Deep Learning & Convolutional Networks In Vision (part 2)

VRML, Paris 2013-07-23

Yann LeCun

Center for Data Science & Courant Institute, NYU

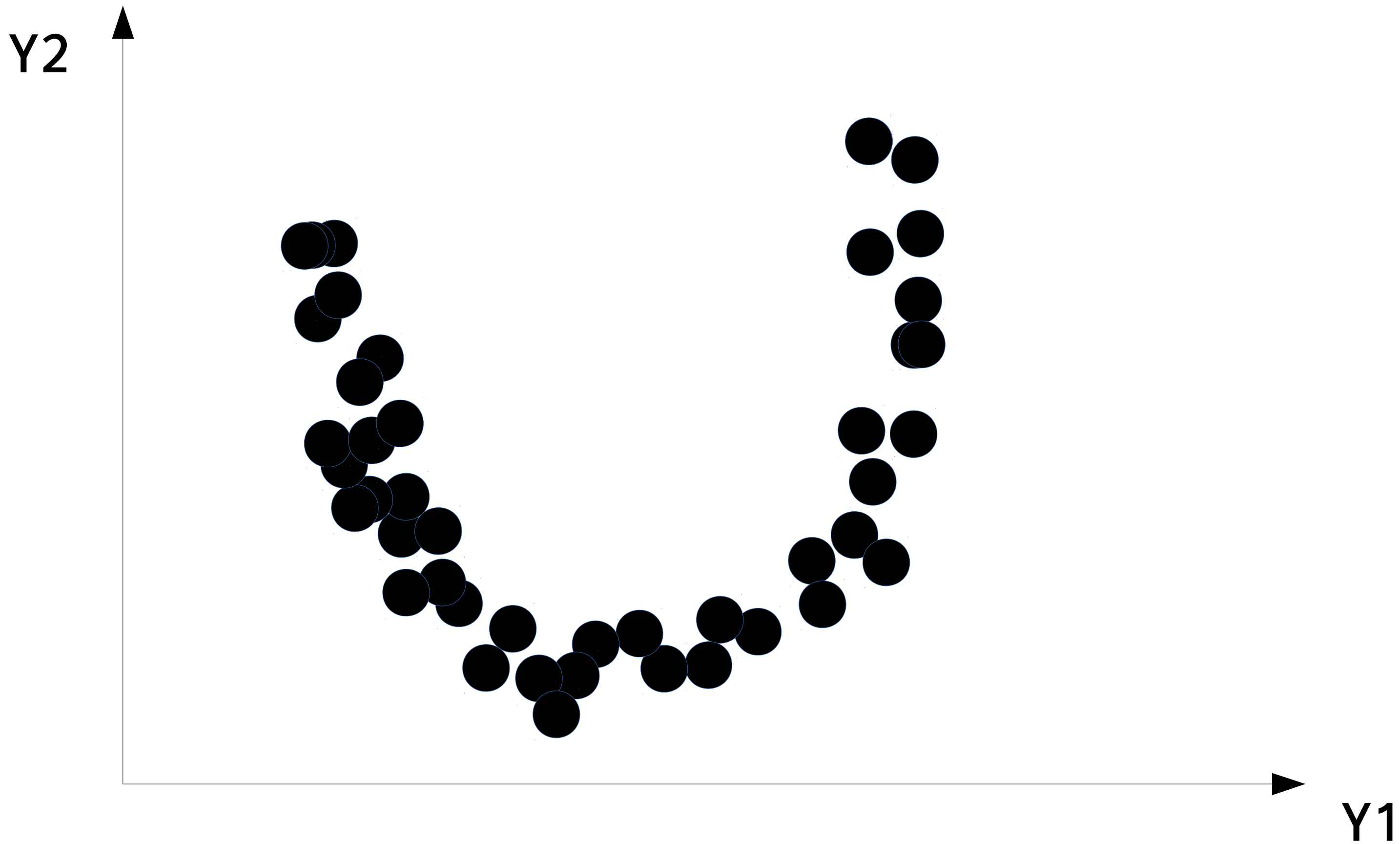
yann@cs.nyu.edu

http://yann.lecun.com

Energy-Based Unsupervised Learning

Energy-Based Unsupervised Learning

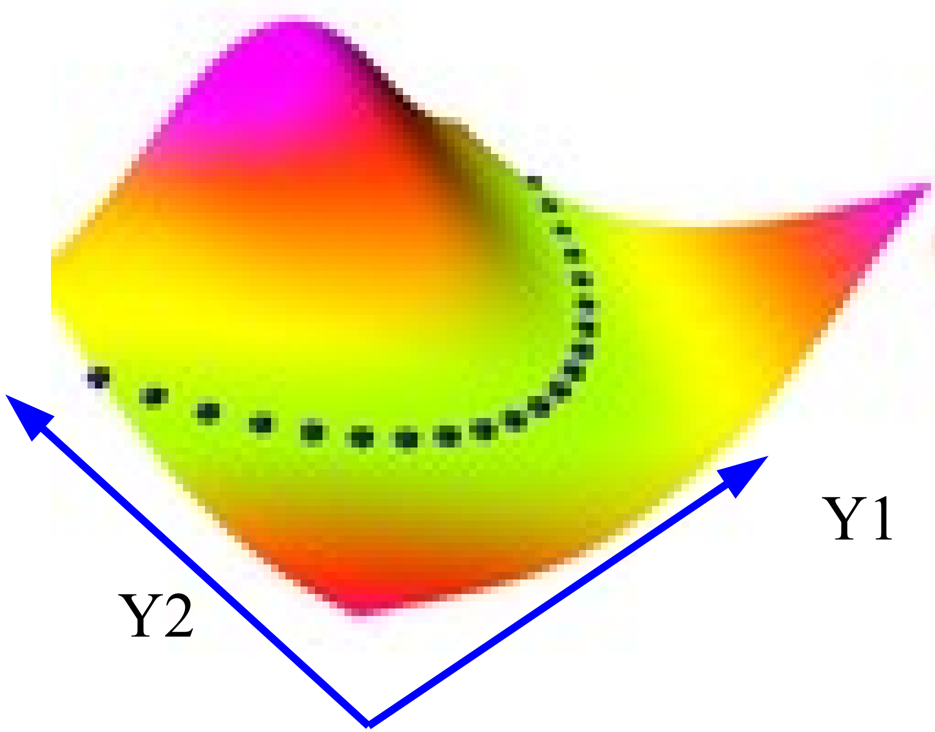
- Learning an energy function (or contrast function) that takes
  - Low values on the data manifold
  - Higher values everywhere else

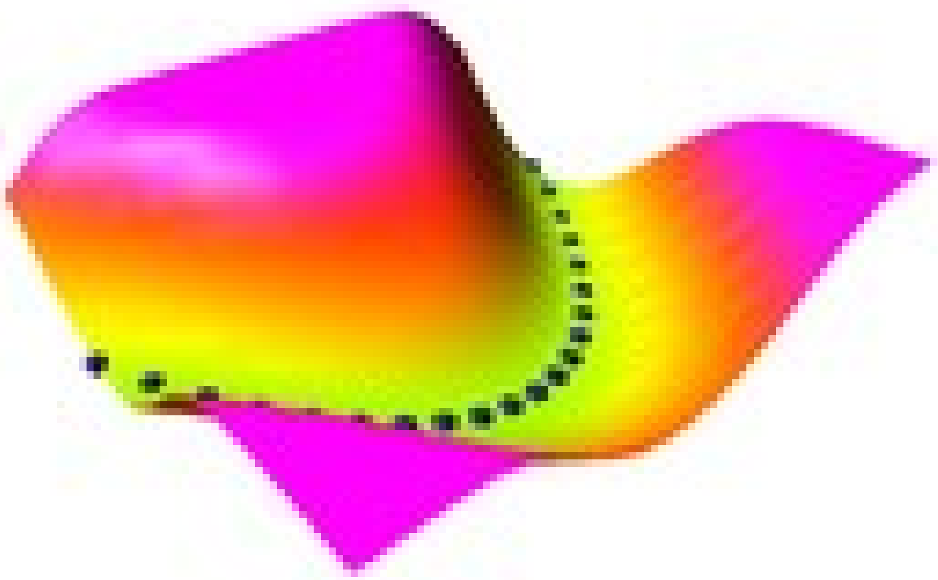


Capturing Dependencies Between Variables with an Energy Function

Y LeCun

- **The energy surface is a "contrast function" that takes low values on the data manifold, and higher values everywhere else**
  - Special case: energy = negative log density
  - Example: the samples live in the manifold $Y_2 = (Y_1)^2$

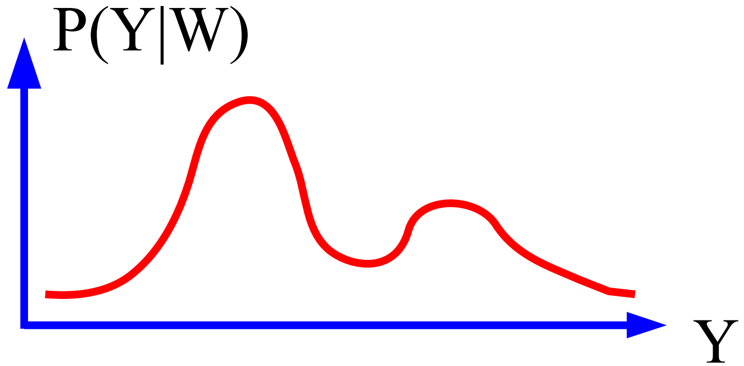




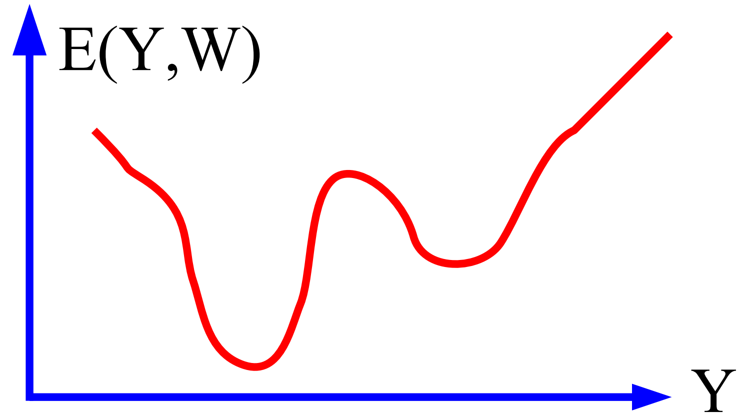
Transforming Energies into Probabilities (if necessary)

- **The energy can be interpreted as an unnormalized negative log density**
- **Gibbs distribution: Probability proportional to exp(-energy)**
  - Beta parameter is akin to an inverse temperature
- **Don't compute probabilities unless you absolutely have to**
  - Because the denominator is often intractable

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$

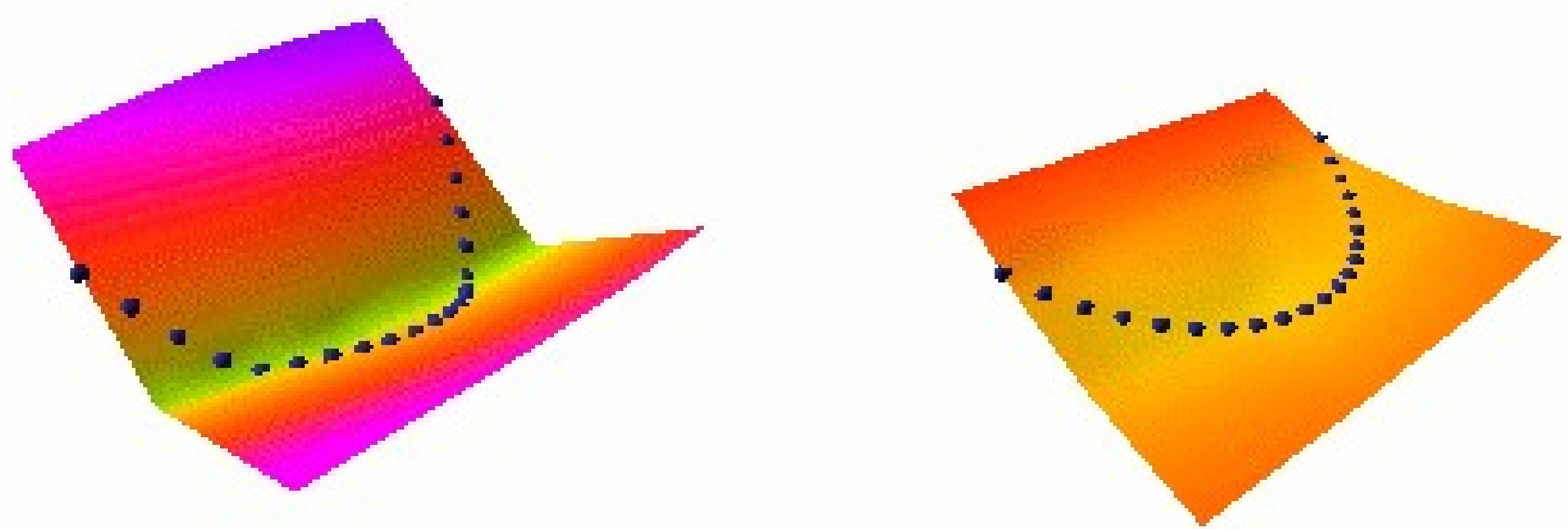


$$E(Y,W) \propto -\log P(Y|W)$$



Learning the Energy Function

- **parameterized energy function E(Y,W)**
  - Make the energy low on the samples
  - Make the energy higher everywhere else
  - Making the energy low on the samples is easy
  - But how do we make it higher everywhere else?



Seven Strategies to Shape the Energy Function

- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA
- 2. push down of the energy of data points, push up everywhere else
  - Max likelihood (needs tractable partition function)
- 3. push down of the energy of data points, push up on chosen locations
  - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
- 4. minimize the gradient and maximize the curvature around data points
  - score matching
- 5. train a dynamical system so that the dynamics goes to the manifold
  - denoising auto-encoder
- 6. use a regularizer that limits the volume of space that has low energy
  - Sparse coding, sparse auto-encoder, PSD
- 7. if E(Y) = ||Y - G(Y)||^2, make G(Y) as "constant" as possible.
  - Contracting auto-encoder, saturating auto-encoder

#1: constant volume of low energy

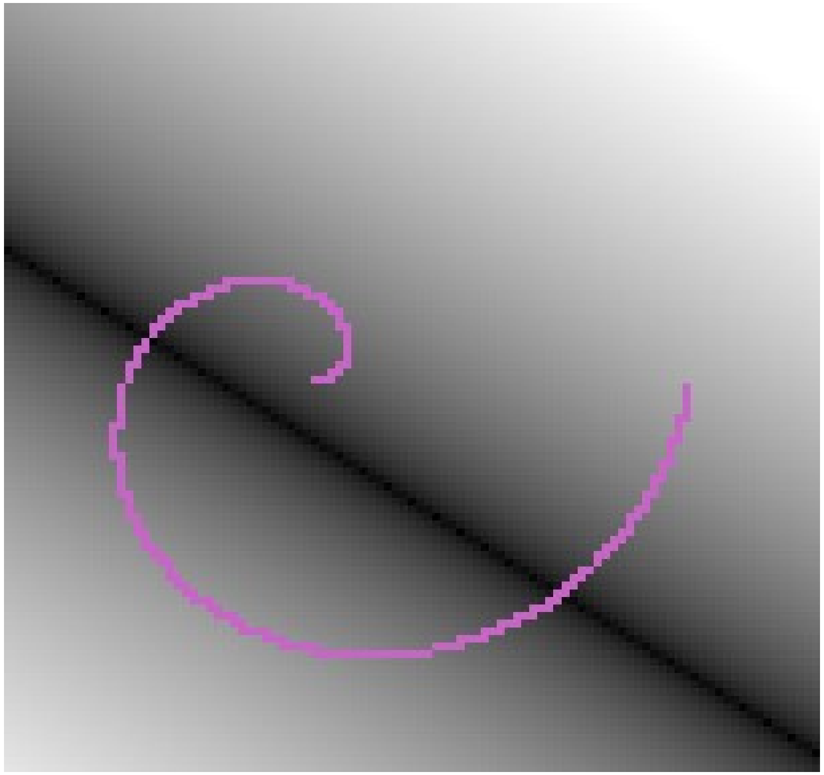
- 1. build the machine so that the volume of low energy stuff is constant
  - PCA, K-means, GMM, square ICA...

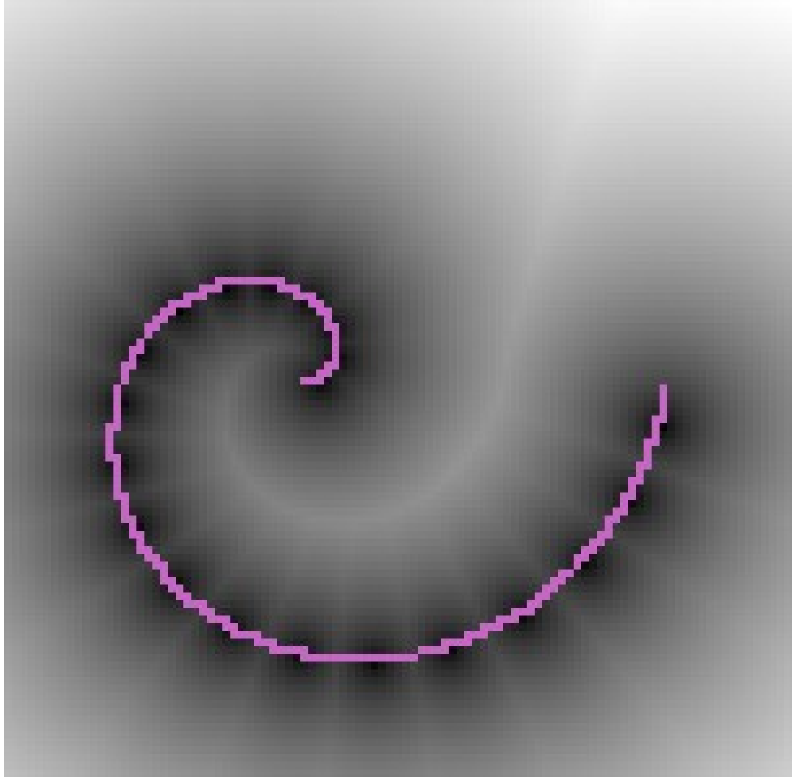
PCA

$$E(Y)=\|W^T WY-Y\|^2$$

K-Means,
Z constrained to 1-of-K code

$$E(Y)=min_z \sum_i \|Y-W_i Z_i\|^2$$





#2: push down of the energy of data points, push up everywhere else

Y LeCun

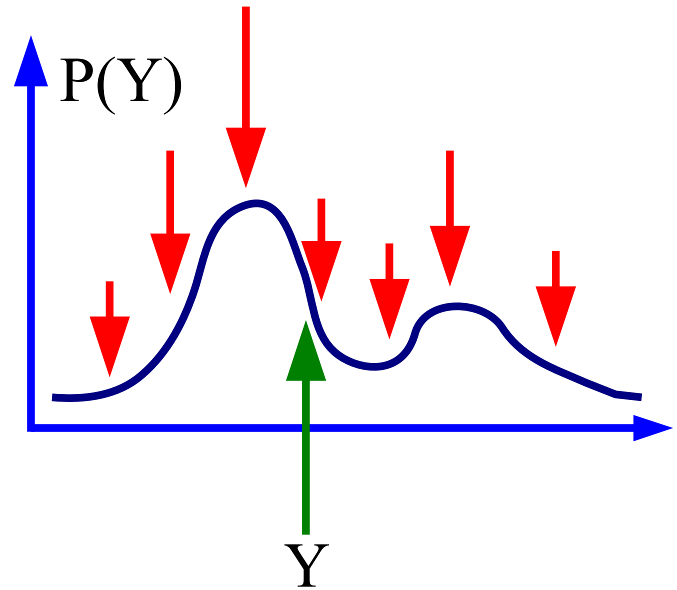
- **Max likelihood (requires a tractable partition function)**

Maximizing P(Y|W) on training samples

make this big

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$

make this small

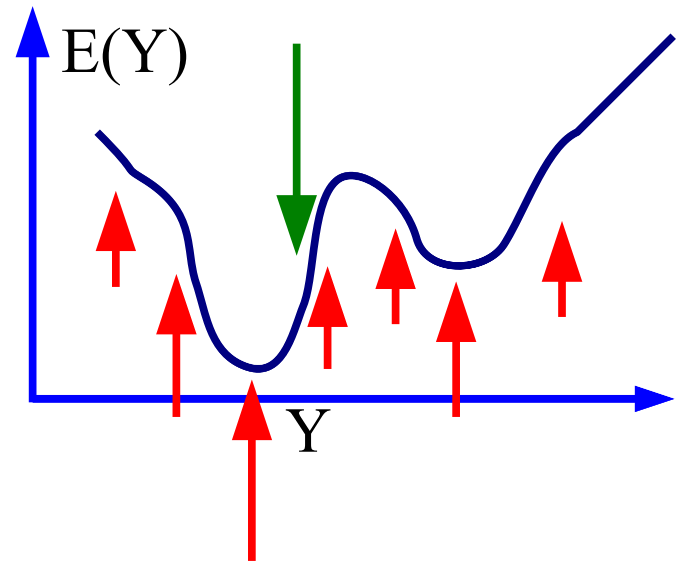


Minimizing -log P(Y,W) on training samples

$$L(Y,W) = E(Y,W) + \frac{1}{\beta} \log \int_y e^{-\beta E(y,W)}$$

make this small

make this big



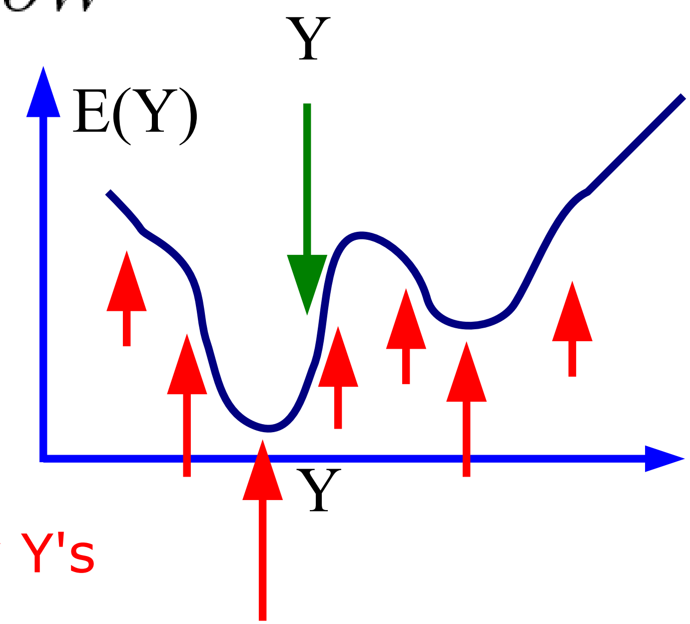
#2: push down of the energy of data points, push up everywhere else

Gradient of the negative log-likelihood loss for one sample Y:

$$\frac{\partial L(Y,W)}{\partial W} = \frac{\partial E(Y,W)}{\partial W} - \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

Gradient descent:

$$W \leftarrow W - \eta \frac{\partial L(Y,W)}{\partial W}$$



Pushes down on the energy of the samples

Pulls up on the energy of low-energy Y's

$$W \leftarrow W - \eta \frac{\partial E(Y,W)}{\partial W} + \eta \int_y P(y|W)\frac{\partial E(y,W)}{\partial W}$$

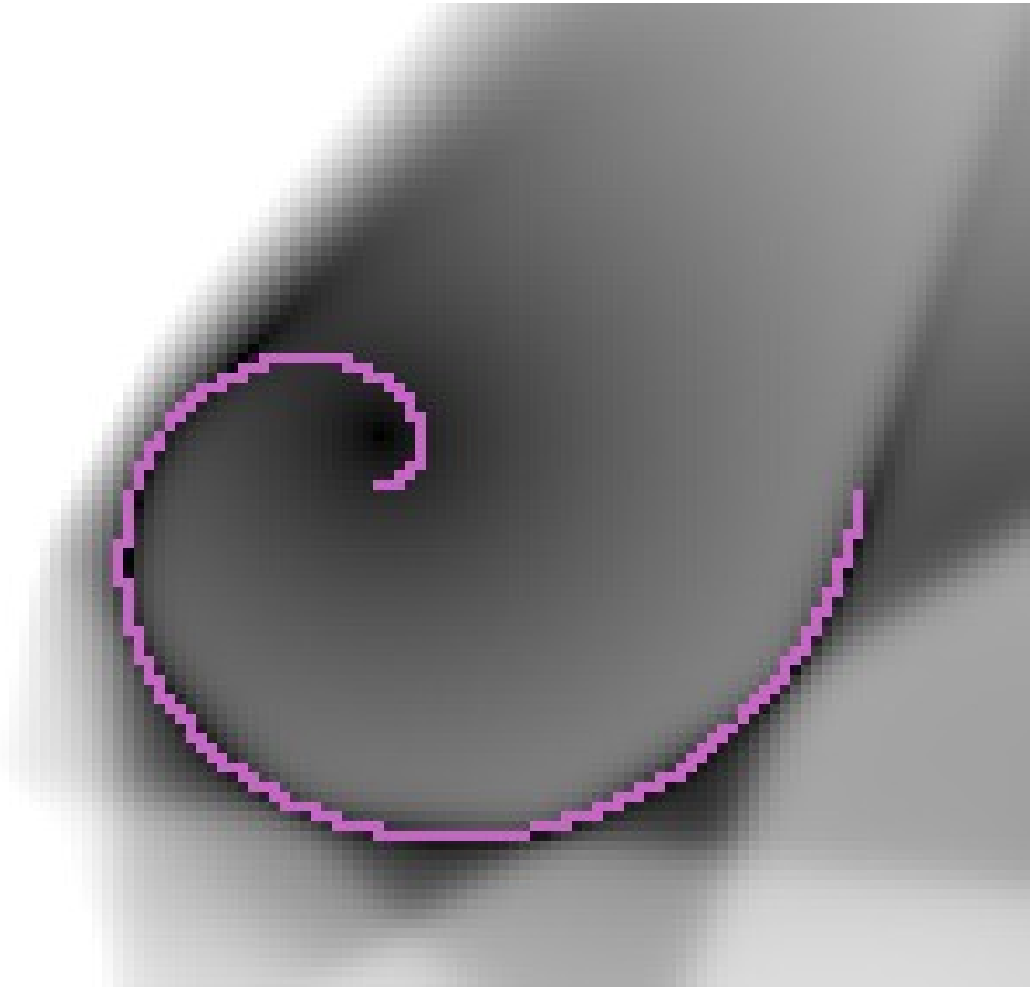
#3. push down of the energy of data points, push up on chosen locations

- **contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow**
- **Contrastive divergence: basic idea**
  - Pick a training sample, lower the energy at that point
  - From the sample, move down in the energy surface with noise
  - Stop after a while
  - Push up on the energy of the point where we stopped
  - This creates grooves in the energy surface around data manifolds
  - CD can be applied to any energy function (not just RBMs)
- **Persistent CD: use a bunch of "particles" and remember their positions**
  - Make them roll down the energy surface with noise
  - Push up on the energy wherever they are
  - Faster than CD
- **RBM**

$$E(Y,Z) = -Z^T W Y \qquad E(Y) = -\log \sum_z e^{Z^T W Y}$$

#6. use a regularizer that limits the volume of space that has low energy

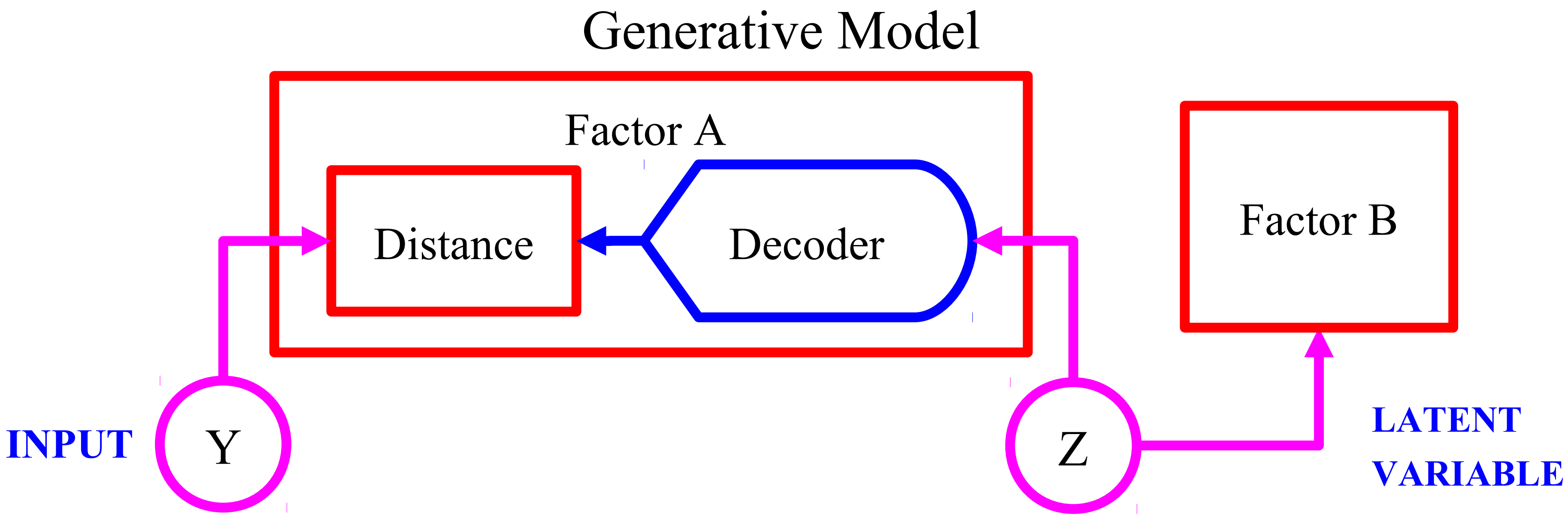
- Sparse coding, sparse auto-encoder, Predictive Saprse Decomposition



Sparse Modeling,
Sparse Auto-Encoders,
Predictive Sparse Decomposition
LISTA

How to Speed Up Inference in a Generative Model?

- Factor Graph with an asymmetric factor
- Inference Z → Y is easy
  - Run Z through deterministic decoder, and sample Y
- Inference Y → Z is hard, particularly if Decoder function is many-to-one
  - MAP: minimize sum of two factors with respect to Z
  - Z* = argmin_z Distance[Decoder(Z), Y] + FactorB(Z)
- Examples: K-Means (1of K), Sparse Coding (sparse), Factor Analysis

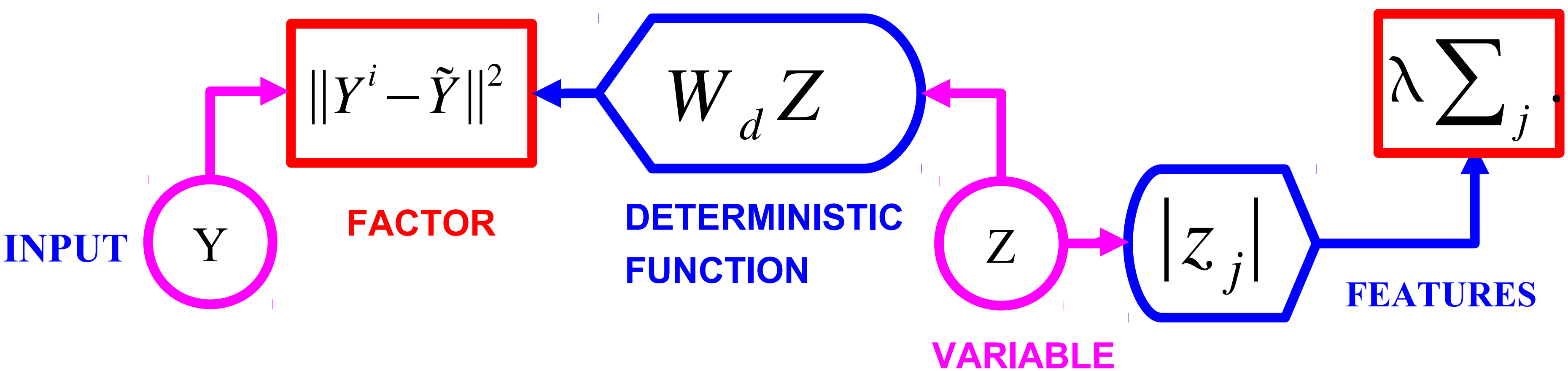


Sparse Coding & Sparse Modeling

[Olshausen & Field 1997]

- Sparse linear reconstruction
- Energy = reconstruction_error + code_prediction_error + code_sparsity

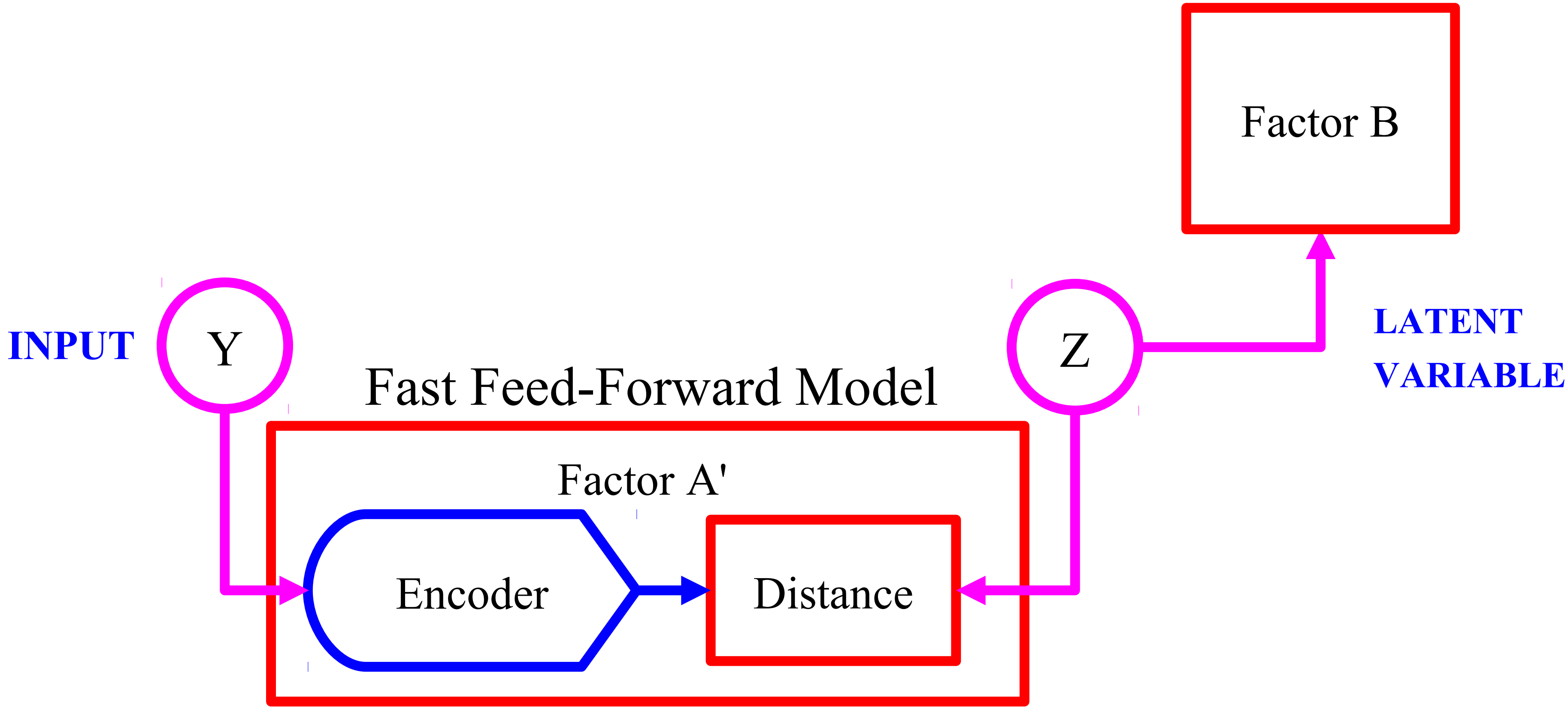
$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \lambda \sum_j |z_j|$$



- Inference is slow $Y \rightarrow \hat{Z} = \mathrm{argmin}_Z E(Y, Z)$

Encoder Architecture

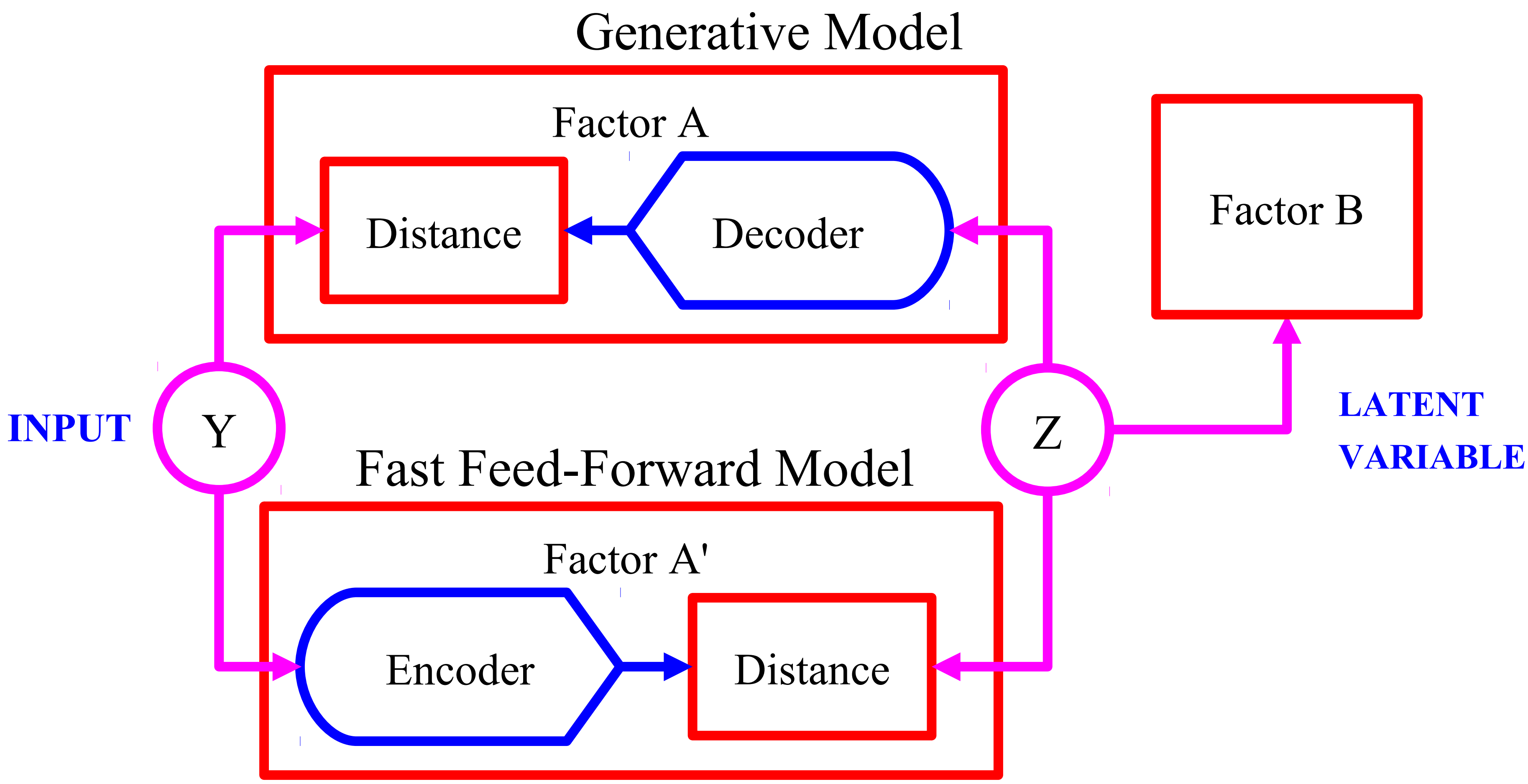
Examples: most ICA models, Product of Experts



Encoder-Decoder Architecture

[Kavukcuoglu, Ranzato, LeCun, rejected by every conference, 2008-2009]

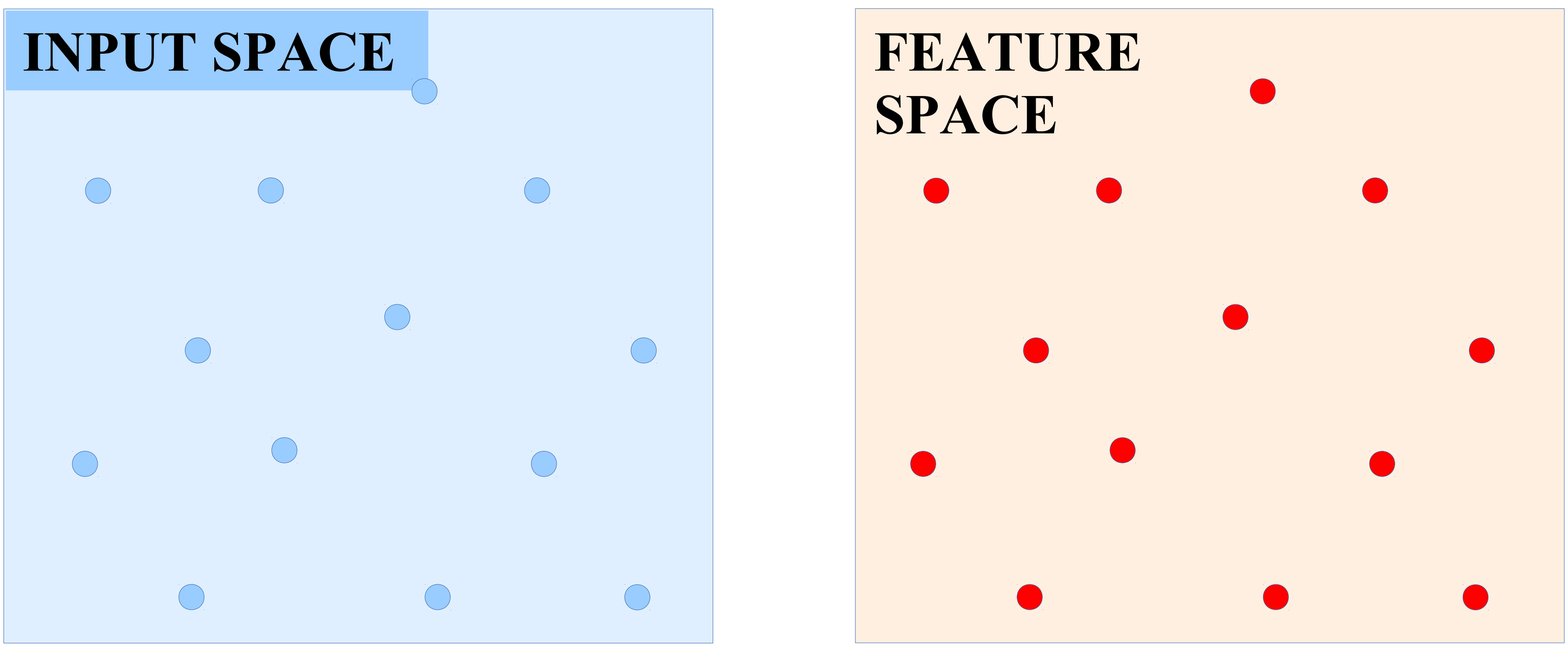
- Train a "simple" feed-forward function to predict the result of a complex optimization on the data points of interest



- 1. Find optimal Zi for all Yi; 2. Train Encoder to predict Zi from Yi

Why Limit the Information Content of the Code?

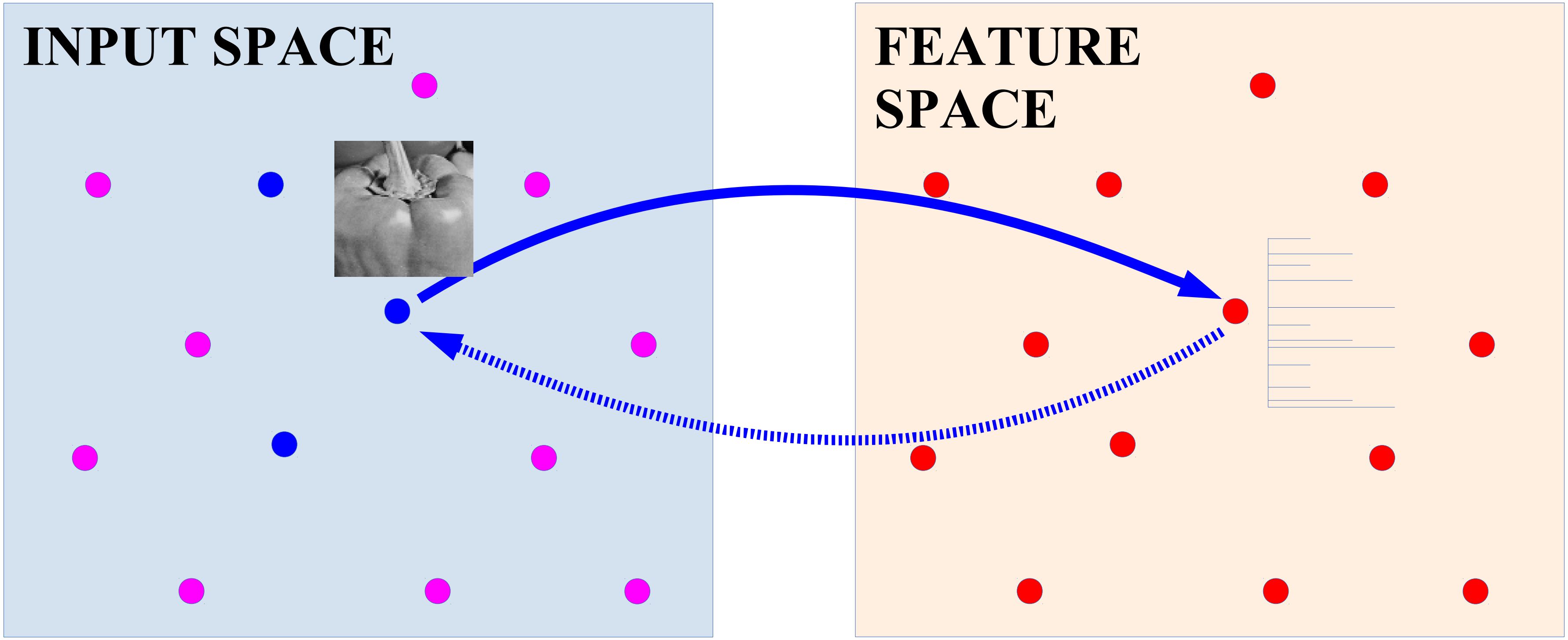
- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**



Why Limit the Information Content of the Code?

- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**

Training based on minimizing the reconstruction error over the training set

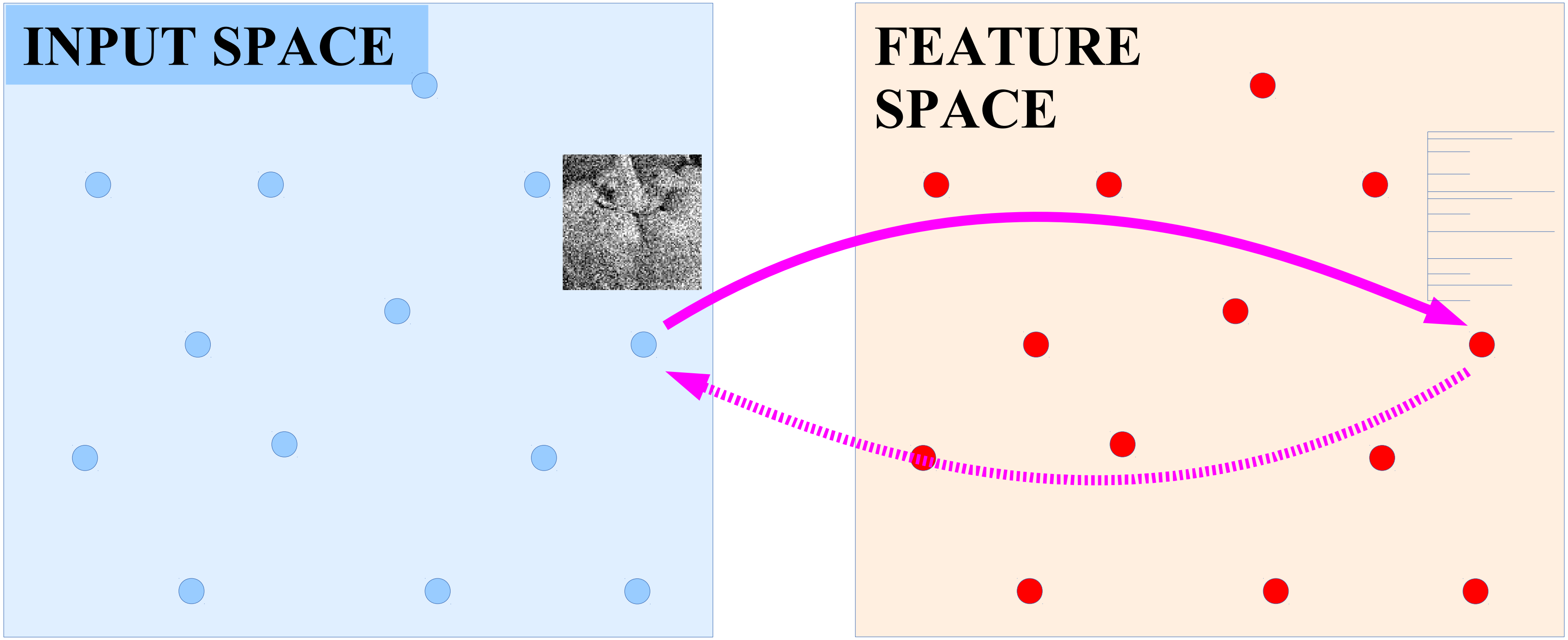


Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

BAD: machine does not learn structure from training data!!

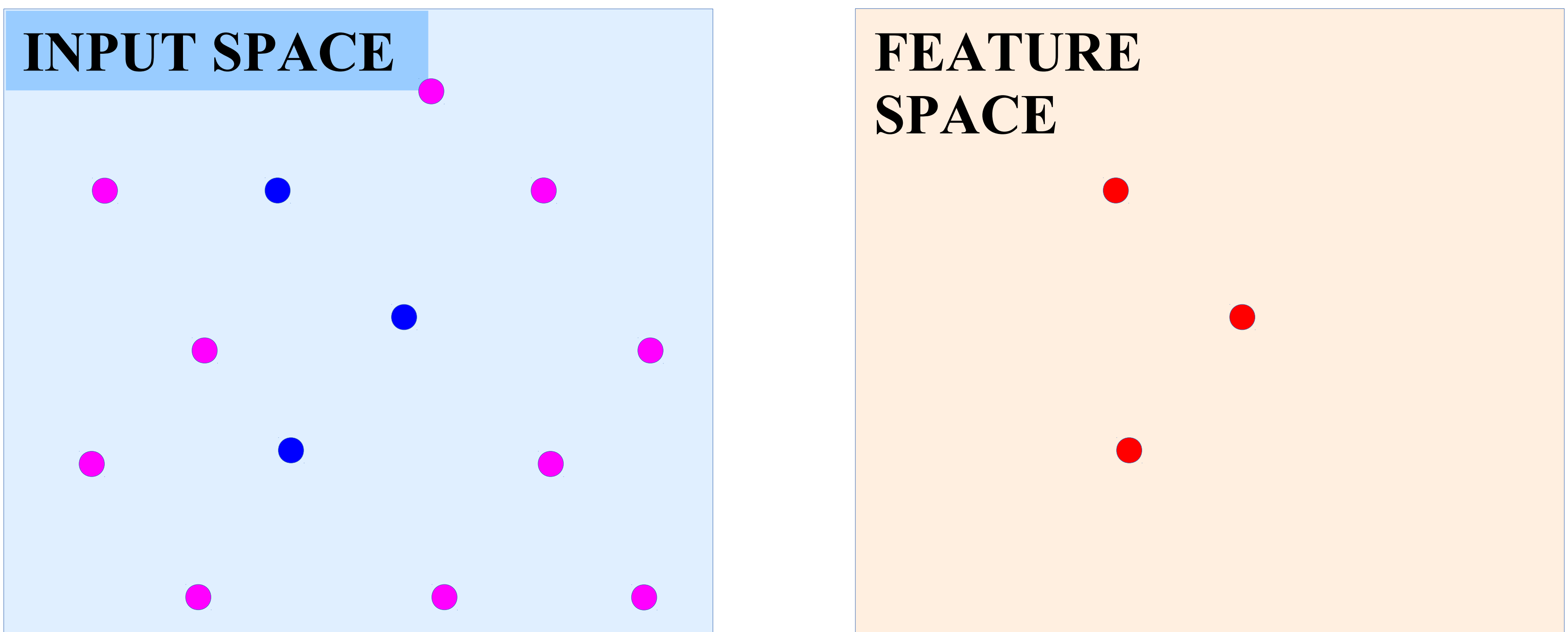
It just copies the data.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

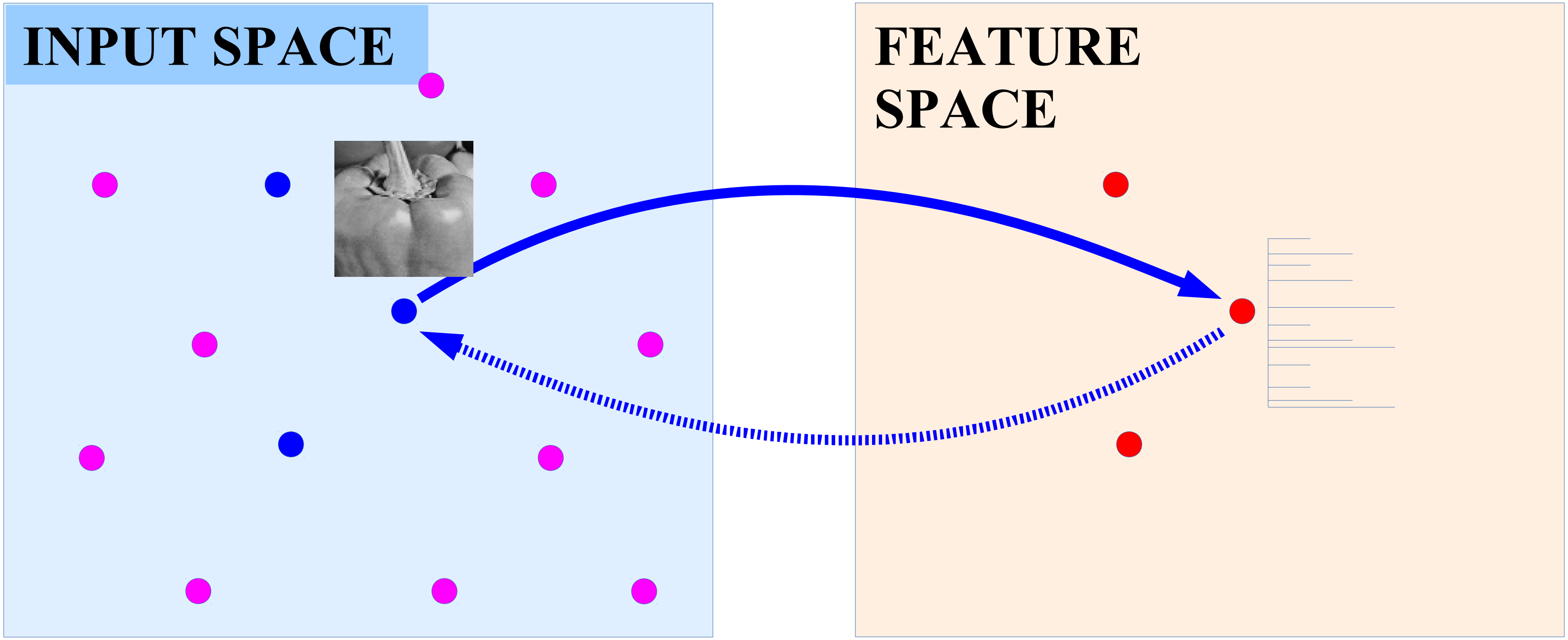
IDEA: reduce number of available codes.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

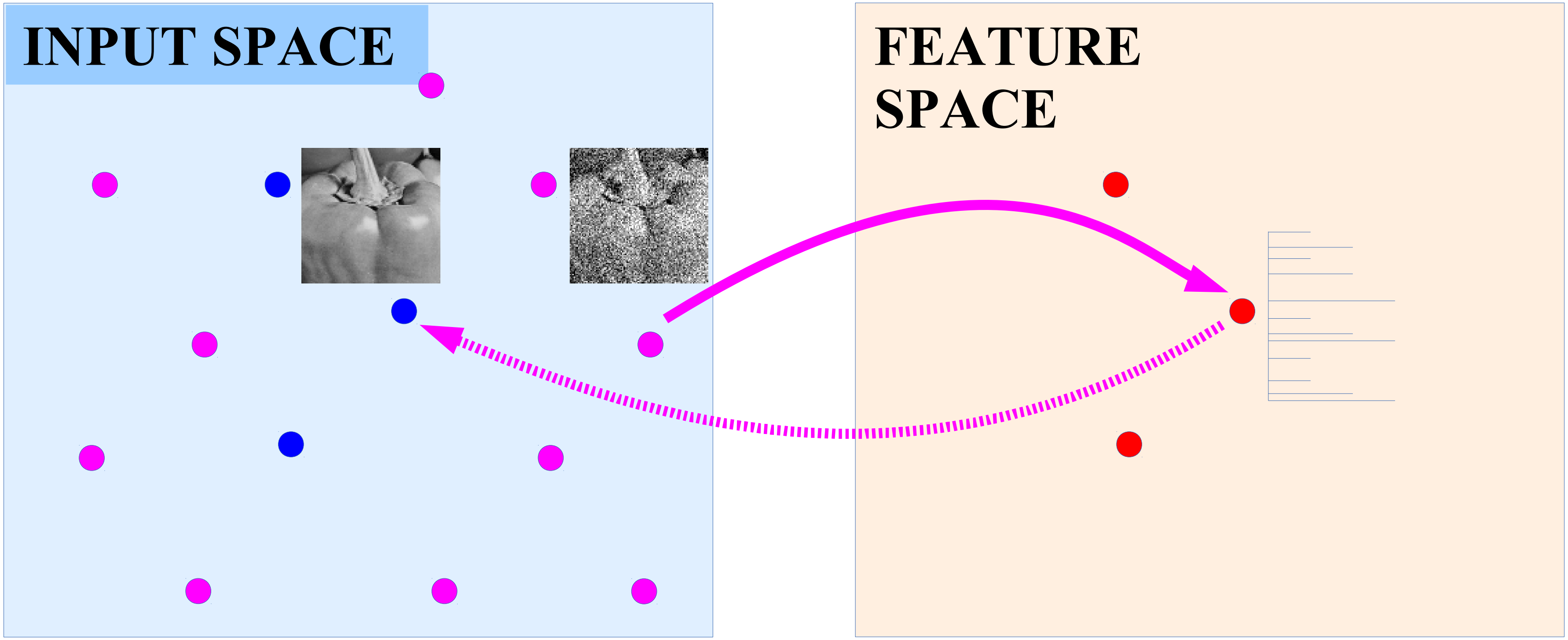
IDEA: reduce number of available codes.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

IDEA: reduce number of available codes.



Predictive Sparse Decomposition (PSD): sparse auto-encoder

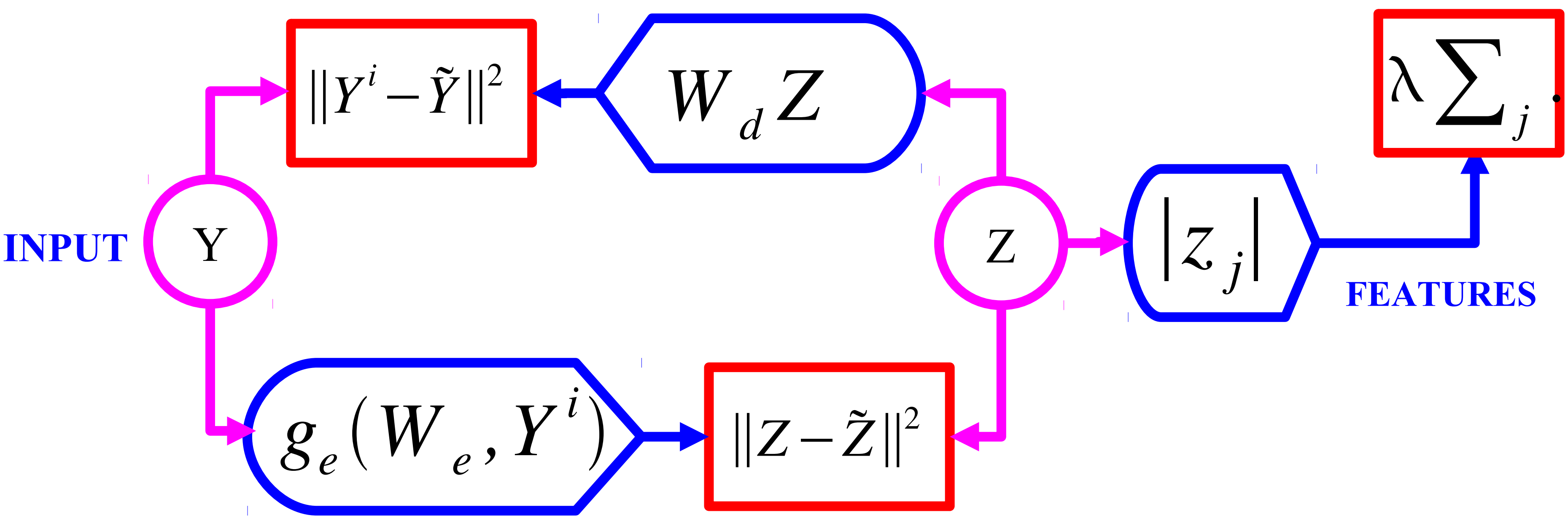
[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

Prediction the optimal code with a trained encoder

Energy = reconstruction_error + code_prediction_error + code_sparsity

$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \|Z - g_e(W_e, Y^i)\|^2 + \lambda \sum_j |z_j|$$

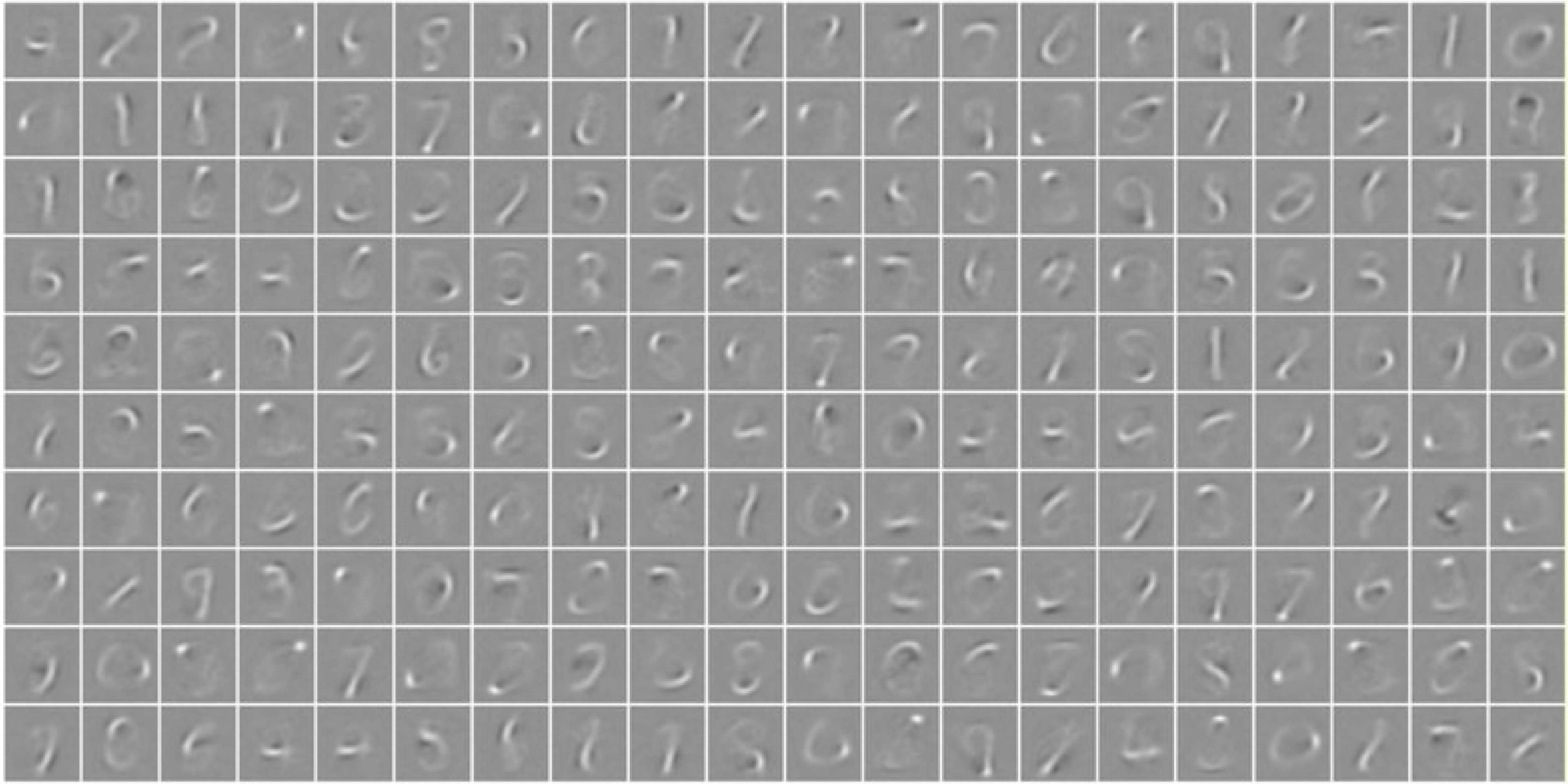
$$g_e(W_e, Y^i) = shrinkage(W_e Y^i)$$



PSD: Basis Functions on MNIST

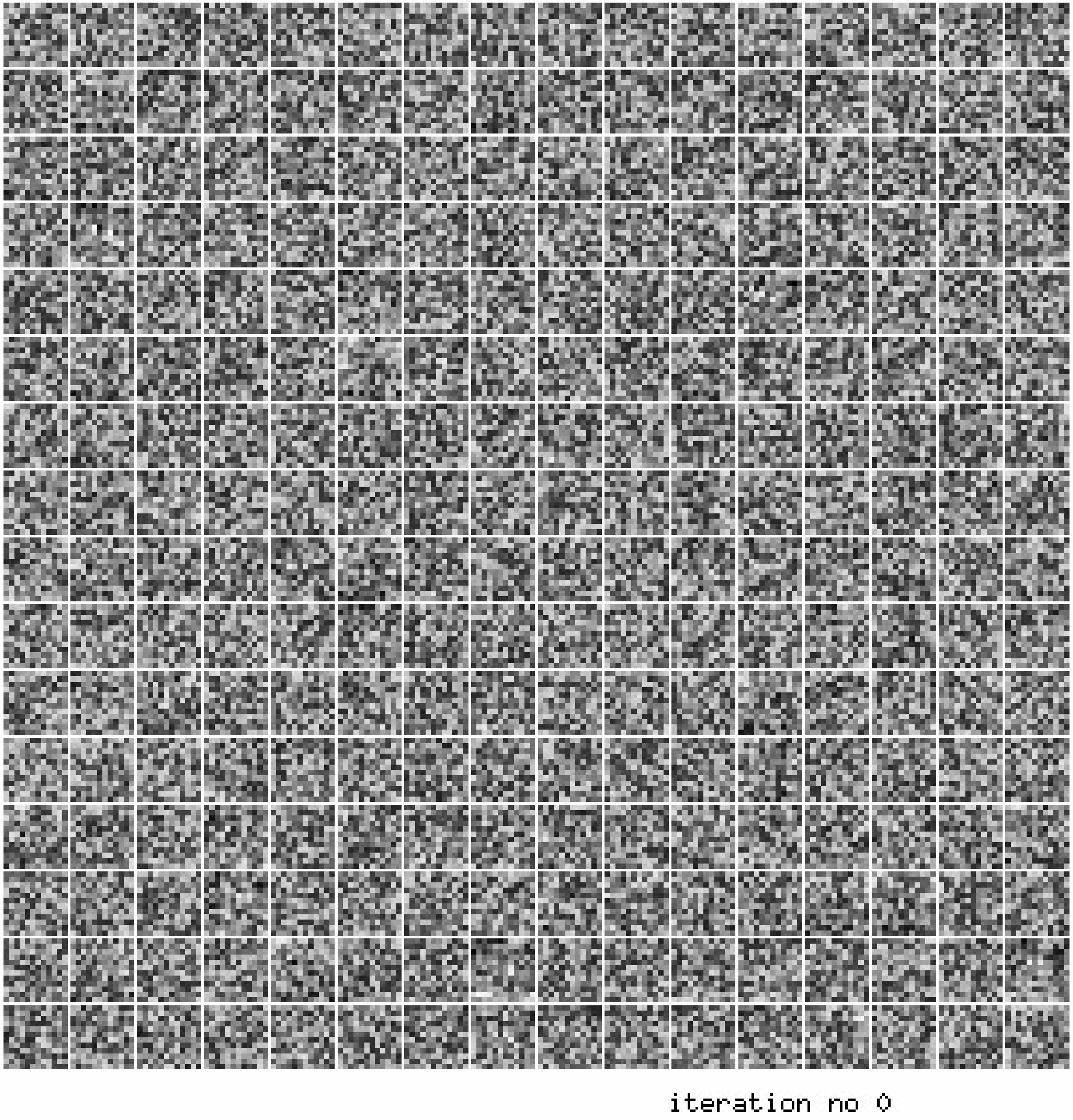
Y LeCun

- **Basis functions (and encoder matrix) are digit parts**

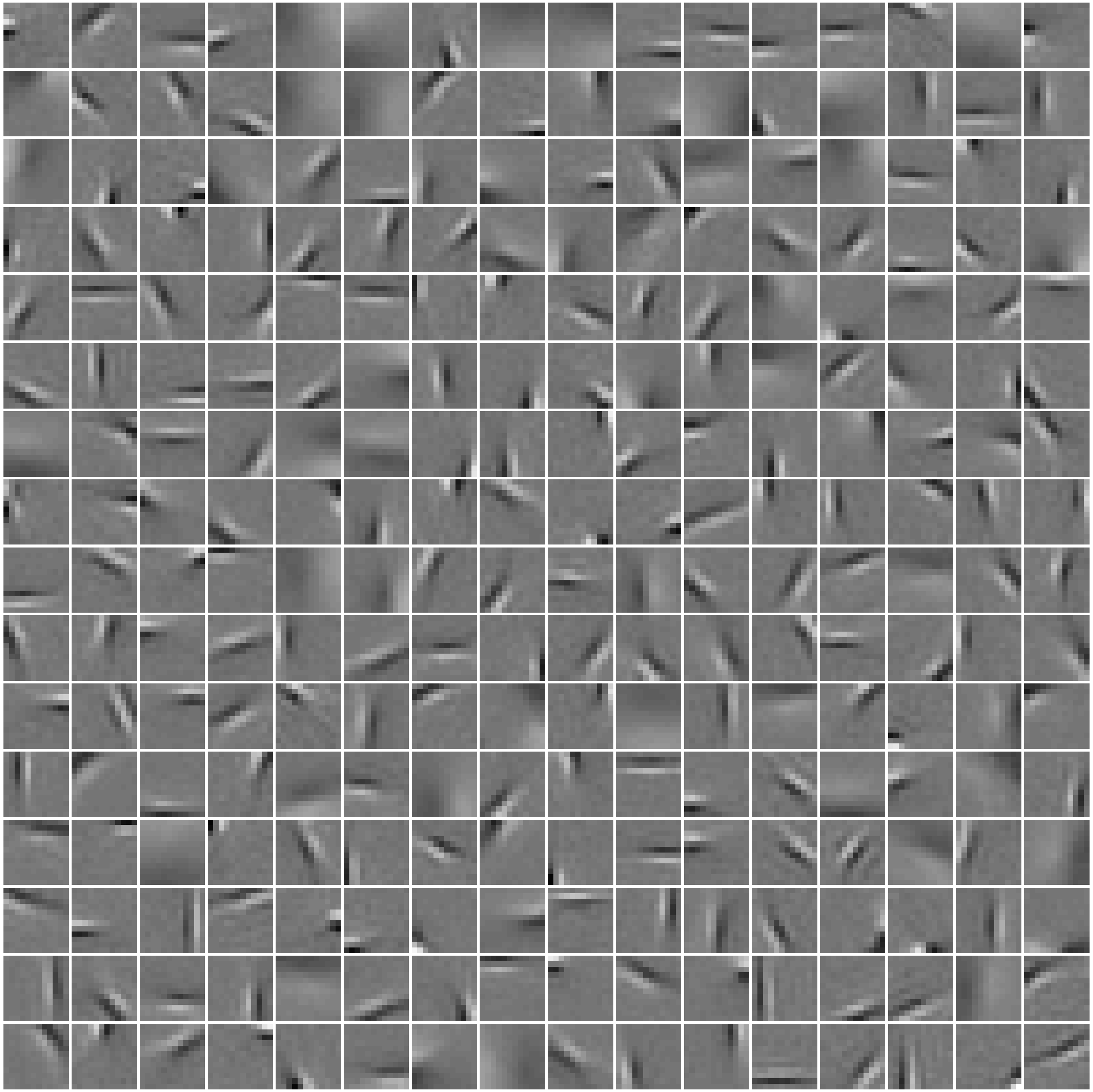


Predictive Sparse Decomposition (PSD): Training

- **Training on natural images patches.**
  - 12X12
  - 256 basis functions



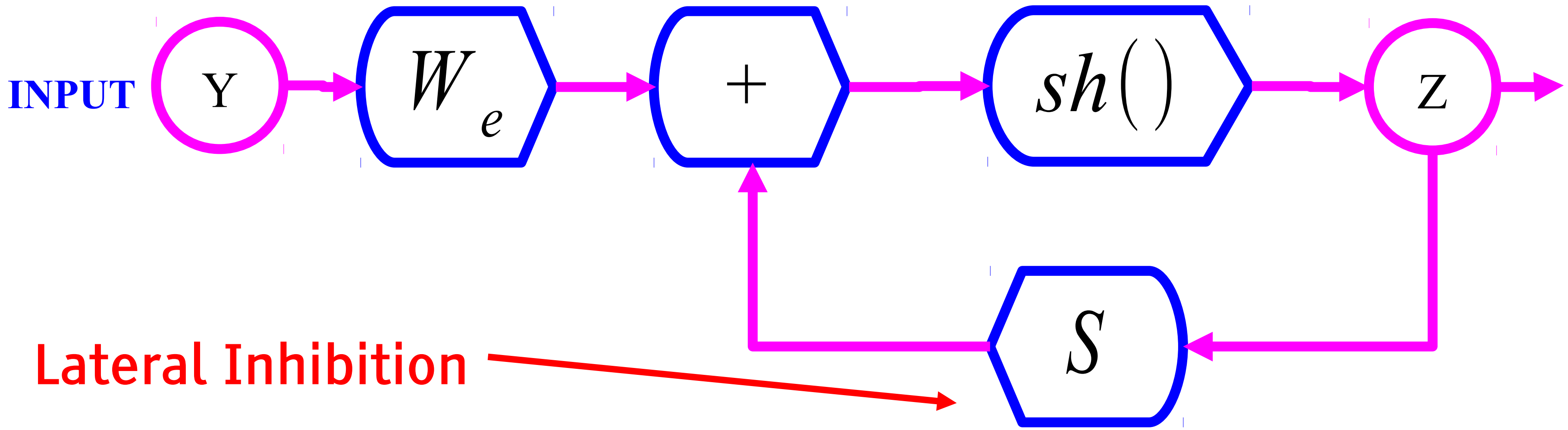
Learned Features on natural patches: V1-like receptive fields



Better Idea: Give the "right" structure to the encoder

- **ISTA/FISTA: iterative algorithm that converges to optimal sparse code**

[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]



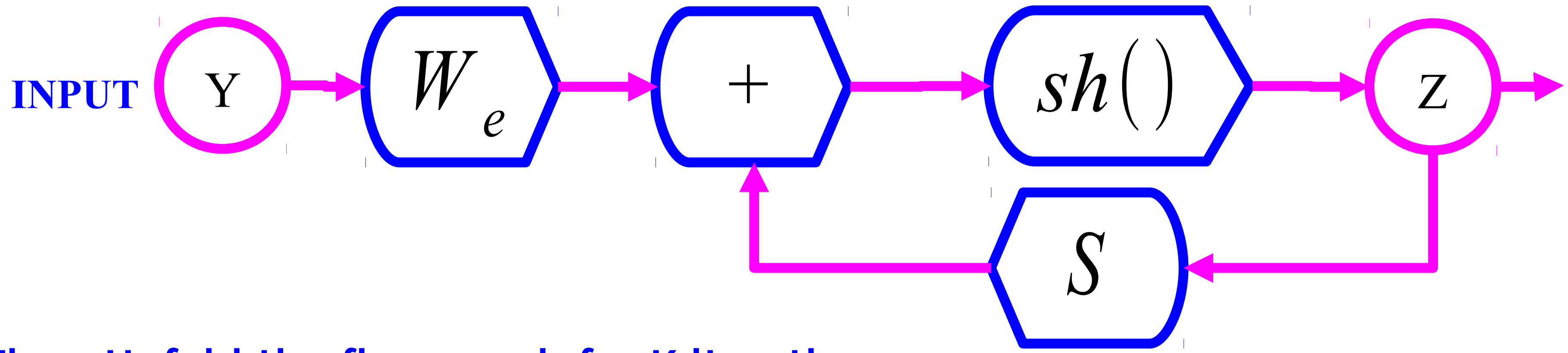
$$Z(t+1) = \text{Shrinkage}_{\lambda/L}\left[Z(t) - \frac{1}{L}W_d^T(W_d Z(t) - Y)\right]$$

$$Z(t+1) = \text{Shrinkage}_{\lambda/L}\left[W_e^T Y + SZ(t)\right]; \quad W_e = \frac{1}{L}W_d; \quad S = I - \frac{1}{L}W_d^T W_d$$

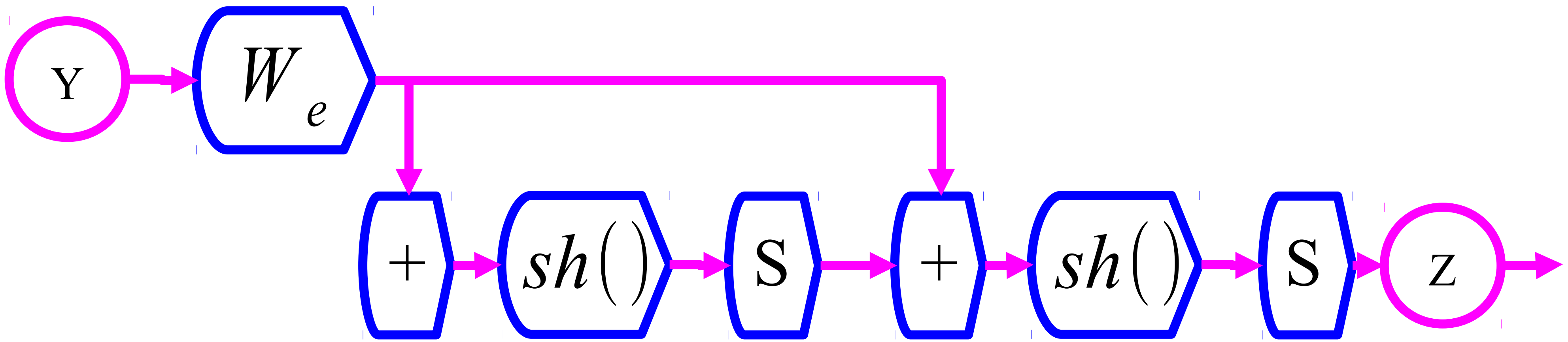
LISTA: Train We and S matrices to give a good approximation quickly

Y LeCun

- Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters

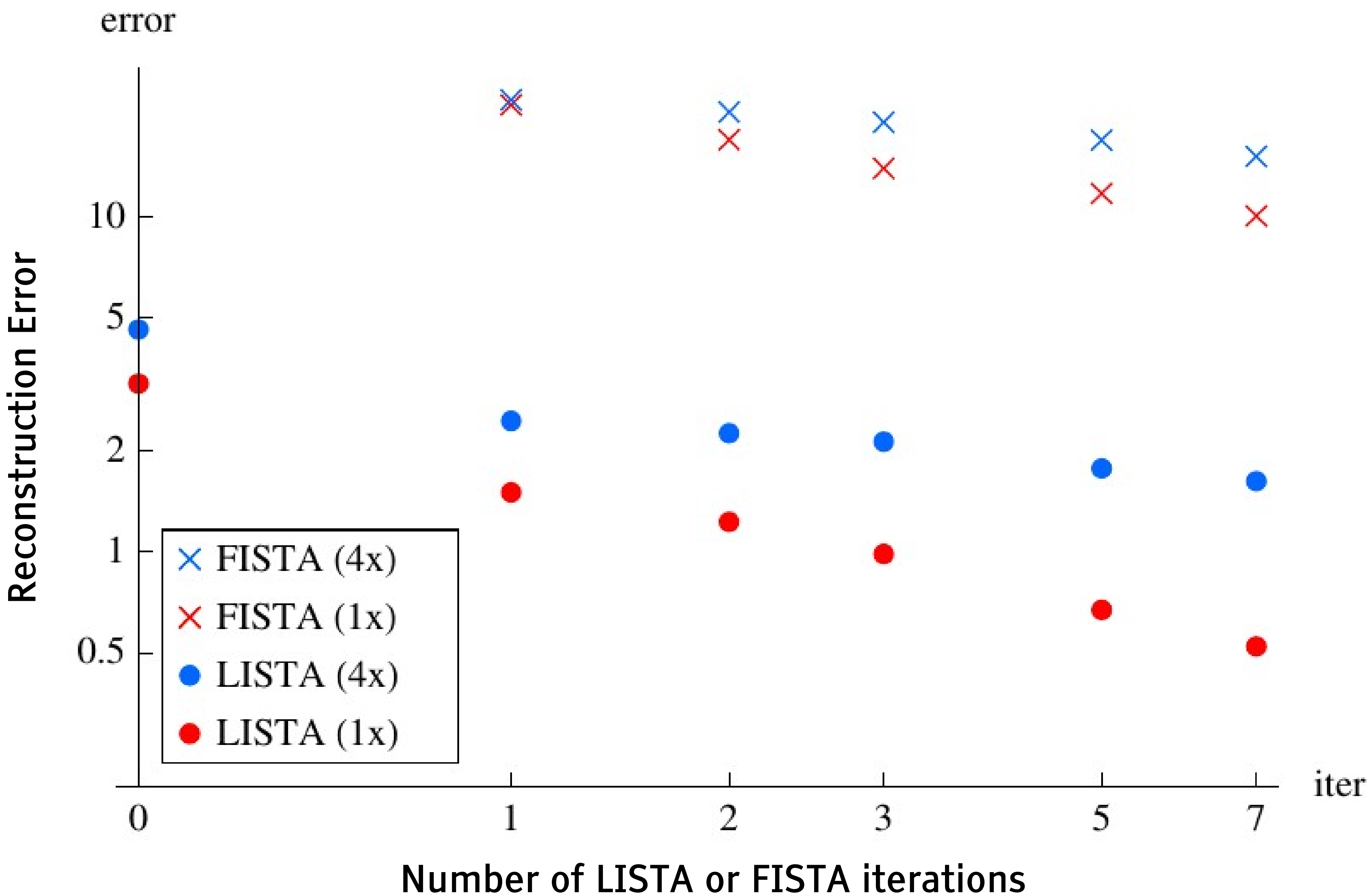


- Time-Unfold the flow graph for K iterations
- Learn the We and S matrices with "backprop-through-time"
- Get the best approximate solution within K iterations



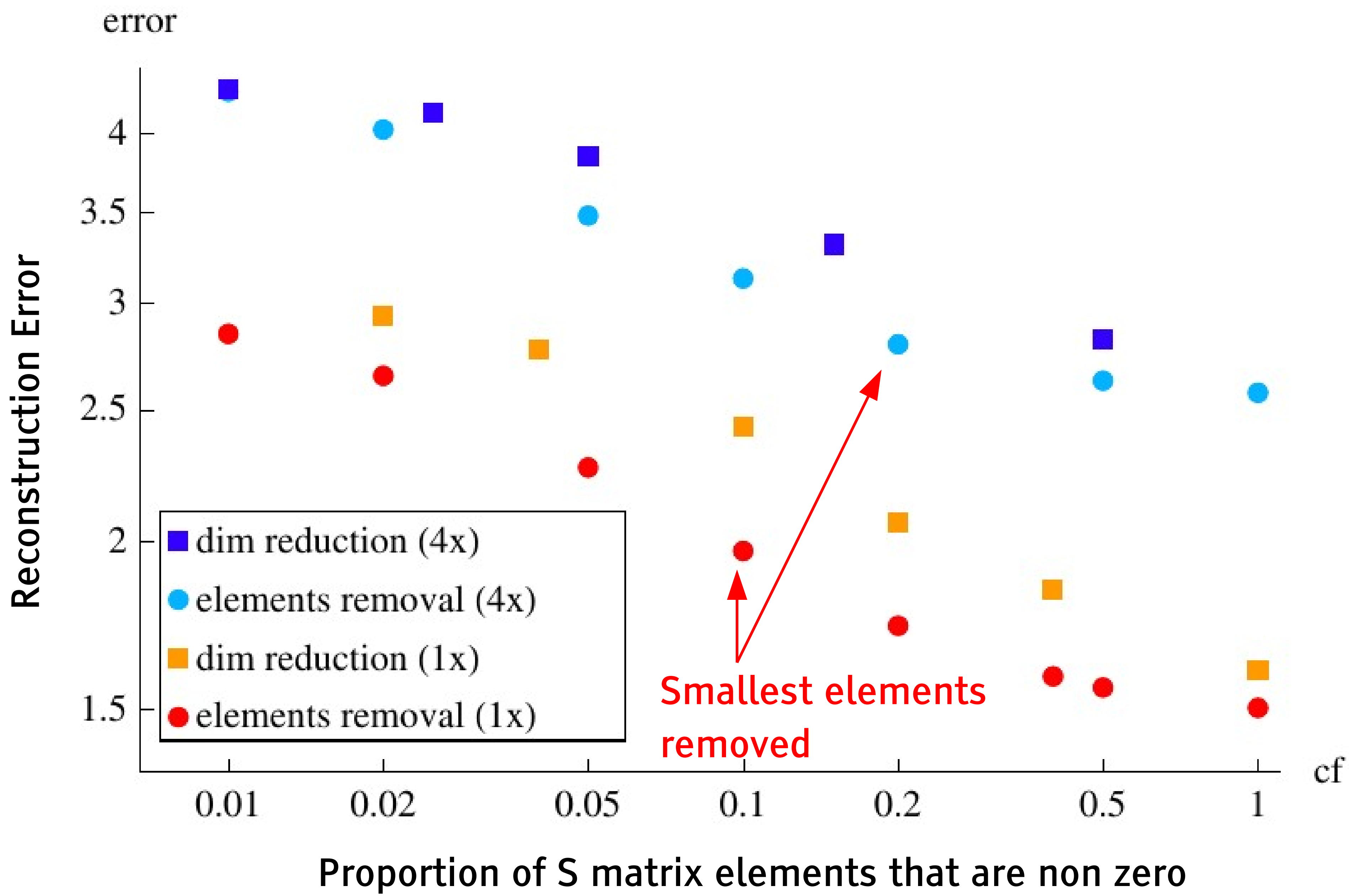
Learning ISTA (LISTA) vs ISTA/FISTA

Y LeCun

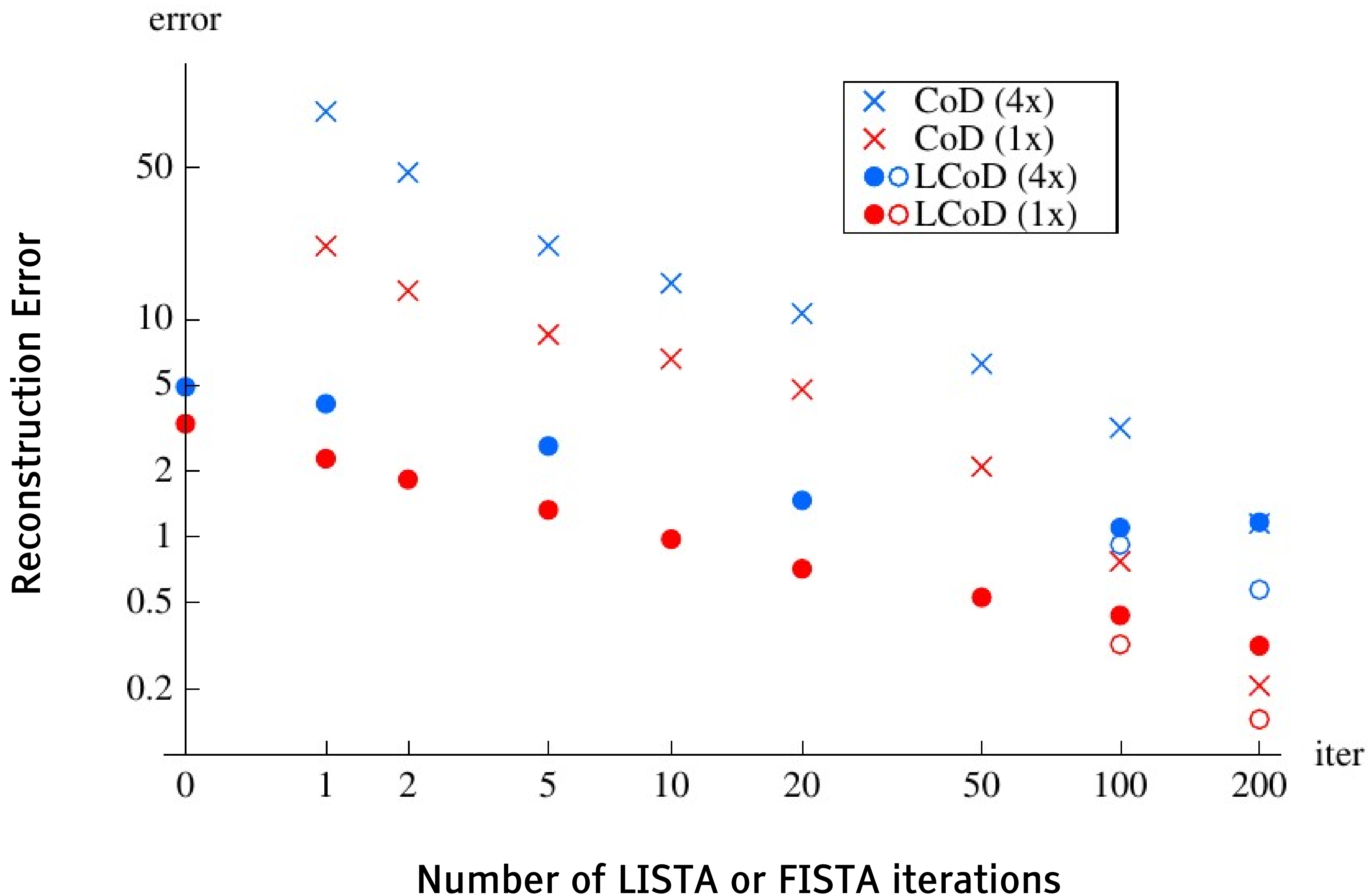


LISTA with partial mutual inhibition matrix

Y LeCun

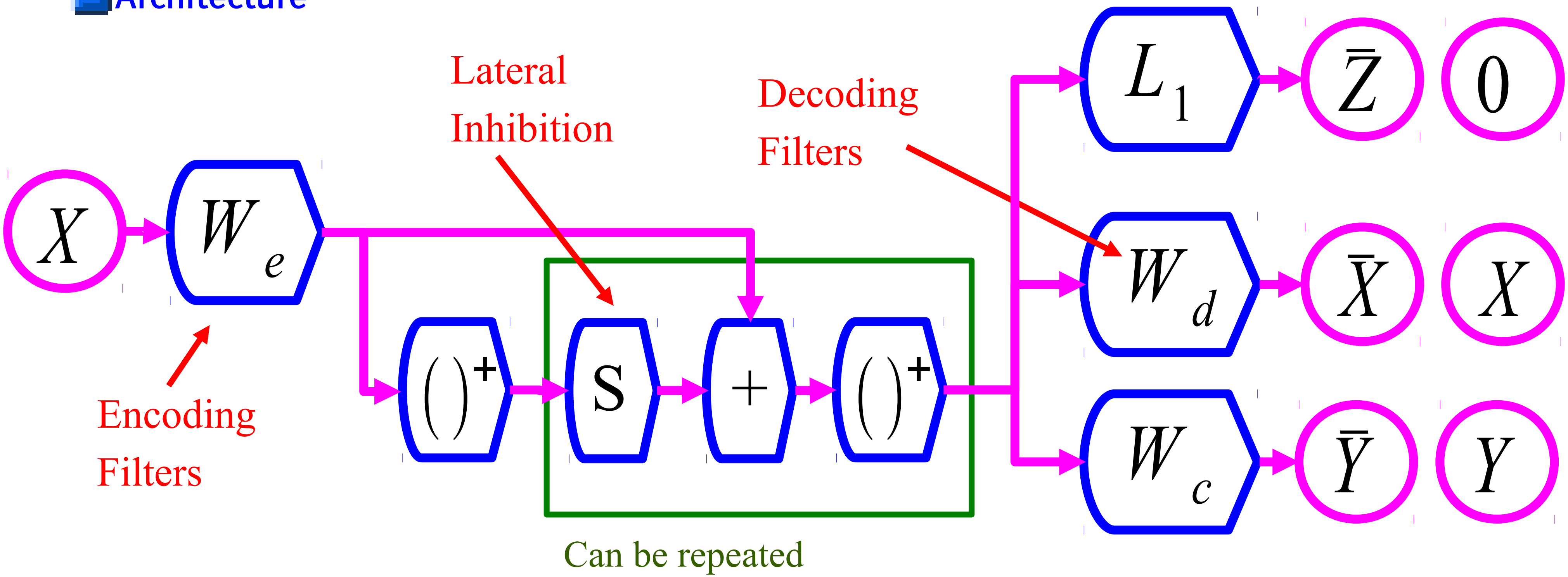


Learning Coordinate Descent (LcoD): faster than LISTA



Discriminative Recurrent Sparse Auto-Encoder (DrSAE)

- Architecture

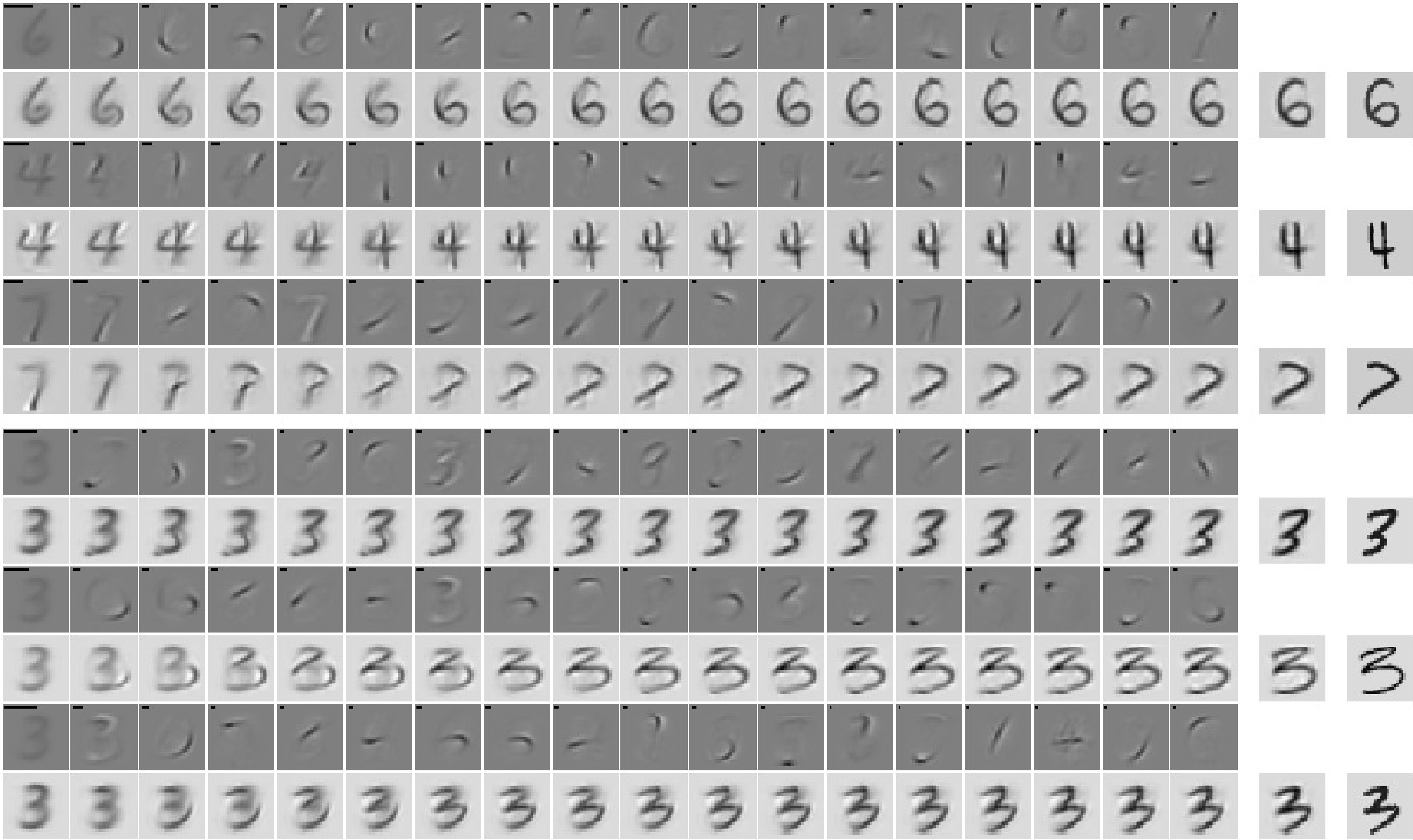


- Rectified linear units
- Classification loss: cross-entropy
- Reconstruction loss: squared error
- Sparsity penalty: L1 norm of last hidden layer
- Rows of Wd and columns of We constrained in unit sphere

[Rolfe & LeCun ICLR 2013]

DrSAE Discovers manifold structure of handwritten digits

- Image = prototype + sparse sum of "parts" (to move around the manifold)

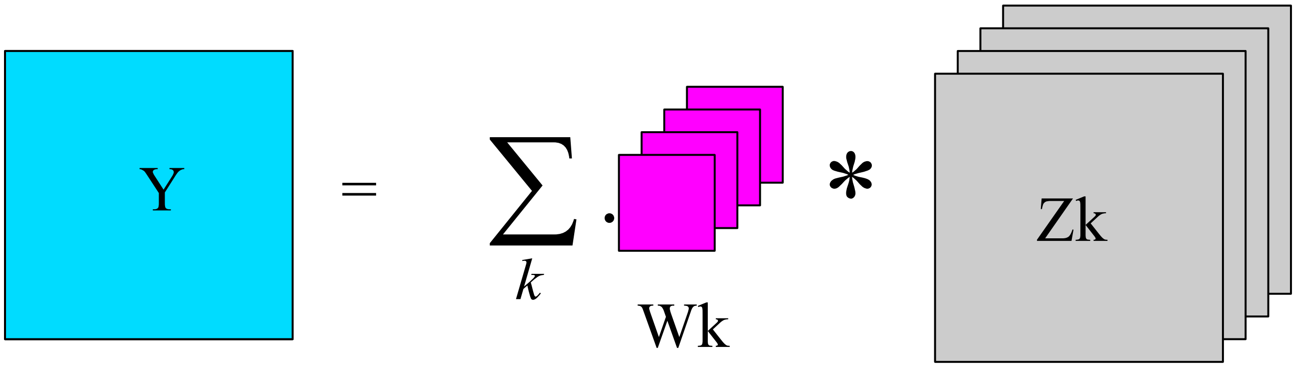


Convolutional Sparse Coding

- **Replace the dot products with dictionary element by convolutions.**
  - Input Y is a full image
  - Each code component Zk is a feature map (an image)
  - Each dictionary element is a convolution kernel

- **Regular sparse coding** $$E(Y,Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$$

- **Convolutional S.C.** $$E(Y,Z) = ||Y - \sum_k W_k * Z_k||^2 + \alpha \sum_k |Z_k|$$

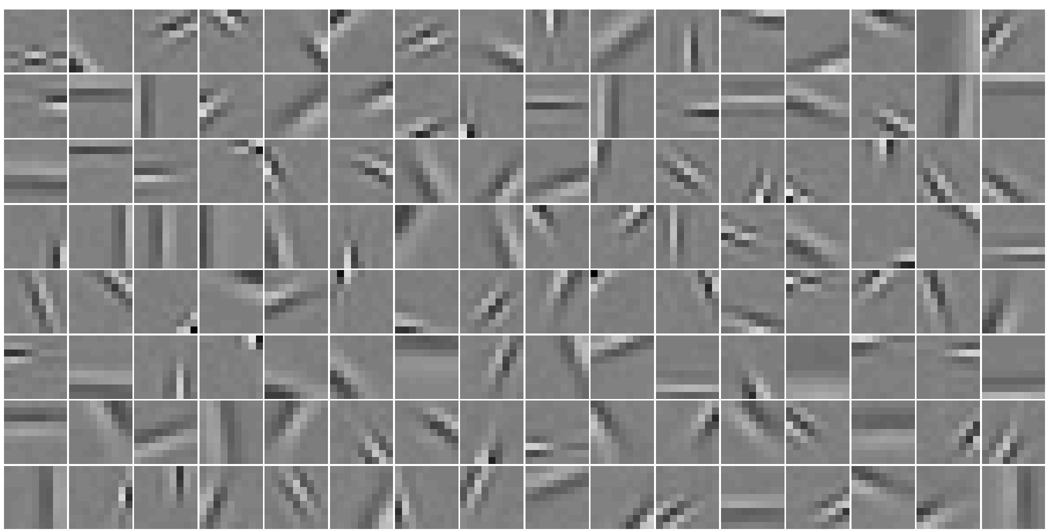


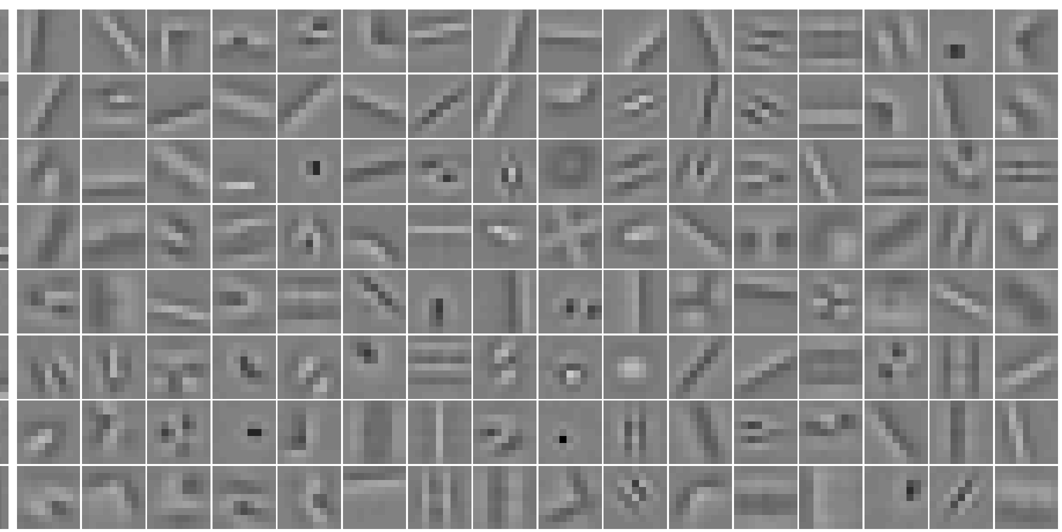
"deconvolutional networks" [Zeiler, Taylor, Fergus CVPR 2010]

Convolutional PSD: Encoder with a soft sh() Function

- **Convolutional Formulation**
  - Extend sparse coding from ***PATCH*** to ***IMAGE***

$$\mathcal{L}(x, z, \mathcal{D}) = \frac{1}{2}||x - \sum_{k=1}^{K} \mathcal{D}_k * z_k||_2^2 + \sum_{k=1}^{K} ||z_k - f(W^k * x)||_2^2 + |z|_1$$

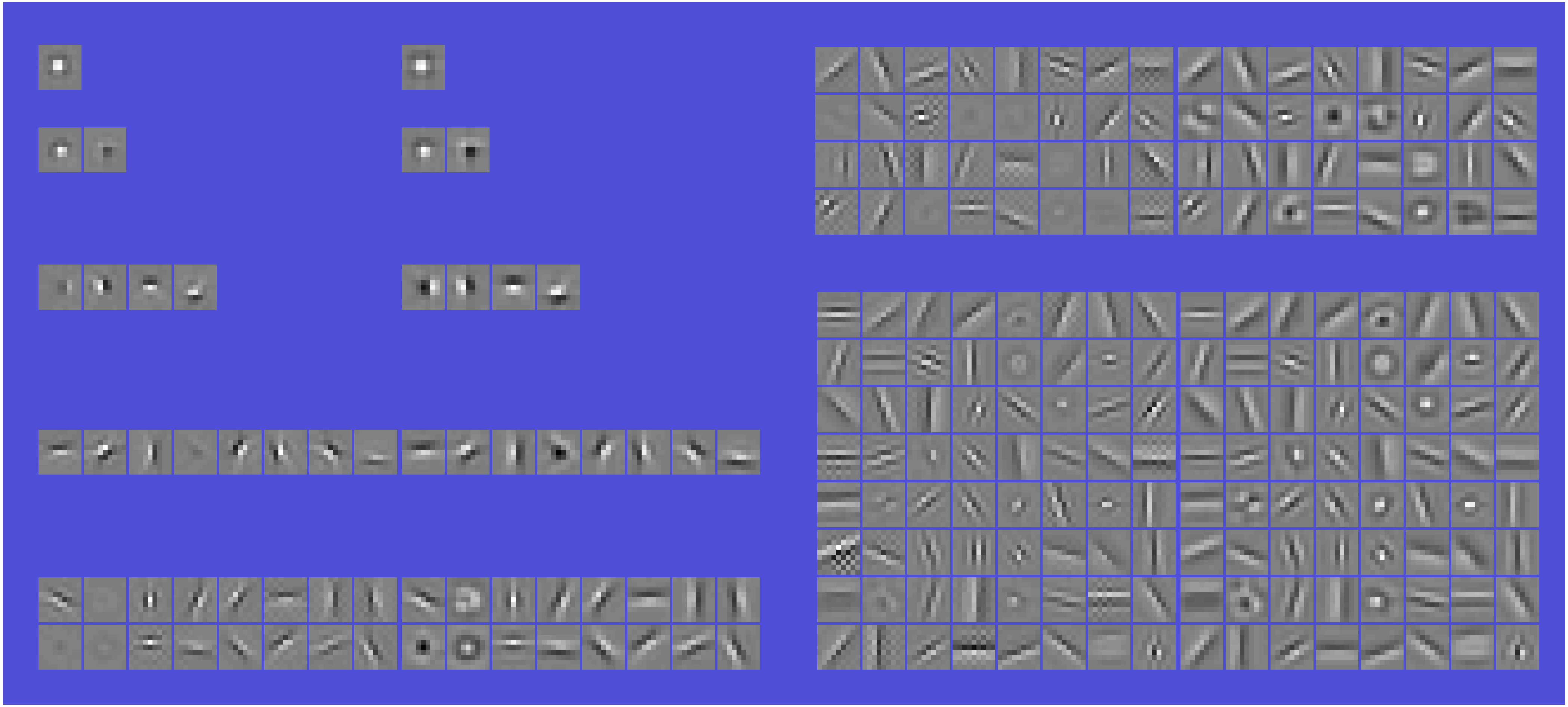




- ***PATCH*** based learning
- ***CONVOLUTIONAL*** learning

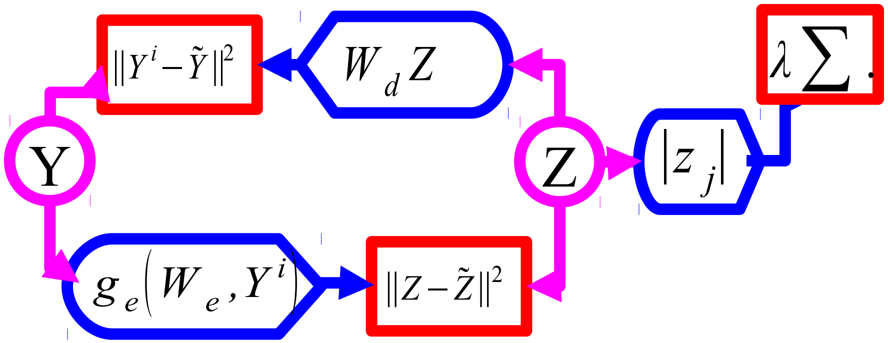
Convolutional Sparse Auto-Encoder on Natural Images

- Filters and Basis Functions obtained with 1, 2, 4, 8, 16, 32, and 64 filters.



Using PSD to Train a Hierarchy of Features

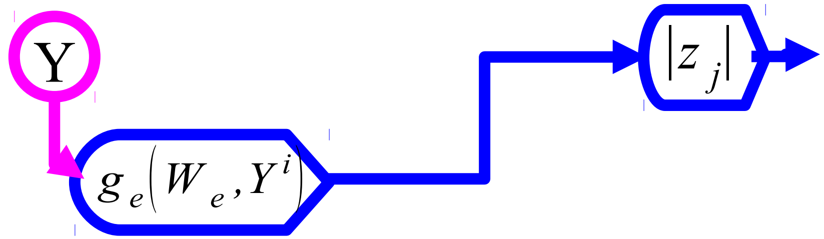
- Phase 1: train first layer using PSD



FEATURES

Using PSD to Train a Hierarchy of Features

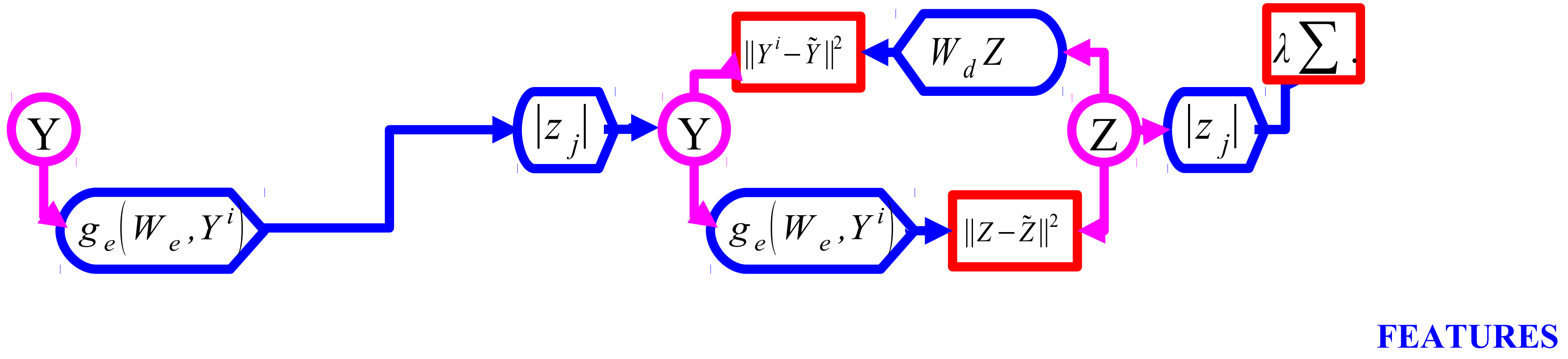
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor



FEATURES

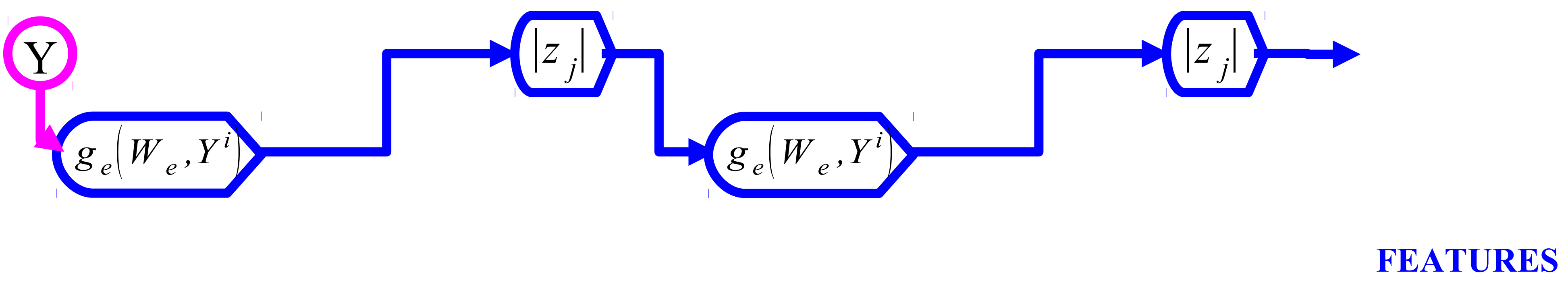
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD



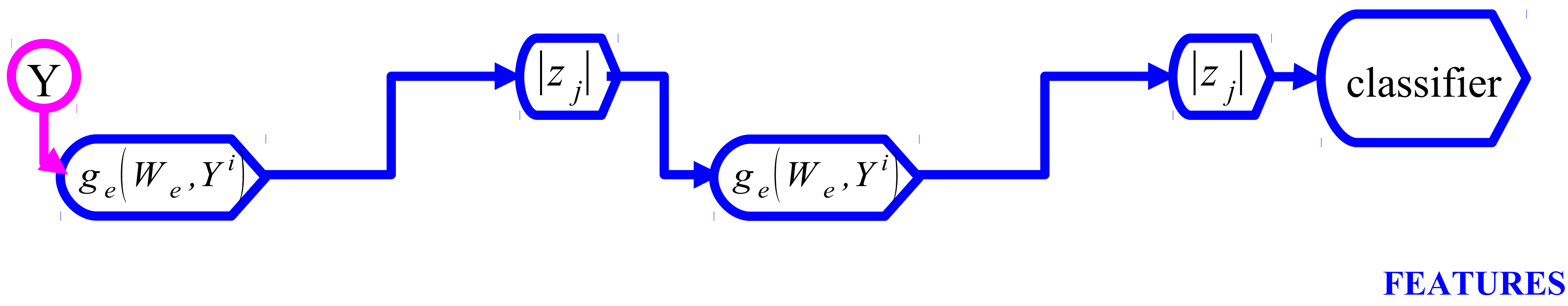
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor

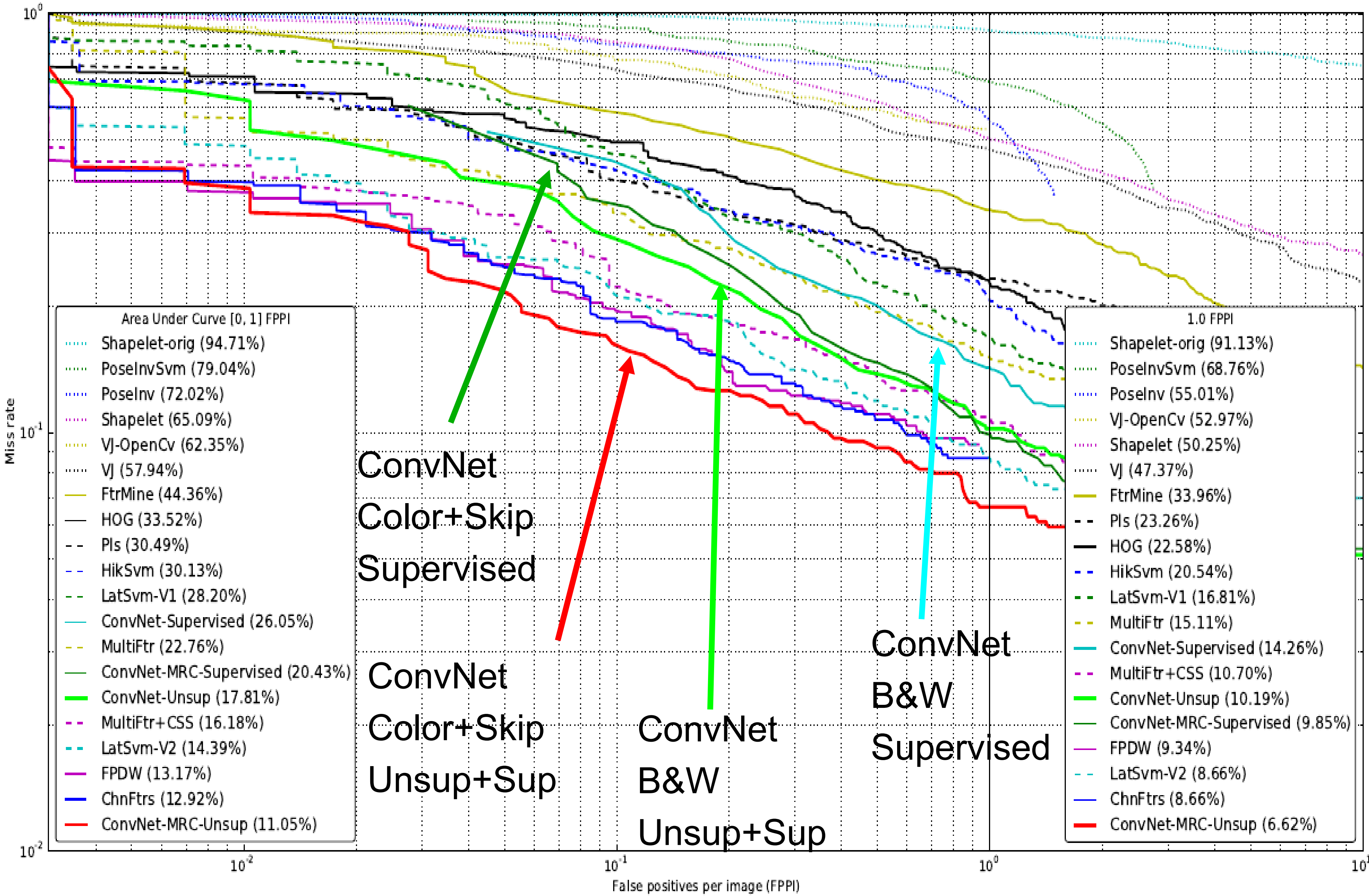


Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor
- Phase 5: train a supervised classifier on top
- Phase 6 (optional): train the entire system with supervised back-propagation



Pedestrian Detection: INRIA Dataset. Miss rate vs false positives



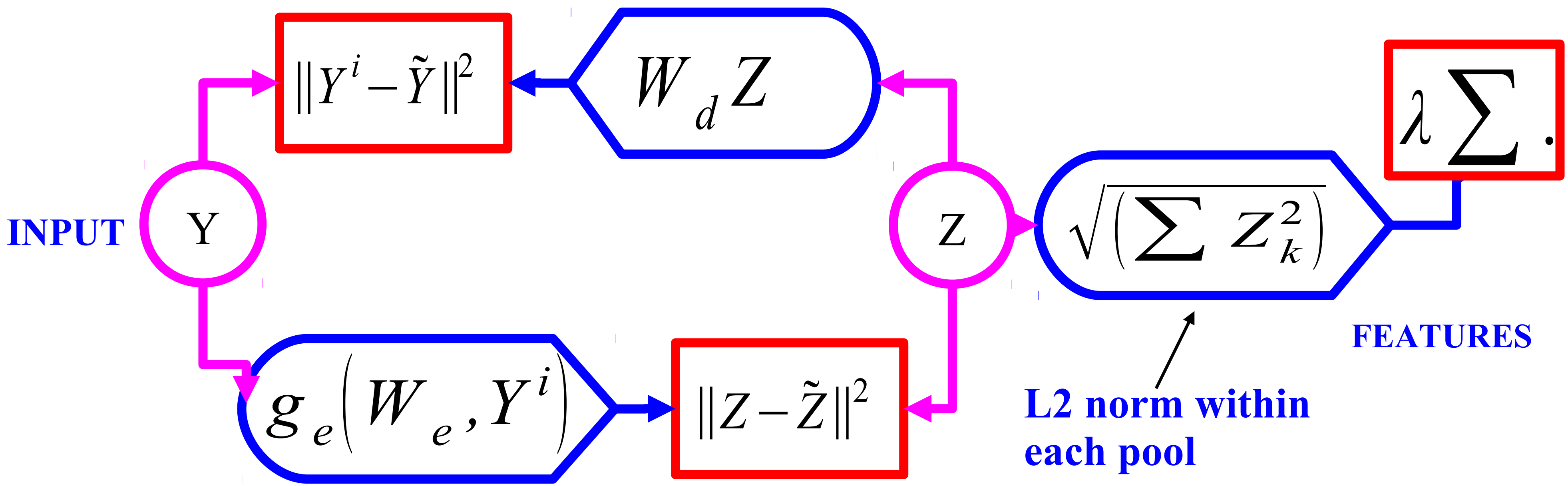
[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]

Unsupervised Learning: Invariant Features

Learning Invariant Features with L2 Group Sparsity

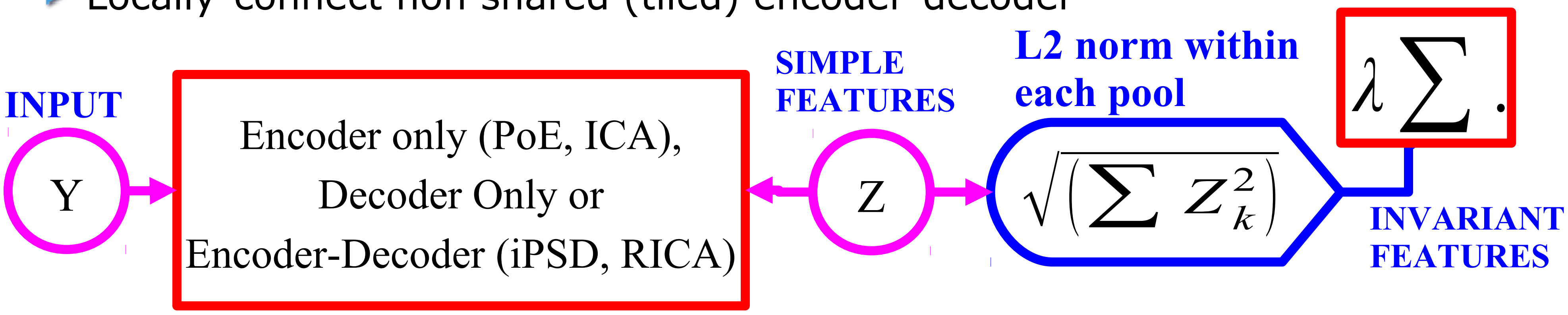
- Unsupervised PSD ignores the spatial pooling step.
- Could we devise a similar method that learns the pooling layer as well?
- Idea [Hyvarinen & Hoyer 2001]: group sparsity on pools of features
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Pools tend to regroup similar features

$$E(Y,Z) = \|Y - W_d Z\|^2 + \|Z - g_e(W_e, Y)\|^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}$$



Learning Invariant Features with L2 Group Sparsity

- **Idea: features are pooled in group.**
  - Sparsity: sum over groups of L2 norm of activity in group.
- **[Hyvärinen Hoyer 2001]: "subspace ICA"**
  - decoder only, square
- **[Welling, Hinton, Osindero NIPS 2002]: pooled product of experts**
  - encoder only, overcomplete, log student-T penalty on L2 pooling
- **[Kavukcuoglu, Ranzato, Fergus LeCun, CVPR 2010]: Invariant PSD**
  - encoder-decoder (like PSD), overcomplete, L2 pooling
- **[Le et al. NIPS 2011]: Reconstruction ICA**
  - Same as [Kavukcuoglu 2010] with linear encoder and tied decoder
- **[Gregor & LeCun arXiv:1006:0448, 2010] [Le et al. ICML 2012]**
  - Locally-connect non shared (tiled) encoder-decoder



Groups are local in a 2D Topographic Map

- The filters arrange themselves spontaneously so that similar filters enter the same pool.
- The pooling units can be seen as complex cells
- Outputs of pooling units are invariant to local transformations of the input
  - For some it's translations, for others rotations, or other transformations.

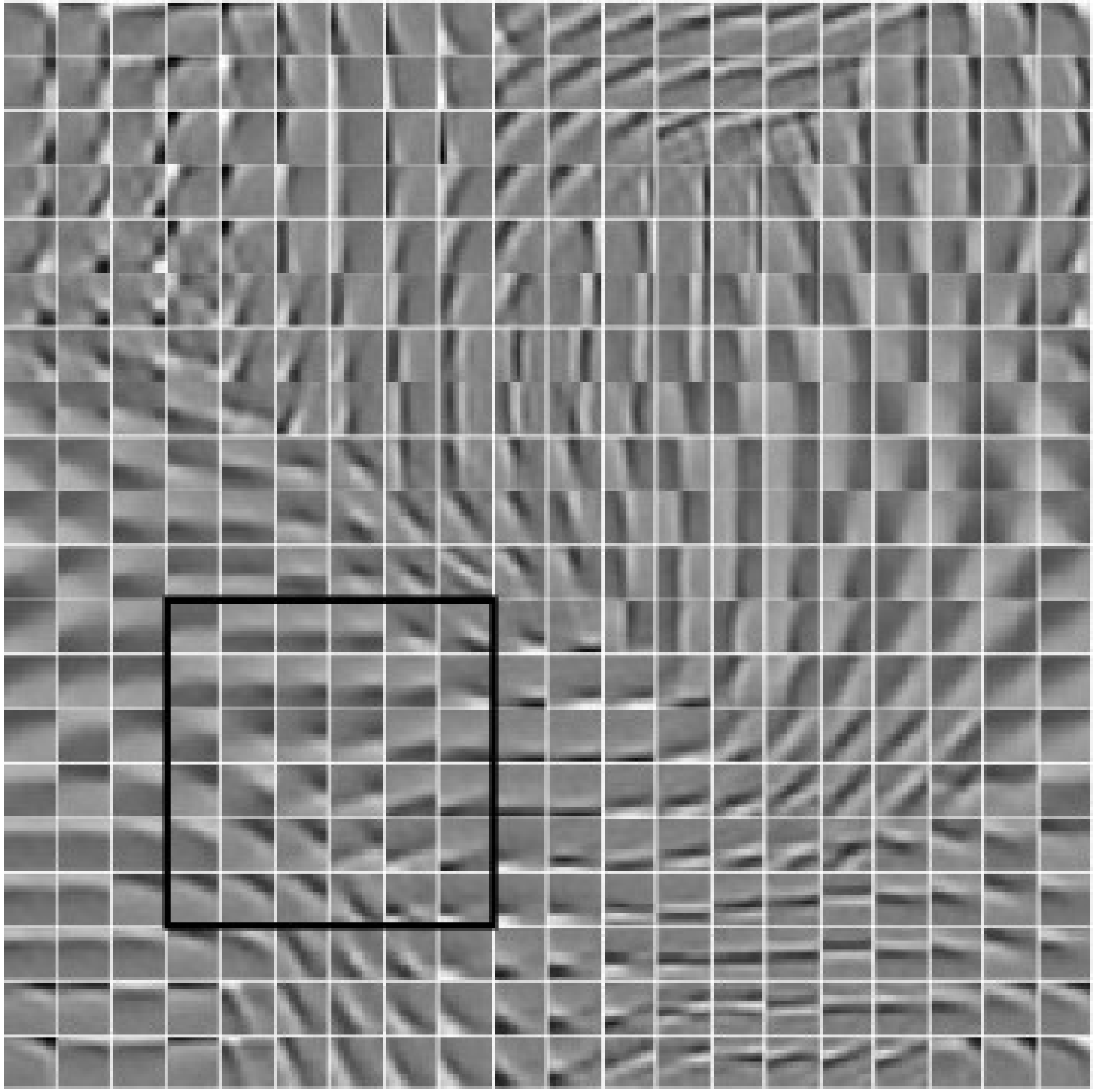


Image-level training, local filters but no weight sharing

- **Training on 115x115 images. Kernels are 15x15 (not shared across space!)**
  - [Gregor & LeCun 2010]
  - Local receptive fields
  - No shared weights
  - 4x overcomplete
  - L2 pooling
  - Group sparsity over pools

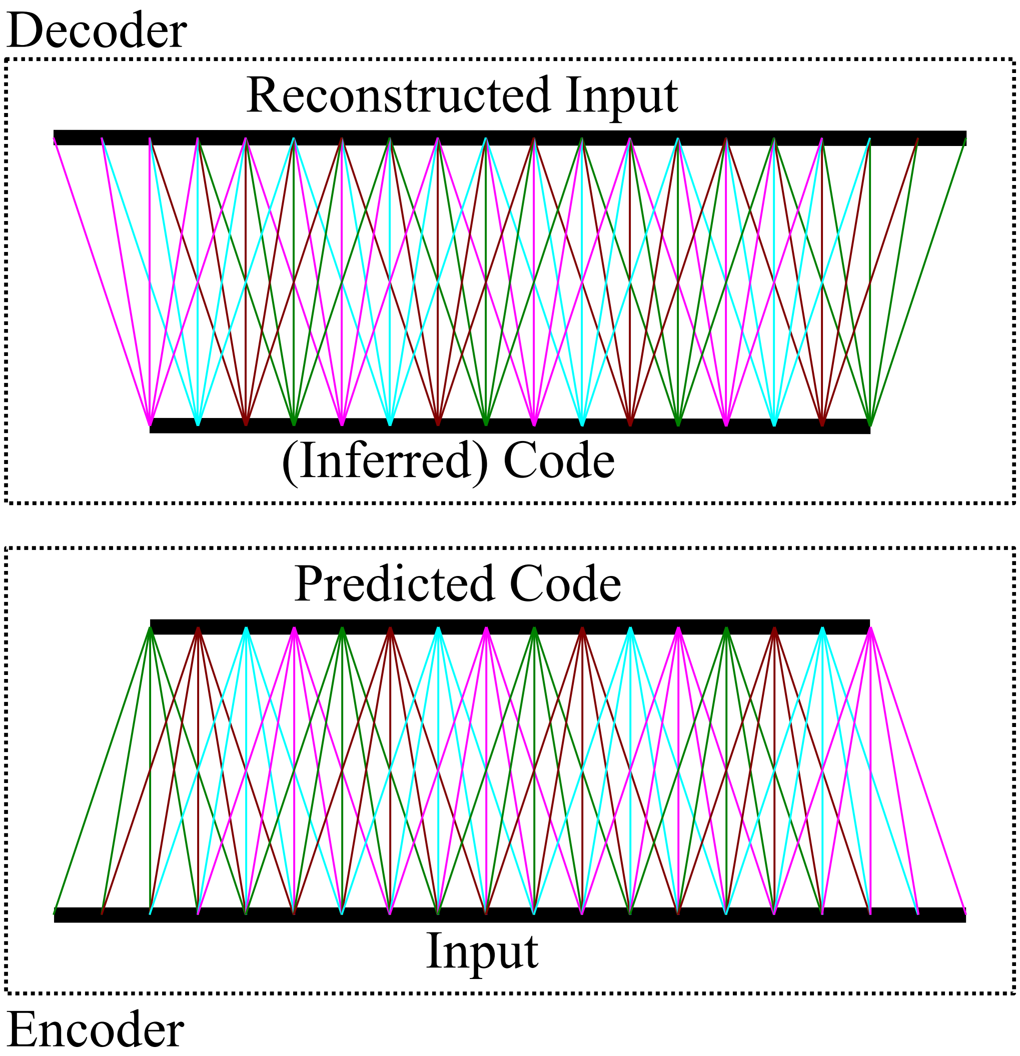
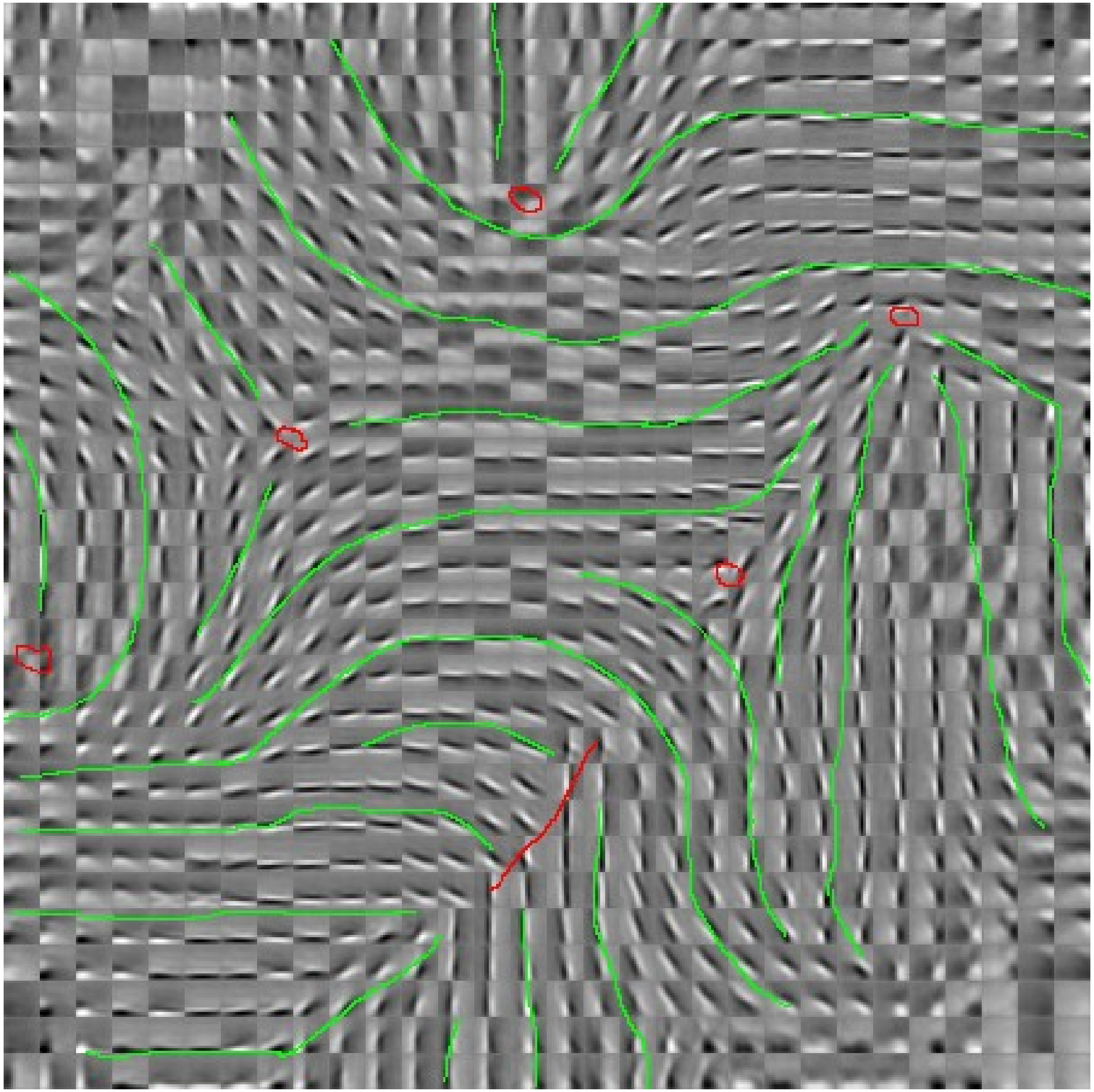
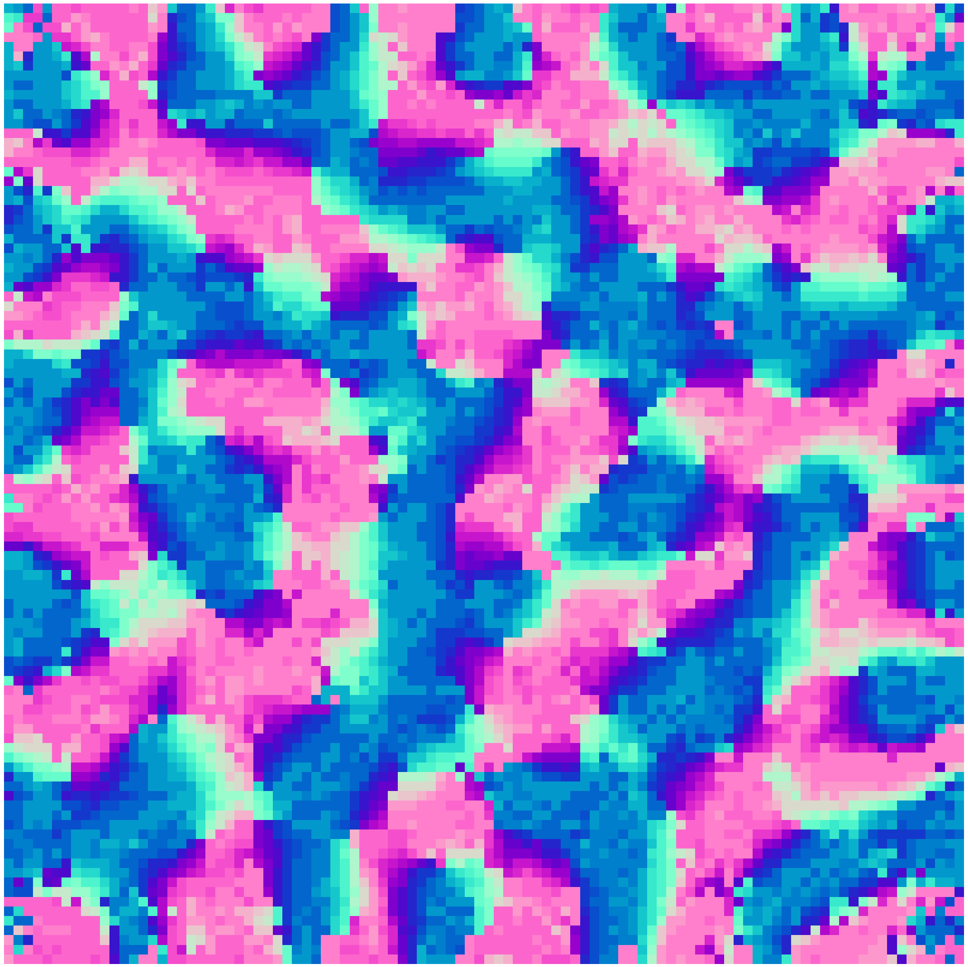


Image-level training, local filters but no weight sharing

- Training on 115x115 images. Kernels are 15x15 (not shared across space!)



Topographic Maps



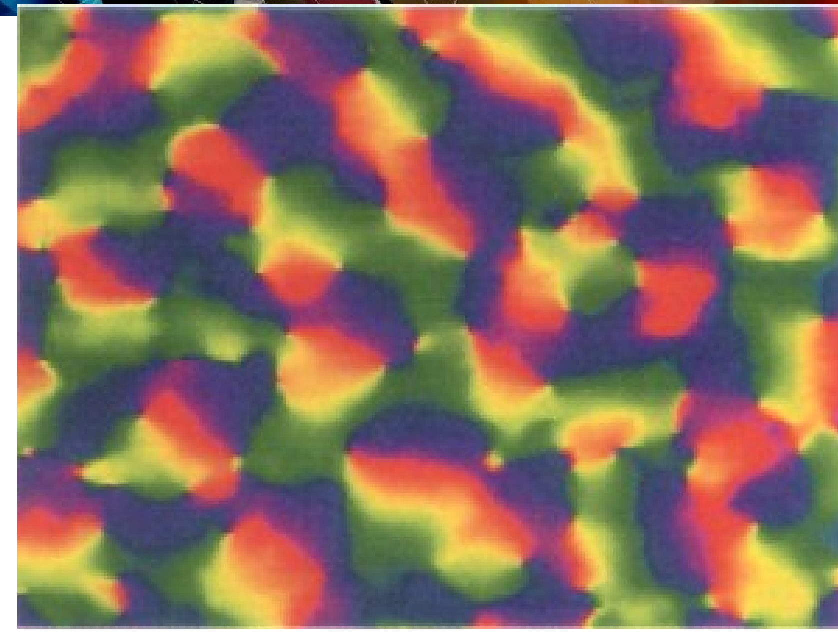
119x119 Image Input

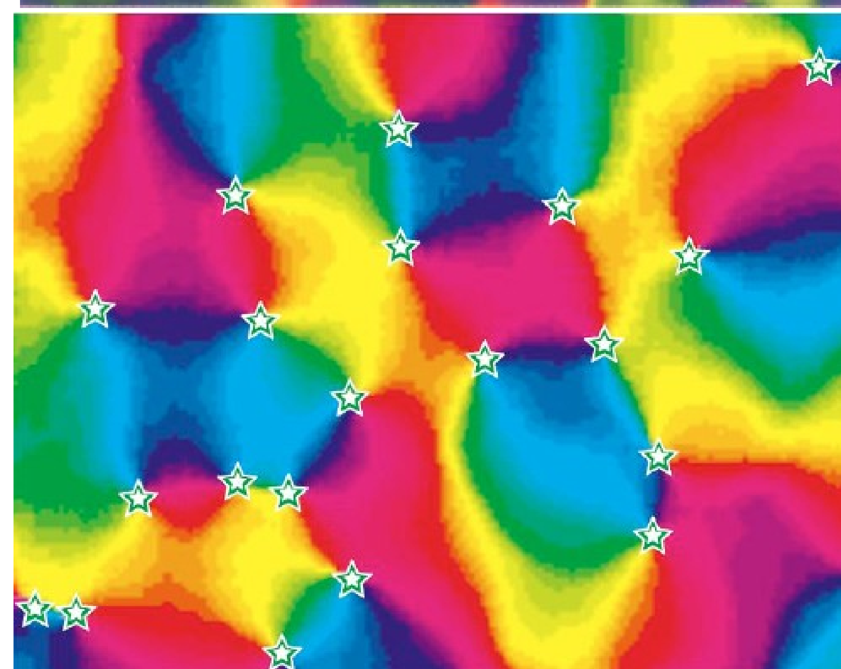
100x100 Code

20x20 Receptive field size

sigma=5

K Obermayer and GG Blasdel, Journal of Neuroscience, Vol 13, 4114-4129 (**Monkey**)

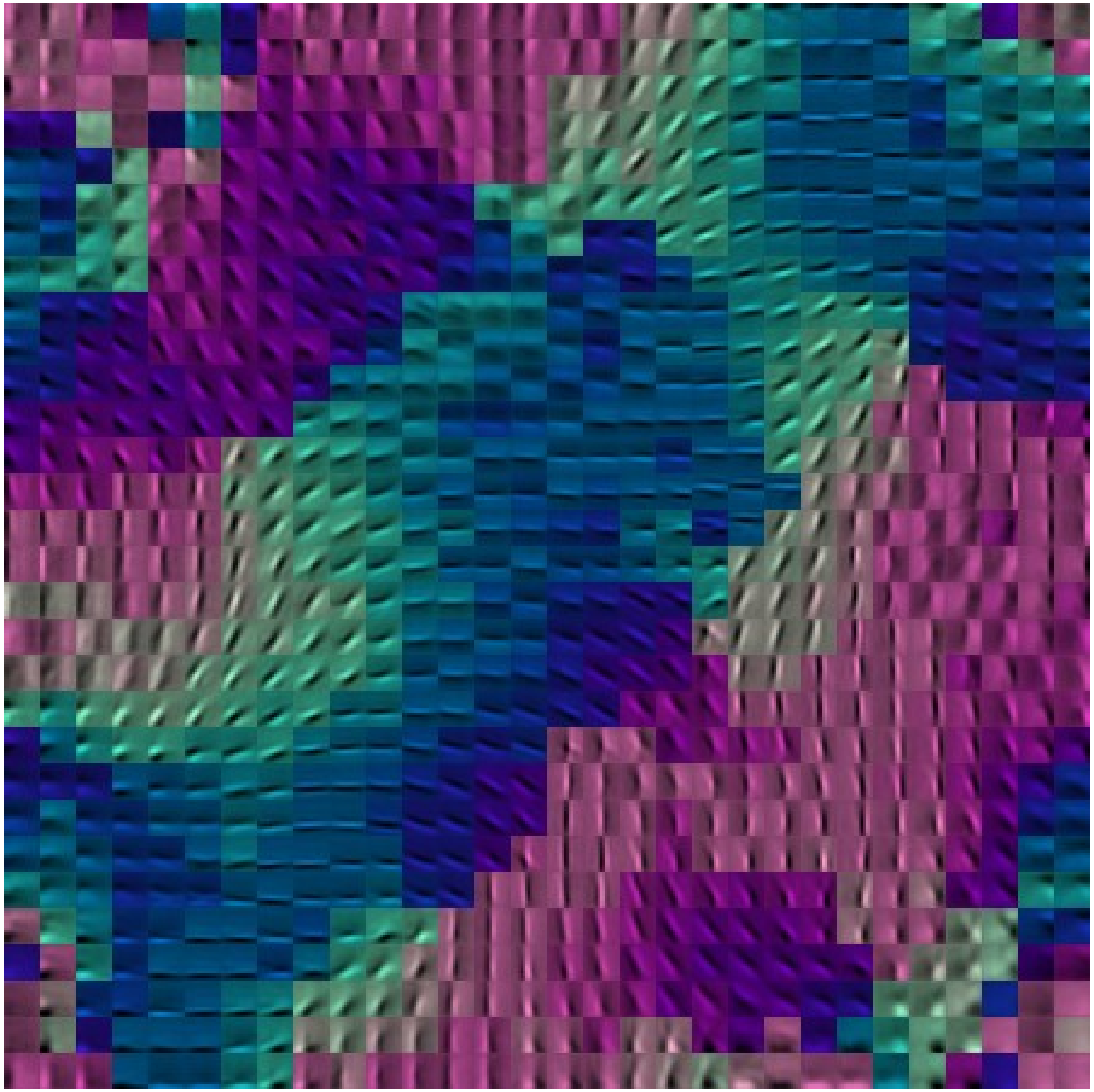




Michael C. Crair, et. al. The Journal of Neurophysiology Vol. 77 No. 6 June 1997, pp. 3381-3385 (**Cat**)

Image-level training, local filters but no weight sharing

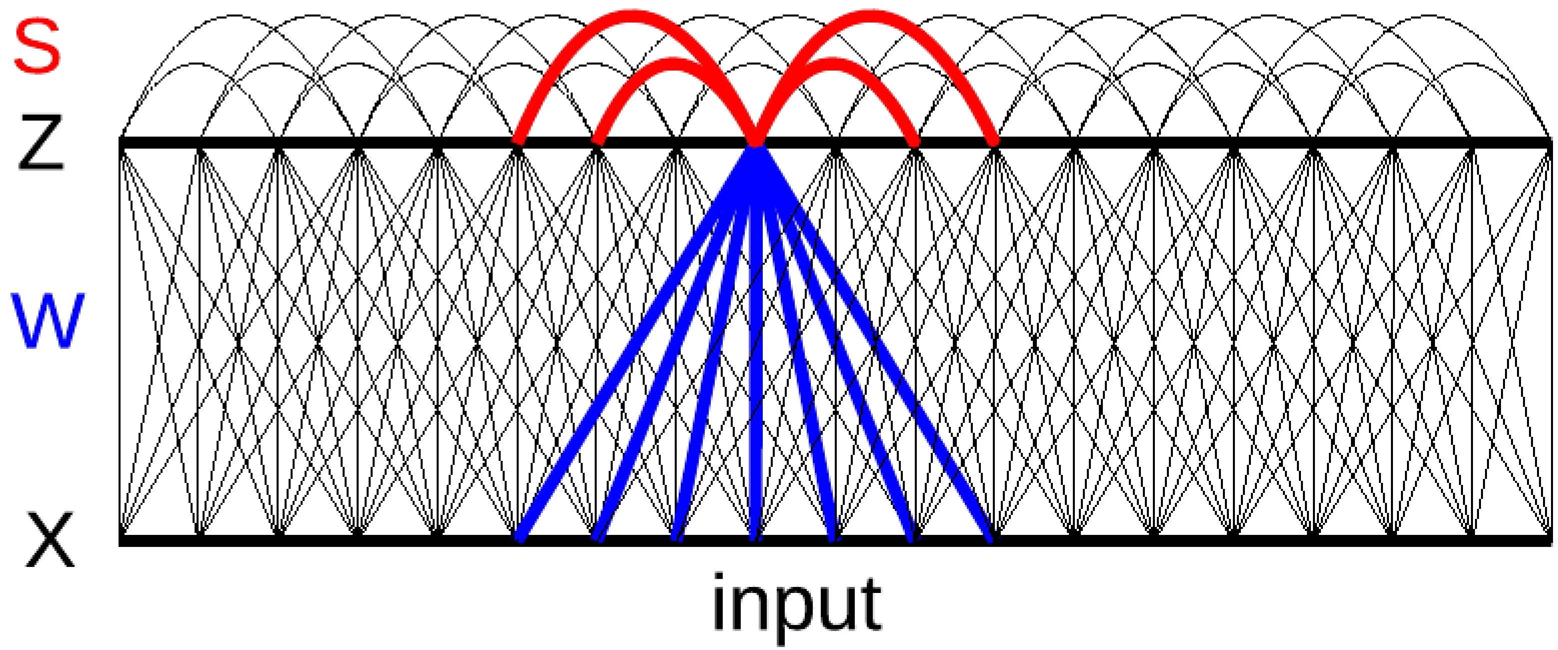
- Color indicates orientation (by fitting Gabors)



Invariant Features Lateral Inhibition

- Replace the L1 sparsity term by a lateral inhibition matrix
- Easy way to impose some structure on the sparsity

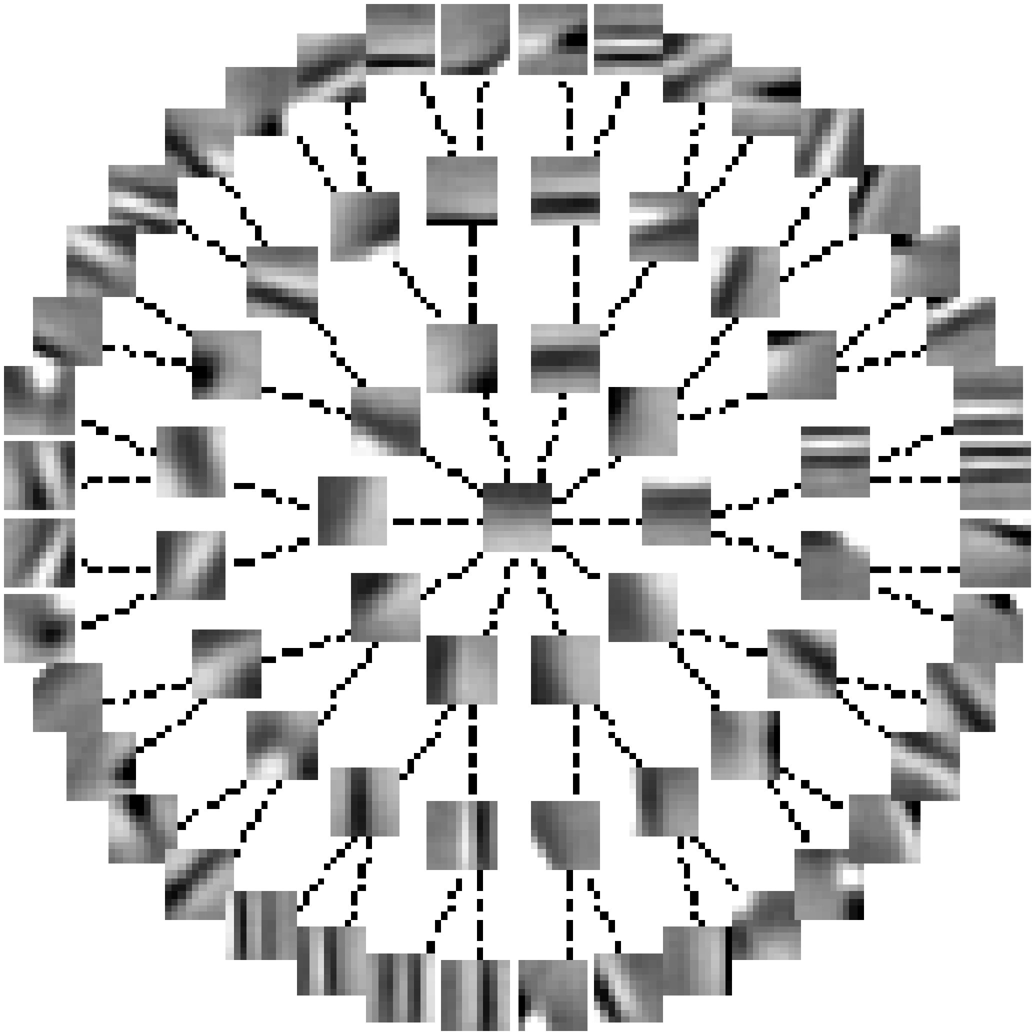
$$\min_{W,Z} \sum_{x \in X} ||Wz - x||^2 + |z|^T S |z|$$



[Gregor, Szlam, LeCun NIPS 2011]

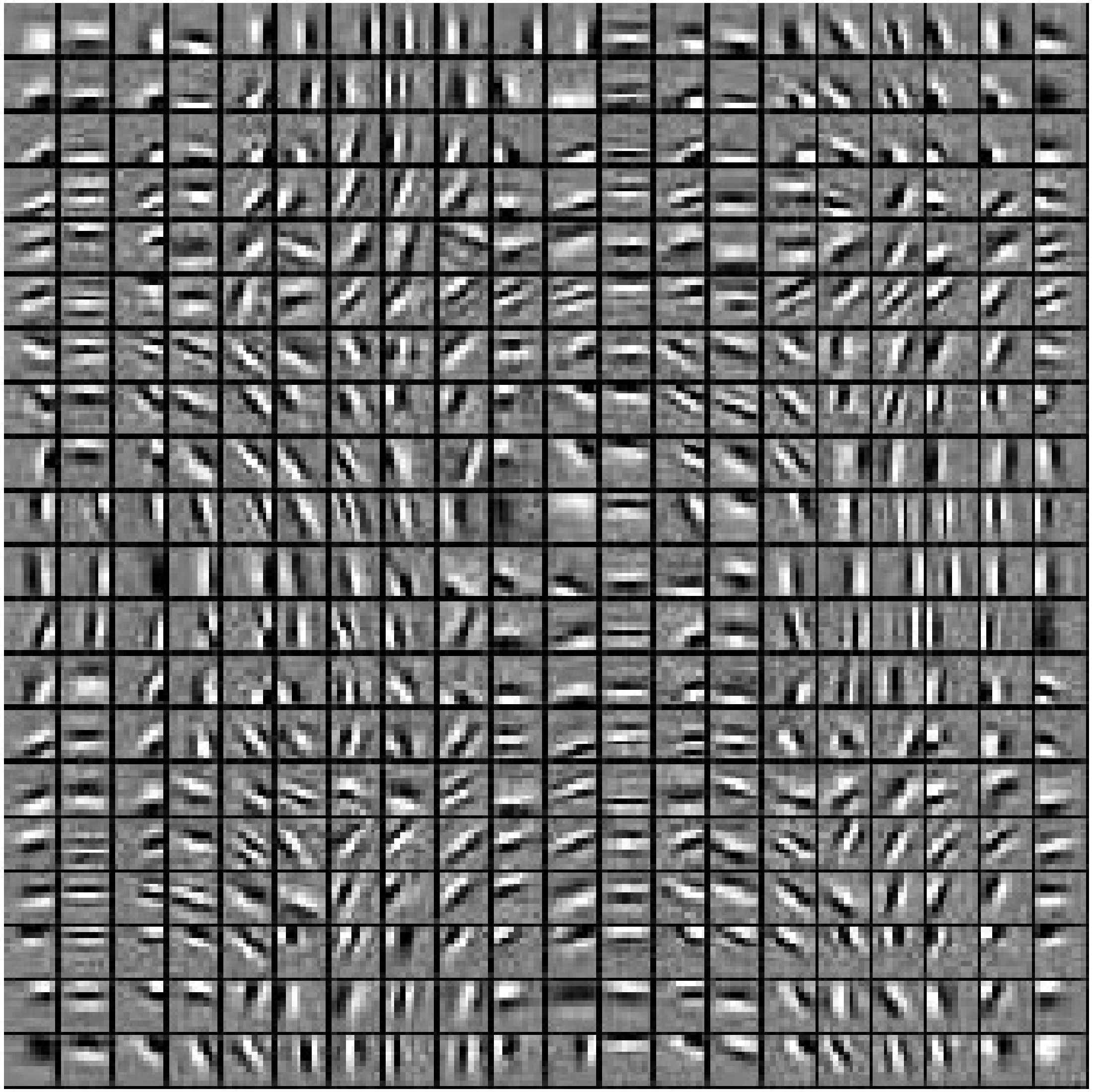
Invariant Features via Lateral Inhibition: Structured Sparsity

- Each edge in the tree indicates a zero in the S matrix (no mutual inhibition)
- Sij is larger if two neurons are far away in the tree



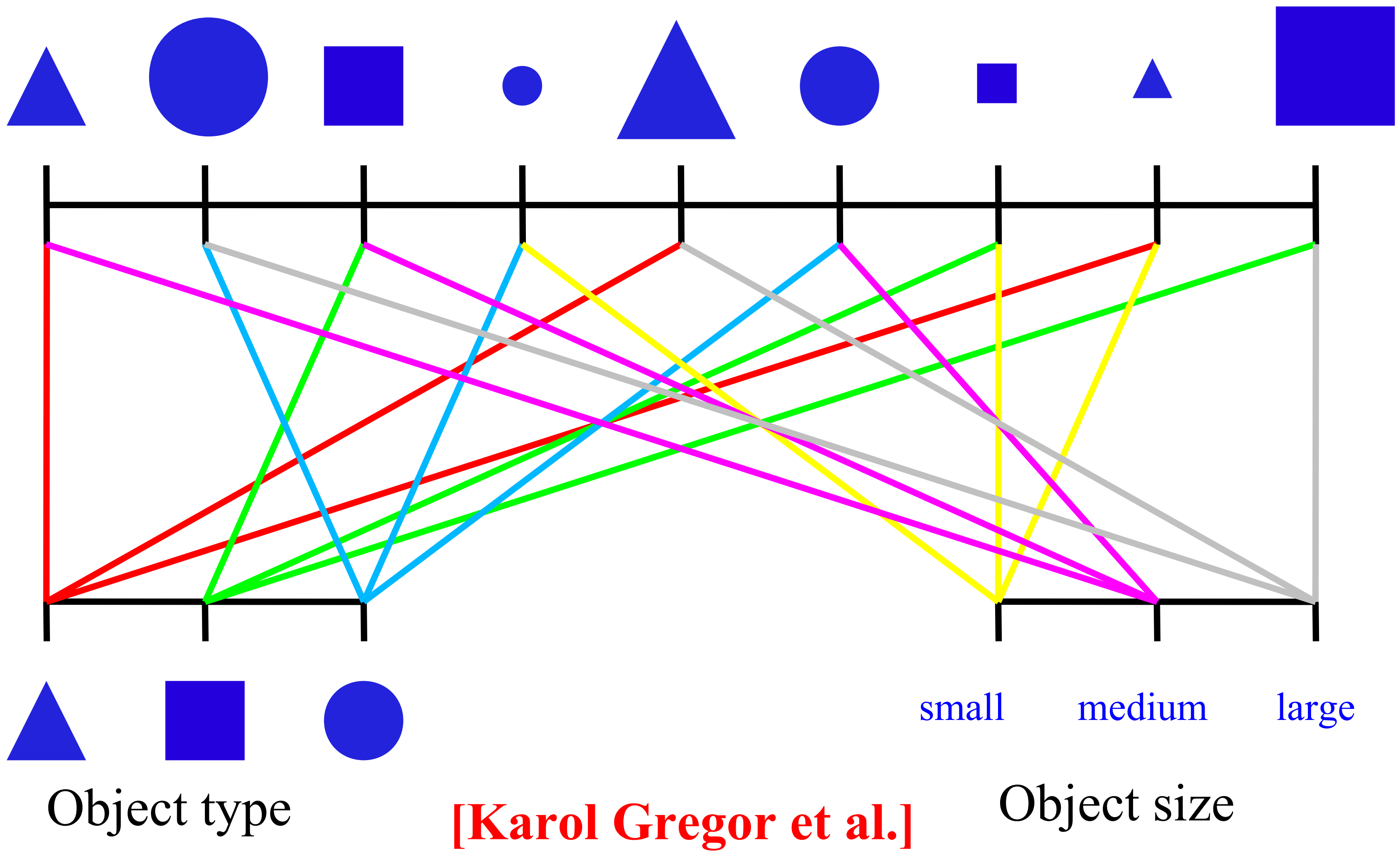
Invariant Features via Lateral Inhibition: Topographic Maps

- **Non-zero values in S form a ring in a 2D topology**
  - Input patches are high-pass filtered



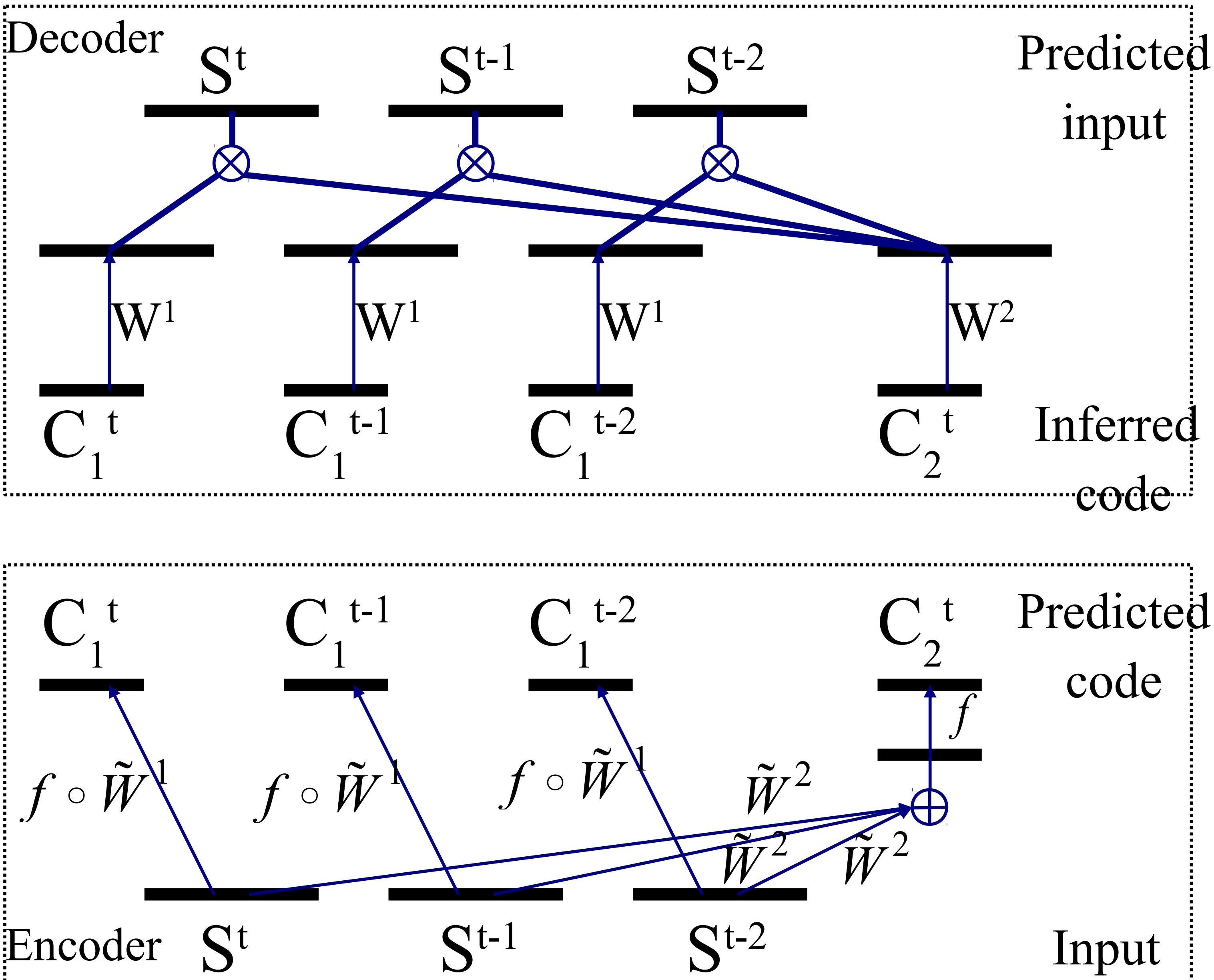
Invariant Features through Temporal Constancy

- **Object is cross-product of object type and instantiation parameters**
  - Mapping units [Hinton 1981], capsules [Hinton 2011]



[Karol Gregor et al.]

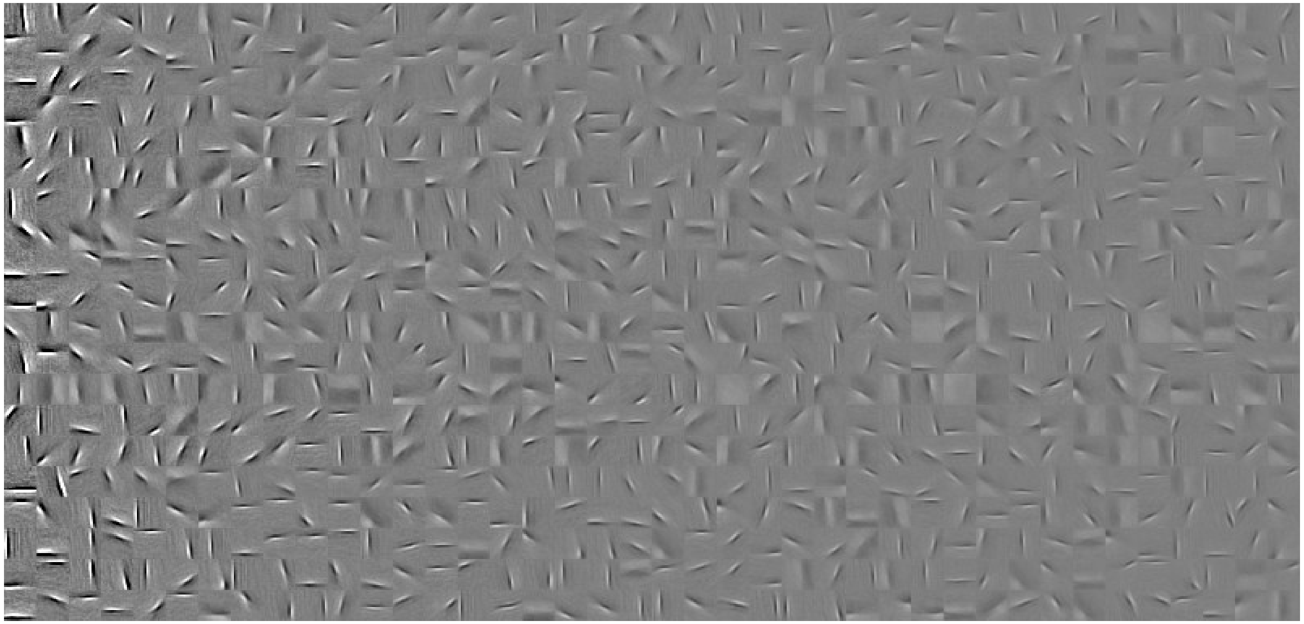
What-Where Auto-Encoder Architecture



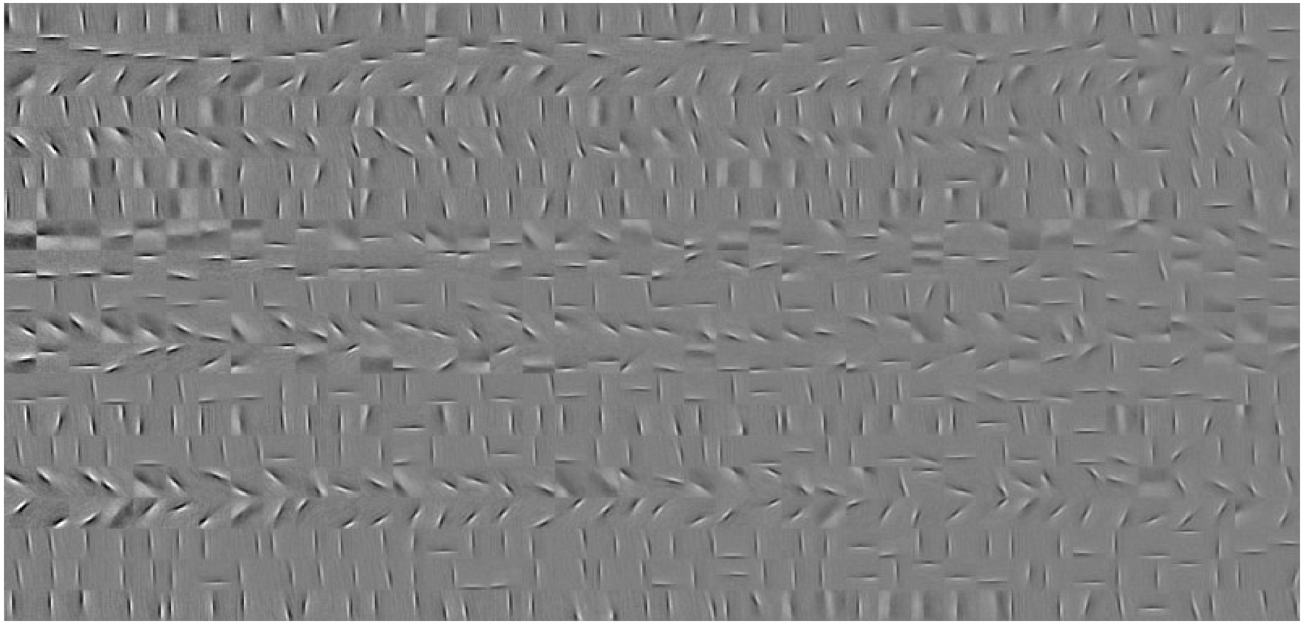
Low-Level Filters Connected to Each Complex Cell

Y LeCun

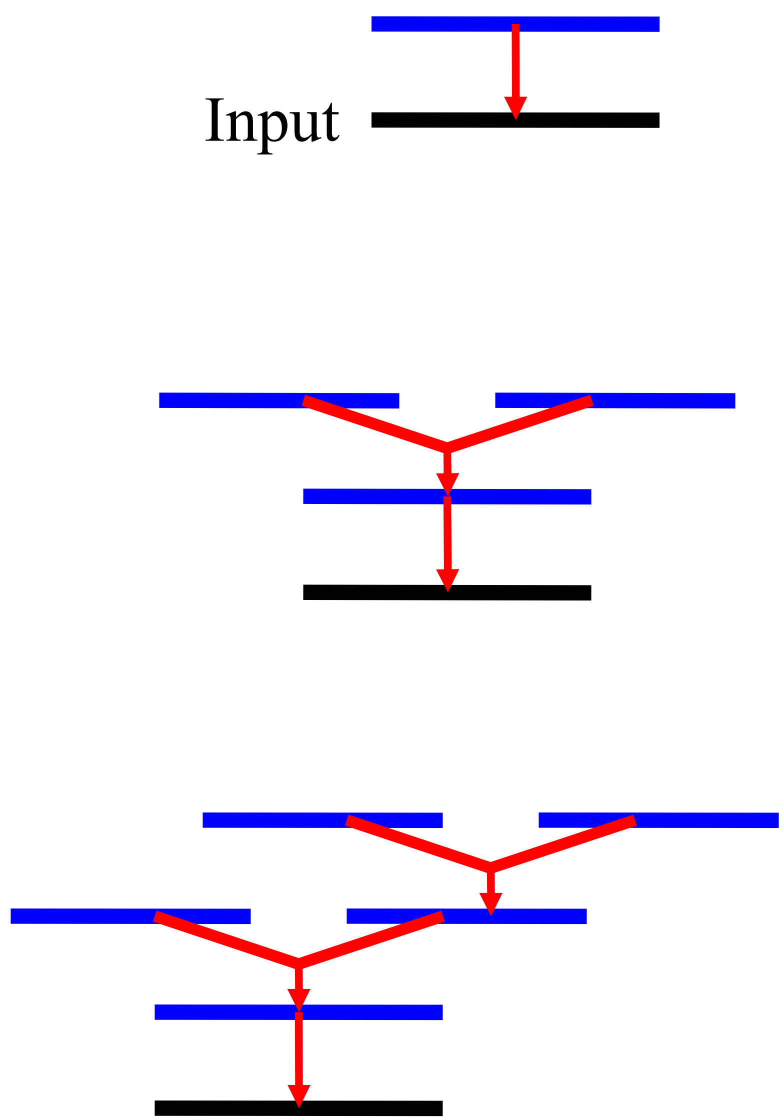
C1
(where)

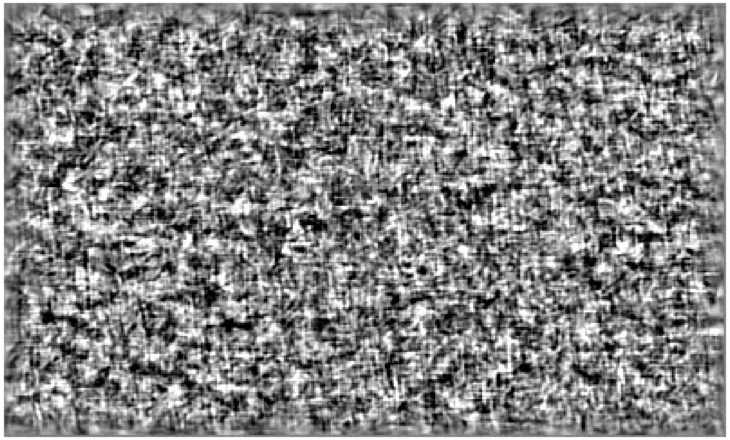


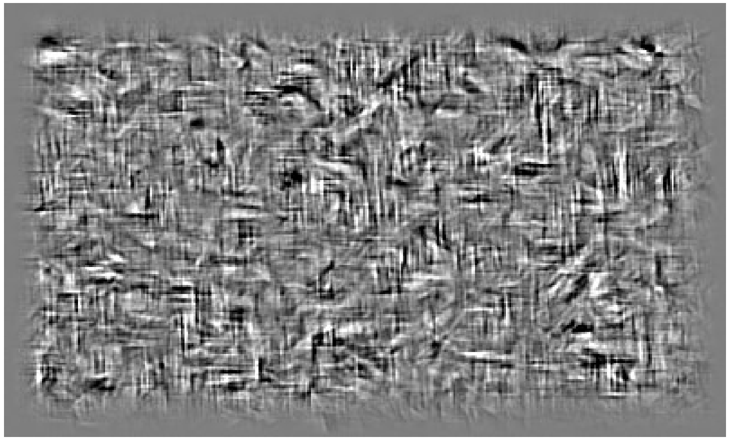
C2
(what)

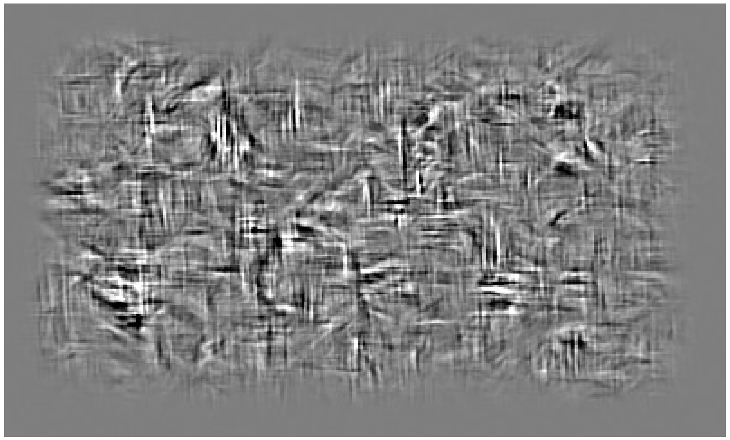


Generating images





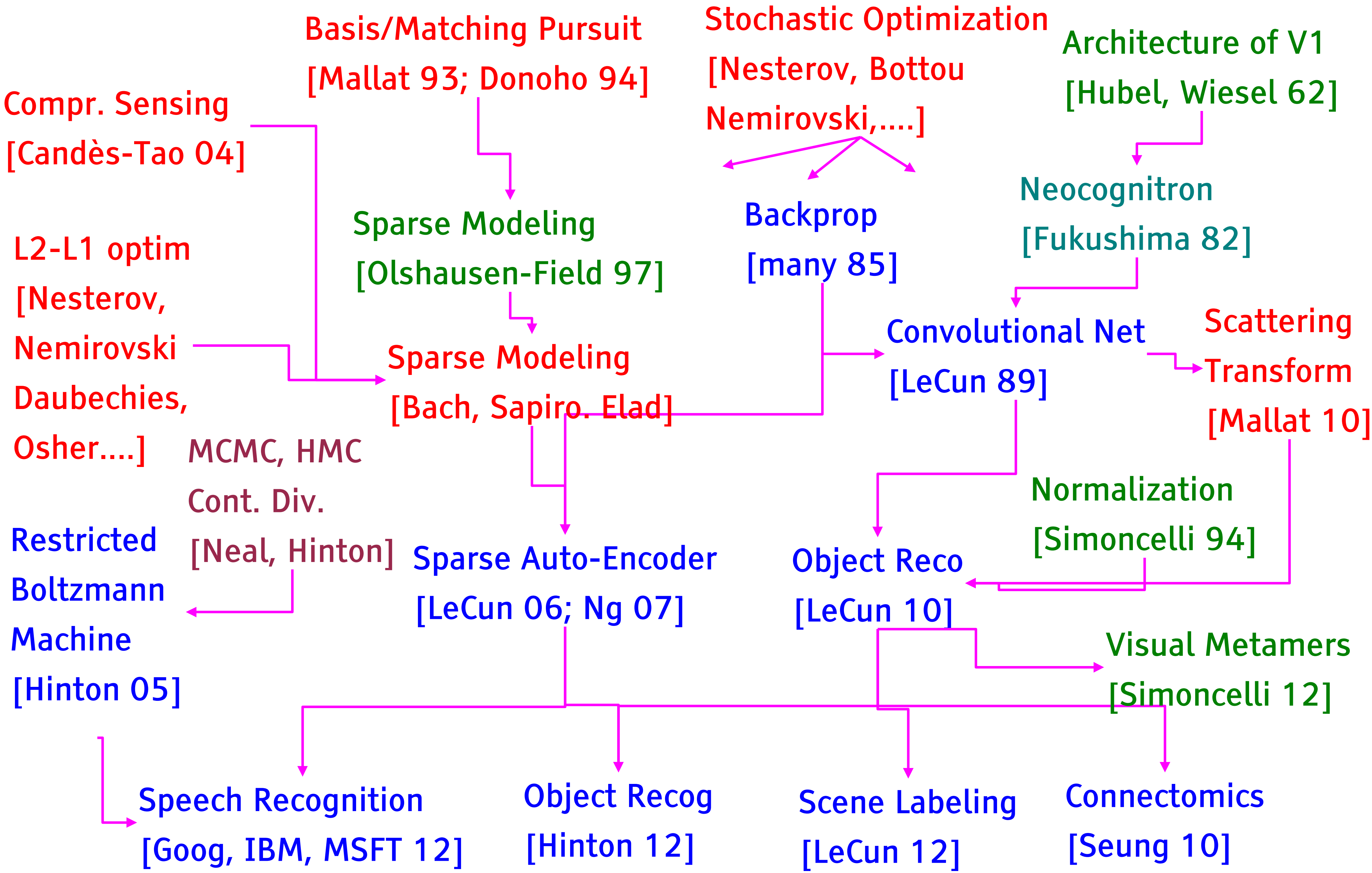




Future Challenges

The Graph of Deep Learning ↔ Sparse Modeling ↔ Neuroscience

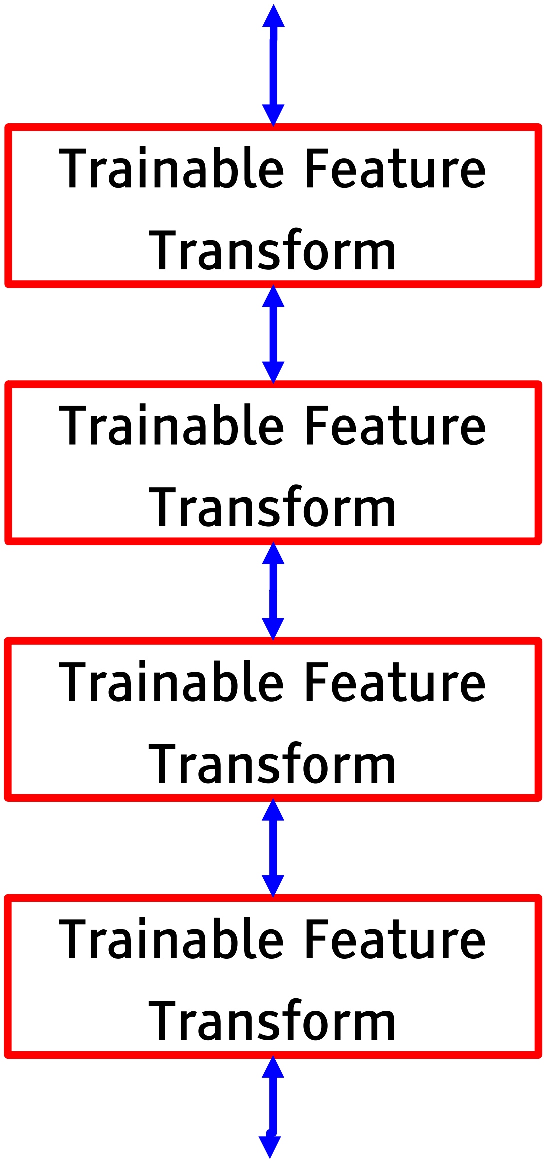
Y LeCun



Integrating Feed-Forward and Feedback

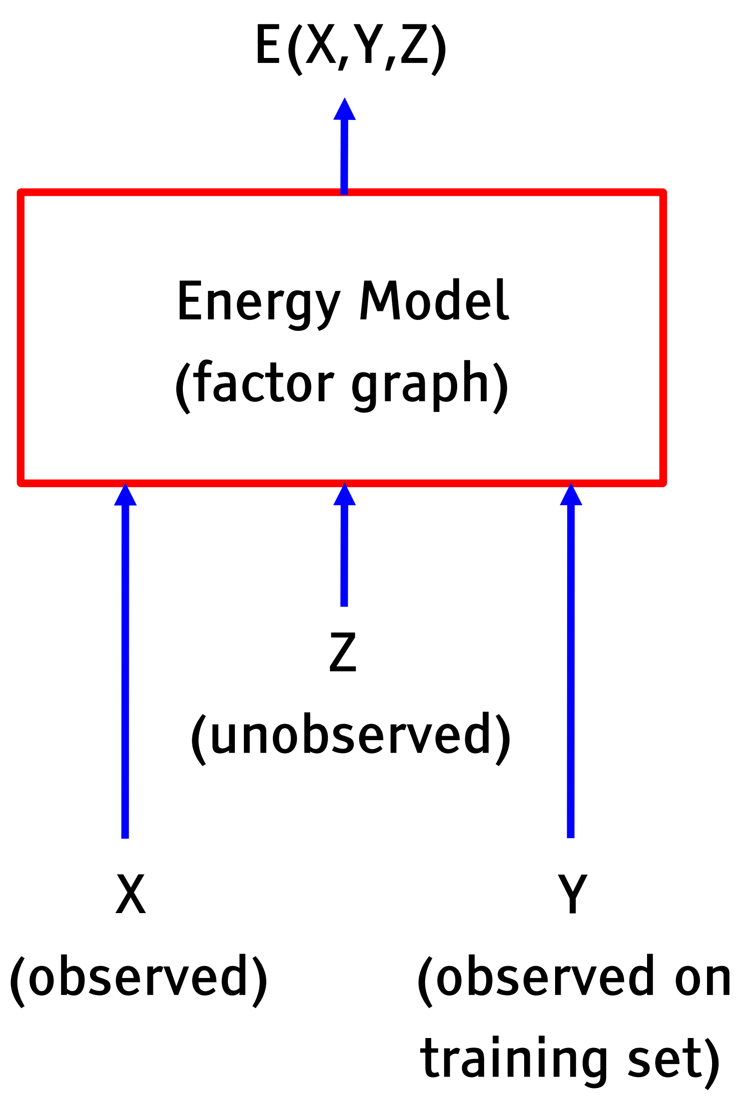
- **Marrying feed-forward convolutional nets with generative "deconvolutional nets"**
  - Deconvolutional networks
    - [Zeiler-Graham-Fergus ICCV 2011]

- **Feed-forward/Feedback networks allow reconstruction, multimodal prediction, restoration, etc...**
  - Deep Boltzmann machines can do this, but there are scalability issues with training



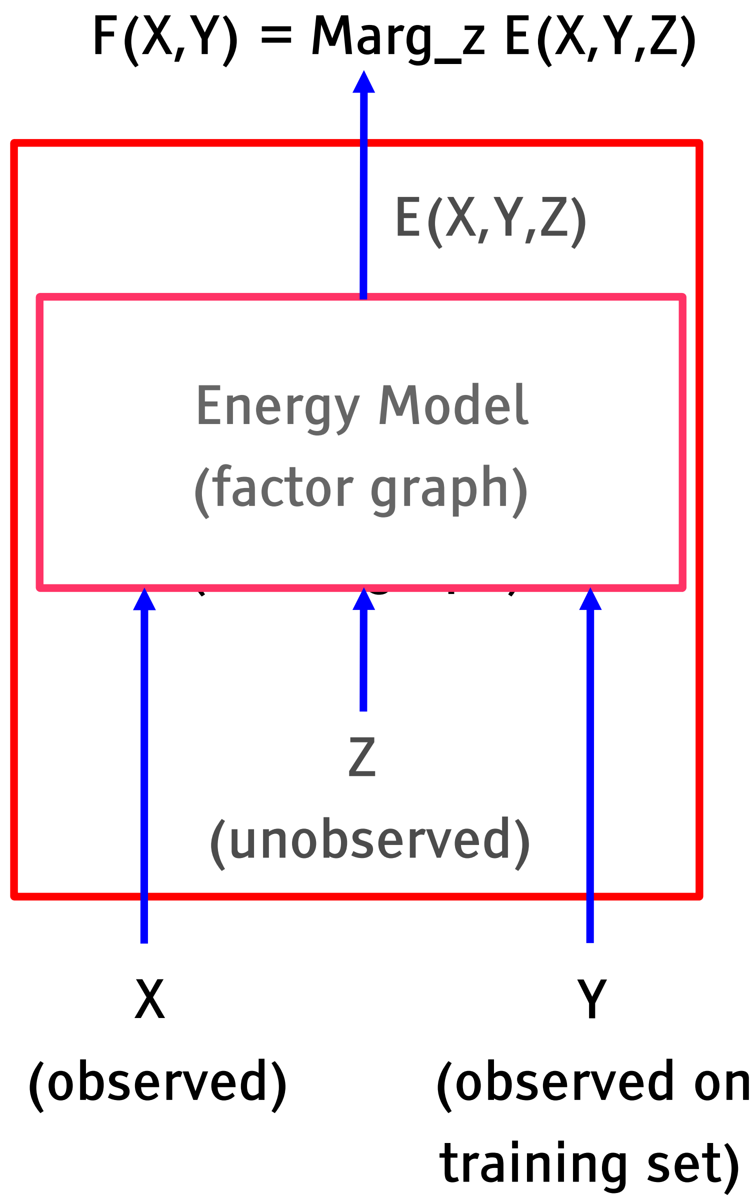
Integrating Deep Learning and Structured Prediction

- **Deep Learning systems can be assembled into factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**



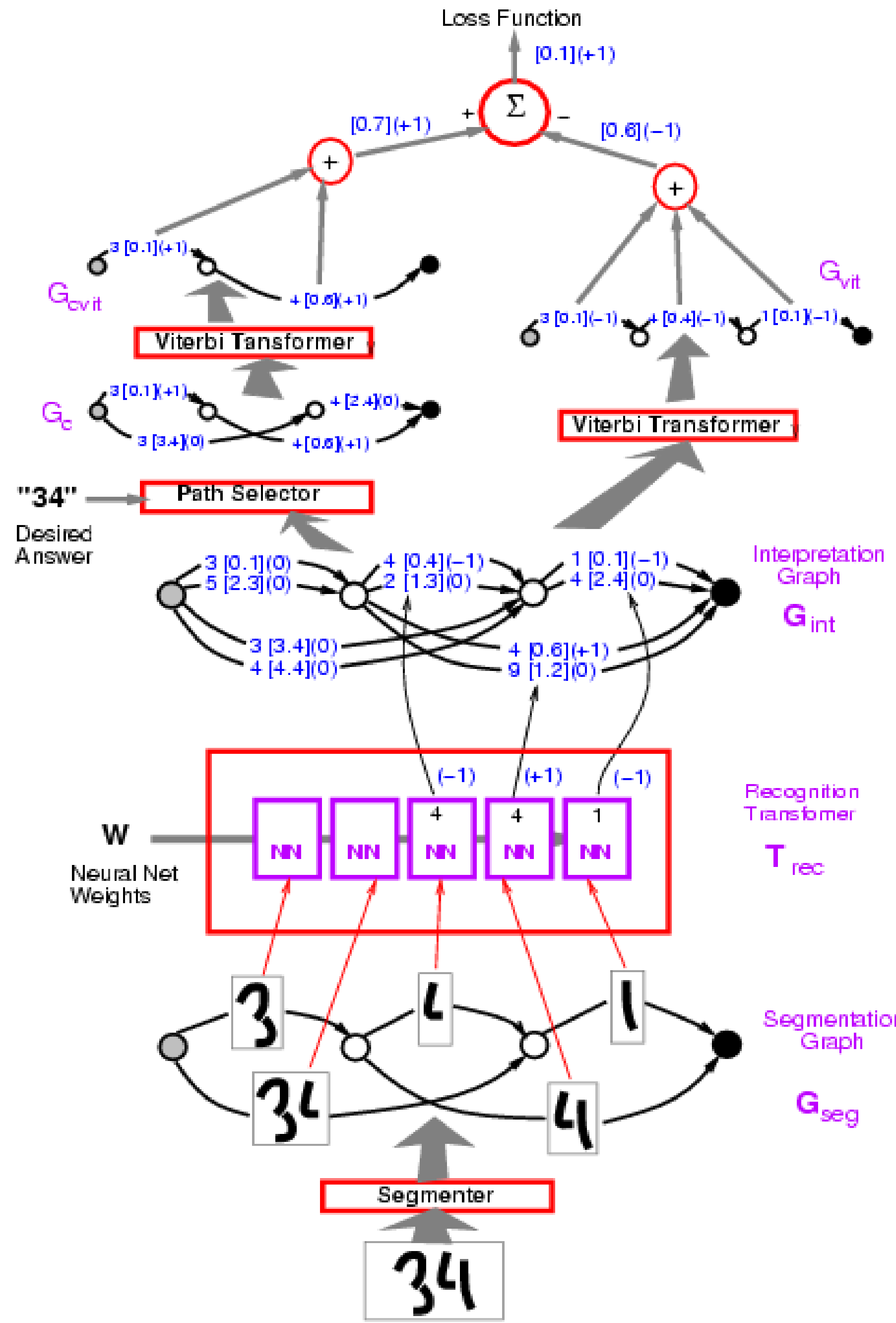
Integrating Deep Learning and Structured Prediction

- **Deep Learning systems can be assembled into factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]



Integrating Deep Learning and Structured Prediction

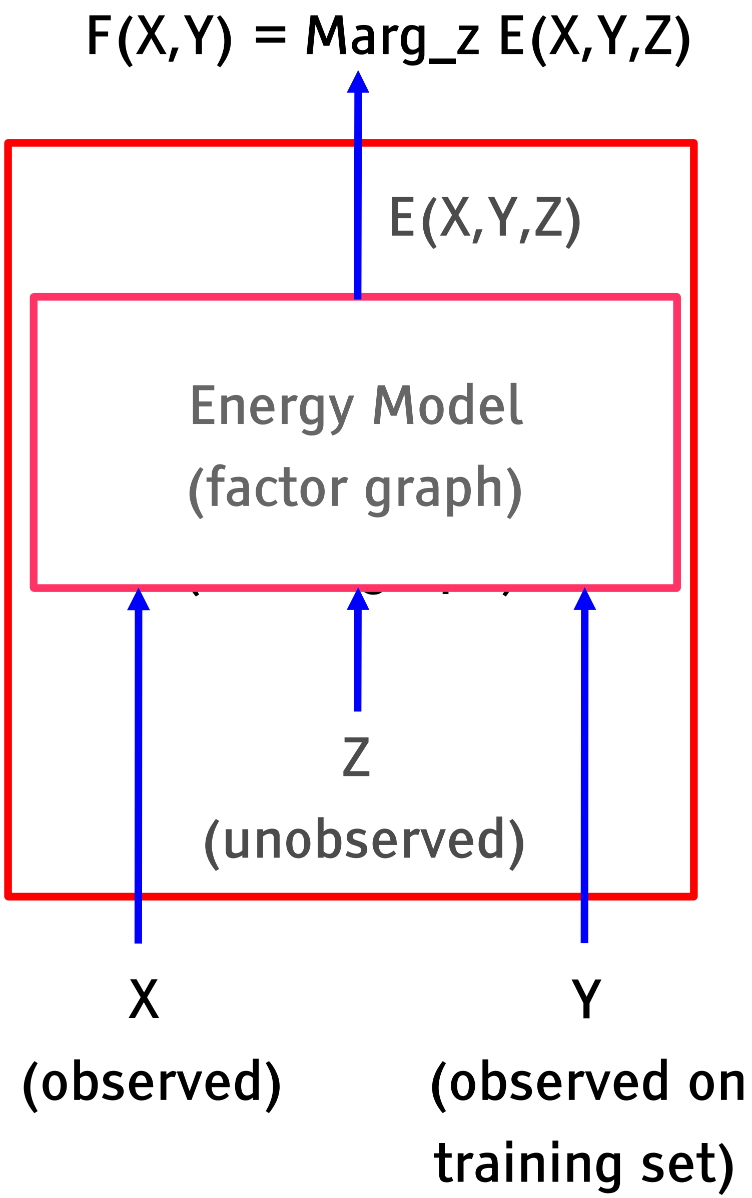
- **Integrting deep learning and structured prediction is a very old idea**
  - In fact, it predates structured prediction
- **Globally-trained convolutional-net + graphical models**
  - trained discriminatively at the word level
  - Loss identical to CRF and structured perceptron
  - Compositional movable parts model
- **A system like this was reading 10 to 20% of all the checks in the US around 1998**



Integrating Deep Learning and Structured Prediction

Y LeCun

- **Deep Learning systems can be assembled into factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]



Future Challenges

- **Integrated feed-forward and feedback**
  - Deep Boltzmann machine do this, but there are issues of scalability.
- **Integrating supervised and unsupervised learning in a single algorithm**
  - Again, deep Boltzmann machines do this, but....
- **Integrating deep learning and structured prediction ("reasoning")**
  - This has been around since the 1990's but needs to be revived
- **Learning representations for complex reasoning**
  - "recursive" networks that operate on vector space representations of knowledge [Pollack 90's] [Bottou 2010] [Socher, Manning, Ng 2011]
- **Representation learning in natural language processing**
  - [Y. Bengio 01],[Collobert Weston 10], [Mnih Hinton 11] [Socher 12]
- **Better theoretical understanding of deep learning and convolutional nets**
  - e.g. Stephane Mallat's "scattering transform", work on the sparse representations from the applied math community....

Communities

- DeepLearning.net
  - http://deeplearning.net
  - Maintained by Yoshua Bengio's group
- International Conference on Learning Representations
  - https://sites.google.com/site/representationlearning2013/
  - Open review system
  - Papers and videos available online
  - Takes place in April
  - Extended version of selected papers published in JMLR
  - https://plus.google.com/communities/108755902083074010353
- “Deep Learning” community on Google+
  - https://plus.google.com/communities/112866381580457264725

SOFTWARE

Torch7: learning library that supports neural net training

– http://www.torch.ch
– http://code.cogbits.com/wiki/doku.php (tutorial with demos by C. Farabet)
- http://eblearn.sf.net (C++ Library with convnet support by P. Sermanet)

Python-based learning library (U. Montreal)

- http://deeplearning.net/software/theano/ (does automatic differentiation)

RNN

– www.fit.vutbr.cz/~imikolov/rnnlm (language modeling)
– http://sourceforge.net/apps/mediawiki/rnnl/index.php (LSTM)

CUDAMat & GNumpy

– code.google.com/p/cudamat
– www.cs.toronto.edu/~tijmen/gnumpy.html

Misc

– www.deeplearning.net//software_links

REFERENCES

Convolutional Nets

– LeCun, Bottou, Bengio and Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

- Krizhevsky, Sutskever, Hinton "ImageNet Classification with deep convolutional neural networks" NIPS 2012

– Jarrett, Kavukcuoglu, Ranzato, LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009

- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural Information Processing Systems (NIPS 2010), 23, 2010

– see yann.lecun.com/exdb/publis for references on many different kinds of convnets.

– see http://www.cmap.polytechnique.fr/scattering/ for scattering networks (similar to convnets but with less learning and stronger mathematical foundations)

REFERENCES

Applications of Convolutional Nets

– Farabet, Couprie, Najman, LeCun, “Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers”, ICML 2012

– Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala and Yann LeCun: Pedestrian Detection with Unsupervised Multi-Stage Feature Learning, CVPR 2013

- D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. NIPS 2012

- Raia Hadsell, Pierre Sermanet, Marco Scoffier, Ayse Erkan, Koray Kavackuoglu, Urs Muller and Yann LeCun: Learning Long-Range Vision for Autonomous Off-Road Driving, Journal of Field Robotics, 26(2):120-144, February 2009

– Burger, Schuler, Harmeling: Image Denoisng: Can Plain Neural Networks Compete with BM3D?, Computer Vision and Pattern Recognition, CVPR 2012,

REFERENCES

Applications of RNNs

– Mikolov “Statistical language models based on neural networks” PhD thesis 2012

– Boden “A guide to RNNs and backpropagation” Tech Report 2002

– Hochreiter, Schmidhuber “Long short term memory” Neural Computation 1997

– Graves “Offline arabic handwrting recognition with multidimensional neural networks” Springer 2012

– Graves “Speech recognition with deep recurrent neural networks” ICASSP 2013

REFERENCES

Deep Learning & Energy-Based Models

– Y. Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.

– LeCun, Chopra, Hadsell, Ranzato, Huang: A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Schölkopf, B. and Smola, A. and Taskar, B. (Eds), Predicting Structured Data, MIT Press, 2006

– M. Ranzato Ph.D. Thesis “Unsupervised Learning of Feature Hierarchies” NYU 2009

Practical guide

– Y. LeCun et al. Efficient BackProp, Neural Networks: Tricks of the Trade, 1998

– L. Bottou, Stochastic gradient descent tricks, Neural Networks, Tricks of the Trade Reloaded, LNCS 2012.

– Y. Bengio, Practical recommendations for gradient-based training of deep architectures, ArXiv 2012