



NEW YORK UNIVERSITY

# Energy-Based Models (part 3)

Yann LeCun

NYU - Courant Institute & Center for Data Science

Facebook AI Research

<http://yann.lecun.com>

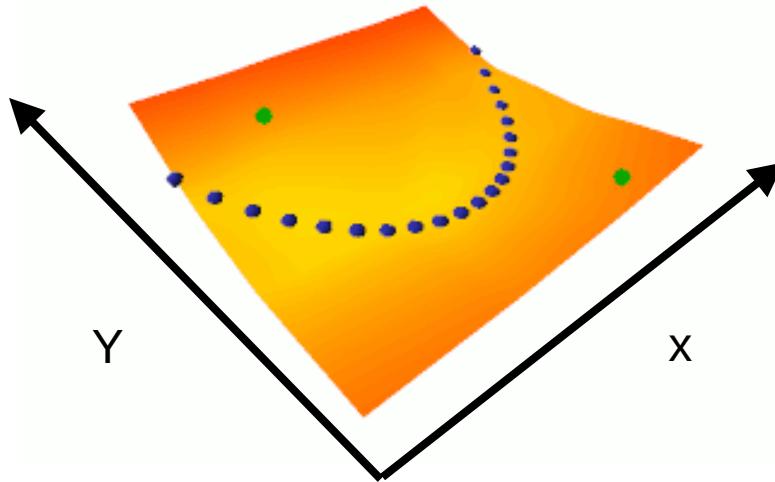
Deep Learning, NYU, Fall 2021

# Contrastive energy-based learning

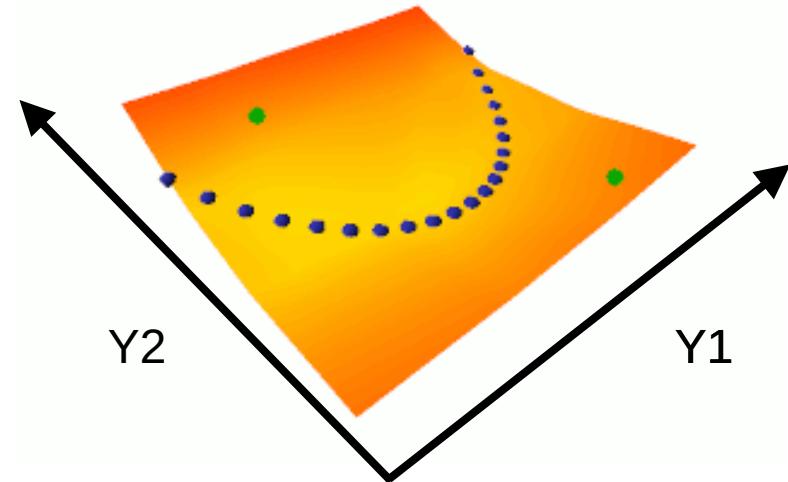
- ▶ Push down on the energy of training samples
- ▶ Pull up on the energy of other “well-chosen” points

$$\mathcal{L}(x_1 \dots x_{p^+}, y_1 \dots y_{p^+}, \hat{y}_1 \dots \hat{y}_{p^-}, w) = H \left( E(x_1, y_1), \dots E(x_{p^+}, y_{p^+}), E(x_1, \hat{y}_1), \dots E(x_{p^+}, \hat{y}_{p^+}), M(Y_{1 \dots p^+}, \hat{Y}_{1 \dots p^-}) \right)$$

conditional



unconditional



# Contrastive Methods vs Regularized/Architectural Methods

- ▶ **Contrastive:** [they all are different ways to pick which points to push up]
  - ▶ C1: push down of the energy of data points, push up everywhere else: Max likelihood (needs tractable partition function or variational approximation)
  - ▶ C2: push down of the energy of data points, push up on chosen locations: max likelihood with MC/MMC/HMC, Contrastive divergence, Metric learning/Siamese nets, Ratio Matching, Noise Contrastive Estimation, Min Probability Flow, adversarial generator/GANs
  - ▶ C3: train a function that maps points off the data manifold to points on the data manifold: denoising auto-encoder, masked auto-encoder (e.g. BERT)
- ▶ **Regularized/Architectural:** [Different ways to limit the information capacity of the latent representation]
  - ▶ A1: build the machine so that the volume of low energy space is bounded: PCA, K-means, Gaussian Mixture Model, Square ICA, normalizing flows...
  - ▶ A2: use a regularization term that measures the volume of space that has low energy: Sparse coding, sparse auto-encoder, LISTA, Variational Auto-Encoders, discretization/VQ/VQVAE.
  - ▶ A3:  $F(x,y) = C(y, G(x,y))$ , make  $G(x,y)$  as "constant" as possible with respect to  $y$ : Contracting auto-encoder, saturating auto-encoder
  - ▶ A4: minimize the gradient and maximize the curvature around data points: score matching

# Architectural and Regularized Methods

- ▶ **Regularized/Architectural:** [Different ways to limit the information capacity of the latent representation]
- ▶ A1: build the machine so that the volume of low energy space is bounded: PCA, K-means, Gaussian Mixture Model, Square ICA, normalizing flows...
- ▶ A2: use a regularization term that measures the volume of space that has low energy: Sparse coding, sparse auto-encoder, LISTA, Variational Auto-Encoders, discretization/VQ/VQVAE.
- ▶ A3:  $F(x,y) = C(y, G(x,y))$ , make  $G(x,y)$  as "constant" as possible with respect to  $y$ : Contracting auto-encoder, saturating auto-encoder
- ▶ A4: minimize the gradient and maximize the curvature around data points: score matching

# Architectural EBM

Construct the machine so that the volume of low-energy space is fixed or limited.

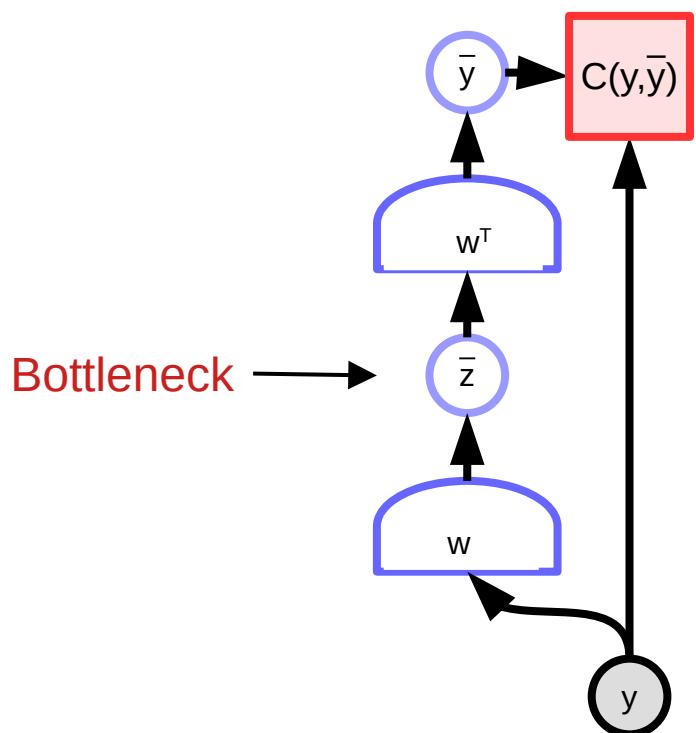
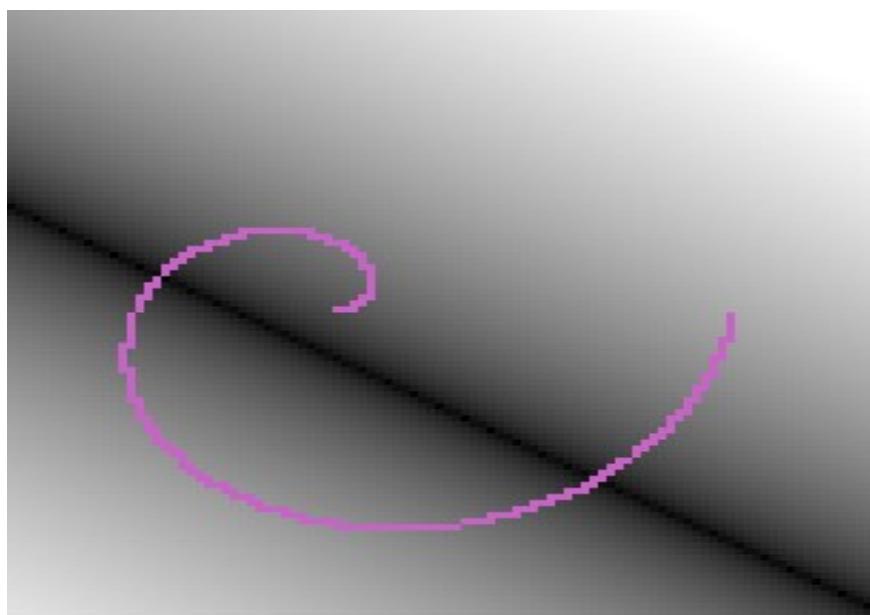


# Architectural Methods

- ▶ Architectural: [Different ways to limit the information capacity of the latent representation]
  - ▶ A1: build the machine so that the volume of low energy space is bounded:
  - ▶ PCA: volume is dimension of principal subspace
  - ▶ K-means: volume limited by number of prototypes
  - ▶ Analytically normalized probabilistic models: volume is fixed Gaussian, and Gaussian Mixture Model (model is normalized)
  - ▶ Square Independent Component Analysis
  - ▶ Latent variable models with fixed latent distribution: volume is less than the “volume” (entropy) of the latent variable prior distribution.
  - ▶ Normalizing flows

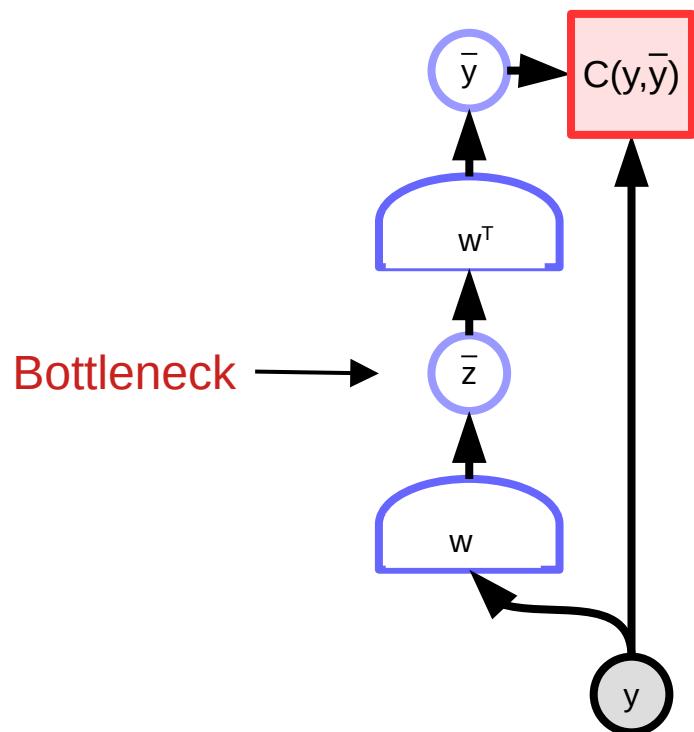
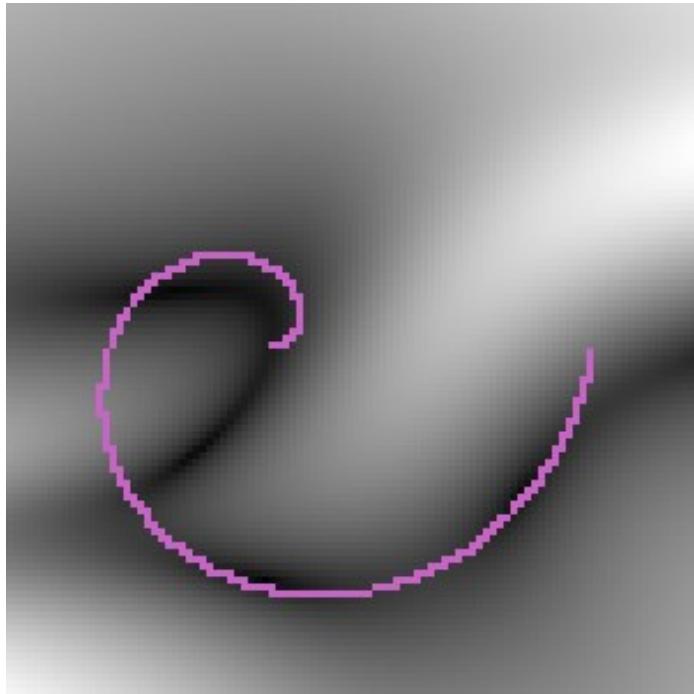
# Principal Component Analysis

- ▶ PCA is a 2-layer linear auto-encoder with a bottleneck.
- ▶ Energy:  $F_w(y) = \|y - Dec(Enc(y))\|^2 = \|y - w^T w\|^2$
- ▶ Loss  $L(y, w) = F_w(y)$



# Auto-Encoder with Bottleneck

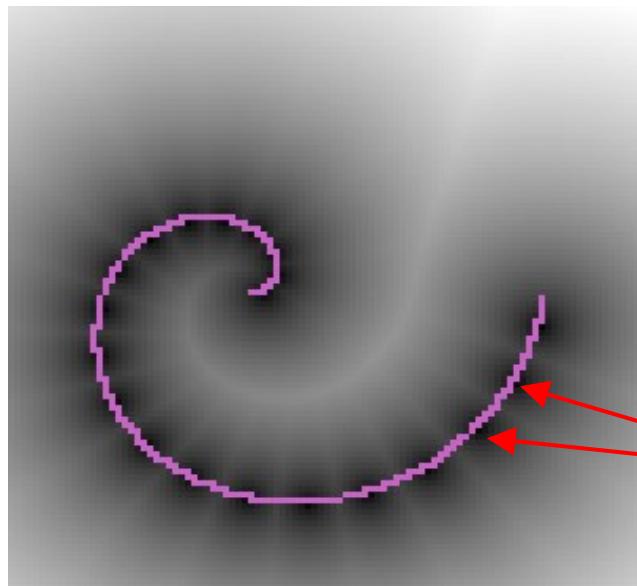
- ▶ non-linear auto-encoder with a bottleneck.
- ▶ Energy:  $F_w(y) = \|y - Dec(Enc(y))\|^2$
- ▶ Loss:  $L(y, w) = F_w(y)$



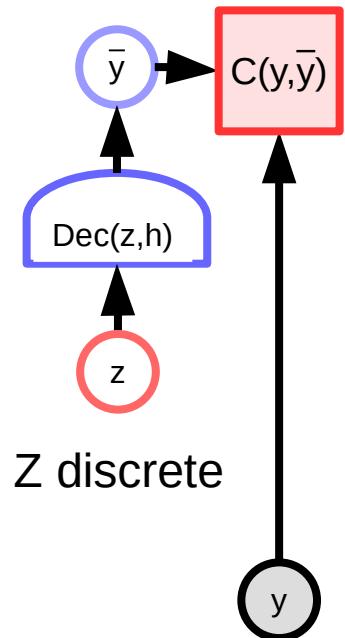
# K-Means

## ► Discrete latent-variable model with linear decoder

- Energy:  $E(y, z) = \|y - Dec(z)\|^2 = \|y - wz\|^2$
- Free Energy  $F(y) = \min_{z \in \mathcal{Z}} E(y, z)$
- Loss:  $L(y, w) = F_w(y)$



- Latent vector  $z$  is constrained to be a 1-hot vector:  $[0,0,\dots,01,0,\dots,0]$
- 1 component selects a column of  $w$
- $F(y)=0$  iff  $y$  is equal to a column of  $w$ .



# Gaussian Mixture Model

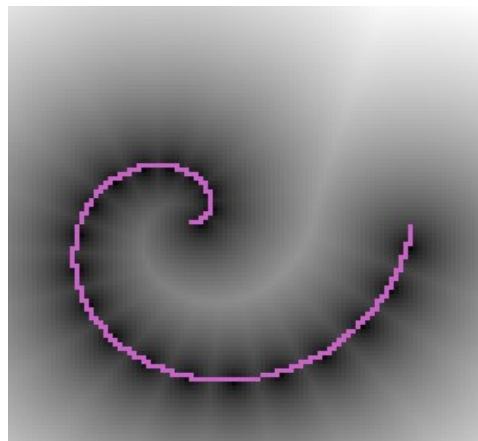
- ▶ Similar to K-means with soft marginalization over latent.

- ▶ Energy:  $E(y, z) = (y - wz)^T(Mz)(y - wz)$

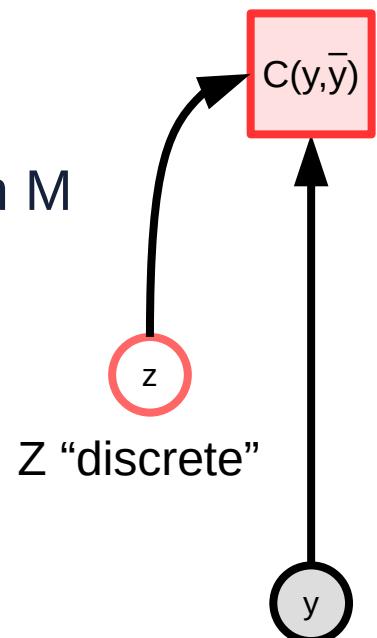
$$(Mz)_{ij} = \sum_k M_{ijk} z_k$$

$$\text{Free Energy } F(y) = -\frac{1}{\beta} \log \sum_{z \in \mathcal{Z}} e^{\beta E(y, z)}$$

- ▶ Loss:  $L(y, w) = F_w(y)$  with normalization constraint on M



- ▶ Latent vector  $z$  is constrained to be a 1-hot vector:  
 $[0,0,\dots,01,0,\dots,0]$
- ▶ But marginalization makes it “soft”



# Regularized EBM

Push down on the energy of data points  
& Penalize/minimize the volume  
of low-energy space

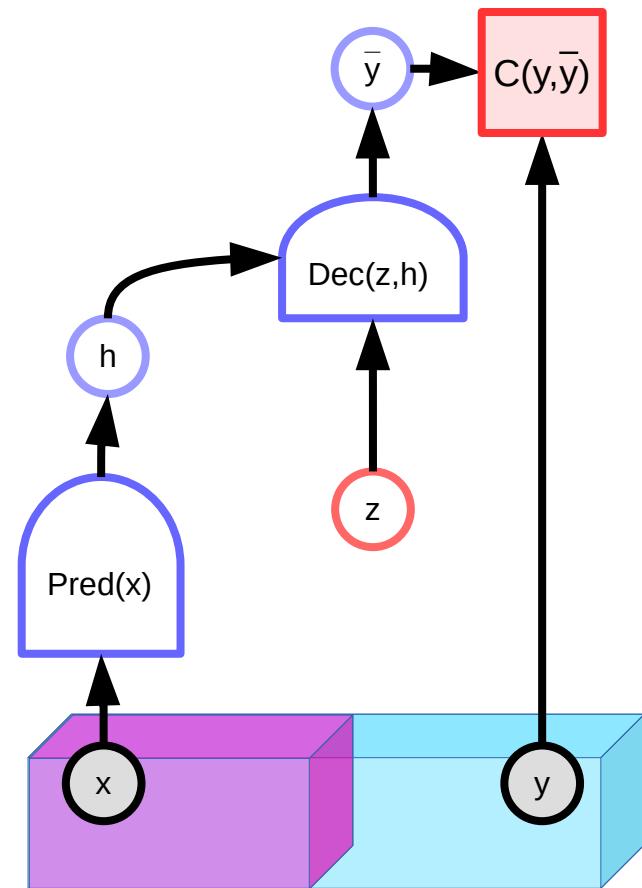


# Prediction with Latent Variables

- ▶ If the Latent has too much capacity...
- ▶ e.g. if it has the same dimension as  $y$
- ▶ ... then the entire  $y$  space could be perfectly reconstructed

$$E(x, y, z) = C(y, \text{Dec}(\text{Pred}(x), z))$$

- ▶ For every  $y$ , there is always a  $z$  that will reconstruct it perfectly
- ▶ The energy function would be zero everywhere
- ▶ This is no a good model....
- ▶ **Solution: limiting the information capacity of the latent variable  $z$ .**

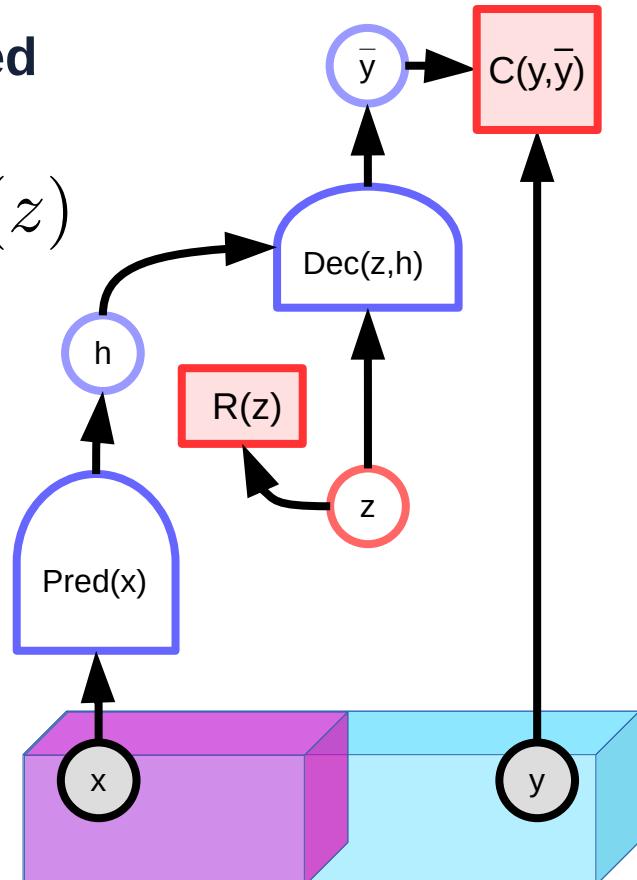


# Regularized Latent Variable EBM

- ▶ Regularizer  $R(z)$  limits the information capacity of  $z$
- ▶ Without regularization, every  $y$  may be reconstructed exactly (flat energy surface)

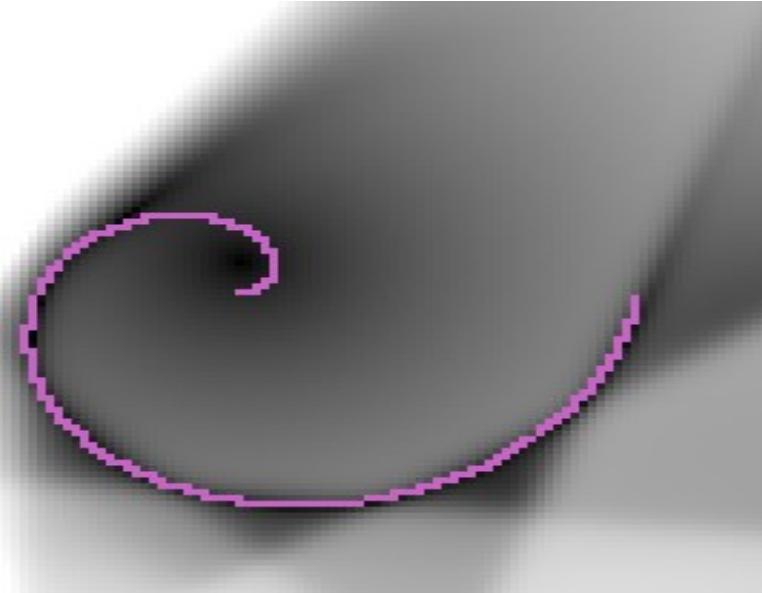
$$E(x, y, z) = C(y, \text{Dec}(\text{Pred}(x), z)) + \lambda R(z)$$

- ▶ Examples of  $R(z)$ :
- ▶ Effective dimension
- ▶ Quantization / discretization
- ▶ L0 norm (# of non-0 components)
- ▶ L1 norm with decoder normalization
- ▶ Maximize lateral inhibition / competition
- ▶ Add noise to  $z$  while limiting its L2 norm (VAE)
- ▶ <your\_information\_throttling\_method\_goes\_here>

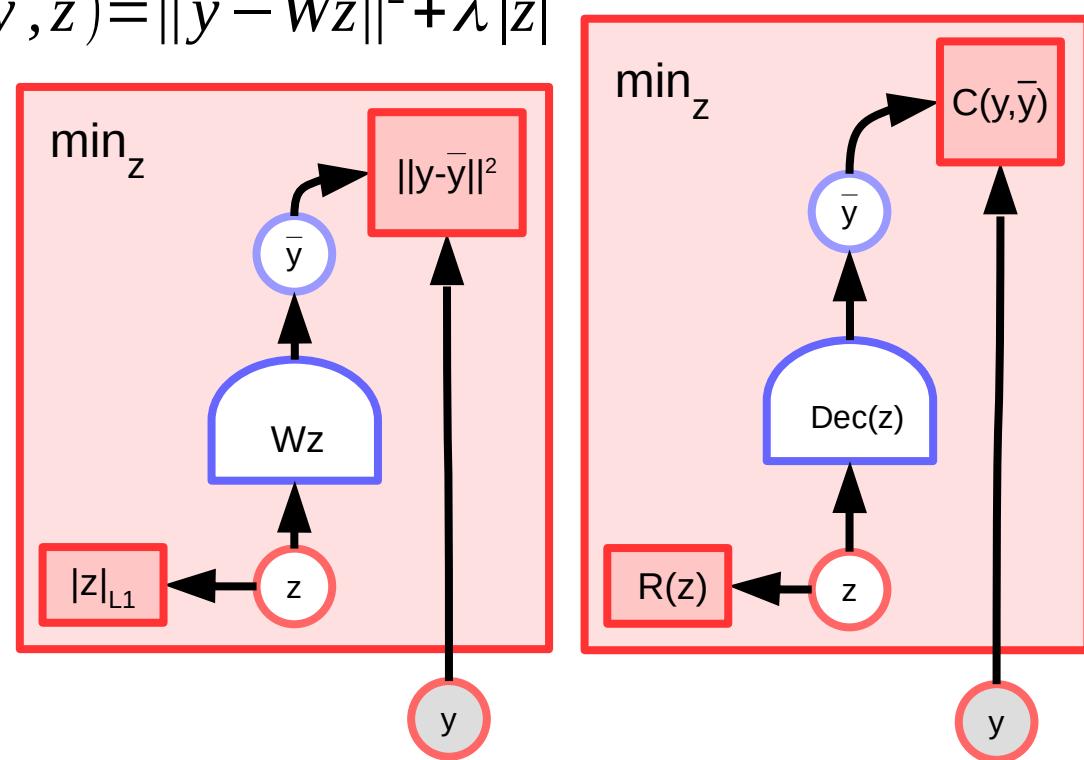


# Unconditional Regularized Latent Variable EBM

- ▶ Unconditional form. Reconstruction (no  $x$ , no predictor).
- ▶ Example: sparse coding / sparse modeling
- ▶ Linear decoder
- ▶ L1 regularizer on  $Z$

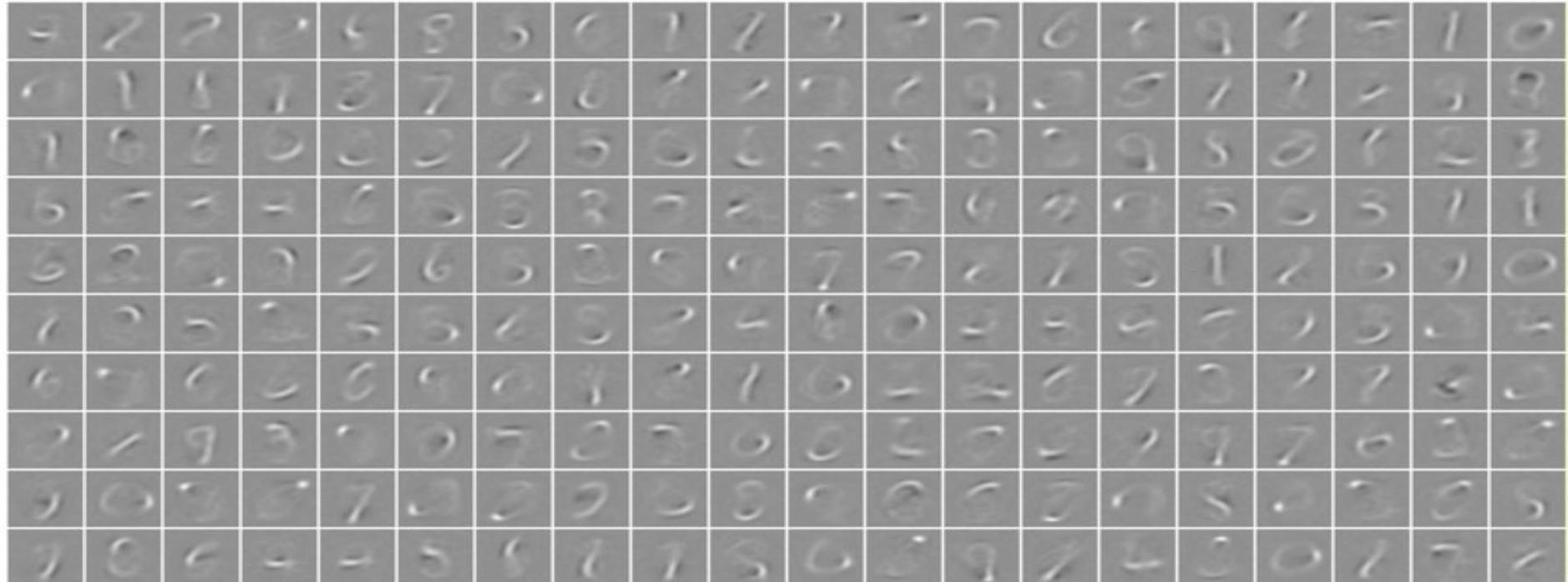


$$E(y, z) = \|y - Wz\|^2 + \lambda |z|_{L1}$$



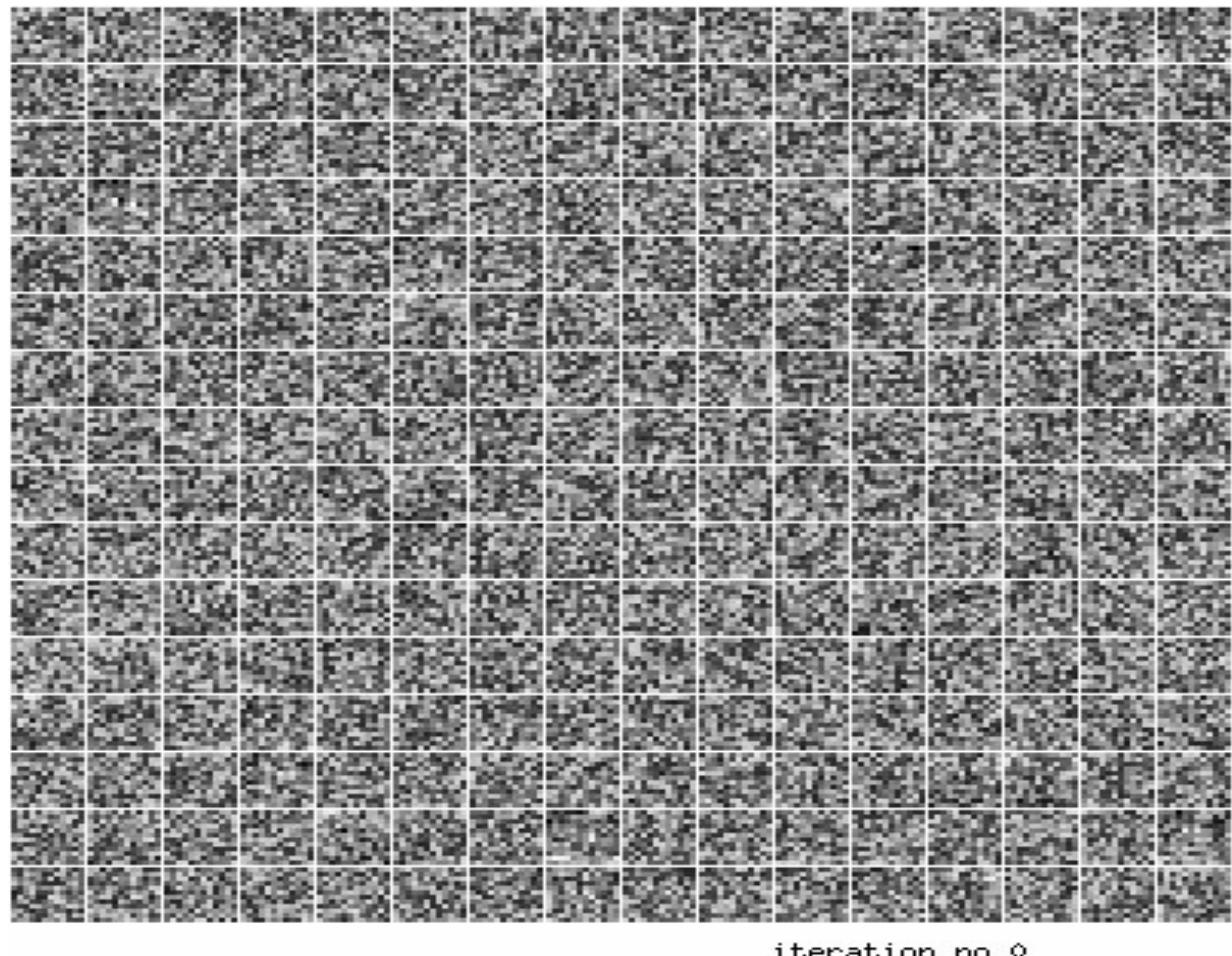
# Sparse Modeling on handwritten digits (MNIST)

- Basis functions (columns of decoder matrix) are digit parts
- All digits are a linear combination of a small number of these

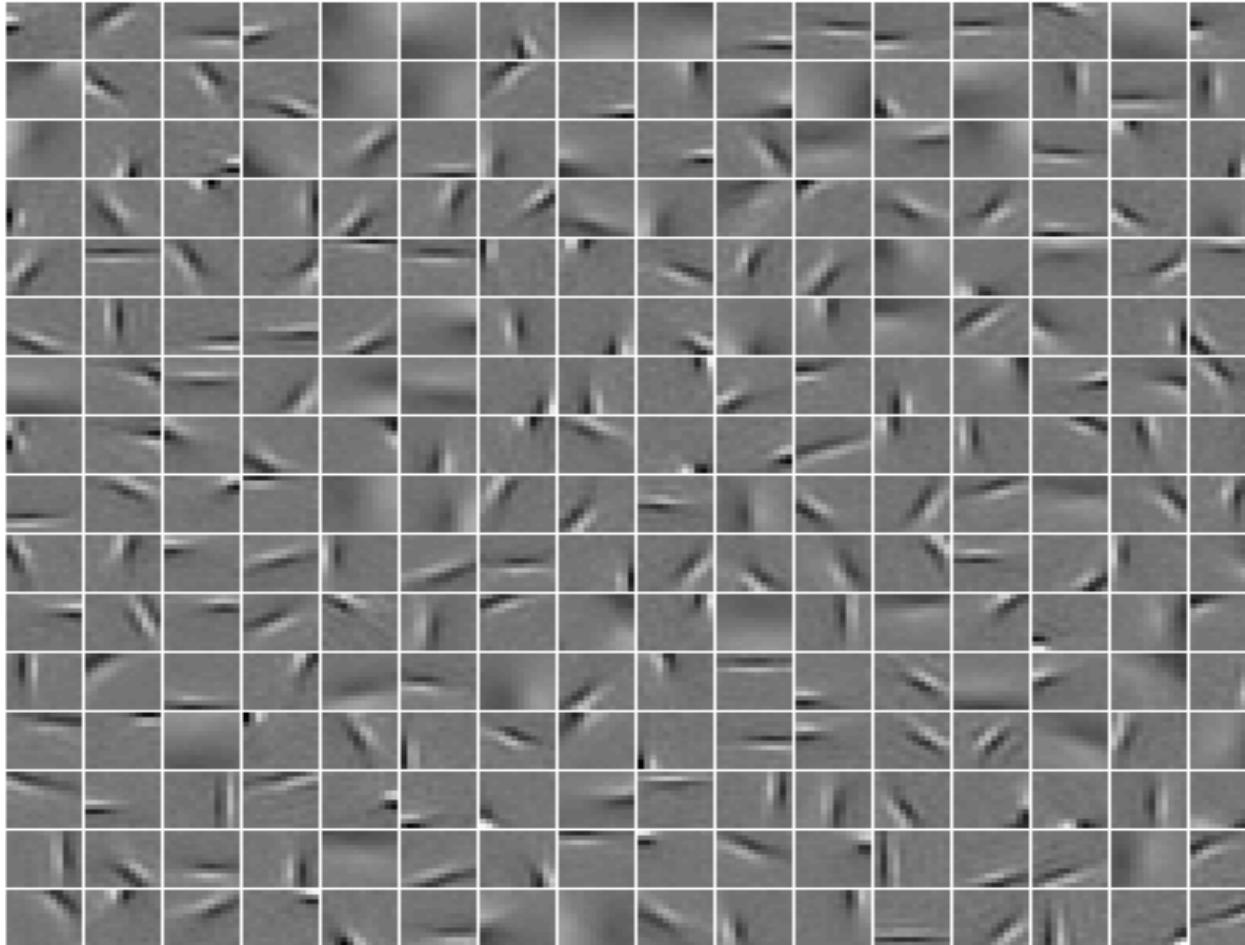


# Predictive Sparse Decomposition (PSD): Training

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



# Learned Features on natural patches: V1-like receptive fields



# Amortized Inference

- ▶ Training an encoder to give an approximate solution to the inference optimization problem

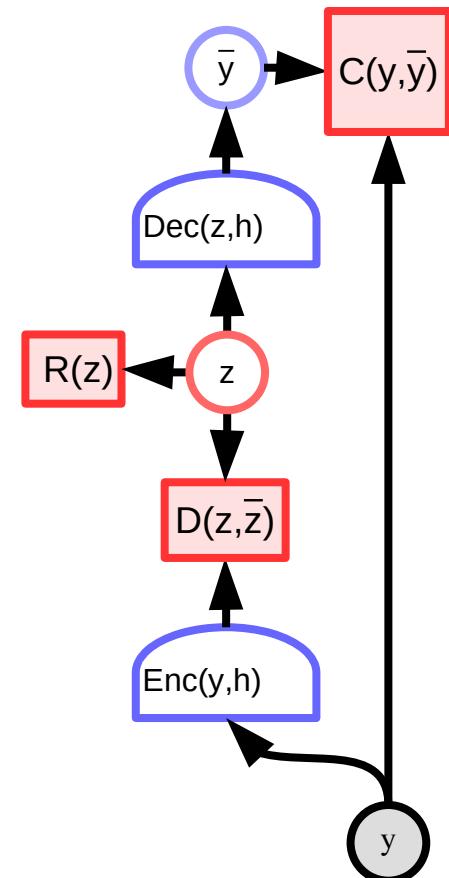
- ▶ Regularized Auto-Encoder, Sparse AE, LISTA

$$E(y, z) = C(y, \text{Dec}((z))) + D(z, \text{Enc}(y)) + \lambda R(z)$$

$$F(y) = \min_z E(y, z)$$

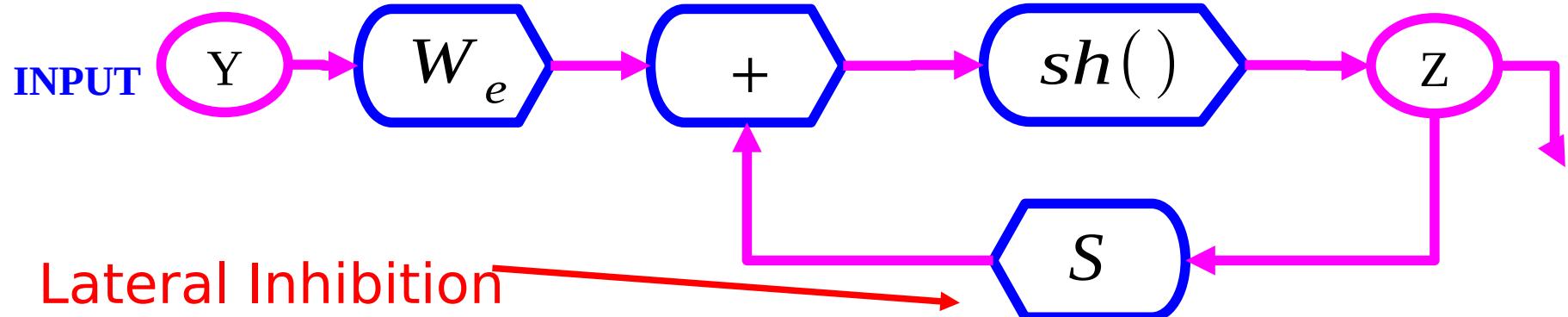
- ▶ Variational AE
- ▶ Approximated by sampling and variational approximation

$$F(y) = -\log \int_z e^{-E(y, z)}$$



# Giving the “right” structure to the encoder

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



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

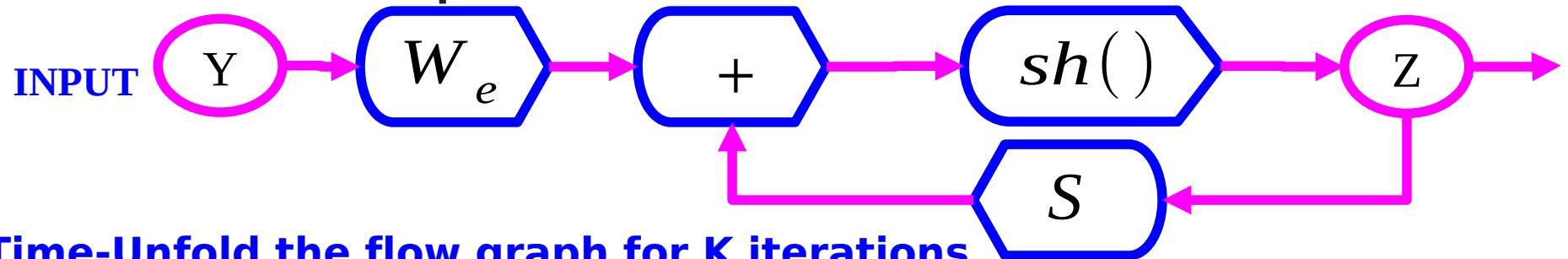
- ISTA/FastISTA reparameterized:

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

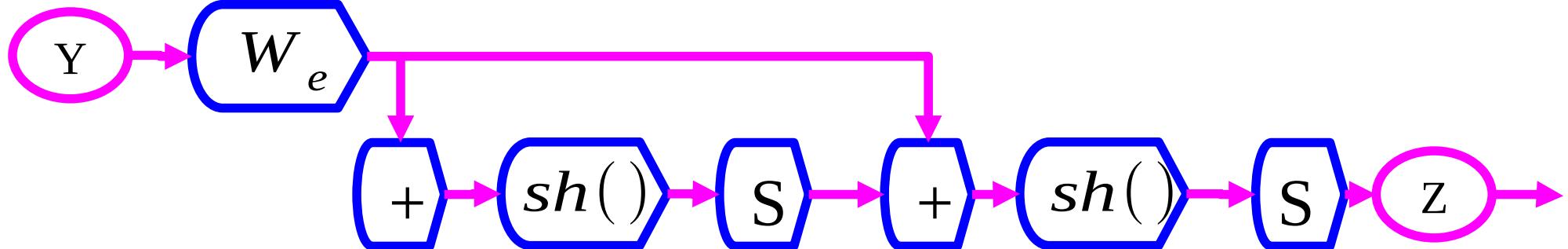
- LISTA (Learned ISTA): learn the  $W_e$  and  $S$  matrices to get fast solutions

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

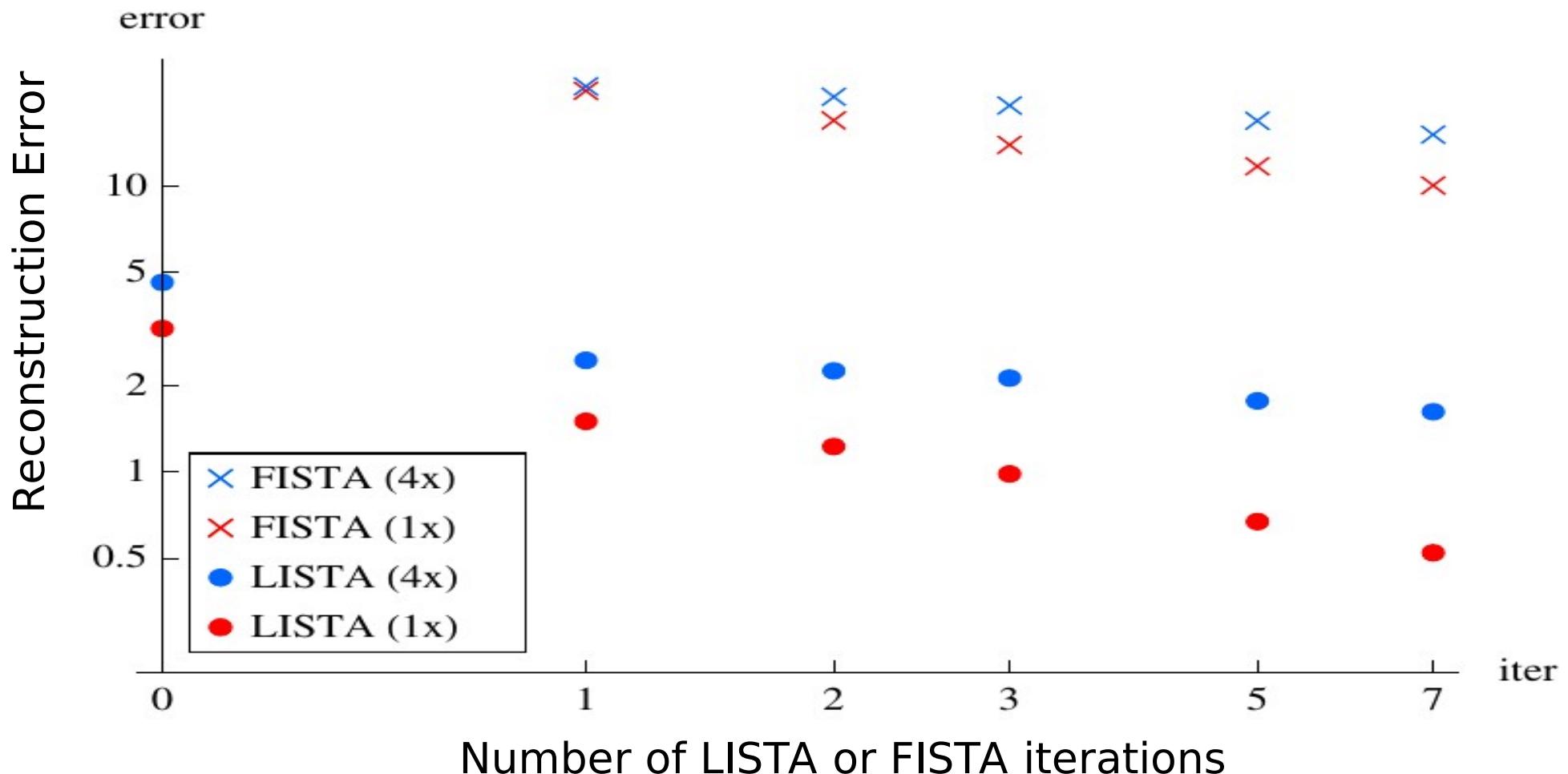
- Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters



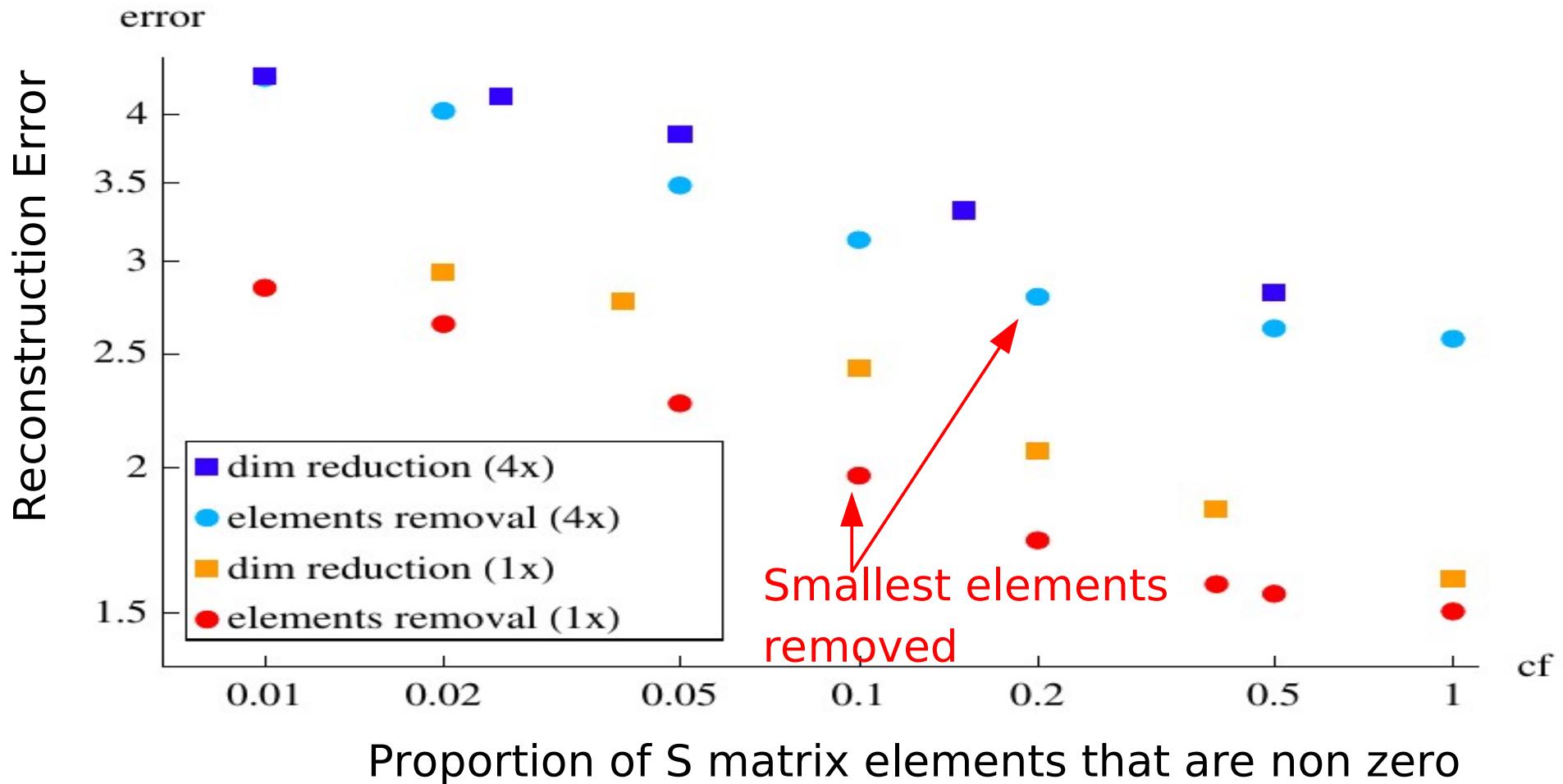
- Time-Unfold the flow graph for K iterations
- Learn the We and S matrices with “backprop-through-time”
- Get the best approximate solution within K iterations



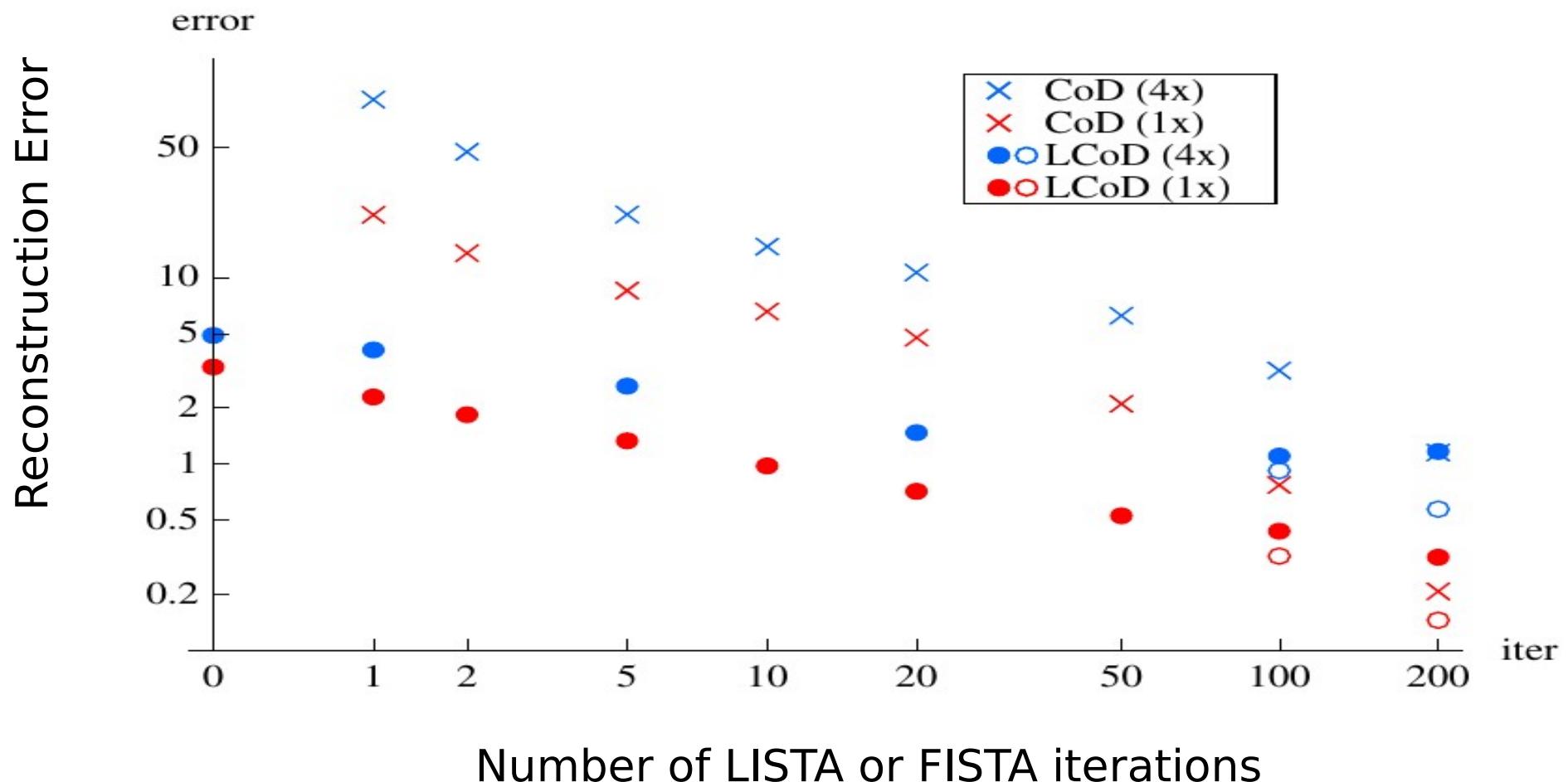
# Learning ISTA (LISTA) vs ISTA/FISTA



# LISTA with partial mutual inhibition matrix



# Learning Coordinate Descent (LcoD): faster than LISTA



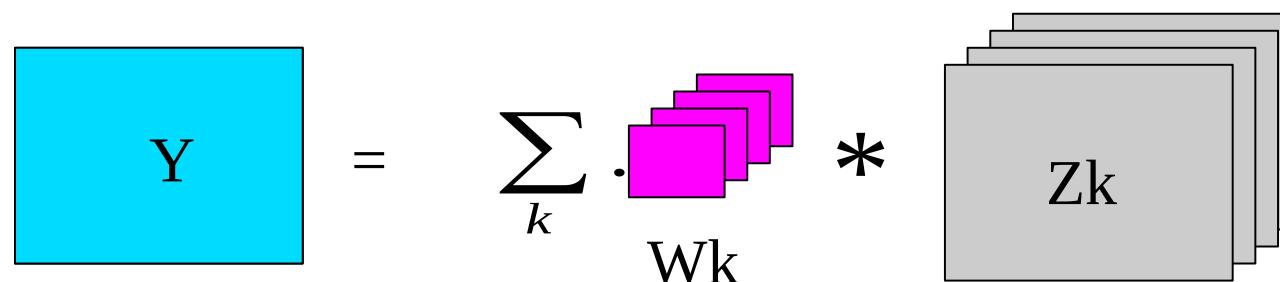
# Convolutional Sparse Coding

Replace the dot products with dictionary element by convolutions.

- ▶ Input  $Y$  is a full image
- ▶ Each code component  $Z_k$  is a feature map (an image)
- ▶ Each dictionary element is a convolution kernel

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

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

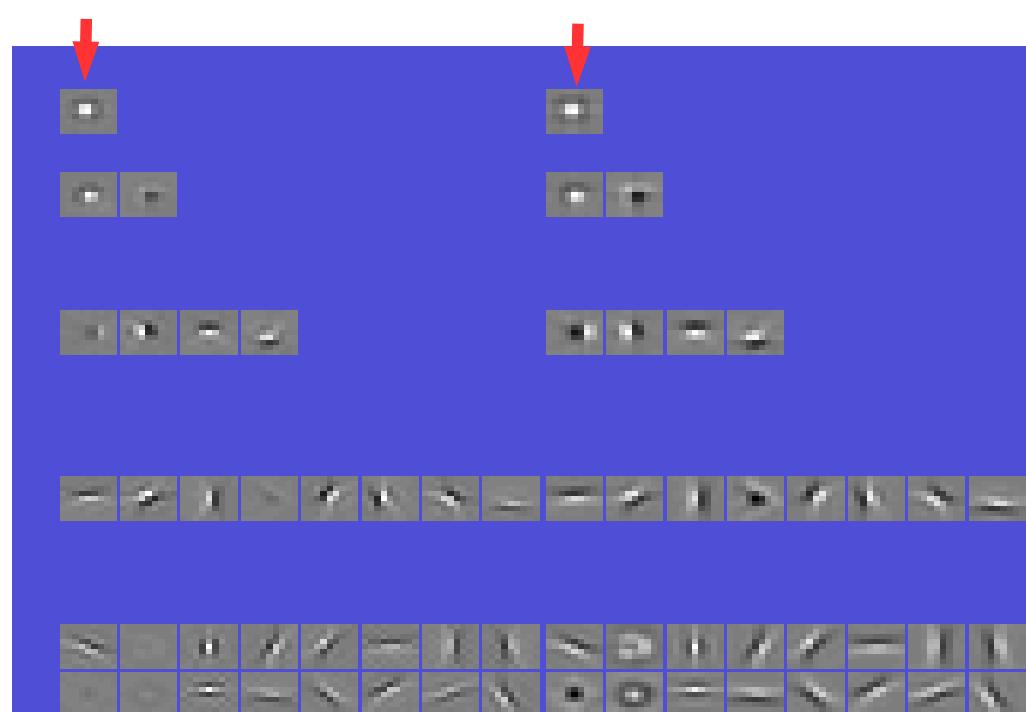


Also used in “deconvolutional networks” [Zeiler, Taylor, Fergus CVPR 2010]

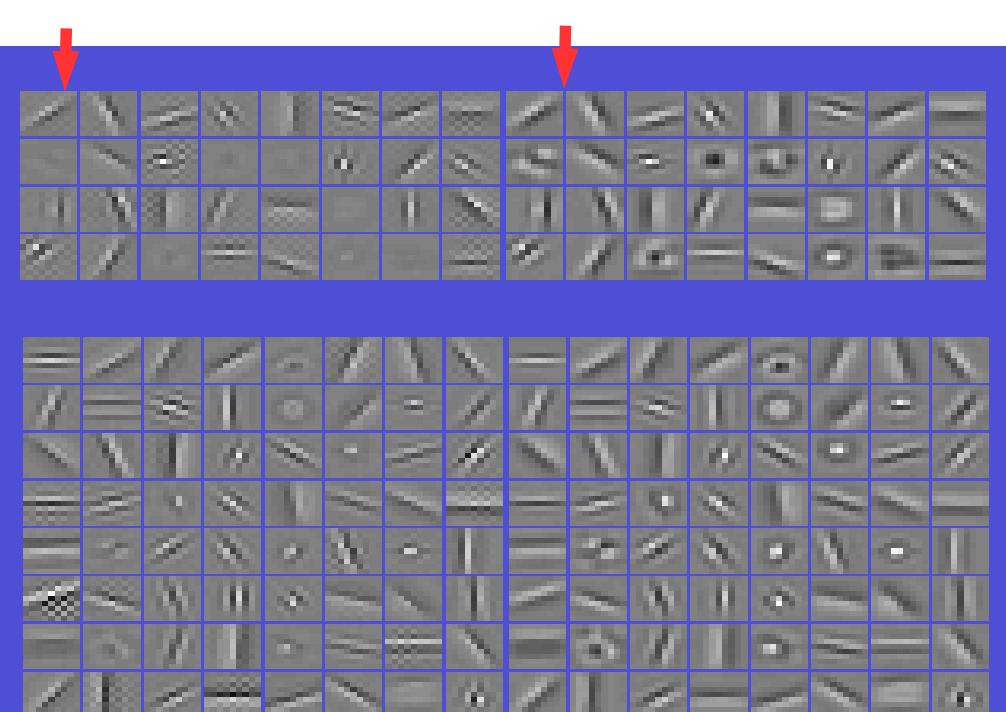
# Convolutional Sparse Auto-Encoder on Natural Images

- ▶ Encoder filters and decoder filters. Decoder is linear (convolutional)
- ▶ with 1, 2, 4, 8, 16, 32, and 64 filters [Kavukcuoglu NIPS 2010]

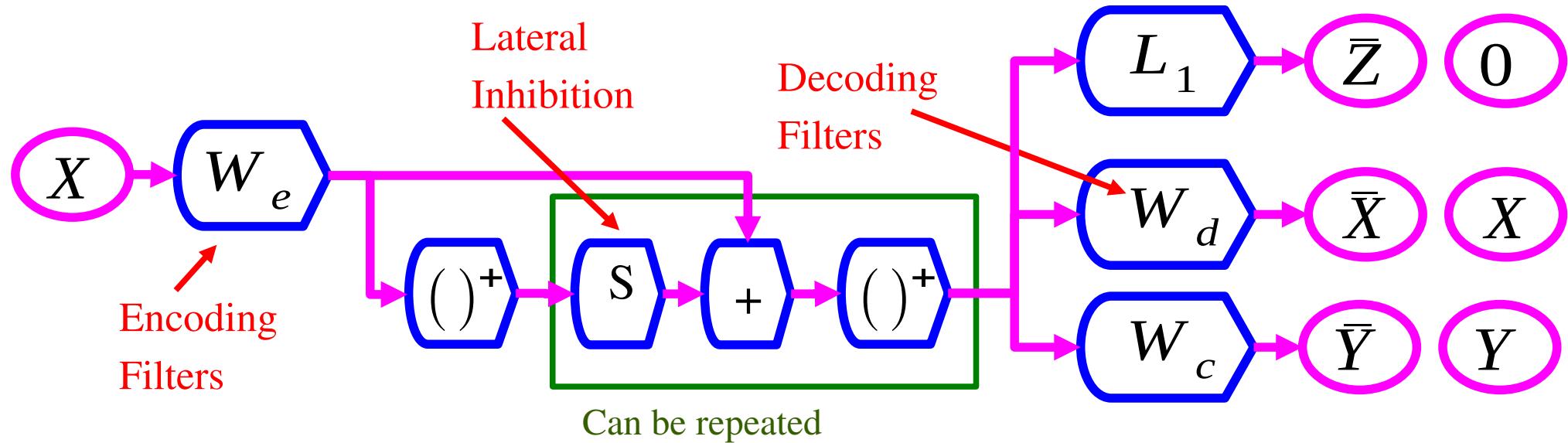
Encoder Filters      Decoder Filters      Encoder Filters      Decoder Filters



Encoder Filters      Decoder Filters



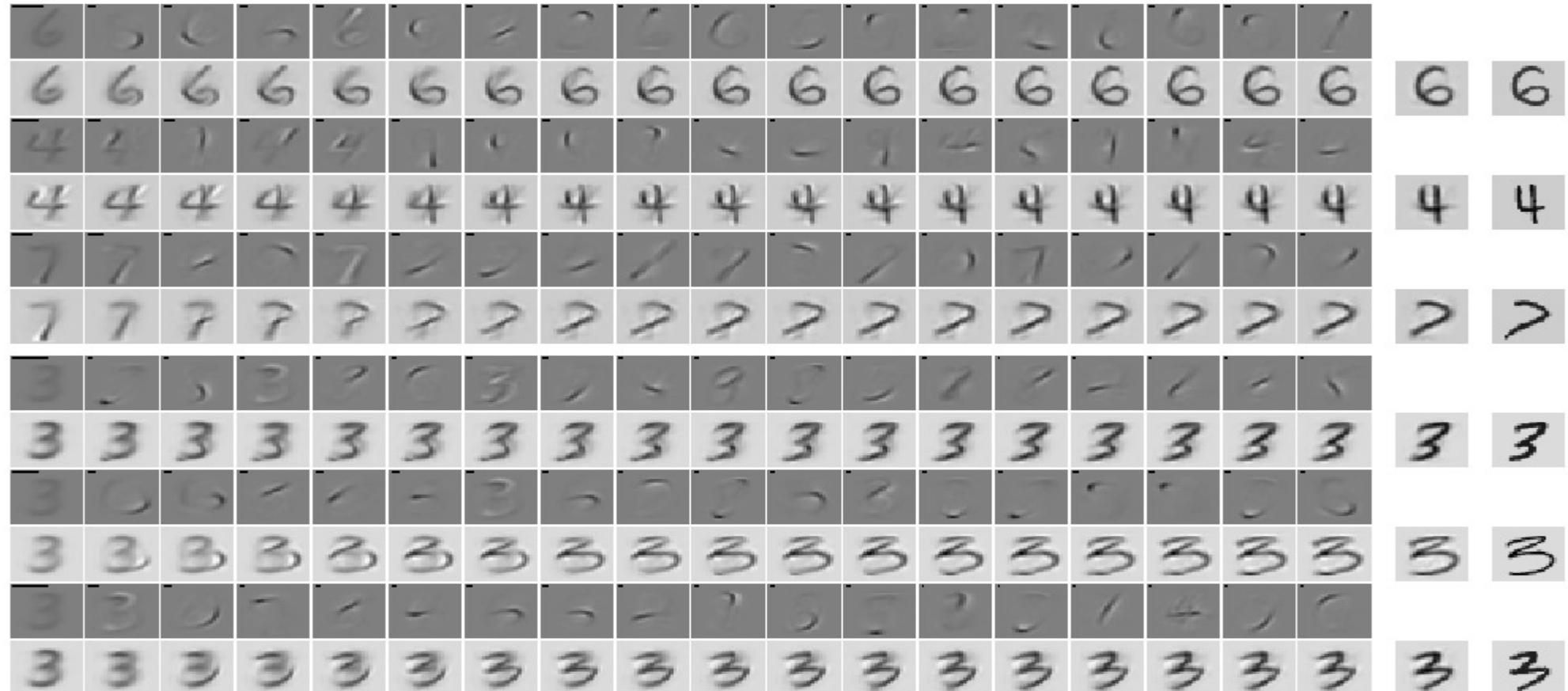
# Discriminative Recurrent Sparse Auto-Encoder (DrSAE)



[Rolle & LeCun ICLR 2013]

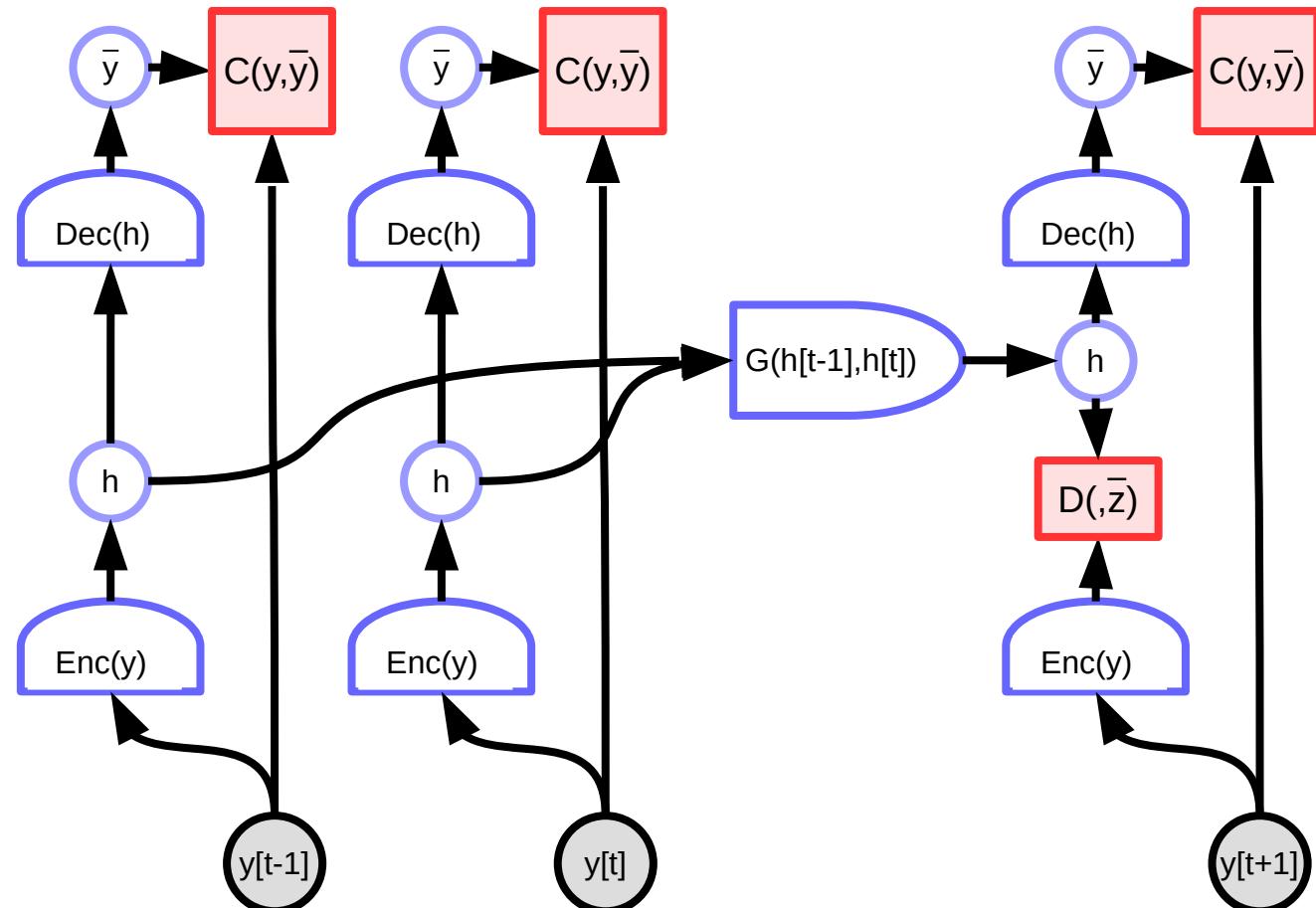
# DrSAE Discovers manifold structure of handwritten digits

Image = prototype + sparse sum of “parts” (to move around the

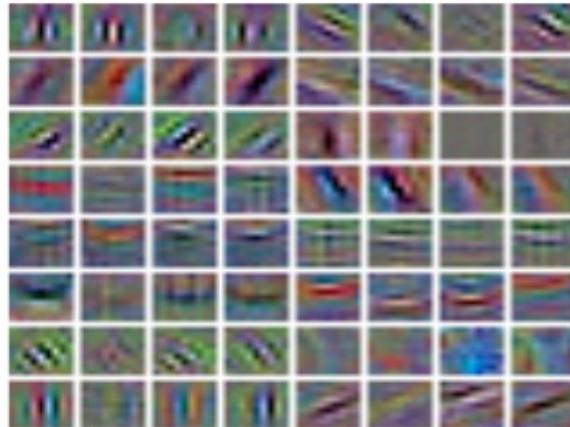


# Temporal Regularization Methods

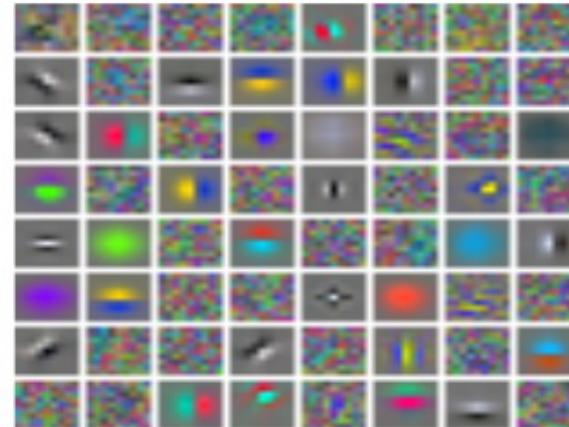
- ▶ Favors “flatness” and predictability of the representation.
  - ▶ Temporal invariance [Goroshin ICCV’15]
  - ▶ Linear predictability [Goroshin NIPS’16]
  - ▶ Minimal curvature [O. Hénaff 2019]
- ▶ Temporal proximity is an instance of similarity graph.
- ▶ Decoder alleviates need for contrastive samples



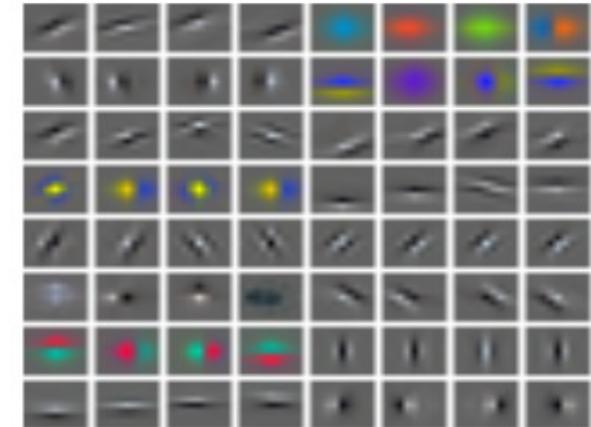
# Sparse Auto-Encoder with “Slow Feature” Penalty



▶ Supervised filters CIFAR10

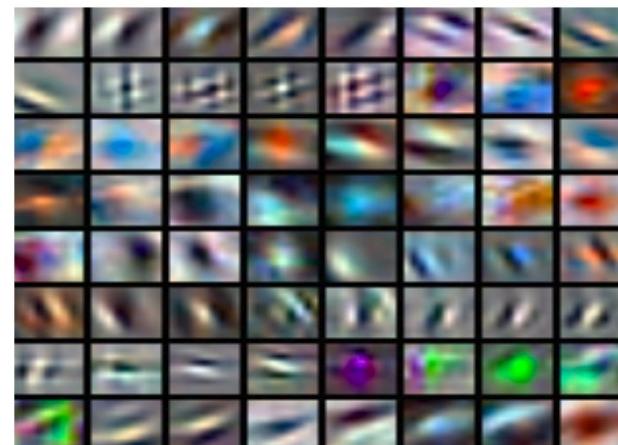
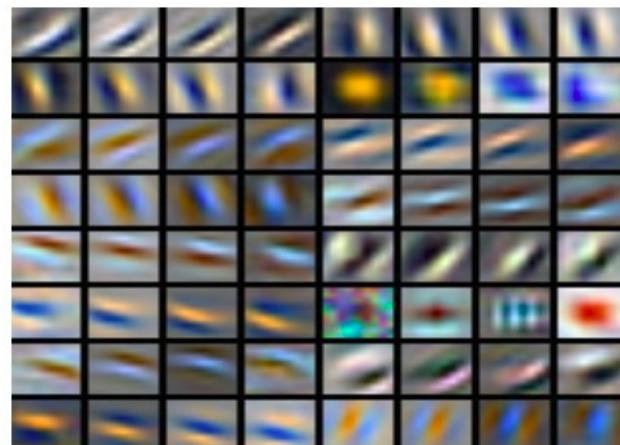


sparse conv. auto-encoder



slow & sparse convolutional AE  
trained on YouTube videos

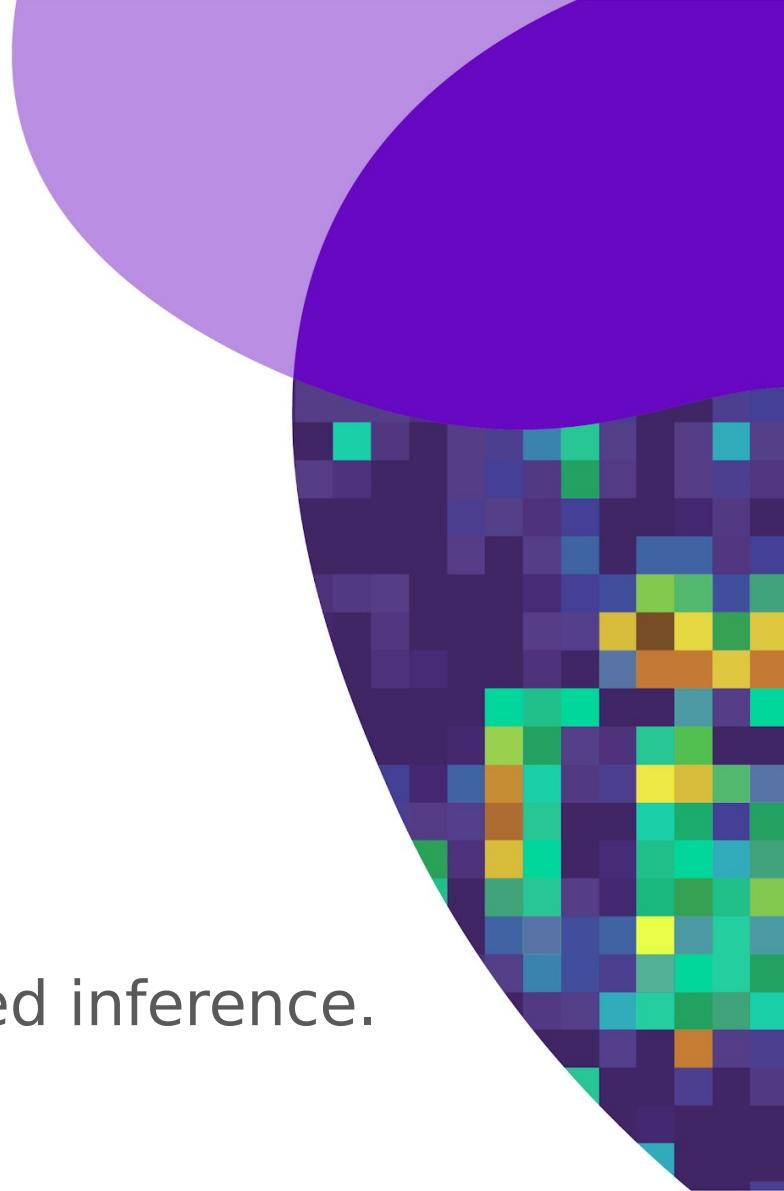
- ▶ Representation is pooled over non-overlapping groups of 4 features



- ▶ Representation is pooled over overlapping groups of 4 features

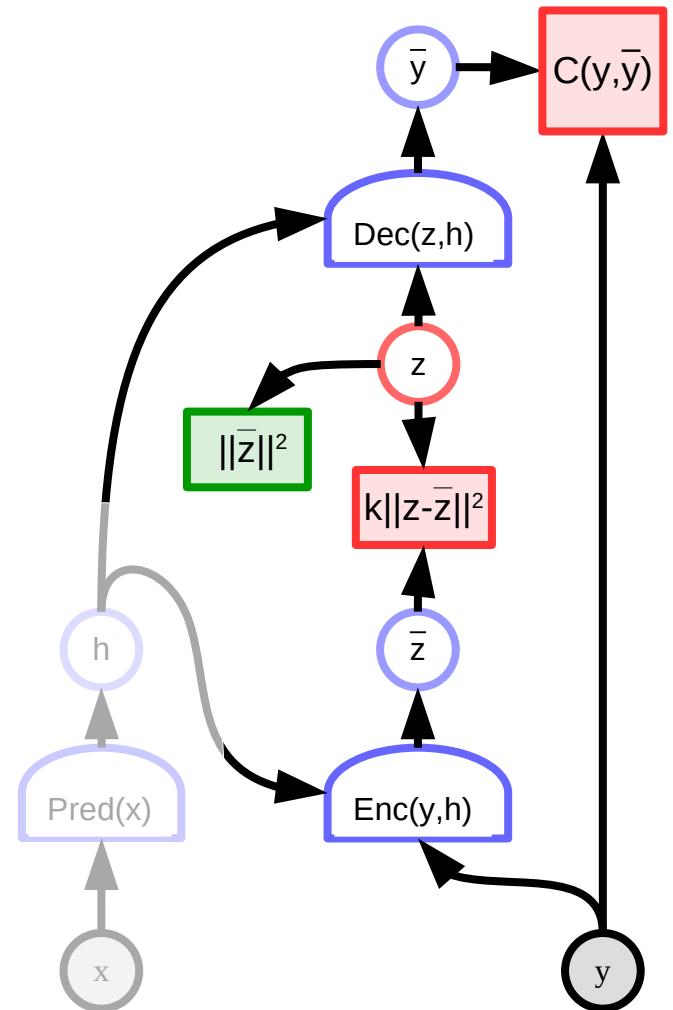
# Variational AE

The energy-based approach.  
It's a kind of noise-regularized  
latent var model with amortized inference.



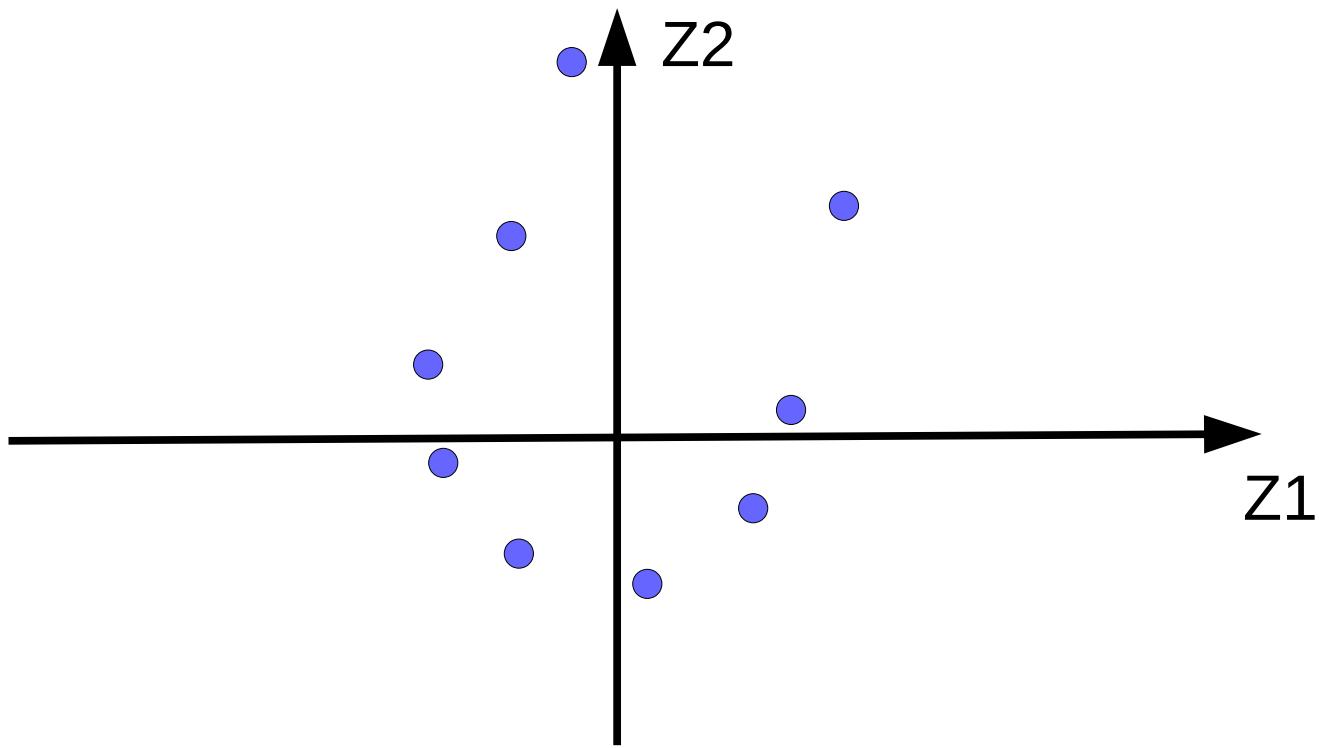
# Variational Auto-Encoder

- ▶ Limiting the information capacity of the code by adding Gaussian noise
- ▶ The energy term  $k\|z-\bar{z}\|^2$  is seen as the log of a prior from which to sample  $z$
- ▶ The encoder output is regularized to have a mean and a variance close to zero.



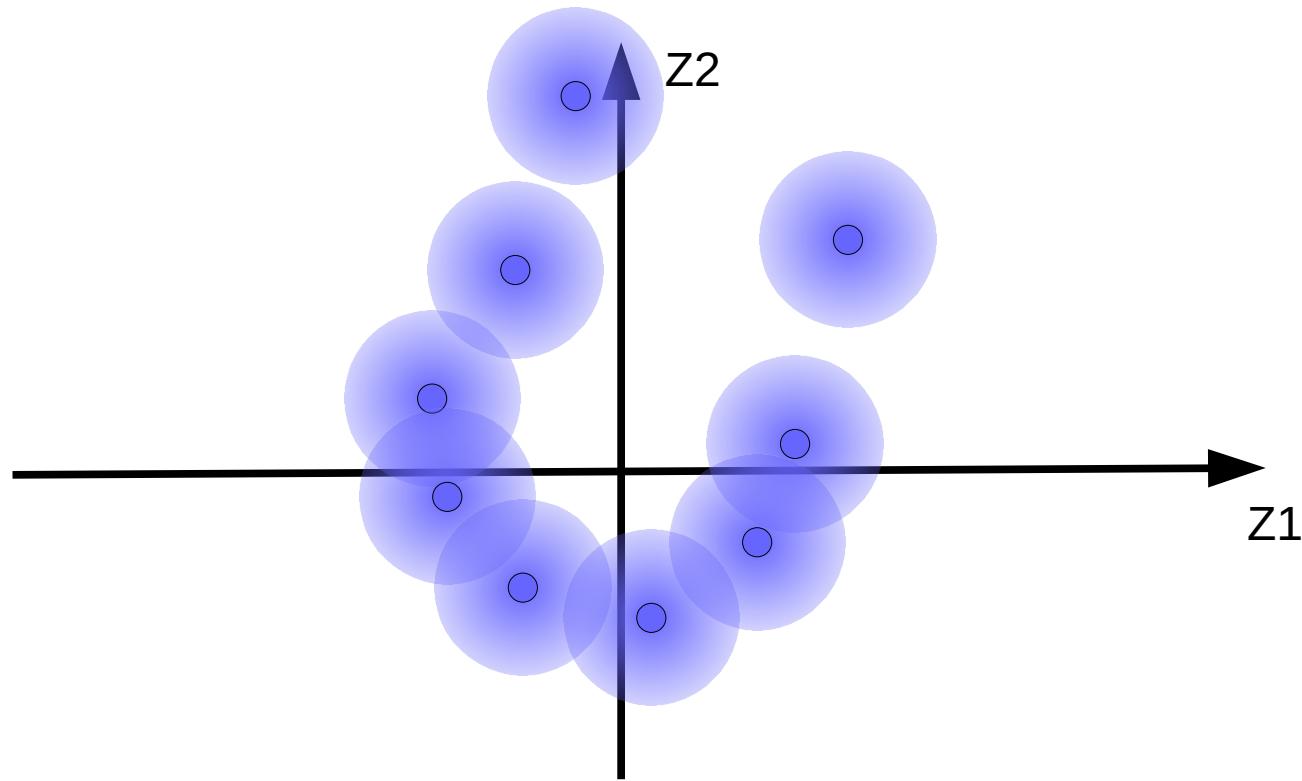
# Variational Auto-Encoder

- ▶ Code vectors for training samples



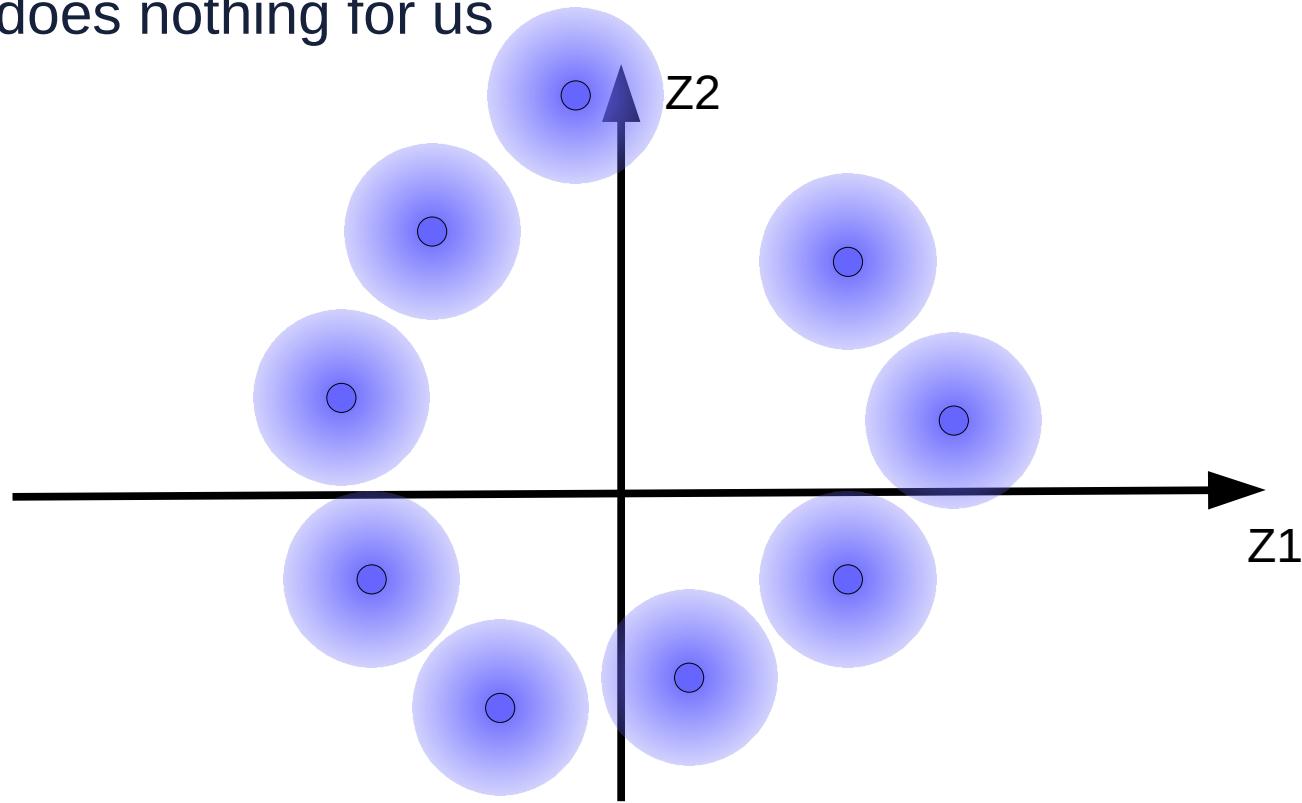
# Variational Auto-Encoder

- ▶ **Code vectors for training sample with Gaussian noise**
- ▶ Some fuzzy balls overlap, causing bad reconstructions



# Variational Auto-Encoder

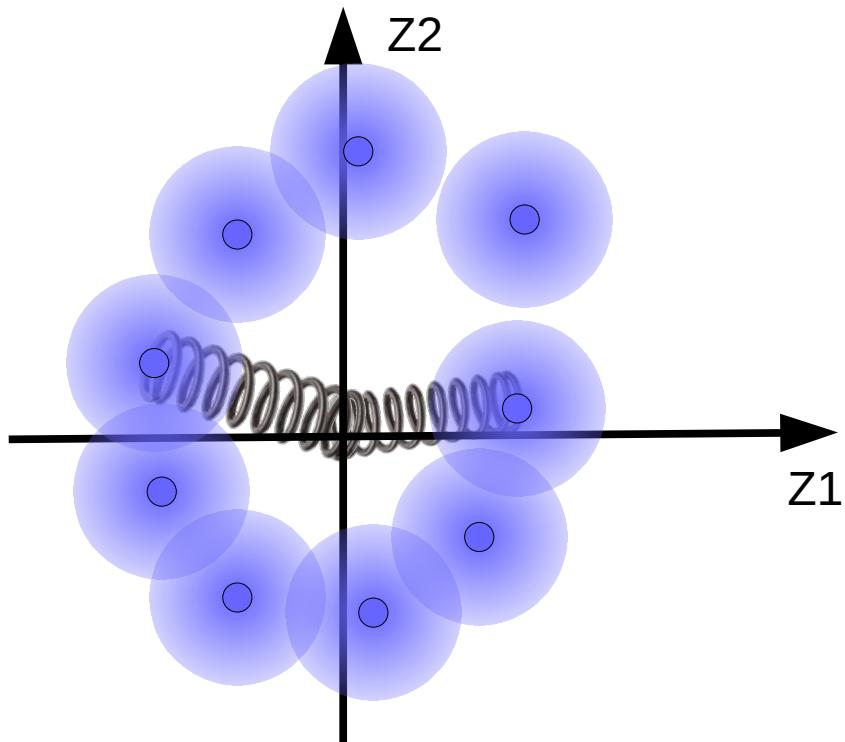
- ▶ The code vectors want to move away from each other to minimize reconstruction error
- ▶ But that does nothing for us



# Variational Auto-Encoder

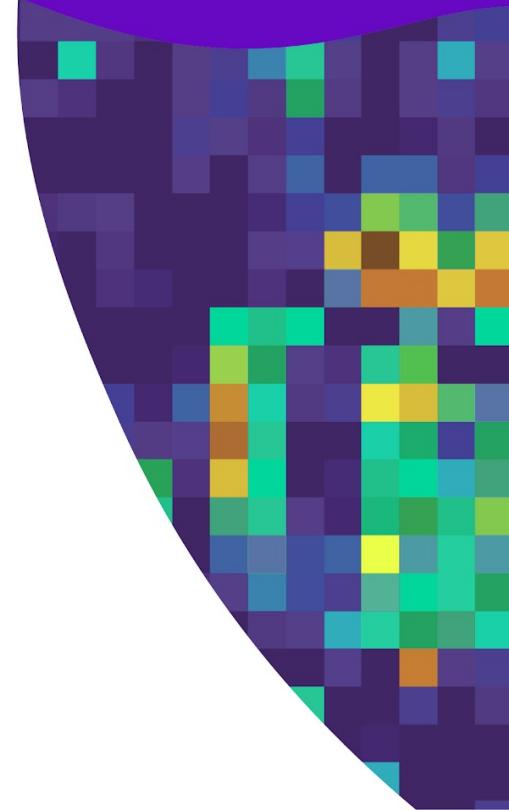
- ▶ Attach the balls to the center with a spring, so they don't fly away
- ▶ Minimize the square distances of the balls to the origin
- ▶ Center the balls around the origin
  - ▶ Make the center of mass zero
- ▶ Make the sizes of the balls close to 1 in each dimension
  - ▶ Through a so-called KL term

Talia Konkle

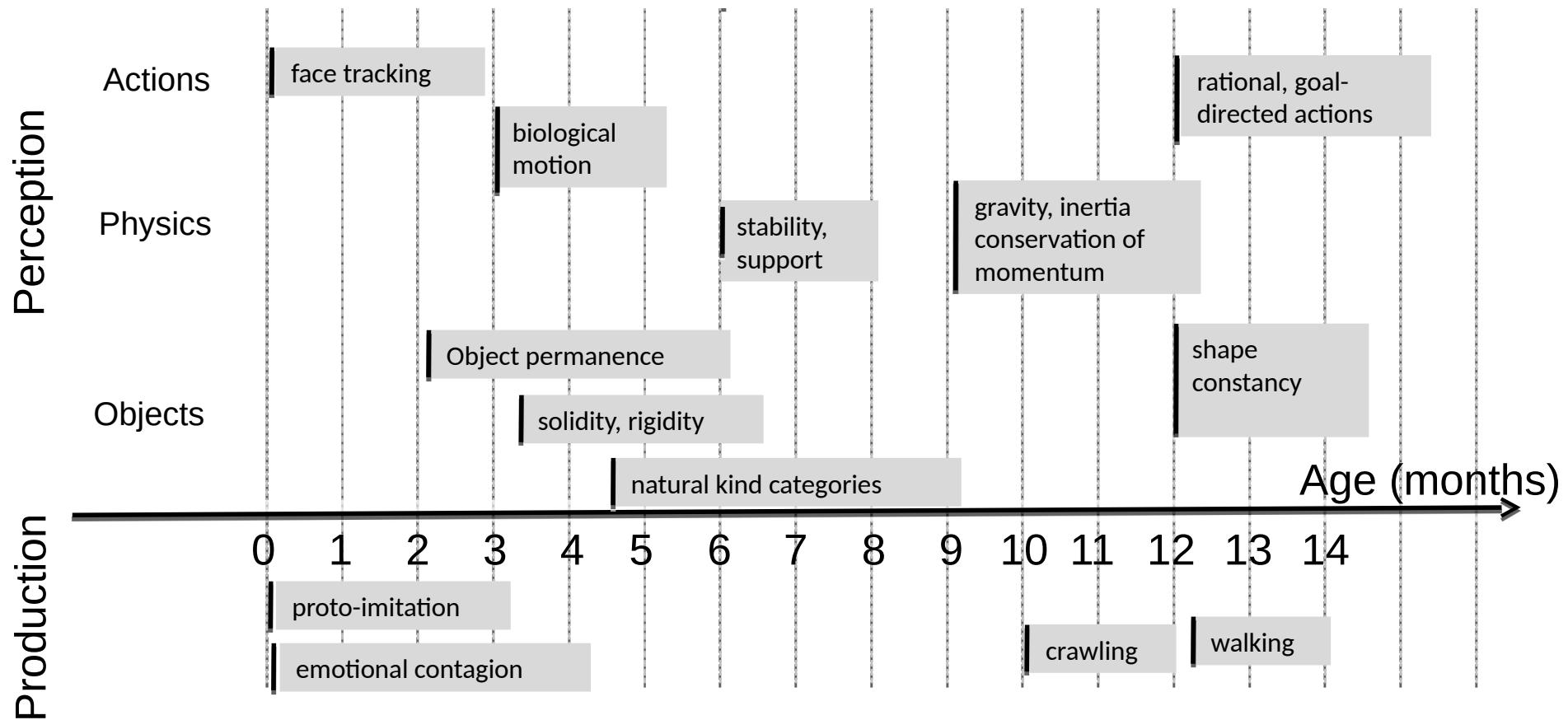


# How do humans and animals learn so quickly?

Not supervised.  
Not Reinforced.



# When infants learn models of the world [after Emmanuel Dupoux]



# How do Human and Animal Babies Learn?

- ▶ How do they learn how the world works?
- ▶ Largely by **observation**, with remarkably little interaction (initially).
- ▶ They accumulate enormous amounts of **background knowledge**
  - ▶ About the structure of the world, like intuitive physics.
- ▶ Perhaps **common sense** emerges from this knowledge?



Photos courtesy of  
Emmanuel Dupoux

# Self-Supervised Learning

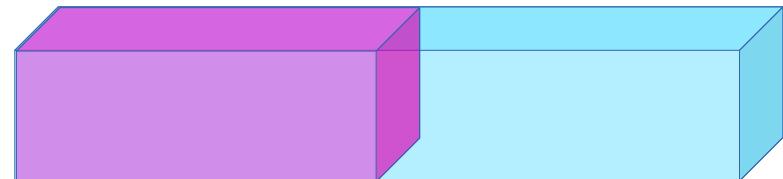
Capture dependencies.  
Predict everything from everything else.



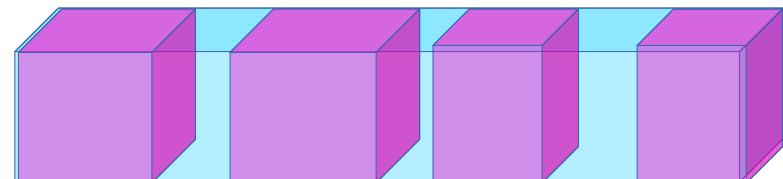
# Self-Supervised Learning = Filling in the Blanks

- ▶ Predict any part of the input from any other part.

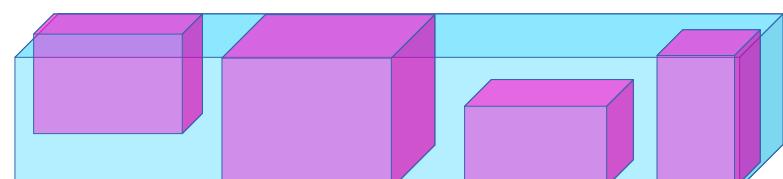
time or space →



- ▶ Predict the **future** from the **past**.



- ▶ Predict the **invisible** from the **visible**.



- ▶ Predict any **occluded, masked, or corrupted part from all available parts**.

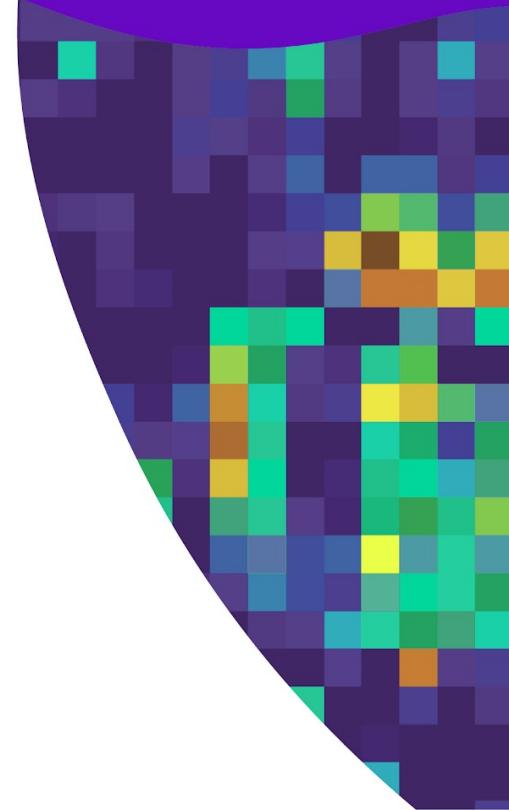
- ▶ Pretend there is a part of the input you don't know and predict that.
- ▶ Reconstruction = SSL when any part could be known or unknown

# Two Uses for Self-Supervised Learning

- ▶ **1. Learning hierarchical representations of the world**
  - ▶ SSL pre-training precedes a supervised or RL phase
- ▶ **2. Learning predictive (forward) models of the world**
  - ▶ Learning models for Model-Predictive Control, policy learning for control, or model-based RL.
- ▶ **Question:** how to represent uncertainty/multi-modality in the prediction?

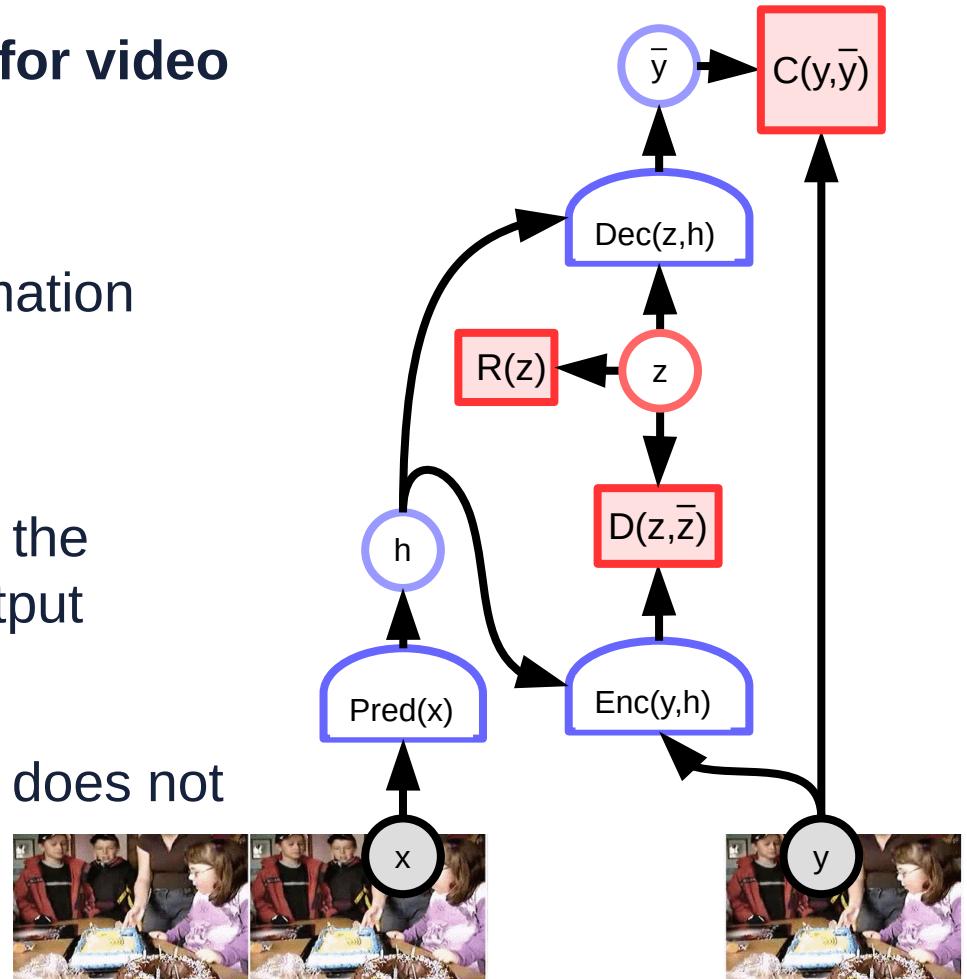
# Conditional RLVEBM for forward prediction

Multimodal video prediction



# Conditional Regularized AE

- ▶ Regularized Latent Variable EBM for video Prediction
- ▶ Predictor captures the useful information from the past in  $h$
- ▶ Regularized latent variable capture the unpredictable information in the output
- ▶ Regularizer ensures latent variable does not capture all the information.

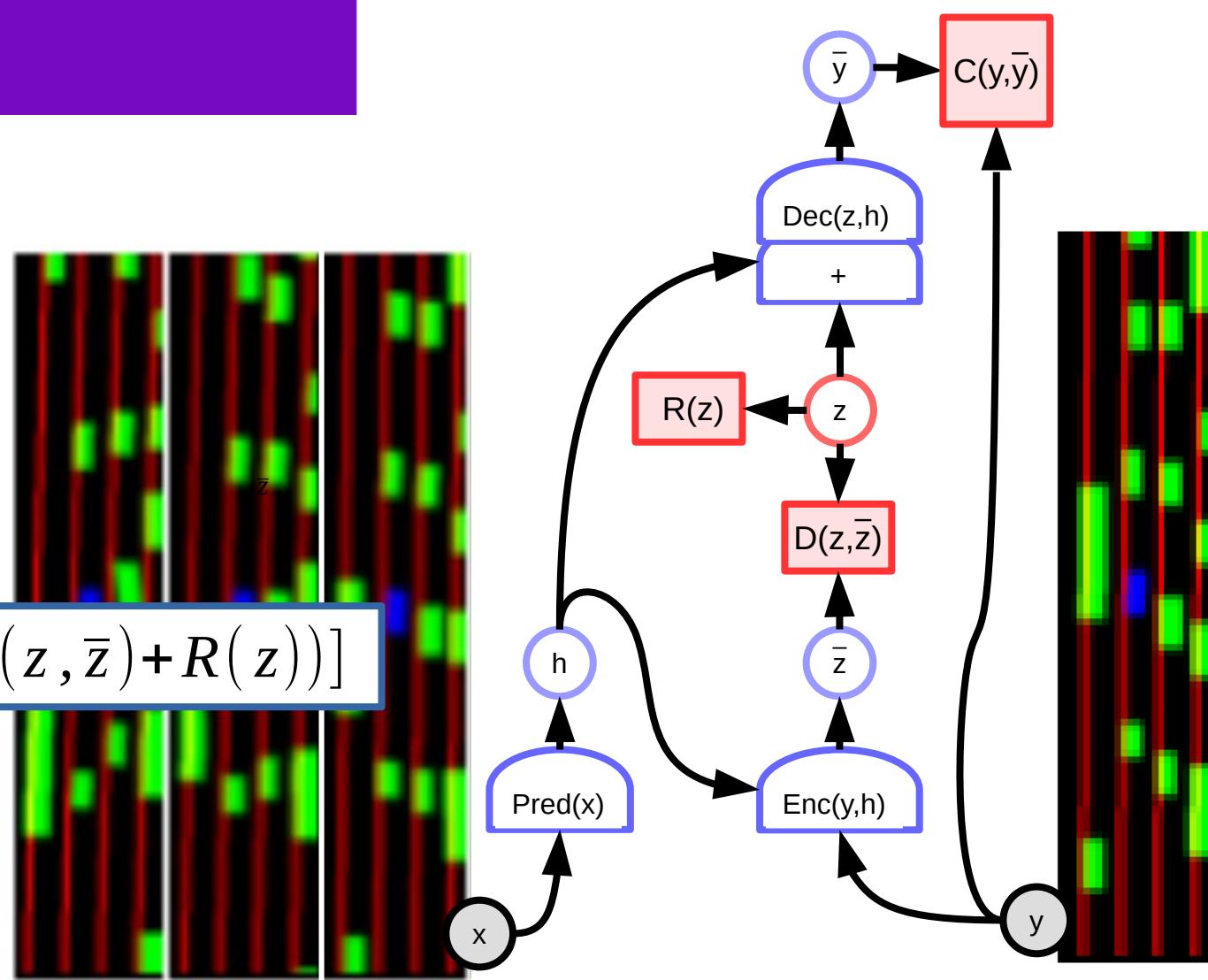


# VAE + Drop Out

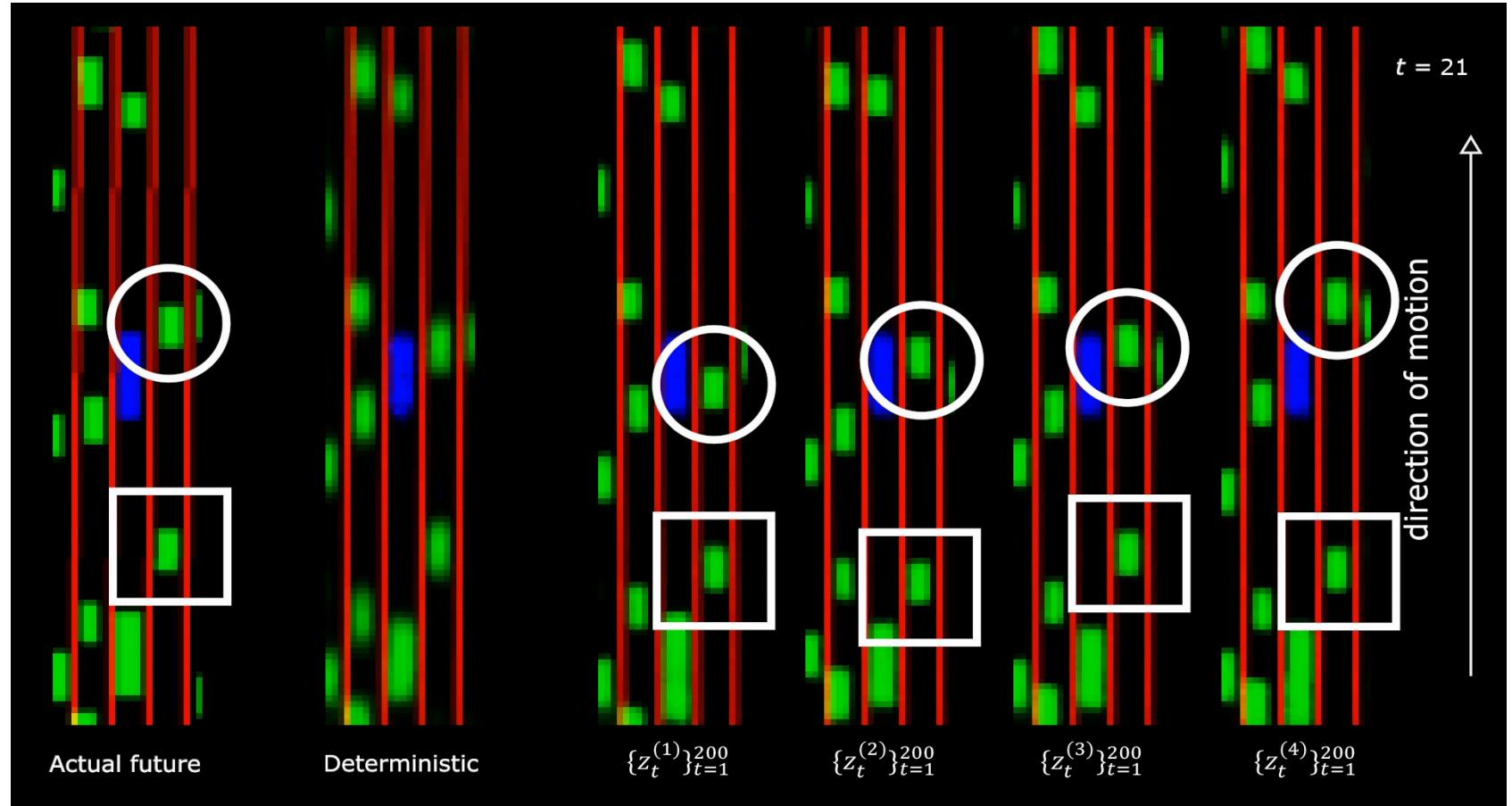
- ▶ **Training:**
  - ▶ Observe frames
  - ▶ Compute  $h$
  - ▶ Predict  $\bar{z}$  from encoder
  - ▶ Sample  $z$ , with:

$$P(z|\bar{z}) \propto \exp[-\beta(D(z, \bar{z}) + R(z))]$$

- ▶ Half the time, set  $z=0$
- ▶ Predict next frame
- ▶ backprop

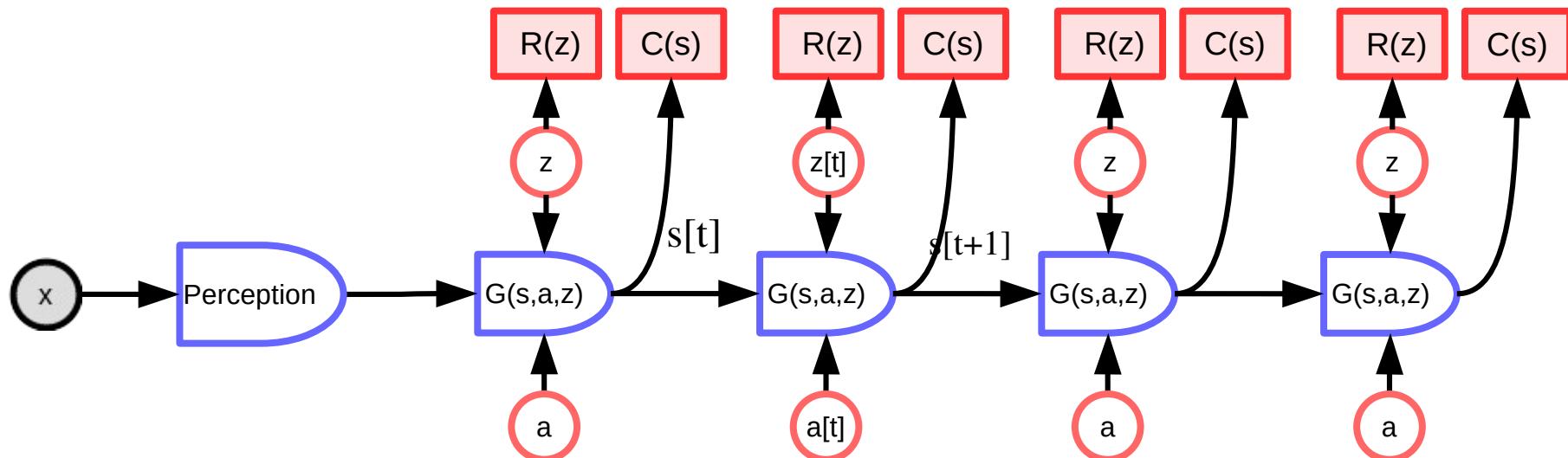


# Actual, Deterministic, VAE+Dropout Predictor/encoder



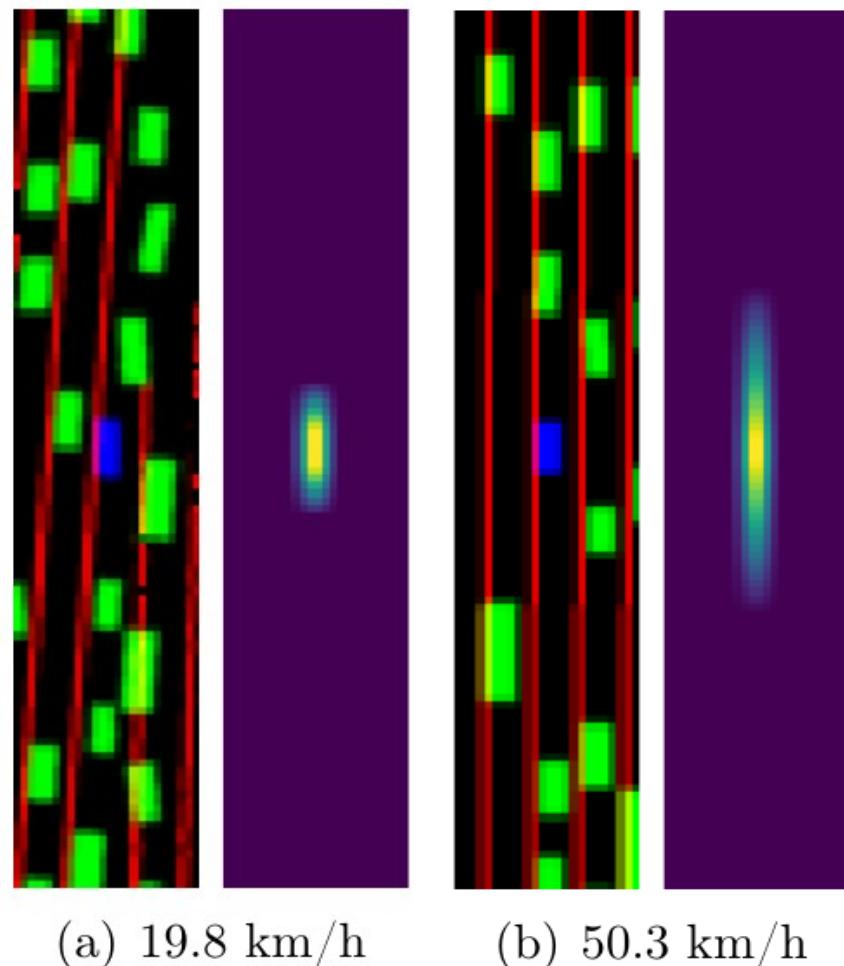
# Forward Model for Model-Predictive Control

- ▶ Forward model:  $s[t+1] = G(s[t], a[t], z[t])$
- ▶ Cost/Energy:  $f[t] = C(s[t])$
- ▶ Latent variable  $z$  sampled from  $q(z)$  proportional to  $\exp(-R(z))$
- ▶ Optimize  $(a[1], a[2], \dots, a[T]) = \operatorname{argmin} \sum_t C(s[t])$   
through backprop (== Kelley-Bryson adjoint state method)



# Cost optimized for Planning & Policy Learning

- ▶ **Differentiable cost function**
  - ▶ Increases as car deviates from lane
  - ▶ Increases as car gets too close to other cars nearby in a speed-dependent way
- ▶ **Uncertainty cost:**
  - ▶ Increases when the costs from multiple predictions (obtained through sampling of drop-out) have high variance.
  - ▶ Prevents the system from exploring unknown/unpredictable configurations that may have low cost.



# Forward Model for Gradient-Based Policy Learning

- ▶ Forward model:  $s[t+1] = G(s[t], a[t], z[t])$
- ▶ Cost/Energy:  $f[t] = C(s[t], a[t])$
- ▶ Latent variable  $z$  sampled from  $q(z)$  proportional to  $\exp(-R(z))$
- ▶ Policy:  $a[t] = P(s[t])$
- ▶ Learn  $P$  through backprop (== Kelley-Bryson adjoint state method)

