



NEW YORK UNIVERSITY

# Energy-Based Models (part 2)

<http://bit.ly/DLSP20>

Yann LeCun

NYU - Courant Institute & Center for Data Science

Facebook AI Research

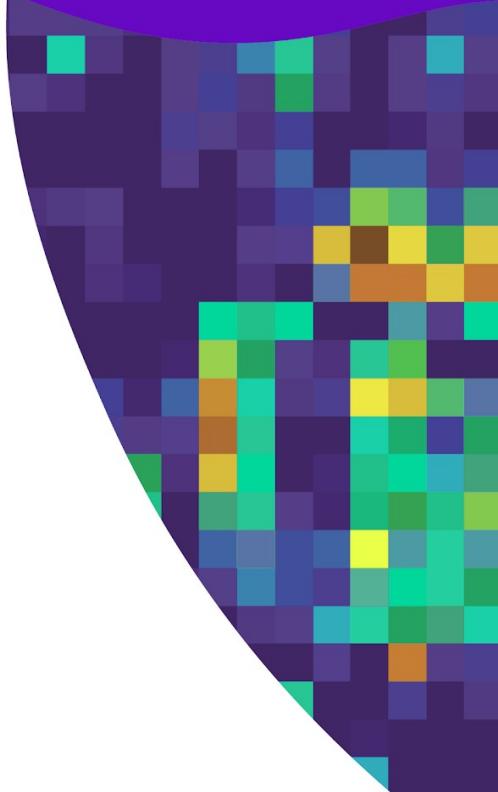
<http://yann.lecun.com>

TAs: Alfredo Canziani, Mark Goldstein

Deep Learning, NYU, Spring 2020

# Self-Supervised Learning

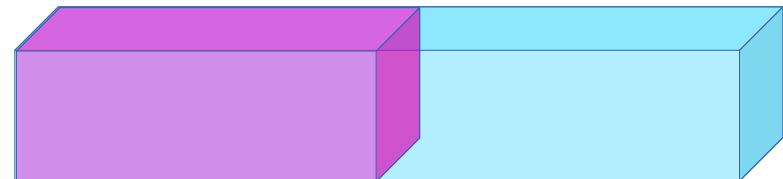
Predict everything  
from everything else



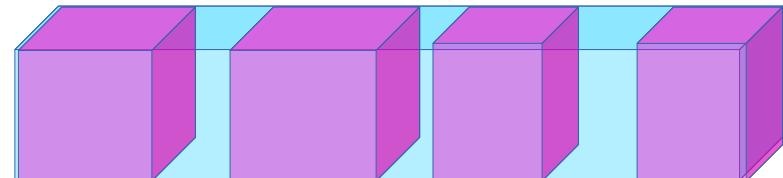
# Self-Supervised Learning = Filling in the Blanks

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

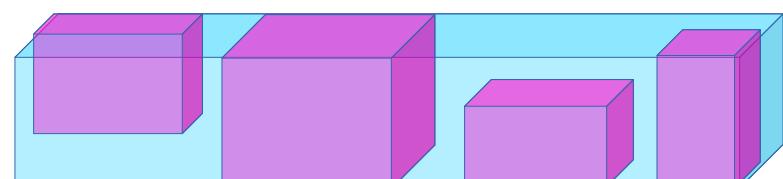
time or space →



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



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

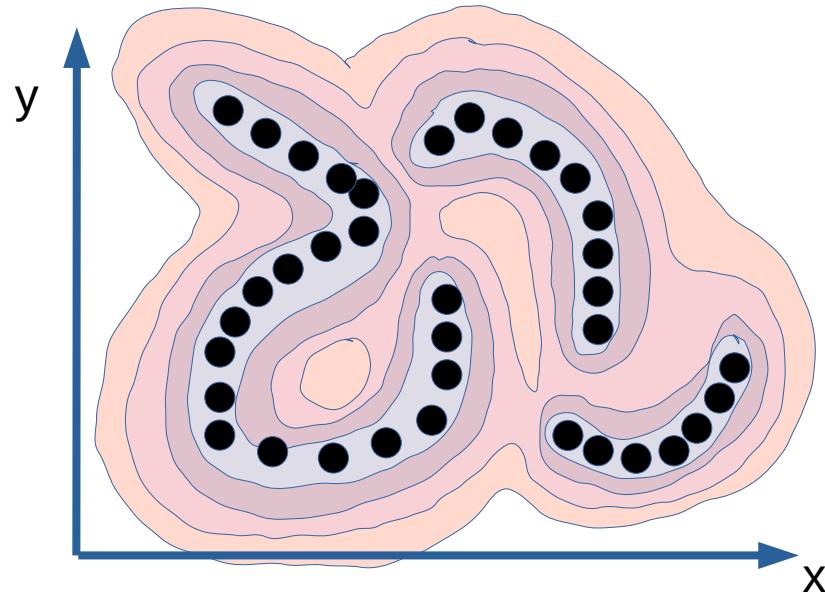


- ▶ Predict the **any occluded part** from all **available parts**.

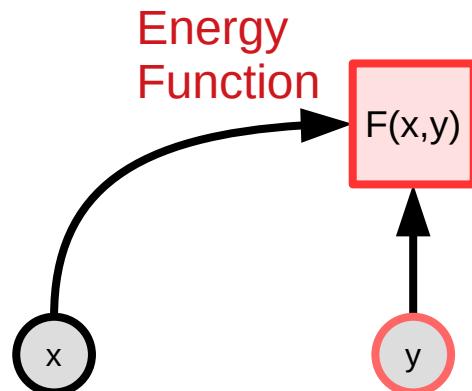
- ▶ Pretend there is a part of the input you don't know and predict that.
- ▶ Reconstruction = SSL when any part could be known or unknown

# Energy-Based Model

- ▶ Energy function that captures the  $x, y$  dependencies:
  - ▶ Low energy near the data points. Higher energy everywhere else.
  - ▶ If  $y$  is continuous,  $F$  should be smooth and differentiable, so we can use gradient-based inference algorithms.

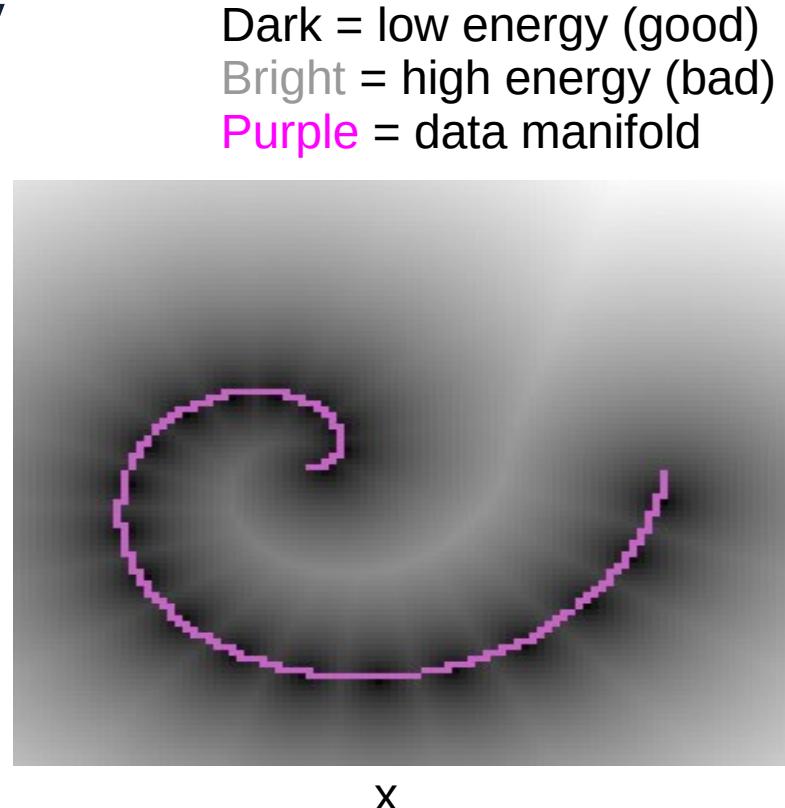
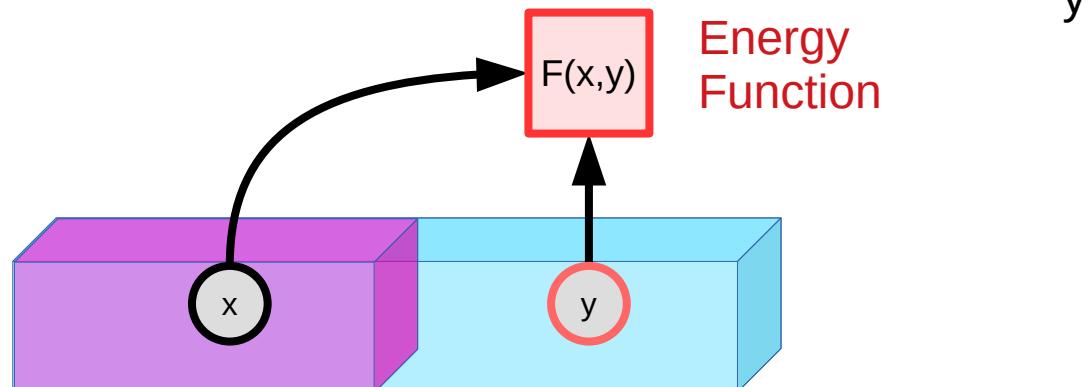


$$\check{y} = \operatorname{argmin}_y F(x, y)$$



# Energy-Based Model

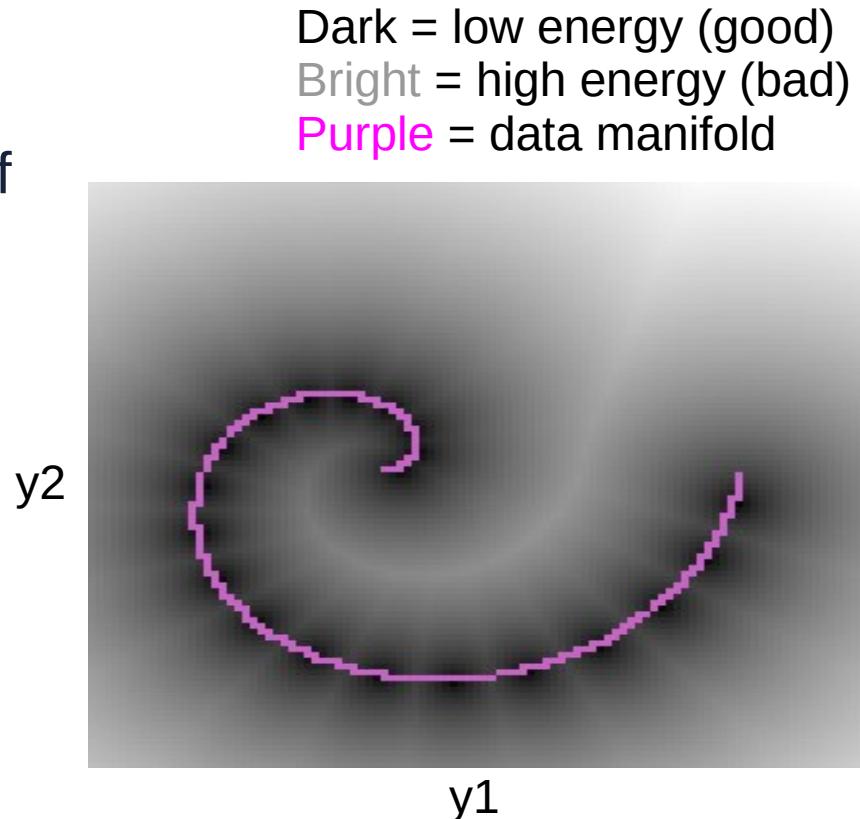
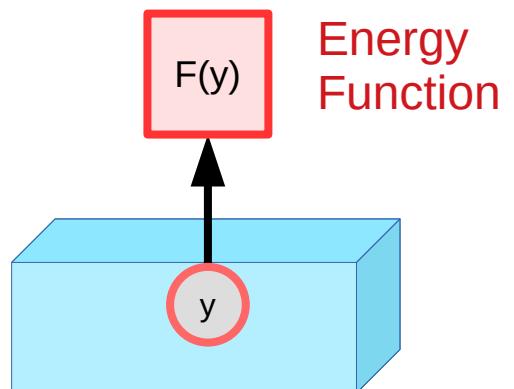
- ▶ **Scalar-valued energy function:  $F(x,y)$**
- ▶ measures the compatibility between  $x$  and  $y$
- ▶ Low energy:  $y$  is good prediction from  $x$
- ▶ High energy:  $y$  is bad prediction from  $x$
  
- ▶ Inference:  $\check{y} = \operatorname{argmin}_y F(x, y)$



[Figure from M-A Ranzato's PhD thesis]

# Energy-Based Model: unconditional version

- ▶ **Scalar-valued energy function:  $F(y)$**
- ▶ measures the compatibility between the components of  $y$
- ▶ If we don't know in advance which part of  $y$  is known and which part is unknown
- ▶ Example: auto-encoders, generative models (energy =  $-\log$  likelihood)



Dark = low energy (good)  
Bright = high energy (bad)  
**Purple** = data manifold

# Latent-Variable EBM

- ▶ Allowing multiple predictions through a latent variable

- ▶ Conditional:

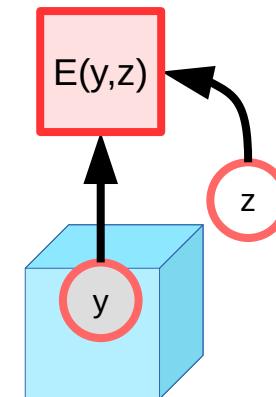
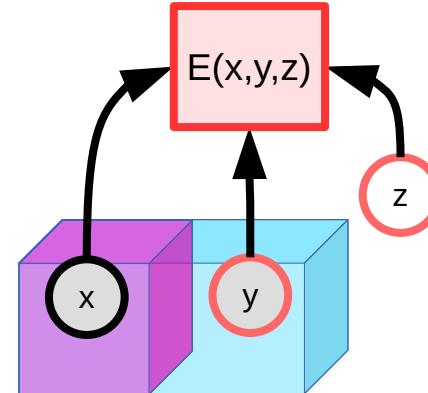
$$F(x, y) = \min_z E(x, y, z)$$

$$F(x, y) = -\frac{1}{\beta} \log \left[ \int_z \exp(-\beta E(x, y, z)) \right]$$

- ▶ Unconditional

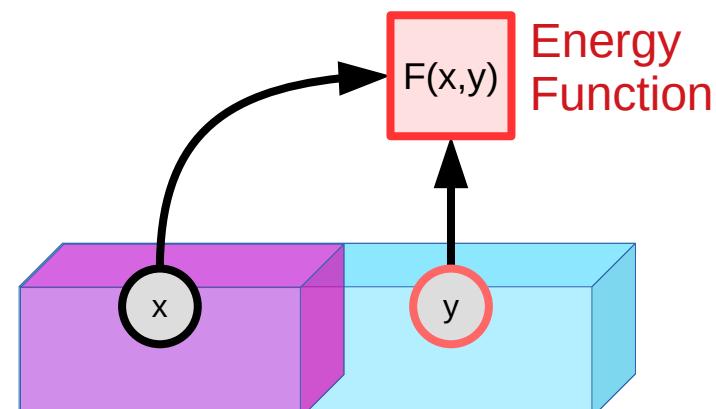
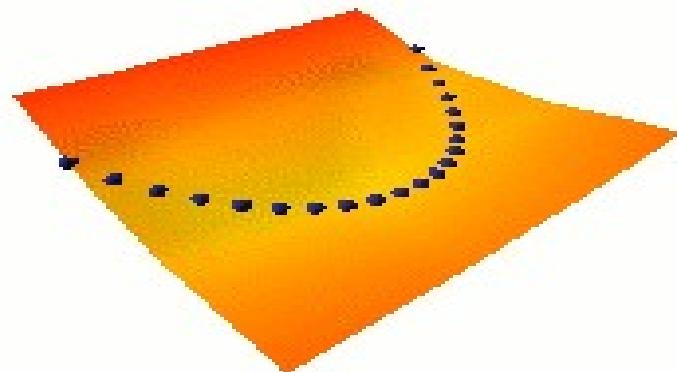
$$F(y) = \min_z E(y, z)$$

$$F(y) = -\frac{1}{\beta} \log \left[ \int_z \exp(-\beta E(y, z)) \right]$$



# Training an Energy-Based Model

- ▶ Parameterize  $F(x,y)$
- ▶ Get training data  $(x[i], y[i])$
- ▶ Shape  $F(x,y)$  so that:
  - ▶  $F(x[i], y[i])$  is strictly smaller than  $F(x[i], y)$  for all  $y$  different from  $y[i]$
  - ▶ Keep  $F$  smooth (probabilistic methods break that!)
- ▶ Two classes of learning methods:
  - ▶ 1. **Contrastive methods**: push down on  $F(x[i], y[i])$ , push up on other points  $F(x[i], y')$
  - ▶ 2. **Architectural Methods**: build  $F(x,y)$  so that the volume of low energy regions is limited or minimized through regularization



# Seven Strategies to Shape the Energy Function

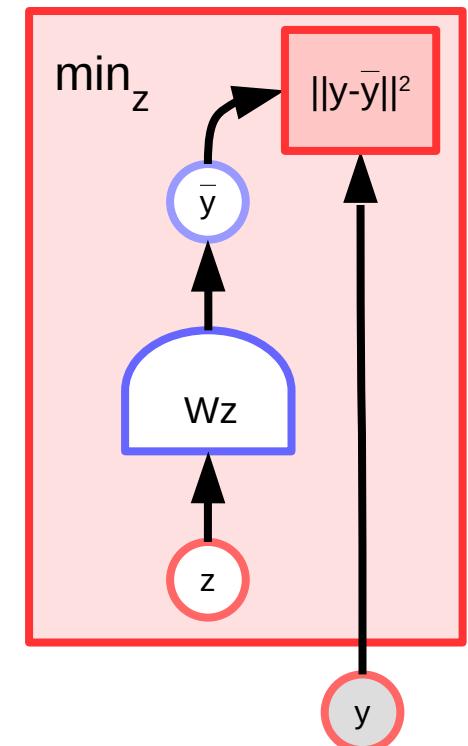
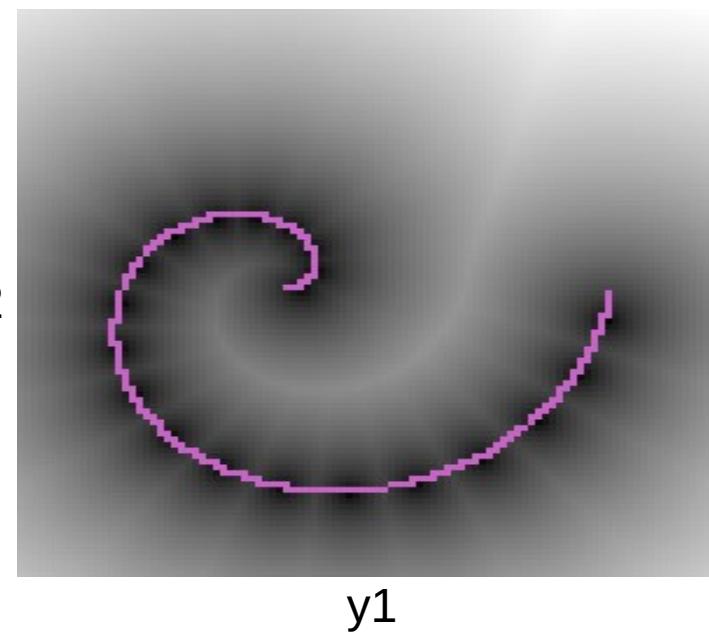
- ▶ **Contrastive:** [they all are different ways to pick which points to push up]
  - ▶ C1: push down of the energy of data points, push up everywhere else: Max likelihood (needs tractable partition function or variational approximation)
  - ▶ C2: push down of the energy of data points, push up on chosen locations: max likelihood with MC/MMC/HMC, Contrastive divergence, [Metric learning](#), 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)
- ▶ **Architectural:** [they all are different ways to limit the information capacity of the code]
  - ▶ A1: build the machine so that the volume of low energy stuff is bounded: PCA, K-means, Gaussian Mixture Model, Square ICA...
  - ▶ A2: use a regularization term that measures the volume of space that has low energy: Sparse coding, [sparse auto-encoder](#), LISTA, Variational auto-encoders
  - ▶ 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

# Latent-Variable EBM example: K-means

- ▶ Decoder is linear,  $z$  is a 1-hot vector (discrete)
- ▶ Energy function:  $E(y, z) = \|y - Wz\|^2 \quad z \in 1\text{hot}$
- ▶ Inference by exhaustive search

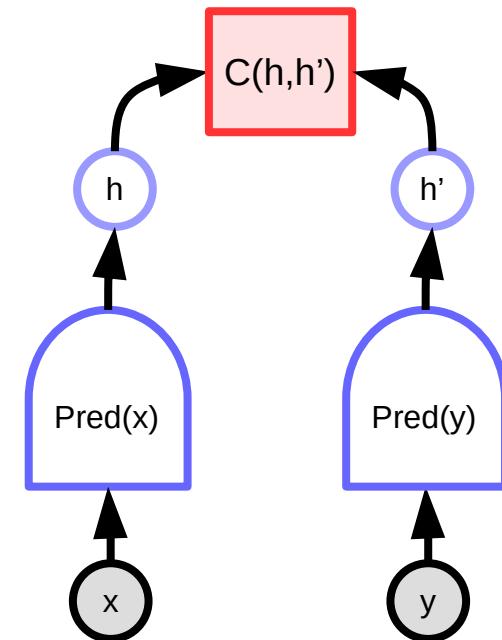
$$F(y) = \min_z E(y, z)$$

- ▶ Volume of low-energy regions limited by number of prototypes  $k$



# Contrastive Embedding

- ▶ Distance measured in feature space
- ▶ Multiple “predictions” through feature invariance
- ▶ Siamese nets, metric learning [YLC NIPS’93,CVPR’05,CVPR’06]
- ▶ **Advantage: no pixel-level reconstruction**
- ▶ **Difficulty: hard negative mining**
- ▶ Successful examples for images:
  - ▶ DeepFace [Taigman et al. CVPR’14]
  - ▶ PIRL [Misra & van der Maaten Arxiv:1912.01991]
  - ▶ MoCo [He et al. Arxiv:1911.05722]
  - ▶ SimCLR [Chen et al. Arxiv:2002.05709]
- ▶ Video / Audio
  - ▶ Temporal proximity [Taylor CVPR’11]
  - ▶ Slow feature [Goroshin NIPS’15]



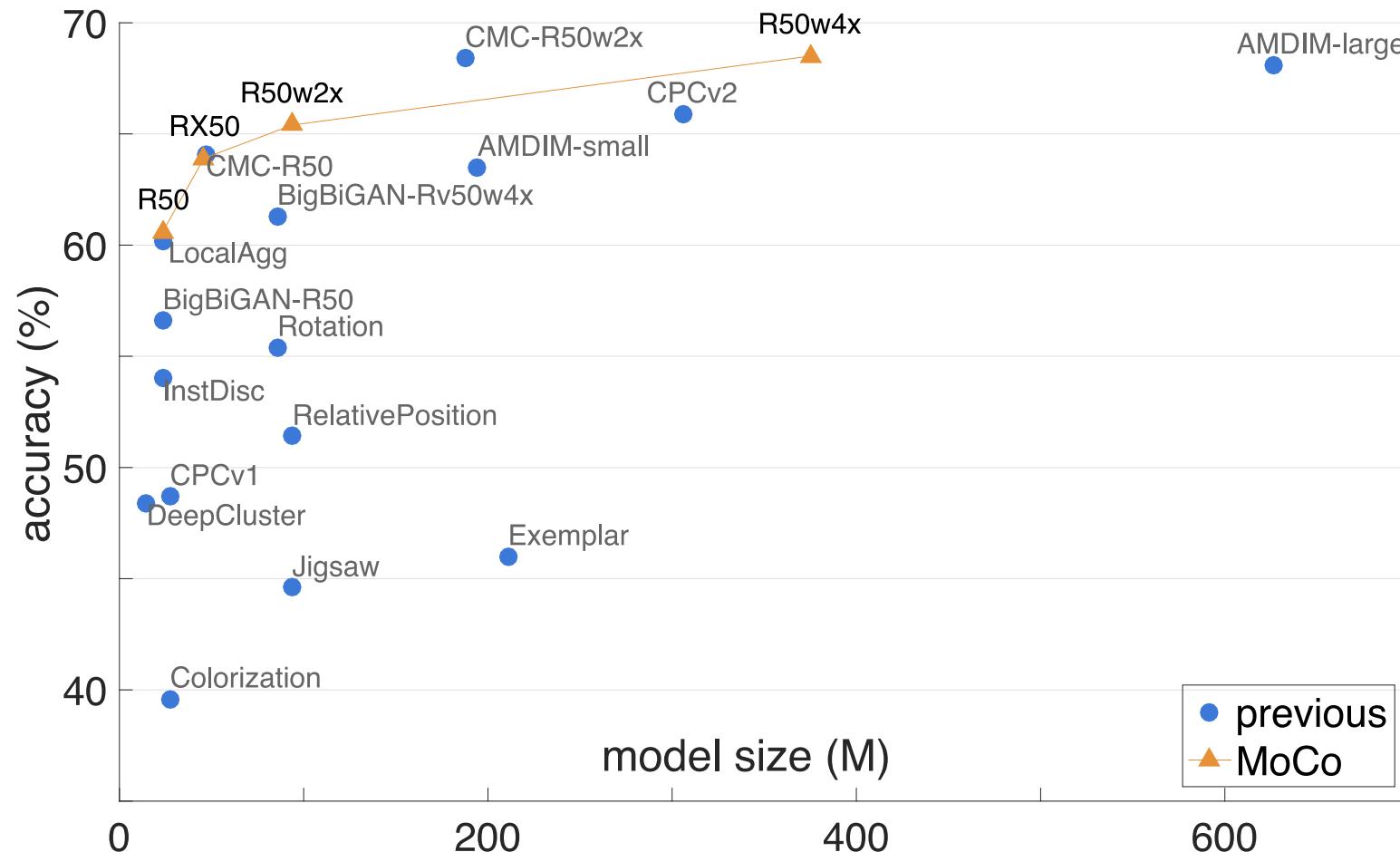
Positive pair:  
Make F small



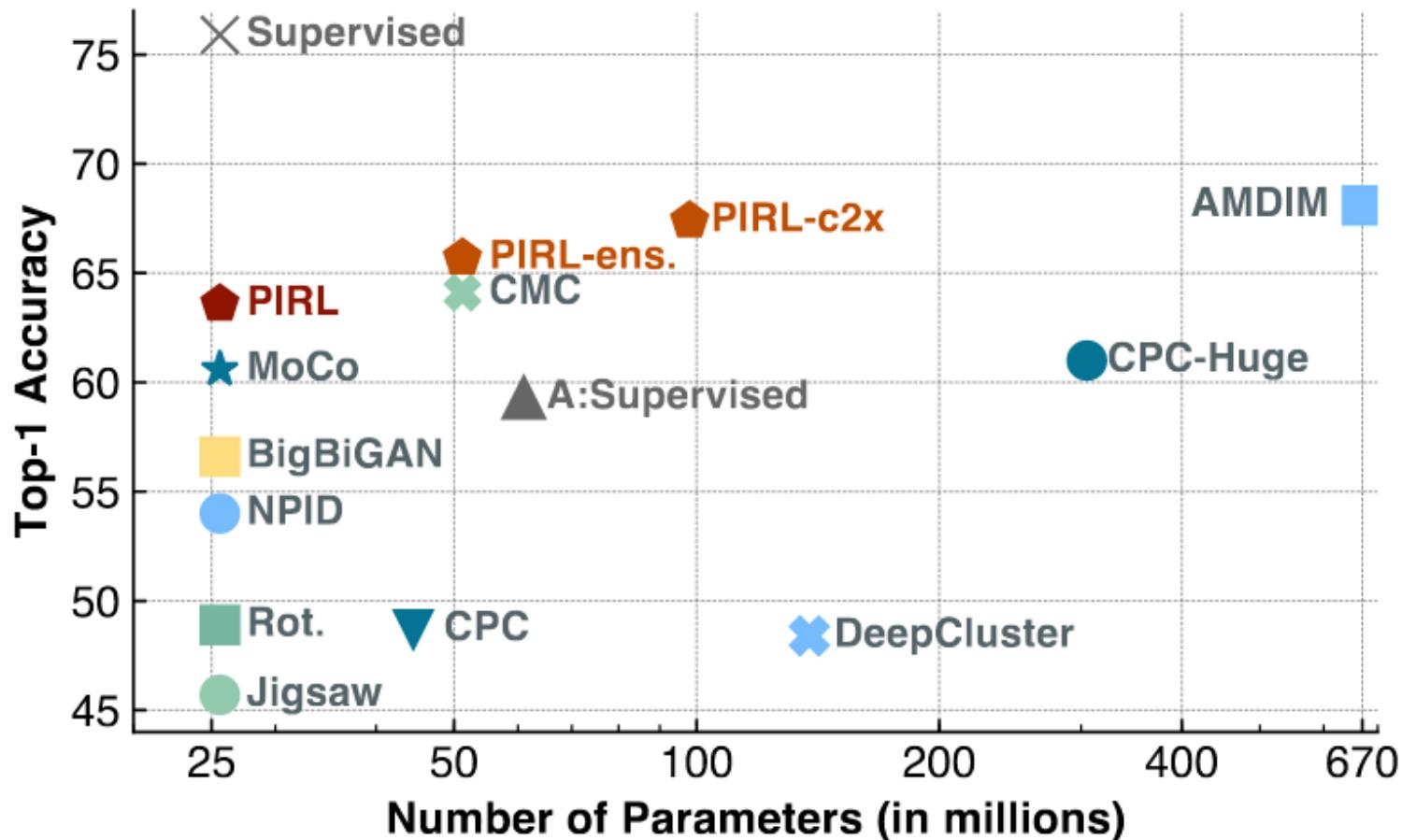
Negative pair:  
Make F large



# MoCo on ImageNet [He et al. Arxiv:1911.05722]



# PIRL on ImageNet [Misra & van der Maaten Arxiv:1912.01991]



# PIRL objective function: NCE

- ▶ **NCE = noise contrastive estimator**
- ▶  $s(u, v)$  = cosine similarity measure between ConvNet outputs
- ▶  $f(u), g(v)$  different “heads” of the convnets to compute representations

$$h(\mathbf{v_I}, \mathbf{v_{I^t}}) = \frac{\exp\left(\frac{s(\mathbf{v_I}, \mathbf{v_{I^t}})}{\tau}\right)}{\exp\left(\frac{s(\mathbf{v_I}, \mathbf{v_{I^t}})}{\tau}\right) + \sum_{\mathbf{I}' \in \mathcal{D}_N} \exp\left(\frac{s(\mathbf{v_{I^t}}, \mathbf{v_{I'}})}{\tau}\right)}$$

$$\begin{aligned} L_{\text{NCE}} (\mathbf{I}, \mathbf{I}^t) &= -\log [h(f(\mathbf{v_I}), g(\mathbf{v_{I^t}}))] \\ &\quad - \sum_{\mathbf{I}' \in \mathcal{D}_N} \log [1 - h(g(\mathbf{v_I^t}), f(\mathbf{v_{I'}}))] \end{aligned}$$

# PIRL Results on ImageNet

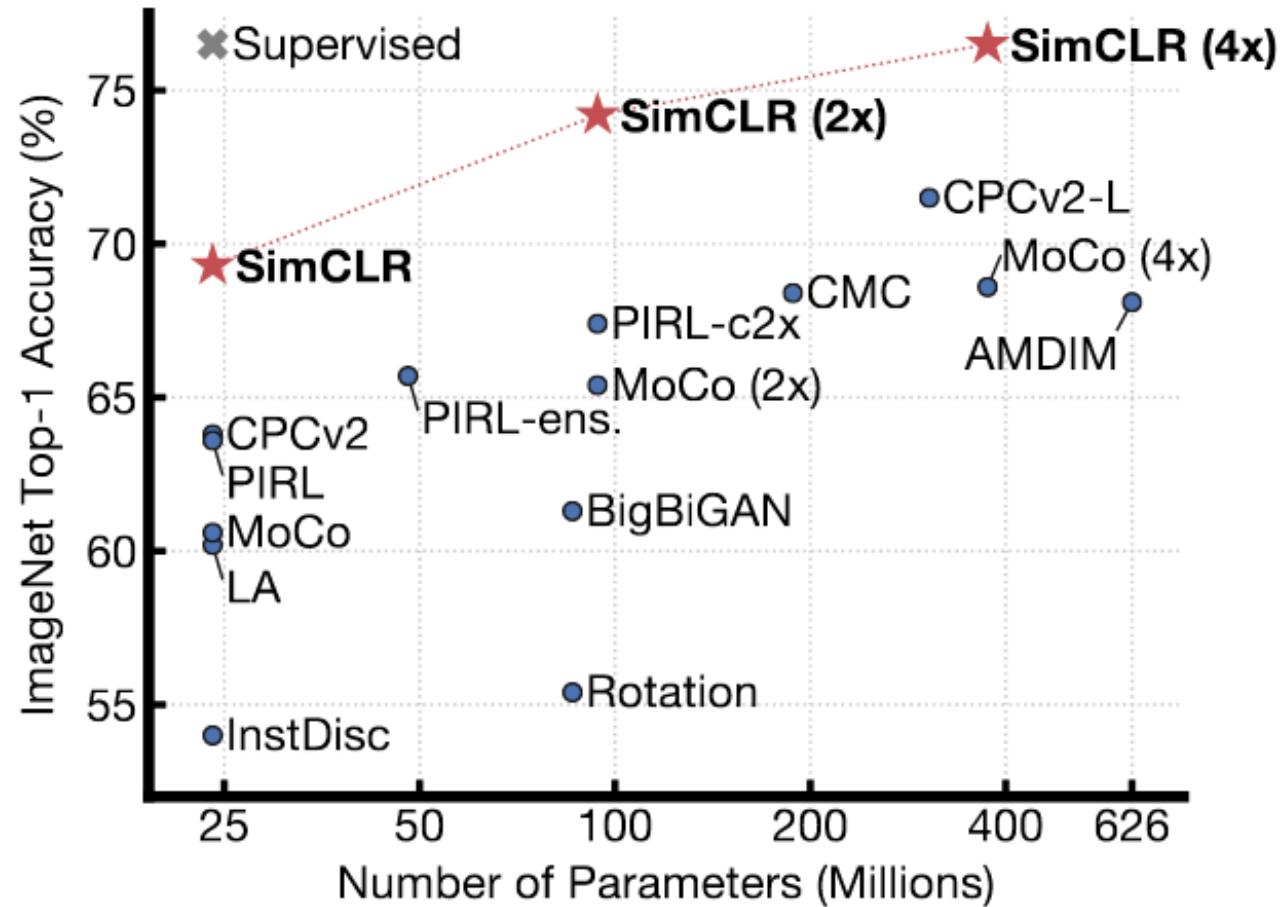
- ▶ Single-crop top-5 accuracy
- ▶ Contrastive training on full training set.
- ▶ Supervised fine tuning on 1% and 10% of training samples

Method	Backbone	Data fraction →	1%	10%
			Top-5 Accuracy	
Random initialization [72]	R-50	22.0	59.0	
NPID [72]	R-50	39.2	77.4	
Jigsaw [19]	R-50	45.3	79.3	
NPID++ [72]	R-50	52.6	81.5	
VAT + Ent Min. [20, 45]	R-50v2	47.0	83.4	
S <sup>4</sup> L Exemplar [75]	R-50v2	47.0	83.7	
S <sup>4</sup> L Rotation [75]	R-50v2	53.4	<b>83.8</b>	
<b>PIRL (ours)</b>	R-50	<b>57.2</b>	<b>83.8</b>	
Colorization [36]	R-152	29.8	62.0	
CPC-Largest [26]	R-170 and R-11	64.0	84.9	

**Table 3: Semi-supervised learning on ImageNet.** Single-crop top-5 accuracy on the ImageNet validation set of self-supervised models that are finetuned on 1% and 10% of the ImageNet training data, following [72]. All numbers except for Jigsaw, NPID++ and PIRL are adopted from the corresponding papers. Best performance is **boldfaced**.

# SimCLR [Chen et al. Arxiv:2002.05709]

- ▶ Sophisticated corruption / data augmentation methods
- ▶ Massive amount of training (on TPU)
  - ▶ Shows the limits of scalability of contrastive methods



# SimCLR [Chen et al. Arxiv:2002.05709]

- ▶ **ImageNet accuracy with supervised training on 1% and 10% of training data**
- ▶ After SimCLR contrastive pretraining.

Method	Architecture	Label fraction		
		1%	10%	Top 5
<i>Methods using other label-propagation:</i>				
Pseudo-label	ResNet50	51.6	82.4	
VAT+Entropy Min.	ResNet50	47.0	83.4	
UDA (w. RandAug)	ResNet50	-	88.5	
FixMatch (w. RandAug)	ResNet50	-	89.1	
S4L (Rot+VAT+En. M.)	ResNet50 (4×)	-	91.2	
<i>Methods using representation learning only:</i>				
InstDisc	ResNet50	39.2	77.4	
BigBiGAN	RevNet-50 (4×)	55.2	78.8	
PIRL	ResNet-50	57.2	83.8	
CPC v2	ResNet-161(*)	77.9	91.2	
SimCLR (ours)	ResNet-50	75.5	87.8	
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2	
SimCLR (ours)	ResNet-50 (4×)	<b>85.8</b>	<b>92.6</b>	

Table 7. ImageNet accuracy of models trained with few labels.

# SimCLR [Chen et al. Arxiv:2002.05709]

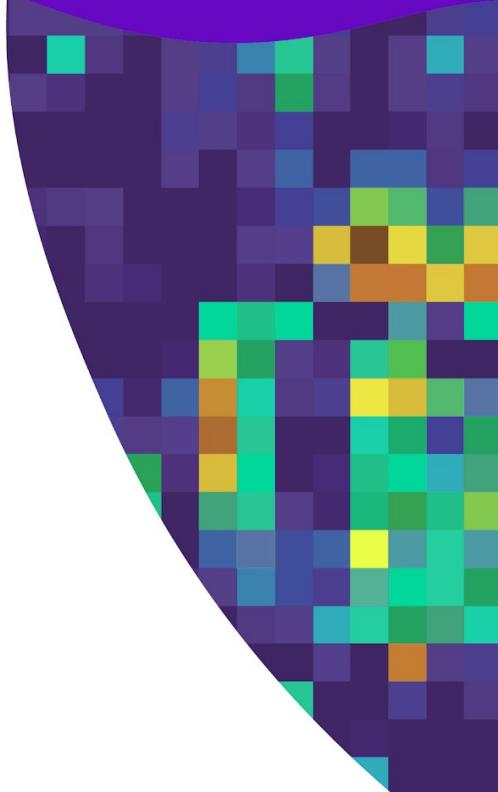
- ▶ Training a linear classifier on the pre-trained features

Method	Architecture	Param.	Top 1	Top 5
<i>Methods using ResNet-50:</i>				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	<b>69.3</b>	<b>89.0</b>
<i>Methods using other architectures:</i>				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (*)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	<b>76.5</b>	<b>93.2</b>

Table 6. ImageNet accuracies of linear classifiers trained on representations learned with different self-supervised methods.

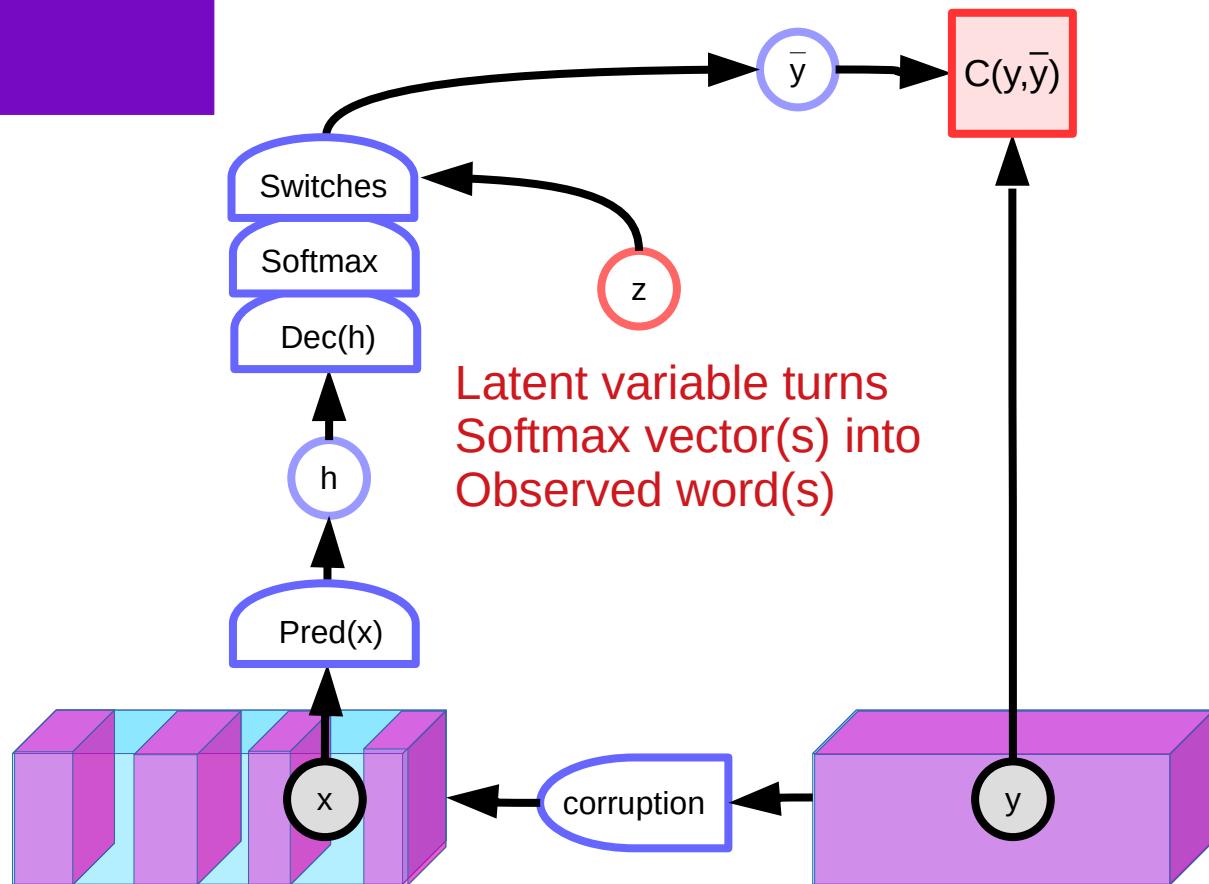
# Denoising Auto-Encoder

Contrastive Self-Supervised  
Learning Method



# Denoising AE: discrete

- ▶ [Vincent et al. JMLR 2008]
- ▶ Masked Auto-Encoder
- ▶ [BERT et al.]
- ▶ Issues:
  - ▶ latent variables are in output space
  - ▶ No abstract LV to control the output
  - ▶ How to cover the space of corruptions?

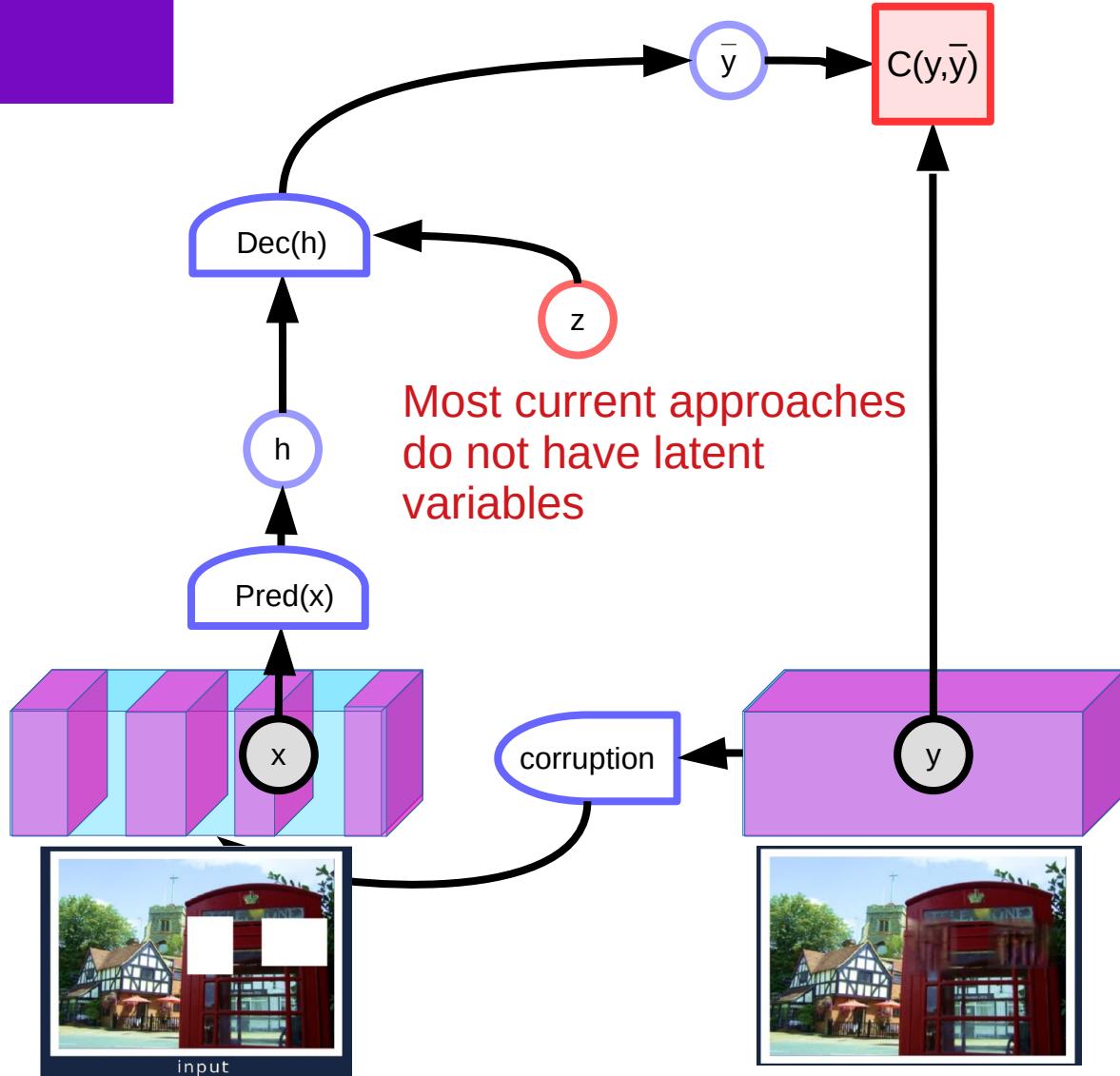
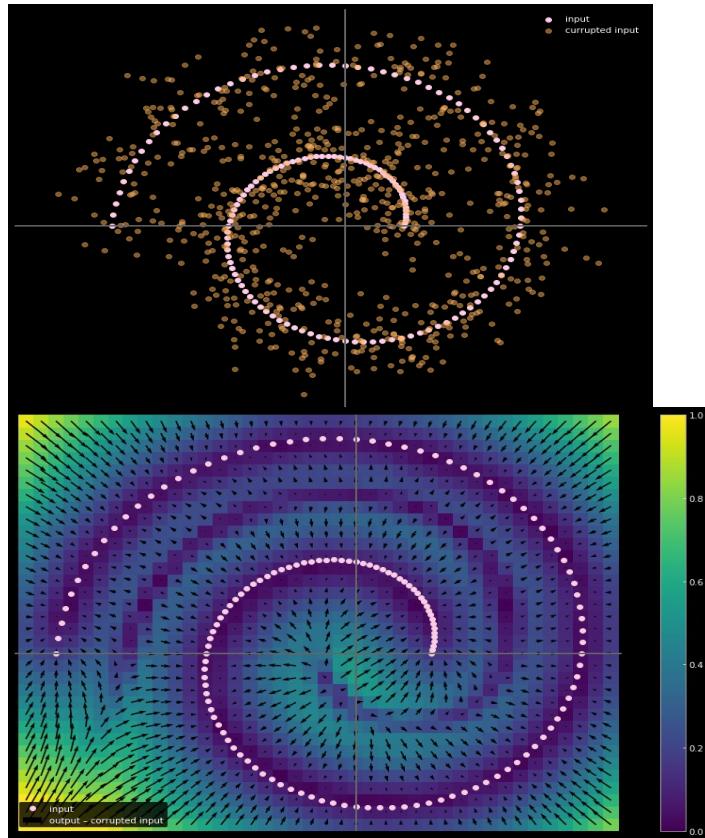


This is a [...] of text extracted  
[...] a large set of [...] articles

This is a piece of text extracted  
from a large set of news articles

# Denoising AE: continuous

- ▶ Image inpainting [Pathak 17]
- ▶ Latent variables? GAN?



# Other Contrastive Methods

■ 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)

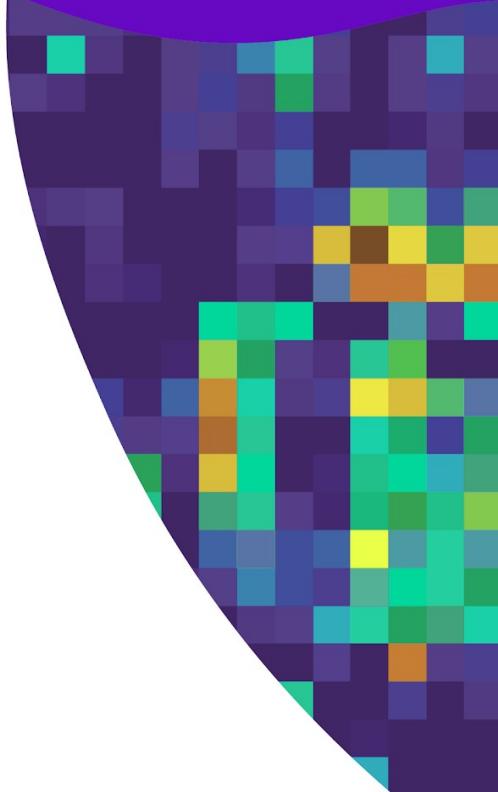
■ Persistent CD: use a bunch of “particles” and remember their positions

- ▶ Make them roll down the energy surface with noise
- ▶ Push up on the energy wherever they are
- ▶ Faster than CD

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

# Regularized Latent-Variable EBM

Architectural SSL methods  
(non contrastive)

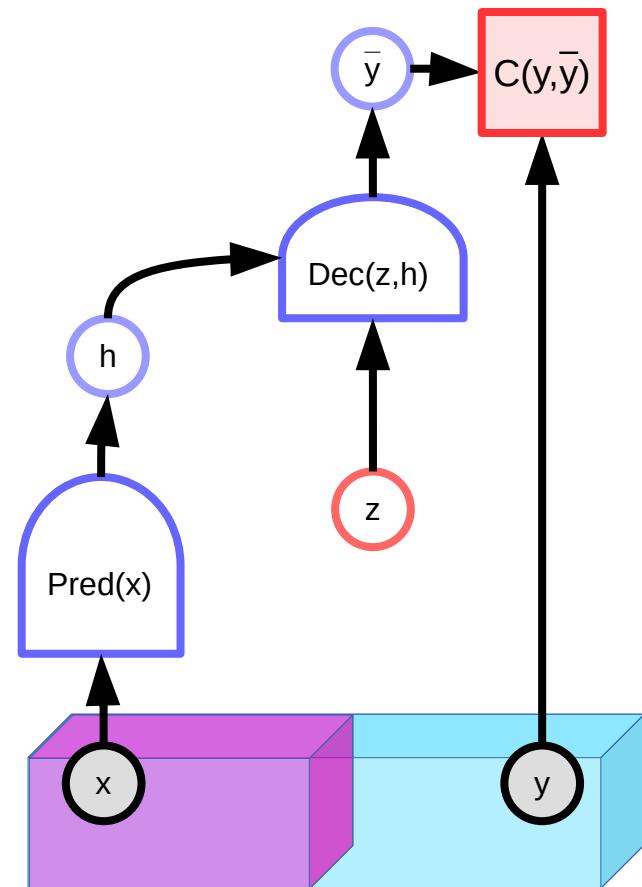


# Prediction with Latent Variables

- ▶ 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  $z$ .**

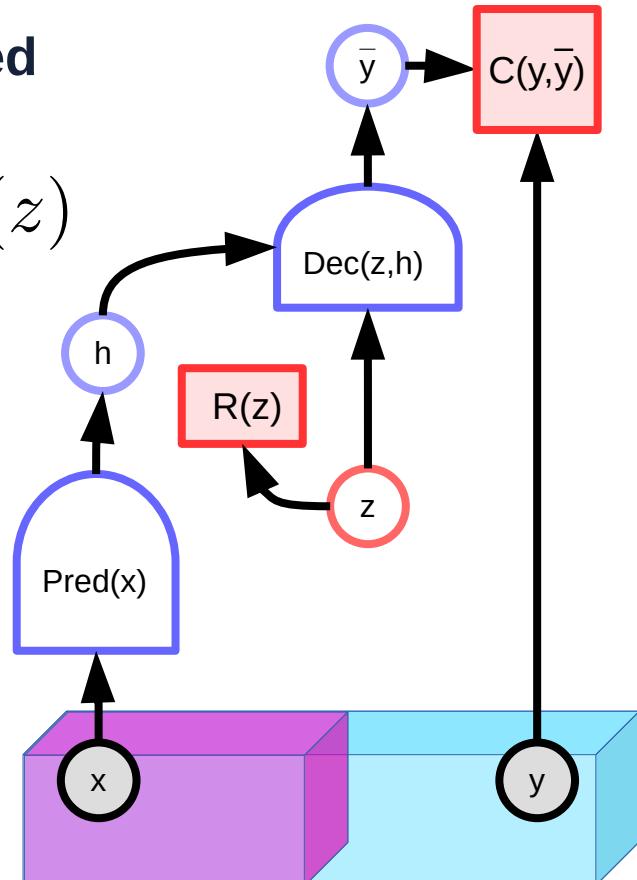


# Regularized Latent Variable EBM

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

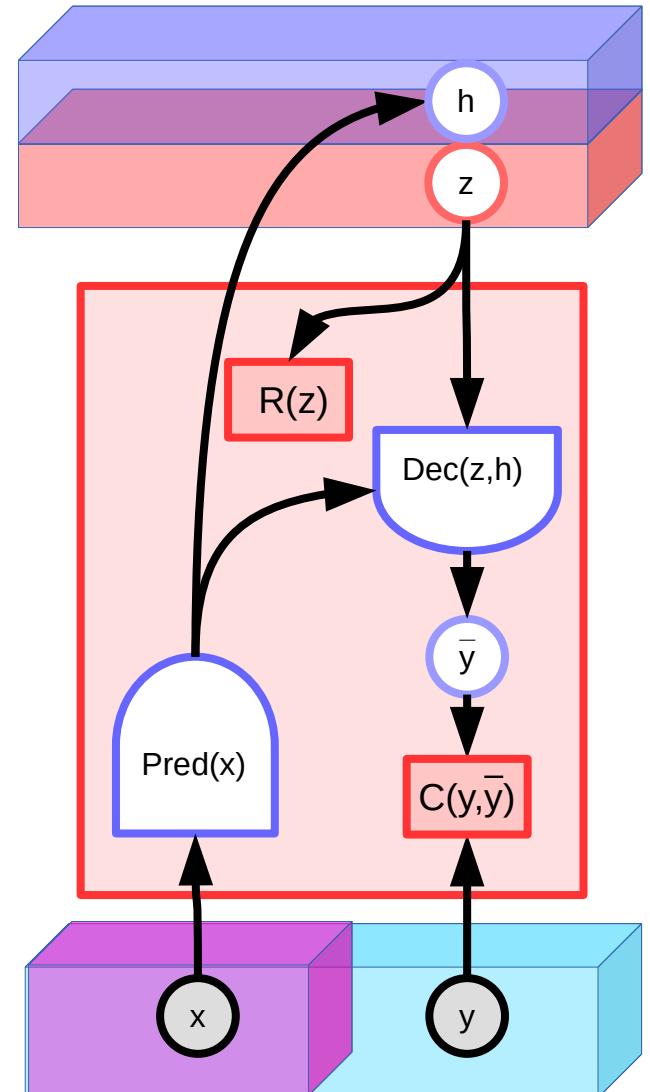
$$E(x, y, z) = C(y, \text{Dec}(\text{Pred}(x), z)) + \lambda R(z)$$

- ▶ 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>



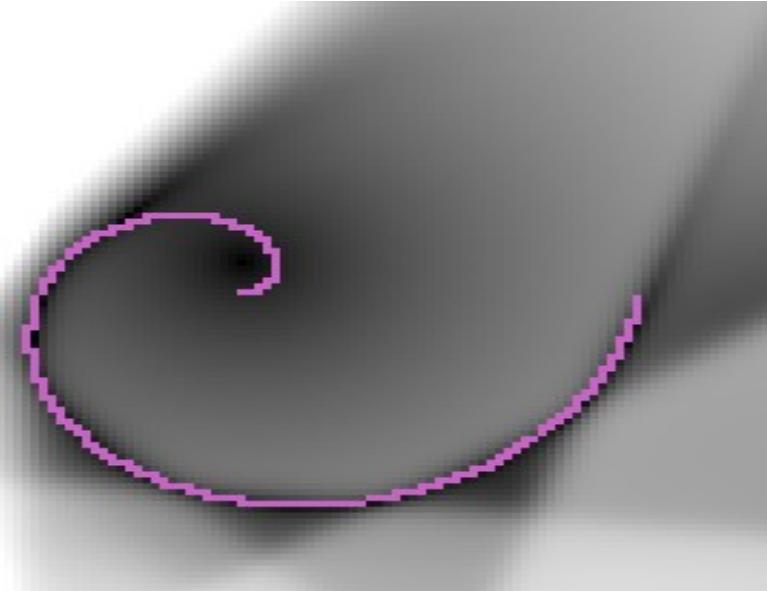
# Sequence → Abstract Features

- ▶ Regularized LV EBM is passed over a sequence (e.g. a video, audio, text)
- ▶ The sequence of corresponding  $h$  and  $z$  is collected
- ▶ It contains all the information about the input sequence
- ▶  $h$  contains the information in  $x$  that is useful to predict  $y$
- ▶  $z$  contains the complementary information, not present in  $x$  or  $h$ .
- ▶ Several such SSL modules can be stacked to learn hierarchical representations of sequences

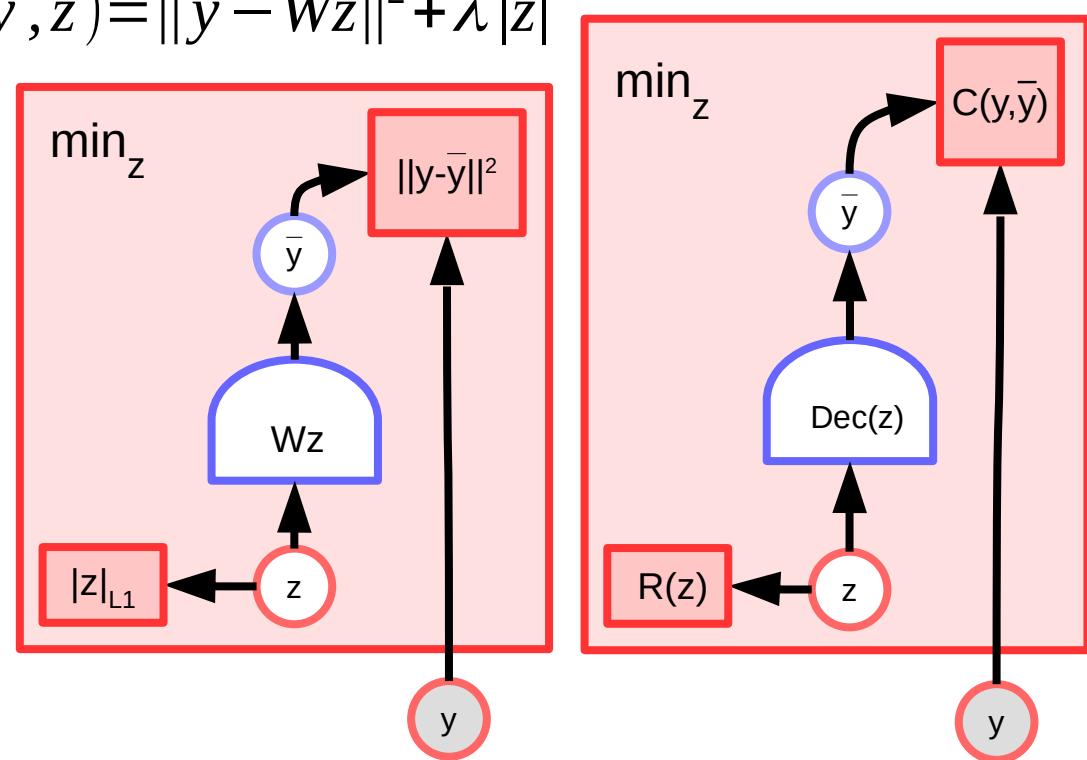


# Unconditional Regularized Latent Variable EBM

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



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



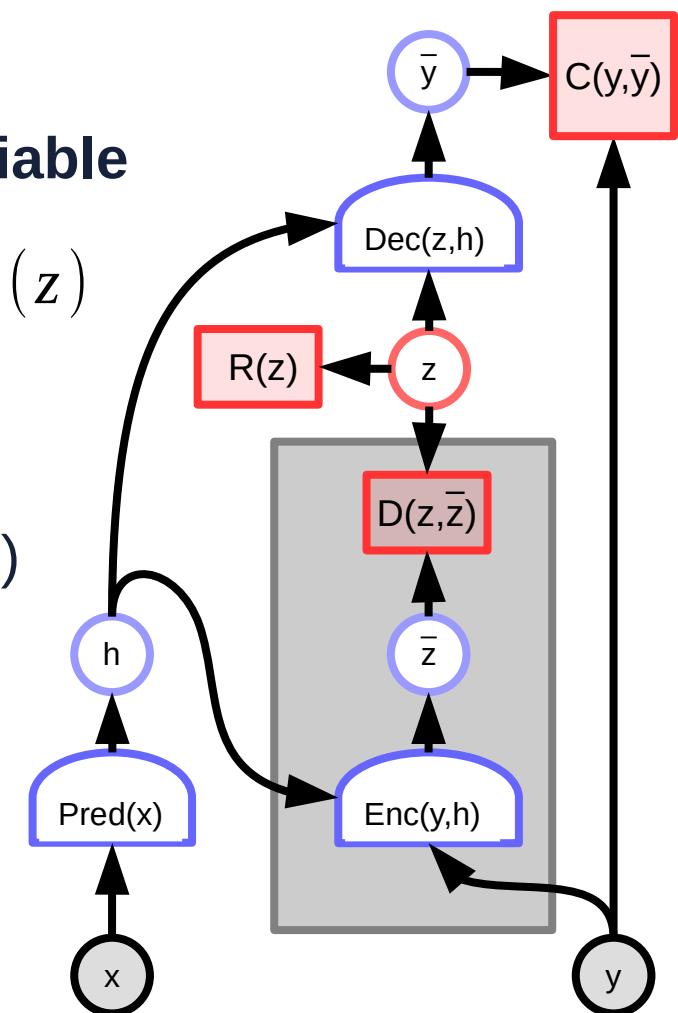
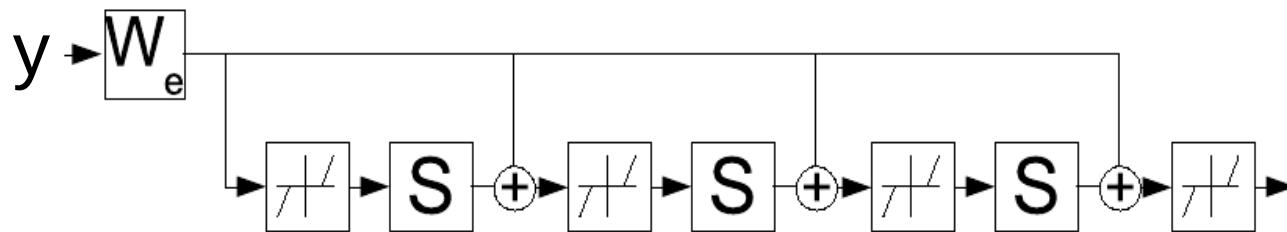
# LatVar inference is expensive!

- ▶ Let's train an encoder to predict the latent variable

$$E(x, y, z) = C(y, Dec(z, h)) + D(z, Enc(x, y)) + \lambda R(z)$$

- ▶ Predictive Sparse Modeling

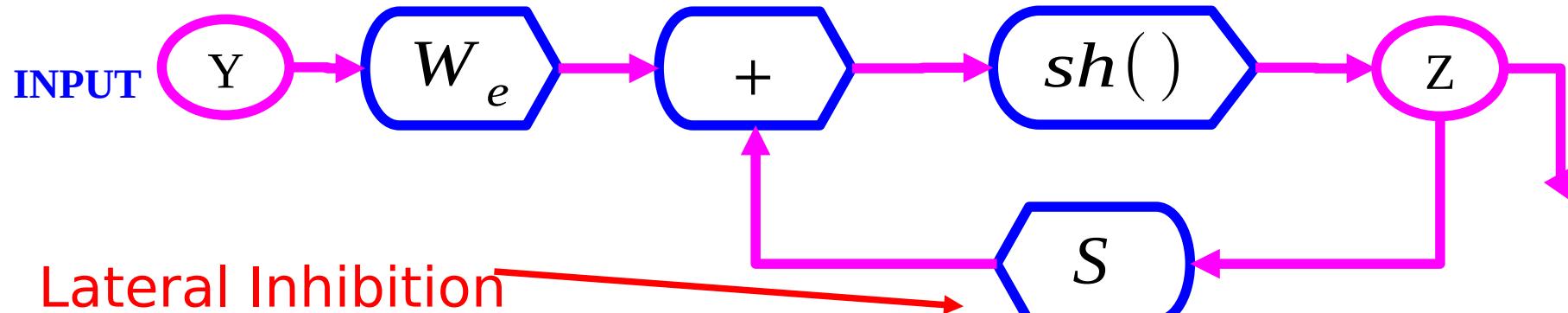
- ▶  $R(z)$  = L1 norm of  $z$
- ▶  $Dec(z, h)$  gain must be bounded (clipped weights)
- ▶ Sparse Auto-Encoder
- ▶ LISTA [Gregor ICML 2010]



# Giving 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], [Rofe & LeCun ICLR 2013]

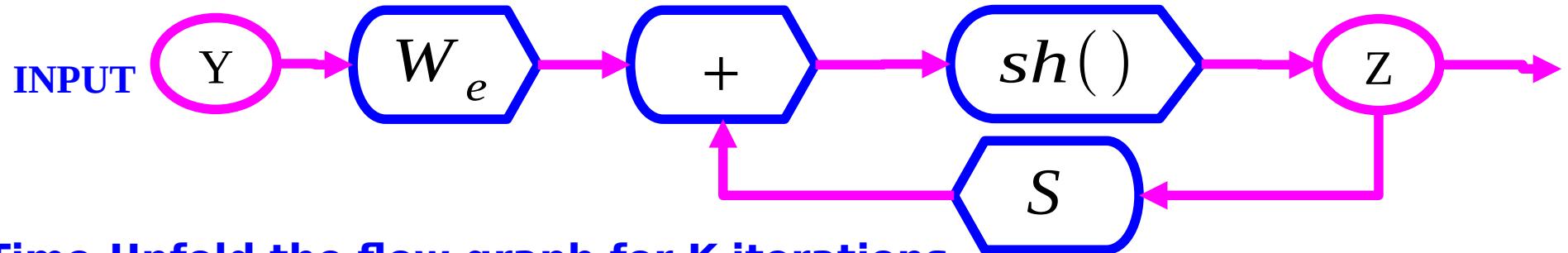


$$Z(t+1) = \text{Shrinkage}_{\lambda/L} \left[ Z(t) - \frac{1}{L} W_d^T (W_d Z(t) - Y) \right]$$

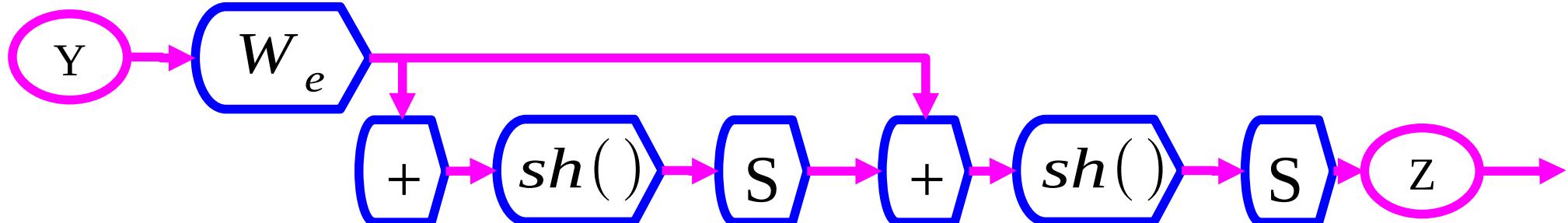
$$Z(t+1) = \text{Shrinkage}_{\lambda/L} [W_e^T Y + S Z(t)]; \quad W_e = \frac{1}{L} W_d; \quad S = I - \frac{1}{L} W_d^T W_d$$

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

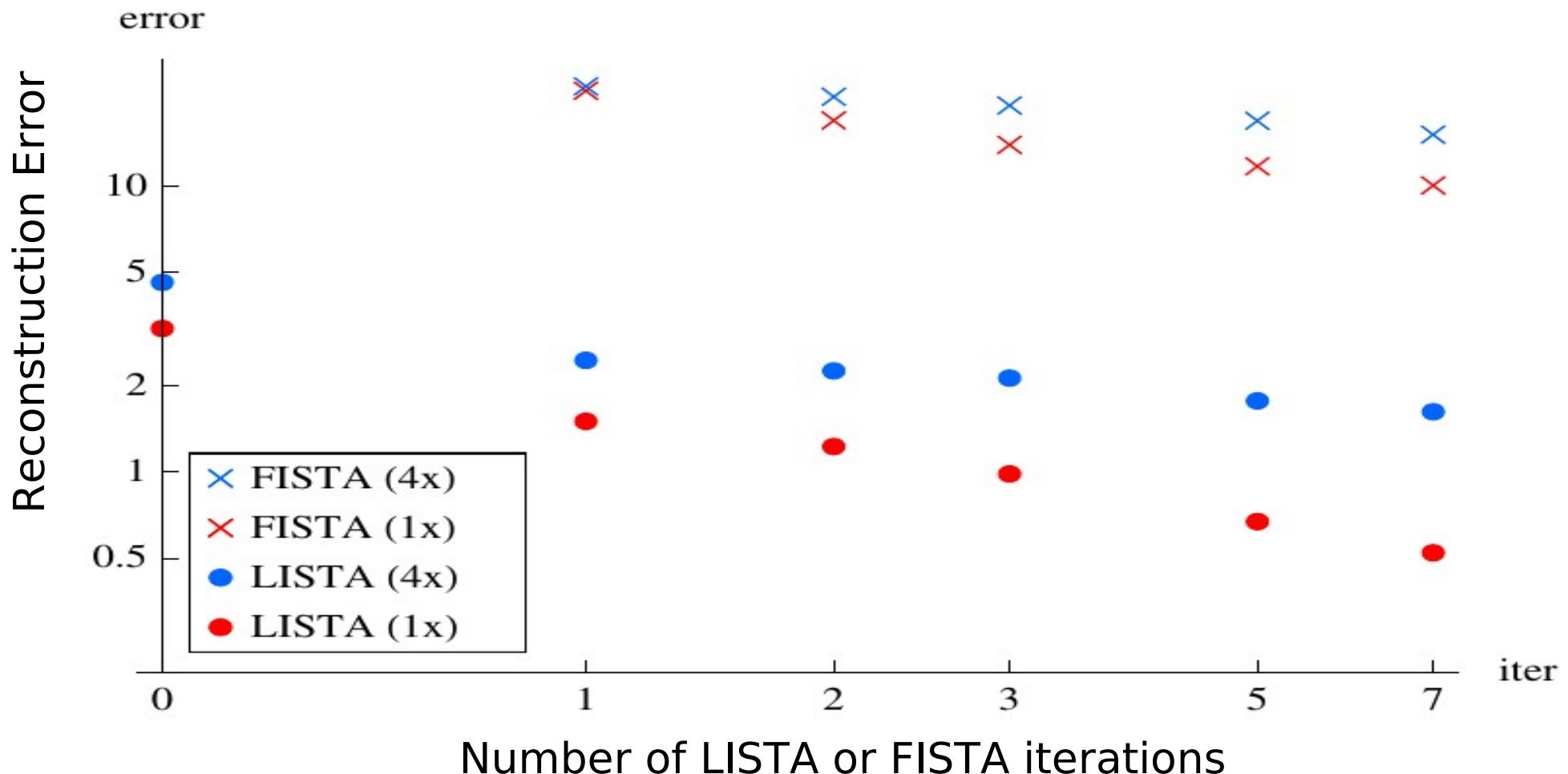
- Think of the FISTA flow graph as a recurrent neural net where  $We$  and  $S$  are trainable parameters



- 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

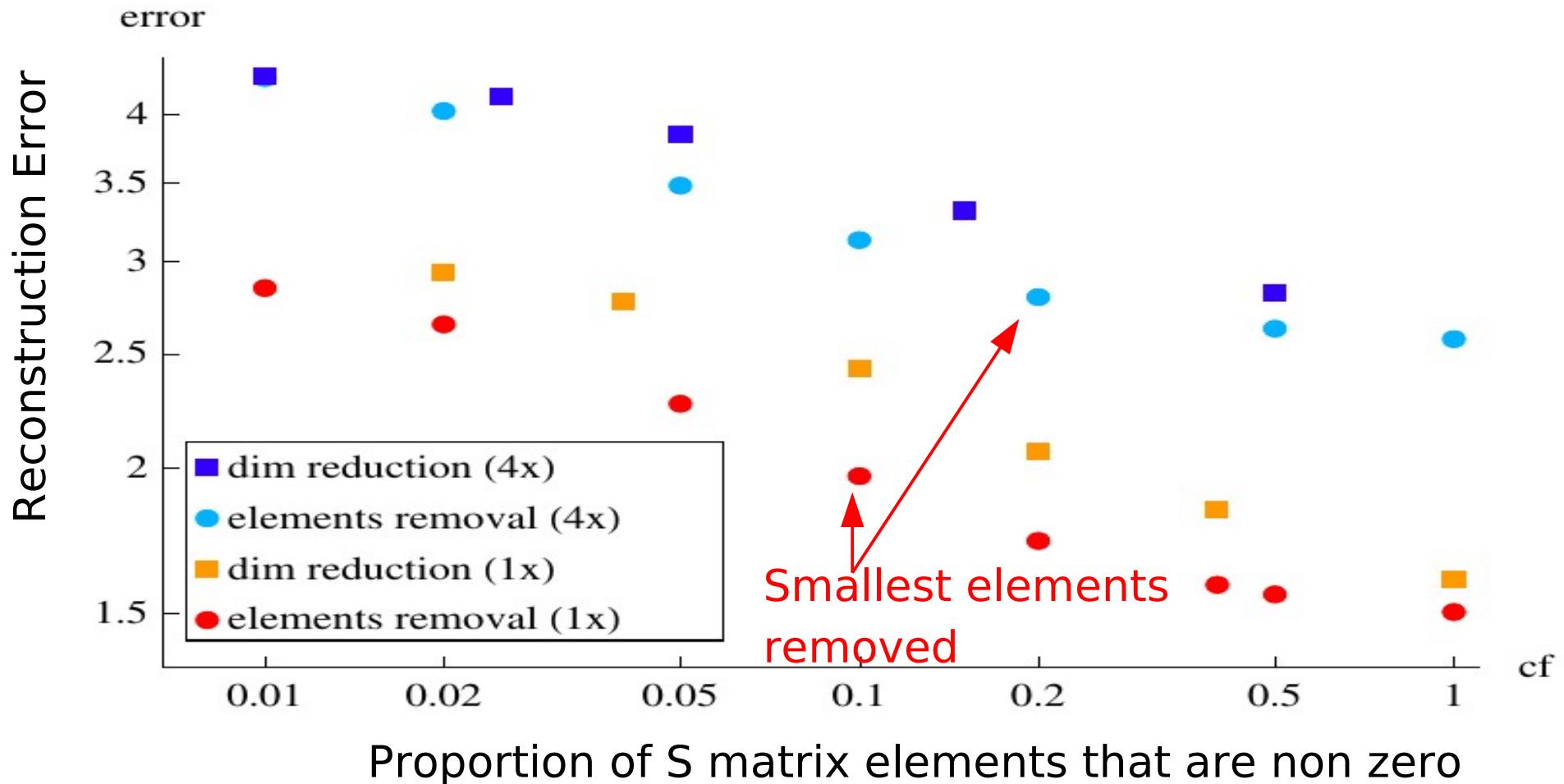


# Learning ISTA (LISTA) vs ISTA/FISTA

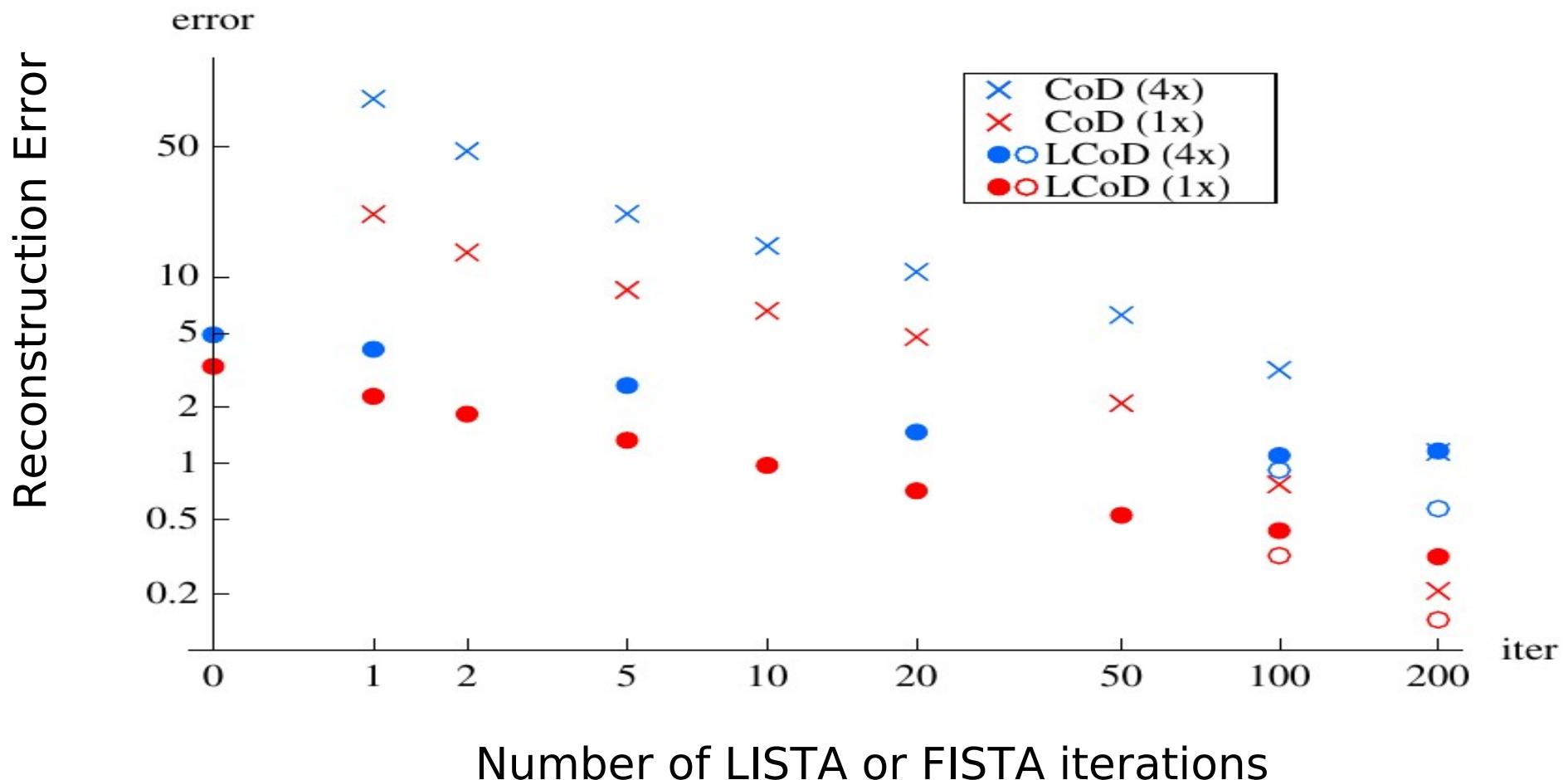


# LISTA with partial mutual inhibition matrix

Y. LeCun

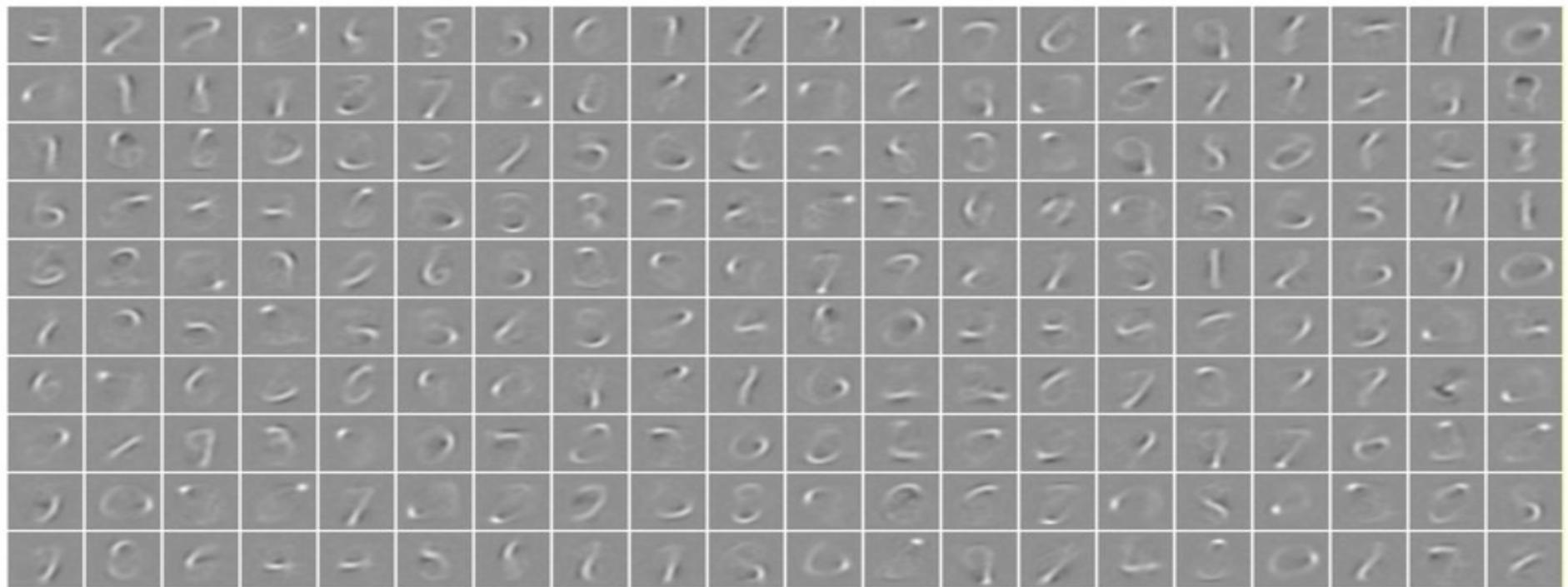


# Learning Coordinate Descent (LcoD): faster than LISTA



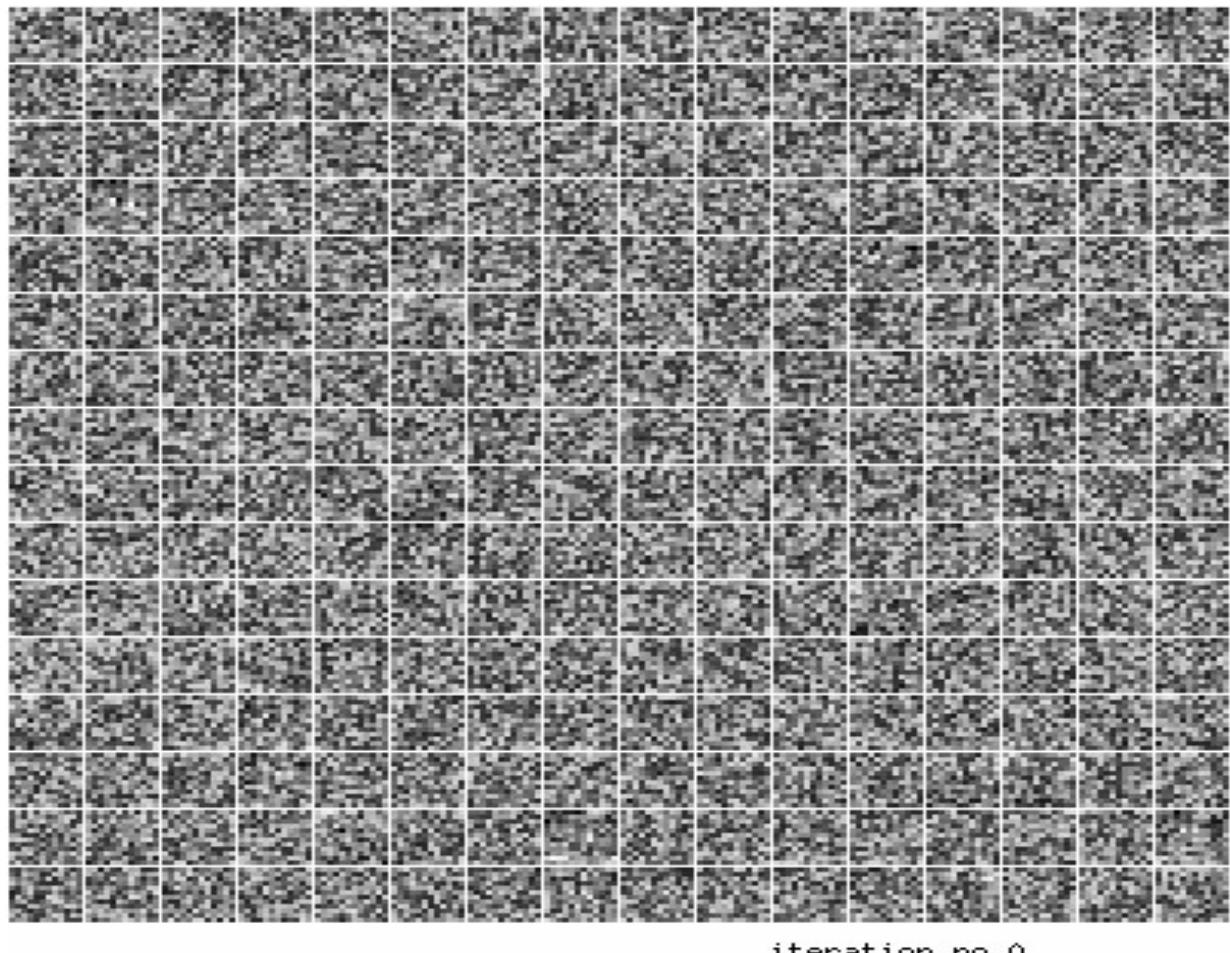
# Sparse AE on handwritten digits (MNIST)

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

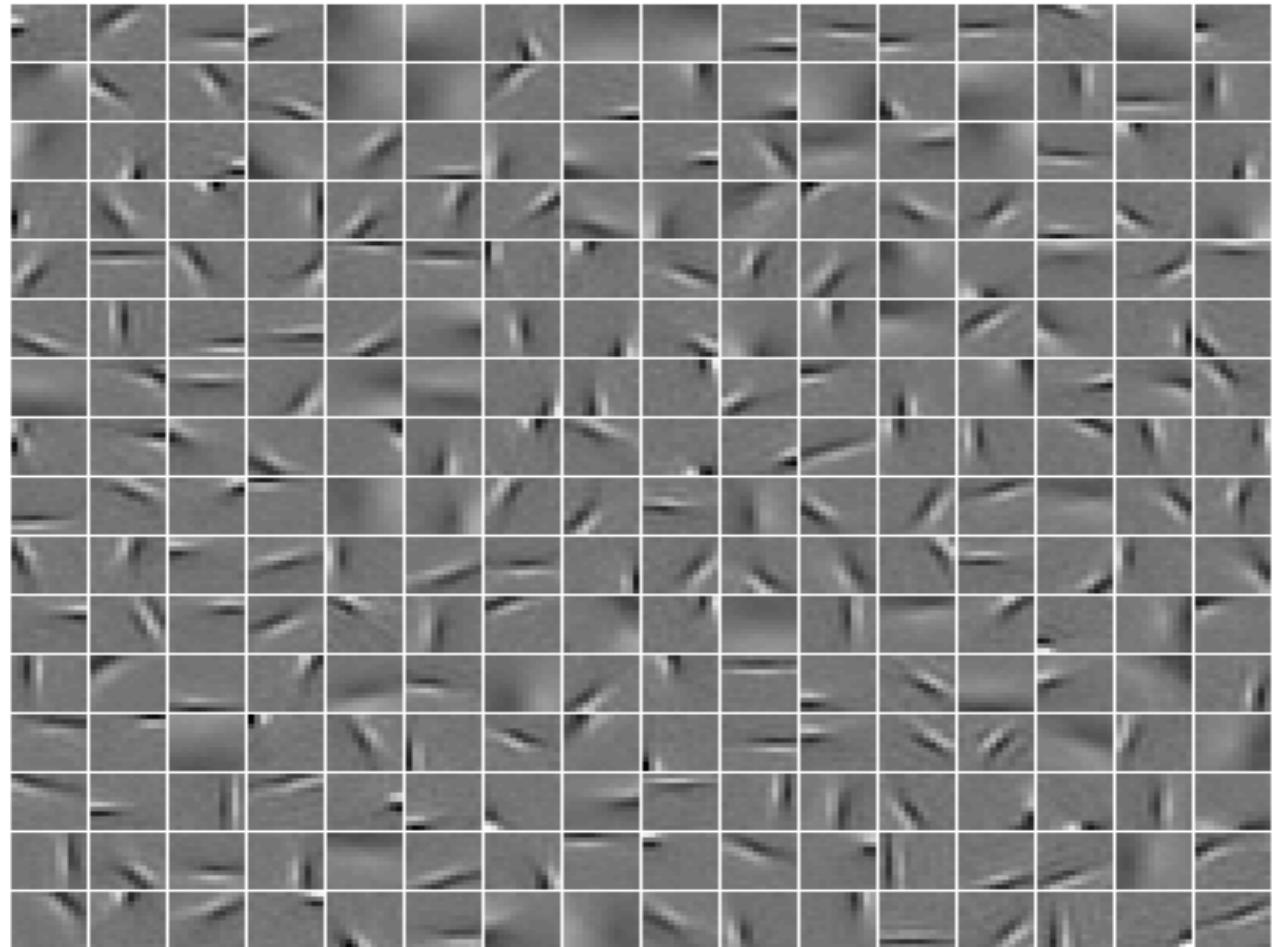


# Predictive Sparse Decomposition (PSD): Training

- ▶ Training on natural images patches.
- ▶ 12X12
- ▶ 256 basis functions
- ▶ [Ranzato 2007]



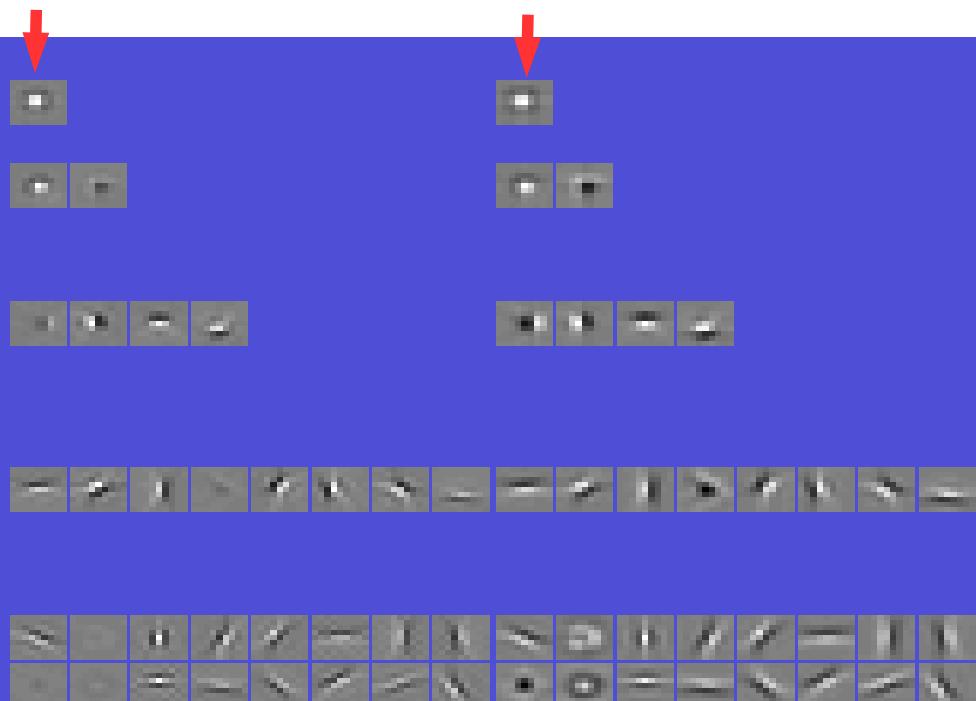
# Learned Features: V1-like receptive fields



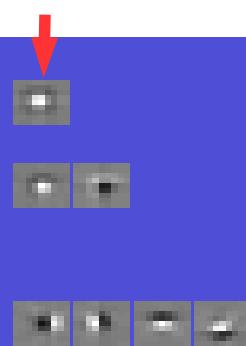
# Convolutional Sparse Auto-Encoder on Natural Images

- ▶ Filters and Basis Functions obtained. Linear decoder (conv)
  - ▶ with 1, 2, 4, 8, 16, 32, and 64 filters [Kavukcuoglu NIPS 2010]

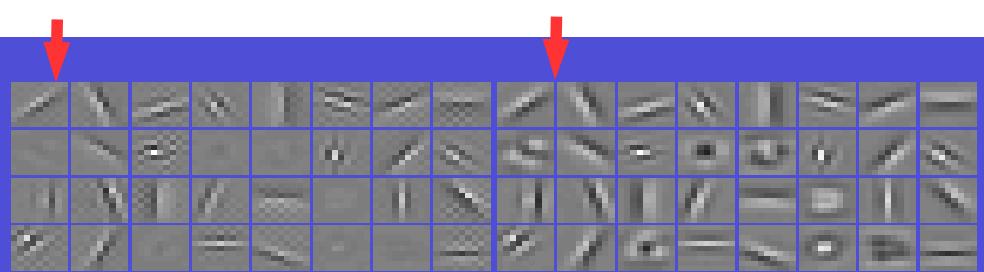
Encoder Filters



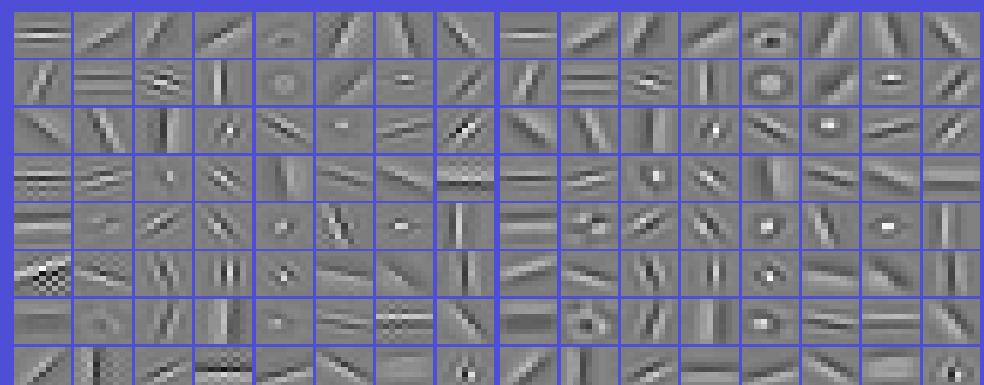
Decoder Filters



Encoder Filters



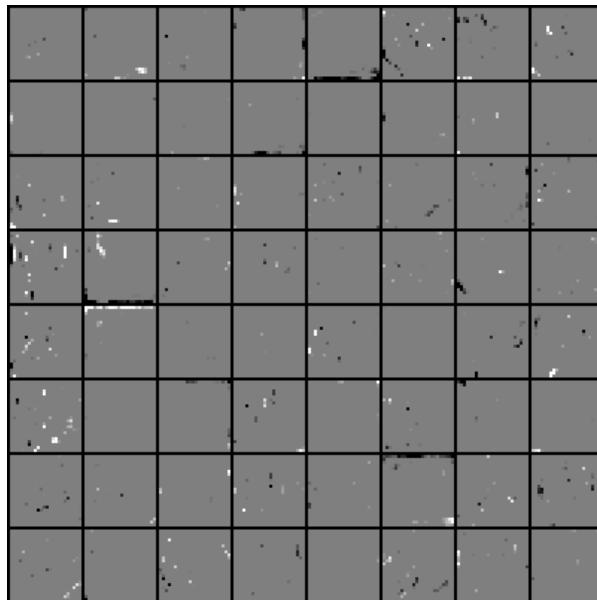
Decoder Filters



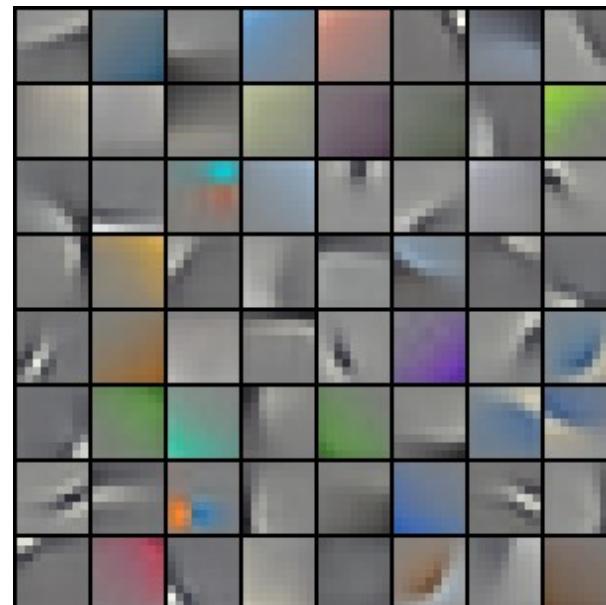
# Convolutional Sparse Auto-Encoder on Natural Images

- ▶ Trained on CIFAR 10 (32x32 color images)
- ▶ Architecture: Linear decoder, LISTA recurrent encoder
- ▶ Pytorch implementation (talk to Jure Zbontar)

sparse codes ( $z$ ) from encoder

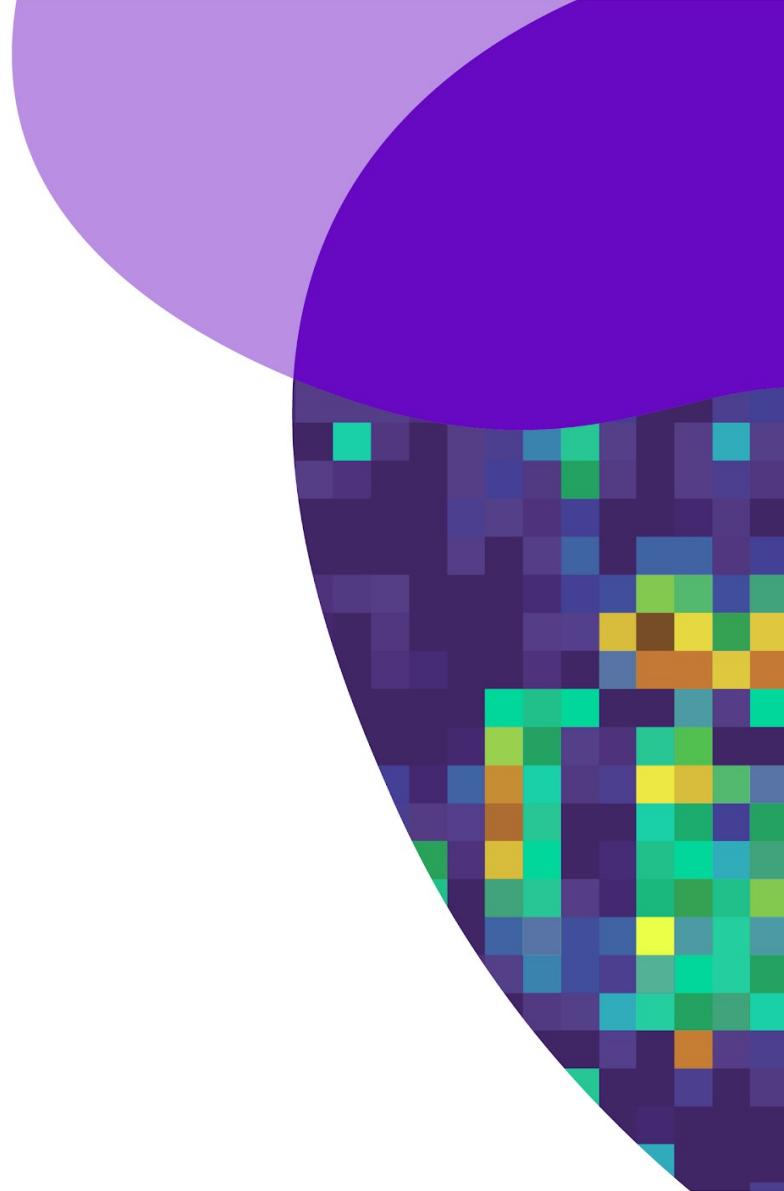


9x9 decoder kernels



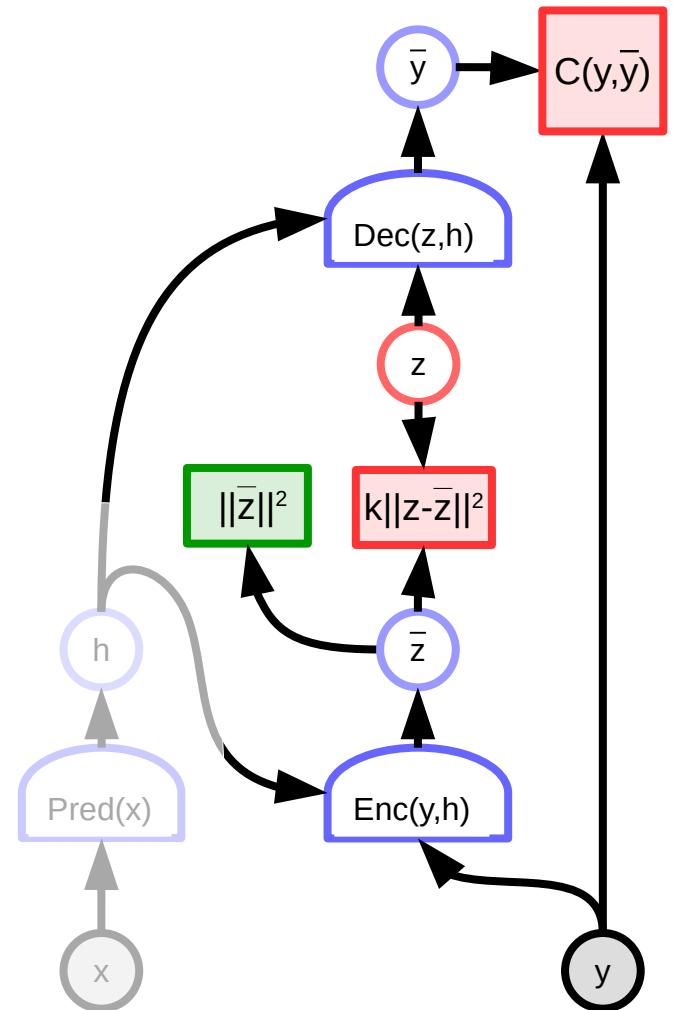
# Variational AE

Architectural SSL methods  
(non contrastive)



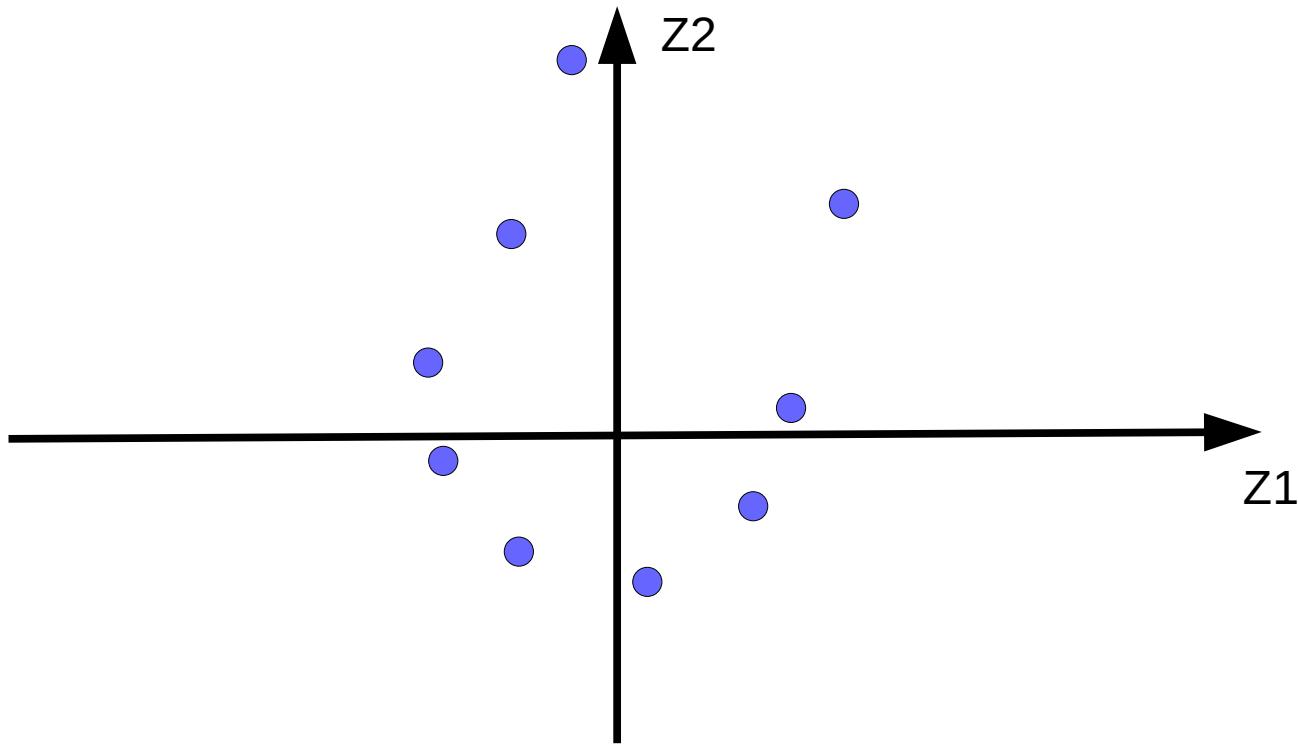
# Variational Auto-Encoder

- ▶ 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.



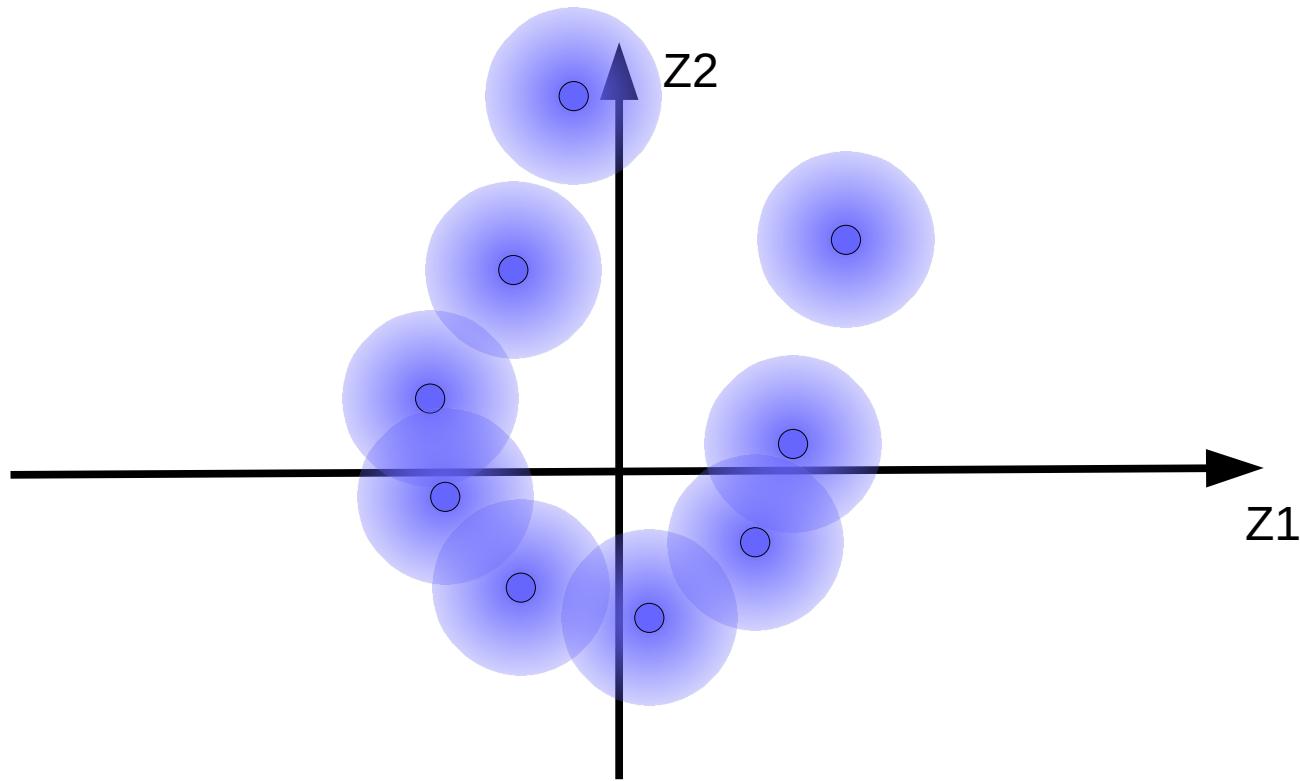
# Variational Auto-Encoder

- ▶ Code vectors for training samples



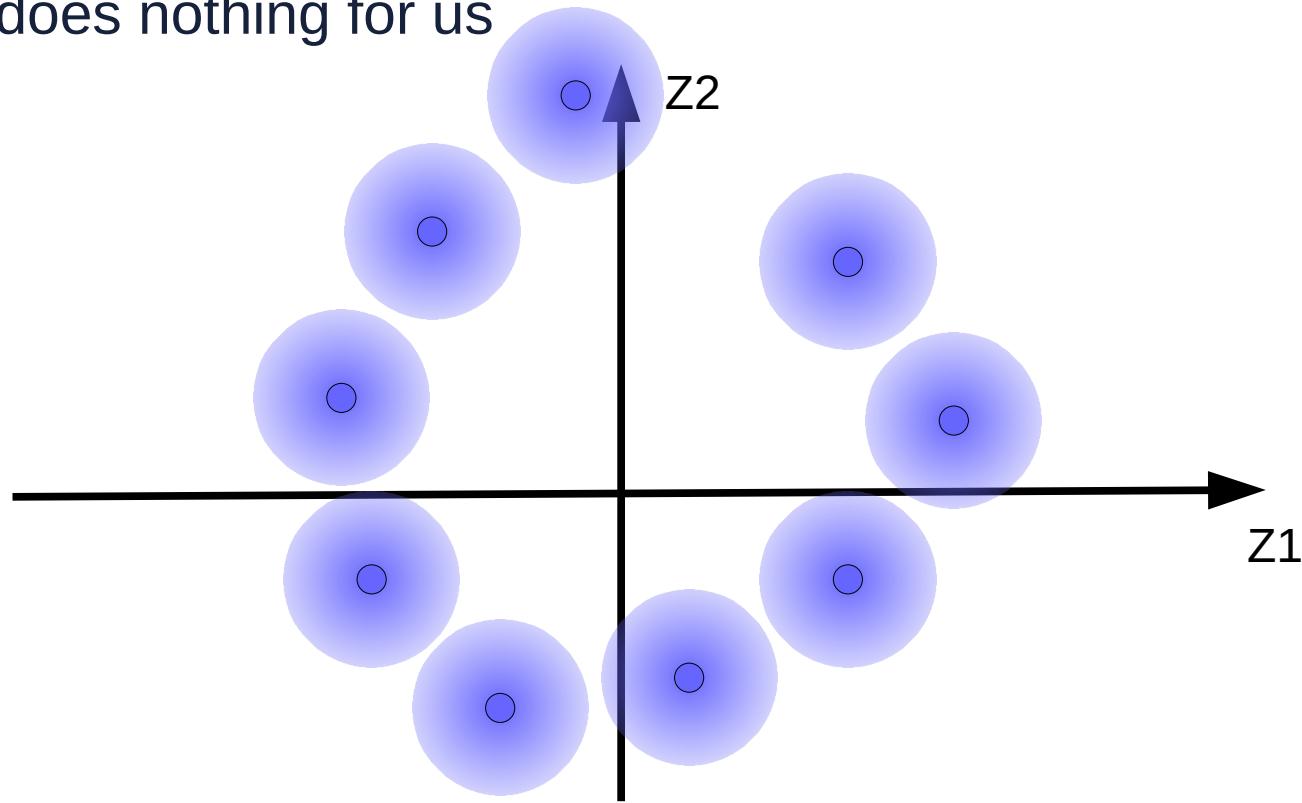
# Variational Auto-Encoder

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



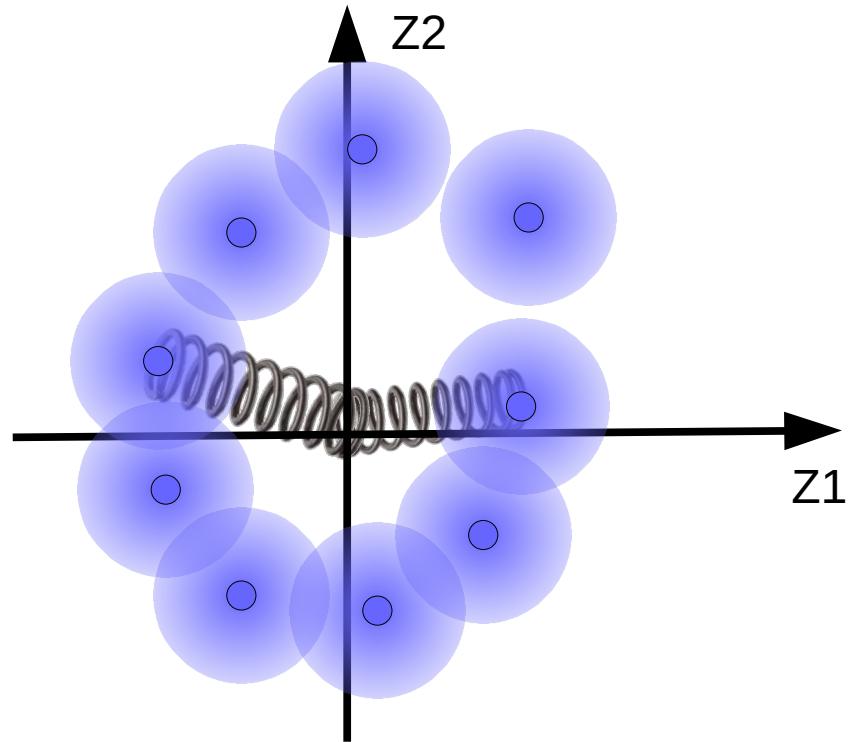
# Variational Auto-Encoder

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



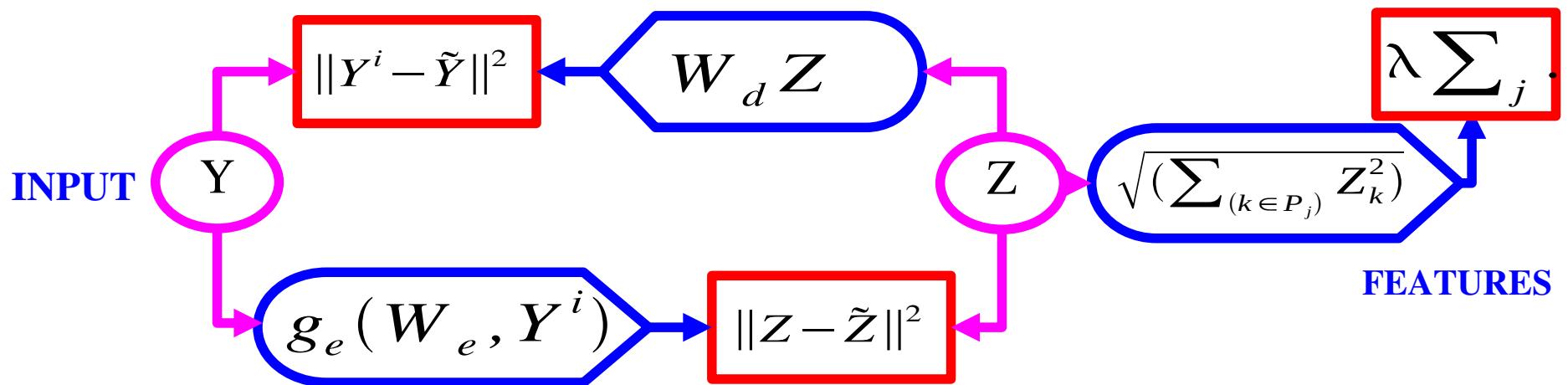
# Variational Auto-Encoder

- ▶ 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



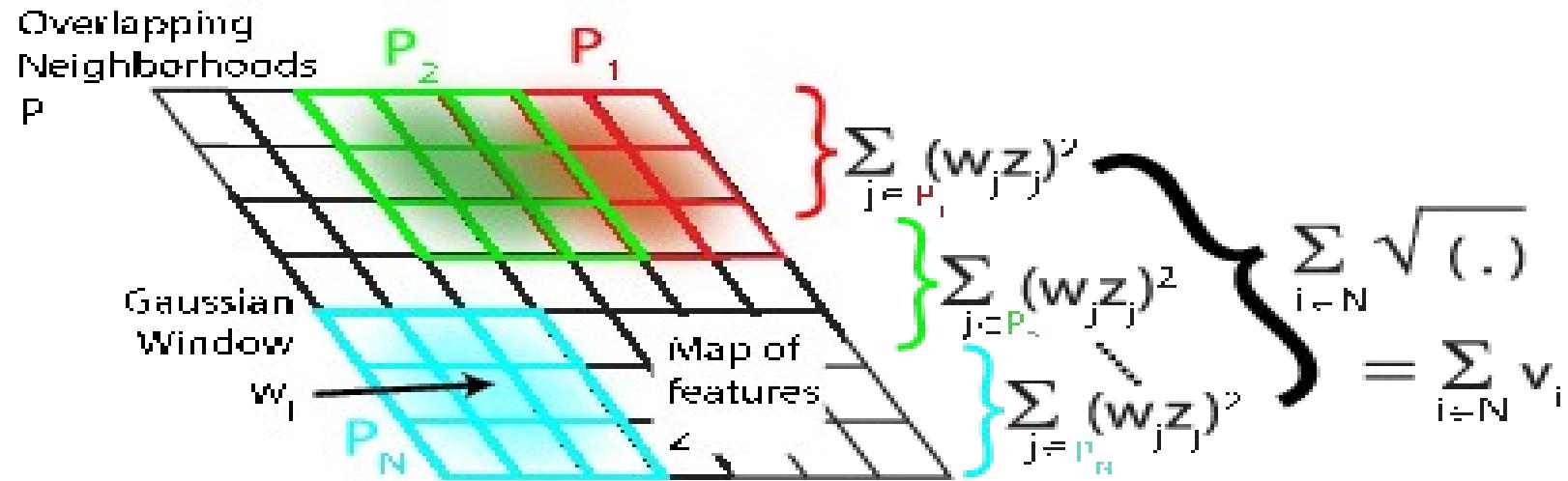
## Group Sparsity [Kavukcuoglu et al. CVPR 2009]

- ▶ 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



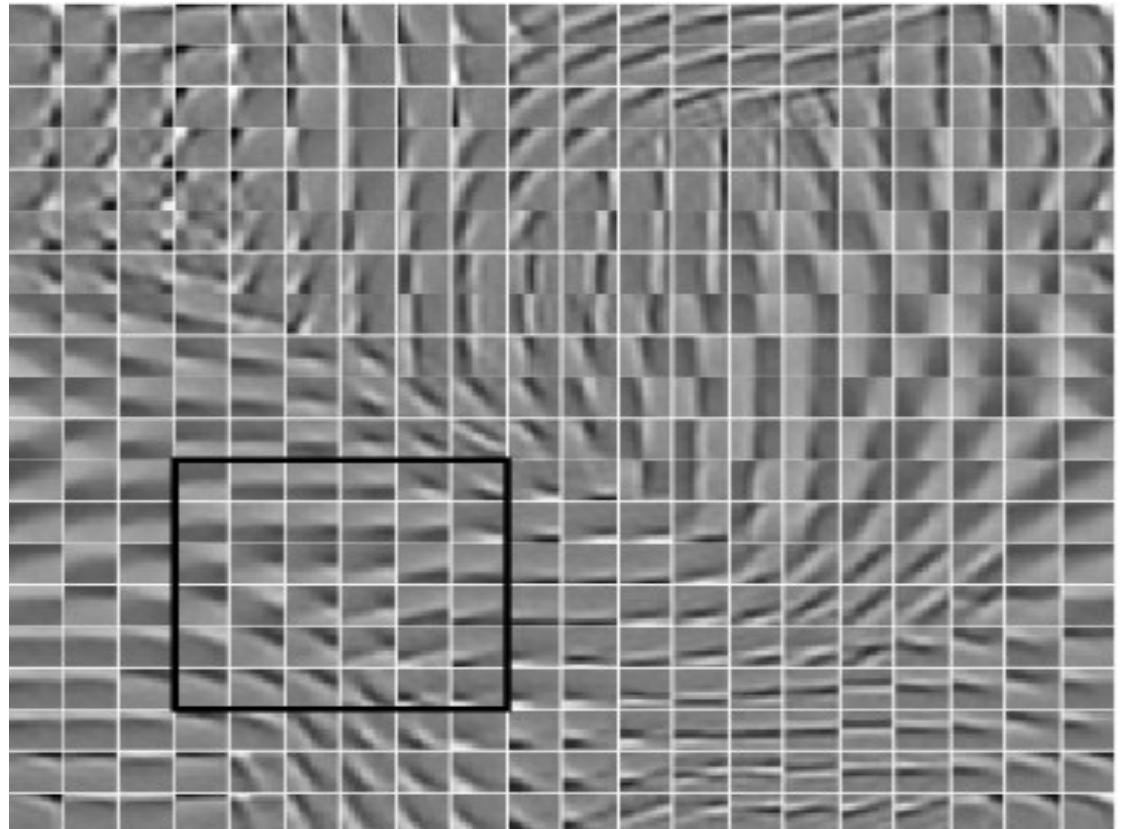
# Why just pool over space? Why not over orientation?

- ▶ Using an idea from Hyvarinen: topographic square pooling (subspace ICA)
- ▶ 1. Apply filters on a patch (with suitable non-linearity)
- ▶ 2. Arrange filter outputs on a 2D plane
- ▶ 3. square filter outputs
- ▶ 4. minimize sqrt of sum of blocks of squared filter outputs



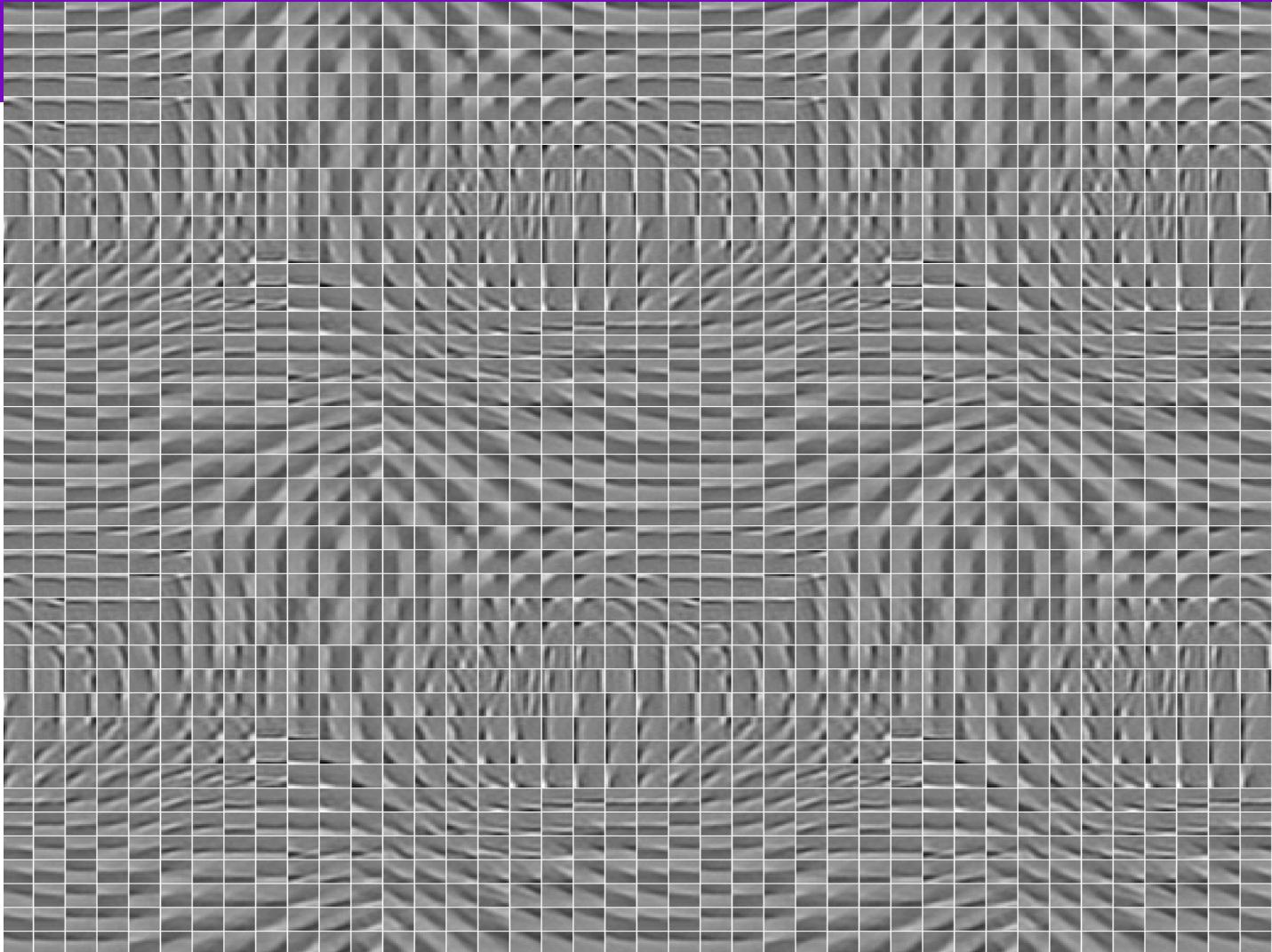
# Why just pool over space? Why not over orientation?

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



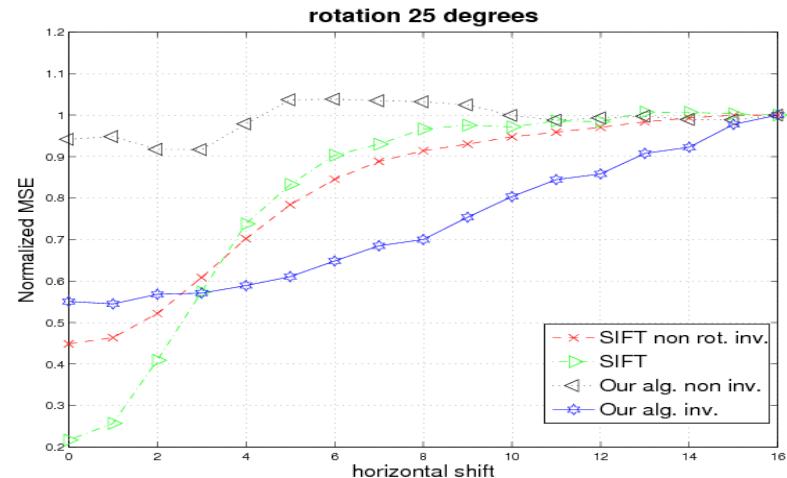
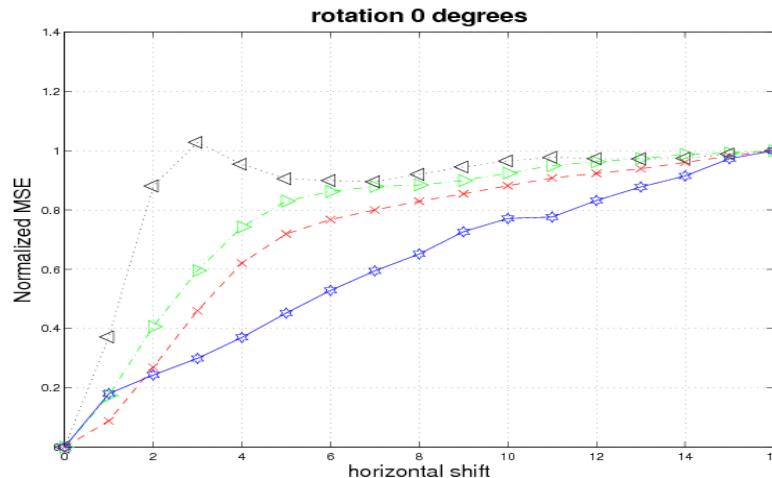
# Pinwheels!

- ▶ Does that look pinwheely to you?



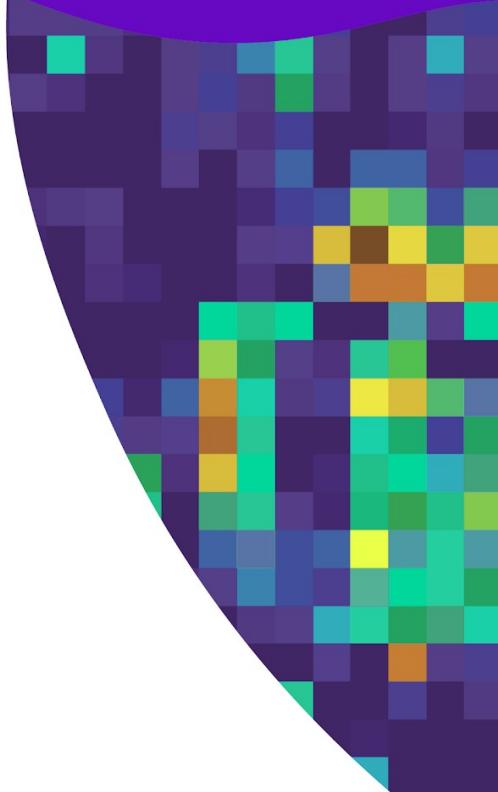
# Invariance Properties Compared to SIFT

- ▶ Measure distance between feature vectors (128 dimensions) of 16x16 patches from natural images
  - ▶ Left: normalized distance as a function of translation
  - ▶ Right: normalized distance as a function of translation when one patch is rotated 25 degrees.



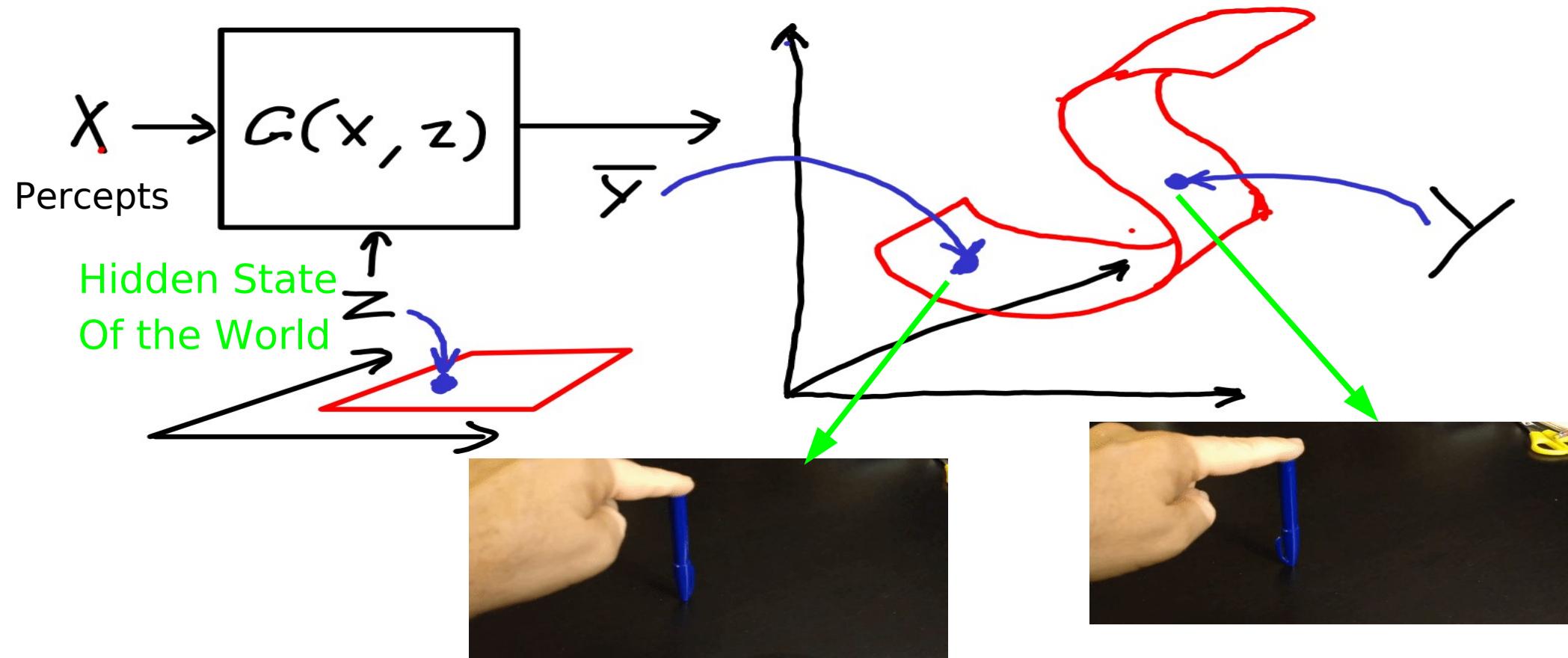
# Generative Adversarial Networks

Contrastive method in which a generator network produces the contrastive samples



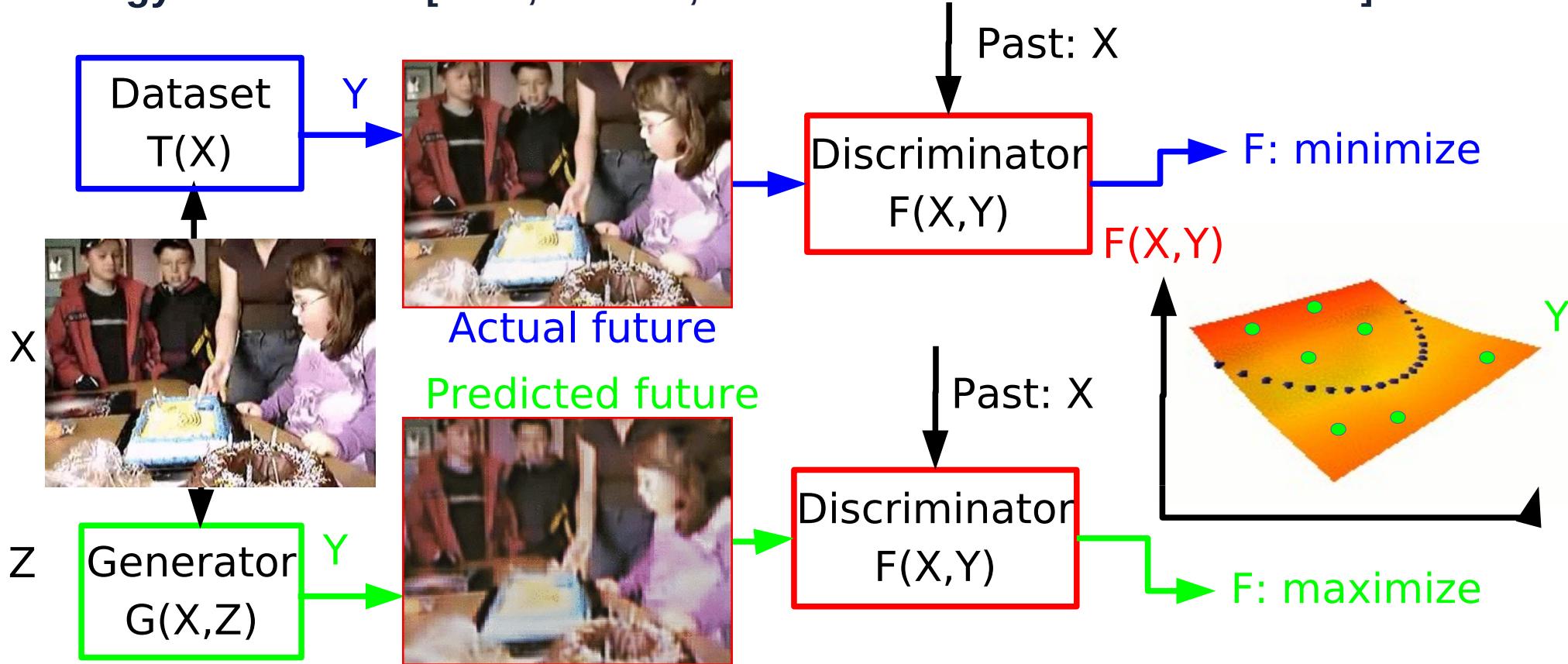
# The Hard Part: Prediction Under Uncertainty

- Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).



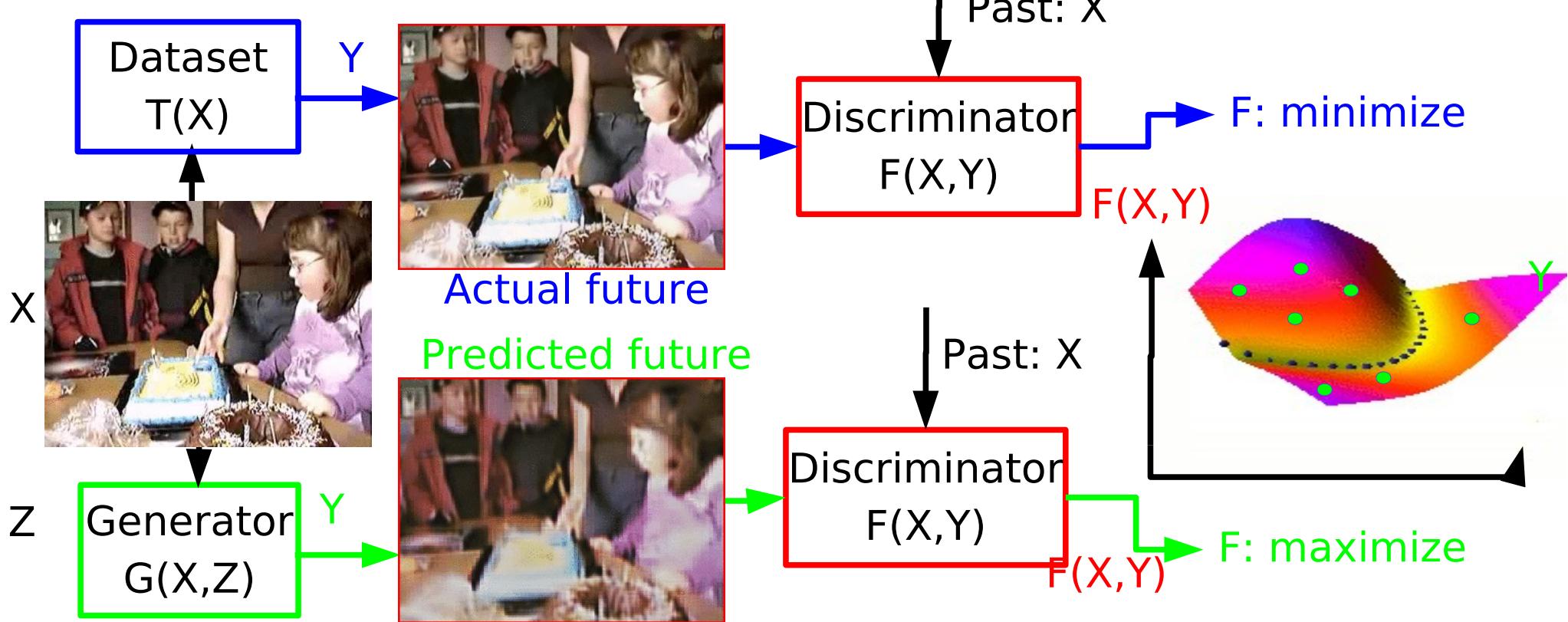
# Adversarial Training: the key to prediction under uncertainty?

- ▶ Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- ▶ Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]



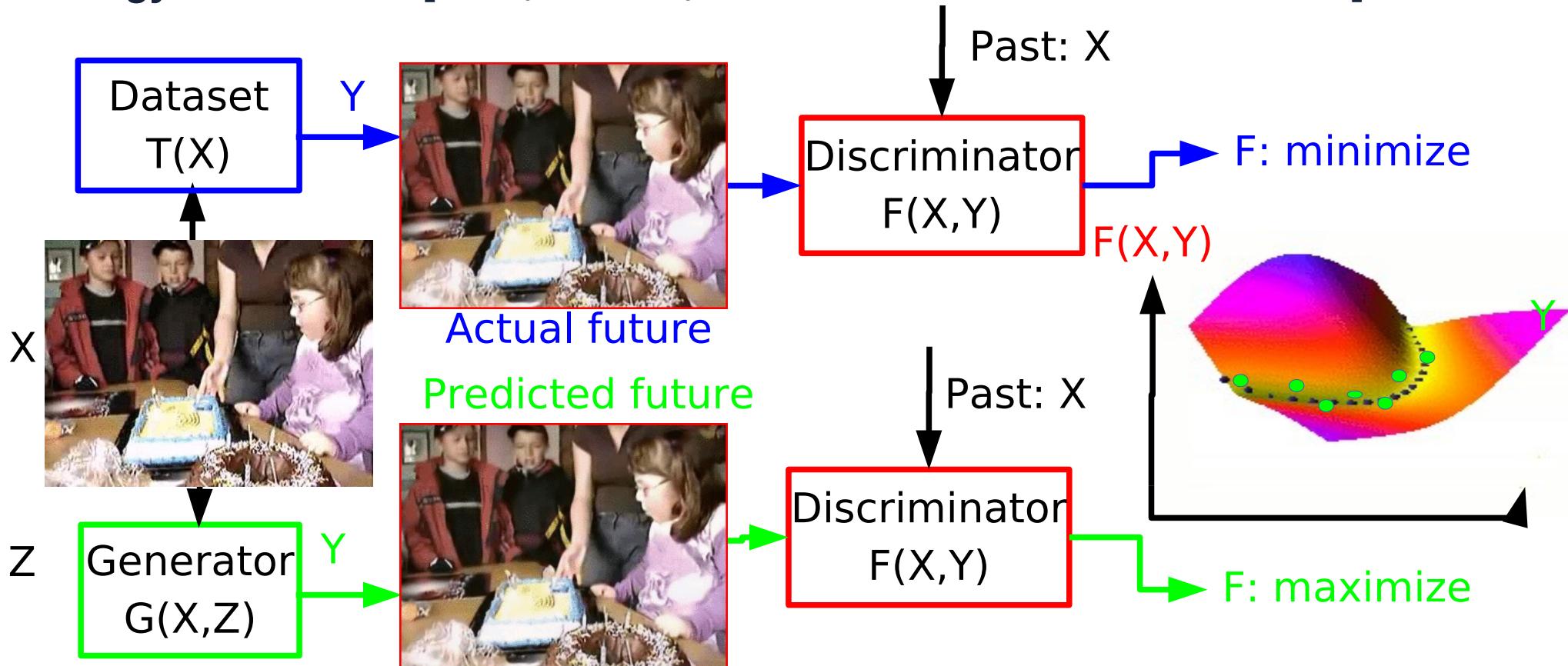
# Adversarial Training: the key to prediction under uncertainty?

- ▶ Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- ▶ Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]



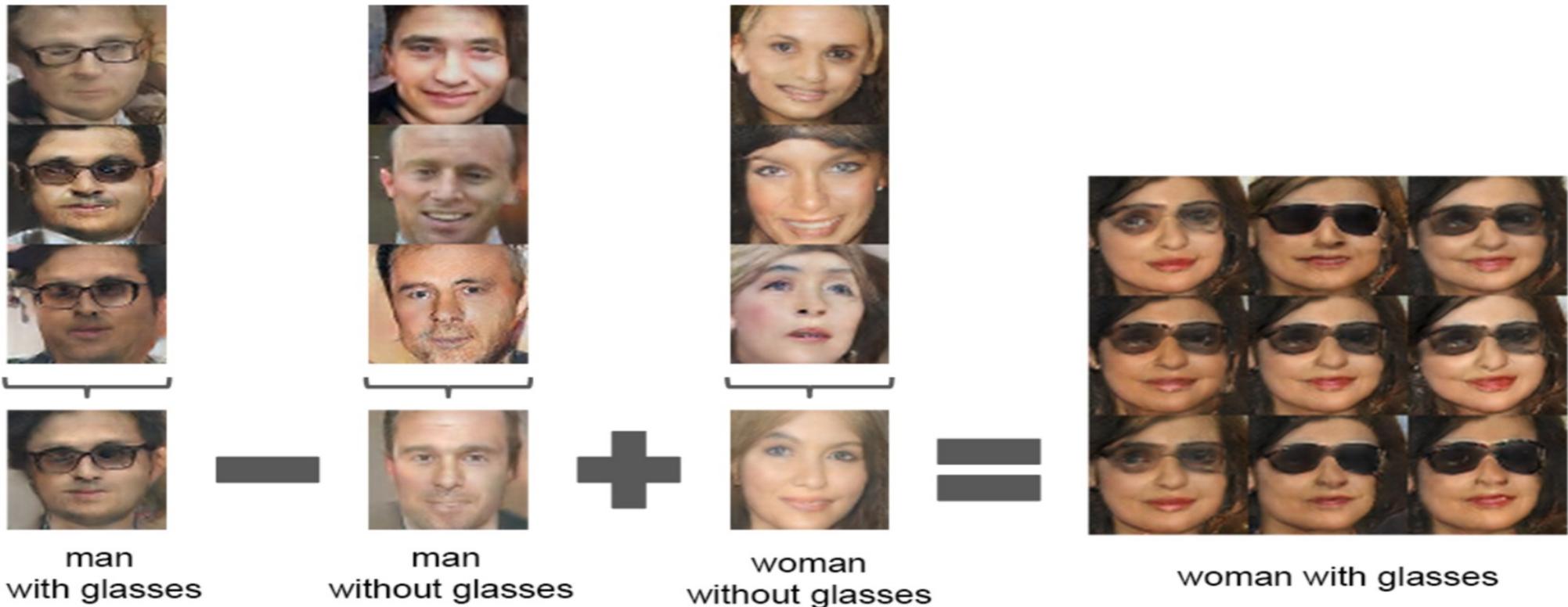
# Adversarial Training: the key to prediction under uncertainty?

- ▶ Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- ▶ Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]



# Face Algebra (in DCGAN space)

- ▶ DCGAN: adversarial training to generate images.
- ▶ [Radford, Metz, Chintala 2015]



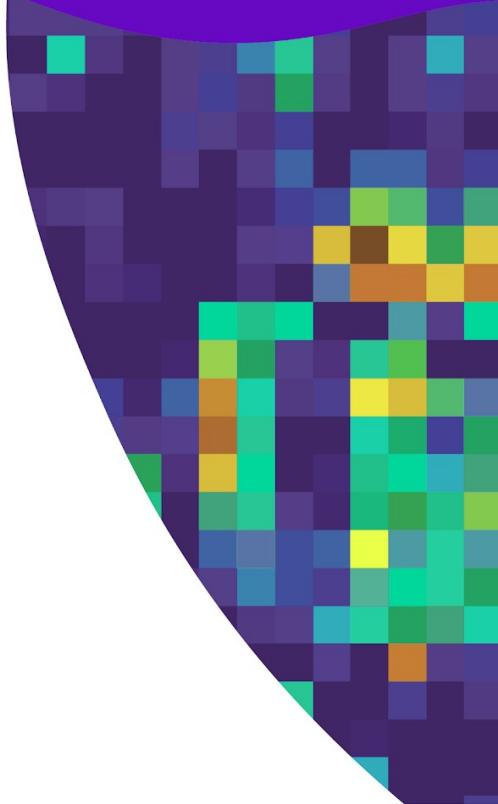
# Faces “invented” by a GAN (Generative Adversarial Network)

- ▶ Random vector → Generator Network → output image [Goodfellow NIPS 2014]  
[Karras et al. ICLR 2018] (from NVIDIA)

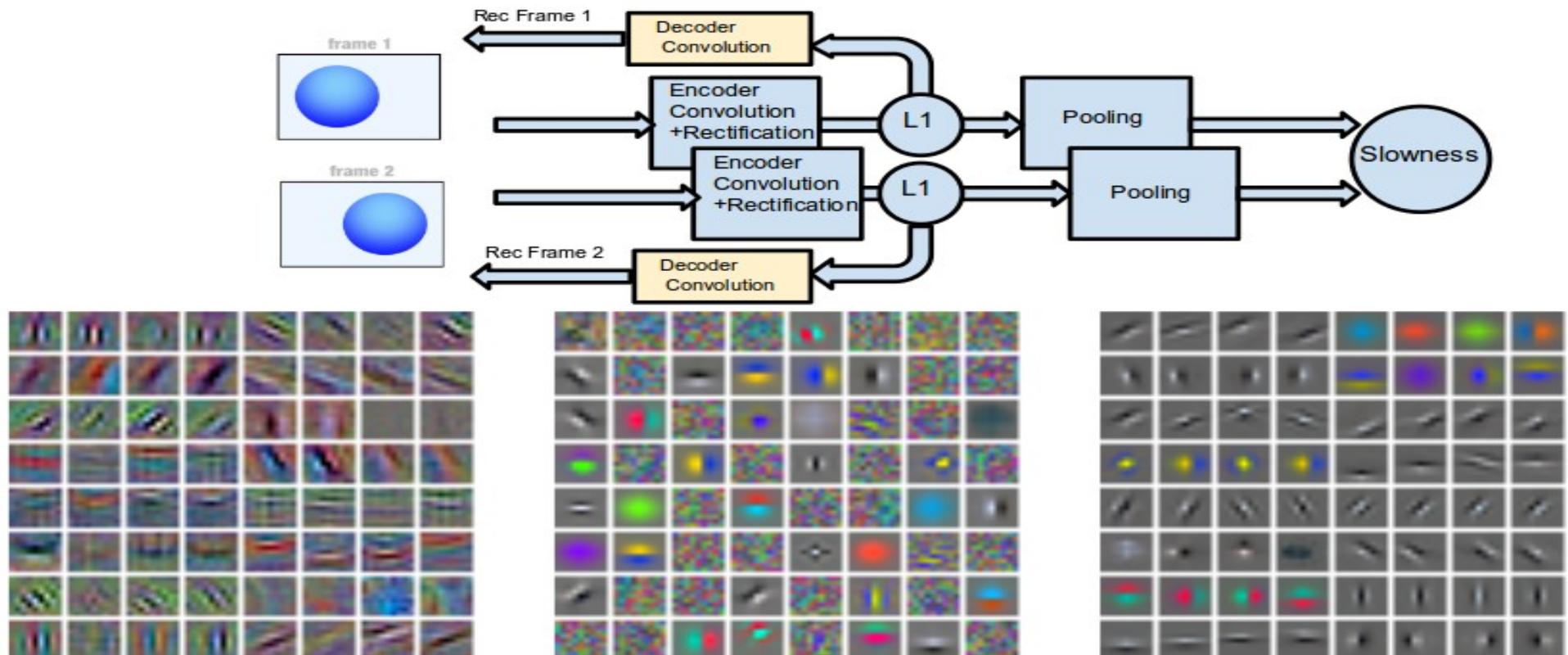


# Other Latent representation regularization

Using temporal constancy



# Sparse Auto-Encoder with “Slow Feature” Penalty



► Supervised filters CIFAR10

► [Goroshin et al. ICCV 2015, Arxiv:1412.6056]

sparse conv. auto-encoder

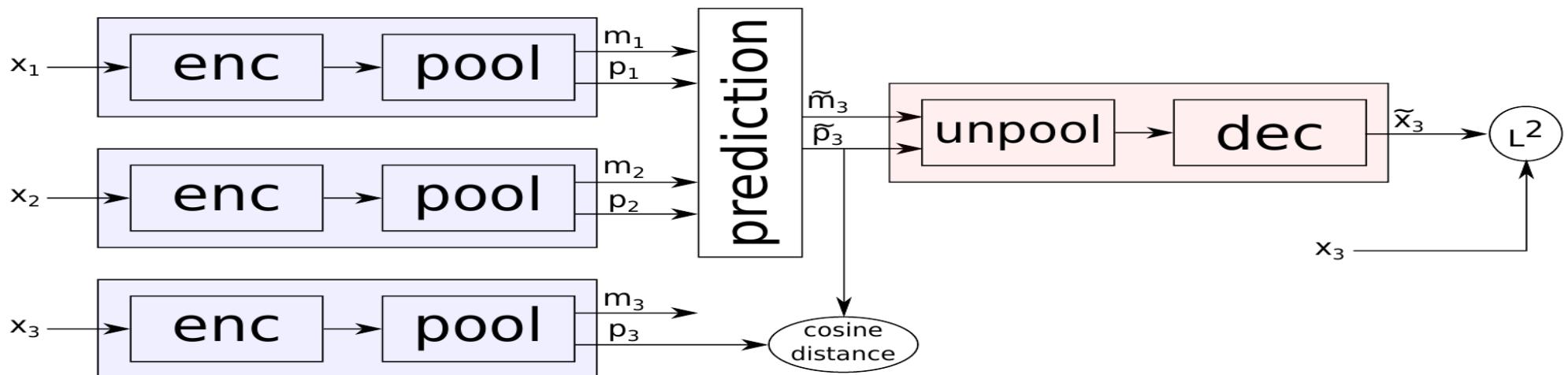
slow & sparse conv. auto-encoder  
Trained on YouTube videos

# Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- Trained on 3 successive frames of video
- Maximize the colinearity between successive changes of feature vectors
- So that “natural” changes stay in linear manifold in feature space.

$$L = \frac{1}{2} \|G_W(\mathbf{a} [z^t \ z^{t-1}]^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\|z^t - z^{t-1}\| \|z^{t+1} - z^t\|}$$



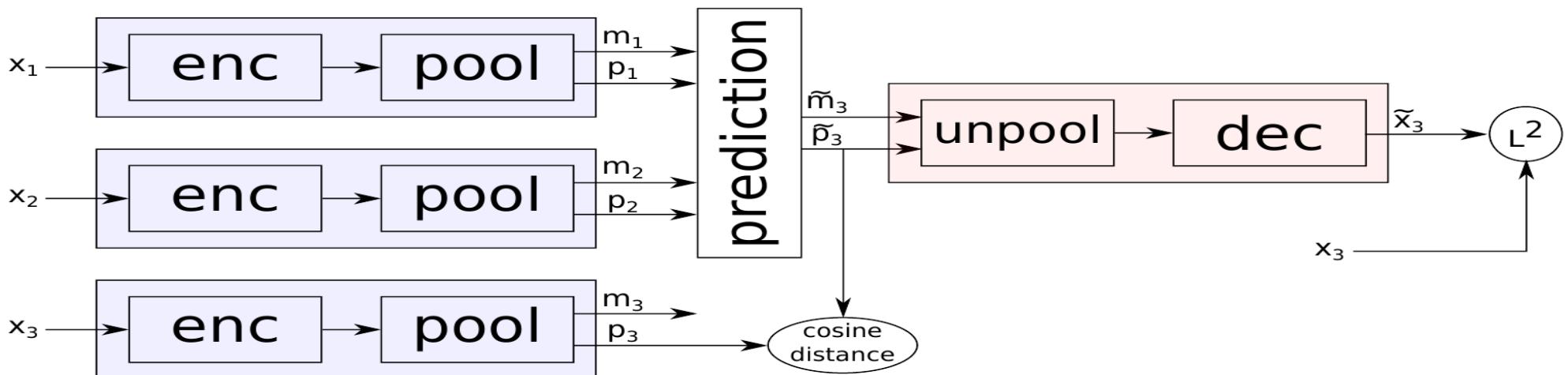
# Unsupervised Learning by Prediction and Linearization

$$L = \frac{1}{2} \|G_W(\mathbf{a} [z^t \ z^{t-1}]^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\|z^t - z^{t-1}\| \|z^{t+1} - z^t\|}$$

- **Magnitude**
  - (soft max)
- **Phase**
  - (soft argmax)

$$m_k = \sum_{N_k} z(f, x, y) \frac{e^{\beta z(f, x, y)}}{\sum_{N_k} e^{\beta z(f', x', y')}} \approx \max_{N_k} z(f, x, y)$$

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f, x, y)}}{\sum_{N_k} e^{\beta z(f', x', y')}} \approx \arg \max_{N_k} z(f, x, y)$$



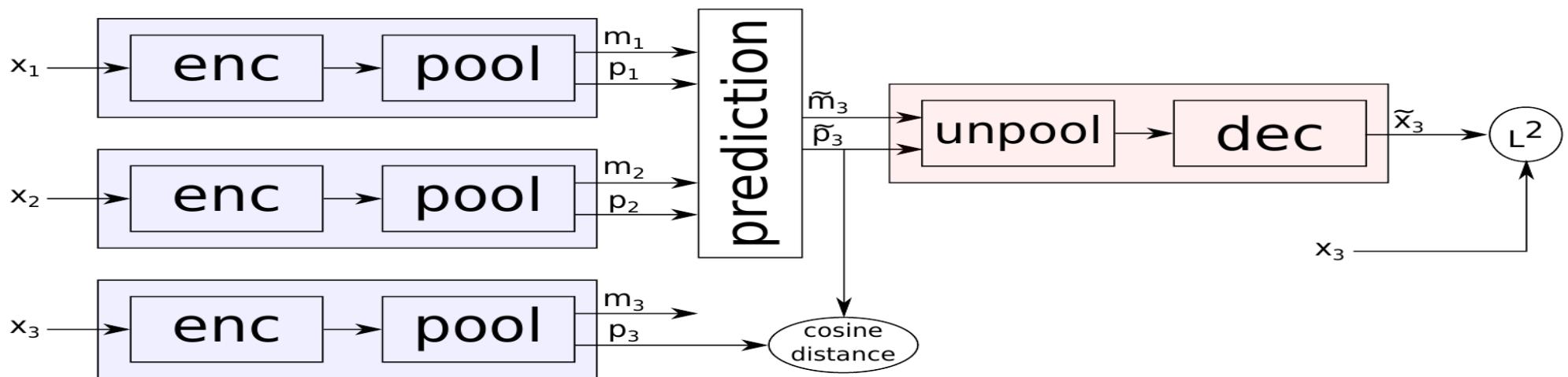
# Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- **Version 2:**

- Make sure the “what”/magnitude changes slowly
- Make sure the “where”/phase changes linearly

$$\begin{aligned} m^{t+1} &= \frac{m^t + m^{t-1}}{2} \\ \mathbf{p}^{t+1} &= 2\mathbf{p}^t - \mathbf{p}^{t-1} \end{aligned}$$



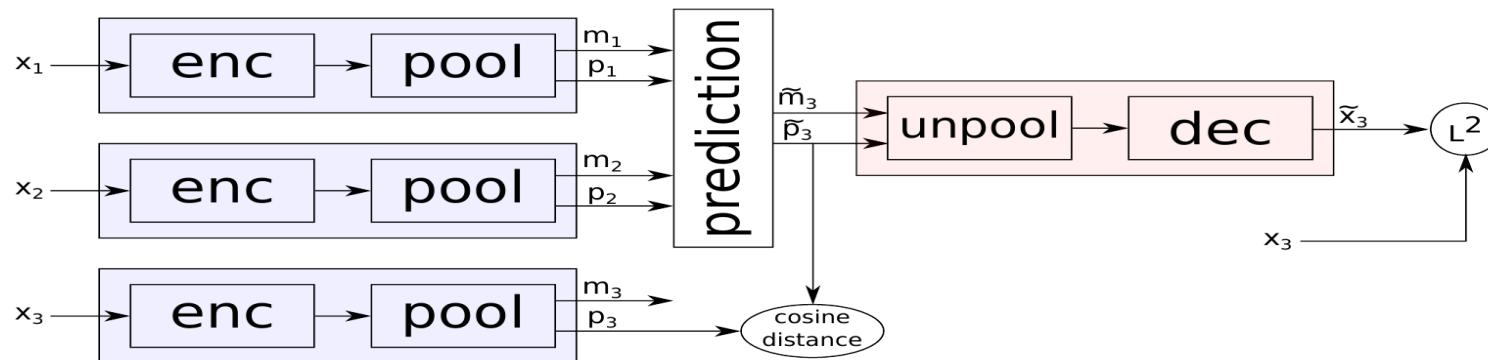
# Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun arXiv:1506.03011]

- **Version 3: the world is unpredictable**
  - Add latent variable to compensate for that.
  - Minimize over them during training

$$\hat{z}_\delta^{t+1} = z^t + (W_1 \delta) \odot \mathbf{a} [z^t \quad z^{t-1}]^T$$

$$L = \min_{\delta} \|G_W(\hat{z}_\delta^{t+1}) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\|z^t - z^{t-1}\| \|z^{t+1} - z^t\|}$$



# Algorithm (with latent variables)

---

**Algorithm 1** Minibatch stochastic gradient descent training for prediction with uncertainty. The number of  $\delta$ -gradient descent steps ( $k$ ) is treated as a hyper-parameter.

---

```

for number of training epochs do
    Sample a mini-batch of temporal triplets  $\{x^{t-1}, x^t, x^{t+1}\}$ 
    Set  $\delta_0 = 0$ 
    Forward propagate  $x^{t-1}, x^t$  through the network and obtain the codes  $z^{t-1}, z^t$  and the prediction  $\hat{x}_0^{t+1}$ 
    for  $i = 1$  to  $k$  do
        Compute the  $L^2$  prediction error
        Back propagate the error through the decoder to compute the gradient  $\frac{\partial L}{\partial \delta^{i-1}}$ 
        Update  $\delta_i = \delta_{i-1} - \alpha \frac{\partial L}{\partial \delta^{i-1}}$ 
        Compute  $\hat{z}_{\delta_i}^{t+1} = z^t + (W_1 \delta_i) \odot \mathbf{a} [z^t \quad z^{t-1}]^T$ 
        Compute  $\hat{x}_i^{t+1} = G_W(z_{\delta_i}^{t+1})$ 
    end for
    Back propagate the full encoder/predictor loss from Equation 7 using  $\delta_k$ , and update the weight matrices  $W$  and  $W_1$ 
end for

```

---

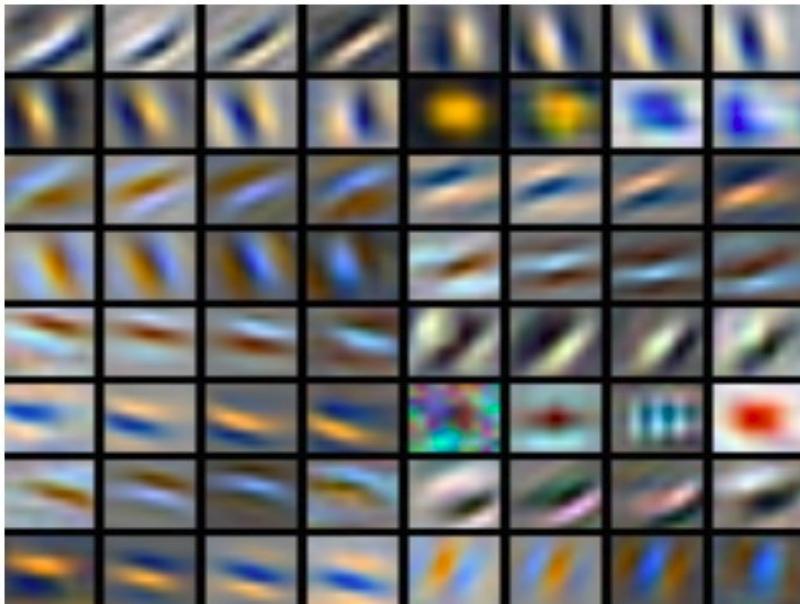
$$\hat{z}_{\delta}^{t+1} = z^t + (W_1 \delta) \odot \mathbf{a} [z^t \quad z^{t-1}]^T$$

$$L = \min_{\delta} \|G_W(\hat{z}_{\delta}^{t+1}) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\|z^t - z^{t-1}\| \|z^{t+1} - z^t\|}$$

# Architectures

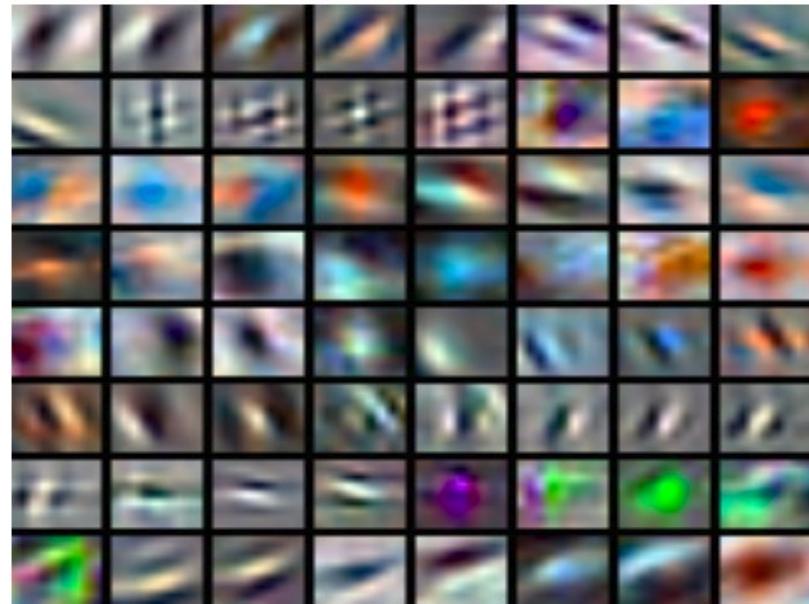
	Encoder	Prediction	Decoder
Shallow Architecture 1	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4	Average Mag. Linear Extrapolation Phase	Conv $64 \times 9 \times 9$
Shallow Architecture 2	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4 stride 2	Average Mag. Linear Extrapolation Phase	Conv $64 \times 9 \times 9$
Deep Architecture 1	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU $8192 \times 4096$	None	FC+ReLU $\mathbf{8192} \times 8192$ Reshape $32 \times 16 \times 16$ SpatialPadding $8 \times 8$ Conv+ReLU $16 \times 9 \times 9$ SpatialPadding $8 \times 8$ Conv $1 \times 9 \times 9$
Deep Architecture 2	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU $8192 \times 4096$	Linear Extrapolation	FC+ReLU $4096 \times 8192$ Reshape $32 \times 16 \times 16$ SpatialPadding $8 \times 8$ Conv+ReLU $16 \times 9 \times 9$ SpatialPadding $8 \times 8$ Conv $1 \times 9 \times 9$
Deep Architecture 3	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU $8192 \times 4096$ Reshape $64 \times 8 \times 8$ Phase Pool $8 \times 8$	Average Mag. Linear Extrapolation Phase	Unpool $8 \times 8$ FC+ReLU $4096 \times 8192$ Reshape $32 \times 16 \times 16$ SpatialPadding $8 \times 8$ Conv+ReLU $16 \times 9 \times 9$ SpatialPadding $8 \times 8$ Conv $1 \times 9 \times 9$

# Filters: pooling over groups of 4 filters.



(a) Shallow Architecture 1

Non-overlapping pooling  
4x4x4 (feat,x,y)

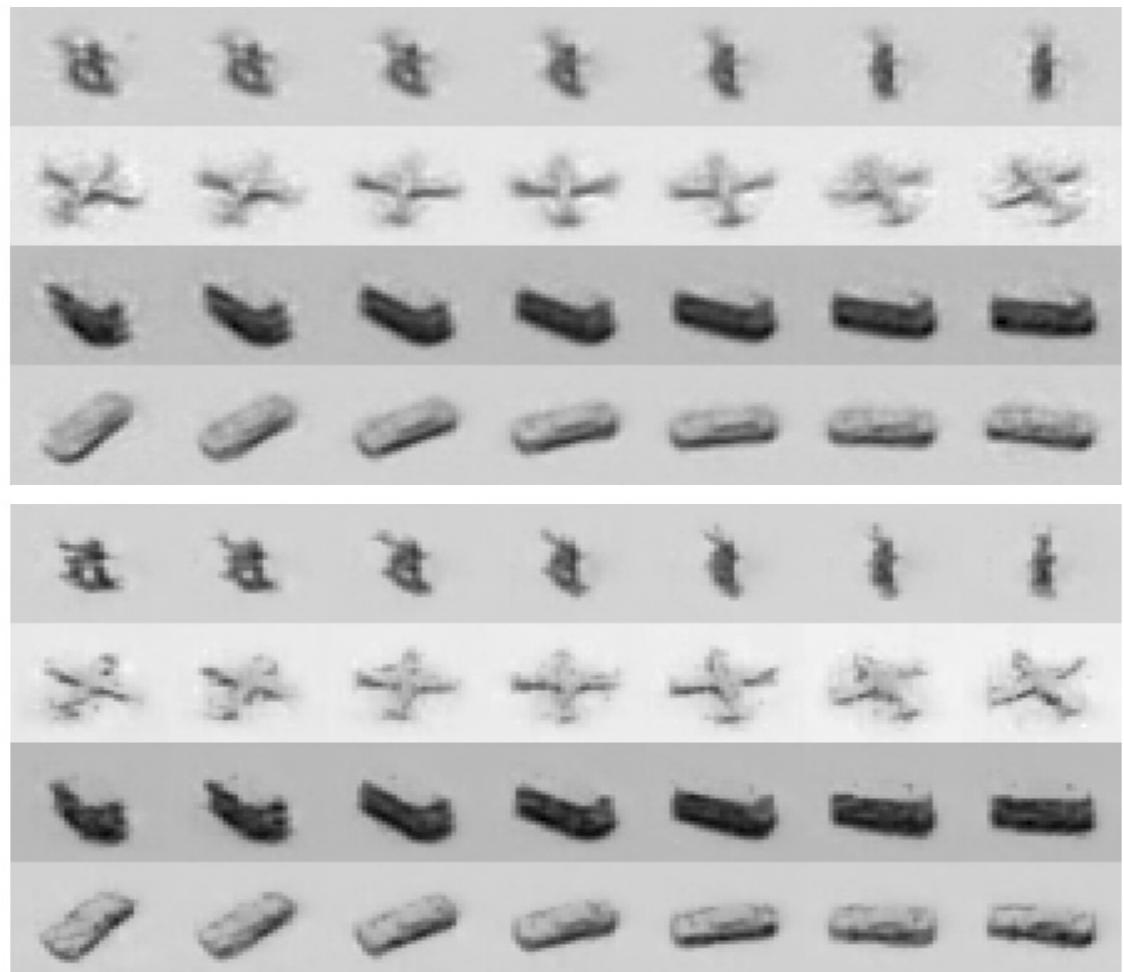


(b) Shallow Architecture 2

Overlapping pooling  
4x4x4 (feat,x,y)  
Stride 2 over features

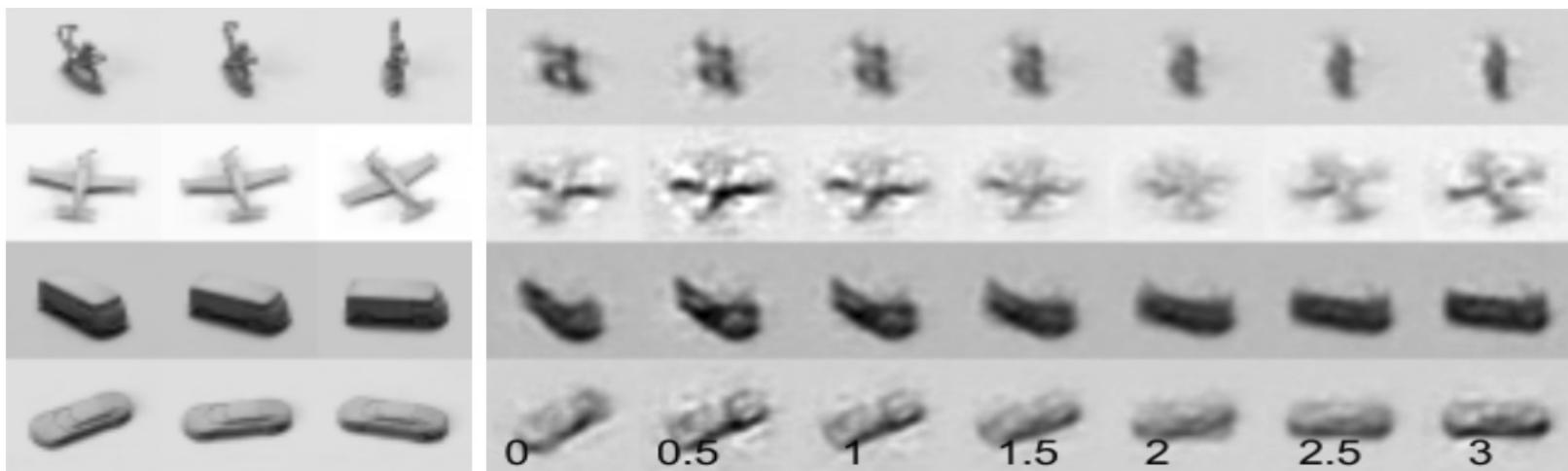
# Image interpolation in feature space: deterministic world

- No phase pooling
  - No curvature regularization
- 
- With phase pooling
  - With curvature regularization



# Image interpolation in feature space: deterministic world

- Image interpolation with the basic model (siamese net)



# Image interpolation in feature space: unpredictable world

- Phase interpolation
  - No latent variable
- 
- Phase interpolation
  - With latent variable

