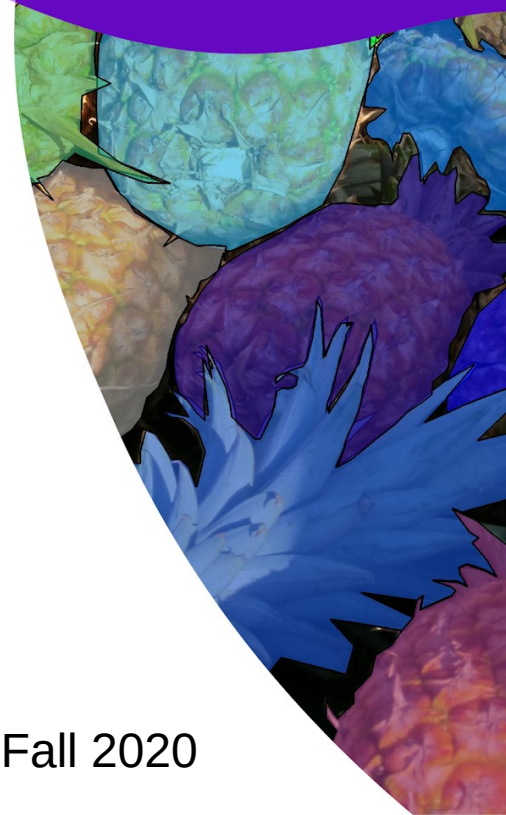
NEW YORK UNIVERSITY

Energy-Based Models (part 2)



Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research
http://yann.lecun.com

Deep Learning, NYU, Fall 2020

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

Contrastive Methods: loss functions

- **Push down on data points, push up of well-chosen other points**

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

- Example: NCE

$$\mathcal{L}(x, y, \hat{y}_1, \dots \hat{y}_{p^-}, w) = \frac{e^{-E_w(x,y)}}{e^{-E_w(x,y)} + \sum_{i=1}^{p^-} e^{-E_w(x,\hat{y}_i,w)}}$$

- General margin loss: $\mathcal{L}(x, y, w) = \sum_{\check{y} \in \mathcal{Y}} H(F_w(x, y), F_w(x, \check{y}), m(y, \check{y}))$
  - H(F⁺,F⁻,m) increasing fn of F⁺ decreasing fn of F⁻, whenever F⁻ - F⁺ < m.
  - Example: simple hinge [Bromley 1993]:

$$\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y)]^+ + [m(y, \hat{y}) - F_w(x, \hat{y})]^+$$

Plenty of Contrastive Loss Functions to Choose From

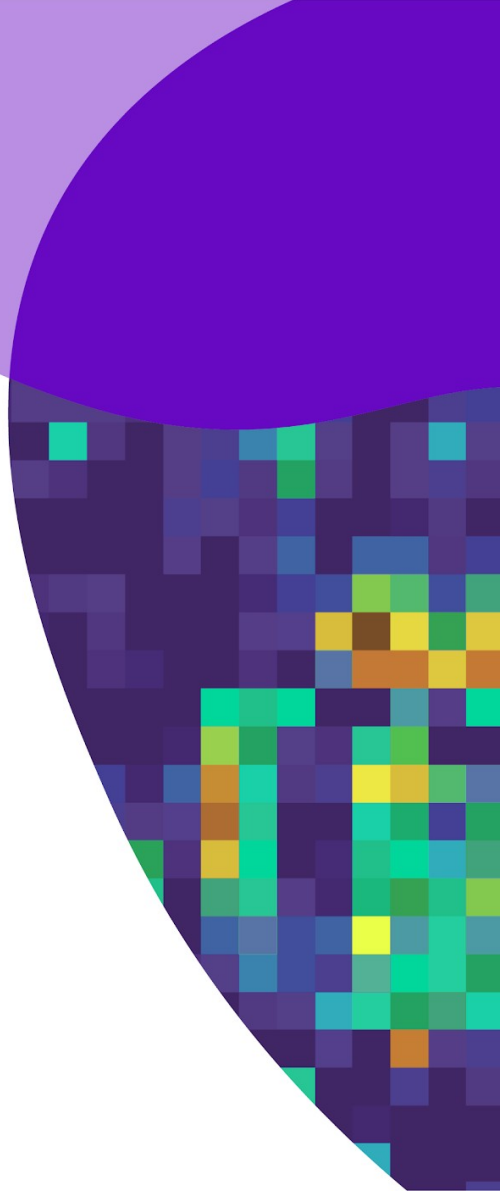
Good and bad loss functions: good ones have non-zero margin

Loss (equation #)	Formula	Margin
energy loss	$E(W, Y^i, X^i)$	none
perceptron	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2	$\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	0
MCE	$\left(1 + e^{-\left(E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)\right)}\right)^{-1}$	> 0
square-square	$E(W, Y^i, X^i)^2 - \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$	m
square-exp	$E(W, Y^i, X^i)^2 + \beta e^{-E(W,\bar{Y}^i,X^i)}$	> 0
NLL/MMI	$E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0
MEE	$1 - e^{-\beta E(W,Y^i,X^i)} / \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0

FACEBOOK AI

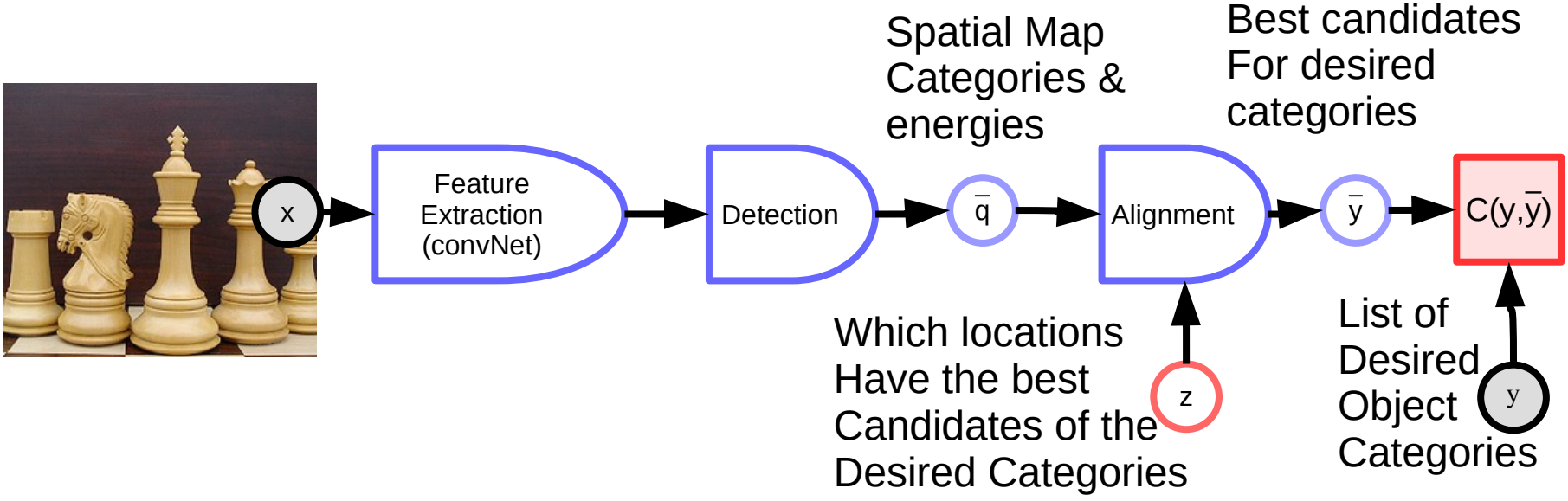
Latent Variable Models in Practice

Structured prediction with latent variable models



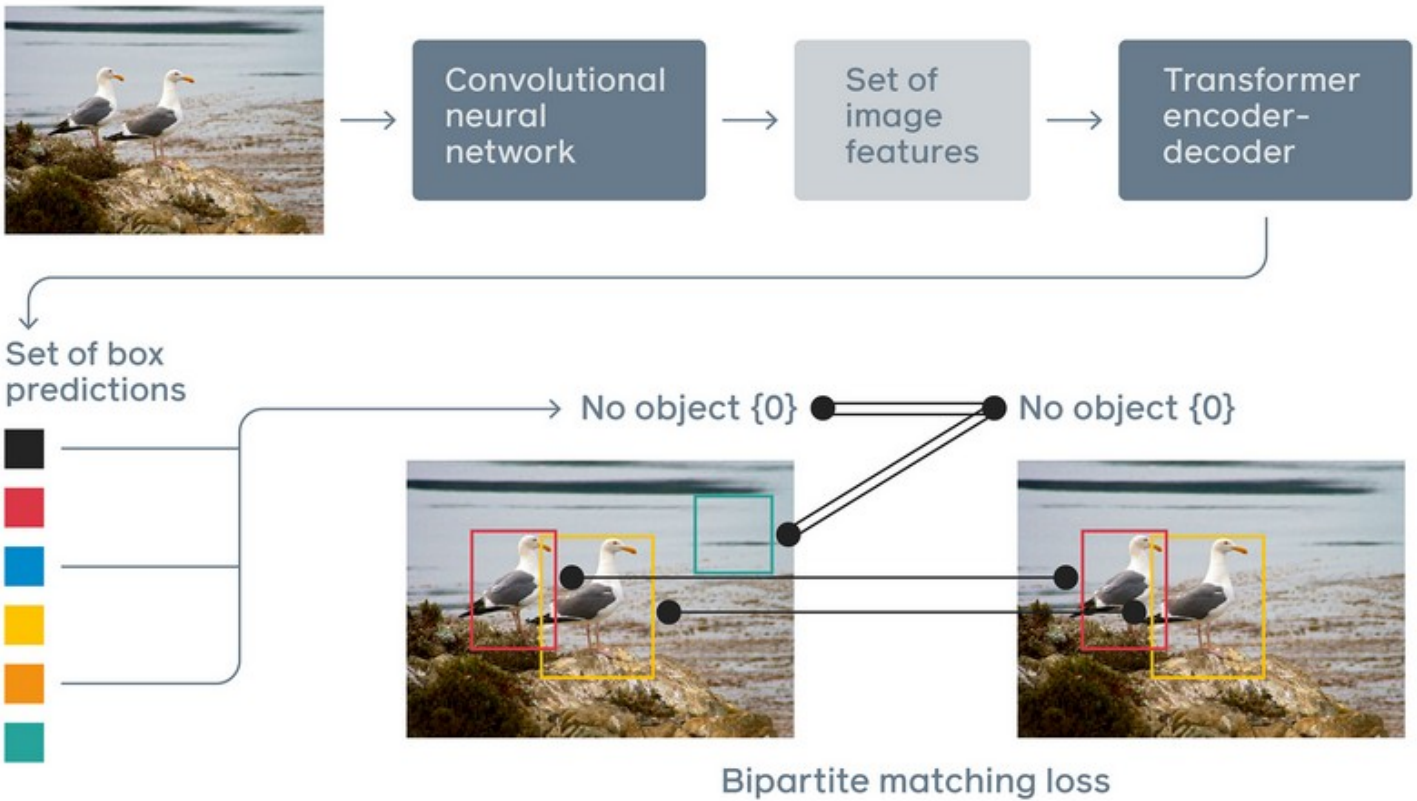
Structured Prediction

- **Complex output with weak/partial supervision**
  - Speech recognition: alignment of audio to text transcription
  - Handwriting recognition: alignment of image to character sequence
  - Object detection: alignment of image features to labeled categories



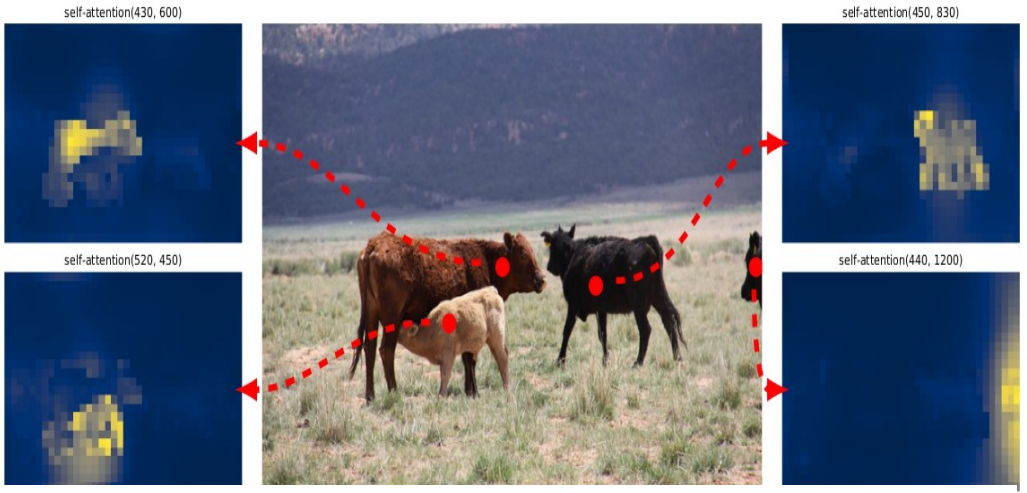
DETR:

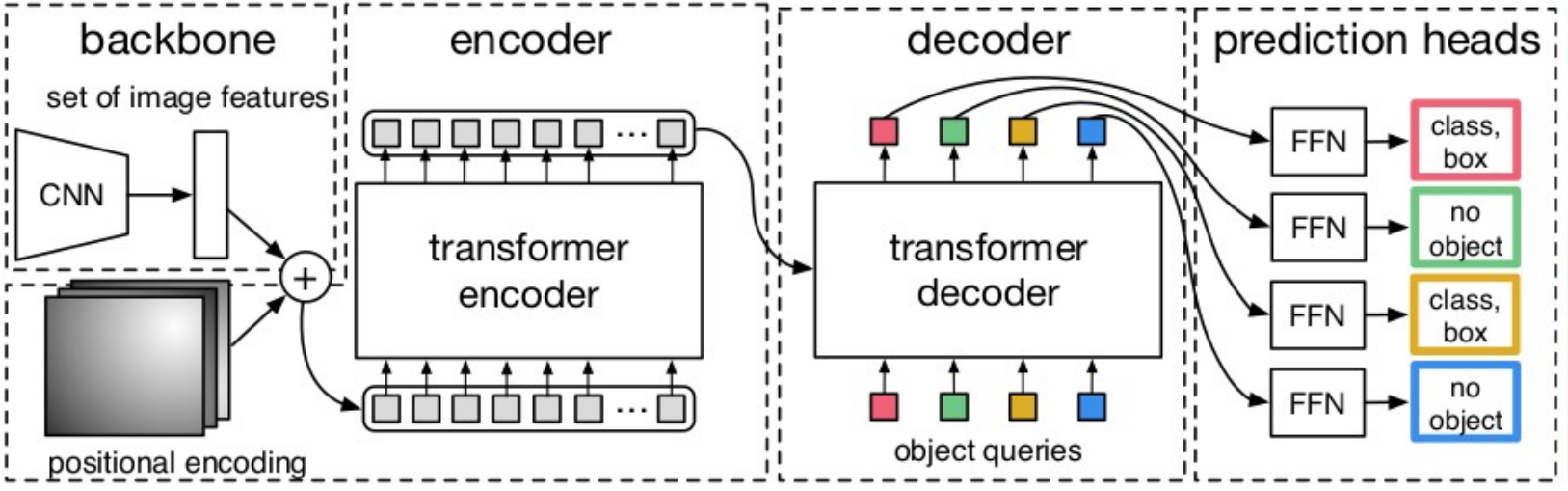
- DETR [Carion et al. ArXiv:2005.12872] https://github.com/facebookresearch/detr
- ConvNet → Transformer
- Object-based visual reasoning

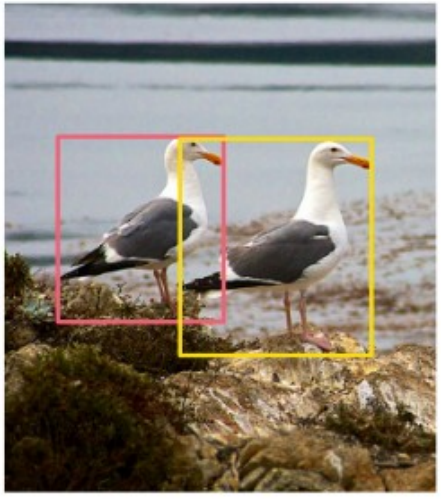


DETR: ConvNet → Transformer for object detection

- **DETR [Carion et al. ArXiv:2005.12872]**
- **https://github.com/facebookresearch/detr**
- **ConvNet → Transformer**
- **Object-based visual reasoning**
- **Transformer: dynamic networks**
  - Through “attention”

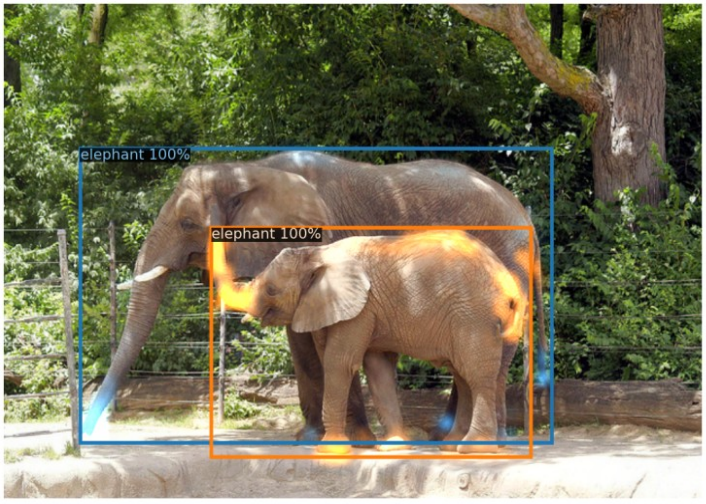


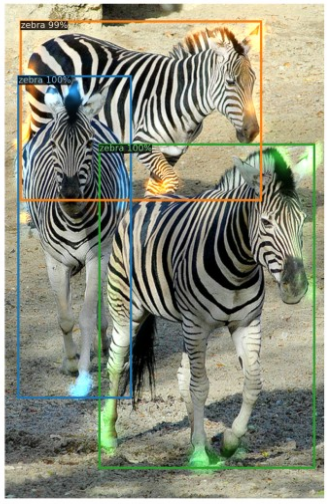


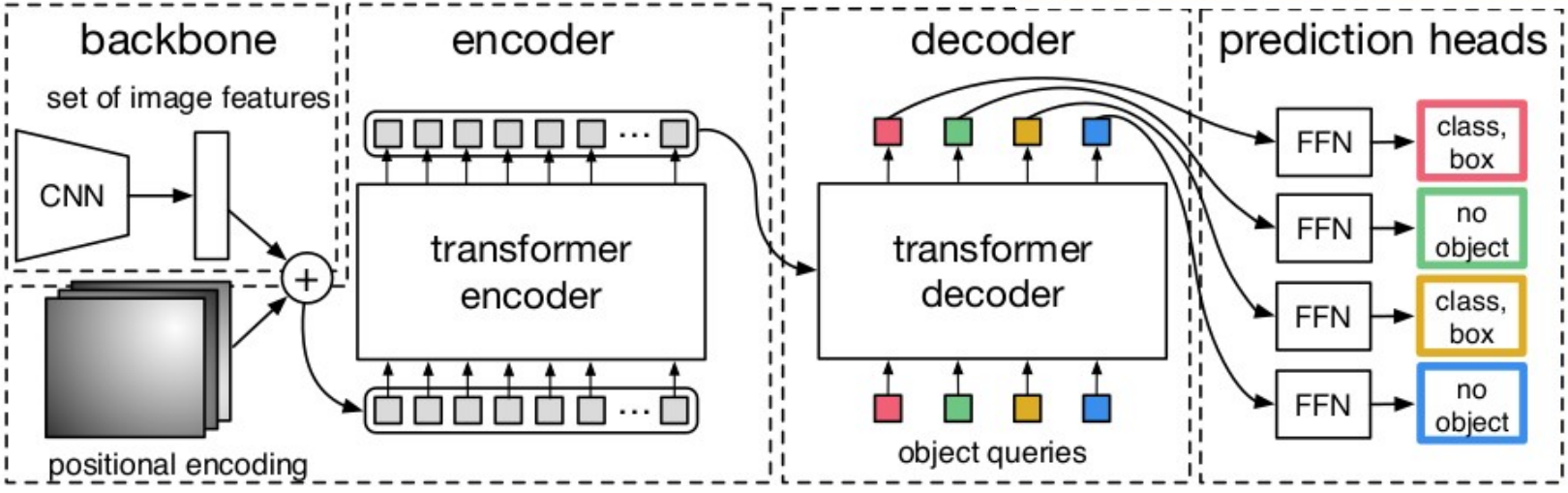


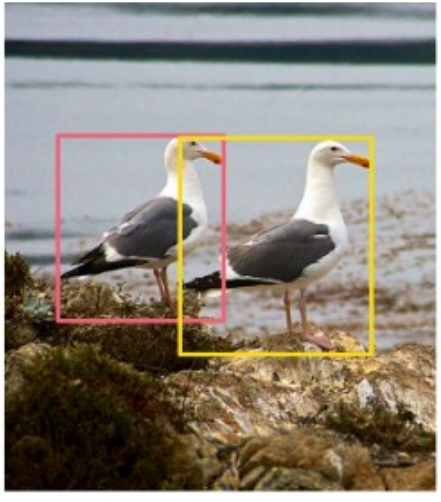
DETR: ConvNet → Transformer for object detection

- **DETR [Carion et al. ArXiv:2005.12872]**
- **https://github.com/facebookresearch/detr**
- **ConvNet → Transformer**
- **Object-based visual reasoning**
- **Transformer: dynamic networks**
  - Through “attention”

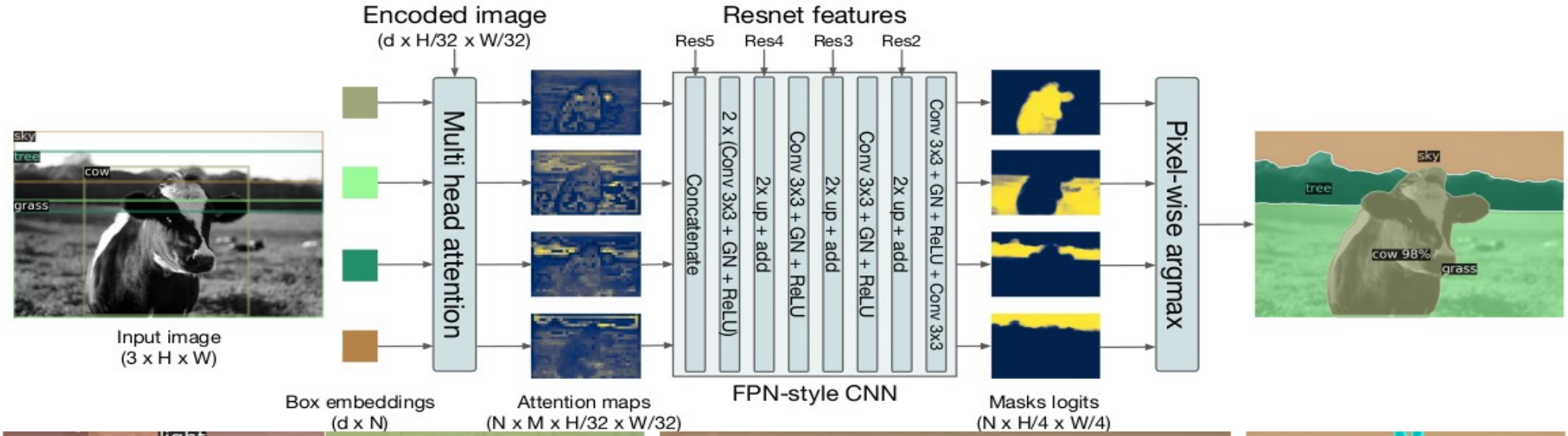


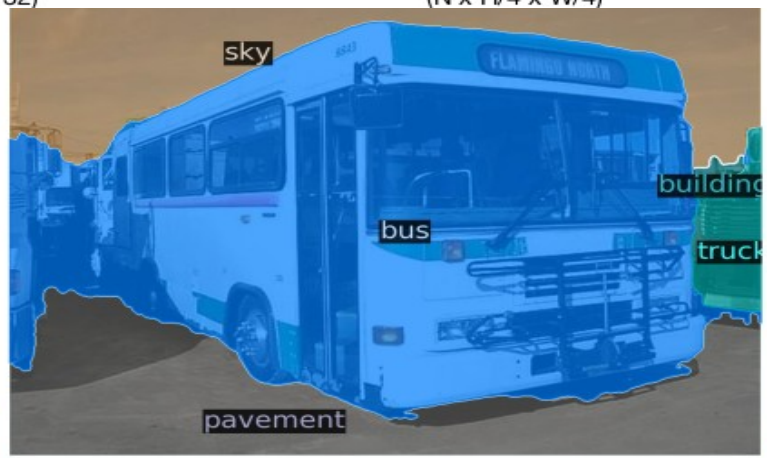


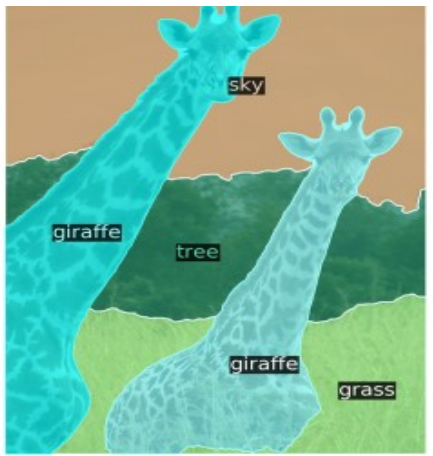




DETR: results on panoptic segmentation



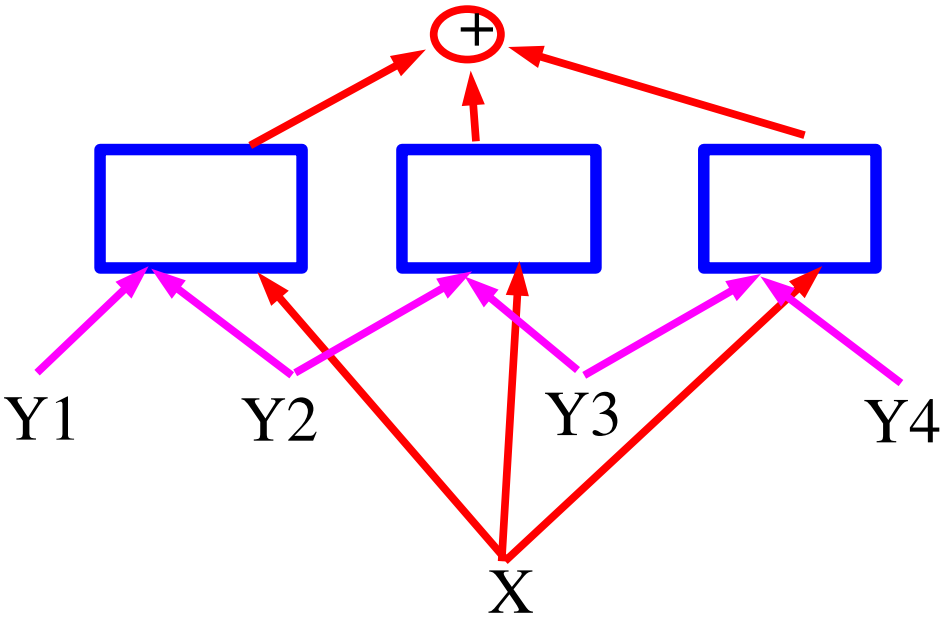




Energy-Based Factor Graphs: Energy = Sum of "factors"

- **Sequence Labeling**
  - Output is a sequence Y1,Y2,Y3,Y4.....
  - NLP parsing, MT, speech/handwriting recognition, biological sequence analysis
  - The factors ensure grammatical consistency
  - They give low energy to consistent sub-sequences of output symbols
  - The graph is generally simple (chain or tree)
  - Inference is easy (dynamic programming, min-sum)

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z, Y, X).$$

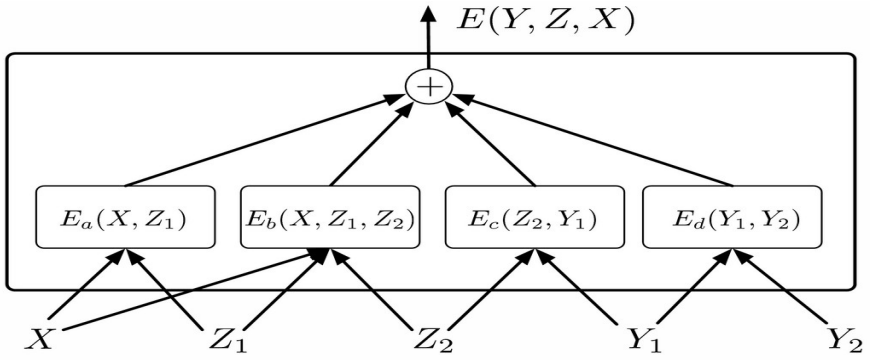


Efficient Inference: Energy-Based Factor Graphs

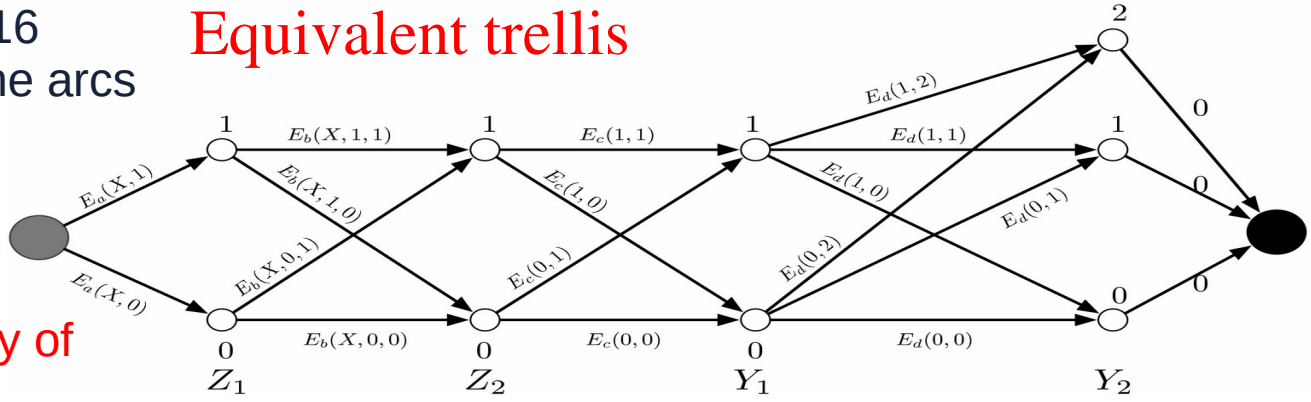
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put them on the arcs in a trellis.
  - A path in the trellis is a config of variable
  - The cost of the path is the energy of the config

- **The energy is a sum of "factor" functions**

Factor graph

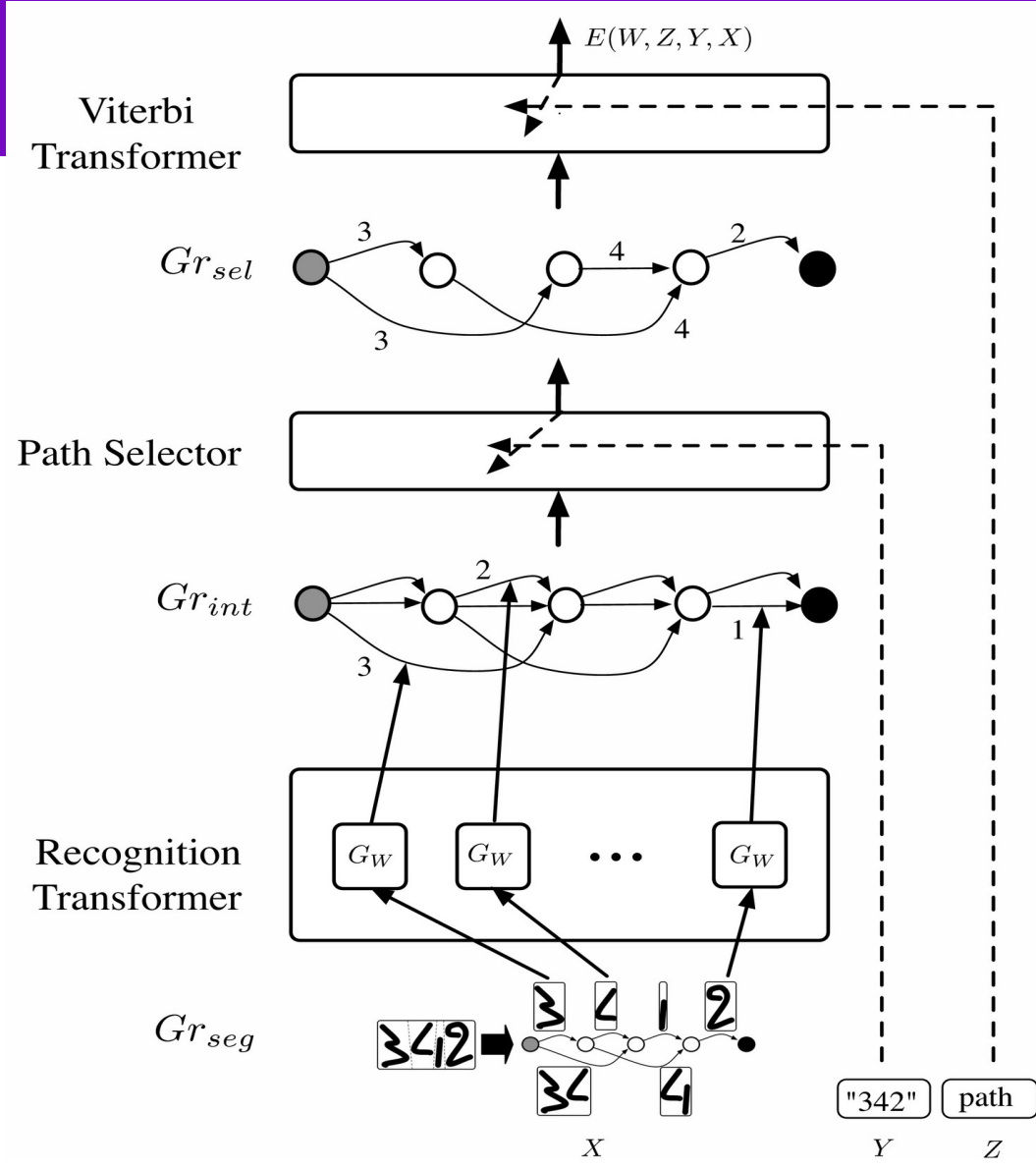


Equivalent trellis



Deep Factors & implicit graphs: GTN

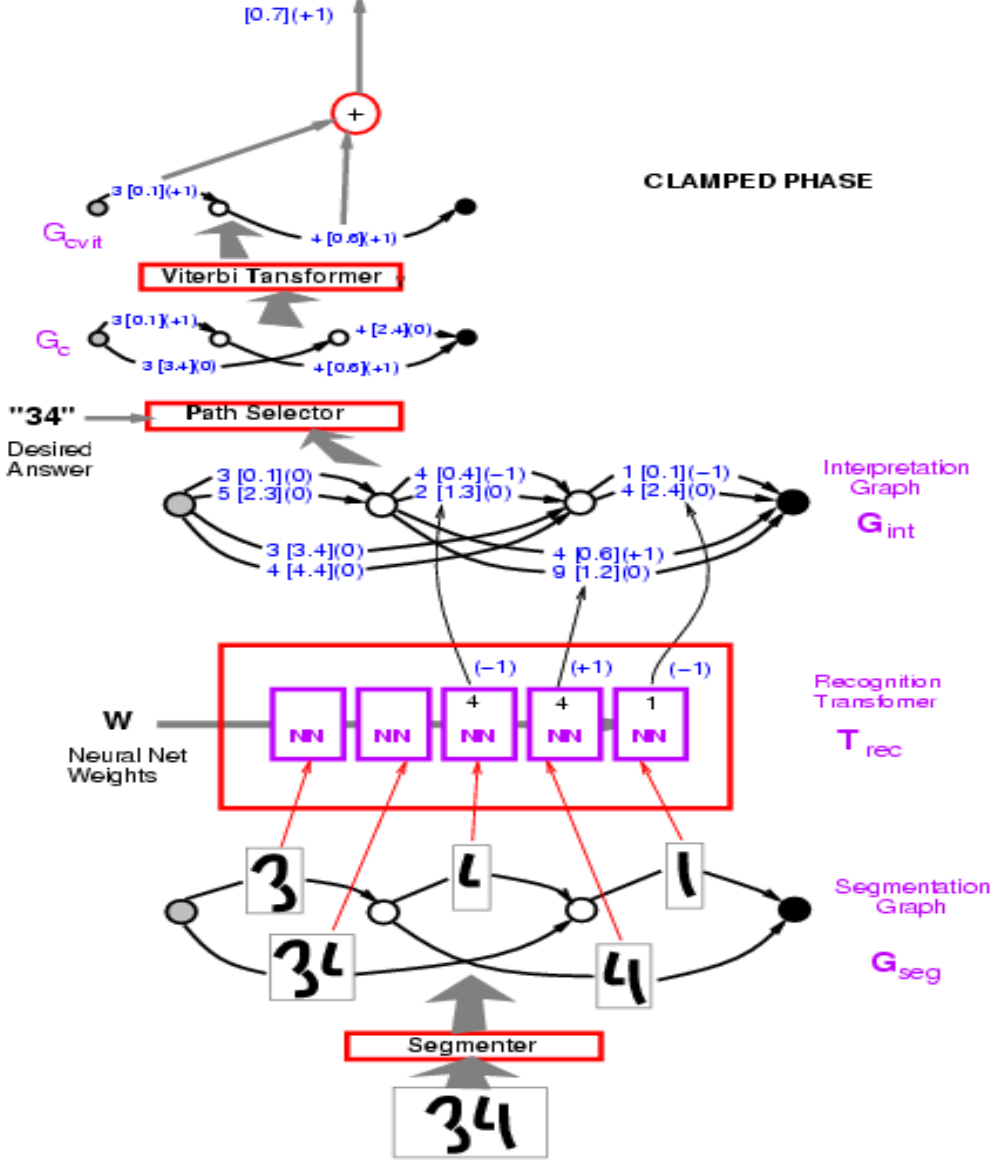
- **Handwriting Recognition with Graph Transformer Networks**
- **Un-normalized hierarchical HMMs**
  - Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
  - Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]
- **Answer = sequence of symbols**
- **Latent variable = segmentation**



Graph Transformer Networks

- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the energy of the desired answer:**

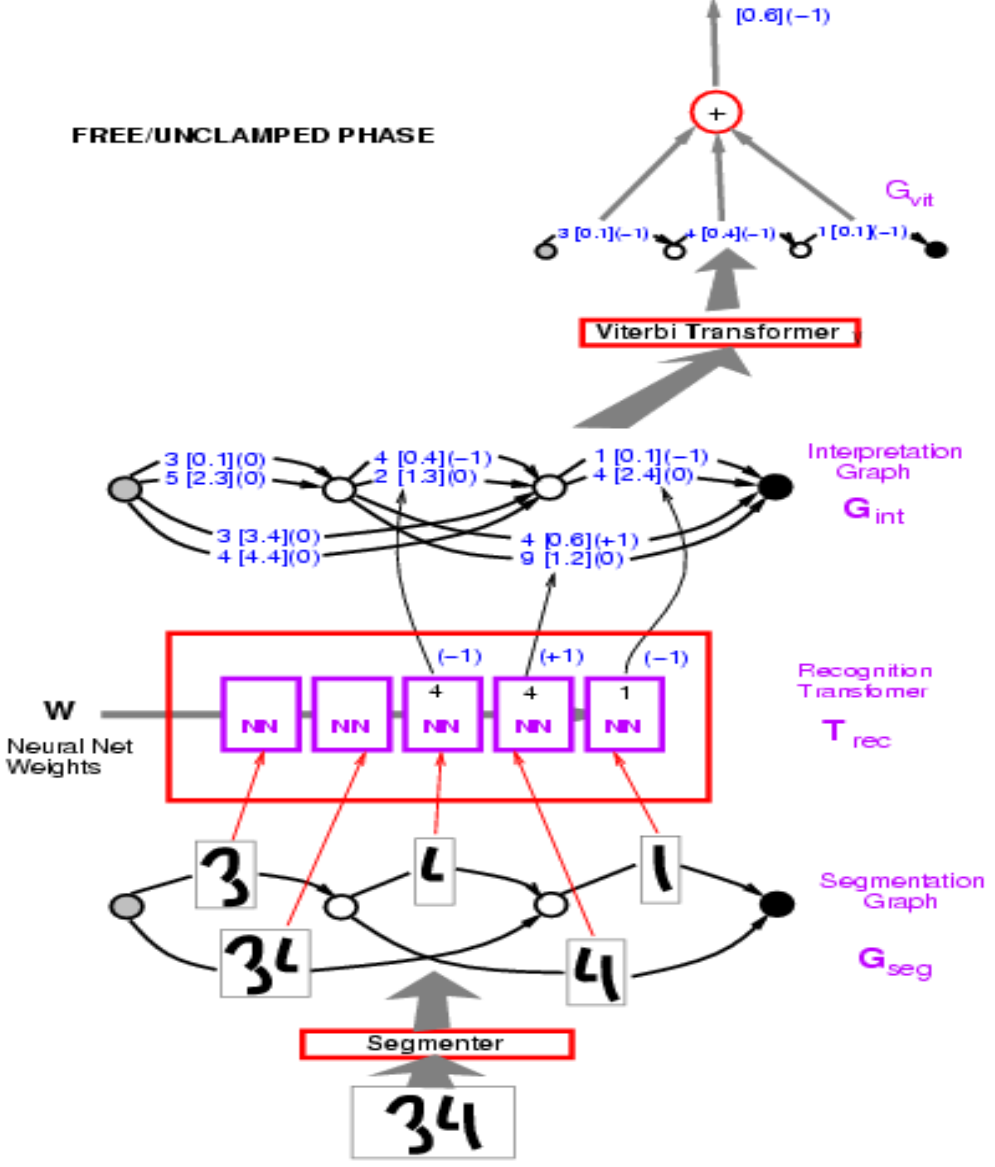
$$E(W, Y, X)$$



Graph Transformer Networks

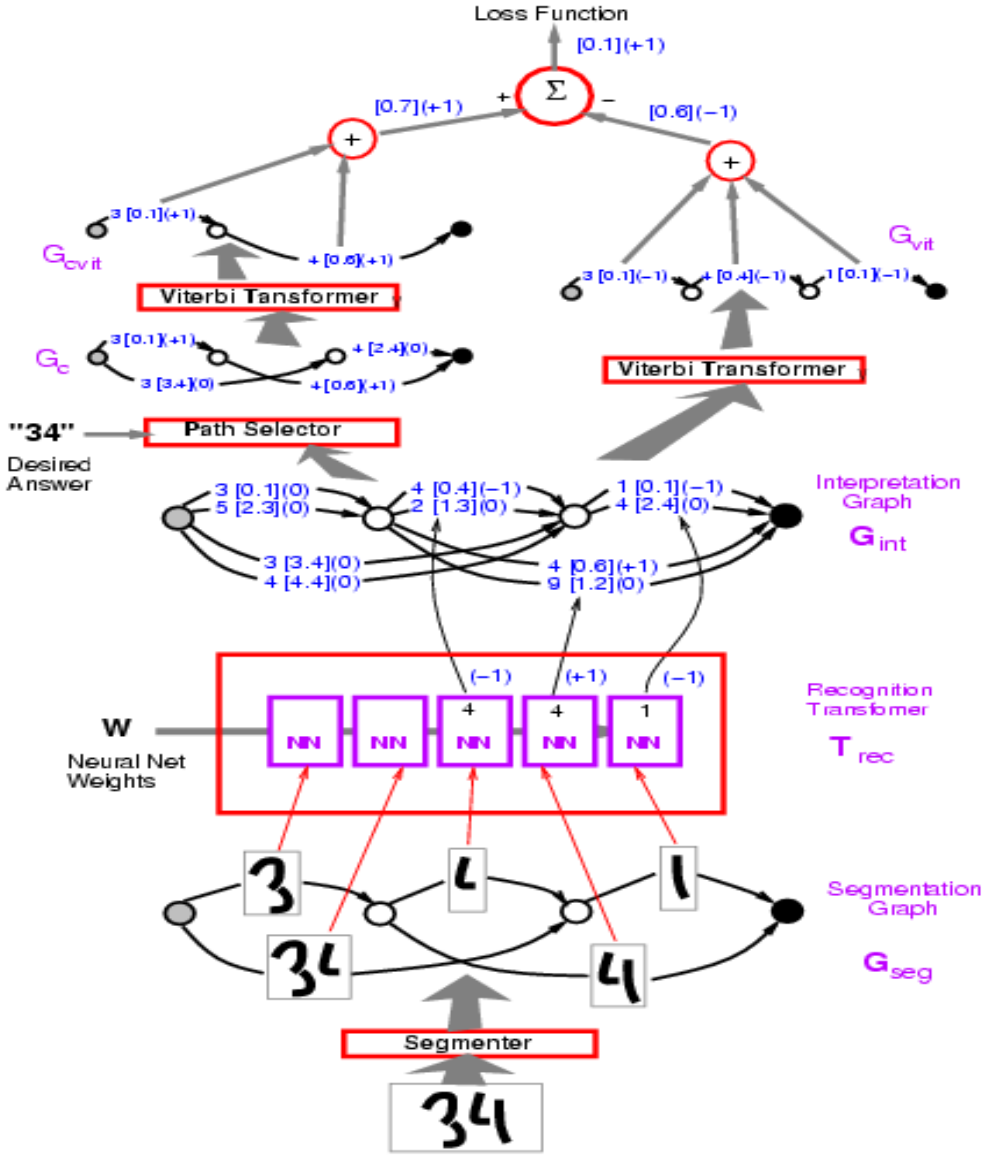
- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the constrastive term:**

$$E(W, \check{Y}, X)$$



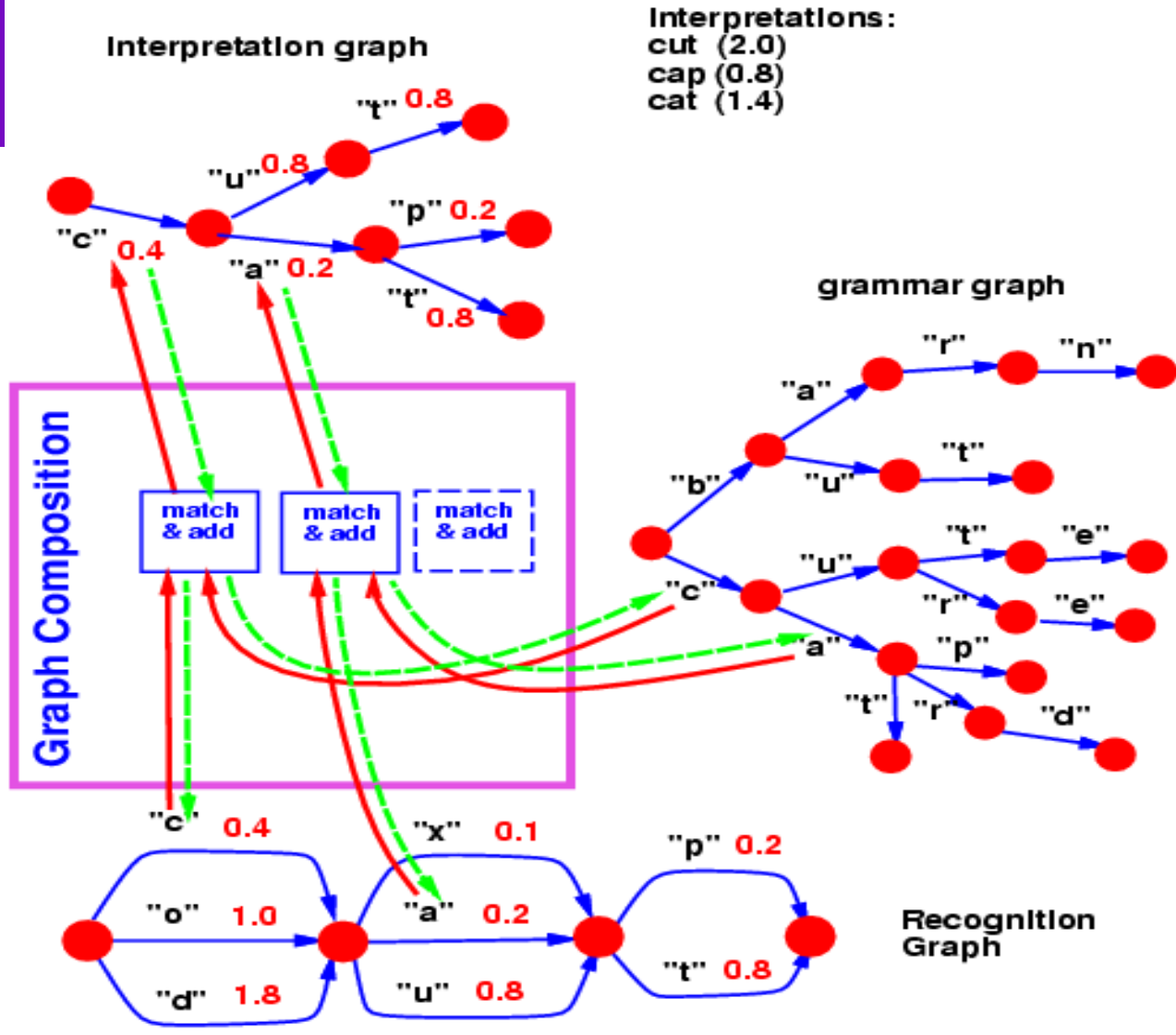
Graph Transformer Networks

- **Example: Perceptron loss**
- **Loss = Energy of desired answer – Energy of best answer.**
  - (no margin)



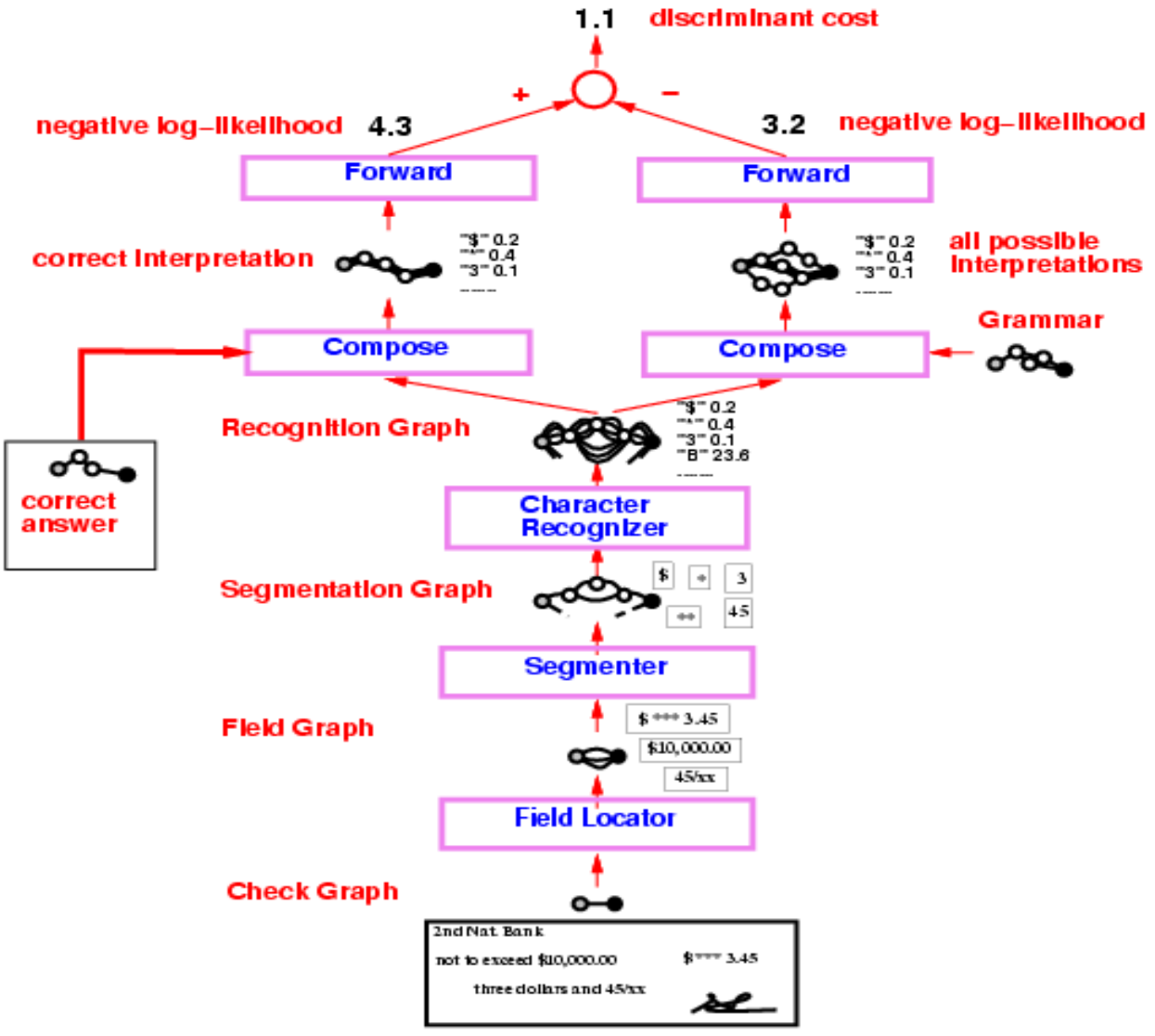
Graph Composition, Transducers.

- The composition of two graphs can be computed, the same way the dot product between two vectors can be computed.
- General theory: semi-ring algebra on weighted finite-state transducers and acceptors.



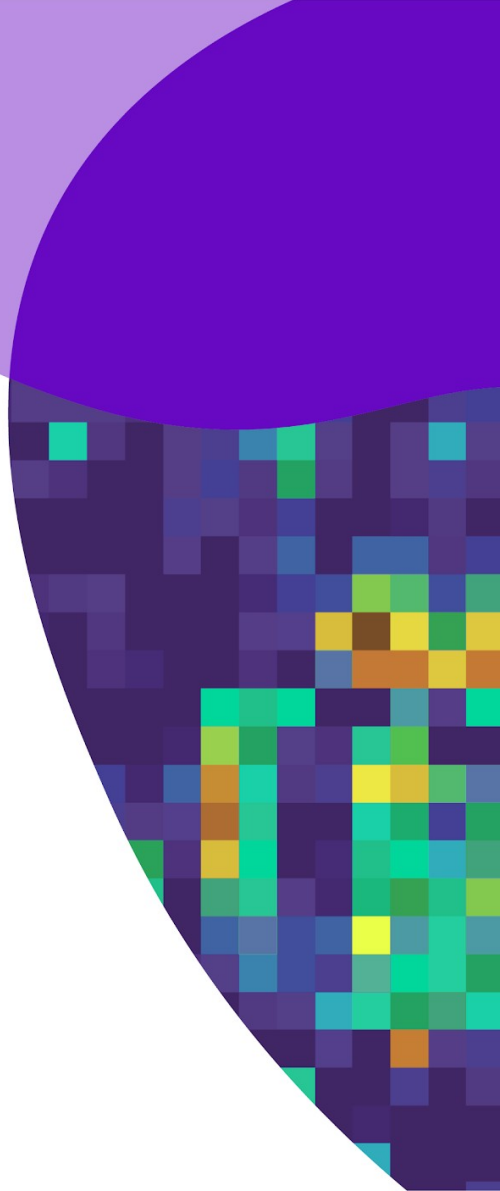
Check Reader

- Graph transformer network trained to read **check amounts.**
- Trained globally with Negative-Log-Likelihood loss.
- 50% percent correct, 49% reject, 1% error (detectable later in the process.
- **Fielded in 1996**, used in many banks in the US and Europe.
- Processes an estimated **10% of all the checks written in the US in the late 1990s**



FACEBOOK AI

Regularized Latent-Variable EBM



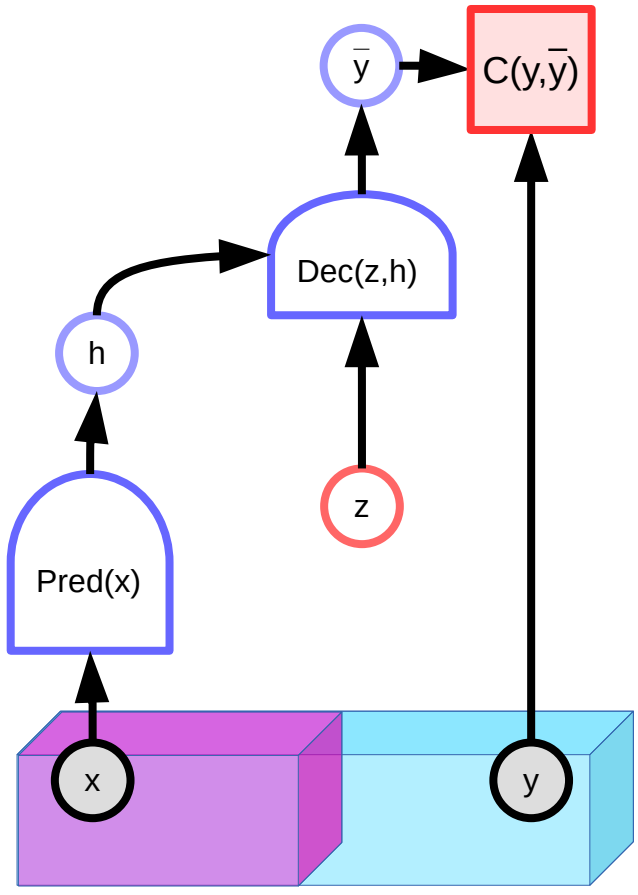
Architectural SSL methods
(non constrastive)

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**

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

- **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.**

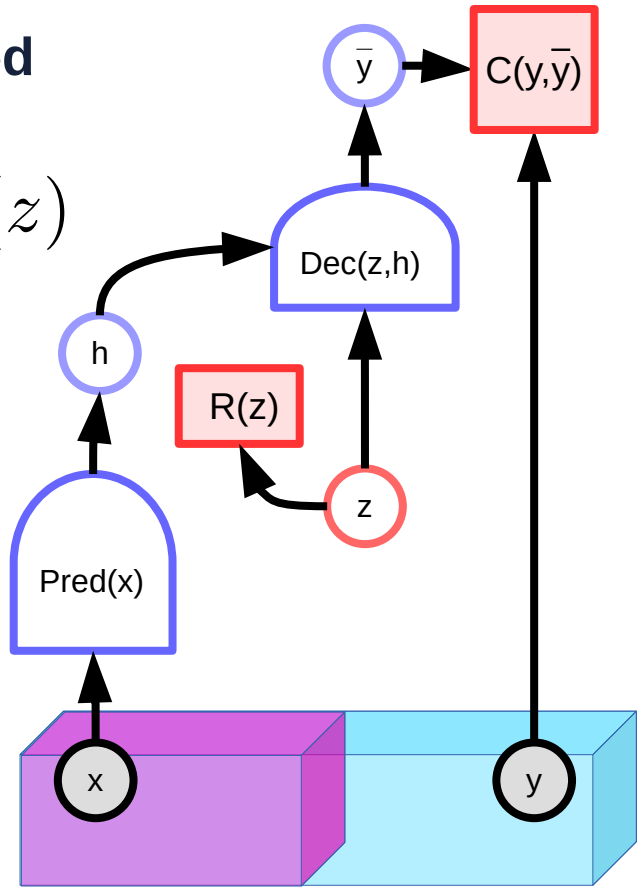


Regularized Latent Variable EBM

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

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

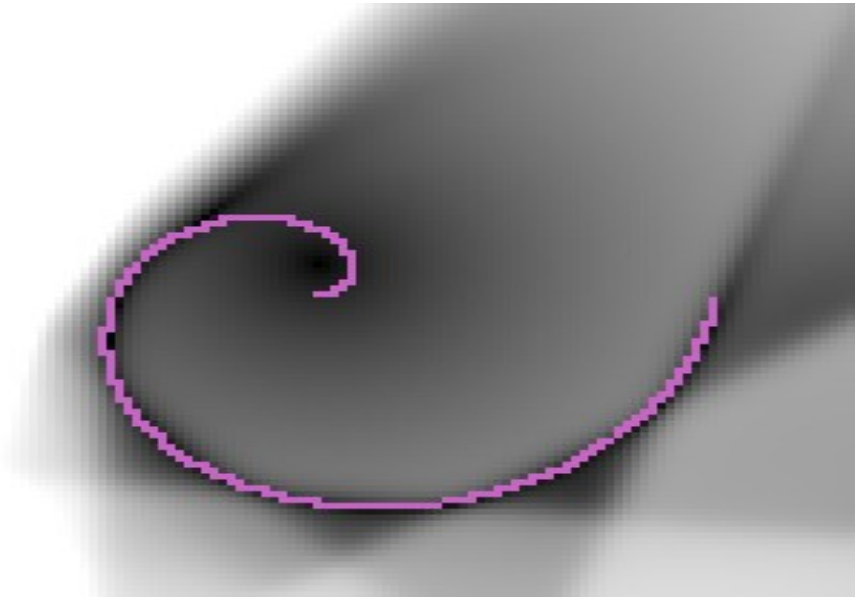
- **Examples of R(z):**
  - Effective dimension
  - Quantization / discretization
  - L0 norm (# of non-0 components)
  - L1 norm with decoder normalization
  - Maximize lateral inhibition / competition
  - Add noise to z while limiting its L2 norm (VAE)
  - <your_information_throttling_method_goes_here>

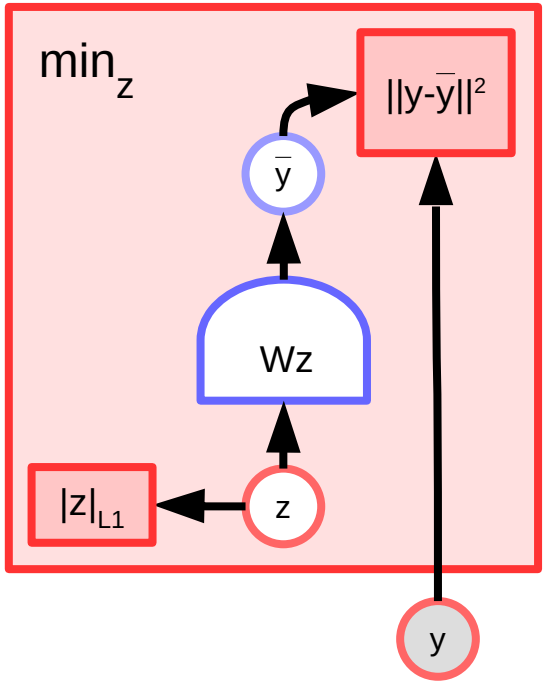


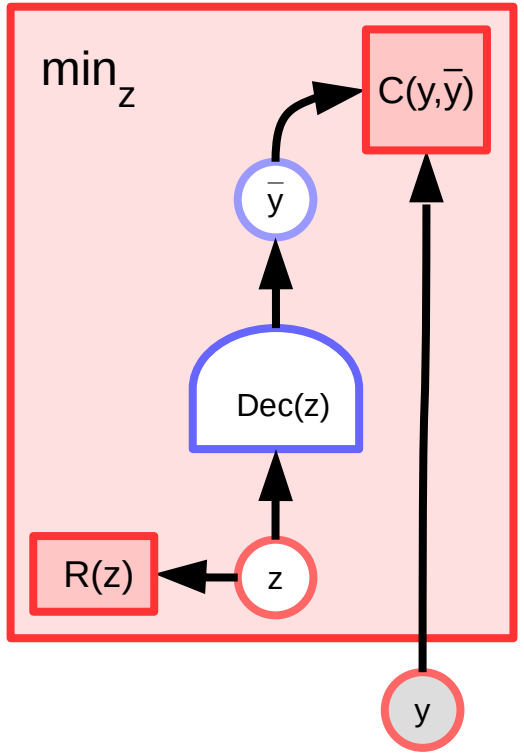
Unconditional Regularized Latent Variable EBM

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

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

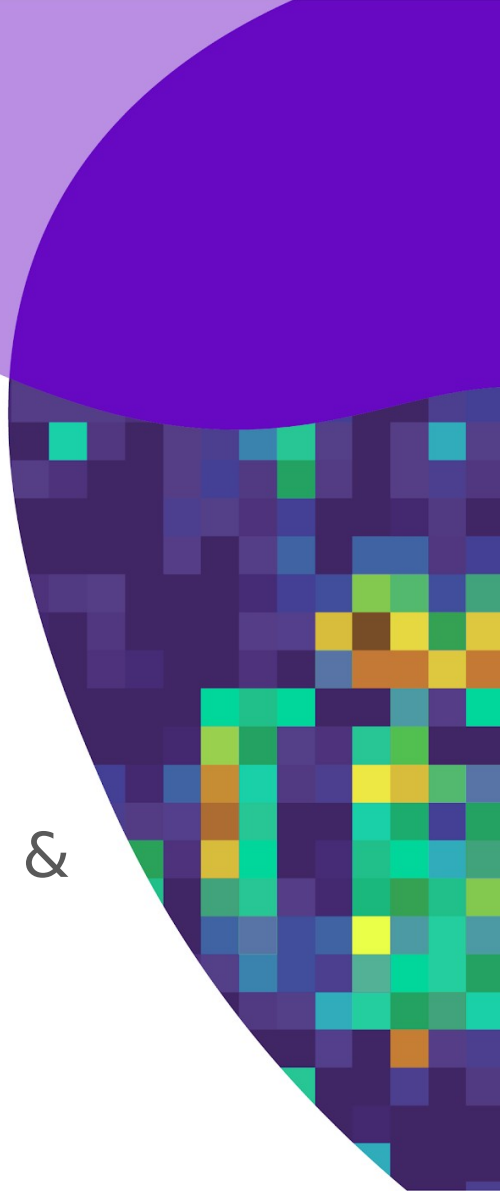






FACEBOOK AI

Architectural and Regularized EBM Training Methods



Push down on the energy of data points &
limit the information capacity
of the representation

Architectural Methods

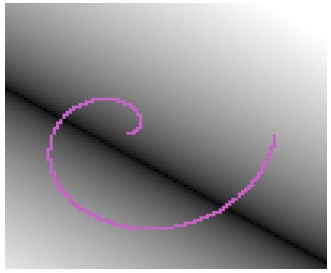
- **A1: build the machine so that the volume of the low energy regions is bounded:**
  - K-means, Gaussian Mixture Model, PCA, Bottleneck AE, Discretized AE (VQVAE),…

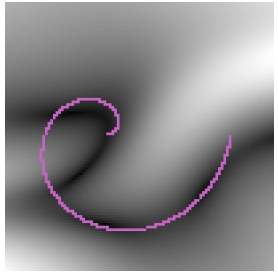
- PCA

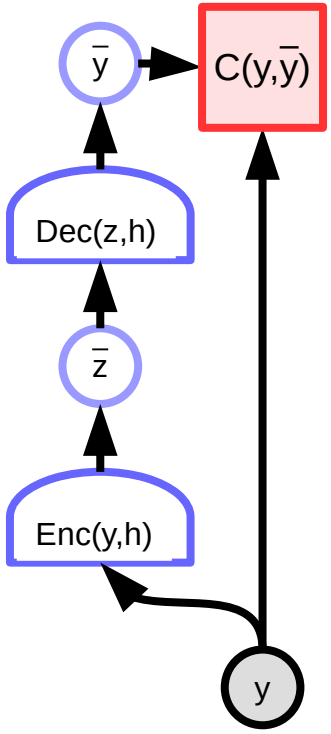
$$F(y) = ||y - w^t w y||^2$$

- Bottleneck AE

$$F(y) = C(y, \bar{y})$$
$$\bar{y} = \text{Dec}(\bar{z})$$
$$\bar{z} = \text{Enc}(y)$$

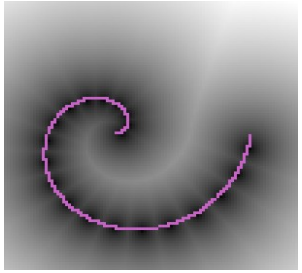


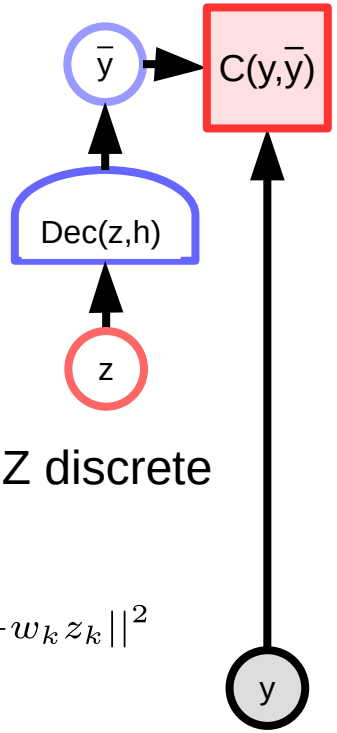




- K-means: z is a k-dim, 1-hot vector

$$E(y, z) = ||y - wz||^2$$
$$F(y) = \min_z ||y - wz||^2$$





Z discrete

- Gaussian Mixture

$$F(y) = -\log \sum_k \frac{e^{u_k}}{\sum_q e^{u_q}} e^{-||y - w_k z_k||^2}$$

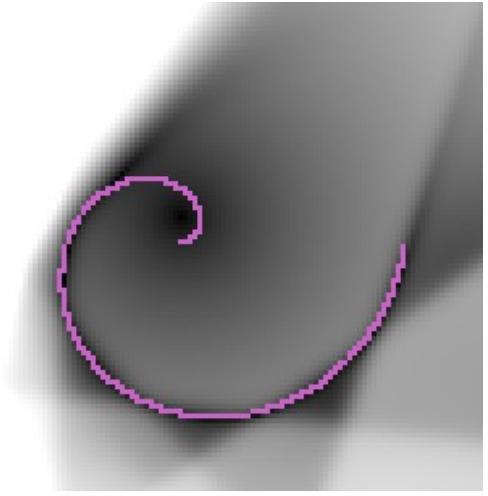
Regularized Latent Variable Methods

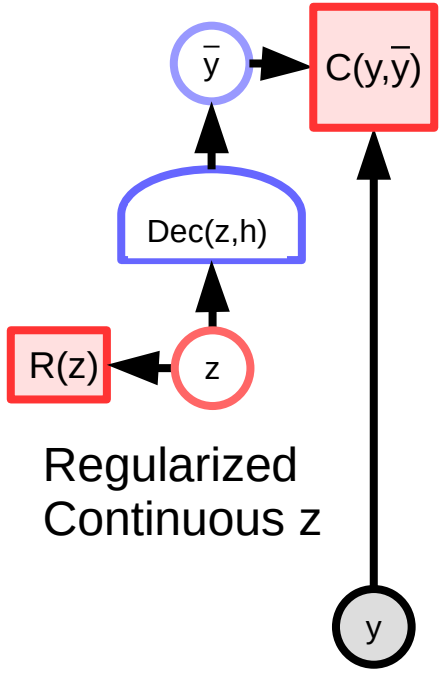
- **A2:** regularize the volume of the low energy regions

- **Sparse coding**

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

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



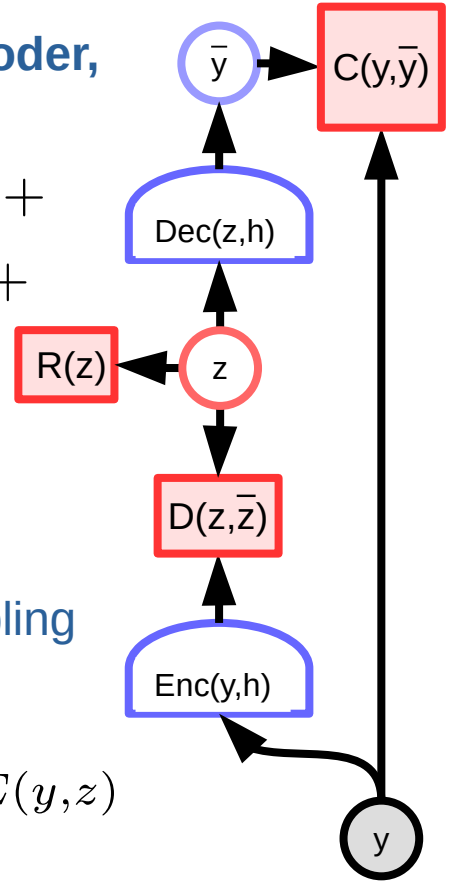


Regularized
Continuous z

- **Regularized Auto-Encoder, Sparse AE, LISTA**

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

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



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

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

Amortized Inference

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

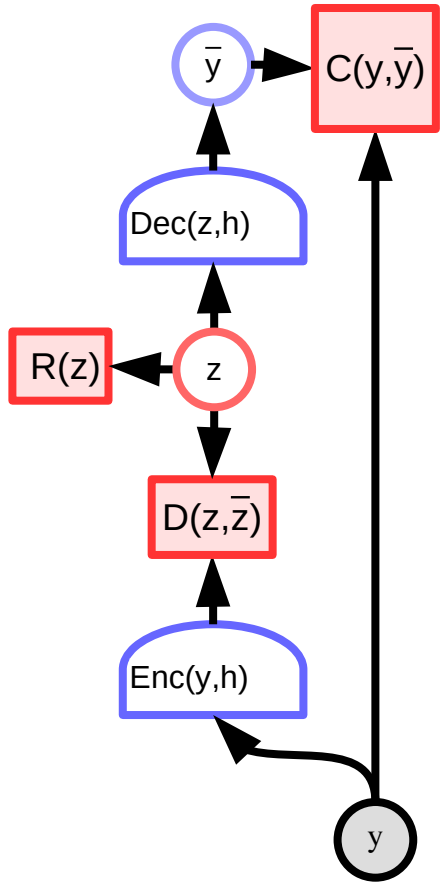
- **Regularized Auto-Encoder, Sparse AE, LISTA**

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

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

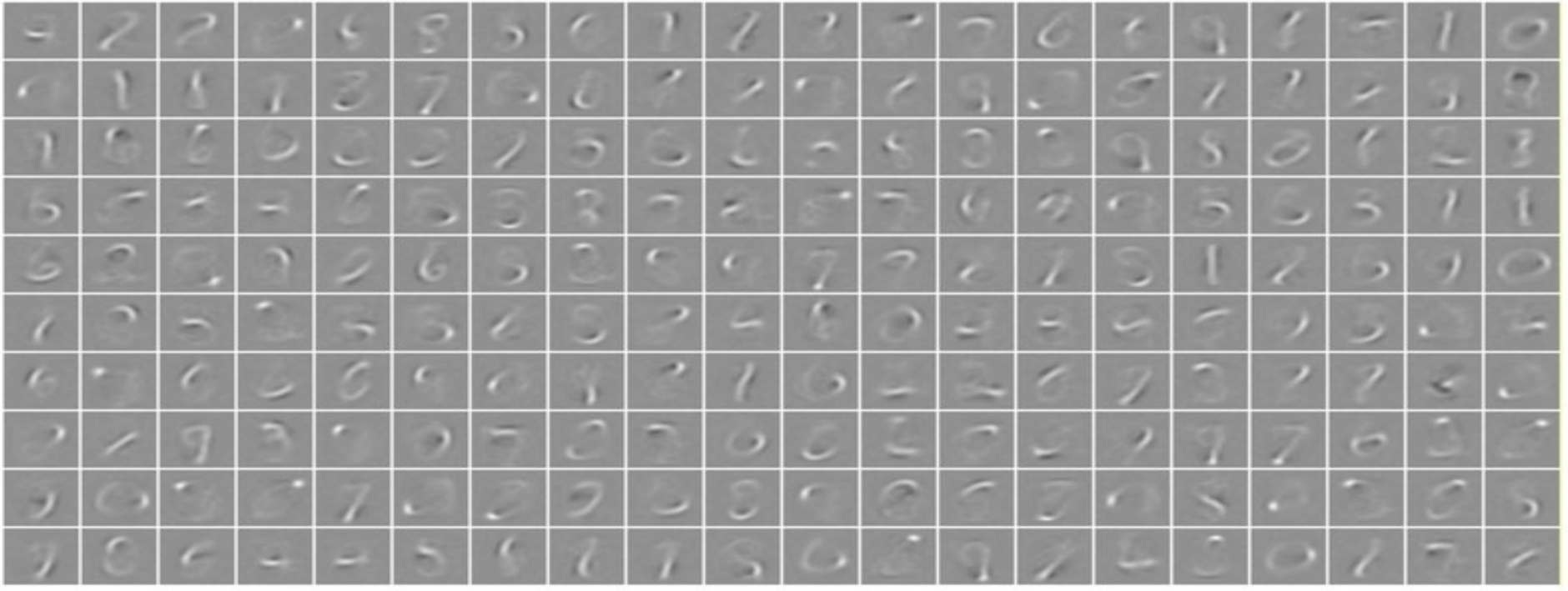
- **Variational AE**
  - Approximated by sampling and variational approximation

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



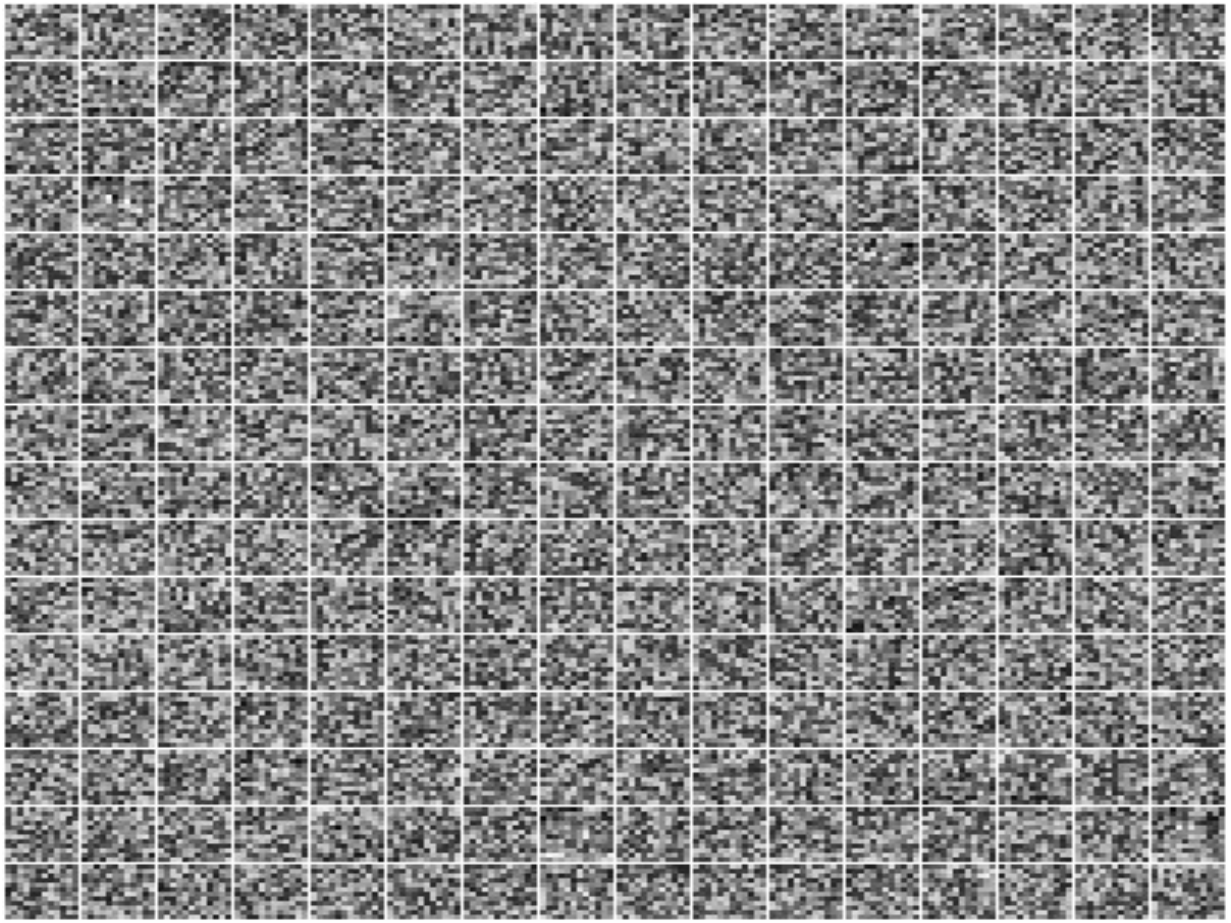
Sparse Modeling on handwritten digits (MNIST)

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

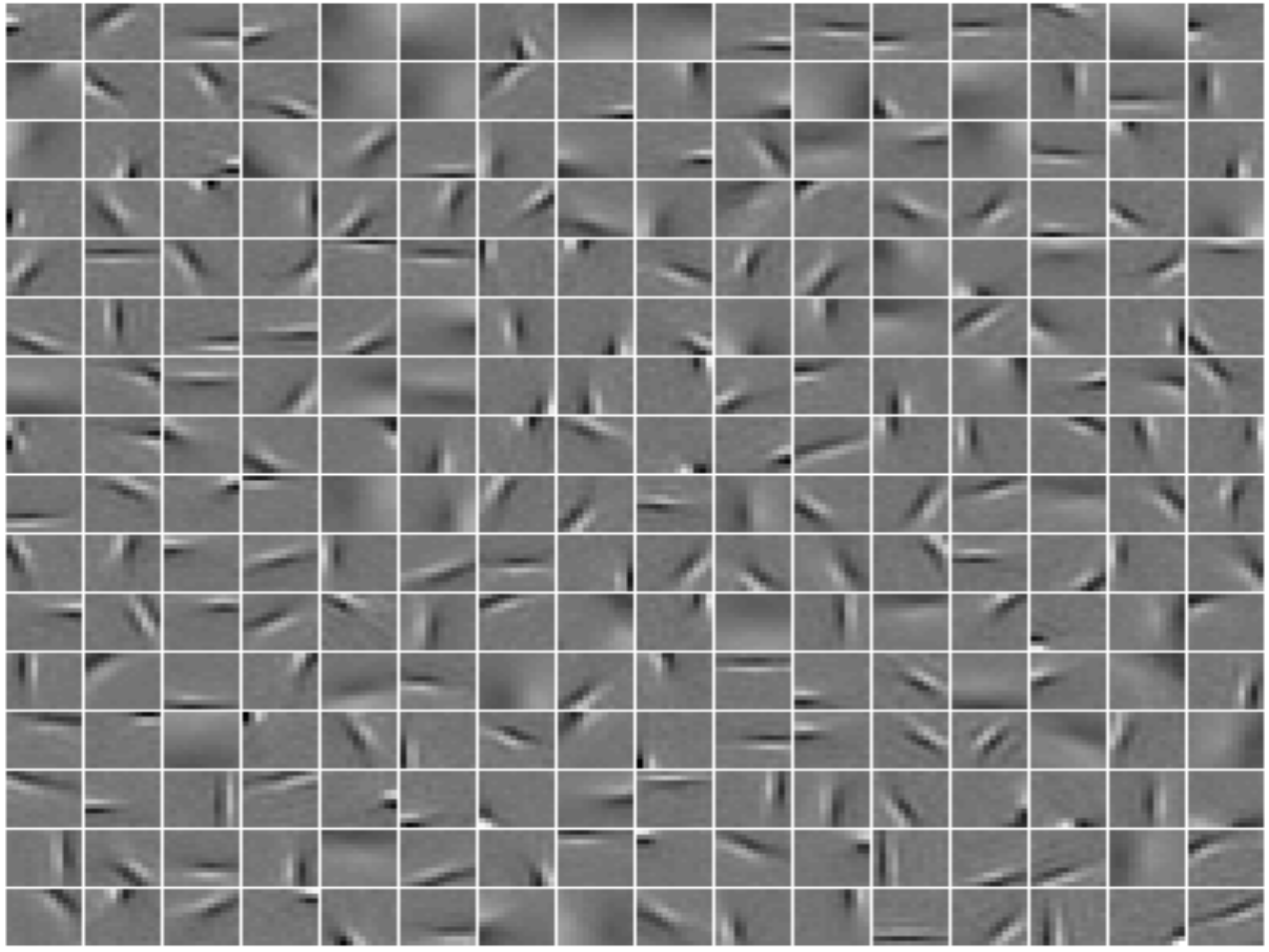


Predictive Sparse Decomposition (PSD): Training

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

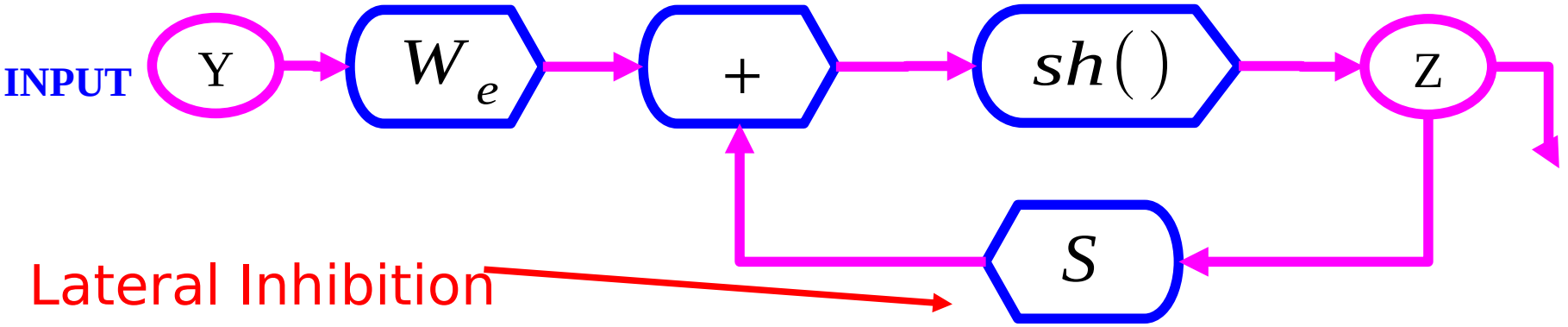


Learned Features on natural patches: V1-like receptive fields



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:**

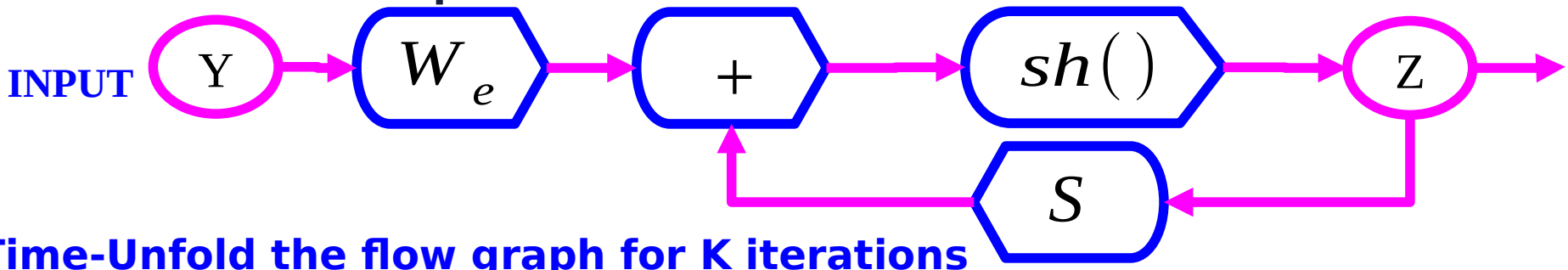
$$Z(t+1) = \text{Shrinkage}_{\lambda/L}\left[W_e^T Y + S Z(t)\right]; \quad W_e = \frac{1}{L}W_d; \quad S = I - \frac{1}{L}W_d^T W_d$$

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

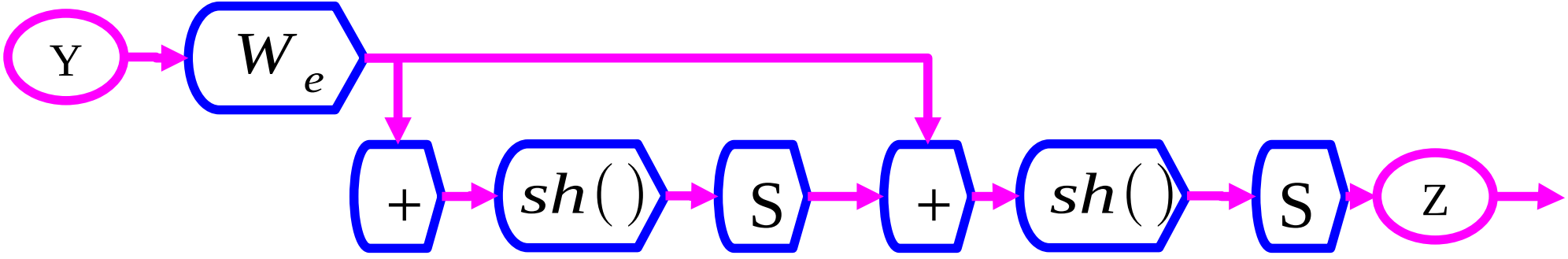
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

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

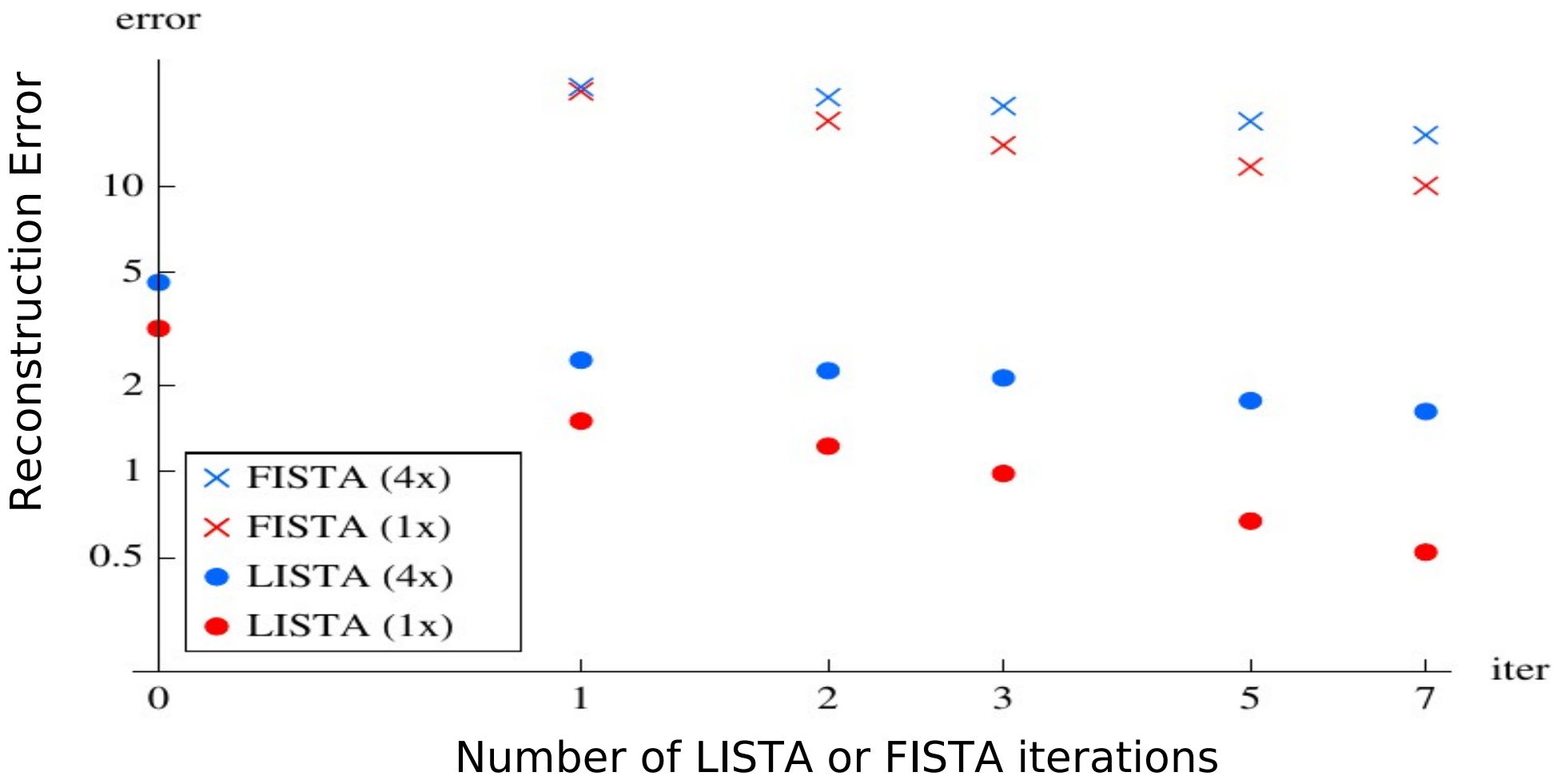
- **Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters**



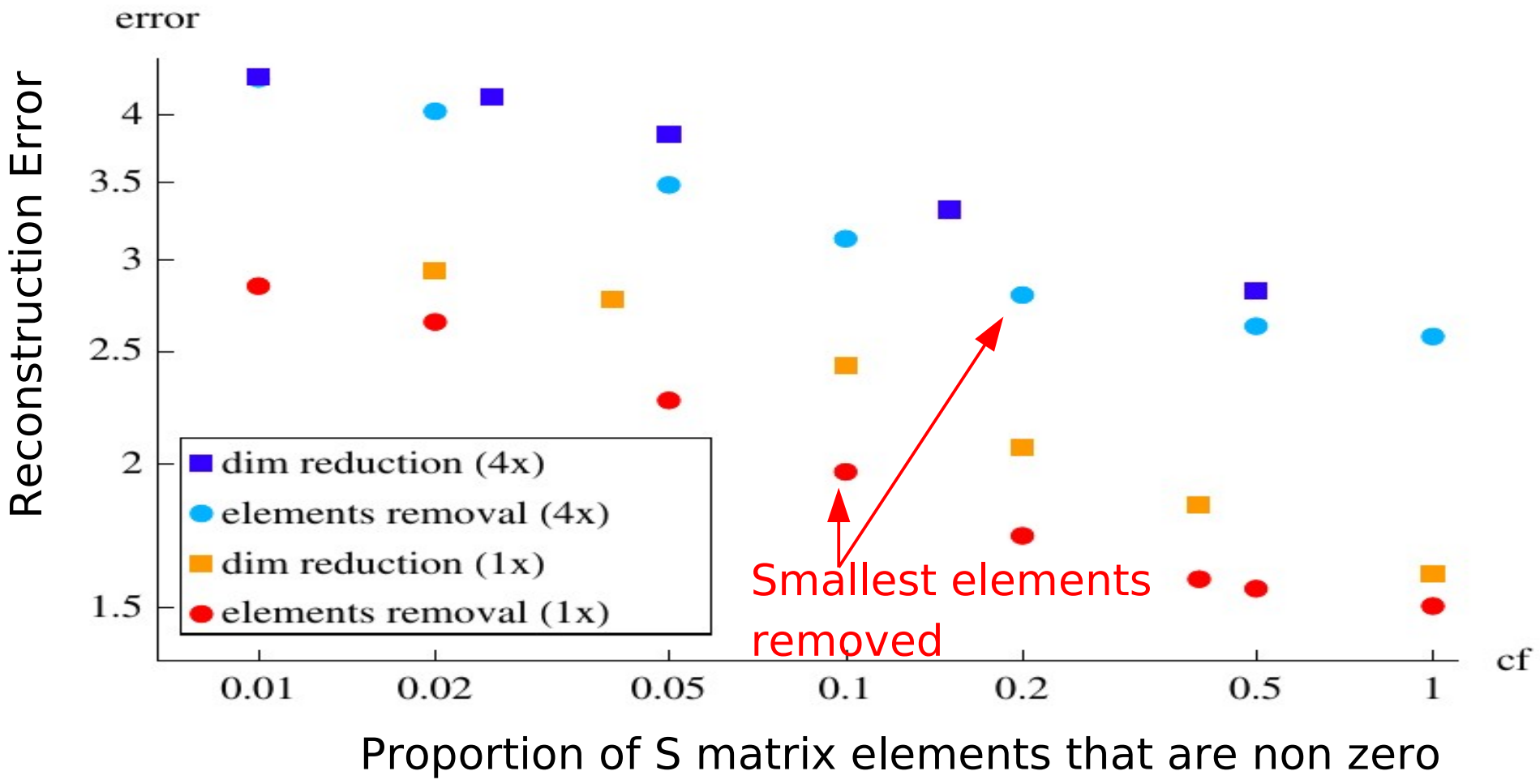
- **Time-Unfold the flow graph for K iterations**
- **Learn the We and S matrices with "backprop-through-time"**
- **Get the best approximate solution within K iterations**



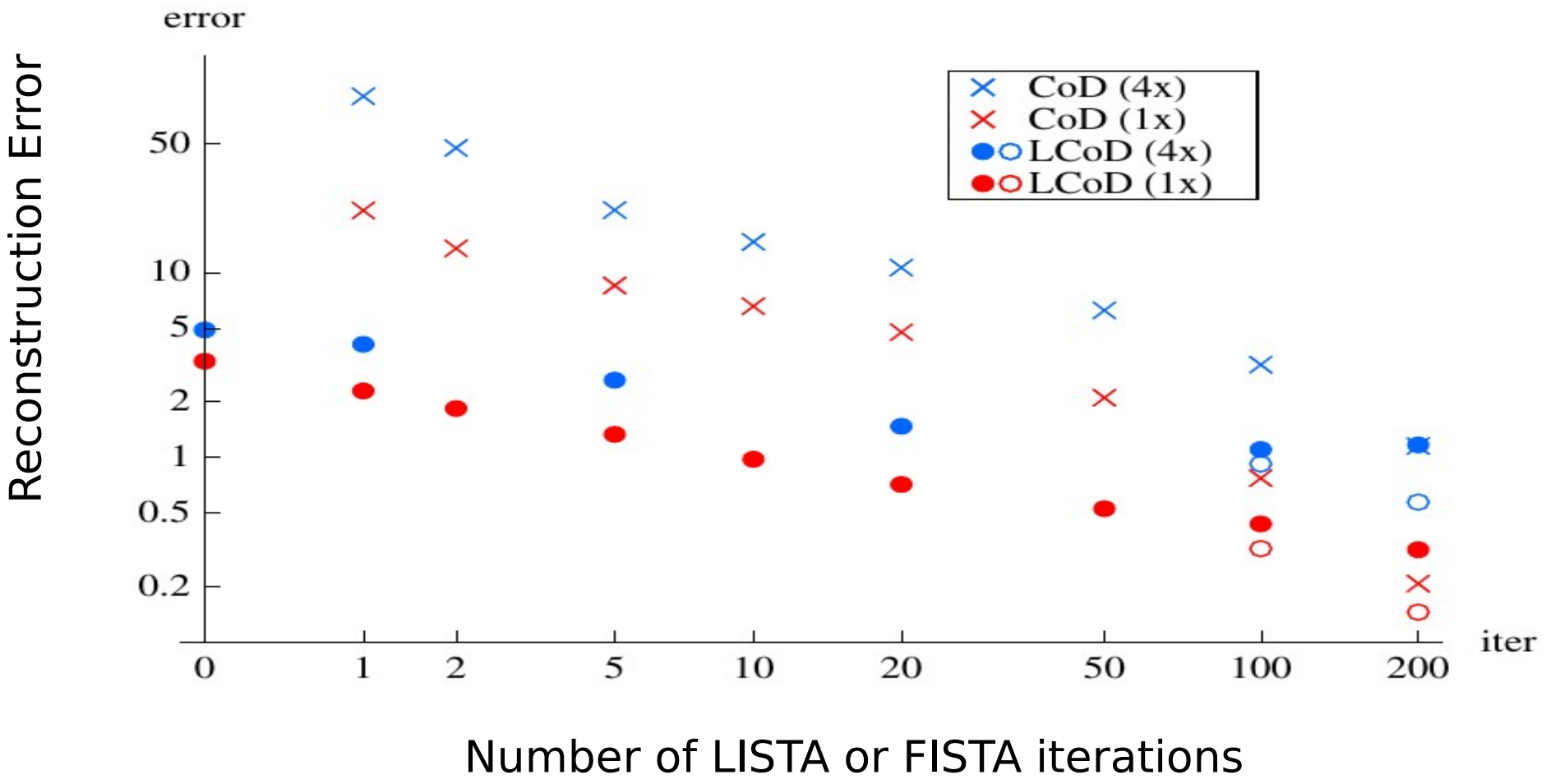
Learning ISTA (LISTA) vs ISTA/FISTA



LISTA with partial mutual inhibition matrix

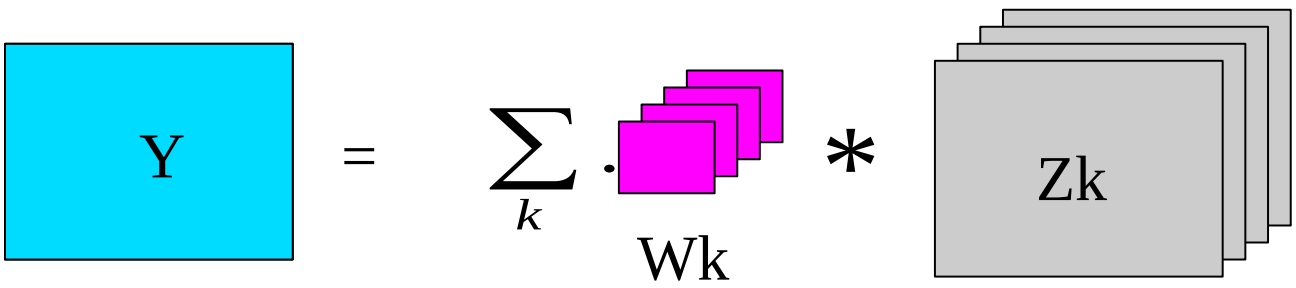


Learning Coordinate Descent (LcoD): faster than LISTA



Convolutional Sparse Coding

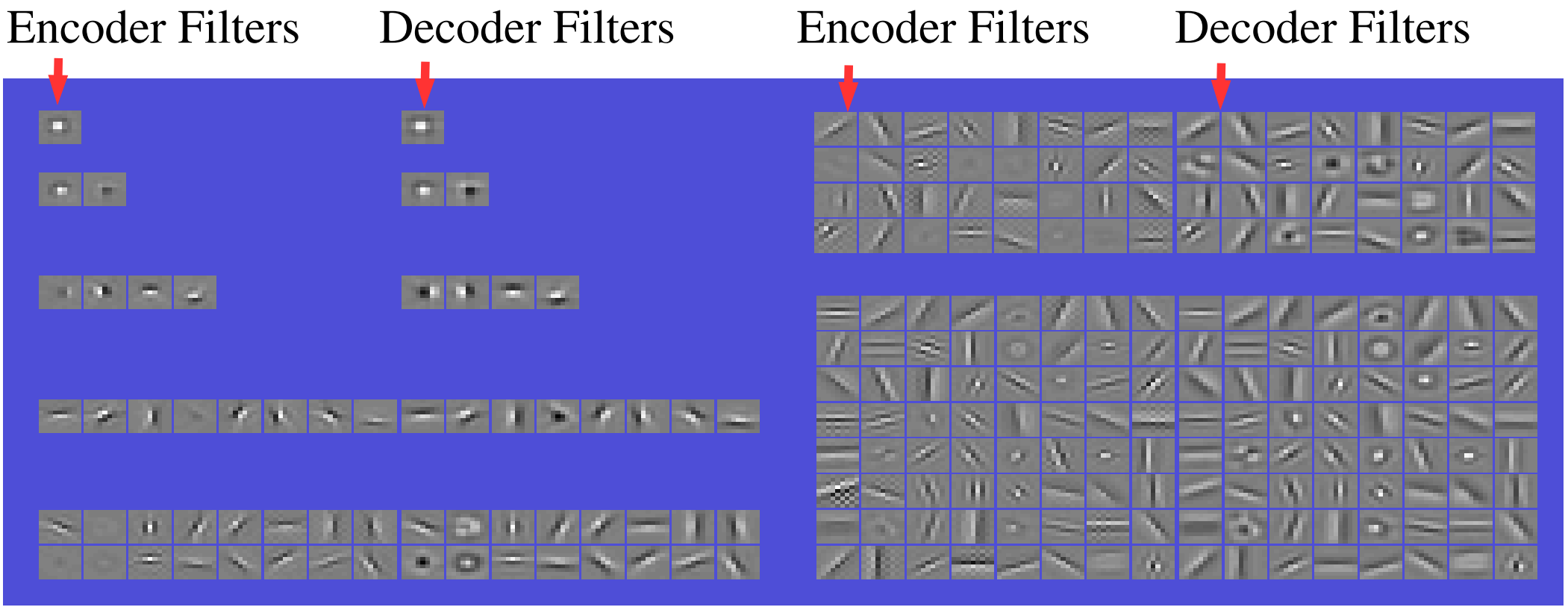
- **Replace the dot products with dictionary element by convolutions.**
  - Input Y is a full image
  - Each code component Zk is a feature map (an image)
  - Each dictionary element is a convolution kernel
- **Regular sparse coding** $E(Y,Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$
- **Convolutional S.C.** $E(Y,Z) = ||Y - \sum_k W_k * Z_k||^2 + \alpha \sum_k |Z_k|$



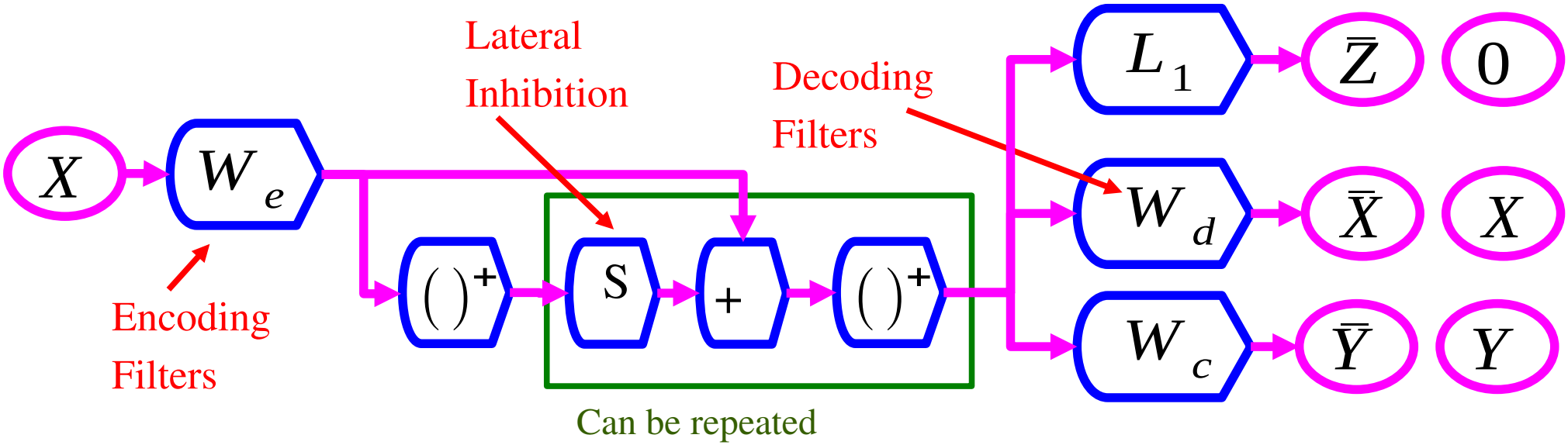
Also used in “deconvolutional networks” [Zeiler, Taylor, Fergus CVPR 2010]

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]



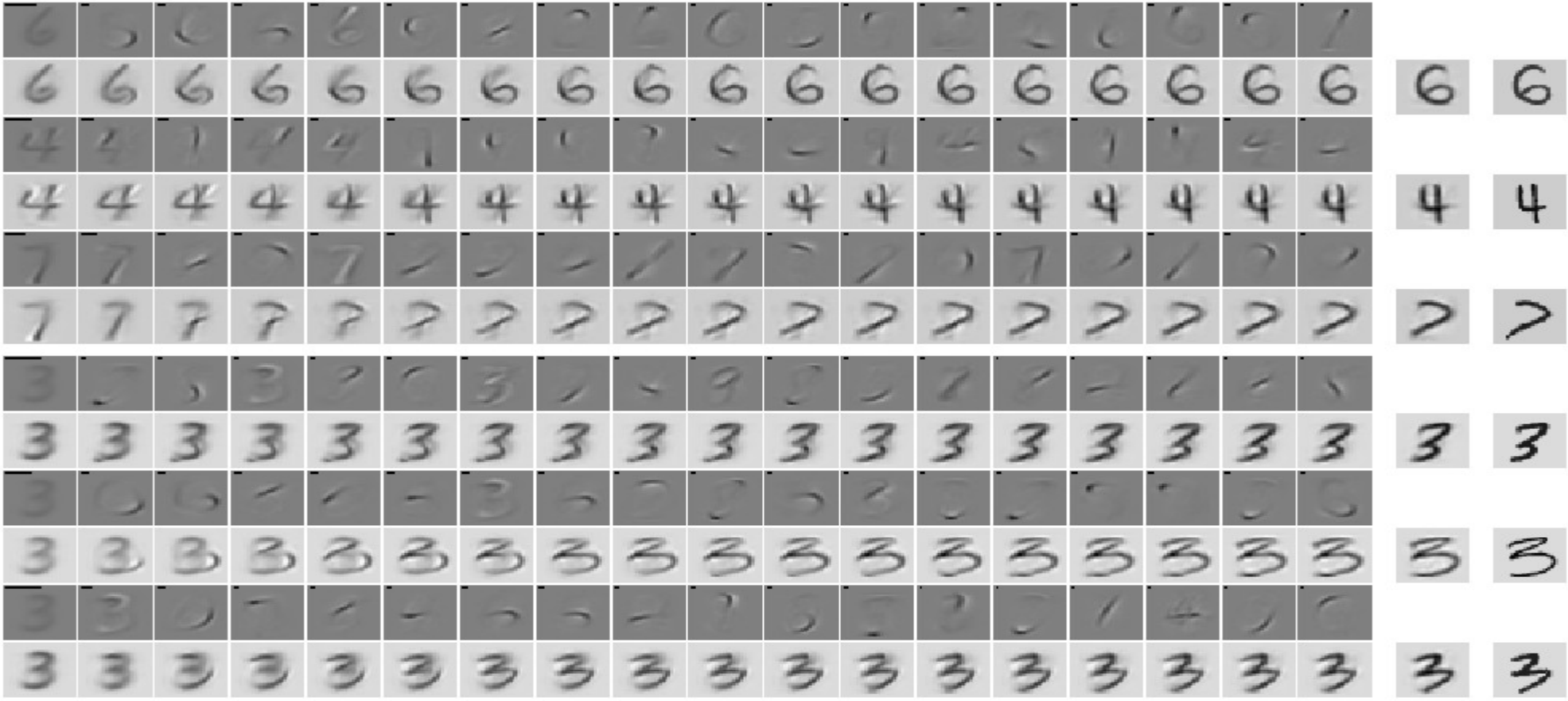
Discriminative Recurrent Sparse Auto-Encoder (DrSAE)



[Rolfe & LeCun ICLR 2013]

DrSAE Discovers manifold structure of handwritten digits

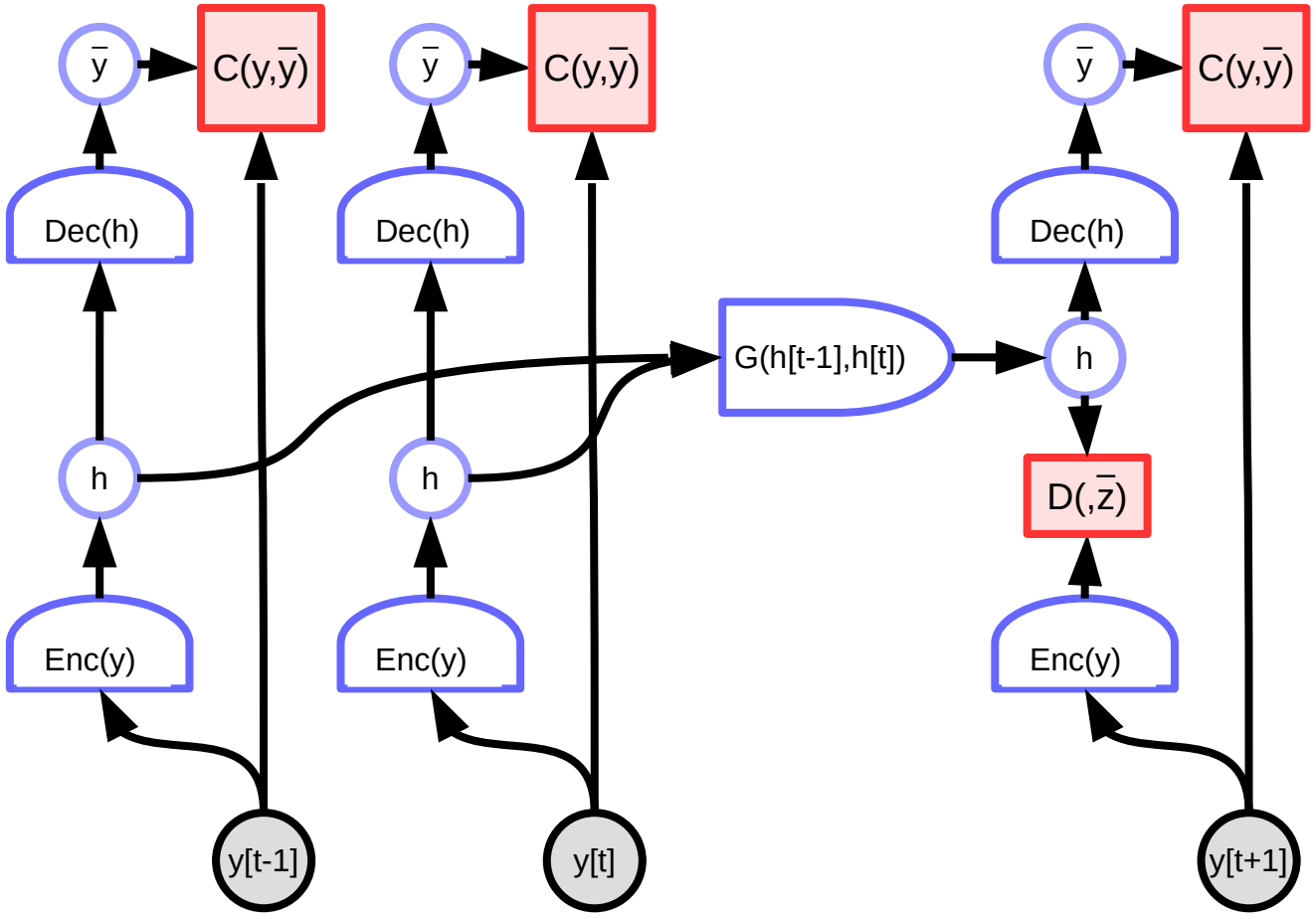
- Image = prototype + sparse sum of “parts” (to move around the



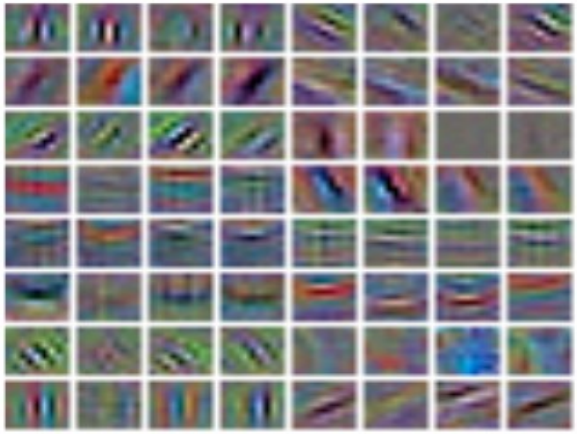
Temporal Regularization Methods

- **Favors "flatness" and predictability of the representation.**
  - Temporal invariance [Goroshin ICCV'15]
  - Linear predictability [Goroshin NIPS'16]
  - Minimal curvature

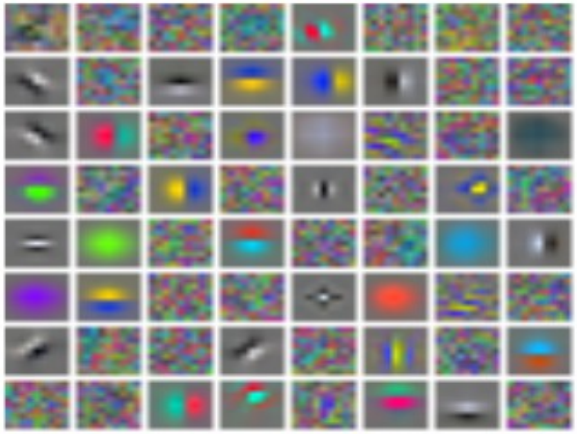
    [O. Hénaff 2019]
- **Temporal proximity is an instance of similarity graph.**
- **Decoder alleviates need for contrastive samples**



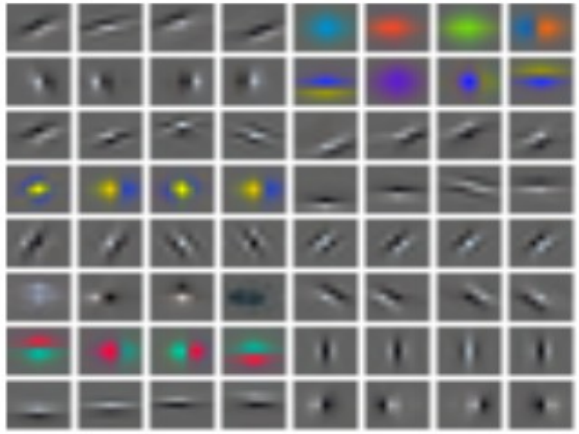
Sparse Auto-Encoder with “Slow Feature” Penalty



Supervised filters CIFAR10

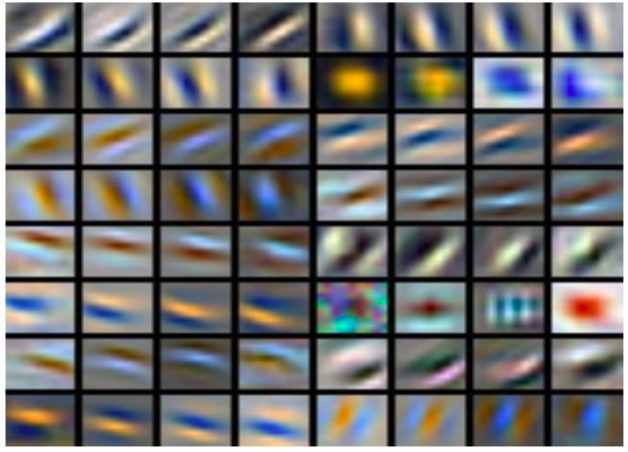


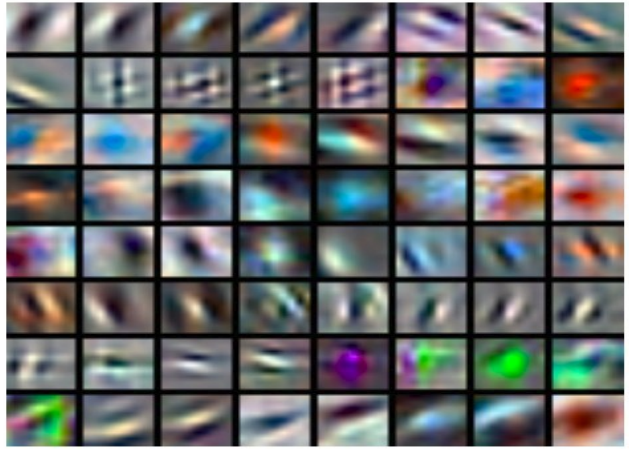
sparse conv. auto-encoder



slow & sparse convolutional AE trained on YouTube videos

Representation is pooled over non-overlapping groups of 4 features



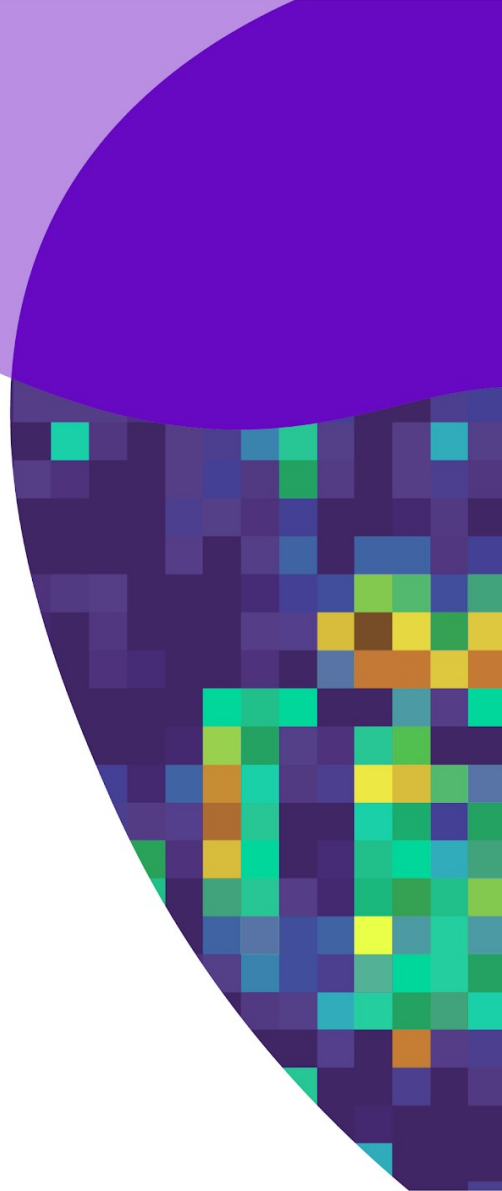


Representation is pooled over overlapping groups of 4 features

FACEBOOK AI

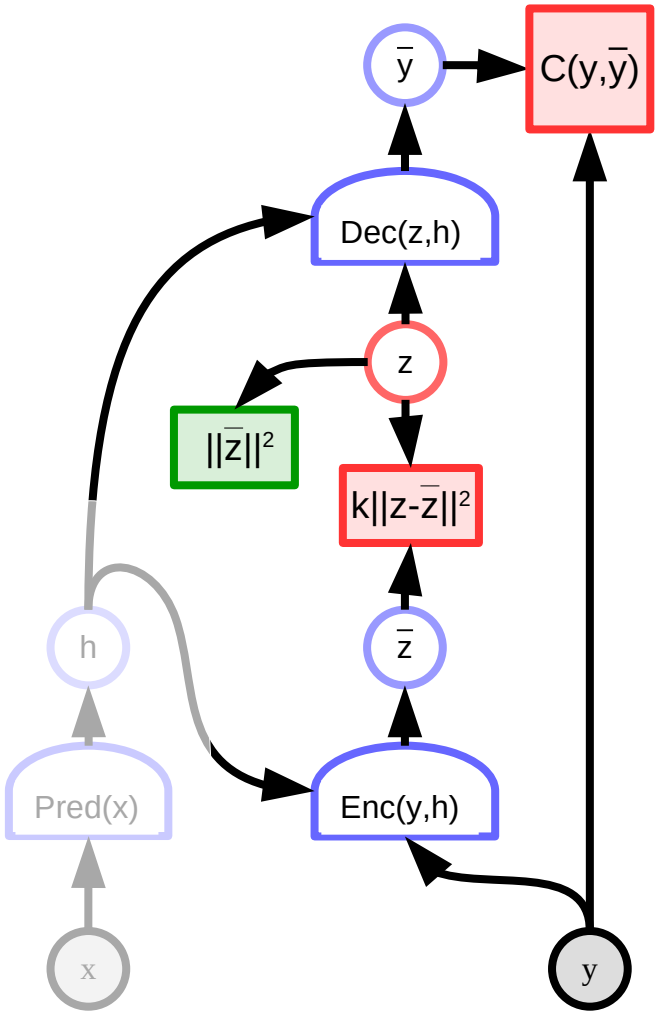
Variational AE

The energy-based approach



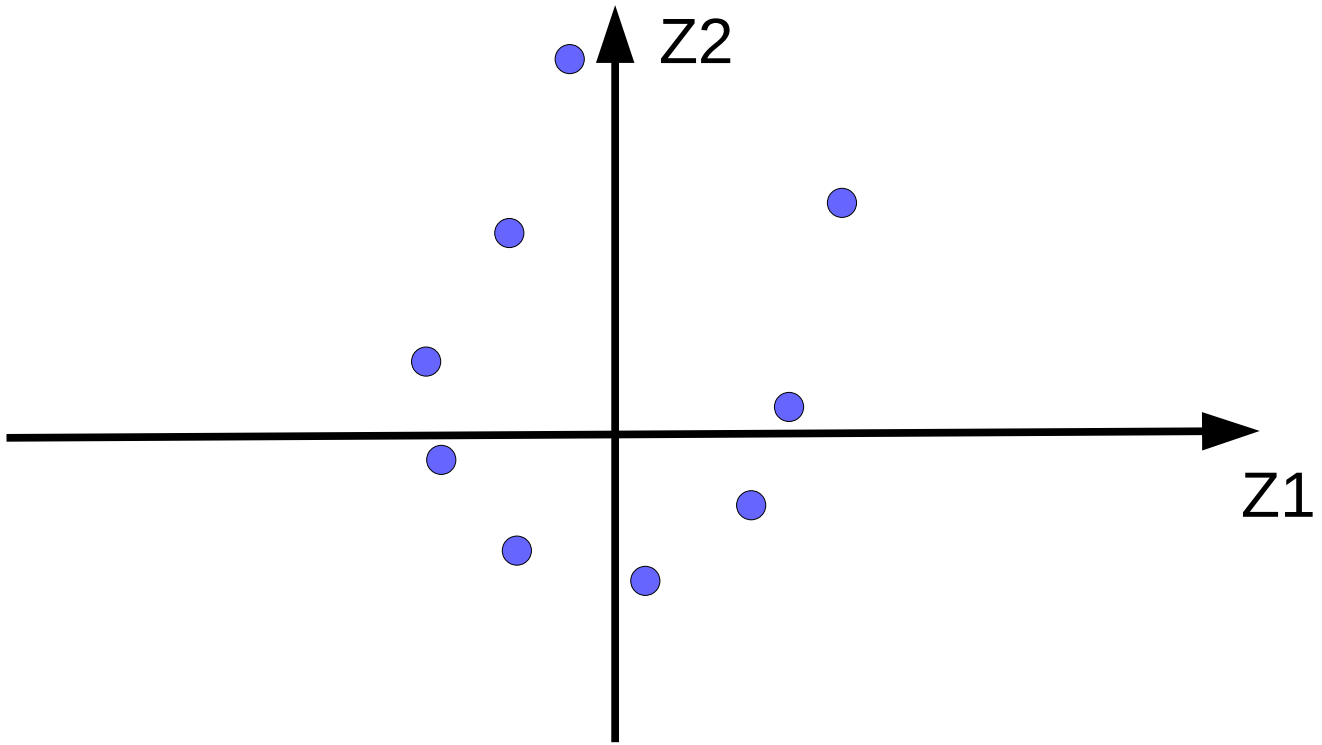
Variational Auto-Encoder

- **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.**



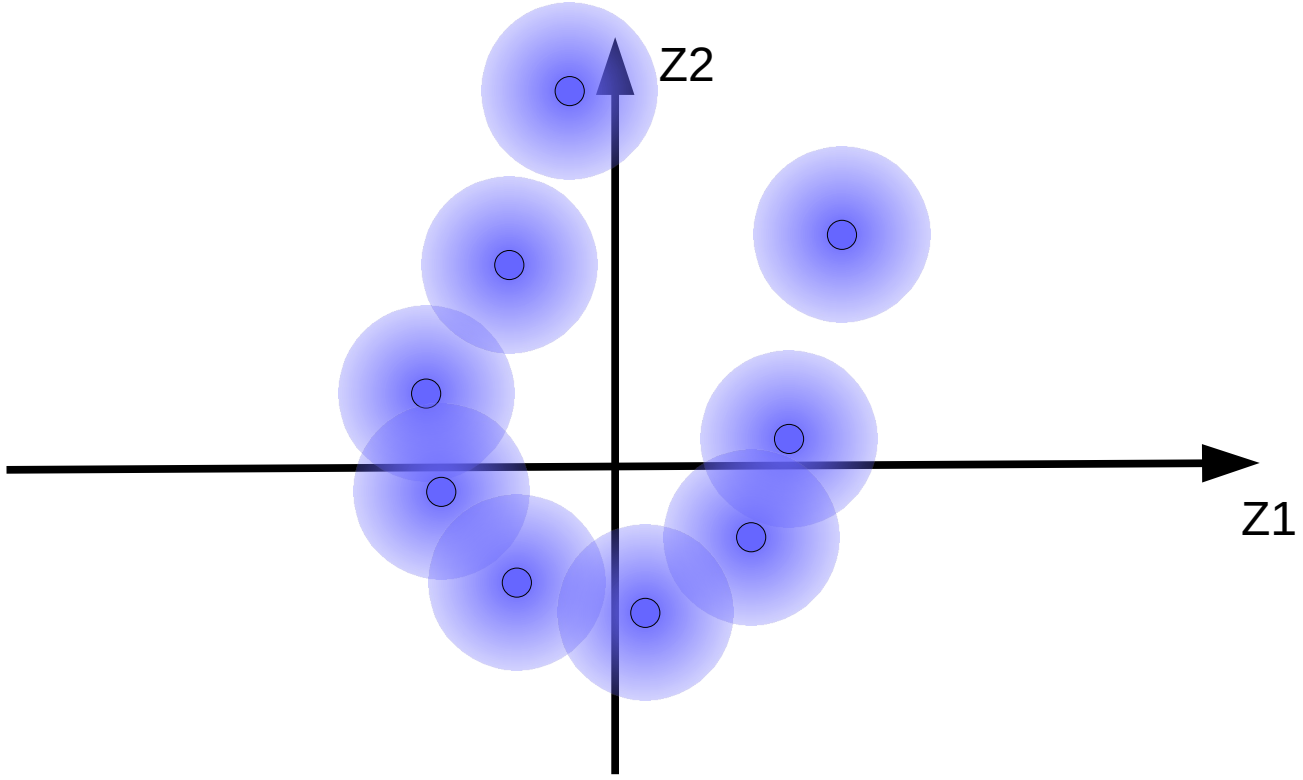
Variational Auto-Encoder

- **Code vectors for training samples**



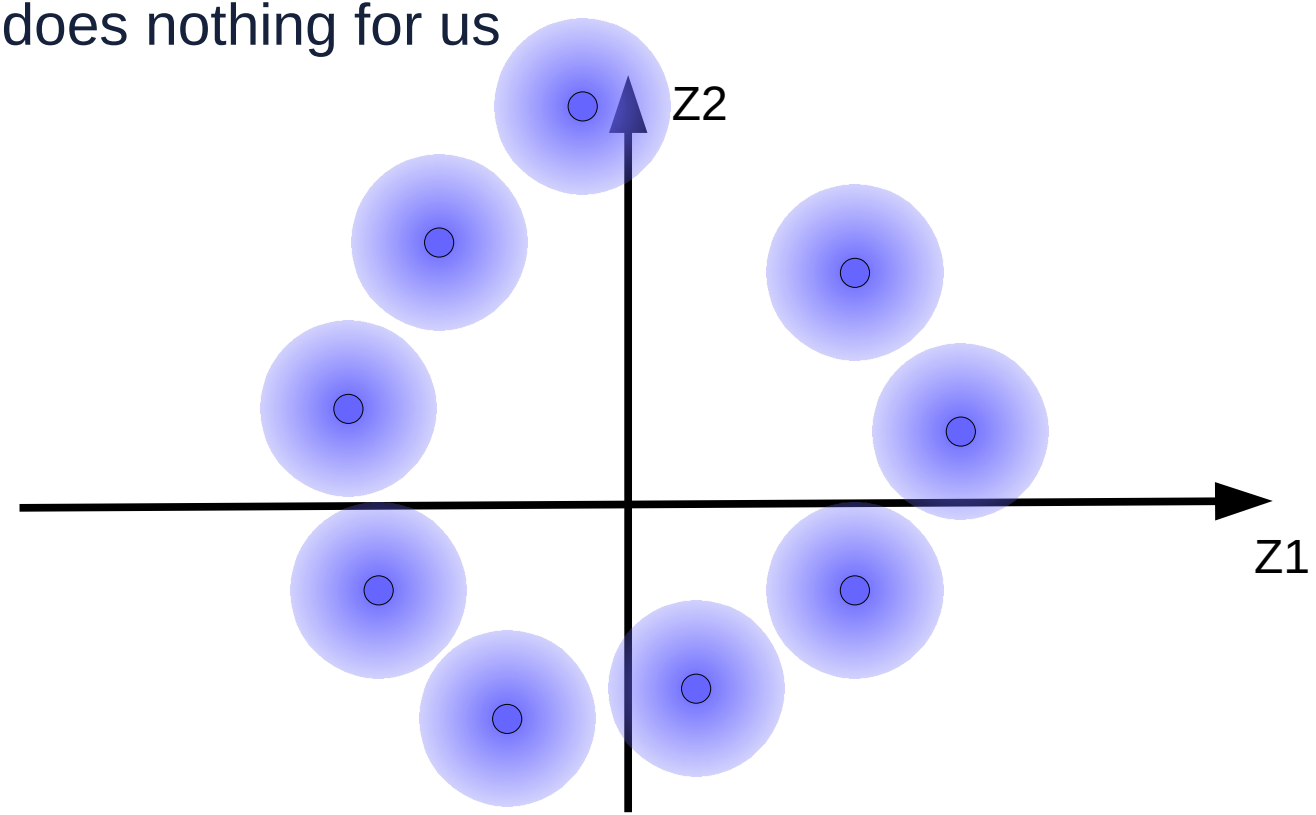
Variational Auto-Encoder

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



Variational Auto-Encoder

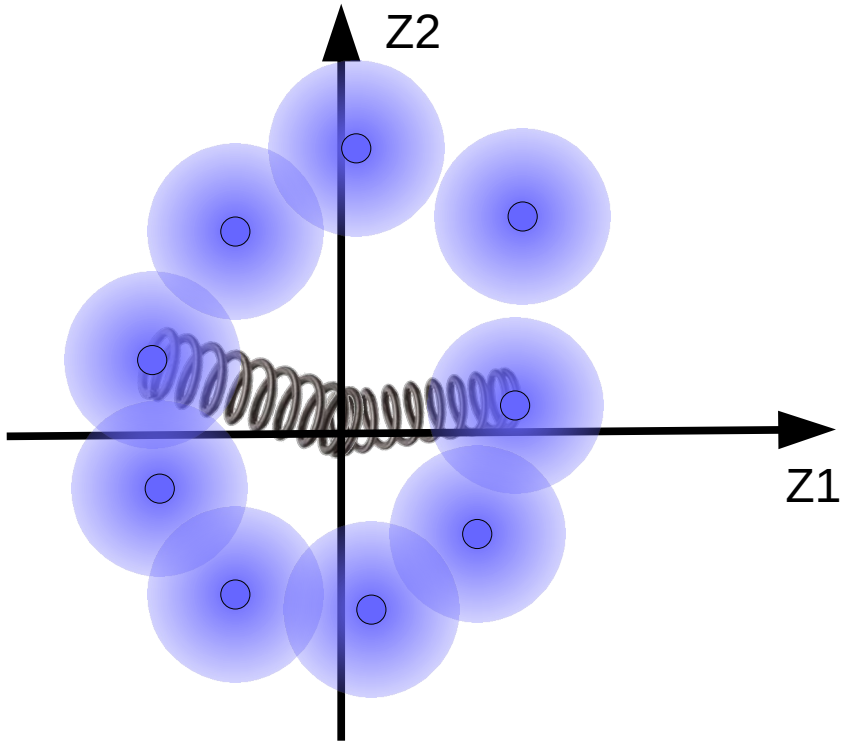
- **The code vectors want to move away from each other to minimize reconstruction error**
  - But that does nothing for us



Variational Auto-Encoder

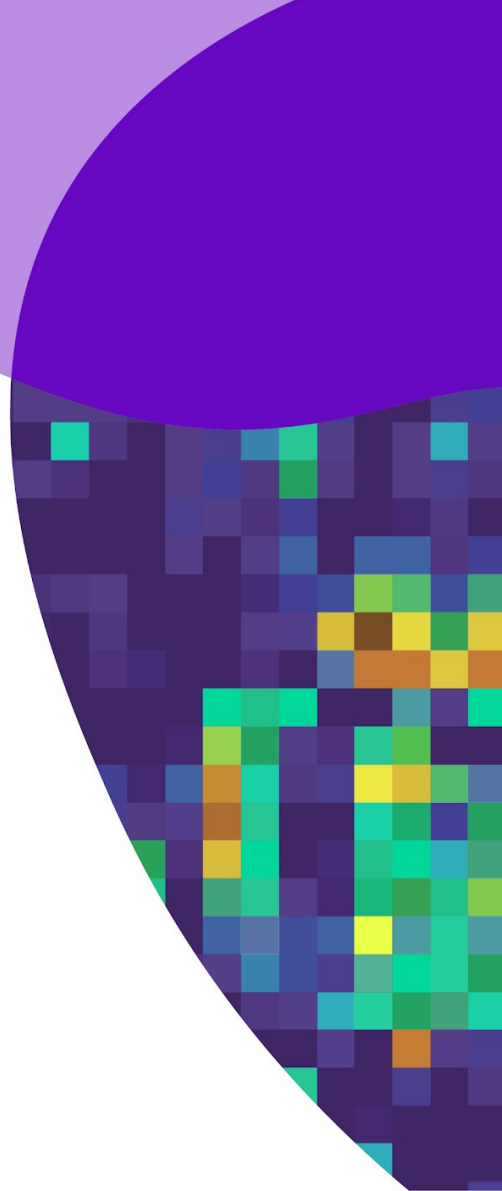
- **Attach the balls to the center with a sping, so they don't fly away**
  - Minimize the square distances of the balls to the origin
- **Center the balls around the origin**
  - Make the center of mass zero
- **Make the sizes of the balls close to 1 in each dimension**
  - Through a so-called KL term

Talia Konkle



FACEBOOK AI

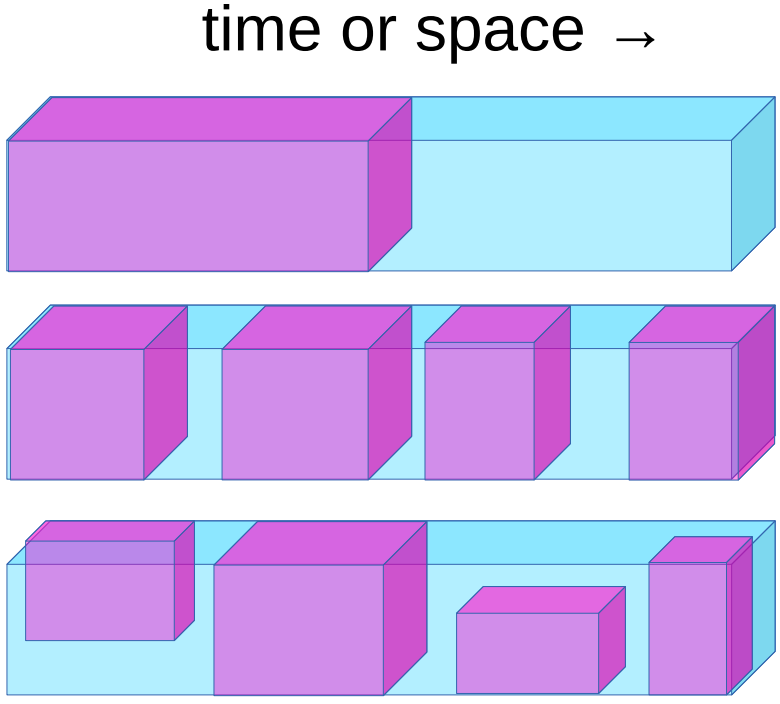
Self-Supervised Learning



Capture dependencies.
Predict everything from everything else.

Self-Supervised Learning = Filling in the Blanks

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

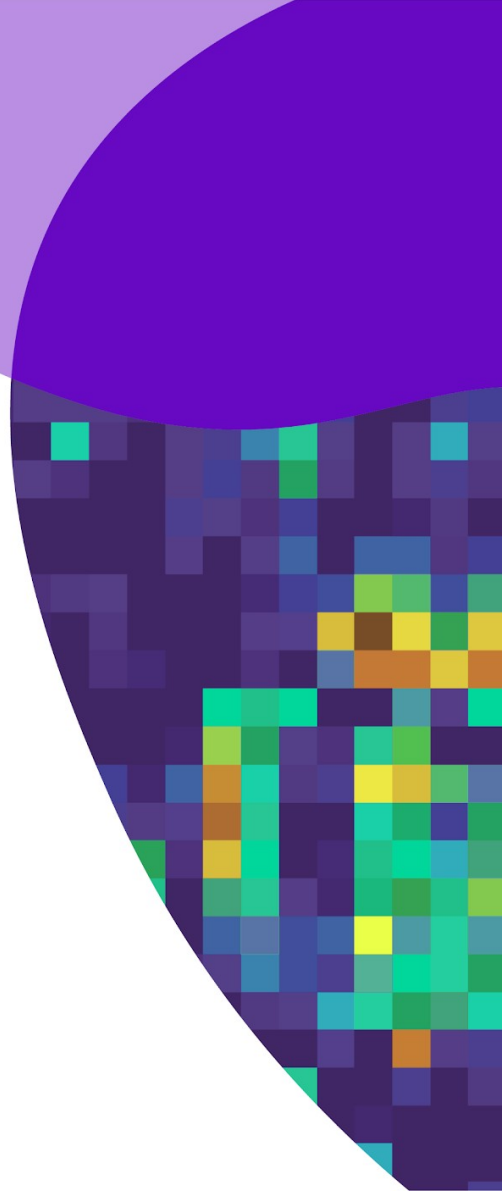
Two Uses for Self-Supervised Learning

- **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?**

FACEBOOK AI

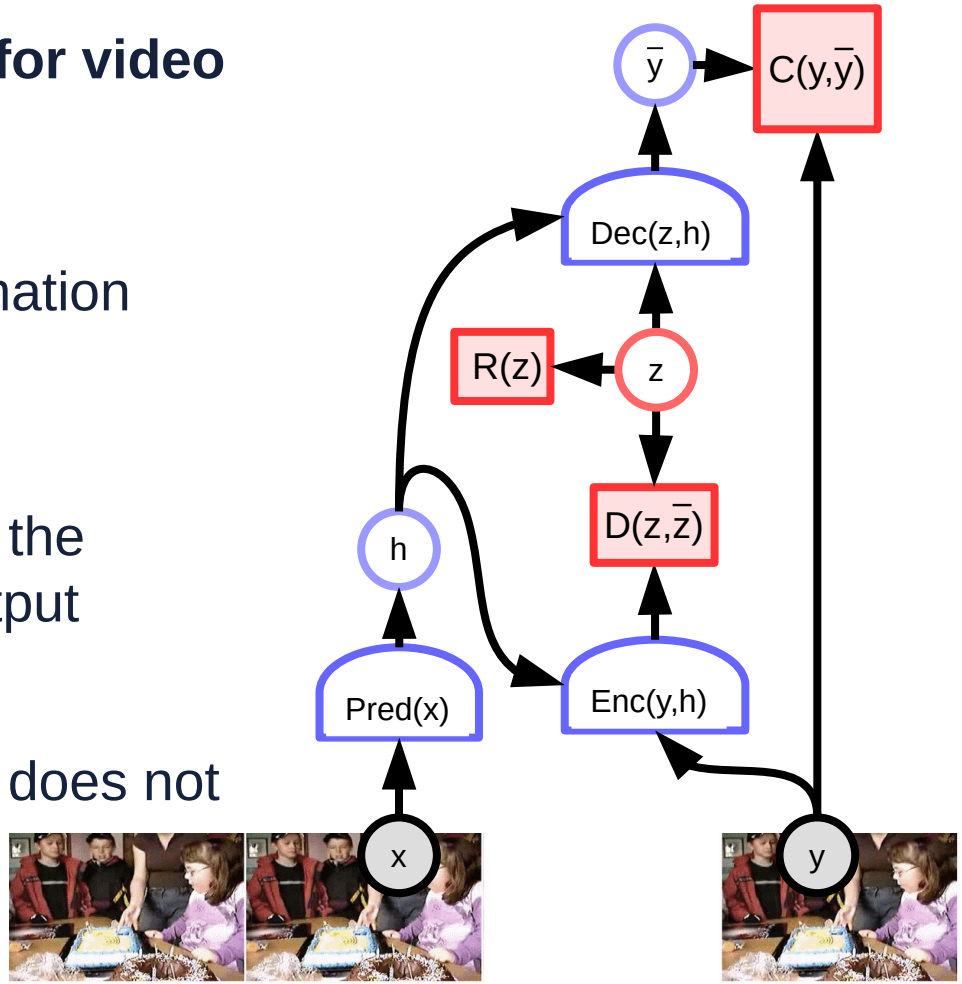
Conditional RLVEBM for forward prediction

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.

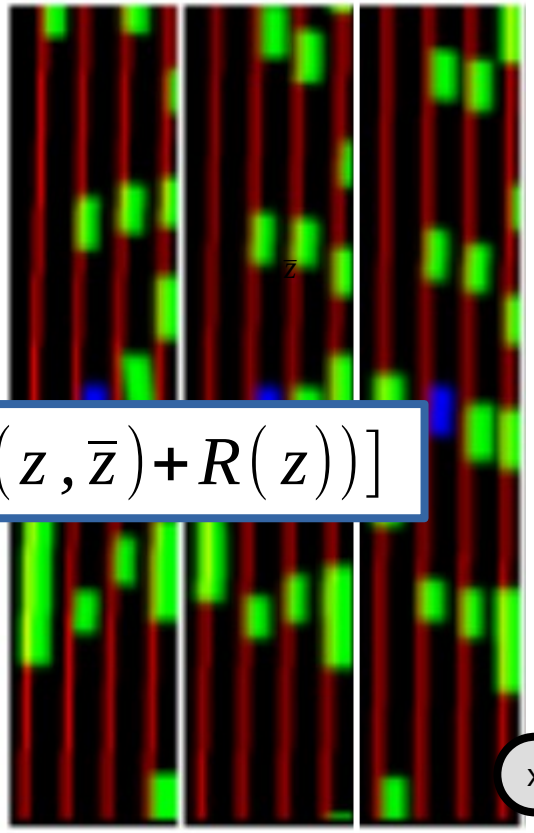


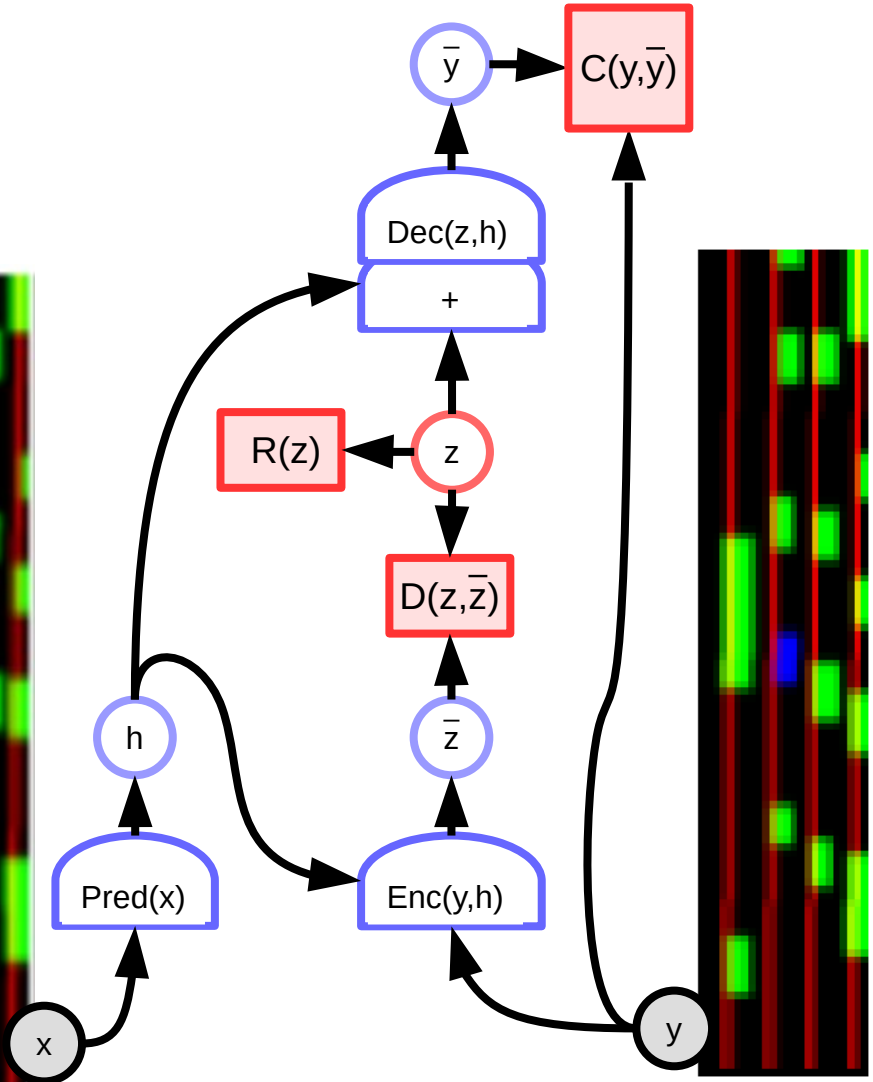
VAE + Drop Out

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

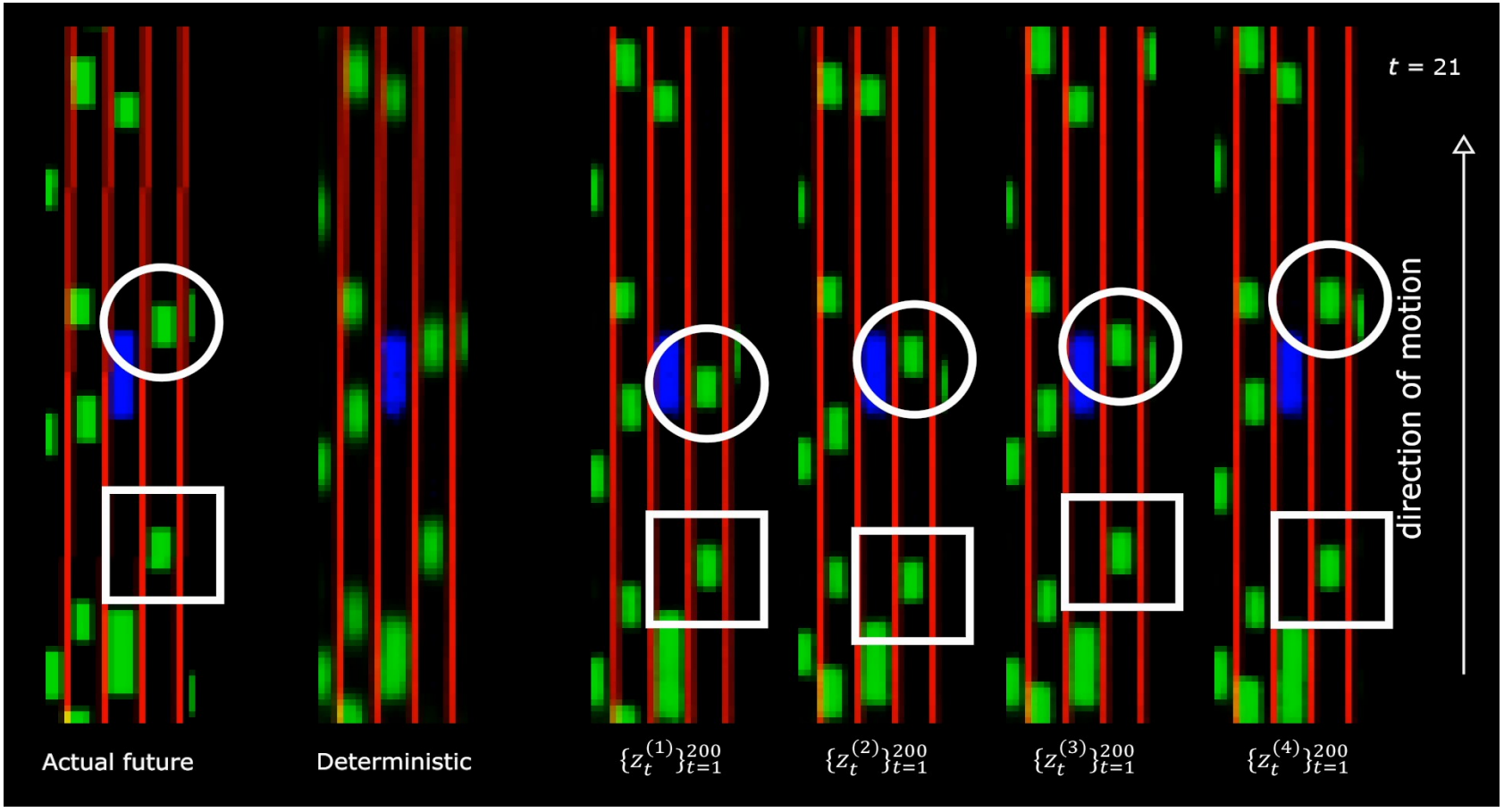
$$P(z/\bar{z}) \propto \exp[-\beta(D(z,\bar{z})+R(z))]$$

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



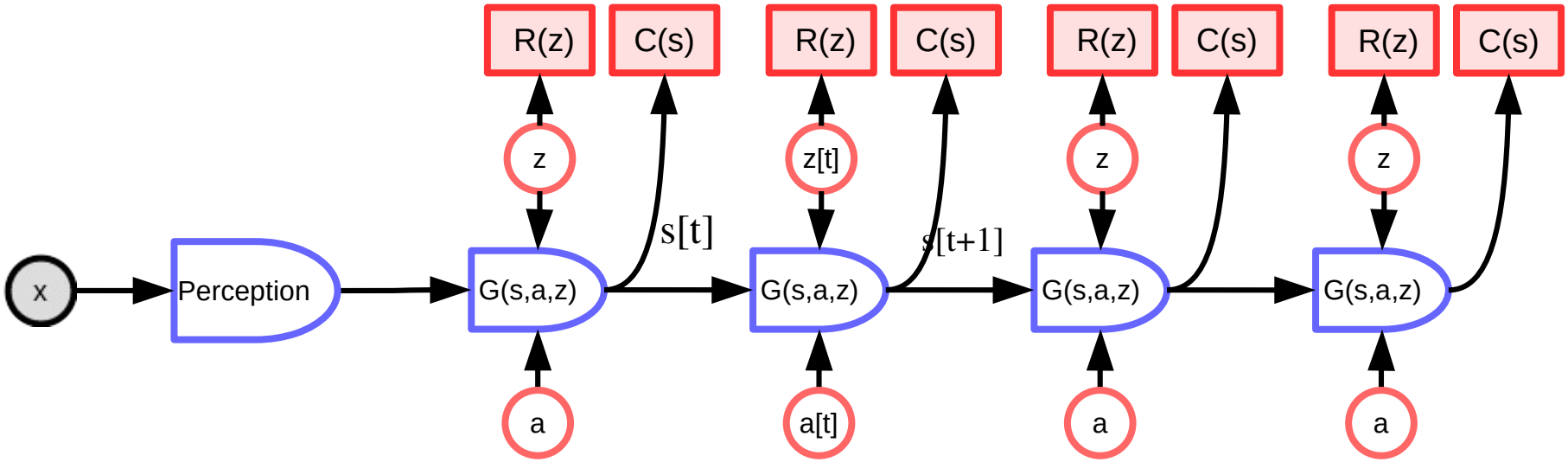


Actual, Deterministic, VAE+Dropout Predictor/encoder



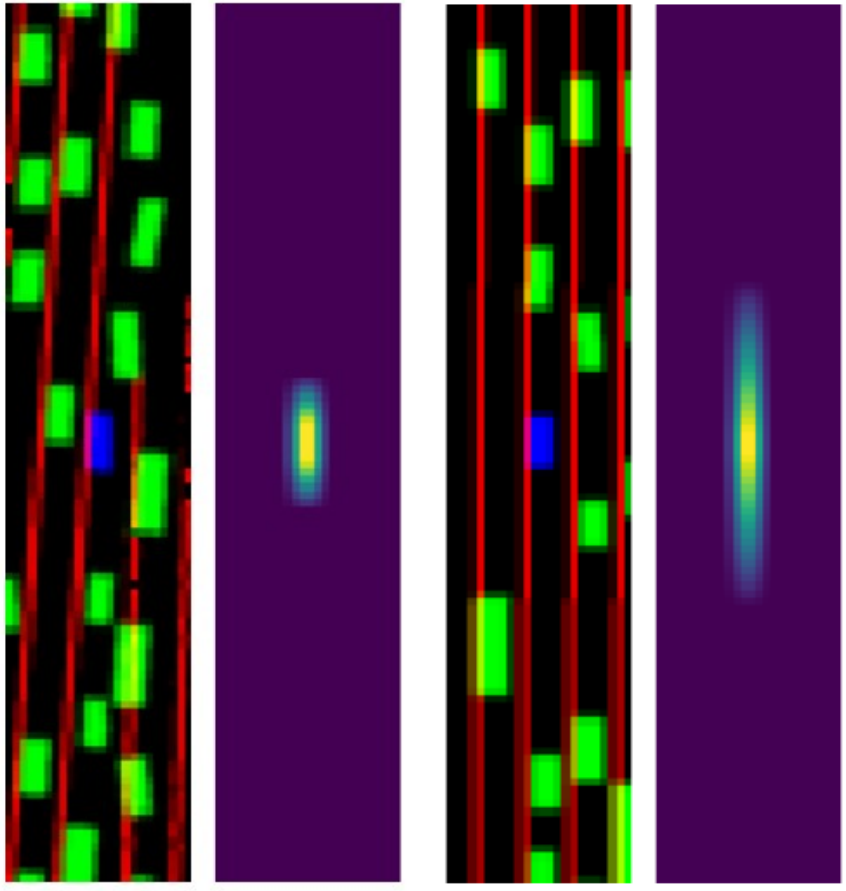
Forward Model for Model-Predictive Control

- **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]…,a[T]) = argmin Σ_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.



(a) 19.8 km/h (b) 50.3 km/h

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)**