Efficient Learning of Sparse Overcomplete Representations with an Energy-Based Model

Marc'Aurelio Ranzato

C. Poultney
S. Chopra
Y. LeCun

(NYU – Courant Institute)

CIAR Summer School – Toronto 2006

Why Extracting Features?

- Preprocessing in image analysis systems

  - Extract information from high dimensional data (e.g. compression, visualization)

  - Map data into higher dimensional where features become linearly separable (e.g. kernel methods for classification)

  - Find representation with a more “meaningful” interpretation (e.g. NMF applied to face images finds “parts”)

Why Extracting Features?

- Biological motivation
  - The brain circuitry extracts information from highly redundant sensory signals
    - **Barlow**: the goal of sensory coding is to transform the input reducing the redundancy among elements in the input stream

[“*Possible principle underlying the transformation of sensory messages*” Sensory Communication 1961]

    - **Hubel, Wiesel**: receptive fields in area V1 of visual cortex code edges at different scales, orientations and spatial locations

[“*Receptive Fields of single neurones in cat's striate cortex*” J. Physiol. 1959]

Why sparse & overcomplete codes?

Surface area (mm^2)..190,000
Thickness (mm)...2.5
Neurons/mm^3...40,000
Synapses/mm^3...7x10^8
Energy available for spikes/min.........................4x10^20 ATP
Energy for 1 spike...2.2x10^9 ATP
Average rate (spike/s/neuron)............................0.16

*"... in strongly driven visual cortex, only a small **fraction of neurons is working at any one time** – between 1 in 25 and **1 in 63**, with the latter the more probable value"*

This might be interpreted as evidence for the use of a sparse and over-complete code.

[P. Lennie "*The cost of cortical computation*" Current Biology 2003]

Why sparse & overcomplete codes?

Sparse and overcomplete representations are also efficient

- compression
- robustness to noise
- better tiling of joint space of location and frequency

Unsupervised Feature Extractor

- PCA
- Kernel-PCA [*Scholkopf, Smola, Muller - 1998*]
- Auto-encoders
- ICA [*Bell, Sejnowski - 1995*]
- NMF [*Lee, Seung - 1999*]

Unsupervised Feature Extractor

- PCA ← COMPACT LINEAR
- Kernel-PCA [*Scholkopf, Smola, Muller - 1998*]
- Auto-encoders
- ICA [*Bell, Sejnowski - 1995*]
- NMF [*Lee, Seung - 1999*]

Unsupervised Feature Extractor

- PCA
- Kernel-PCA [*Scholkopf, Smola, Muller - 1998*]
- Auto-encoders
- ICA [*Bell, Sejnowski - 1995*]
- NMF [*Lee, Seung - 1999*]

COMPACT
+
NON LINEAR

Unsupervised Feature Extractor

- PCA
- Kernel-PCA [*Scholkopf, Smola, Muller - 1998*]
- Auto-encoders
- ICA [*Bell, Sejnowski - 1995*]
- NMF [*Lee, Seung - 1999*]

UNDER-COMPLETE
LINEAR

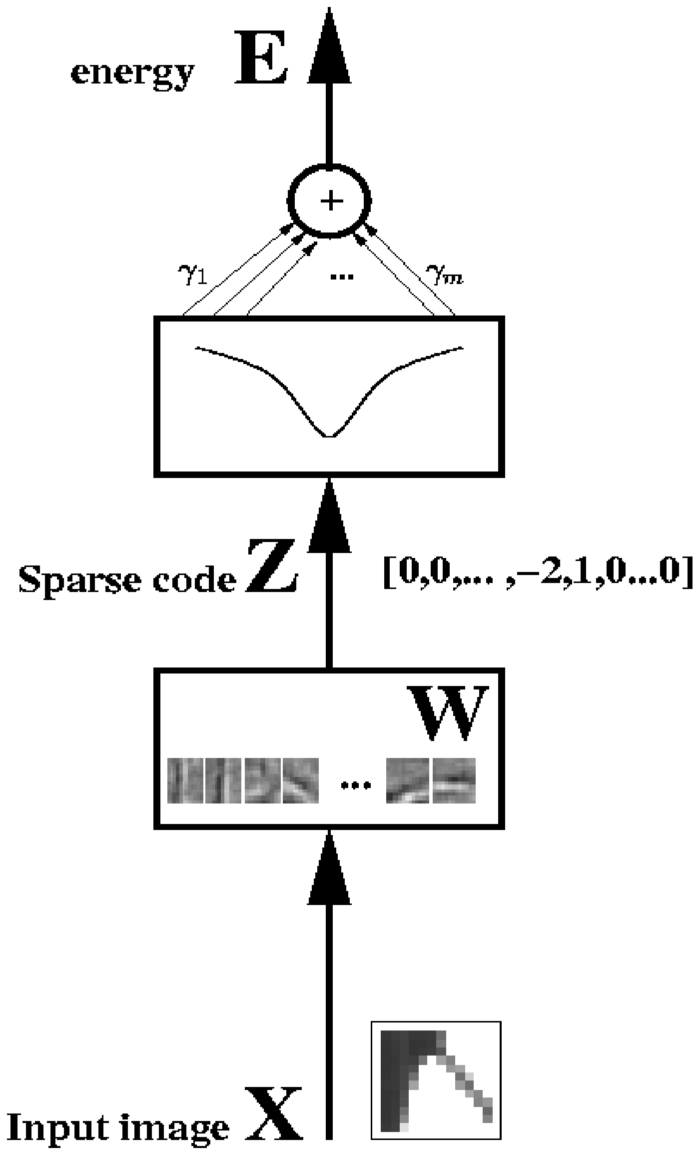
Unsupervised Feature Extractor

- PCA
- Kernel-PCA [*Scholkopf, Smola, Muller - 1998*]
- Auto-encoders
- ICA [*Bell, Sejnowski - 1995*]
- NMF [*Lee, Seung – 1999*]

NOT LEARNED FROM DATA

- Gabor-Wavelets [*Simoncelli, Freeman, Adelson, Heeger - 1998*]

Product of Experts



- non causal model
- system has energy: $E=\sum_i \gamma_i \log(1+z_i^2)$
- Loss is: $L=\sum_j E(X_j) + \log\int_X e^{-E(X)}$
- Update weights:

$$\frac{dL}{dw_{ij}}=\left\langle\frac{dE}{dw_{ij}}\right\rangle_{p^0}-\left\langle\frac{dE}{dw_{ij}}\right\rangle_{p^\infty}$$

replaced by Contrastive Divergence:

$$\Delta w_{ij}=\left\langle\frac{dE}{dw_{ij}}\right\rangle_{p^0}-\left\langle\frac{dE}{dw_{ij}}\right\rangle_{p^1}$$

[Teh, Welling, Osindero, Hinton "*Energy-Based Models for Sparse Overcomplete Representations*" JMLR 2003]

Product of Experts

STRENGTHS

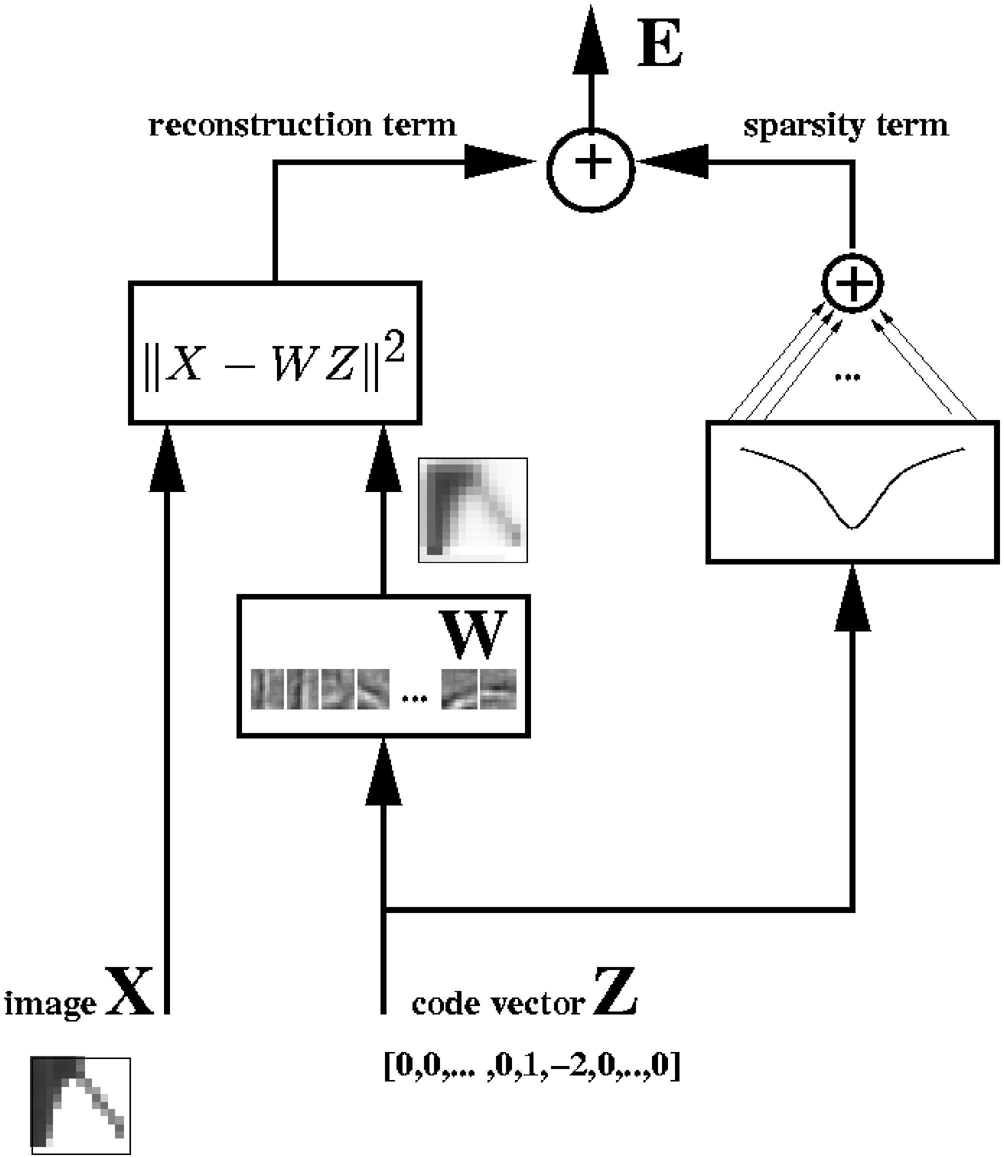
- Fast and easy inference
- Works for any number of code units
- Can be extended to multi-layer architectures

LIMITATIONS

- Training is expensive because of sampling
- Preprocessing is necessary to improve convergence
- How can we reconstruct a patch given its code?

[Teh, Welling, Osindero, Hinton "*Energy-Based Models for Sparse Overcomplete Representations*" JMLR 2003]

Reverse Approach



In energy terms:

$$E(X,Z,W) = \|X - WZ\|^2 + \lambda \sum S(Z_i)$$

Learning:

$$\hat{W} = argmin_W \langle min_Z E(X,Z,W) \rangle$$

1) find optimal Z, given X and W (inference, E-step)
2) update W given X and Z (learning, M-step)

But, the norm of each filter in W must be normalized by:

$$\left[\frac{\langle Z_i \rangle^2}{\sigma^2_{goal}}\right]^{\alpha}$$

[Olshausen, Field "*Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?*" VR 1996]

Reverse Approach

STRENGTHS

- Simple learning algorithm
- Straightforward probabilistic interpretation
- Neural circuitry *might* work on the same principle of sparsity...

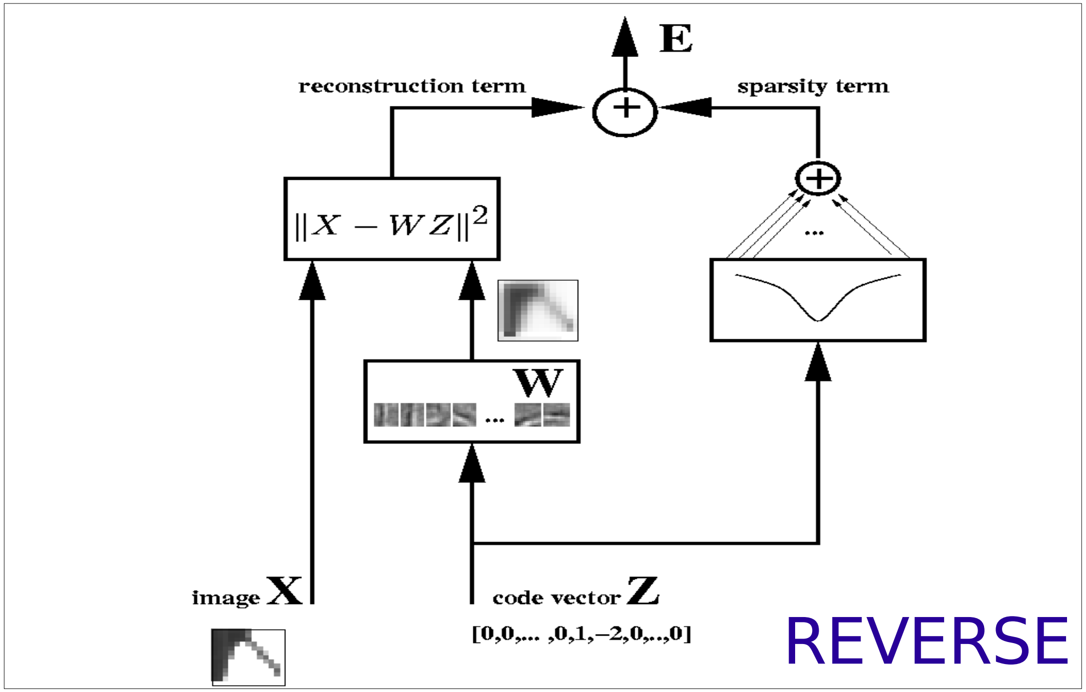
LIMITATIONS

- Normalization tweak
- Inference is expensive
- Preprocessing is required to get convergence
- A hierarchical approach might be needed
- Do real causes independently and linearly mix?
  No, they occlude one another.

[Olshausen, Field "*Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?*" VR 1996]

Combining two approaches

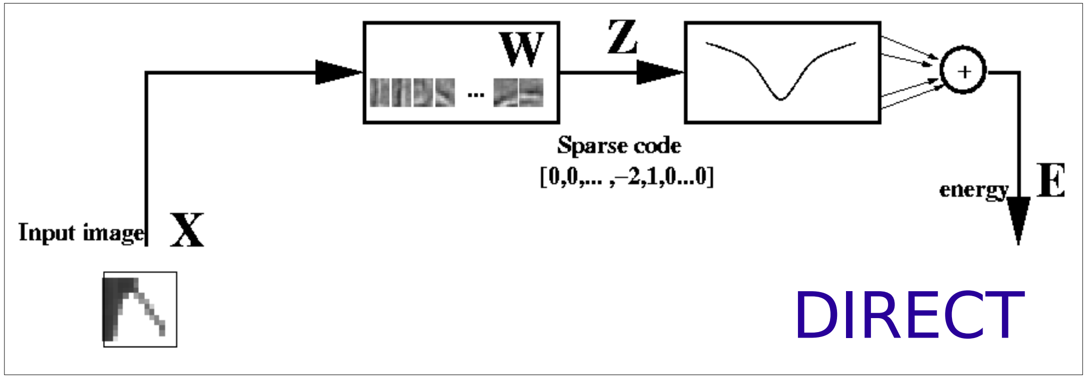
- no normalization
- no slow inference
- no preprocessing



- easy learning

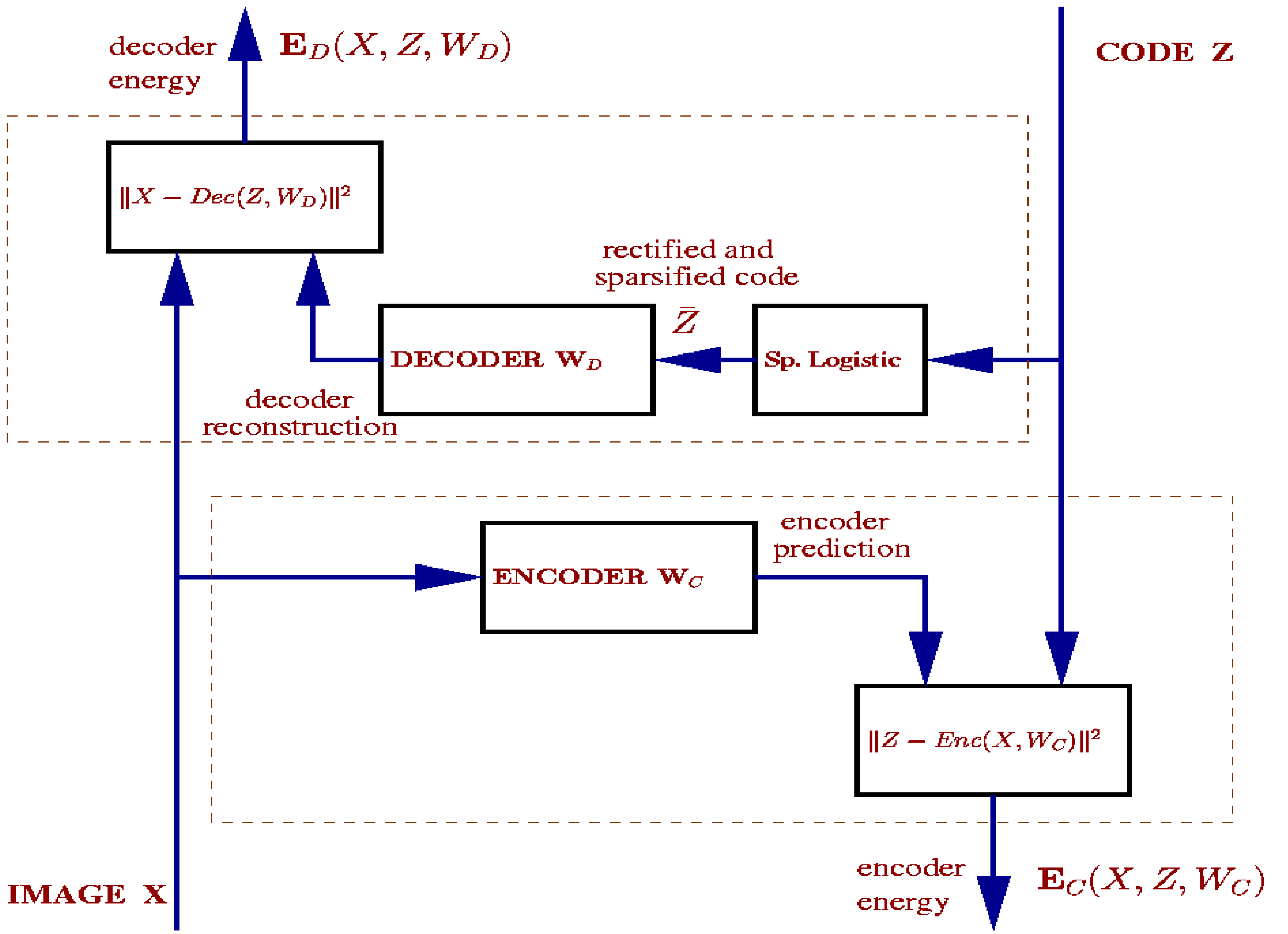
+

- no difficult learning
- no sampling
- no preprocessing

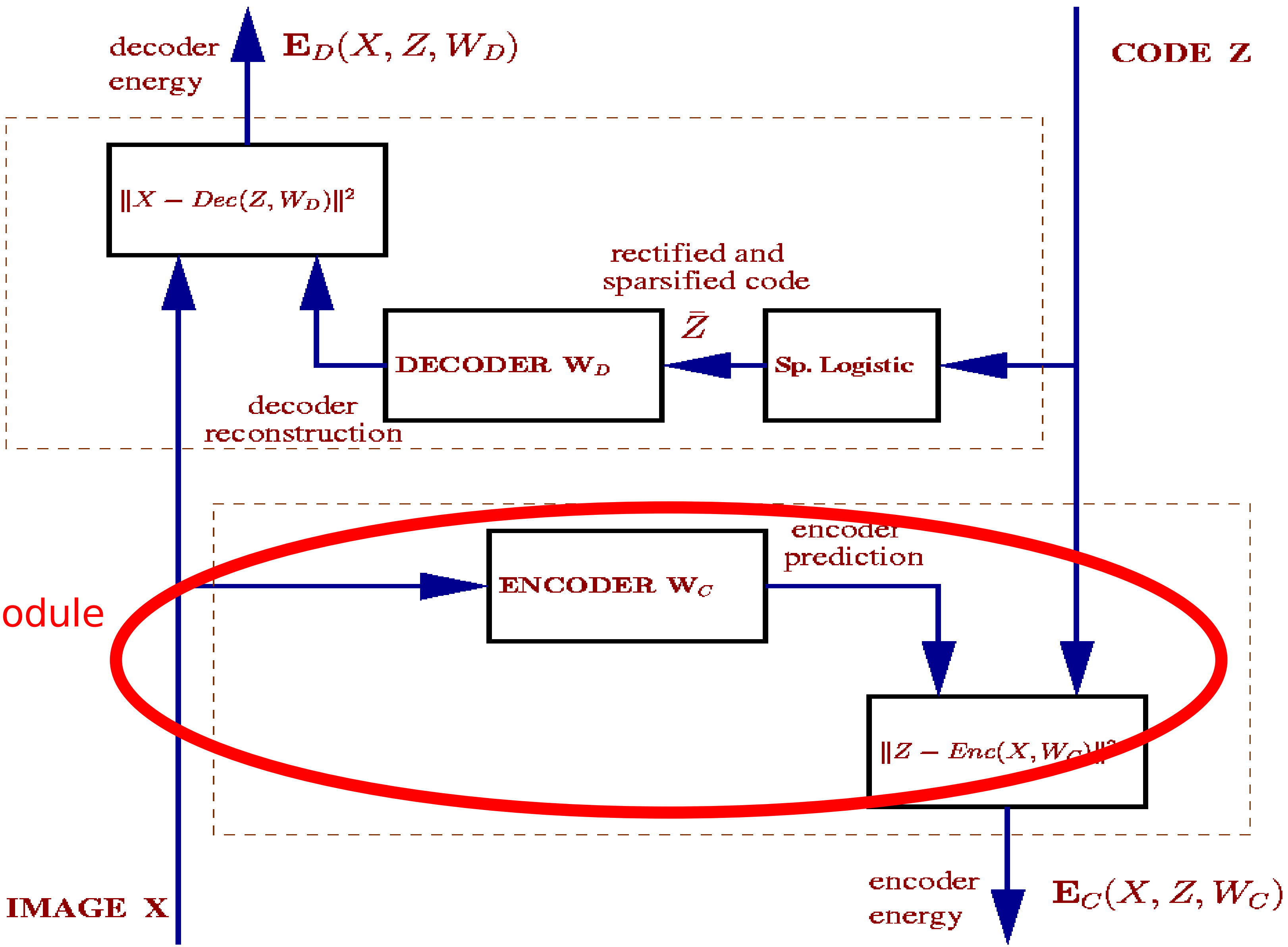


- fast inference

EBM for Sparse Representations



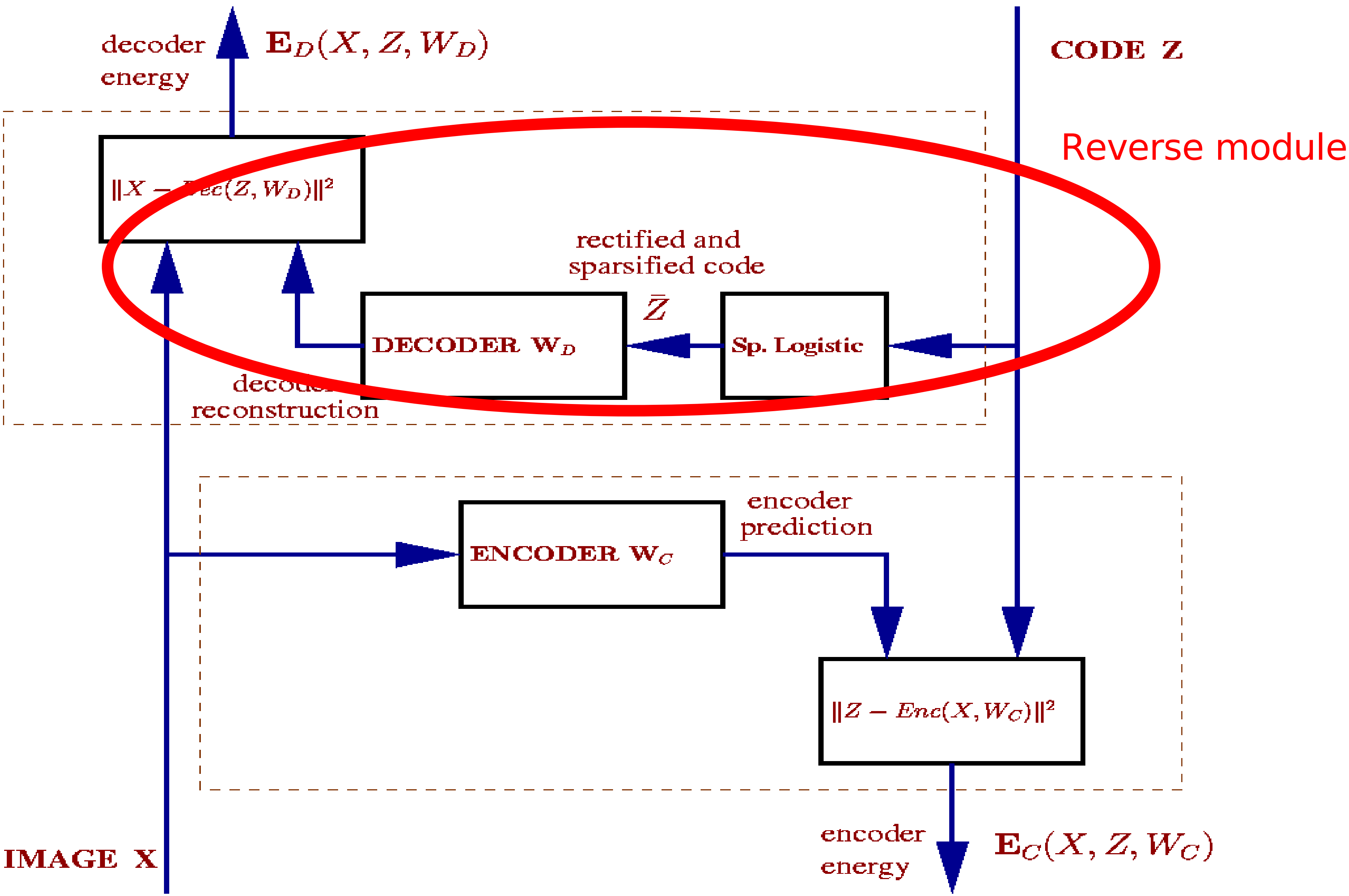
EBM for Sparse Representations



Direct module

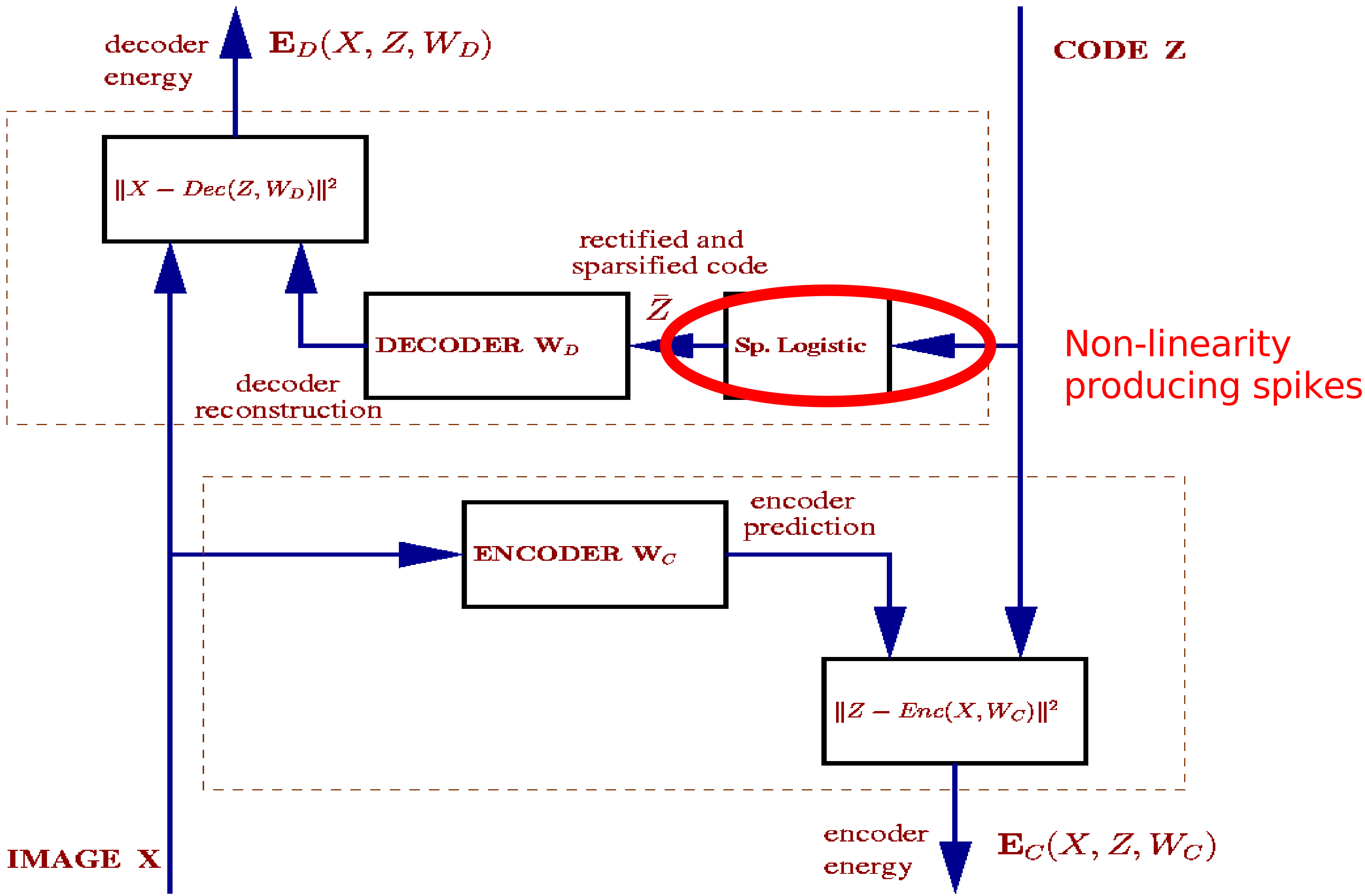
[Teh, Welling, Osindero, Hinton “*Energy-Based Models for Sparse Overcomplete Representations*” JMLR 2003]

EBM for Sparse Representations



[Olshausen, Field "*Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?*" VR 1996]

EBM for Sparse Representations



[Pillow, Paninski, Uzzell, Simoncelli, Chichilnisky *"Prediction and decoding of RGC responses with a probabilistic spiking model"* J.Neurosc. 05]

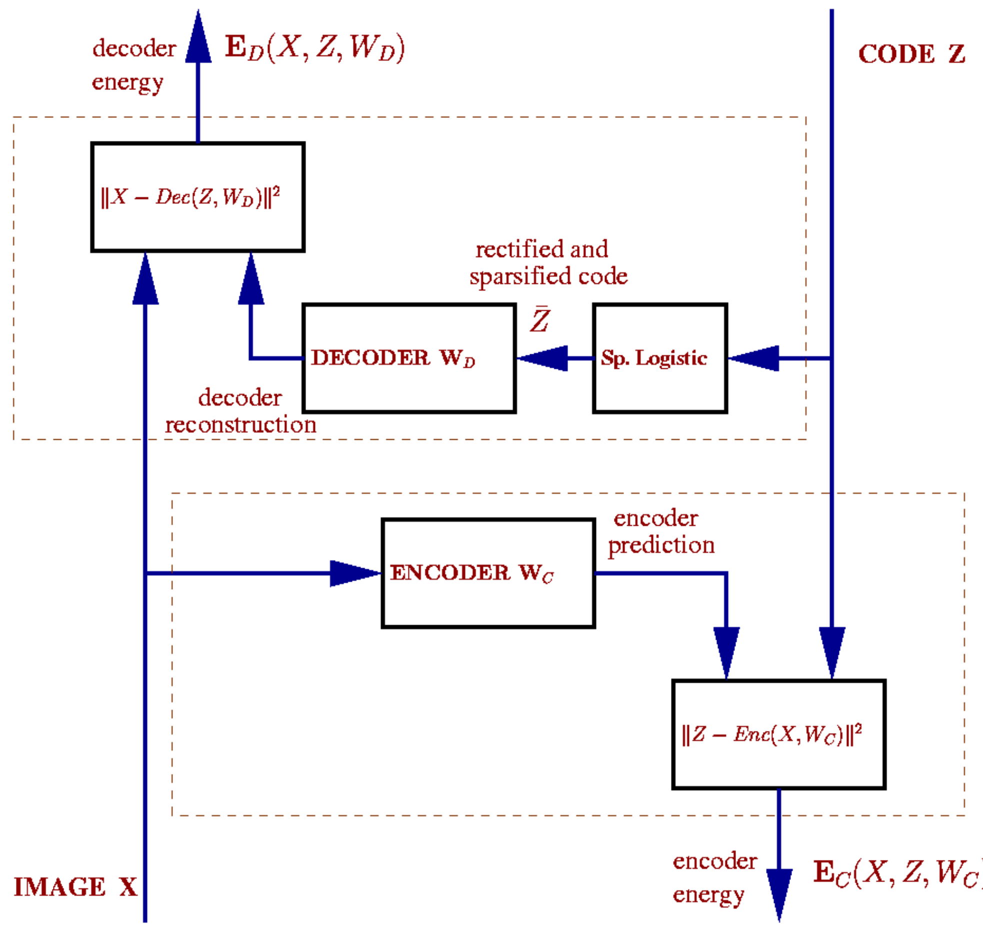
EBM for Sparse Representations

$$P(X,Z|W_c,W_d) \;\alpha\; \exp(-\beta E(X,Z,W_c,W_d))$$

$$E(X,Z,W_c,W_d) = E_C(X,Z,W_c)+E_D(X,Z,W_d)$$

$$E_C(X,Z,W_c) = \frac{1}{2}\|Z-W_c X\|^2 = \frac{1}{2}\sum(z_i-W_c^i X)^2$$

$$E_D(X,Z,W_d) = \frac{1}{2}\|X-W_d \bar{Z}\|^2 = \frac{1}{2}\sum(x_i-W_d^i \bar{Z})^2$$



Inference & Learning

- *Inference*

$$\tilde{Z} = argmin_Z \; E(X,Z,W) = argmin_Z \left[E_C(X,Z,W) + E_D(X,Z,W) \right]$$

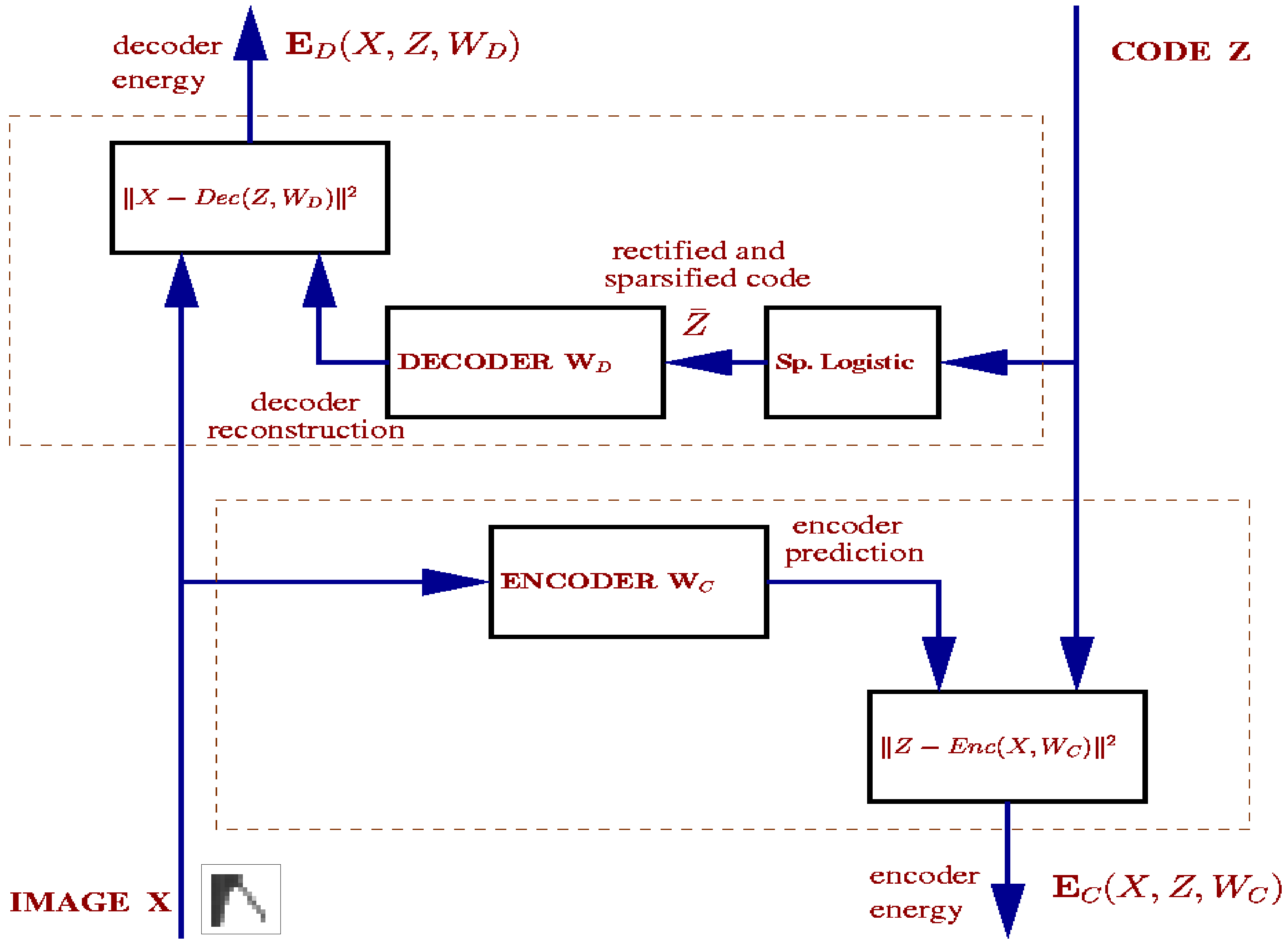
  - let Z(0) be the encoder prediction
  - find code which minimizes total energy
  - gradient descent optimization

- *Learning*

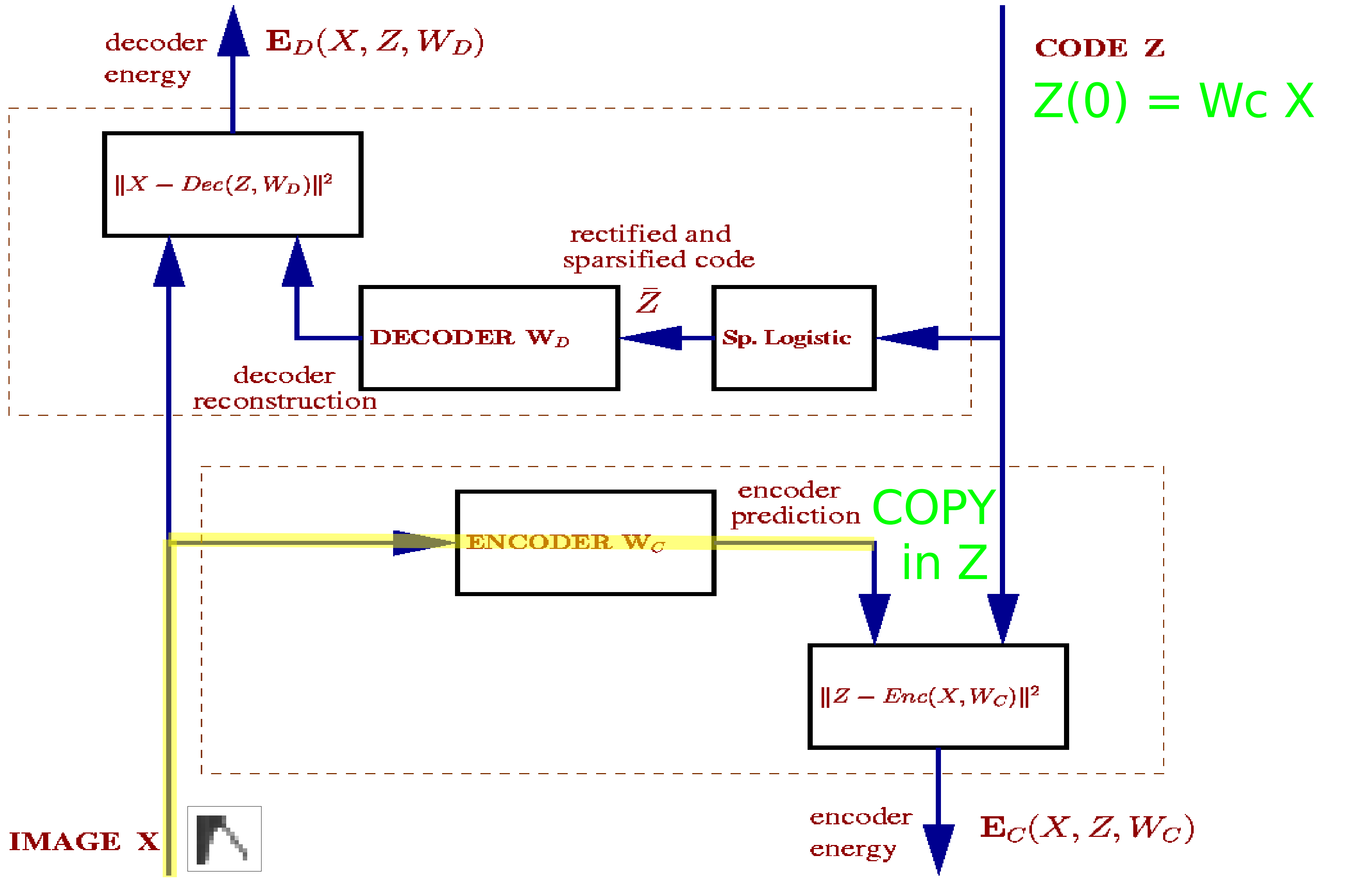
$$W \leftarrow W - \partial E(X,\tilde{Z},W) / \partial W$$

  - using the optimal code, minimize E w.r.t. the weights W
  - gradient descent optimization

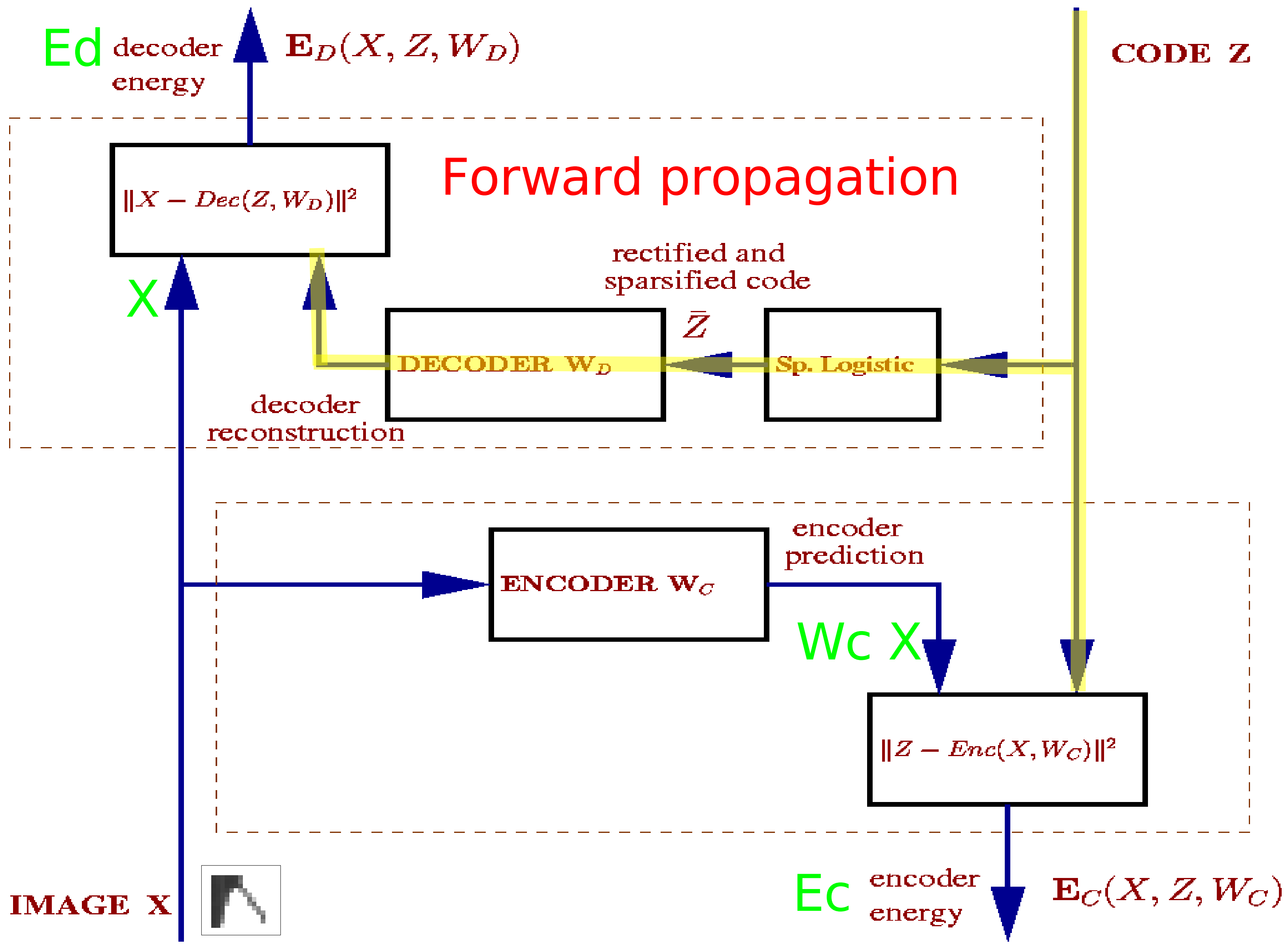
Inference & Learning



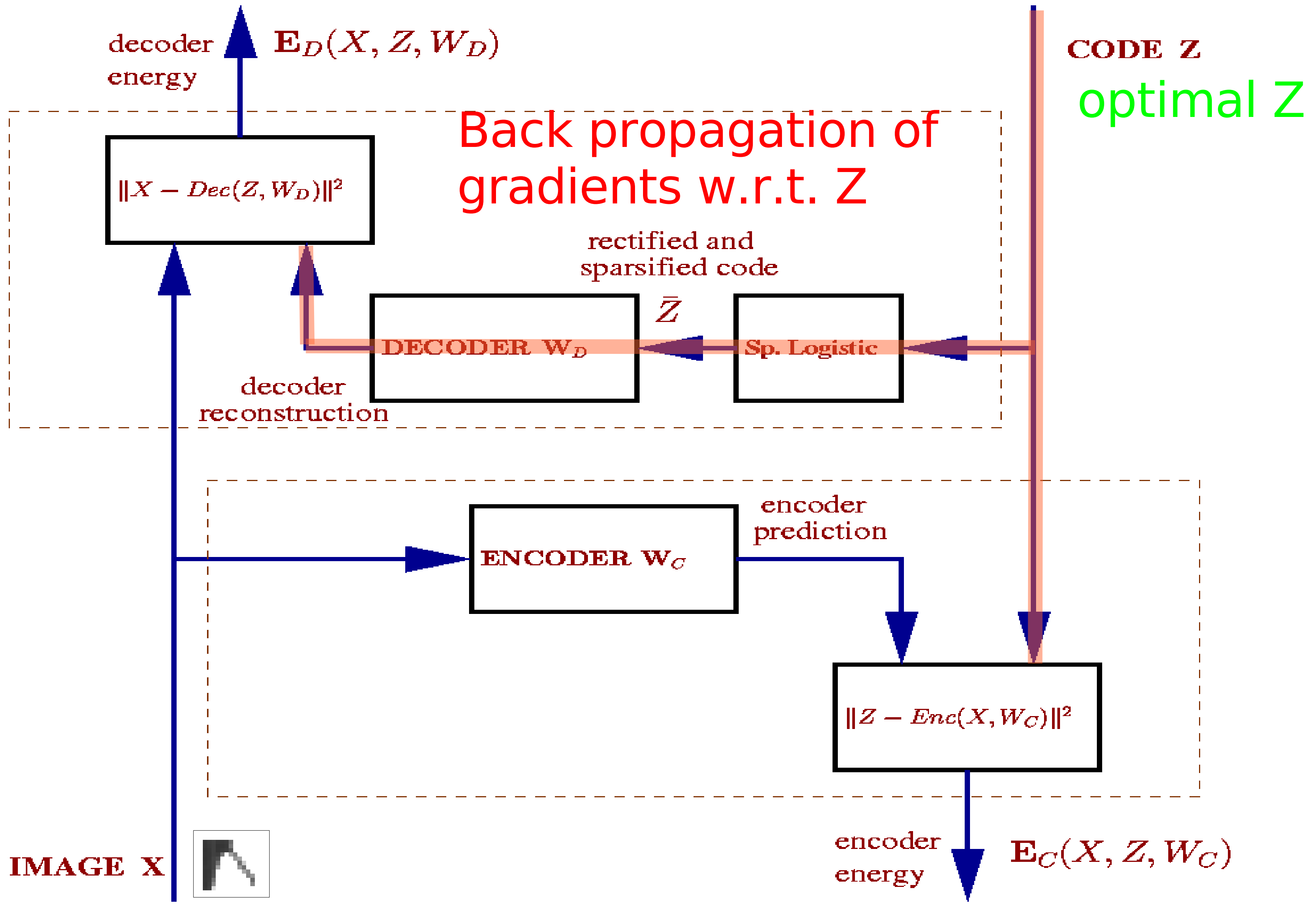
Inference - step 1



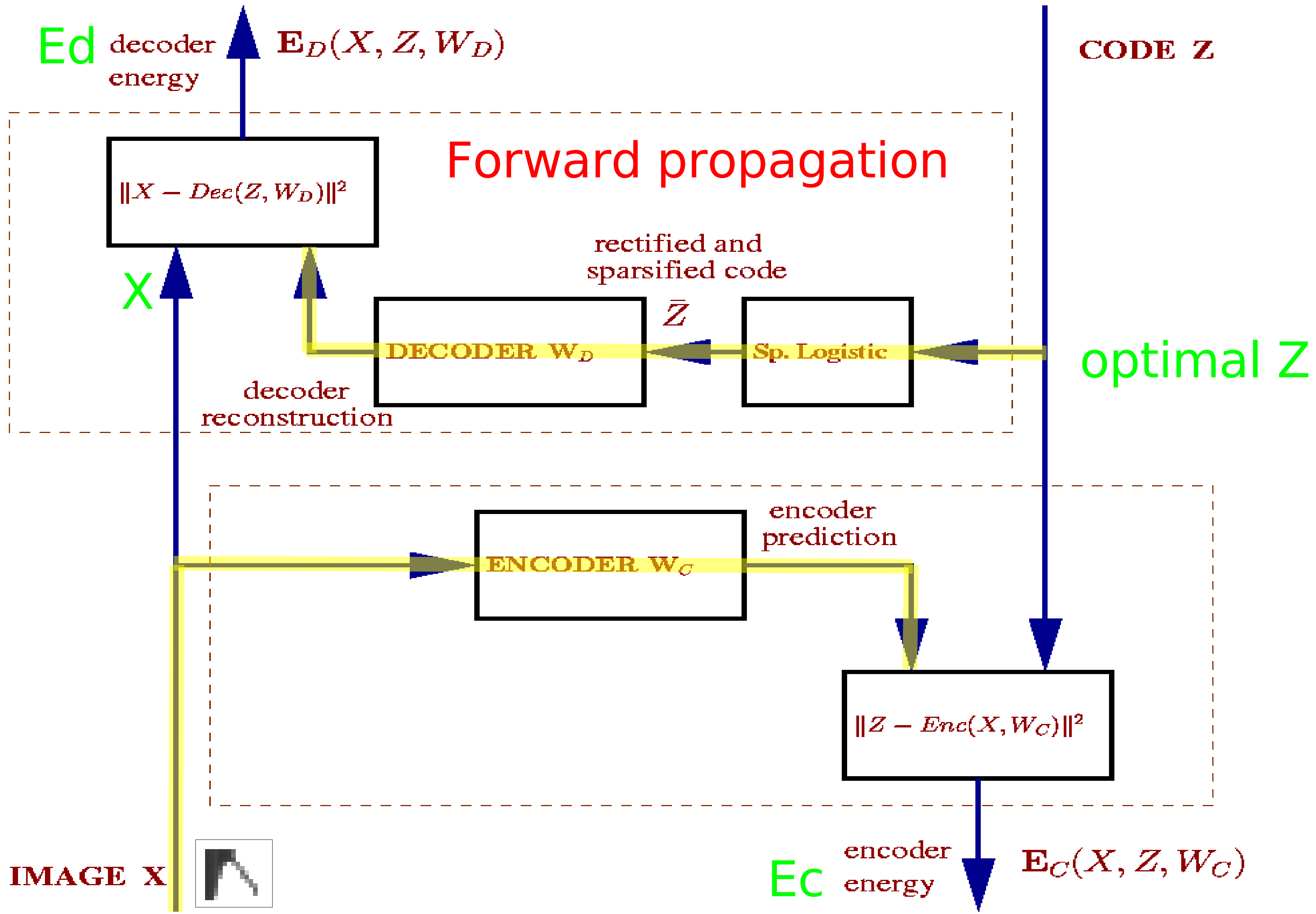
Inference - step 1



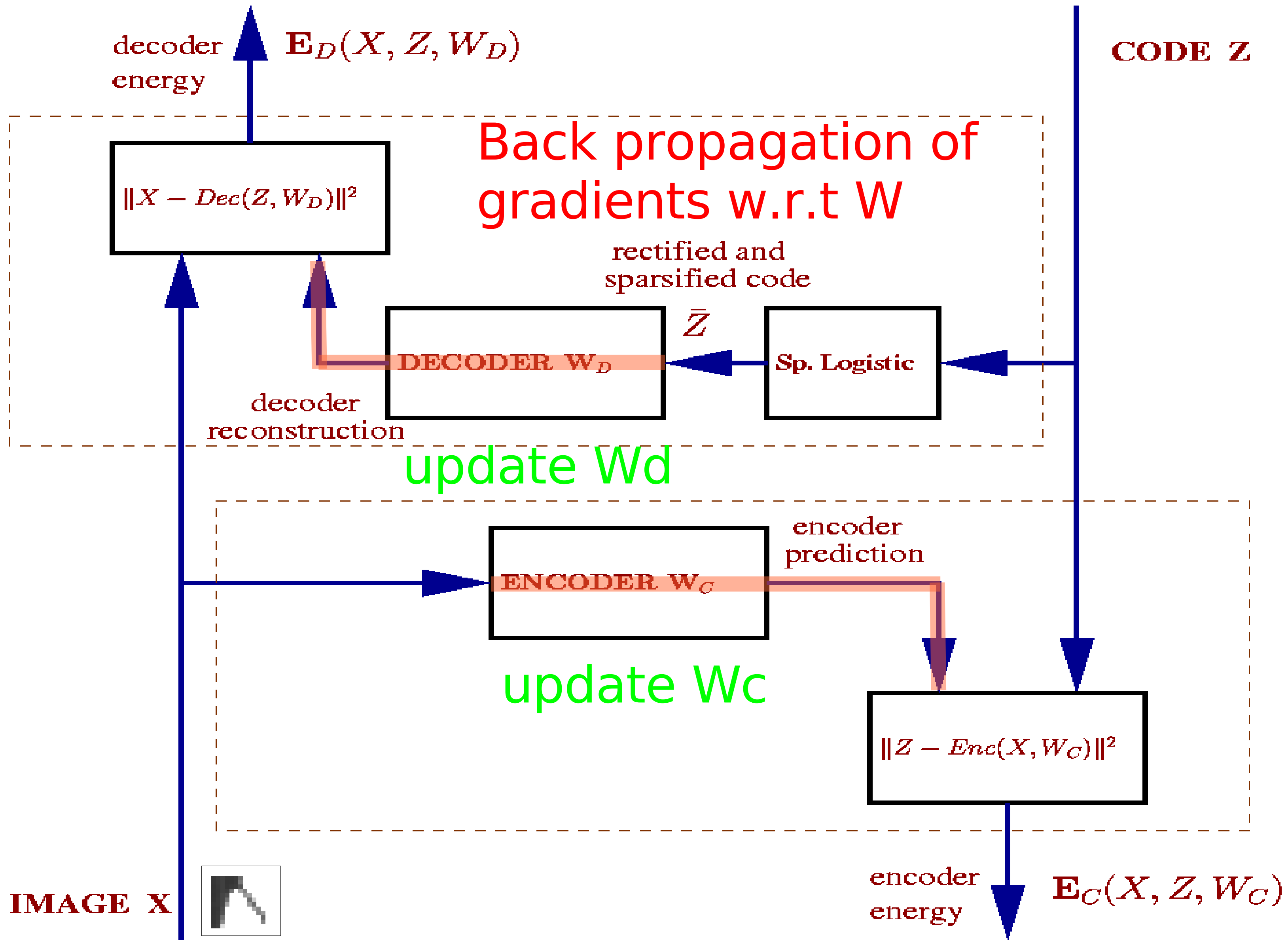
Inference - step 1



Learning - step 2



Learning - step 2



Sparsifying Logistic

Sparsifying non linearity mapping a code vector into a sparse code vector with components between 0 and 1.

- z_i input unit
- $\bar{z}_i$ corresponding output unit

$$\bar{z}_i(k) = \frac{\eta \, \mathrm{e}^{\beta z_i(k)}}{\xi_i(k)}, \quad i \in [1..m], k \in [1..P]$$

$$\xi_i(k) = \eta \, \mathrm{e}^{\beta z_i(k)} + (1-\eta)\xi_i(k-1)$$ (recursive equation)

Expanding the denominator, we have:

$$\bar{z}_i(k) = \frac{\eta \, \mathrm{e}^{\beta z_i(k)}}{\eta \, e^{\beta z_i(k)} + \eta(1-\eta) e^{\beta z_i(k-1)} + \eta(1-\eta)^2 e^{\beta z_i(k-2)} + \ldots}$$

This is a *logistic* function with an adaptive bias:

$$\bar{z}_i(k) = \frac{1}{1 + e^{-\beta[z_i(k) - \frac{1}{\beta}\log(\frac{1-\eta}{\eta}\xi_i(k-1))]}}$$

Sparsifying Logistic

$$\overline{z}_i(k) = \frac{\eta\, e^{\beta z_i(k)}}{\xi_i(k)}, \quad i \in [1..m], k \in [1..P]$$

$$\xi_i(k) = \eta\, e^{\beta z_i(k)} + (1-\eta)\,\xi_i(k-1)$$

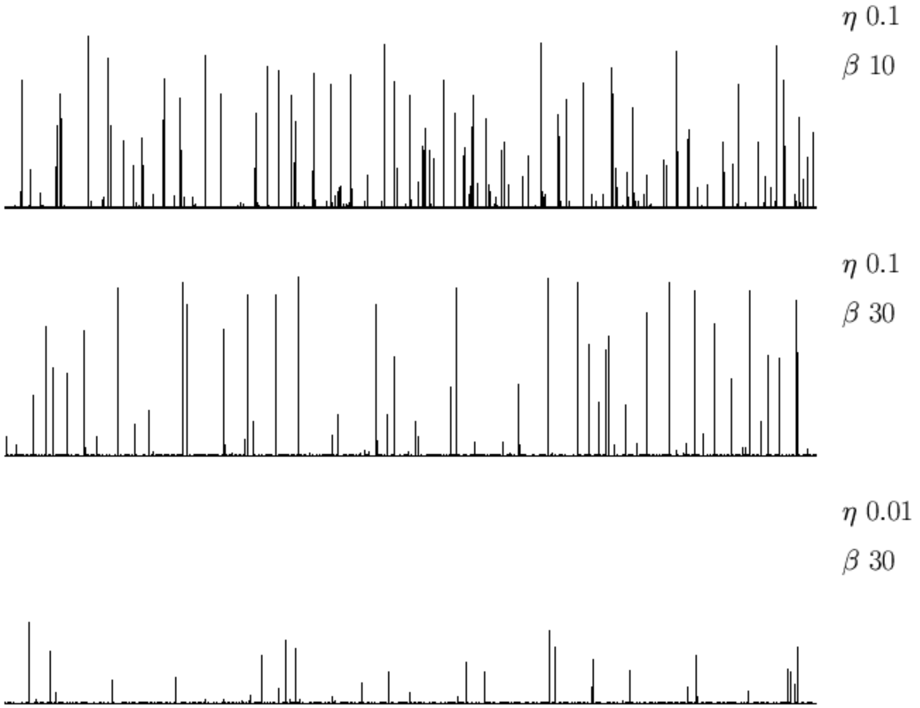
EXAMPLE

Input: random variable uniformly distributed in [-1,1]

Output: a Poisson process with firing rate determined by η and β.

- Increasing β the gain is increased and the output takes almost binary values.

- Increasing η more importance is given to the current sample, a spike will be more likely to occur.



Sparsifying Logistic

$$\bar{z}_i(k) = \frac{\eta\, e^{\beta z_i(k)}}{\xi_i(k)}, \quad i \in [1..m], k \in [1..P]$$

$$\xi_i(k) = \eta\, e^{\beta z_i(k)} + (1-\eta)\xi_i(k-1)$$

- "across samples" vs. "spatial" sparsity => no normalization is required

- ξ is treated as a learned parameter after training

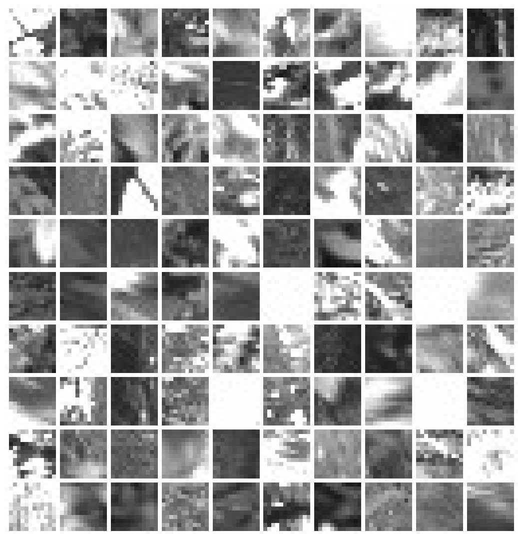
- ξ is saturated during training to allow units to have different sparsity.

- it is biologically inspired

[Pillow, Paninski, Uzzell, Simoncelli, Chichilnisky
"Prediction and Decoding of Retinal Ganglion Cell Responses with a Probabilistic Spiking Model"
J. Neuroscience 2005]

[Foldiak "*Forming Sparse Representations by Local Anti-Hebbian Learning*" Biological Cybernetics 1990]

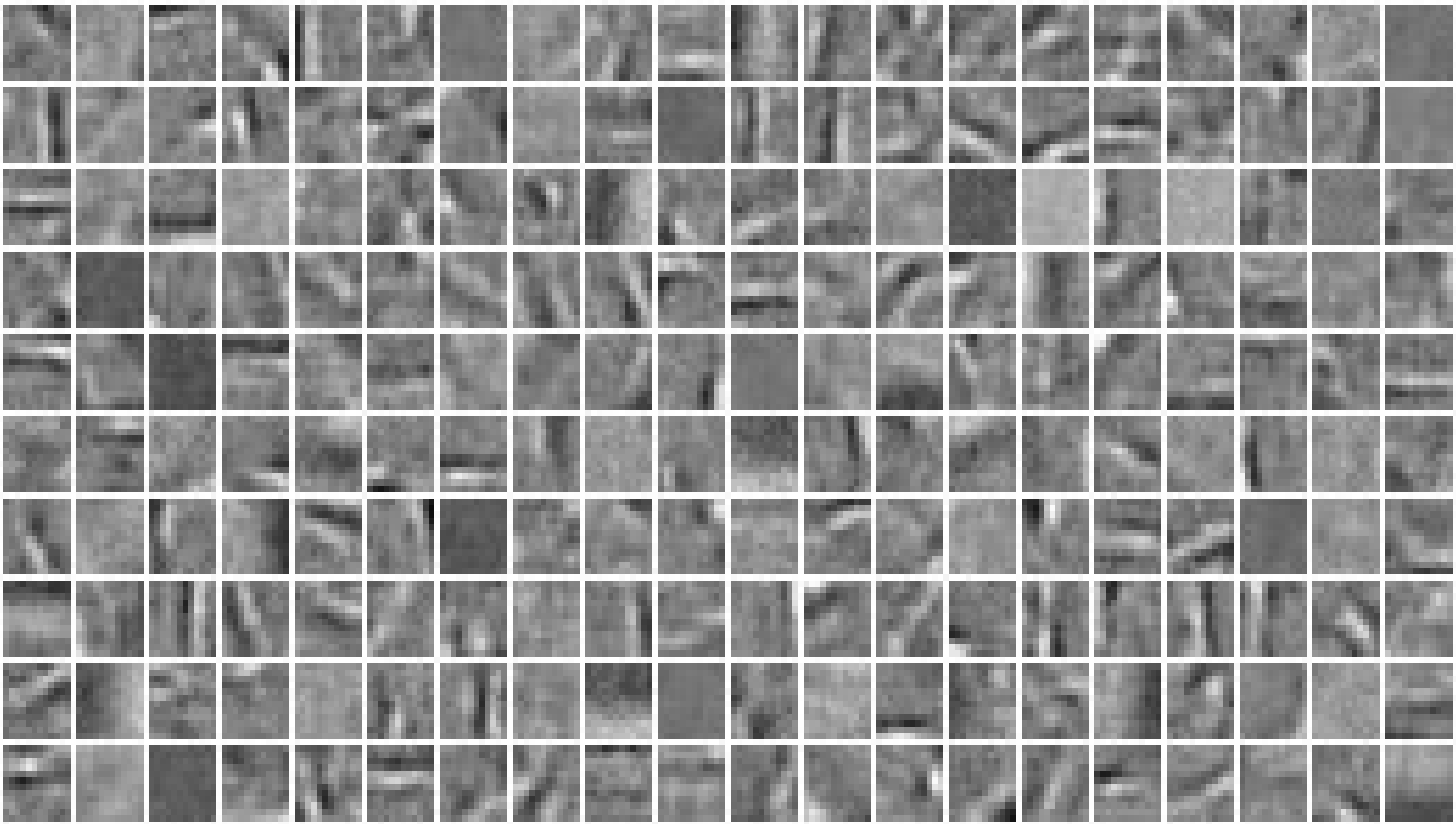
Natural image patches - Berkeley



Berkeley data set

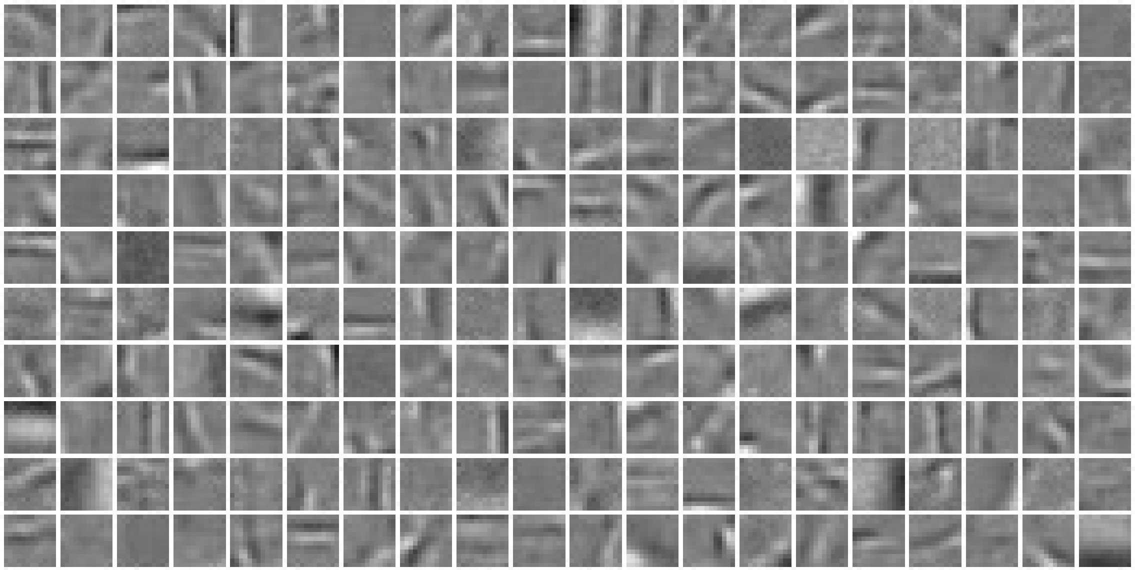
- 100,000 12x12 patches
- 200 units in the code
- η 0.02
- β 1
- learning rate 0.001
- L1, L2 regularizer 0.001
- fast convergence: < 30min.

Natural image patches - Berkeley

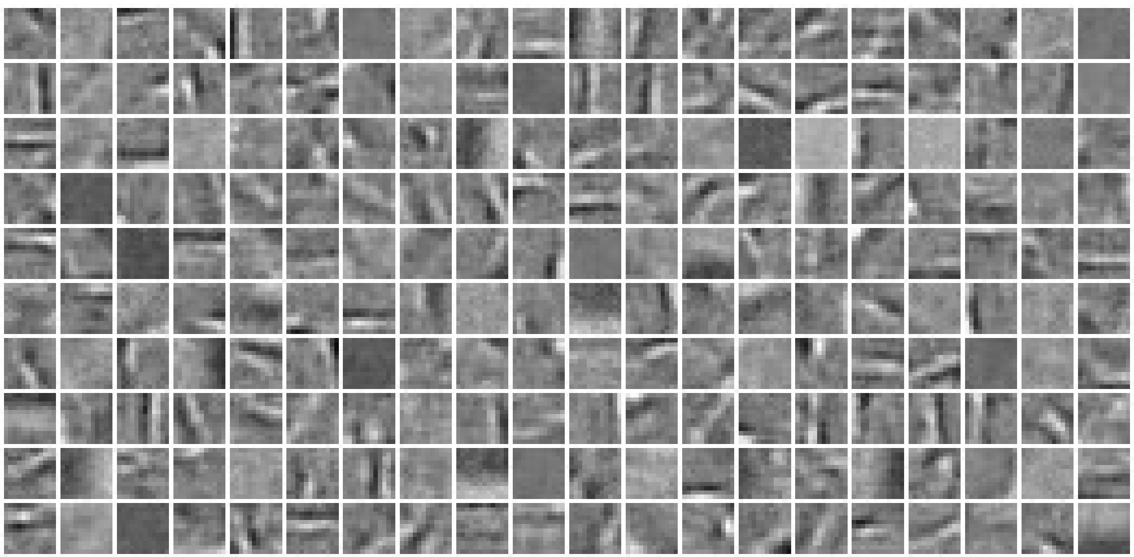


200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

Natural image patches - Berkeley

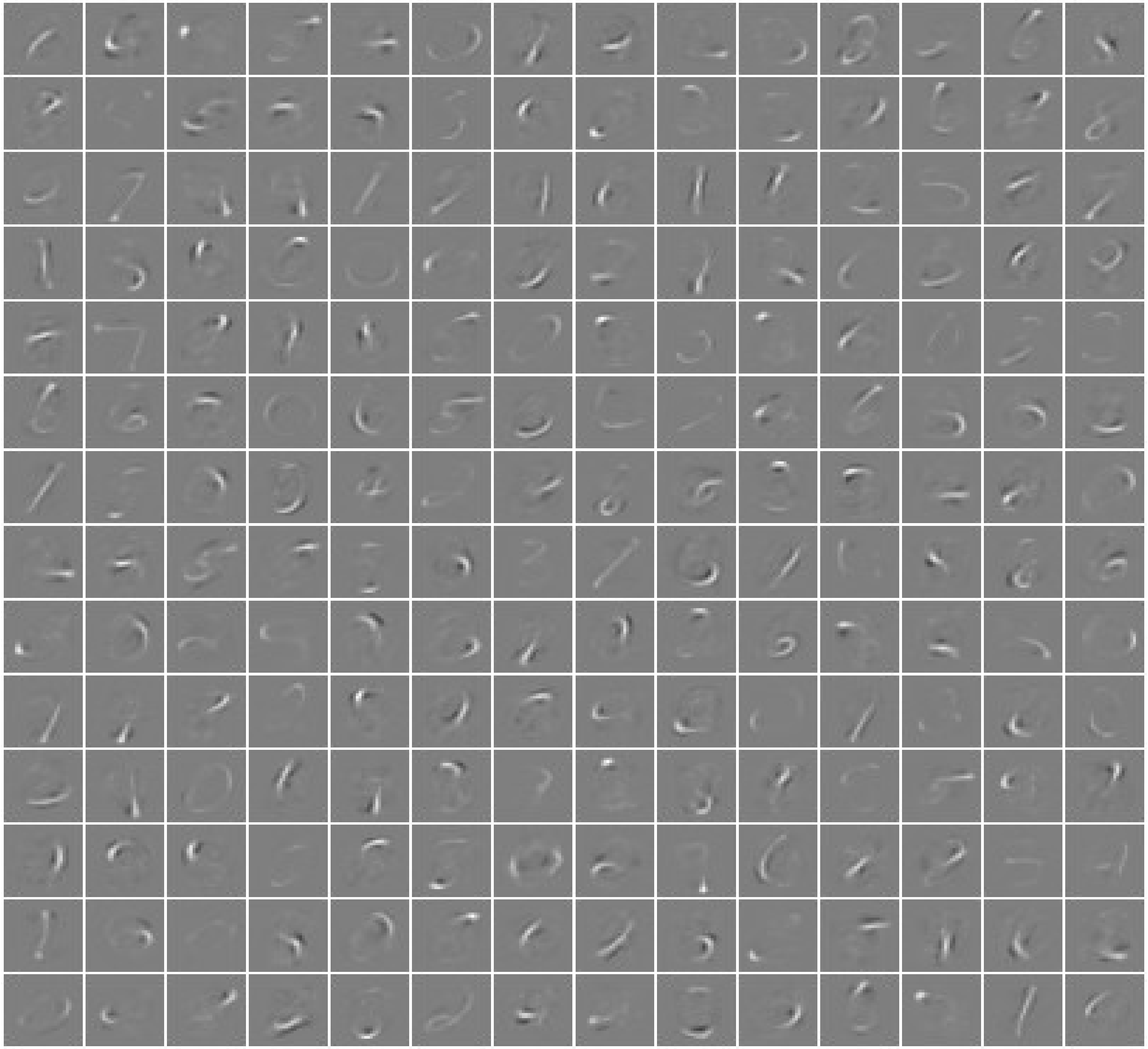


Encoder *direct* filters
(rows of $\mathbf{W_c}$)



Decoder *reverse* filters
(cols. of $\mathbf{W_d}$)

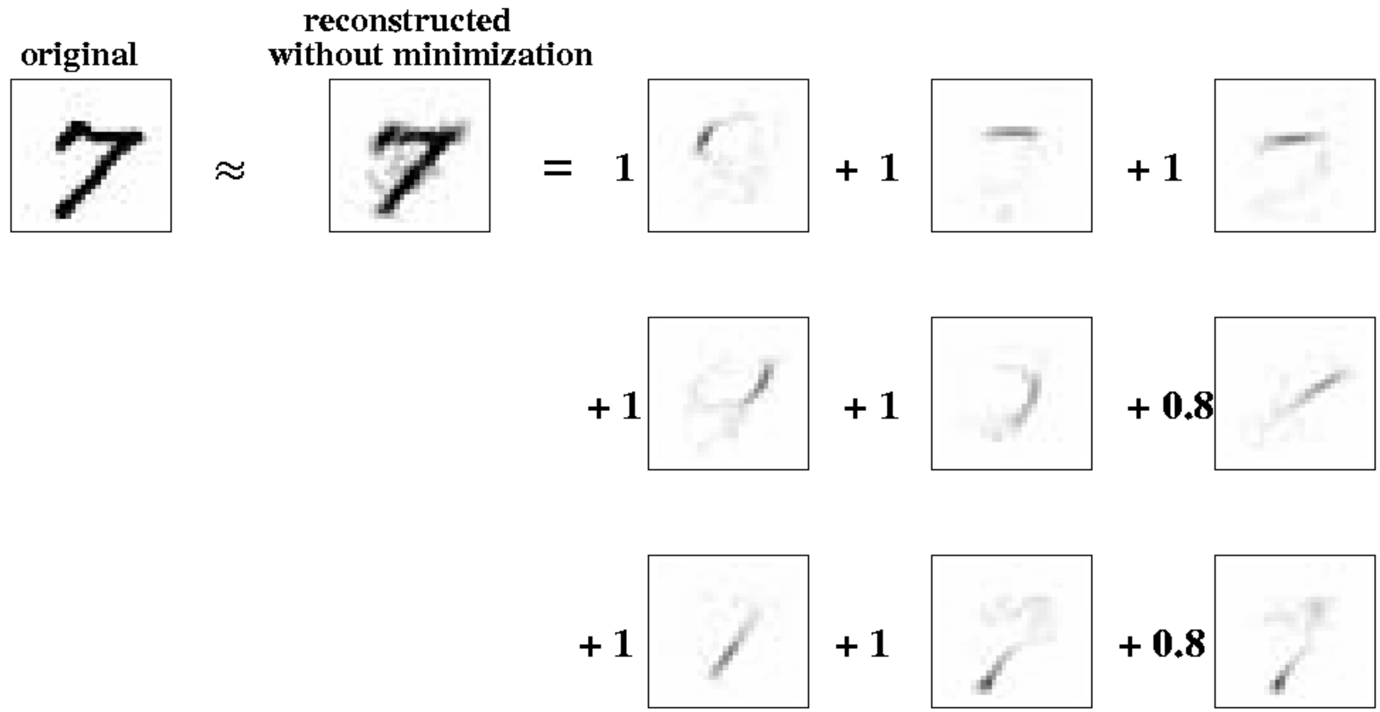
Handwritten digits - MNIST

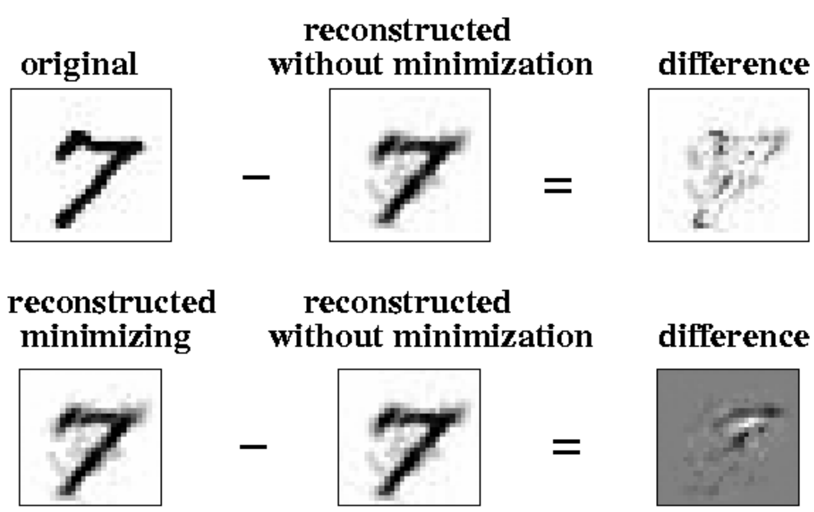


- 60,000 28x28 images
- 196 units in the code
- η 0.01
- β 1
- learning rate 0.001
- L1, L2 regularizer 0.005

Encoder *direct* filters

Handwritten digits - MNIST





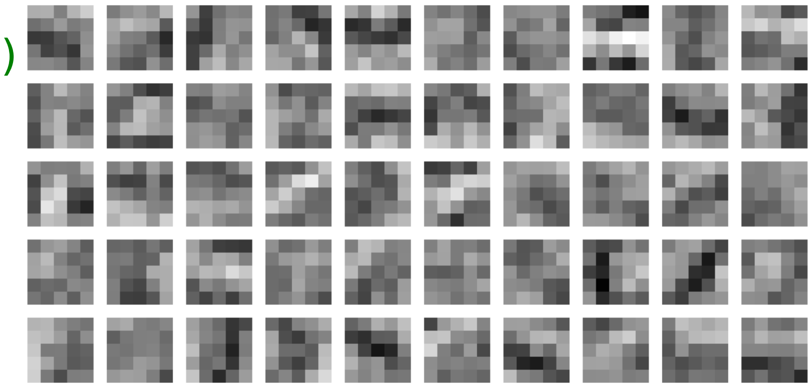
forward propagation through encoder and decoder

after training there is no need to minimize in code space

Handwritten digits - MNIST

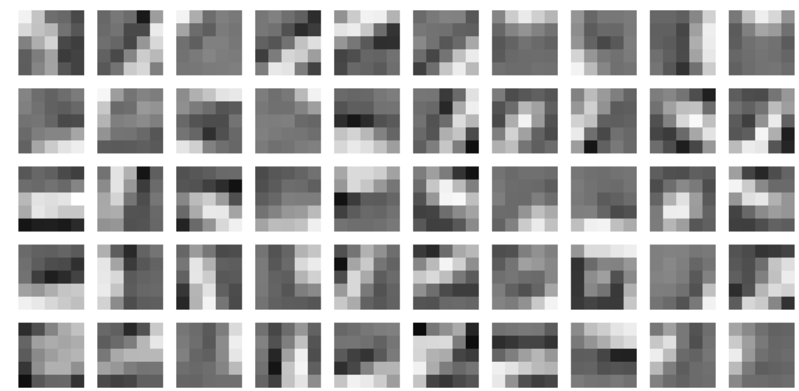
CLASSIFICATION EXPERIMENTS

sparse representations & *lenet6* (1->50->50->200->10)



- The baseline: *lenet6* initialized randomly

Test error rate: 0.70%. Training error rate: 0.01%.



- *Experiment 1*
  - Train on 5x5 patches to find 50 features
  - Use the scaled filters in the encoder to initialize the kernels in the first convolutional layer

Test error rate: 0.60%. Training error rate: 0.00%.

- *Experiment 2*
  - Same as experiment 1, but training set augmented by elastically distorted digits (random initialization gives test error rate equal to 0.49%).

Test error rate: 0.39%. Training error rate: 0.23%.

Summary

STRENGTHS

- Simple and fast learning algorithm
- Fast inference
- Simple preprocessing (no whitening)
- No normalization
- Can generate data from codes

LIMITATIONS

- No rigorous probabilistic interpretation
- No guarantee to find optimal solution

Summary

FUTURE WORK

- Multi-layer encoder and decoder
- Hierarchical architectures
- Applications: classification, clustering, density estimation, denoising
- Theoretical analysis of convergence

THANK YOU