Deep Learning for Generic Object Recognition

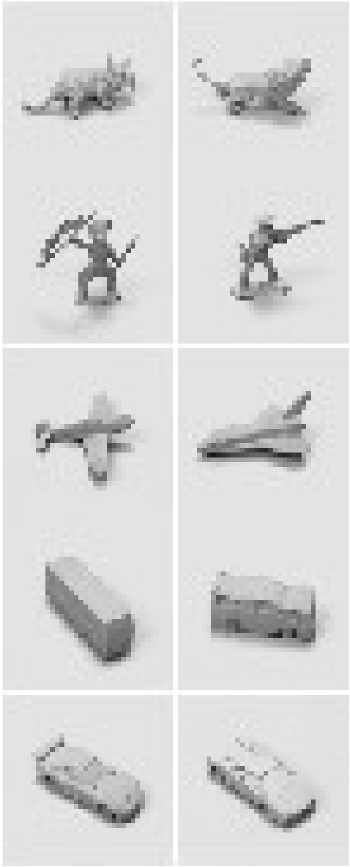
Yann LeCun,
Computational and Biological Learning Lab
The Courant Institute of Mathematical Sciences
New York University
Collaborators: Marc'Aurelio Ranzato, Fu Jie Huang,

http://yann.lecun.com

http://www.cs.nyu.edu/~yann

Generic Object Detection and Recognition with Invariance to Pose, Illumination and Clutter

- **Computer Vision and Biological Vision are getting back together again after a long divorce (Hinton, LeCun, Poggio, Perona, Ullman, Lowe, Triggs, S. Geman, Itti, Olshausen, Simoncelli, ...).**
- **What happened? (1) Machine Learning, (2) Moore's Law.**
- **Generic Object Recognition** is the problem of detecting and classifying objects into generic categories such as “cars”, “trucks”, “airplanes”, “animals”, or “human figures”
- **Appearances are highly variable within a category** because of shape variation, position in the visual field, scale, viewpoint, illumination, albedo, texture, background clutter, and occlusions.
- **Learning invariant representations is key.**
- **Understanding the neural mechanism behind invariant recognition is one of the main goals of Visual Neuroscience.**

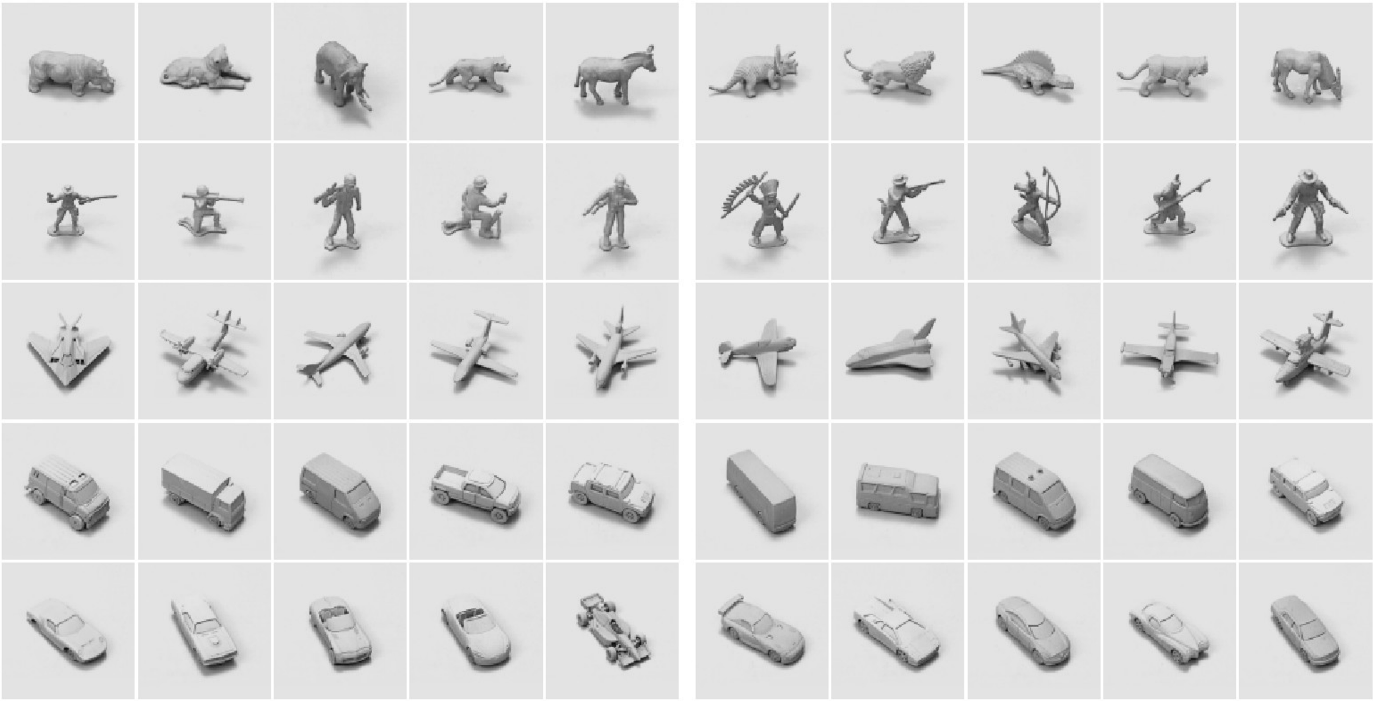


Why do we need “Deep” Architectures?

- **Conjecture: we won't solve the perception problem without solving the problem of learning in deep architectures [Hinton]**
  - Neural nets with lots of layers
  - Deep belief networks
  - Factor graphs with a “Markov” structure
- **We will not solve the perception problem with kernel machines**
  - Kernel machines are glorified template matchers
  - You can't handle complicated invariances with templates (you would need too many templates)
- **Many interesting functions are “deep”**
  - Any function can be approximated with 2 layers (linear combination of non-linear functions)
  - But many interesting functions a more efficiently represented with multiple layers
  - Stupid examples: binary addition

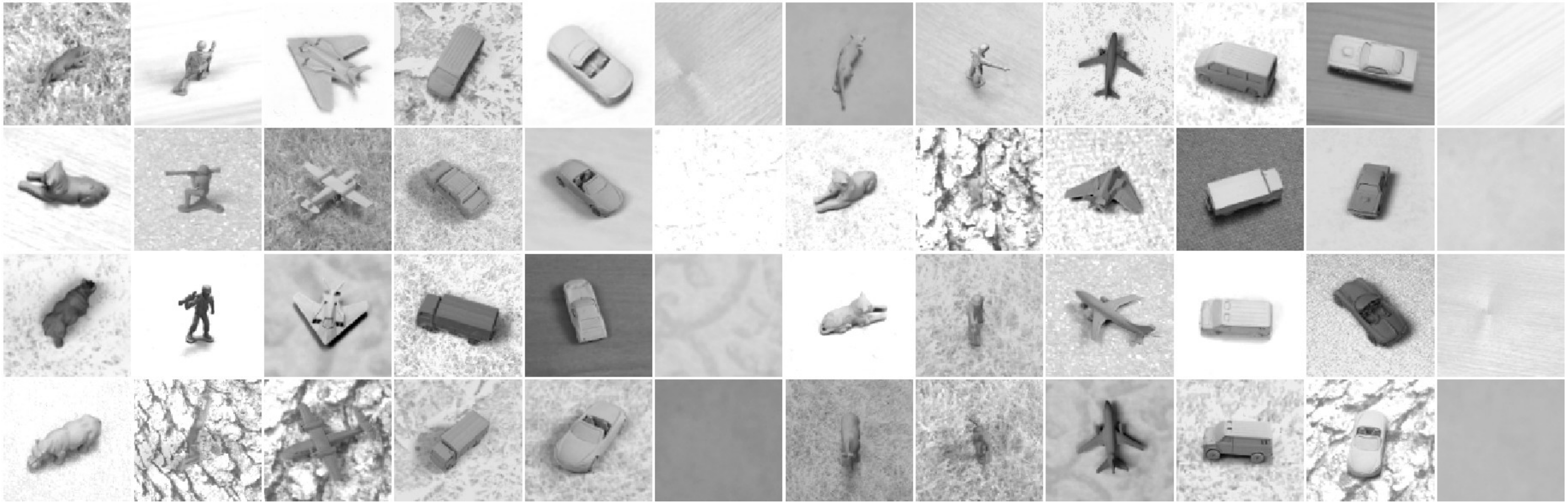
Generic Object Detection and Recognition with Invariance to Pose and Illumination

- **50** toys belonging to 5 categories: **animal, human figure, airplane, truck, car**
- **10** instance per category: 5 instances used for training, 5 instances for testing
- **Raw dataset: 972** stereo pair of each object instance. **48,600** image pairs total.
- **For each instance:**
- **18 azimuths**
  - 0 to 350 degrees every 20 degrees
- **9 elevations**
  - 30 to 70 degrees from horizontal every 5 degrees
- **6 illuminations**
  - on/off combinations of 4 lights
- **2 cameras (stereo)**
  - 7.5 cm apart
  - 40 cm from the object

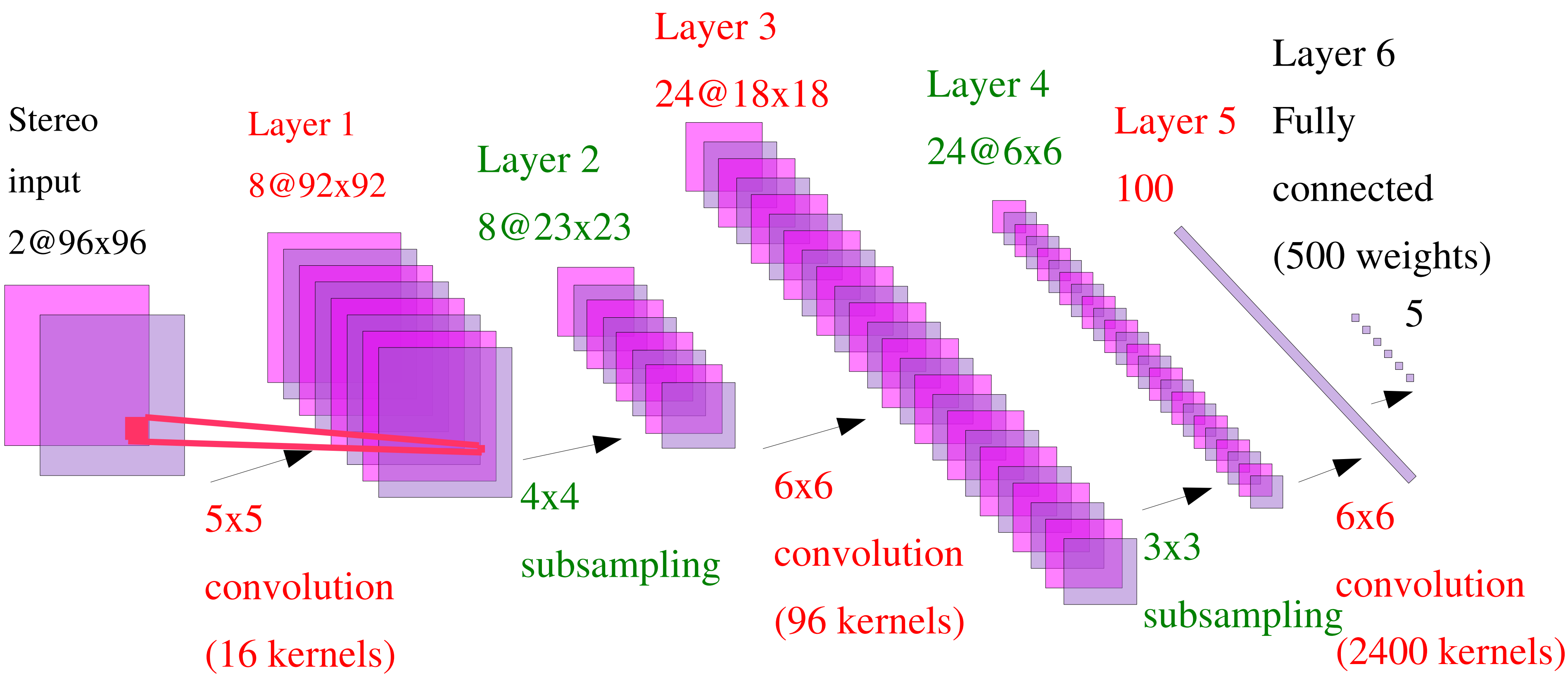


Training instances **Test instances**

Textured and Cluttered Datasets

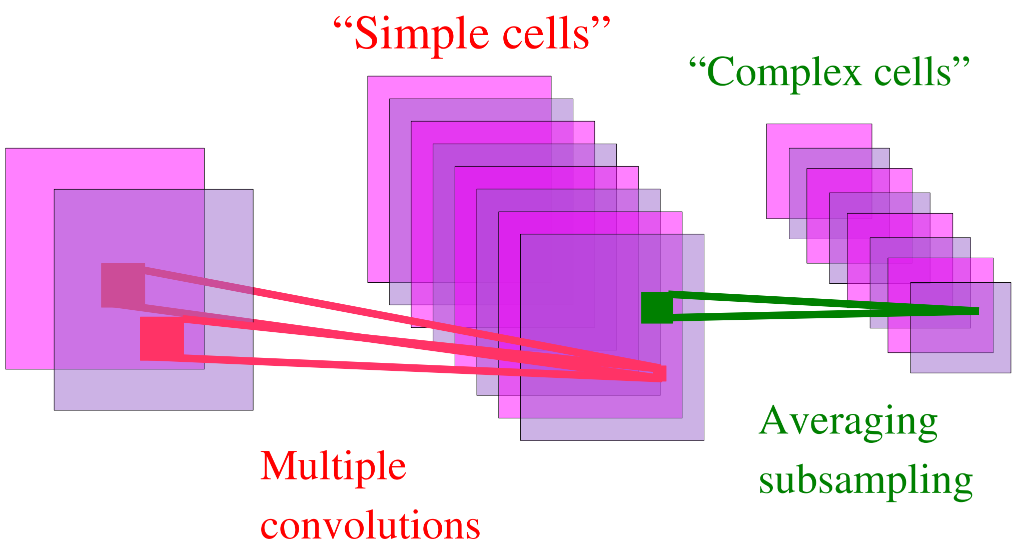


Convolutional Network

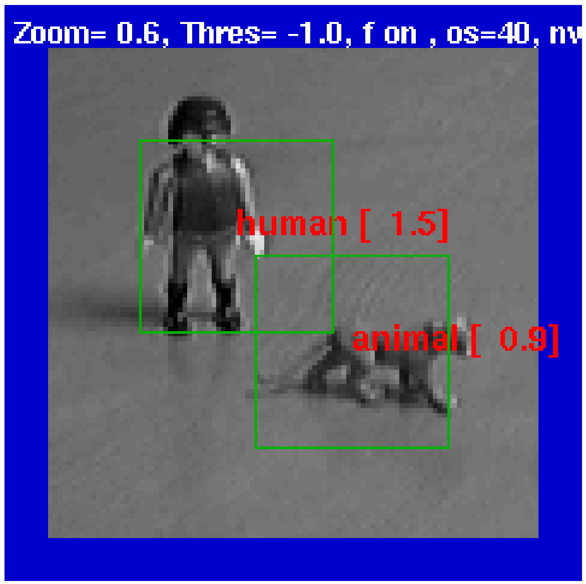


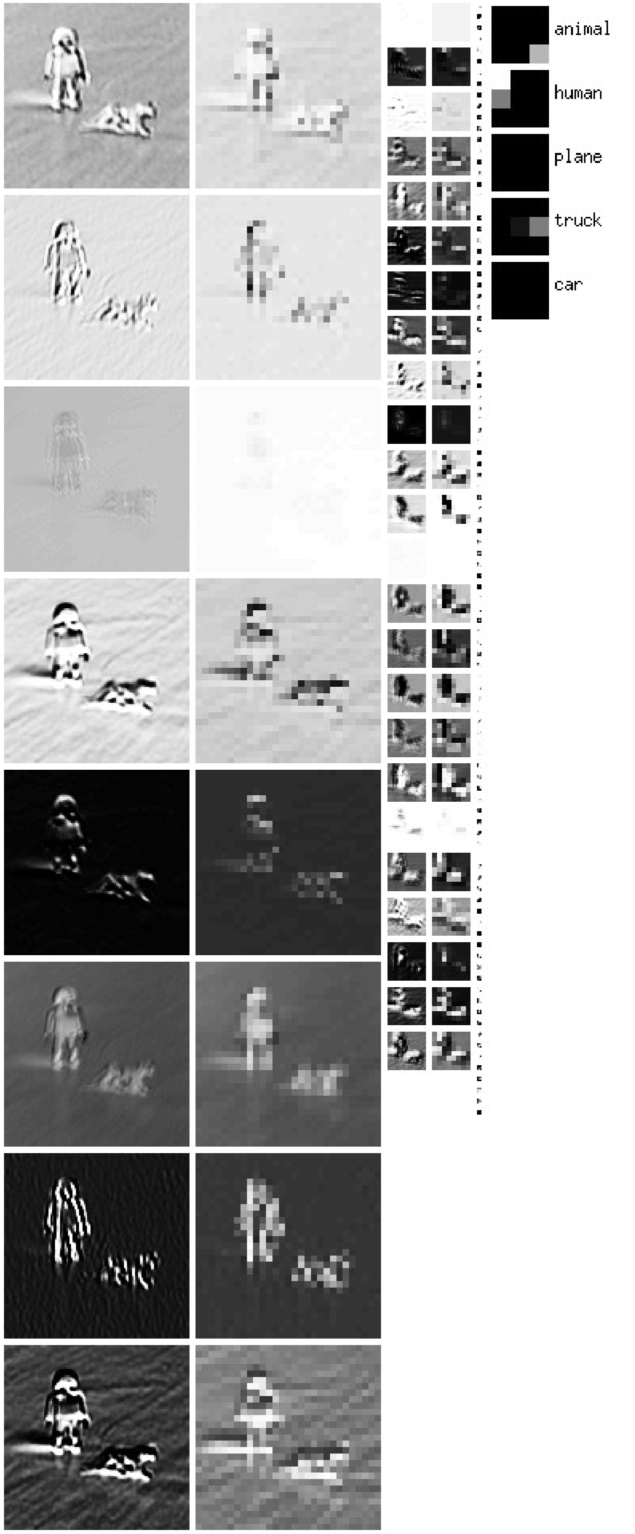
- **90,857 free parameters, 3,901,162 connections.**
- The architecture alternates convolutional layers (feature detectors) and subsampling layers (local feature pooling for invariance to small distortions).
- **The entire network is trained end-to-end** (all the layers are trained simultaneously).
- A gradient-based algorithm is used to minimize a supervised loss function.

Alternated Convolutions and Subsampling



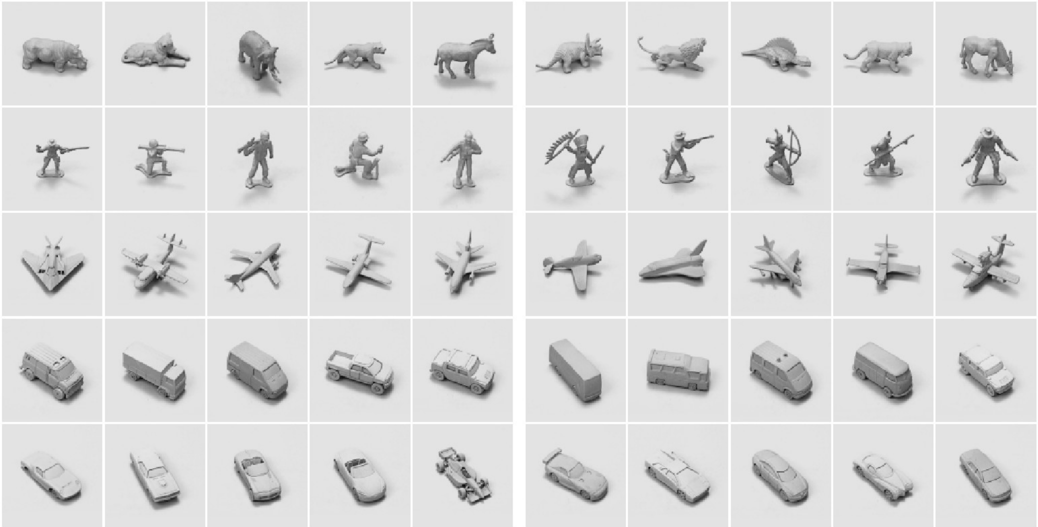
- Local features are extracted everywhere.
- averaging/subsampling layer builds robustness to variations in feature locations.
- Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....





Normalized-Uniform Set: Error Rates

- Linear Classifier on raw stereo images: 30.2% error.
- K-Nearest-Neighbors on raw stereo images: 18.4% error.
- K-Nearest-Neighbors on PCA-95: 16.6% error.
- Pairwise SVM on 96x96 stereo images: 11.6% error
- Pairwise SVM on 95 Principal Components: 13.3% error.
- Convolutional Net on 96x96 stereo images: 5.8% error.



Training instances Test instances

Normalized-Uniform Set: Learning Times

	SVM	Conv Net				SVM/Conv
test error	11.6%	10.4%	6.2%	5.8%	6.2%	5.9%
train time (min*GHz)	480	64	384	640	3,200	50+
test time per sample (sec*GHz)	0.95	0.03				0.04+
#SV	28%					28%
parameters	σ=2,000 C=40					dim=80 σ=5 C=0.01

SVM: using a parallel implementation by Graf, Durdanovic, and Cosatto (NEC Labs)

Chop off the last layer of the convolutional net and train an SVM on it

Jittered-Cluttered Dataset

- **Jittered-Cluttered Dataset:**
- **291,600** tereo pairs for training, **58,320** for testing
- Objects are jittered: position, scale, in-plane rotation, contrast, brightness, backgrounds, distractor objects,...
- Input dimension: 98x98x2 (approx 18,000)

Experiment 2: Jittered-Cluttered Dataset

- 291,600 training samples, 58,320 test samples
- SVM with Gaussian kernel 43.3% error
- Convolutional Net with binocular input: 7.8% error
- Convolutional Net + SVM on top: 5.9% error
- Convolutional Net with monocular input: 20.8% error
- Smaller mono net (DEMO): 26.0% error
- Dataset available from http://www.cs.nyu.edu/~yann

Jittered-Cluttered Dataset

	SVM	Conv Net			SVM/Conv
test error	43.3%	16.38%	7.5%	7.2%	5.9%
train time (min*GHz)	10,944	420	2,100	5,880	330+
test time per sample (sec*GHz)	2.2	0.04			0.06+
#SV	5%				2%
parameters	σ=10^4 C=40				dim=100 σ=5 C=1

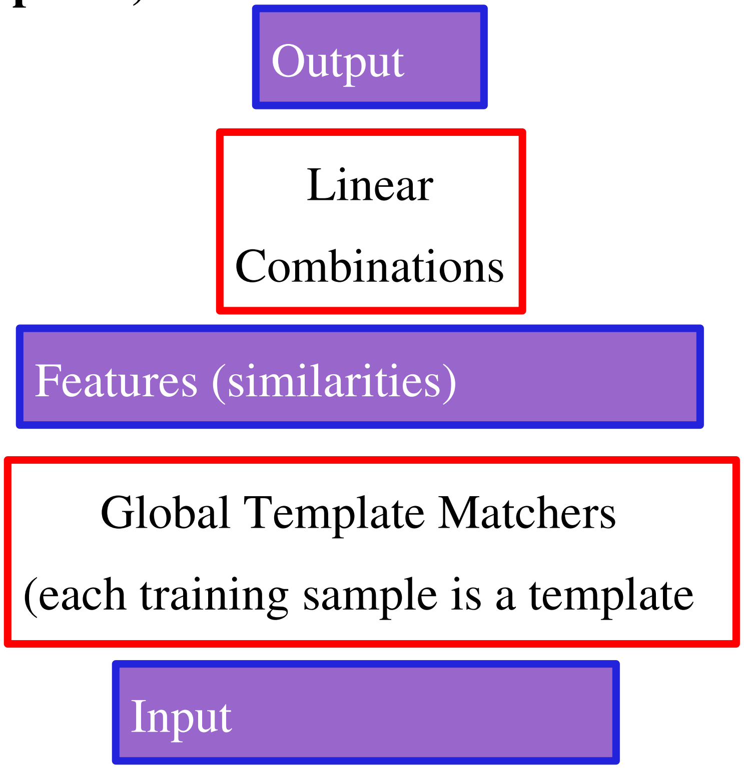
OUCH!

The convex loss, VC bounds and representers theorems don't seem to help

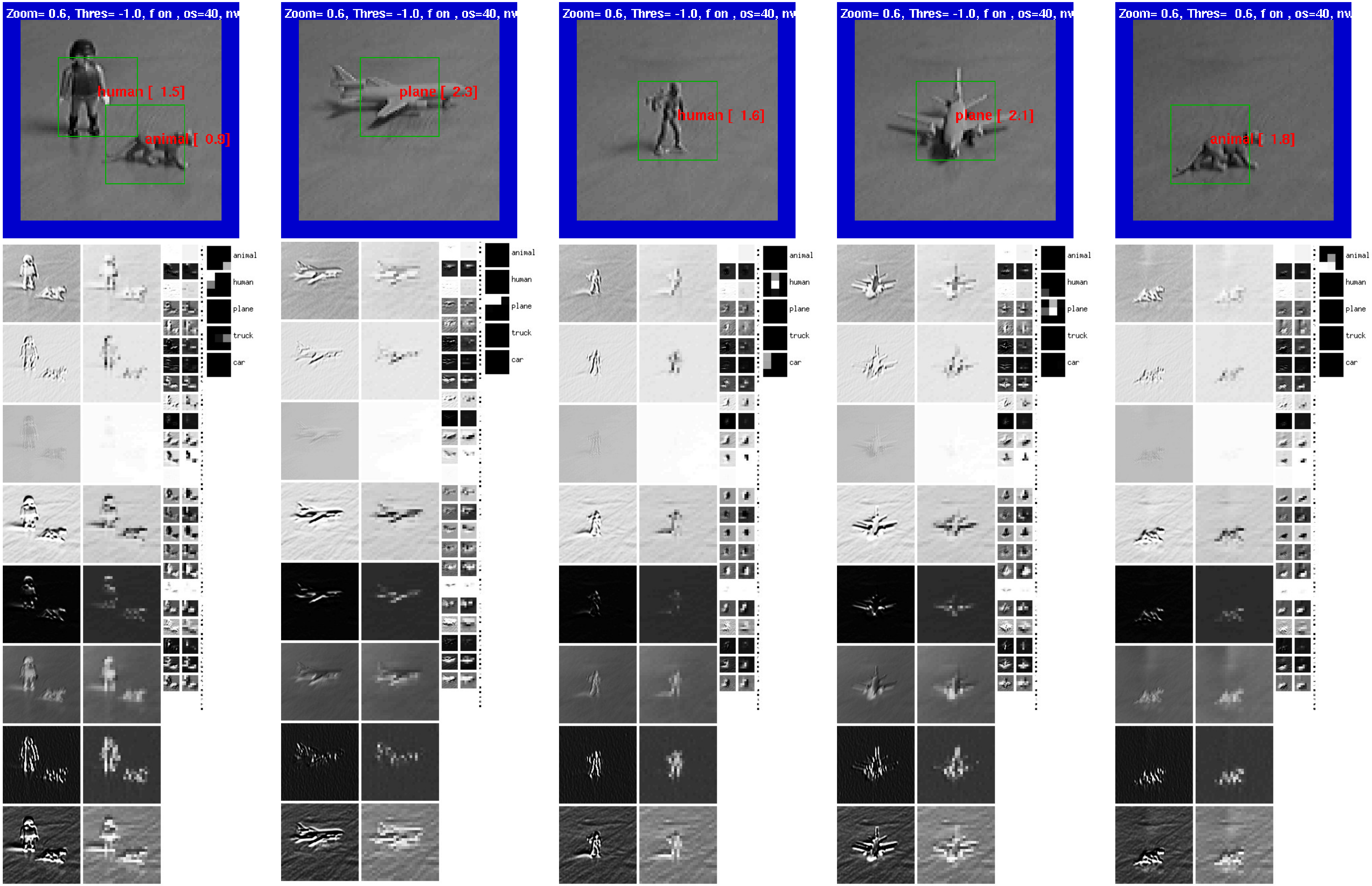
Chop off the last layer, and train an SVM on it it works!

What's wrong with K-NN and SVMs?

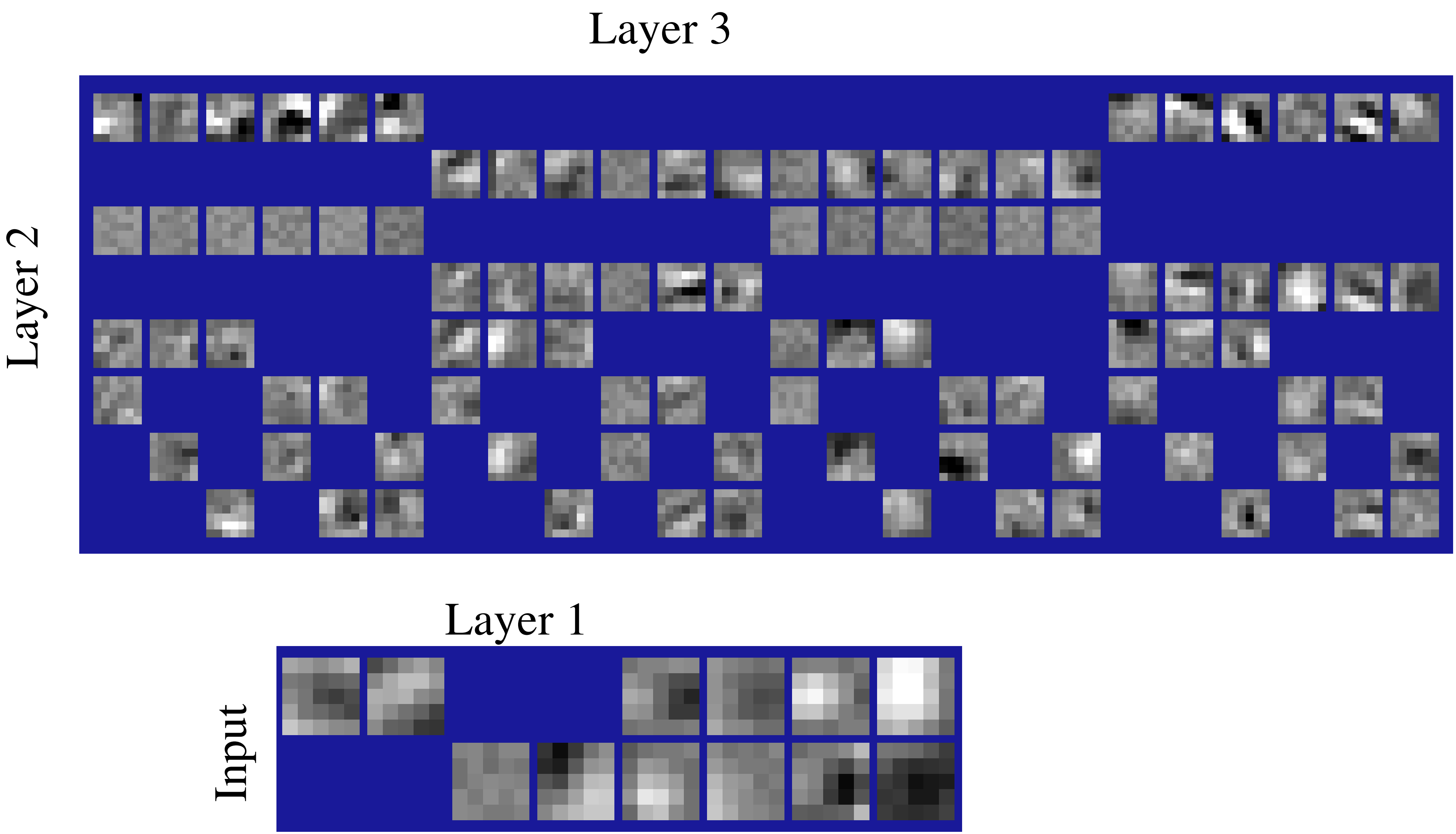
- K-NN and SVM with Gaussian kernels are based on matching global templates
- Both are "shallow" architectures
- There is now way to learn invariant recognition tasks with such naïve architectures (unless we use an impractically large number of templates).
  - The number of necessary templates grows exponentially with the number of dimensions of variations.
  - Global templates are in trouble when the variations include: category, instance shape, configuration (for articulated object), position, azimuth, elevation, scale, illumination, texture, albedo, in-plane rotation, background luminance, background texture, background clutter, ...



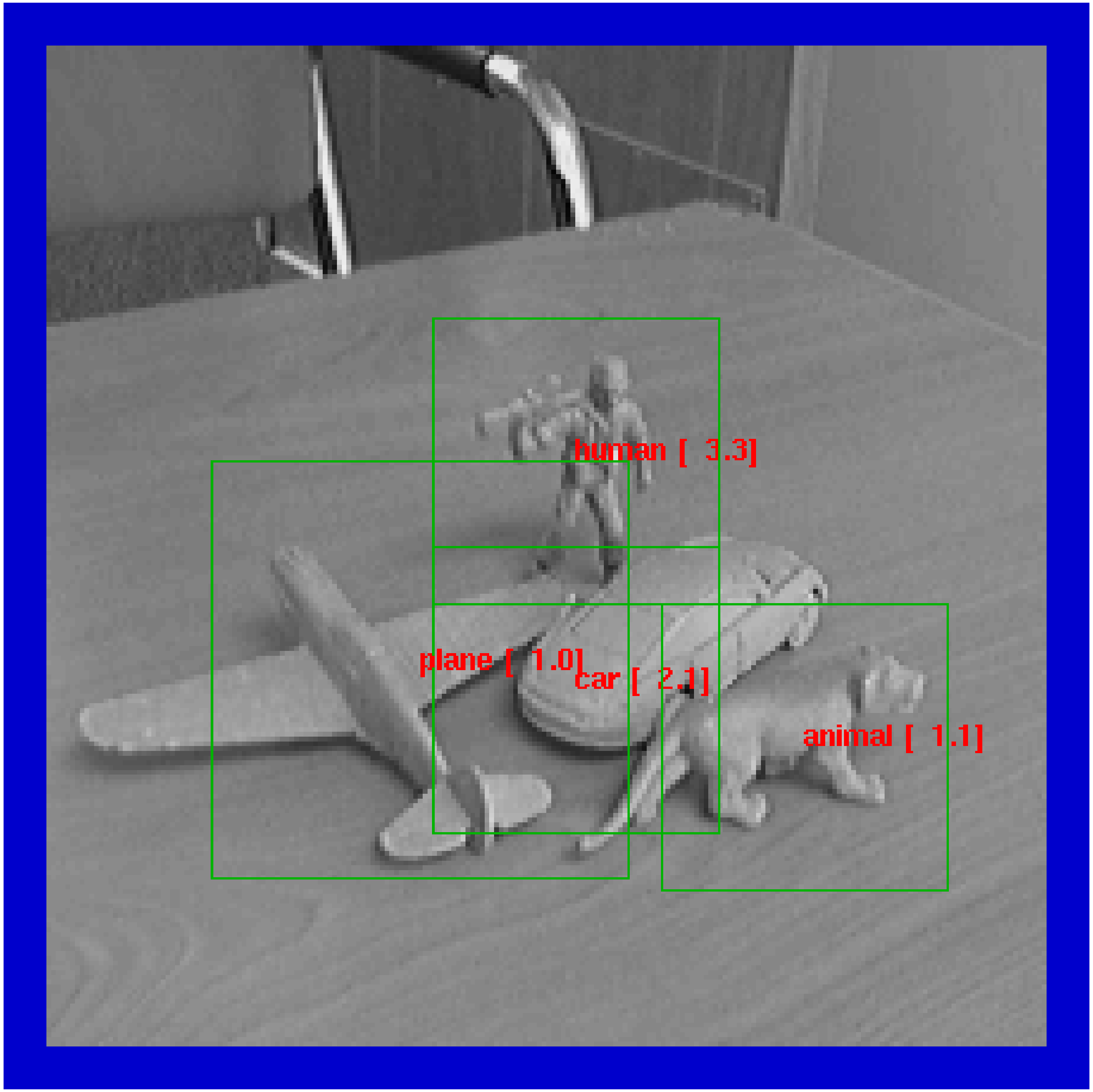
Examples (Monocular Mode)



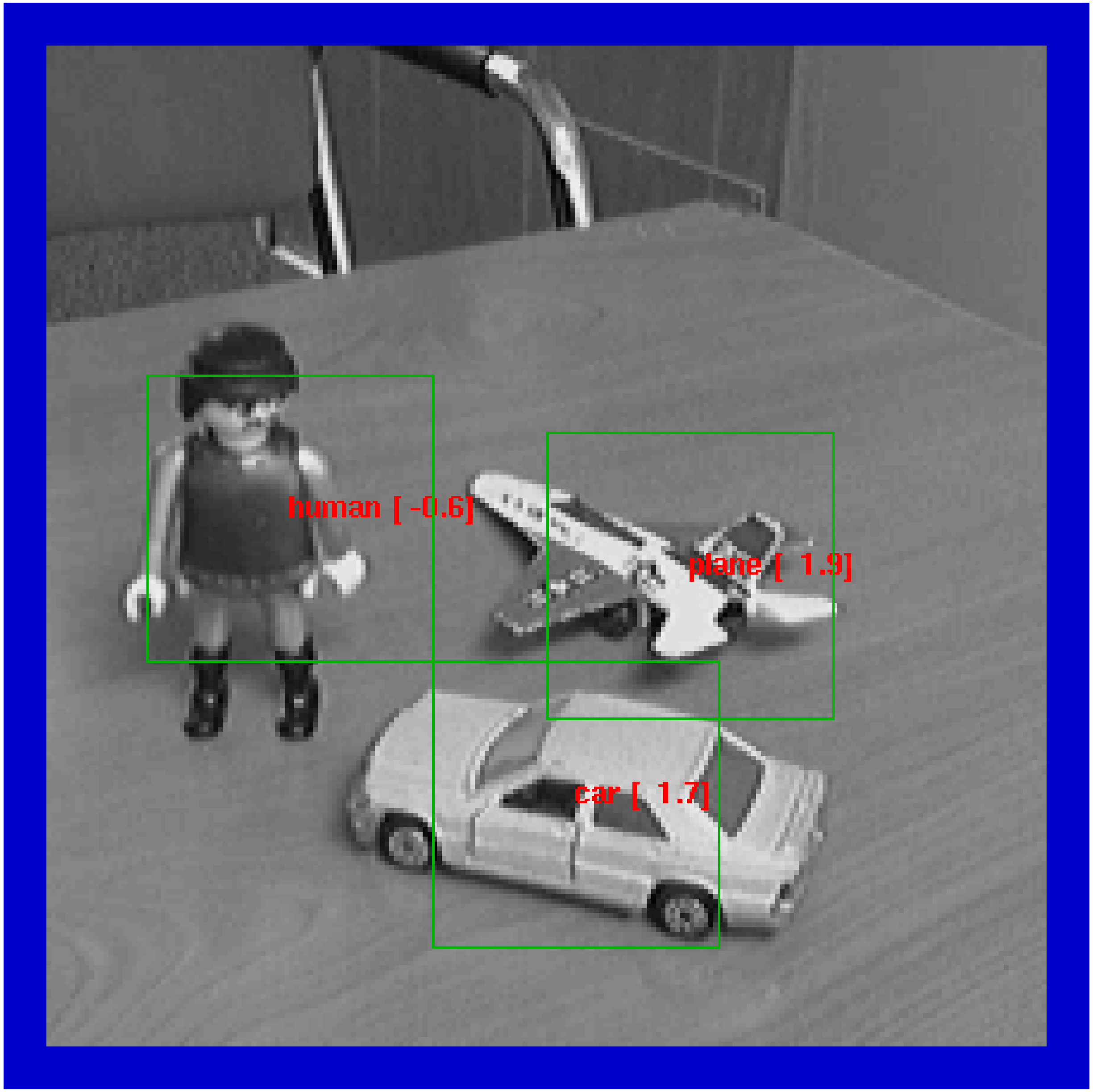
Learned Features



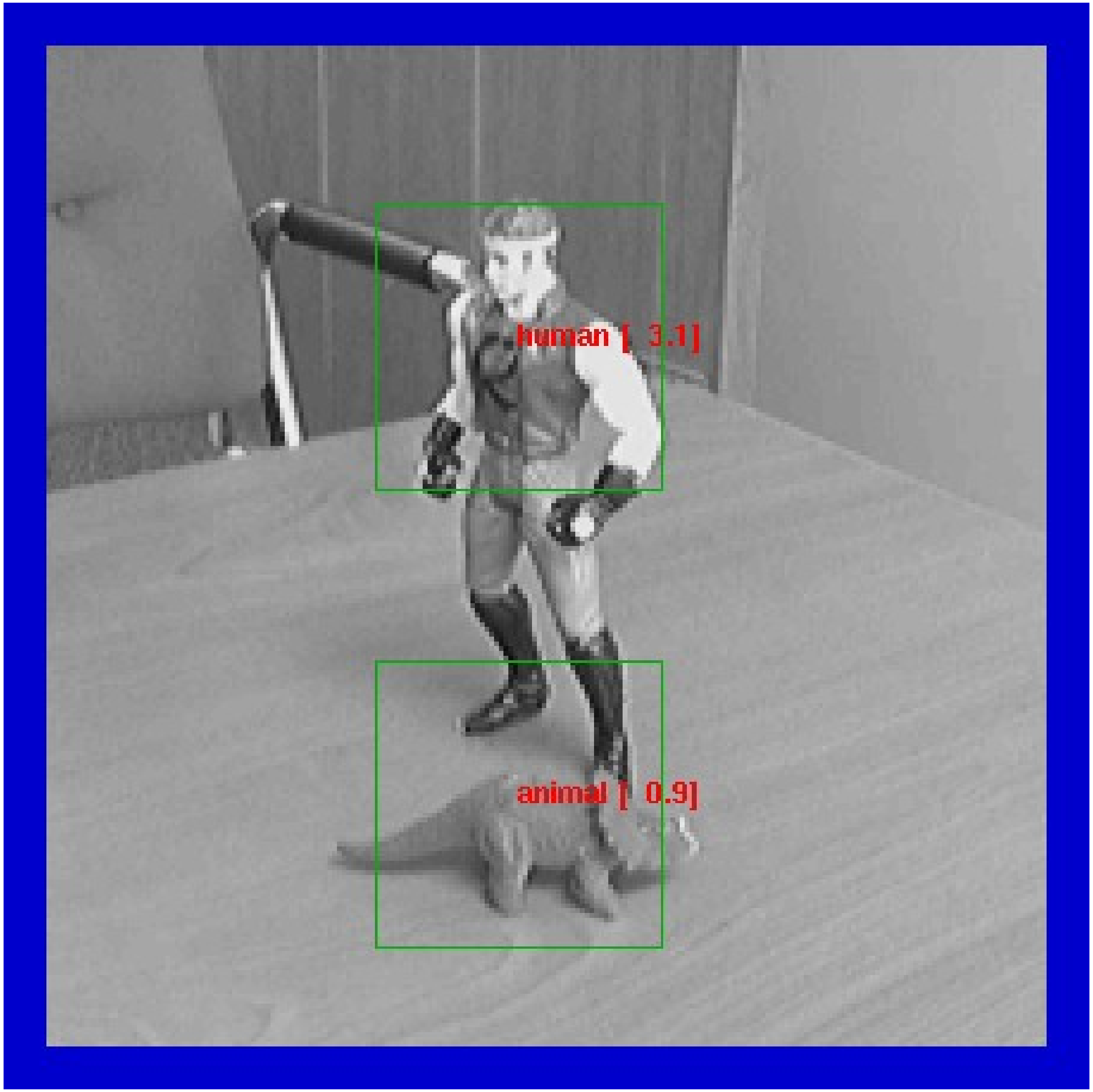
Examples (Monocular Mode)



Examples (Monocular Mode)



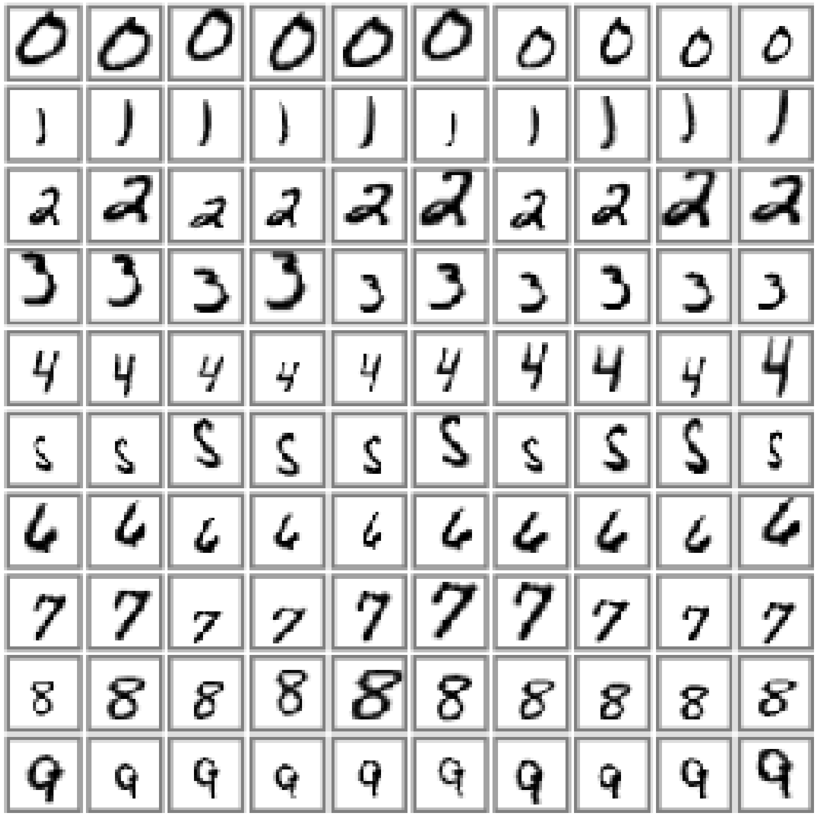
Examples (Monocular Mode)



Supervised Learning in "Deep" Architectures

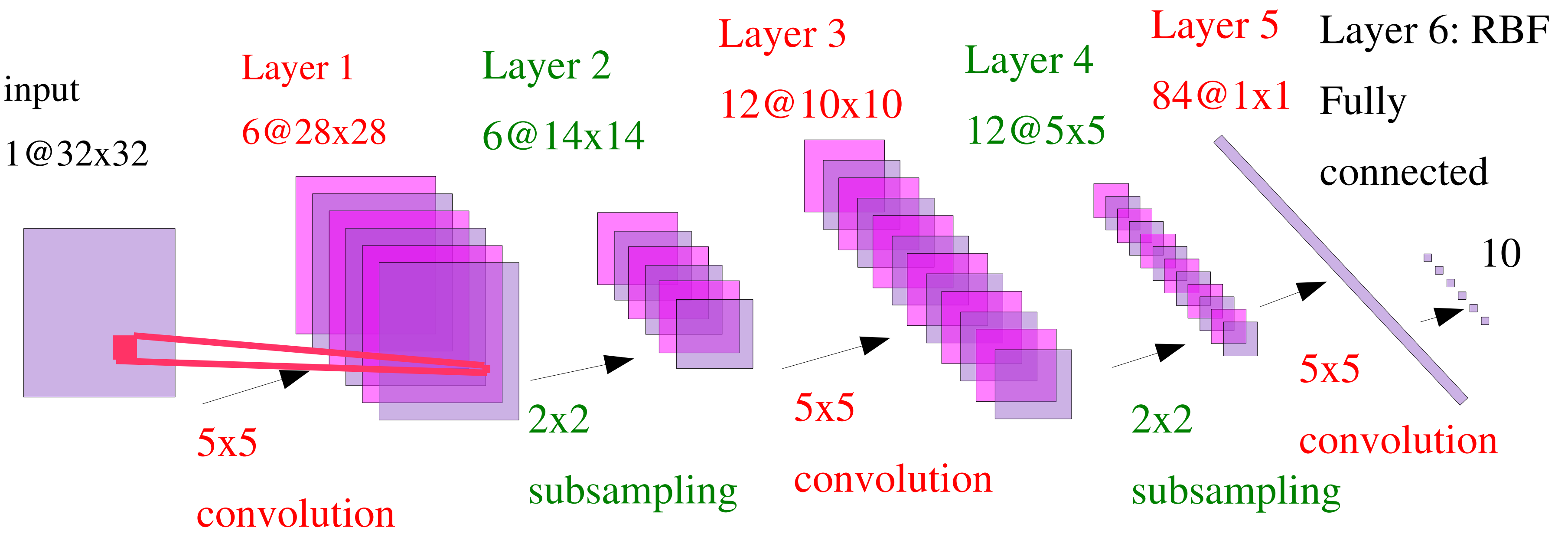
- **Backprop can train "deep" architectures reasonably well**
  - It works better if the architecture has some structure (e.g. A convolutional net)
- **Deep architectures with some structure (e.g. Convolutional nets) beat shallow ones (e.g. Kernel machines) on image classification tasks:**
  - Handwriting recognition
  - Face detection
  - Generic object recognition
- **Deep architectures are inherently more efficient for representing complex functions.**
- **Have we solved the problem of training deep architectures?**
  - Can we do backprop with lots of layers?
  - Can we train deep belief networks?
- **NO!**

MNIST Dataset



- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

Handwritten Digit Recognition with a Convolutional Network



- **60,000 free parameters, 400,000 connections.**
- The architecture alternates convolutional layers (feature detectors) and subsampling layers (local feature pooling for invariance to small distortions).
- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples
- **The entire network is trained end-to-end** (all the layers are trained simultaneously).
- Test Error Rate: 0.8%

Results on MNIST Handwritten Digits (P=60,000)

	CLASSIFIER	DEFORMATION	PREPROCESSING	ERROR	Reference
	linear classifier (1-layer NN)		none	12.00	LeCun et al. 1998
	linear classifier (1-layer NN)		deskewing	8.40	LeCun et al. 1998
	pairwise linear classifier		deskewing	7.60	LeCun et al. 1998
	K-nearest-neighbors, (L2)		none	3.09	K. Wilder, U. Chicago
	K-nearest-neighbors, (L2)		deskewing	2.40	LeCun et al. 1998
	K-nearest-neighbors, (L2)		deskew, clean, blur	1.80	K. Wilder, U. Chicago
Best	K-NN L3, 2 pixel jitter		deskew, clean, blur	1.22	K. Wilder, U. Chicago
Hand-crafted	K-NN, shape context matching		shape context feature	0.63	Belongie PAMI 02
	40 PCA + quadratic classifier		none	3.30	LeCun et al. 1998
	1000 RBF + linear classifier		none	3.60	LeCun et al. 1998
	K-NN, Tangent Distance		subsamp 16x16 pixels	1.10	LeCun et al. 1998
	SVM, Gaussian Kernel		none	1.40	Many
	SVM deg 4 polynomial		deskewing	1.10	Cortes/Vapnik
	Reduced Set SVM deg 5 poly		deskewing	1.00	Scholkopf
	Virtual SVM deg-9 poly	Affine	none	0.80	Scholkopf
Best	V-SVM, 2-pixel jittered		none	0.68	DeCoste/Scholkopf, MLJ'02
Kernel-based	V-SVM, 2-pixel jittered		deskewing	0.56	DeCoste/Scholkopf, MLJ'02
	2-layer NN, 300 HU, MSE		none	4.70	LeCun et al. 1998
	2-layer NN, 300 HU, MSE,	Affine	none	3.60	LeCun et al. 1998
	2-layer NN, 300 HU		deskewing	1.60	LeCun et al. 1998
	3-layer NN, 500+150 HU		none	2.95	LeCun et al. 1998
Best fully-c	3-layer NN, 500+150 HU	Affine	none	2.45	LeCun et al. 1998
Neural Net	3-layer NN, 500+300 HU, CE, reg		none	1.53	Hinton, in press, 2005
	2-layer NN, 800 HU, CE		none	1.60	Simard et al., ICDAR 2003
	2-layer NN, 800 HU, CE	Affine	none	1.10	Simard et al., ICDAR 2003
	2-layer NN, 800 HU, MSE	Elastic	none	0.90	Simard et al., ICDAR 2003
Best know-	2-layer NN, 800 HU, CE	Elastic	none	0.70	Simard et al., ICDAR 2003
Ledge-free	Stacked RBM + backprop		none	0.95	Hinton, in press, 2005
	Convolutional net LeNet-1		subsamp 16x16 pixels	1.70	LeCun et al. 1998
	Convolutional net LeNet-4		none	1.10	LeCun et al. 1998
	Convolutional net LeNet-5,		none	0.95	LeCun et al. 1998
	Convolutional net LeNet-5,	Affine	none	0.80	LeCun et al. 1998
	Boosted LeNet-4	Affine	none	0.70	LeCun et al. 1998
	Convolutional net, CE	Affine	none	0.60	Simard et al., ICDAR 2003
Best overall	Convolutional net, CE	Elastic	none	0.40	Simard et al., ICDAR 2003

Best Results on MNIST (from raw images: no preprocessing)

CLASSIFIER	DEFORMATION	ERROR %	Reference
Knowledge-free methods			
2-layer NN, 800 HU, CE		1.60	Simard et al., ICDAR 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton, in press, 2005
SVM, Gaussian Kernel		1.40	Cortes 92 + Many others
Convolutional nets			
Convolutional net LeNet-5,		0.80	LeCun 2005 Unpublished
Convolutional net LeNet-6,		0.70	LeCun 2006 Unpublished
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al., ICDAR 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., ICDAR 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., ICDAR 2003
Convolutional net, CE	Elastic	0.40	Simard et al., ICDAR 2003

Convolutional Nets are the best known method for handwriting recognition

Problems with Supervised Learning in Deep Architectures

- **vanishing gradient, symmetry breaking**
  - The first layers have a hard time learning useful things
  - How to break the symmetry so that different units do different things
- **Idea [Hinton]:**
  - 1 – Initialize the first (few) layers with unsupervised training
  - 2 – Refine the whole network with backprop
- **Problem: How do we train a layer in unsupervised mode?**
  - Auto-encoder: only works when the first layer is smaller than the input
  - What if the first layer is larger than the input?
  - Reconstruction is trivial!
- **Solution: sparse over-complete representations**
  - Keep the number of bits in the first layer low
  - Hinton uses a Restricted Boltzmann Machine in which the first layer uses stochastic binary units

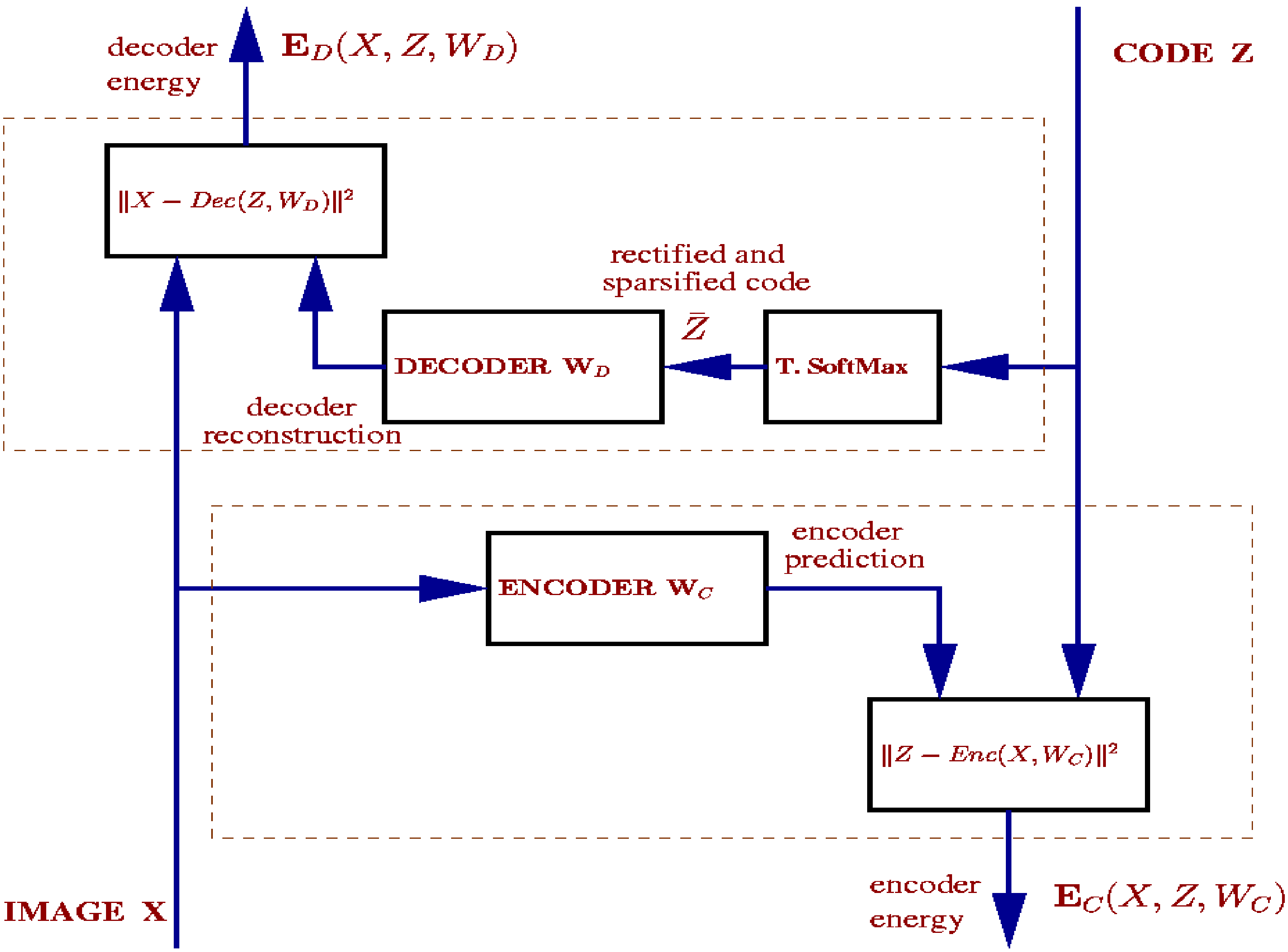
Best Results on MNIST (from raw images: no preprocessing)

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods			
2-layer NN, 800 HU, CE		1.60	Simard et al., ICDAR 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton, in press, 2005
SVM, Gaussian Kernel		1.40	Cortes 92 + Many others
Unsupervised Stacked RBM + backprop		0.95	Hinton, in press, 2005
Convolutional nets			
Convolutional net LeNet-5,		0.80	LeCun 2005 Unpublished
Convolutional net LeNet-6,		0.70	LeCun 2006 Unpublished
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al., ICDAR 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., ICDAR 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., ICDAR 2003
Convolutional net, CE	Elastic	0.40	Simard et al., ICDAR 2003

Unsupervised Learning of Sparse Over-Complete Features

- **Classification is easier with over-complete feature sets**
- **Existing Unsupervised Feature Learning (non sparse/overcomplete):**
  - PCA, ICA, Auto-Encoder, Kernel-PCA
- **Sparse/Overcomplete Methods**
  - Non-Negative Matrix Factorization
  - Sparse-Overcomplete basis functions (Olshausen and Field 1997)
  - Product of Experts (Teh, Welling, Osindero, Hinton 2003)

Symmetric Product of Experts



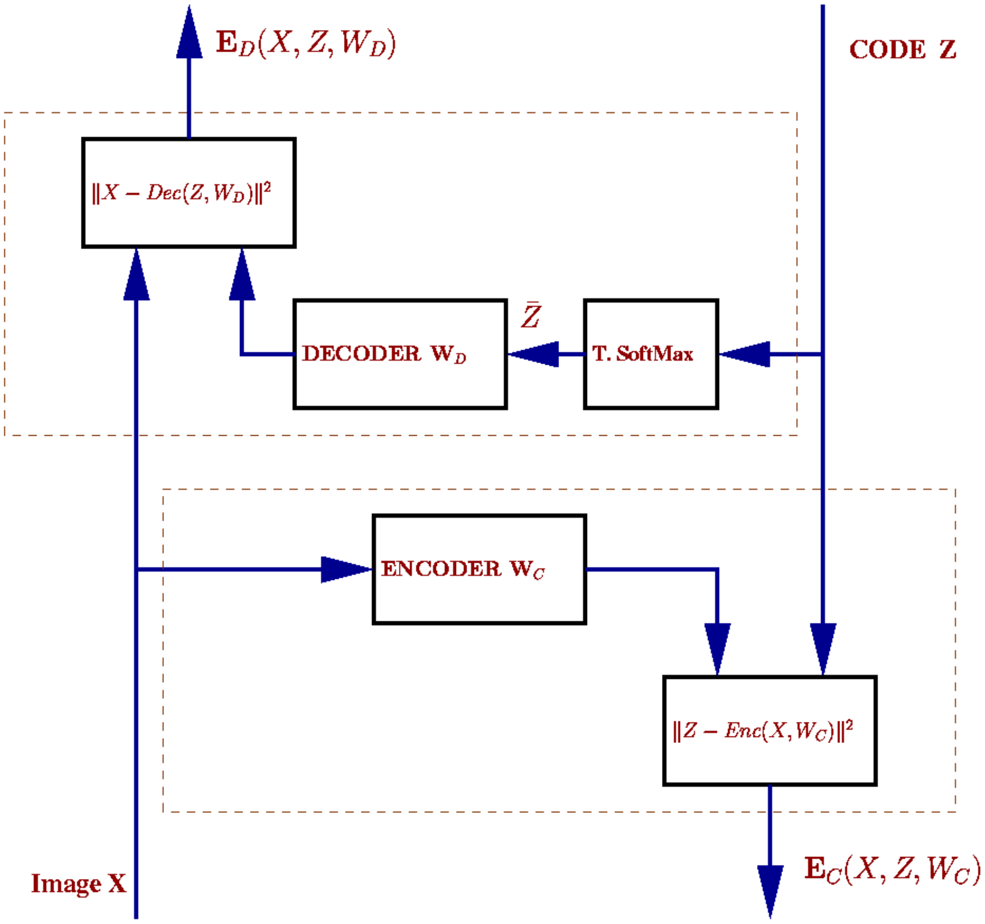
Symmetric Product of Experts

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

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

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

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



Inference & Learning

- *Inference*

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

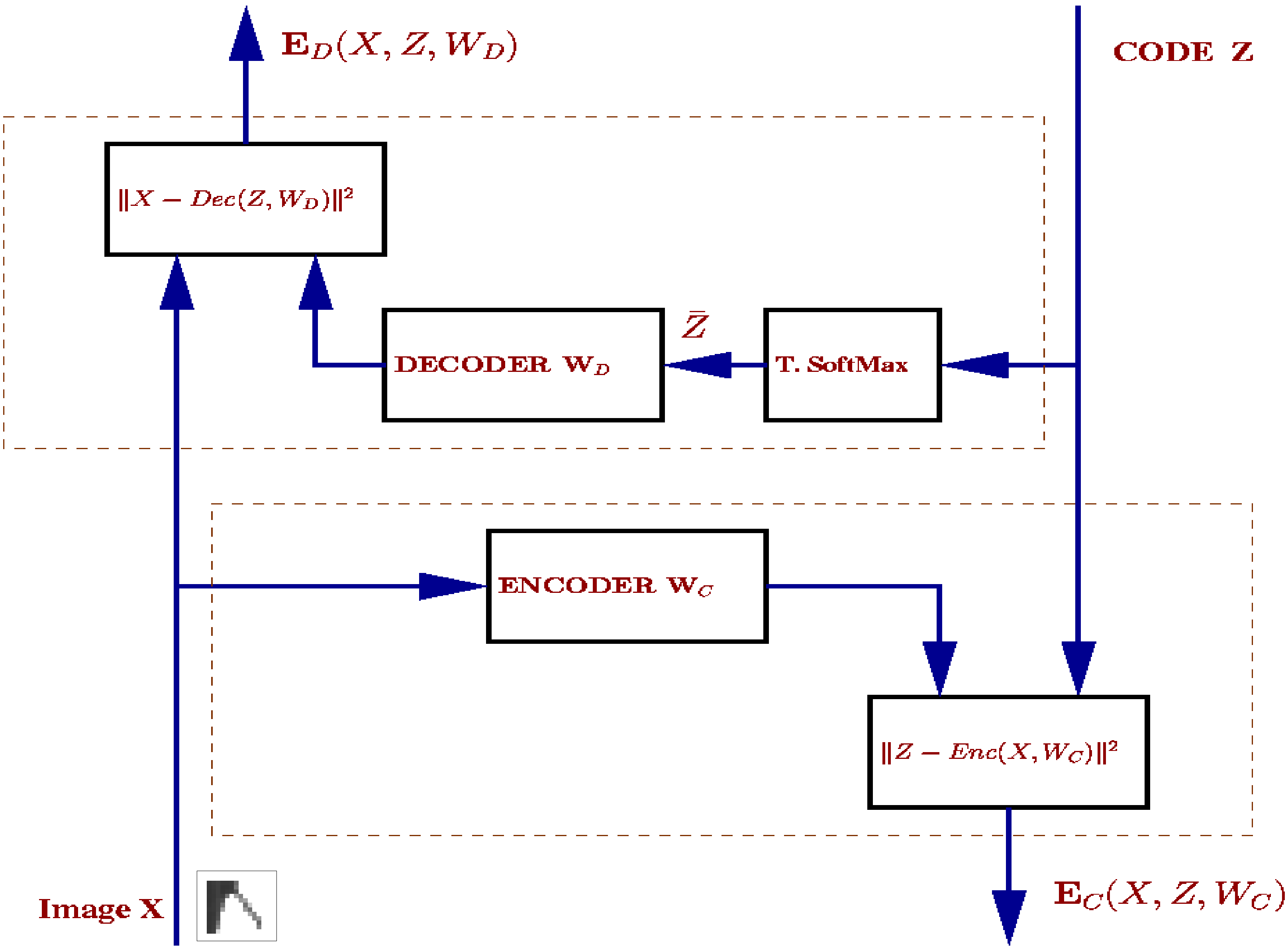
  - let Z(0) be the encoder prediction
  - find code which minimizes total energy
  - gradient descent optimization

- *Learning*

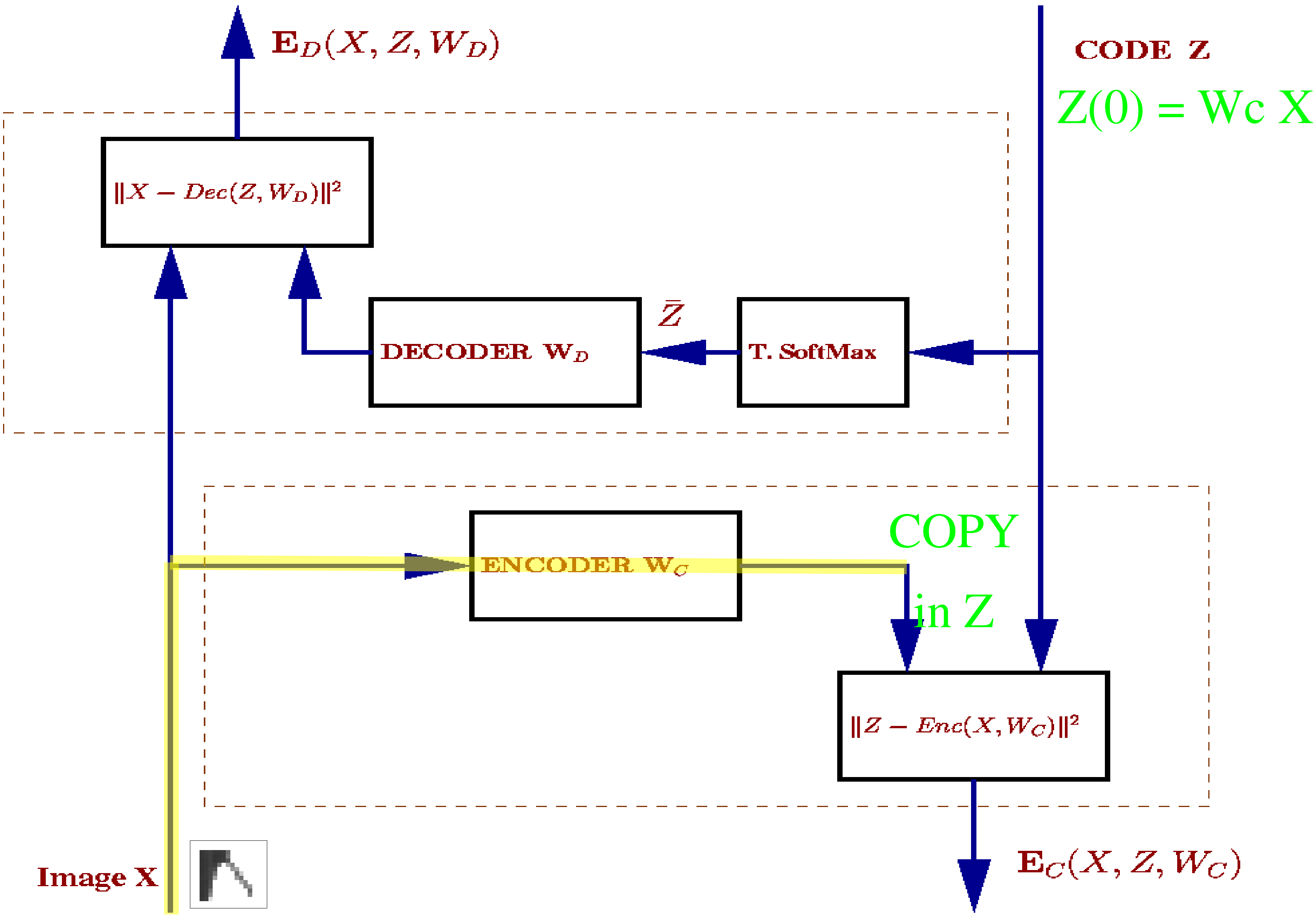
$$W \leftarrow W - \partial E(X, \tilde{Z}, W) / \partial W$$

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

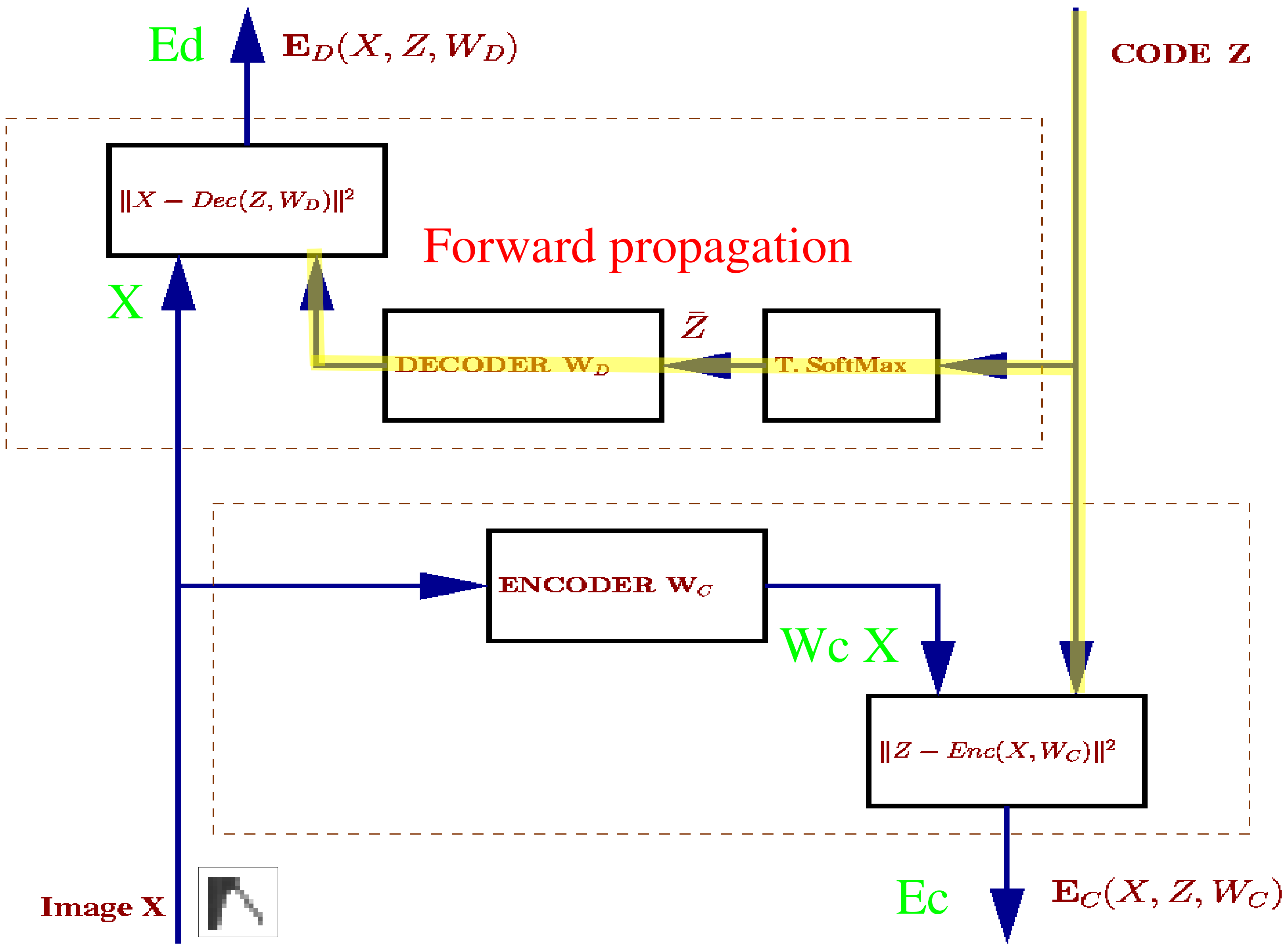
Inference & Learning



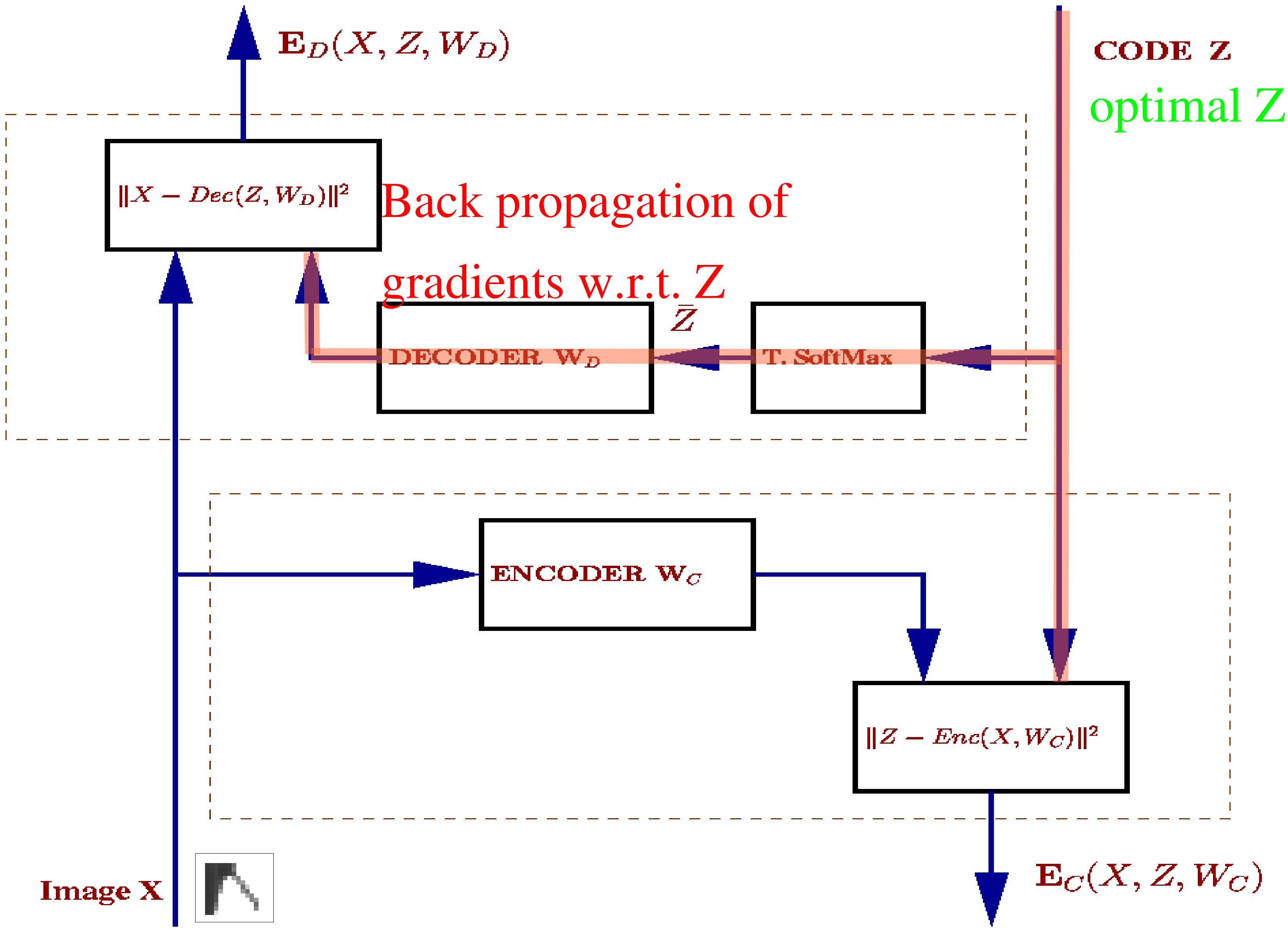
Inference - step 1



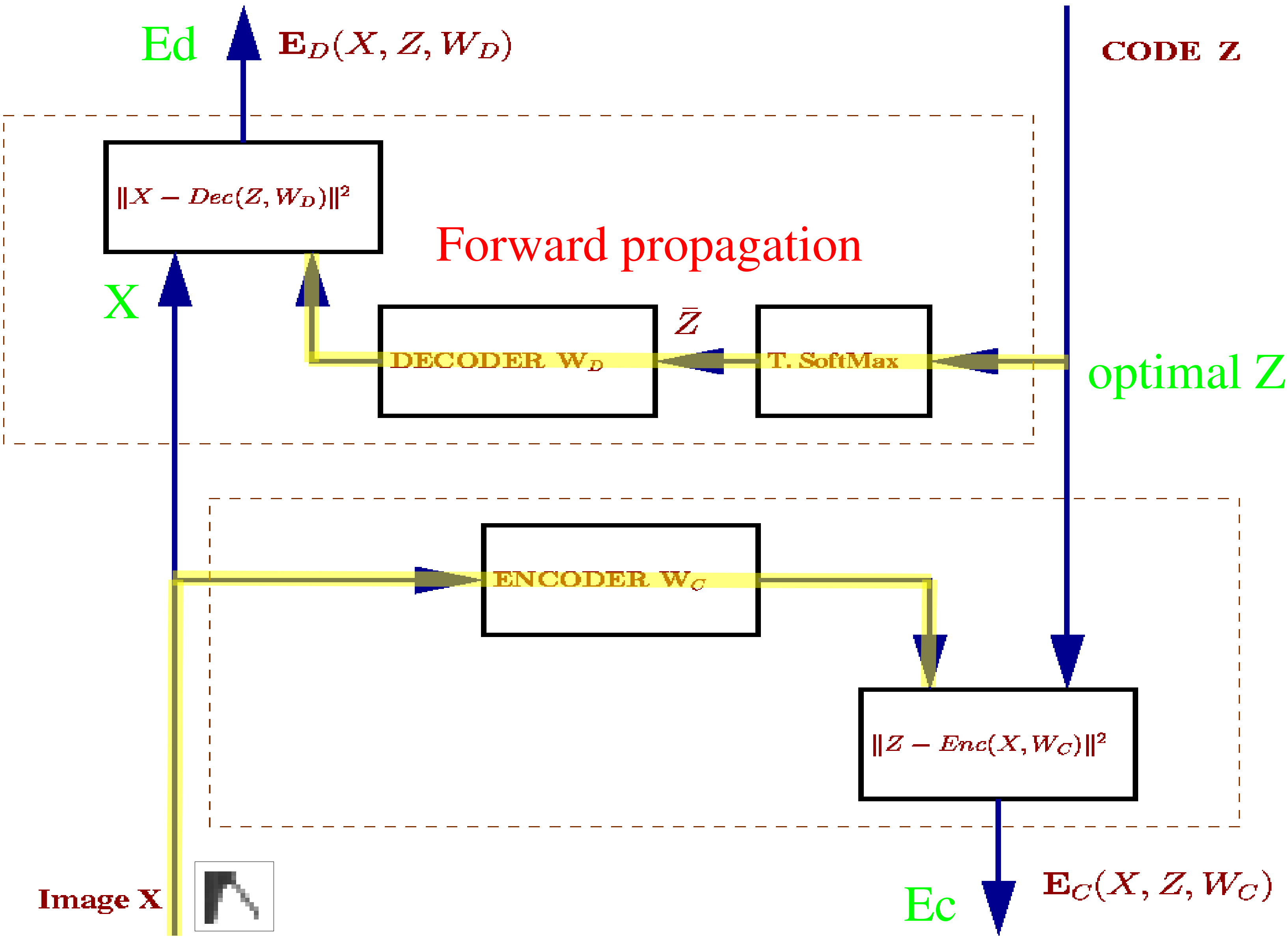
Inference - step 1



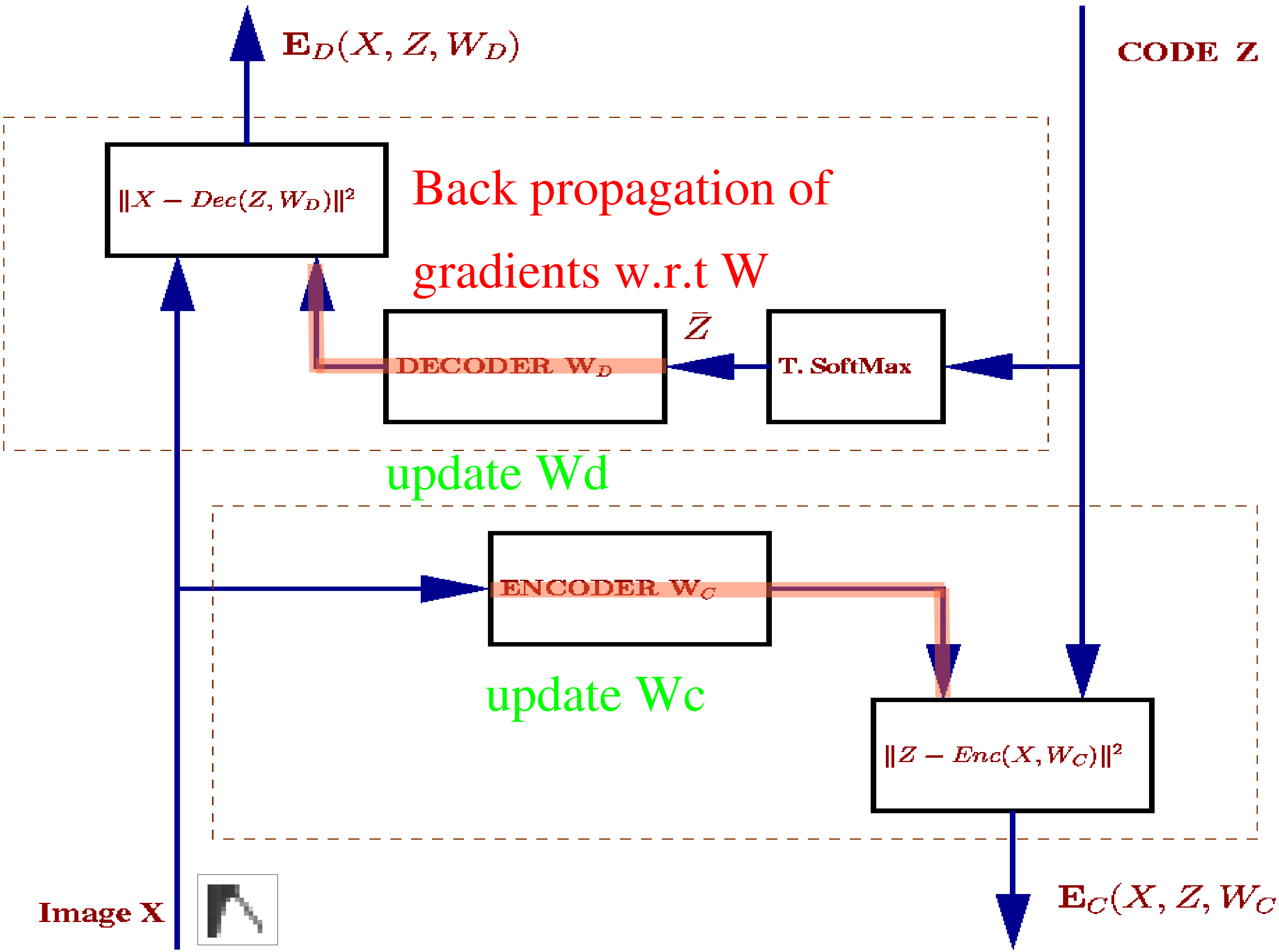
Inference - step 1



Learning - step 2



Learning - step 2



Sparsifying Logistic

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

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

- temporal vs. spatial sparsity

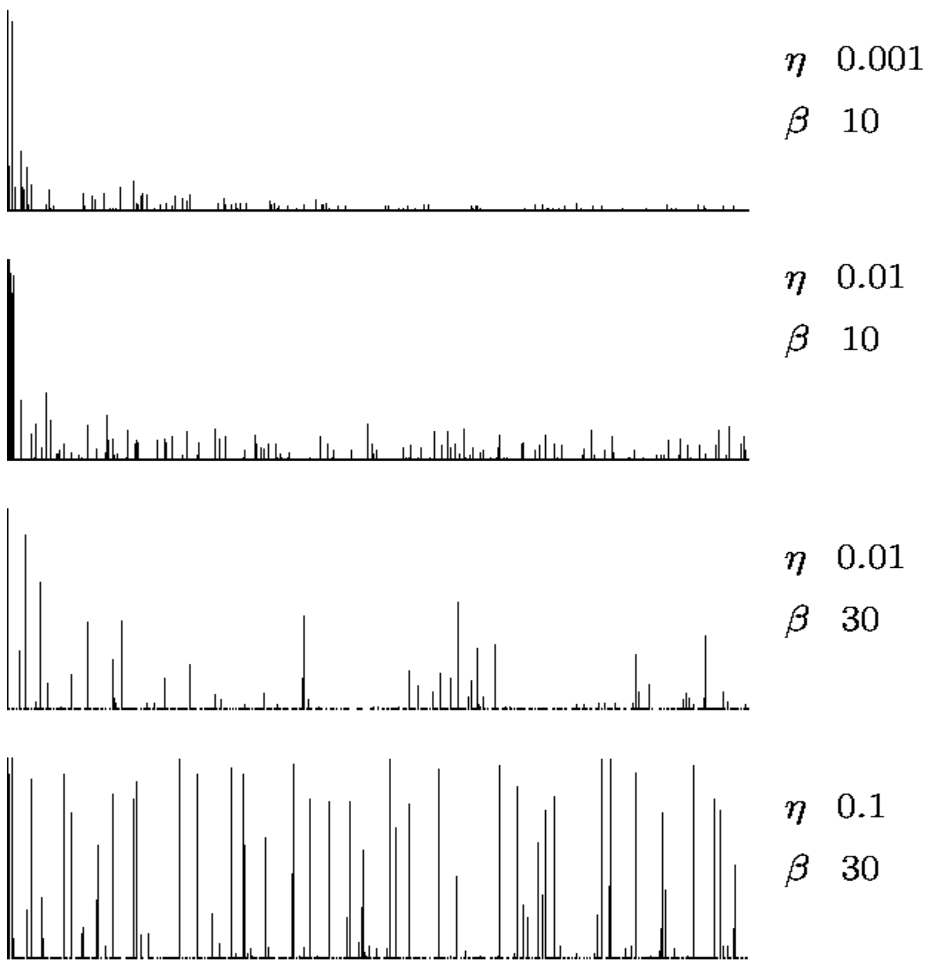
  => no normalization

- ξ is treated as a learned parameter

  => TSM is a sigmoid function with a special bias

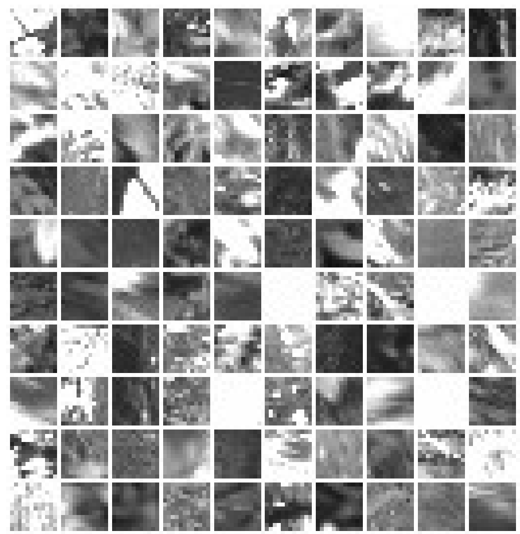
$$\bar{z}_i(t) = \frac{1}{1 + B e^{-\beta z_i(t)}}$$

- ξ is saturated during training to allow units to have different sparseness



input uniformly distributed in [-1,1]

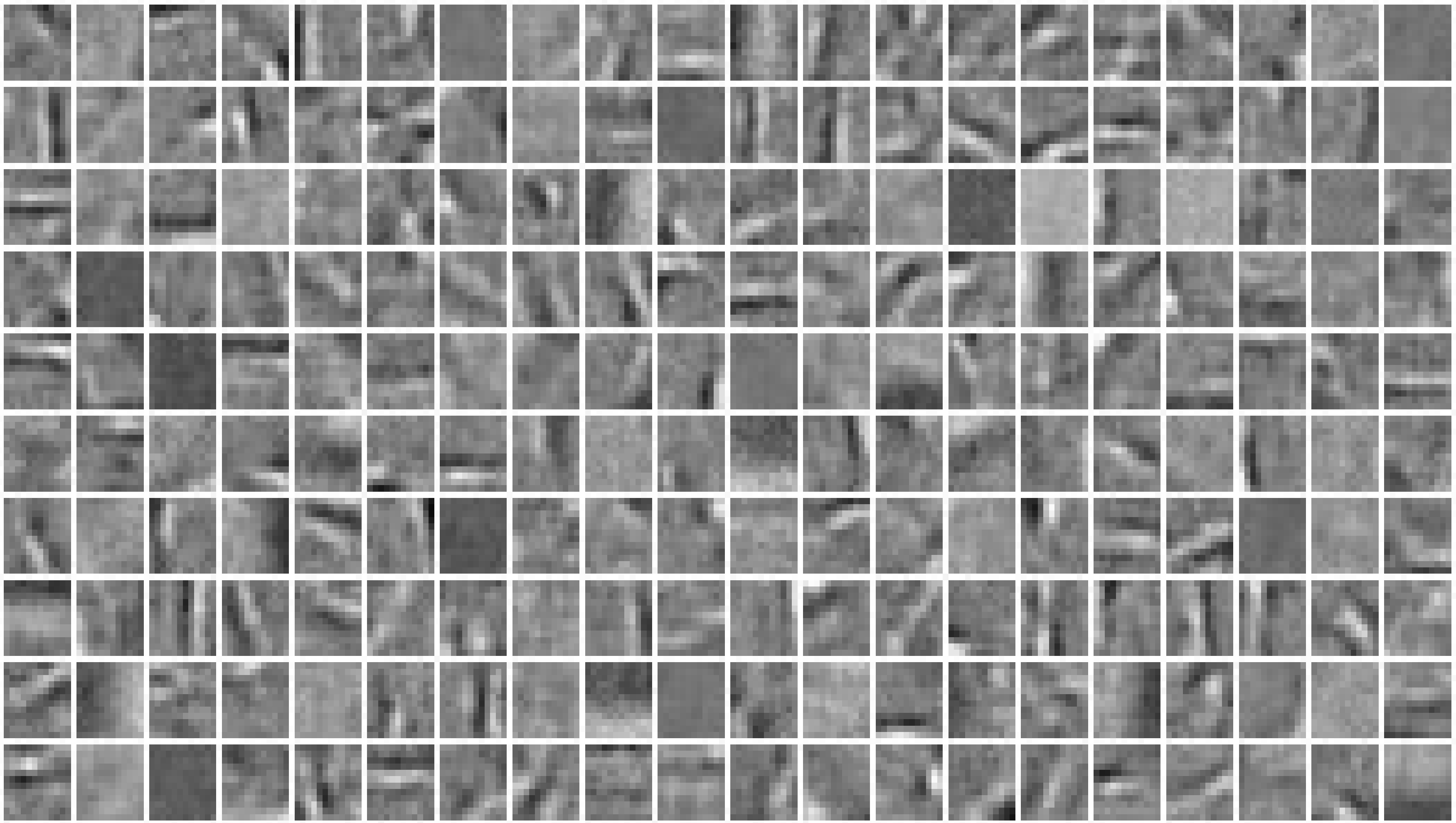
Natural image patches - Berkeley



Berkeley data set

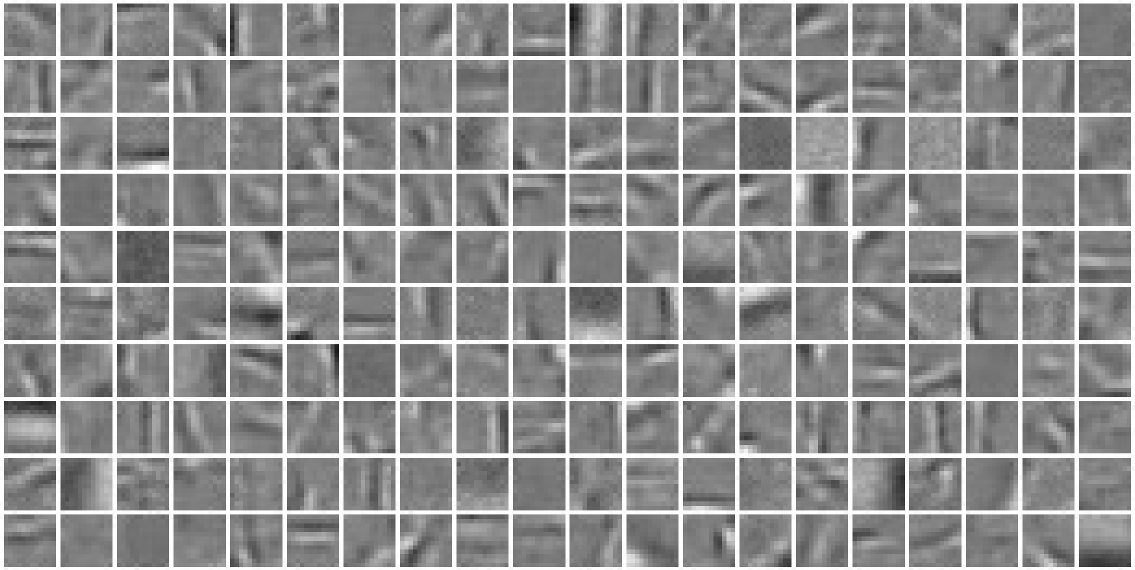
- 100,000 12x12 patches
- 200 units in the code
- η β 0.02
- 1
- learning rate 0.001
- L1, L2 regularizer 0.001
- fast convergence: < 30min.

Natural image patches - Berkeley



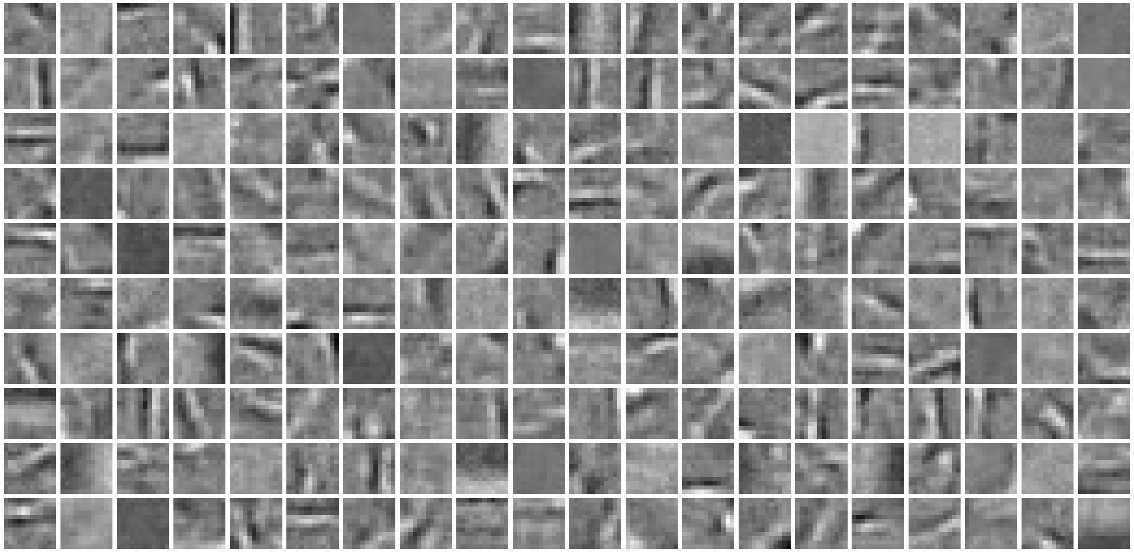
200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

Natural image patches - Berkeley



Encoder *direct* filters

(rows of $\mathbf{W_c}$)

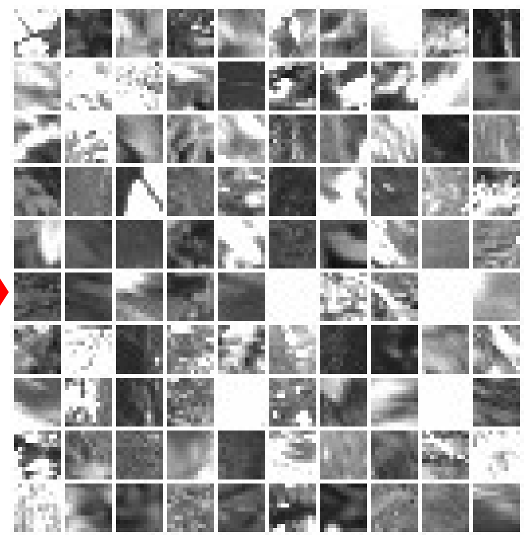


Decoder *reverse* filters

(cols. of $\mathbf{W_d}$)

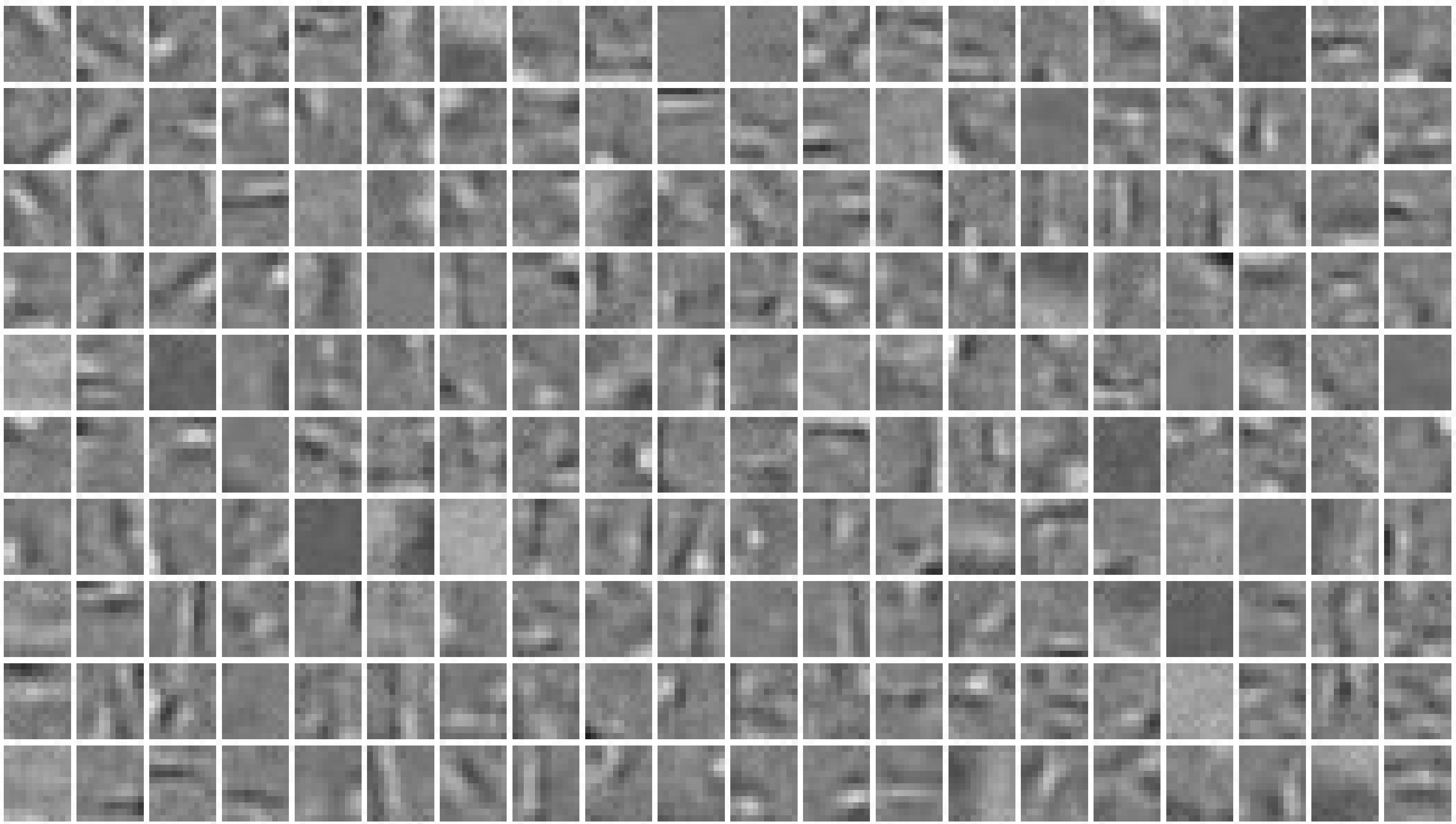
Natural image patches - Forest

Forest data set



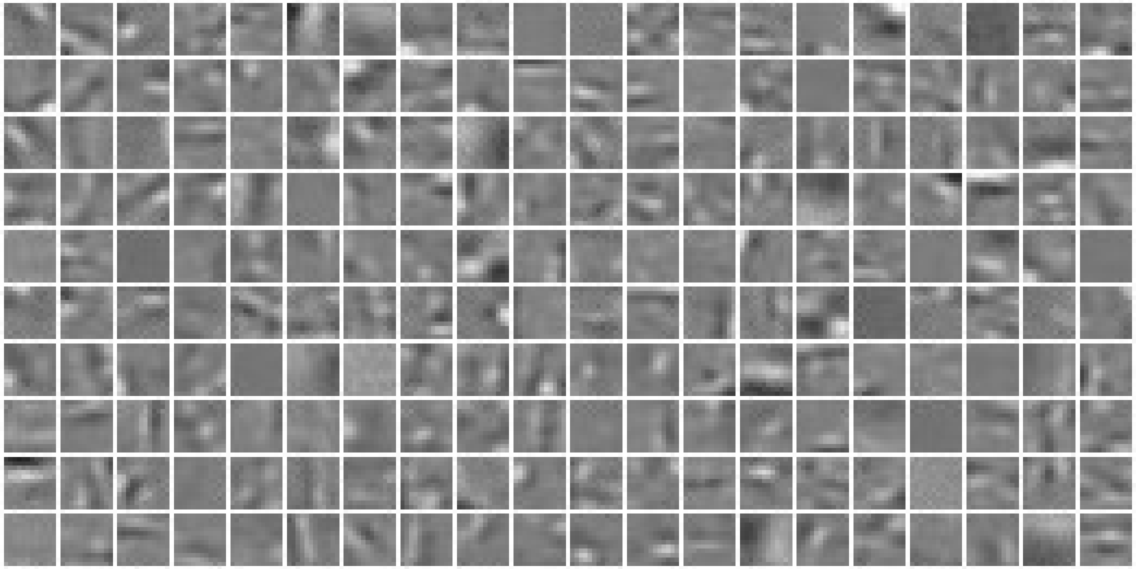
- 100,000 12x12 patches
- 200 units in the code
- η β 0.02
- 1
- learning rate 0.001
- L1, L2 regularizer 0.001
- fast convergence: < 30min.

Natural image patches - Forest



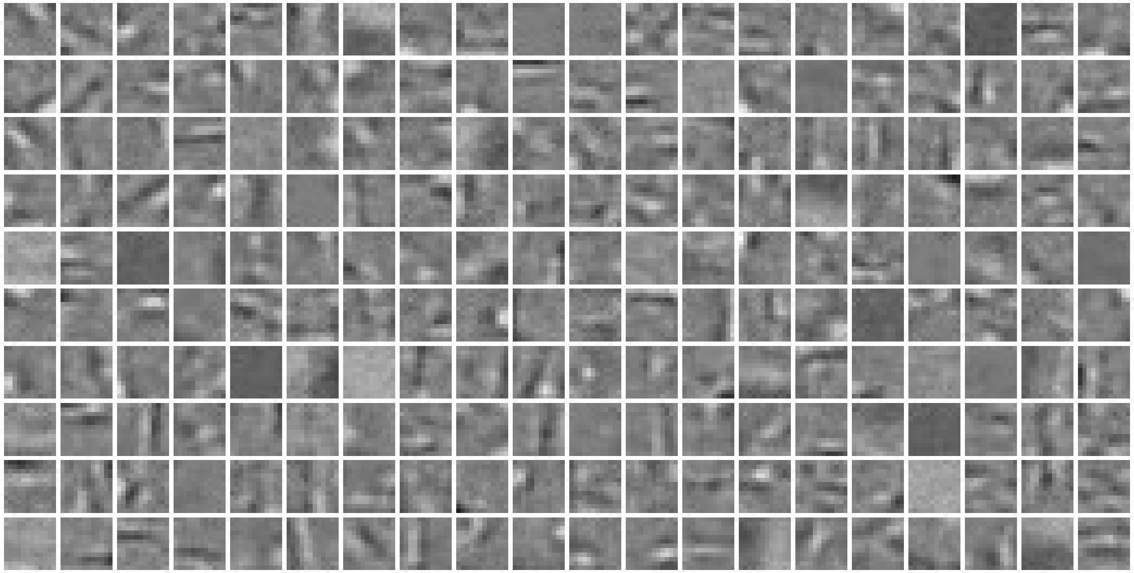
200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

Natural image patches - Forest



Encoder *direct* filters

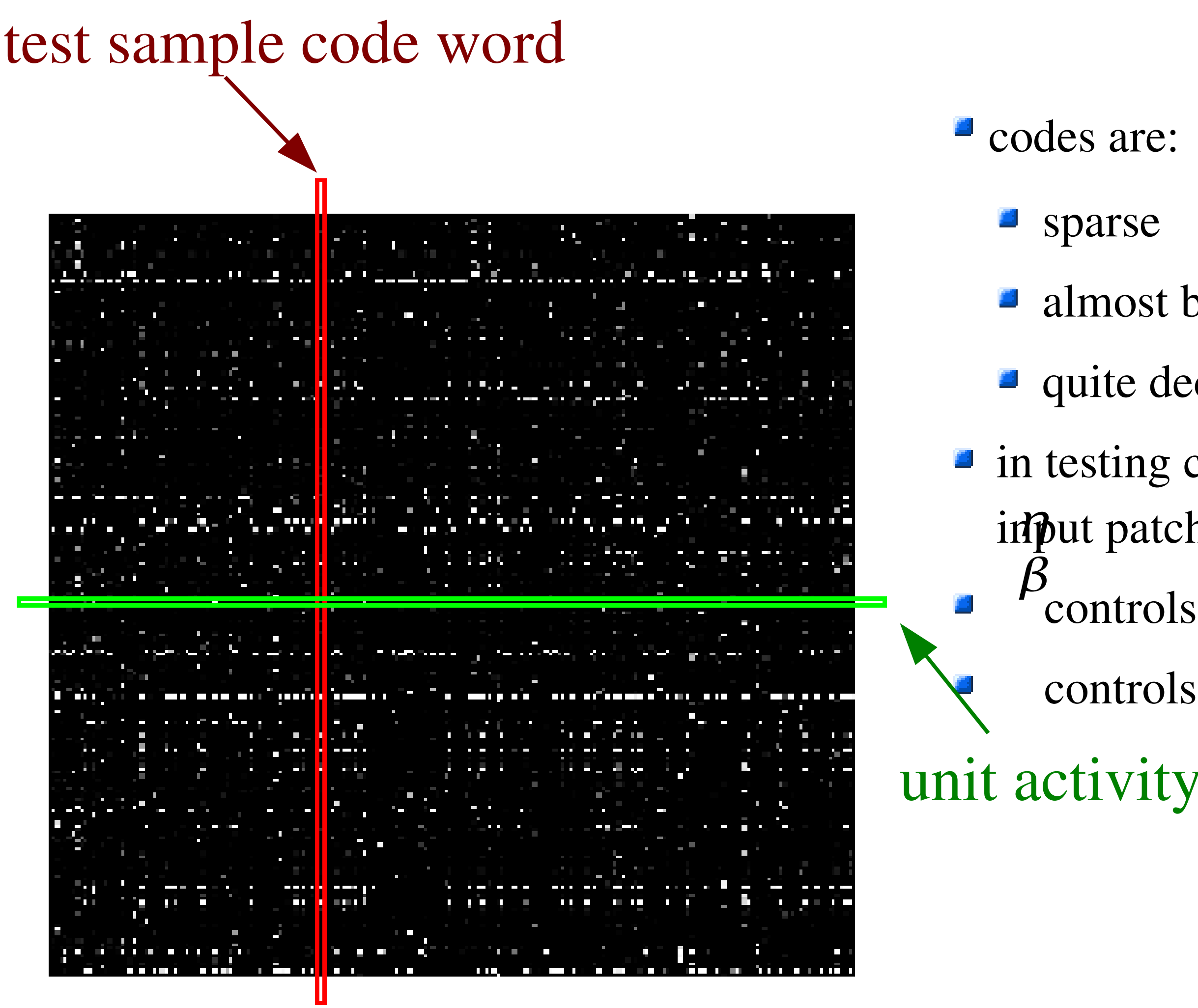
(rows of $\mathbf{W_c}$)



Decoder *reverse* filters

(cols. of $\mathbf{W_d}$)

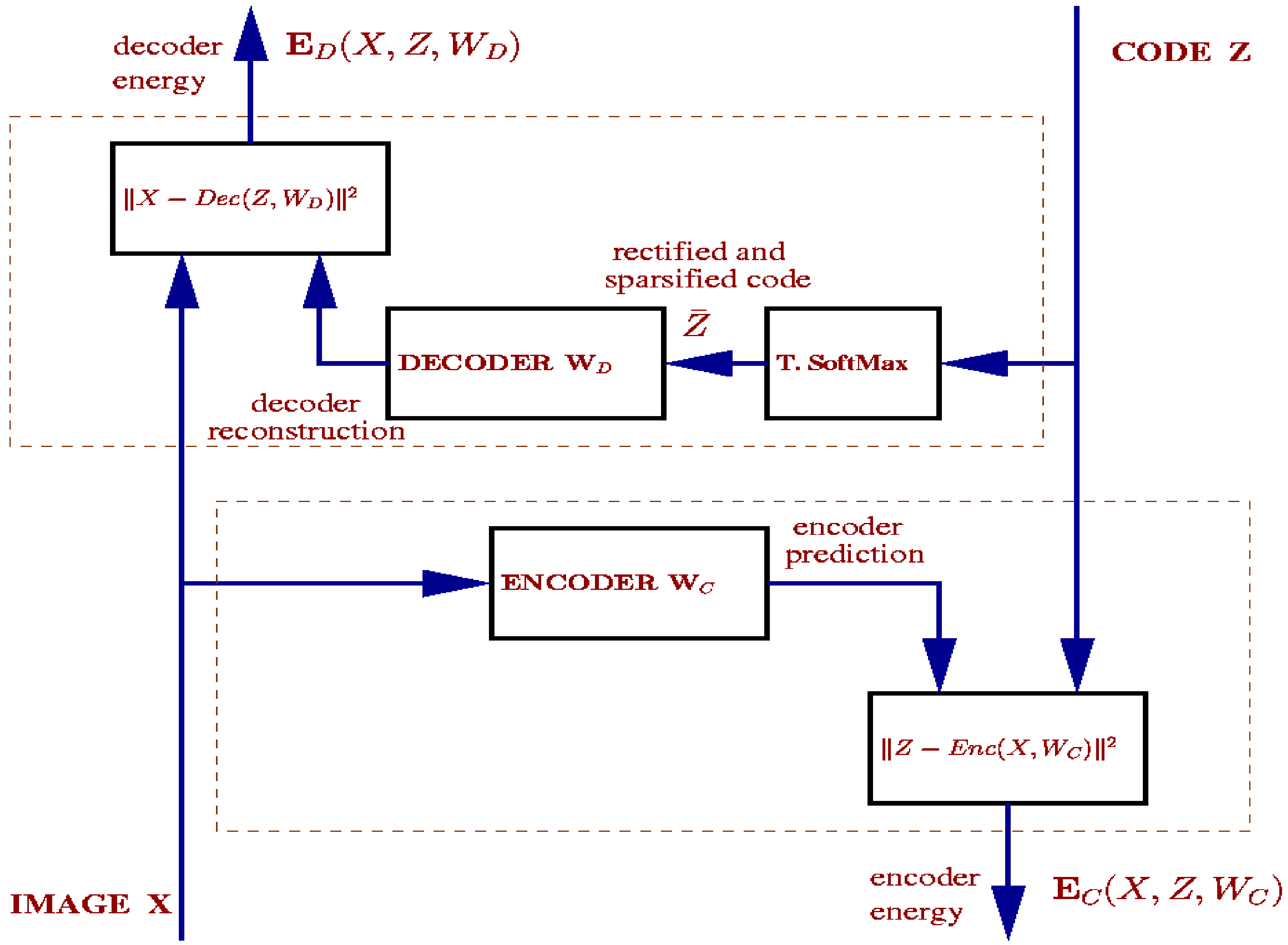
Natural image patches - Forest



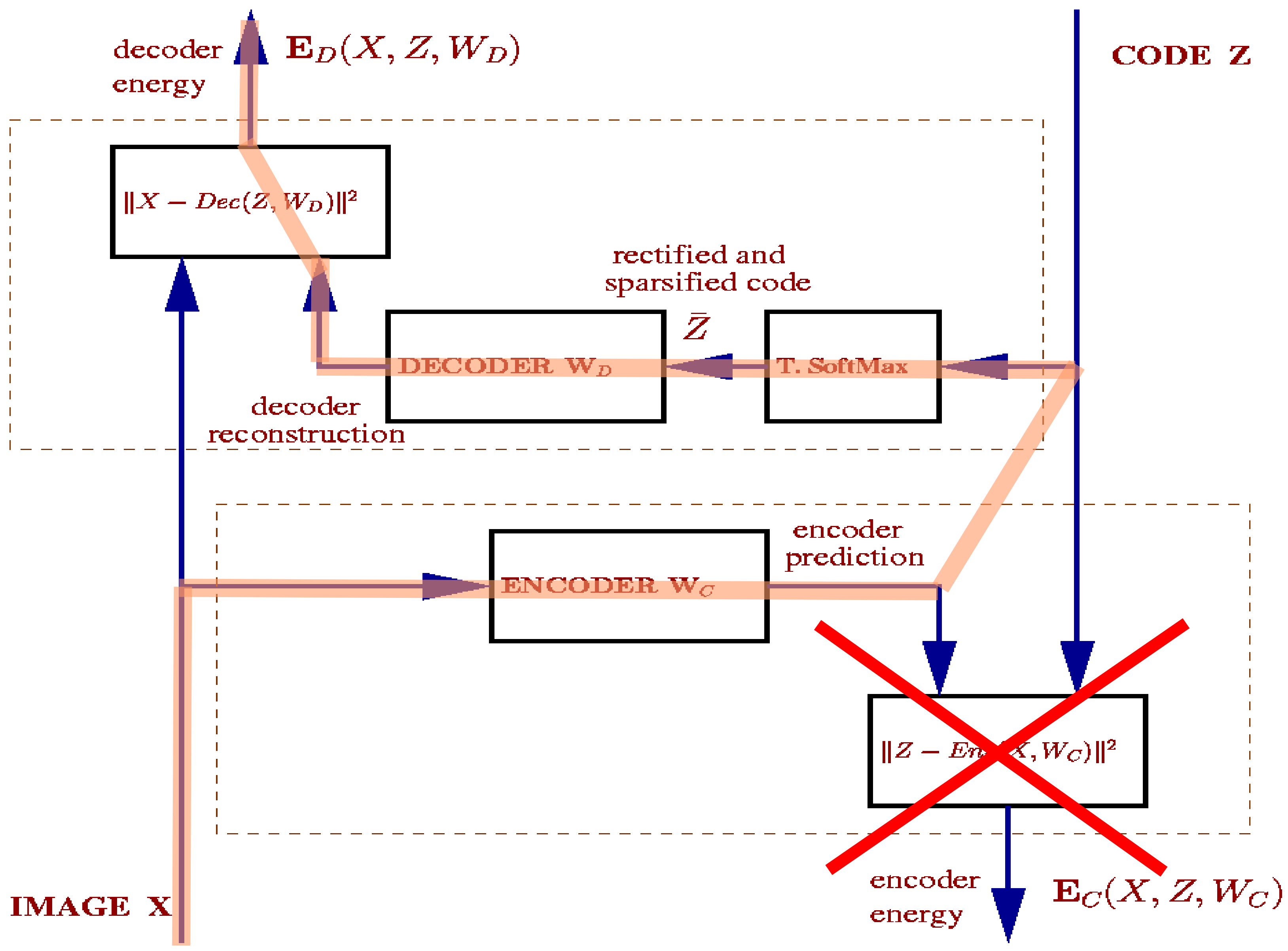
- codes are:
  - sparse
  - almost binary
  - quite decorrelated
- in testing codes are produced by propagating the input patch through encoder and TSM
- η controls sparsity
- β controls the "bit content" in each code unit

code words from 200 randomly selected test patches

What about an autoencoder?

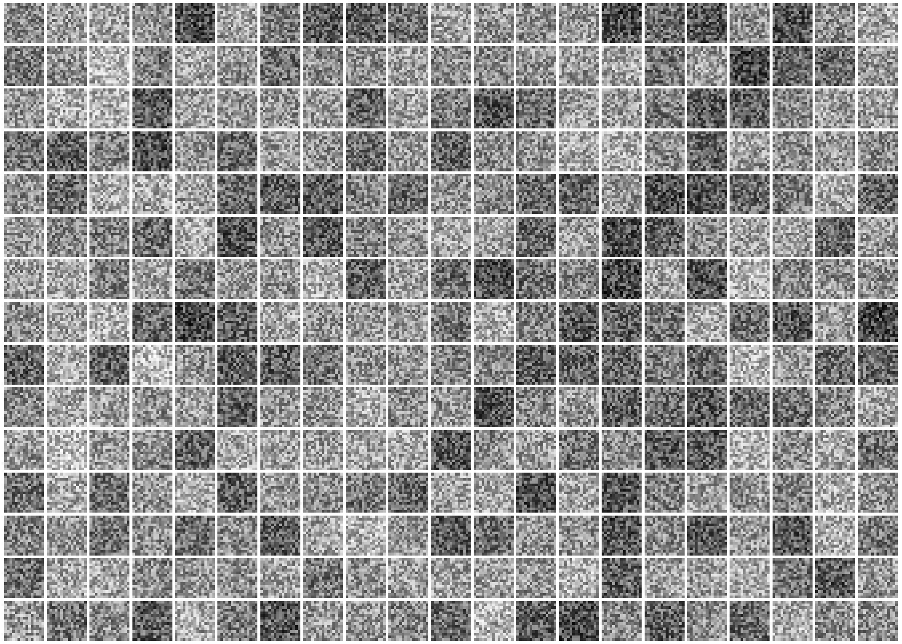


What about an autoencoder?

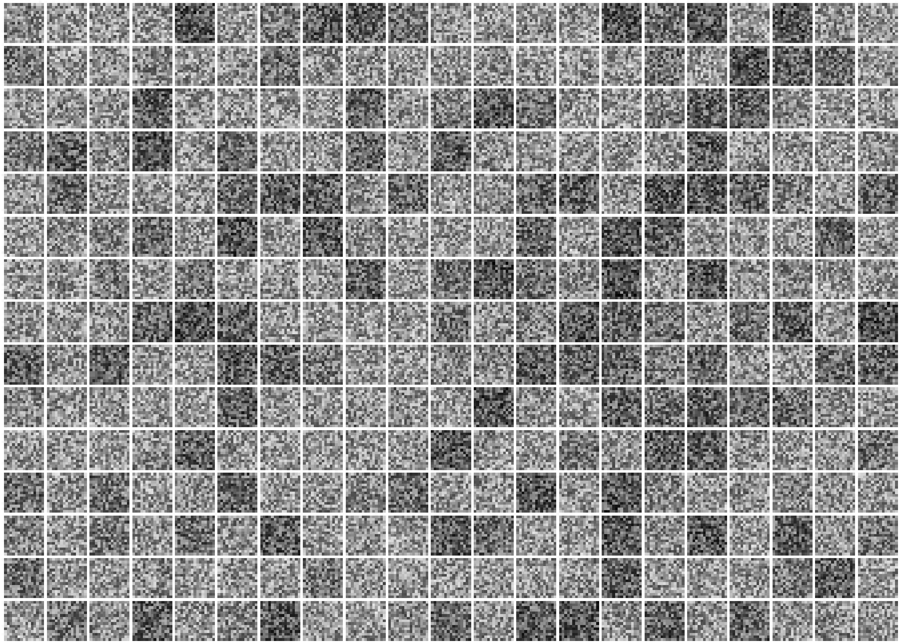


What about an autoencoder?

encoder filters



decoder filters

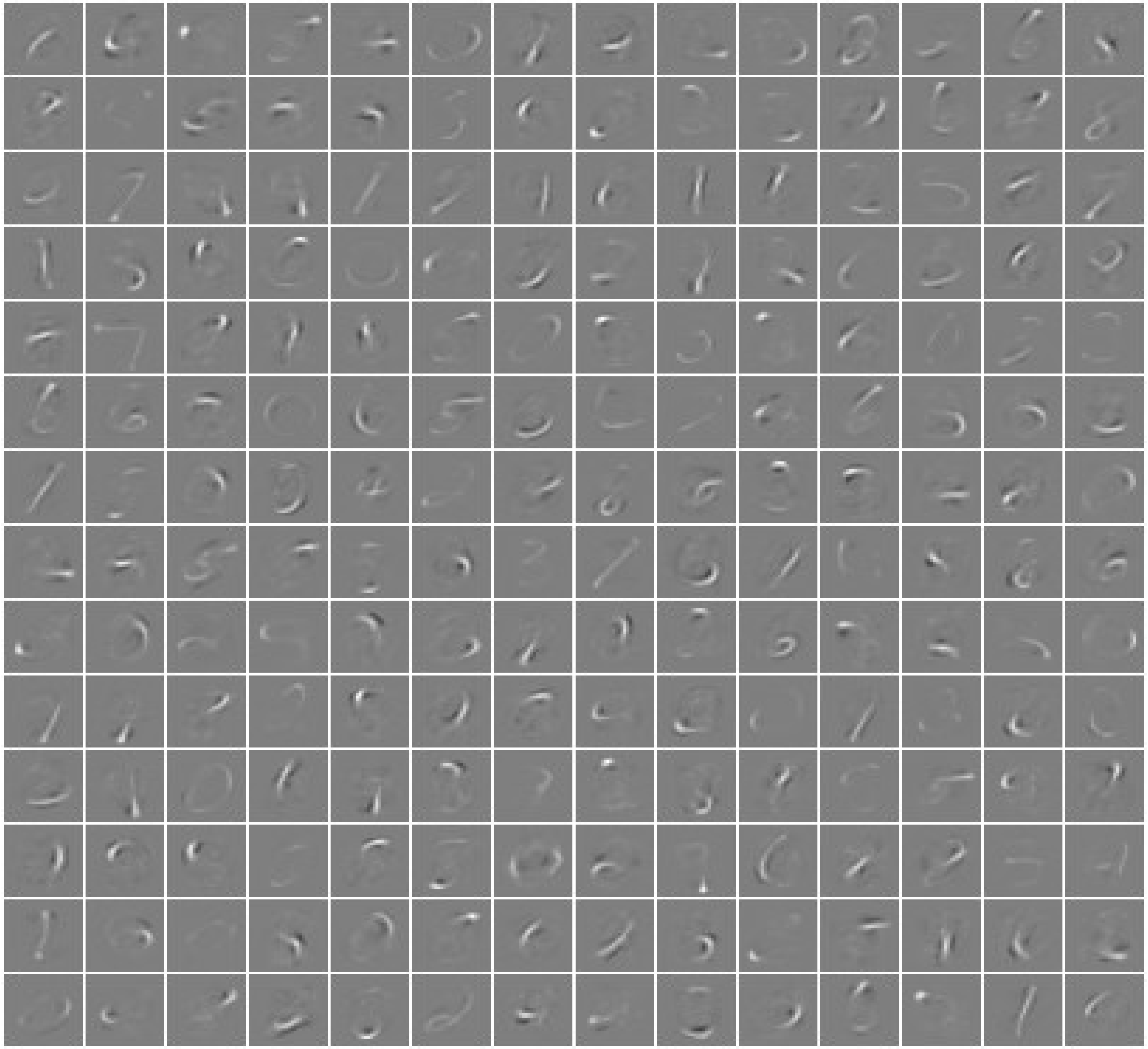


- filters are random
- convergence only for large η and small β

η 0.1

β 0.5

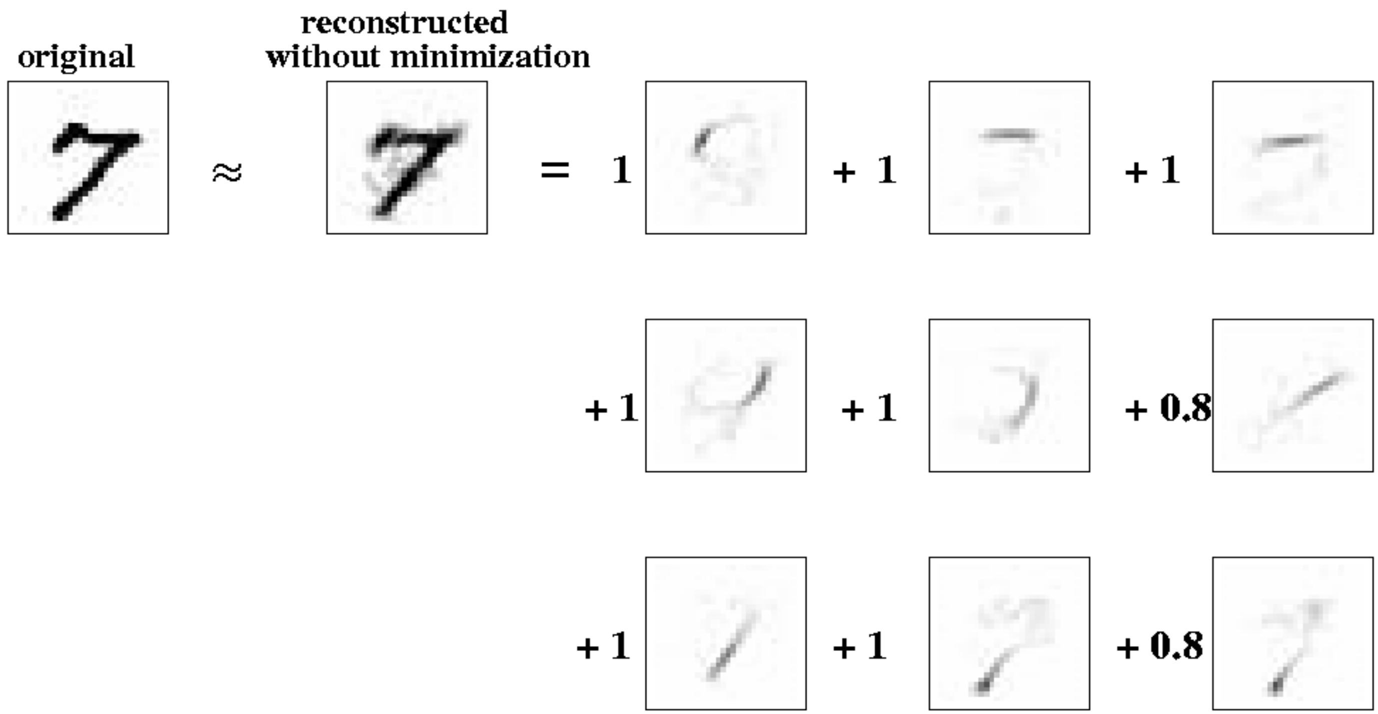
Handwritten digits - MNIST

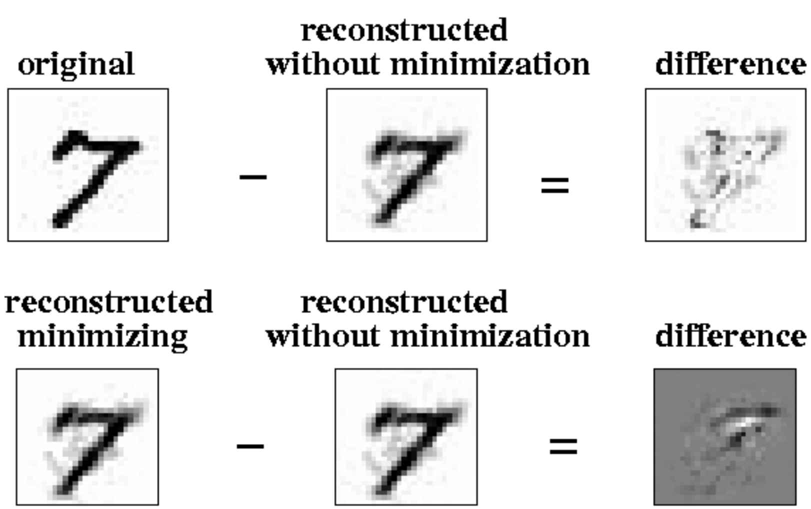


Encoder *direct* filters

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

Handwritten digits - MNIST



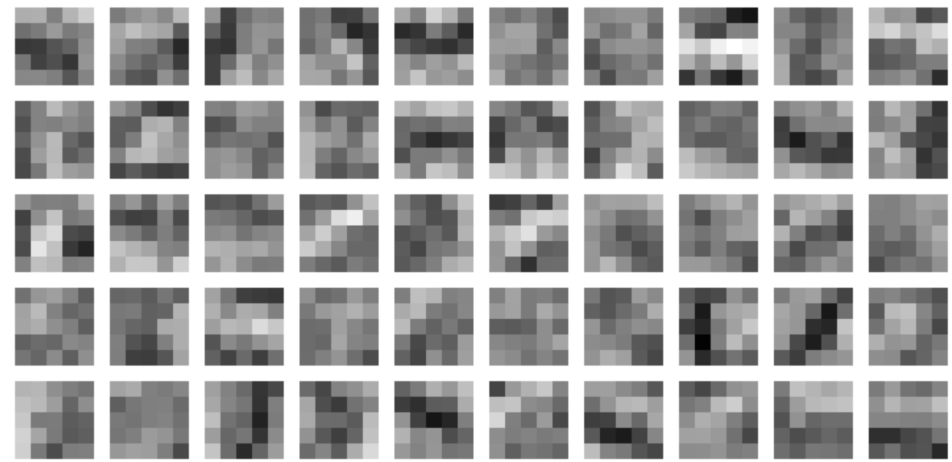


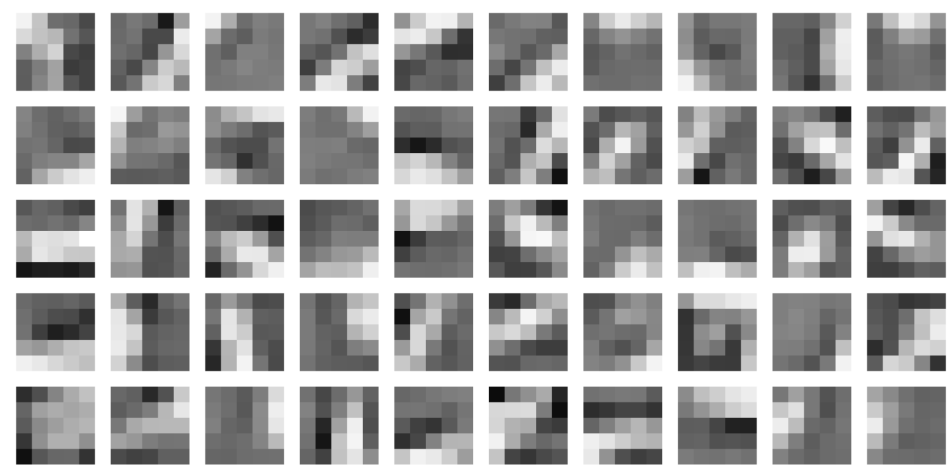
forward propagation through encoder and decoder

after training there is no need to minimize in code space

Initializing a Convolutional Net with SPoE

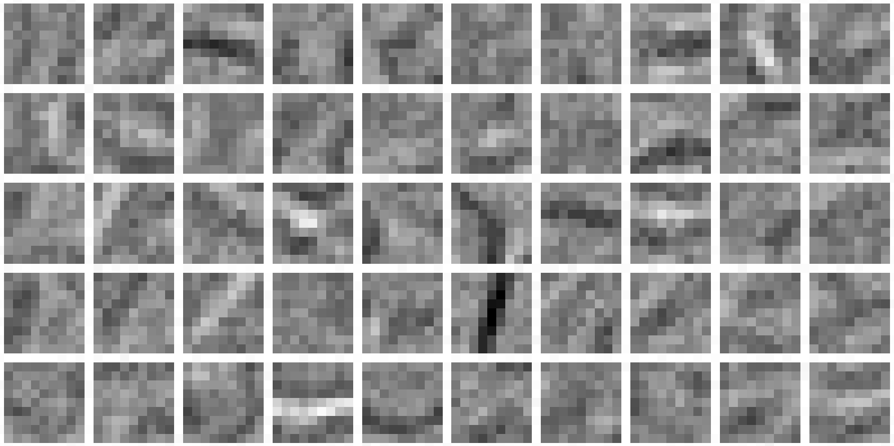
- **Architecture: LeNet-6**
  - 1->50->50->200->10
- **Baseline: random initialization**
  - 0.7% error on test set
- **First Layer Initialized with SpoE**
  - 0.6% error on test set
- **Training with elastically-distorted samples:**
  - 0.38% error on test set



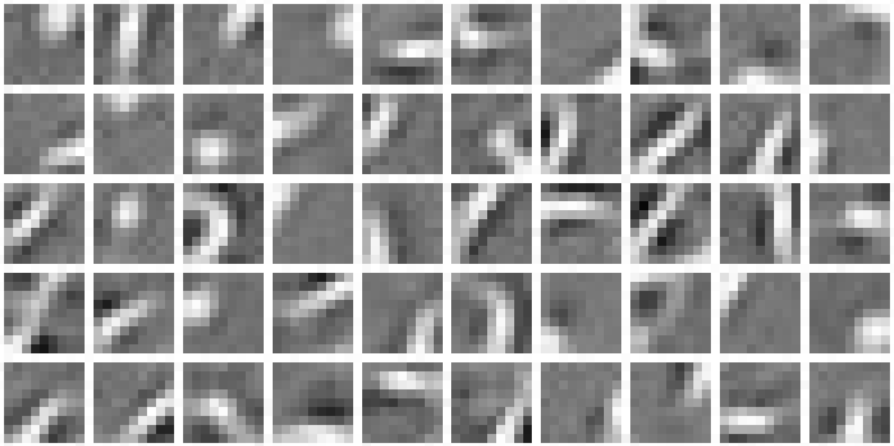


Initializing a Convolutional Net with SPoE

- **Architecture: LeNet-6**
  - 1->50->50->200->10
  - 9x9 kernels instead of 5x5
- **Baseline: random initialization**



- **First Layer Initialized with SpoE**



Best Results on MNIST (from raw images: no preprocessing)

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods			
2-layer NN, 800 HU, CE		1.60	Simard et al., ICDAR 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton, in press, 2005
SVM, Gaussian Kernel		1.40	Cortes 92 + Many others
Unsupervised Stacked RBM + backprop		0.95	Hinton, in press, 2005
Convolutional nets			
Convolutional net LeNet-5,		0.80	LeCun 2005 Unpublished
Convolutional net LeNet-6,		0.70	LeCun 2006 Unpublished
Conv. net LeNet-6- + unsup learning		0.60	LeCun 2006 Unpublished
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al., ICDAR 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., ICDAR 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., ICDAR 2003
Convolutional net, CE	Elastic	0.40	Simard et al., ICDAR 2003
Conv. net LeNet-6- + unsup learning	Elastic	0.38	LeCun 2006 Unpublished

Conclusion

- Deep architectures are better than shallow ones
- We haven't solved the deep learning problem yet
- Larger networks are better
- Initializing the first layer(s) with unsupervised learning helps
- WANTED: a learning algorithm for deep architectures that seamlessly blends supervised and unsupervised learning