

Learning a Deep Hierarchy of Sparse and Invariant Features

Yann LeCun

The Courant Institute of Mathematical Sciences

New York University

Collaborators:

Marc'Aurelio Ranzato, Y-Lan Boureau, Fu-Jie Huang, Sumit Chopra

See: [Ranzato et al. NIPS 2007], [Ranzato et al. CVPR 2007],
[Ranzato et al. AI-Stats 2007], [Ranzato et al. NIPS 2006]
[Bengio & LeCun “Scaling Learning Algorithms towards AI, 2007]
<http://yann.lecun.com/exdb/publis/>

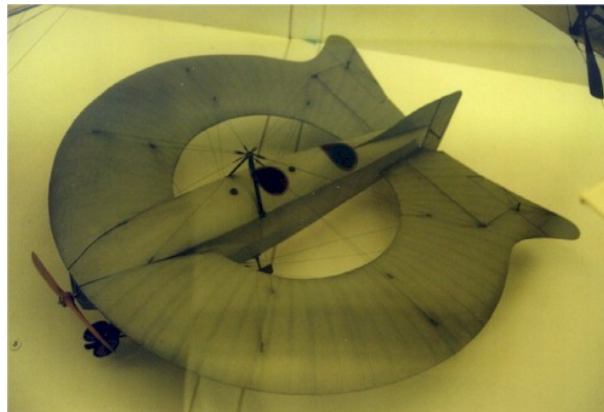
Challenges of Computer Vision (and Visual Neuroscience)

• How do we learn “invariant representations”?

- ▶ From the image of an airplane, how do we extract a representation that is invariant to pose, illumination, background, clutter, object instance....
- ▶ How can a human (or a machine) learn those representations by just looking at the world?

• How can we learn visual categories from just a few examples?

- ▶ I don't need to see many airplanes before I can recognize every airplane (even really weird ones)



The visual system is “deep” and learned

• The primate's visual system is “deep”

- ▶ It has 10-20 layers of neurons from the retina to the infero-temporal cortex (where object categories are encoded).
- ▶ How can it train itself by just looking at the world?

• Is there a magic bullet for visual learning?

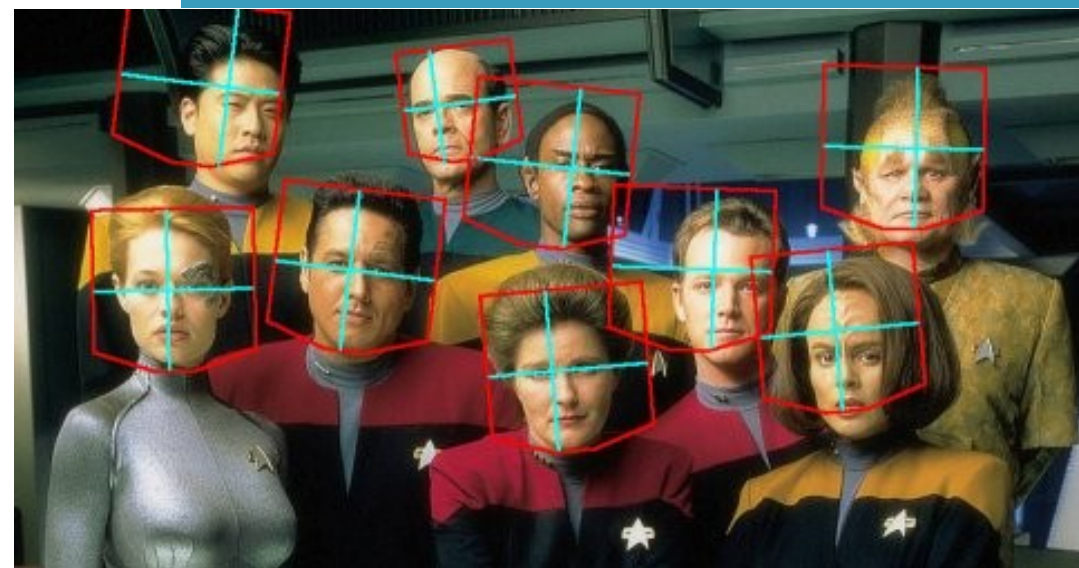
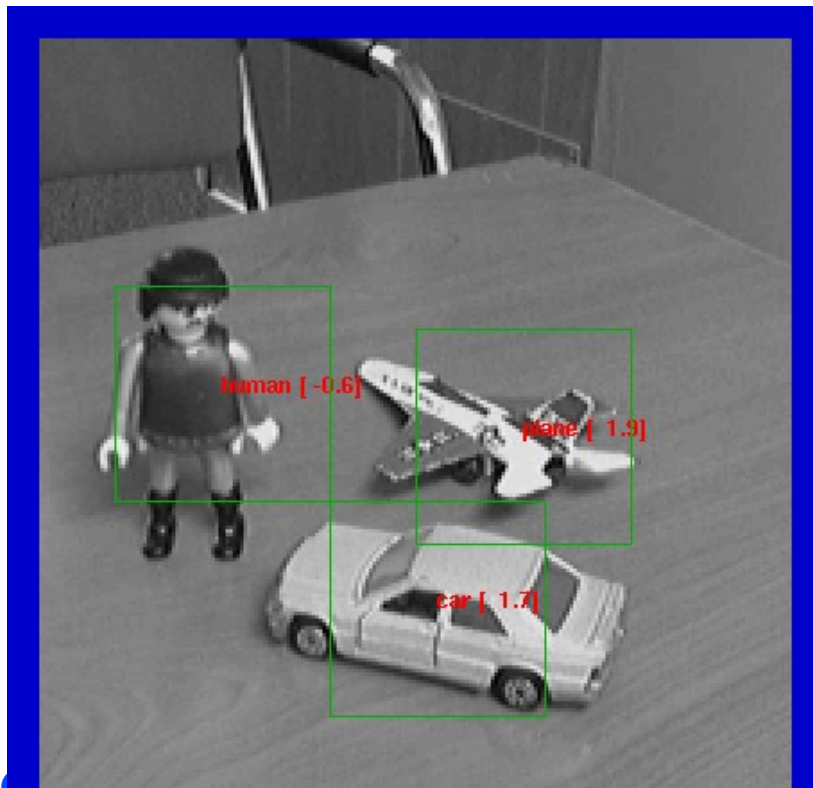
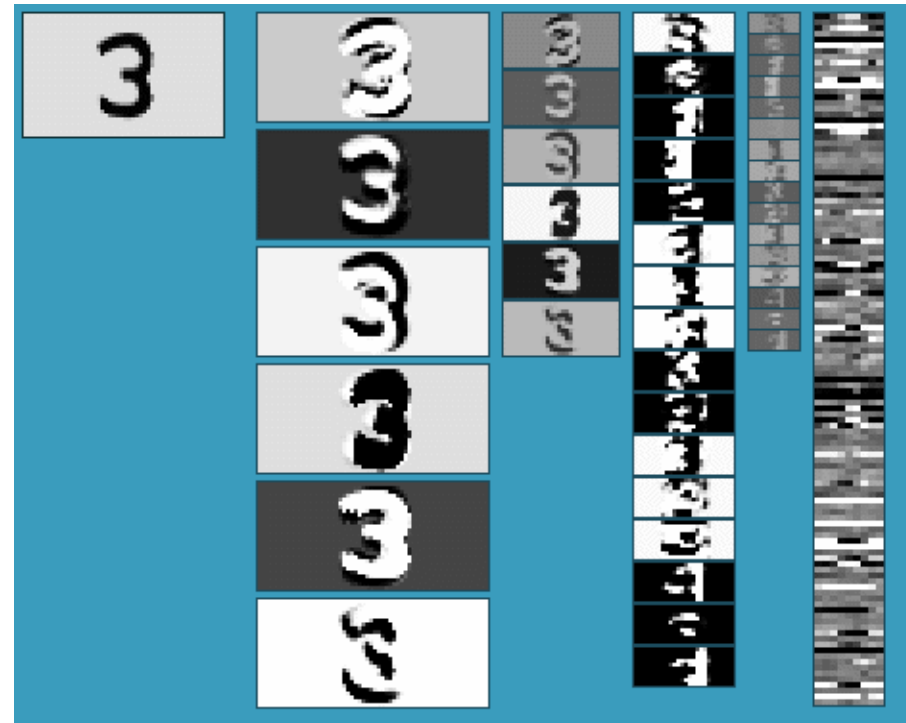
- ▶ The neo-cortex is pretty much the same all over
- ▶ The “learning algorithm” it implements is not specific to a modality (what works for vision works for audition)
- ▶ There is evidence that everything is learned, down to low-level feature detectors in V1
- ▶ Is there a **universal learning algorithm/architecture** which, given a small amount of appropriate prior structure, can produce an intelligent vision system?
- ▶ Or do we have to keep accumulating a large repertoire of pre-engineered “modules” to solve every specific problem an intelligent vision system must solve?

Deep Supervised Learning works well with lots of data

Supervised Convolutional nets work

very well for:

- ▶ handwriting recognition (winner on MNIST)
- ▶ face detection
- ▶ object recognition with few classes and lots of training samples



Learning Deep Feature Hierarchies

• The visual system is deep, and is learned

- ▶ How do we learn deep hierarchies of invariant features?

• On recognition tasks **with lots of training samples**, deep supervised architecture outperform shallow architectures in speed and accuracy

• Handwriting Recognition:

- ▶ raw MNIST: 0.62% for convolutional nets [Ranzato 07]
- ▶ raw MNIST: 1.40% for SVMs [Cortes 92]
- ▶ distorted MNIST: 0.40% for conv nets [Simard 03, Ranzato 06]
- ▶ distorted MNIST: 0.67% for SVMs [Bordes 07]

• Object Recognition

- ▶ small NORB: 6.0% for conv nets [Huang 05]
- ▶ small NORB: 11.6% for SVM [Huang 05]
- ▶ big NORB: 7.8% for conv nets [Huang 06]
- ▶ big NORB: 43.3% for SVM [Huang 06]

Learning Deep Feature Hierarchies

- On recognition tasks **with few labeled samples**, deep supervised architectures don't do so well
 - ▶ a purely supervised convolutional net gets only 20% correct on Caltech-101 with 30 training samples/class
- We need **unsupervised** learning methods that can learn invariant feature hierarchies
- This talk will present methods to learn hierarchies of **sparse and invariant features**
- **Sparse features are good for two reasons:**
 - ▶ they are easier to deal with for a classifier
 - ▶ we will show that using sparsity constraints is a way to upper bound the partition function.

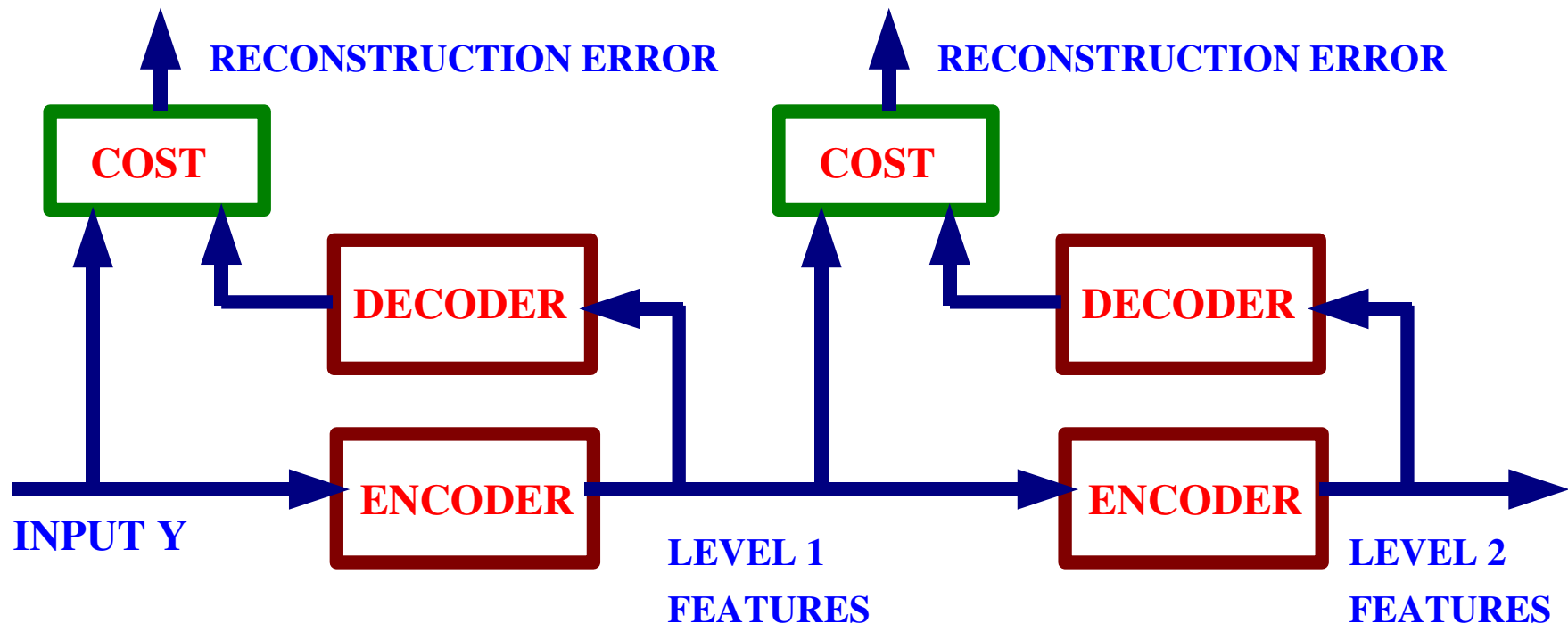
The Basic Idea for Training Deep Feature Hierarchies

Each stage is composed of

- ▶ an encoder that produces a feature vector from the input
- ▶ a decoder that reconstruct the input from the feature vector
 - (RBM is a special case)

Each stage is trained one after the other

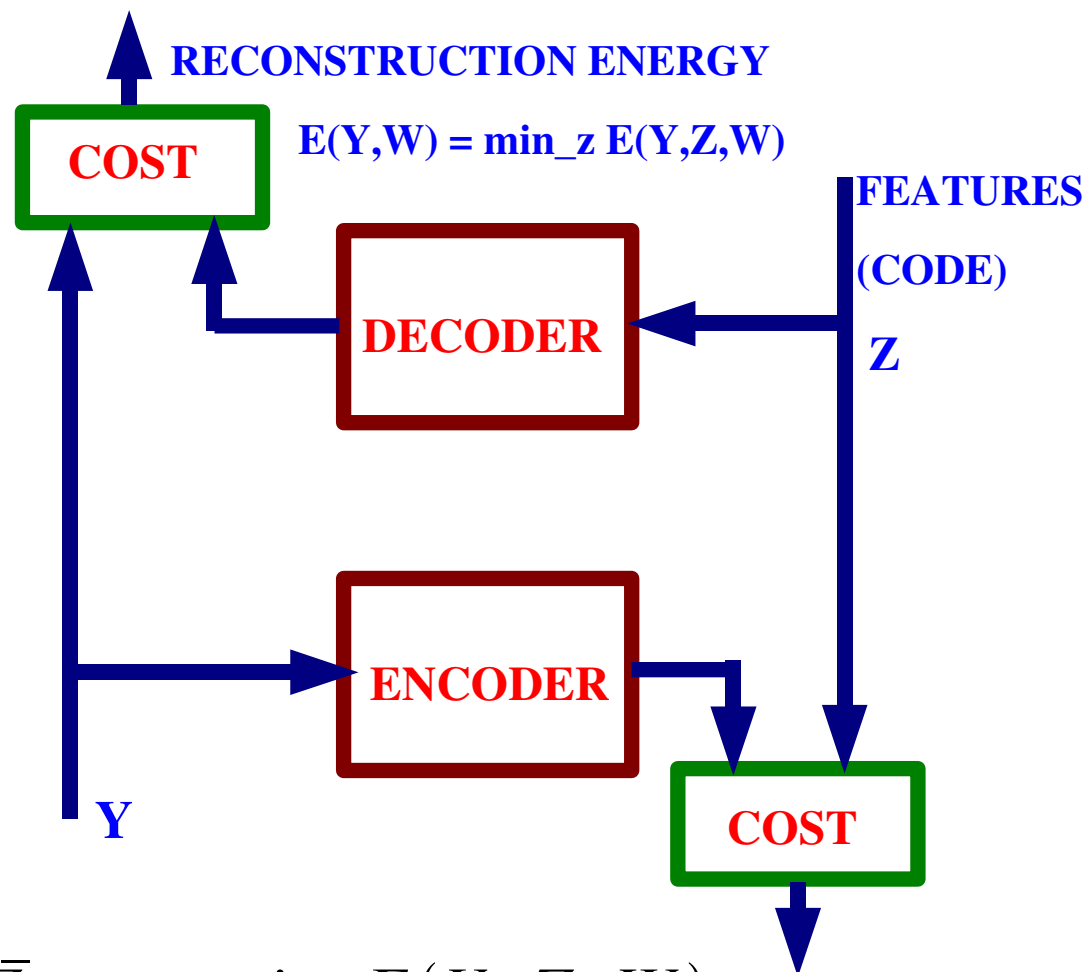
- ▶ the input to stage $i+1$ is the feature vector of stage i .



Feature Learning with the Encoder-Decoder Architecture

- A principle on which unsupervised algorithms can be built is **reconstruction of the input from a code (feature vector)**

- ▶ RBM: $E(Y, Z, W) = -Y'WZ$
- ▶ PCA: $E(Y) = ||Y - W'WY||^2$
- ▶ K-means, Olshausen-Field, Hoyer-Hyvarinen.....

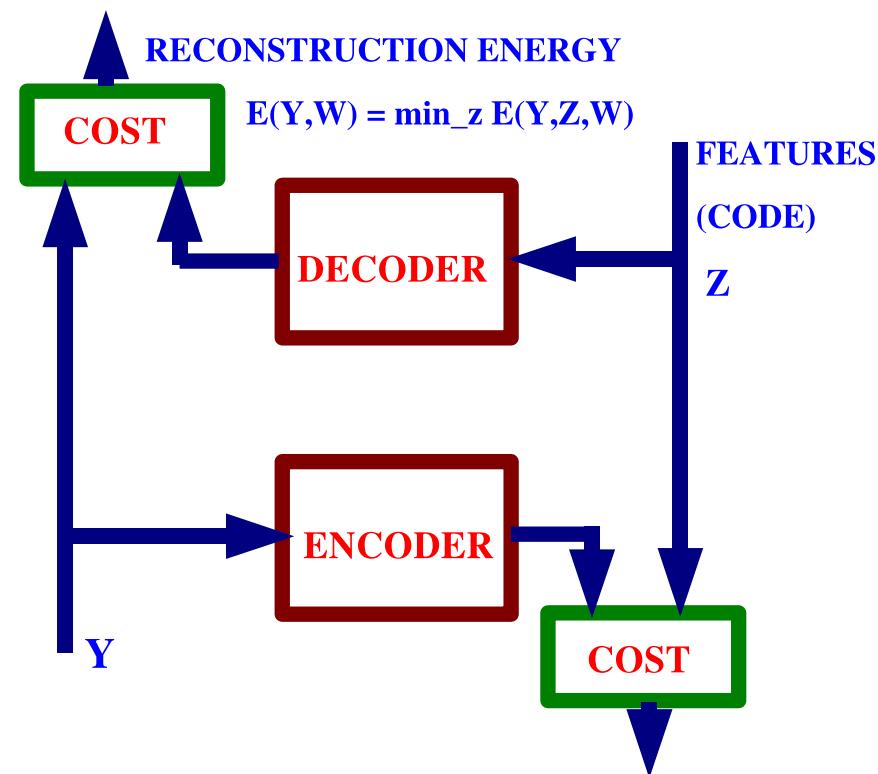


$$\bar{Z}_Y = \operatorname{argmin}_z E(Y, Z, W)$$

$$E(Y, W) = \min_z E(Y, Z, W)$$

Preventing “Flat” Energy Surfaces

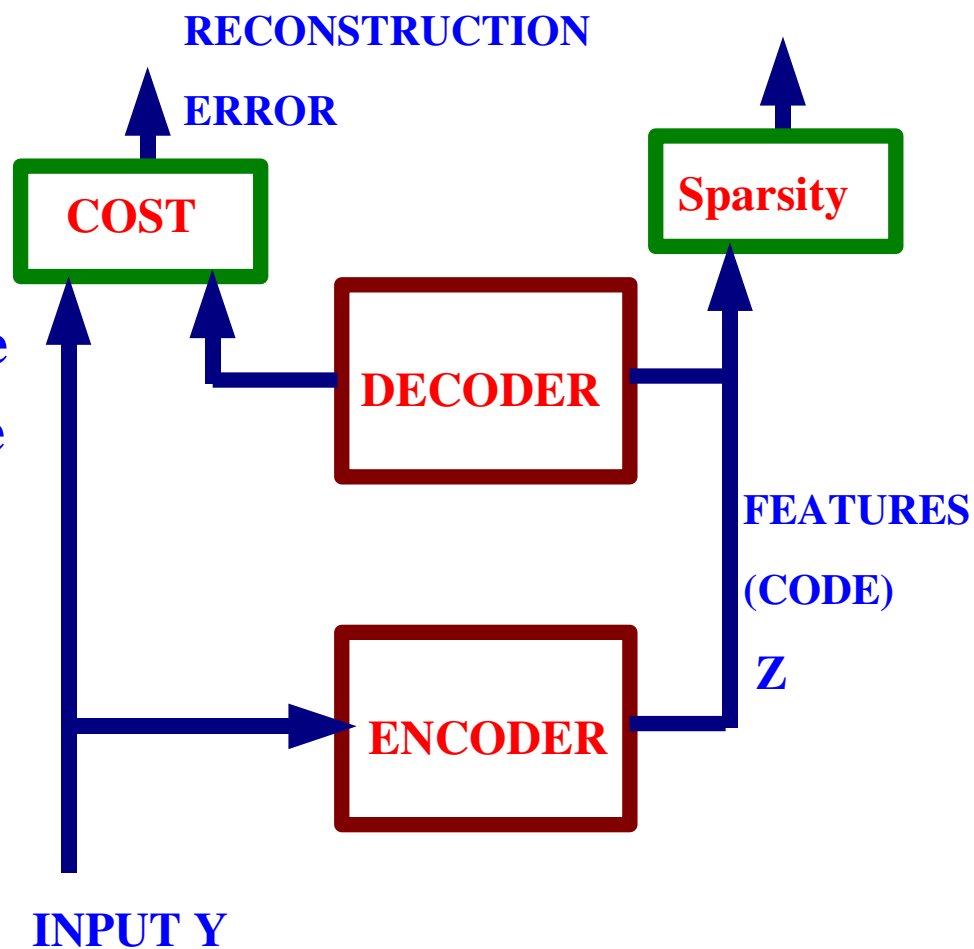
- If the dimension of Z is larger than the input, the system can reconstruct any input vector exactly in a trivial way
- **This is BAD**
- We want low energies for training samples, and high energies for everything else
- Contrastive divergence is a nice trick to pull up the energy of configurations around the training samples



$$\bar{Z}_Y = \operatorname{argmin}_Z E(Y, Z, W)$$
$$E(Y, W) = \min_Z E(Y, Z, W)$$

Learning Sparse Features

- If the feature vector is larger than the input, the system can learn the identity function in a trivial way
- To prevent this, we force the feature vector to be sparse
- By sparsifying the feature vector, we limit its information content, and we prevent system from being able to reconstruct everything perfectly
 - ▶ technically, code sparsification puts an upper bound on the partition function of $P(Y)$ under the model [Ranzato AI-Stats 07]



Why sparsity puts an upper bound on the partition function

Imagine the code has no restriction on it

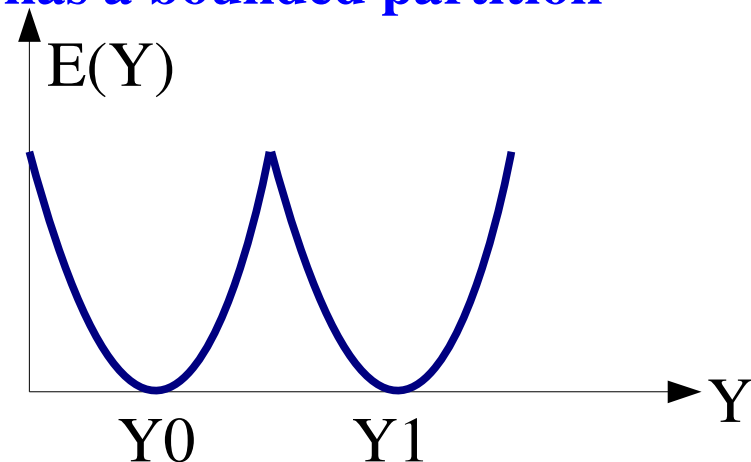
- ▶ The energy (or reconstruction error) can be zero everywhere, because every Y can be perfectly reconstructed. The energy is flat, and the partition function is unbounded

Now imagine that the code is binary ($Z=0$ or $Z=1$), and that the reconstruction cost is quadratic $E(Y) = \|Y - \text{Dec}(Z)\|^2$

- ▶ Only two input vectors can be perfectly reconstructed:
- ▶ $Y_0 = \text{Dec}(0)$ and $Y_1 = \text{Dec}(1)$.
- ▶ All other vectors have a higher reconstruction error

The corresponding probabilistic model has a bounded partition function:

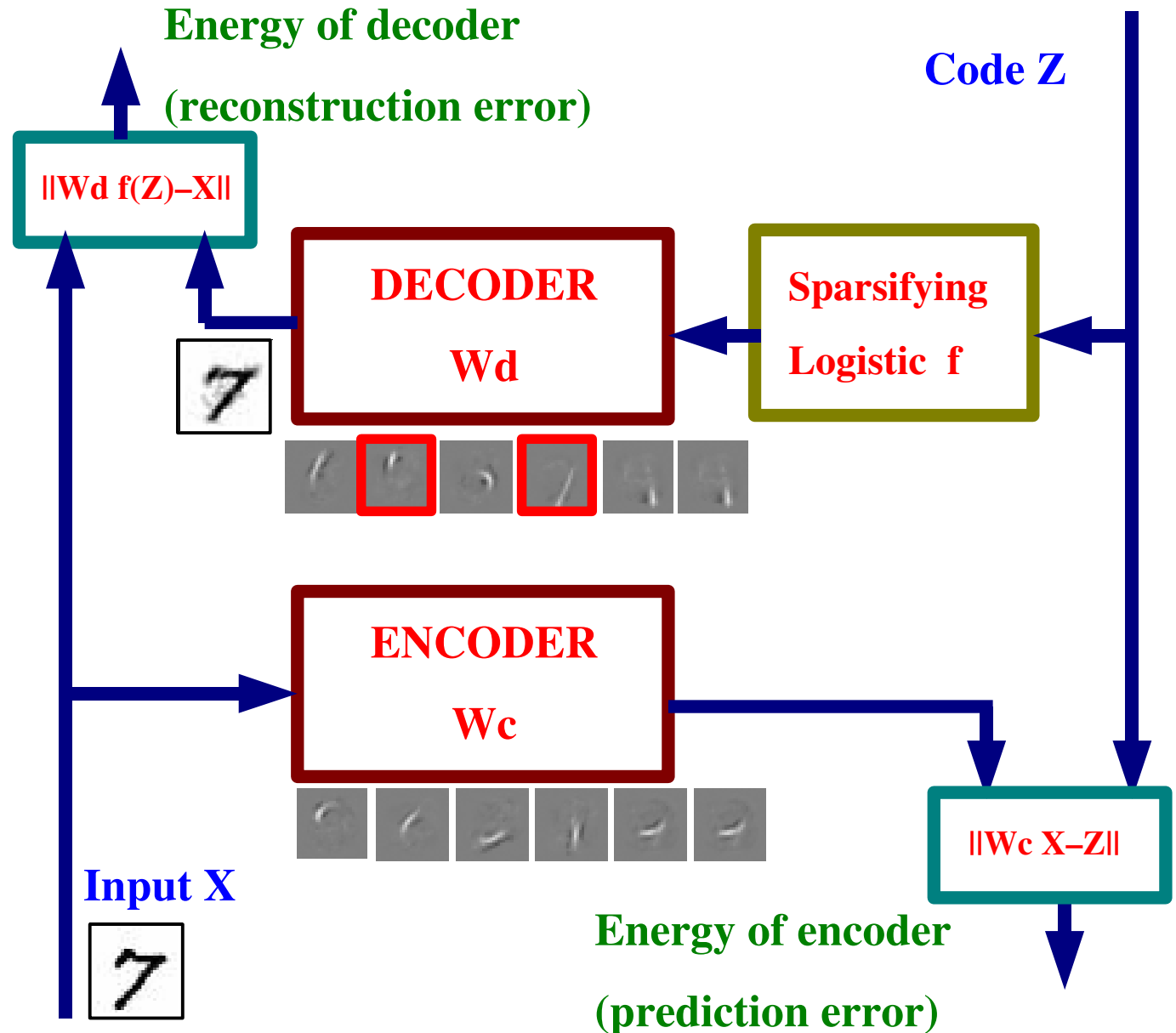
$$P(Y) = \frac{e^{-E(Y)}}{\int_y e^{-E(y)}}$$



Sparsifying with a high-threshold logistic function

Algorithm:


1. find the code Z that minimizes the reconstruction error AND is close to the encoder output
2. Update the weights of the decoder to decrease the reconstruction error
3. Update the weights of the encoder to decrease the prediction error



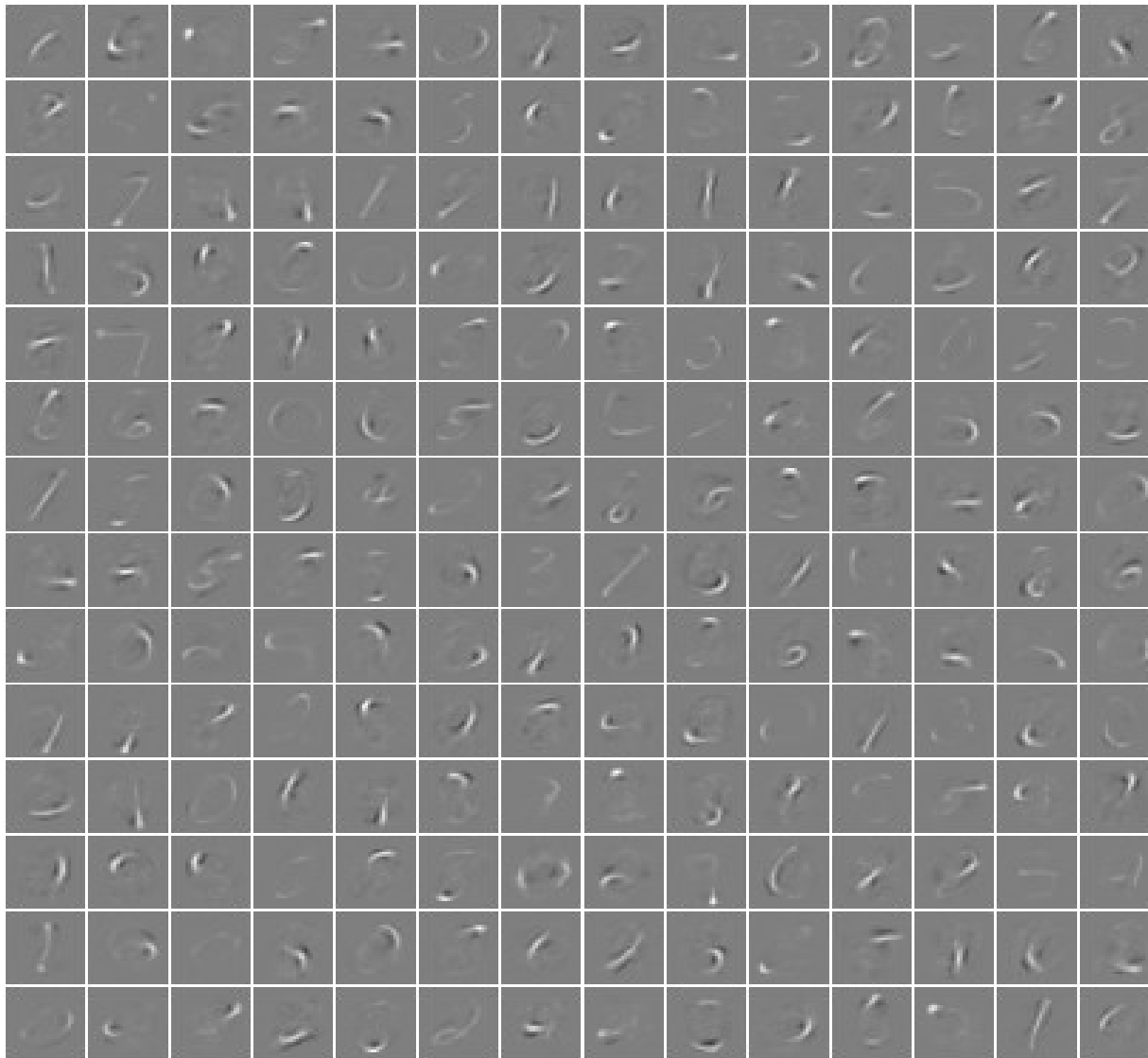
MNIST Dataset

3 6 8 1 7 9 6 6 4 1
6 7 5 7 8 6 3 4 8 5
2 1 7 9 7 1 2 8 4 5
4 8 1 9 0 1 8 8 9 4
7 6 1 8 6 4 1 5 6 0
7 5 9 2 6 5 8 1 9 7
2 2 2 2 2 3 4 4 8 0
0 2 3 8 0 7 3 8 5 7
0 1 4 6 4 6 0 2 4 3
7 1 2 8 7 6 9 8 6 1

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

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

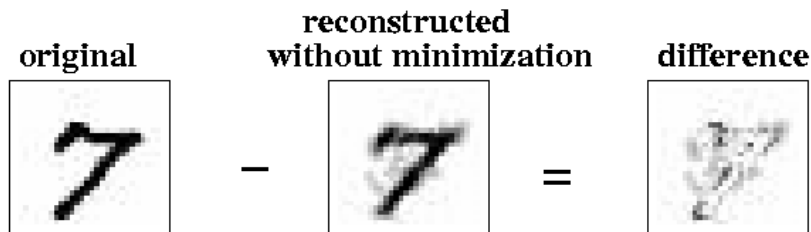
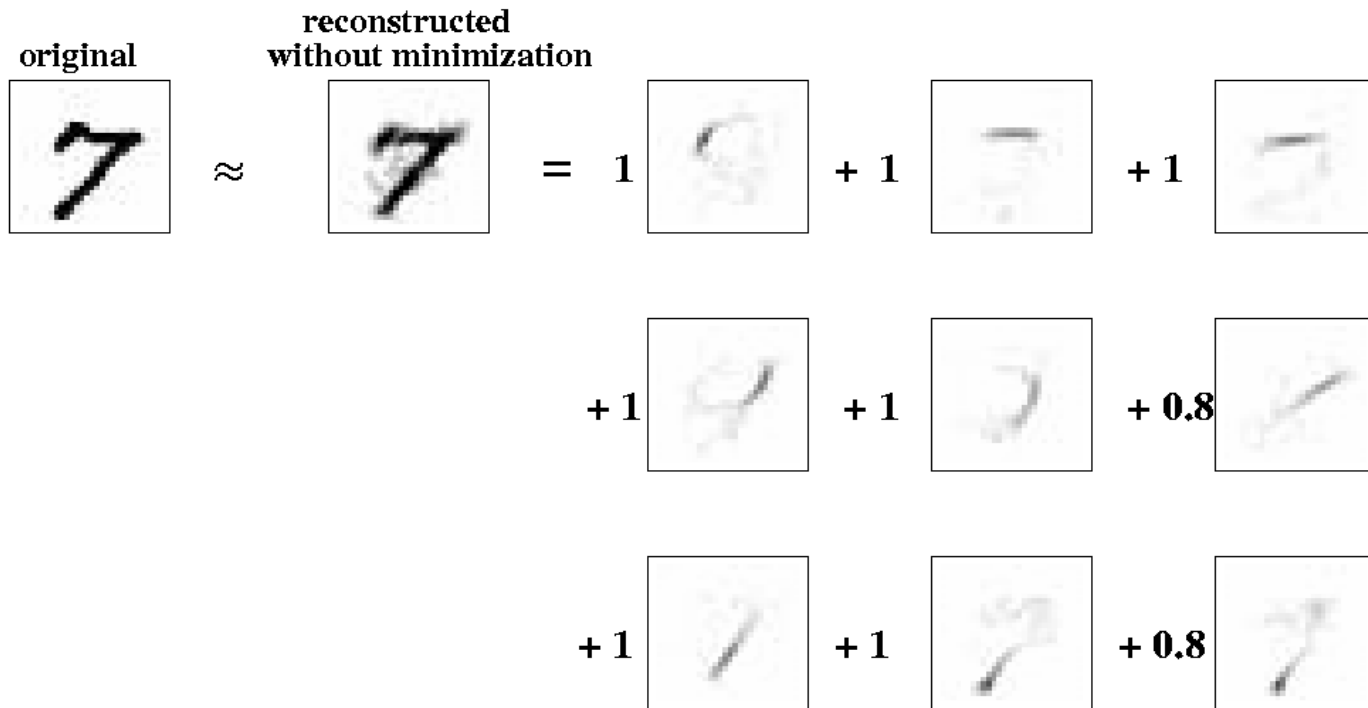
Training on handwritten digits



- ◆ 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



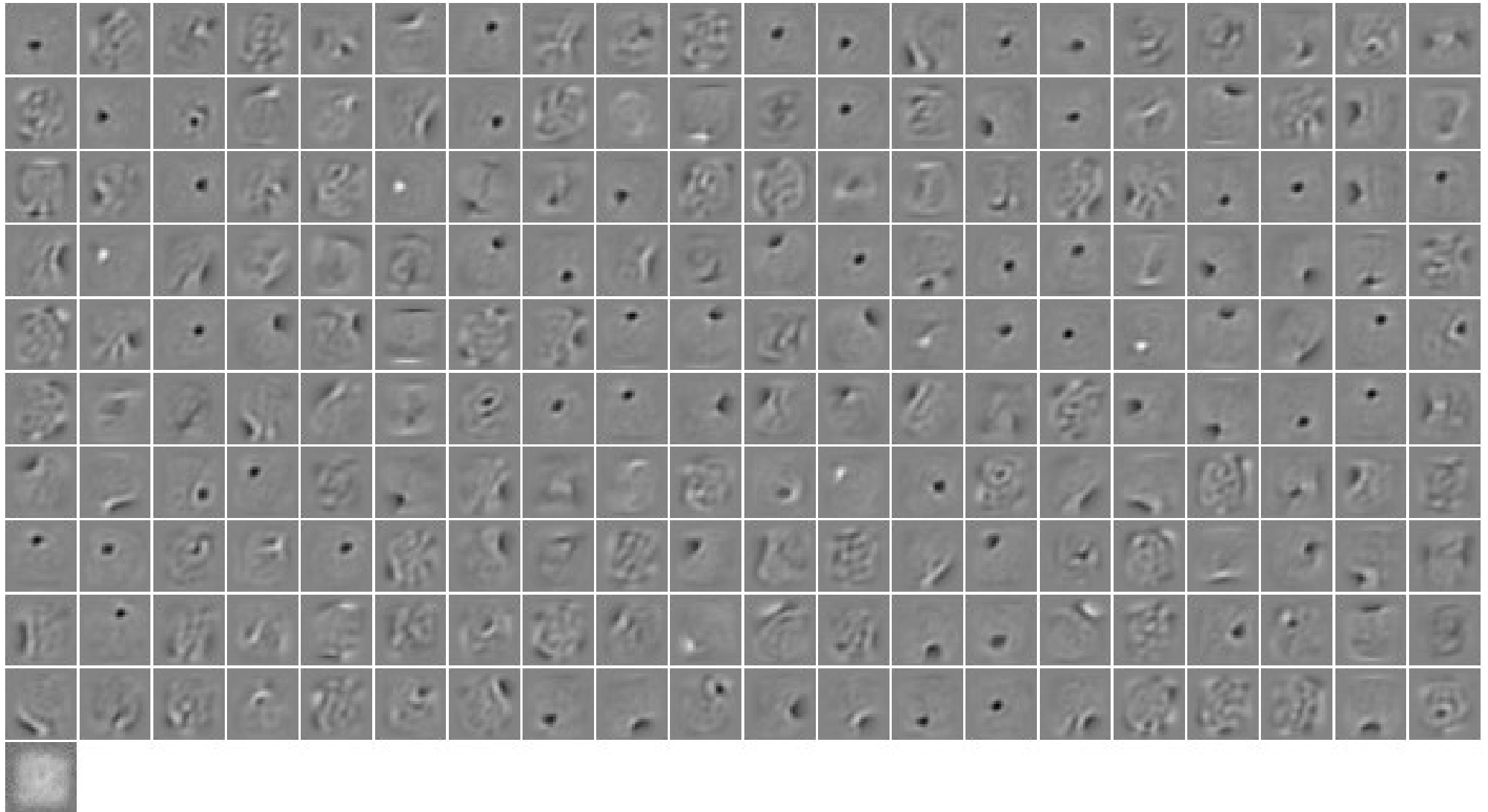
forward propagation through encoder and decoder



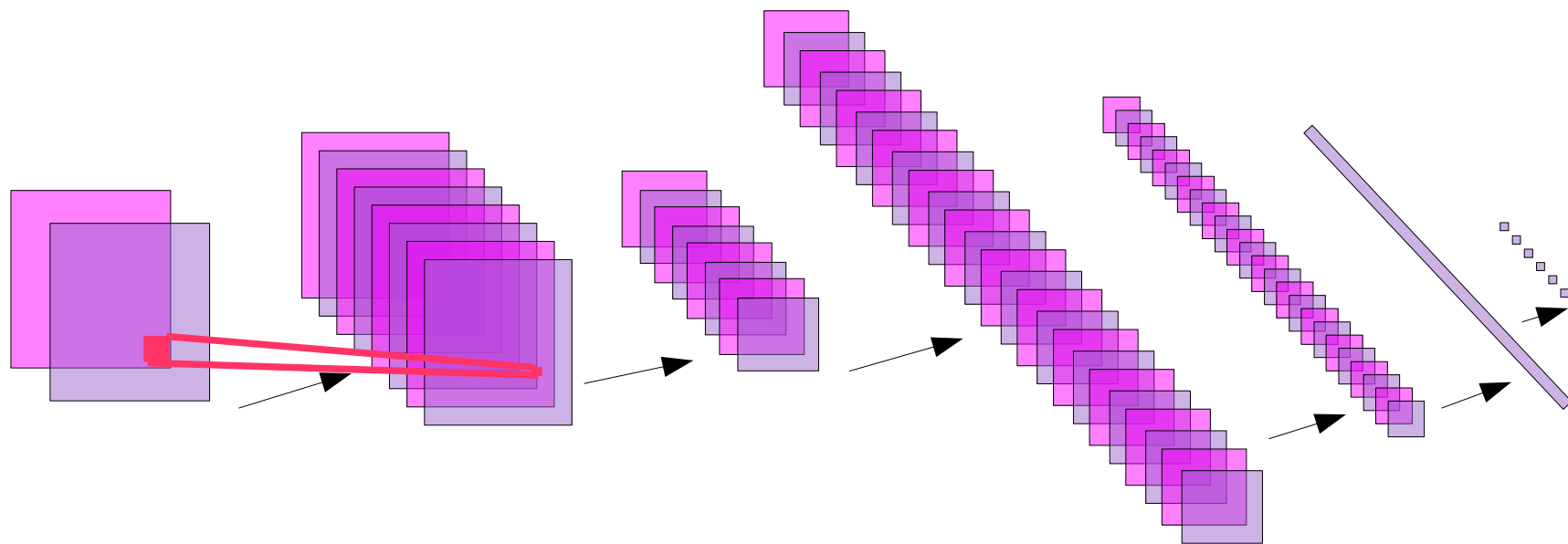
after training there is no need to minimize in code space

RBM: filters trained on MNIST

“bubble” detectors



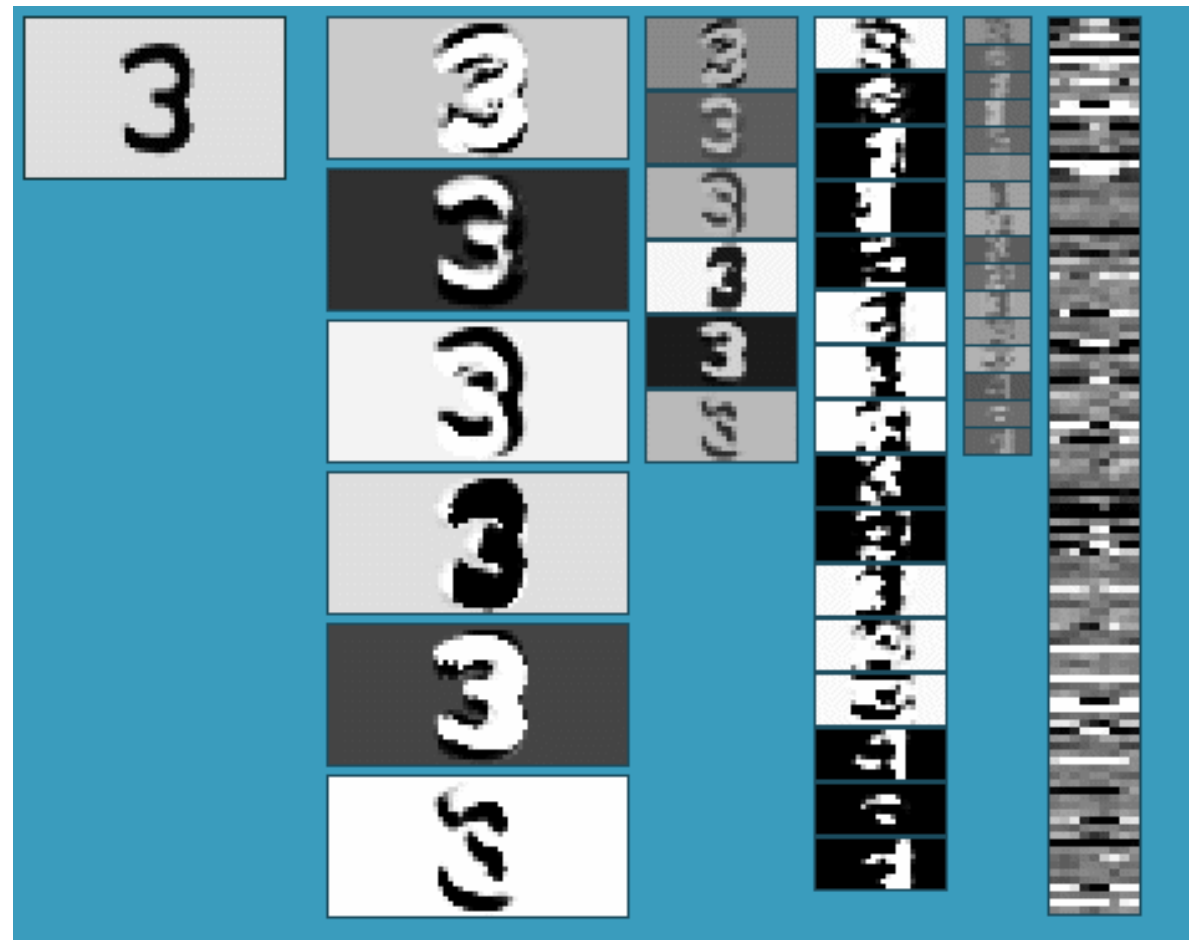
Training the filters of a Convolutional Network



The Multistage Hubel-Wiesel Architecture

Building a complete artificial vision system:

- ▶ Stack multiple stages of simple cells / complex cells layers
- ▶ Higher stages compute more global, more invariant features
- ▶ Stick a classification layer on top
- ▶ [Fukushima 1971-1982]
 - neocognitron
- ▶ [LeCun 1988-2007]
 - convolutional net
- ▶ [Poggio 2002-2006]
 - HMAX
- ▶ [Ullman 2002-2006]
 - fragment hierarchy
- ▶ [Lowe 2006]
 - HMAX



Unsupervised Training of Convolutional Filters

CLASSIFICATION EXPERIMENTS

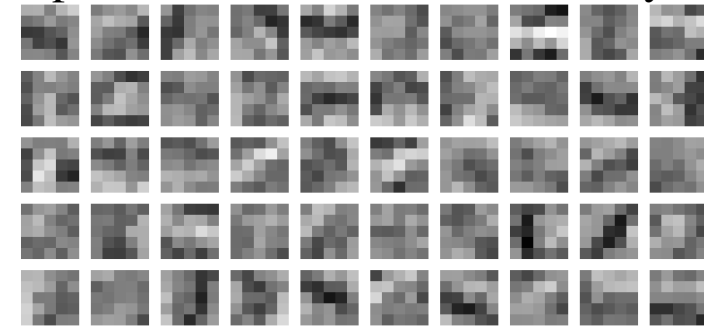
IDEA: improving supervised learning by pre-training with the unsupervised method (*)

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

- The **baseline**: *lenet6* initialized randomly

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

supervised filters in first conv. layer

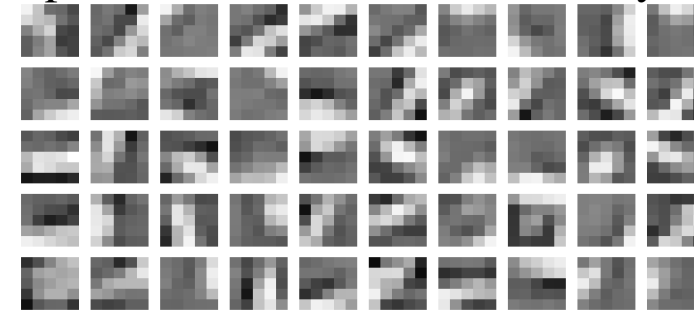


• *Experiment 1*

- 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%.

unsupervised filters in first conv. layer



• *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%.

(*)[Hinton, Osindero, Teh "A fast learning algorithm for deep belief nets" Neural Computatn 2006]

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, Neur Comp 2006
Convolutional nets			
Convolutional net LeNet-5,		0.80	Ranzato et al. NIPS 2006
Convolutional net LeNet-6,		0.70	Ranzato et al. NIPS 2006
Conv. net LeNet-6- + unsup learning		0.60	Ranzato et al. NIPS 2006
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	0.39	Ranzato et al. NIPS 2006

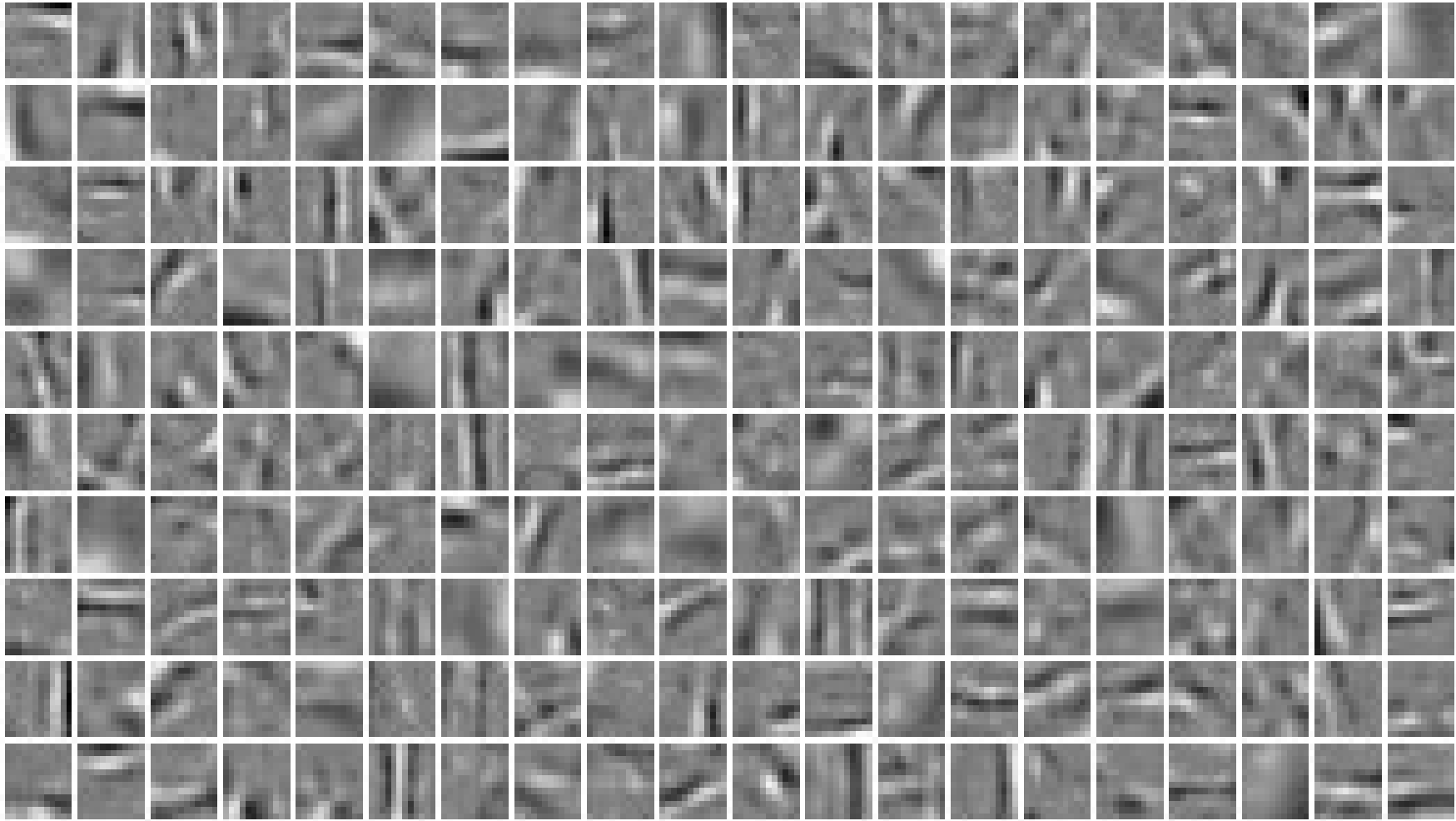
Training on natural image patches



Berkeley data set

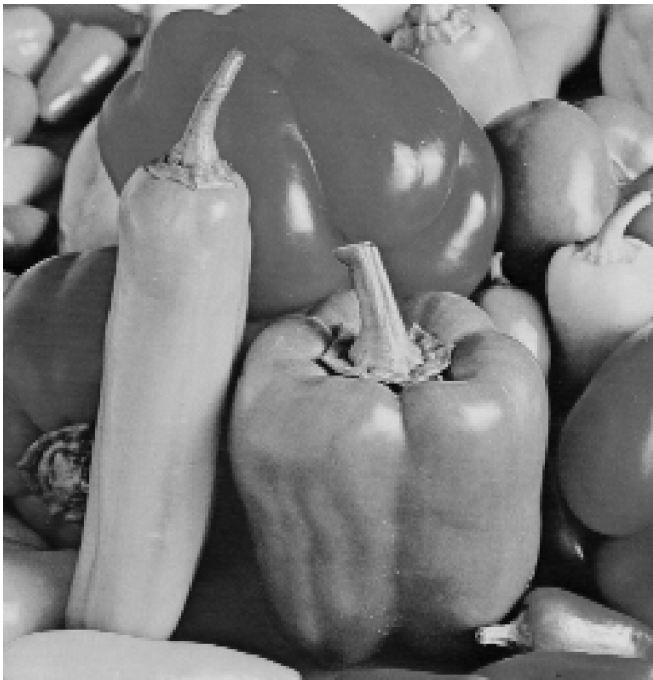
- ◆ 100,000 12x12 patches
- ◆ 200 units in the code
- ◆ η 0.02
- ◆ β 1
- ◆ learning rate 0.001
- ◆ L1 regularizer 0.001
- ◆ fast convergence: < 30min.

Natural image patches: Filters

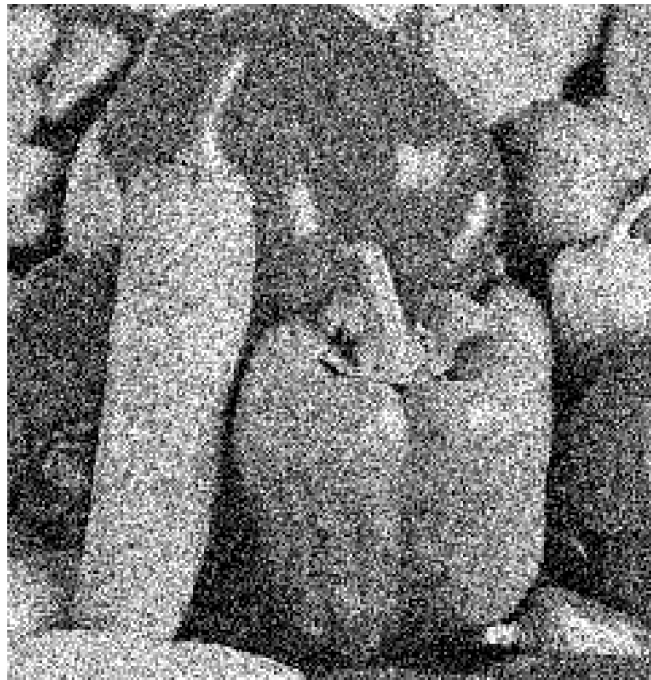


200 decoder filters (reshaped columns of matrix \mathbf{W}_d)

Denoising



original image

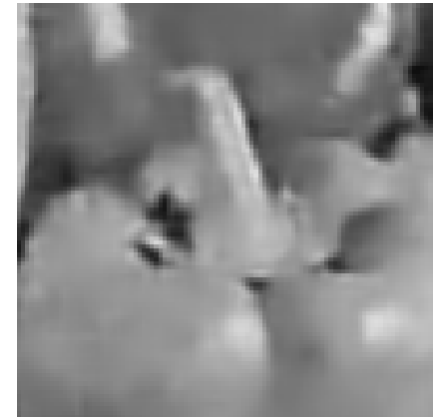


noisy image: PSNR 14.15dB
(std. dev. noise 50)



denoised image
PSNR 26.50dB

ZOOM ->



Denoising

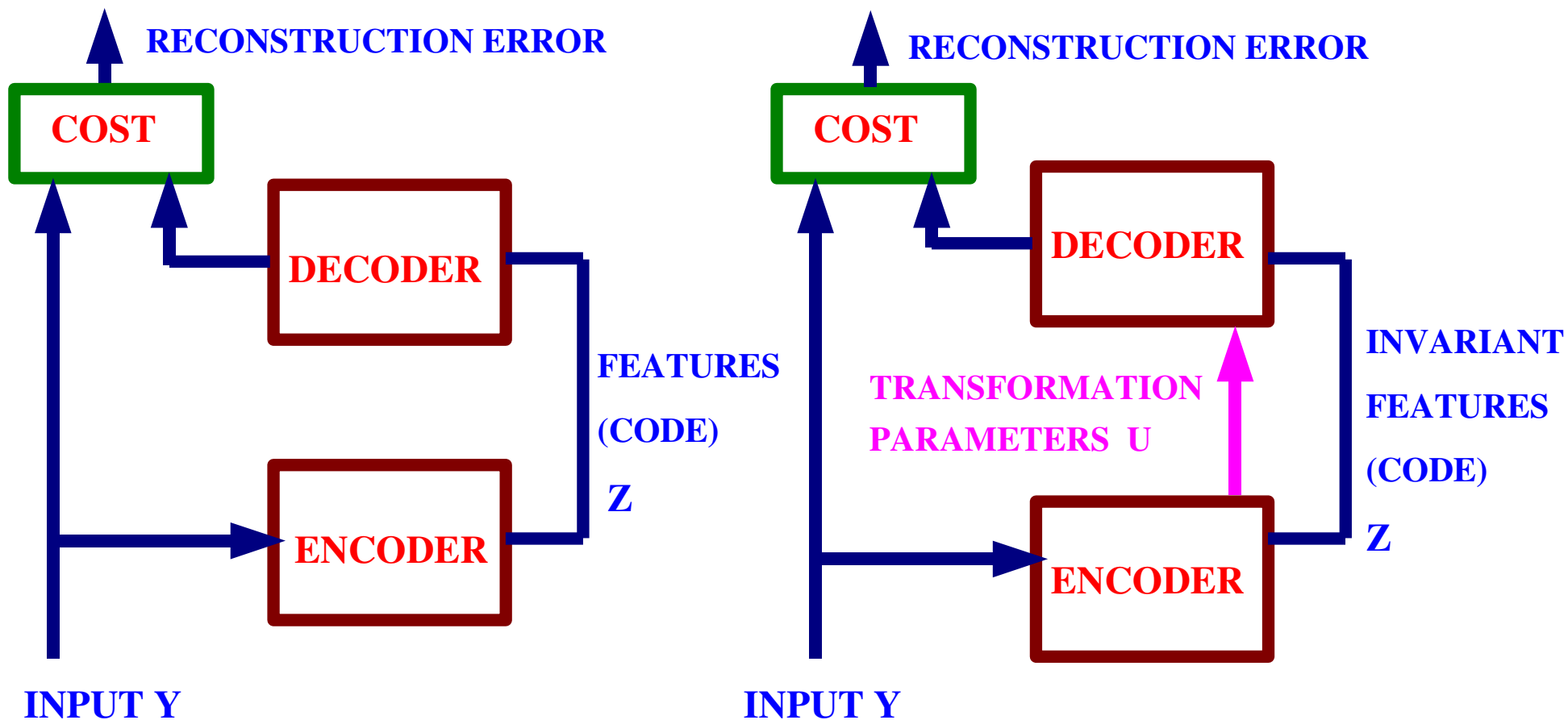
<i>s.d. / PSNR</i>	<i>Lena</i>				<i>Barbara</i>				<i>Boat</i>				<i>House</i>				<i>Peppers</i>			
50 / 14.15	27.86	28.61	27.79	26.49	23.46	25.48	25.47	23.15	26.02	26.38	25.95	24.53	27.85	28.26	27.95	26.74	26.35	25.90	26.13	24.52
75 / 10.63	25.97	26.84	25.80	24.13	22.46	23.65	23.01	21.36	24.31	24.79	23.98	22.48	25.77	26.41	25.22	24.13	24.56	24.00	23.69	21.68
100 / 8.13	24.49	25.64	24.46	21.87	21.77	22.61	21.89	19.77	23.09	23.75	22.81	20.80	24.20	25.11	23.71	21.66	23.04	22.66	21.75	19.60

Comparison between:

- our method [first column]
- Portilla et al. IEEE Trans. Image Processing (2003) [second column]
- Elad and Aharon CVPR 2006 [third column]
- Roth and Black CVPR 2005 [fourth column]

Learning Invariant Features

Separating the “what” from the “where”

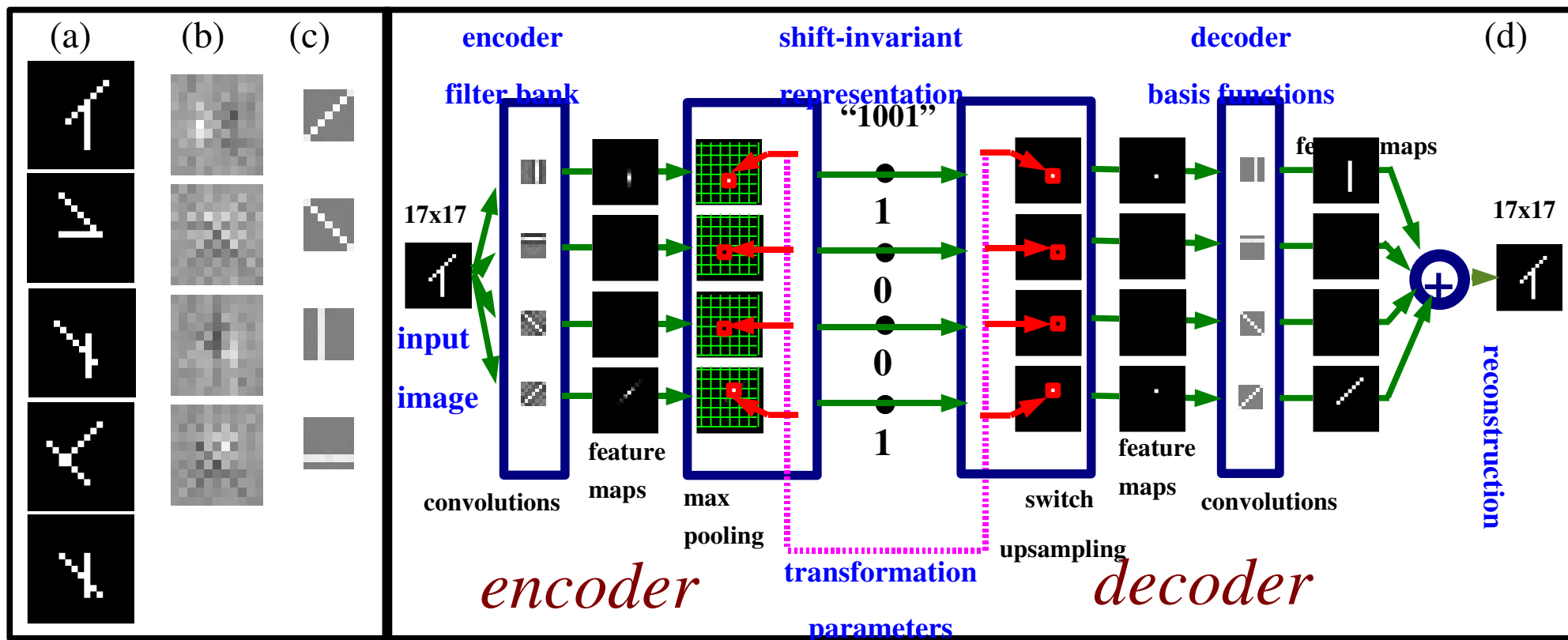


Standard Feature Extractor

Invariant Feature Extractor

Learning Invariant Feature Hierarchies

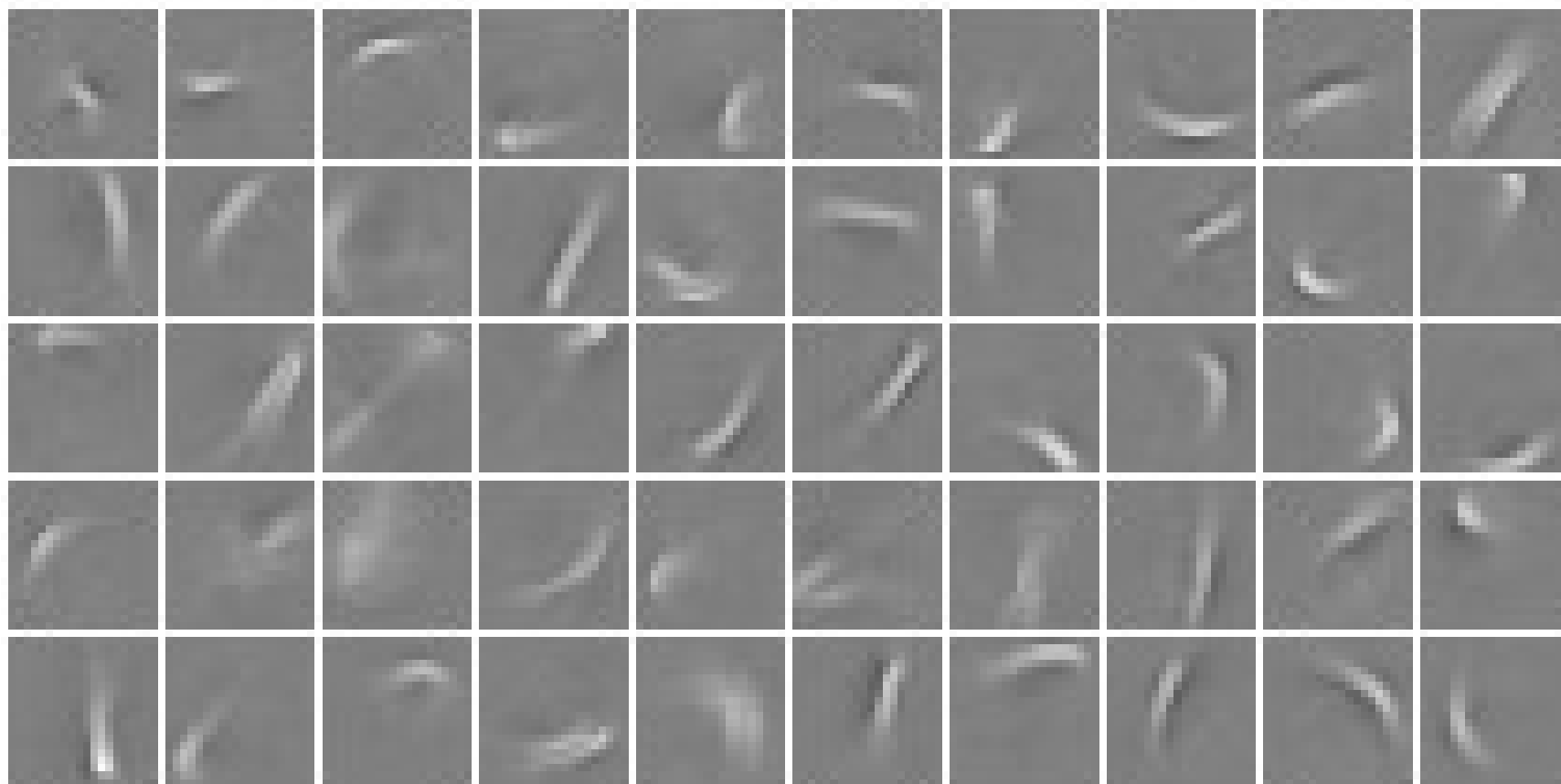
Learning Shift Invariant Features



Shift Invariant Global Features on MNIST

Learning 50 Shift Invariant Global Features on MNIST:

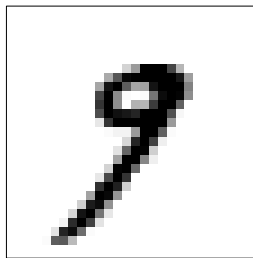
- ▶ 50 filters of size 20x20 movable in a 28x28 frame (81 positions)
- ▶ movable strokes!



Example of Reconstruction

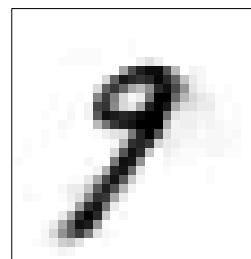
- Any character can be reconstructed as a linear combination of a small number of basis functions.

ORIGINAL
DIGIT



\approx

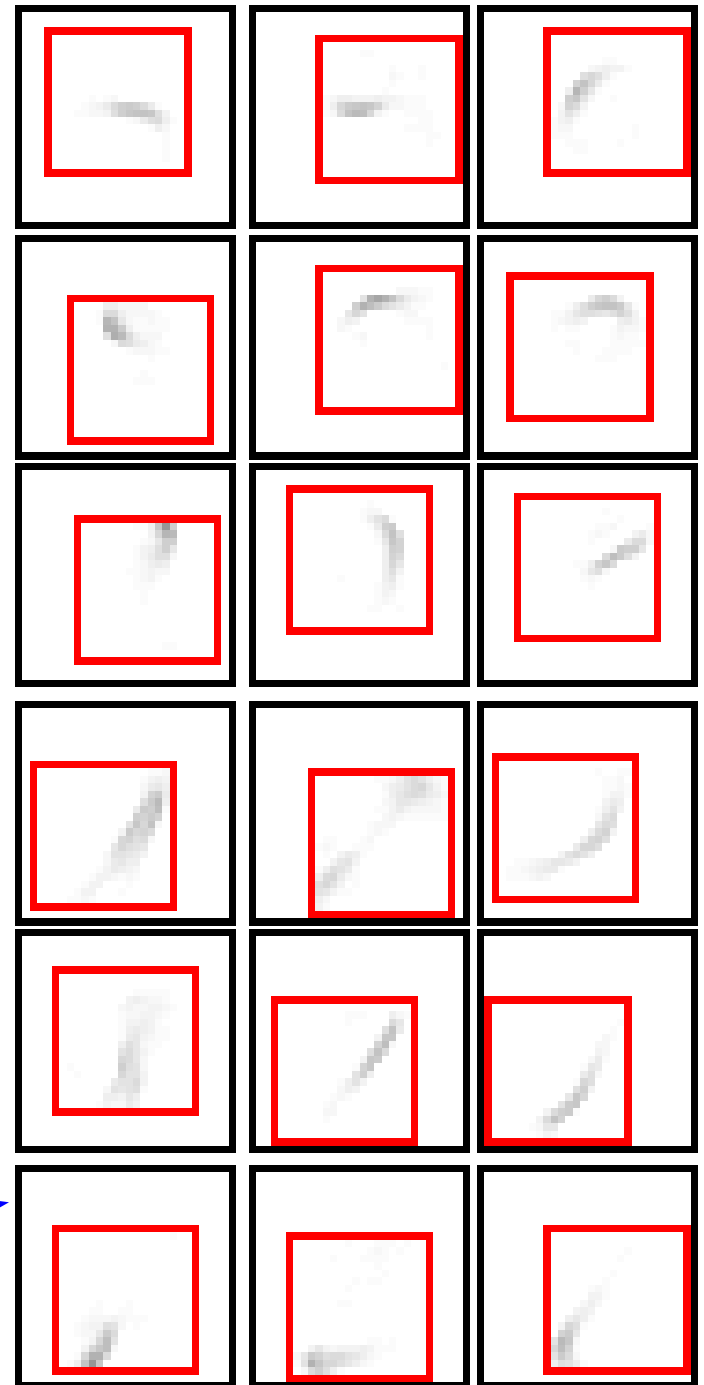
RECONSTRUCTION



$=$

Σ

ACTIVATED DECODER
BASIS FUNCTIONS
(in feed-back layer)

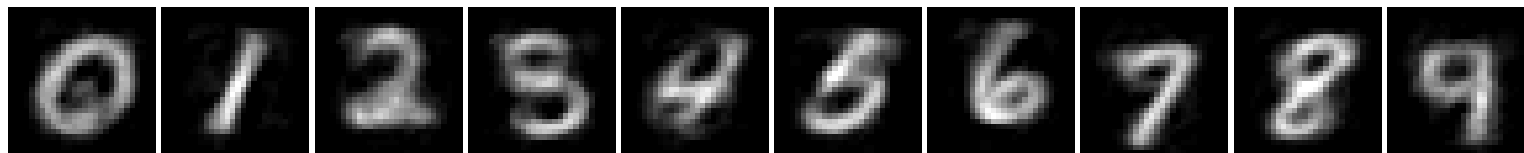
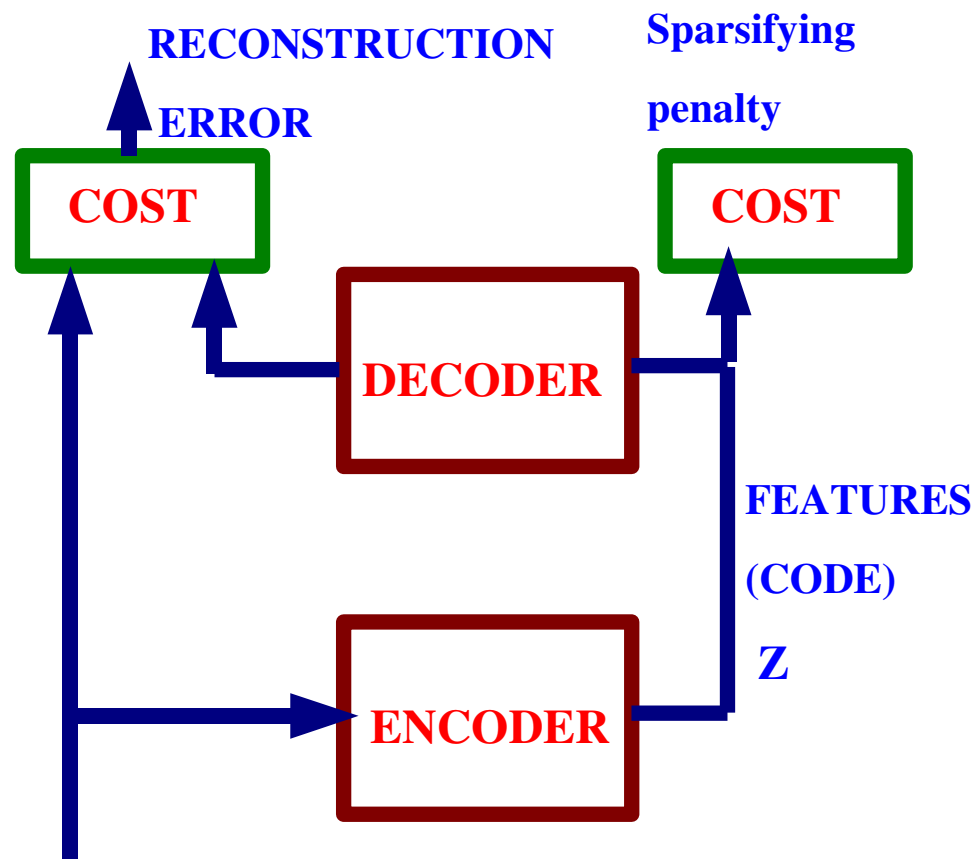


red squares: decoder bases

Sparse Encoding Symmetric Machine (SESM)

[Ranzato et al. NIPS 2007]

- Encoder and Decoder are symmetric (they have the same parameters)
- There is a log Student-T penalty on components of the code



Learning Invariant Filters in a Convolutional Net

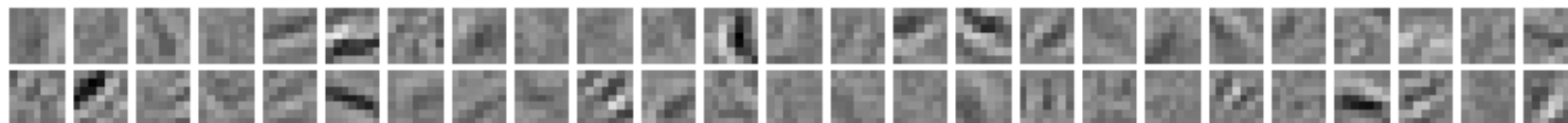


Figure 1: 50 7×7 filters in the first convolutional layer that were learned by the network trained supervised from *random* initial conditions with 600K digits.



Figure 2: 50 7×7 filters that were learned by the unsupervised method (on 60K digits), and that are used to initialize the first convolutional layer of the network.

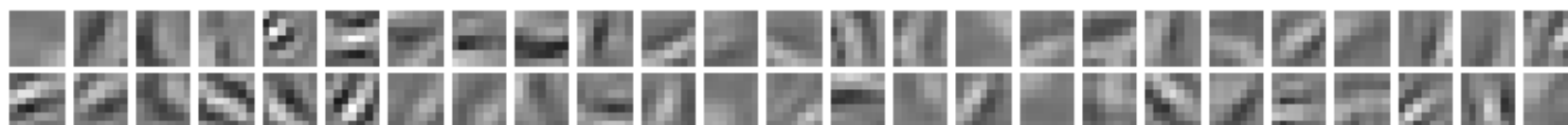
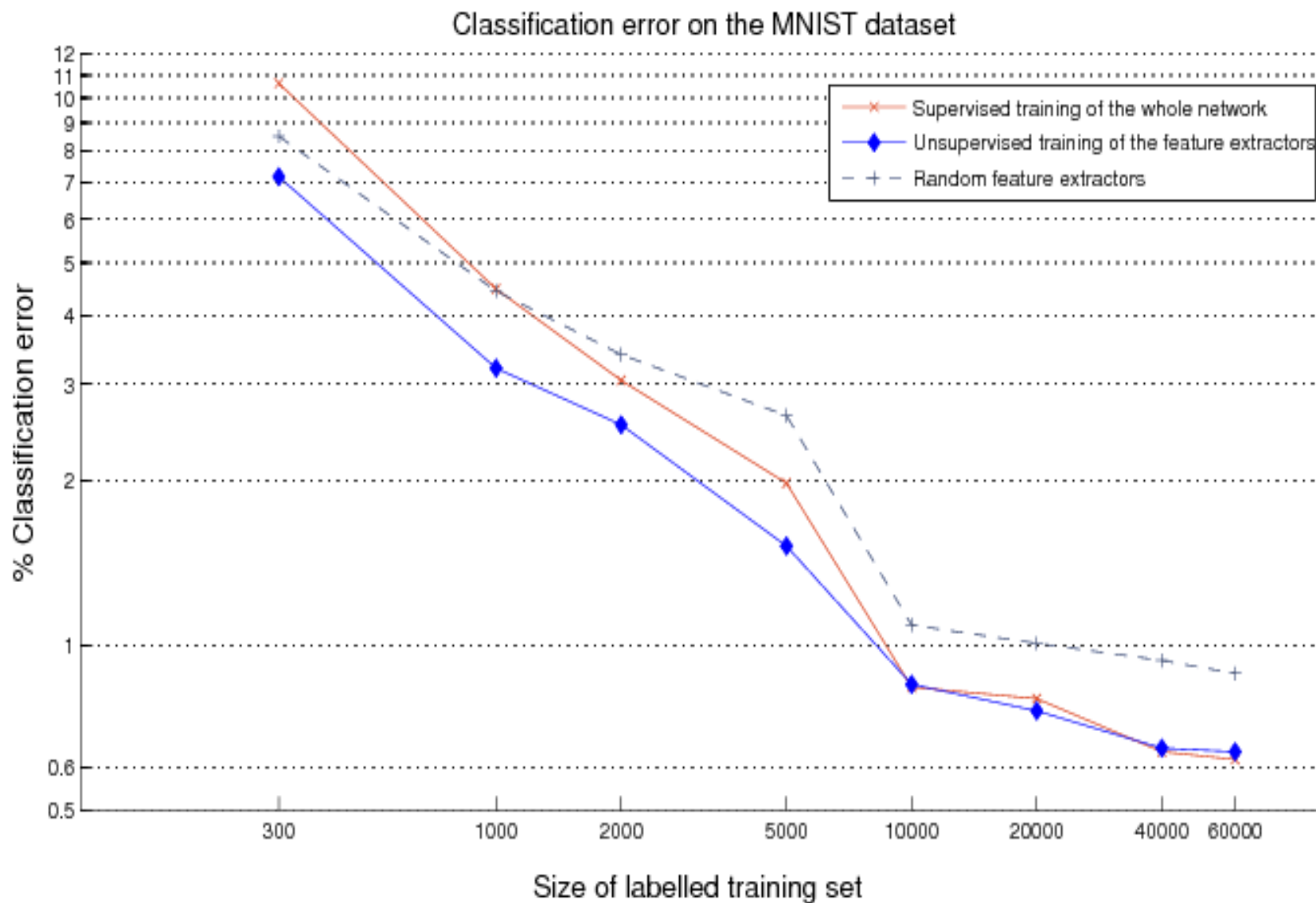


Figure 3: 50 7×7 filters in the first convolutional layer that were learned by the network trained supervised from the initial conditions given by the *unsupervised method* (see fig.2) with 600K digits.

Influence of Number of Training Samples



Generic Object Recognition: 101 categories + background

Caltech-101 dataset: 101 categories

▶ accordion airplanes anchor ant barrel bass beaver binocular bonsai brain
brontosaurus buddha butterfly camera cannon car_side ceiling_fan cellphone
chair chandelier cougar_body cougar_face crab crayfish crocodile crocodile_head
cup dalmatian dollar_bill dolphin dragonfly electric_guitar elephant emu
euphonium ewer Faces Faces_easy ferry flamingo flamingo_head garfield
gerenuk gramophone grand_piano hawksbill headphone hedgehog helicopter ibis
inline_skate joshua_tree kangaroo ketch lamp laptop Leopards llama lobster
lotus mandolin mayfly menorah metronome minaret Motorbikes nautilus octopus
okapi pagoda panda pigeon pizza platypus pyramid revolver rhino rooster
saxophone schooner scissors scorpion sea_horse snoopy soccer_ball stapler
starfish stegosaurus stop_sign strawberry sunflower tick trilobite umbrella watch
water_lilly wheelchair wild_cat windsor_chair wrench yin_yang

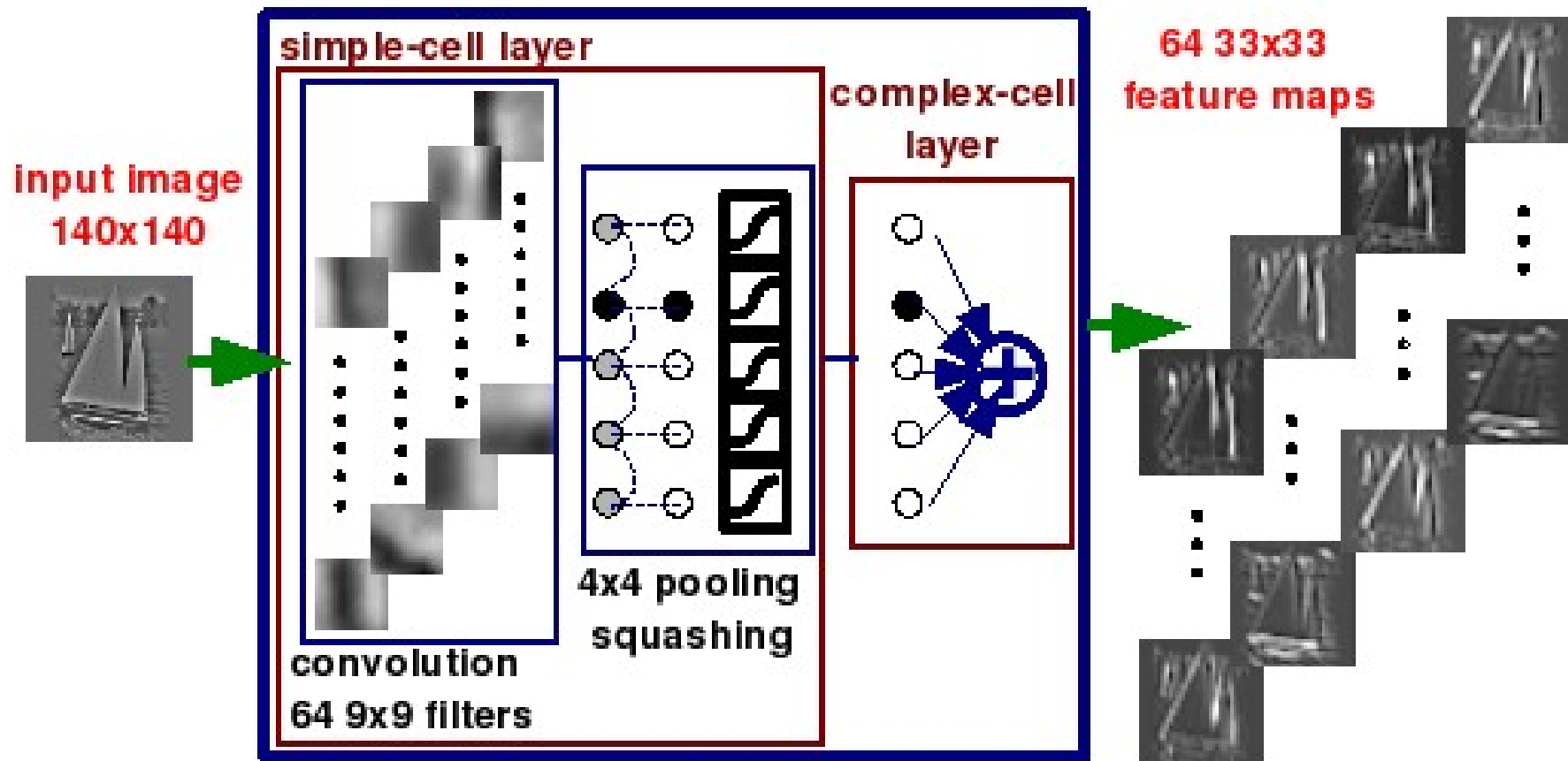
● **Only 30 training examples per category!**

● **A convolutional net trained with backprop (supervised) gets 20% correct recognition.**

● **Training the filters with the sparse invariant unsupervised method**

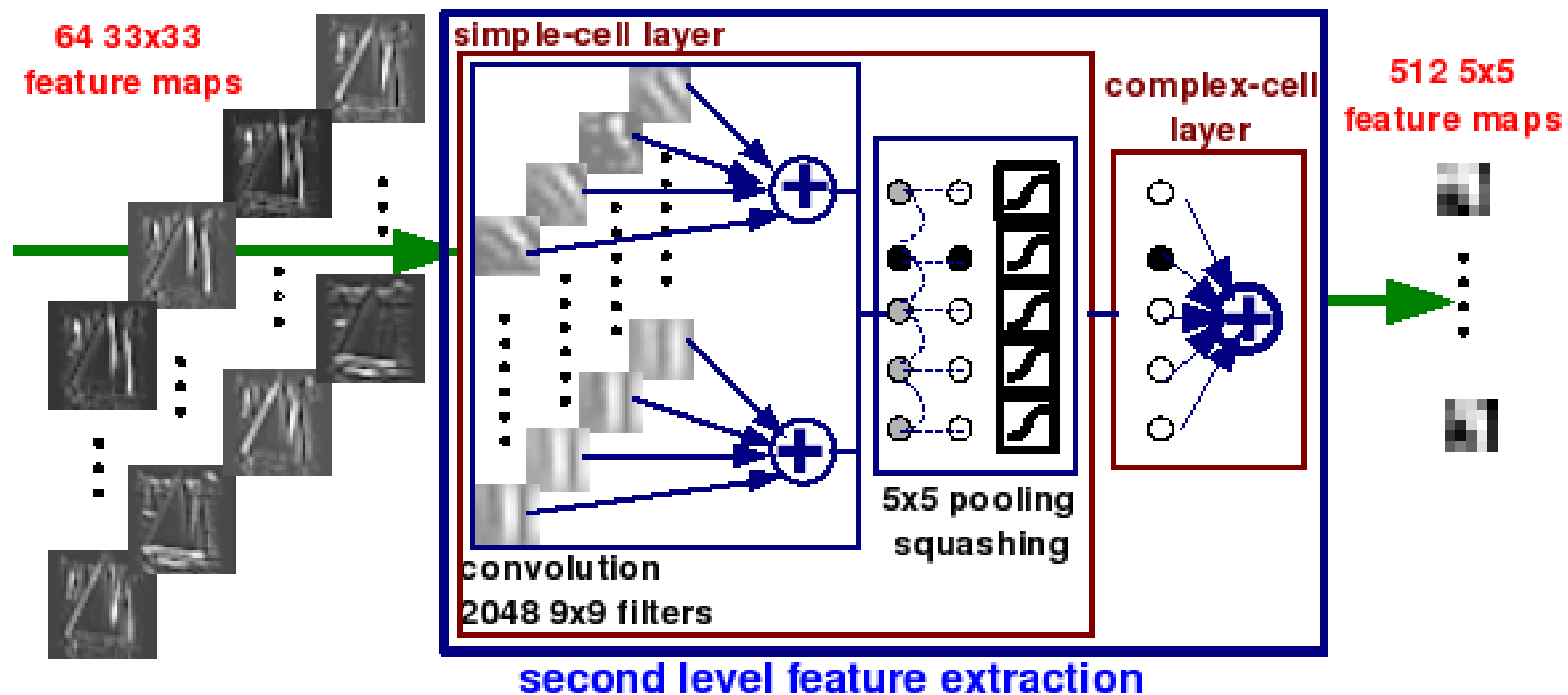
Training the 1st stage filters

- 12x12 input windows (complex cell receptive fields)
- 9x9 filters (simple cell receptive fields)
- 4x4 pooling



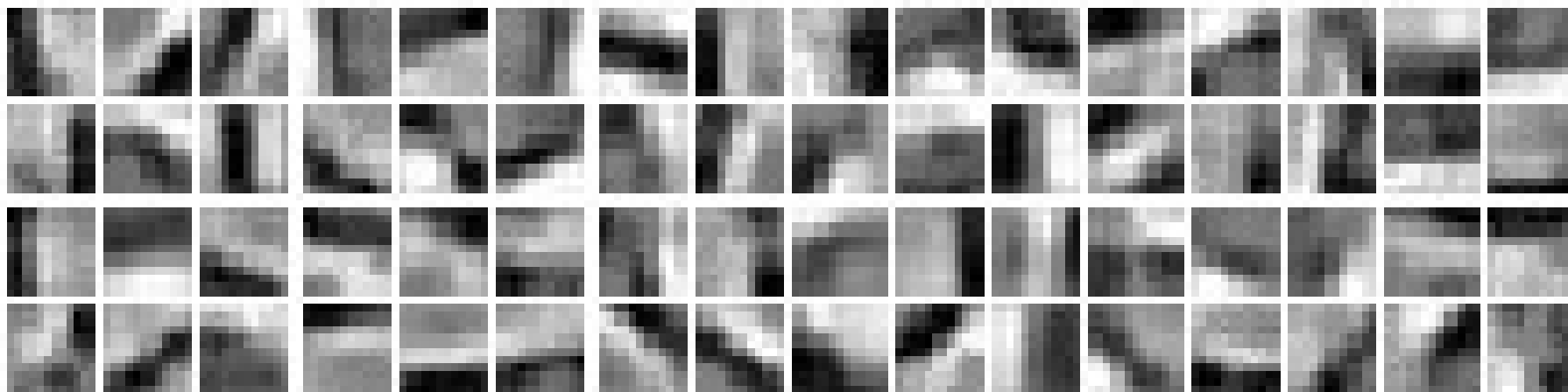
Training the 2nd stage filters

- 13x13 input windows (complex cell receptive fields on 1st features)
- 9x9 filters (simple cell receptive fields)
- Each output feature map combines 4 input feature maps
- 5x5 pooling

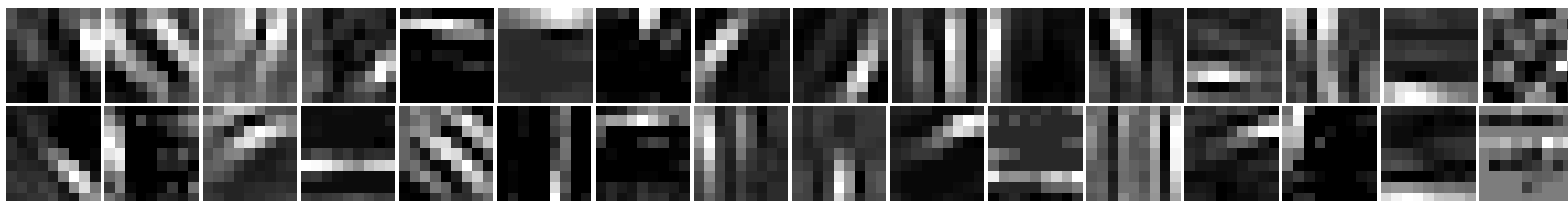


Generic Object Recognition: 101 categories + background

9x9 filters at the first level

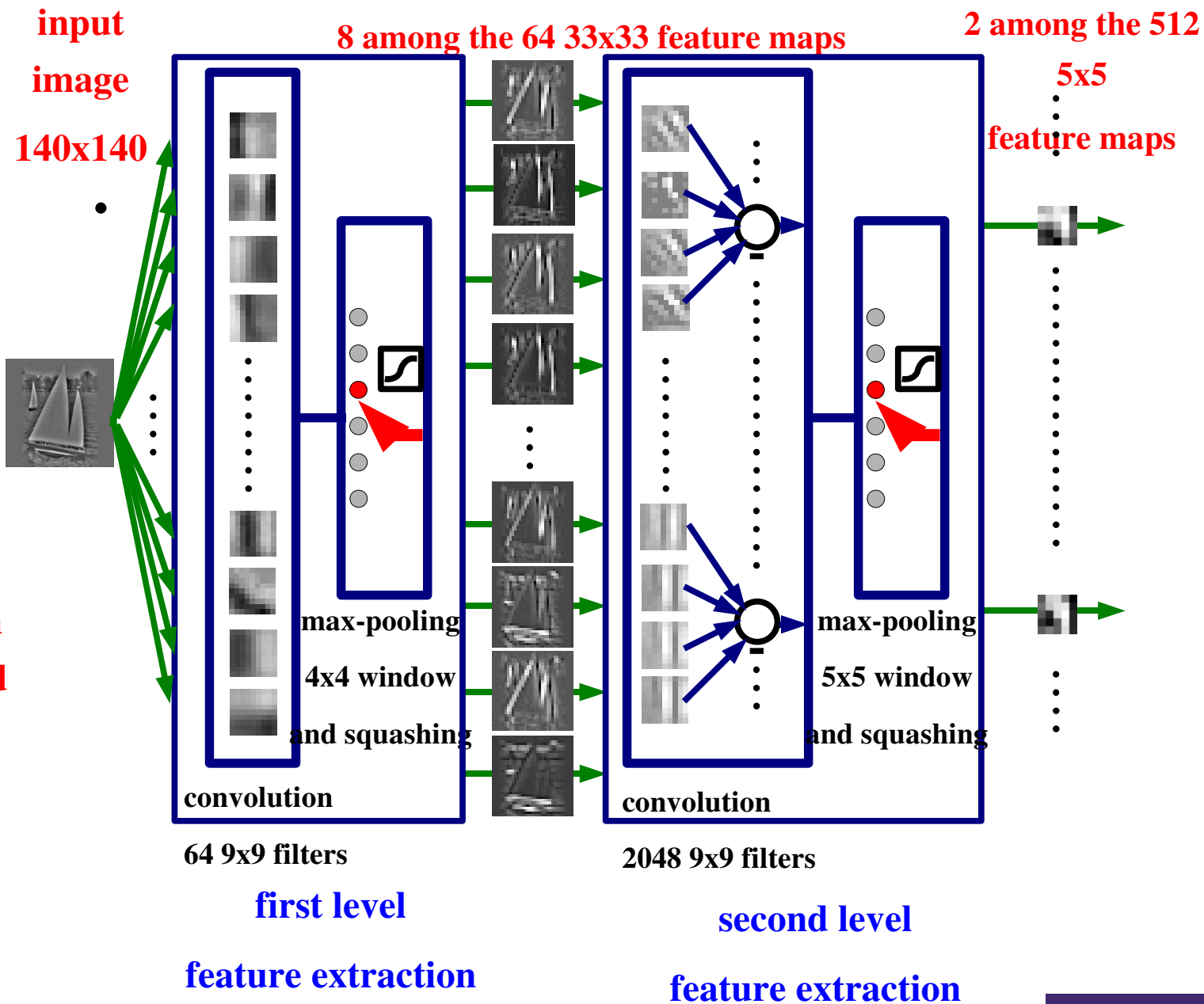


9x9 filters at the second level

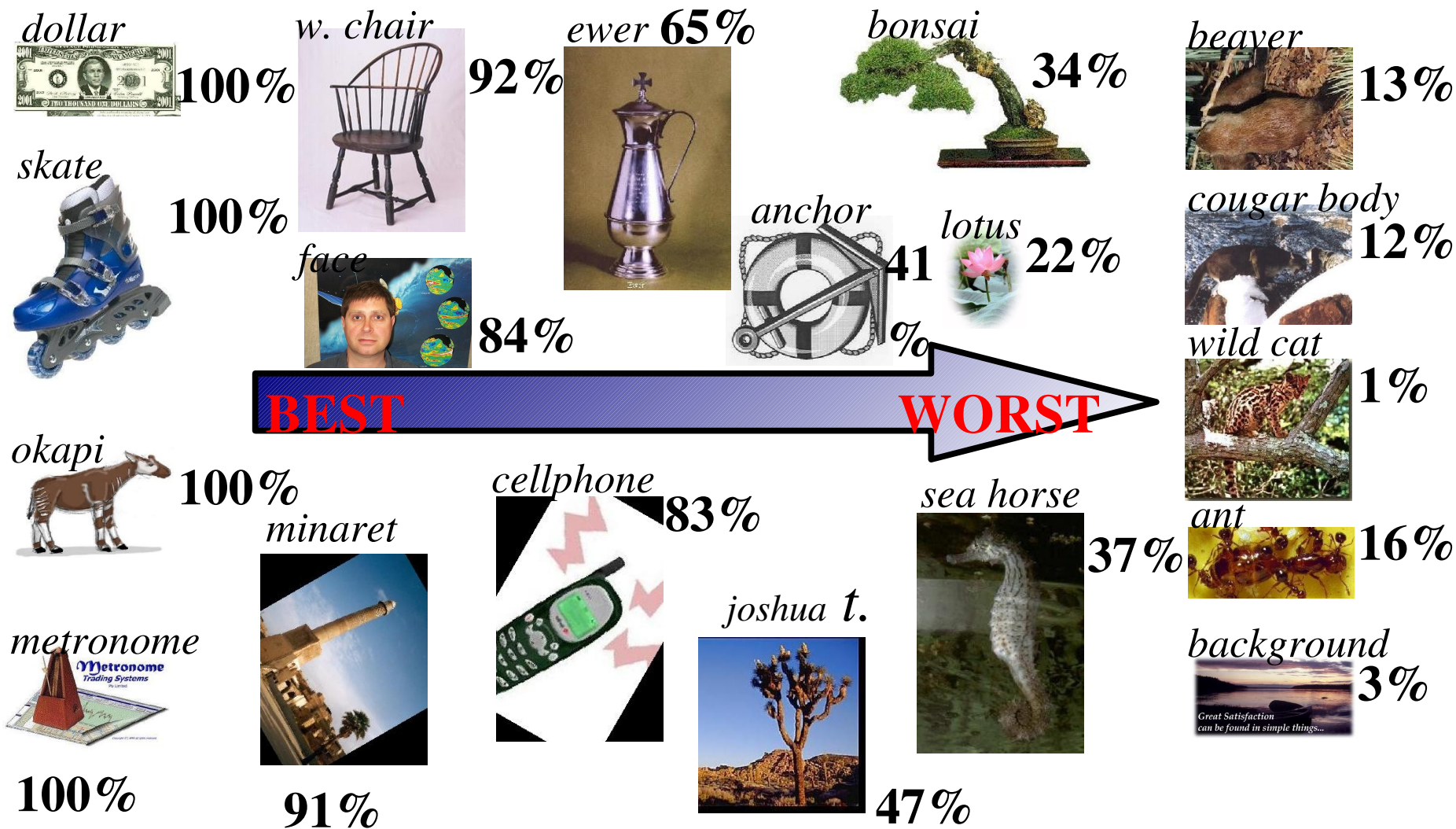


Shift-Invariant Feature Hierarchies on Caltech-101

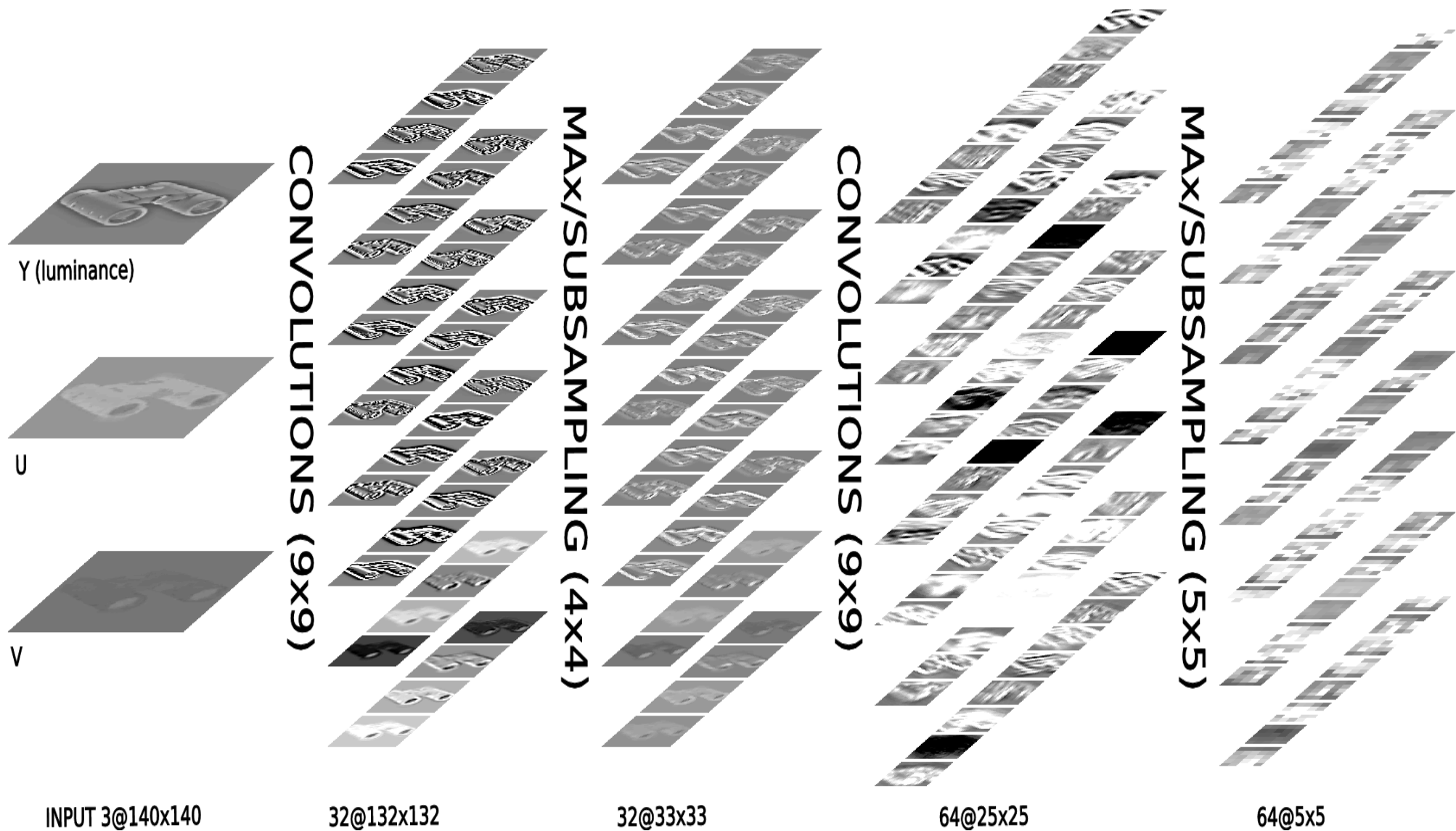
- 2 layers of filters trained unsupervised
- supervised classifier on top.
- 54% correct on Caltech-101 with 30 examples per class
- 20% correct with purely supervised backprop



Recognition Rate on Caltech 101



Smaller Network



The End