

What's Wrong With Deep Learning?

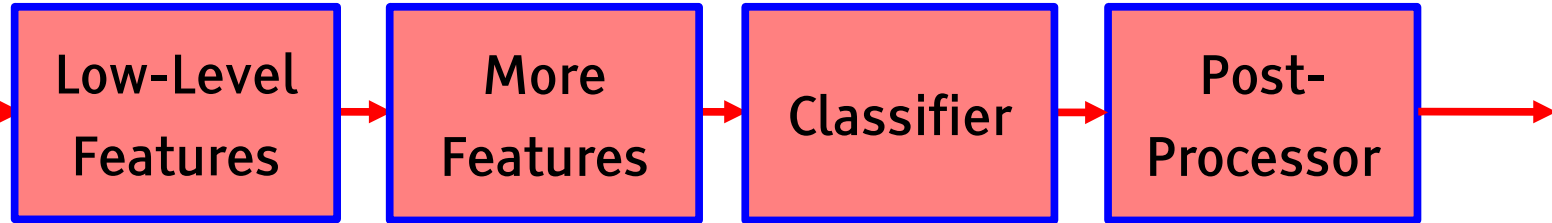
Yann LeCun

Facebook AI Research &
Center for Data Science, NYU

yann@cs.nyu.edu

<http://yann.lecun.com>





■ The motivation for ConvNets and Deep Learning: **end-to-end learning**

- ▶ Integrating feature extractor, classifier, contextual post-processor

■ A bit of archeology: ideas that have been around for a while

- ▶ Kernels with stride, non-shared local connections, metric learning...
- ▶ “fully convolutional” training

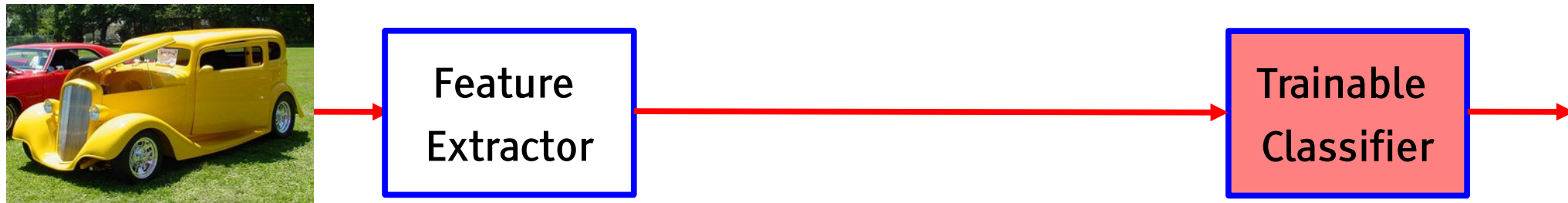
■ What's missing from deep learning?

- ▶ 1. Theory
- ▶ 2. Reasoning, structured prediction
- ▶ 3. Memory, short-term/working/episodic memory
- ▶ 4. Unsupervised learning that actually works

Deep Learning = Learning Hierarchical Representations

Y LeCun

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor



Mainstream Modern Pattern Recognition: Unsupervised mid-level features



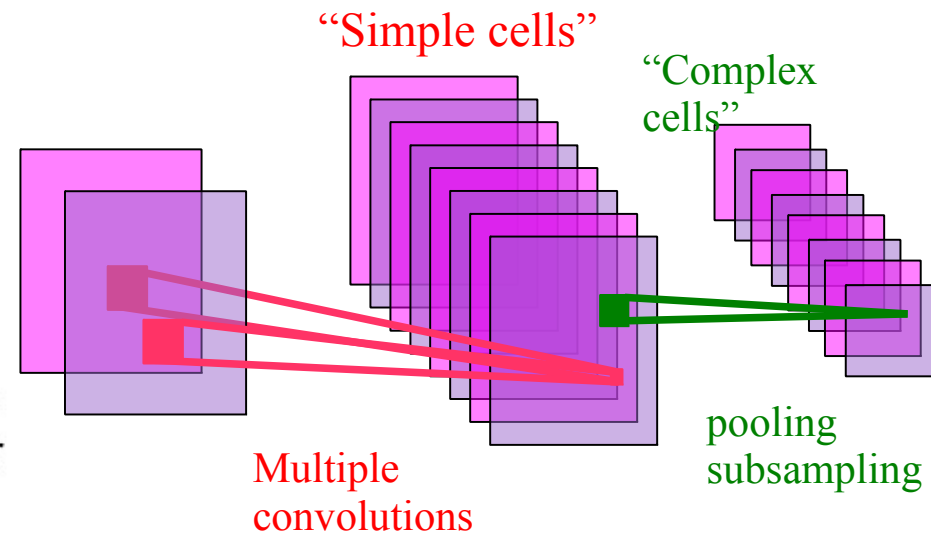
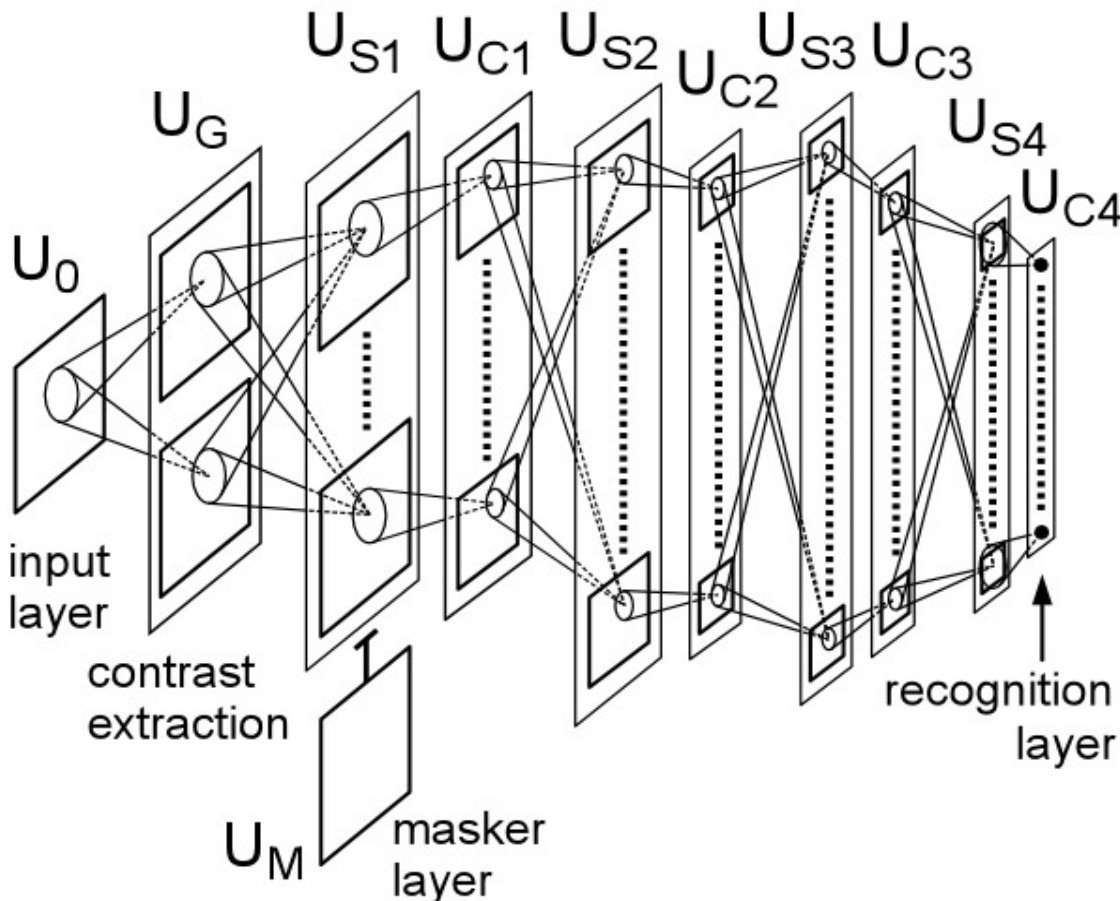
Deep Learning: Representations are hierarchical and trained



Early Hierarchical Feature Models for Vision

■ [Hubel & Wiesel 1962]:

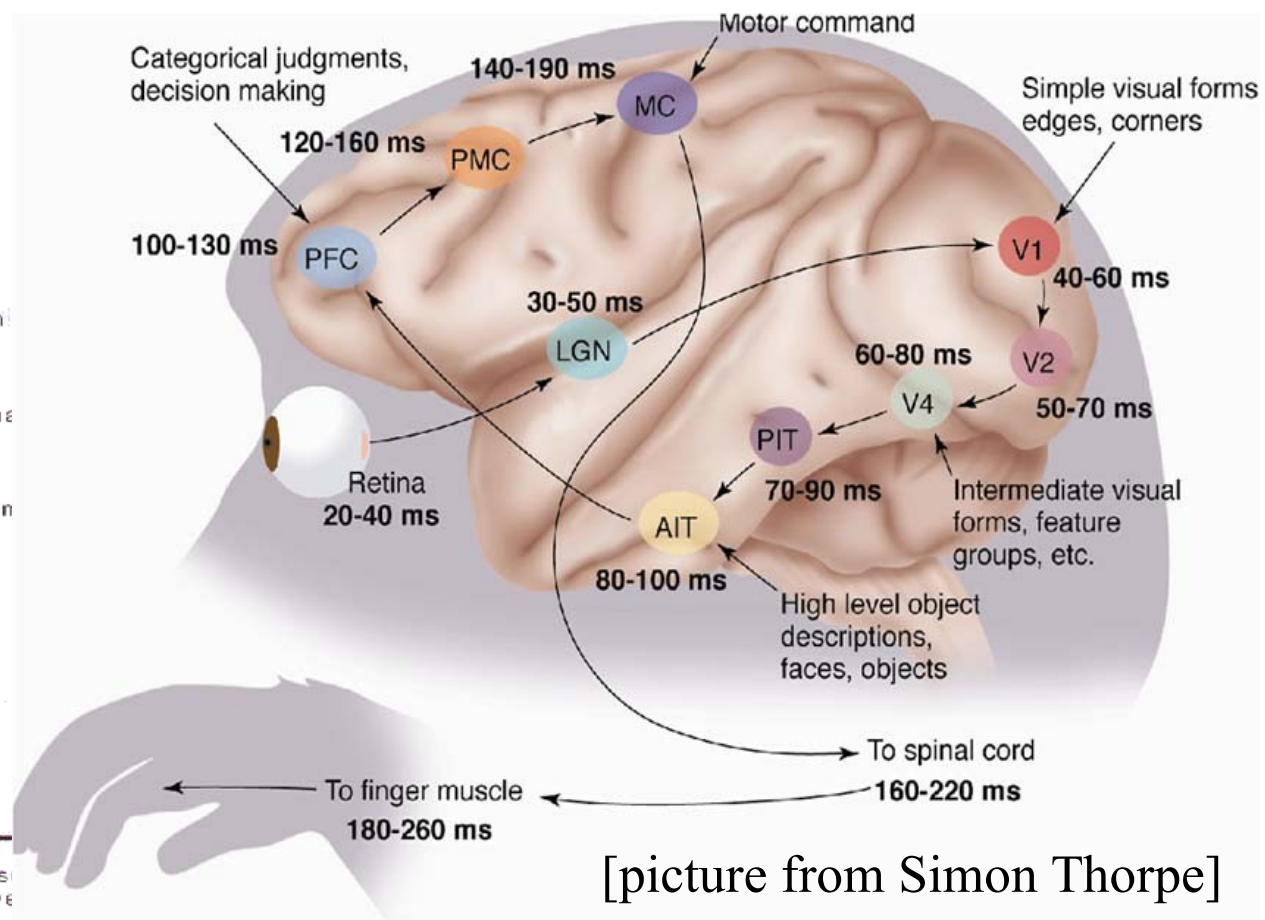
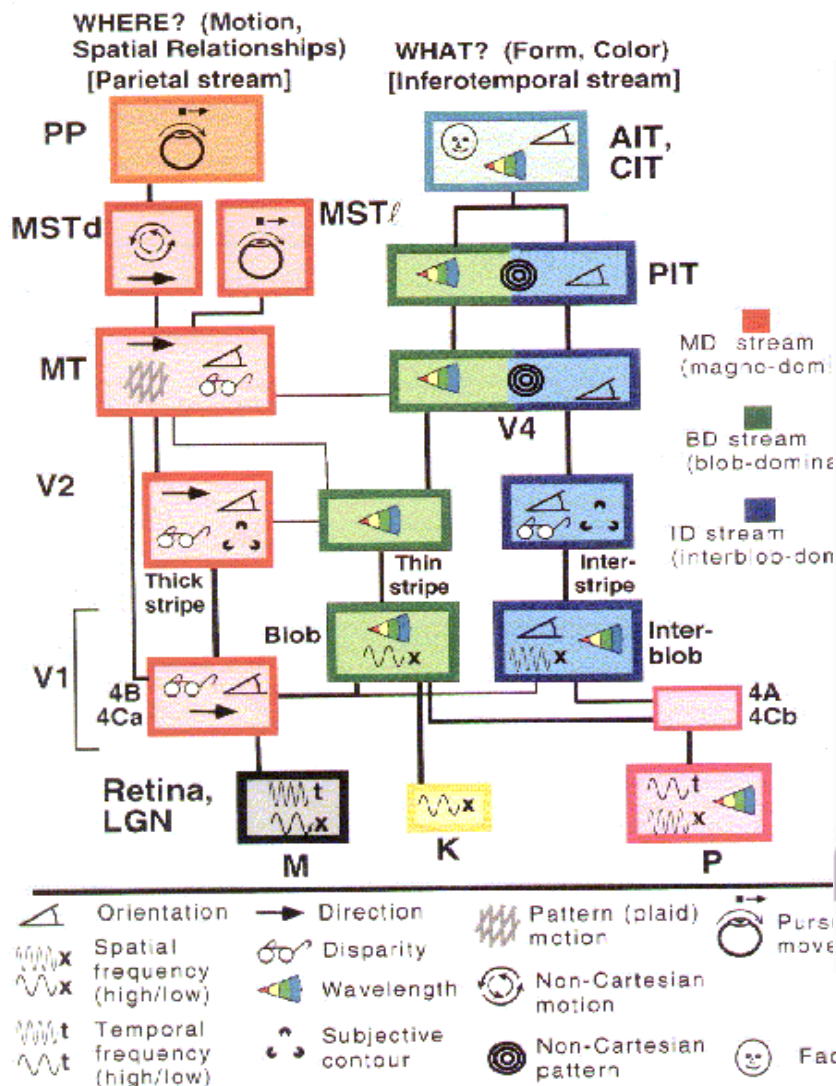
- ▶ **simple cells** detect local features
- ▶ **complex cells** “pool” the outputs of simple cells within a retinotopic neighborhood.



Cognitron & Neocognitron [Fukushima 1974-1982]

The Mammalian Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT
- Lots of intermediate representations



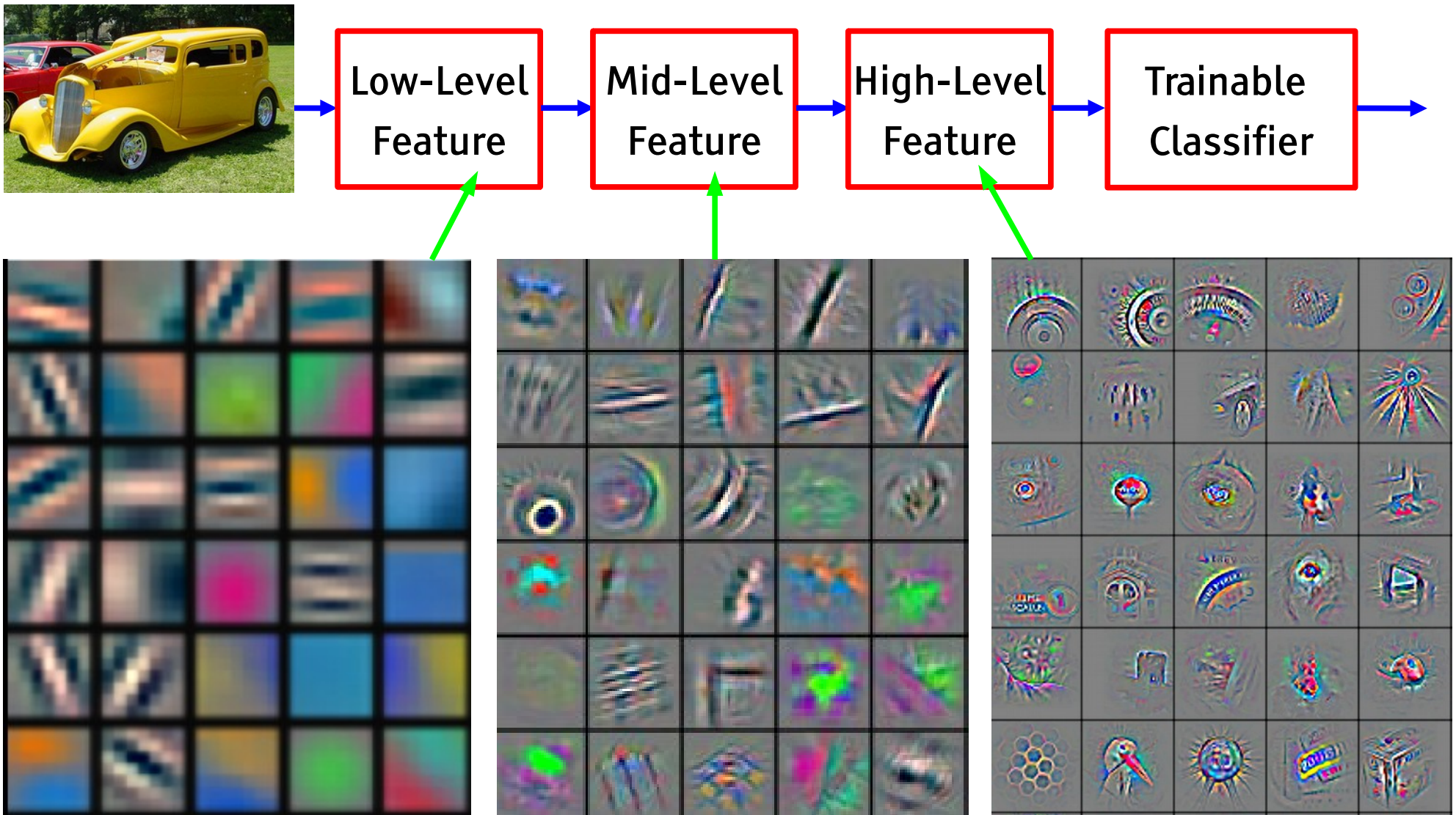
[picture from Simon Thorpe]

[Gallant & Van Essen]

Deep Learning = Learning Hierarchical Representations

Y LeCun

It's **deep** if it has **more than one stage** of non-linear feature transformation

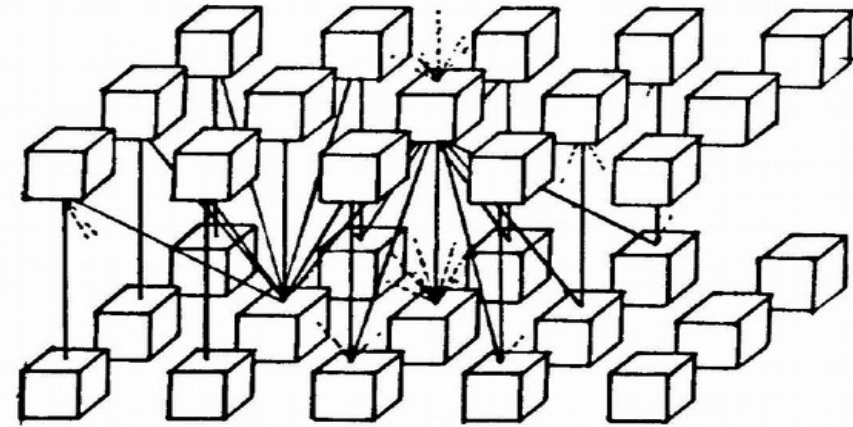
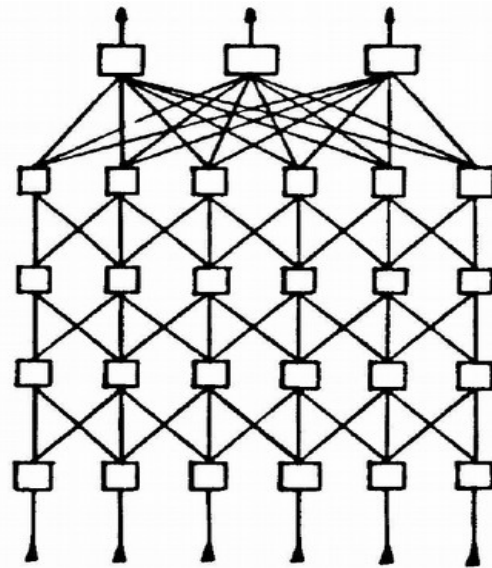


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

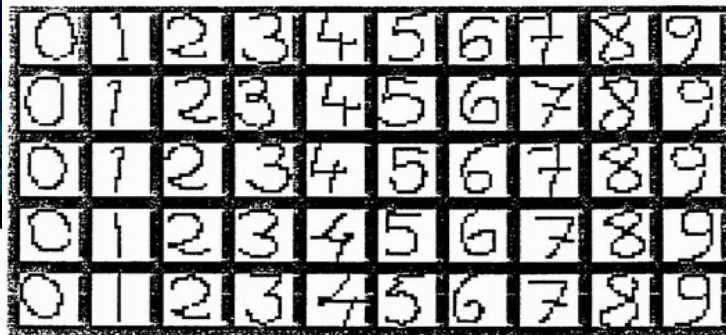
Early Networks [LeCun 85, 86]

Binary threshold units
trained supervised
with "target prop"

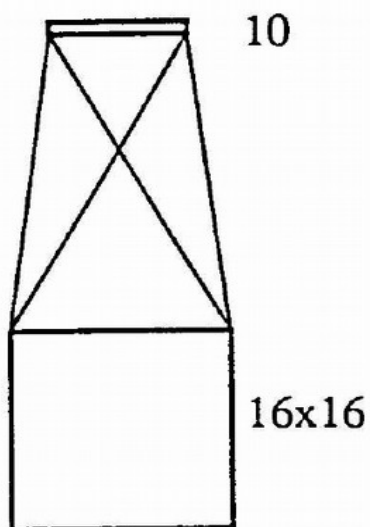
Hidden units compute a
virtual target



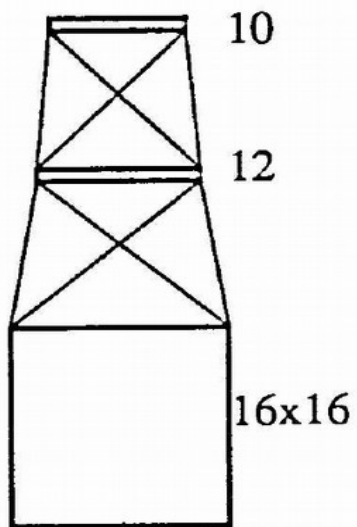
First ConvNets (U Toronto) [LeCun 88, 89]



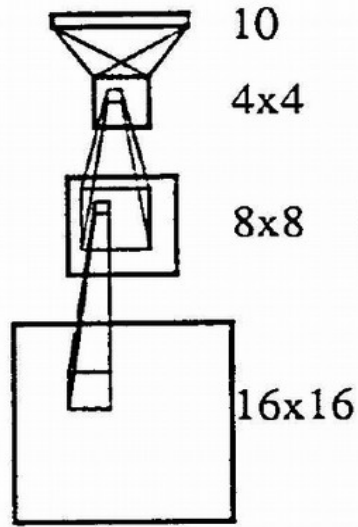
Trained with Backprop. 320 examples.



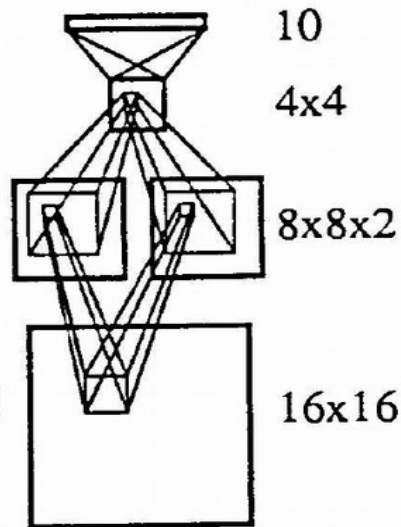
Single layer



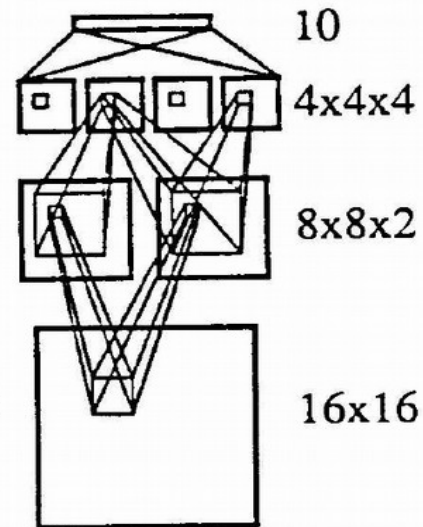
Two layers FC



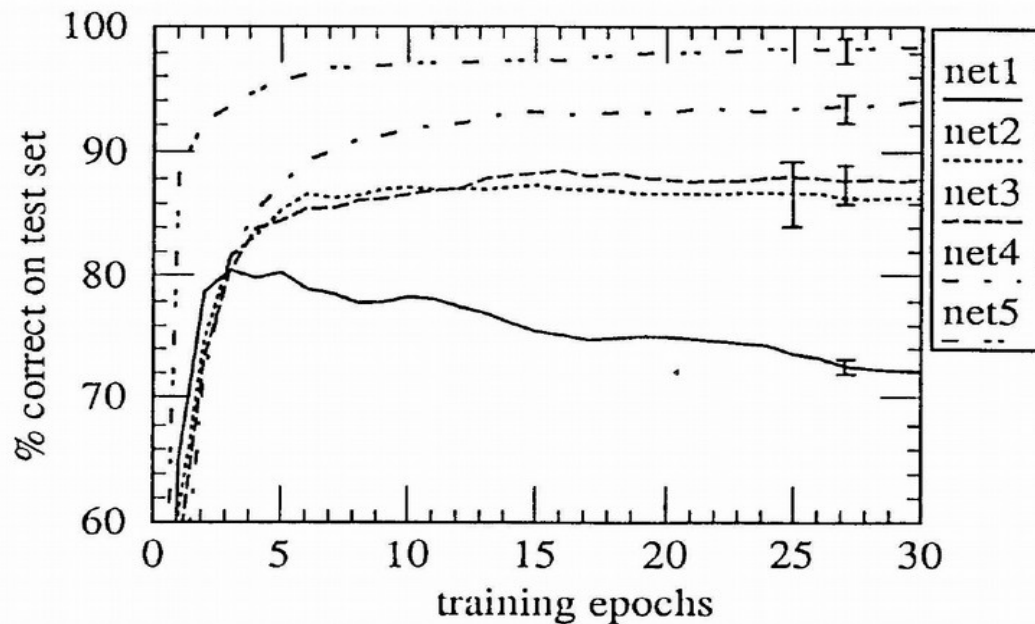
locally connected



Shared weights



Shared weights



- Convolutions with stride (subsampling)
- No separate pooling layers

network architecture	links	weights	performance
single layer network	2570	2570	80 %
two layer network	3240	3240	87 %
locally connected	1226	1226	88.5 %
constrained network	2266	1132	94 %
constrained network 2	5194	1060	98.4 %

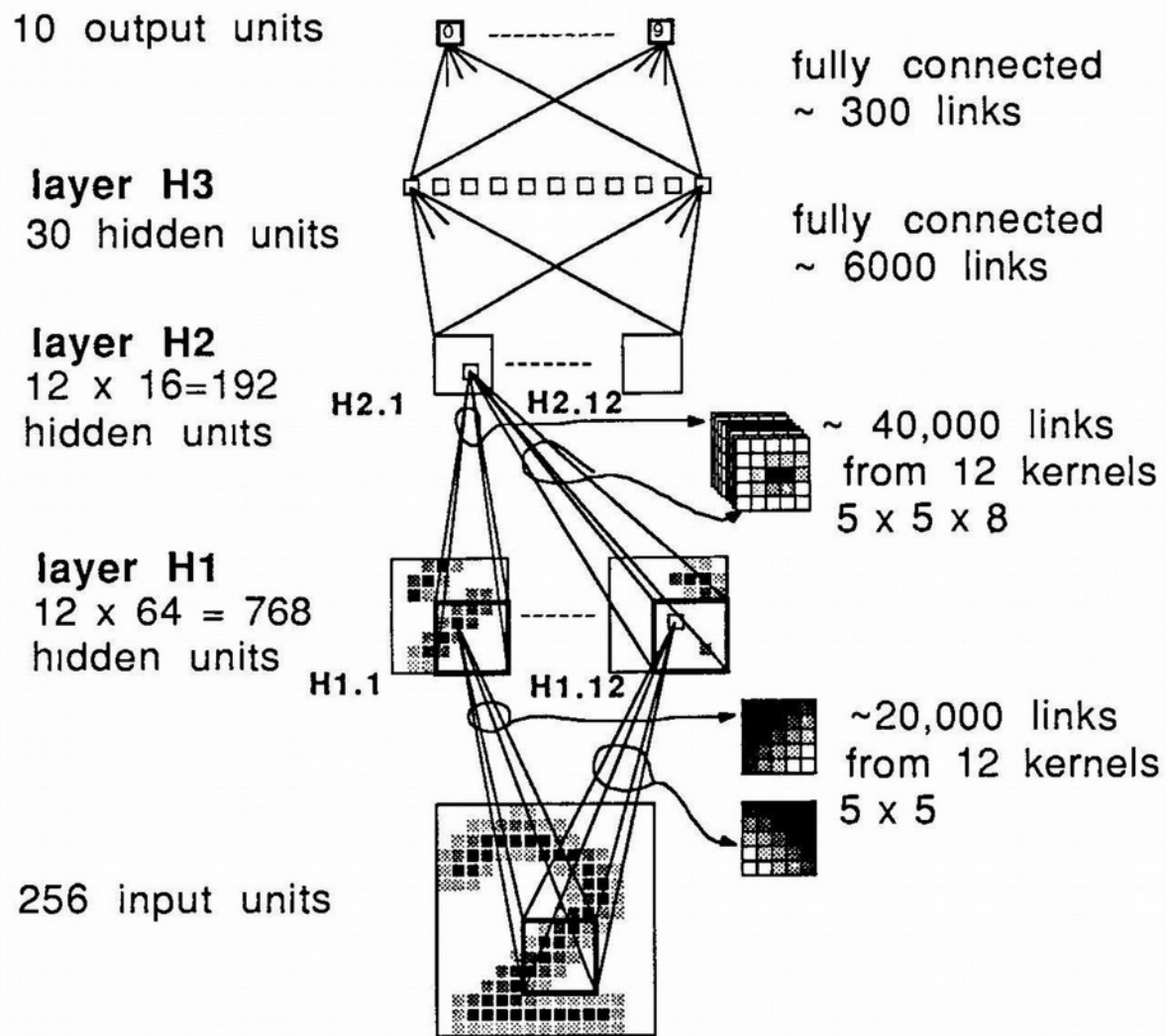
First "Real" ConvNets at Bell Labs [LeCun et al 89]

Y LeCun

Trained with Backprop.

USPS Zipcode digits: 7300 training, 2000 test.

Convolution with stride. No separate pooling.



80322-4129 80206

40004 14310

37879 05153

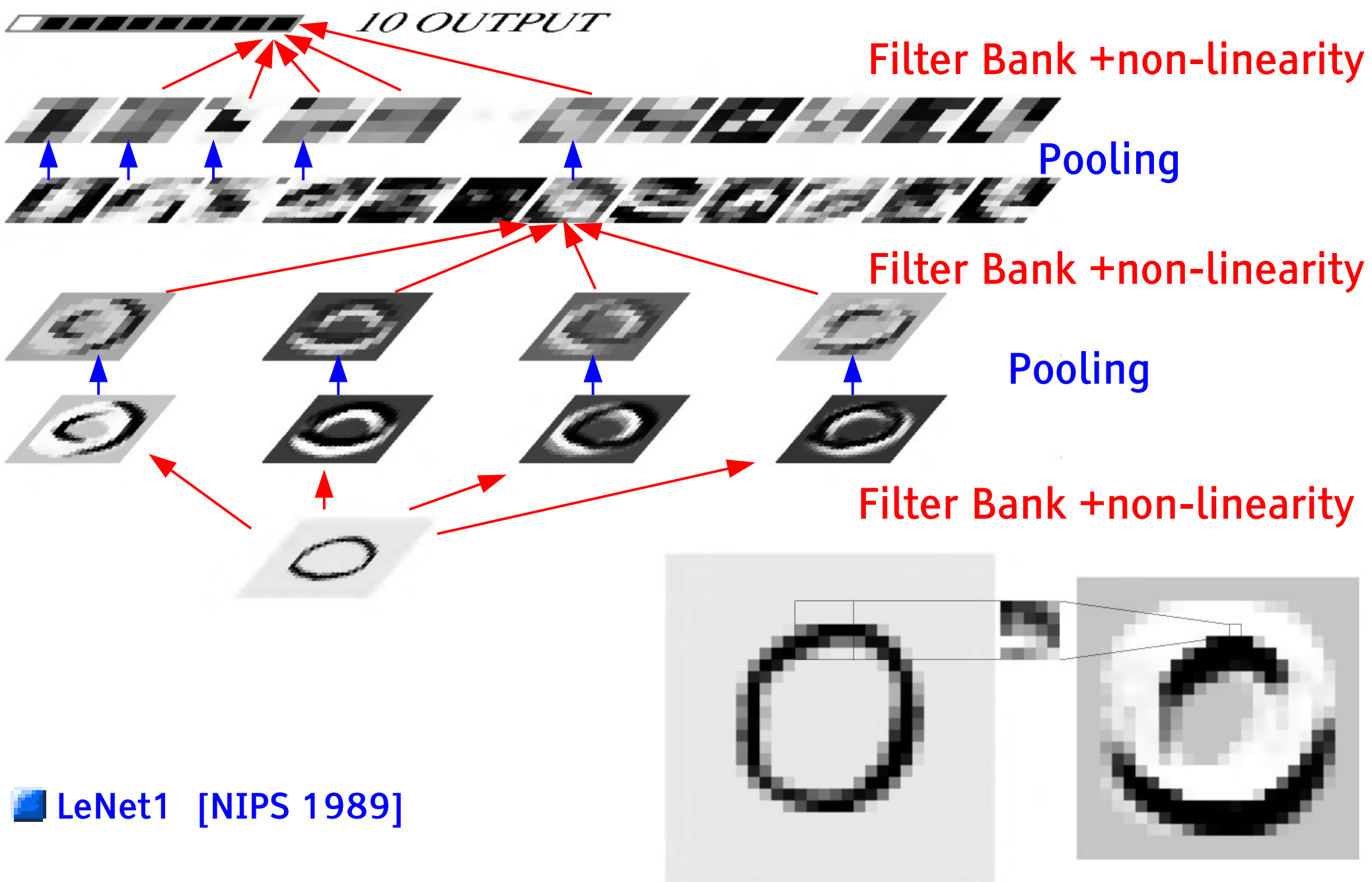
~~5502~~ 75216

35460 44209

1611913485726803226414186
 6359720299299722510046701
 3084111591010615406103631
 1064111030475262009979966
 8912056708557131427955460
 2018750187112991089970984
 0109707597331972015519055
 1075318255182814358090943
 1787521655460554603546055
 18255108503047520439401

ConvNet with separate pooling layer [LeCun et al 90]

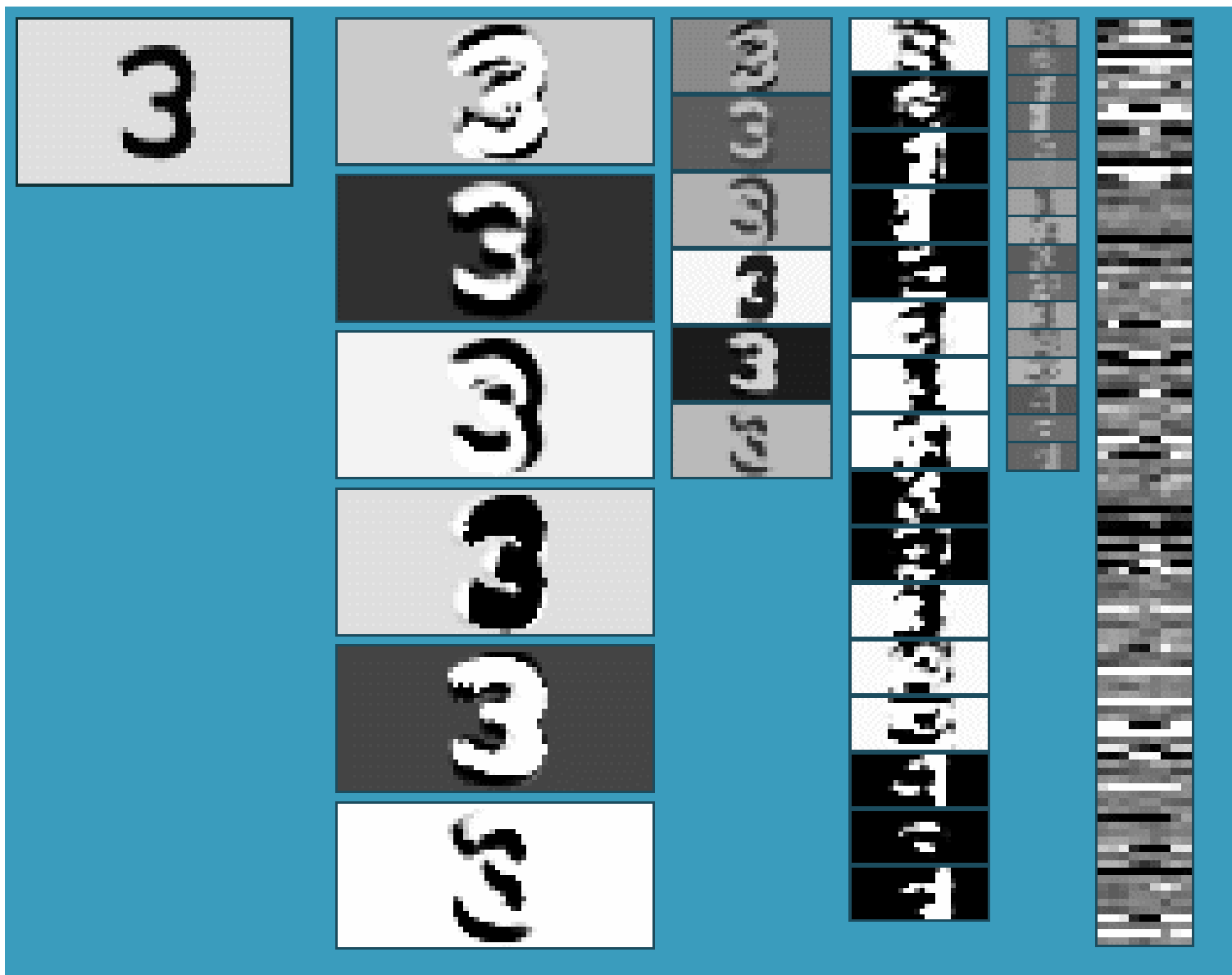
Y LeCun



LeNet1 [NIPS 1989]

Convolutional Network (vintage 1992)

Filters-tanh → pooling → filters-tanh → pooling → filters-tanh

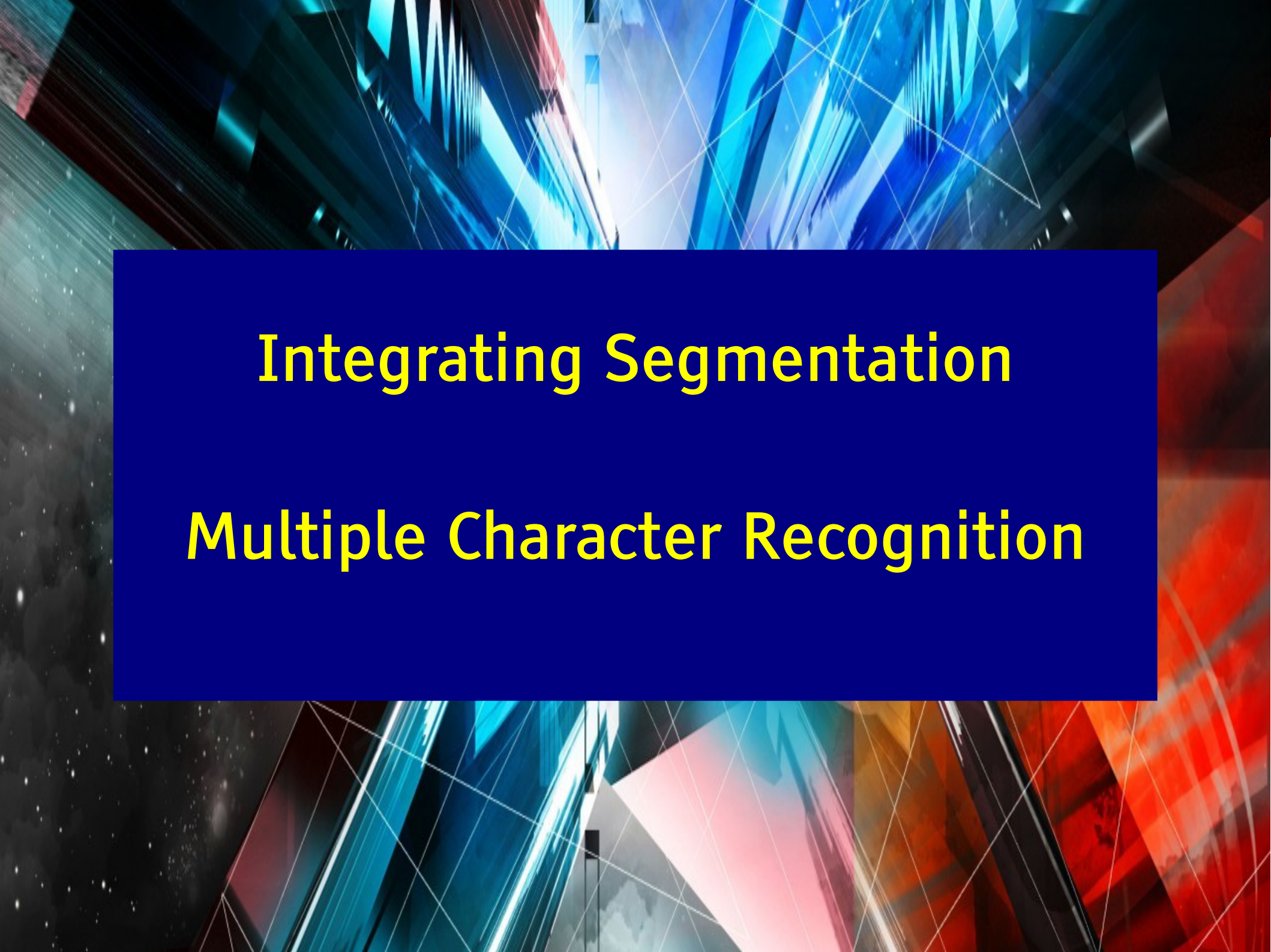


LeNet1 Demo from 1993

Y LeCun

Running on a 486 PC with an AT&T DSP32C add-on board (20 Mflops!)

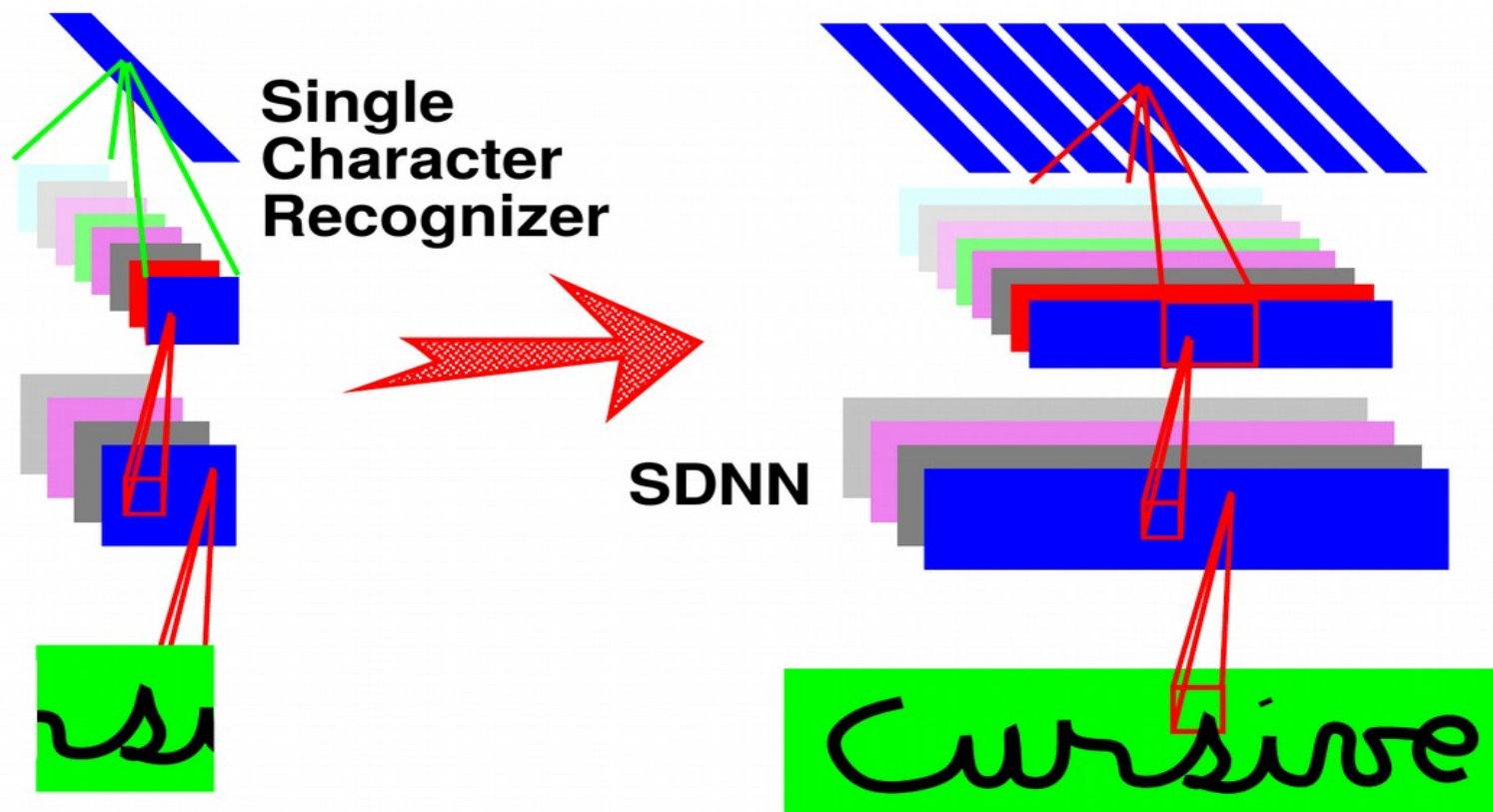




Integrating Segmentation Multiple Character Recognition

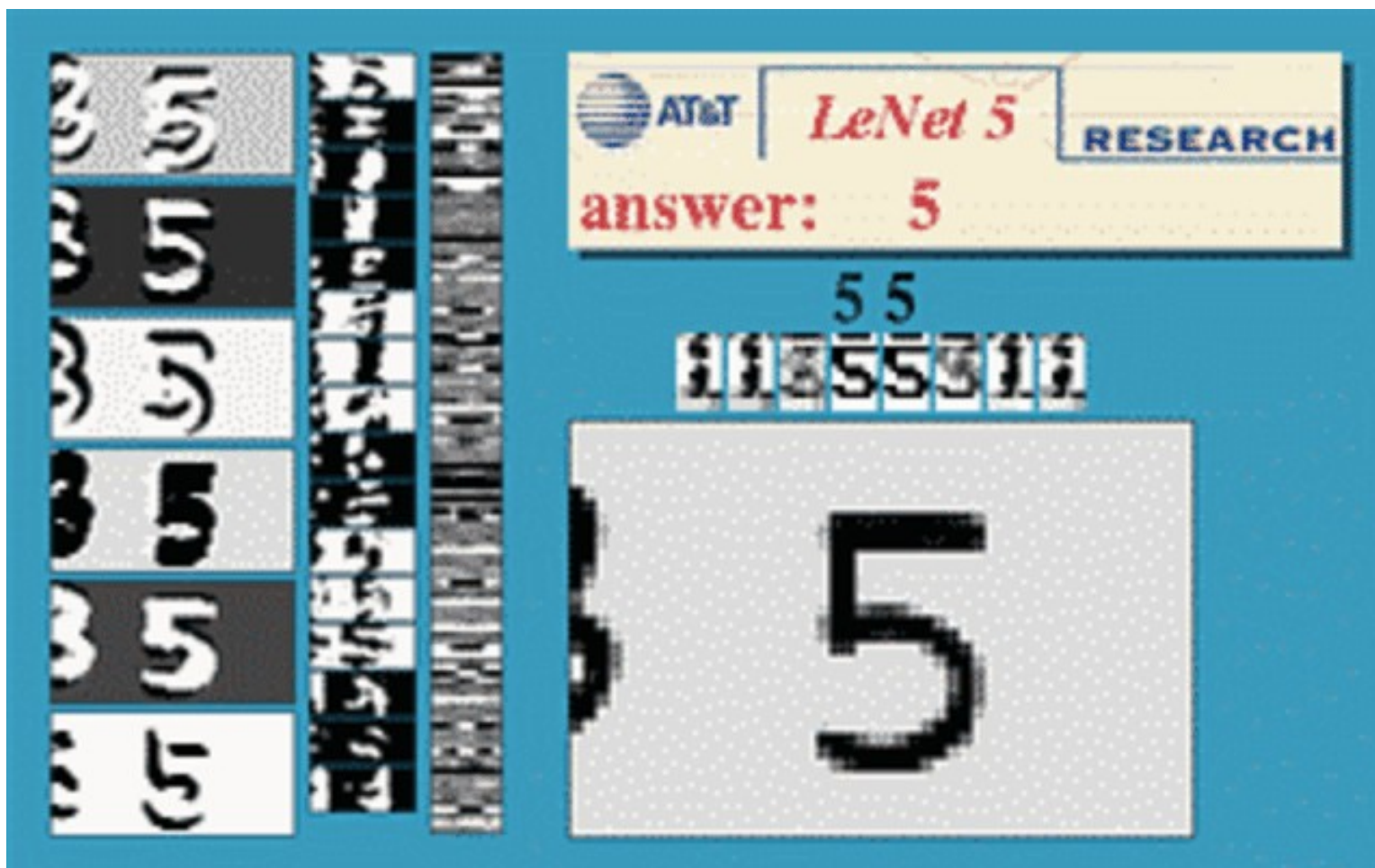
Multiple Character Recognition [Matan et al 1992]

- SDNN: Space Displacement Neural Net
- Also known as “replicated convolutional net”, or just ConvNet
 - (are we going to call this “fully convolutional net” now?)
- There is no such thing as a “fully connected layer”
- they are actually convolutional layers with 1x1 convolution kernels.



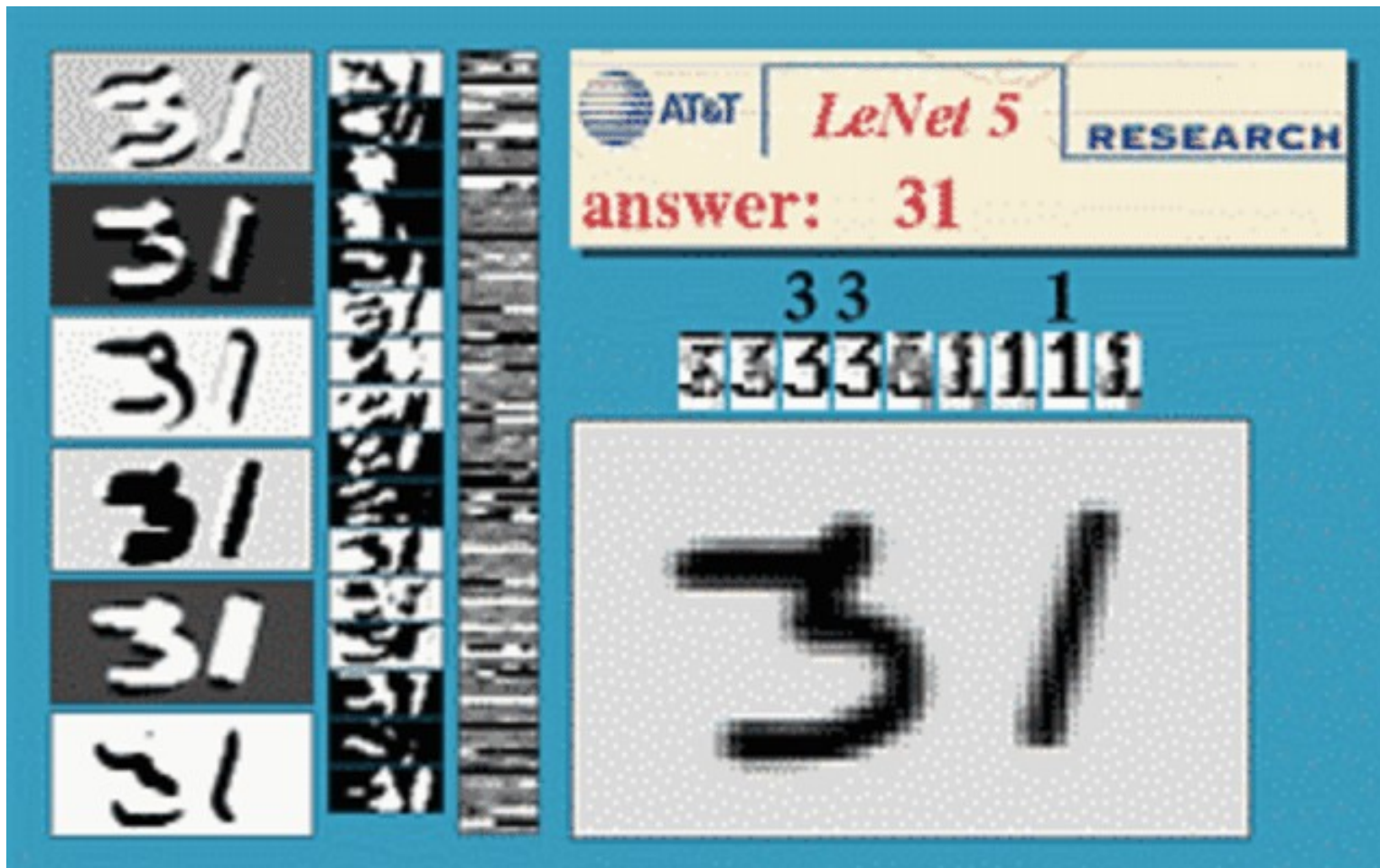
Multiple Character Recognition. Integrated Segmentation

- Trained with “semi synthetic” data
 - the individual character positions are known
- Training sample: a character painted with flanking characters or a inter-character space



Multiple Character Recognition. Integrated Segmentation

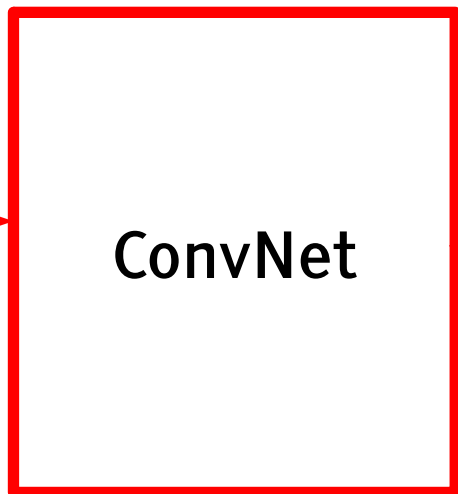
Y LeCun



Word-level training with weak supervision [Matan et al 1992]

Y LeCun

- Word-level training
- No labeling of individual characters
- How do we do the training?
- We need a "deformable part model"



5

4

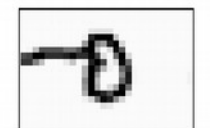
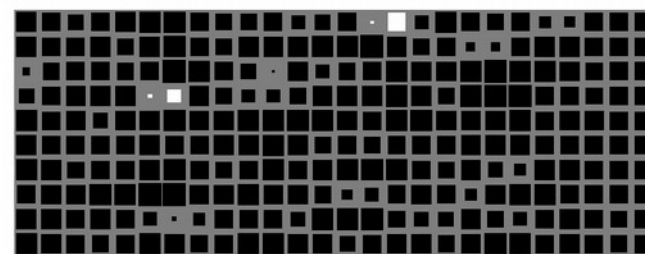
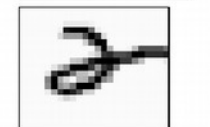
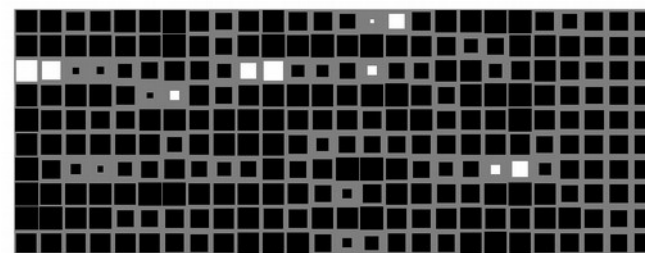
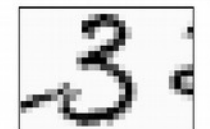
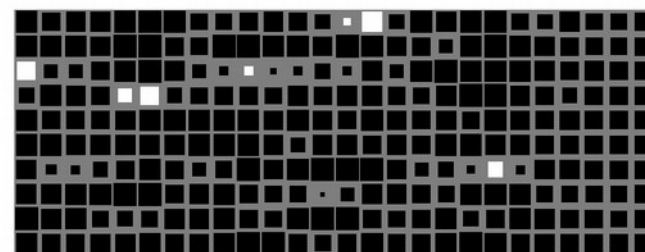
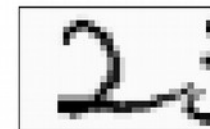
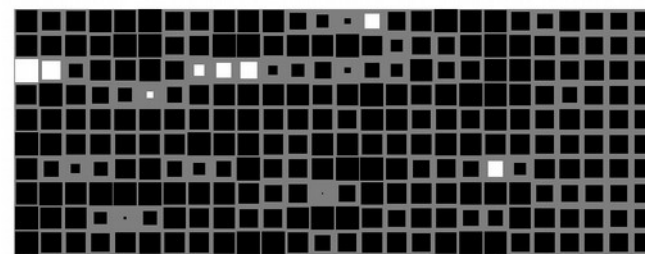
3

2

window width of each classifier

Multiple classifiers

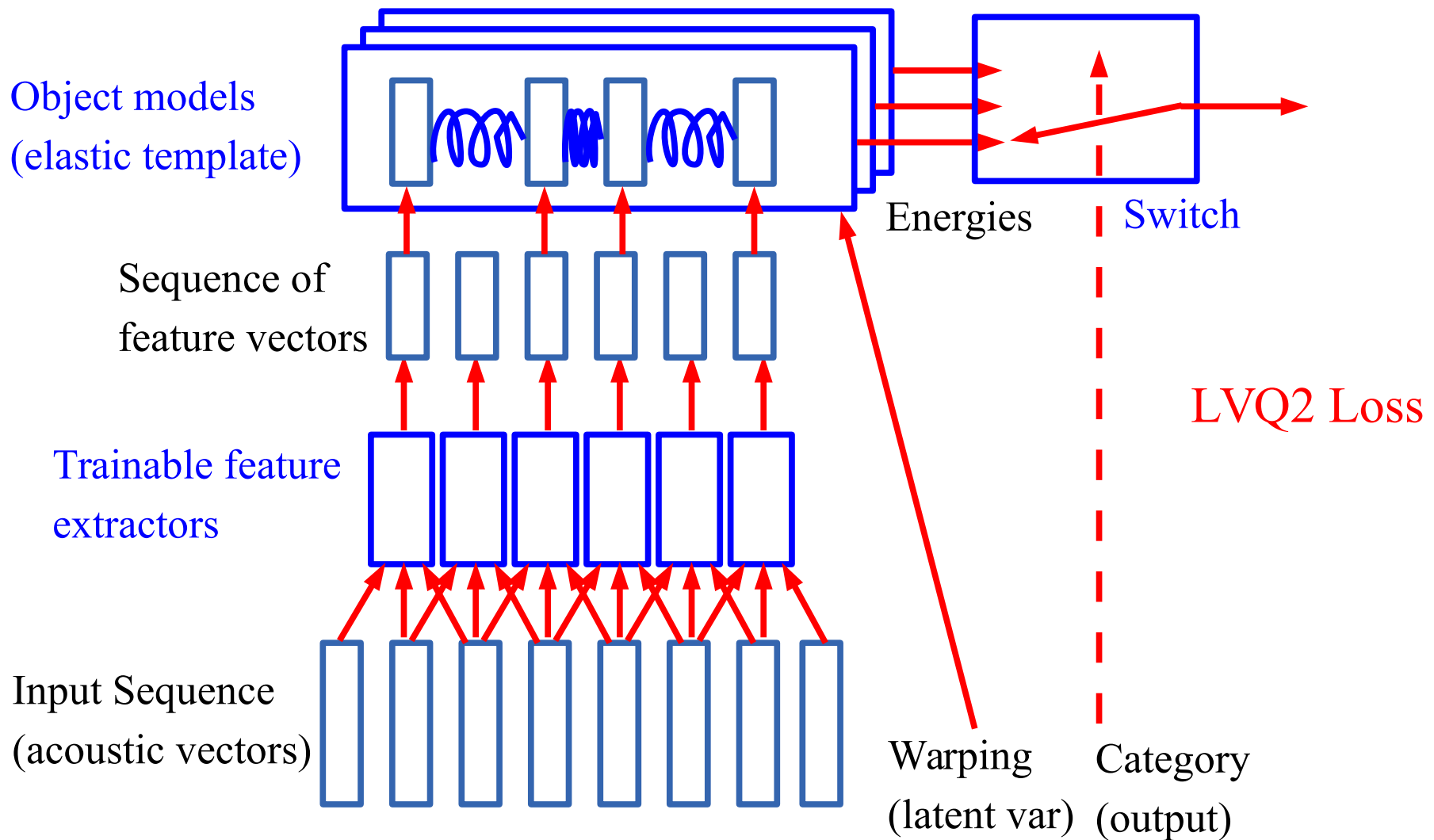
2 3 2 0 6



"Deformable Part Model" on top of a ConvNet [Driancourt, Bottou 1991]

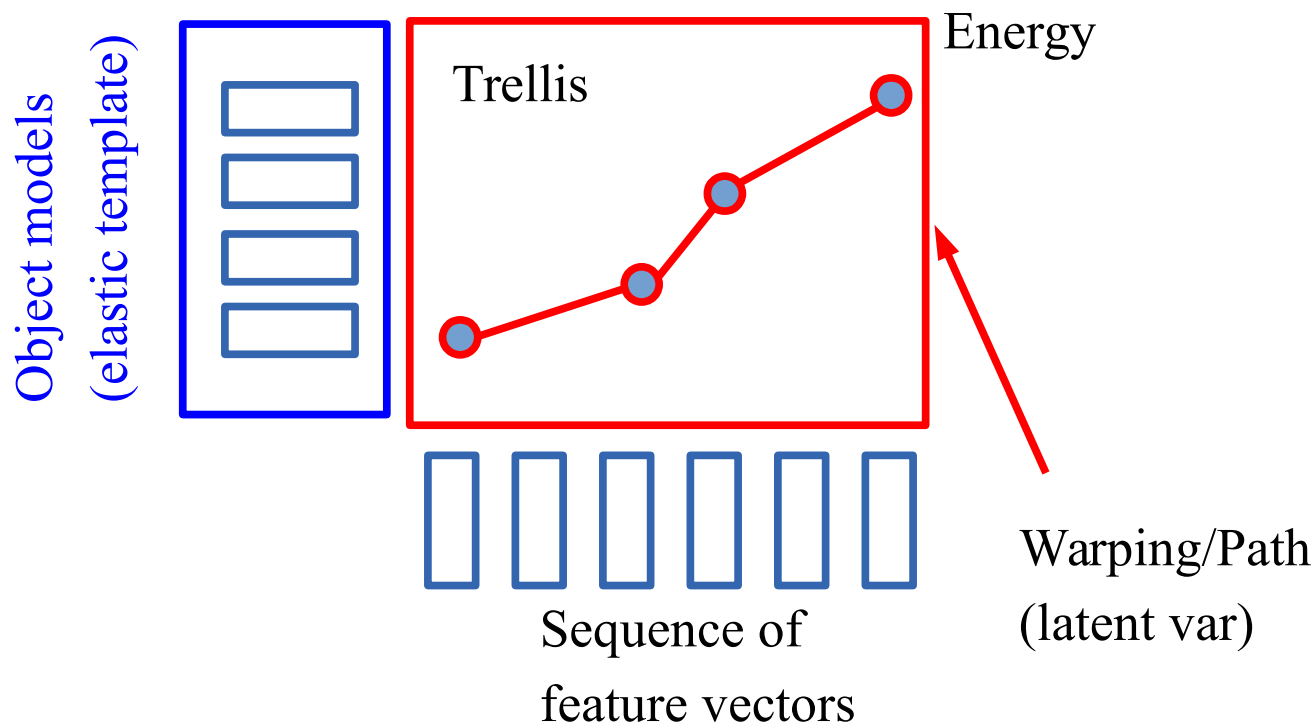
Y LeCun

Spoken word recognition with trainable elastic word templates.
First example of structured prediction on top of deep learning
[Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]



Word-level training with elastic word models

- Isolated spoken word recognition
- trainable elastic templates and trainable feature extraction
- Globally trained at the word level
- Elastic matching using dynamic time warping
 - Viterbi algorithm on a trellis.



[Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]

The Oldest Example of Structured Prediction & Deep Learning

Trainable Automatic Speech Recognition system with a **convolutional net** (TDNN) and dynamic time warping (DTW)

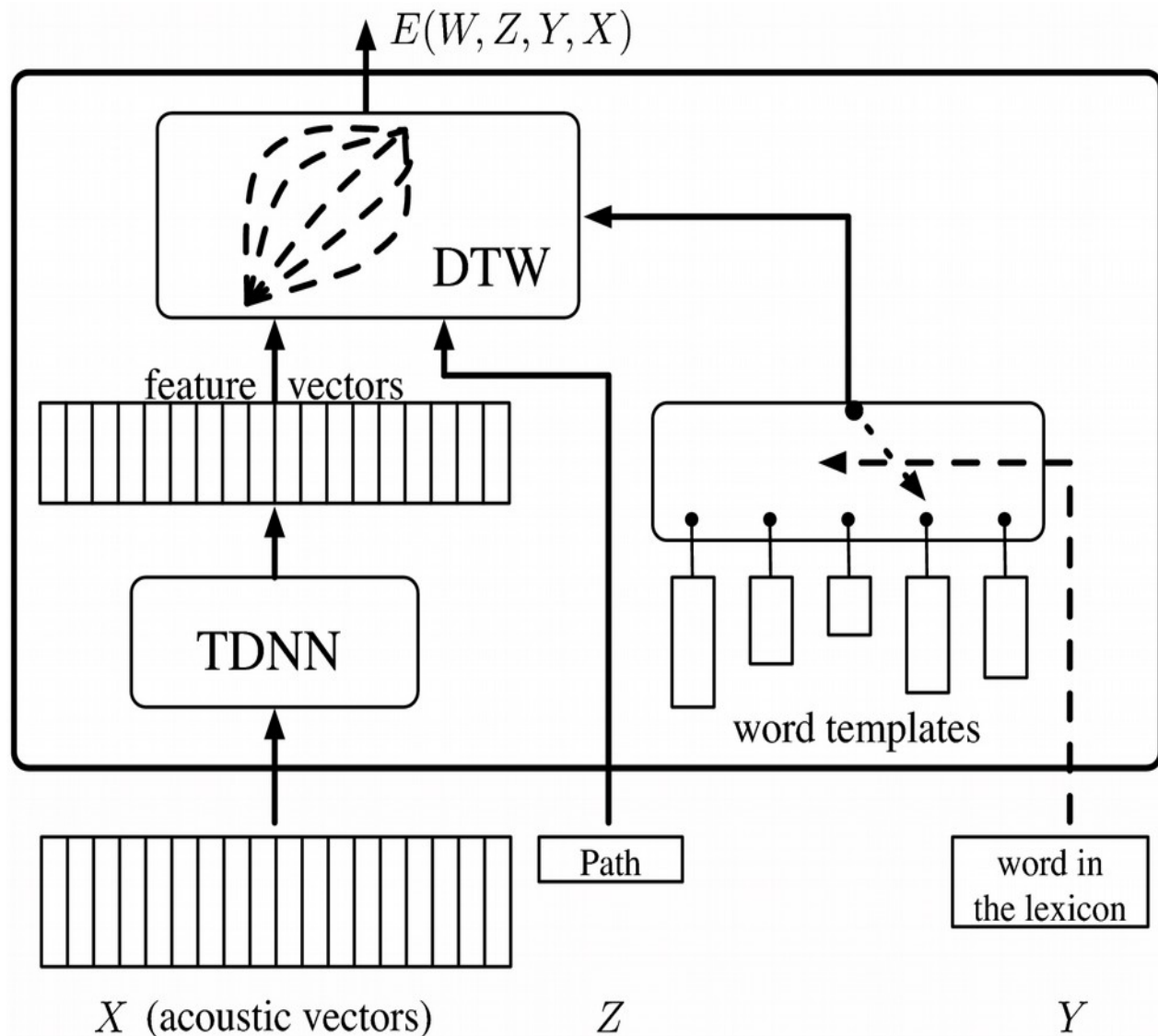
The feature extractor and the structured classifier are trained simultaneously in an integrated fashion.

with the LVQ2 Loss :

- Driancourt and Bottou's speech recognizer (1991)

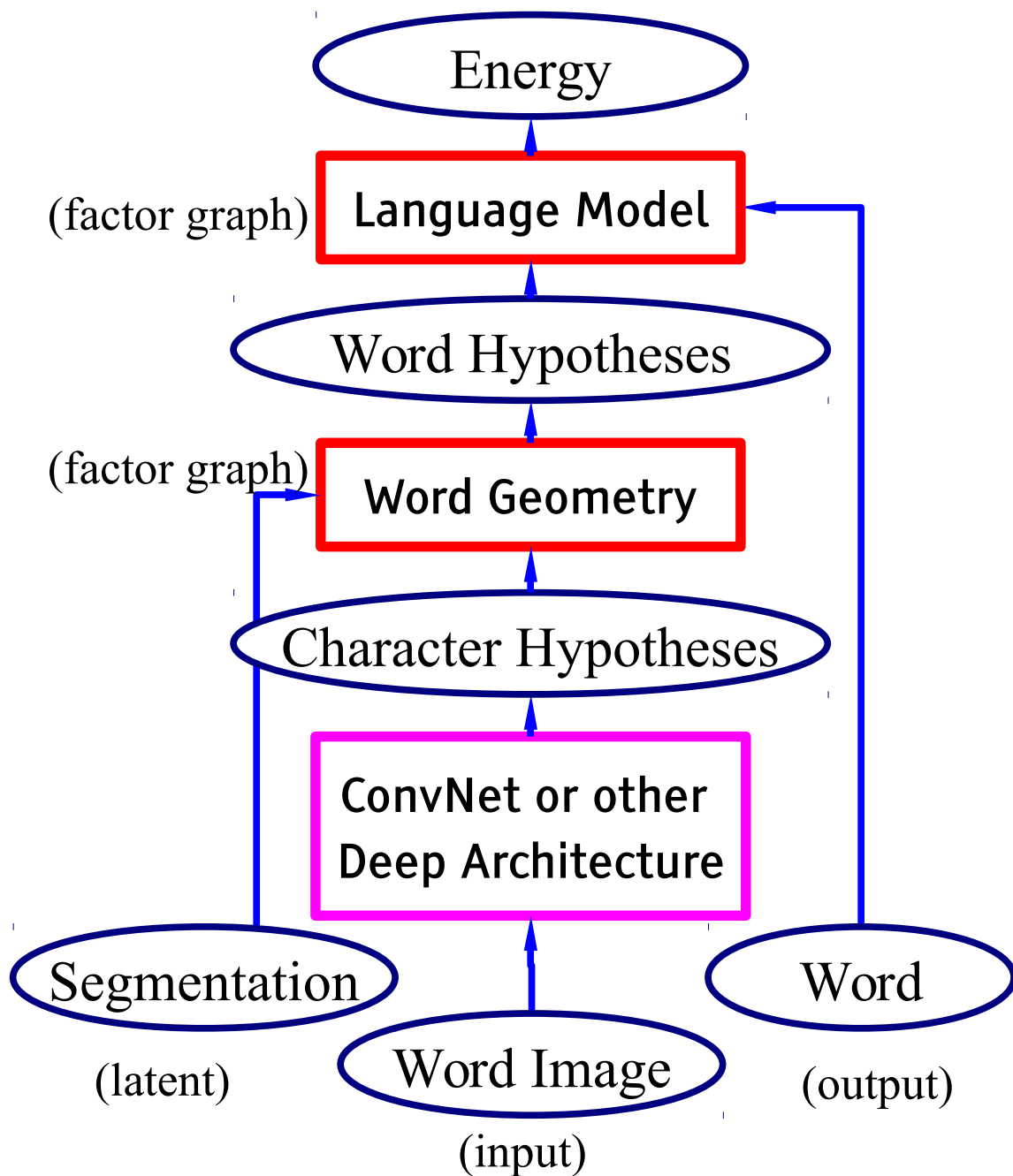
with Neg Log Likelihood:

- Bengio's speech recognizer (1992)
- Haffner's speech recognizer (1993)



End-to-End Learning – Word-Level Discriminative Training

Y LeCun



Making every single module in the system trainable.

Every module is trained simultaneously so as to optimize a **global loss function**.

Includes the feature extractor, the recognizer, and the contextual post-processor (graphical model)

Problem: back-propagating gradients through the graphical model.

"Shallow" Structured Prediction

Energy function is linear in the parameters

$$E(X, Y, Z) = \sum_i W_i^T h_i(X, Y, Z)$$

with the NLL Loss :

- **Conditional Random Field**
[Lafferty, McCallum, Pereira 2001]

with Hinge Loss:

- **Max Margin Markov Nets** and **Latent SVM** [Taskar, Altun, Hofmann...]

with Perceptron Loss

- **Structured Perceptron**
[Collins...]

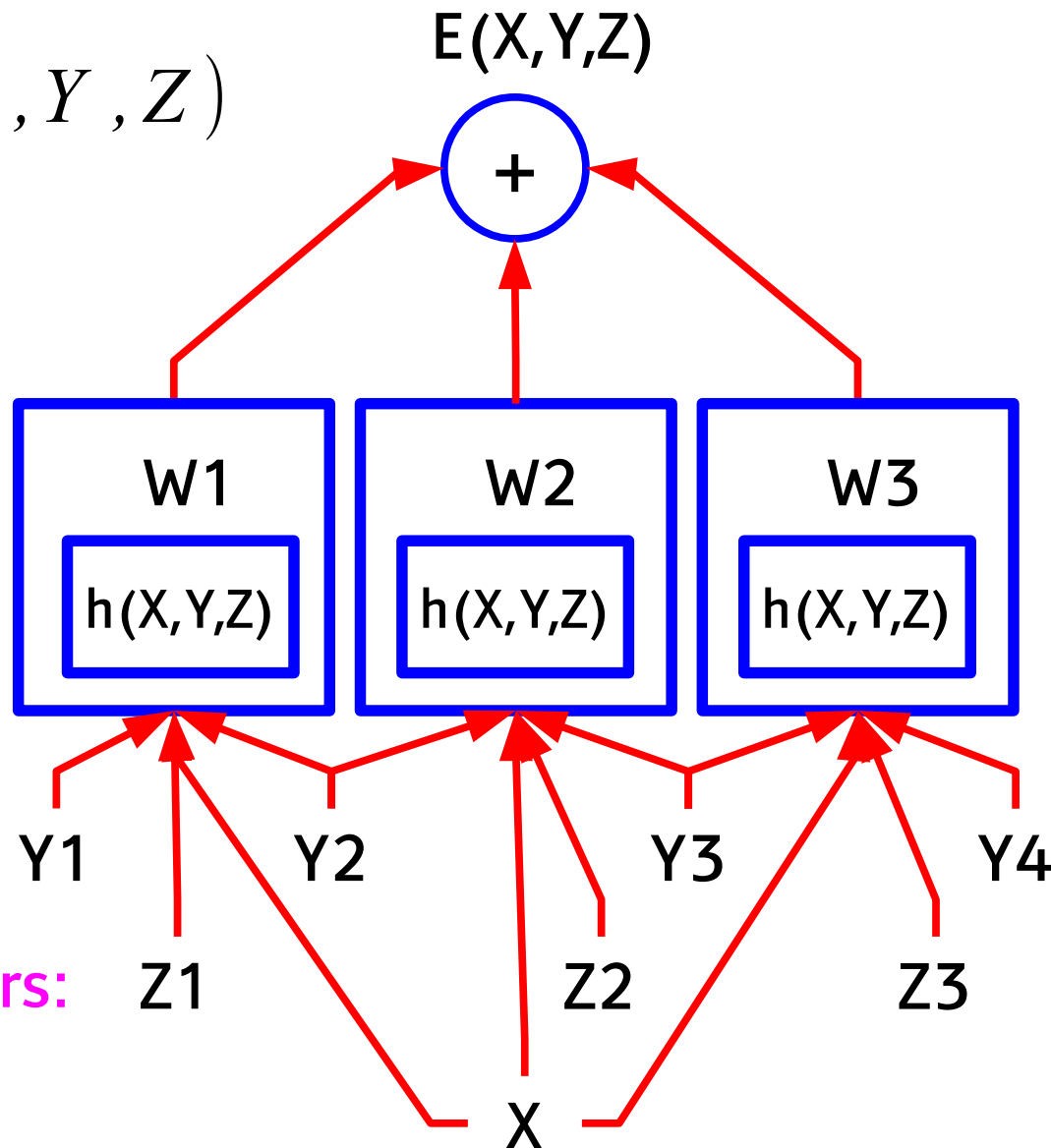
Params

Features

Outputs:

Latent Vars:

Input:



Deep Structured Prediction

Energy function is linear in the parameters

$$E(X, Y, Z) = \sum_i g_i(X, Y, Z, W_i)$$

Graph Transformer Networks

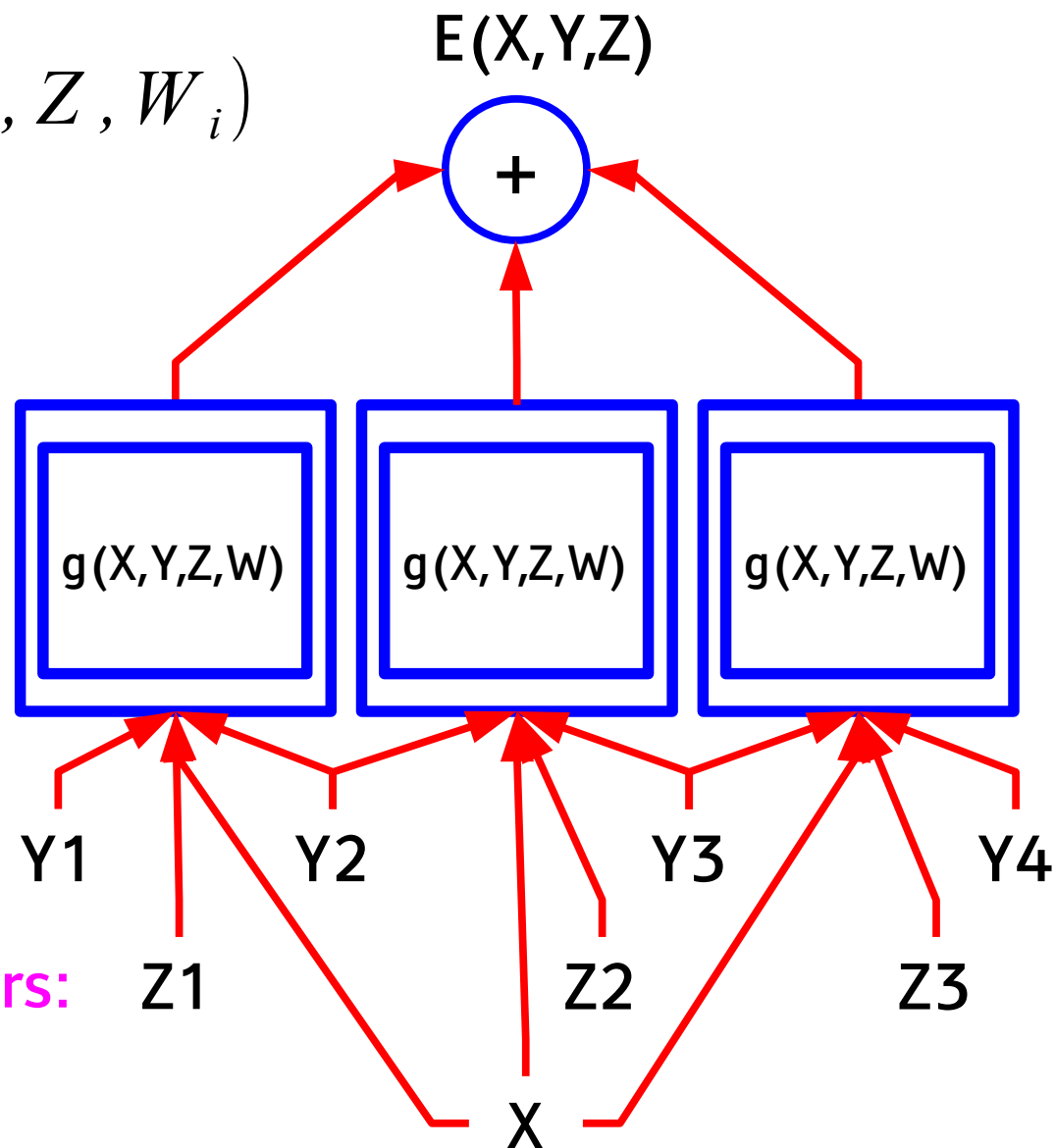
- [LeCun, Bottou, Bengio, Haffner 97,98]
- NLL loss
- Perceptron loss

ConvNet

Outputs:

Latent Vars:

Input:



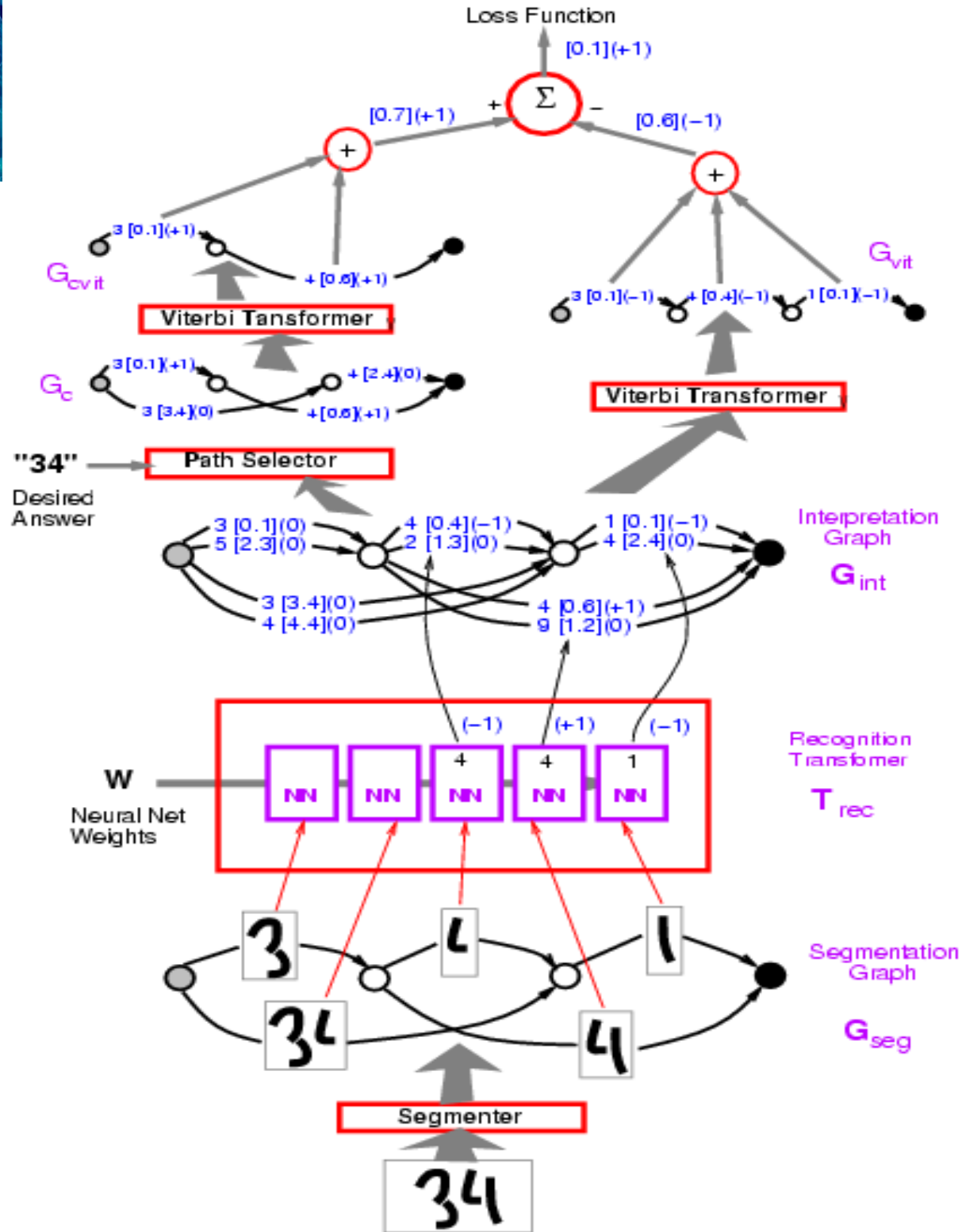
Graph Transformer Networks

Structured Prediction
on top of Deep Learning

This example shows the structured perceptron loss.

In practice, we used negative log-likelihood loss.

Deployed in 1996 in check reading machines.



Check Reader

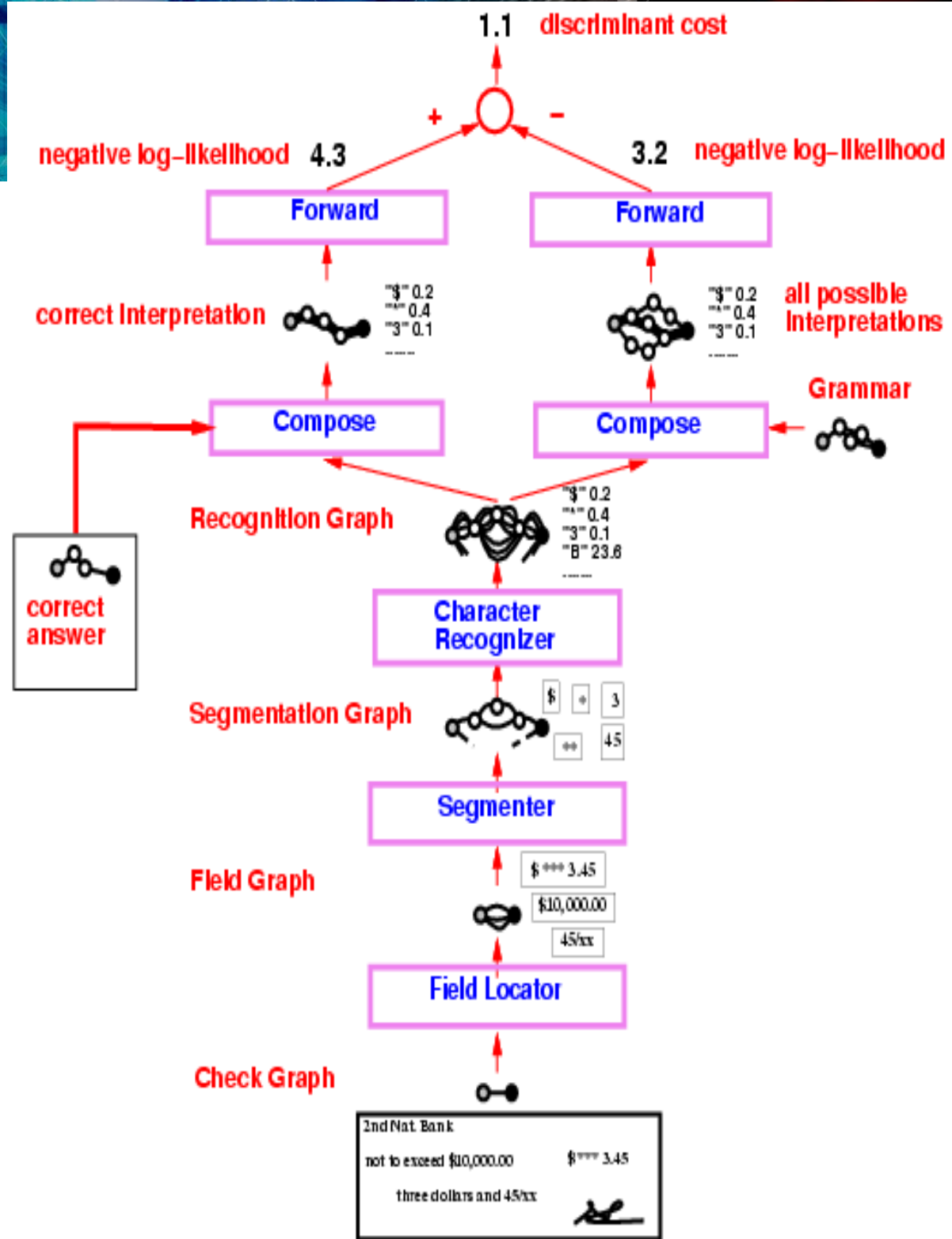
Graph transformer network trained to read **check amounts**.

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% to 20%** of all the checks written in the US.

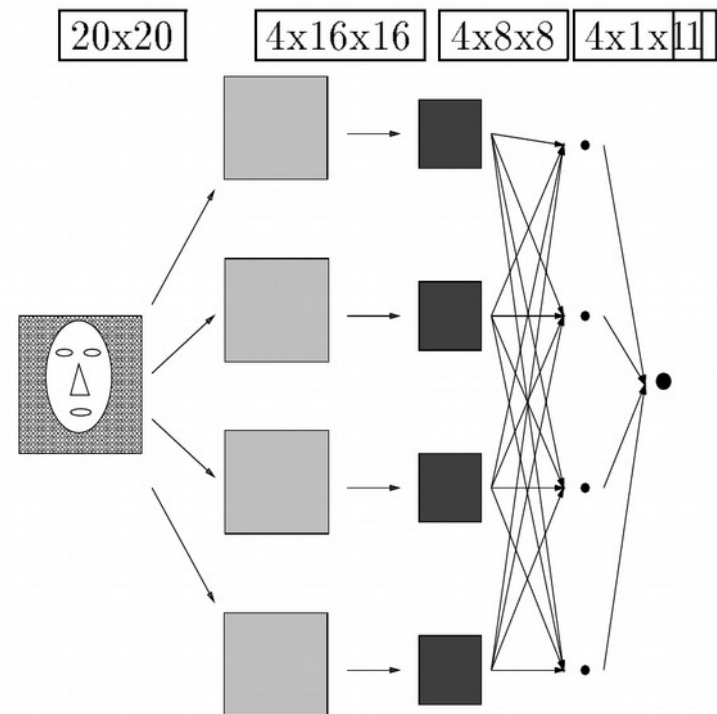




Object Detection

Face Detection [Vaillant et al. 93, 94]

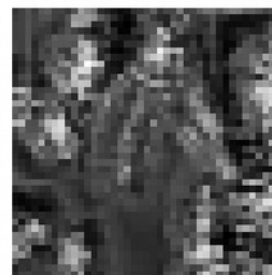
- ConvNet applied to large images
- Heatmaps at multiple scales
- Non-maximum suppression for candidates
- 6 second on a Sparcstation for 256x256 image



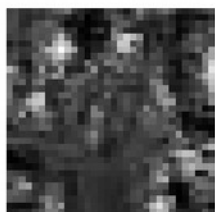
Scale 3



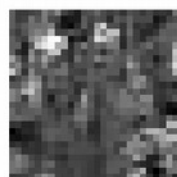
Scale 4



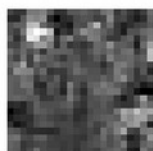
Scale 5



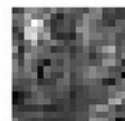
Scale 6



Scale 7



Scale 8



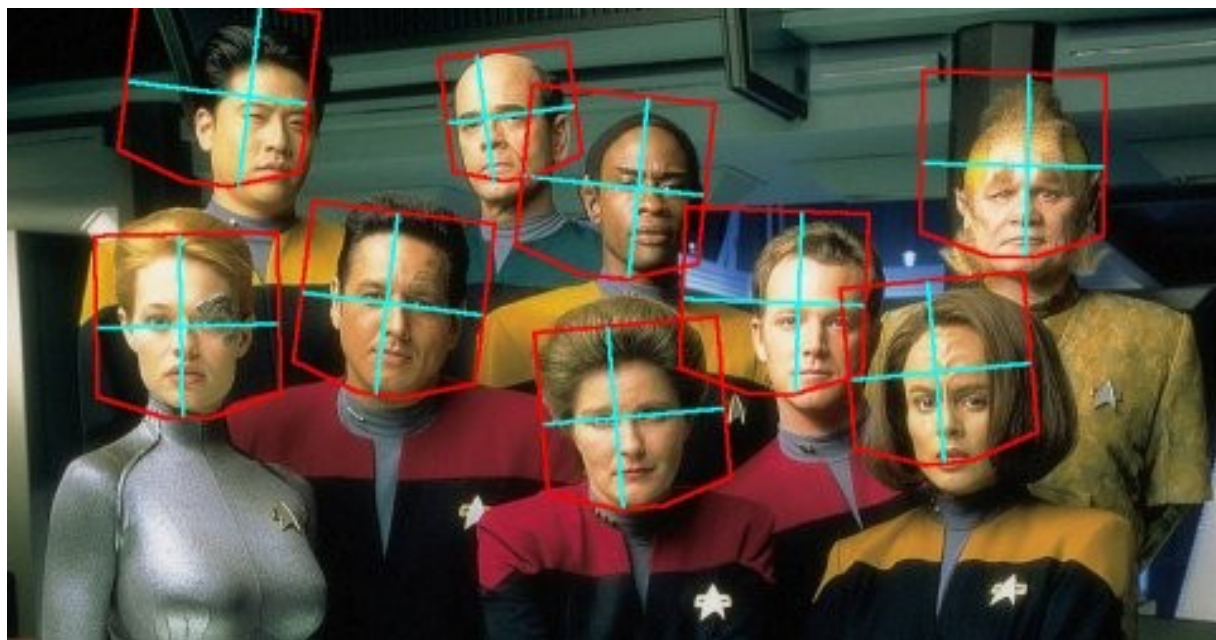
Scale 9



mid 2000s: state of the art results on face detection

Y LeCun

<i>Data Set-></i>	TILTED		PROFILE		MIT+CMU	
<i>False positives per image-></i>	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	

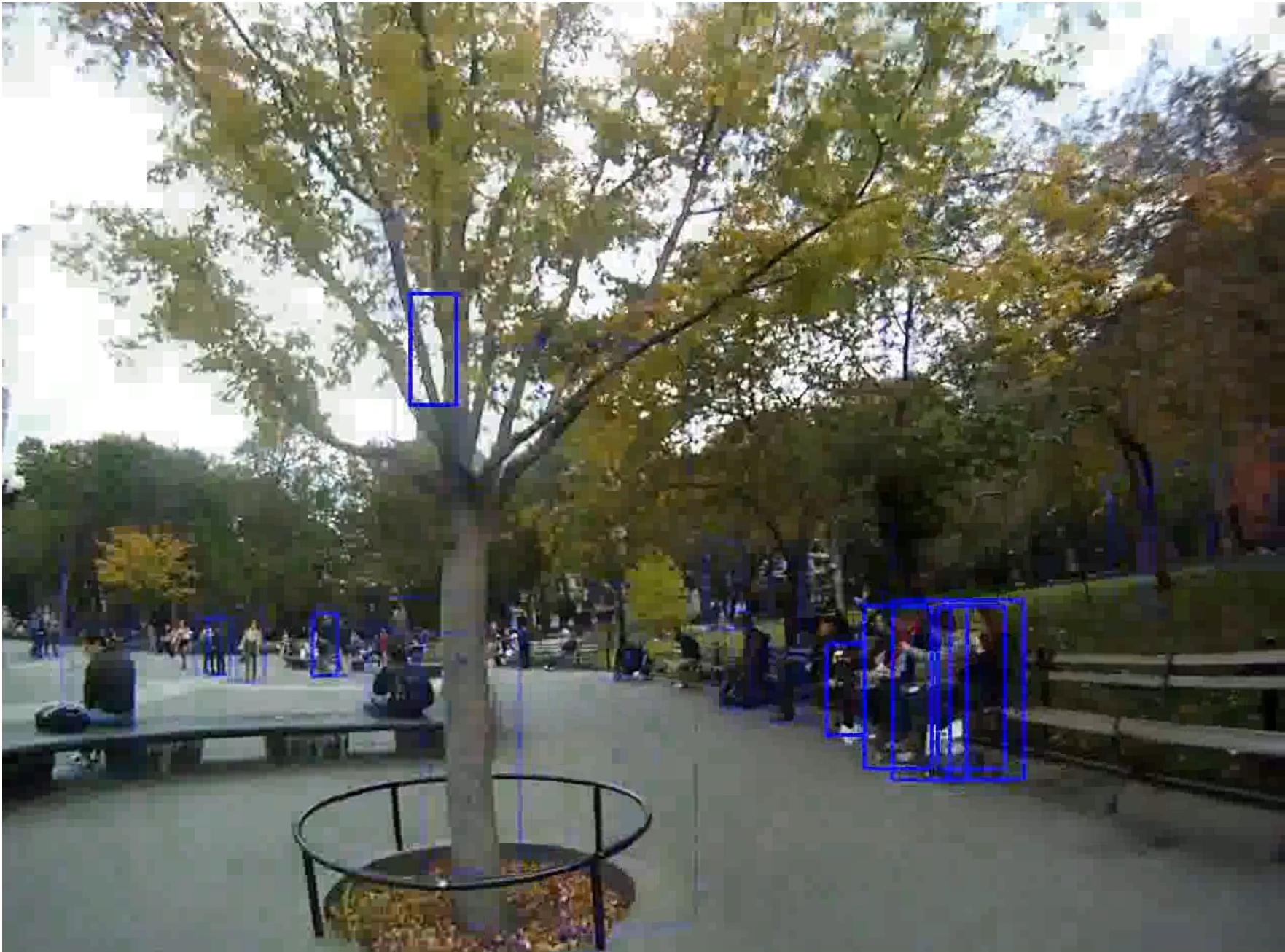


[Garcia & Delakis 2003] [Osadchy et al. 2004] [Osadchy et al, JMLR 2007]

Simultaneous face detection and pose estimation

Y LeCun







Semantic Segmentation

ConvNets for Biological Image Segmentation

Y LeCun

■ Biological Image Segmentation

▶ [Ning et al. IEEE-TIP 2005]

■ Pixel labeling with large context using a convnet

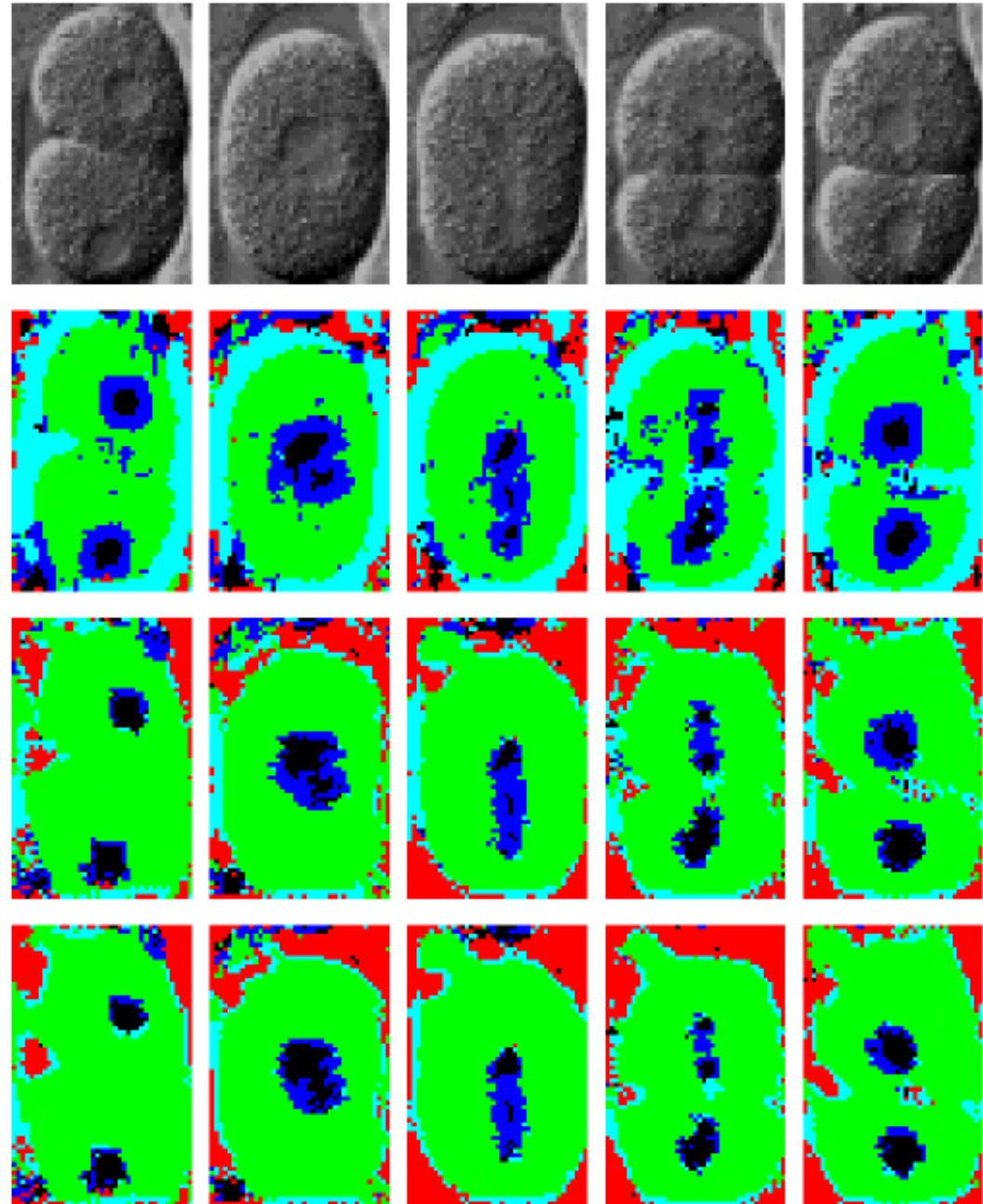
■ ConvNet takes a window of pixels and produces a label for the central pixel

■ Cleanup using a kind of conditional random field (CRF)

▶ Similar to a field of expert, but conditional.

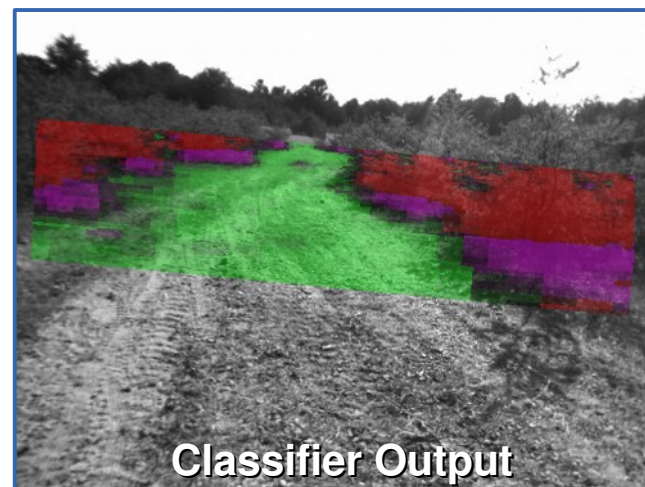
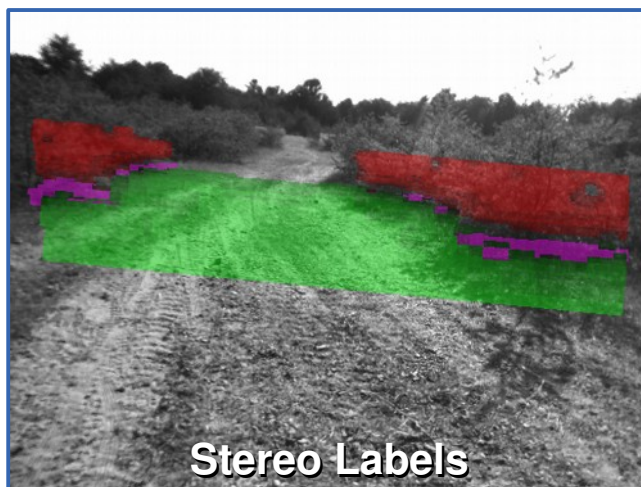
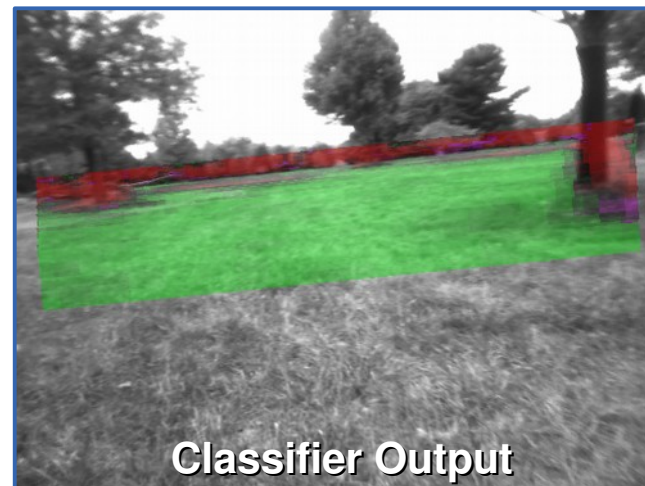
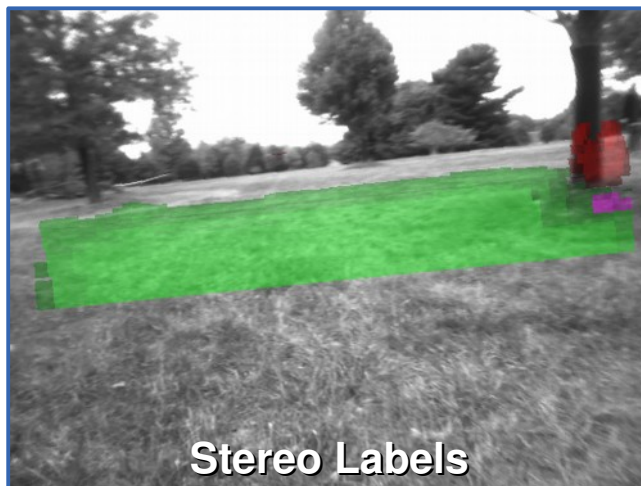
■ 3D version for connectomics

▶ [Jain et al. 2007]



ConvNet for Long Range Adaptive Robot Vision (DARPA LAGR program 2005-2008)

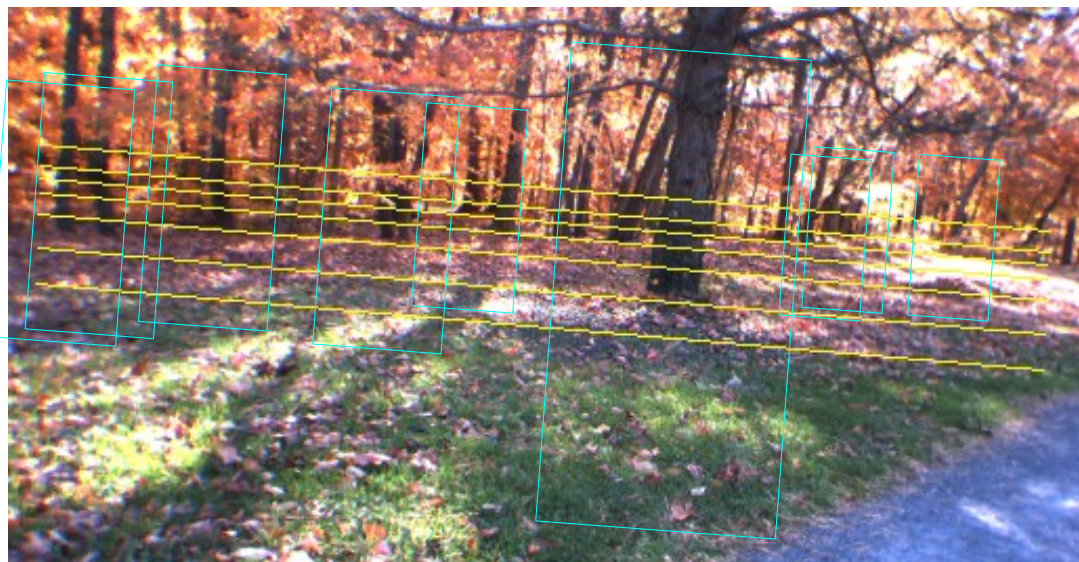
Y LeCun



[Hadsell et al., J. Field Robotics 2009]

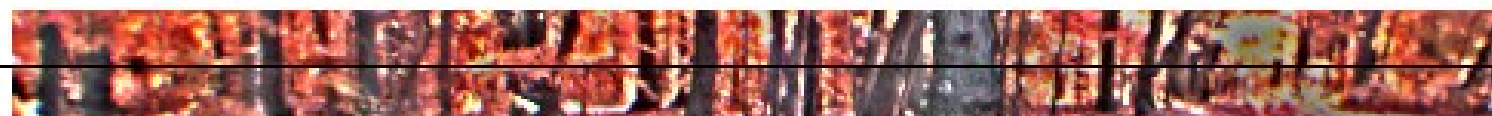
Long Range Vision with a Convolutional Net

Y LeCun



Pre-processing (125 ms)

- Ground plane estimation
- Horizon leveling
- Conversion to YUV + local contrast normalization
- Scale invariant pyramid of distance-normalized image "bands"



112.3m to INF, scale: 1.0



50.7m to INF, scale: 1.4



24.2m to INF, scale: 1.9



13.8m to 86.8m, scale: 2.6



9.0m to 34.5m, scale: 3.5



5.8m to 17.6m, scale: 5.0



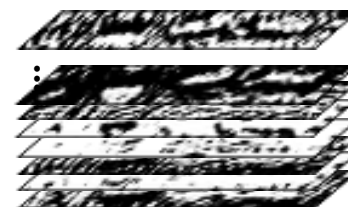
4.1m to 11.3m, scale: 6.7

Convolutional Net Architecture

Y LeCun

100 features per
3x12x25 input window

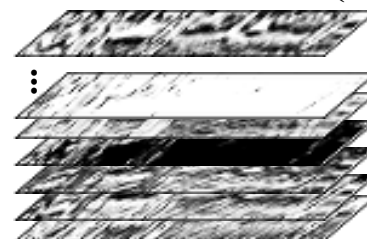
100@25x121



CONVOLUTIONS (6x5)

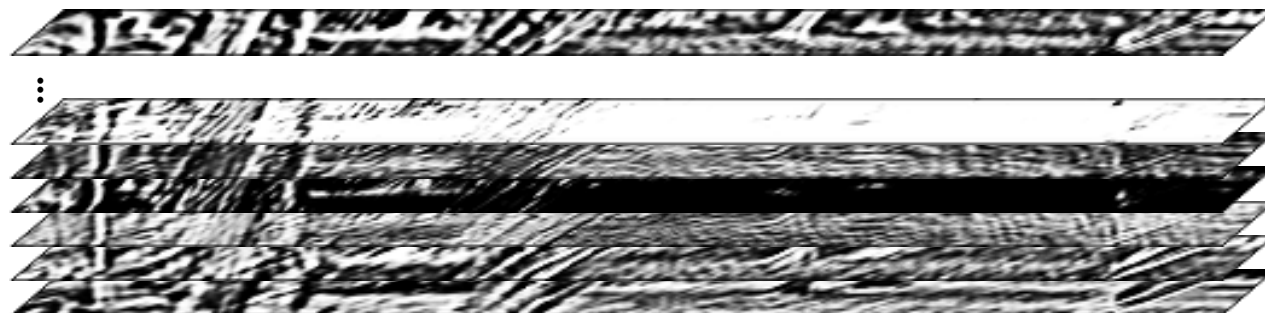
YUV image band
20-36 pixels tall,
36-500 pixels wide

20@30x125



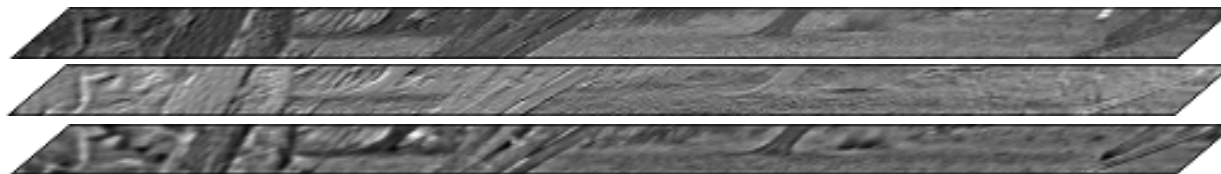
MAX SUBSAMPLING (1x4)

20@30x484



CONVOLUTIONS (7x6)

3@36x484



YUV input

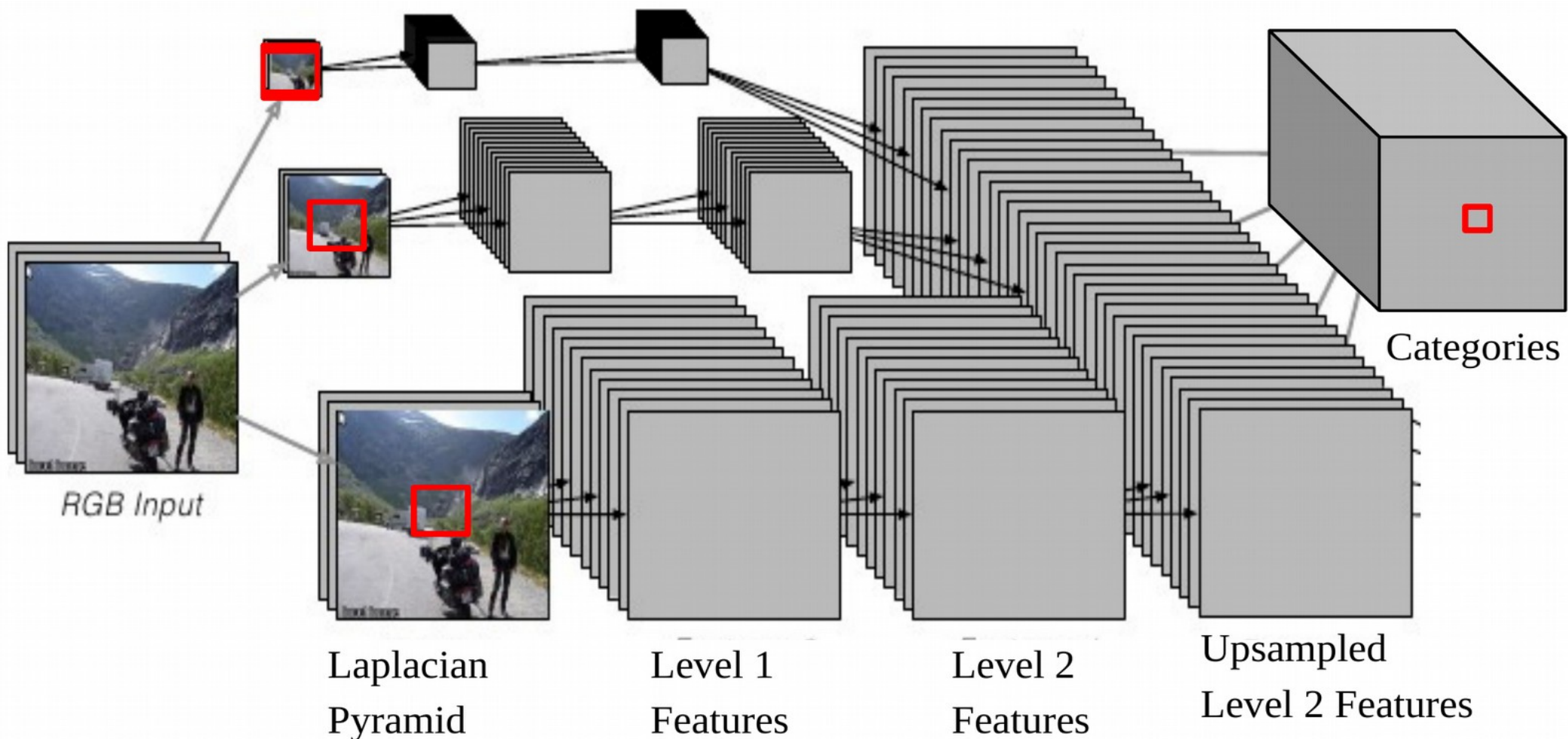


Scene Parsing/Labeling: Multiscale ConvNet Architecture

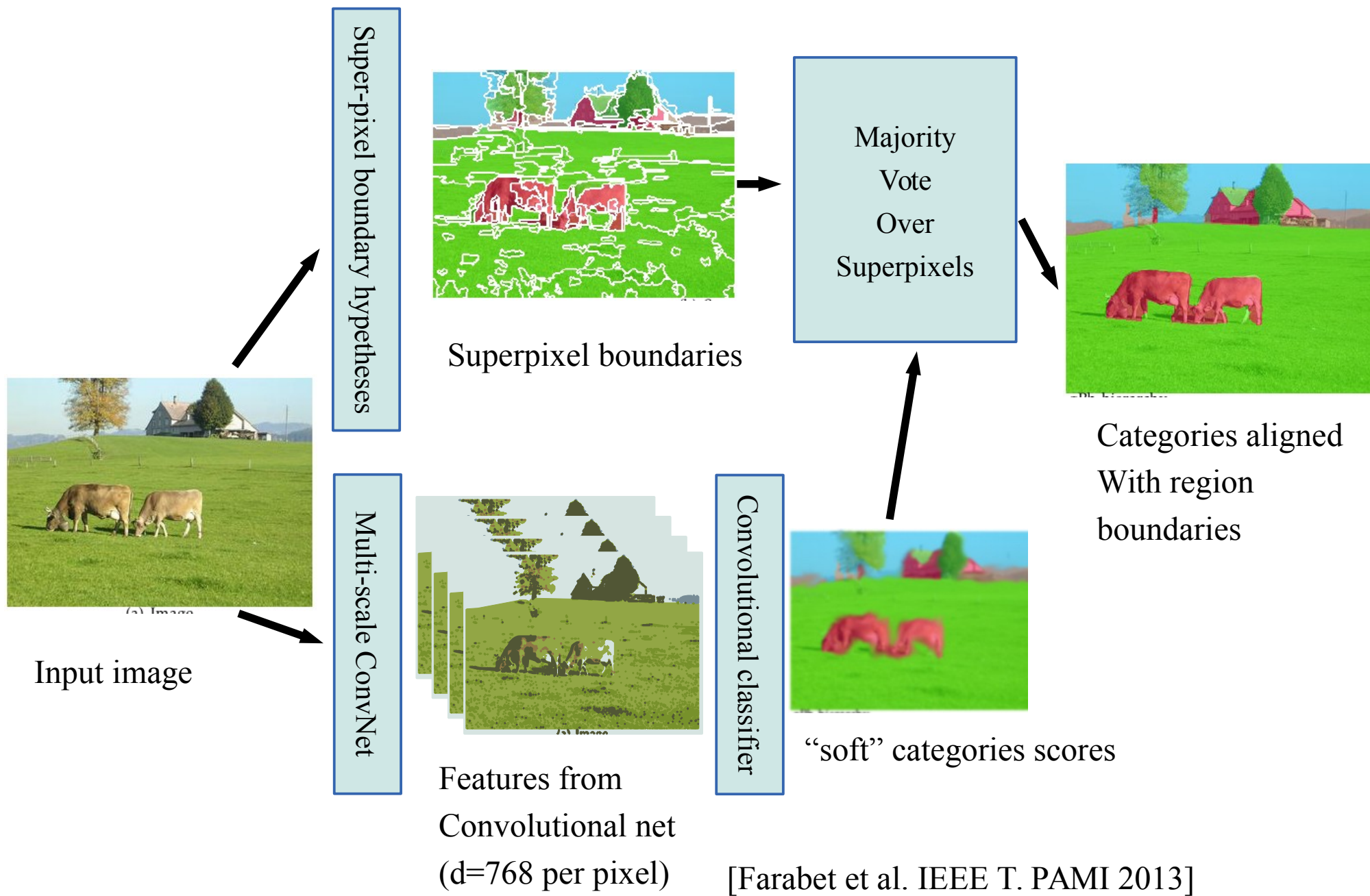
Y LeCun

Each output sees a large input context:

- ▶ **46x46** window at full rez; **92x92** at $\frac{1}{2}$ rez; **184x184** at $\frac{1}{4}$ rez
- ▶ [7x7conv]->[2x2pool]->[7x7conv]->[2x2pool]->[7x7conv]->
- ▶ Trained supervised on fully-labeled images

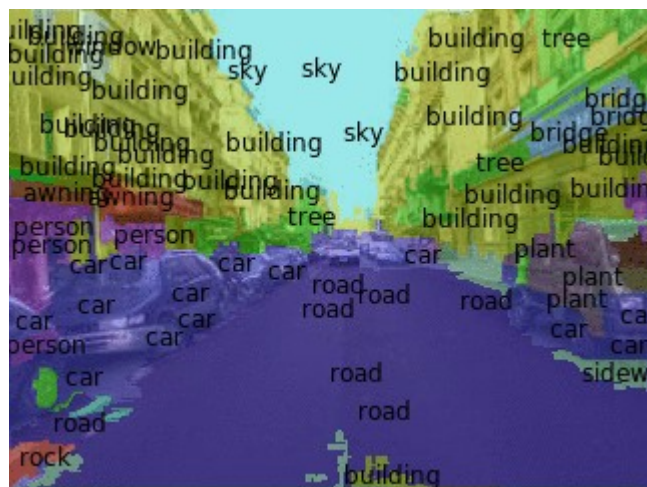


Method 1: majority over super-pixel regions



Scene Parsing/Labeling

Y LeCun



[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling on RGB+Depth Images

Y LeCun

Legend for scene parsing labels:

red	wall	blue	books	purple	chair	teal	furniture	green	sofa	red	object	brown	TV
orange	bed	cyan	ceiling	dark blue	floor	yellow	pict./deco	orange	table	dark red	window	gray	uknw



Ground truths



Our results

[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]

Scene Parsing/Labeling: Performance

Y LeCun

■ Stanford Background Dataset [Gould 1009]: 8 categories

	Pixel Acc.	Class Acc.	CT (sec.)
Gould <i>et al.</i> 2009 [14]	76.4%	-	10 to 600s
Munoz <i>et al.</i> 2010 [32]	76.9%	66.2%	12s
Tighe <i>et al.</i> 2010 [46]	77.5%	-	10 to 300s
Socher <i>et al.</i> 2011 [45]	78.1%	-	?
Kumar <i>et al.</i> 2010 [22]	79.4%	-	< 600s
Lempitzky <i>et al.</i> 2011 [28]	81.9%	72.4%	> 60s
singlescale convnet	66.0 %	56.5 %	0.35s
multiscale convnet	78.8 %	72.4%	0.6s
multiscale net + superpixels	80.4%	74.56%	0.7s
multiscale net + gPb + cover	80.4%	75.24%	61s
multiscale net + CRF on gPb	81.4%	76.0%	60.5s

[Rejected from CVPR 2012]

[Farabet et al. ICML 2012][Farabet et al. IEEE T. PAMI 2013]

Scene Parsing/Labeling: Performance

Y LeCun

	Pixel Acc.	Class Acc.
Liu <i>et al.</i> 2009 [31]	74.75%	-
Tighe <i>et al.</i> 2010 [44]	76.9%	29.4%
raw multiscale net ¹	67.9%	45.9%
multiscale net + superpixels ¹	71.9%	50.8%
multiscale net + cover ¹	72.3%	50.8%
multiscale net + cover ²	78.5%	29.6%

- SIFT Flow Dataset
- [Liu 2009]:
- 33 categories

- Barcelona dataset
- [Tighe 2010]:
- 170 categories.

	Pixel Acc.	Class Acc.
Tighe <i>et al.</i> 2010 [44]	66.9%	7.6%
raw multiscale net ¹	37.8%	12.1%
multiscale net + superpixels ¹	44.1%	12.4%
multiscale net + cover ¹	46.4%	12.5%
multiscale net + cover ²	67.8%	9.5%

[Farabet et al. IEEE T. PAMI 2012]

Scene Parsing/Labeling

Y LeCun



[Farabet et al. ICML 2012, PAMI 2013]

Scene Parsing/Labeling

Y LeCun



- No post-processing
- Frame-by-frame
- ConvNet runs at 50ms/frame on Virtex-6 FPGA hardware
 - ▶ But communicating the features over ethernet limits system performance

Then., two things happened...

Y LeCun

The ImageNet dataset [Fei-Fei et al. 2012]

- ▶ 1.2 million training samples
- ▶ 1000 categories

Fast Graphical Processing Units (GPU)

- ▶ Capable of 1 trillion operations/second



Matchstick



Sea lion



Flute



Strawberry



Bathing cap



Backpack

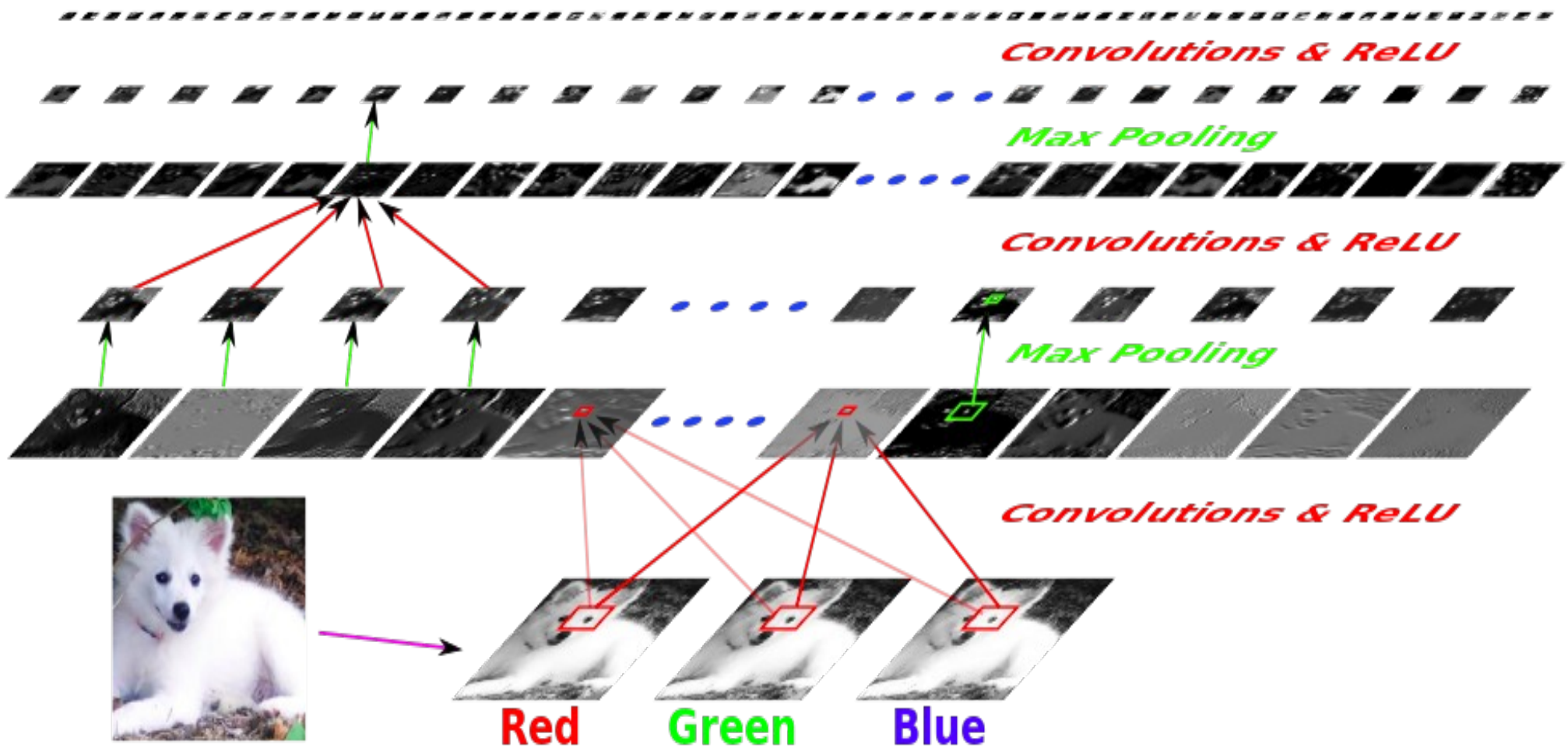


Racket



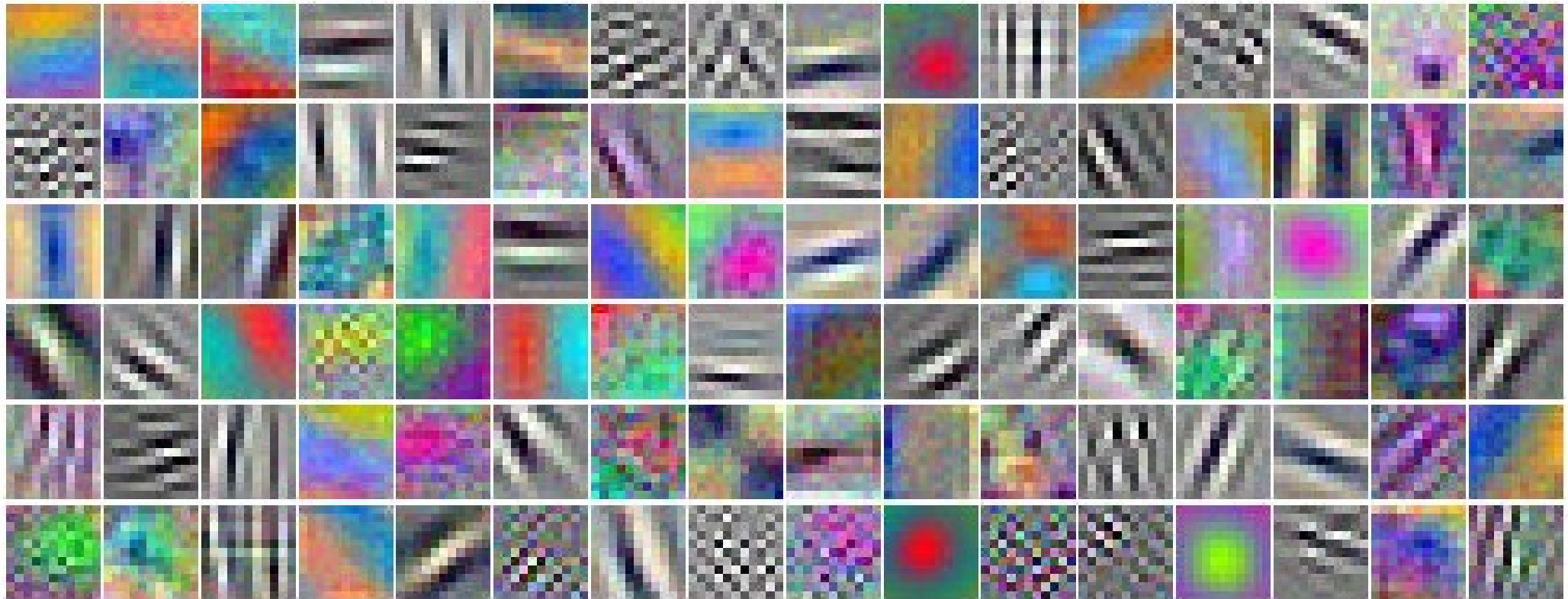
f Very Deep ConvNet for Object Recognition

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic Fox (1.0); Eskimo Dog (0.6); White Wolf (0.4); Siberian Husky (0.4)



Kernels: Layer 1 (11x11)

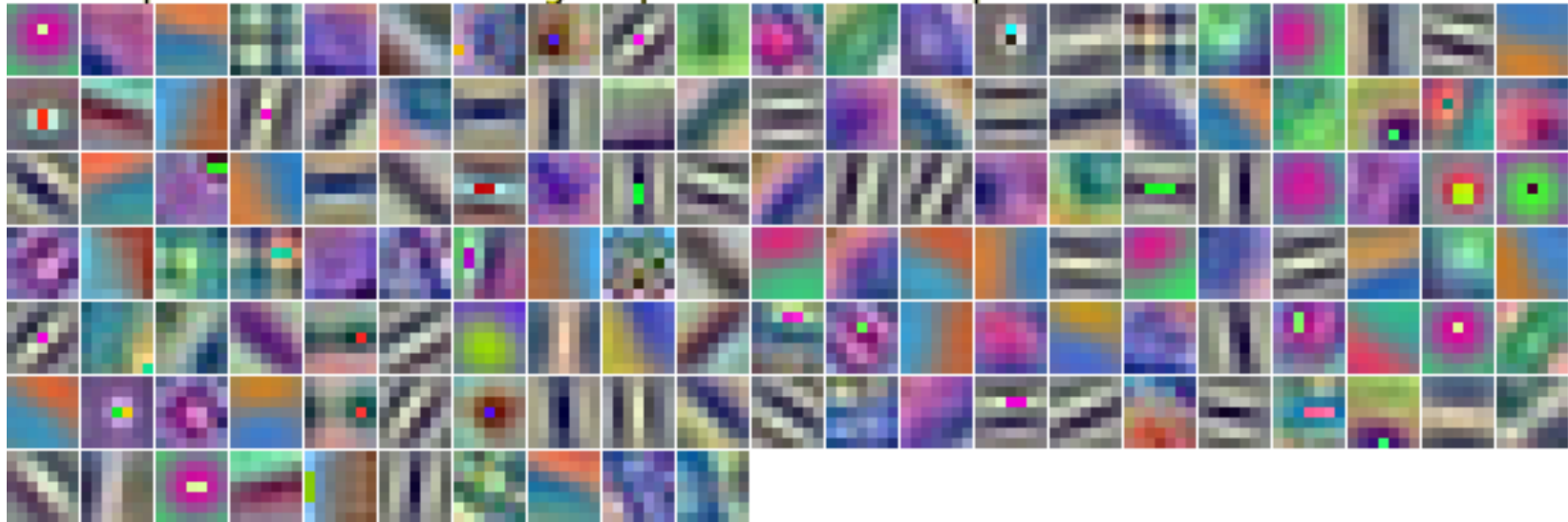
Layer 1: 3x96 kernels, RGB->96 feature maps, 11x11 Kernels, stride 4



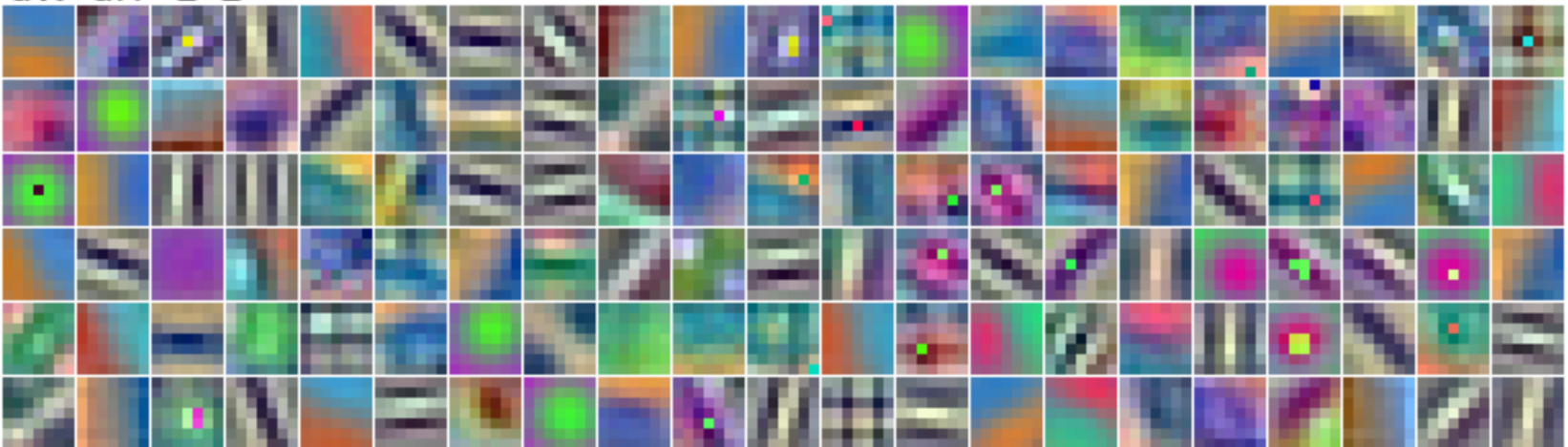
Kernels: Layer 1 (11x11)

Layer 1: 3x512 kernels, 7x7, 2x2 stride.

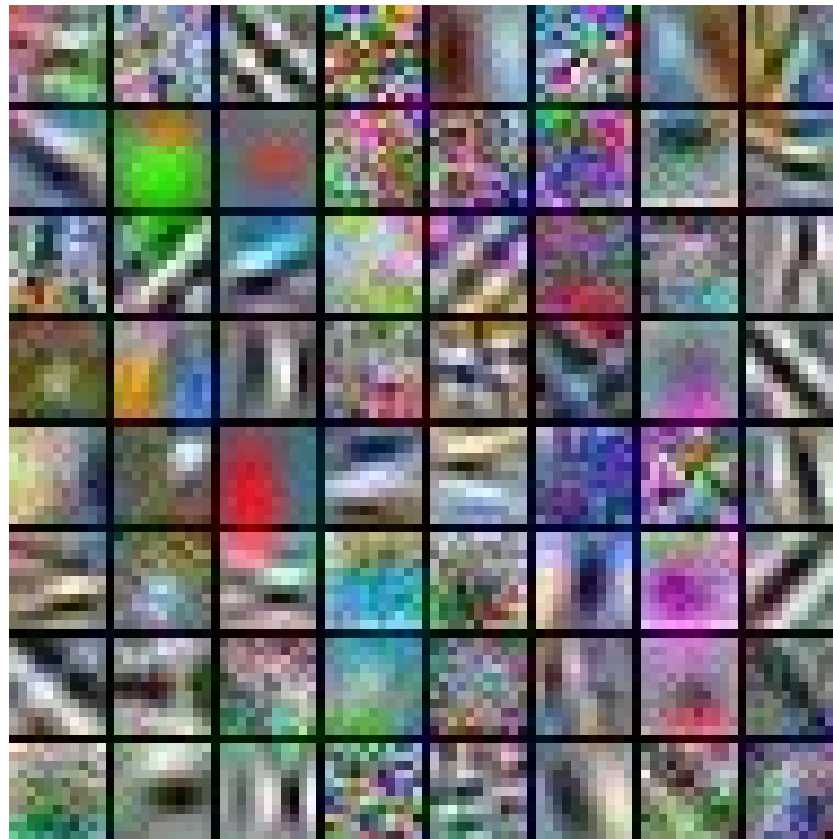
1: nn.SpatialConvolutionRing nInputPlane=3 nOutputPlane=512 kW*kH=7*7 dW:



dW*dH=2*2

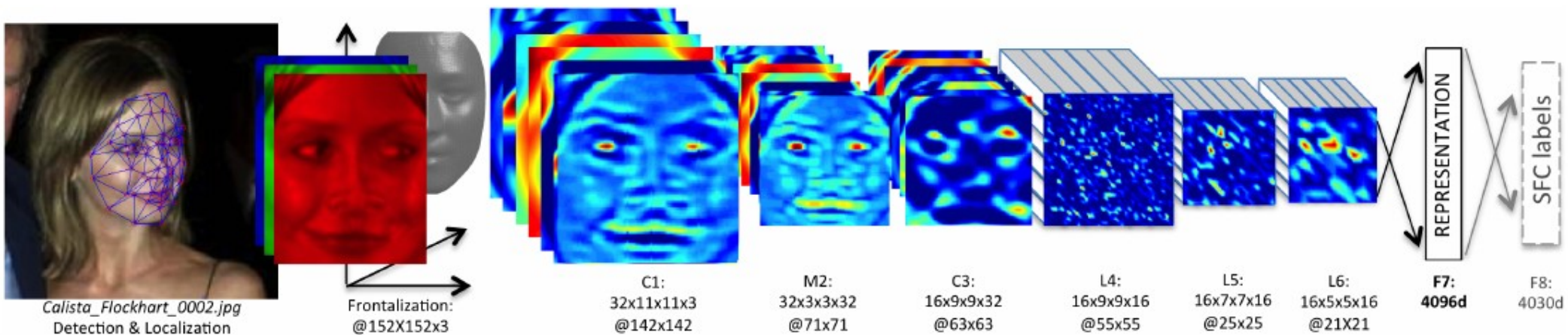
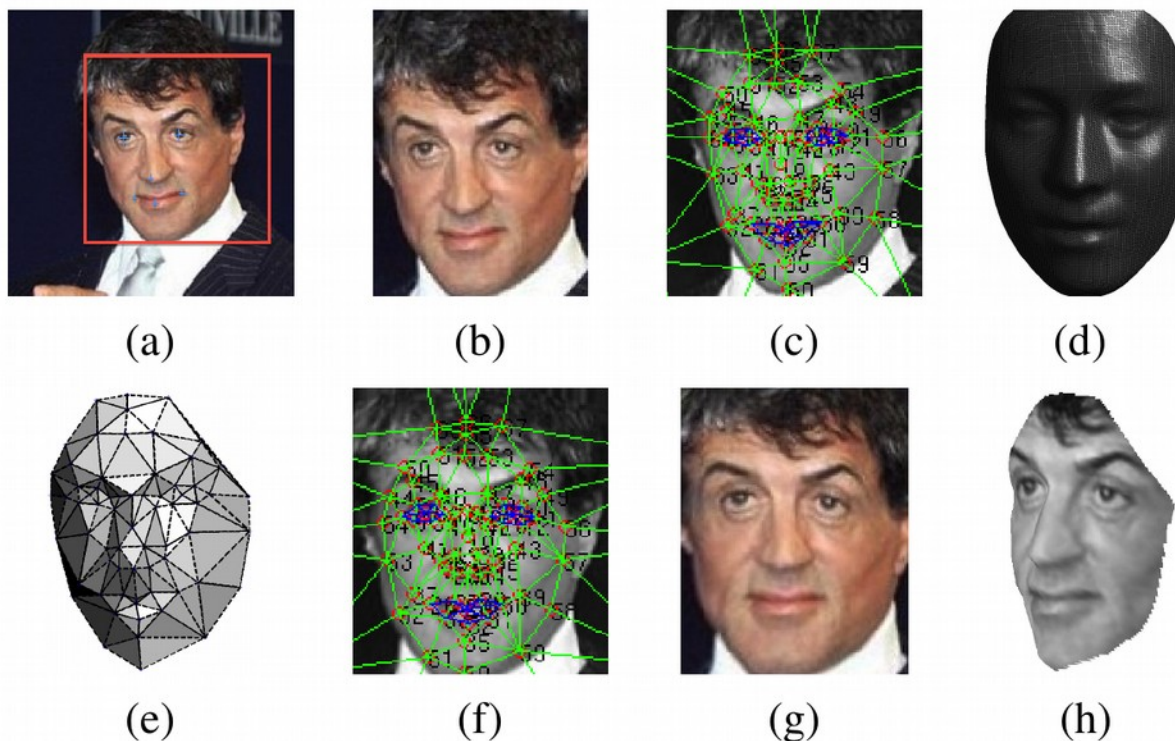


- How the filters in the first layer learn



[Taigman et al. CVPR 2014]

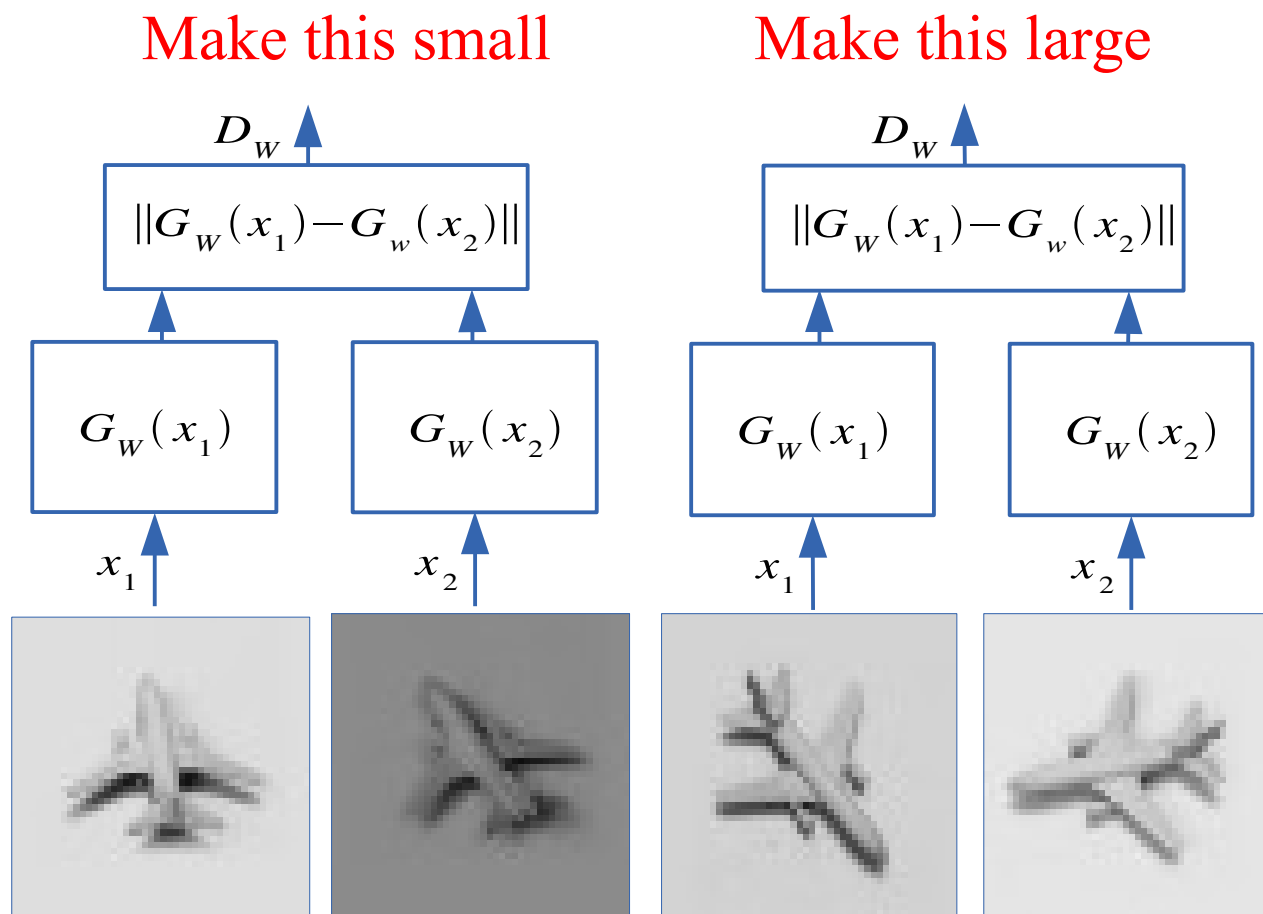
- ▶ Alignment
- ▶ ConvNet
- ▶ Metric Learning



Siamese Architecture and loss function

Loss function:

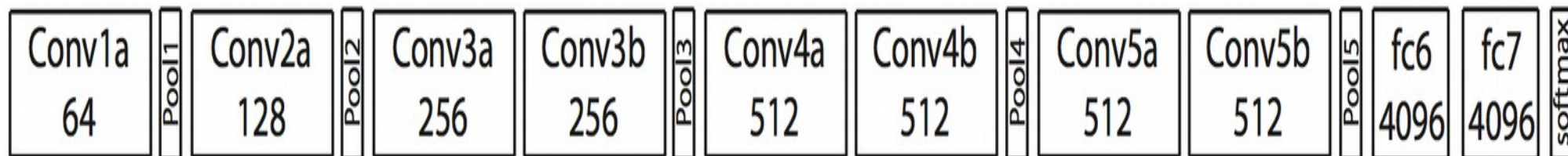
- Outputs corresponding to input samples that are neighbors in the neighborhood graph should be nearby
- Outputs for input samples that are not neighbors should be far away from each other



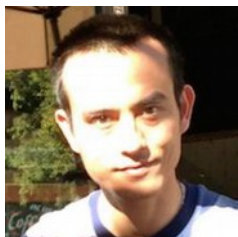
Similar images (neighbors
in the neighborhood graph)

Dissimilar images
(non-neighbors in the
neighborhood graph)

Learning Video Features with C3D



- **C3D Architecture**
 - 8 convolution, 5 pool, 2 fully-connected layers
 - 3x3x3 convolution kernels
 - 2x2x2 pooling kernels
- **Dataset: Sports-1M [Karpathy et al. CVPR'14]**
 - 1.1M videos of 487 different sport categories
 - Train/test splits are provided



Du Tran
(1,2)



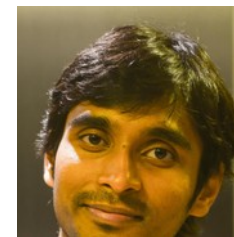
Lubomir Bourdev
(2)



Rob Fergus
(2,3)



Lorenzo Torresani
(1)



Manohar Paluri
(2)

Sport Classification Results



Method	Number of Nets	Clip hit@1	Video hit@1	Video hit@5
Deep Video's Single-Frame + Multires [19]	3 nets	42.4	60.0	78.5
Deep Video's Slow Fusion [19]	1 net	41.9	60.9	80.2
C3D (trained from scratch)	1 net	44.9	60.0	84.4
C3D (fine-tuned from I380K pre-trained model)	1 net	46.1	61.1	85.2

- Using a spatio-temporal ConvNet

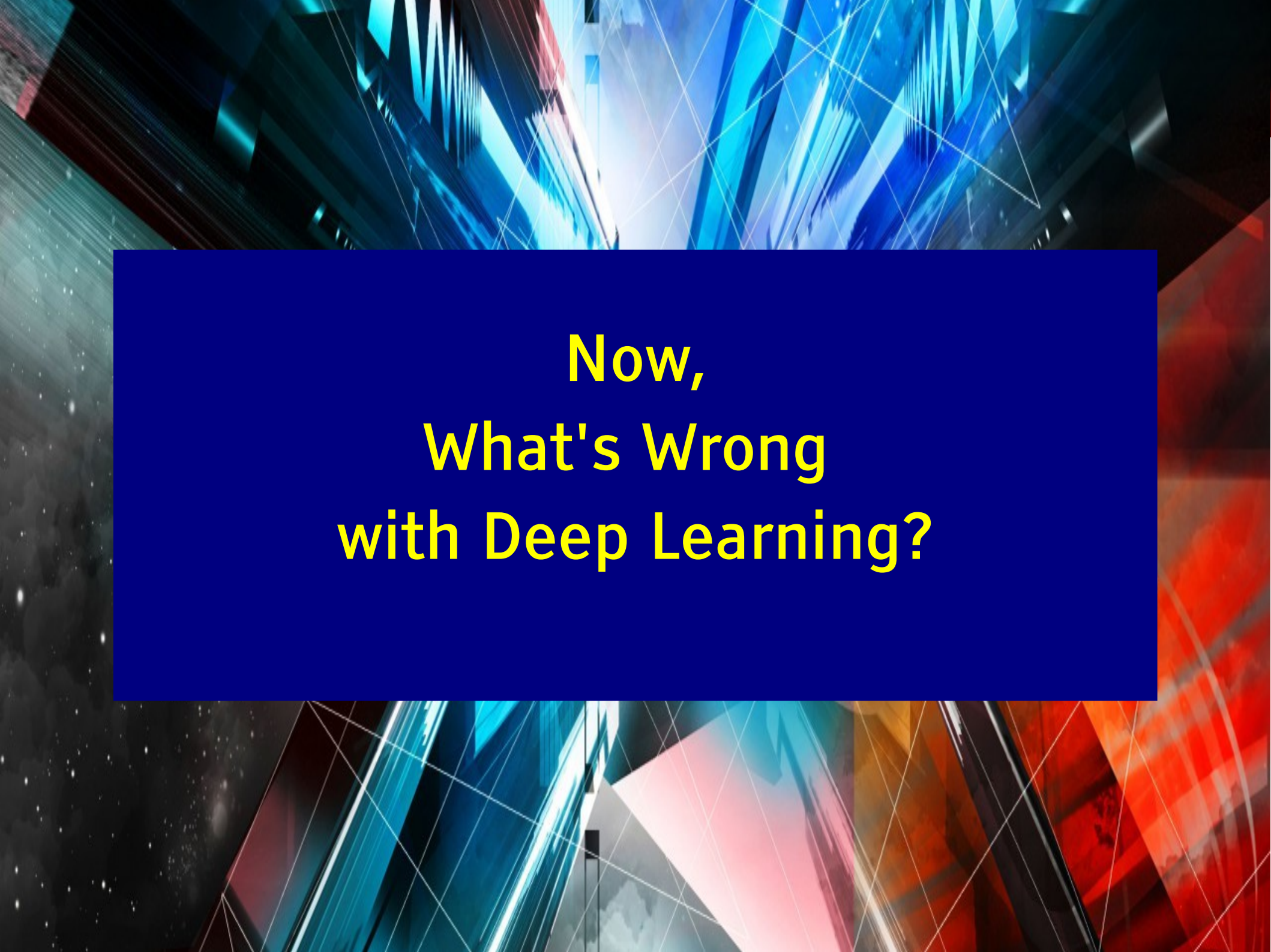


- Using a spatio-temporal ConvNet



- Spatio-temporal ConvNet





**Now,
What's Wrong
with Deep Learning?**



Missing Some Theory

Why are ConvNets a good architecture?

- Scattering transform
- Mark Tygert's "complex ConvNet"

How many layers do we really need?

- Really?

How many effective free parameters are there in a large ConvNet

- The weights seem to be awfully redundant

What about Local Minima?

- Turns out almost all the local minima are equivalent
- Local minima are degenerate (very flat in most directions)
- Random matrix / spin glass theory comes to the rescue
- [Choromanska, Henaff, Mathieu, Ben Arous, LeCun AI-stats 2015]

Deep Nets with ReLUs: Objective Function is Piecewise Polynomial

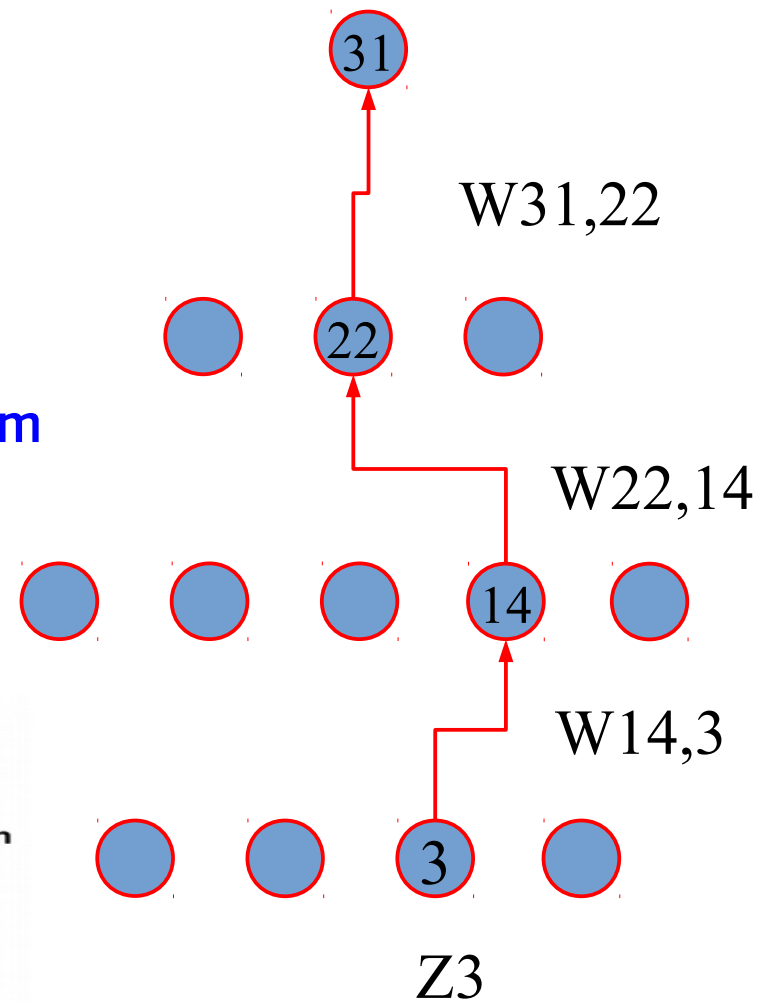
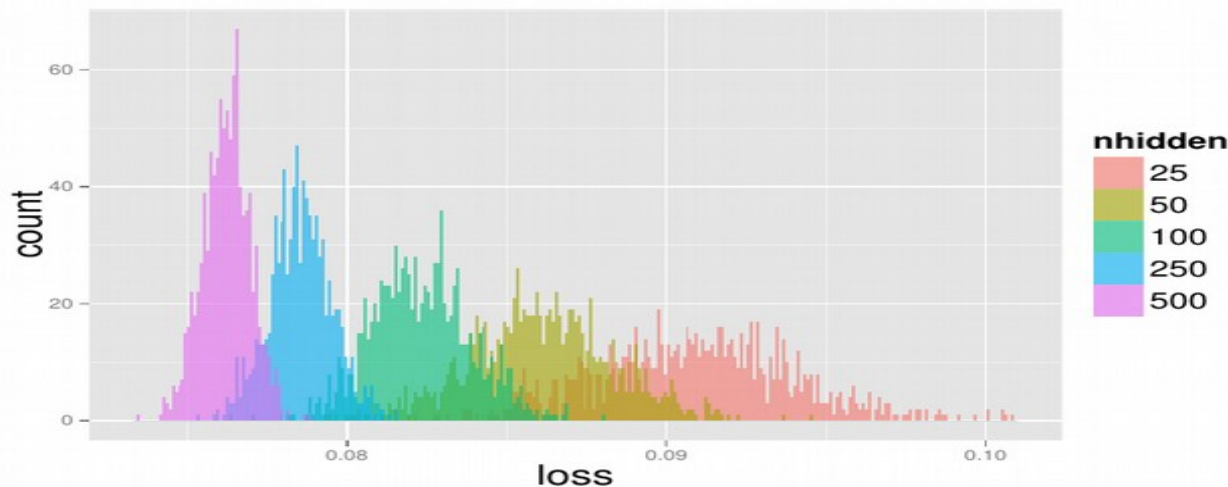
If we use a hinge loss, delta now depends on label Y_k :

$$L(W) = \sum_P C_p(X, Y, W) \left(\prod_{(ij) \in P} W_{ij} \right)$$

Piecewise polynomial in W with random coefficients

A lot is known about the distribution of critical points of polynomials on the sphere with random (Gaussian) coefficients [Ben Arous et al.]

- ▶ High-order spherical spin glasses
- ▶ Random matrix theory





Missing: Reasoning

Reasoning as Energy Minimization (structured prediction++)

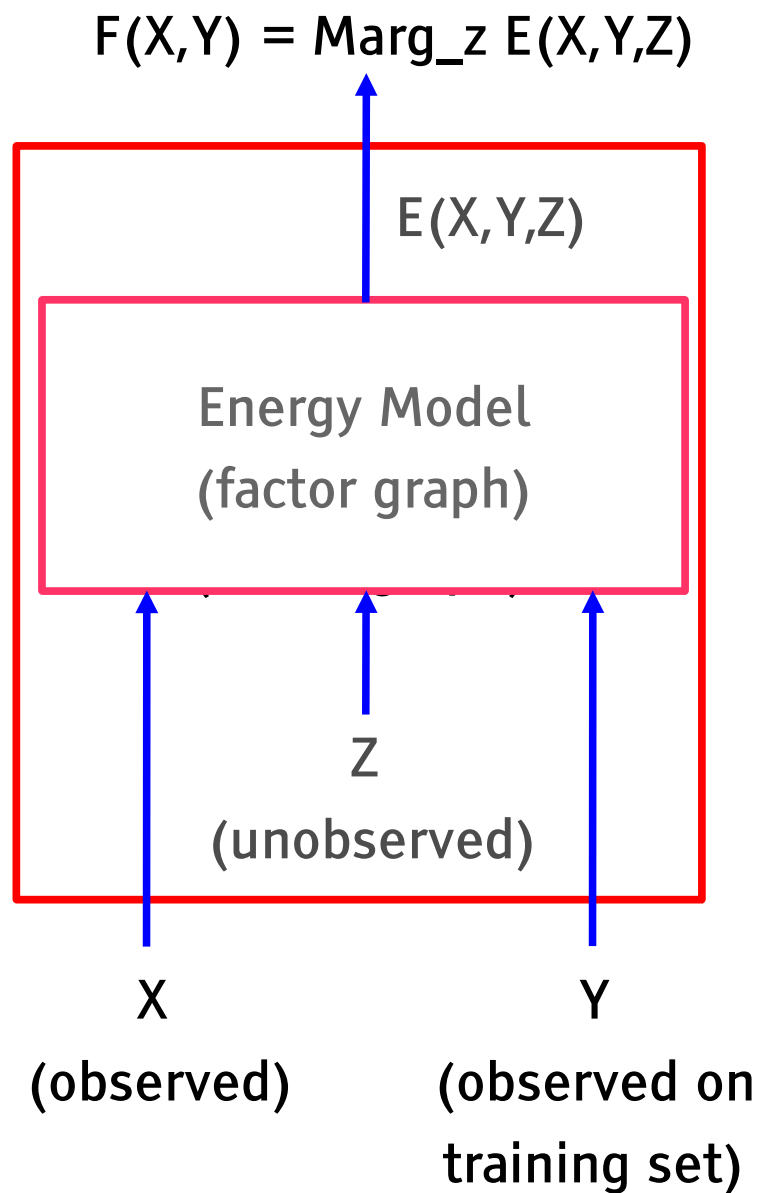
Y LeCun

Deep Learning systems can be assembled into energy models AKA factor graphs

- ▶ Energy function is a sum of factors
- ▶ Factors can embed whole deep learning systems
- ▶ X: observed variables (inputs)
- ▶ Z: never observed (latent variables)
- ▶ Y: observed on training set (output variables)

Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X

- ▶ $F(X,Y) = \text{MIN}_z E(X,Y,Z)$
- ▶ $F(X,Y) = -\log \text{SUM}_z \exp[-E(X,Y,Z)]$



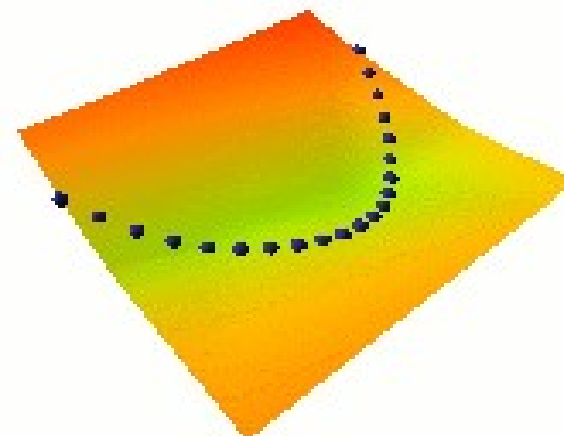
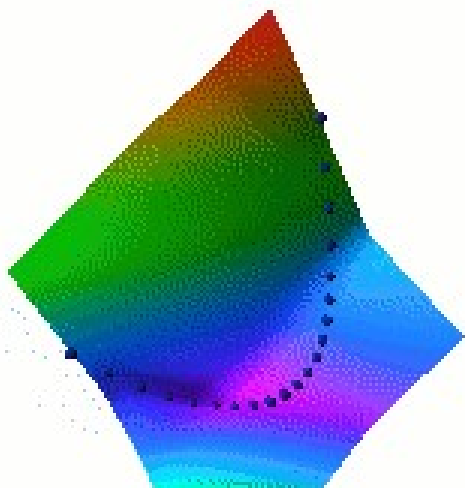
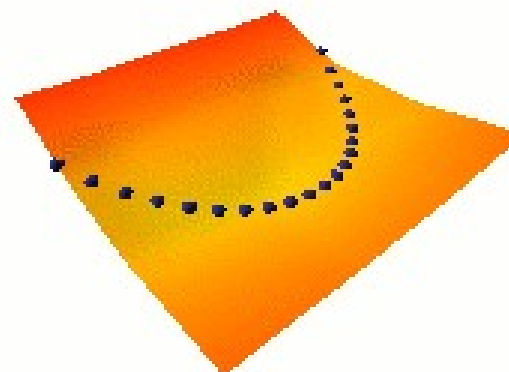
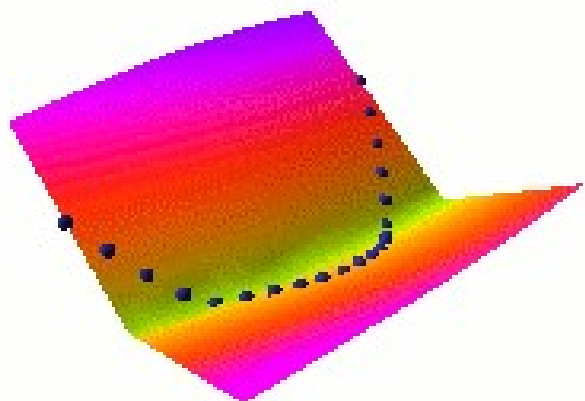
Energy-Based Learning [LeCun et al. 2006]

Y LeCun

Push down on the energy of desired outputs

Push up on everything else

[LeCun et al 2006] "A tutorial on energy-based learning"





Stick a CRF on top of a ConvNet

Pose Estimation and Attribute Recovery with ConvNets

Y LeCun

Pose-Aligned Network for Deep Attribute Modeling

[Zhang et al. CVPR 2014] (Facebook AI Research)



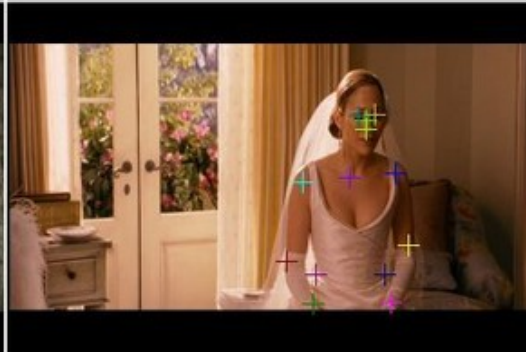
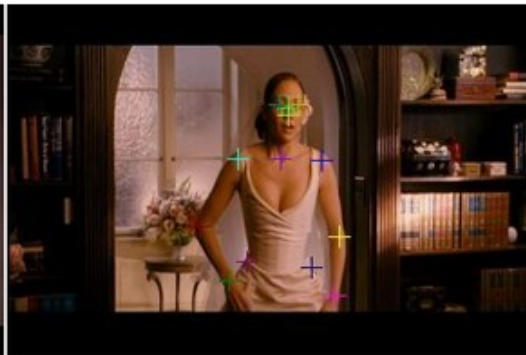
(a) Highest scoring results for people wearing glasses.



(b) Highest scoring results for people wearing a hat.

Real-time hand pose recovery

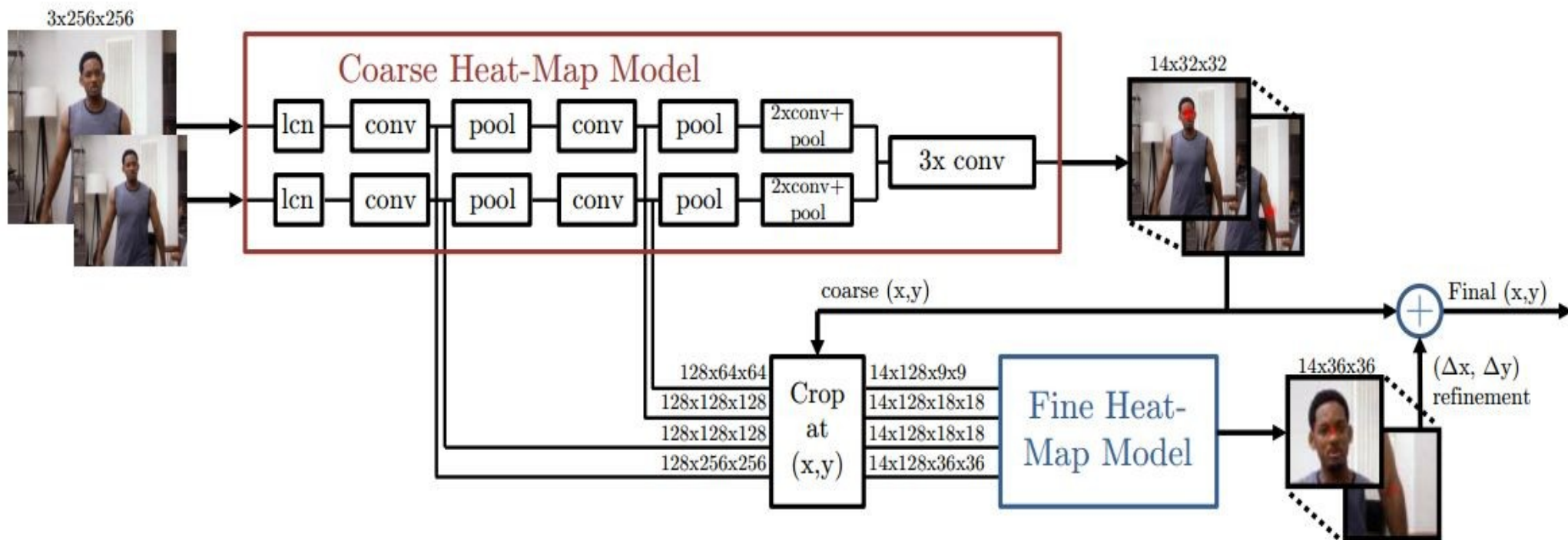
[Tompson et al. Trans. on Graphics 14]



Body pose estimation [Tompson et al. ICLR, 2014]

Person Detection and Pose Estimation

[Tompson, Goroshin, Jain, LeCun, Bregler CVPR 2015]



Person Detection and Pose Estimation

Y LeCun

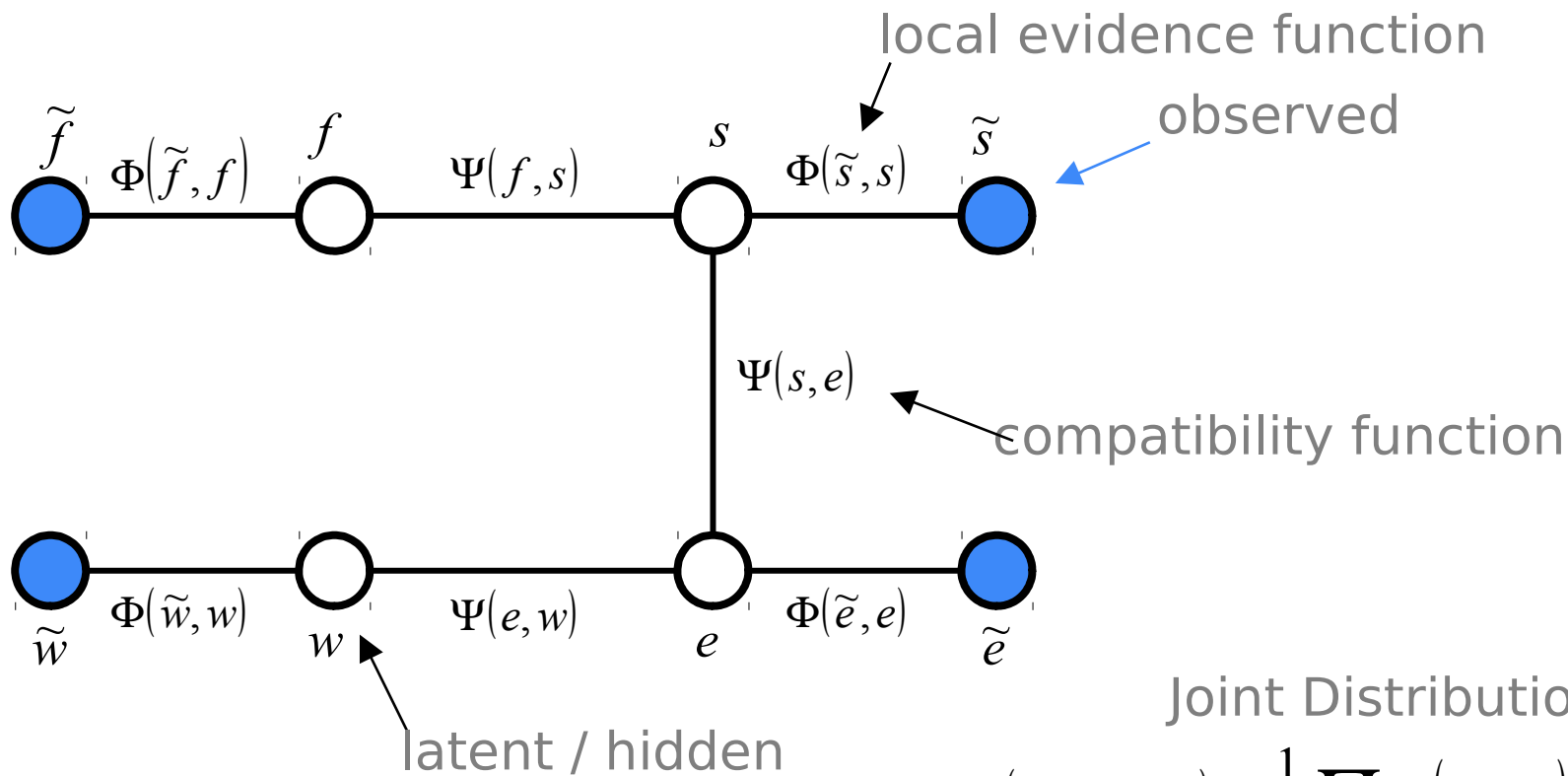
Tompson, Goroshin, Jain, LeCun, Bregler arXiv:1411.4280 (2014)



SPATIAL MODEL

Start with a tree graphical model

MRF over spatial locations



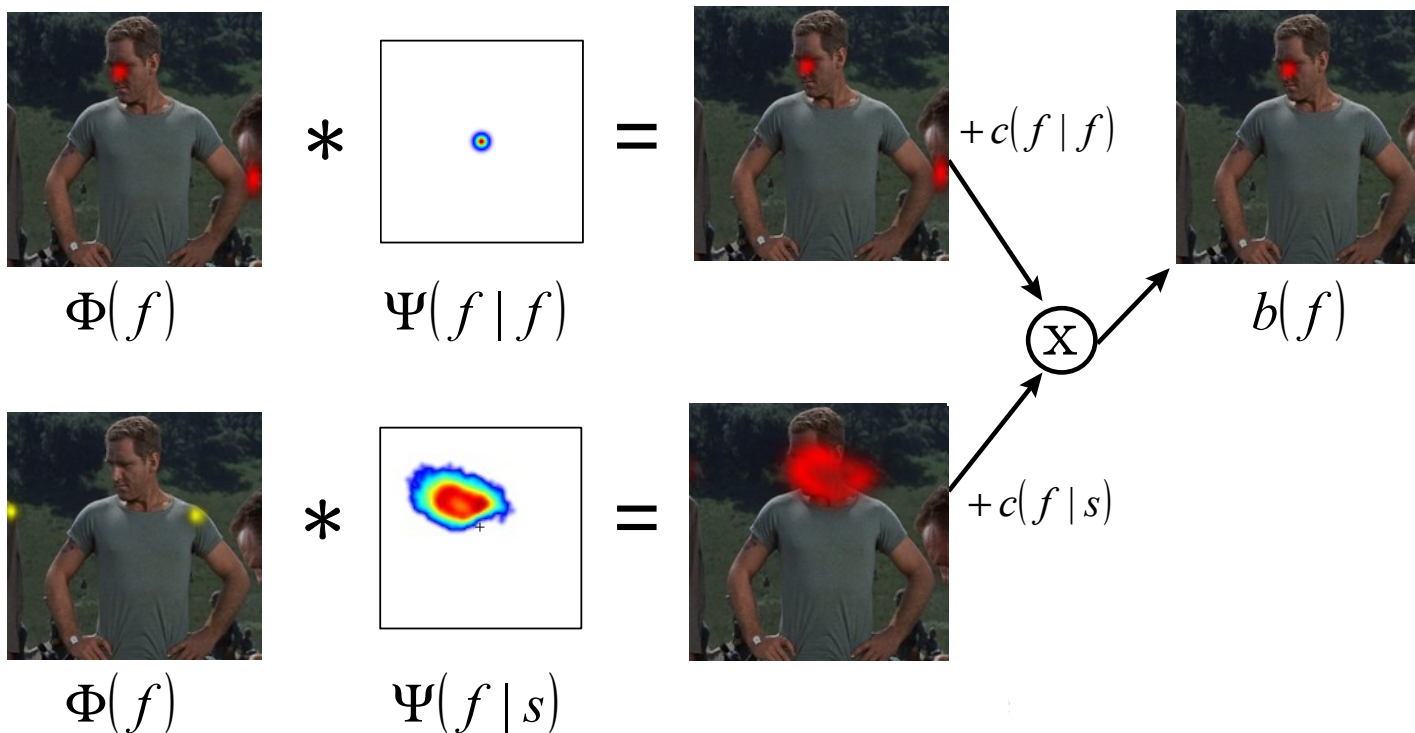
Joint Distribution:

$$P(f, s, e, w) = \frac{1}{Z} \prod_{i,j} \Psi(x_i, x_j) \prod_i \Phi(x_i, \tilde{x}_i)$$

Start with a tree graphical model

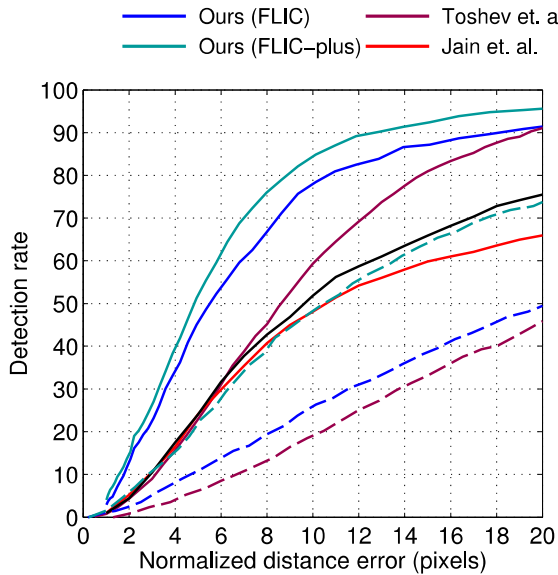
... And approximate it

$$b(f) = \Phi(f) \prod_i (\Phi(x_i) * \Psi(f | x_i) + c(f | x_i))$$

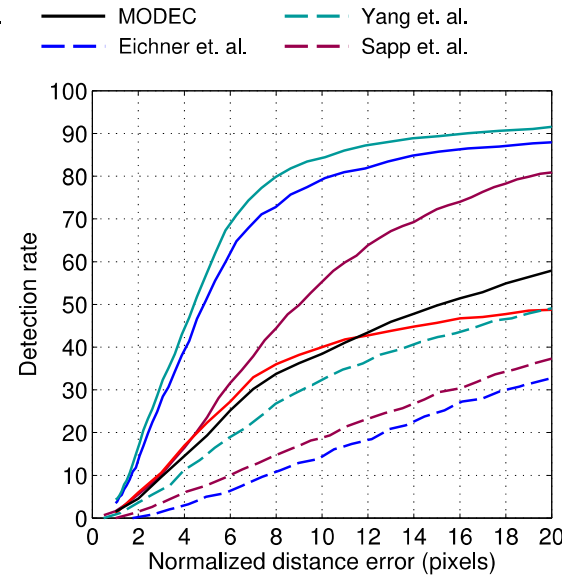


SPATIAL MODEL: RESULTS

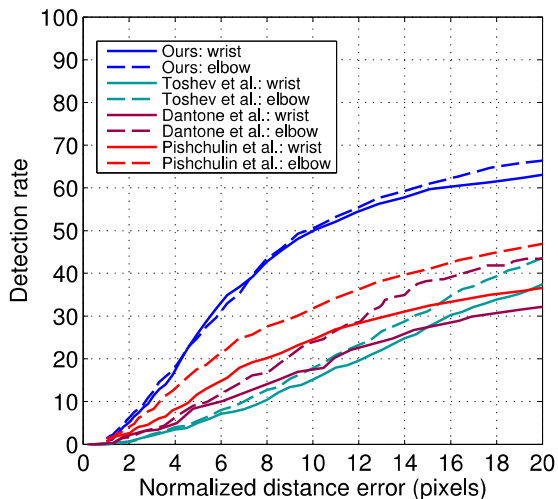
FLIC⁽¹⁾
Elbow



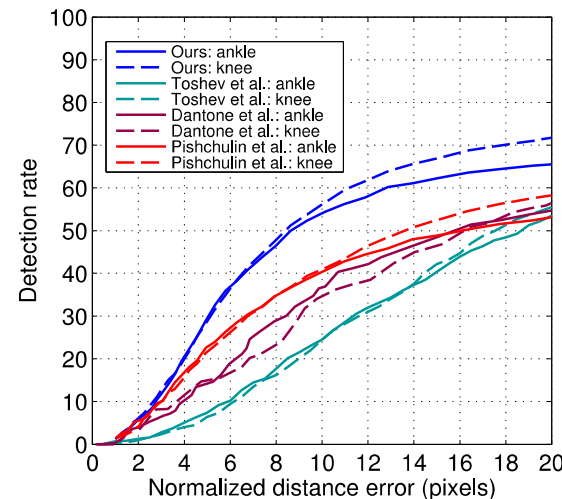
FLIC⁽¹⁾
Wrist



LSP⁽²⁾
Arms



LSP⁽¹⁾
Legs



(1) B. Sapp and B. Taskar. MODEC: Multimodel decomposition models for human pose estimation. CVPR'13

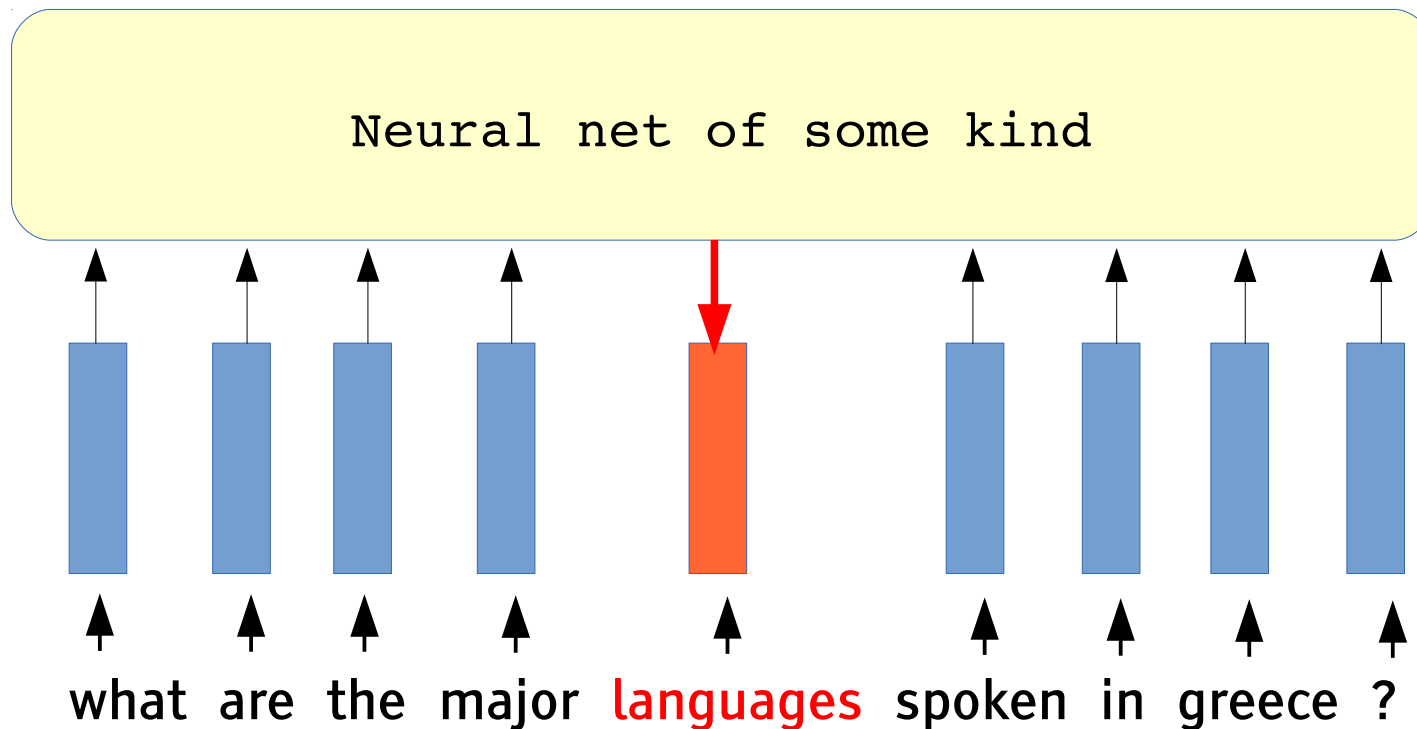
(2) S. Johnson and M. Everingham. Learning Effective Human Pose Estimation for Inaccurate Annotation. CVPR'11



Missing: Memory

Word Embedding in continuous vector spaces

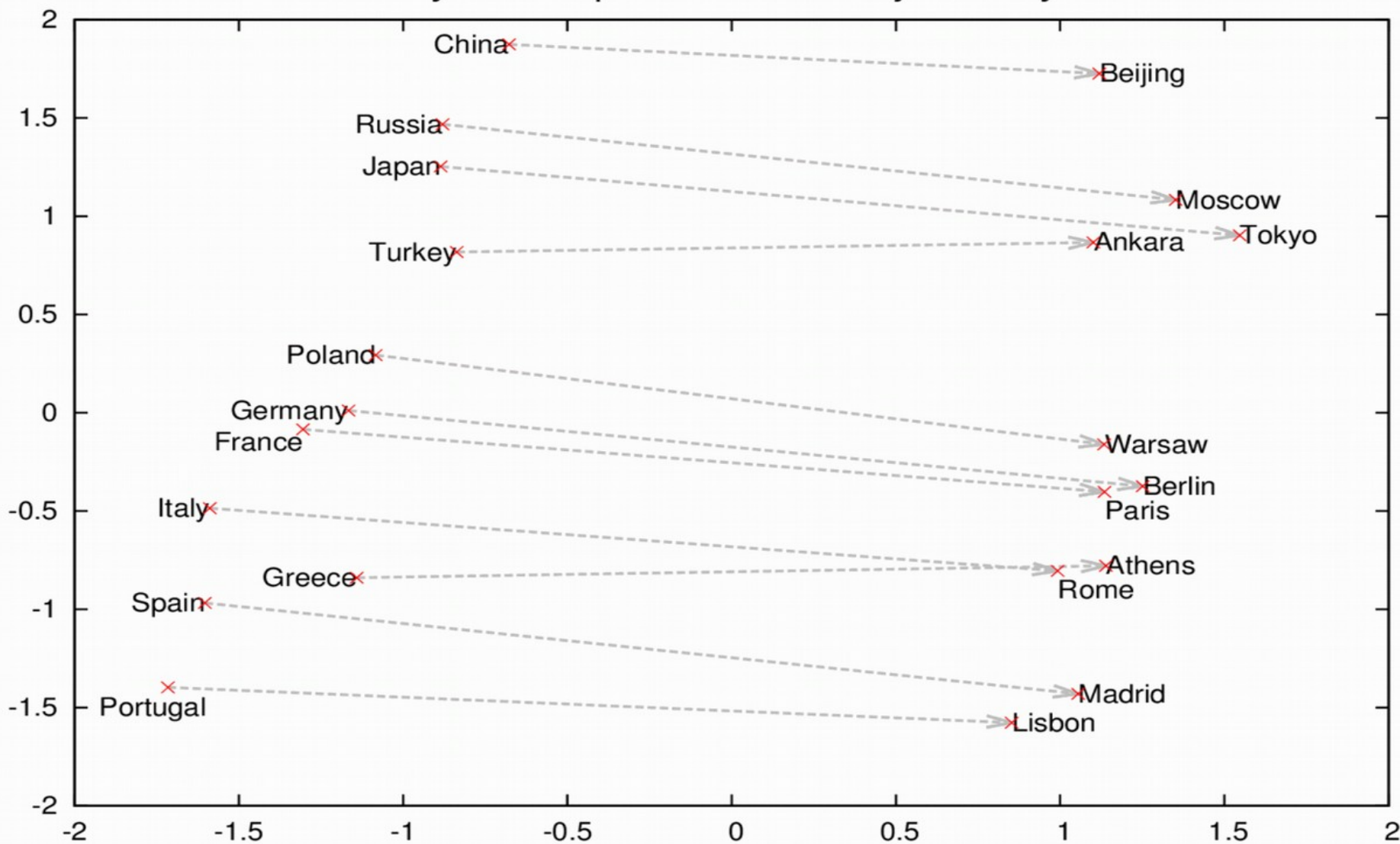
- ▶ [Bengio 2003][Collobert & Weston 2010]
- ▶ Word2Vec [Mikolov 2011]
- ▶ Predict a word from previous words and/or following words



Compositional Semantic Property

Beijing – China + France = Paris

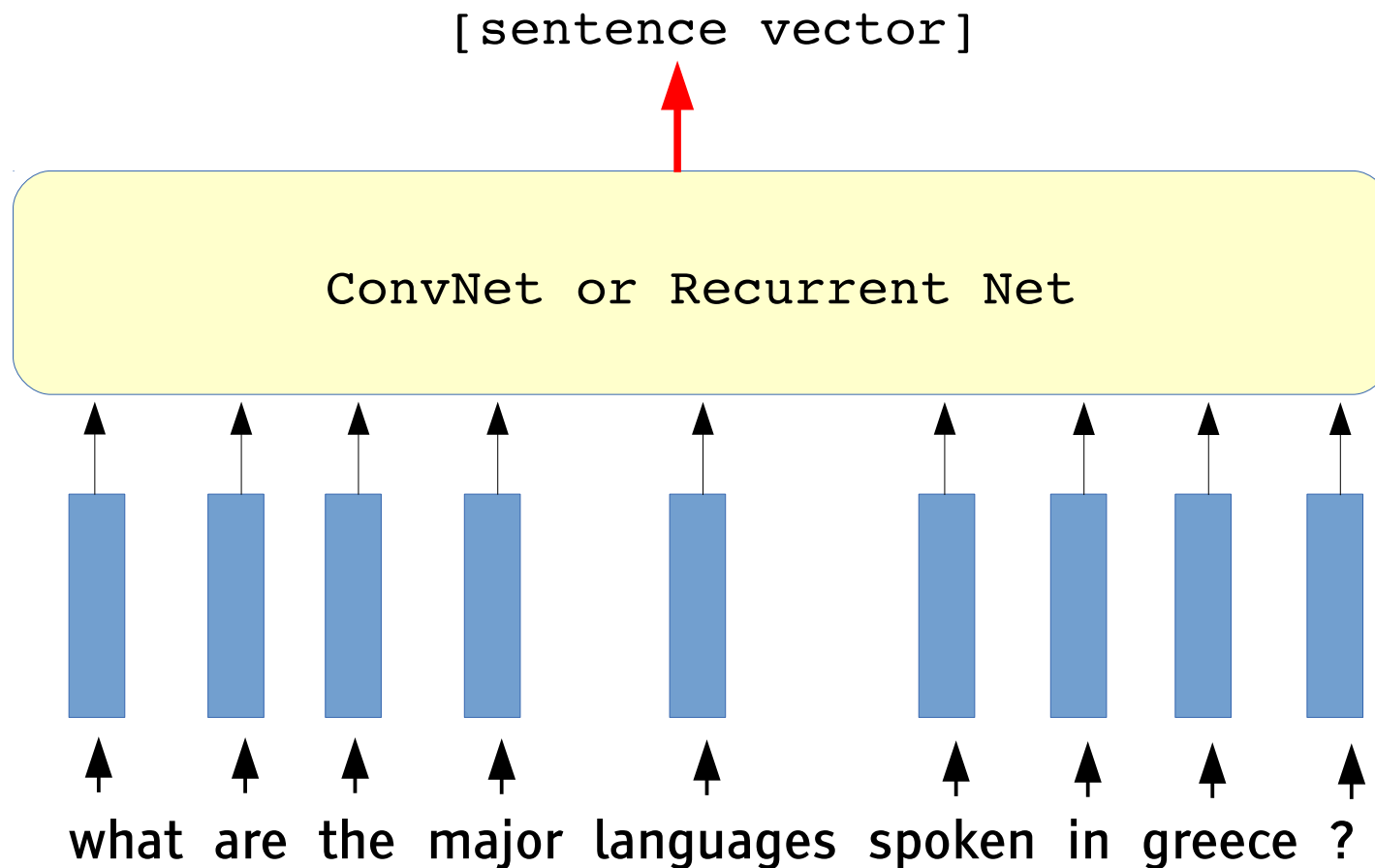
Country and Capital Vectors Projected by PCA



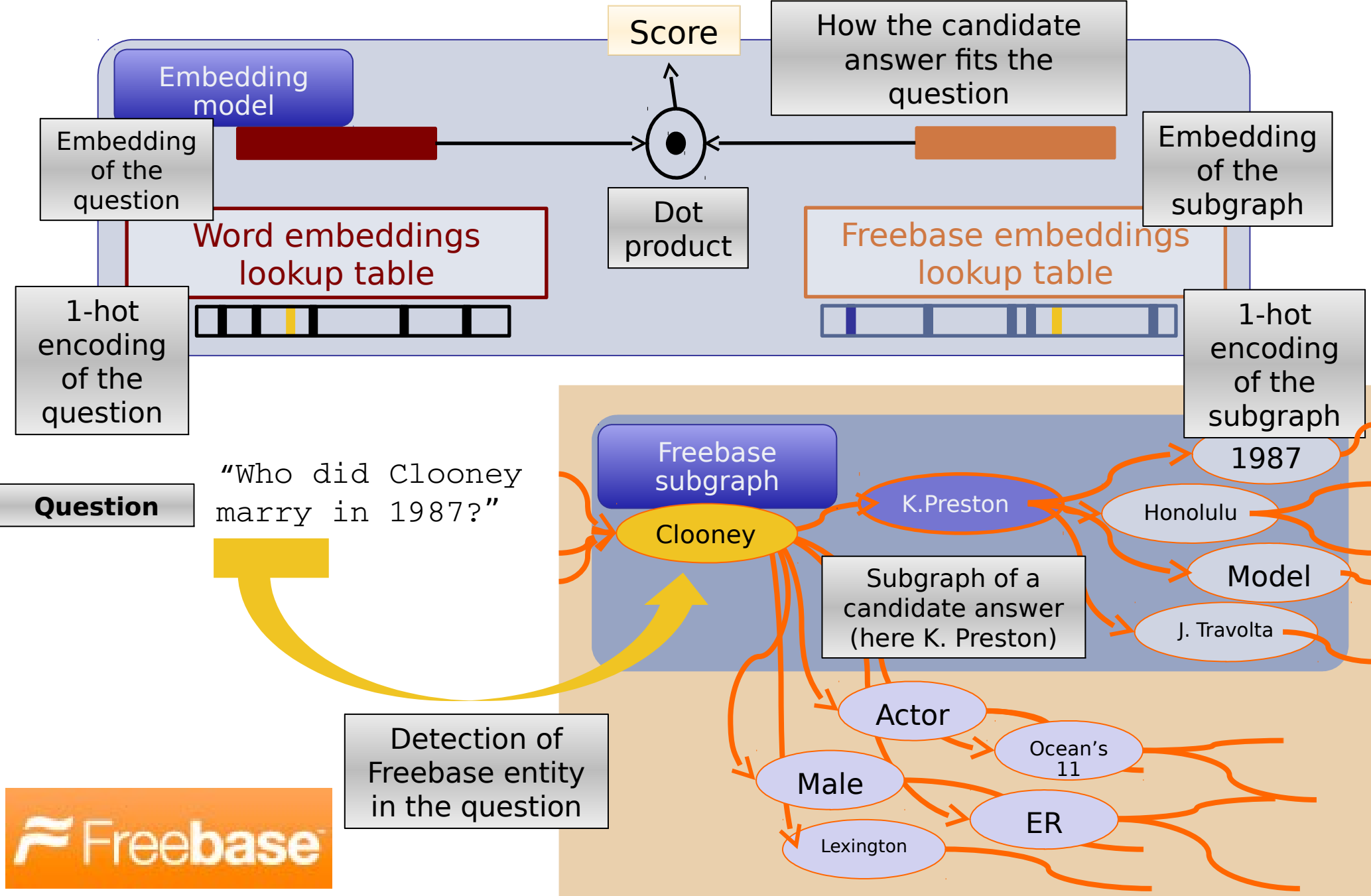
Embedding Text (with convolutional or recurrent nets)

Embedding sentences into vector spaces

- ▶ Using a convolutional net or a recurrent net.



f Question-Answering System



what are bigos?

["stew"] ["stew"]

what are dallas cowboys colors?

["navy_blue", "royal_blue", "blue", "white", "silver"] ["blue", "navy_blue", "white", "royal_blue", "silver"]

how is egyptian money called?

["egyptian_pound"] ["egyptian_pound"]

what are fun things to do in sacramento ca?

["sacramento_zoo"] ["raging_waters_sacramento", "sutter_s_fort", "b_street_theatre", "sacramento_zoo", "california_state_capitol_museum",]

how are john terry's children called?

["georgie_john_terry", "summer_rose_terry"] ["georgie_john_terry", "summer_rose_terry"]

what are the major languages spoken in greece?

["greek_language", "albanian_language"] ["greek_language", "albanian_language"]

what was laura ingalls wilder famous for?

["writer", "author"] ["writer", "journalist", "teacher", "author"]

f NLP: Question-Answering System

who plays sheldon cooper mother on the big bang theory?

["jim_parsons"] ["jim_parsons"]

who does peyton manning play football for?

["denver_broncos"] ["indianapolis_colts", "denver_broncos"]

who did vladimir lenin marry?

["nadezhda_krupskaya"] ["nadezhda_krupskaya"]

where was teddy roosevelt's house?

["new_york_city"] ["manhattan"]

who developed the tcp ip reference model?

["vint_cerf", "robert_e._kahn"] ["computer_scientist", "engineer"]

f Representing the world with “thought vectors”

■ Every object, concept or “thought” can be represented by a vector

- ▶ $[-0.2, 0.3, -4.2, 5.1, \dots]$ represent the concept “cat”
- ▶ $[-0.2, 0.4, -4.0, 5.1, \dots]$ represent the concept “dog”
- ▶ The vectors are similar because cats and dogs have many properties in common

■ Reasoning consists in manipulating thought vectors

- ▶ Comparing vectors for question answering, information retrieval, content filtering
- ▶ Combining and transforming vectors for reasoning, planning, translating languages

■ Memory stores thought vectors

- ▶ MemNN (Memory Neural Network) is an example

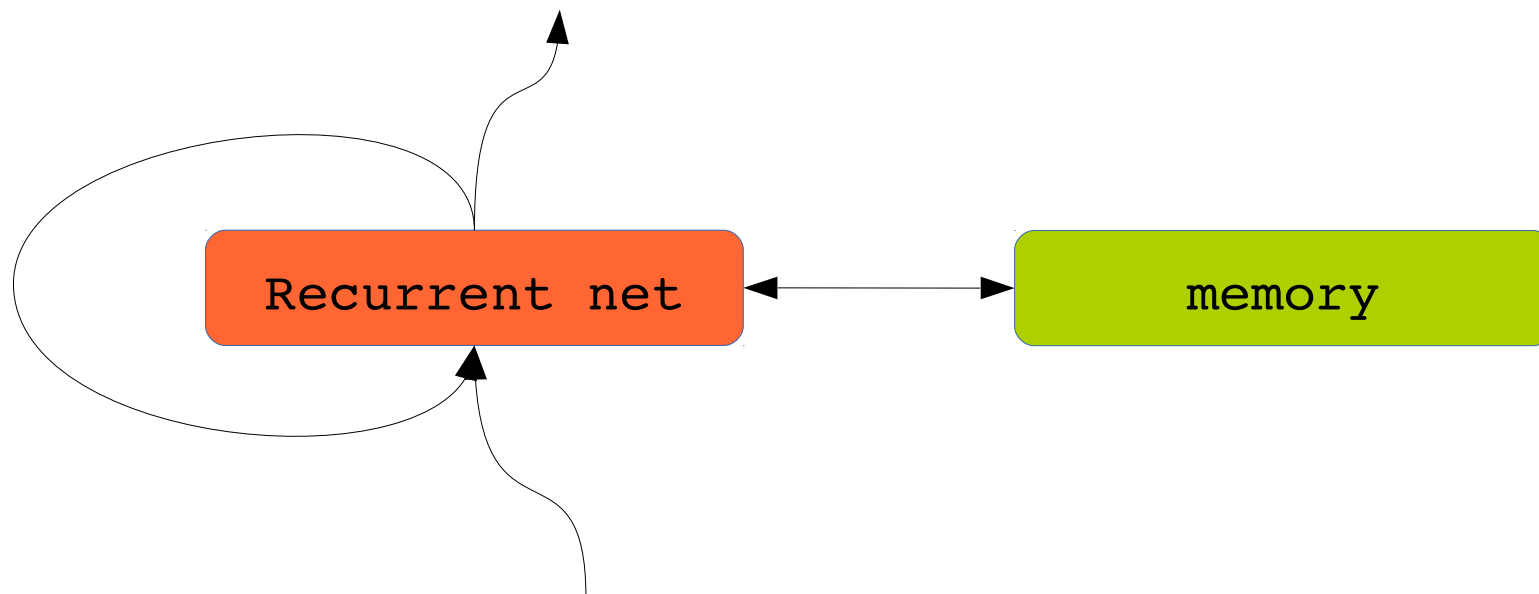
■ At FAIR we want to “embed the world” in thought vectors

We call this **World2vec**

But How can Neural Nets Remember Things?

Y LeCun

- Recurrent networks cannot remember things for very long
 - ▶ The cortex only remember things for 20 seconds
- We need a “hippocampus” (a separate memory module)
 - ▶ LSTM [Hochreiter 1997], registers
 - ▶ **Memory networks** [Weston et 2014] (FAIR), associative memory
 - ▶ NTM [DeepMind 2014], “tape”.



f Memory Network [Weston, Chopra, Bordes 2014]

■ Add a short-term memory to a network

<http://arxiv.org/abs/1410.3916>

- I: (input feature map) – converts the incoming input to the internal feature representation.
- G: (generalization) – updates old memories given the new input.
- O: (output feature map) – produces a new output (in the feature representation space), given the new input and the current memory.
- R: (response) – converts the output into the response format desired. For example, a textual response or an action.

Method	F1
(Fader et al., 2013) 4	0.54
(Bordes et al., 2014) 3	0.73
MemNN	0.71
MemNN (with BoW features)	0.79

Bilbo travelled to the cave.
 Gollum dropped the ring there.
 Bilbo took the ring.
 Bilbo went back to the Shire.
 Bilbo left the ring there.
 Frodo got the ring.
 Frodo journeyed to Mount-Doom.
 Frodo dropped the ring there.
 Sauron died.
 Frodo went back to the Shire.
 Bilbo travelled to the Grey-havens.
 The End.
 Where is the ring? **A: Mount-Doom**
 Where is Bilbo now? **A: Grey-havens**
 Where is Frodo now? **A: Shire**

Results on
 Question Answering
 Task

Fig. 2. An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 4.2 and had never seen many of these words before, e.g. Bilbo, Frodo and Gollum.



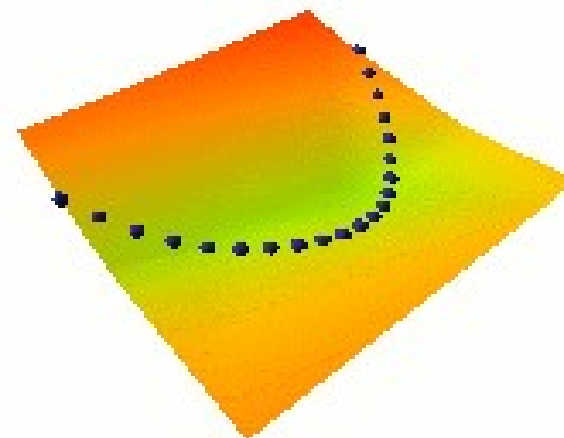
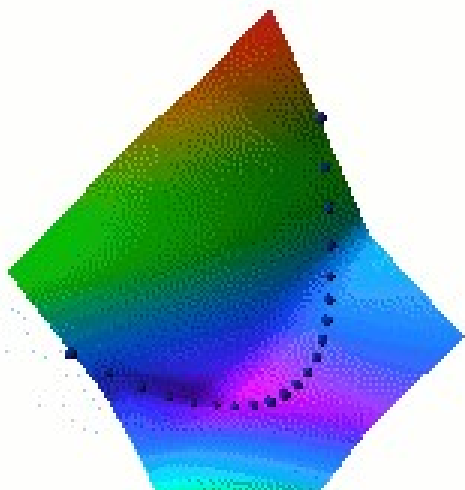
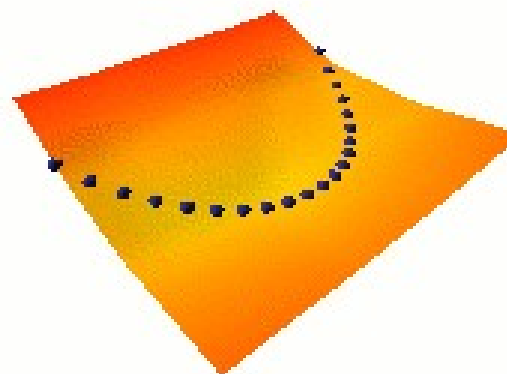
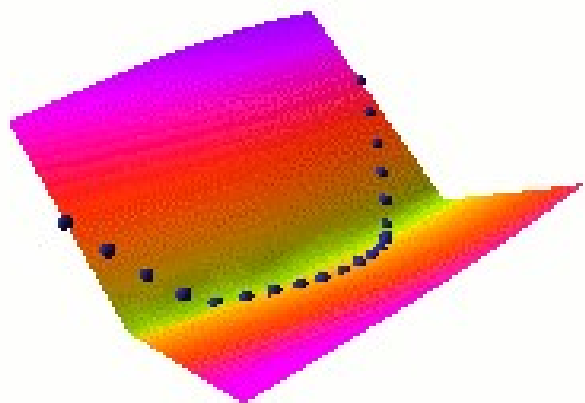
Missing: Unsupervised Learning

Energy-Based Unsupervised Learning

Y LeCun

Push down on the energy of desired outputs

Push up on everything else



Seven Strategies to Shape the Energy Function

Y LeCun

1. build the machine so that the volume of low energy stuff is constant
 - ▶ PCA, K-means, GMM, square ICA
2. push down of the energy of data points, push up everywhere else
 - ▶ Max likelihood (needs tractable partition function)
3. push down of the energy of data points, push up on chosen locations
 - ▶ contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow
4. minimize the gradient and maximize the curvature around data points
 - ▶ score matching
5. train a dynamical system so that the dynamics goes to the manifold
 - ▶ denoising auto-encoder
6. use a regularizer that limits the volume of space that has low energy
 - ▶ Sparse coding, sparse auto-encoder, PSD
7. if $E(Y) = \|Y - G(Y)\|^2$, make $G(Y)$ as "constant" as possible.
 - ▶ Contracting auto-encoder, saturating auto-encoder

#1: constant volume of low energy Energy surface for PCA and K-means

1. build the machine so that the volume of low energy stuff is constant

▶ PCA, K-means, GMM, square ICA...

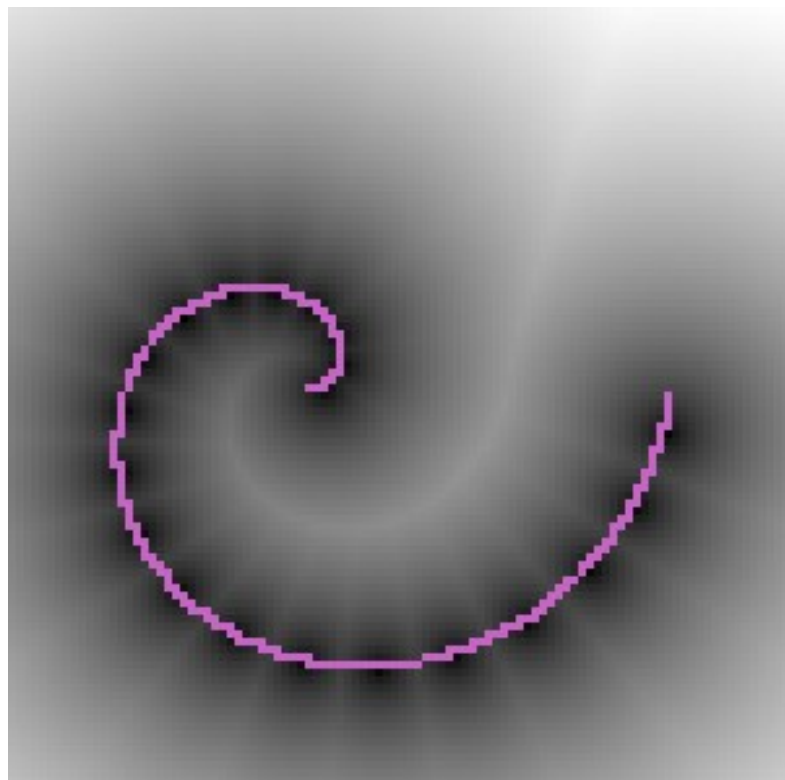
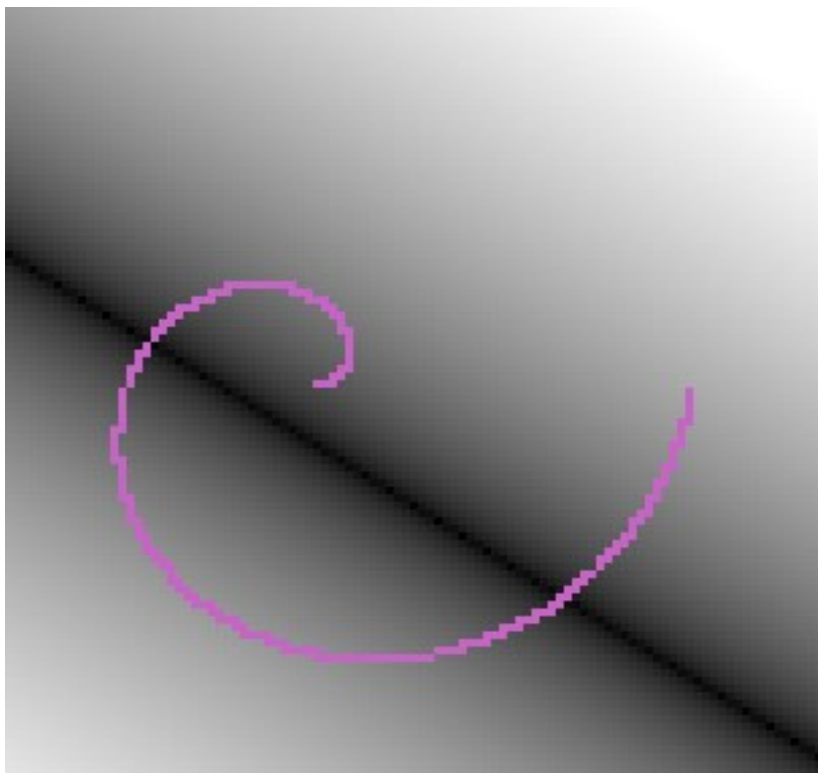
PCA

$$E(Y) = \|W^T WY - Y\|^2$$

K-Means,

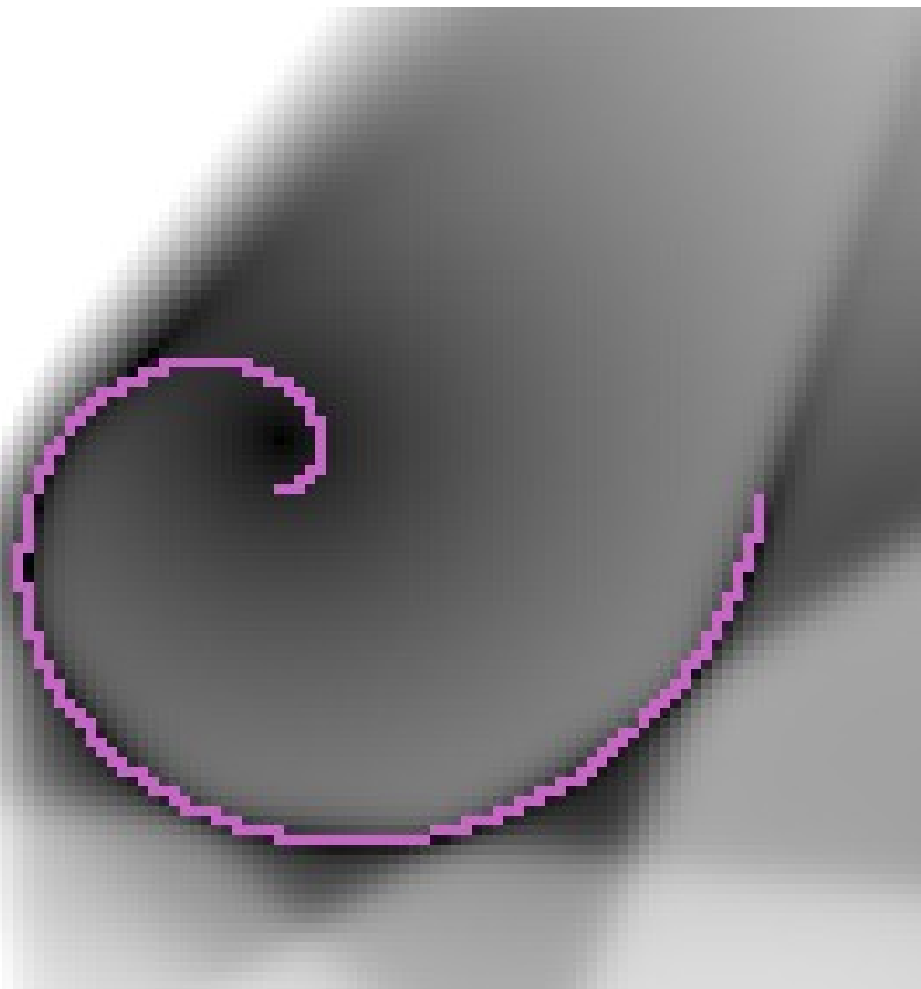
Z constrained to 1-of-K code

$$E(Y) = \min_z \sum_i \|Y - W_i Z_i\|^2$$

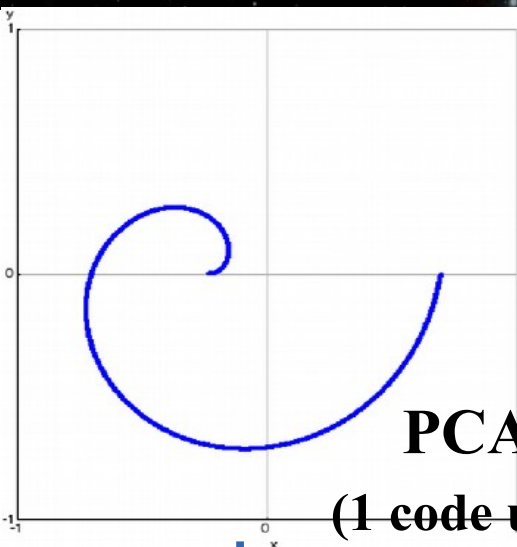


#6. use a regularizer that limits the volume of space that has low energy

■ Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition



Energy Functions of Various Methods



PCA
(1 code unit)

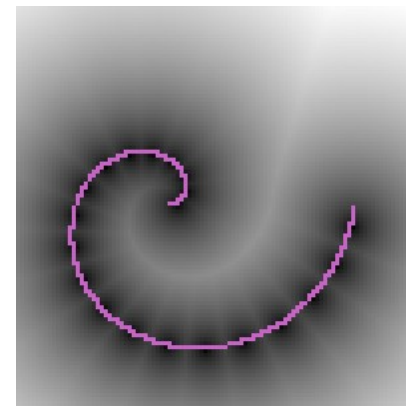
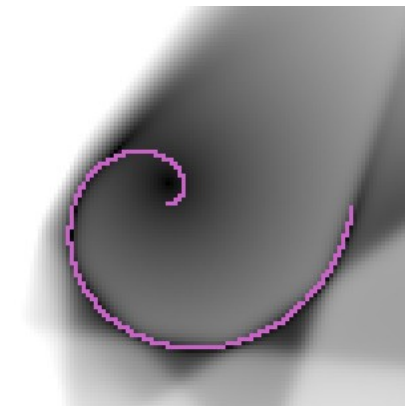
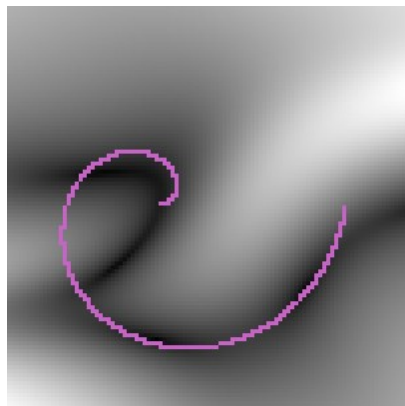
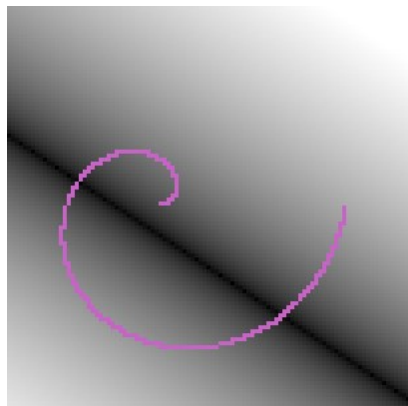
- 2 dimensional toy dataset: spiral
- Visualizing energy surface
 - (black = low, white = high)


autoencoder
(1 code unit)

sparse coding
(20 code units)

K-Means
(20 code units)

	PCA (1 code unit)	autoencoder (1 code unit)	sparse coding (20 code units)	K-Means (20 code units)
encoder	$W'Y$	$\sigma(W_e Y)$	$\sigma(W_e Z)$	—
decoder	WZ	$W_d Z$	$W_d Z$	WZ
energy	$\ Y - WZ\ ^2$	$\ Y - WZ\ ^2$	$\ Y - WZ\ ^2$	$\ Y - WZ\ ^2$
loss	$F(Y)$	$F(Y)$	$F(Y)$	$F(Y)$
pull-up	dimens.	dimens.	sparsity	1-of-N code

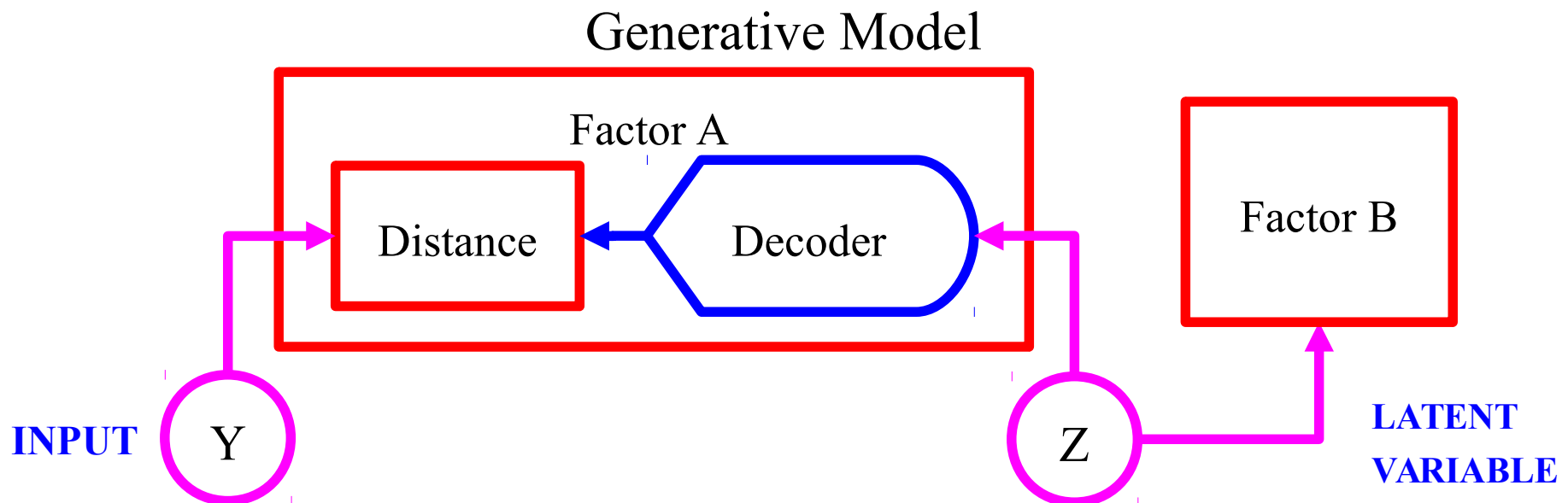




**Dictionary Learning With
Fast Approximate Inference:
Sparse Auto-Encoders**

How to Speed Up Inference in a Generative Model?

- Factor Graph with an asymmetric factor
- Inference $Z \rightarrow Y$ is easy
 - ▶ Run Z through deterministic decoder, and sample Y
- Inference $Y \rightarrow Z$ is hard, particularly if Decoder function is many-to-one
 - ▶ MAP: minimize sum of two factors with respect to Z
 - ▶ $Z^* = \operatorname{argmin}_z \text{Distance}[\text{Decoder}(Z), Y] + \text{FactorB}(Z)$
- Examples: K-Means (1 of K), Sparse Coding (sparse), Factor Analysis



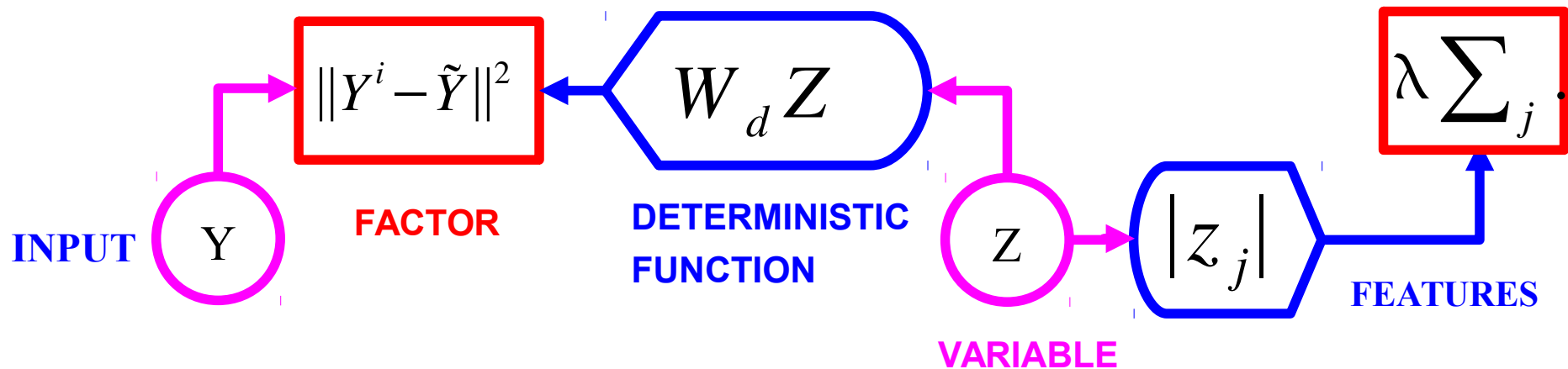
Sparse Modeling: Sparse Coding + Dictionary Learning

[Olshausen & Field 1997]

■ Sparse linear reconstruction

■ Energy = reconstruction_error + code_prediction_error + code_sparsity

$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \lambda \sum_j |z_j|$$

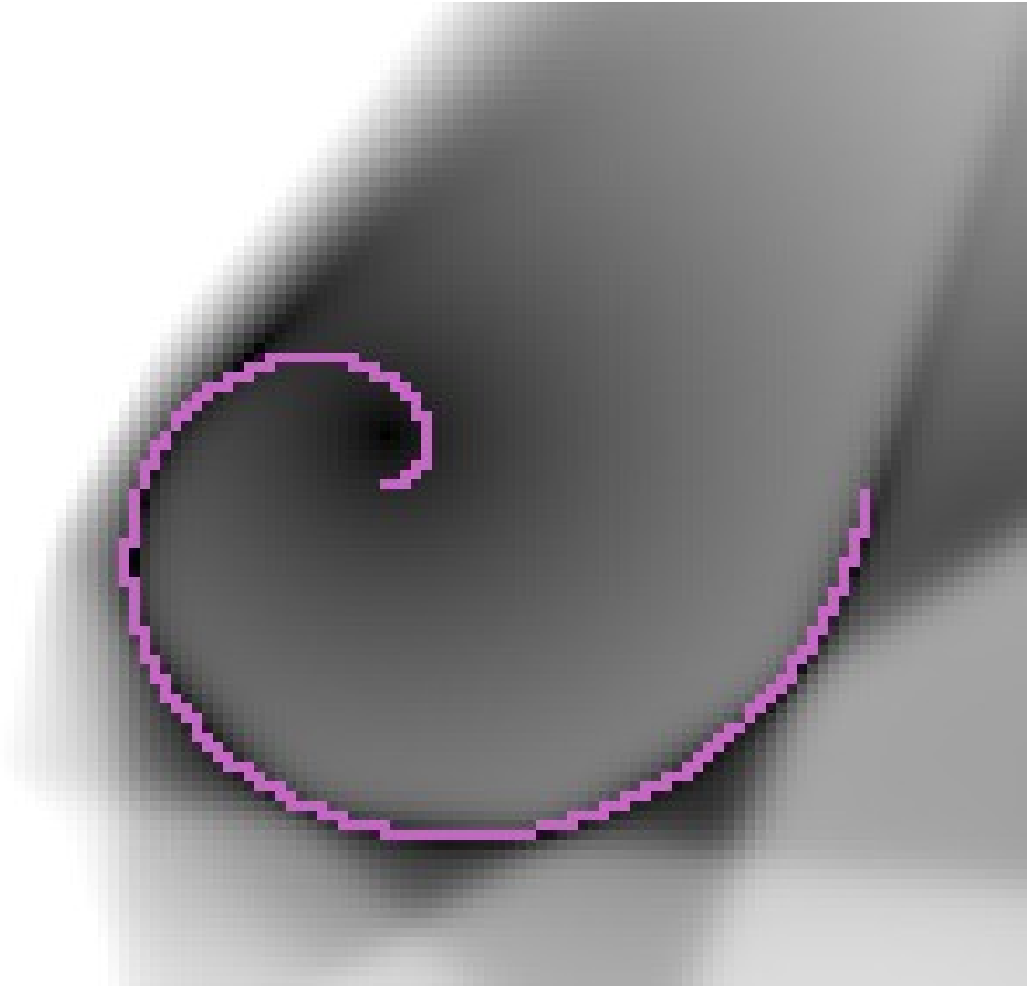


■ Inference is expensive: ISTA/FISTA, CGIHT, coordinate descent....

$$Y \rightarrow \hat{Z} = \operatorname{argmin}_Z E(Y, Z)$$

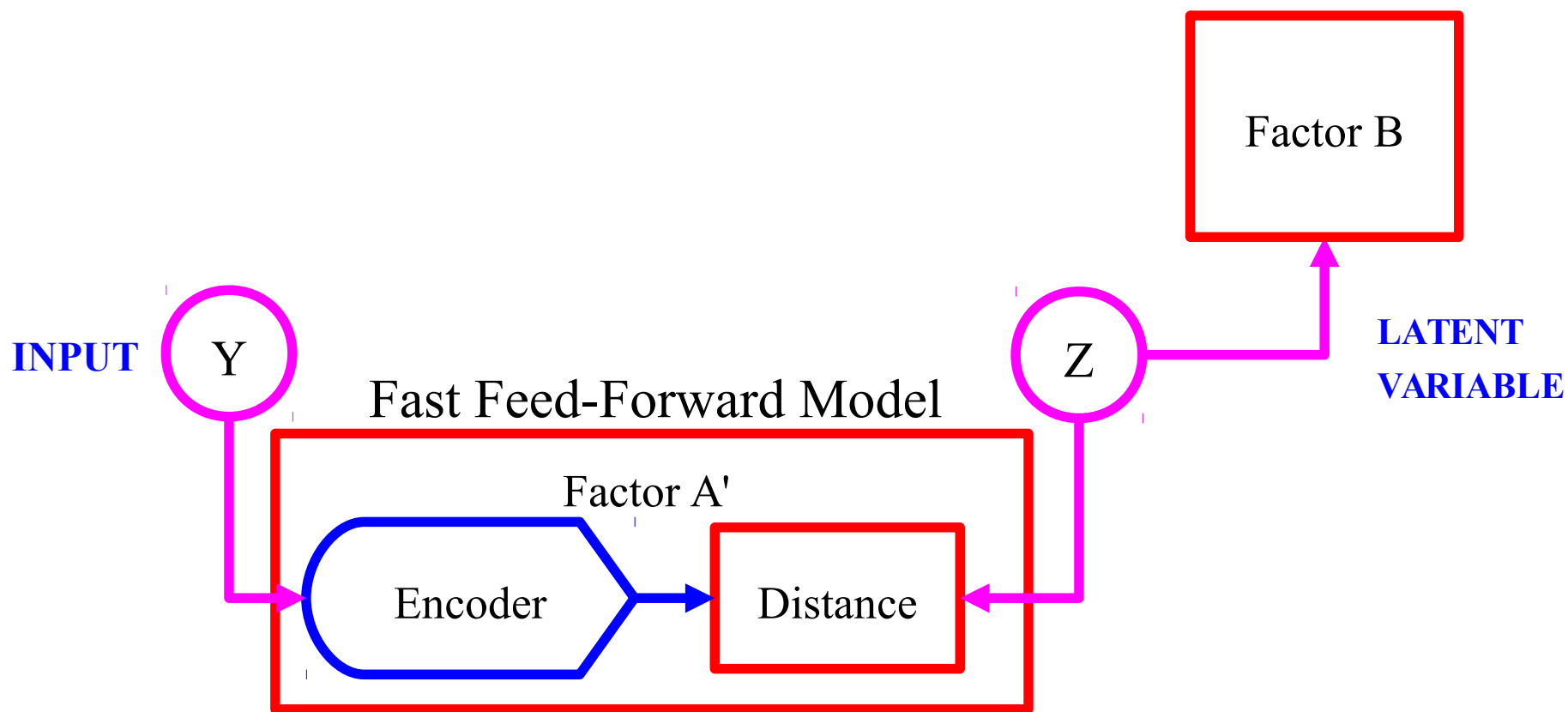
#6. use a regularizer that limits the volume of space that has low energy

■ Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition



Encoder Architecture

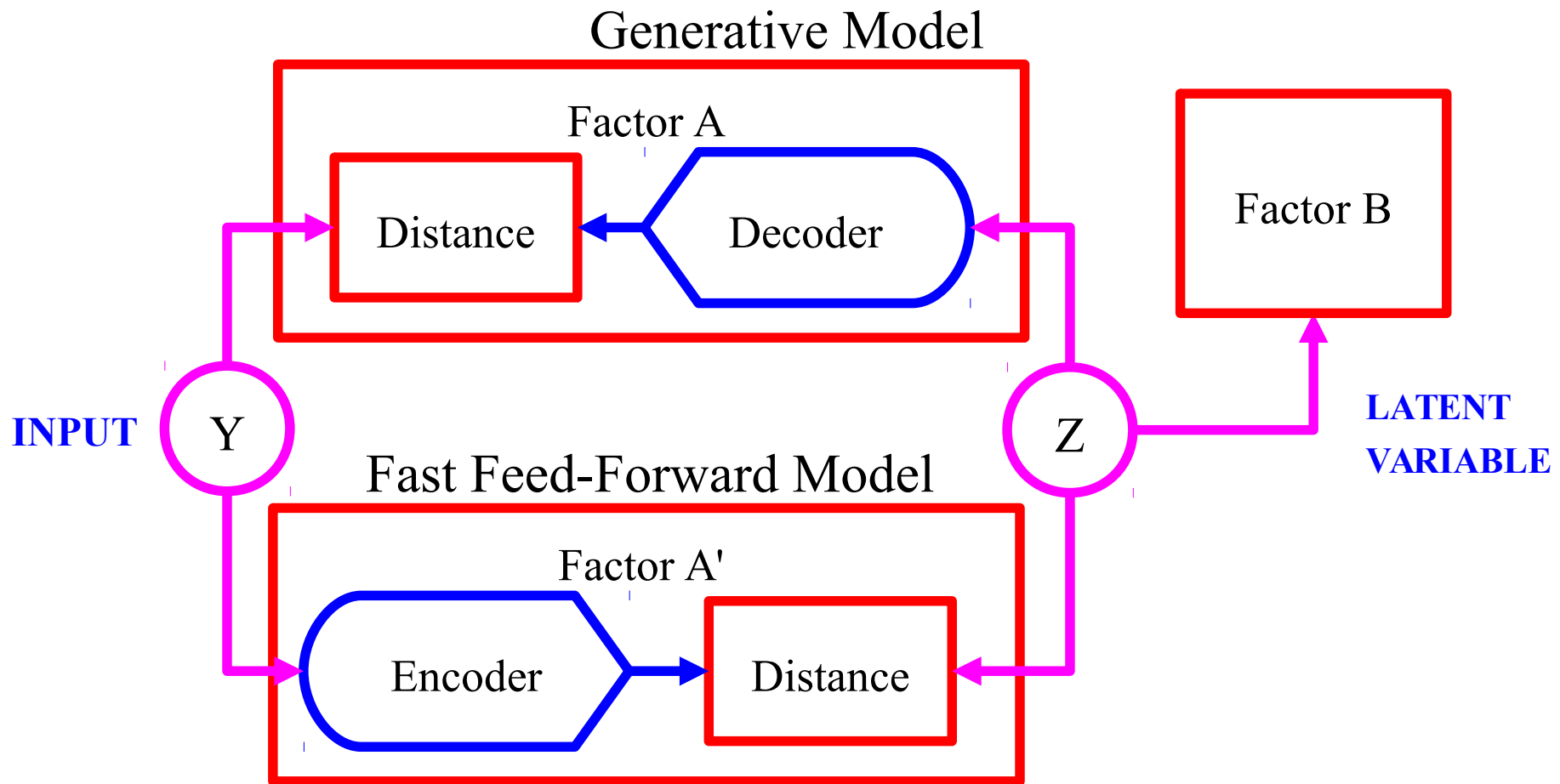
■ Examples: most ICA models, Product of Experts




Encoder-Decoder Architecture

[Kavukcuoglu, Ranzato, LeCun, rejected by every conference, 2008-2009]

- Train a "simple" feed-forward function to predict the result of a complex optimization on the data points of interest



- 1. Find optimal Z_i for all Y_i ; 2. Train Encoder to predict Z_i from Y_i



**Learning to Perform
Approximate Inference:
Predictive Sparse Decomposition
Sparse Auto-Encoders**

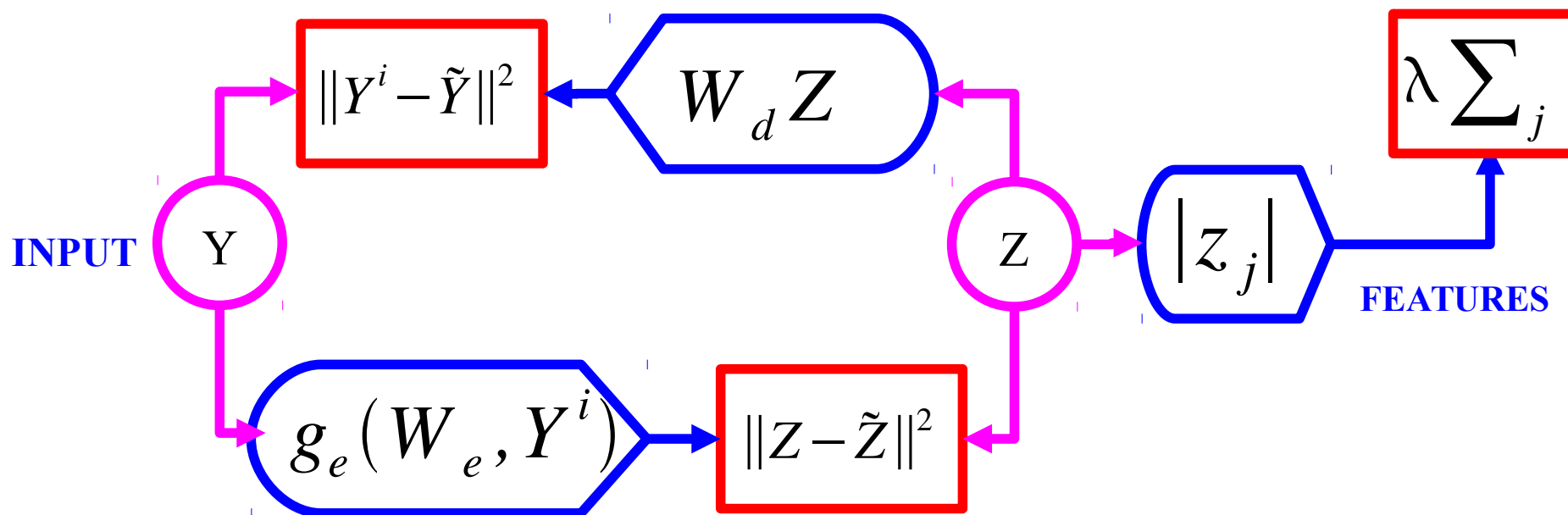
Sparse auto-encoder: Predictive Sparse Decomposition (PSD)

[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

- Prediction the optimal code with a **trained encoder**
- Energy = reconstruction_error + code_prediction_error + code_sparsity

$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \|Z - g_e(W_e, Y^i)\|^2 + \lambda \sum_j |z_j|$$

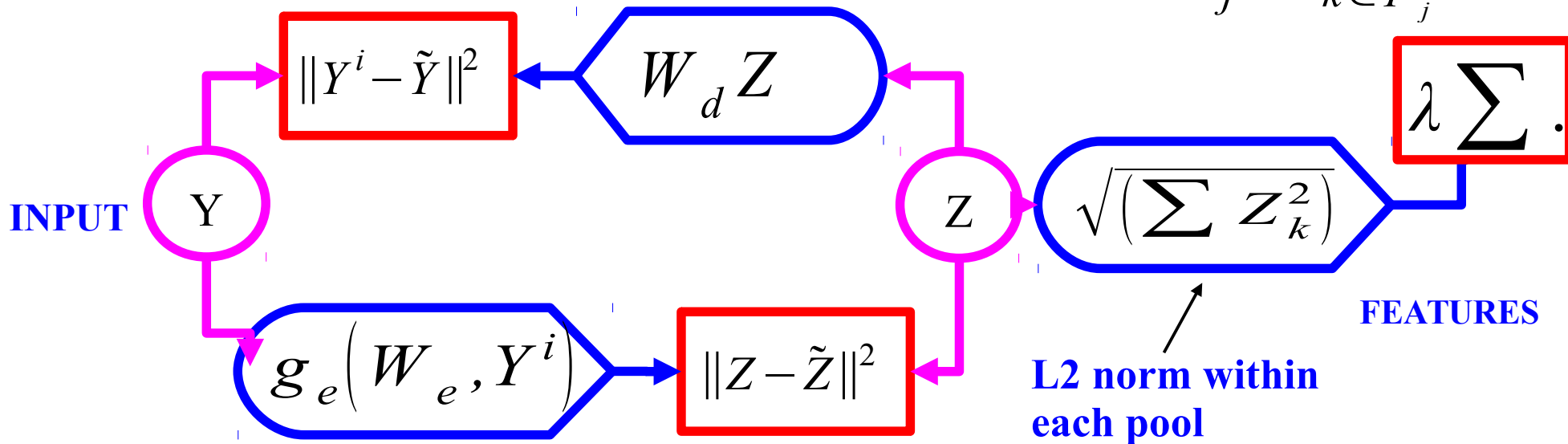
$$g_e(W_e, Y^i) = \text{shrinkage}(W_e Y^i)$$



Regularized Encoder-Decoder Model (auto-Encoder) for Unsupervised Feature Learning

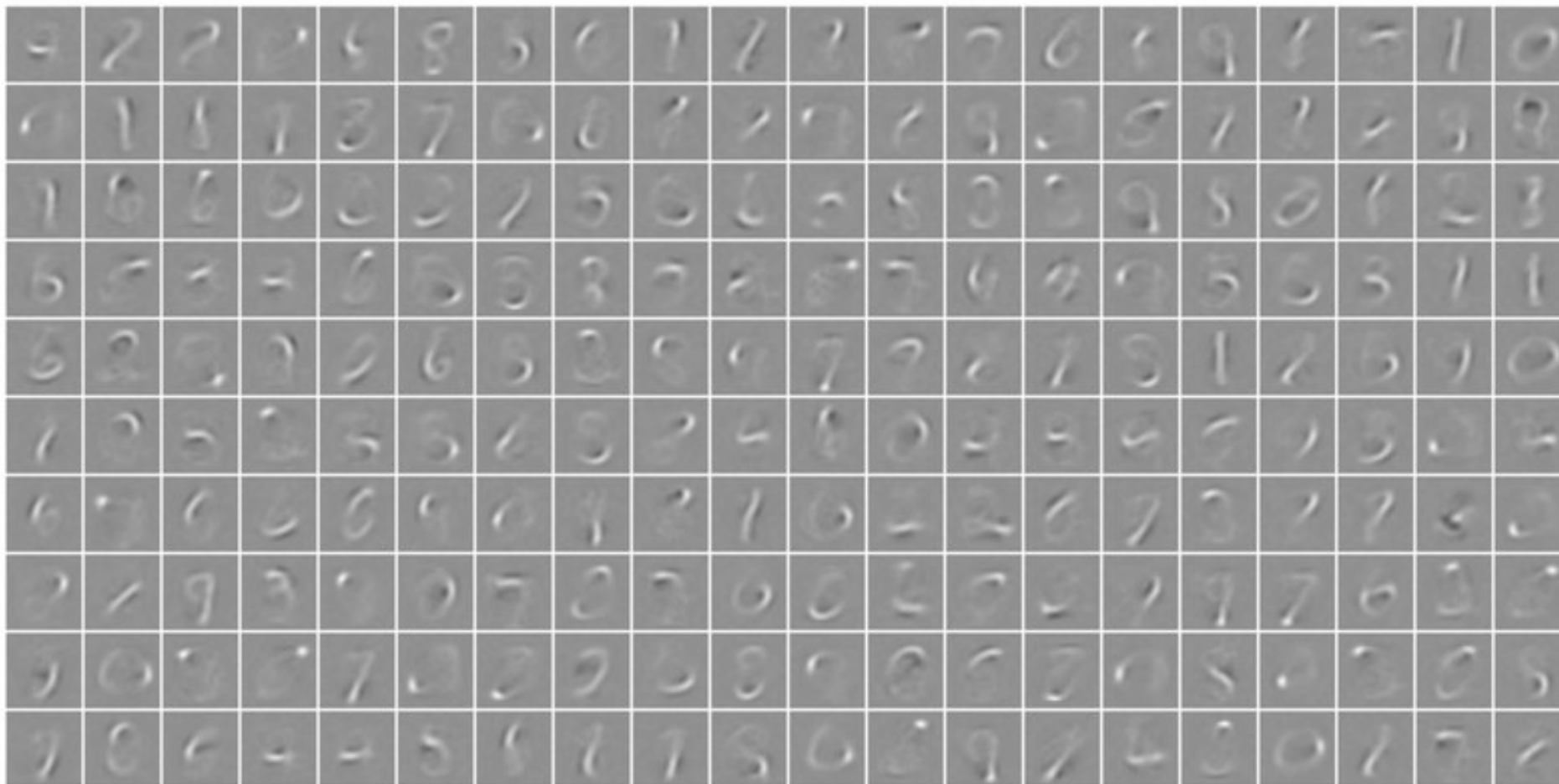
- Encoder: computes feature vector Z from input X
- Decoder: reconstructs input X from feature vector Z
- Feature vector: high dimensional and regularized (e.g. sparse)
- Factor graph with energy function $E(X,Z)$ with 3 terms:
 - Linear decoding function and reconstruction error
 - Non-Linear encoding function and prediction error term
 - Pooling function and regularization term (e.g. sparsity)

$$E(Y,Z) = \|Y - W_d Z\|^2 + \|Z - g_e(W_e, Y)\|^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}$$



PSD: Basis Functions on MNIST

■ Basis functions (and encoder matrix) are digit parts



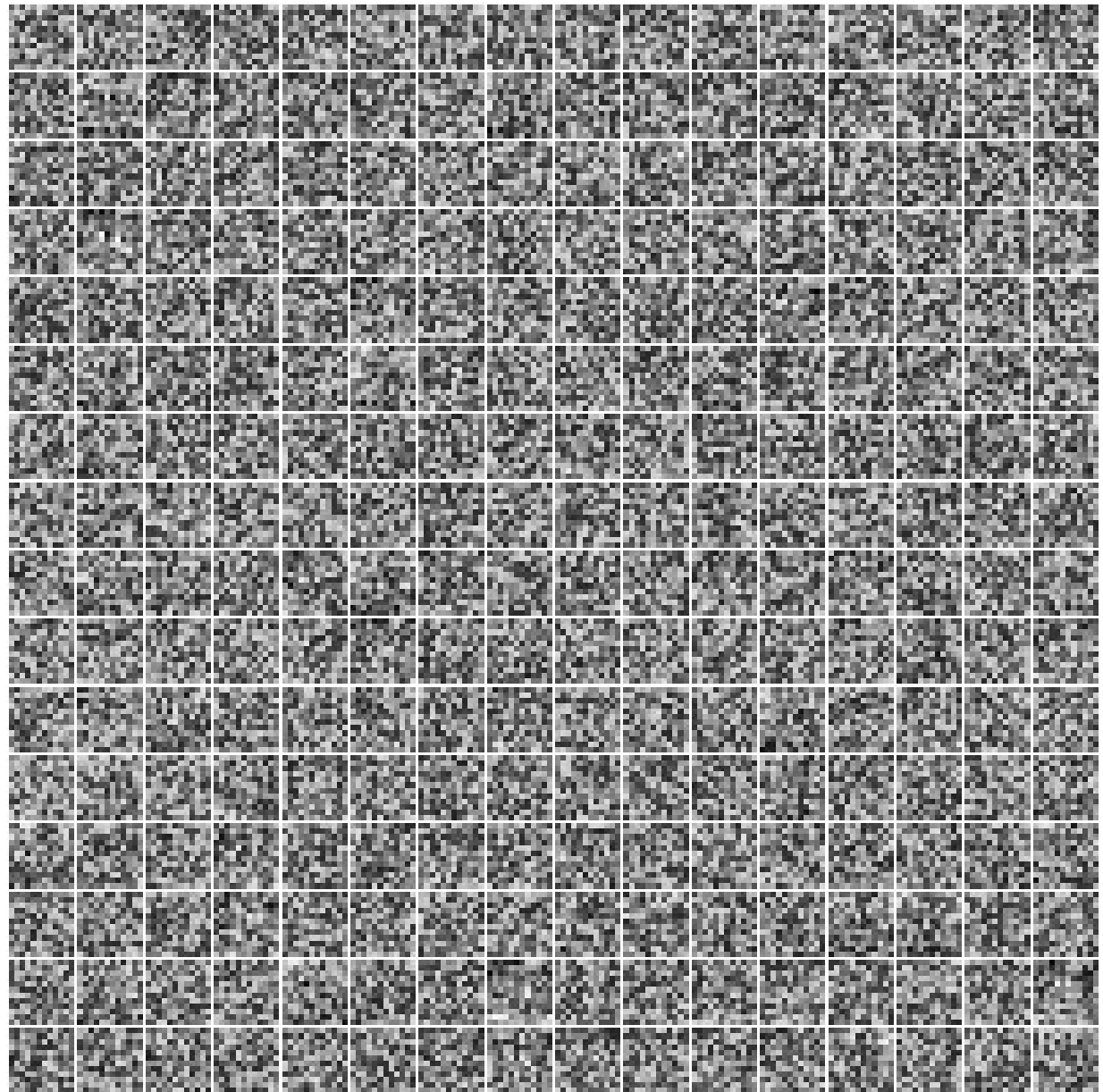
Predictive Sparse Decomposition (PSD): Training

Y LeCun

- Training on natural images patches.

- ▶ 12X12

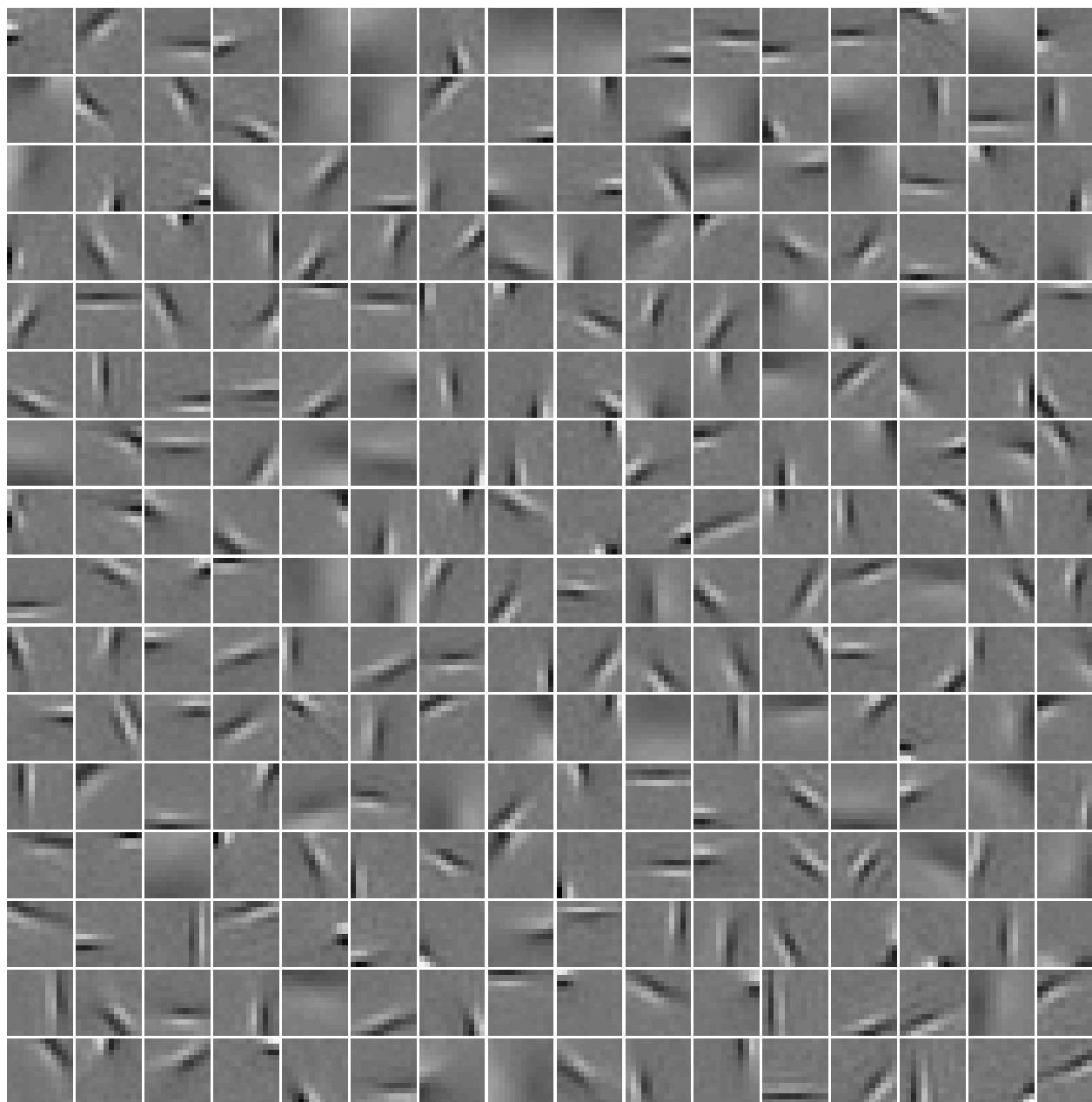
- ▶ 256 basis functions




iteration no 0

Learned Features on natural patches: V1-like receptive fields

Y LeCun



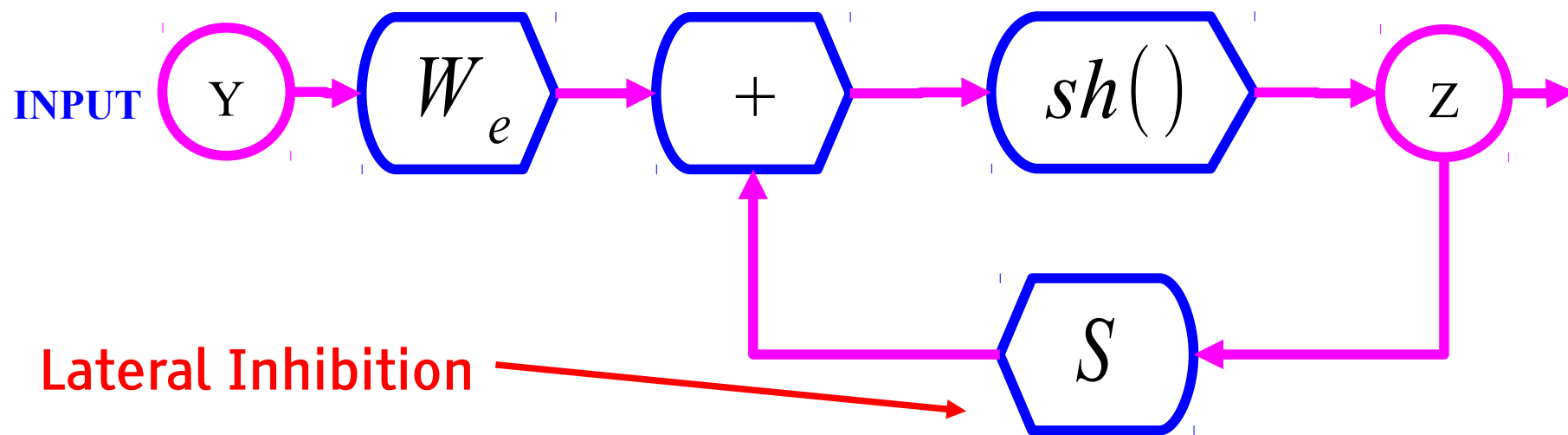


**Learning to Perform
Approximate Inference
LISTA**

Better Idea: Give the “right” structure to the encoder

Y LeCun

- 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/FISTA reparameterized:

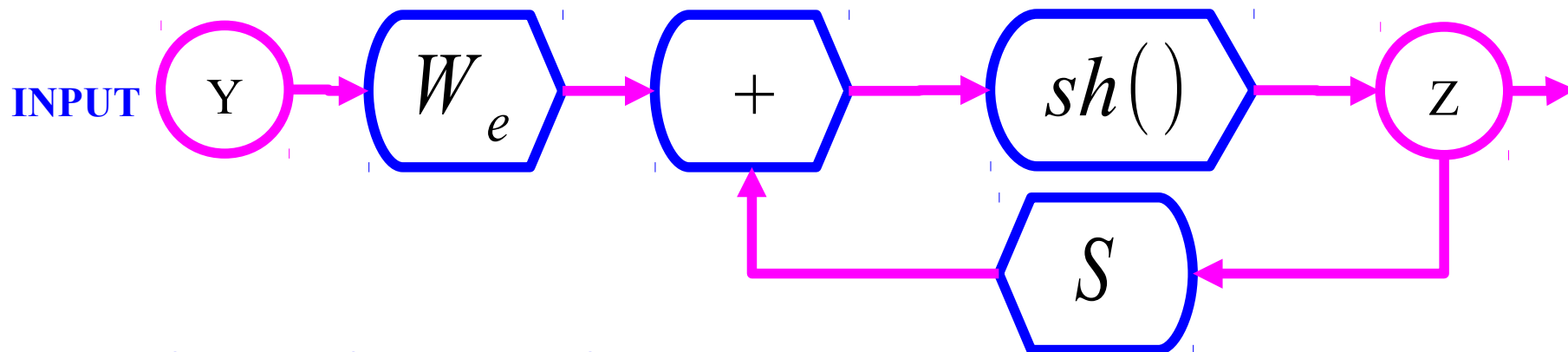
$$Z(t+1) = \text{Shrinkage}_{\lambda/L} [W_e^T Y + S Z(t)]; \quad W_e = \frac{1}{L} W_d; \quad S = I - \frac{1}{L} W_d^T W_d$$

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

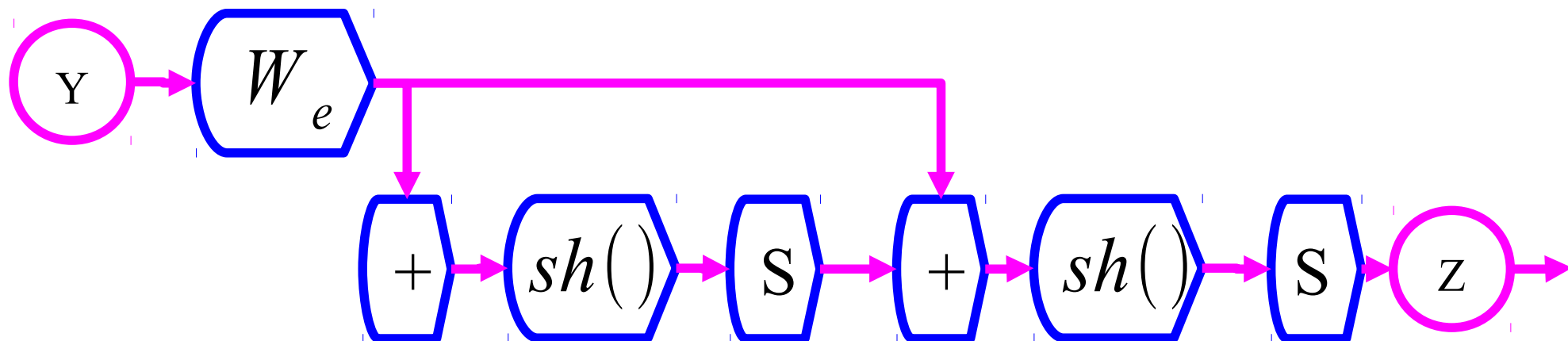
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

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

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

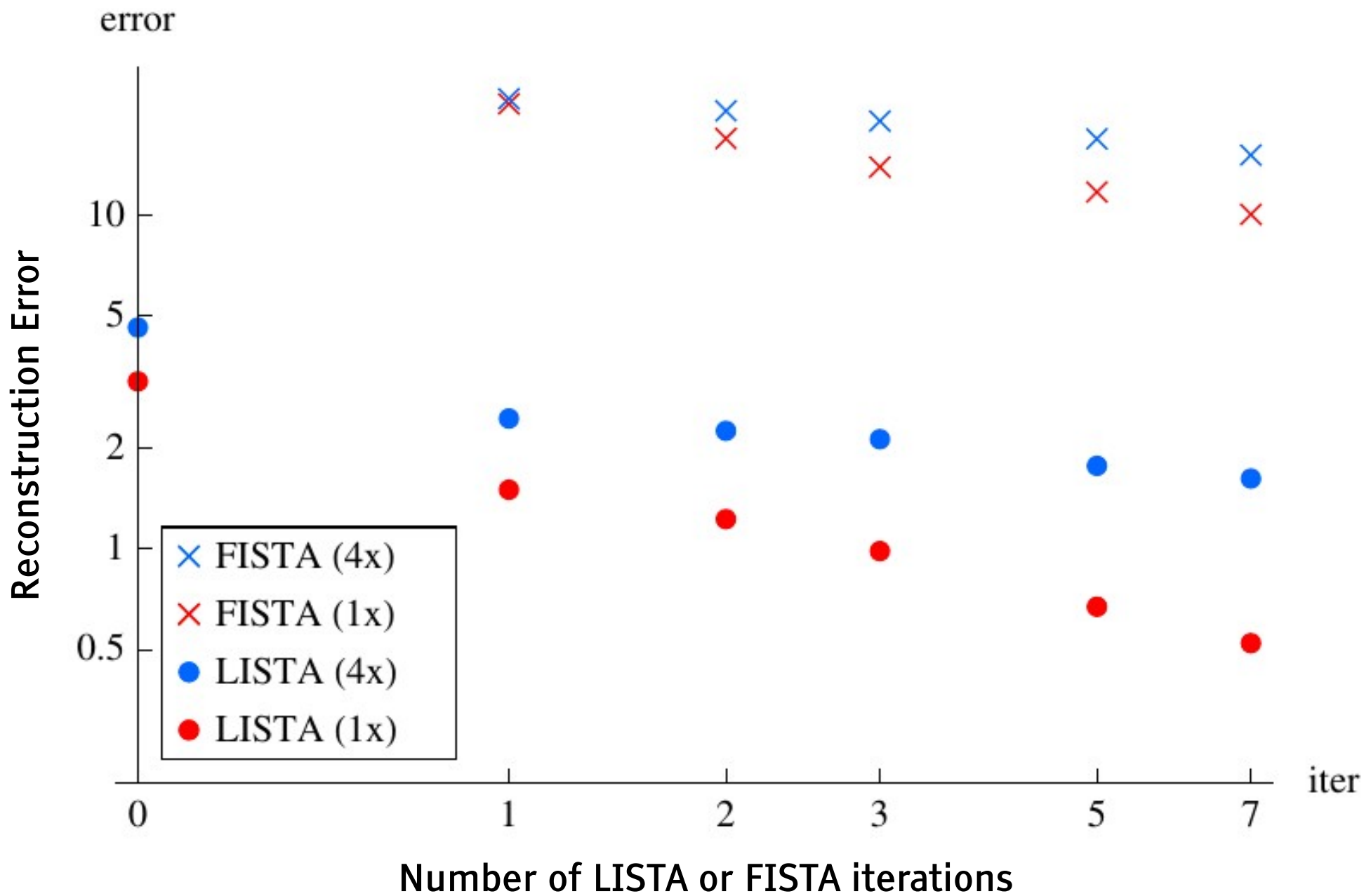


- Time-Unfold the flow graph for K iterations
- Learn the W_e and S matrices with "backprop-through-time"
- Get the best approximate solution within K iterations



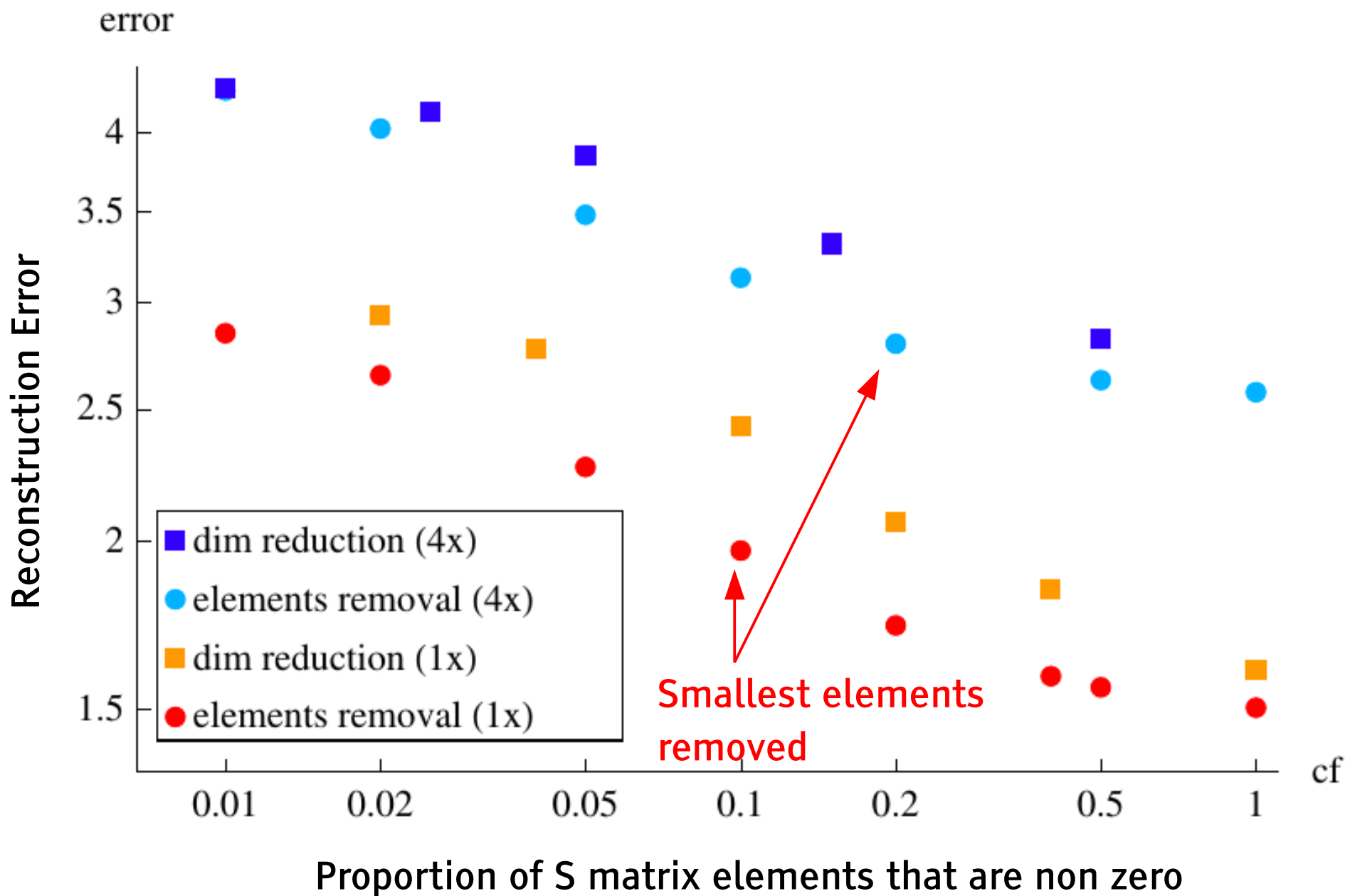
Learning ISTA (LISTA) vs ISTA/FISTA

Y LeCun



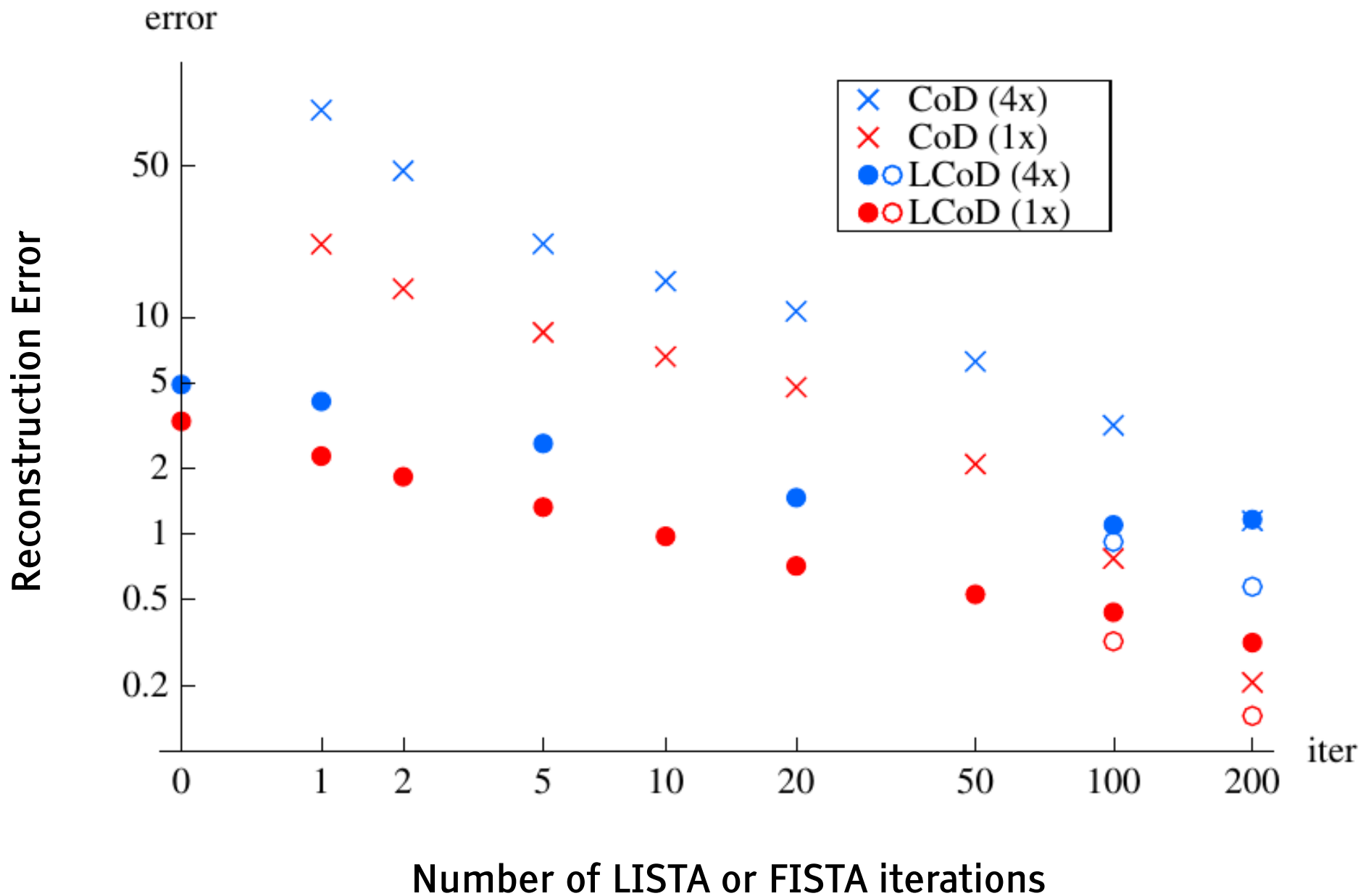
LISTA with partial mutual inhibition matrix

Y LeCun



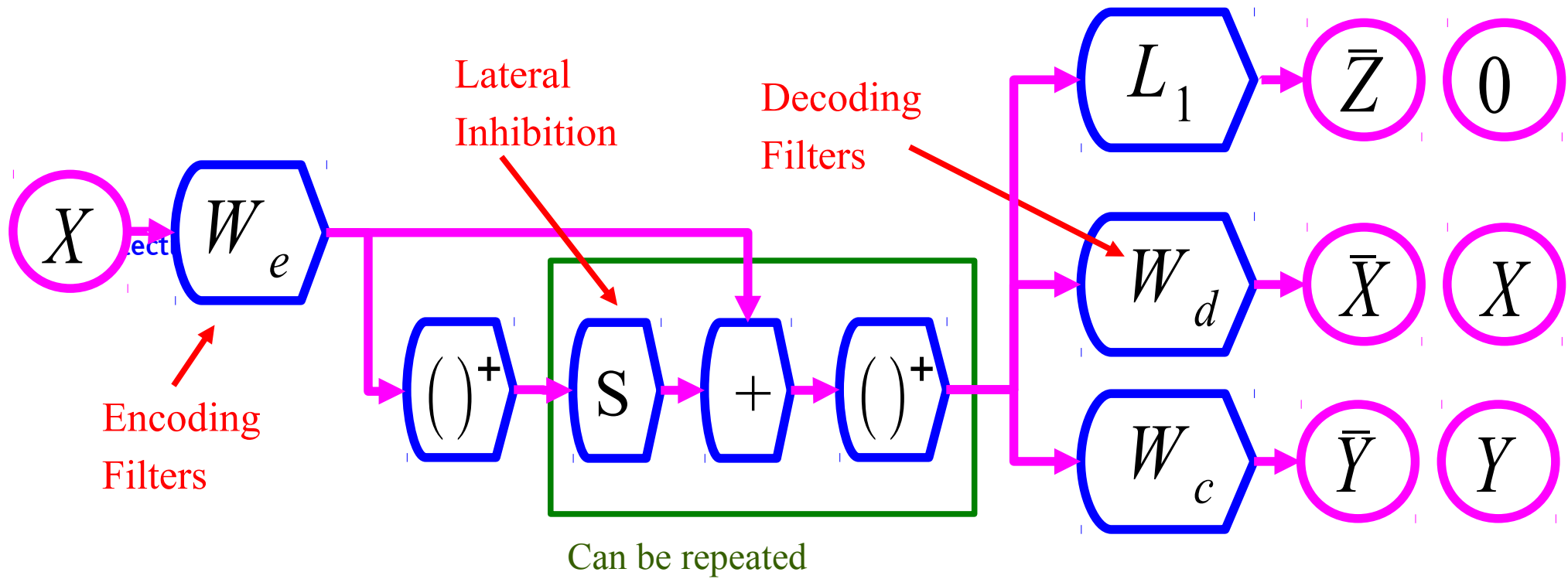
Learning Coordinate Descent (LCoD): faster than LISTA

Y LeCun



Discriminative Recurrent Sparse Auto-Encoder (DrSAE)

Y LeCun



[Rolfe & LeCun ICLR 2013]

- Rectified linear units
- Classification loss: cross-entropy
- Reconstruction loss: squared error
- Sparsity penalty: L1 norm of last hidden layer
- Rows of W_d and columns of W_e constrained in unit sphere

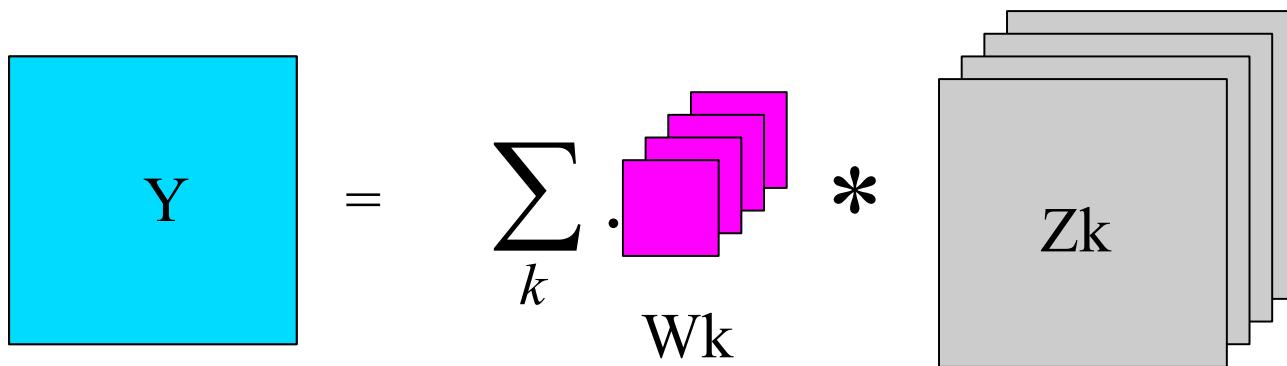
Convolutional Sparse Coding

Replace the dot products with dictionary element by convolutions.

- ▶ Input Y is a full image
- ▶ Each code component Z_k 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. $E(Y, Z) = \|Y - \sum_k W_k * Z_k\|^2 + \alpha \sum_k |Z_k|$



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

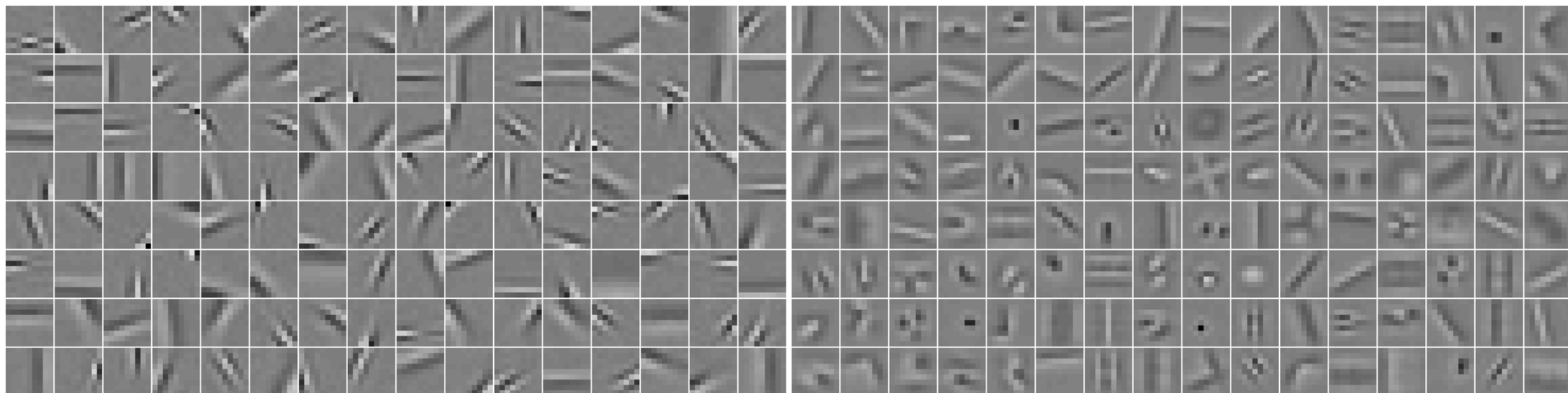
Convolutional PSD: Encoder with a soft sh() Function

Y LeCun

Convolutional Formulation

- ▶ Extend sparse coding from **PATCH** to **IMAGE**

$$\mathcal{L}(x, z, \mathcal{D}) = \frac{1}{2} \left\| x - \sum_{k=1}^K \mathcal{D}_k * z_k \right\|_2^2 + \sum_{k=1}^K \left\| z_k - f(W^k * x) \right\|_2^2 + |z|_1$$



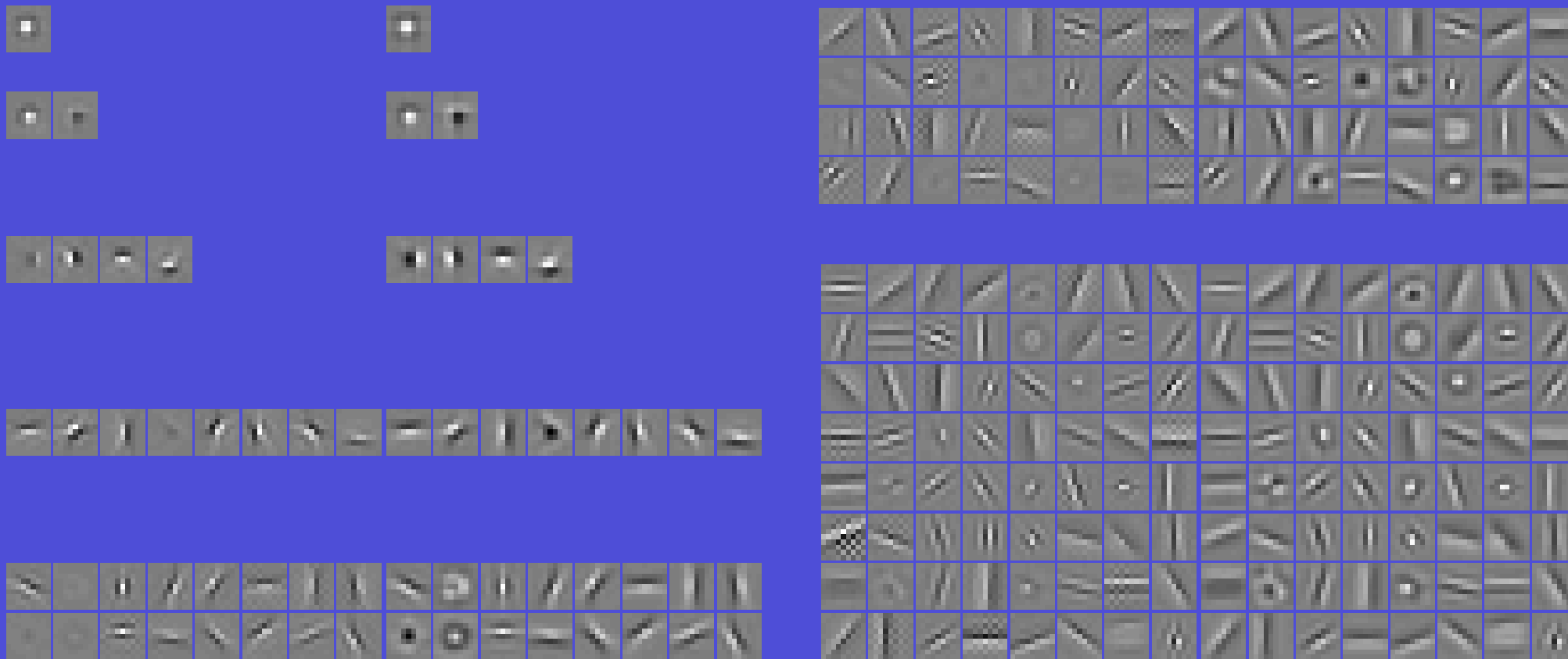
- ▶ **PATCH** based learning

- ▶ **CONVOLUTIONAL** learning

Convolutional Sparse Auto-Encoder on Natural Images

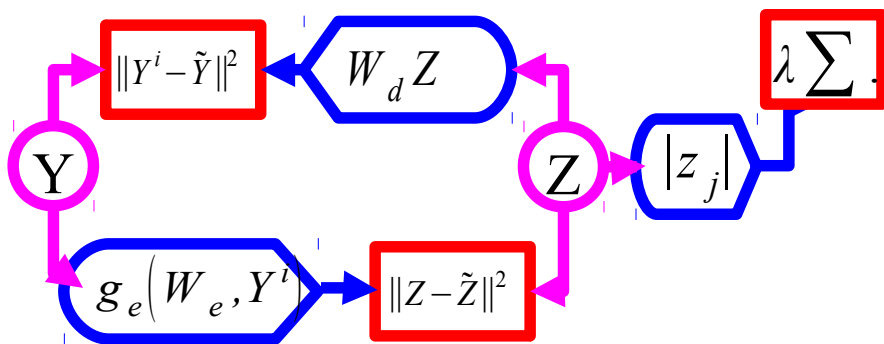
Y LeCun

- Filters and Basis Functions obtained with 1, 2, 4, 8, 16, 32, and 64 filters.



Using PSD to Train a Hierarchy of Features

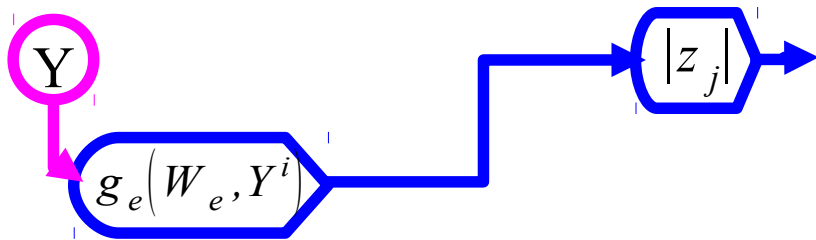
Phase 1: train first layer using PSD



FEATURES

Using PSD to Train a Hierarchy of Features

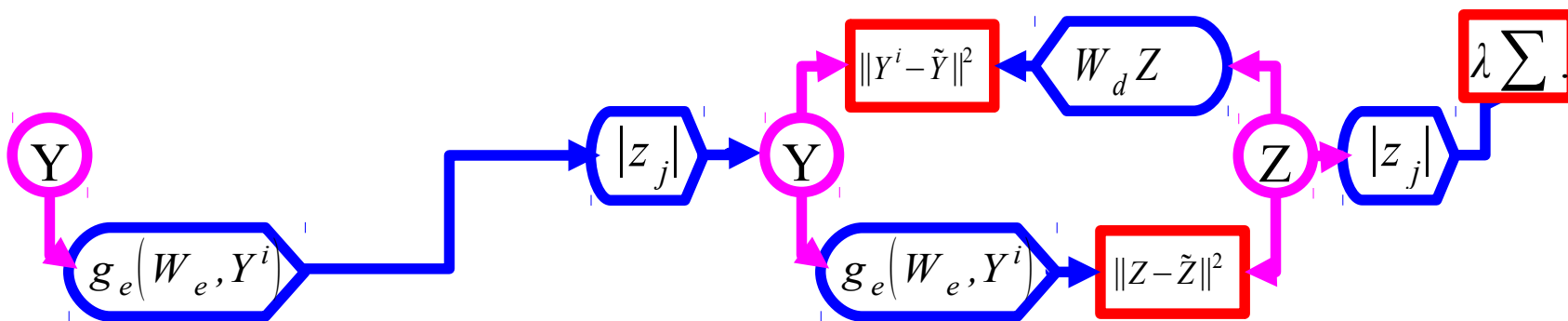
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor



FEATURES

Using PSD to Train a Hierarchy of Features

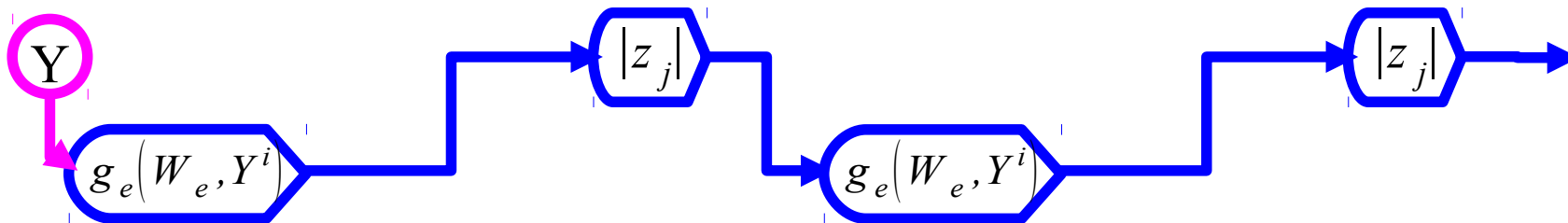
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD



FEATURES

Using PSD to Train a Hierarchy of Features

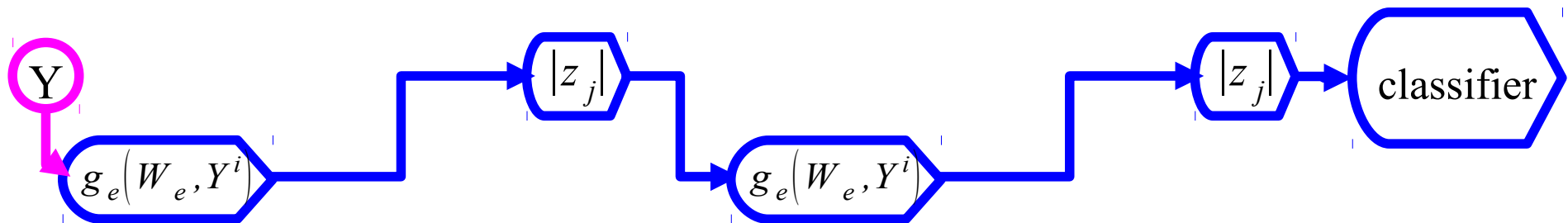
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor



FEATURES

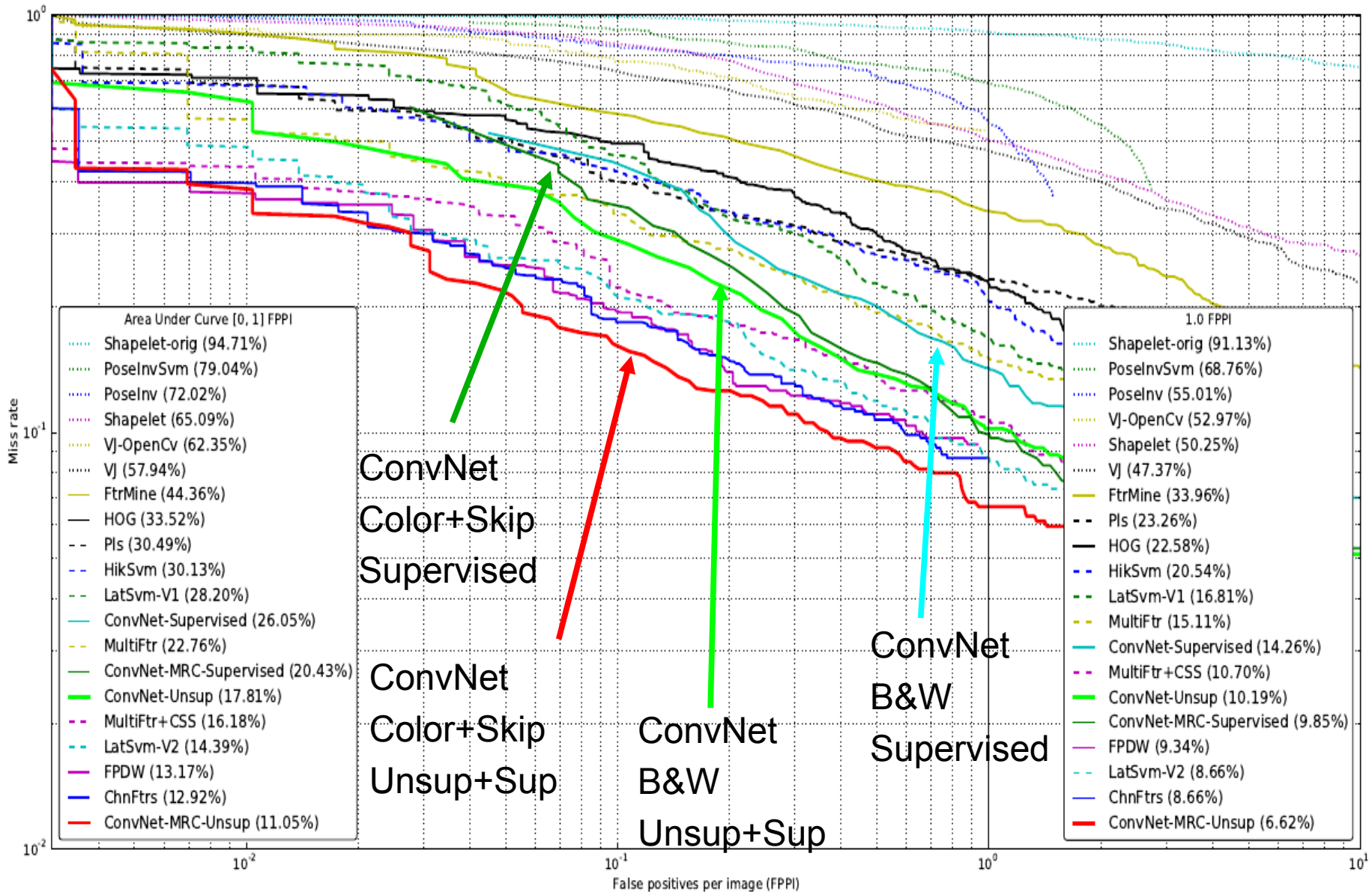
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor
- Phase 5: train a supervised classifier on top
- Phase 6 (optional): train the entire system with supervised back-propagation



FEATURES

Pedestrian Detection: INRIA Dataset. Miss rate vs false positives



[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]

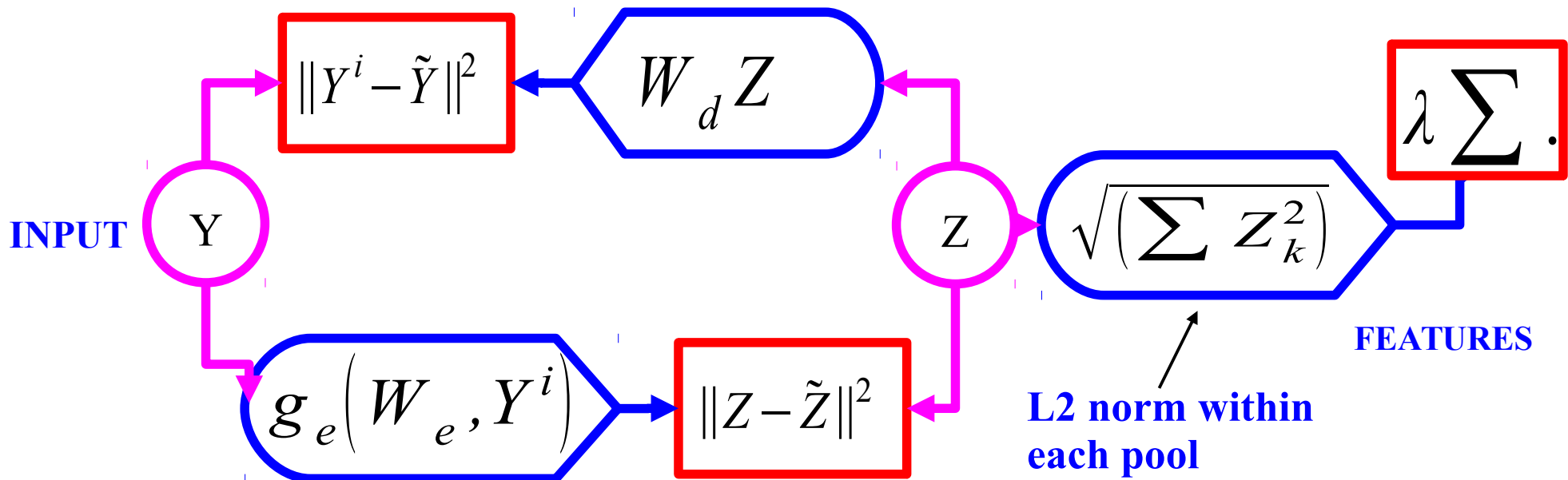


Unsupervised Learning: Invariant Features

Learning Invariant Features with L2 Group Sparsity

- Unsupervised PSD ignores the spatial pooling step.
- Could we devise a similar method that learns the pooling layer as well?
- Idea [Hyvarinen & Hoyer 2001]: **group sparsity** on pools of features
 - ▶ Minimum number of pools must be non-zero
 - ▶ Number of features that are on within a pool doesn't matter
 - ▶ Pools tend to regroup similar features

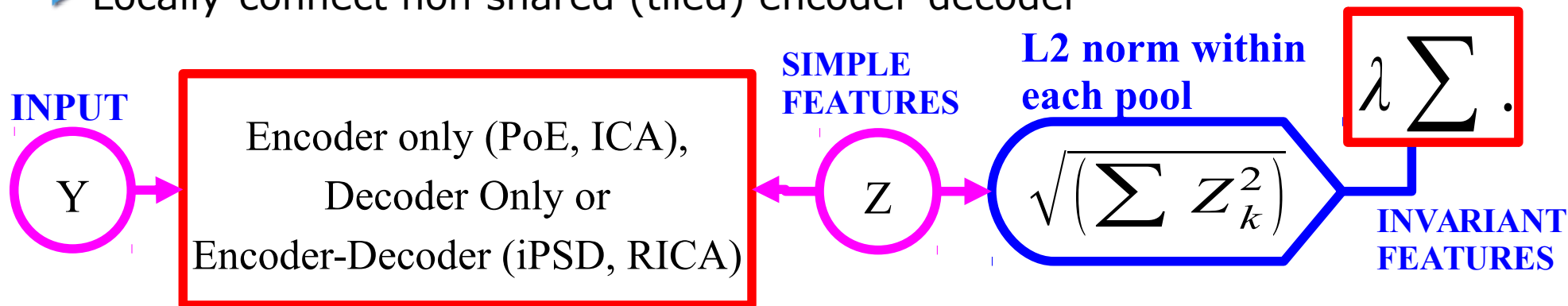
$$E(Y, Z) = \|Y - W_d Z\|^2 + \|Z - g_e(W_e, Y)\|^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}$$



Learning Invariant Features with L2 Group Sparsity

Y LeCun

- **Idea: features are pooled in group.**
 - ▶ Sparsity: sum over groups of L2 norm of activity in group.
- **[Hyvärinen Hoyer 2001]: "subspace ICA"**
 - ▶ decoder only, square
- **[Welling, Hinton, Osindero NIPS 2002]: pooled product of experts**
 - ▶ encoder only, overcomplete, log student-T penalty on L2 pooling
- **[Kavukcuoglu, Ranzato, Fergus LeCun, CVPR 2010]: Invariant PSD**
 - ▶ encoder-decoder (like PSD), overcomplete, L2 pooling
- **[Le et al. NIPS 2011]: Reconstruction ICA**
 - ▶ Same as [Kavukcuoglu 2010] with linear encoder and tied decoder
- **[Gregor & LeCun arXiv:1006:0448, 2010] [Le et al. ICML 2012]**
 - ▶ Locally-connect non shared (tiled) encoder-decoder



Groups are local in a 2D Topographic Map

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

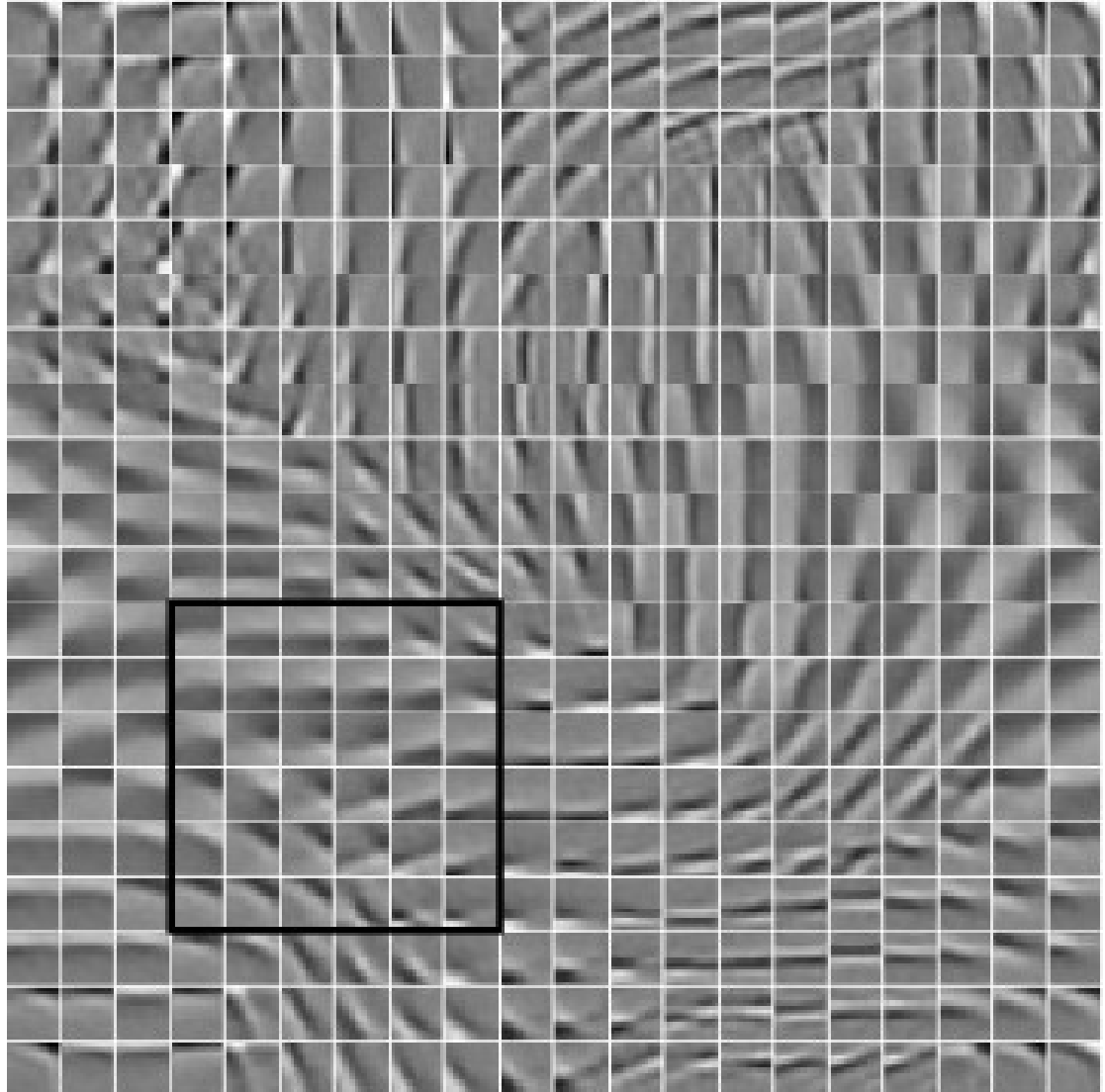
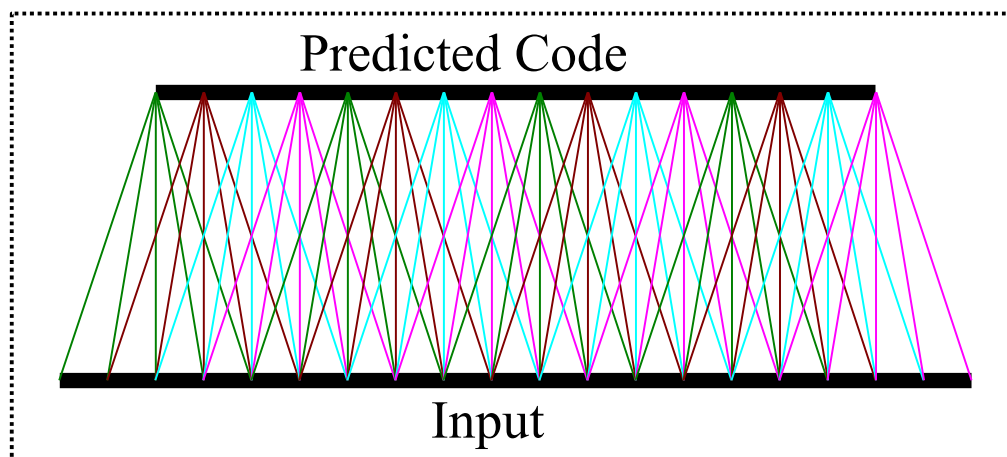
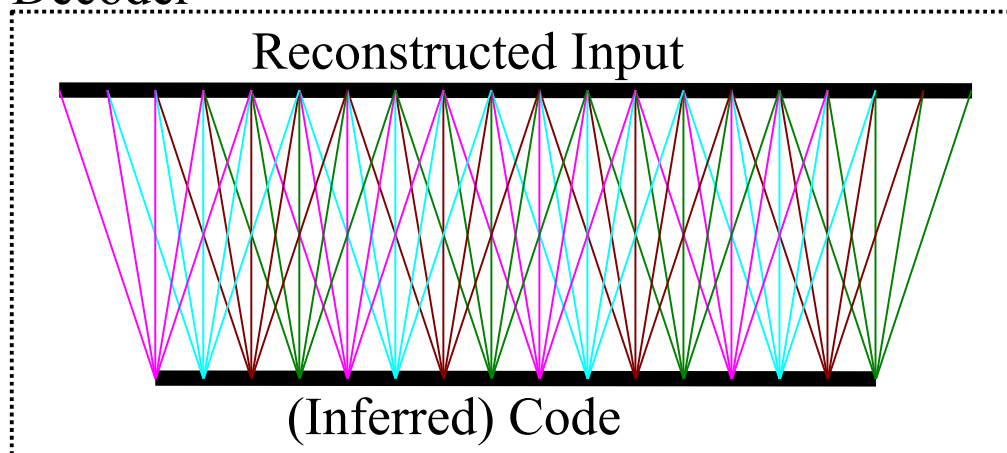


Image-level training, local filters but no weight sharing

■ Training on 115x115 images. Kernels are 15x15 (not shared across space!)

- ▶ [Gregor & LeCun 2010]
- ▶ Local receptive fields
- ▶ No shared weights
- ▶ 4x overcomplete
- ▶ L2 pooling
- ▶ Group sparsity over pools

Decoder

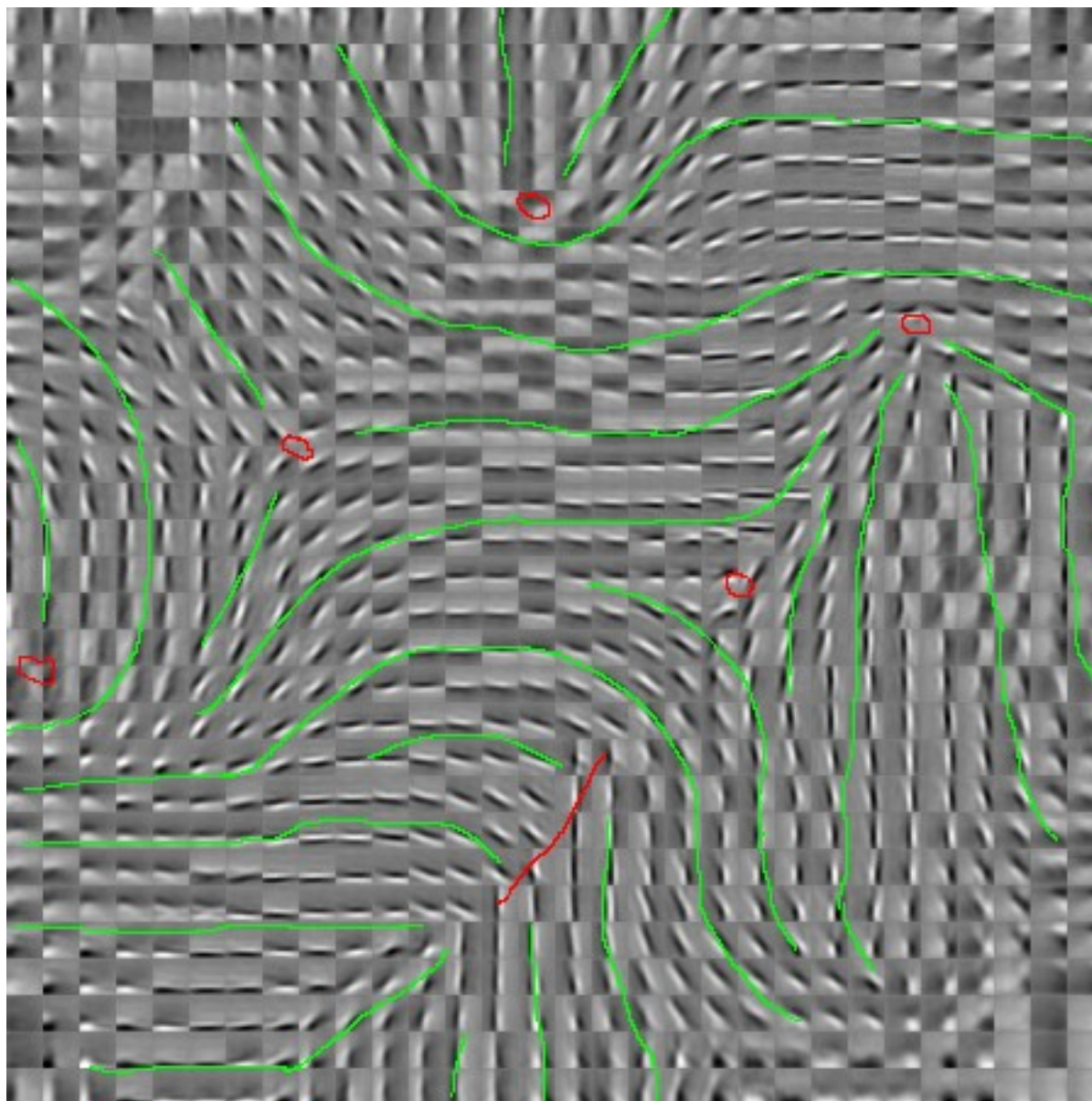


Encoder

Image-level training, local filters but no weight sharing

Y LeCun

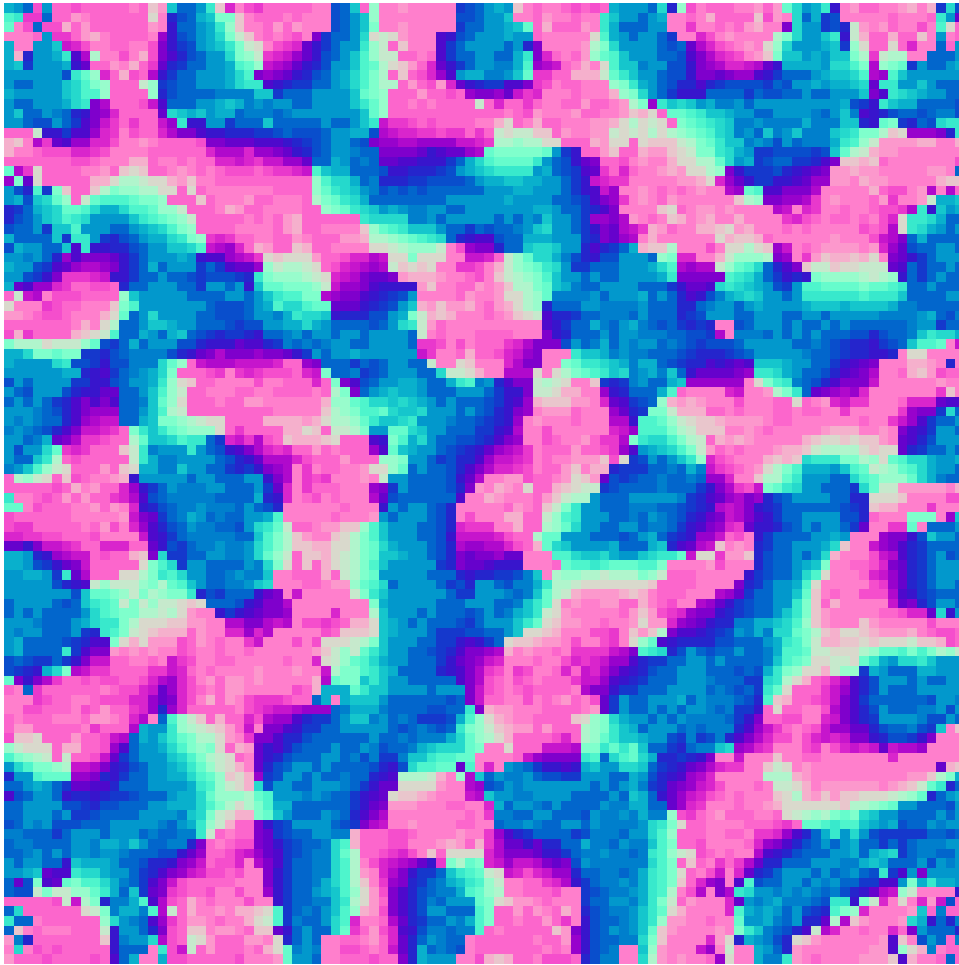
- Training on 115×115 images. Kernels are 15×15 (not shared across space!)



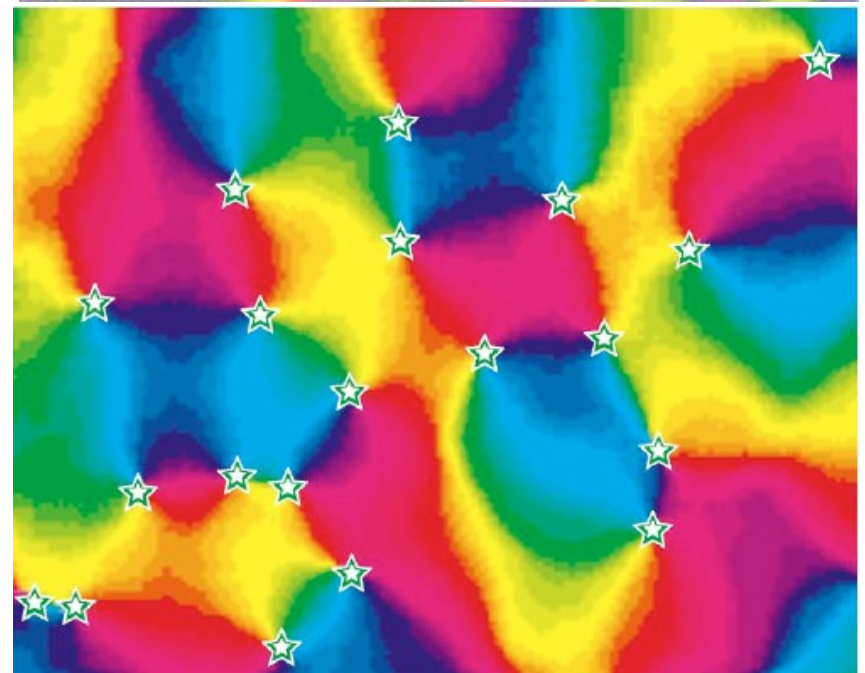
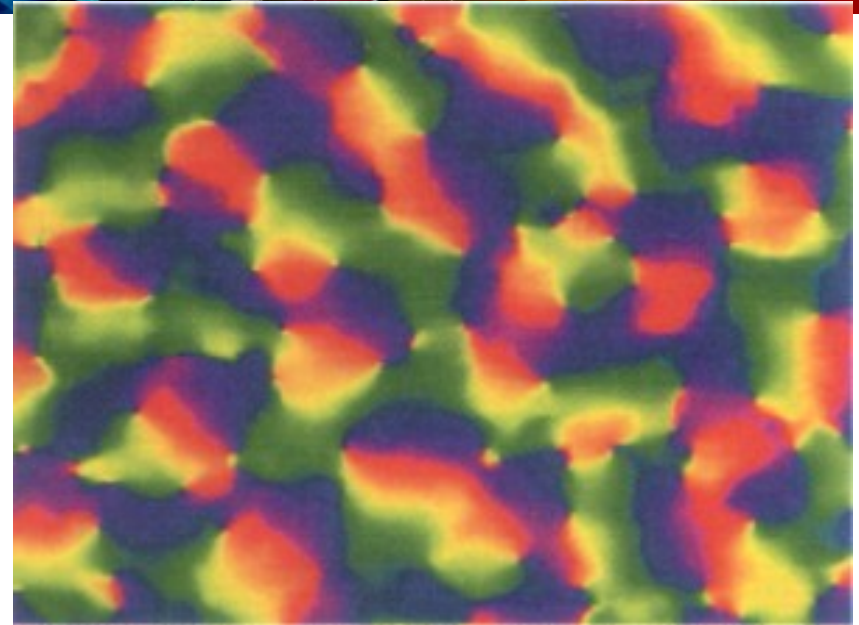
Topographic Maps

K Obermayer and GG Blasdel, Journal of Neuroscience, Vol 13, 4114-4129 (Monkey)

Y LeCun



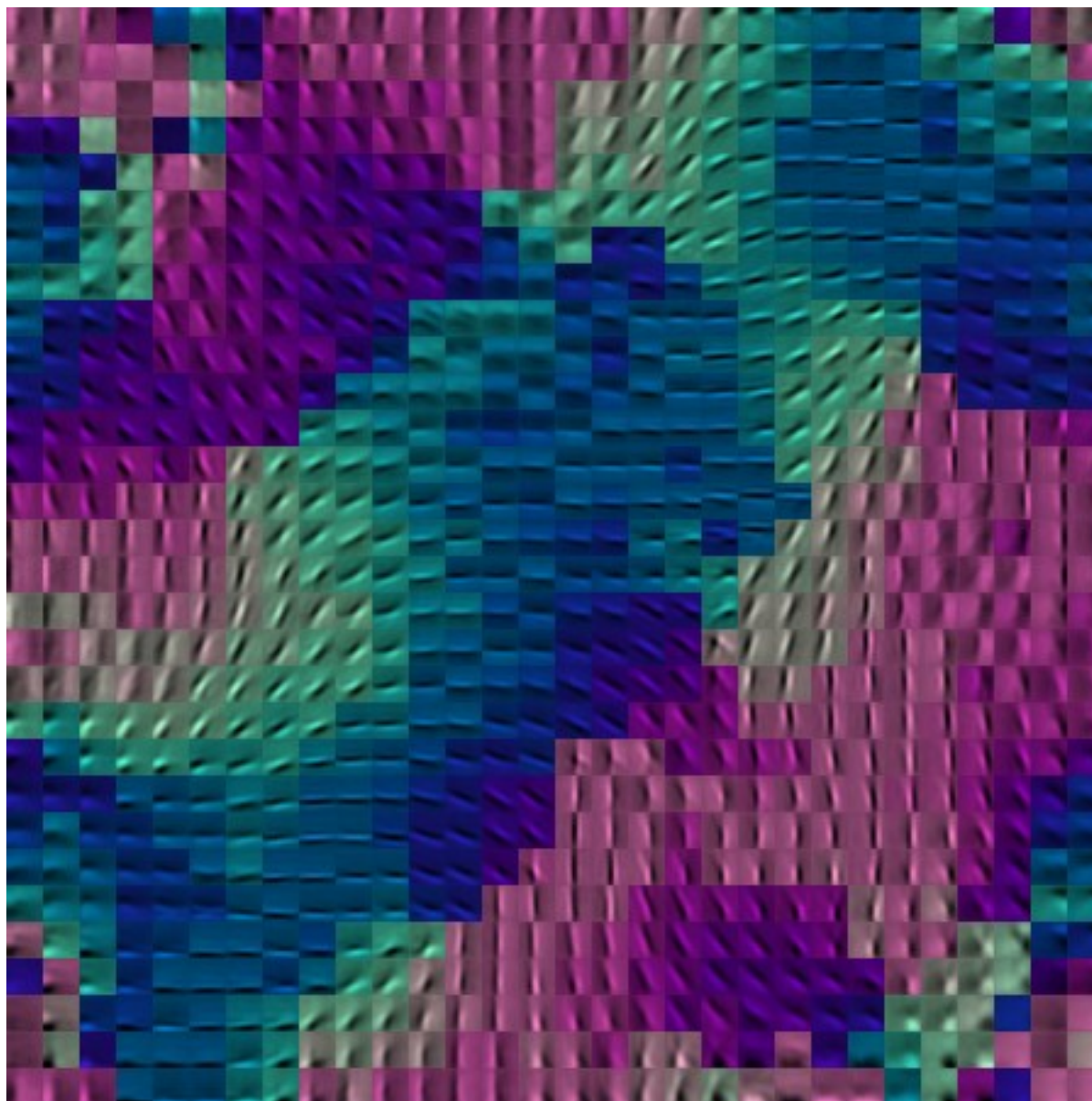
119x119 Image Input
100x100 Code
20x20 Receptive field size
 $\sigma=5$



Michael C. Crair, et. al. The Journal of Neurophysiology
Vol. 77 No. 6 June 1997 pp. 3381-3385 (Cat)

Image-level training, local filters but no weight sharing

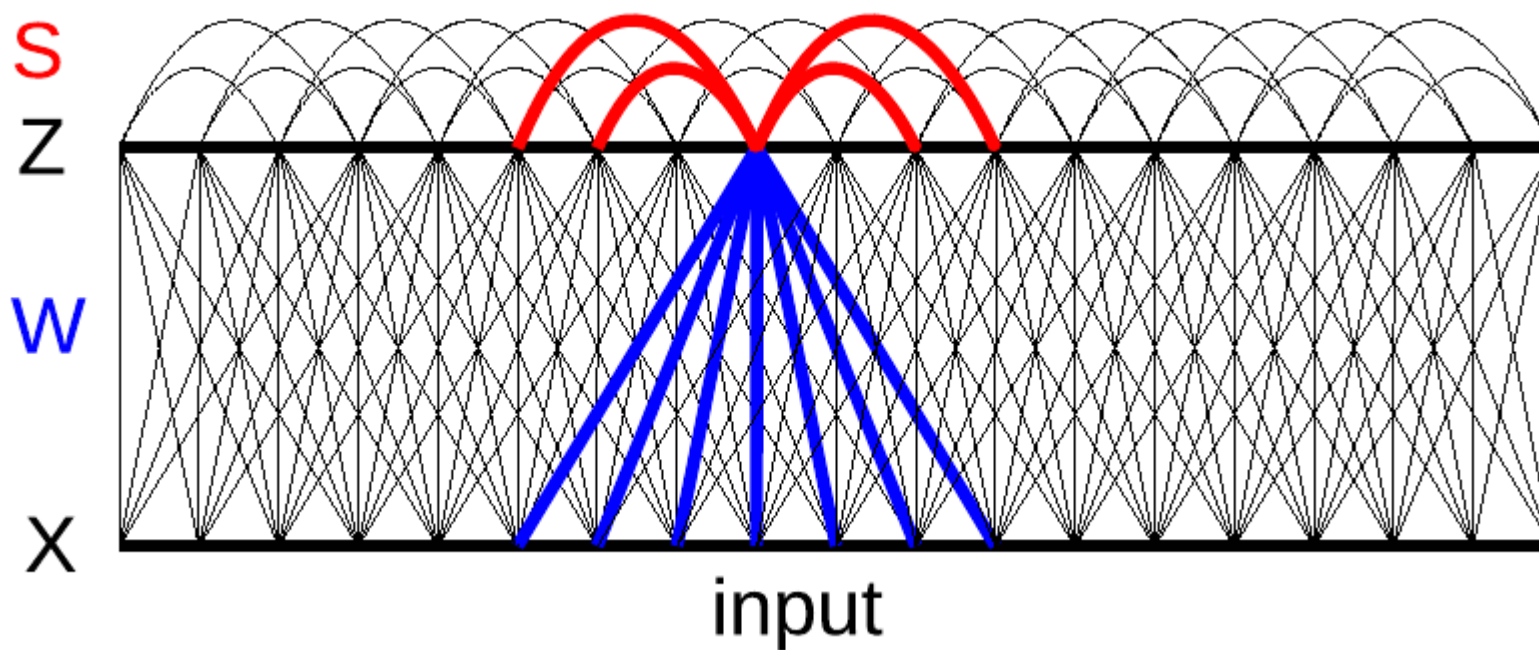
- Color indicates orientation (by fitting Gabors)



Invariant Features Lateral Inhibition

- Replace the L1 sparsity term by a lateral inhibition matrix
- Easy way to impose some structure on the sparsity

$$\min_{W, Z} \sum_{x \in X} \|Wz - x\|^2 + |z|^T S |z|$$

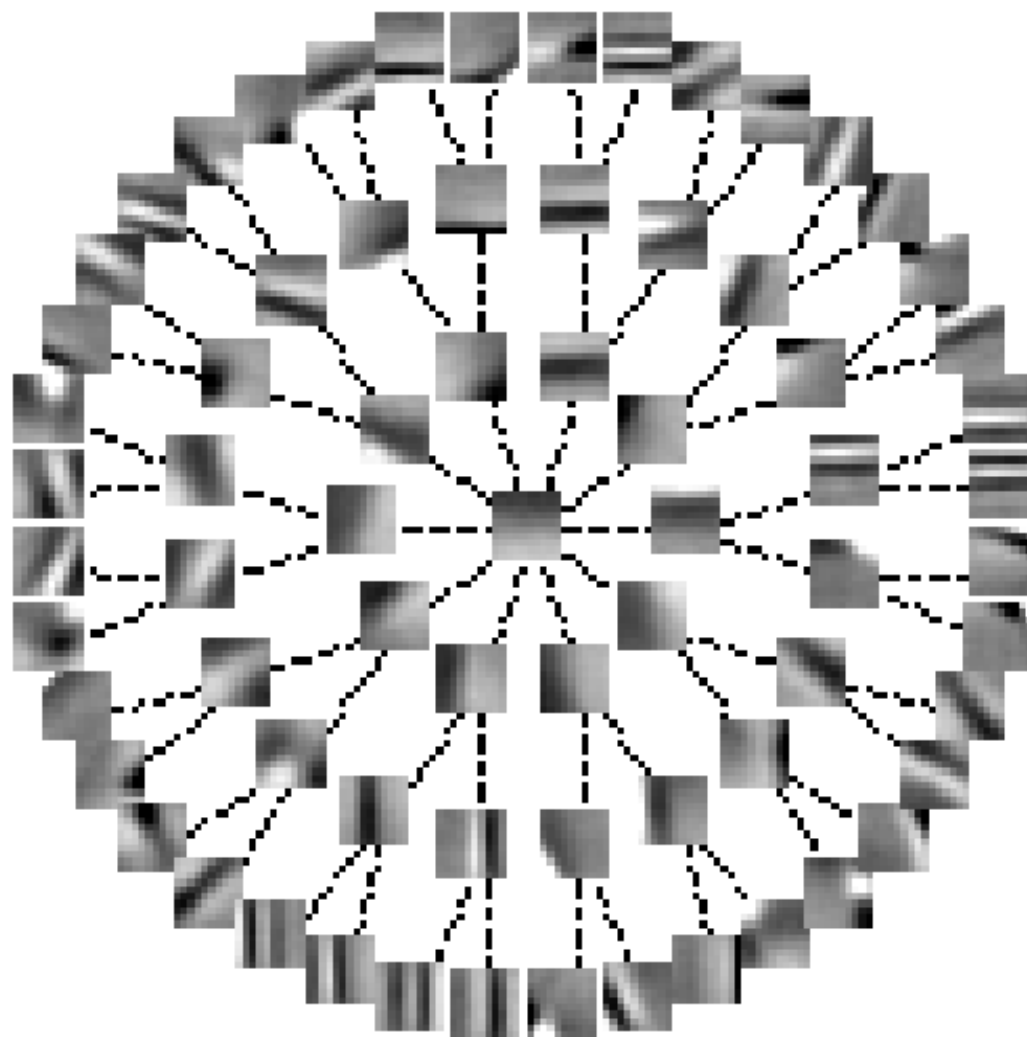


[Gregor, Szlam, LeCun NIPS 2011]

Invariant Features via Lateral Inhibition: Structured Sparsity

Y LeCun

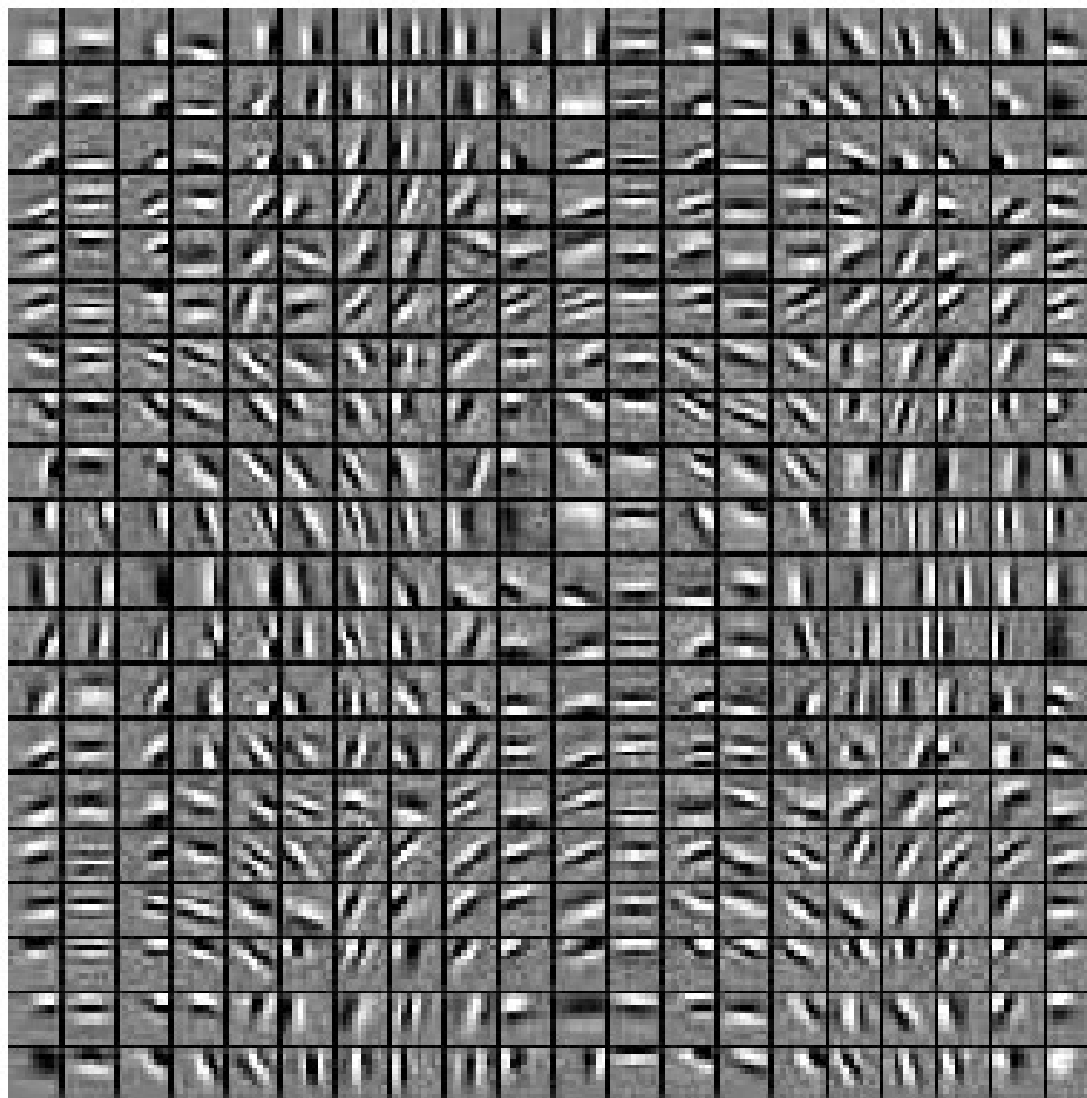
- Each edge in the tree indicates a zero in the S matrix (no mutual inhibition)
- S_{ij} is larger if two neurons are far away in the tree



Invariant Features via Lateral Inhibition: Topographic Maps

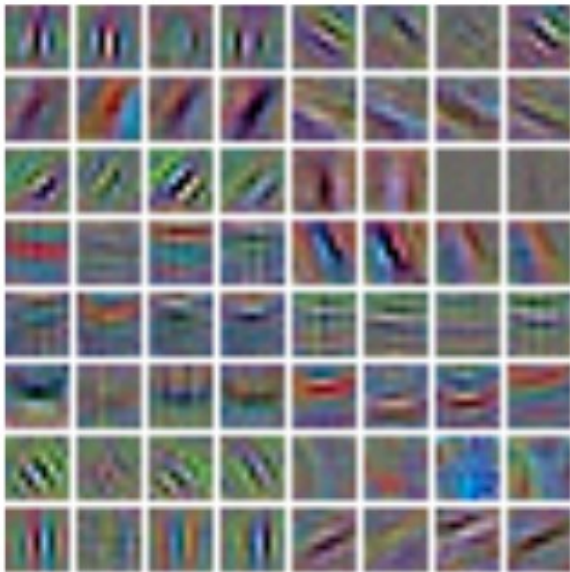
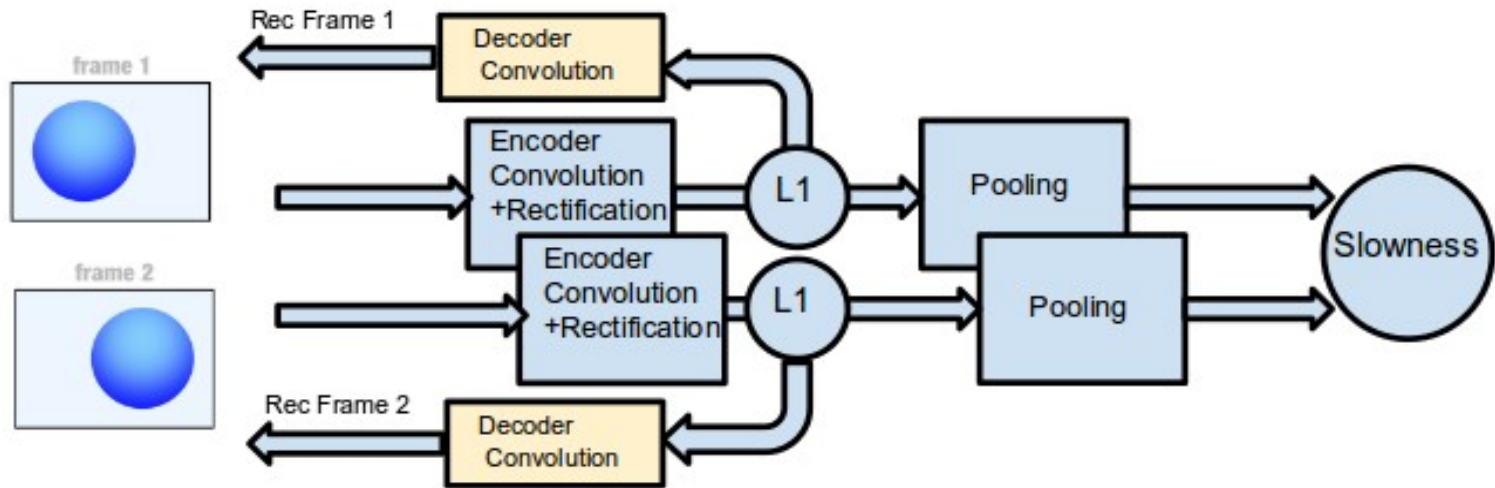
Y LeCun

- Non-zero values in S form a ring in a 2D topology
 - ▶ Input patches are high-pass filtered

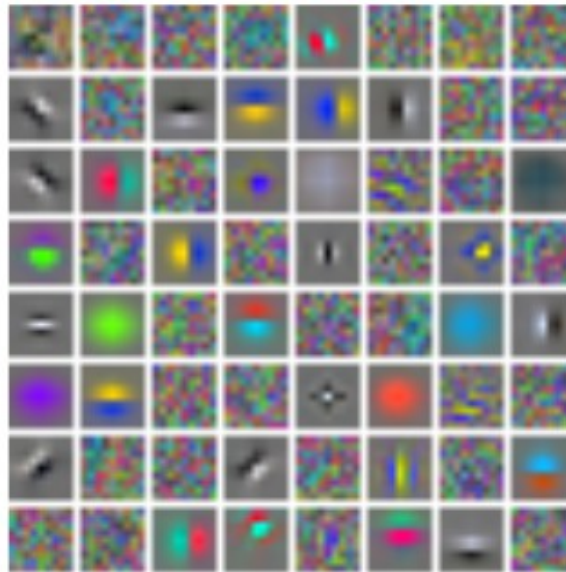


Sparse Auto-Encoder with "Slow Feature" Penalty

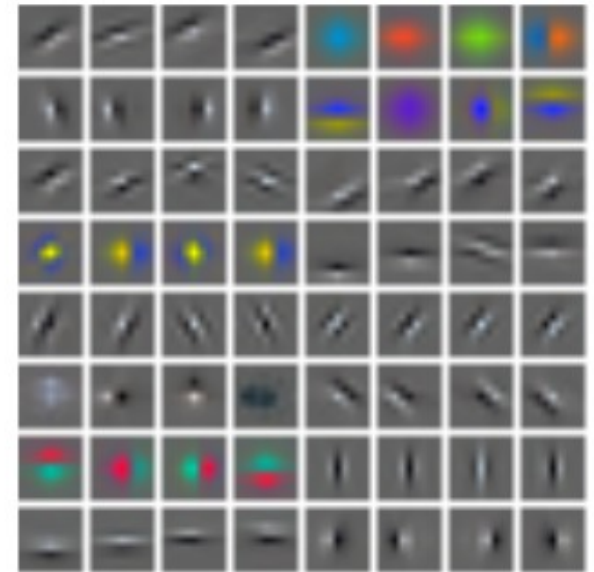
Y LeCun



Supervised filters CIFAR10



sparse conv. auto-encoder



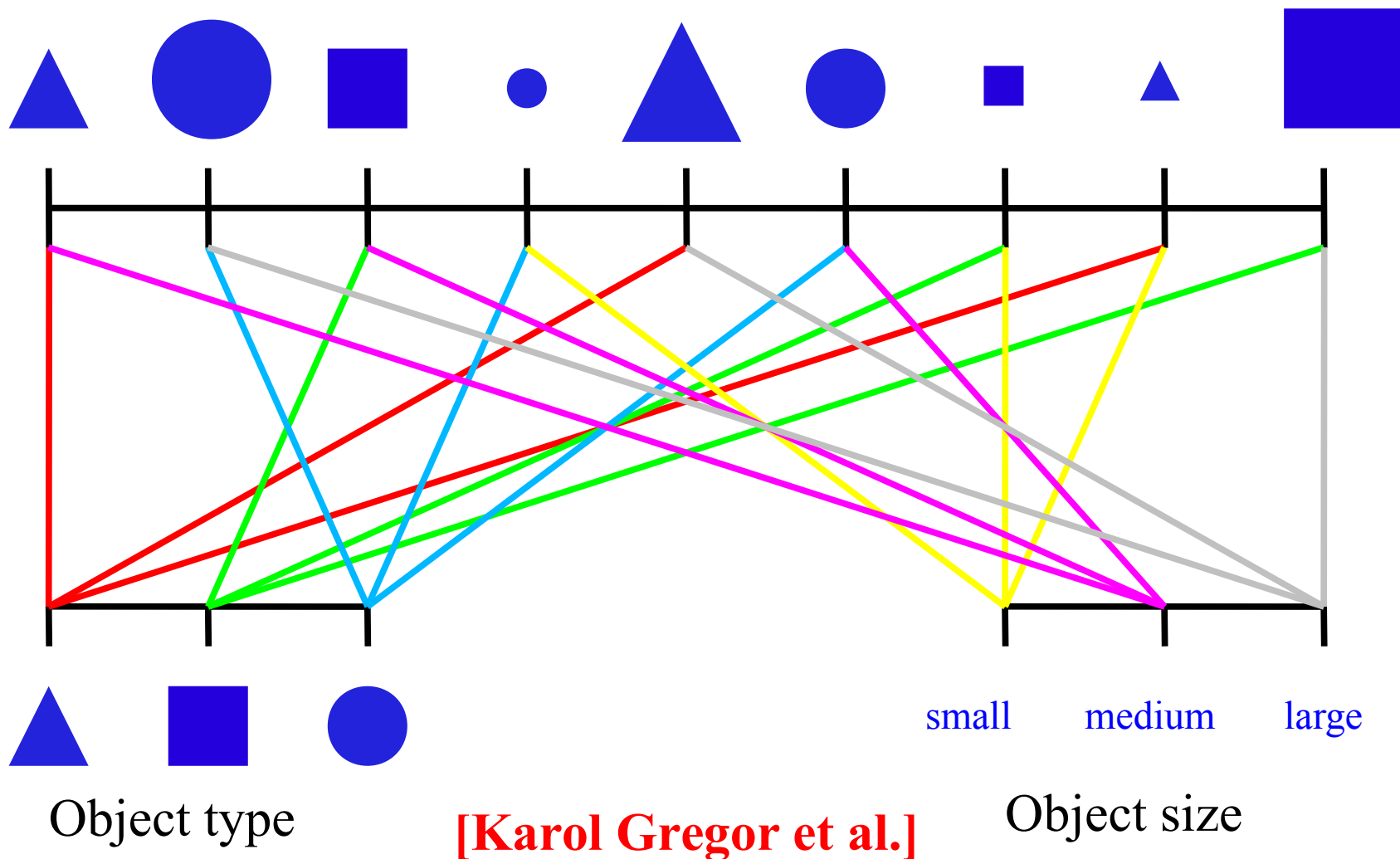
slow & sparse conv. auto-encoder
Trained on YouTube videos

[Goroshin et al. Arxiv:1412.6056]

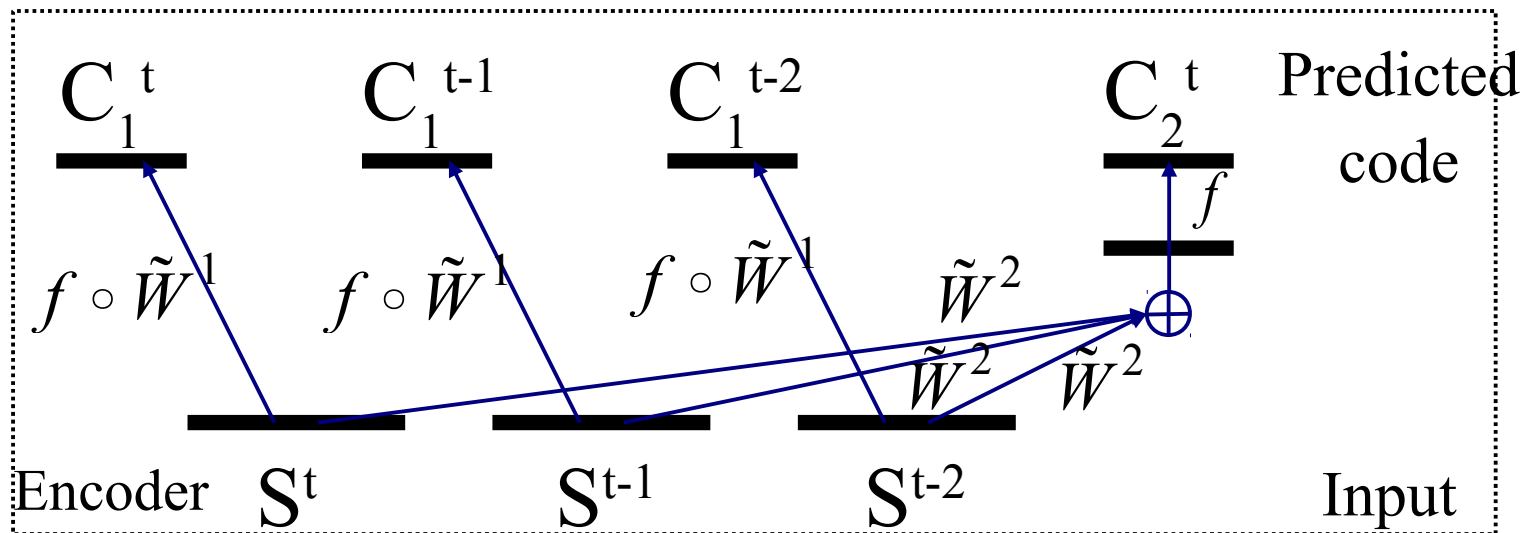
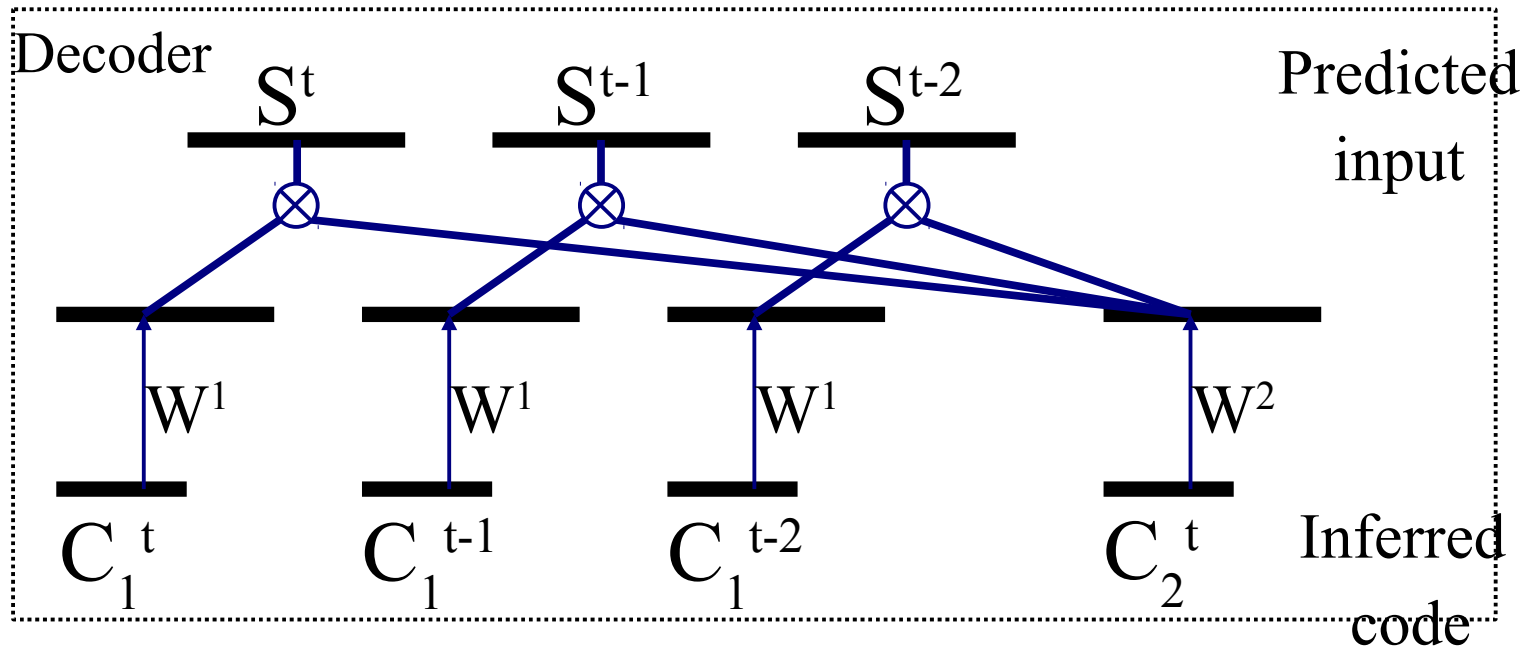
Invariant Features through Temporal Constancy

Y LeCun

- Object is **cross-product** of object type and instantiation parameters
 - ▶ Mapping units [Hinton 1981], capsules [Hinton 2011]

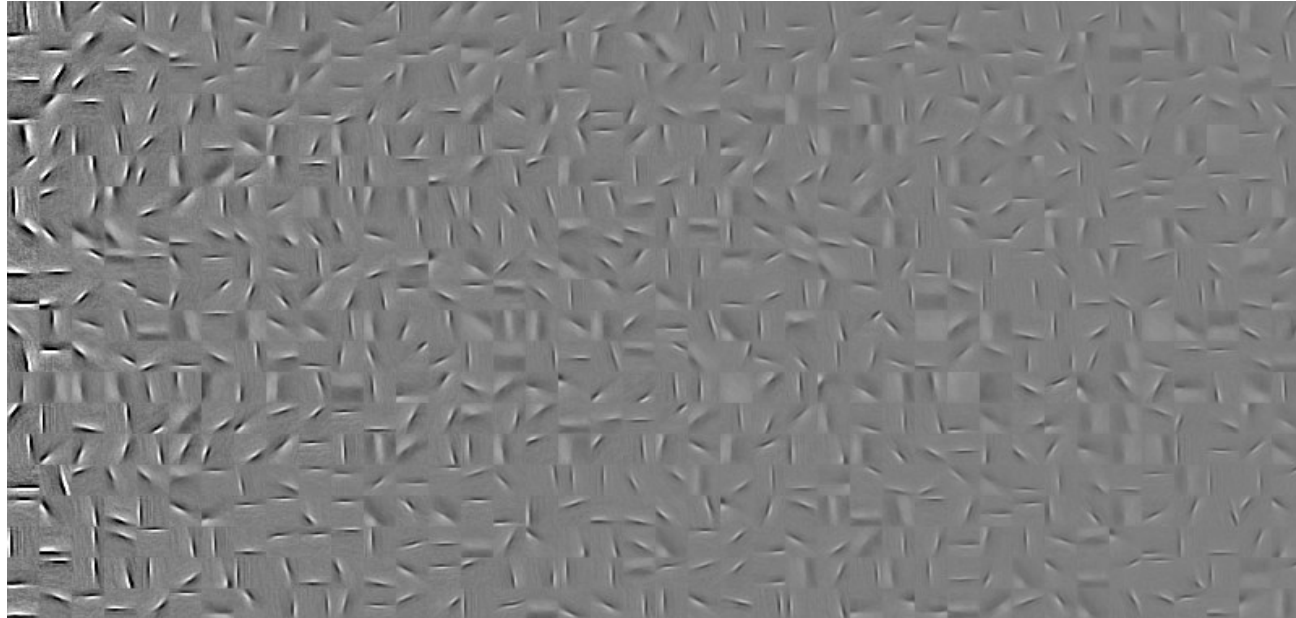


What-Where Auto-Encoder Architecture

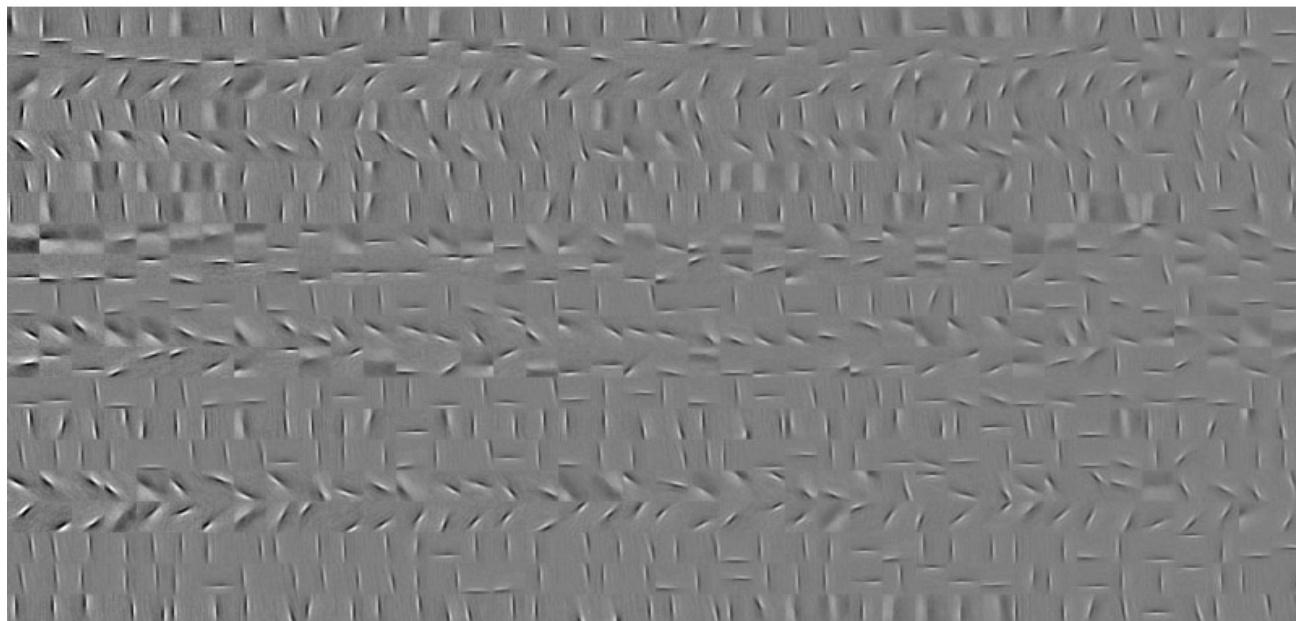


Low-Level Filters Connected to Each Complex Cell

C1
(where)



C2
(what)



Integrated Supervised & Unsupervised Learning

[Zhao, Mathieu, LeCun arXiv:1506.02351]

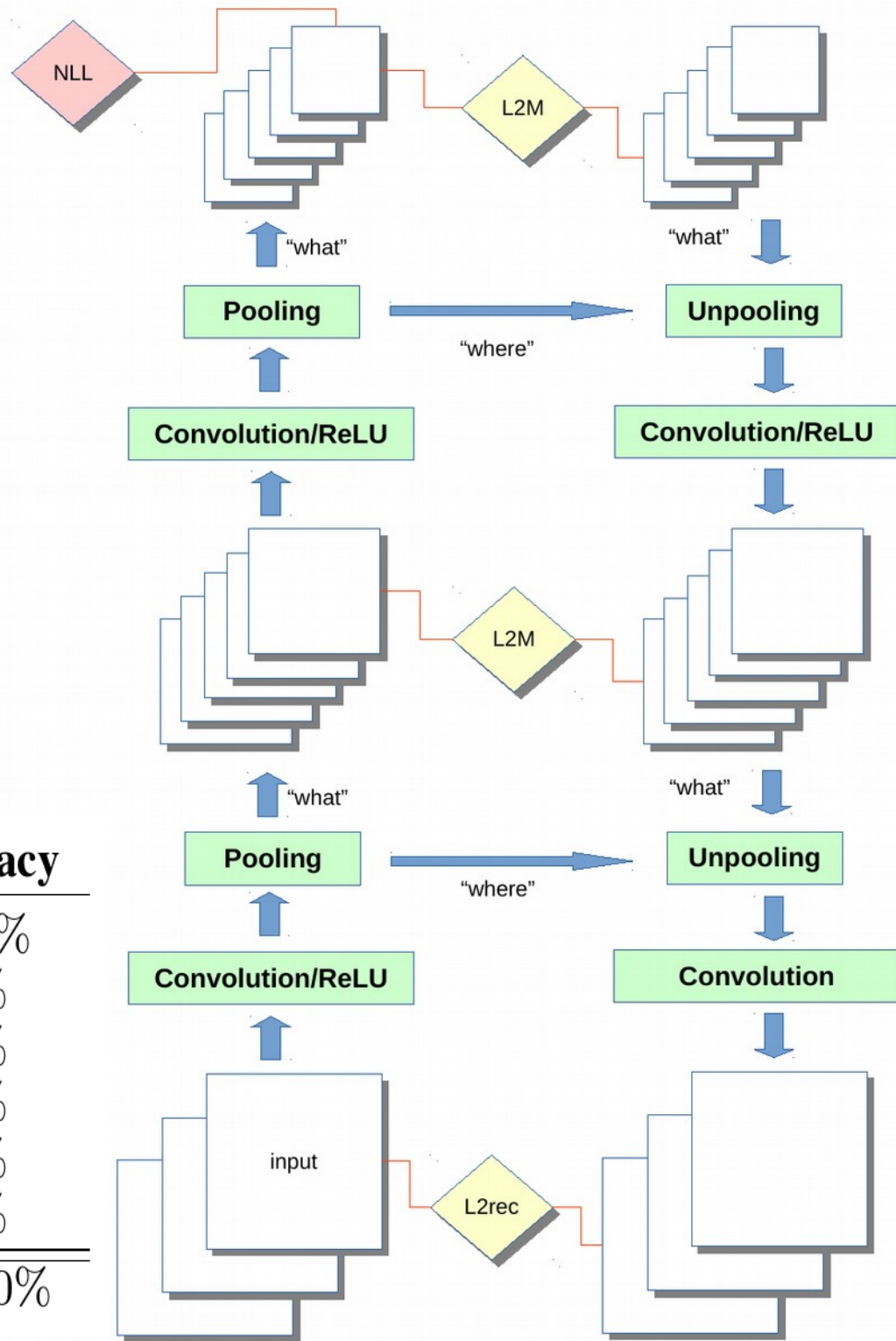
Stacked What-Where Auto-Encoder



model

accuracy

Convolutional Kernel Networks [18]	62.32%
HMP [1]	64.5%
NOMP [17]	67.9%
Multi-task Bayesian Optimization [31]	70.1%
Zero-bias ConvNets + ADCU [22]	70.2%
Exemplar ConvNets [2]	72.8%
WWAE-4layer	¹ 74.80%



The background features a complex, abstract composition of overlapping geometric shapes and lines. The top half is dominated by blue and cyan tones, with sharp, angular forms and a network of thin white lines. The bottom half transitions into warmer colors, including reds, oranges, and pinks, also with geometric patterns. A large, solid blue rectangle is centered horizontally and vertically, serving as a backdrop for the text.

The End

The bAbI Tasks

Questions that an AI system
ought to be able to answer

(1) Basic Factoid QA with Single Supporting Fact

Our first task consists of questions where a single supporting fact, previously given, provides the answer.

We test simplest cases of this, by asking for the location of a person.

A small sample of the task is thus:

John is in the playground.

Bob is in the office.

Where is John? **A:playground**

This kind of synthetic data was already used with MemNNs.

It can be considered the simplest case of some real world QA datasets such as in Fader et al., '13.

(2) Factoid QA with Two Supporting Facts

A harder task is to answer questions where two supporting statements have to be chained to answer the question:

John is in the playground.

Bob is in the office.

John picked up the football.

Bob went to the kitchen.

Where is the football? **A:playground**

Where was Bob before the kitchen?

A:office

E.g. to answer the first question *Where is the football?* both *John picked up the football* and *John is in the playground* are supporting facts.

Again, this kind of task was already used with MemNNs.

(2) *Shuffled Factoid QA with Two Supporting Facts*

- Note that, to show the difficulty of these tasks for a learning machine with no other knowledge we can shuffle the letters of the alphabet and produce equivalent datasets:

Sbdm ip im vdu yonrckblms.

Abf ip im vdu bhhigu.

Sbdm yigaus ly vdu hbbvfnoo.

Abf zumv vb vdu aivgdum.

Mduku ip vdu hbbvfnoo?

A:yonrckblms

Mduku znp Abf fuhbku vdu aivgdum?

A:bhhigu

(3) Factoid QA with Three Supporting Facts

Similarly, one can make a task with three supporting facts:

John picked up the apple.

John went to the office.

John went to the kitchen.

John dropped the apple.

Where was the apple before the kitchen?

A:office

The first three statements are all required to answer this.

(4) Two Argument Relations: Subject vs. Object

To answer questions the ability to differentiate and recognize subjects and objects is crucial.

We consider the extreme case: sentences feature re-ordered words:

The office is north of the bedroom.

The bedroom is north of the bathroom.

What is north of the bedroom? **A:office**

What is the bedroom north of?

A:bathroom

Note that the two questions above have exactly the same words, but in a different order, and different answers.

So a bag-of-words will not work.

(5) Three Argument Relations

- Similarly, sometimes one needs to differentiate three separate arguments, such as in the following task:

Mary gave the cake to Fred.

Fred gave the cake to Bill.

Jeff was given the milk by Bill.

Who gave the cake to Fred? A:Mary

Who did Fred give the cake to? A:Bill

What did Jeff receive? A:milk

Who gave the milk? A:Bill

The last question is potentially the hardest for a learner as the first two can be answered by providing the actor that is not mentioned in the question.

(6) Yes/No Questions

- This task tests, in the simplest case possible (with a single supporting fact) the ability of a model to answer true/false type questions:

John is in the playground.

Daniel picks up the milk.

Is John in the classroom? **A:no**

Does Daniel have the milk? **A:yes**

(7) Counting

- This task tests the ability of the QA system to perform simple counting operations, by asking about the number of objects with a certain property:

Daniel picked up the football.

Daniel dropped the football.

Daniel got the milk.

Daniel took the apple.

How many objects is Daniel holding?

A:two

(8) Lists/Sets

- While many of our tasks are designed to have single word answers for simplicity, this task tests the ability to produce a set of single word answers in the form of a list:

Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
What is Daniel holding? **A:milk,football**

The task above can be seen as a QA task related to database search.

Note that we could also consider the following question types:

Intersection: *Who is in the park carrying food?*

Union: *Who has milk or cookies?*

Set difference: *Who is in the park apart from Bill?*

However, we leave those for future work.

(9) Simple Negation

- We test one of the simplest types of negation, that of supporting facts that imply a statement is false:

Sandra travelled to the office.

Fred is no longer in the office.

Is Fred in the office? A:no

Is Sandra in the office? A:yes

The Yes/No task (6) is a prerequisite.

Slightly harder: we could add things like “Is Fred with Sandra?”

(10) Indefinite knowledge

- This task tests if we can model statements that describe possibilities rather than certainties:

John is either in the classroom or the playground.

Sandra is in the garden.

Is John in the classroom? A:maybe

Is John in the office? A:no

The Yes/No task (6) is a prerequisite.

Slightly harder: we could add things like “Is John with Sandra?”

(11) Basic Coreference

- This task tests the simplest type of coreference, that of detecting the nearest referent, for example:

Daniel was in the kitchen.
Then he went to the studio.
Sandra was in the office.
Where is Daniel? **A:studio**

Next level of difficulty: flip order of last two statements, and it has to learn the difference between 'he' and 'she'.

Much harder difficulty: adapt a real coref dataset into a question answer format.

(12) Conjunction

- This task tests referring to multiple subjects in a single statement, for example:

Mary and Jeff went to the kitchen.

Then Jeff went to the park.

Where is Mary? **A:kitchen**

(13) Compound Coreference

- This task tests coreference in the case where the pronoun can refer to multiple actors:

Daniel and Sandra journeyed to the office.

Then they went to the garden.

Sandra and John travelled to the kitchen.

After that they moved to the hallway.

Where is Daniel? **A:garden**

(14) Time manipulation

- While our tasks so far have included time implicitly in the *order* of the statements, this task tests understanding the use of time expressions within the statements:

In the afternoon Julie went to the park.
Yesterday Julie was at school.
Julie went to the cinema this evening.
Where did Julie go after the park?

A:cinema

Much harder difficulty: adapt a real time expression labeling dataset into a question answer format, e.g. Uzzaman et al., '12.

(15) Basic Deduction

- This task tests basic deduction via inheritance of properties:

Sheep are afraid of wolves.

Cats are afraid of dogs.

Mice are afraid of cats.

Gertrude is a sheep.

What is Gertrude afraid of? **A:wolves**

Deduction should prove difficult for MemNNs because it effectively involves search, although our setup might be simple enough for it.

(16) Basic Induction

- This task tests basic induction via inheritance of properties:

Lily is a swan.

Lily is white.

Greg is a swan.

What color is Greg? **A:white**

Induction should prove difficult for MemNNs because it effectively involves search, although our setup might be simple enough for it.

(17) Positional Reasoning

- This task tests spatial reasoning, one of many components of the classical SHRDLU system:

The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.

Is the red sphere to the right of the blue square? **A:yes**

Is the red square to the left of the triangle?
A:yes

The Yes/No task (6) is a prerequisite.

(18) Reasoning about size

- This task requires reasoning about relative size of objects and is inspired by the commonsense reasoning examples in the Winograd schema challenge:

The football fits in the suitcase.

The suitcase fits in the cupboard.

The box of chocolates is smaller than the football.

Will the box of chocolates fit in the suitcase?

Tasks 3 (yes) and 6 (Yes/No) are prerequisites.

(19) Path Finding

- In this task the goal is to find the path between locations:

The kitchen is north of the hallway.

The den is east of the hallway.

How do you go from den to kitchen?

A:west,north

This is going to prove difficult for MemNNs because it effectively involves search.

(The original MemNN can also output only one word \rightarrow)

(20) Reasoning about Agent's Motivations

- This task tries to ask *why* an agent performs a certain action.
- It addresses the case of actors being in a given state (hungry, thirsty, tired, ...) and the actions they then take:

John is hungry.

John goes to the kitchen.

John eats the apple.

Daniel is hungry.

Where does Daniel go? **A:kitchen**

Why did John go to the kitchen? **A:hungry**

One way of solving these tasks: Memory Networks!!

MemNNs have four component networks (which may or may not have shared parameters):

- **I:** (input feature map) this converts incoming data to the internal feature representation.
- **G:** (generalization) this updates memories given new input.
- **O:** this produces new output (in featurerepresentation space) given the memories.
- **R:** (response) converts output O into a response seen by the outside world.



Experiments

- **Protocol:** 1000 training QA pairs, 1000 for test.

“Weakly supervised” methods:

- Ngram baseline, uses bag of Ngram features from sentences that share a word with the question.
- LSTM

Fully supervised methods (for train data, have supporting facts labeled):

- Original MemNNs, *and all our variants.*

Table 1. Test accuracy (%) on our 20 Tasks for various methods (training with 1000 training examples on each). Our proposed extensions to MemNNs are in columns 5-9: with adaptive memory (AM), N -grams (NG), nonlinear matching function (NL), multilinear matching (ML), and combinations thereof. Bold numbers indicate tasks where our extensions achieve $\geq 95\%$ accuracy but the original MemNN model of (Weston et al., 2014) did not. The last two columns (10-11) give extra analysis of the MemNN method. Column 10 gives the amount of training data for each task needed to obtain $\geq 95\%$ accuracy, or *FAIL* if this is not achievable with 1000 training examples. The final column gives the accuracy when training on all data at once, rather than separately.

TASK	N -grams	LSTM	MemNN (Weston et al., 2014)	MemNN ADAPTIVE MEMORY	MemNN AM + N-GRAMS	MemNN AM + NONLINEAR	MemNN AM + MULTILINEAR	MemNN AM + NG + NL	No. of ex. req. ≥ 95	Multi-Task Training
3.1 - Single Supporting Fact	36	50	100	100	100	100	100	100	250 ex.	100
3.2 - Two Supporting Facts	2	20	100	100	100	100	100	100	500 ex.	100
3.3 - Three Supporting Facts	7	20	20	100	99	100	99	100	500 ex.	98
3.4 - Two Arg. Relations	50	61	71	69	100	73	100	100	500 ex.	80
3.5 - Three Arg. Relations	20	70	83	83	86	86	98	98	1000 ex.	99
3.6 - Yes/No Questions	49	48	47	52	53	100	100	100	500 ex.	100
3.7 - Counting	52	49	68	78	86	83	90	85	FAIL	86
3.8 - Lists/Sets	40	45	77	90	88	94	91	91	FAIL	93
3.9 - Simple Negation	62	64	65	71	63	100	100	100	500 ex.	100
3.10 - Indefinite Knowledge	45	44	59	57	54	97	96	98	1000 ex.	98
3.11 - Basic Coreference	29	72	100	100	100	100	100	100	250 ex.	100
3.12 - Conjunction	9	74	100	100	100	100	100	100	250 ex.	100
3.13 - Compound Coreference	26	94	100	100	100	100	100	100	250 ex.	100
3.14 - Time Reasoning	19	27	99	100	99	100	99	99	500 ex.	99
3.15 - Basic Deduction	20	21	74	73	100	77	100	100	100 ex.	100
3.16 - Basic Induction	43	23	27	100	100	100	100	100	100 ex.	94
3.17 - Positional Reasoning	46	51	54	46	49	57	60	65	FAIL	72
3.18 - Size Reasoning	52	52	57	50	74	54	89	95	1000 ex.	93
3.19 - Path Finding	0	8	0	9	3	15	34	36	FAIL	19
3.20 - Agent's Motivations	76	91	100	100	100	100	100	100	250 ex.	100
Mean Performance	34	49	75	79	83	87	93	93		92

Action Recognition Results

	Method	Accuracy (%)
Baselines	Imagenet	68.8
	iDT	76.2
Use raw pixel inputs	Deep networks [19]	65.4
	Spatial stream network [36]	72.6
	LRCN [7]	71.1
	LSTM composite model [39]	75.8
	C3D (1 net)	82.3
	C3D (3 nets)	85.2
Use optical flows	iDT with Fisher vector [31]	87.9
	Temporal stream network [36]	83.7
	Two-stream networks [36]	88.0
	LRCN [7]	82.9
	LSTM composite model [39]	84.3
	Multi-skip feature stacking [26]	89.1
	C3D (3 nets) + iDT	90.4