


Deconvolutional Networks

Matthew D. Zeiler
Graham W. Taylor
Rob Fergus

Dept. of Computer Science, Courant Institute,
New York University

Matt Zeiler

Overview

- "Generative" image model
- Convolutional form of sparse coding
- Pooling with latent variables (what/where)
  - Integrated into cost function (differentiable)
- Learn features for object recognition

Talk Overview

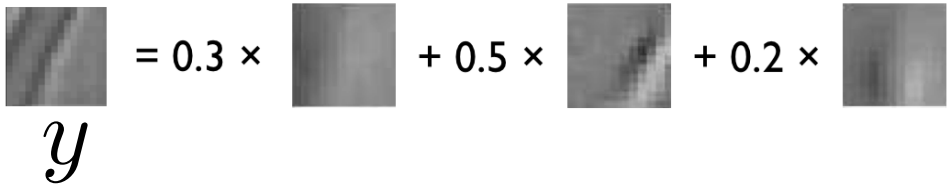
- Single layer
  - Convolutional Sparse Coding
  - Gaussian Pooling
- Multiple layers
  - Multi-layer inference
  - Filter learning
- Related work
- Experiments

Talk Overview

- Single layer
  - Convolutional Sparse Coding
  - Gaussian Pooling
- Multiple layers
  - Multi-layer inference
  - Filter learning
- Related work
- Experiments

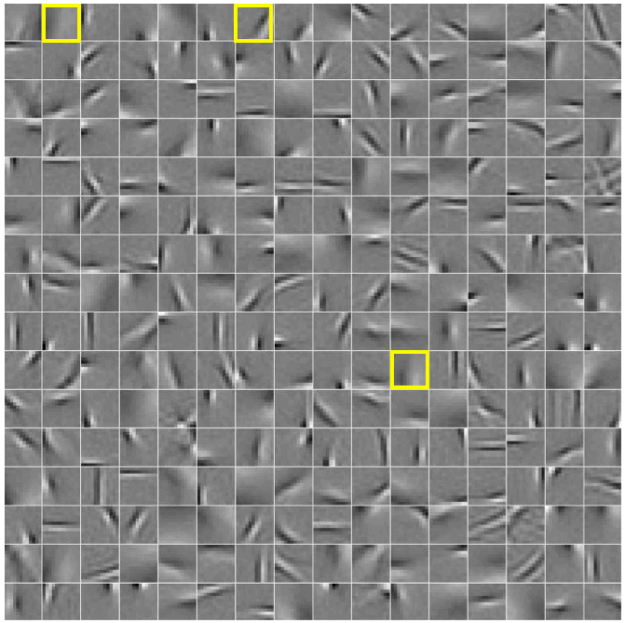
Recap: Sparse Coding (Patch-based)

- Over-complete linear decomposition of input y using dictionary D



y

$$C(y, D) = \underset{p}{argmin} \frac{\lambda}{2} \|Dp - y\|_2^2 + |p|_1$$



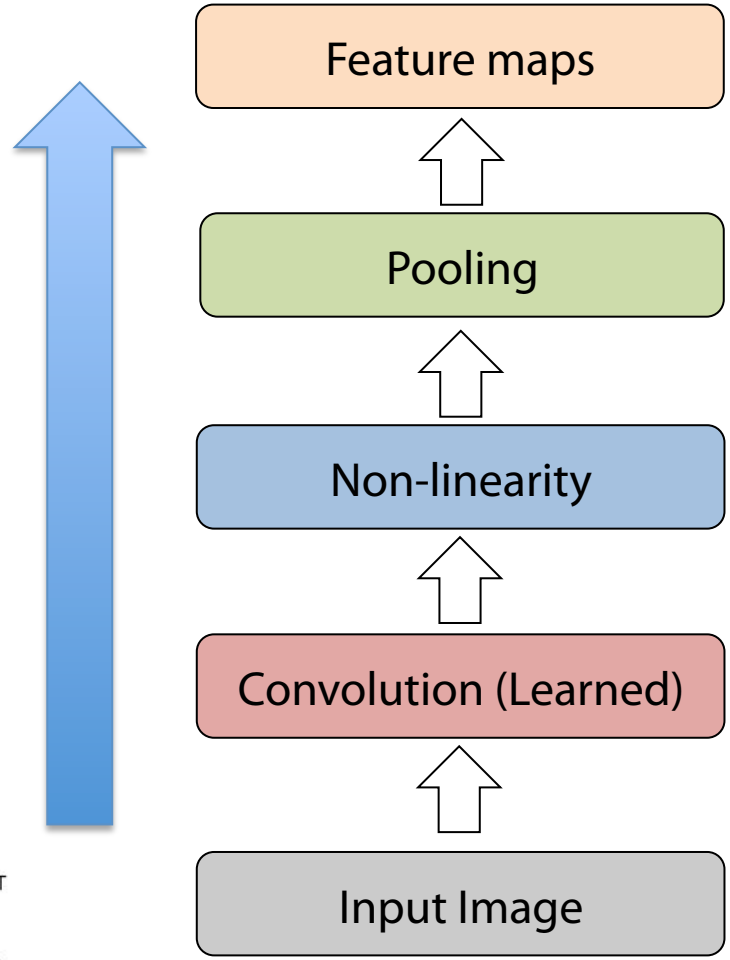
Dictionary D

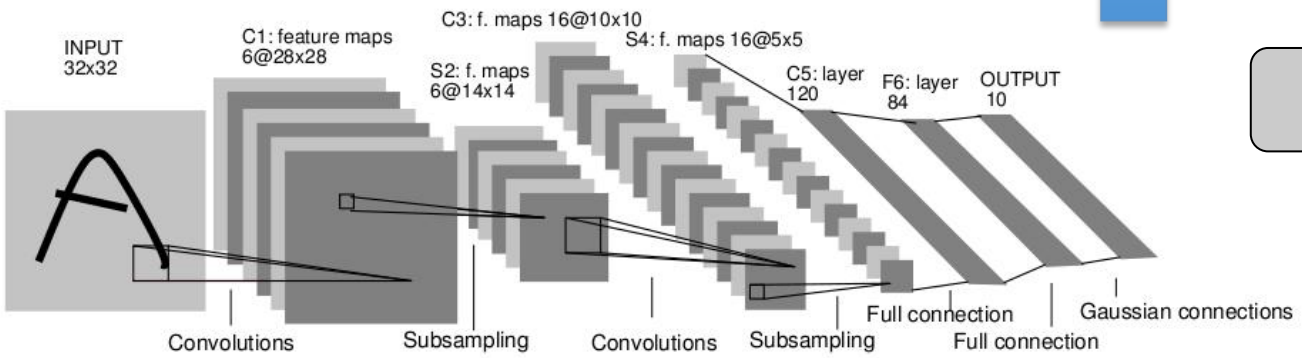
- ℓ_1 regularization yields solutions with few non-zero elements:

$$p = [0, 0.3, 0, \ldots, 0.5, \ldots, 0.2, \ldots, 0]$$

Convolutional Networks

- Feed-forward:
  - Convolve input
  - Non-linearity
  - Pooling
- Supervised
- Encoder-only



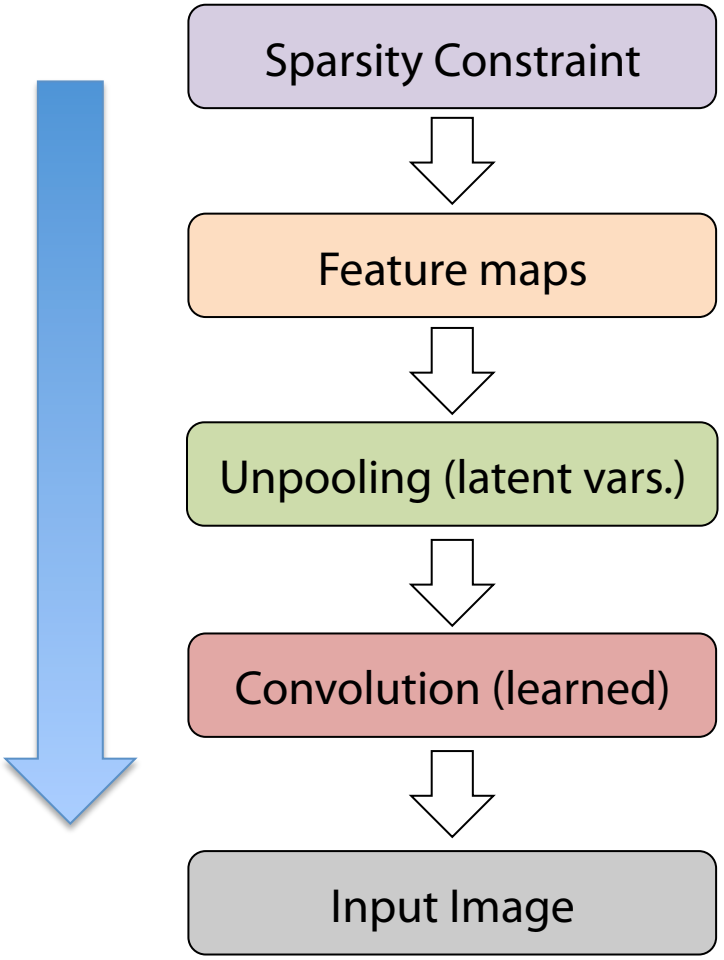


LeCun et al. 1998

Deconvolutional Networks

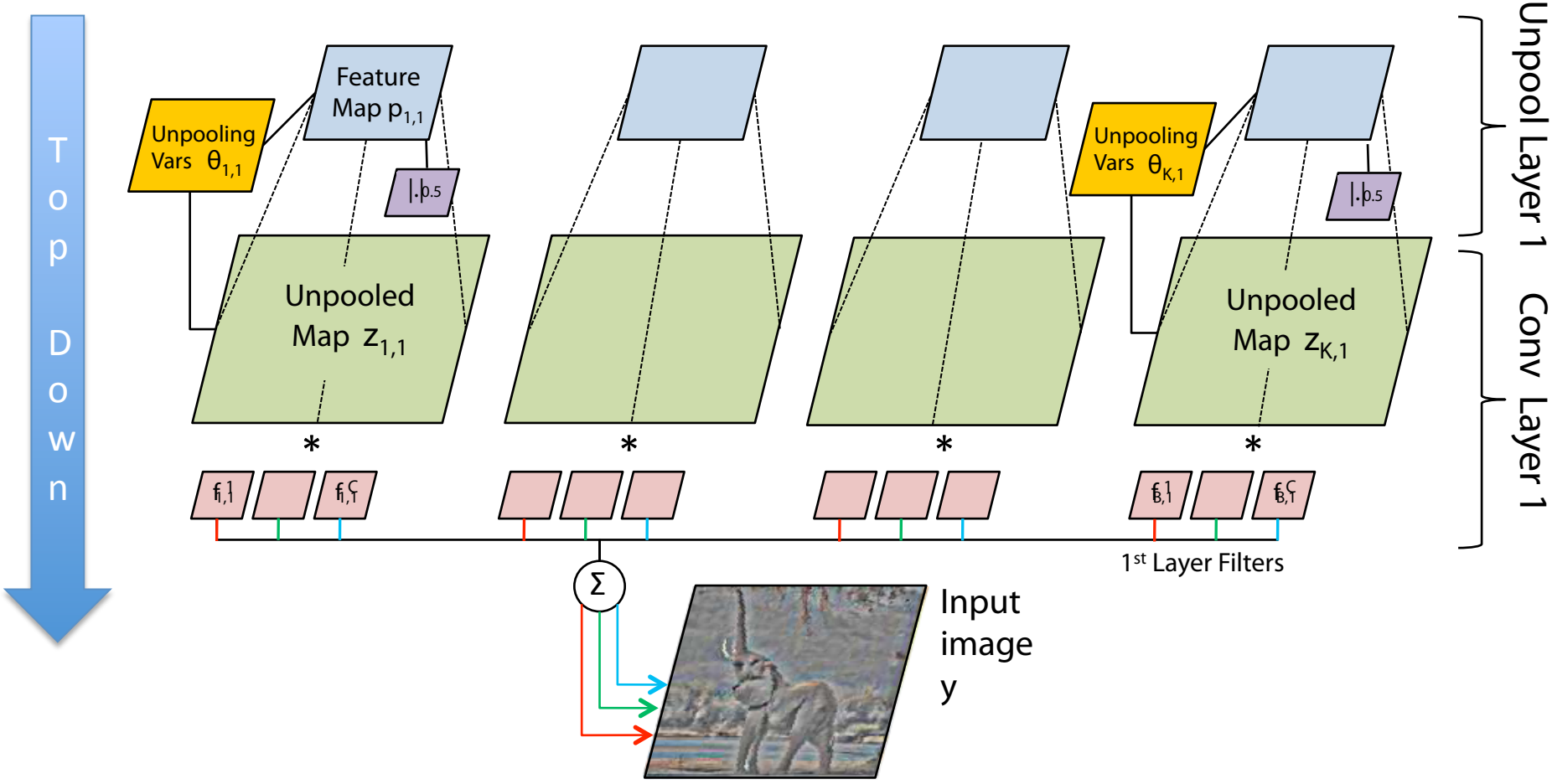
[Zeiler et al. CVPR'10, ICCV'11]

- Feed-back:
  - Unpool feature maps
    - Using inferred latent variables
  - Convolve unpooled maps
    - Learned filters
- Unsupervised
  - Must reconstruct input
  - Sparsity constraint
- Decoder-only
  - Have to infer features



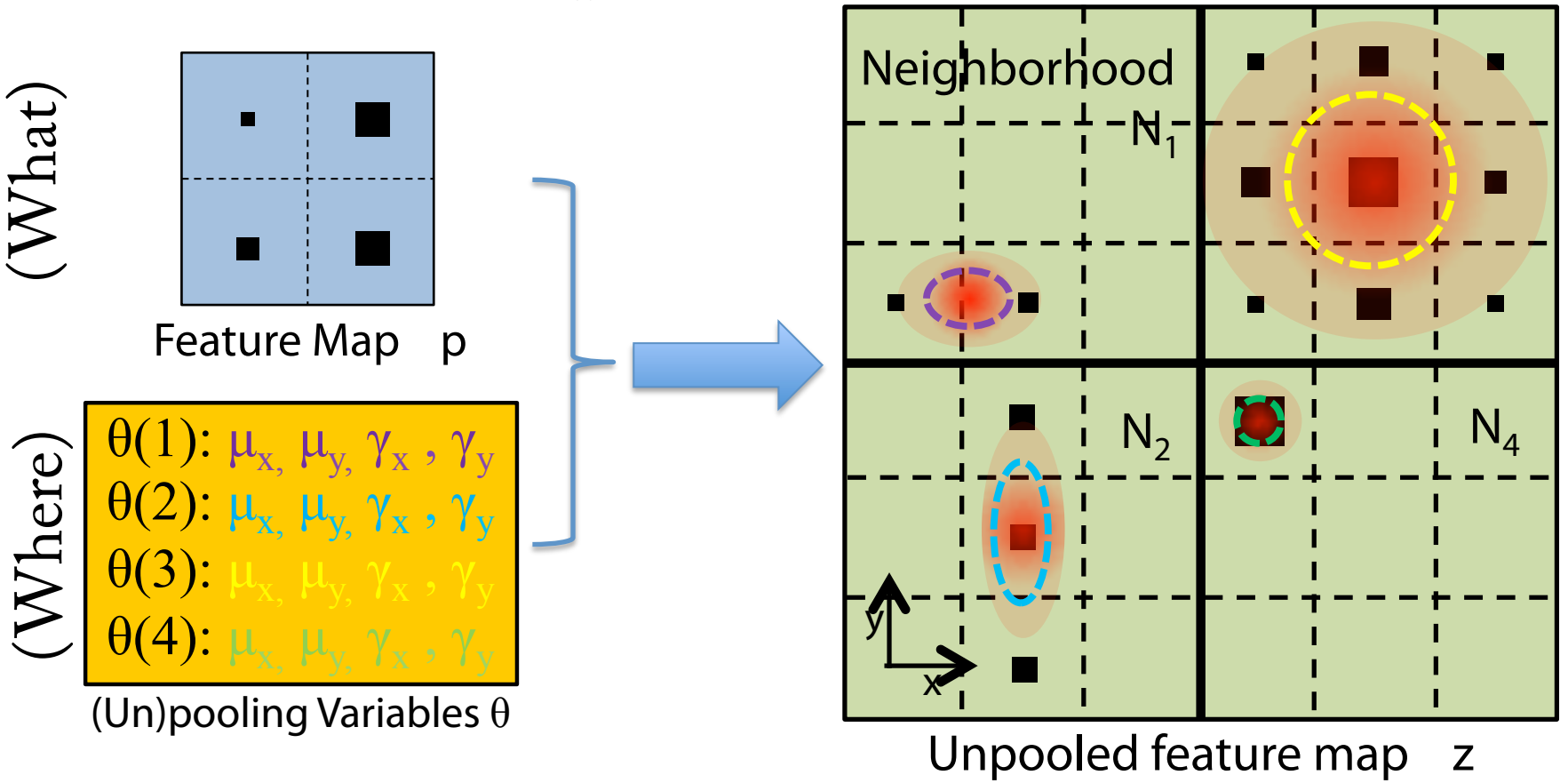
Single Layer Architecture

- Decomposition of input image
  - Over-complete → per-element sparsity constraint

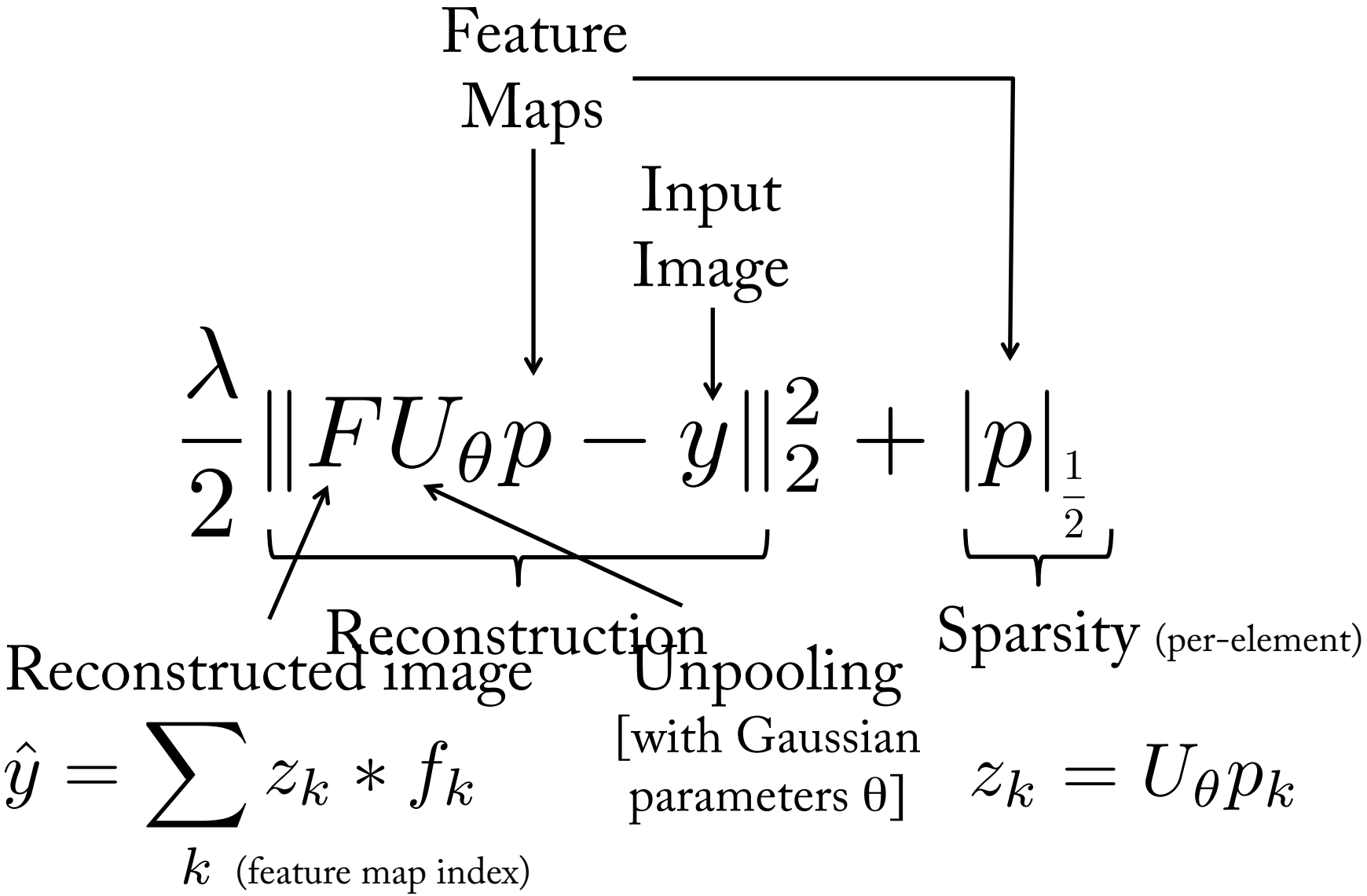


Gaussian Unpooling

- Each unpooling region has its own 2D Gaussian
- Gaussian weights scaled by feature map activation
- Differentiable representation



Single Layer Cost Function



Single Layer Cost Function

Learned (share across all images)

Feature Maps

Input Image

Infer for each image

$$\frac{\lambda}{2}\|FU_\theta p - y\|_2^2 + |p|_{\frac{1}{2}}$$

Reconstructed image

$$\hat{y} = \sum_k z_k * f_k$$

k (feature map index)

Unpooling [with Gaussian parameters θ]

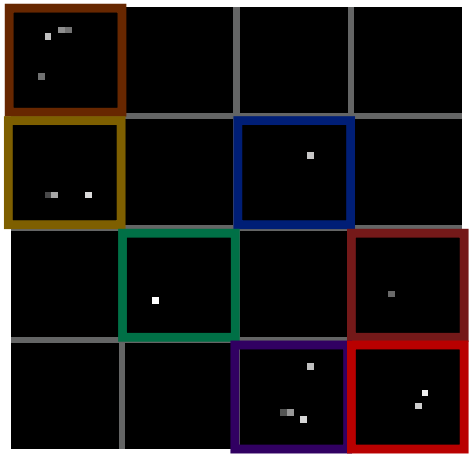
$$z_k = U_\theta p_k$$

Single Layer Inference

- Feature Maps p (What)
  - Fix convolution filters f and pooling variables θ
  - Convolutional form of sparse coding
  - Use ISTA [Beck & Teboulle SIAM J. Imaging Sciences 2009]:
    - Gradient step on reconstruction term (Quadratic)
    - Gradient step on sparsity term
    - Project to be non-negative

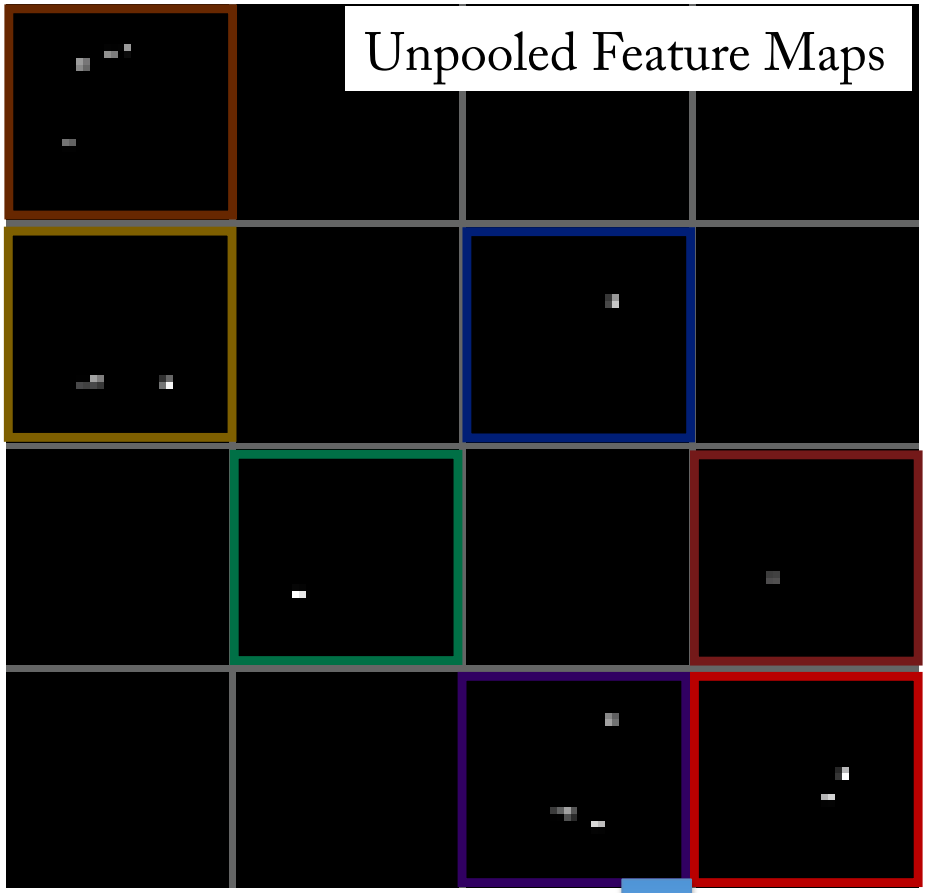
- Pooling variables θ (Where)
  - Fix convolution filters f and feature maps p
  - Chain rule of derivatives to update mean & precision of each Gaussian pooling neighborhood

Single Layer Example



16 Feature Maps

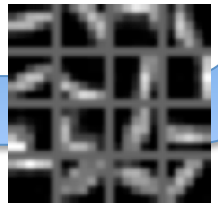
Unpooling



Input Image

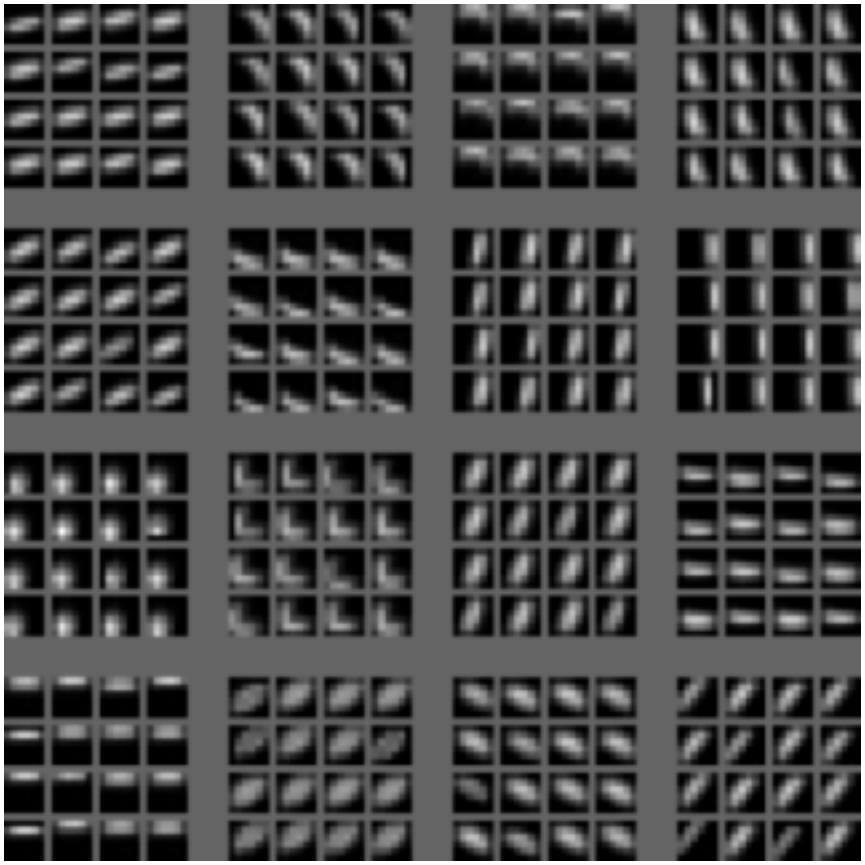
Reconstruction

Convolution
& Sum

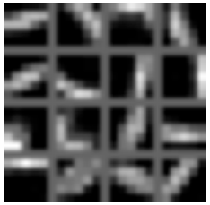


Filters

Effect of Pooling Variables



Pixel-space projections of sample feature map activations



Filter coefficients

Reconstruction Examples

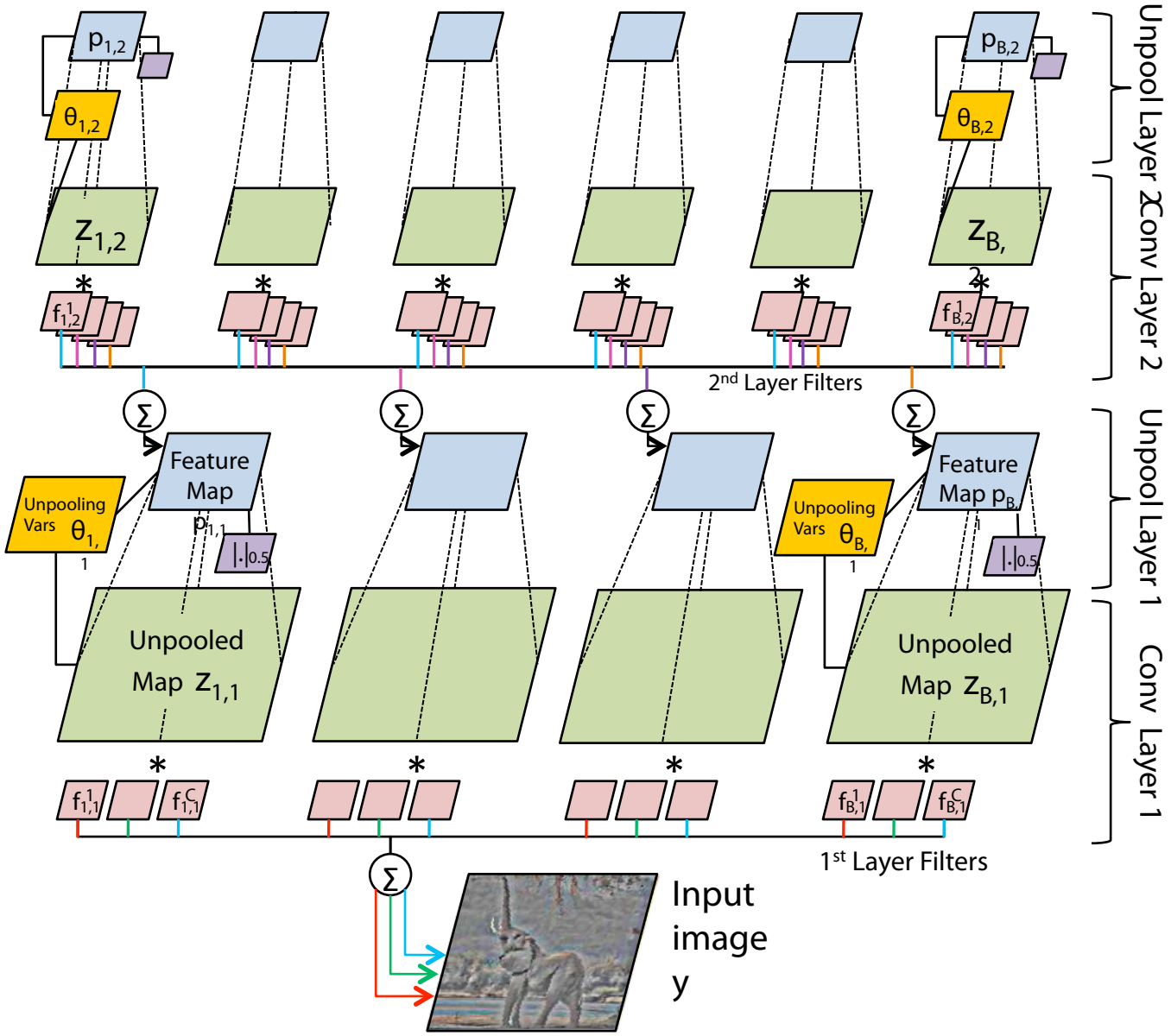
Talk Overview

- Single layer
  - Convolutional Sparse Coding
  - Gaussian Pooling
- Multiple layers
  - Multi-layer inference
  - Filter learning
- Related work
- Experiments

Stacking the Layers

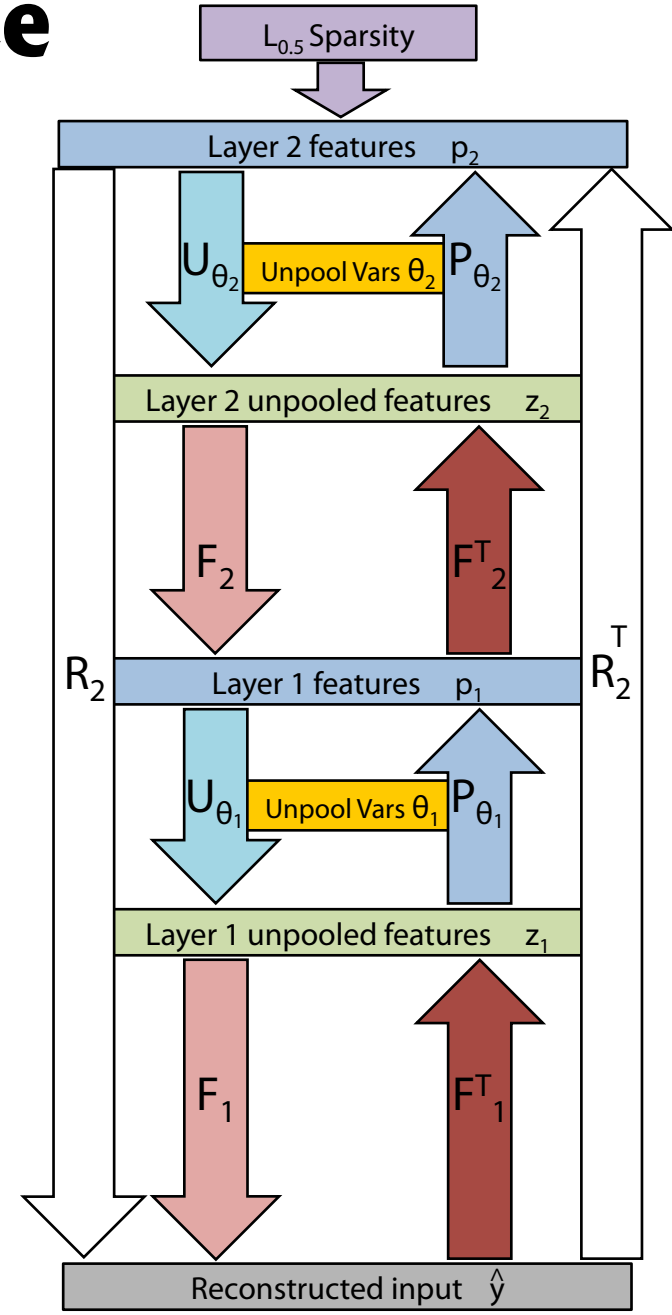
- Take pooled maps as input to next layer
- Joint inference over all layers
  - Only possible with differentiable pooling
- Objective is reconstruction error of input image
  - Not layer below, like most deep models
- Sparsity & pooling make model non-linear
  - No explicit non-linearities (e.g. sigmoid)

Overall Architecture (2 layers)



Multi-layer Joint Inference

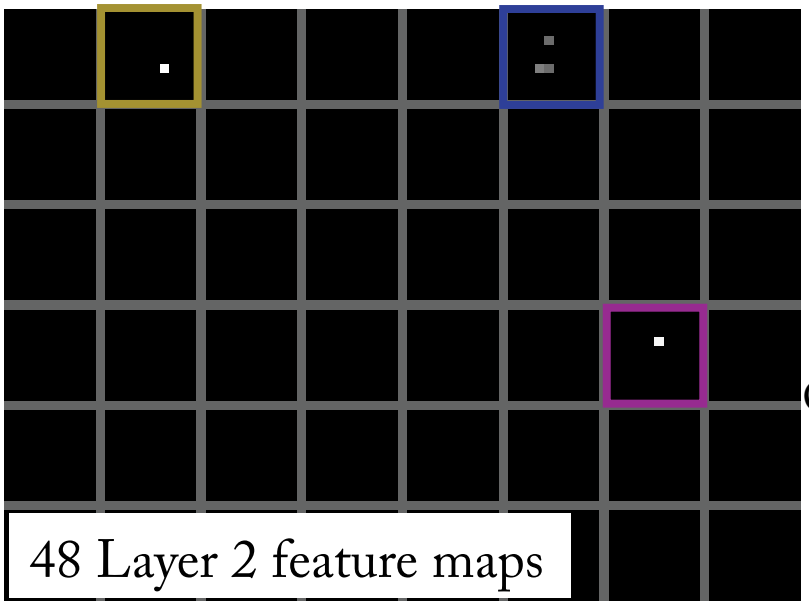
- Consider layer 2 inference:
  - Want to minimize reconstruction error of input image $\|\hat{y} - y\|_2^2$, subject to sparsity.
  - Update feature maps at top (p_2) and pooling variables (θ_1, θ_2) from both layers



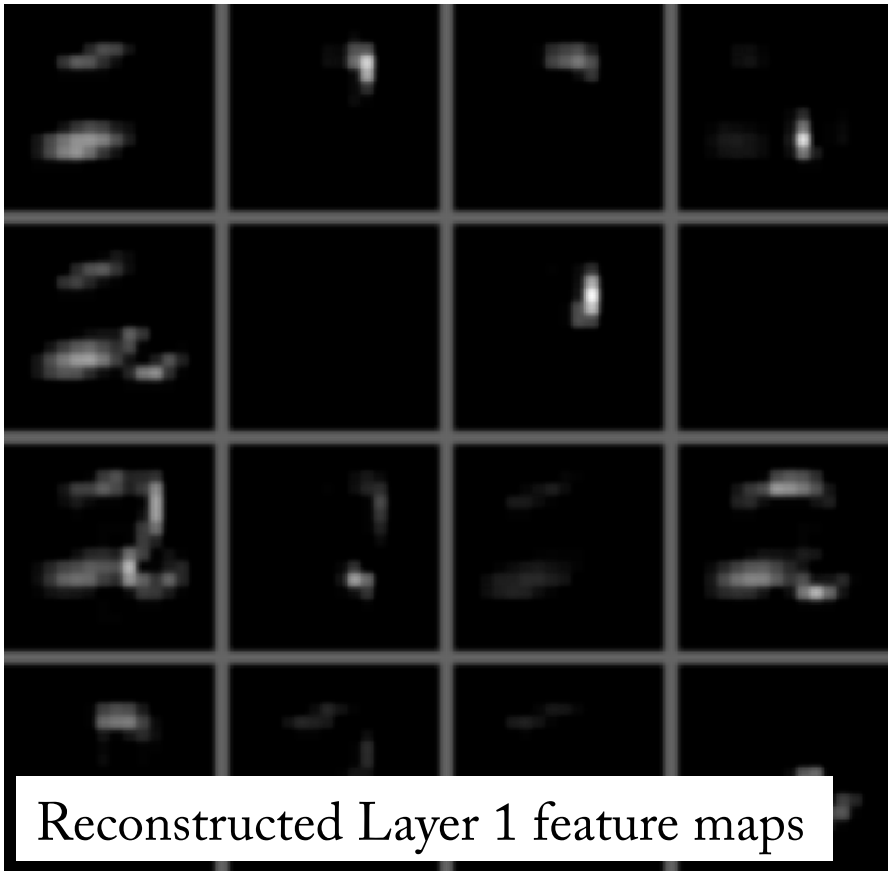
Filter Learning

- Goal: update convolutional filters f
- Fixed feature maps p & pooling variables θ
  → from inference on all training images
- Over-constrained least-squares problem
- Use Conjugate Gradients
- Normalize to unit L2 length & project positive
- Learn filters layer-by-layer
  - Joint training doesn't seem to work

Two Layer Example

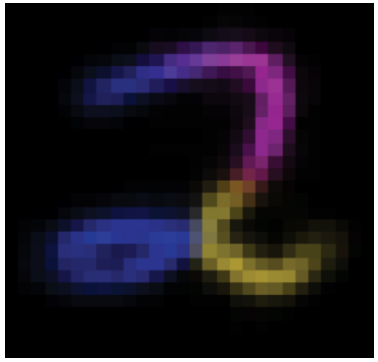


Unpool & Convolution with Layer 2 filters



Input Image

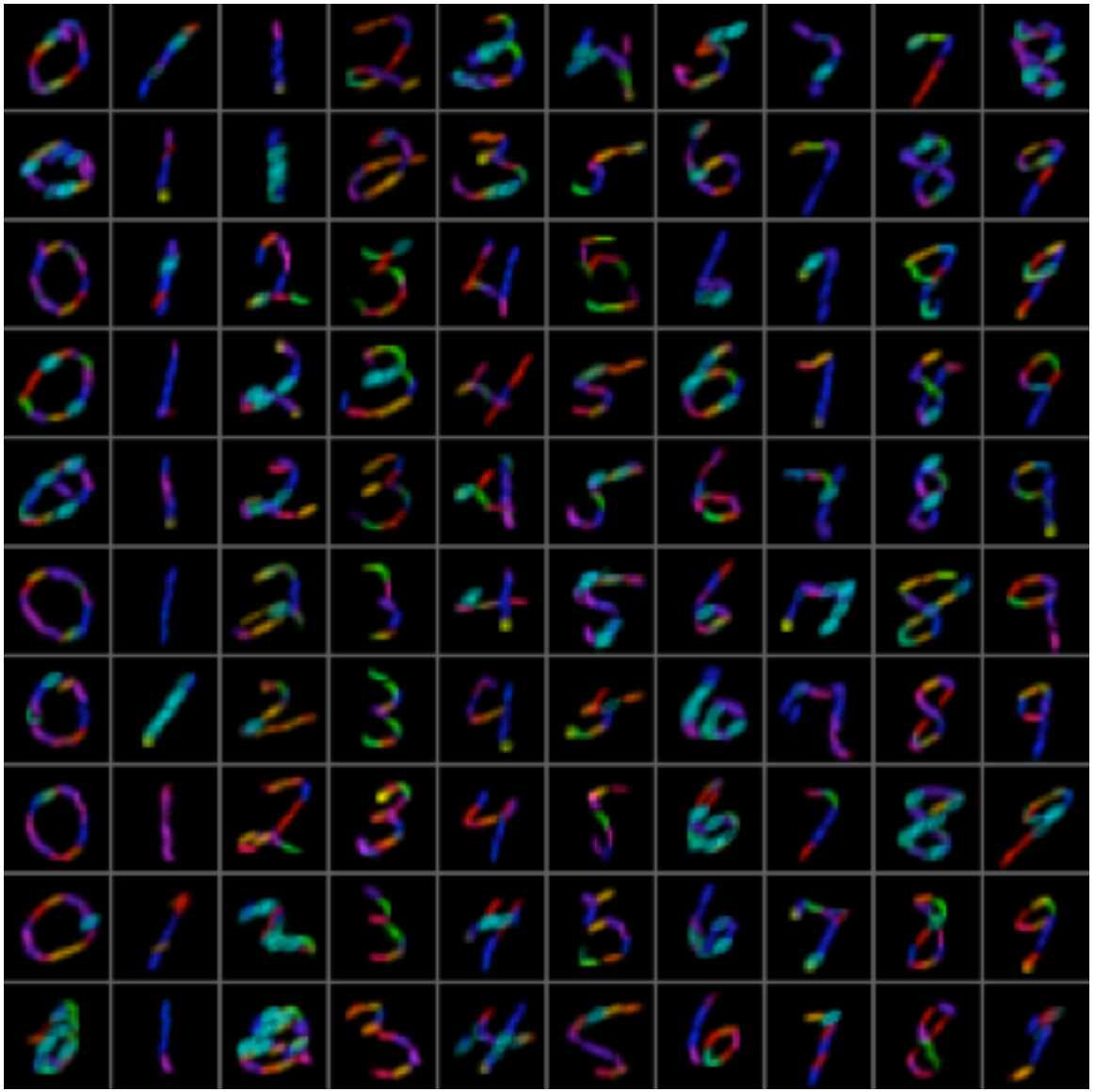
Reconstruction



Unpool & Convolution with Layer 1 filters

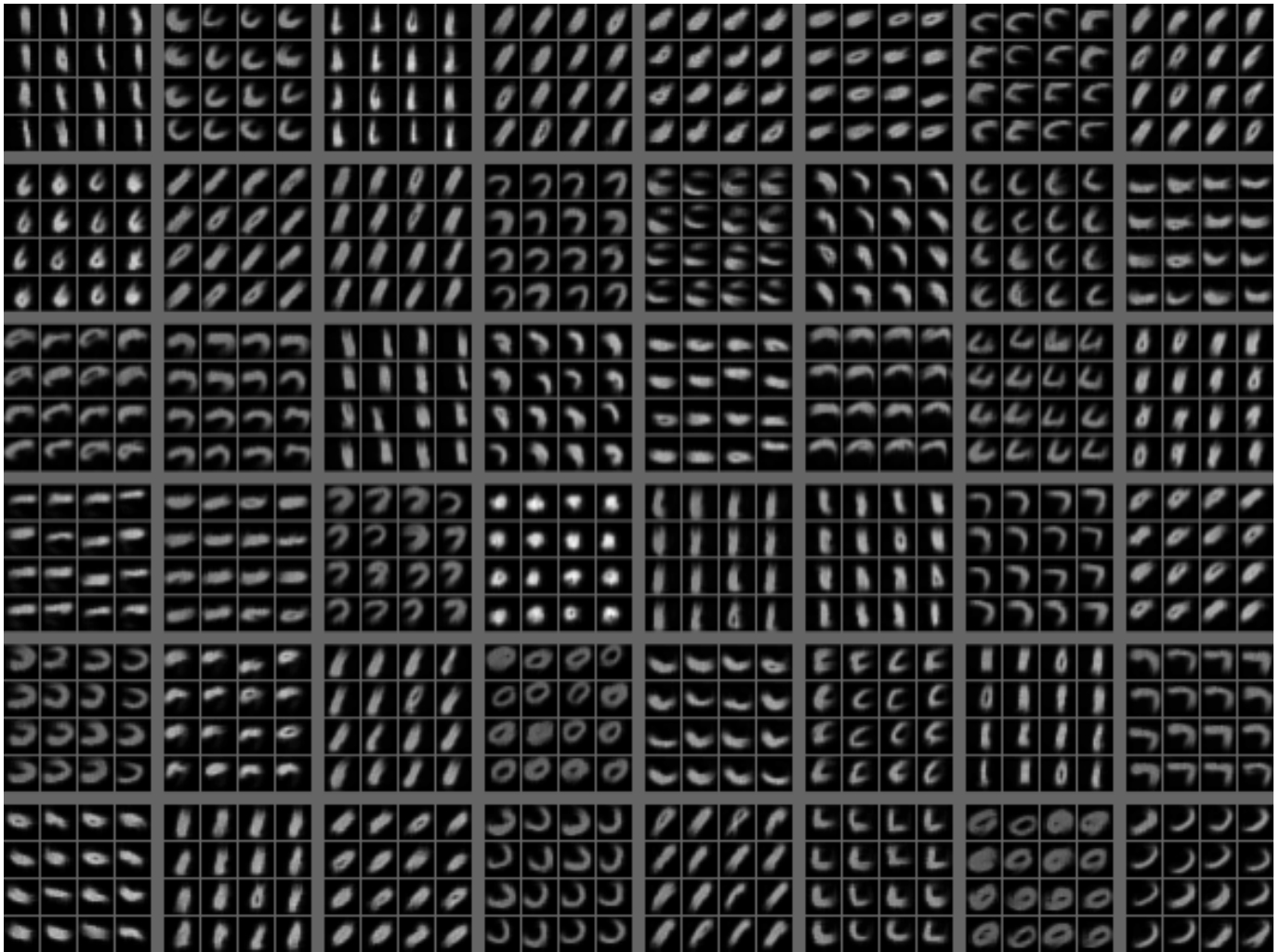
Reconstructions from 2 layer model

Reconstructions from 1 layer model



Gaussian Pooling in 2 layer model

- Projection of 2nd layer features activations down to pixels, using inferred pooling variables



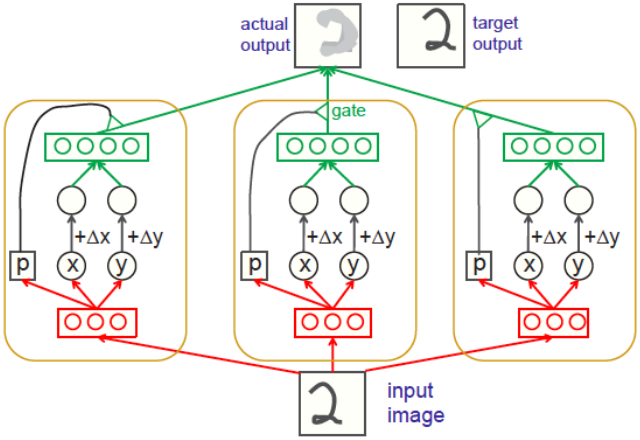
Algorithm 1 Learning with Differentiable Pooling in Deconvolutional Networks

Require: Training set Y, # layers L, # epochs E, # ISTA steps T
Require: Regularization coefficients λ_l, # feature maps B_l
Require: Pooling step sizes β_{U_l}

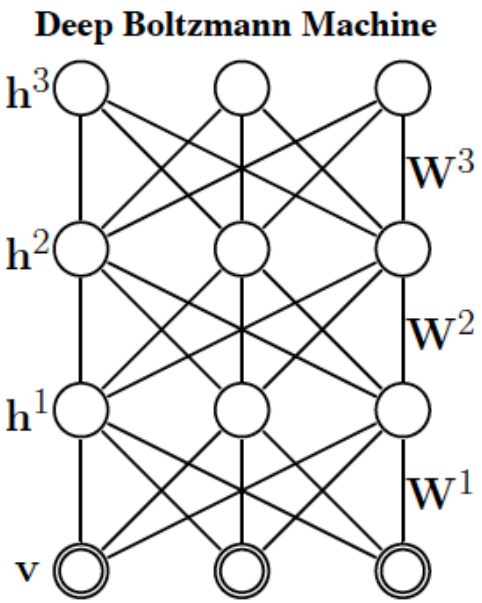
for $l = 1 : L$ **do** %% Loop over layers
  Init. features/filters: $p_l^i \sim 0$, $f_l \sim \mathcal{N}(0, \epsilon)$
  Init. switches: $\theta_l^i = Fit(R_l^T y_i) \ \ \forall i$
  for epoch $= 1 : E$ **do** %% Epoch iteration
    for $i = 1 : N$ **do** %% Loop over images
      for $t = 1 : T$ **do** %% ISTA iteration
        Reconstruct input: $\hat{v_l}^i = R_l p_l^i$
        Compute reconstruction error: $e = \hat{v_l}^i - v^i$
        Propagate error up to layer l: $\nabla p_l = R_l^T e$
        Estimate step size β_{p_l} as in Eqn. 15
        Take gradient step on p: $p_l^i = p_l^i - \lambda_l \beta_{p_l} \nabla p_l$
        Perform shrink: $p_l^i = \max(|p_l^i| - \beta_{p_l}, 0)\text{sign}(p_l^i)$
        Project to positive: $p_l^i = \max(p_l^i, 0)$
        for $k = 1 : l$ **do** %% Loop over lower layers
          Take gradient step on θ: $\theta_k^i = \theta_k^i - \lambda_l \beta_{U_k} \nabla \theta_k$
        end for
      end for
    end for
    Update f_l by solving Eqn. 26 using CG
    Project f_l to positive and unit length
  end for
end for
Output: filters f, feature maps p and pooling variables θ.

Related Work

- What / Where Separation
  - Transforming Auto-Encoders [Hinton et al. ICANN'11]
  - Gregor & LeCun [http://arxiv.org/pdf/1006.0448v1.pdf]
  - [Ranzato et al. CVPR'07]



- Convolutional Sparse Coding
  - Zeiler, Krishnan, Taylor & Fergus [CVPR '10]
  - Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu & LeCun [NIPS '10]
  - Chen, Spario, Dunson & Carin [JMLR submitted]
    - Only 2 layer models

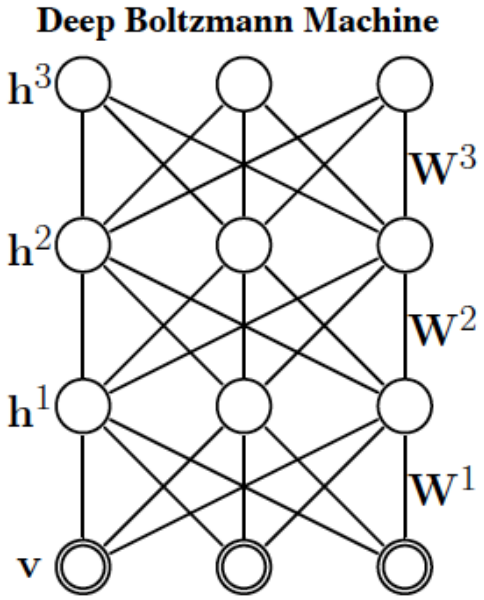


- Reconstruct all the way down to input
  - Deep Boltzmann Machines
    - Salakhutdinov & Hinton [AISTATS'09]
  - Deep Energy Models
    - Ngiam et al. [ICML'11]

Related Work

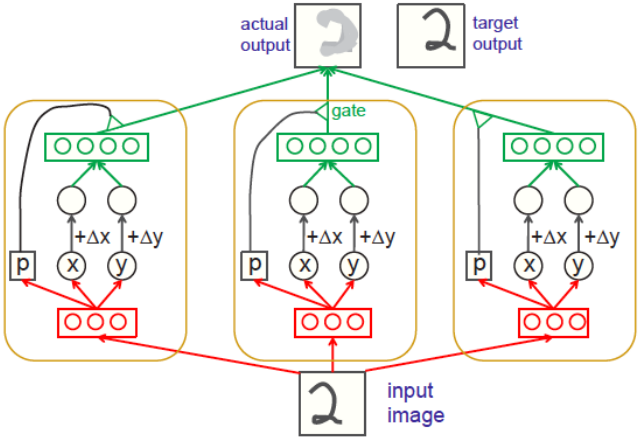
- Convolutional Sparse Coding
  - Zeiler, Krishnan, Taylor & Fergus [CVPR'10]
  - Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu & LeCun [NIPS'10]
  - Chen, Spario, Dunson & Carin [JMLR submitted]
  - Only 2 layer models

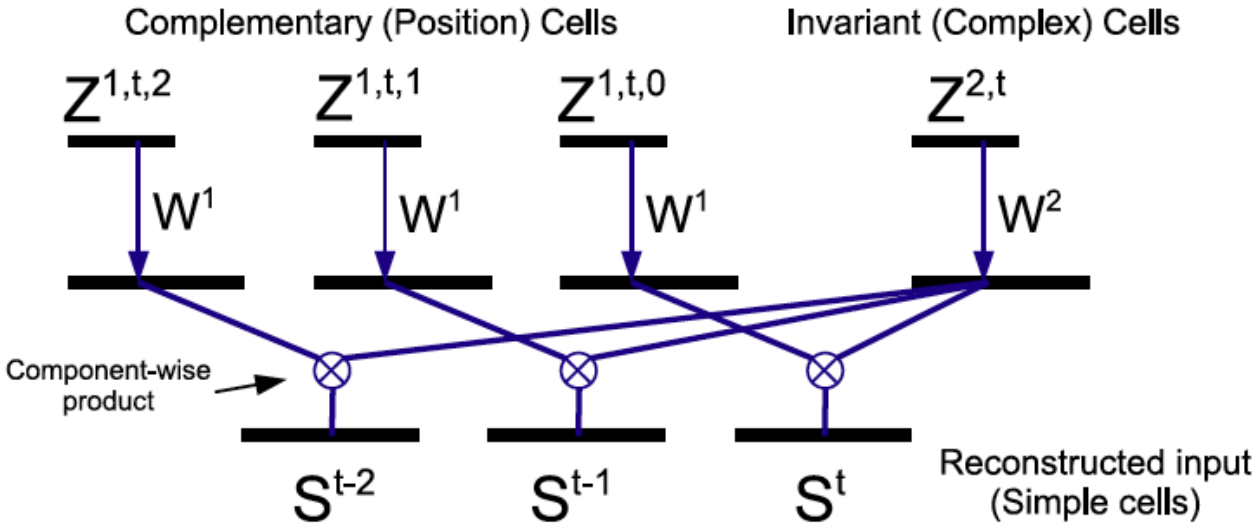
- Reconstruct all the way down
  - Deep Boltzmann Machines
    - Salakhutdinov & Hinton [AISTATS'09]
  - Deep Energy Models
    - Ngiam et al. [ICML'11]



Related Work

- What / Where Separation
  - Transforming Auto-Encoders [Hinton et al. ICANN'11]
  - [Ranzato et al. CVPR'07]
  - [Gregor & LeCun, Arxiv'10] http://arxiv.org/pdf/1006.0448v1.pdf





Talk Overview

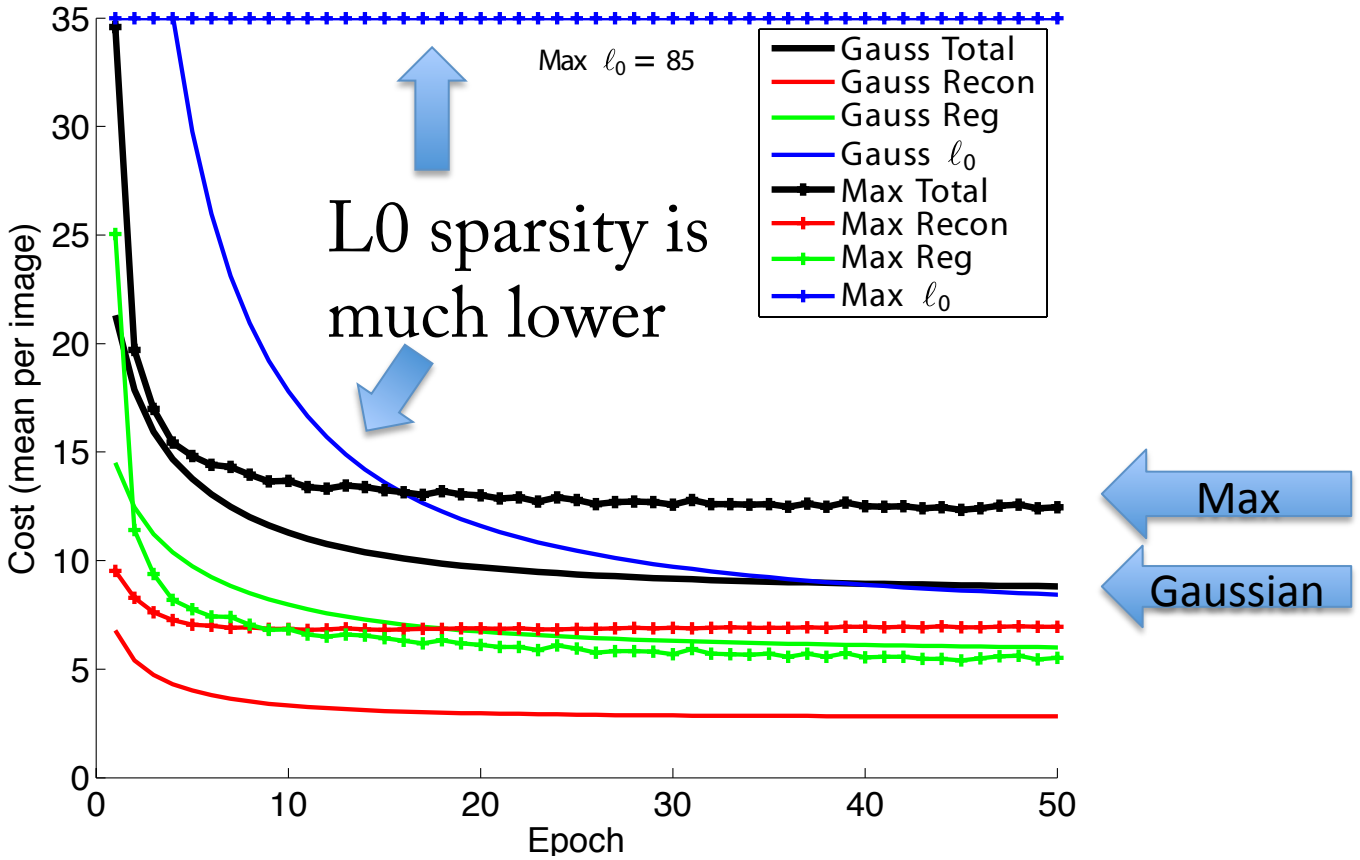
- Single layer
  - Convolutional Sparse Coding
  - Gaussian Pooling
- Multiple layers
  - Multi-layer inference
  - Filter learning
- Related work
- Experiments

MNIST Experiments

- 2 layers:
  - Layer 1: 16 filters (5x5 pixels)
  - Layer 2: 48 filters (5x5 pixels)
  - 2x2 (Un)pooling at each layer
- 50 inference iterations per image
  - 100 frames/sec on GPU
- Unsupervised training
  - No fine tuning
- Classification: patchify feature maps → linear SVM

Gaussian vs Max Pooling

- Compare objective cost to conventional max pooling



Classification:

	Layer 1	Layer 2
Max Pooling	1.30%	1.25%
Gaussian Pooling	1.38%	0.84%

Joint vs Separate Inference

Separate inference (2 phases):
(i): Layer 1 infer (θ_1, p_1) then fix θ_1
(ii): Layer 2 infer (θ_2, p_2)

1.39% error

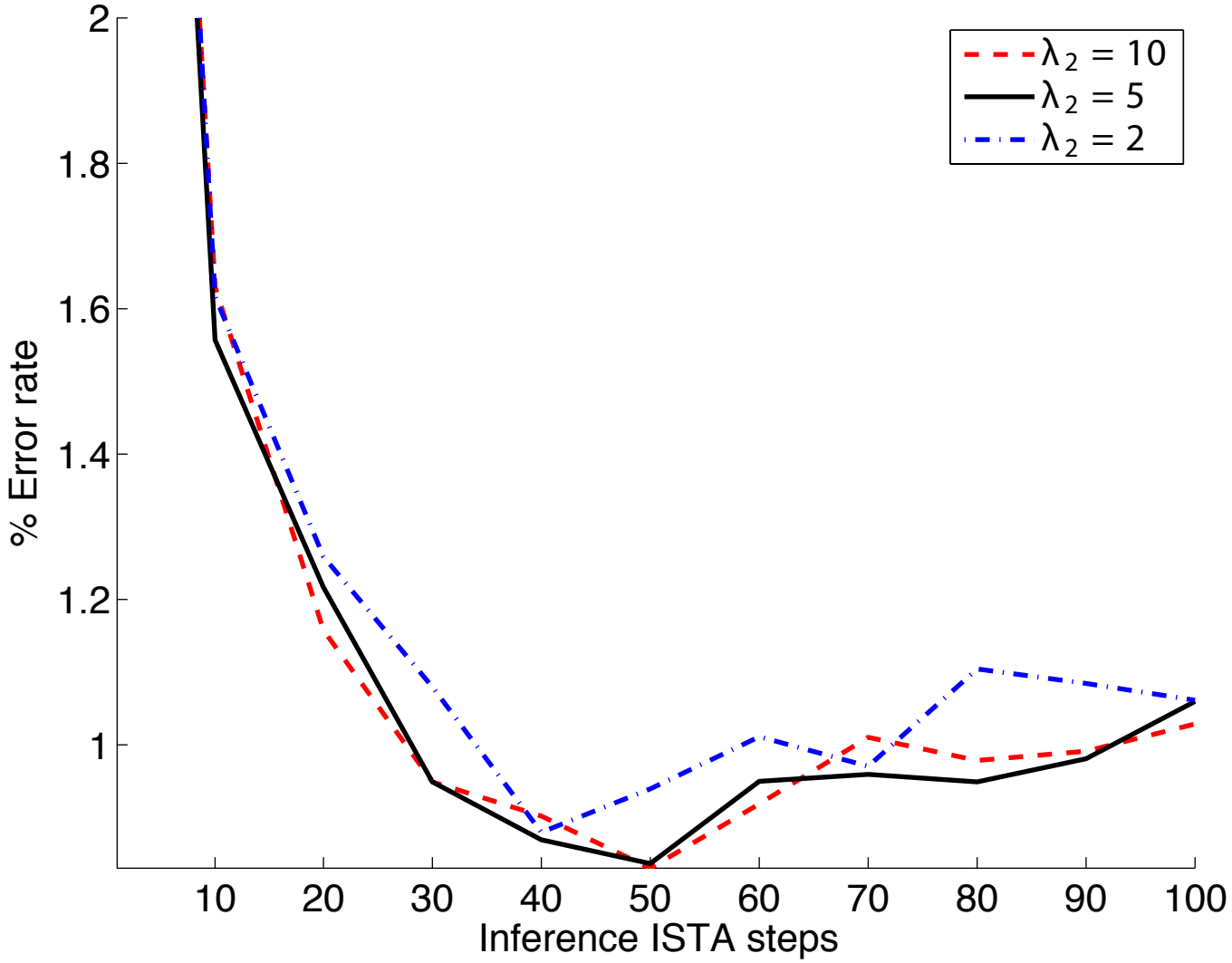
Joint Inference (1 phase):
(i): Layer 1 & 2 infer (θ_1, θ_2, p_2)

0.84% error

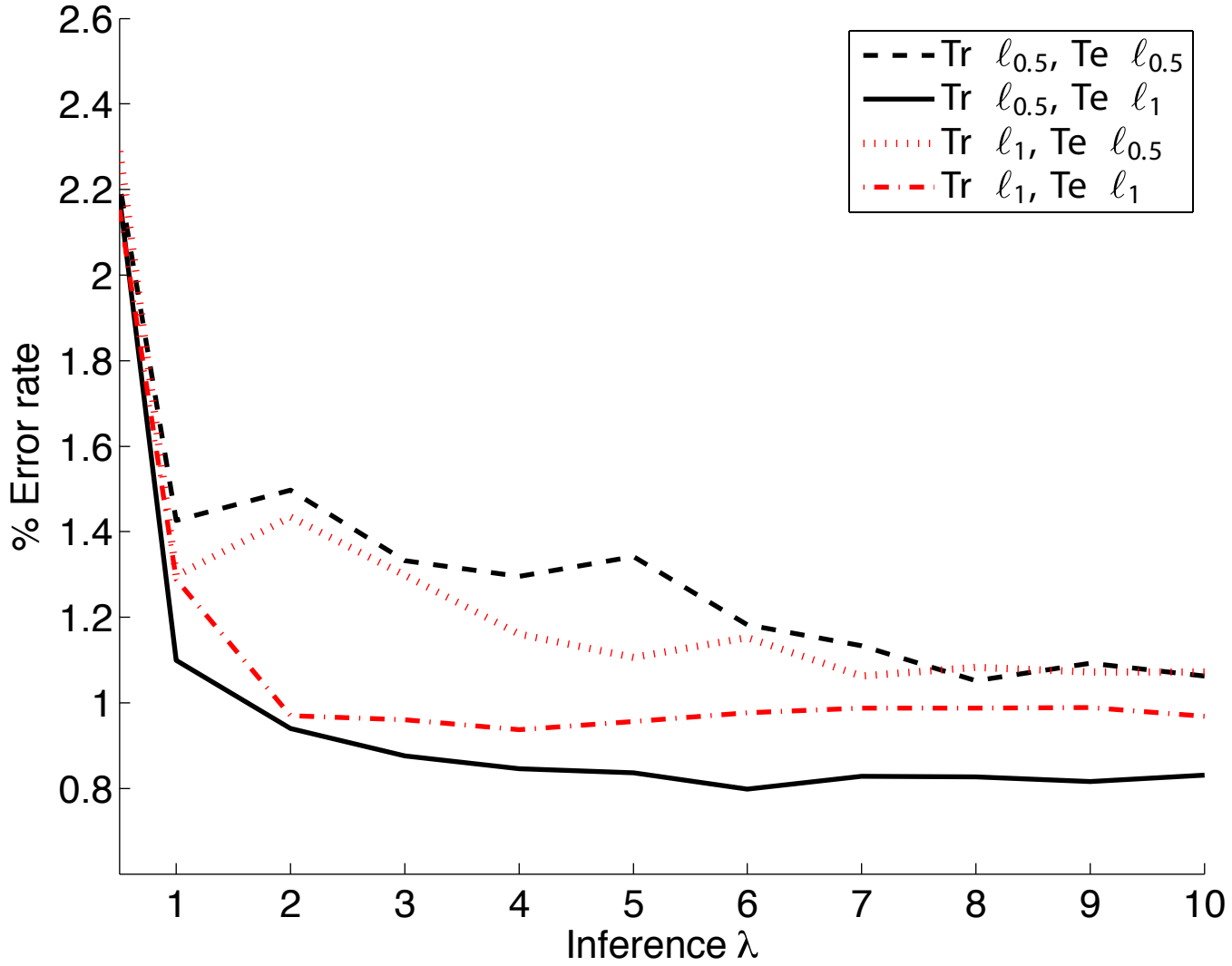
Using top-down information to set lower pooling variables:

40% reduction in error

Number of ISTA Iterations (Test)



L0.5 vs L1 Sparsity in Train/Test



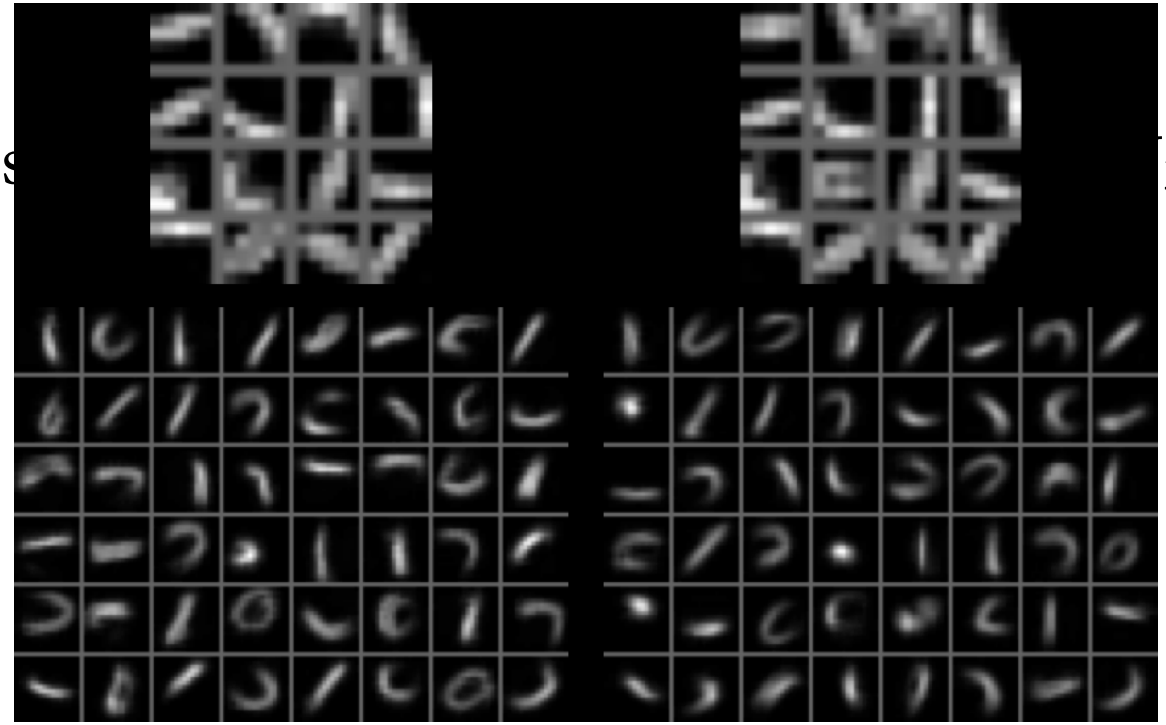
Non-Negativity Results

	Positive/Negative	Non-negative
Max Pooling	2.04%	1.25%
Gaussian Pooling	2.32%	0.84%

Table 3. MNIST error rate for Max and Gaussian models trained with and without the non-negativity constraint.

Resetting Trick

- Periodically set feature maps to zero during training

With res  ithout reset

Trained with No Reset	Trained with Reset
1.00%	0.84%

Table 4. MNIST error rates for 2 layer models trained with and without resetting the feature maps.

MNIST Results Comparison

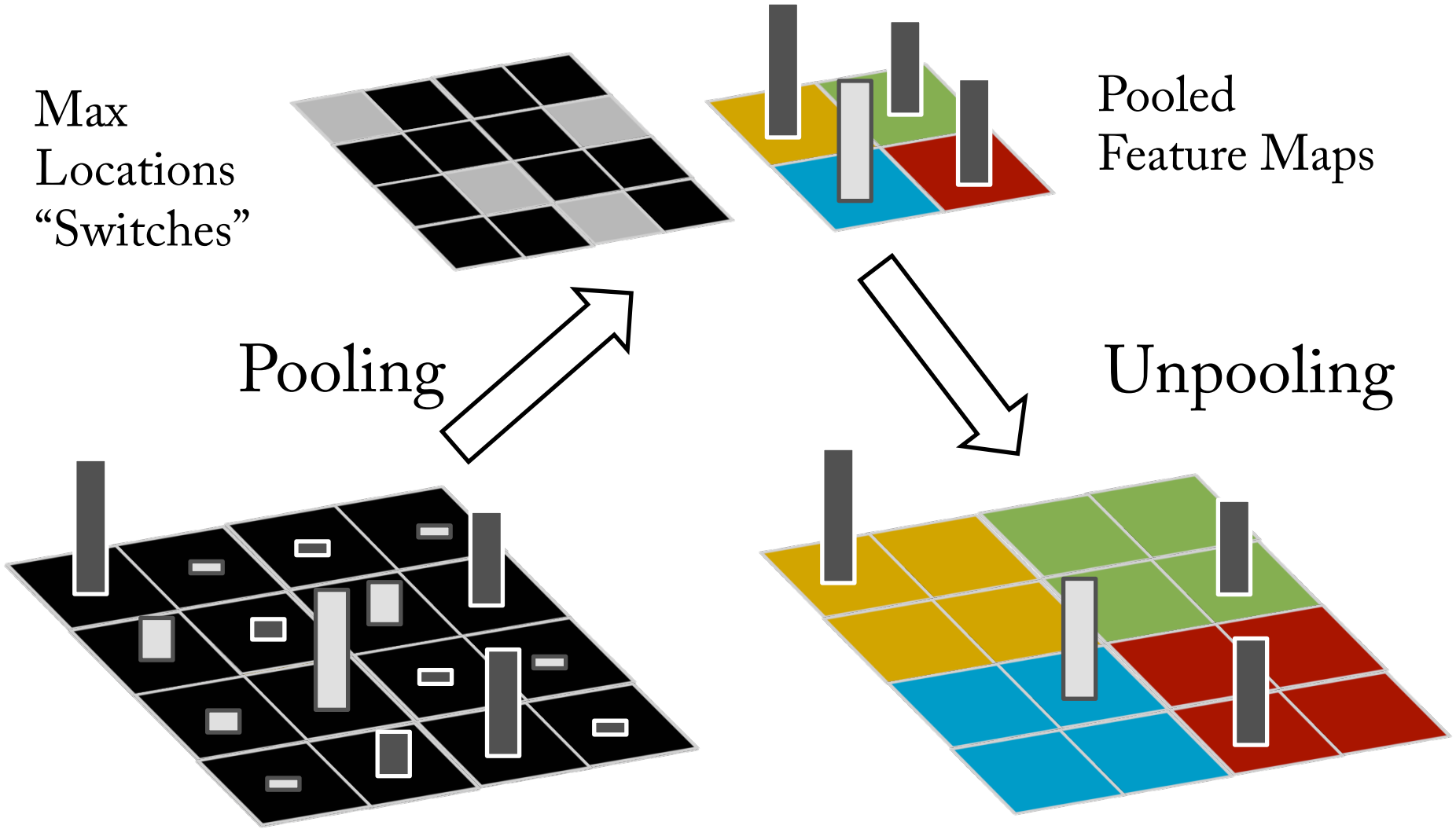
	Pre-training	Fine-tuning
Our Method	0.84%	-
Convolutional RBM [Lee et al. ICML'09]	0.82%	-
Deep Belief Net [Hinton et al. 2006]	2.5%	1.18%
Deep Boltzmann Machine [S & H AISTATS'09]	-	0.95%

Caltech 101 Experiments

- Experiment using older model (from Zeiler et al. ICCV'11) using Max pooling NOT Gaussian pooling

- Unsupervised training on 3060 images from Caltech 101
- Resized/padded to 150x150 gray-scale
- Subtractive & divisive contrast normalization

Reversible Max Pooling



Model Parameters/Statistics

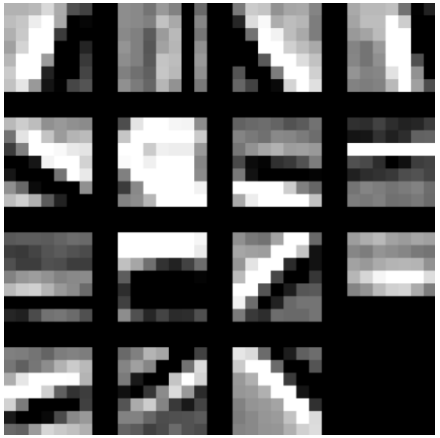
Property	Layer 1	Layer 2	Layer 3	Layer 4
# Feature maps K_l	15	50	100	150
Pooling size	3x3x3	3x3x2	3x3x2	3x3x2
λ_l	2	0.1	0.005	0.001

- 7x7 filters at all layers

Model Reconstructions

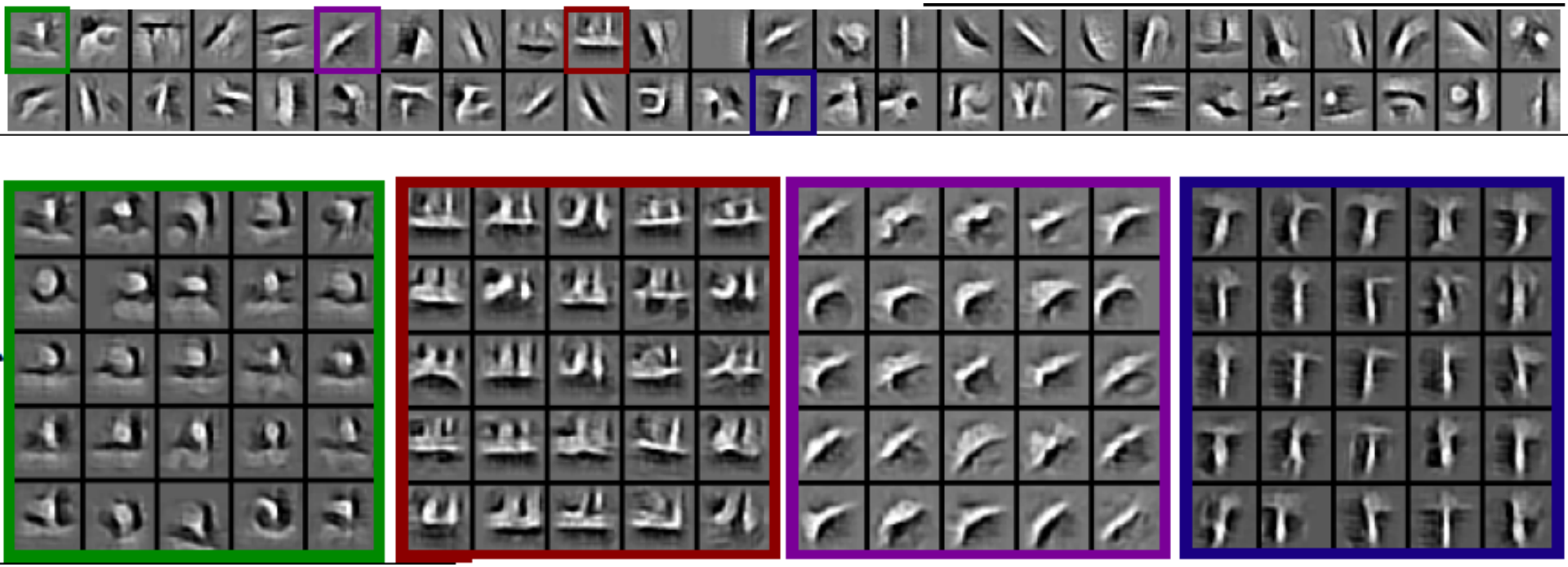
Layer 1 Filters

- 15 filters/feature maps



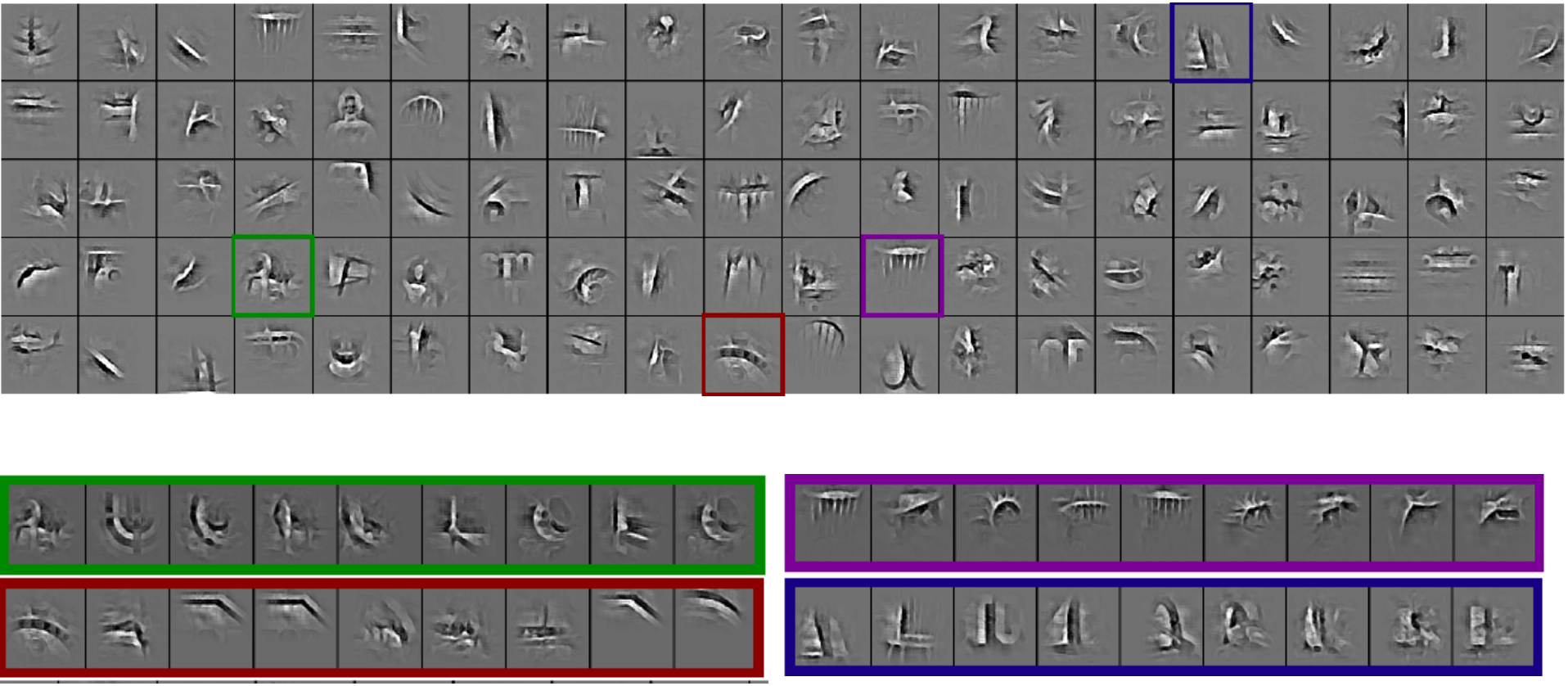
Layer 2 Filters

- 50 filters/feature maps, showing max for each map projected down to image



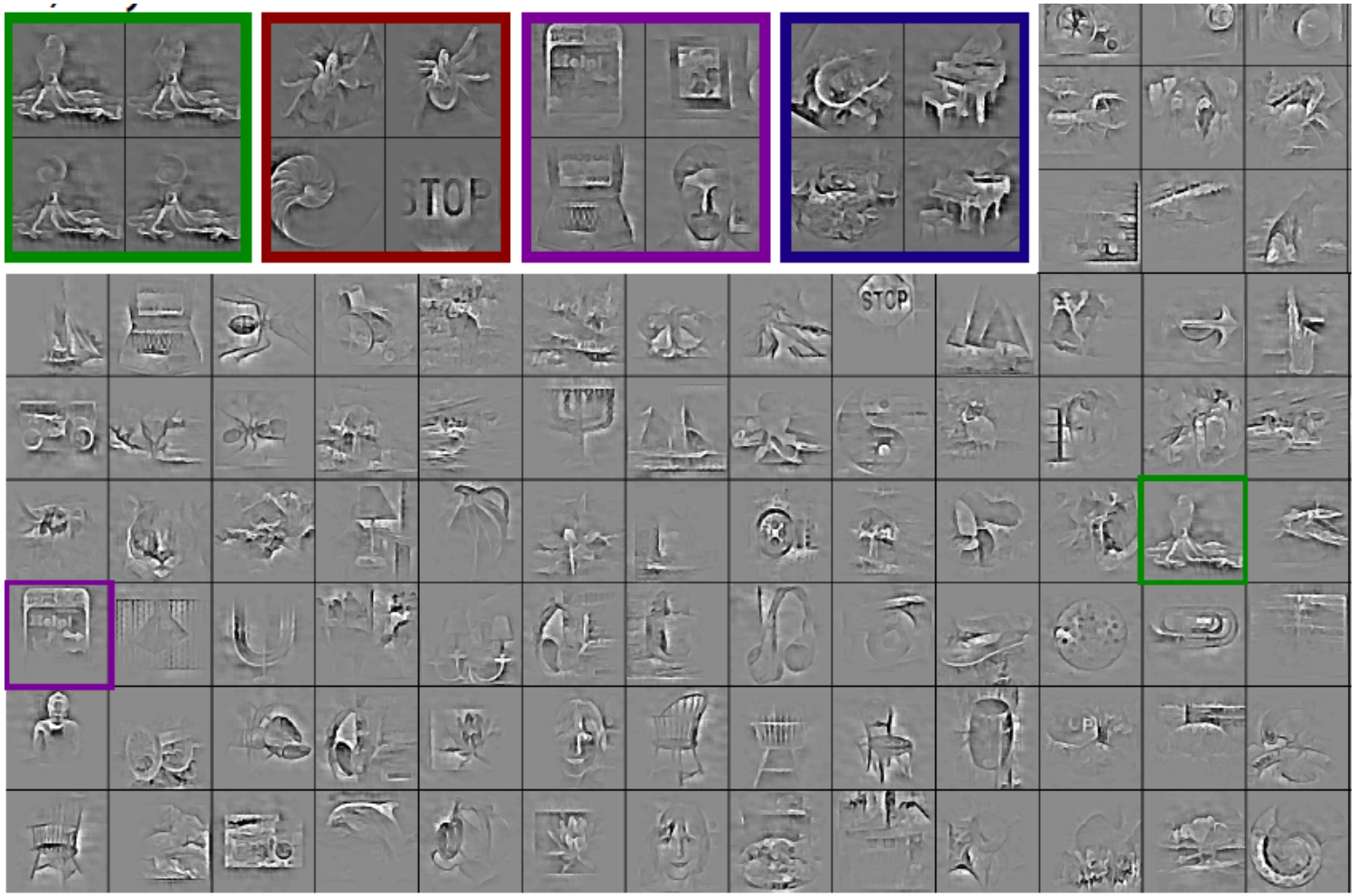
Layer 3 filters

- 100 filters/feature maps, showing max for each map

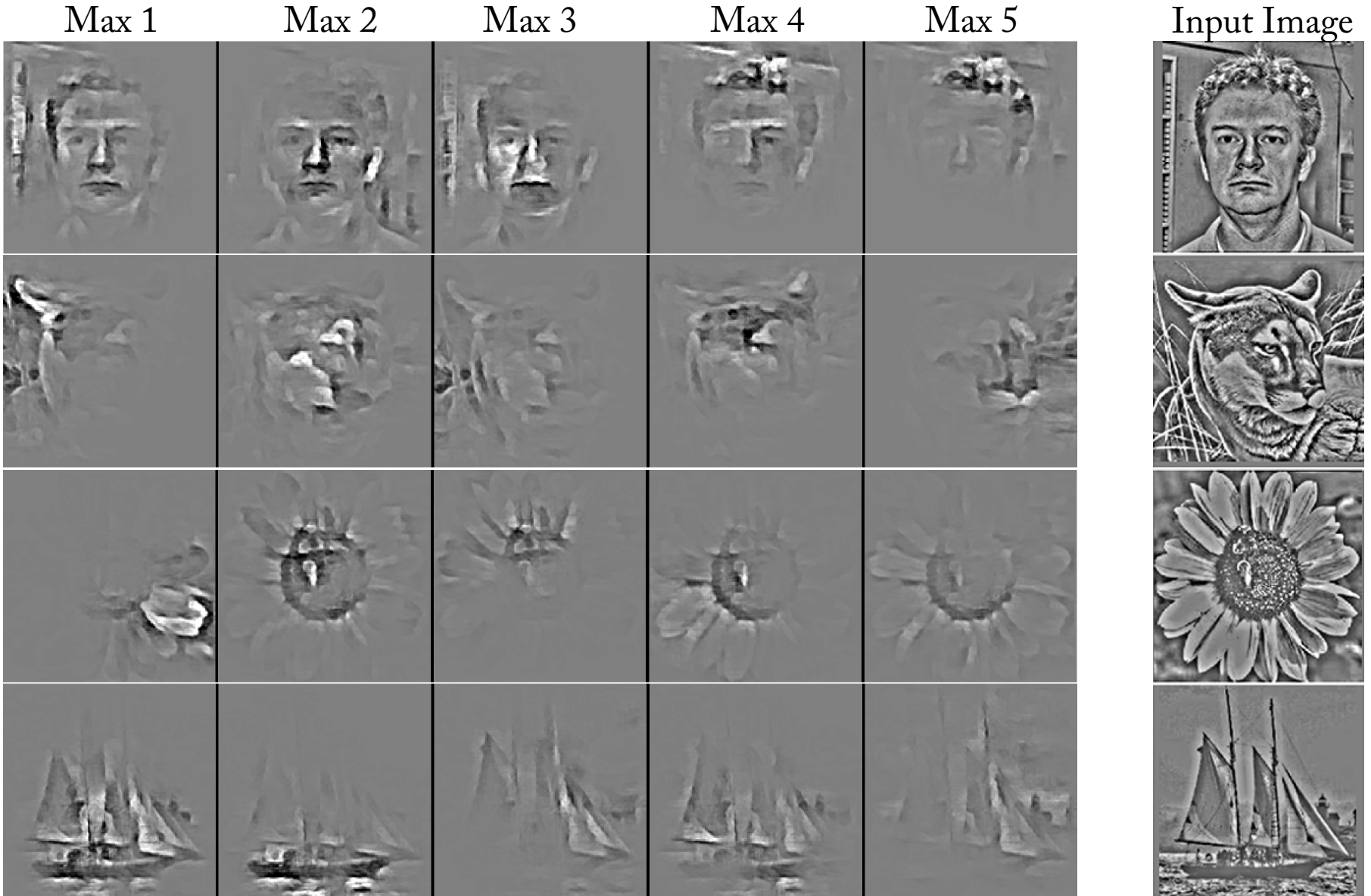


Layer 4 filters

- 150 in total; receptive field is entire image



Largest 5 activations at top layer



Classification Results: Caltech 101

- Use features as input to Spatial Pyramid Matching (SPM) of Lazebnik et al. [CVPR'06]

Our model - layer 1	$67.8 \pm 1.2\%$
Chen *et al.* [3] layer-1+2 (ConvFA)	$65.7 \pm 0.7\%$
Kavukcuoglu *et al.* [8] (ConvSC)	$65.7 \pm 0.7\%$
Zeiler *et al.* [20] layer-1+2 (DN)	$66.9 \pm 1.1\%$
Boureau *et al.* [2] (Macrofeatures)	$70.9 \pm 1.0\%$
Jarrett *et al.* [7] (PSD)	$65.6 \pm 1.0\%$
Lazebnik *et al.* [9] (SPM)	$64.6 \pm 0.7\%$
Lee *et al.* [11] layer-1+2 (CDBN)	$65.4 \pm 0.5\%$

Convolutional Sparse Coding

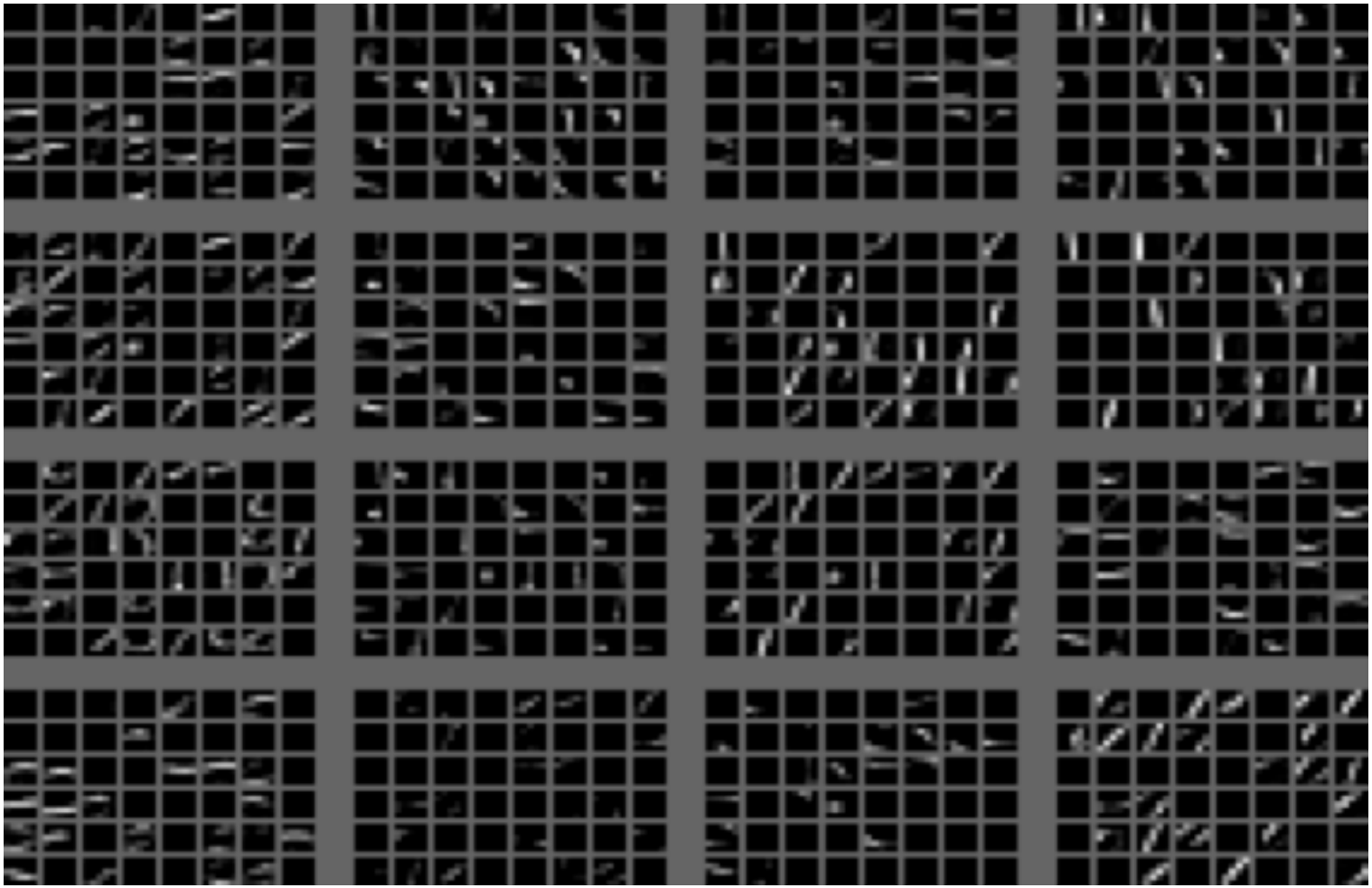
Other approaches using SPM with Hard quantization

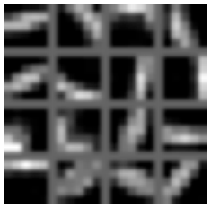
But some way below SOA for single feature (~78%)

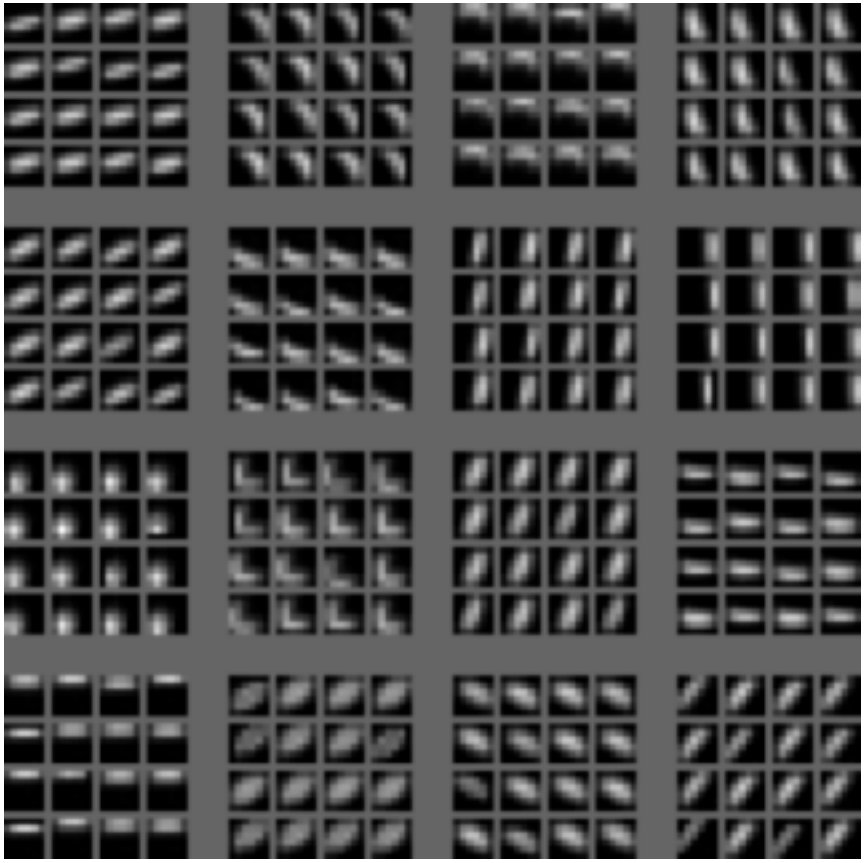
Summary

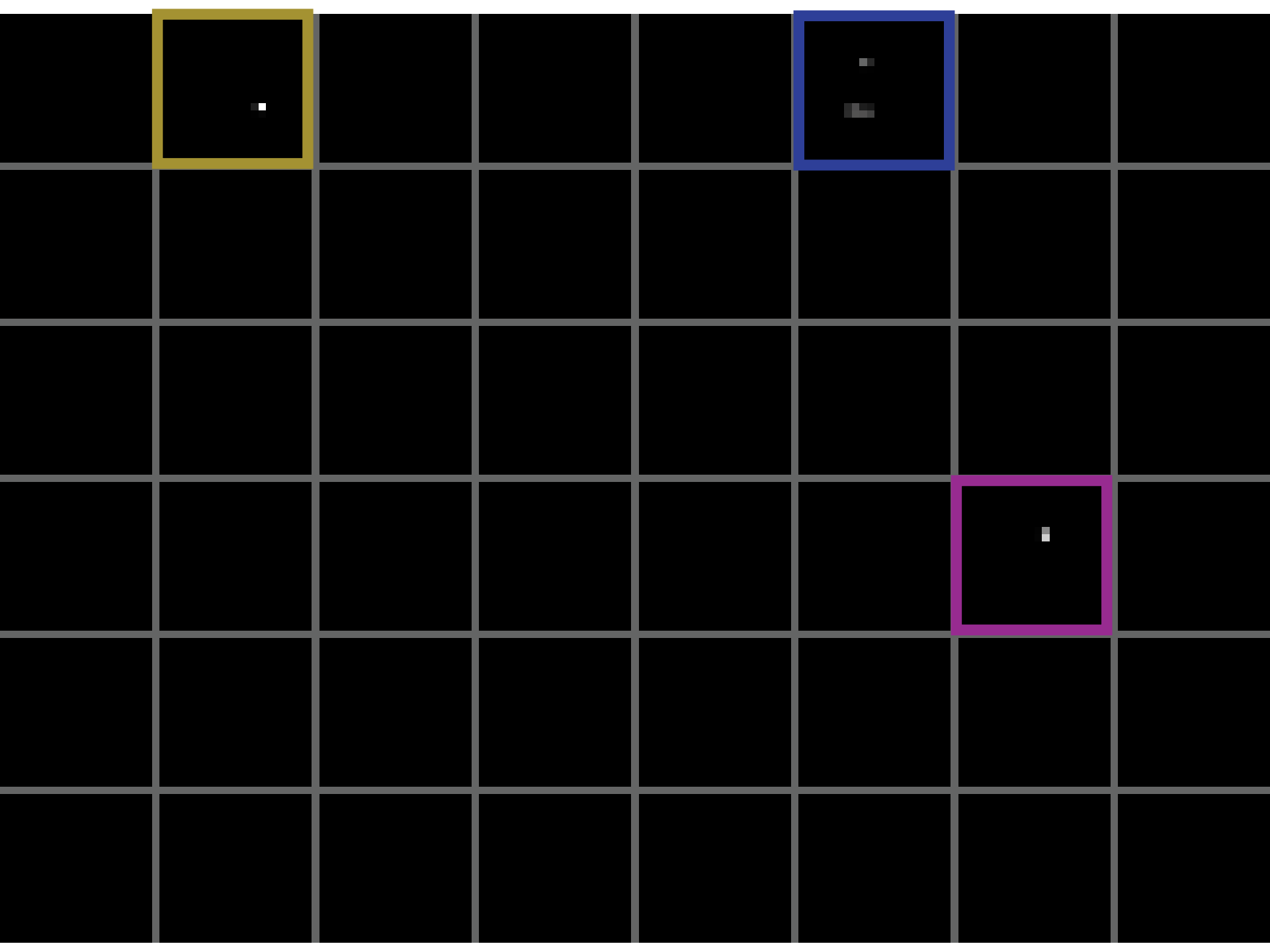
- Variant of sparse coding with novel Gaussian pooling
- Integrating pooling into cost function gives better & sparser features
- Joint inference: top-down influence on lower levels helps performance
- Code & papers on Matt Zeiler's webpage
  - http://www.matthewzeiler.com/pubs/arxiv2012/arxiv2012.pdf

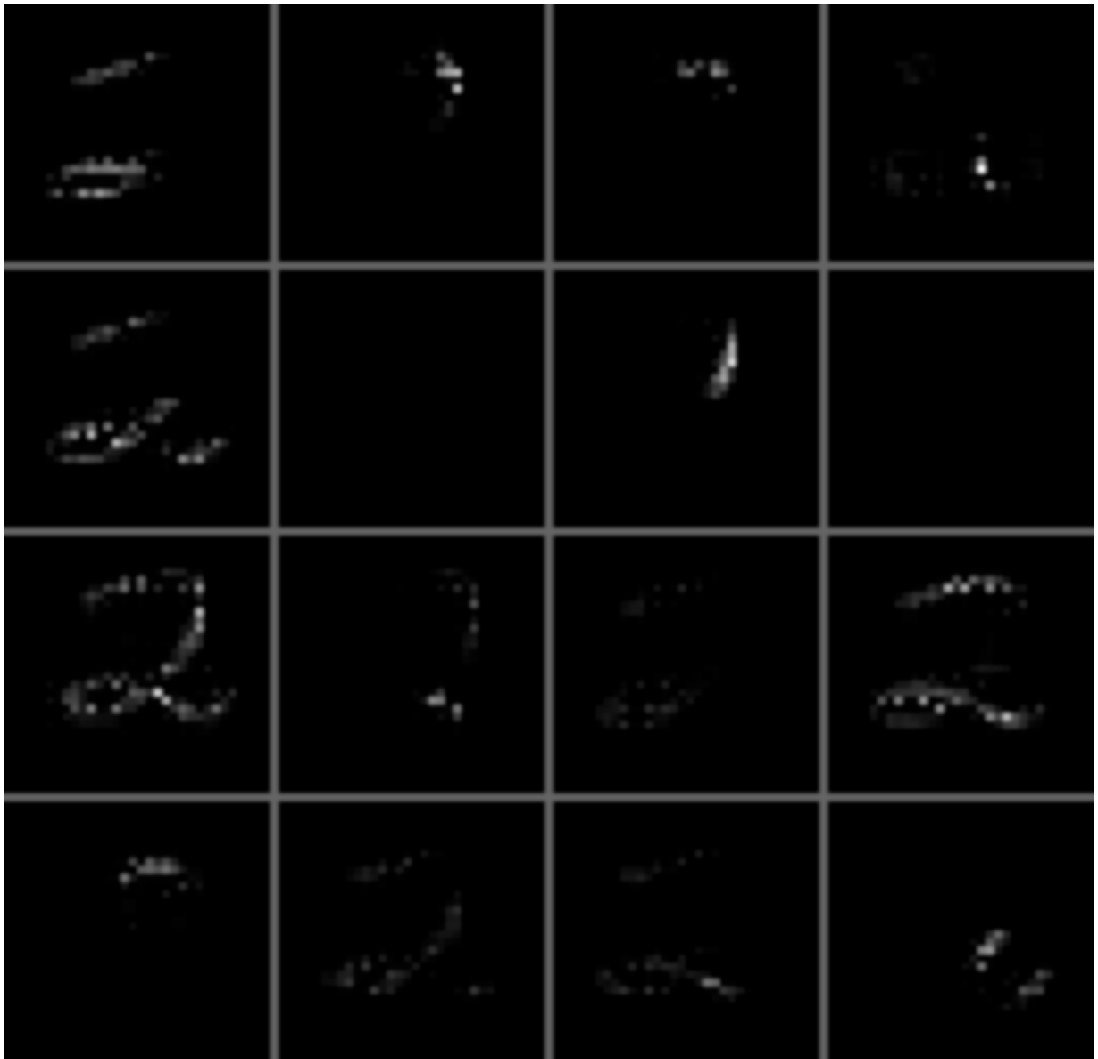
Layer 2 Filter Coefficients









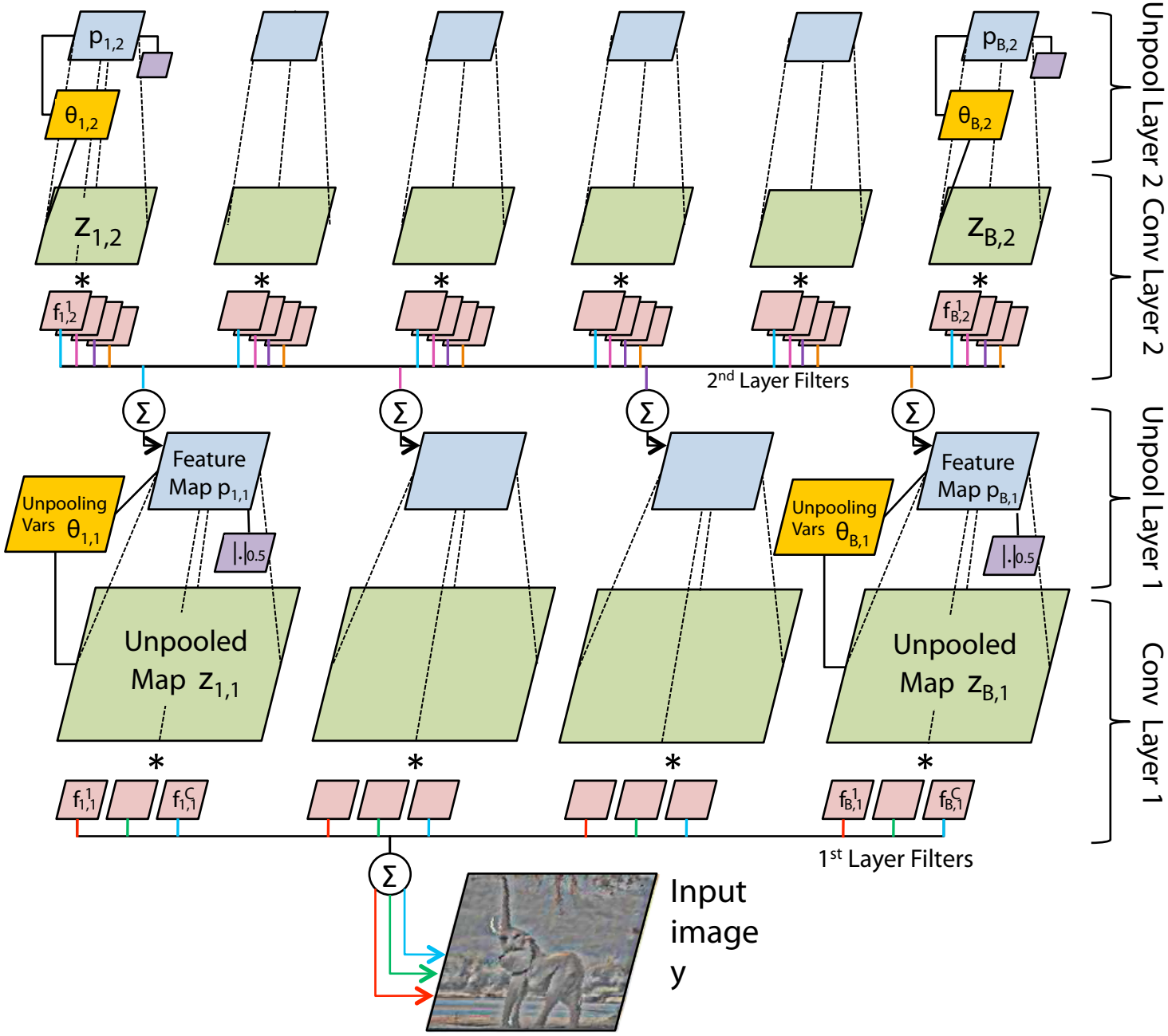


Max Pooling Projections (Aliasing)

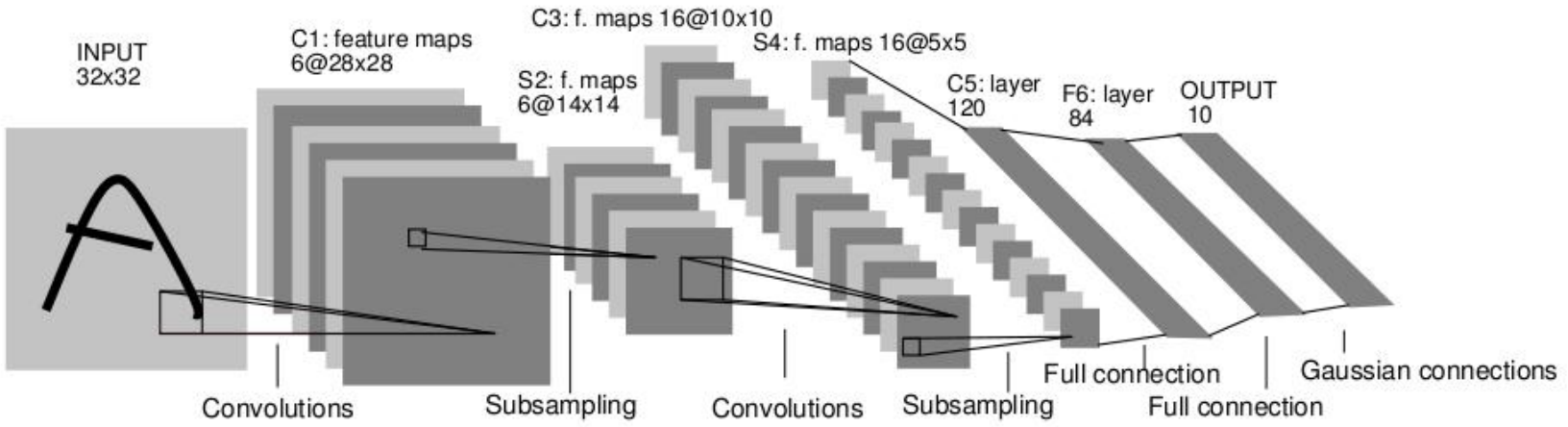
Joint vs Separate Inference

Training	Infer 2 phases	Infer 1 phase (no U_1)	Infer 1 phase
Updating F_1 U_{w_1}	1.79%	1.63%	1.40%
Updating F_1	1.71%	1.21%	1.10%
Updating U_{w_1}	1.39%	1.04%	0.84%
No Layer 1 Updates	1.46%	0.99%	1.03%

Table 2. Comparison of joint training techniques. Each row is a trained two layer model that updates select variables in layer 1 during training (in addition to F_2 and U_{w_2}). The three columns use these models but run inference at test time in 2 phases, 1 phase without updating U_1, and 1 phase with all updates respectively.


Unpool Layer 2
Conv Layer 2
Unpool Layer 1
Conv Layer 1
$p_{1,2}$
$p_{B,2}$
$\theta_{1,2}$
$\theta_{B,2}$
$z_{1,2}$
$z_{B,2}$
$f^1_{1,2}$
$f^1_{B,2}$
2nd Layer Filters
Feature Map $p_{1,1}$
Feature Map $p_{B,1}$
Unpooling Vars $\theta_{1,1}$
Unpooling Vars $\theta_{B,1}$
$|.|_{0.5}$
Unpooled Map $z_{1,1}$
Unpooled Map $z_{B,1}$
$f^1_{1,1}$
$f^C_{1,1}$
$f^1_{B,1}$
$f^C_{B,1}$
1st Layer Filters
Input image y

Comparison: Convolutional Nets



LeCun *et al.* 1989

Convolutional Networks

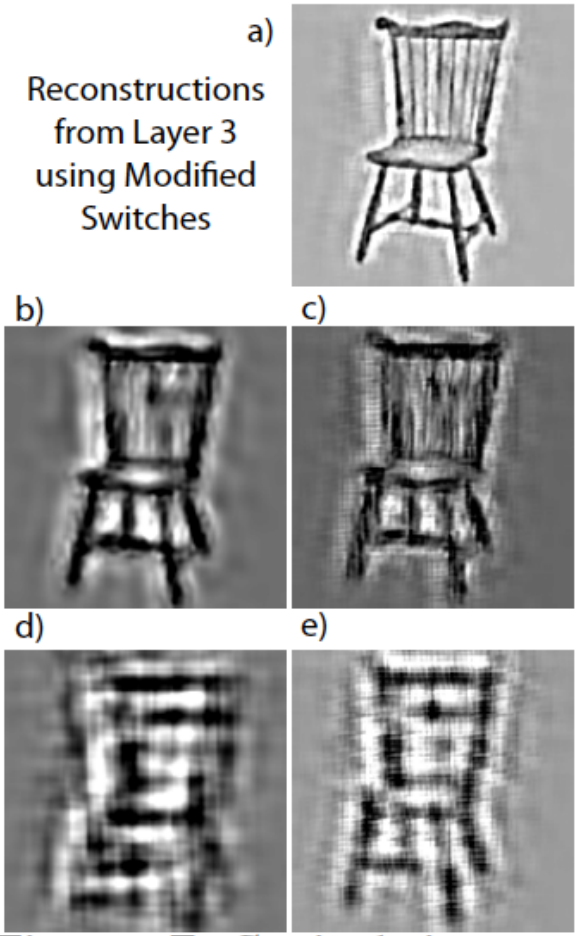
- Bottom-up filtering with convolutions in image space.
- Trained supervised requiring labeled data.

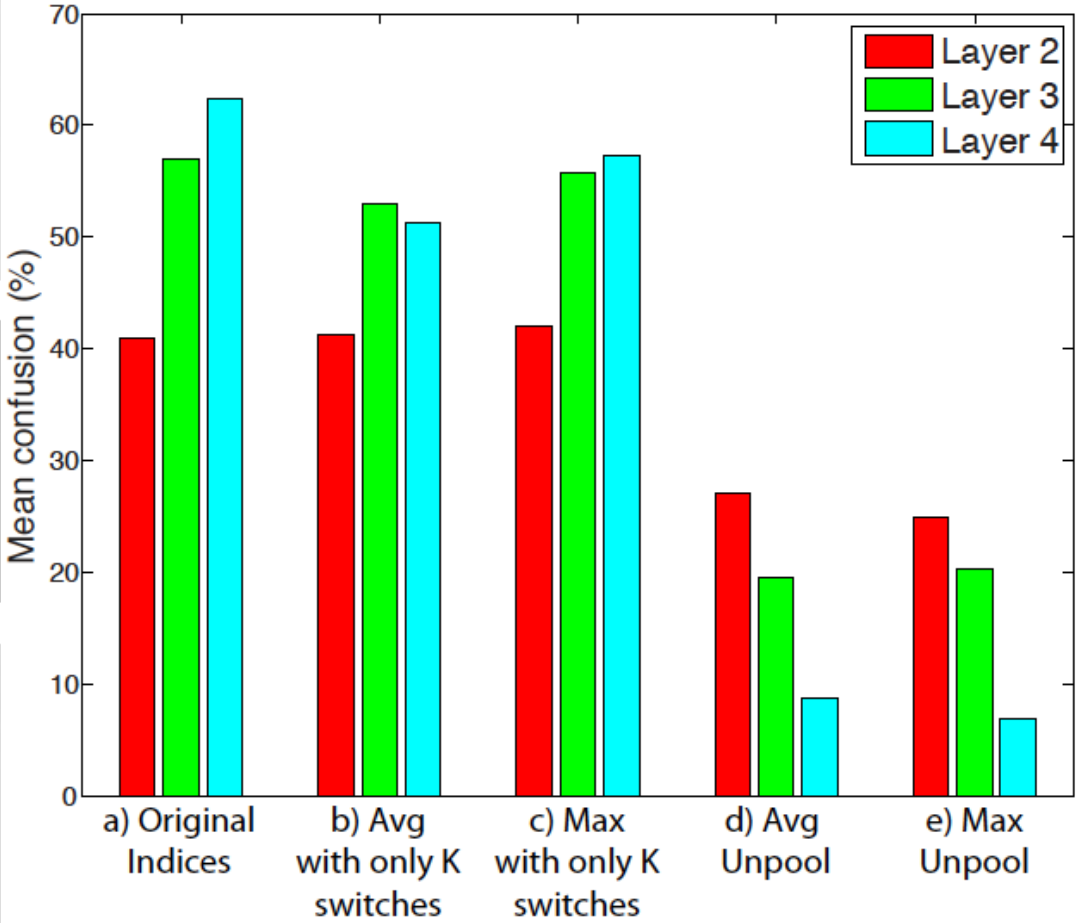
Deconvolutional Networks

- Top-down decomposition with convolutions in feature space.
- Non-trivial unsupervised optimization procedure involving sparsity.

Analysis of Switch Settings

- Recons. and classification with various unpooling.





Related Work

- Convolutional Sparse Coding
  - Zeiler, Krishnan, Taylor & Fergus [CVPR '10]
  - Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu & LeCun [NIPS '10]
  - Chen, Spario, Dunson & Carin [JMLR submitted]
    - Only 2 layer models
- Deep Learning
  - Hinton & Salakhutdinov [Science '06]
  - Ranzato, Poultney, Chopra & LeCun [NIPS '06]
  - Bengio, Lamblin, Popovici & Larochelle [NIPS '05]
  - Vincent, Larochelle, Bengio & Manzagol [ICML '08]
  - Lee, Grosse, Ranganth & Ng [ICML '09]
  - Jarrett, Kavukcuoglu, Ranzato & LeCun [ICCV '09]
  - Ranzato, Mnih, Hinton [CVPR'11]
  - Reconstruct layer below, not input
- Deep Boltzmann Machines
  - Salakhutdinov & Hinton [AIStats '09]

Related Work

- Hierarchical vision models
  - Zhu & Mumford [F&T '06]
  - Tu & Zhu [IJCV '06]
  - Serre, Wolf & Poggio [CVPR '05]