Deconvolutional Networks

Matthew D. Zeiler
Graham W. Taylor
Rob Fergus

Dept. of Computer Science, Courant Institute,
New York University
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 \( D \)

\[
C(y, D) = \arg\min_p \frac{\lambda}{2} \| Dp - y \|_2^2 + |p|_1
\]

- \( \ell_1 \) 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

- **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 $\rightarrow$ per-element sparsity constraint
**Gaussian Unpooling**

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

(What)

Feature Map $p$

(Where)

(1): $\mu_x, \mu_y, \gamma_x, \gamma_y$

(2): $\mu_x, \mu_y, \gamma_x, \gamma_y$

(3): $\mu_x, \mu_y, \gamma_x, \gamma_y$

(4): $\mu_x, \mu_y, \gamma_x, \gamma_y$

(Un)pooling Variables $\theta$

Unpooled feature map $z$

Neighborhood $N_1$, $N_2$, $N_4$
Single Layer Cost Function

\[ \frac{\lambda}{2} \left\| FU_{\theta}p - \hat{y} \right\|_2^2 + \left\| p \right\|_1^{1/2} \]

Reconstructed image:
\[ \hat{y} = \sum_k z_k \ast f_k \]

Unpooling:
\[ z_k = U_{\theta}p_k \]

Feature Maps
Input Image
Reconstruction
Sparsity (per-element)

[with Gaussian parameters \( \theta \)]
Single Layer Cost Function

\[
\hat{y} = \sum_k z_k * f_k
\]

\[
\hat{y} = \sum_k z_k * f_k
\]

\[
\frac{\lambda}{2} \| FU_{\theta}p - y \|_2^2 + |p|_\alpha^{1/2}
\]

Learned (share across all images)

Infer for each image

Feature Maps

Input Image

Reconstructed image

Unpooling [with Gaussian parameters \(\theta\)]

\[
z_k = U_{\theta}p_k
\]
Single Layer Inference

**Feature Maps** $p$  (What)
- Fix convolution filters $f$ and pooling variables $\theta$
- 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** $\theta$  (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

Unpooled Feature Maps
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 ($\theta_1, \theta_2$) from both layers.

- Update features (ISTA):
  1. Reconstruct input
  2. Compute error
  3. Forward prop. error
  4. Gradient step
  5. Shrinkage

- Update Gaussian pooling variables:
  Combine top-down with bottom-up error chain rule to

  - No explicit non-linearities between layers
  - But still get very non-linear behavior

Multi-layer Joint Inference

Layer 1 features $p_1$

Reconstructed input $\hat{y}$

Layer 2 features $p_2$

$L_0.5$ Sparsity
Filter Learning

- Goal: update convolutional filters $f$
- Fixed feature maps $p$ & pooling variables $\theta$
  - 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

- **Input Image**
- **Reconstruction**
- **Reconstructed Layer 1 feature maps**
- **48 Layer 2 feature maps**

Unpool & Convolution with Layer 2 filters

Unpool & Convolution with Layer 1 filters
Reconstructions from 2 layer model
Reconstructions from 1 layer model
Gaussian Pooling in 2 layer model

- Projection of 2\textsuperscript{nd} layer features activations down to pixels, using inferred pooling variables
Algorithm 1 Learning with Differentiable Pooling in De-convolutional Networks

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

1: for $l = 1 : L$ do % Loop over layers
2:   Init. features/filters: $p^i_l \sim 0$, $f_l \sim \mathcal{N}(0, \epsilon)$
3:   Init. switches: $\theta^i_l = \text{Fit}(R^T_l y^i) \quad \forall i$
4: for epoch $= 1 : E$ do % Epoch iteration
5:   for $i = 1 : N$ do % Loop over images
6:     for $t = 1 : T$ do % ISTA iteration
7:       Reconstruct input: $\hat{v}^i_l = R_l p^i_l$
8:       Compute reconstruction error: $e = \hat{v}^i_l - v^i$
9:       Propagate error up to layer $l$: $\nabla p_l = R^T_l e$
10:      Estimate step size $\beta_{p_l}$ as in Eqn. 15
11:     Take gradient step on $p$: $p^i_l = p^i_l - \lambda_l \beta_{p_l} \nabla p_l$
12:     Perform shrink: $p^i_l = \max(|p^i_l| - \beta_{p_l}, 0) \text{sign}(p^i_l)$
13:     Project to positive: $p^i_l = \max(p^i_l, 0)$
14:     for $k = 1 : l$ do % Loop over lower layers
15:       Take gradient step on $\theta$: $\theta^i_k = \theta^i_k - \lambda_l \beta_{U_k} \nabla \theta_k$
16:     end for
17: end for
18: end for
19: Update $f_l$ by solving Eqn. 26 using CG
20: Project $f_l$ to positive and unit length
21: end for
22: end for
23: Output: filters $f$, feature maps $p$ and pooling variables $\theta$. 
Related Work

• What / Where Separation
  – Transforming Auto-Encoders [Hinton et al. ICANN’11]
  – [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]
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 $\rightarrow$ linear SVM
Gaussian vs Max Pooling

- Compare objective cost to conventional max pooling

Classification:

<table>
<thead>
<tr>
<th></th>
<th>Layer 1</th>
<th>Layer 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Pooling</td>
<td>1.30%</td>
<td>1.25%</td>
</tr>
<tr>
<td>Gaussian Pooling</td>
<td>1.38%</td>
<td>0.84%</td>
</tr>
</tbody>
</table>

L0 sparsity is much lower
Joint vs Separate Inference

Separate inference (2 phases):
(i): Layer 1 infer \((\theta_1, p_1)\) then fix \(\theta_1\)  
(ii): Layer 2 infer \((\theta_2, p_2)\)

Joint Inference (1 phase):
(i): Layer 1 & 2 infer \((\theta_1, \theta_2, p_2)\)

0.84% error

1.39% error

Using top-down information to set lower pooling variables: 40% reduction in error
Number of ISTA Iterations (Test)

Inference ISTA steps

% Error rate

$\lambda_2 = 10$

$\lambda_2 = 5$

$\lambda_2 = 2$
L0.5 vs L1 Sparsity in Train/Test
## Non-Negativity Results

<table>
<thead>
<tr>
<th></th>
<th>Positive/Negative</th>
<th>Non-negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Pooling</td>
<td>2.04%</td>
<td>1.25%</td>
</tr>
<tr>
<td>Gaussian Pooling</td>
<td>2.32%</td>
<td>0.84%</td>
</tr>
</tbody>
</table>

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 reset | Without 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Pre-training</th>
<th>Fine-tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.84%</td>
<td>-</td>
</tr>
<tr>
<td>Convolutional RBM [Lee et al. ICML'09]</td>
<td>0.82%</td>
<td>-</td>
</tr>
<tr>
<td>Deep Belief Net [Hinton et al. 2006]</td>
<td>2.5%</td>
<td>1.18%</td>
</tr>
<tr>
<td>Deep Boltzmann Machine [S &amp; H AISTATS'09]</td>
<td>-</td>
<td>0.95%</td>
</tr>
</tbody>
</table>
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 contrast normalization
Reversible Max Pooling

Max Locations
“Switches”

Pooling

Unpooling
## Model Parameters/Statistics

<table>
<thead>
<tr>
<th>Property</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Layer 3</th>
<th>Layer 4</th>
</tr>
</thead>
<tbody>
<tr>
<td># Feature maps $K_l$</td>
<td>15</td>
<td>50</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>Pooling size</td>
<td>3x3x3</td>
<td>3x3x2</td>
<td>3x3x2</td>
<td>3x3x2</td>
</tr>
<tr>
<td>$\lambda_l$</td>
<td>2</td>
<td>0.1</td>
<td>0.005</td>
<td>0.001</td>
</tr>
</tbody>
</table>

- 7x7 filters at all layers
Model Reconstructions

Input
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

Max 1
Max 2
Max 3
Max 4
Max 5

Input Image
## Classification Results: Caltech 101

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our model - layer 1</td>
<td>67.8 ± 1.2%</td>
</tr>
<tr>
<td>Chen et al. [3] layer-1+2 (ConvFA)</td>
<td>65.7 ± 0.7%</td>
</tr>
<tr>
<td>Kavukcuoglu et al. [8] (ConvSC)</td>
<td>65.7 ± 0.7%</td>
</tr>
<tr>
<td>Zeiler et al. [20] layer-1+2 (DN)</td>
<td>66.9 ± 1.1%</td>
</tr>
<tr>
<td>Boureau et al. [2] (Macrofeatures)</td>
<td>70.9 ± 1.0%</td>
</tr>
<tr>
<td>Jarrett et al. [7] (PSD)</td>
<td>65.6 ± 1.0%</td>
</tr>
<tr>
<td>Lazebnik et al. [9] (SPM)</td>
<td>64.6 ± 0.7%</td>
</tr>
<tr>
<td>Lee et al. [11] layer-1+2 (CDBN)</td>
<td>65.4 ± 0.5%</td>
</tr>
</tbody>
</table>

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
Layer 2 Filter Coefficients
Max Pooling Projections (Aliasing)
# Joint vs Separate Inference

<table>
<thead>
<tr>
<th>Training</th>
<th>Infer 2 phases</th>
<th>Infer 1 phase (no $U_1$)</th>
<th>Infer 1 phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updating $F_1$ $U_{w_1}$</td>
<td>1.79%</td>
<td>1.63%</td>
<td>1.40%</td>
</tr>
<tr>
<td>Updating $F_1$</td>
<td>1.71%</td>
<td>1.21%</td>
<td>1.10%</td>
</tr>
<tr>
<td>Updating $U_{w_1}$</td>
<td>1.39%</td>
<td>1.04%</td>
<td>0.84%</td>
</tr>
<tr>
<td>No Layer 1 Updates</td>
<td>1.46%</td>
<td>0.99%</td>
<td>1.03%</td>
</tr>
</tbody>
</table>

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.
Comparison: Convolutional Nets

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.

LeCun et al. 1989
Analysis of Switch Settings

- Recons. and classification with various unpooling.

![Reconstructions from Layer 3 using Modified Switches](Image)

![Graphs showing mean confusion for different layers](Image)
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]