Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers

Clément Farabet, Camille Couprie, Laurent Najman, Yann LeCun
Published in the proceeding of ICML 2012

Presented by Bo Chen
10.05, 2012

Most of slide content can be found at Clément’s homepage.
Outline

1. Scene parsing
2. Classic Approaches
3. Laplacian pyramid
4. Convolutional networks
5. The methodology in this paper
6. Experimental results
Scene Parsing

- The goal of Scene Parsing is to find a decomposition of a scene into objects.

Detection Recognition Delineation
Classic Approach

1. Representation
   a) decompose the scene into candidate segments (super-pixels)
   b) represent each segment by a feature vector (SIFT/HoG, color and geometry features)

2. Prediction
   a) Supervised models such as Markov random field and conditional random fields
   b) Jointly model the objects in the scene based on spatial location and local appearance
   c) Approximate inference

Other methods:
   recursive neural networks and hierarchical cluster tree
Laplacian Pyramid

The figure got from http://jbhuang1.projects.cs.illinois.edu/cs498dwh/proj1/
Convolutional Networks

1. Process full image
2. Each layer is fully trainable
3. Nonlinear function with pooling operation
Methodology

3. Optimal cover of the pixels: finding the purest labeling, *i.e.* for each pixel the parent component with the purest distribution is chosen

2. Purity Tree: using a hierarchy of segmentations and a class purity criterion, to decide if a segment contains one or more objects/classes

1. Multi-scale Feature Learning: using a multiscale convolutional network, trained to produce local to global features for region classification
Multiscale Convolutional Networks

1. Three-scale Laplacian pyramid
2. Parallelly process all scales by CovNet with pixelwise linear classifier, where weights are shared by different scales

\[ H_0 = X_s \longrightarrow H_{lp} = \text{maxp} \left( \tanh(b_{lp} + \sum_{q \in \text{parents}(p)} w_{lpq} \ast H_{l-1,q}) \right) \longrightarrow F_S = H_N \]

multiscale convnet: \[ F = \{F_1, u(F_2), \ldots, u(F_N)\} \]

supervised dataset: \( \{X, c\} \)

classification loss: \( L_{\text{cat}} = \sum_{i \in \text{pixels}} l_{\text{cat}}(\hat{c}_i, c_i) \)

pixelwise classifier: \( \hat{c}_{i,a} = \frac{e^{w_a^T F_i}}{\sum_{b \in \text{classes}} e^{w_b^T F_i}} \)

\( l_{\text{cat}}(\hat{c}_i, c_i) = -\sum_{a \in \text{classes}} c_{i,a} \ln(\hat{c}_{i,a}) \)
An Example by Convolutional Factor Analysis

1. Image content is, in principle, scale invariant
2. Performance decreases when removing the weight-sharing.

Reconstructed images by the left weight at different scales

Weights shared across scales

Laplacian map
Feature Extraction Before Purity Tree

Given the features achieved by the trained convolutional networks:

Since it is not easy to predict label only based on a single pixel feature vector, we define the neighborhood of a pixel as a connected component that contains this pixel, which is a spatial grouping of feature vectors around the pixel, i.e. a neighborhood.

Each component is described by a grid of feature vectors:
- a 3x3 grid of 768-dim feature vectors
- that yields a 7000-dim descriptor

Good properties:
(1) elongated, or in general ill-shaped objects, are nicely handled
(2) the dominant features are used to represent the object, combined with background subtraction.
Purity Tree

Purity tree:
1. its leaves are the pixels of the image
2. its nodes are groupings of pixels
3. its root is the complete image
4. each node is described by:
   1) a cost (the purity of the node)
   2) a histogram of classes present in the node
Train Parameters in Purity Tree

1. Each descriptor is then projected through a 2-layer neural network, with 1024 hidden units, to produce an n-dim distribution (n classes):

\[ y_k = W_2 \tanh(W_1 x_k + b_1), \]

\[ \hat{d}_{k,a} = \frac{e^{y_{k,a}}}{\sum_{b \in \text{classes}} e^{y_{k,b}}}, \]

Predicted class distribution

\[ S_k = -\sum_{a \in \text{classes}} \hat{d}_{k,a} \ln(\hat{d}_{k,a}). \]

Cost with the distribution

2. This network is trained using KL-divergence, to approximate the histogram of classes present in each component

\[ l_{\text{div}} = \sum_{a \in \text{classes}} \hat{d}_{k,a} \ln\left(\frac{\hat{d}_{k,a}}{d_{k,a}}\right) \]
Optimal Cover

Optimal cover of the pixels:

1. Each component in the tree is now described by an estimate of the class histogram
2. Each pixel in the scene belongs to multiple pixel groupings in the tree
3. The one that has the lowest entropy is chosen to explain the pixel, and assign its class
4. A cover has interesting properties, as it is a set of non-disjoint components (allows for hierarchical classes)
Experimental Results I

1. Stanford background data: 572 images used for training, and 143 for testing

<table>
<thead>
<tr>
<th>8-class problem</th>
<th>Gould et al.</th>
<th>Munoz et al.</th>
<th>Socher et al.</th>
<th>Kumar et al.</th>
<th>Lempitsky et al.</th>
<th>Multiscale ConvNet + Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>76.4</td>
<td>76.9</td>
<td>78.1</td>
<td>79.4</td>
<td><strong>81.9</strong></td>
<td>79.5</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>66.2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td><strong>72.4</strong></td>
</tr>
<tr>
<td>Inference Time</td>
<td>1 to 10min</td>
<td>10sec</td>
<td>?</td>
<td>1 to 10min</td>
<td>more than 1min</td>
<td><strong>1sec</strong></td>
</tr>
</tbody>
</table>

results on the Stanford Background Dataset (Gould et al)
Experimental Results II

2. SIFTFlow: 2,488 training images and 200 test images

<table>
<thead>
<tr>
<th>33-class problem</th>
<th>Liu et al.</th>
<th>Tighe et al.</th>
<th>MCNet+Tree Natural Freqs</th>
<th>MCNet+Tree Freq Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>76.4</td>
<td>76.9</td>
<td><strong>78.5</strong></td>
<td>74.2</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>29.4</td>
<td><strong>29.6</strong></td>
<td><strong>46.0</strong></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>170-class problem</th>
<th>Tighe et al.</th>
<th>MCNet+Tree Natural Freqs</th>
<th>MCNet+Tree Freq Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>66.9</td>
<td><strong>67.8</strong></td>
<td>39.1</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>7.6</td>
<td>9.5</td>
<td><strong>10.7</strong></td>
</tr>
</tbody>
</table>
Experimental Results III
Experimental Results IV

self-contained battery-powered system for real-time 360-degree scene parsing
Experimental Results V

trained on only 2500 images, 33 classes, tested (here) on arbitrary New York scenes (360 degree)