Real-time Scene Parsing with multi-scale feature learning, and a touch of custom hardware

joint work with:
Camille Couprie
Laurent Najman
Yann LeCun
Berin Martini
Eugenio Culurciello

Clément Farabet

NYU – Courant Institute of Mathematical Sciences | Université Paris Est | Purdue University
Real-time Scene Parsing

Finding a decomposition of the scene into objects (objects need not be disjoint)

implies 3 sub-problems:

- Detection,
- Recognition,
- Delineation

of all objects in the scene
OVERVIEW
in pixel space, prediction of high-level concepts (e.g. object categories, attributes) is hard, if not impossible

the image needs to be encoded into a representation that allows easy (i.e. linear) prediction

in most recent works, this encoding involves two steps:

- decomposing the scene into candidate segments, which is typically done using a graph representation of the image and some merging criterion

- representing each segment by a feature vector, which is typically based on SIFT/HoG + color + geometry features
prediction of object categories and/or attributes is often done using a global, scene-level model, e.g. conditional random fields

such models capture joint probabilities between objects in the scene, based on their spatial relationship, and local appearance

these models typically involve loopy graphs, and rely on approximate inference to achieve decent compute times

To simplify and speed up prediction, we need a better representation!
We use **convolutional networks** (convnets) to:

- learn **hierarchical representations** of images, with little prior on their statistics (any images can be modeled, not just RGB)
- learn **global descriptors** that encode both **local appearance** and **scene-level, contextual** relationships between objects
- **alleviate the need for “complex” inference procedures**: the convnet produces a powerful representation that simplifies prediction

Better representation = simpler prediction
Convolutional Networks:

- can learn **hierarchical representations** of images (any modalities)

- rely on a sequence of **linear layers** (simple cells) interspersed with **non-linear pooling layers** (complex cells)

- **simple cells** are learned at all layers, to maximize their utility, and yield efficient/compact codes

- **complex cells**, or pooling layers, enforce invariances into the model, that are gradually increasing through the hierarchy
Convolutonal Networks:

- **detection, recognition** and **delineation** can be solved within a single energy-based model
- the model can parse a complete scene (convolutionally) and produce a dense map of probability distributions (classes)
- the model can be **trained end-to-end**: each layer/stage in the model is a non-linear, fully trainable transform: no engineered features or prior knowledge are required
Convolutional Networks:

- the model has an **homogenous architecture**: a minimal set of operators is sufficient to compute it

- the model is **intrinsically parallel**: hidden units could be computed by thousands of asynchronous/parallel processors

- inference is **feed-forward**, and therefore deterministic in time

- the model can accommodate **low numeric precision** (this low precision can be even be accounted for during training)
- **Multi-scale Feature Learning**
  using a multiscale convolutional network, trained to produce local to global features for region classification

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

- **Optimal (minimum entropy) cover of the pixels**
  finding the purest labeling, i.e. for each pixel the parent component with the purest distribution is chosen
Multiscale Feature Learning

- our model is a **multi-scale generalization of convolutional networks**: weights at each layer are **replicated** both **spatially** and in **scale-space**

- each feature vector represents a **local focus point** as well as its **context**, up to the complete scene (analogy with the fovea: **foveated focus**)

- a complete scene can be **parsed efficiently**, and **densely**
Multiscale Feature Learning

- normalized RGB on Laplacian pyramid

- 3-layer network, replicated for each scale
  - L1: 16 features (max-pooled over 2x2 neighborhoods)
  - L2: 64 features (max-pooled over 2x2 neighborhoods)
  - L3: 256 features

- 3-scale pyramid yields a 768-dim feature vector at each location
Multiscale Feature Learning

**Model**

- Purity Tree

**Results**

**Overview**

**Perspectives**

Optimal cover

Convnet at each scale:

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

Multiscale convnet:

\[ F = [F_1, u(F_2), \ldots, u(F_N)] \]

Supervised dataset:

\[ \{X, c\} \]

Pixelwise training:

\[ \hat{c}_{i,a} = \frac{e^{w^T a F_i}}{\sum_{b \in \text{classes}} e^{w^T b F_i}} \]

Classification loss:

\[ L_{\text{cat}} = \sum_{i \in \text{pixels}} l_{\text{cat}}(\hat{c}_i, c_i) \]

\[ l_{\text{cat}}(\hat{c}_i, c_i) = -\sum_{a \in \text{classes}} c_{i,a} \ln(\hat{c}_{i,a}) \]
What is a purity tree?

- it is a tree!
- its leaves are the pixels of the image
- its nodes are groupings of pixels
- its root is the complete image
- each node is described by:
  - a cost (the purity of the node)
  - a histogram of classes present in the node
we compute the purity tree by:

- computing a segmentation tree over the pixels (lots of standard methods to do that: plain gradient-based, mean-shift, surface-normalized, ...)
- predicting the histogram of classes for each component
- setting the “impurity” cost of each component as the entropy of that histogram
Purity Tree

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

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

\[
\begin{align*}
\mathbf{y}_k &= \mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{x}_k + \mathbf{b}_1), \\
\hat{d}_{k,a} &= \frac{e^{\mathbf{y}_{k,a}}}{\sum_{b \in \text{classes}} e^{\mathbf{y}_{k,b}}}, \\
S_k &= -\sum_{a \in \text{classes}} \hat{d}_{k,a} \ln(\hat{d}_{k,a}).
\end{align*}
\]

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

\[
l_{\text{div}} = \sum_{a \in \text{classes}} \hat{d}_{k,a} \ln\left(\frac{\hat{d}_{k,a}}{d_{k,a}}\right)
\]
we produce an optimal cover:

- each component in the tree is now described by an estimate of the class histogram (k-dim)

- we find an optimal cover of the pixels:
  - each pixel in the scene belongs to multiple pixel groupings in the tree
  - the one that has the lowest entropy is chosen to explain the pixel, and assign its class
  - a cover has interesting properties, as it is a set of non-disjoint components (allows for hierarchical classes)

\[ k^*(i) = \arg\min_{k \mid i \in C_k} S_k \]
RESULTS
<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>81.9</td>
<td>79.5</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>66.2</td>
<td>?</td>
<td>?</td>
<td>72.4</td>
<td>74.3</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>1sec</td>
</tr>
</tbody>
</table>

results on the Stanford Background Dataset (Gould et al)
<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>78.5</td>
<td>74.2</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>29.4</td>
<td>29.6</td>
<td>46.0</td>
</tr>
</tbody>
</table>

results on the SiftFlow Dataset (Liu et al)
### 170-class problem

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

results on the Barcelona Dataset (Tighe et al)
**OVERVIEW**

Accuracy

**RESULTS**

Compute Time

**Min-cover**  
10ms

**Histograms**  
400ms

**SegmTree**  
70ms

**Features**  
500ms

**REAL-WORLD RESULTS**

**Intel i7, 4-core**
neuFlow

is a dataflow processor architecture running on off-the-shelf FPGAs

- reduced flow control
- a grid of dataflow processing units:
  - each unit is 100% data-driven
  - data flows across units using efficient data buses
- the grid can be efficiently reconfigured at runtime to unroll algorithms in time

Joint work:
C.Farabet, B.Martini
E.Culurciello, Y.LeCun
simplified model for neuFlow:

**Classification**  10ms

**Flat cut in tree**  30ms

**Features**  80ms
results on the SiftFlow Dataset (Liu et al)
trained on only 2500 images, 33 classes, 
tested (here) on arbitrary new york scenes (360 degree)
our current model is based on an arbitrary segmentation tree, which contains few, if any object-level component.

- Arbelaez and Malik proposed a way of learning hierarchies of segmentations, and demonstrated nice results on the Berkeley dataset. Their method is extremely slow (> 1 minute per image).

- Turaga et al. proposed a way of learning a ultrametric distance to produce object-level segmentation. This type of approach could be applied to produce better segmentation trees.
learning our representation yields a non-convex optimization problem, which requires some tuning (learning rate, frequency balancing, ...)

- our representation could be pre-trained using deep-learning methods, e.g. with greedy layer-wise pre-training

- non-convexity does not seem to be a big problem anyway: given enough data, the function is eventually learned
THANK YOU!

www.clement.farabet.net