Regionlet Object Detector with Hand-crafted and CNN Feature

Xiaoyu Wang
Snapchat Research

Xiaoyu Wang
Snapchat Research

Ming Yang
Horizon Robotics

Shenghuo Zhu
Alibaba Group

Yuanqing Lin
Baidu
Overview of this section

- Regionlet Object Detector
- Regionlet Localizer (re-localization)
- Regionlet with Deep CNN Feature
  - CNN Feature Extraction
  - Support Pixel Integral Image
- Application Examples
  - Car Detection for Fine-grained Image Classification
  - Pedestrian, Car, Cyclist Detection for Autonomous Driving
What is Regionlet Object Detector

• A significant extension to traditional boosting object detector

• Together with OverFeat and R-CNN, the Regionlet detector is one of the first several detectors that successfully adopt deep CNN features for generic object detection.
How does Regionlet detector connect to past/future

RealBoost
Segmentation as Selective Search
Low-level Feature
Deep CNN

Boosting Feature Selection
Object Proposal
Generalized Spatial Pyramid for CNN Feature Pooling

Spatial Pyramid Pooling in SPP-Net
RoI Pooling in Fast R-CNN

Boosting Object Detector

$$f(X) = \sum_{i=0}^{N} \beta_i h_i(x_i)$$

A detection window

A sub-region where weak classifier is built based on

Weak classifier
Traditional Boosting Detection Framework

- Operate on multiple scales to detect objects in different scales
- Use multiple components to detect objects with various aspect ratios

How about a single model, but flexible during testing, no feature pyramids, no multiple components
What the Regionlet Detector Proposed

• A boosting classifier that can take inputs of different scales

• A boosting classifier that can take inputs of different viewpoints

• A boosting classifier containing feature pooling learning
Regionlet: Definition

- **Region**($R$): Feature extraction region
- **Regionlet**($r_1, r_2, r_3$): A sub-region in a feature extraction area whose position/resolution are relative and normalized to a detection window
Regionlet: Definition (cont.)

• Regionlet coordinates are normalized

\[ (l, t, r, b) \]
\[ (.25, .25, .90, .90) \]
Regionlet: Definition (cont.)

• Regionlet definition = Generalized Spatial Pyramid
  • Similar
    • Both use relative coordinates
  • Difference
    • Regionlet: coordinates are relative to the detection window (not the image)
    • Regionlet: coordinates are flexible (do not have to evenly divide the image/window)
• Regionlet feature extraction = Generalized Spatial Pyramid Pooling
Connection to other methods in pooling design

Object Proposal

Generalized Spatial Pyramid for CNN Feature Pooling

CNN-based Object Detection

Spatial Pyramid Pooling in SPP-Net\(^1\)

RoI Pooling in Fast R-CNN\(^2\)

Regionlet: Feature extraction

Could be Hand-crafted features or deep CNN features, whatever feature your like!
Regionlet Classifier

- Each weak classifier is based on a 1-D feature extracted from a region

\[
h(x) = \sum_{o=1}^{n-1} v^o \mathbb{1}(B(x) = 0)\]

\[
H(X) = \sum_{i=1}^{T} \beta_i h_i(x_i)\]
Detection Framework


(a) : Input image
(b) : Generate object regions\(^1,2,3\)
(c) : Feature extraction and pooling
  - Generalized Spatial Pyramid Pooling inside Regionets
  - Low-level features
  - CNN features (will talk later)
  - Max-pooling among Regionlets
Multiple scale & viewpoints Handling

Regionlet Model

Adjusting the model to a candidate bounding box

Not a motorbike
Multiple scale & viewpoints Handling

Regionlet Model

Motorbike Detected

Adjusting the model to a candidate bounding box
Weak Classifier Construction

- Weak learner on each REGION
  - Eight lots lookup table
  - Lookup table is learned
  - Lot value is learned
  - One lot is activated for one feature

\[ H(X) = \sum_{i=1}^{N} LUT_i(x_i) \]

Regionlet feature \( x_i \) (after pooling)

Assign lot

<table>
<thead>
<tr>
<th>Lot Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
</tr>
<tr>
<td>-0.2</td>
</tr>
<tr>
<td>-0.5</td>
</tr>
<tr>
<td>-0.4</td>
</tr>
<tr>
<td>0.02</td>
</tr>
<tr>
<td>0.15</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.3</td>
</tr>
</tbody>
</table>

Weak learner output: -0.5
Regionlet Training

- How to get regions and regionlets
  - Regions
    - Regions are randomly sampled
    - Effective Regions are greedily selected to reduce learning cost
  - Regionlets
    - Each Region & Regionlet configuration are randomly configured
    - A Region and its regionlets configuration are selected simultaneously
  - Region & Regionlet pool is fixed for each cascade learning
Regionlet: Training

• Constructing the regions/regionlets pool
  • Small region, fewer regionlets -> fine spatial layout
  • Large region, more regionlets -> robust to deformation

• Learning realBoost\(^1\) cascades
  • 16K region/regionlets candidates for each cascade
  • Learning of each cascade stops when the error rate is achieved (1% for positive, 37.5% for negative)
  • Last cascade stops after collecting 5000 weak classifiers
  • Result in 4-7 cascades
  • 2-3 hours to finish training one category on a 8-core machine

Regionlet: Testing

• No image resizing
• Any scale, any aspect ratio
• Adapt the model size to the same size as the object candidate bounding box
Overview of this section

- Regionlet Object Detector
- Regionlet Localizer
- Regionlet with Deep CNN Feature
  - CNN Feature Extraction
  - Support Pixel Integral Image
- Application Examples
  - Car Detection
  - Pedestrian, Car, Cyclist Detection for Autonomous Driving
Regionlet Localizer (object re-localization)

- Why a localizer is needed (classification & localization precision dilemma)

Data augmentation during **training** to accommodate inaccurate localization

As accurate location as possible during **testing**
Regionlet Localizer

- Regionlet feature can be reused for localization

- Each Regionlet feature is associated with a spatial location

- The location is learned during classifier training
Regionlet Localizer

- Regionlet feature can be reused for localization

Regionlet classifier 1

0 0 0 0 0 0 1 0

Regionlet classifier N

0 0 0 0 0 0 0 1 0

8N dimensional binary vector
Regionlet Localizer

$$\Delta l, \Delta t, \Delta r, \Delta b$$

$$\times W$$

\[\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\cdots & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\]
Regionlet Localizer Training

- Random sample examples which have > 0.6 overlap with ground truth
  - Less overlap gives poor results

- The regression task learns the location difference
Regionlet Localizer

- Experiment result on our car dataset for autonomous driving
  - 17501 cars for training
  - 12546 cars for testing

Detection performance (% AP)

<table>
<thead>
<tr>
<th></th>
<th>0.5 overlap</th>
<th>0.7 overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regionlet</td>
<td>62.7%</td>
<td>34.6%</td>
</tr>
<tr>
<td>Regionlet + localization</td>
<td><strong>65.3%</strong></td>
<td><strong>43.9%</strong></td>
</tr>
<tr>
<td>Improvement</td>
<td>2.6%</td>
<td>9.1%</td>
</tr>
</tbody>
</table>
Overview of this section

• Regionlet Object Detector
• Regionlet Localizer
• Regionlet with Deep CNN Feature
  • CNN Feature Extraction
  • Support Pixel Integral Image
• Application Examples
  • Car Detection
  • Pedestrian, Car, Cyclist Detection for Autonomous Driving
Regionlet with DCNN

- Deep CNN
  - Deep structure learns high-level information
  - Max-pooling is robust to parts misalignment
  - Information are jointly learned

- How to establish a bridge for DCNN and Regionlet object detection framework?
Regionlet with DCNN

- Deep CNN structure
  - Features from convolution layers retain spatial information

Convolutional layers
Regionlet with DCNN

- Deep CNN structure
  - Features from convolution layers retain spatial information
Regionlets with DCNN

- Deep CNN structure
  - ‘image-convolution’ to generate features for the whole image

(a) Convolution kernels

(b) Output the dense neural patterns
Support Pixel Integral Image

• CNN Feature
  • Not available on each pixel
  • Feature dimension is high for integral image

\[
I(x, y) = \sum_{i=0, j=0}^{i \leq x, j \leq y} p(i, j)
\]

\[
= \sum_{i=0, j=y}^{i \leq x, j=y} p(i, j) + \sum_{i=0, j=0}^{i \leq x, j \leq y-1} p(i, j)
\]

\[
= \sum_{i=0}^{x} p(i, y) + I(x, y-1)
\]

\[
= \text{RowSum}(x, y) + I(x, y-1).\]

Does not change if \( \text{rowsum} = 0 \)

We want dense integral feature

We want to save memory

\[
I(x, y) = \sum_{j=0}^{y} p(x, y) + I(x-1, y)
\]

\[
= \text{ColSum}(x, y) + I(x-1, y).
\]

Does not change if \( \text{colsum} = 0 \)
Support Pixel Integral Image

- Support Pixel
  - Where the integral vector computation is inevitable
Support Pixel Integral Image

• Support Pixel Integral Image for CNN Feature\(^1\)

Regionlet with DCNN

- Deep CNN feature for detection\(^1,\,^2\)

1. Zou et al. Generic Object Detection with Dense Neural Patterns and Regionlets. BMVC, 2014
Experiment Results

• Regionlet + CNN feature (No fine-tuning)

Average precision on PASCAL VOC 2007 (%)

<table>
<thead>
<tr>
<th></th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regionlet</td>
<td>41.7</td>
</tr>
<tr>
<td>Regionlet-CNN pool5</td>
<td>49.3</td>
</tr>
<tr>
<td>R-CNN pool5</td>
<td>44.2</td>
</tr>
<tr>
<td>R-CNN FT fc7 BB</td>
<td>58.5</td>
</tr>
</tbody>
</table>
Experiment Results

• Visualization of selected neuron patterns

Figure 8. Visualization of the high-level information encoded by neural patterns from the fifth convolutional layer. The patches are obtained by: 1) Determine the most frequently selected neural pattern dimension (1 out of 256) for an object category. 2) Run the neural pattern extractor as a detector, using the value of the extracted neural patterns as detection scores. 3) Collect and rank detection results, visual patches with larger neural pattern values are ranked top.
Running speed

• Regionlet detector + localizer: 5 fps using a single CPU core, (>30 fps) using 8 cores
• Regionlet + CNN: 3fps using a single CPU core
Application examples

- Car, pedestrian, cyclist detection for autonomous driving
  - Ranked #1 on KITTI detection dataset

### Car Detection (mAP %)

<table>
<thead>
<tr>
<th></th>
<th>Moderate</th>
<th>Easy</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DVP</td>
<td>75.77%</td>
<td>87.46%</td>
<td>65.38%</td>
</tr>
<tr>
<td>SS</td>
<td>74.30%</td>
<td>85.03%</td>
<td>59.48%</td>
</tr>
<tr>
<td>SVM-Res</td>
<td>67.49%</td>
<td>78.11%</td>
<td>54.28%</td>
</tr>
<tr>
<td>Regionlet</td>
<td><strong>76.45%</strong></td>
<td>84.75%</td>
<td>59.70%</td>
</tr>
</tbody>
</table>
Application examples

- Car, pedestrian, cyclist detection for autonomous driving
  - Ranked #1 on KITTI detection dataset

Pedestrian Detection (mAP %)

<table>
<thead>
<tr>
<th>Method</th>
<th>Moderate</th>
<th>Easy</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>pAUCEnsT $^1$</td>
<td>54.49%</td>
<td>65.26%</td>
<td>48.60%</td>
</tr>
<tr>
<td>R-CNN</td>
<td>50.13%</td>
<td>61.61%</td>
<td>44.79%</td>
</tr>
<tr>
<td>Fusion-DPM</td>
<td>46.67%</td>
<td>59.51%</td>
<td>42.05%</td>
</tr>
<tr>
<td>Regionlet</td>
<td>61.15%</td>
<td>73.14%</td>
<td>55.21%</td>
</tr>
</tbody>
</table>

1. S. Paisitkriangkrai, C. Shen and A. Hengel, Pedestrian Detection with Spatially Pooled Features and Structured Ensemble Learning
Application examples

- Car, pedestrian, cyclist detection for autonomous driving
  - Ranked #1 on KITTI detection dataset

### Cyclist Detection (mAP %)

<table>
<thead>
<tr>
<th>Method</th>
<th>Moderate</th>
<th>Easy</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>pAUCEnsT</td>
<td>38.03%</td>
<td>51.62%</td>
<td>33.38%</td>
</tr>
<tr>
<td>R-CNN</td>
<td>34.47%</td>
<td>50.07%</td>
<td>32.12%</td>
</tr>
<tr>
<td>DF+DPM+ROI</td>
<td>30.90%</td>
<td>41.86%</td>
<td>27.75%</td>
</tr>
<tr>
<td>Regionlet</td>
<td><strong>58.72%</strong></td>
<td><strong>70.41%</strong></td>
<td><strong>51.83%</strong></td>
</tr>
</tbody>
</table>
Application examples

- Car detection
  - Totally 170K cars for fine-grained car recognition
  - 11000 labelled for training detector
  - 2745 labelled for testing
  - 100% AP
Take Away Messages

• Regionlet extend the traditional boosting object detector
  • It operates on object proposal (the \textit{generalized spatial pyramid} pooling is the key)
  • It integrates a pooling process
  • It can easily integrate various features

• Regionlet is \textbf{FAST} (5 fps, single core CPU)

• Regionlet is \textbf{FAST even with CNN Feature} (3fps, single core CPU)
Thank you!

We are hiring!