Recognize Objects

Gluing Pixels

CIS 680, Spring '17
recognize objects: Describe and Discriminate

- **Describe** positives by commonalities of shape
- **Discriminate** negatives by uniqueness of shape
recognize objects: Frame ‘it’ right
Challenge 1: maintain discriminative under deformation

Similarity of appearance near feature points

Similarity in configuration of the feature points
The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features

Kristen Grauman
Trevor Darrell
Sets of features

\[ X = \{ \vec{x}_1, \ldots, \vec{x}_m \} \]

\[ Y = \{ \vec{y}_1, \ldots, \vec{y}_n \} \]
Pyramid match

Related: Earth Moving Distance (EMD)

$X = \{\vec{x}_1, \ldots, \vec{x}_m\}$  $\vec{x}_i \in \mathbb{R}^d$

$Y = \{\vec{y}_1, \ldots, \vec{y}_n\}$  $\vec{y}_i \in \mathbb{R}^d$

$$\max_{\pi: X \rightarrow Y} \sum_{x_i \in X} S(x_i, \pi(x_i))$$

optimal partial matching
Pyramid match overview

Pyramid match kernel measures similarity of a partial matching between two sets:

- Place multi-dimensional, multi-resolution grid over point sets
- Consider points matched at finest resolution where they fall into same grid cell
- Approximate similarity between matched points with worst case similarity at given level

No explicit search for matches!
Pyramid match kernel

\[ K_\Delta = \sum_{i=0}^{L} w_i N_i \]

Number of newly matched pairs at level \( i \)

Approximate partial match similarity

Measure of difficulty of a match at level \( i \)
Feature extraction

\[ X = \{ \bar{x}_1, \ldots, \bar{x}_m \}, \quad \bar{x}_i \in \mathbb{R}^d \]

Histogram pyramid: level \( i \) has bins of size \( 2^i \)

\[ \Psi(X) = [H_0(X), \ldots, H_L(X)] \]
Counting matches

Histogram intersection

\[ I(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j) \]
Counting new matches

Histogram intersection

\[ \mathcal{I}(H(X), H(Y)) = \sum_{j=1}^{r} \min(H(X)_j, H(Y)_j) \]

matches at this level

\[ N_i = \mathcal{I}(H_i(X), H_i(Y)) - \mathcal{I}(H_{i-1}(X), H_{i-1}(Y)) \]

matches at previous level

Difference in histogram intersections across levels counts *number of new pairs* matched
Pyramid match kernel

\[ K_\Delta (\Psi(X), \Psi(Y)) = \sum_{i=0}^{L} \frac{1}{2^i} \left( \mathcal{I}(H_i(X), H_i(Y)) - \mathcal{I}(H_{i-1}(X), H_{i-1}(Y)) \right) \]

- Weights inversely proportional to bin size
- Normalize kernel values to avoid favoring large sets

measure of difficulty of a match at level \( i \)
Example pyramid match

Level 0

$X$

$Y$

$H_0(X)$  $H_0(Y)$

$N_0 = 2$
$w_0 = 1$

$I_0 = 2$
Example pyramid match

Level 1

\[ N_1 = 4 - 2 = 2 \]
\[ \omega_1 = \frac{1}{2} \]

\[ H_1(X) \]
\[ H_1(Y) \]
\[ I_1 = 4 \]
Example pyramid match

Level 2

\[ N_2 = 5 - 4 = 1 \]
\[ w_2 = \frac{1}{4} \]

\[ H_2(X) \]
\[ H_2(Y) \]
\[ I_2 = 5 \]
Example pyramid match

\[ K_\Delta = \sum_{i=0}^{L} w_i N_i \]

\[ = 1(2) + \frac{1}{2}(2) + \frac{1}{4}(1) = 3.25 \]

Optimal match

\[ K = \max_{\pi: X \to Y} \sum_{x_i \in X} S(x_i, \pi(x_i)) \]

\[ = 1(2) + \frac{1}{2}(3) = 3.5 \]
100 sets with 2D points, cardinalities vary between 5 and 100.

Approximation of the optimal partial matching.
Localization

• Inspect intersections to obtain correspondences between features

• Higher confidence correspondences at finer resolution levels
Challenge: maintain discriminative under occlusion
Challenge 2: maintain discriminative under occlusion

Per Pixel Classification

Texton Boost for pixel level semantic classification

Jamie Shotton
Step 1: Texton Map generation (17 filters, K=400)

- Inputs
  - Texton Map
  - (Rectangle mask $r$, texton query $t$)
  - Pixel location $i$

- Output
  - Area in rectangle mask that match $t$

Step 2: Shape Filter

- For each texton $t$
  - End result is a texton histogram of area responses

How does this capture shape?
Shape Filters

- Pair: \((, )\)
  - rectangle \(r\)
  - texton \(t\)

- Feature responses \(v(i, r, t)\)

- Large bounding boxes enable long range interactions

- Integral images

```
\[ v(i_1, r, t) = a \]
\[ v(i_2, r, t) = 0 \]
\[ v(i_3, r, t) = a/2 \]
```

up to 200 pixels

appearance context
Shape as Texton Layout

Slides from Shotton’s ECCV talk

Shape as Texton Layout

\[(r_1, t_1) = (\text{texton map}, \text{ground truth})\]

\[(r_2, t_2) = (\text{feature response image})\]
Slides from Shotton’s ECCV talk

Shape as Texton Layout

\[(r_1, t_1) = \begin{pmatrix} \text{ground truth} \\ \text{texton map} \end{pmatrix} \]

\[(r_2, t_2) = \begin{pmatrix} \text{summed response images} \\ \text{texton map} \end{pmatrix} \]

\[v(i, r_1, t_1) + v(i, r_2, t_2)\]
Failures
Benchmark Challenges

VOC: 20 classes

COCO: 200 classes
Challenge 3: Dense vs Sparse

Learning Pictorial Structure (DPM)

Sliding windows.
- Score every subwindow.

Deformable part models (DPM)
- Uses HOG features
- Very fast

![Image showing sliding window and deformable part models for person and bottle]
1) fine level with deformable parts
2) coarse level with a fixed template model
combined cost of root (neck) locations

part detection cost

transformed cost

part detection cost
Challenge 3: Dense vs Sparse

Selective Search for Object Recognition

J.R.R. Uijlings\textsuperscript{*1,2}, K.E.A. van de Sande\textsuperscript{+2}, T. Gevers\textsuperscript{2}, and A.W.M. Smeulders\textsuperscript{2}

\textsuperscript{1}University of Trento, Italy
\textsuperscript{2}University of Amsterdam, the Netherlands

Technical Report 2012, submitted to IJCV
Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.
Figure 3: The training procedure of our object recognition pipeline. As positive learning examples we use the ground truth. As negatives we use examples that have a 20-50% overlap with the positive examples. We iteratively add hard negatives using a retraining phase.

Figure 5: Examples of locations for objects whose Best Overlap score is around our Mean Average Best Overlap of 0.879. The green boxes are the ground truth. The red boxes are created using the “Quality” selective search.
Lessons and Hard Cases:

**Hard Negative Mining**

Imbalance between positive and negative examples.

Use negative examples with higher confidence score.

**Non Maximum Suppression**

If output boxes overlap, only consider the most confident.
Region based classification: Proposal (where) and Checking (what)

R-CNN: Regions with CNN Features (Girshick et al., 2014)
Object Detection

Apply bounding-box regressors
Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Input image

Post hoc component

R-CNN: Regions with CNN Features (Girshick et al., 2014)
Training RCNN

Step 1: train your own CNN model for classification (or use existing model), using ImageNet dataset.
Training RCNN

Step 2: focus on 20 classes + 1 background. Remove the last FC layer and replace it with a smaller layer and fine-tune the model using PASCAL VOC dataset.
Training RCNN

Step 3: Extract Feature

Image

Crop & Warp

Convolution and Pooling

Store all the features after pool 5 layer and save to disk

Proposals

It is about ~ 200G features
Classify regions by SVM

- linear SVM per class
  (With the softmax classifier from fine-tuning mAP decreases from 54% to 51%)
- greedy NMS (non-maximum suppression) per class: rejects a region if it has an intersection-overunion (IoU) overlap with a higher scoring selected region larger than a learned threshold
Object proposal refinement
- Linear bounding-box regression on CNN features (pool_5 feature: mAP ~4% up)
Training R-CNN

• Bounding-box labeled detection data is scarce
• Use supervised pre-training on a data-rich auxiliary task and transfer to detection

Supervised pre-training
Pre-train CNN on ILSVRC2012 (1.2 million 1000-way image classification) using image-level annotations only

Domain-specific fine-tuning
Adapt to new task (detection) and new domain (warped proposal)

• random initialize (N+1)-way classification layer (N classes + background)
• Positives: 0.5 IoU overlap with a ground-truth box. Negative: o.w.
• SGD: learning rate: 0.001 (1/10 of original) mini-batch: 32 pos & 96 neg

Train binary SVM

• IoU overlap threshold: grid search over \{0, 0.1, \ldots, 0.5\}
  \( \text{IoU} = 0.5 : \text{mAP} \sim 5\% \text{ down} \)
  \( \text{IoU} = 0.0 : \text{mAP} \sim 4\% \text{ down} \)
Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

SPP-Net

Crop & Warp → Convolution and Pooling → Fully connected layer → Softmax loss

SPP pooling to a fix length layer (max pooling) → Softmax loss
SPP-Net

fully-connected layers ($fc_6$, $fc_7$)

fixed-length representation

......

spatial pyramid pooling layer

feature maps of conv3

window

convolutional layers

input image

Goal: Achieve a class-agnostic scalable object detection by predicting a set of bounding boxes.

Train a neural network to directly predict:

- The upper-left and lower-right coordinates of each bounding box
- The confidence score for the box containing an object
Fast RCNN

Share convolution layers for proposals from the same image

Faster and More accurate than RCNN

ROI Pooling

Fast R-CNN
Ross Girshick
Apr 2015

Fast RCNN

RoI Pooling

image ➔ Conv layer ➔ Max pooling ➔ Divided into h * w region ➔ Differentiable
What is bbox-regressor?
Bounding box regression

- Convolution and Pooling
- Fully connected layer
- Softmax loss
- Total loss
- L2 loss OR L1 loss
- 4 value (x, y, w, h)
- DeepPose, R-CNN
- Overfeat, VGG
Obstacle #1: 'Differentiable' RoI pooling

RoI pooling / 'SPP' is just like max pooling, 'except that' pooling regions overlap
Obstacle#'1:'Differentiable'RoI pooling

RoI pooling'/'SPP'is'just'like'max'pooling,'except'that'pooling'regions'
overlap

Obstacle #1: Differentiable RoI pooling

RoI pooling / SPP is just like max pooling, except that pooling regions overlap.

![Diagram showing RoI pooling]

Obstacle #1: Differentiable RoI pooling

RoI pooling / SPP is just like max pooling, except that pooling regions overlap.

\[
\frac{\partial L}{\partial x_i} = \sum \sum [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}}
\]

Partial from next layer

\[i^*(0,2) = 23\]
\[i^*(1,0) = 23\]

Object Detection

Fast R-CNN (Girshick, 2015)
Obstacle '#2: Making SGD steps efficient

Slow R)CNN and SPP net use region-wise sampling to make mini-batches

- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability
Obstacle '#2: Making SGD steps efficient

Note the receptive field for one example RoI is often very large

- Worst case: the receptive field is the entire image
Obstacle #2: Making SGD steps efficient

Worst case cost per mini-batch (crude model of computational complexity)

- \( \frac{128 \times 600 \times 1000}{(128 \times 224 \times 224)} = 12x \) computation than slow R-CNN
Obstacle #2: Making SGD steps efficient

Solution: use hierarchical sampling to build mini-batches

- Sample a small number of images (2)
- Sample many examples from each image (64)

Obstacle '#2: 'Making SGD steps efficient

Use the test time trick from SPP net during training

• Share computation between overlapping examples from the same image

Example 'RoI 1
Example 'RoI 2
Example 'RoI 3

Example 'RoI 1
Example 'RoI 2
Example 'RoI 3

Union of RoIs' receptive fields (shared computation)
Obstacle #2: Making SGD steps efficient

Cost per mini-batch compared to slow R-CNN (same crude cost model)

- $2 \times 600 \times 1000 / (128 \times 224 \times 224) = 0.19x < \text{computation than slow R-CNN}$
Obstacle '#2: 'Making SGD's steps efficient

Are the examples from just 2 images diverse enough?

- Concern: examples from the sample image may be too correlated
Fast R-CNN outcome

Better training time and testing time with better accuracy than slow R-CNN or SPP-net

• Training time: 84 hours / 25.5 hours / 8.75 hours (Fast R-CNN)
• VOC07 test mAP: 66.0% / 63.1% / 68.1%
• Testing time per image: 47s / 2.3s / 0.32s
  • Plus 0.2 to > 2s per image depending on proposal method
  • With selective search: 49s / 4.3s / 2.32s

Updated numbers from the ICCV paper based on implementation improvements

Experimental findings

• End-to-end training is important for very deep networks
• Softmax is a fine replacement for SVMs
• Multi-task training is beneficial
• Single-scale testing is a good tradeoff (noted by Kaiming)
• Fast training and testing enables new experiments
  • Comparing proposals
## Result compare

<table>
<thead>
<tr>
<th></th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train time (h)</td>
<td>9.5</td>
<td>84</td>
</tr>
<tr>
<td>-speedup</td>
<td>8.8x</td>
<td>1x</td>
</tr>
<tr>
<td>Test time/image</td>
<td>0.32s</td>
<td>47.00 s</td>
</tr>
<tr>
<td>-test speedup</td>
<td>146x</td>
<td>1x</td>
</tr>
<tr>
<td>mAP</td>
<td>66.9</td>
<td>66</td>
</tr>
</tbody>
</table>

Trained using VGG 16 on Pascal VOC 2007 dataset
Not including proposal time

Source: R. Girshick
## Result compare

<table>
<thead>
<tr>
<th></th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time/image</td>
<td>0.32s</td>
<td>47.00 s</td>
</tr>
<tr>
<td>-test speedup</td>
<td>146x</td>
<td>1x</td>
</tr>
<tr>
<td>Test time/image with proposal</td>
<td>2s</td>
<td>50 s</td>
</tr>
<tr>
<td>-test speedup</td>
<td>25x</td>
<td>1x</td>
</tr>
</tbody>
</table>

Source: cs231 standford
Direct region proposal evaluation

- VGG_CNN_M_1024
- Training takes < '2' hours
- Fast training makes these experiments possible

Drawback of R-CNN and the modification:
1. Training is a multi-stage pipeline. -> End-to-end joint training.
2. Training is expensive in space and time. -> Convolutional layer sharing. Classification in memory.
For SVM and regressor training, features are extracted from each warped object proposal in each image and written to disk. (VGG16, 5k VOC07 trainval images: 2.5 GPU days). Hundreds of gigabytes of storage.
3. Test-time detection is slow. -> Single scale testing, SVD fc layer.
At test-time, features are extracted from each warped proposal in each img. (VGG16: 47s / image).

Contributions:
1. Higher detection quality (mAP) than R-CNN
2. Training is single-stage, using a multi-task loss
3. All network layers can be updated during training
4. No disk storage is required for feature caching
• **Mini-batch Sampling**
  128: 2 randomly sampled images with 64 PoI sampled from each image
  25% positive: IoU > 0.5
  75% background: IoU in [0.1, 0.5)
  horizontally flipped with prob = 0.5

**Truncated SVD for faster detection**

mAP ~ 0.3% down; speed ~ 30% up

number of RoI for detection is large -> time spent on fc

\[ W \sim U \Sigma_t V^T \] (U : u*t, Sigma_t: t*t, V: v*t)

Compression : \((Wx + b) \text{ fc} \rightarrow (\Sigma_t V^T x) \text{ fc} + (Ux + b) \text{ fc}\)
Faster RCNN

Don’t need to have external regional proposals

RPN - Regional Proposal Network

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun
Jun 2015

What is Faster R-CNN?

• Presented in Kaiming’s section

• Review:
  Faster R-CNN = Fast R-CNN + Region Proposal Networks
  • Does not depend on an external region proposal algorithm
  • Does object detection in a single forward pass
Training goal: Share features

Goal: "share" so CNN'A' == CNN'B
One'nnet','four'losses

1. CNN: 
   - Feature map

2. Region Proposal Network: 
   - Proposals

3. Classification loss
4. Bounding box regression loss

5. RoI pooling

...
**Faster RCNN**

2k scores

*cls layer*

4k coordinates

*reg layer*

256-d

intermediate layer

sliding window

conv feature map

k anchor boxes

• Translation-Invariant Anchors

At each sliding window loc, predict k proposal: 4k outputs for reg layer, 2k outputs for cls layer (binary softmax).

Anchor: centered at sliding window with scale and aspect ratio: \([128^2, 256^2, 512^2; 1:2, 2:1, 1:1]\)

For a conv feature map: \(W \times H \times k\) (k=9 anchors) \((2+4)\times9\) output layer

• Optimization

fcn trained by end-to-end by bp and sgd
image-centric sampling strategy, sample 256 anchors in an image(Pos:neg = 1:1)
new layer initialization \(\sim N(0, 0.01)\)
tune ZFnet and conv3_1 and up for VGGnet, lr=0.001 for 60k batches, 0.0001 for 20k on PASCAL
Loss function for Learning Region Proposal

positive label: the anchor has highest IoU with a gt-box or has an IoU > 0.7 with any gt-box

negative label: IoU < 0.3 for all gt-box

Objective function with multi-task loss: Similar to Fast R-CNN.

\[ L(p_i, t_i) = L_{cls}(p_i, p_i^*) + \lambda p_i^* L_{reg}(t_i, t_i^*) \]

where \( p_i^* \) is 1 if the anchor is labeled positive, and is 0 if the anchor is negative.

\( \lambda = 10 \) bias towards better box location
Share Convolutional Features for Region Proposal and Objection Detection

Four-step training algorithm:
1. Train RPN, initialized with ImageNet pre-trained model
2. Train a separate detection network by Fast R-CNN using proposals generated by step-1 RPN, initialized by ImageNet pre-trained model
3. Fix conv layer, fine-tune unique layers to RPN, initialized by detector network in Step2
4. Fix conv layer, fine-tune fc-layers of Fast R-CNN
## Result compare

<table>
<thead>
<tr>
<th></th>
<th>Faster R-CNN</th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time/image</td>
<td>0.2S</td>
<td>2s</td>
<td>50s</td>
</tr>
<tr>
<td>With proposal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-test speedup</td>
<td>250x</td>
<td>25x</td>
<td>1x</td>
</tr>
<tr>
<td>mAP</td>
<td>66.9</td>
<td>66.9</td>
<td>66</td>
</tr>
</tbody>
</table>

Trained using Pascal VOC 2007 dataset

Source: cs231 standford
Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. “Region-wise” includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

<table>
<thead>
<tr>
<th>model</th>
<th>system</th>
<th>conv</th>
<th>proposal</th>
<th>region-wise</th>
<th>total</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG</td>
<td>SS + Fast R-CNN</td>
<td>146</td>
<td>1510</td>
<td>174</td>
<td>1830</td>
<td>0.5 fps</td>
</tr>
<tr>
<td>VGG</td>
<td>RPN + Fast R-CNN</td>
<td>141</td>
<td>10</td>
<td>47</td>
<td>198</td>
<td>5 fps</td>
</tr>
<tr>
<td>ZF</td>
<td>RPN + Fast R-CNN</td>
<td>31</td>
<td>3</td>
<td>25</td>
<td>59</td>
<td>17 fps</td>
</tr>
</tbody>
</table>
Recap: Faster RCNN

https://pan.baidu.com/s/1pIKIIB
• Recognition = Description + Discrimination. We are doing well on Discrimination

• Challenges of Deformation and Occlusion. This is also known as hard negative mining and non-maximum suppression. The past solutions assume the features are hard coded. CNN solutions extract ‘deformable’ features. Occlusion is dealt with per pixel labels over many object classes.

• R-CNN, Fast R-CNN, Faster R-CNN: sparse sampling of ‘where’ jointly with ‘what’. Using one shared feature extraction front-end, and multiple regression objectives at the FC layers. Enough information from the front-end CNN features allows both detection (of location) and classification (of classes).

• Memory allocation and flow is important for CNN learning and speed up.