Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren  Kaiming He
Ross Girshick  Jian Sun

Present by:
Yixin Yang
Mingdong Wang

Object Detection
Applications

• Basic task for image understanding
  • Output only bounding boxes

• Enables many downstream applications
  • Almost everything requires detection!
  • Vehicle – traffic, accident etc.
  • Face – security, APPs
  • …
Applications

Applications

• [http://how-old.net/](http://how-old.net/) By Microsoft
Basic Idea: Detection as classification

• Select a region (by any means)
• Classify this region

• Example
  • Cat? No
  • Dog? No

Basic Idea: Detection as classification

• Select a region (by any means)
• Classify this region

• Example
  • Cat? No
  • Dog? Yes
Basic Idea: Detection as classification

• Select a region (by any means)
• Classify this region

• Example
  • Cat? No
  • Dog? No

Basic Idea: Detection as classification

• Problem: Too many proposals
  • Slow
• Solution: Only look at possible regions

• Region proposal
  • Find regions that are likely to have object in it
  • Class-invariant
Region Proposal

• Selective Search
  • Feature-based
  • Bottom-up
  • Merging


Region Proposal

• Many other solutions
• EdgeBoxes is the best in practice

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Outputs Segments</th>
<th>Outputs Score</th>
<th>Control #proposals</th>
<th>Time (sec)</th>
<th>Repeatability</th>
<th>Recall Results</th>
<th>Detection Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing [18]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.2</td>
<td>***</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>CPM [19]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>250</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EdgeBoxes [20]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.3</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Endres [21]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100</td>
<td>-</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Geodesic [22]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>***</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>HOG [23]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
<td>***</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Objectness [24]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>ReLu [25]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>RandomizedPrim’s [26]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>Rastalankila [27]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>-</td>
<td>**</td>
</tr>
<tr>
<td>Riger [28]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>-</td>
<td>**</td>
</tr>
<tr>
<td>SelectiveSearch [29]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SlidingWindow</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td>***</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Superpixels</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>1</td>
<td>*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Hosang et al, “What makes for effective detection proposals?”, PAMI 2015
Outline

• Previous work
  • Region-based Convolutional Neural Network (R-CNN)
  • Spatial Pyramid Pooling Network (SPP-Net)
  • Fast R-CNN
• Faster R-CNN
  • Region Proposal Network (RPN)
  • Detection
• Experiments

R-CNN

• Region Proposals + CNN

• Three Steps:
  • Use Selective Search to get region proposals (~2k)
  • Warp every region proposal to 227x227, then extract feature by CNN
  • Classify: Support Vector Machine (SVM)
R-CNN

Warp Image:
- The inputs of CNN should be the same size

Training:
- Pre-train CNN for image classification
- Fine-tune CNN for object detection
- Train linear predictor for object detection

What is wrong with R-CNN?
- Training and testing is slow
- Takes a lot of disk space

Reason
- a ConvNet forward pass for each object proposal

How to solve?
- 1 CNN for whole image -> Spatial Pyramid Pooling Net (Spp-net)
Spatial Pyramid Pooling Net

• Much similar with R-CNN, but only 1 CNN for the whole image

• In fact, it is the fully-connect layer that needs the fix-size input

Spatial Pyramid Pooling Net

• 1 CNN for the input image and get the feature map

• Add a SPP layer after the last convolutional layer
Spatial Pyramid Pooling Net

• SPP layer Example

• The input of the spp layer can be of arbitrary size

• The output size is 21x256 no matter the input size
Spatial Pyramid Pooling Net

- The improvement of SPP-net
  - Makes training and testing fast
- What is wrong with SPP-net?
  - Cannot update parameters below SPP layer during training
- How to solve?
  - Use Softmax classifier instead of SVM -> Fast R-CNN

Fast R-CNN

- There are two differences in fast R-CNN
  - Add a Region of Interest (RoI) pooling layer after the last convolutional layer
  - Two output vectors per RoI: softmax probabilities and bounding-box regression on offset. The architecture is trained end-to-end with a multi-task loss.
Fast R-CNN

• RoI pooling layer
  • Use max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of $H \times W$

Fast R-CNN

• RoI pooling layer (example)
  • Input:
    a single 8×8 feature map and a region proposal (arbitrary size)
  • Output:
    2x2 feature map for future use
Fast R-CNN

- RoI pooling layer
  - Pooling sections
  - $h/H \&\& w/W$, in this case: $5/2 \&\& 7/2$
  - Notice that the size of the region of interest doesn’t have to be perfectly divisible by the number of pooling sections

---

Fast R-CNN

- RoI pooling layer
  - Max pooling:
    Get the max values in each of the sections
Fast R-CNN

- Two output layers
  - discrete probability distribution (per RoI), \( p = (p_0, p_1, ..., p_K) \) over \( K+1 \) categories
  - bounding-box regression offsets, \( t^k = (t^k_x, t^k_y, t^k_w, t^k_h) \), for each of the \( K \) object classes

Fast R-CNN

- Multi-task loss

\[
L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v).
\]

\( u \) is the ground-truth class

\( v \) is ground-truth bounding-box regression target

Where \( L_{\text{cls}}(p, u) = -\log p_u \) is the log loss for true class \( u \)
Fast R-CNN

• Multi-task loss

\[ L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v). \]

Where

\[ L_{loc}(t^u, v) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t^u_i - v_i) \]

\[ \text{smooth}_{L_1}(x) = \begin{cases} 
0.5x^2 & \text{if } |x| < 1 \\
|x| - 0.5 & \text{otherwise}
\end{cases} \]

Fast R-CNN

• What is wrong with Fast R-CNN
  • Use Selective Search to get the regions proposal, which is time consuming.

• How to solve?
  • Use Convolutional Neural Network to generate region proposal -> Faster R-CNN
Faster R-CNN

• Object proposal is the bottleneck
  • Selective search ~ 2s
  • EdgeBoxes ~ 0.2s
  • As much as the detection network
• Feature map used by detector can also used for generating proposals
• Why not also use CNN?
  • Better performance
  • Sharing computation

Region Proposal Network (RPN)

• Input
  • Image (feature map) of any size
• Output
  • A set of rectangular object proposals
  • Each with an objectness score

Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.
RPN

- Insert the RPN after the last conv layer
- RPN will produce region proposal directly, serves as ‘attention’
- After RPN, use RoI pooling and bounding-box regressor just like fast R-CNN
RPN: Anchor

- For each anchor
- Propose k anchor boxes
  - Related to this anchor
  - Prior assumption
- For each box, regress
  - Objectness score
  - Coordinates

Loss function

- L_cls: classification error
- L_reg: bbox coords regression error
- p_i/p_i*: predicted/ground truth classification
- t_i/t_i*: predicted/ground truth bbox coords

\[
L({p_i}, {t_i}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)
\]
RPN training

• End to end trainable by SGD
  • Stochastic gradient descent
• Each mini-batch arises from a single image
• Randomly sample 256 anchors in the image
  • Try to make pos/neg ~ 1
  • Otherwise negative will dominate -> biased

RPN & Fast RCNN detector

• If trained separately, they will have different conv params
• Try to share conv layers
• How to write the update formula?
Joint training

- Exact derivatives are hard to get
  - Derivatives of RoI pooling layer w.r.t. box coords
- Alternating
  - Train RPN, then use proposals to train detector, then train RPN...
- Approximate
  - Ignore the derivatives of bbox coords as if they are fixed
- 4-step training
  - Train RPN on pretrained ImageNet network
  - Train a detector with RPN proposals but using different conv params
  - Use detector params to initialize RPN network, but fix shared layers, finetune RPN
  - Fix shared layers, finetune detector

Other details

- Anchor box area & ratio
  - Select without carefully tested
- Cross image boundaries handling
  - Ignore when training
  - Crop when testing
- RPN proposals overlapping
  - NMS to reduce proposals
Experiments

Table 2: Detection results on PASCAL VOC 2007 test set (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF but using various proposal methods for training and testing.

<table>
<thead>
<tr>
<th>train-time region proposals</th>
<th>test-time region proposals</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td># boxes</td>
<td>method</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>SS</td>
</tr>
<tr>
<td>EB</td>
<td>2000</td>
<td>EB</td>
</tr>
<tr>
<td>RPN+ZF, shared</td>
<td>2000</td>
<td>RPN+ZF, shared</td>
</tr>
</tbody>
</table>

ablation experiments follow below

| method | # boxes | method | # proposals |
|-----------------------------|----------------------------|---------|
| RPN+ZF, unshared | 2000 | RPN+ZF, unshared | 300 | 58.7 |
| SS  | 2000 | RPN+ZF | 100 | 55.1 |
| SS  | 2000 | RPN+ZF | 300 | 56.8 |
| SS  | 2000 | RPN+ZF | 1000 | 56.3 |
| SS  | 2000 | RPN+ZF (no NMS) | 6000 | 55.2 |
| SS  | 2000 | RPN+ZF (no cls) | 100 | 44.6 |
| SS  | 2000 | RPN+ZF (no cls) | 300 | 51.4 |
| SS  | 2000 | RPN+ZF (no cls) | 1000 | 55.8 |
| SS  | 2000 | RPN+ZF (no reg) | 300 | 52.1 |
| SS  | 2000 | RPN+ZF (no reg) | 1000 | 51.3 |
| SS  | 2000 | RPN+VGG | 300 | 59.2 |

Experiments

Table 3: Detection results on PASCAL VOC 2007 test set. The detector is Fast R-CNN and VGG-16. Training data: "07"; VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

| method | # proposals | data | mAP (%) |
|-----------------------------|----------------------------|---------|
| SS  | 2000 | 07 | 66.9† |
| SS  | 2000 | 07+12 | 70.0 |
| RPN+VGG, unshared | 300 | 07 | 68.5 |
| RPN+VGG, unshared | 300 | 07+12 | 73.2 |
| RPN+VGG, shared | 300 | COCO+07+12 | 75.8 |


| method | # proposals | data | mAP (%) |
|-----------------------------|----------------------------|---------|
| SS  | 2000 | 07 | 65.7 |
| SS  | 2000 | 07++ | 68.4 |
| RPN+VGG, shared† | 300 | 07 | 67.0 |
| RPN+VGG, shared‡ | 300 | 07+12 | 70.4 |
| RPN+VGG, shared‡ | 300 | COCO+07++12 | 75.9 |
Experiments

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</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</td>
</tr>
<tr>
<td>ZF</td>
<td>RPN + Fast R-CNN</td>
<td>31</td>
<td>3</td>
<td>25</td>
<td>50</td>
<td>17</td>
</tr>
</tbody>
</table>

Figure 4: Recall vs. IoU overlap ratio on the PASCAL VOC 2007 test set.
Follow Up

- YOLO (You Only Look Once: Unified, Real-Time Object)
  - Instead of using region proposal + classification, doing the regression of the position and class of bounding box
  - Convert the objection detection to a Regression problem
- SDD (Single Shot MultiBox Detector)
- YOLO2
- ......

Summary

- Find a variable number of objects by classifying image regions
- R-CNN
  - Selective Search + CNN + SVM
  - ~30s / img
- Fast RCNN
  - Swap order of convolutions and region extraction
  - 2 s / img
- Faster RCNN
  - Compute region proposals within the network
  - 0.2 s / img
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Thanks for listening

Q && A