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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Presented by
Tushar Bansal
Objective

1. Get bounding box for all objects (of trained classes) in an image
2. Classify bounding boxes with labels
3. Train a network fast enough for real-time object detection
Overview

1. Background
2. R-CNN
   a. Architecture
   b. Selective Search
3. Fast R-CNN
   a. Architecture
   b. ROI Pooling
4. Faster R-CNN
   a. Region Proposal Network
5. Experiments/Performance Comparison
6. Conclusion
Background

R-CNN: Region-based Convolutional Neural Networks (2013)

- Selective Search for region proposals
- CNN extracts feature map to classify regions using SVM

Fast R-CNN (2015)

- Share computation of convolutional layers for proposals (ROI pooling)

Faster R-CNN (2015)

- Introduces RPN with shared network weights to reduce proposal time
R-CNN

- Works on classification model for Imagenet (Alexnet/VGG-16)
- Model is trained using positive/negative regions from detection images
- Features from pool5 layer are saved in memory
R-CNN

- SVM over image features for classification
- bbox regression: Linear regression model to map from cached features to new detection window
Selective Search

Bottom-up segmentation, merging regions at multiple scales

J. R. Uijlings, et. al, “Selective search for object recognition”
R-CNN Limitations

- Not end-to-end training: CNN layers are not trained over feedback from regressor and SVM
- Each region proposal requires a full pass of convolutional network
- Large memory usage
- Slow detection speed-not suitable for real-time usage
Fast R-CNN

- End to end system training for detection
- Share computation of convolutional layers between proposals for an image - thanks to ROI Pooling layers
- Proposals still from selective search
ROI Pooling

Problem: Region proposal is $C \times H \times W$ but the fully connected layers expects $C \times h \times w$ conv features.
ROI Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Hi-res conv features: C x H x W with region proposal

RoI conv features: C x h x w for region proposal

Fully-connected layers expect low-res conv features: C x h x w
Faster R-CNN

- Insert a Region Proposal Network (RPN) after last conv layer
- Proposals from RPN are fed to ROI pooling layer followed by classifier and bbox regressor similar to Fast R-CNN
- Authors explore two models- Zeiler and Fergus (5 shareable conv layers) and VGG-16 (13 shareable conv layers)
Faster R-CNN: Region Proposal Network

- Slide a small network, with nxn window (n=3 here) over the convolutional feature map output by last shared convolutional layer.
- Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG).
- This feature is fed into two sibling fully-connected layers—a box regression layer (reg) and a box-class layer (cls).
Faster R-CNN: Region Proposal Network
Anchors

Problem with region proposal using a convolutional network?
Anchors

Problem with region proposal using a convolutional network?

Multiple scales of objects possible.

Possible solutions:

1. Multiple scaled images
2. Multiple filter sizes (usually adopted jointly with 1)

but these approaches are time consuming.
Anchors

A more cost-efficient approach built on pyramid of anchors

- Method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios
- Requires convolutional features computed on a single-scale image
- Key-component for sharing features without extra-cost for addressing scales
- Approach is translation invariant
Anchors

For bounding box regression, the parameterizations of 4 coordinates is done as:

\[ t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \]
\[ t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \]
\[ t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \]
\[ t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a), \]

where \( x, y, w, \) and \( h \) denote box’s center coordinates and its width and height

Variables \( x, x_a, x^* \) are for predicted box, anchor box and ground truth box
Loss

For training RPNs, binary class labels are assigned to anchors.

- **Positive:**
  - Anchor that has an IOU overlap higher than 0.7 with any ground truth box
  - Anchors with highest Intersection-over-Union (IOU). *Why do we need this?*

- **Negative:**
  - Non-positive anchor if its IOU ratio is lower than 0.3 for all ground truth boxes

Anchors neither positive nor negative do not contribute to the training objective
Loss (RPN)

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

Classification loss, \( L_{cls} \) is log loss over two classes (object vs not object)
Regression loss, \( L_{reg} \) is robust \( L_1 \) loss

\( i \): index of anchor in mini-batch
\( p_i \): predicted probability of anchor being an object
\( p_i^* \): ground truth (1 if anchor +ve, 0 if -ve)
\( N_{cls}, N_{reg} \) = normalization terms, \( \lambda \) = balancing parameter
Loss (Fast R-CNN)

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

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

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

\(u, v\): ground truth class and bounding box
Training

- Randomly sample 256 anchors in an image to compute the loss function, with positive and negative anchors having ratio up to 1:1
- Shared layers are initialized by pre-trained ImageNet classification
- New layers initialized by zero mean gaussian with 0.01 standard dev
Training

Algorithms for training RPN and Fast R-CNN together:

1. Alternative Training:
   a. First train RPN
   b. Use proposals to train Fast R-CNN
   c. Use this network to initialize RPN and keep iterating.

2. Approximate Joint Training:
   a. In forward pass, proposals are treated like fixed, pre-computed proposals
   b. In backward pass, signals from RPN and Fast R-CNN is combined for shared layers
   c. Ignores derivative w.r.t. proposal boxes coordinates, so is approximate

3. Non-approximate Joint Training
   a. Includes derivative w.r.t. proposal boxes coordinates
Training

4 Step Alternating Training:

1. Train RPN initialized with ImageNet pre-trained network
2. Train separate detection network by Fast R-CNN using proposals generated by RPN trained in step 1
3. Use the detector network weights to fix shared convolutional layers and fine tune only layers unique to RPN
4. Now keeping all other layers fixed, fine tune unique layers of Fast R-CNN
Experiments

- **PASCAL VOC 2007**
  - 5k trainval images and 5k test images
  - 20 object categories

- **MS COCO**
  - 80k images in training set, 40k in validation set and 20k images on test-dev set
  - 80 object categories

All images are re-scaled such that their shorter side is 600 pixels.

Why are flexible image sizes not a problem for the network?
Experiments

- **PASCAL VOC 2007**
  - 5k trainval images and 5k test images
  - 20 object categories

- **MS COCO**
  - 80k images in training set, 40k in validation set and 20k images on test-dev set
  - 80 object categories

All images are re-scaled such that their shorter side is 600 pixels.

*Why are flexible image sizes not a problem for the network?*

Because of conv layers and ROI pooling before any fully connected layers.
Experiments

Anchors with 3 scales- 128, 256 and 512 pixels and 3 aspect ratios- 1:1, 1:2 and 2:1 were used

Since some RPN proposals highly overlap with each other, Non-maximum suppression (NMS) with IOU threshold 0.7 is used to reduce redundancy.

mAP: mean average precision (mAP@0.5 for VOC, mAP@[0.5,0.95] for COCO)

<table>
<thead>
<tr>
<th>train-time region proposals</th>
<th>test-time region proposals</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>method</td>
</tr>
<tr>
<td>SS</td>
<td>SS</td>
</tr>
<tr>
<td>EB</td>
<td>EB</td>
</tr>
<tr>
<td>RPN+ZF, shared</td>
<td>RPN+ZF, shared</td>
</tr>
</tbody>
</table>
Experiments: Ablation

1. Removing NMS
2. Removing cls layer: Randomly sample regions
3. Removing the reg layer: Proposals become anchor boxes

<table>
<thead>
<tr>
<th>train-time region proposals</th>
<th>test-time region proposals</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>method</strong></td>
<td><strong># boxes</strong></td>
<td><strong>method</strong></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:

<table>
<thead>
<tr>
<th>RPN+ZF, unshared</th>
<th>RPN+ZF, unshared</th>
<th>300</th>
<th>58.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF</td>
<td>100</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF</td>
<td>300</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF</td>
<td>1000</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no NMS)</td>
<td>6000</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no cls)</td>
<td>100</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no cls)</td>
<td>300</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no cls)</td>
<td>1000</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no reg)</td>
<td>300</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+ZF (no reg)</td>
<td>1000</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>RPN+VGG</td>
<td>300</td>
</tr>
</tbody>
</table>
### Experiments: Training Data

<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2000</td>
<td>07</td>
<td>66.9†</td>
</tr>
<tr>
<td>SS</td>
<td>2000</td>
<td>07+12</td>
<td>70.0</td>
</tr>
<tr>
<td>RPN+VGG, unshared</td>
<td>300</td>
<td>07</td>
<td>68.5</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07</td>
<td>69.9</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>73.2</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>COCO+07+12</td>
<td>78.8</td>
</tr>
</tbody>
</table>
Experiments: Timing

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>
Experiments: Anchors

<table>
<thead>
<tr>
<th>settings</th>
<th>anchor scales</th>
<th>aspect ratios</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 scale, 1 ratio</td>
<td>$128^2$</td>
<td>1:1</td>
<td>65.8</td>
</tr>
<tr>
<td></td>
<td>$256^2$</td>
<td>1:1</td>
<td>66.7</td>
</tr>
<tr>
<td>1 scale, 3 ratios</td>
<td>$128^2$</td>
<td>{2:1, 1:1, 1:2}</td>
<td>68.8</td>
</tr>
<tr>
<td></td>
<td>$256^2$</td>
<td>{2:1, 1:1, 1:2}</td>
<td>67.9</td>
</tr>
<tr>
<td>3 scales, 1 ratio</td>
<td>{128$^2$, 256$^2$, 512$^2$}</td>
<td>1:1</td>
<td>69.8</td>
</tr>
<tr>
<td>3 scales, 3 ratios</td>
<td>{128$^2$, 256$^2$, 512$^2$}</td>
<td>{2:1, 1:1, 1:2}</td>
<td>69.9</td>
</tr>
</tbody>
</table>
## Experiments: One vs Two Stages

<table>
<thead>
<tr>
<th></th>
<th>proposals</th>
<th>detector</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Stage</td>
<td>RPN + ZF, unshared</td>
<td>Fast R-CNN + ZF, 1 scale</td>
<td>58.7</td>
</tr>
<tr>
<td>One-Stage</td>
<td>dense, 3 scales, 3 aspect ratios</td>
<td>Fast R-CNN + ZF, 1 scale</td>
<td>53.8</td>
</tr>
<tr>
<td>One-Stage</td>
<td>dense, 3 scales, 3 aspect ratios</td>
<td>Fast R-CNN + ZF, 5 scales</td>
<td>53.9</td>
</tr>
</tbody>
</table>
### Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image (with proposals)</td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
<td>250x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Results
Conclusion

- Shared weights improve the speed significantly
- Scale (thanks to anchors) and Translational invariant
- Region Proposal time is insignificant compared to detection
- Both networks could be trained together rather than the alternating approach
Extensions

- Two stage process could be made a single network - already done by some networks after Faster R-CNN
- Network could be generalized to other tasks like pose estimation (Mask R-CNN)

SVCL Lab at UCSD have developed MS-CNN, which has shown better latency and performance* compared to Faster R-CNN.
References

- Stanford slides - Fei Fei Li & Andrej Karpathy & Justin Johnson