2D Object Detection

Renjie Liao

largely reuse materials in slides and papers of Ross Girshick, Kaiming He, et. al.
2D Object Detection

Localization
Where?

Recognition
What?

car: 1.000
human: 0.992
horse: 0.993
dog: 0.967

Figure adapted from Kaiming He
Object detection renaissance (2013-present)

mean Average Precision (mAP)

Before deep convnets

Using deep convnets

PASCAL VOC
Object detection renaissance (2013-present)

![Graph showing the improvement in mean Average Precision (mAP) over years with a significant increase after the use of deep convnets.](image_url)
Object detection renaissance (2013-present)

PASCAL VOC

Mean Average Precision (mAP)

Year

Fast R-CNN
- Accurate
- Fast
- Streamlined

R-CNNv1
- Accurate
- Slow
- Inelegant
R-CNN
R-CNN: Regions with CNN features

- Input image
- Extract region proposals (~2k / image)
- Compute CNN features
- Classify regions (linear SVM)
R-CNN at test time: **Step 1**

Proposal-method agnostic, many choices
- Selective Search [van de Sande, Uijlings et al.] **(Used in this work)**
- Objectness [Alexe et al.]
- Category independent object proposals [Endres & Hoiem]
- CPMC [Carreira & Sminchisescu]

Active area, at this CVPR
- BING [Ming et al.] – *fast*
- MCG [Arbelaez et al.] – *high-quality segmentation*
R-CNN at test time: **Step 2**

Input image → Extract region proposals (~2k / image) → Compute CNN features

- aeroplane? no.
- person? yes.
- tvmonitor? no.
R-CNN at test time: **Step 2**

Input image → Extract region proposals (~2k / image) → Compute CNN features → Dilate proposal

- aeroplane? no.
- person? yes.
- tvmonitor? no.
R-CNN at test time: **Step 2**

Input image

Extract region proposals (~2k / image)

Compute CNN features

- a. Crop
- aeroplane? no.
- person? yes.
- tvmonitor? no.
**R-CNN at test time: Step 2**

**Input image**

**Extract region proposals (~2k / image)**

**Compute CNN features**

- **a. Crop**
- **b. Scale (anisotropic)**

- **aeroplane? no.**
- **person? yes.**
- **tvmonitor? no.**

**Image dimensions:** 227 x 227
R-CNN at test time: Step 2

1. Crop
2. Scale (anisotropic)
3. Forward propagate

Output: “fc7” features
R-CNN at test time: **Step 3**

Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions

- aeroplane? no.
- tvmonitor? no.

- person? yes.

4096-dimensional fc7 feature vector

linear classifiers (SVM or softmax)

- person? 1.6
- horse? -0.3
Step 4: Object proposal refinement

Original proposal  \[\text{Linear regression on CNN features}\]  Predicted object bounding box

Bounding-box regression
Slow R-CNN

Girshick et al. CVPR14.
Slow R-CNN

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

Input image

Girshick et al. CVPR14.
Slow R-CNN

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

Warped image regions

Input image

Girshick et al. CVPR14.
Slow R-CNN

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

Warped image regions

Forward each region through ConvNet

Input image

Girshick et al. CVPR14.
**Slow R-CNN**

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

Warped image regions

Forward each region through ConvNet

Classify regions with SVMs

Input image

Post hoc component

Girshick et al. CVPR14.
Slow R-CNN

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

Warped image regions

Forward each region through ConvNet

Classify regions with SVMs

Apply bounding-box regressors

Input image

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

Post hoc component

Girshick et al. CVPR14.
Training R-CNN

Bounding-box labeled detection data is scarce

Key insight:
Use supervised pre-training on a data-rich auxiliary task and transfer to detection
Supervised pre-training
Train a SuperVision CNN* for the 1000-way ILSVRC image classification task

*Network from Krizhevsky, Sutskever & Hinton. NIPS 2012
Also called “AlexNet”
R-CNN training: Step 2

Fine-tune the CNN for detection
Transfer the representation learned for ILSVRC classification to PASCAL (or ImageNet detection)

Target task:
PASCAL VOC detection
(~25k object labels)
R-CNN training: Step 3

Train detection SVMs
(With the softmax classifier from fine-tuning mAP decreases from 54% to 51%)
Fast R-CNN
What’s wrong with slow R-CNN?
What’s wrong with slow R-CNN?

• **Ad hoc training objectives**
  • Fine-tune network with softmax classifier (log loss)
  • Train post-hoc linear SVMs (hinge loss)
  • Train post-hoc bounding-box regressors (squared loss)
What’s wrong with slow R-CNN?

• Ad hoc training objectives
  • Fine-tune network with softmax classifier (log loss)
  • Train post-hoc linear SVMs (hinge loss)
  • Train post-hoc bounding-box regressors (squared loss)

• Training is slow (84h), takes a lot of disk space
What’s wrong with slow R-CNN?

• Ad hoc training objectives
  • Fine-tune network with softmax classifier (log loss)
  • Train post-hoc linear SVMs (hinge loss)
  • Train post-hoc bounding-box regressions (least squares)
• Training is slow (84h), takes a lot of disk space
• Inference (detection) is slow
  • 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  • Fixed by SPP-net [He et al. ECCV14]

~2000 ConvNet forward passes per image
SPP-net

He et al. ECCV14.
SPP-net

He et al. ECCV14.
SPP-net

Regions of Interest (RoIs) from a proposal method

Forward whole image through ConvNet

“conv5” feature map of image

He et al. ECCV14.
SPP-net

Regions of Interest (RoIs) from a proposal method

Forward whole image through ConvNet

Spatial Pyramid Pooling (SPP) layer

“conv5” feature map of image

He et al. ECCV14.
SPP-net

Regions of Interest (RoIs) from a proposal method

Forward whole image through ConvNet

“conv5” feature map of image

Spatial Pyramid Pooling (SPP) layer

Fully-connected layers

Classify regions with SVMs

Post hoc component

He et al. ECCV14.
**SPP-net**

- Forward *whole* image through ConvNet
- “conv5” feature map of image
- Spatial Pyramid Pooling (SPP) layer
- Fully-connected layers
- Classify regions with SVMs
- Apply bounding-box regressors
- Bbox reg
- SVMs
- FCs

Regions of Interest (RoIs) from a proposal method

He et al. ECCV14.
What’s good about SPP-net?

• Fixes one issue with R-CNN: makes testing fast
What’s wrong with SPP-net?

- Inherits the rest of R-CNN’s problems
  - Ad hoc training objectives
  - Training is slow (25h), takes a lot of disk space
What’s wrong with SPP-net?

• Inherits the rest of R-CNN’s problems
  • Ad hoc training objectives
  • Training is slow (though faster), takes a lot of disk space

• Introduces a new problem: cannot update parameters below SPP layer during training
SPP-net: the main limitation

He et al. ECCV14.
Fast R-CNN

• Fast test-time, like SPP-net
Fast R-CNN

• Fast test-time, like SPP-net
• One network, trained in one stage
Fast R-CNN

• Fast test-time, like SPP-net
• One network, trained in one stage
• Higher mean average precision than slow R-CNN and SPP-net
Fast R-CNN (test time)

Regions of Interest (RoIs) from a proposal method

“conv5” feature map of image

Forward whole image through ConvNet

Input image

ConvNet
Fast R-CNN (test time)

Regions of Interest (RoIs) from a proposal method

“RoI Pooling” (single-level SPP) layer

“conv5” feature map of image

Forward whole image through ConvNet

Input image

ConvNet
Fast R-CNN (test time)

1. Forward whole image through ConvNet
2. "conv5" feature map of image
3. "RoI Pooling" (single-level SPP) layer
4. Fully-connected layers
5. Linear + softmax
6. Softmax classifier
7. Regions of Interest (RoIs) from a proposal method

ConvNet
Fast R-CNN (test time)

Regions of Interest (RoIs) from a proposal method

Softmax classifier

Linear + softmax

Linear

Bounding-box regressors

Fully-connected layers

“RoI Pooling” (single-level SPP) layer

“conv5” feature map of image

Forward whole image through ConvNet

ConvNet

Input image
Fast R-CNN (training)

ConvNet

Linear + softmax

Linear

FCs
Fast R-CNN (training)

Log loss + smooth L1 loss

Linear + softmax

Linear

FCs

ConvNet

Multi-task loss
Fast R-CNN (training)

- Log loss + smooth L1 loss
- Linear + softmax
- Linear
- FCs
- ConvNet

Multi-task loss

Trainable
Obstacle #1: Differentiable RoI pooling

\[
\frac{\partial L}{\partial x_i} = \sum_r \sum_j 1_{i = i^*(r, j)} \frac{\partial L}{\partial y_{rj}}
\]

Partial for \( x_i \)

Over regions \( r \), locations \( j \)

Partial from next layer

Max pooling "switch" (i.e., argmax back-pointer)
Faster R-CNN
Faster R-CNN
Region Proposal Network
Training Scheme

• Alternating Training:
  1. Train RPN
  2. Fixed Proposals, train Fast R-CNN
  3. Fixed Shared CNN, train RPN with Fast R-CNN fixed
  4. Fixed Shared CNN, fine-tune Fast R-CNN

• Approximate Joint Training

• Non-approximate Joint Training
Residual Network
Deep Residual Learning

• Plaint net

\[ x \]

\[ \text{weight layer} \]

\[ \text{relu} \]

\[ \text{weight layer} \]

\[ \text{relu} \]

\[ H(x) \]

\( H(x) \) is any desired mapping, hope the 2 weight layers fit \( H(x) \)

Deep Residual Learning

- Residual net

\[ H(x) \] is any desired mapping,

hope the 2 weight layers fit \( H(x) \)

hope the 2 weight layers fit \( F(x) \)

let \( H(x) = F(x) + x \)
Deep Residual Learning

- $F(x)$ is a **residual** mapping w.r.t. identity

$$H(x) = F(x) + x$$

- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations
Network “Design”

• Keep it simple

• Our basic design (VGG-style)
  • all 3x3 conv (almost)
  • spatial size /2 => # filters x2
  • Simple design; just deep!

• Other remarks:
  • no max pooling (almost)
  • no hidden fc
  • no dropout

Object Detection (brief)

• Simply “Faster R-CNN + ResNet”

<table>
<thead>
<tr>
<th>Faster R-CNN baseline</th>
<th>mAP@.5</th>
<th>mAP@.5:.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>41.5</td>
<td>21.5</td>
</tr>
<tr>
<td>ResNet-101</td>
<td><strong>48.4</strong></td>
<td><strong>27.2</strong></td>
</tr>
</tbody>
</table>

COCO detection results
(ResNet has 28% relative gain)

Object Detection (brief)

- RPN learns proposals by extremely deep nets
  - We use only 300 proposals (no SS/EB/MCG!)

- Add what is just missing in Faster R-CNN...
  - Iterative localization
  - Context modeling
  - Multi-scale testing

- All are based on CNN features; all are end-to-end (train and/or inference)

- All benefit more from deeper features – cumulative gains!

Our results on COCO – too many objects, let’s check carefully!


*the original image is from the COCO dataset*

*the original image is from the COCO dataset*
Thanks!