CSE 252C: Advanced Computer Vision

Manmohan Chandraker

Lecture 15: Object Detection
Object Detection

[Ren et al., Faster R-CNN]
Object Detection

Object 1: \((x_1, y_1, w_1, h_1)\), dog
Object 2: \((x_2, y_2, w_2, h_2)\), cat
Object Detection

Object 1: \((x_1, y_1, w_1, h_1)\), dog
Object 2: \((x_2, y_2, w_2, h_2)\), cat
Object 3: \((x_3, y_3, w_3, h_3)\), dog
Object Detection

Need to handle outputs of variable lengths

CSE 252C, SP20: Manmohan Chandraker
Object Detection: Sliding Windows

Given new image:
1. Slide window
2. Score by classifier

Car or non-car Classifier

Feature extraction

Training examples

CSE 252C, SP20: Manmohan Chandraker
Object Detection: Sliding Windows

### HoG + SVM

- **Input Image**
- **Gradient Image**
- **Orientation Voting**
- **Overlapping Blocks**
- **Local Normalization**

**Dalal and Triggs, CVPR 2005**

### DPM

- **Image pyramid**
- **HOG feature pyramid**
- **Part filters**
- **Root filter**

**Score is sum of filter scores minus deformation costs**

*Multiscale model captures features at two-resolutions*

*Parts are represented at twice the resolution of the root filter.*

Sliding Windows

- Need to consider windows at several different positions and scales
- Either use a simple feature extractor, or evaluate only few candidates

Dog? YES
Cat? NO
Background? NO

Dog? NO
Cat? NO
Background? YES
Region Proposals

- Find blob-like regions of image that might be objects
- Do not consider object type and tolerate high rate of false positives
Step 1: Train (or download) a classification model for ImageNet (AlexNet)

---

[Girshick et al., Rich feature hierarchies]
**Step 2:** Fine-tune model for detection
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive and negative regions from detection images
Step 3: Extract features
- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!
**R-CNN Training**

**Step 4:** Train one binary SVM per class to classify region features

- **Training image regions**
  - Positive samples for cat SVM
  - Negative samples for cat SVM

- **Cached region features**

[Slide from: Tamara Berg]  
[Girshick et al., Rich feature hierarchies]
R-CNN Training

**Step 4:** Train one binary SVM per class to classify region features

Training image regions

Cached region features

Negative samples for dog SVM

Positive samples for dog SVM

[Girshick et al., Rich feature hierarchies]
**R-CNN Training**

**Step 5** (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals.

- **Training image regions**
- **Cached region features**
- **Regression targets**
  - $(dx, dy, dw, dh)$
  - Normalized coordinates
- **Proposal is good**: $(0, 0, 0, 0)$
- **Proposal too far to left**: $(.25, 0, 0, 0)$
- **Proposal too wide**: $(0, 0, -0.125, 0)$

[Girshick et al., Rich feature hierarchies]
R-CNN Regression

**Step 5** (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals.

![Diagram of R-CNN Regression]

Regression targets:

\[
\begin{align*}
  t_x &= (g_x - p_x)/p_w \\
  t_y &= (g_y - p_y)/p_h \\
  t_w &= \log(g_w/p_w) \\
  t_h &= \log(g_h/p_h)
\end{align*}
\]

Train a regressor to find offset function \( d_i \) that reaches closest to regression target:

\[
\mathcal{L}_{\text{reg}} = \sum_{i \in \{x, y, w, h\}} (t_i - d_i(p))^2 + \lambda \|w\|^2
\]
R-CNN Improvements

• Significant gain over pre-CNN methods
• Bounding box regression helps
• Deeper feature extractor leads to large gain
Issues with R-CNN

• Very expensive for inference
  • Full forward pass of CNN for each region proposal

• Classification and regression are disjoint from feature extraction
  • CNN features not updated together with detection

• Complex training pipeline

• Need mechanism to “connect” CNN features to classifier and regressor
Fast R-CNN

- Extract convolutional features just once for whole image
- Need method to share computation of convolutional layers between proposals for an image
- Allows several advantages
  - Faster inference
  - Connects CNN to classifier
  - Easier training
- Proposals still from selective search

Ghirshick, Fast R-CNN
Fast R-CNN: 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

Problem: Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: 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

Project region proposal onto conv feature map

Convolution and Pooling

Fully-connected layers

Problem: Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: RoI Pooling

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

Convolution and Pooling

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

Divide projected region into h x w grid

Fully-connected layers

Problem: Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: 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

**Problem:** Fully-connected layers expect fixed size low-res features: C x h x w

**RoI Pooling**

\[ y_{rj} = x_{i^*(r,j)} \]  
(RoI r, output j)

\[ i^*(r,j) = \arg\max_{i \in \mathcal{R}(r,j)} x_i \]

Pooling region
**Fast R-CNN: 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

**Problem:** Fully-connected layers expect fixed size low-res features: C x h x w

**RoI Pooling**

\[
y_{rj} = x_{i^*(r,j)}
\]

\[
i^*(r,j) = \arg \max_{i' \in \mathcal{R}(r,j)} x_{i'}
\]

Pooling region

**Backpropagation**

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

Gradient accumulated if \( i \) is the index chosen for max pooling.
Fast R-CNN: Improvements over R-CNN

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time:</td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Faster R-CNN

- Proposals still from separate mechanism in Fast R-CNN
- Insert a Region Proposal Network (RPN) after last convolutional layer
- RPN trained to produce region proposals directly, no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and regressor just like Fast R-CNN

[Ren et al., Faster R-CNN]
Faster R-CNN: Region Proposal Network

- Slide a small window on the feature map

- Build a small network for:
  - classifying object or not-object
  - regressing bounding box locations

- Position of sliding window gives location information with respect to the image

- Box regression gives finer localization with respect to this sliding window
Faster R-CNN: Anchors

- Problem with region proposal using a convolutional network
  - Multiple scales of objects possible

- Possible solutions
  - Multiple scaled images
  - Filters of different sizes (usually together with above)

- But such approaches are time-consuming.
Faster R-CNN: Anchors

• Consider single object localization
  • Simply have a classification and regression head

• But now consider multiple objects, possibly overlapping
  • Regression for objects will interfere with each other

• Anchors: a set of reference positions on the feature map
  • Anchor box with high ground truth overlap responsible for regressing position
  • Determines reference and spatial extent for predicting object

Typically: 3 scales and 3 aspect ratios
Faster R-CNN: Anchors

• A cost-efficient way to achieve multi-scale outputs
  • Relies on image and feature maps at single scale
  • Feature computation is shared across anchor boxes at different scales

• Translation invariance
  • Use same convolutional RPN, anchor boxes for spatial localization
  • A translated object will lead to accordingly shifted proposals

Typically: 3 scales and 3 aspect ratios
Training RPN

• Place anchors uniformly across image, n boxes at every position (typically n = 9)
  • For 40 x 60 feature map, about 21k anchor boxes

• RPN output: 4n regression (x, y, w, h) and 2n classification (object, background)

• Non-maximum suppression to pick about 2k boxes
  • Remove boxes that overlap with others of higher score

• Labels for RPN classification: compute IoU of anchor boxes with ground truth
  • Assign IoU > 0.7 as object and IoU < 0.3 as background

• Bounding box regression
  • Displacement target: distance between centers of ground truth and anchor boxes
  • Size target: log ratio of anchor and ground truth dimensions

• Sample 256 anchors to form mini-batch for training

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

Cross-entropy | Bounding box label | Smooth L1 | Regression target
Training Overall Detector

- Train CNN + RPN
  - Initialize with ImageNet pretrained weights
  - Train end-to-end for region proposal task

- Train CNN + Detector
  - Initialize with ImageNet pretrained weights
  - Use fixed proposals from above RPN
  - Train Fast R-CNN for detection task

- Fine-tune RPN
  - Use CNN from above step
  - Fine-tune RPN for proposal task

- Fine-tune detector
  - Keep CNN fixed
  - Fine-tune Fast R-CNN layers for detection task

- Subsequently, a joint training is also available with all four losses
  - RPN: classification (object or background), regression (anchor to proposal)
  - Fast R-CNN: classification (object category), regression (proposal to bounding box)
## Faster R-CNN Improvements

<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>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Faster R-CNN Improvements

ImageNet Detection (mAP)

- NekNet ensemble (2015): 53.57
- Faster R-CNN single (2015): 42.94
- GoogleNet ensemble (2014): 43.93
- NUS ensemble (2014): 37.21
- SPP ensemble (2014): 35.11
- UVA-Euvision (2013): 22.58
- Overfeat (2013): 19.4
Object Detection

Huang et al. Speed/Accuracy Tradeoffs for Modern Convolutional Object Detectors. CVPR 2017