CSE 252C: Advanced Computer Vision
Manmohan Chandraker

Lecture 17: Object Detection
Object Detection

Need to handle outputs of variable lengths
• Need to consider windows at several different positions and scales
• Either use a simple feature extractor, or evaluate only few candidates
Region Proposals

- Find blob-like regions of image that might be objects
- Do not consider object type and tolerate high rate of false positives

Bottom-up segmentation, merging regions at multiple scales

[Uijlings et al., Selective search for object recognition]
R-CNN

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

- Expensive
  - No feature reuse
  - 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

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

- **RoI pooling**: connects convolutional layers to classifier
  - Features and classifier trained together
- Feature extraction just once for the whole image
  - Faster inference and training
- Proposals still from external mechanism

[Girshick, Fast R-CNN]
## 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 and shapes 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 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 anchors at different scales

- Translation invariance
  - Use same convolutional RPN, anchors 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, \( n \) at every position (typically \( n = 9 \))
  - For 40 x 60 feature map, about 21k anchor positions

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

- Labels for RPN classification: compute IoU of bounding 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)

mAP

62.07, 58.85, 53.57, 42.94, 43.93, 37.21, 35.11, 22.58, 19.4

Single-Shot Detector

- Two step process of Faster R-CNN can be expensive
  - Box classification and regression are done twice
  - RPN produces boxes, but they are used to pool features
  - Separate classifier then used for evaluation

- Key motivation for SSD: achieve good accuracy-speed trade-off
  - Use something similar to RPN for directly scoring anchors
  - Extract anchors at multiple scales to get accuracy with minimal overhead
Single-Shot Detector

- Standard VGG (or ResNet) base network

- Add further layers with progressively decreasing size
  - Used for predicting detections at multiple scales
  - Layers with wider receptive fields expected to detect larger objects

[Liu et al., SSD]
SSD: Default boxes (anchors)

- SSD uses default boxes, which are similar to anchors in Faster R-CNN.
- Default boxes located at each cell on the feature map.
  - Multiple boxes corresponding to different scales and aspect ratios.

Each position has anchor boxes of different aspect ratio.

\[
\begin{align*}
\text{Feature Map} & \\
\frac{S_k}{\sqrt{a_r}} \times S_k \sqrt{a_r} & \\
S_k \times S_k & \\
S_k \sqrt{a_r} \times \frac{S_k}{\sqrt{a_r}} & \\
\end{align*}
\]

\( a_r \) : Aspect ratio
\( a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\} \)
SSD: Default boxes (anchors)

- Anchor boxes of different feature maps have different scales
  - Various feature maps responsible for detecting objects of different sizes
- Let $m$ be the number of feature maps used for prediction
- Let $s_k$ denote the scale of feature map $k$
- Define $s_{\text{min}} = 0.2$ and $s_{\text{max}} = 0.9$

$$s_k = s_{\text{min}} + \frac{s_{\text{max}} - s_{\text{min}}}{m - 1}(k - 1), \quad k \in [1, m]$$
SSD: Convolution prediction at multiple scales

- On each feature map, two types of convolutional filters applied
  - c filters for class prediction, where c is the number of object categories
  - 4 filters for bounding box regression, for coordinates x, y, w, h
- With k default boxes at every cell, there are \((c+4)k\) filters for each feature map
- Output for \(m \times n\) feature map is a \(m \times n \times (c+4)k\) map
  - Represents category label and regression offsets for each default box
SSD: Training

- Ground truth positives: match ground truth to each default box with IoU > 0.5
- Ground truth negatives: all other default boxes
**SSD: Training**

- **Ground truth positives:** match ground truth to each default box with IoU > 0.5
- **Ground truth negatives:** all other default boxes
- **Loss:** Combination of classification confidence and localization accuracy
- **Classification loss** is softmax for multi-class prediction

\[
L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}
\]

\[x_{ij}^p = 1 \text{ if default box } i \text{ matches ground truth box } j \text{ of class } p, \text{ otherwise } 0\]

- **Smooth L1 loss for localization**

\[
L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)
\]

Offsets  Predicted location  Ground truth for box j
SSD: Training

- Ground truth positives: match ground truth to each default box with IoU > 0.5
- Ground truth negatives: all other default boxes
- Loss: Combination of classification confidence and localization accuracy
- Classification loss is softmax for multi-class prediction
  \[
  L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}
  \]
  \[x_{ij}^p = 1 \text{ if default box } i \text{ matches ground truth box } j \text{ of class } p, \text{ otherwise } 0\]
- Smooth L1 loss for localization
  \[
  L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)
  \]
  Offsets Predicted location Ground truth for box \(j\)
- Number of negatives far exceeds positives
  - Hard negative mining: choose default boxes with highest classification loss
SSD: Analysis

- Helps to use multiple layers for prediction

<table>
<thead>
<tr>
<th>Prediction source layers from:</th>
<th>mAP</th>
<th>use boundary boxes?</th>
<th># Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv4.3 conv7 conv8.2 conv9.2 conv10.2 conv11.2</td>
<td>74.3</td>
<td>Yes</td>
<td>8732</td>
</tr>
<tr>
<td></td>
<td><strong>74.6</strong></td>
<td>No</td>
<td>8764</td>
</tr>
<tr>
<td></td>
<td>73.8</td>
<td>Yes</td>
<td>8942</td>
</tr>
<tr>
<td></td>
<td>70.7</td>
<td>No</td>
<td>9864</td>
</tr>
<tr>
<td></td>
<td>64.2</td>
<td>Yes</td>
<td>9025</td>
</tr>
<tr>
<td></td>
<td>62.4</td>
<td>No</td>
<td>8664</td>
</tr>
</tbody>
</table>

- In general, most errors in SSD are due to poorer localization
  - Explicit pooling of region proposals in Faster R-CNN prevents this
  - Possibly also cause for more confusion between similar categories in SSD

- SSD also tends to do better for larger objects compared to smaller ones
Object detection: Accuracy and speed

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