Presented by: Roi Shikler & Gil Elbaz
Advisor: Prof. Michael Lindenbaum

Topics of the lecture:

- Problem statement
- Review of slow R-CNN
- Review of Fast R-CNN
- Review of Faster R-CNN
- Compare with other methods
- Take away

PASCAL Visual Object Classes

"The main goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes" [from the challenge homepage]

The twenty object classes that have been selected are:

- **Person**: person
- **Animal**: bird, cat, cow, dog, horse, sheep
- **Vehicle**: airplane, bicycle, boat, bus, car, motorbike, train
- **Indoor**: bottle, chair, dining table, potted plant, sofa, TV/monitor
mAP: Mean Average Precision.

Mean average precision for a set of queries is the mean of the average precision scores for each query.

\[ mAP = \frac{\sum P_\in Q \text{AvgP}(\phi)}{|Q|} \]

Plateau & Increasing complexity
Topics of the lecture:

- Problem statement
- Review of slow R-CNN
  - Review of Fast R-CNN
  - Review of Faster R-CNN
  - Compare with other methods
  - Take away

R-CNN - Article (2013)

Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN:

Input image
Extract region proposals (~2k/image)
Compute CNN features
Classify regions (linear SVM)

PASCAL VOC detection history

R-CNN at test time: Step 1
RCNN at test time: Step 1

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

Selective Search [van de Sande, Uijlings et al.] (Agnostic proposal method)

RCNN at test time: Step 2

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

Dilate Proposal

RCNN at test time: Step 2

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

a. Crop

RCNN at test time: Step 2

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

a. Crop

b. Scale (anisotropic)

RCNN at test time: Step 2

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

b. Scale

c. Forward propagate

Output: “fc7” features

RCNN at test time: Step 2

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

b. Scale

c. Forward propagate

Output: “fc7” features

Five connected convolutional layers, and two fully connected layers.

Input: 227*227*3 = 150528

Output: 4096-dimensional feature vector
**RCNN at test time:** Step 2

- Input image
- Extract region proposals (~2k / image)
- Compute CNN features
- b. Scale
- Output: "fc7" features

**RCNN at test time:** Step 3

- Input image
- Extract region proposals (~2k / image)
- Compute CNN features
- b. Scale
- 4096-dimensional fc7 feature vector
- Linear classifiers (SVM)

**Step 4: Object proposal refinement**

- Course Detection (using HOG + BOVW)
- Feature Extraction from Color and Edges
- Relocalization + Resizing with Graph-Cuts
- Linear regression on CNN features

**Step 4: Object proposal refinement**

- Original proposal
- Predicted object bounding box

**In RCNN from Selective Search**

**Course Detection**

- Using HOG + BOVW
- (~20-30% overlap)

**Feature Extraction from Color and Edges**

- Color
- Edge

**Relocalization + Resizing with Graph-Cuts**

- [fine tuning]
Steps for training a slow R-CNN detector:

1. [offline] $M \leftarrow \text{Pre-train a ConvNet for ImageNet classification}$
2. $M' \leftarrow \text{Fine-tune } M \text{ for object detection (softmax classifier)}$
3. $F \leftarrow \text{Cache feature vectors to disc using } M'$
4. Train post hoc linear SVMs on $F$ for object classification
5. Train post hoc linear bounding box regressors on $F$

* “post hoc” means the parameters are learned after the ConvNet is fixed
Review slow 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)

- Training is slow, takes a lot of disk space
  - The training process takes about 84h

- Detection (inference) is slow
  - About 47s per image
  - ~2k ConvNet forward passes per image

Topics of the lecture:

- Problem statement
- Review of slow R-CNN
- Review of Fast R-CNN
- Review of Faster R-CNN
- Compare with other methods
- Take away

PASCAL VOC detection history
Fast R-CNN objectives
Fix most of what’s wrong with slow R-CNN and SPP-net
• Train the detector in a single stage, end-to-end
  • No caching features to disk
  • No post hoc training steps
• Train all layers of the network
  [Training the conv layers is important for very deep networks]
Review of the fast R-CNN training pipeline

Benefits of end-to-end training
- Faster training
  - No reading/writing features from/to disk
  - No training post hoc SVMs and bounding-box regressors
- Verified empirically: optimizing a single multi-task objective is more accurate than optimizing objectives independently

Slow R-CNN vs. Fast R-CNN
- Training time: 84 hours / 8.75 hours
- VOC07 test mAP: 66.0% / 68.1%
- Listing time per image: 47s / 0.32s
  - With selective search: 49s / 2.05s (+2x per image)
Topics of the lecture:

- Problem statement
- Review of slow R-CNN
- Review of Fast R-CNN
- Review of Faster R-CNN

- Compare with other methods
- Take away

**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 the sliding window provides localization information with reference to the image.

Box regression provides finer localization information with reference to this sliding window.

**Review of the faster R-CNN**

- In Fast R-CNN: Single loss
  - Region Proposal Network
  - Classifying: object or not
  - Regressing: bounding-boxes

- Single loss
  - Anchor boxes:
  - Translation invariant
  - Regression: gives offsets from anchor boxes
  - Classification gives the probability that each regressed box shows an object
Topics of the lecture:

- Problem statement
- Review of slow R-CNN
- Review of Fast R-CNN
- Review of Faster R-CNN
- Compare with other methods
  - Take away

---

**R-CNN Results**

- Big improvement using CNN
- Features from deeper network helps a lot

---

**R-CNN Results**

- Results by using pre-CNN methods

---

**R-CNN Results**

- Bounding box regression shows a 5% improvement

---

**Fast R-CNN Results**

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time</td>
<td>54 hrs</td>
<td>9.0 hrs</td>
</tr>
<tr>
<td>Speedup</td>
<td>1x</td>
<td>8.5x</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 secs</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>Speedup</td>
<td>1x</td>
<td>148x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
<tr>
<td>Test time per image with Selective Search</td>
<td>50 seconds</td>
<td>2 seconds</td>
</tr>
<tr>
<td>Speedup</td>
<td>1x</td>
<td>26x</td>
</tr>
</tbody>
</table>
Faster R-CNN Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time per image (with proposals)</td>
<td>90 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>95.0</td>
<td>66.5</td>
<td>66.0</td>
</tr>
</tbody>
</table>

Faster R-CNN Results

YOLO – You Only Look Once
Detection as Regression

Faster than Faster R-CNN, but not as good

Topics of the lecture:

- Problem statement
- Review of slow R-CNN
- Review of Fast R-CNN
- Review of Faster R-CNN
- Compare with other methods

Object Detection code links:

- R-CNN (Caffe + MATLAB): [GitHub](https://github.com/rbgirshick/rcnn)
  [Probably don’t use this; too slow]
- Fast R-CNN (Caffe + MATLAB): [GitHub](https://github.com/rbgirshick/fast_rcnn)
- Faster R-CNN (Caffe + MATLAB): [GitHub](https://github.com/ShaoqingRen/fasterrcnn)
- Faster R-CNN (Caffe + Python): [GitHub](https://github.com/rbgirshick/faster_rcnn)
- YOLO: [GitHub](https://github.com/y0lo)

Take Away

- Classification and detection of objects in images is approaching high quality and real-time frame rate results.
- The major breakthroughs came from:
  1. Utilizing CNN instead of classical computer vision methods
  2. Making deeper neural networks with multiple objectives
  3. Switching all stages of algorithm with one unified network (end-to-end)
- These algorithms are all available and applicable to real-world problems! Once you know how they work, you can change and adjust them to your specific research needs.
Any Questions?

THANK YOU