Lecture 8:
Spatial Localization and Detection
Administrative

- Project Proposals were due on Saturday
- Homework 2 due Friday 2/5
- Homework 1 grades out this week
- Midterm will be in-class on Wednesday 2/10
Convolution

Summary: To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Pooling

224x224x64 \rightarrow \text{pool} \rightarrow 112x112x64

2x2 max pooling

\begin{array}{cccc}
1 & 1 & 2 & 4 \\
5 & 6 & 7 & 8 \\
3 & 2 & 1 & 0 \\
1 & 2 & 3 & 4 \\
\end{array}
Case Studies

LeNet
(1998)

AlexNet
(2012)

ZFNet
(2013)
Case Studies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv layers</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-128</td>
</tr>
<tr>
<td></td>
<td>conv3-64</td>
<td>conv3-128</td>
<td>conv3-256</td>
</tr>
<tr>
<td></td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-512</td>
</tr>
<tr>
<td></td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td></td>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC-4096</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC-4096</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC-1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Revolution of Depth

ImageNet Classification top-5 error (%)

- ILSVRC'15 ResNet: 3.57%
- ILSVRC'14 GoogleNet: 6.7%
- ILSVRC'14 VGG: 7.3%
- ILSVRC'13: 11.7%
- ILSVRC'12 AlexNet: 16.4%
- ILSVRC'11: 25.8%
- ILSVRC'10: 28.2%

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Localization and Detection

Results from Faster R-CNN, Ren et al 2015
Computer Vision Tasks

- **Classification**
  - Single object: CAT

- **Classification + Localization**
  - Single object: CAT

- **Object Detection**
  - Multiple objects: CAT, DOG, DUCK

- **Instance Segmentation**
  - Multiple objects: CAT, DOG, DUCK
Computer Vision Tasks

Classification + Localization

Object Detection

Instance Segmentation
Classification + Localization: Task

**Classification**: C classes
- **Input**: Image
- **Output**: Class label
- **Evaluation metric**: Accuracy

**Localization**:
- **Input**: Image
- **Output**: Box in the image (x, y, w, h)
- **Evaluation metric**: Intersection over Union

**Classification + Localization**: Do both
Classification + Localization: ImageNet

1000 classes (same as classification)

Each image has 1 class, at least one bounding box

~800 training images per class

Algorithm produces 5 (class, box) guesses

Example is correct if at least one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)

Krizhevsky et al. 2012
Idea #1: Localization as Regression

**Input:** image

**Output:**
Box coordinates (4 numbers)

**Correct output:**
box coordinates (4 numbers)

**Loss:**
L2 distance

Only one object, simpler than detection
Simple Recipe for Classification + Localization

**Step 1:** Train (or download) a classification model (AlexNet, VGG, GoogLeNet)
Simple Recipe for Classification + Localization

**Step 2**: Attach new fully-connected “regression head” to the network
Simple Recipe for Classification + Localization

**Step 3**: Train the regression head only with SGD and L2 loss
Simple Recipe for Classification + Localization

**Step 4**: At test time use both heads
Per-class vs class agnostic regression

Assume classification over C classes:

Classification head:
C numbers (one per class)

Class agnostic:
4 numbers (one box)

Class specific:
C x 4 numbers (one box per class)
Where to attach the regression head?

- **After conv layers:**
  - Overfeat, VGG

- **After last FC layer:**
  - DeepPose, R-CNN

**Diagram:**
- Image → Convolution and Pooling → Final conv feature map → Fully-connected layers → Class scores → Softmax loss
Aside: Localizing multiple objects

Want to localize exactly $K$ objects in each image

(e.g. whole cat, cat head, cat left ear, cat right ear for $K=4$)
Aside: Human Pose Estimation

Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

(Details: Normalized coordinates, iterative refinement)

Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014
Localization as Regression

Very simple

Think if you can use this for projects
Idea #2: Sliding Window

- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction
Sliding Window: Overfeat

Winner of ILSVRC 2013 localization challenge

Image: 3 x 221 x 221

Convolution + pooling

Feature map: 1024 x 5 x 5

4096

Class scores: 1000

Softmax loss

Euclidean loss

4096

1024

Boxes: 1000 x 4

Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014
Sliding Window: Overfeat

Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257
Sliding Window: Overfeat

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores: P(cat)

0.5
Sliding Window: Overfeat

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores: P(cat)

0.5  0.75
Sliding Window: Overfeat

Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

Classification scores:
P(cat) = 0.5, 0.75, 0.6
Sliding Window: Overfeat

Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

Classification scores:
P(cat)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>0.6</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Sliding Window: Overfeat

Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

Classification scores:
P(cat)

<table>
<thead>
<tr>
<th>0.5</th>
<th>0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Sliding Window: Overfeat

Greedily merge boxes and scores (details in paper)

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification score: P (cat)
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales.

Window positions + score maps
Box regression outputs
Final Predictions

Efficient Sliding Window: Overfeat

Image: 3 x 221 x 221

Convolution + pooling

Feature map: 1024 x 5 x 5

Boxes: 1000 x 4

Class scores: 1000

4096 FC 4096 FC 1024 FC Boxes: 1000 x 4

FC
Efficient Sliding Window: Overfeat

Efficient sliding window by converting fully-connected layers into convolutions.

Image: 3 x 221 x 221

Convolution + pooling

Feature map: 1024 x 5 x 5

4096 x 1 x 1 → 1024 x 1 x 1

5 x 5 conv

5 x 5 conv

1 x 1 conv

1 x 1 conv

Class scores: 1000 x 1 x 1

Box coordinates: (4 x 1000) x 1 x 1

4096 x 1 x 1 → 1024 x 1 x 1
Efficient Sliding Window: Overfeat

**Training time:** Small image, 1 x 1 classifier output

**Test time:** Larger image, 2 x 2 classifier output, only extra compute at yellow regions

Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014
ImageNet Classification + Localization

- **AlexNet**: Localization method not published
- **Overfeat**: Multiscale convolutional regression with box merging
- **VGG**: Same as Overfeat, but fewer scales and locations; simpler method, gains all due to deeper features
- **ResNet**: Different localization method (RPN) and much deeper features
Computer Vision Tasks

- Classification
- Classification + Localization
- Object Detection
- Instance Segmentation
Computer Vision Tasks

- Classification
- Classification + Localization
- Object Detection
- Instance Segmentation
Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers
Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers
Detection as Regression?

CAT, (x, y, w, h) → ... → CAT (x, y, w, h) = many numbers

Need variable sized outputs
Detection as Classification

CAT? NO
DOG? NO
Detection as Classification

CAT? YES!

DOG? NO
Detection as Classification

CAT? NO
DOG? NO
Detection as Classification

**Problem:** Need to test many positions and scales

**Solution:** If your classifier is fast enough, just do it
Histogram of Oriented Gradients

- Compute HOG of the whole image at multiple resolutions
- Score every subwindow of the feature pyramid
- Apply non-maxima suppression

Dalal and Triggs, “Histograms of Oriented Gradients for Human Detection”, CVPR 2005
Slide credit: Ross Girshick
Deformable Parts Model (DPM)

Aside: Deformable Parts Models are CNNs?

Girshick et al, “Deformable Part Models are Convolutional Neural Networks”, CVPR 2015
Detection as Classification

**Problem:** Need to test many positions and scales, and use a computationally demanding classifier (CNN)

**Solution:** Only look at a tiny subset of possible positions
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions
Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes

Region Proposals: Many other choices

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Outputs Segments</th>
<th>Outputs Score</th>
<th>Control #proposals</th>
<th>Time (sec.)</th>
<th>Repeatability</th>
<th>Recall Results</th>
<th>Detection Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing [18]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.2</td>
<td>***</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CPMC [19]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>250</td>
<td>-</td>
<td>**</td>
<td>***</td>
</tr>
<tr>
<td>EdgeBoxes [20]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.3</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Endres [21]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100</td>
<td>-</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Geodesic [22]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>MCG [23]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
<td>*</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Objectness [24]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Raitu [25]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>RandomizedPrim’s [26]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Rantalankila [27]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Rigor [28]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>*</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>SelectiveSearch [29]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SlidingWindow</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Superpixels</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>1</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hosang et al, “What makes for effective detection proposals?”, PAMI 2015
Region Proposals: Many other choices

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Outputs Segments</th>
<th>Outputs Score</th>
<th>Control #proposals</th>
<th>Time (sec.)</th>
<th>Repeatability</th>
<th>Recall Results</th>
<th>Detection Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing [18]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.2</td>
<td>***</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CPMC [19]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>250</td>
<td>-</td>
<td>**</td>
<td>*</td>
</tr>
<tr>
<td><strong>EdgeBoxes [20]</strong></td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.3</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Endres [21]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100</td>
<td>-</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Geodesic [22]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>MCG [23]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
<td>*</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Objectness [24]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Rahtu [25]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>RandomizedPrim’s [26]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Rantalankila [27]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Rigor [28]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>*</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>SelectiveSearch [29]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SlidingWindow</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>***</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Superpixels</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Hosang et al, “What makes for effective detection proposals?”, PAMI 2015
Putting it together: R-CNN


Slide credit: Ross Girshick
R-CNN Training

**Step 1**: Train (or download) a classification model for ImageNet (AlexNet)
R-CNN Training

**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 / negative regions from detection images
R-CNN Training

**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

Cached region features

Positive samples for cat SVM

Negative samples for cat SVM
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
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

(0, 0, 0, 0) Proposal is good
(.25, 0, 0, 0) Proposal too far to left
(0, 0, -0.125, 0) Proposal too wide
## Object Detection: Datasets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of classes</td>
<td>20</td>
<td>200</td>
<td>80</td>
</tr>
<tr>
<td>Number of images (train + val)</td>
<td>~20k</td>
<td>~470k</td>
<td>~120k</td>
</tr>
<tr>
<td>Mean objects per image</td>
<td>2.4</td>
<td>1.1</td>
<td>7.2</td>
</tr>
</tbody>
</table>
Object Detection: Evaluation

We use a metric called “mean average precision” (mAP)

Compute average precision (AP) separately for each class, then average over classes

A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)

Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

TL;DR mAP is a number from 0 to 100; high is good
R-CNN Results

R-CNN Results

Big improvement compared to pre-CNN methods

Mean Average Precision (mAP)

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM (2011)</td>
<td>33.7</td>
<td>29.6</td>
</tr>
<tr>
<td>Regionlets (2013)</td>
<td>41.7</td>
<td>39.7</td>
</tr>
<tr>
<td>R-CNN (2014, AlexNet)</td>
<td>54.2</td>
<td>50.2</td>
</tr>
<tr>
<td>R-CNN + bbox reg (AlexNet)</td>
<td>58.5</td>
<td>53.7</td>
</tr>
<tr>
<td>R-CNN (VGG-16)</td>
<td>66.0</td>
<td>62.9</td>
</tr>
</tbody>
</table>
R-CNN Results

Bounding box regression helps a bit

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM (2011)</td>
<td>33.7</td>
<td>29.6</td>
</tr>
<tr>
<td>Regionlets (2013)</td>
<td>41.7</td>
<td>39.7</td>
</tr>
<tr>
<td>R-CNN (2014, AlexNet)</td>
<td>54.2</td>
<td>50.2</td>
</tr>
<tr>
<td>R-CNN + bbox reg (AlexNet)</td>
<td>58.5</td>
<td>53.7</td>
</tr>
<tr>
<td>R-CNN (VGG-16)</td>
<td>66</td>
<td>62.9</td>
</tr>
</tbody>
</table>
R-CNN Results

Features from a deeper network help a lot

Mean Average Precision (mAP)

- DPM (2011): 33.7
- Regionlets (2013): 41.7
- R-CNN (2014, AlexNet): 54.2
- R-CNN + bbox reg (AlexNet): 58.5
- R-CNN (VGG-16): 66.0

VOC 2007

VOC 2010
R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal

2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline
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

Slide credit: Ross Girschick
R-CNN Problem #1:
Slow at test-time due to independent forward passes of the CNN

Solution:
Share computation of convolutional layers between proposals for an image
Fast R-CNN (training)

R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

R-CNN Problem #3:
Complex training pipeline

Solution:
Just train the whole system end-to-end all at once!

Slide credit: Ross Girshick
Fast R-CNN: Region of Interest 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

Problem: Fully-connected layers expect low-res conv features: C x h x w

Fully-connected layers
**Fast R-CNN: Region of Interest 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 low-res conv features: C x h x w

Convolution and Pooling

Project region proposal onto conv feature map

Fully-connected layers
Fast R-CNN: Region of Interest 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 low-res conv features: C x h x w
Fast R-CNN: Region of Interest 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

Max-pool within each grid cell

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

Fully-connected layers expect low-res conv features: C x h x w
Fast R-CNN: Region of Interest 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

Convolution and Pooling

Can back propagate similar to max pooling

Fully-connected layers
# 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>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
## Fast R-CNN Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training Time:</strong></td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
<tr>
<td><strong>Test time per image</strong></td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
# Fast R-CNN Results

<table>
<thead>
<tr>
<th>Faster!</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>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>Better!</td>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Fast R-CNN Problem:

Test-time speeds don’t include region proposals

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</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>25x</td>
</tr>
</tbody>
</table>
Fast R-CNN Problem Solution:

Test-time speeds don’t include region proposals
Just make the CNN do region proposals too!

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</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>25x</td>
</tr>
</tbody>
</table>
Faster R-CNN:

Insert a **Region Proposal Network (RPN)** after the 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 bbox regressor just like Fast R-CNN.


Slide credit: Ross Girschick
Faster R-CNN: Region Proposal Network

Slide a small window on the feature map

Build a small network for:
- classifying object or not-object, and
- regressing bbox 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

Slide credit: Kaiming He
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object
Faster R-CNN: Training

In the paper: Ugly pipeline
- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!
One network, four losses
- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)
## 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>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>
Object Detection State-of-the-art: ResNet 101 + Faster R-CNN + some extras

<table>
<thead>
<tr>
<th>training data</th>
<th>COCO train</th>
<th>COCO trainval</th>
<th>test data</th>
<th>COCO val</th>
<th>COCO test-dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>@.5</td>
<td>@ [.5, .95]</td>
<td>@.5</td>
<td>@ [.5, .95]</td>
<td></td>
</tr>
<tr>
<td>baseline Faster R-CNN (VGG-16)</td>
<td>41.5</td>
<td>21.2</td>
<td>baseline Faster R-CNN (ResNet-101)</td>
<td>48.4</td>
<td>27.2</td>
</tr>
<tr>
<td>+ box refinement</td>
<td>49.9</td>
<td>29.9</td>
<td>+ context</td>
<td>51.1</td>
<td>30.0</td>
</tr>
<tr>
<td>+ multi-scale testing</td>
<td>53.8</td>
<td>32.5</td>
<td>ensemble</td>
<td>59.0</td>
<td>37.4</td>
</tr>
</tbody>
</table>

ImageNet Detection 2013 - 2015

![ImageNet Detection (mAP)](chart.png)

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet ensemble (2015)</td>
<td>62.07</td>
</tr>
<tr>
<td>ResNet single (2015)</td>
<td>58.85</td>
</tr>
<tr>
<td>NeoxNet ensemble (2015)</td>
<td>53.57</td>
</tr>
<tr>
<td>Faster R-CNN single (2015)</td>
<td>42.94</td>
</tr>
<tr>
<td>GoogLeNet ensemble (2015)</td>
<td>43.93</td>
</tr>
<tr>
<td>NUS ensemble (2014)</td>
<td>37.21</td>
</tr>
<tr>
<td>SPP ensemble (2014)</td>
<td>35.11</td>
</tr>
<tr>
<td>UvA-Euvisio (2013)</td>
<td>22.58</td>
</tr>
<tr>
<td>Overfeat (2013)</td>
<td>19.4</td>
</tr>
</tbody>
</table>
YOLO: You Only Look Once
Detection as Regression

Divide image into $S \times S$ grid

Within each grid cell predict:
- B Boxes: 4 coordinates + confidence
- Class scores: $C$ numbers

Regression from image to $7 \times 7 \times (5 \times B + C)$ tensor

Direct prediction using a CNN

YOLO: You Only Look Once
Detection as Regression

Faster than Faster R-CNN, but not as good

Object Detection code links:

**R-CNN**
(Caffe + MATLAB): [https://github.com/rbgirshick/rcnn](https://github.com/rbgirshick/rcnn)
Probably don’t use this; too slow

**Fast R-CNN**
(Caffe + MATLAB): [https://github.com/rbgirshick/fast-rcnn](https://github.com/rbgirshick/fast-rcnn)

**Faster R-CNN**
(Caffe + MATLAB): [https://github.com/ShaoqingRen/faster_rcnn](https://github.com/ShaoqingRen/faster_rcnn)
(Caffe + Python): [https://github.com/rbgirshick/py-faster-rcnn](https://github.com/rbgirshick/py-faster-rcnn)

**YOLO**
Maybe try this for projects?
Recap

Localization:
- Find a fixed number of objects (one or many)
- L2 regression from CNN features to box coordinates
- Much simpler than detection; consider it for your projects!
- Overfeat: Regression + efficient sliding window with FC -> conv conversion
- Deeper networks do better

Object Detection:
- Find a variable number of objects by classifying image regions
- Before CNNs: dense multiscale sliding window (HoG, DPM)
- Avoid dense sliding window with region proposals
- R-CNN: Selective Search + CNN classification / regression
- Fast R-CNN: Swap order of convolutions and region extraction
- Faster R-CNN: Compute region proposals within the network
- Deeper networks do better