Automatic Object Detection

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html
Advanced Approaches for Object Detection

• Object Detection

• The RCNN Object Detector (2014)

• The Fast RCNN Object Detector (2015)

• The Faster RCNN Object Detector (2016)
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation

Single object

Multiple objects

CAT

CAT

CAT, DOG, DUCK

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 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

Neural Net 

**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)

1. **Convolution and Pooling**
2. **Final conv feature map**
3. **Fully-connected layers**
4. **Class scores**
5. **Softmax loss**

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Simple Recipe for Classification + Localization

**Step 2:** Attach new fully-connected “regression head” to the network

![Diagram of image processing steps]

- **Image**
- **Convolution and Pooling**
- **Final conv feature map**
- **“Classification head”**
  - Fully-connected layers
  - Class scores
- **“Regression head”**
  - Fully-connected layers
  - Box coordinates
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)
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 $K=4$)
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
Object Detection: *sliding windows following by classification*
Object Detection as Classification of sliding windows (proposals)
Object Detection as Classification of sliding windows (proposals)
Object Detection as Classification of sliding windows (proposals)

CNN

deer?
cat?
background?
Object Detection as Classification of sliding windows (proposals)

CNN →

deer?
cat?
background?
Object Detection as Classification with Box Proposals
Sliding Window: Overfeat

Winner of ILSVRC 2013 localization challenge

Image: 3 x 221 x 221

Convolution + pooling

Feature map: 1024 x 5 x 5

Class scores: 1000

Softmax loss

Euclidean loss


Boxes: 1000 x 4
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)
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>
</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></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

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

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)

0.8
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

4096 → FC

4096 → FC

Class scores: 1000

1024 → FC

Boxes: 1000 x 4
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

5 x 5 conv

4096 x 1 x 1

1 x 1 conv

5 x 5 conv

1024 x 1 x 1

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

1 x 1 conv

1024 x 1 x 1

1 x 1 conv
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

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

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

![Diagram](https://via.placeholder.com/150)

- Image
- Convolution and Pooling
- Final conv feature map
- Fully-connected layers
- Class scores: 21 classes
- Softmax loss

Re-initialize this layer: was 4096 x 1000, now will be 4096 x 21
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
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.
## 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 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
Rich feature hierarchies for accurate object detection and semantic segmentation.
Girshick et al. CVPR 2014.
RCNN

First stage: generate category-independent region proposals.

- 2000 Region proposals for every image

Selective Search: combine the strength of both an exhaustive search and segmentation. Uijlings et al. IJCV 2013.

ref
**RCNN**

First stage: generate category-independent region proposals.
- 2000 Region proposals for every image

Second stage: extracts a fixed-length feature vector from each region.
- A 4096-dimensional feature vector from each region proposal

Arbitrary rectangles?  
A fixed size input? 227 x 227  
5 conv layers + 2 fully connected layers
First stage: generate category-independent region proposals.
- 2000 Region proposals for every image

Second stage: extracts a fixed-length feature vector from each region.
- A 4096-dimensional feature vector from each region proposal

Third stage: a set of class-specific linear SVMs.
- Object category and location
RCNN

• Simple and scalable.
• Improves mAP.

• A multistage pipeline.
• Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
• Object detection is slow.

Fast-RCNN

?
Fast-RCNN

https://arxiv.org/abs/1504.08083
Fast R-CNN. Girshick. ICCV 2015.

Idea: No need to recompute features for every box independently
Fast-RCNN

Process the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map.

A region of interest (RoI) pooling layer extracts a fixed-length feature vector from the region feature map.

K + 1 categories

Four real-valued numbers for each of the K object classes.
Fast R-CNN (test time)

Regions of Interest (RoIs) from a proposal method

ConvNet

Input image

Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
Fully-connected layers
“Rol Pooling” (single-level SPP) layer
“conv5” feature map of image

Forward whole image through ConvNet

Girsichick, “Fast R-CNN”, ICCV 2015

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

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

Convolutions 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

Convolution and Pooling
Project region proposal onto conv feature map

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

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

Convolution and Pooling

Hi-res conv features: \( C \times H \times W \) with region proposal

Max-pool within each grid cell

RoI conv features: \( C \times h \times w \) for region proposal

Fully-connected layers expect low-res conv features: \( C \times h \times w \)
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

Can back propagate similar to max pooling

Fully-connected layers expect low-res conv features: C x h x w
# 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></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>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</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>
**RCNN**

- Simple and scalable.
- Improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.

**Fast-RCNN**

- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.
- Proposals are the computational bottleneck in detection systems.

**Faster-RCNN**
Faster-RCNN

Idea: Integrate the Bounding Box Proposals as part of the CNN predictions

https://arxiv.org/abs/1506.01497
Ren et al. NIPS 2015.
Faster-RCNN

Region Proposal Networks:

- **2k scores**: object or not object
- **4k coordinates**: bounding box proposal

- **1x1 conv layer**: cls layer, reg layer
- **n xn conv layer**: feature map

- **k anchors boxes**: sliding window, n xn

Shared conv layers

RPN

Object is a cat

Classification loss

Bounding-box regression loss

proposals

Region Proposal Network

feature map

Rol pooling

Last conv layer

Fast-RCNN
RCNN

- Simple and scalable.
- Improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.

Fast-RCNN

- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.
- Proposals are the computational bottleneck in detection systems.

Faster-RCNN

- Compute proposals with a deep convolutional neural network -- Region Proposal Network (RPN)
- Merge RPN and Fast R-CNN into a single network, enabling nearly cost-free region proposals.
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 Girshick
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</td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>(with proposals)</td>
<td></td>
<td></td>
<td></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>