Two-stage object detectors
Types of object detectors

- One-stage detectors
  - Image
  - Feature extraction
  - Classification
  - Localization
  - Class score (cat, dog, person)
  - Bounding box ($x, y, w, h$)

- Two-stage detectors
  - Image
  - Feature extraction
  - Extraction of object proposals
  - Classification
  - Localization
  - Refine bounding box ($\Delta x, \Delta y, \Delta w, \Delta h$)
Types of object detectors

- **One-stage detectors**

- **Two-stage detectors**
Localization

- Bounding box regression

Image

Feature extraction (this time with a Neural Network)

Output:
Box coordinates (x,y,w,h)

Ground truth: Box coordinates

L2 loss function
Localization

- Bounding box regression
Localization and classification

• Bounding box regression

Image → Convolutional Neural Network → Fully connected → Output: Box coordinates (x, y, w, h)
Localization and classification

- Bounding box regression

Image → Convolutional Neural Network → Fully connected
Output: Box coordinates (x, y, w, h)

L2 loss
Softmax loss

Output: Class scores
Localization and classification

- Bounding box regression

Image → Convolutional Neural Network → Regression head
- Output: Box coordinates (x, y, w, h)

Convolutional Neural Network

Classification head
- Output: Class scores

Output:
- Box coordinates (x, y, w, h)
- Class scores

CV3DST | Prof. Leal-Taixé
Localization and classification

- It was typical to train the classification head first, freeze the layers
- Then train the regression head
- At test time, we use both!

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

- Sliding window + box regression + classification

Image
(221 x 221 x 3)

Convolutional Neural Network

Feature map
(5 x 5 x 1024)

Boxes
(1000 x 4)

Class scores
1000

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

- Sliding window + box regression + classification

Image (468 x 356 x 3)

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

- Sliding window + box regression + classification

Image (468 x 356 x 3)

Overfeat

- Sliding window + box regression + classification

Image (468 x 356 x 3)

Overfeat

- **Sliding window** + **box regression** + **classification**

Image (468 x 356 x 3)

Overfeat

• Sliding window + box regression + classification

We end up with many predictions and we have to combine them for a final detection (in Overfeat they have a greedy method)

Image (468 x 356 x 3)

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

- Sliding window + box regression + classification

We end up with many predictions and we have to combine them for a final detection (in Overfeat they have a greedy method)

Image (468 x 356 x 3)

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

- In practice: use many sliding window locations and multiple scales

Window positions + score maps

Box regression outputs

Final Predictions

Overfeat

- Sliding window + box regression + classification

What prevents us from dealing with any image size?

What about multiple objects?

- Localization: Regression
- How about detection?
What about multiple objects?

- Localization: Regression
- How about detection?

3 objects means having an output of 12 numbers (3 x 4)
What about multiple objects?

- Localization: Regression
- How about detection?

14 objects means having an output of 56 numbers (14 x 4)
What about multiple objects?

• Localization: Regression
• How about detection?

• Having a variable sized output is not optimal for Neural Networks

• There are a couple of workarounds:
Detection as classification?

- Localization: Regression
- How about detection? Regression

Is this a Flamingo? NO
Detection as classification?

- Localization: Regression
- How about detection? Regression

Is this a Flamingo? NO
Detection as classification?

- Localization: Regression
- How about detection? Regression

Is this a Flamingo? YES!
Detection as classification?

• Localization: Regression

• How about detection? Classification

• Problem:
  – Expensive to try all possible positions, scales and aspect ratios
  – How about trying only on a subset of boxes with most potential?
Region Proposals

- We have already seen a method that gives us “interesting” regions in an image that potentially contain an object

- Step 1: Obtain region proposals
- Step 2: Classify them.
The R-CNN family
R-CNN

Regression head to refine the bounding box location

Classification head

Extract features

Warping to a fix size 227 x 227

R-CNN

• Training scheme:
  – 1. Pre-train the CNN on ImageNet
  – 2. Fine-tune the CNN on the number of classes the detector is aiming to classify (softmax loss)
  – 3. Train a linear Support Vector Machine classifier to classify image regions. One SVM per class! (hinge loss)
  – 4. Train the bounding box regressor (L2 loss)
R-CNN

• **PROS:**
  
  – The pipeline of proposals, feature extraction and SVM classification is well-known and tested. Only features are changed (CNN instead of HOG).
  
  – CNN summarizes each proposal into a 4096 vector (much more compact representation compared to HOG).
  
  – Leverage transfer learning: the CNN can be pre-trained for image classification with C classes. One needs only to change the FC layers to deal with Z classes.
R-CNN

**CONS:**

- Slow! 47s/image with VGG16 backbone. One considers around 2000 proposals per image, they need to be warped and forwarded through the CNN.
- Training is also slow and complex
- The object proposal algorithm is fixed. Feature extraction and SVM classifier are trained separately → not exploiting learning to its full potential.

Let us try to solve this first
How do we “pool” these features into a common size

SPP-Net

• It solved the R-CNN problem of being slow at test time

• It still has some problems inherited from R-CNN:
  – Training is still slow (a bit faster than R-CNN)
  – Training scheme is still complex
  – Still no end-to-end training

Fast R-CNN
Fast R-CNN

- 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
- Regions of Interest (RoIs) from a proposal method

Shared computation at test time (like SPP)


Slide credit: Ross Girshick
Fast R-CNN

Region of Interest Pooling

ConvNet

Input image

Forward whole image through ConvNet

“conv5” feature map of image

“RoI Pooling” (single-level SPP) layer

Fully-connected layers

Softmax classifier

Linear + softmax

Linear

Bounding-box regressors

Regions of Interest (RoIs) from a proposal method


Slide credit: Ross Girshick
Fast R-CNN: RoI Pooling

- Region of Interest Pooling

Image (N x M x 3) → Convolutional Neural Network → Feature map (L x K x C) → FC layers expect a fixed size (H x W x C)

Boxes (1000 x 4)
Class scores 1000
Fast R-CNN: RoI Pooling

- Region of Interest Pooling

Image (N x M x 3) → Convolutional Neural Network → Feature map (L x K x C) → FC layers expect a fixed size (H x W x C)

Boxes (1000 x 4)
Class scores 1000

We have to transform this feature map into size (H x W x C)
Fast R-CNN: RoI Pooling

• Region of Interest Pooling

Feature map
(L x K x C)

Zoom in

Boxes
(1000 x 4)

Class scores
1000

FC layers
expect a fixed size
(H x W x C)
Fast R-CNN: RoI Pooling

- Region of Interest Pooling

Feature map \((L \times K \times C)\)

Zoom in

We put a \(H \times W\) grid on top

Boxes \((1000 \times 4)\)

Class scores 1000

FC layers expect a fixed size \((H \times W \times C)\)
Fast R-CNN: RoI Pooling

• Region of Interest Pooling

Feature map 
(L x K x C)

We put a H x W grid on top

Pooling

Feature map 
(H x W x C)

Zoom in

Boxes 
(1000 x 4)

Class scores 
1000

FC layers 
expect a fixed size 
(H x W x C)

Class scores 
1000
Fast R-CNN: RoI Pooling

- RoI Pooling: how do you do backpropagation?

Feature map (L x K x C) → We put a H x W grid on top → Pooling → Feature map (H x W x C) → FC layers expect a fixed size (H x W x C)

Zoom in

Like max-pooling!

Class scores 1000

Boxes (1000 x 4)
**Fast R-CNN Results**

- **VGG-16 CNN on Pascal VOC 2007 dataset**

<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>
</tbody>
</table>
| (Speedup)        | 1x      | 8.8x       | Faster!
## Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

<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>
</tbody>
</table>

Faster! Faster!
Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

<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>
## Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

<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>

The test times do not include proposal generation!
Fast R-CNN Results

- VGG-16 CNN on Pascal VOC 2007 dataset

<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>50 seconds</td>
<td>2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Faster R-CNN
Faster R-CNN:

- Solution: Have the proposal generation integrated with the rest of the pipeline.
- Region Proposal Network (RPN) trained to produce region proposals directly.
- After RPN, everything is like Fast R-CNN.

Region proposal network

- How to extract proposals
  - Image \((N \times M \times 3)\)
  - Zoom in
  - \((H \times W \times 4096)\)
  - Extract proposals

- How many proposals?
  - ✓ We need to decide a fixed number

- Where are they placed?
  - ✓ Densely
Region proposal network

- We fix the number of proposals by using a set of $n=9$ anchors per location.
- $9$ anchors = 3 scales and 3 aspect ratios

Region proposal network

- We fix the number of proposals by using a set of n=9 anchors per location.
- 9 anchors = 3 scales and 3 aspect ratios
- We extract a descriptor per location

Region proposal network

• How to extract proposals

Image
(N x M x 3)

(H x W x 4096)

3x3 conv

(H x W x 256)

#anchors per image?
(H x W x n)
Region proposal network

• How to extract proposals

Image
\((N \times M \times 3)\)

\(\rightarrow\)

\((H \times W \times 4096)\)

3x3 conv

\((H \times W \times 256)\)

1 classification score per proposal (object/non-object)

Anchor regression to proposal box

\(\rightarrow\)

\((H \times W \times (2n+4n))\)

1x1 conv

\((H \times W \times n)\)

#anchors per image?
Region proposal network

- How to extract proposals

Image 
(N x M x 3)

3x3 conv
(H x W x 4096)

1 classification score per proposal (object/non-object)

1x1 conv
(H x W x 256)

Anchor regression to proposal box

Per feature map location, I get a set of anchor correction and classification into object/non-object

RPN

(H x W x (2n+4n))
RPN: training and losses

• Classification ground truth: We compute $p^*$ which indicates how much an anchor overlaps with the ground truth bounding boxes

\[
p^* = 1 \quad if \quad \text{IoU} > 0.7
\]

\[
p^* = 0 \quad if \quad \text{IoU} < 0.3
\]

• 1 indicates the anchor represent an object (foreground) and 0 indicates background object. The rest do not contribute to the training.
RPN: training and losses

• For an image, we randomly sample 256 anchors to form a mini-batch (balanced objects vs. non-objects).
• We calculate the classification loss (binary cross-entropy).
• Those anchors that do contain an object are used to compute the regression loss.
RPN: training and losses

- Each anchor is described by the center position, width and height $x_a, y_a, w_a, h_a$
RPN: training and losses

• Each anchor is described by the center position, width and height $x_a, y_a, w_a, h_a$
• What the network actually predicts are $t_x, t_y, t_w, t_h$

Normalized x

$$t_x = (x - x_a)/w_a,$$  

Normalized y

$$t_y = (y - y_a)/h_a,$$  

Normalized width

$$t_w = \log(w/w_a),$$  

Normalized height

$$t_h = \log(h/h_a),$$

• Smooth L1 loss on regression targets
Faster R-CNN: Training

- First implementation, training of RPN separate from the rest.
- Now we can train jointly!

- Four losses:
  1. RPN classification (object/non-object)
  2. RPN regression (anchor -> proposal)
  3. Fast R-CNN classification (type of object)
  4. Fast R-CNN regression (proposal -> box)
Faster R-CNN

• 10x faster at test time wrt Fast R-CNN
• Trained end-to-end including feature extraction, region proposals, classifier and regressor
• More accurate, since proposals are learned. RPN is fully convolutional
# 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>
Two-stage object detectors
Related works