So far: Image Classification

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector:
4096

Fully-Connected:
4096 to 1000

Slide by: Justin Johnson
Other Computer Vision Tasks

Semantic Segmentation

Classification + Localization

Object Detection

Instance Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

CAT

Single Object

DOG, DOG, CAT

Multiple Object

Slide by: Justin Johnson
Classification + Localization

- **GRASS, CAT, TREE, SKY**
  - No objects, just pixels

- **Single Object**
  - CAT
  - DOG, DOG, CAT
  - DOG, DOG, CAT

Slide by: Justin Johnson
Classification + Localization

Class Scores
- Cat: 0.9
- Dog: 0.05
- Car: 0.01
- ...

Fully Connected: 4096 to 1000

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates
- (x, y, w, h)

Treat localization as a regression problem!

Slide by: Justin Johnson
Classification + Localization

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct box: (x’, y’, w’, h’)

Box Coordinates (x, y, w, h)

L2 Loss

Correct label: Cat

Treat localization as a regression problem!

Slide by: Justin Johnson
Classification + Localization

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Multitask Loss

Vector: 4096

Treat localization as a regression problem!

Fully Connected: 4096 to 1000

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x’, y’, w’, h’)

Treat localization as a regression problem!

May 10, 2017

Classification + Localization

Slide by: Justin Johnson
Classification + Localization

Often pretrained on ImageNet (Transfer learning)

Treat localization as a regression problem!
Plan for this lecture

• Fully supervised detection
  – Pre-CNN: Deformable part models
  – Detection with region proposals: R-CNN, Fast/er R-CNN
  – Detection without region proposals: YOLO
  – Semantic and instance segmentation: FCN, Mask R-CNN

• Weak or out-of-domain supervision
  – Weakly supervised object detection
  – Domain adaptation
Object Detection

No objects, just pixels

Single Object

Multiple Object
Object detection: basic framework

• Build/train object model
• Generate candidate regions in new image
• Score the candidates
Window-template-based models
Building an object model

Given the representation, train a binary classifier
Window-template-based models
Generating and scoring candidates

Car/non-car Classifier

Kristen Grauman
Window-template-based object detection: recap

Training:
1. Obtain training data
2. Define features
3. Define classifier

Given new image:
1. Slide window
2. Score by classifier

Training examples

Feature extraction

Car/non-car Classifier

Kristen Grauman
Dalal-Triggs pedestrian detector

1. Extract fixed-sized (64x128 pixel) window at multiple positions and scales
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
Histograms of oriented gradients (HOG)

Divide image into 8x8 regions

Orientation: 9 bins (for unsigned angles)

Votes weighted by magnitude

Histograms in 8x8 pixel cells

Adapted from Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
Train SVM for pedestrian detection using HoG

$0.16 = w^T x + b$

$\text{sign}(0.16) = 1$

$\Rightarrow$ pedestrian
Remove overlapping detections

Non-max suppression

Score = 0.1
Score = 0.8
Score = 0.8

Adapted from Derek Hoiem
Are window templates enough?

• Many objects are articulated, or have parts that can vary in configuration

Images from Caltech-256, D. Ramanan

• Many object categories look very different from different viewpoints, or from instance to instance

Adapted from N. Snavely, D. Tran
Parts-based Models

Define object by collection of parts modeled by:

1. Appearance
2. Spatial configuration
How to model spatial relations?

• One extreme: fixed template
Fixed part-based template

- Object model = sum of scores of features at fixed positions

\[ +3 +2 -2 -1 -2.5 = -0.5 \quad > \quad 7.5 \]
Non-object

\[ +4 +1 +0.5 +3 +0.5 = 10.5 \quad > \quad 7.5 \]
Object
How to model spatial relations?

• Another extreme: bag of words
How to model spatial relations?

• Star-shaped model
Parts-based Models

- Articulated parts model
  - Object is configuration of parts
  - Each part is detectable and can move around

Adapted from Derek Hoiem, images from Felzenszwalb
Discriminative part-based models

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 32(9), 2010
Discriminative part-based models

Multiple components

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, PAMI 32(9), 2010
Scoring an object hypothesis

• The score of a hypothesis is the sum of appearance scores minus the sum of deformation costs

\[ z = (p_0, \ldots, p_n) \]

\( p_0 \): location of root

\( p_1, \ldots, p_n \): location of parts

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F'_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i, dy_i) + b
\]

Appearance weights  Part features  Deformation weights

i.e. how much we’ll penalize the part \( p_i \) for moving from its expected location

\[
(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)
\]

Displacements

i.e. how much the part \( p_i \) moved from its expected anchor location in the \( x, y \) directions

Felzenszwalb et al.
Car model

Component 1

Component 2
Car detections

high scoring true positives

high scoring false positives
Person model
Person detections

high scoring true positives

high scoring false positives (not enough overlap)
Cat model
Cat detections

high scoring true positives

high scoring false positives (not enough overlap)
“Sliding window” detector
Plan for this lecture

• Fully supervised detection
  – Pre-CNN: Deformable part models
  – Detection with region proposals: R-CNN, Fast/er R-CNN
  – Detection without region proposals: YOLO
  – Semantic and instance segmentation: FCN, Mask R-CNN

• Weak or out-of-domain supervision
  – Weakly supervised object detection
  – Domain adaptation
Complexity and the plateau

[Source: http://pascallin.ics.uci.edu/challenges/VOC/voc20{07,08,09,10,11,12}/results/index.html]

Impact of Deep Learning

![Graph showing the impact of deep learning on mean average precision (mAP) from 2006 to 2016. The graph indicates a significant improvement in mAP after the introduction of deep convolutional neural networks (convnets).]
Object Detection as Regression?

CAT: \((x, y, w, h)\)

DOG: \((x, y, w, h)\)

DUCK: \((x, y, w, h)\)

Slide by: Justin Johnson
Object Detection as Regression?

Each image needs a different number of outputs!
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Dog? NO
Cat? NO
Background? YES
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Dog? YES  
Cat? NO  
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? YES
Cat? NO
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? NO
Cat? YES
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU
Speeding up detection: Restrict set of windows we pass through SVM to those w/ high “objectness”

**Fig. 1: Desired behavior of an objectness measure.** The desired objectness measure should score the blue windows, partially covering the objects, lower than the ground truth windows (green), and score even lower the red windows containing only stuff or small parts of objects.
Fig. 3: **CC success and failure.** **Success:** the windows containing the objects (cyan) have high color contrast with their surrounding ring (yellow) in images (a) and (b). **Failure:** the color contrast for windows in cyan in image (c) is much lower.

Alexe et al., “Measuring the objectness of image windows”, PAMI 2012 and CVPR 2010
Proposals cue: no segments “straddling” the object box

Fig. 5: The SS cue. Given the segmentation (b) of image (a), for a window $w$ we compute $SS(w, \theta_{SS})$ (eq. 4). In (c), most of the surface of $w_1$ is covered by superpixels contained almost entirely inside it. Instead, all superpixels passing by $w_2$ continue largely outside it. Therefore, $w_1$ has a higher SS score than $w_2$. The window $w_3$ has an even higher score as it fits the object tightly.
Proposals cue:
many edges wholly contained inside box
R-CNN

Input image

R-CNN

Regions of Interest (RoI) from a proposal method (~2k)

Input image

R-CNN

R-CNN

R-CNN

- R-CNN
R-CNN

Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

R-CNN: Regions with CNN features

R-CNN at test time: Step 1

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

Proposal-method agnostic, many choices
- Selective Search [van de Sande, Uijlings et al.] (Used in this work)
- Objectness [Alexe et al.]
- Category independent object proposals [Endres & Hoiem]
- CPMC [Carreira & Sminchisescu]

R-CNN at test time: Step 2

Input image  

Extract region proposals (~2k / image)  

Compute CNN features

\[ \text{aeroplane? no.} \]  
\[ \text{person? yes.} \]  
\[ \text{tvmonitor? no.} \]

R-CNN at test time: Step 2

Input image

Extract region proposals (~2k/image)

Compute CNN features

Dilate proposal

R-CNN at test time: Step 2

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

- aeroplane? no.
- person? yes.
- tvmonitor? no.

R-CNN at test time: Step 2

Input image

Extract region proposals (~2k / image)

Compute CNN features

aeroplane? no.

person? yes.

tvmonitor? no.

227 x 227

R-CNN at test time: Step 2

1. **Input image**
2. **Extract region proposals (~2k / image)**
3. **Compute CNN features**
   - **Crop**
   - **Scale (anisotropic)**
   - **Forward propagate**

Output: “fc7” features

---

R-CNN at test time: Step 3

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

- aeroplane? no.
- person? yes.
- tvmonitor? no.
- person? 1.6
- horse? -0.3

- 4096-dimensional fc7 feature vector
- linear classifiers (SVM or softmax)

Step 4: Object proposal refinement

Original proposal

Linear regression on CNN features

Predicted object bounding box

Bounding-box regression

R-CNN on ImageNet detection

ILSVRC2013 detection test set mAP

- *R-CNN BB: 31.4%
- *OverFeat (2): 24.3%
- UvA-Euvision: 22.6%
- *NEC-MU: 20.9%
- *OverFeat (1): 19.4%
- Toronto A: 11.5%
- SYSU_Vision: 10.5%
- GPU_UCLA: 9.8%
- Delta: 6.1%
- UIUC-IFP: 1.0%

R-CNN
Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

Regions of Interest (RoI) from a proposal method (~2k)

Input image

Post hoc component

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 regressions (least squares)

• Training is slow (84h), takes a lot of disk space

• Inference (detection) is slow
  • 47s / image with VGG16 [Simonyan & Zisserman, ICLR15]
Fast R-CNN

• Fast test time
• One network, trained in one stage
• Higher mean average precision

Fast R-CNN

Fast R-CNN

“conv5” feature map of image

ConvNet

Forward whole image through ConvNet

Input image

Fast R-CNN

Fast R-CNN

Fast R-CNN

Fast R-CNN (Training)

- Log loss + Smooth L1 loss
- Linear + softmax
- Linear
- FCs
- ConvNet
- Input image

Multi-task loss

Fast R-CNN (Training)

Log loss + Smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

ConvNet

Input image

## Fast R-CNN vs R-CNN

<table>
<thead>
<tr>
<th></th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train time (h)</td>
<td>9.5</td>
<td>84</td>
</tr>
<tr>
<td>Speedup</td>
<td>8.8x</td>
<td>1x</td>
</tr>
<tr>
<td>Test time / image</td>
<td>0.32s</td>
<td>47.0s</td>
</tr>
<tr>
<td>Test speedup</td>
<td>146x</td>
<td>1x</td>
</tr>
<tr>
<td>mAP</td>
<td>66.9%</td>
<td>66.0%</td>
</tr>
</tbody>
</table>

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

*Girshick, “Fast R-CNN”, ICCV 2015*
Cropping Features: RoI Pool

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Project proposal onto features

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Project proposal onto features

“Snap” to grid cells

Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Cropping Features: RoI Pool

Project proposal onto features

“Snap” to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)


Johnson, Yeung, Fei-Fei
Cropping Features: RoI Pool

Project proposal onto features

“Snap” to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

Region features
(here 512 x 2 x 2; In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!


Cropping Features: RoI Pool

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)
Cropping Features: RoI Pool

Project proposal onto features

“Snap” to grid cells

Divide into 2x2 grid of (roughly) equal subregions

Max-pool within each subregion

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)

Problem: Region features slightly misaligned

Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

Cropping Features: **RoI Align**

Project proposal onto features

No “snapping”!

Input Image  
(e.g. 3 x 640 x 480)

CNN

Image features  
(e.g. 512 x 20 x 15)

He et al, "Mask R-CNN", ICCV 2017
Cropping Features: RoI Align

Project proposal onto features

No “snapping”!

Sample at regular points in each subregion using bilinear interpolation

Input Image (e.g. 3 x 640 x 480)

Image features (e.g. 512 x 20 x 15)

He et al, "Mask R-CNN", ICCV 2017
Cropping Features: RoI Align

Project proposal onto features

Sample at regular points in each subregion using bilinear interpolation

Feature $f_{xy}$ for point $(x, y)$ is a linear combination of features at its four neighboring grid cells.

He et al., "Mask R-CNN", ICCV 2017

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)

No “snapping”!
Cropping Features: RoI Align

Project proposal onto features

No “snapping”!

Sample at regular points in each subregion using bilinear interpolation

Max-pool within each subregion

Region features (here 512 x 2 x 2; In practice e.g 512 x 7 x 7)

Input Image (e.g. 3 x 640 x 480)

Image features (e.g. 512 x 20 x 15)

He et al, "Mask R-CNN", ICCV 2017

Johnson, Yeung, Fei-Fei
Faster R-CNN

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

---

Faster R-CNN

R-CNN Test-Time Speed

- R-CNN: 49
- SPP-Net: 4.3
- Fast R-CNN: 2.3
- Faster R-CNN: 0.2
Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

CNN

Image features
(e.g. 512 x 20 x 15)
Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map.

Input Image (e.g. 3 x 640 x 480)

Image features (e.g. 512 x 20 x 15)
Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map.

At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression).

Anchor is an object? 1 x 20 x 15

Input Image (e.g. 3 x 640 x 480)

Image features (e.g. 512 x 20 x 15)

CNN

Conv

Johnson, Yeung, Fei-Fei
Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map.

Input Image (e.g. 3 x 640 x 480)

Image features (e.g. 512 x 20 x 15)

For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel).

Anchor is an object? 1 x 20 x 15

Box transforms 4 x 20 x 15

CNN

Conv

Johnson, Yeung, Fei-Fei
Region Proposal Network

In practice use $K$ different anchor boxes of different size / scale at each point

Anchor is an object? $K \times 20 \times 15$

Box transforms $4K \times 20 \times 15$

Input Image (e.g. $3 \times 640 \times 480$)

Image features (e.g. $512 \times 20 \times 15$)

CNN

Conv
Region Proposal Network

In practice use $K$ different anchor boxes of different size / scale at each point.

Anchor is an object? $K \times 20 \times 15$

Box transforms $4K \times 20 \times 15$

Sort the $K \times 20 \times 15$ boxes by their “object” score, take top ~300 as our proposals.
Accurate object detection is slow!

<table>
<thead>
<tr>
<th></th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>.05 FPS</td>
</tr>
</tbody>
</table>

⅓ Mile, 1760 feet

Accurate object detection is slow!

<table>
<thead>
<tr>
<th>Method</th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>.05 FPS</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>70.0</td>
<td>.5 FPS</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>73.2</td>
<td>7 FPS</td>
</tr>
<tr>
<td>YOLO</td>
<td>69.0</td>
<td>45 FPS</td>
</tr>
</tbody>
</table>

Plan for this lecture

• Fully supervised detection
  – Pre-CNN: Deformable part models
  – Detection with region proposals: R-CNN, Fast/er R-CNN
  – Detection without region proposals: YOLO
  – Semantic and instance segmentation: FCN, Mask R-CNN

• Weak or out-of-domain supervision
  – Weakly supervised object detection
  – Domain adaptation
Detection without Proposals: YOLO / SSD

Input image
3 x H x W

Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3

Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)


Slide by: Justin Johnson
Split the image into a grid

Each cell predicts boxes and confidences: $P(\text{Object})$
Each cell also predicts a probability $P(\text{Class} \mid \text{Object})$

<table>
<thead>
<tr>
<th>Bicycle</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dining Table</td>
</tr>
</tbody>
</table>
Combine the box and class predictions

Finally do NMS and threshold detections

This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

$$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = 1470 \text{ outputs}$$

YOLO works across many natural images

It also generalizes well to new domains

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
YOLOv2: Fast, Accurate Detection

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Typically use softmax over all classes

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Can’t just mash classes together...

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Can’t just mash classes together...

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Can’t just mash classes together...

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Can’t just mash classes together...

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Each node is a conditional probability

\[
P(\text{canine } | \text{mammal}) \quad \cdots \\
P(\text{dog } | \text{canine}) \\
P(\text{terrier } | \text{dog}) \\
\quad \quad \quad \quad P(\text{Bedlington terrier } | \text{terrier}) \\
P(\text{Norfolk terrier } | \text{terrier}) \\
P(\text{Yorkshire terrier } | \text{terrier})
\]

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Each node is a conditional probability

\[
P(\text{Bedlington terrier}) = P(\text{object}) \times P(\text{living thing} \mid \text{object}) \times \ldots \times P(\text{canine} \mid \text{mammal}) \times P(\text{dog} \mid \text{canine}) \times P(\text{terrier} \mid \text{dog}) \times P(\text{Bedlington terrier} \mid \text{terrier})
\]

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Atlantic bottlenose dolphin

skin-diver

Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Plan for this lecture

• Fully supervised detection
  – Pre-CNN: Deformable part models
  – Detection with region proposals: R-CNN, Fast/er R-CNN
  – Detection without region proposals: YOLO
  – Semantic and instance segmentation: FCN, Mask R-CNN

• Weak or out-of-domain supervision
  – Weakly supervised object detection
  – Domain adaptation
Semantic Segmentation

Grass, Cat, Tree, Sky

No objects, just pixels

Single Object

Dog, Dog, Cat

Multiple Object
Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels
Semantic Segmentation Idea: Sliding Window

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014
Semantic Segmentation Idea: Sliding Window

Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Input: $3 \times H \times W$

Convolutions:
$D \times H \times W$

Scores: $C \times H \times W$

Predictions: $H \times W$

Slide by: Justin Johnson
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Problem: convolutions at original image resolution will be very expensive ...

Input: 3 x H x W

Convolutions: D x H x W

Scores: C x H x W

Predictions: H x W
Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!


Slide by: Justin Johnson
Semantic Segmentation Idea: Fully Convolutional

**Downsampling:** Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

**Input:** 3 x H x W

**High-res:**
- D₁ x H/2 x W/2
- D₂ x H/4 x W/4
- D₃ x H/4 x W/4

**Med-res:**
- D₂ x H/4 x W/4

**Low-res:**
- D₃ x H/4 x W/4

**High-res:**
- D₁ x H/2 x W/2

**Predictions:** H x W


Slide by: Justin Johnson
In-Network upsampling: “Unpooling”

Nearest Neighbor

```
1 2
3 4
```

Input: 2 x 2

```
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
```

Output: 4 x 4

“Bed of Nails”

```
1 2
3 4
```

Input: 2 x 2

```
1 0 2 0
0 0 0 0
3 0 4 0
0 0 0 0
```

Output: 4 x 4

Slide by: Justin Johnson
In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Input: 4 x 4

Max Unpooling
Use positions from pooling layer

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Input: 2 x 2

Output: 4 x 4

Rest of the network

Corresponding pairs of downsampling and upsampling layers
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2
Output: 4 x 4

Slide by: Justin Johnson
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

Slide by: Justin Johnson
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

Sum where output overlaps

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Slide by: Justin Johnson
Transposing a 1D convolution involves applying the convolution operation in the opposite direction. The process involves taking the transpose of the filter and sliding it across the input signal, computing the dot product at each position to produce the output. This operation is useful in scenarios where the input signal is time-reversed, such as in certain types of signal processing or neural network architectures. The diagram illustrates this process visually, with the input signal and filter being convolved to produce the output. The output contains copies of the filter weighted by the input, summing at where they overlap.
Semantic Segmentation Idea: Fully Convolutional

**Downsampling:**
Pooling, strided convolution

**Upsampling:**
Unpooling or strided transpose convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

- **Input:** $3 \times H \times W$
- **High-res:** $D_1 \times H/2 \times W/2$
- **Low-res:** $D_3 \times H/4 \times W/4$
- **Med-res:** $D_2 \times H/4 \times W/4$
- **High-res:** $D_1 \times H/2 \times W/2$
- **Predictions:** $H \times W$


Slide by: Justin Johnson
Instance Segmentation

No objects, just pixels

Single Object

Multiple Object

GRASS, CAT, TREE, SKY

CAT

DOG, DOG, CAT

DOG, DOG, CAT

Slide by: Justin Johnson
Mask R-CNN

He et al, "Mask R-CNN", ICCV 2017

What is Mask R-CNN: Parallel Heads

• Easy, fast to implement and use

(slow) R-CNN

Fast/er R-CNN

Mask R-CNN

Slide by: Kaiming He
Mask R-CNN

He et al, "Mask R-CNN", ICCV 2017

Classification Scores: C
Box coordinates (per class): 4 * C

Predict a mask for each of C classes

Adapted from Justin Johnson
Mask R-CNN: Example Mask Training Targets
Plan for this lecture

• Fully supervised detection
  – Pre-CNN: Deformable part models
  – Detection with region proposals: R-CNN, Fast/er R-CNN
  – Detection without region proposals: YOLO
  – Semantic and instance segmentation: FCN, Mask R-CNN

• Weak or out-of-domain supervision
  – Weakly supervised object detection
  – Domain adaptation
What if no bounding boxes to train?

• Weakly supervised object detection
  – Image-level class labels
  – Image-level captions
Class activation maps

Class activation maps

- Let $f_k(x, y)$ be the activation in the $k$-th map at location $(x, y)$
- Global average pooling: $F^k = \sum_{x,y} f_k(x, y)$
- Input to softmax is $S_c = \sum_k w^c_k F^k$ where $w^c_k$ is the weight for class $c$ and map $k$

$$S_c = \sum_k w^c_k \sum_{x,y} f_k(x, y) = \sum_{x,y} \sum_k w^c_k f_k(x, y)$$

- Map for class $c$:

$$M_c(x, y) = \sum_k w^c_k f_k(x, y)$$

Class activation maps

Class activation maps of top 5 predictions

Class activation maps for one object class

Table 3. Localization error on the ILSVRC test set for various weakly- and fully-supervised methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>supervision</th>
<th>top-5 test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet-GAP (heuristics)</td>
<td>weakly</td>
<td>37.1</td>
</tr>
<tr>
<td>GoogLeNet-GAP</td>
<td>weakly</td>
<td>42.9</td>
</tr>
<tr>
<td>Backprop [23]</td>
<td>weakly</td>
<td>46.4</td>
</tr>
<tr>
<td>OverFeat [22]</td>
<td>full</td>
<td>29.9</td>
</tr>
<tr>
<td>AlexNet [25]</td>
<td>full</td>
<td>34.2</td>
</tr>
</tbody>
</table>

Class activation maps

Figure 5. Class activation maps from CNN-GAPs and the class-specific saliency map from the backpropagation methods.

Figure 6. a) Examples of localization from GoogleNet-GAP. b) Comparison of the localization from GoogleNet-GAP (upper two) and the backpropagation using AlexNet (lower two). The ground-truth boxes are in green and the predicted bounding boxes from the class activation map are in red.

Localization from captions

Ye et al., “Learning to discover and localize visual objects with open vocabulary”, arxiv 2018
Localization from sound

Harwath et al., “Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input”, ECCV 2018
Localization from sound

Fig. 7: On the left are shown two images and their speech signals. Each color corresponds to one connected component derived from two matchmaps from a fully random MISA network. The masks on the right display the segments that correspond to each speech segment. We show the caption words obtained from the ASR transcriptions below the masks. Note that those words were never used for learning, only for analysis.

Harwath et al., “Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input”, ECCV 2018
Detection from documentaries

Chen et al., “Discover and Learn New Objects from Documentaries”, CVPR 2017
What if test data very diff from train?

- Adapt detection models
Adapting classifiers

(a) Visual variability of the class “horse” across domains

(b) Overview of our method

Thomas and Kovashka, “Artistic Object Recognition by Unsupervised Style Adaptation”, ACCV 2018
Adapting classifiers

Fig. 2. Training with multiple modalities and style-invariance constraint. We train networks on real and synthetic data. We show an example of style transfer transforming photos into labeled synthetic cartoons. The style-invariance loss trains the FC2 layer to predict which modality the image came from. During backpropagation, we reverse its gradient before propagating it to the layers used by both classifiers. This encourages those layers to learn style-invariant features.
Adapting detectors

Figure 1. Illustration of different datasets for autonomous driving: From top to bottom-right, example images are taken from: KITTI [17], Cityscapes [5], Foggy Cityscapes [49], SIM10K [30]. Though all datasets cover urban scenes, images in those datasets vary in style, resolution, illumination, object size, etc. The visual difference between those datasets presents a challenge for applying an object detection model learned from one domain to another domain.
Adapting detectors

**Figure 2. An overview of our Domain Adaptive Faster R-CNN model:** we tackle the domain shift on two levels, the image level and the instance level. A domain classifier is built on each level, trained in an adversarial training manner. A consistency regularizer is incorporated within these two classifiers to learn a domain-invariant RPN for the Faster R-CNN model.
Adapting detectors

<table>
<thead>
<tr>
<th></th>
<th>img</th>
<th>ins</th>
<th>cons</th>
<th>car AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>✔</td>
<td></td>
<td></td>
<td>30.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33.03</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
<td>35.79</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>37.86</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td><strong>38.97</strong></td>
</tr>
</tbody>
</table>

Table 1. The average precision (AP) of Car on the Cityscapes validation set. The models are trained using the SIM 10k dataset as the source domain and the Cityscapes training set as the target domain. **img** is short for image-level alignment, **ins** for instance-level alignment and **cons** is short for our consistency loss.

Figure 3. Error Analysis of Top Ranked Detections

Chen et al., “Domain Adaptive Faster R-CNN for Object Detection in the Wild”, CVPR 2018