Different Flavors of Object Recognition

Semantic Segmentation

Classification + Localization

Object Detection

Instance Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

CAT

Single Object

DOG, DOG, CAT

Multiple Object

Adapted from Justin Johnson
Plan for the next few lectures

• Detection approaches
  – Pre-CNNs
    • Detection with whole windows: Pedestrian detection
    • Part-based detection: Deformable Part Models
  – Post-CNNs
    • Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    • Detection without region proposals: YOLO, SSD

• Segmentation approaches
  – Semantic segmentation
    • Fully-Convolutional Networks (FCN)
  – Instance segmentation
    • Mask R-CNN
    • Segment Anything

• Learning from noisy web image-text data
  – Contrastive Language-Image Pretraining (CLIP)
    • Prompting
  – Open-vocabulary object detection
Object Detection

- **Object Detection**
  - **Single Object**
  - **Multiple Object**

**Images:**
- **GRASS, CAT, TREE, SKY**
  - No objects, just pixels
- **CAT**
- **DOG, DOG, CAT**
- **DOG, DOG, CAT**

**May 10, 2017**

Slide by: Justin Johnson
Object detection: basic framework

• Build/train object model
• Generate candidate regions in new image
• Score the candidates

Adapted from Kristen Grauman
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
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

Feature extraction

Training examples

Car/non-car Classifier
Evaluating detection methods

\[
\text{mAP} = \frac{1}{|\text{classes}|} \sum_{c \in \text{classes}} \frac{\#\text{TP}(c)}{\#\text{TP}(c) + \#\text{FP}(c)}
\]

- True Positive - TP(c): a predicted bounding box (\text{pred\_bb}) was made for class c, there is a ground truth bounding box (\text{gt\_bb}) of class c, and IoU(\text{pred\_bb}, \text{gt\_bb}) \geq 0.5.
- False Positive - FP(c): a pred\_bb was made for class c, and there is no gt\_bb of class c. Or there is a gt\_bb of class c, but IoU(\text{pred\_bb}, \text{gt\_bb}) < 0.5.
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
Histograms of oriented gradients (HOG)

Divide image into 8x8 regions

Orientation: 9 bins (for unsigned angles)

Histograms in 8x8 pixel cells

Votes weighted by magnitude

Adapted from Pete Barnum

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

\[ 0.16 = w^T x + b \]

\[ \text{sign}(0.16) = 1 \]

\[ \Rightarrow \text{pedestrian} \]

Adapted from Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05
Remove overlapping detections

Non-max suppression

Score = 0.8

Score = 0.1

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

\begin{align*}
\text{Non-object} & : +3 +2 -2 -1 -2.5 = -0.5 > 7.5 \\
\text{Object} & : +4 +1 +0.5 +3 +0.5 = 10.5 > 7.5
\end{align*}
How to model spatial relations?

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

- Star-shaped model

![Image of star-shaped models](image-url)
Parts-based Models

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

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

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

- Appearance weights
- Part features
- Deformation weights

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

Displacements

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

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

Felzenszwalb et al.
Detection

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
Complexity and the plateau

[Source: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc20{07,08,09,10,11,12}/results/index.html]

PASCAL VOC challenge dataset

Impact of Deep Learning

Slide by: Justin Johnson
Before: Image Classification with CNNs

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

Vector: 4096

Fully-Connected: 4096 to 1000
Classification + Localization

- **No objects, just pixels**
- **Single Object**
- **Multiple Object**

Slide by: Justin Johnson
Classification + Localization

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

Vector: 4096

Fully Connected: 4096 to 1000

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
...

Vector: 4096

Fully Connected: 4096 to 1000

Box Coordinates
(x, y, w, h)

Fully Connected: 4096 to 4

Correct box:
(x', y', w', h')

Correct label:
Cat

Softmax Loss

L2 Loss

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

Multitask Loss

Vector: 4096

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!

Slide by: Justin Johnson
Classification + Localization

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

Vector: 4096

Fully Connected: 4096 to 1000

Box Coordinates
(x, y, w, h)

Fully Connected: 4096 to 4

Correct label:
Cat

Softmax Loss

Correct box:
(x', y', w', h')

Treat localization as a regression problem!

Often pretrained on ImageNet
(Transfer learning)

Slide by: Justin Johnson
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
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

Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Slide by: Justin Johnson
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.
Objectness cue #1: Where people look

Fig. 2: MS success and failure.
Objectness cue #2: color contrast at boundary

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.
Objectness cue #3: 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.
R-CNN

R-CNN

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

Input image

R-CNN

Warped image regions

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

Input image

R-CNN

R-CNN

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

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

Input image

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)

Input image

R-CNN: Regions with CNN features

Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions (linear SVM)

R-CNN at test time: Step 1

Input image \rightarrow 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

R-CNN at test time: Step 2

Input image

Extract region proposals (~2k/image)

Compute CNN features

Dilate proposal

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

R-CNN at test time: Step 2

Input image

Extract region proposals (~2k / image)

Compute CNN features

a. Crop

R-CNN at test time: Step 2

Input image

Extract region proposals (~2k / image)

Compute CNN features

a. Crop
b. Scale (anisotropic)

R-CNN at test time: Step 2

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

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.

Step 4: Object proposal refinement

Original proposal

Predicted object bounding box

Linear regression on CNN features

Bounding-box regression

R-CNN on ImageNet detection

ILSVRC2013 detection test set mAP

- *R-CNN BB: 31.4%
- *OverFeat (2): 24.3%
- UvA-Euvison: 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

Warped image regions

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
  • Train post-hoc linear SVMs (hinge loss)
  • Train post-hoc bounding-box regressions (L2 loss)

• Training is slow (84h), takes a lot of disk space
  • Need to store all region crops

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

Adapted from Girshick, “Fast R-CNN”, ICCV 2015
Fast R-CNN

• One network, applied one time, not 2000 times
• Trained end-to-end (in one stage)
• Fast test time
• Higher mean average precision

Adapted from Girshick, “Fast R-CNN”, ICCV 2015
Fast R-CNN

Fast R-CNN

“conv5” feature map of image

Forward whole image through ConvNet

ConvNet

Input image

Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“conv5” feature map of image

Forward whole image through ConvNet

ConvNet

Input image

Fast R-CNN

Fast R-CNN

Fast R-CNN

Fast R-CNN (Training)

Input image

ConvNet

FCs

Linear

Linear + softmax

Log loss + Smooth L1 loss

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*
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

---

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></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></td>
<td></td>
<td>14 s/img</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>.05 FPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 s/img</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>70.0</td>
<td>.5 FPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 s/img</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>73.2</td>
<td>7 FPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>140 ms/img</td>
</tr>
<tr>
<td>YOLO</td>
<td>69.0</td>
<td>45 FPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 ms/img</td>
</tr>
</tbody>
</table>

Detection without Proposals: YOLO

Each cell predicts boxes and confidences: $P(\text{Object})$
Each cell also predicts a probability
\[ P(\text{Class} \mid \text{Object}) \]
Combine the box and class predictions
Finally do NMS and threshold detections
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)

Input image
$3 \times H \times W$

Divide image into grid
$7 \times 7$

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

Output:
$7 \times 7 \times (5 \times B + C)$

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

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

YOLO works across many natural images
It also generalizes well to new domains

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
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017
Plan for the next few lectures

• Detection approaches
  – Pre-CNNs
    • Detection with whole windows: Pedestrian detection
    • Part-based detection: Deformable Part Models
  – Post-CNNs
    • Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    • Detection without region proposals: YOLO, SSD

• Segmentation approaches
  – Semantic segmentation
    • Fully-Convolutional Networks (FCN)
  – Instance segmentation
    • Mask R-CNN
    • Segment Anything

• Learning from noisy web image-text data
  – Contrastive Language-Image Pretraining (CLIP)
    • Prompting
  – Open-vocabulary object detection
Semantic Segmentation

- **GRASS, CAT, TREE, SKY**
  - No objects, just pixels
- **CAT**
  - Single Object
- **DOG, DOG, CAT**
  - Multiple Object

Slide by: Justin Johnson
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

Slide by: Justin Johnson
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: $H \times W$

Semantic Segmentation Idea: Fully Convolutional
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: $H \times W$

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

Slide by: Justin Johnson
Semantic Segmentation Idea: Fully Convolutional

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

Input: \(3 \times H \times W\)

High-res: \(D_1 \times H/2 \times W/2\)

Med-res: \(D_2 \times H/4 \times W/4\)

Low-res: \(D_3 \times H/4 \times W/4\)

High-res: \(D_1 \times H/2 \times W/2\)

Predictions: \(H \times W\)

---


---

Slide by: Justin Johnson
Semantic Segmentation Idea: Fully Convolutional

**Input:**
3 x H x W

**Prediction:**
H x W

**Downsampling:**
Pooling, strided convolution

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

**High-res:**
D₃ x H/4 x W/4

**Med-res:**
D₂ x H/4 x W/4

**Low-res:**
D₁ x H/2 x W/2

**Upsampling:**
???


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

Nearest Neighbor

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

Input: 2 x 2

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

Output: 2 x 2

Rest of the network

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

Corresponding pairs of downsampling and upsampling layers

May 10, 2017

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
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

May 10, 2017

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
Transpose Convolution: 1D Example

Input: a, b

Filter: x, y, z

Output: ax, ay, az, bx, by, bz

Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.

Adapted from Justin Johnson
Instance Segmentation

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

CAT
Single Object

DOG, DOG, CAT
Multiple Object

May 10, 2017
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

Adapted from Justin Johnson
Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a promptable segmentation task, a segmentation model (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a data engine for collecting SA-1B, our dataset of over 1 billion masks.
Plan for the next few lectures

• Detection approaches
  – Pre-CNNs
    • Detection with whole windows: Pedestrian detection
    • Part-based detection: Deformable Part Models
  – Post-CNNs
    • Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    • Detection without region proposals: YOLO, SSD

• Segmentation approaches
  – Semantic segmentation
    • Fully-Convolutional Networks (FCN)
  – Instance segmentation
    • Mask R-CNN
    • Segment Anything

• Learning from noisy web image-text data
  – Contrastive Language-Image Pretraining (CLIP)
    • Prompting
  – Open-vocabulary object detection
Learning from noisy web data

• Massive datasets of image-text pairs from the web
  – E.g. alt text, Flickr, Reddit, Wikipedia, etc
• Images and their co-occurring text assumed related (text provides a reasonable description of image?)
• Train text and image feature extractors using the objective that matched (co-occurring) image-text should be more similar than mismatched ones
• Great performance at a low annotation cost (data already existed)
Contrastive Language-Image Pretraining (CLIP)

Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset’s classes.

Using CLIP for Object Recognition

• Compute dot product of image and prompt for each class, e.g. “A photo of dog”
• Return class with highest dot product for each image
• Prompt can be optimized manually or through training
• Can extend idea for object detection
Fig. 1 Prompt engineering vs Context Optimization (CoOp). The former needs to use a held-out validation set for words tuning, which is inefficient; the latter automates the process and requires only a few labeled images for learning.


Figure 1: Schematic of the method. (Left) The standard method of a zero-shot open vocabulary image classification model (e.g., CLIP [Radford et al., 2021]). (Right) Our method of CuPL. First, an LLM generates descriptive captions for given class categories. Next, an open vocabulary model uses these captions as prompts for performing classification.
Figure 2: An overview of using ViLD for open-vocabulary object detection. ViLD distills the knowledge from a pretrained open-vocabulary image classification model. First, the category text embeddings and the image embeddings of cropped object proposals are computed, using the text and image encoders in the pretrained classification model. Then, ViLD employs the text embeddings as the region classifier (ViLD-text) and minimizes the distance between the region embedding and the image embedding for each proposal (ViLD-image). During inference, text embeddings of novel categories are used to enable open-vocabulary detection.