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

No objects, just pixels

Single Object

Multiple Object

GRASS, CAT, TREE, SKY

CAT

DOG, DOG, CAT

DOG, DOG, CAT

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
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
Evaluating detection methods

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

- True Positive - TP(c): a predicted bounding box (pred_bb) was made for class c, there is a ground truth bounding box (gt_bb) of class c, and IoU(pred_bb, gt_bb) >= 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(pred_bb, 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 = \mathbf{w}^T \mathbf{x} + b

\text{sign}(0.16) = 1

\Rightarrow \text{pedestrian}
Remove overlapping detections

Non-max suppression

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

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

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

\[ p_0 : \text{location of root} \]

\[ p_1, \ldots, p_n : \text{location of parts} \]

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

Appearance weights  Part features  Deformation weights

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

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

\( \text{Displacements} \)
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]

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

Classification + Localization

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

DOG, DOG, CAT

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

Treat localization as a regression problem!

Slide by: Justin Johnson
Classification + Localization

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

Fully Connected: 4096 to 1000

Vector: 4096

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

Softmax Loss

L2 Loss

Correct label: Cat

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
- ...

Softmax Loss

Multitask Loss

Fully Connected: 4096 to 1000

Vector: 4096

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

Treat localization as a regression problem!

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

L2 Loss

May 10, 2017

Correct label: Cat

Treat localization as a regression problem!

Classification + Localization Slide by: Justin Johnson
Classification + Localization

Often pretrained on ImageNet (Transfer learning)

Treat localization as a regression problem!

Correct label:
Cat

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

Vector:
4096

Fully Connected:
4096 to 1000

Softmax Loss

Box Coordinates
(x, y, w, h)

L2 Loss

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

Fully Connected:
4096 to 4
Object Detection as Regression?

CAT: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)

DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

....
Object Detection as Regression?

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

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

DUCK: \((x, y, w, h)\)  
Many numbers!

Each image needs a different number of outputs!

Slide by: Justin Johnson
Object Detection as Classification:
Sliding Window

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

Slide by: Justin Johnson
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 “bloppy” 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
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 \( \text{SS}(w, \theta_{\text{SS}}) \) (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

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 → 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

- aeroplane? no.
- person? yes.
- 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

a. Crop

- 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
- b. Scale (anisotropic)

R-CNN at test time: Step 2

1. Crop
2. Scale (anisotropic)
3. 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.

4096-dimensional fc7 feature vector

linear classifiers (SVM or softmax)

person? 1.6
horse? -0.3
...
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-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%

**Legend:**
- Red: post competition result
- Blue: competition result

R-CNN

Linear Regression for bounding box offsets

Classification of 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

Fast R-CNN

Regions of Interest (RoIs) from a proposal method

"conv5" feature map of image

Forward whole image through ConvNet

Input image

Fast R-CNN

Regions of Interest (RoIs) from a proposal method

“RoI Pooling” layer

“conv5” feature map of image

Forward whole image through ConvNet

Input image

Fast R-CNN

Fast R-CNN

Fast R-CNN (Training)

Multi-task loss

Log loss + Smooth L1 loss

Linear + softmax

Linear

FCs

ConvNet

Input image

Fast R-CNN (Training)

### 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*
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

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>Model</th>
<th>Pascal 2007 mAP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5</td>
<td>33.7</td>
<td>0.07 FPS</td>
</tr>
<tr>
<td>R-CNN</td>
<td>66.0</td>
<td>0.05 FPS</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>70.0</td>
<td>0.5 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>

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
Detection without Proposals: YOLO

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)

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
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

No objects, just pixels

Single Object

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
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
Input: $3 \times H \times W$

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

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 ...

May 10, 2017

Semantic Segmentation Idea: Fully Convolutional

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:** \(D_1 \times H/2 \times W/2\)

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

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

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

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

Predictions: \(H \times W\)

---


---

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!

**Upsampling:**
???

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: $H \times W$

---


In-Network upsampling: “Unpooling”

Nearest Neighbor

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

Input: 2 x 2

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

Output: 4 x 4

“Bed of Nails”

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

Input: 2 x 2

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

Output: 4 x 4

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

Max Pooling
Remember which element was max!

Max Unpooling
Use positions from pooling layer

Corresponding pairs of downsampling and upsampling layers

Input: 4 x 4
Output: 2 x 2

Rest of the network

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

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

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
Transpose Convolution: 1D Example

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

Fig. 2 Overview of Context Optimization (CoOp). The main idea is to model a prompt’s context using a set of learnable vectors, which can be optimized through minimizing the classification loss. Two designs are proposed: one is unified context, which shares the same context vectors with all classes; and the other is class-specific context, which learns for each class a specific set of context vectors.
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. The framework of the Knowledge-guided Context Optimization for prompt tuning. $L_{ce}$ is the standard cross-entropy loss, and $L_{kg}$ is the proposed Knowledge-guided Context Optimization contraint to minimize the discrepancy between the special knowledge (learnable textual embeddings) and the general knowledge (the textual embeddings generated by the hand-crafted prompt).

\[
L_{kg} = \frac{1}{N_c} \sum_{i=1}^{N_c} \| w_i - w_i^{clip} \|_2^2,
\]

where $\| \cdot \|$ is the euclidean distance, $N_c$ is the number of seen classes. Meanwhile, the standard contrastive loss is:

\[
L_{ce} = -\sum_{x \in X} \log \frac{\exp(d(x, w_y)/\tau)}{\sum_{i=1}^{N_c} \exp(d(x, w_i)/\tau)},
\]

where $y$ is the corresponding label of the image embedding.

By combining the standard cross-entropy loss $L_{ce}$, the final objective is:

\[
\mathcal{L} = L_{ce} + \lambda L_{kg},
\]

where $\lambda$ is used balance the effect of $L_{kg}$.
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.