Lecture: Object detection

Juan Carlos Niebles and Ranjay Krishna
Stanford Vision and Learning Lab
What we will learn today

• Object detection
  – Task and evaluation
• A simple detector
• Deformable parts model
What we will learn today

• Object detection
  – Task and evaluation
• A simple detector
• Deformable parts model
Object detection

• What do you see in the image?
Object Detection

- **Problem**: Detecting and localizing generic objects from various categories, such as cars, people, etc.

- **Challenges**:
  - Illumination,
  - viewpoint,
  - deformations,
  - intraclass variability
Vision Contests

• PASCAL VOC Challenge

• 20 categories
• Annual classification, detection, segmentation, ... challenges
How do we evaluate object detection?
How do we evaluate object detection?

True positive:
- The overlap of the prediction with the ground truth is MORE than 0.5
How do we evaluate object detection?

True positive:
- The overlap of the prediction with the ground truth is LESS than 0.5

False positive:
- The overlap of the prediction with the ground truth is LESS than 0.5
How do we evaluate object detection?

True positive:
False positive:
False negative:
- The objects that our model doesn’t find
How do we evaluate object detection?

- True positive: The objects that our model finds
- False positive: The objects that our model incorrectly predicts
- False negative: The objects that our model doesn’t find

What is a True Negative?
<table>
<thead>
<tr>
<th></th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>true positive</td>
<td>false negative</td>
</tr>
<tr>
<td>True 0</td>
<td>false positive</td>
<td>true negative</td>
</tr>
<tr>
<td>True 1</td>
<td>Predicted 1</td>
<td>Predicted 0</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>True 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **True Positive**
- **False Negative**
- **False Positive**
- **True Negative**

- **TP**
- **FN**
- **FP**
- **TN**
<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td><strong>True 0</strong></td>
</tr>
<tr>
<td>true positive</td>
<td>false negative</td>
</tr>
<tr>
<td>false positive</td>
<td>true negative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td><strong>True 0</strong></td>
</tr>
<tr>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td><strong>True 0</strong></td>
</tr>
<tr>
<td>hits</td>
<td>misses</td>
</tr>
<tr>
<td>false alarms</td>
<td>correct rejections</td>
</tr>
<tr>
<td>True 1</td>
<td>Predicted 1</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>True 1</td>
<td>true positive</td>
</tr>
<tr>
<td>True 0</td>
<td>false positive</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>True 1</th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>True 0</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

**Precision**:

\[
\text{precision} = \frac{TP}{TP + FP}
\]

**Recall**:

\[
\text{recall} = \frac{TP}{TP + FN}
\]
How do we evaluate object detection?

- predictions
- ground truth

True positive: 1
False positive: 2
False negative: 1

So what is the precision?
- recall?
Precision versus recall

• Precision:
  – how many of the object detections are correct precision

• Recall:
  – how many of the ground truth objects can the model detect?
In reality, our model makes a lot of predictions with varying scores between 0 and 1.

Here are all the boxes that are predicted with score > 0.

This means that our
- Recall is perfect!
- But our precision is BAD!
In reality, our model makes a lot of predictions with varying scores between 0 and 1.

There are no boxes that are predicted with score $= 1$.

This means that our
- Precision is undefined!
- And our recall is BAD!
How do we evaluate object detection?

Here are all the boxes that are predicted with score $> 0.5$

We are setting a threshold of 0.5
Precision – recall curve (PR curve)
Which model is the best?
Which model is the best?
Properties of ROC

- 1.0: perfect prediction
- 0.9: excellent prediction
- 0.8: good prediction
- 0.7: mediocre prediction
- 0.6: poor prediction
- 0.5: random prediction
- <0.5: something went wrong!
Properties of ROC

• Slope is non-increasing
• Each point on ROC represents different tradeoff (cost ratio) between false positives and false negatives
• ROC Area represents performance averaged over all possible thresholds
• If two ROC curves do not intersect, one method dominates the other
• If two ROC curves intersect, one method is better for some thresholds, and other method is better for other thresholds
True Positives - Person

UoCTTI_LSVM-MDPM

MIZZOU_DEF-HOG-LBP

NECUIUC_CLS-DTCT
False Positives - Person

UoCTTI_LSV-MDPM

MIZZOU_DEF-HOG-LBP

NECUIUC_CLS-DTCT
“Near Misses” - Person

UoCTTI_LSV-HOG-MDPM

MIZZOU_DEF-HOG-LBP

NECUIUC_CLS-DTCT
True Positives - Bicycle

UoCTTI_LSVM-MDPM

OXFORD_MKL

NECUIUC_CLS-DTCT
False Positives - Bicycle

UoCTTI_LSVM-MDPM

OXFORD_MKL

NECUIUC_CLS-DTCT
What we will learn today

• Object detection
  – Task and evaluation
• A simple detector
• Deformable parts model
Dalal-Triggs method

sliding window
Recap – HOG filters
Sliding window + hog features

• Slide through the image and check if there is an object at every location

No person here
Sliding window + hog features

- Slide through the image and check if there is an object at every location

YES!! Person match found
Sliding window + hog features

• But what if we were looking for buses?

No bus found
Sliding window + hog features

• But what if we were looking for buses?

No bus found
Sliding window + hog features

- We will never find the object we don’t choose our window size wisely!

No bus found
Sliding window + hog features

• We need to do multi scale sliding window
Create a feature pyramid

Filter $F$

Score of $F$ at position $p$ is $F \cdot \phi(p, H)$

$\phi(p, H) = \text{concatenation of HOG features from subwindow specified by } p$
What we will learn today

• Object detection
  – Task and evaluation
• A simple detector
• Deformable parts model
Recap – bag of visual words

• We can present images as a set of words
  – Where each word represents a part of the image.

• Can we do the same for objects within those images?
Deformable parts model

• Represents an object as a collection of parts arranged in a deformable configuration
• Each part represents local appearances
• Spring-like connections between certain pairs of parts

Fischler and Elschlager, Pictoral Structures, 1973
Deformable parts model

- The parts of an object form pairwise relationships.
- We can model this using a “star model”
  - where every part is defined relative to a root.
Detecting a person with their parts

- For example, a person can be modelled as having a head, left arm, right arm, etc.
- All parts can be modelled relative to the global person detector
Deformable parts model

• Mixture of deformable part models
• Each component has global component + deformable parts

• Part filters have finer details
Deformable parts model

• Each model will have a **global** filter. And a set of **part** filters. Here is an example of a global person filter with it’s ‘head’ part filter:
Two-component bicycle model

“side” component

“frontal” component
Six-component person model
Six-component car model

side view

frontal view

root filters (coarse)  part filters (fine)  deformation models

Stanford University  Lecture 12  14-Nov-17
Remember from Dalal and Triggs

Filter $F$

Score of $F$ at position $p$ is $F \cdot \phi(p, H)$

$\phi(p, H) =$ concatenation of HOG features from subwindow specified by $p$
Deformable parts model

- A model for an object with $n$ parts is a $(n + 2)$ tuple:

$$ (F_0, P_1, \ldots, P_n, b) $$

- Each part-based model defined as:

$$ (F_i, v_i, d_i) $$

$F_i$ filter for the $i$-th part

$v_i$ “anchor” position for part $i$ relative to the root position

$d_i$ defines a deformation cost for each possible placement of the part relative to the anchor position
Deformable parts calculates a score for each part along with a global score

\[ p_i = (x_i, y_i, l_i) \] specifies the level and position of the \( i \)-th filter

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

- \( p_0 \): location of root
- \( p_1, \ldots, p_n \): location of parts
Calculating the score for a detection

The score for a detection is defined as the score for the global detector minus the sum of deformation costs for each part.

This means that if a detection’s parts are really far away from where they should be, it’s probably a false positive.
Calculating the score for a detection

The score for a detection is defined as the score for the global detector minus the sum of deformation costs for each part.
Calculating the score for a detection

The score for a detection is defined as the score for the global detector minus the sum of deformation costs for each part.

\[
\text{detection score} = \prod_{i=0}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2)
\]
Calculating the score for a detection

\[
detection \ score = \prod_{i=0}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2)
\]

Scores for each part filter + global filter (same as Dalal and Triggs).
Calculating the score for a detection

detection score

\[
= \prod_{i=0}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2)
\]

The deformation costs for each part. \(dx_i\) measures the distance in the x-direction from where part \(i\) should be. \(dy_i\) measures the same in the y-axis direction. \(d_i\) is the weight associated for part \(i\) that penalizes the part for being away.
Calculating the score for a detection

detection score

\[ = \prod_{i=0}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2) \]

If \( d_i = (0, 0, 1, 0) \). What does this mean?
Detection pipeline

- So, to make a detection, we use the sliding window technique and use the global filter first.
- Whenever, the global filters detects an object, we use the part filters to calculate it’s score.
Overall detection pipeline

Let’s break this down
Detection pipeline

First, make sure you have filters for the global and the parts: $F_i$
Detection pipeline

Apply the filters:

\[
\prod_{i=0}^{n} F_i \phi(p_i, H)
\]
Transformation

Given the location for the detected head, we can guess where the body should be.

The body should be in the direction calculated from the root person filter.
Detection pipeline

Now apply the spatial costs:

detection score

\[
= \prod_{i=1}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2)
\]
Detection pipeline

Now add the global filter:

\[ \text{detection score} \]

\[
F_0 + \prod_{i=1}^{n} F_i \phi(p_i, H) - \sum_{i=1}^{n} d_i (dx_i, dy_i, dx_i^2, dy_i^2)
\]
DPM - bicycle

root filters
coarse resolution

part filters
finer resolution

deformation models
DPM - person

root filters
coarse resolution

part filters
finer resolution

deformation models
DPM - bottle

- Root filters
  - Coarse resolution
- Part filters
  - Finer resolution
- Deformation models
Results – car detection

high scoring true positives

high scoring false positives
Results – Person detection

high scoring true positives

high scoring false positives
(not enough overlap)
Results – horse detection

high scoring true positives

high scoring false positives
DPM - discussion

• Approach
  – Manually selected set of parts - Specific detector trained for each part
  – Spatial model trained on part activations
  – Evaluate joint likelihood of part activations

• Advantages
  – Parts have intuitive meaning.
  – Standard detection approaches can be used for each part.
  – Works well for specific categories.

• Disadvantages
  – Parts need to be selected manually
  – Semantically motivated parts sometimes don’t have a simple appearance distribution
  – No guarantee that some important part hasn’t been missed

• When switching to another category, the model has to be rebuilt from scratch.
Extensions - From star shaped model to constellation model

“Star” shape model

- e.g. ISM (Implicit Shape Model)
- Parts mutually independent
- Recognition complexity: $O(N^P)$
- Method: Generalized Hough Transform

Fully connected shape model

- e.g. Constellation Model
- Parts fully connected
- Recognition complexity: $O(N^P)$
- Method: Exhaustive search
What we have learned today

• Object detection
  – Task and evaluation
• A simple detector
• Deformable parts model