CPSC 425: Computer Vision

Instructor: Jim Little
little@cs.ubc.ca

Department of Computer Science
University of British Columbia

Lecture Notes 2016/2017 Term 2
Menu March 30, 2017

Topics:
Object Detection (cont.)
   — Deformable part model
   — Object proposals
Convolutional Neural Networks (preview)

Reading:
Today: Forsyth & Ponce (2nd ed.) 17.1–17.2

Reminders:
Assignment 7 due Tuesday, April 5
www: http://www.cs.ubc.ca/~little/cpsc425/
Today’s Fun Example
Google DeepMind’s AlphaGo

Lecture 21: Re-cap

- **Sliding window** strategy: Train an image classifier. ‘Slide’ a detection window across the image and evaluate the classifier on each window.

- Viola and Jones’ face detector is a classic example
  - An integral image enables constant-time summing of a rectangular region
  - A classifier cascade tries to quickly rule out negative windows using fast classifiers early on

- A deformable part model consists of
  - root: coarse model that gives overall appearance of the object
  - parts: object components that have reliable appearance but might appear at somewhat different locations for different instances
Each part has an appearance model and a natural location relative to the root

Finding a window that looks a lot like the part close to that part's natural location relative to the root yields evidence that the object is present
Deformable Part Model

A parts model for a bicycle, containing a root and 6 parts

Figure source: Felzenszwalb et al., 2010
The learned root model is a set of linear weights $\beta^{(r)}$ applied to the feature descriptor of the root window.

The $i$th learned part model consists of:
- a set of linear weights $\beta^{(p_i)}$ applied to the feature descriptor of the part window
- a natural location (offset) relative to the root $v^{(p_i)} = (u^{(p_i)}, v^{(p_i)})$
- a set of distance weights $d^{(p_i)} = (d_1^{(p_i)}, d_2^{(p_i)}, d_3^{(p_i)}, d_4^{(p_i)})$

Figure source: Felzenszwalb et al., 2010
The overall score of the deformable parts model at a particular window will be the sum of several scores

— A root score compares the root to the window

— Each part has its own score, consisting of an appearance score and a location score

\[
\text{Model score} = \text{Root score} + \sum_{i} \text{Part } i \text{ score}
\]  

(1)
Sliding Window with Deformable Part Model

Denote by $\phi(x, y)$ the feature descriptor of a part window at offset $(x, y)$ relative to the root.

Denote by $(dx, dy) = (u^{(pi)}, v^{(pi)}) - (x, y)$ the difference from the part’s natural offset relative to the root.

The score for part $i$ at offset $(x, y)$ is given by

$$S^{(pi)}(x, y; \beta^{(pi)}, d^{(pi)}, v^{(pi)}) = \beta^{(pi)} \phi(x, y) - (d_1^{(pi)} dx + d_2^{(pi)} dy + d_3^{(pi)} dx^2 + d_4^{(pi)} dy^2)$$
Sliding Window with Deformable Part Model

Denote by $\phi(x, y)$ the feature descriptor of a part window at offset $(x, y)$ relative to the root.

Denote by $(dx, dy) = (u^{(p_i)}, v^{(p_i)}) - (x, y)$ the difference from the part’s natural offset relative to the root.

The score for part $i$ at offset $(x, y)$ is given by

$$S^{(p_i)}(x, y; \beta^{(p_i)}, d^{(p_i)}, v^{(p_i)}) = \beta^{(p_i)} \phi(x, y)$$

$$- (d_1^{(p_i)} dx + d_2^{(p_i)} dy + d_3^{(p_i)} dx^2 + d_4^{(p_i)} dy^2)$$

The final part $i$ score is the best score found over all possible offsets $(x, y)$

$$\text{Part } i \text{ score} = \max_{(x, y)} S^{(p_i)}(x, y; \beta^{(p_i)}, d^{(p_i)}, v^{(p_i)})$$
Learning a Deformable Part Model

- Learning the model can be tricky. Why?
Learning a Deformable Part Model

- Learning the model can be tricky. Why?

- A class model can consist of multiple component models representing different canonical views — e.g. a front and lateral model of a bicycle

- We do not know which component model should respond to which training example
Learning a Deformable Part Model

- Learning the model can be tricky. Why?

- A class model can consist of multiple component models representing different canonical views — e.g. a front and lateral model of a bicycle

- We do not know which component model should respond to which training example

- We also do not know the locations of the parts in the training examples
Learning a Deformable Part Model

However, notice that if the component and the part locations for each training example are given (fixed), we can simply train a linear SVM as usual.
Learning a Deformable Part Model

However, notice that if the component and the part locations for each training example are given (fixed), we can simply train a linear SVM as usual.

This observation leads to the following iterative strategy:

- Assume components and part locations are given (fixed). Compute appearance and offset models.

- Assume appearance and offset models are given (fixed). Re-estimate components and part locations.
Deformable Part Model: Hard Negative Mining

- Sliding window detectors must search over an immense number of windows
  - Even a small false positive rate becomes noticeable

- As a result, we want to train on as many negative examples as possible, but remain computationally feasible

- Hard negative mining: As we train the classifier, apply it to the negative examples (e.g. ‘not a bicycle’) and keep track of ones that get a strong response (e.g. are mistakenly detected as bicycles). Include these in the next round of training.
Deformable Part Model: Examples

A learned car model

Figure source: Felzenszwalb et al., 2010
Deformable Part Model: Examples

A learned cat model

Figure source: Felzenszwalb et al., 2010
Recall: Sliding Window

- Train a window classifier. ‘Slide’ a detection window across the image and evaluate the classifier on each window.

![Image](image_url)

Image credit: KITTI Vision Benchmark

- This is a lot of possible windows! And most will not contain the object we are looking for.
Object Proposals

- **Object proposal** algorithms generate a short list of regions that have generic object-like properties

  — These regions are likely to contain *some* kind of foreground object instead of background texture

- The object detector then considers these candidate regions only, instead of exhaustive sliding window search
Object Proposals

First introduced by Alexe et al., who asked ‘what is an object?’ and defined an ‘objectness’ score based on several visual cues.

Figure credit: Alexe et al., 2012
Object Proposals

- First introduced by Alexe et al., who asked ‘what is an object?’ and defined an ‘objectness’ score based on several visual cues.

This work argued that objects typically — are unique within the image and stand out as salient — have a contrasting appearance from surroundings and/or — have a well-defined closed boundary in space.
Object Proposals

- Multiscale Saliency
  - Favours regions with a unique appearance within the image

Figure credit: Alexe et al., 2012
Object Proposals

- Colour Contrast
  - Favours regions with a contrasting colour appearance from immediate surroundings

Figure credit: Alexe et al., 2012
Object Proposals

- Superpixels Straddling
  - Favours regions with a well-defined closed boundary
  - Measures the extent to which superpixels (obtained by image segmentation) contain pixels both inside and outside of the window

Figure credit: Alexe et al., 2012
Object Proposals

- Superpixels Straddling
  - Favours regions with a well-defined closed boundary
  - Measures the extent to which superpixels (obtained by image segmentation) contain pixels both inside and outside of the window

Figure credit: Alexe et al., 2012
Object Proposals

TABLE 2: For each detector [11, 18, 33] we report its performance (left column) and that of our algorithm 1 using the same window scoring function (right column). We show the average number of windows evaluated per image \#\text{win} and the detection performance as the mean average precision (mAP) over all 20 classes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>0.186</td>
<td>0.268</td>
<td>0.127</td>
</tr>
<tr>
<td>#\text{win}</td>
<td>79945</td>
<td>18562</td>
<td>183501</td>
</tr>
<tr>
<td></td>
<td>0.162</td>
<td>0.225</td>
<td>0.125</td>
</tr>
<tr>
<td></td>
<td>1349</td>
<td>1358</td>
<td>2997</td>
</tr>
</tbody>
</table>

Table credit: Alexe et al., 2012

Object Proposals: Other Strategies (Aside)

1. Hierarchical segmentation

- Generate object proposals using multiple hierarchical segmentations. Perform segmentations in colour spaces with different sensitivities to shadows, shading, and highlights.

Figure credit: K.E.A. van de Sande
2. Seed growing

- Solve several figure-ground separation problems from different initial seeds

Figure credit: Carreira and Sminchisescu, 2012
3. Edges and image gradients

- Score windows based on the number and strength of edges contained within the window

Figure credit: Zitnick and Dollar, 2014
Recent years have seen increasing interest in applying neural network (‘deep learning’) techniques to solve visual recognition problems.

The basic unit of computation in a neural network is a neuron. A neuron accepts some number of input signals, computes their weighted sum, and applies an **activation function** (or non-linearity) to the sum. Common activation functions include sigmoid and rectified linear unit (ReLU).
Sigmoid

\[ f(x) = \frac{1}{1 + e^{-x}} \]

- Common in many early neural networks
- Biological analogy to saturated firing rate of neurons
- Maps the input to the range [0,1]

Figure credit: Fei-Fei and Karpathy
ReLU

\[ f(x) = \max(0, x) \]

- Found to accelerate convergence during learning
- Used in the most recent neural networks

Figure credit: Fei-Fei and Karpathy
Neural Networks

- A neural network comprises neurons connected in an acyclic graph
- The outputs of neurons can become inputs to other neurons
- Neural networks typically contain multiple layers of neurons

Example of a neural network with three inputs, a single hidden layer of four neurons, and an output layer of two neurons

Figure credit: Fei-Fei and Karpathy
Neural Networks

- Training a neural network requires estimating a large number of parameters
- Modern convolutional neural networks contain 10-20 layers and on the order of 100 million parameters
Detection scores in the deformable part model are based on both appearance and location.

The deformable part model is trained iteratively by alternating the steps:

1. Assume components and part locations given; compute appearance and offset models.
2. Assume appearance and offset models given; compute components and part locations.

An object proposal algorithm generates a short list of regions with generic object-like properties that can be evaluated by an object detector in place of an exhaustive sliding window search.