Content of the Lecture

• Single-Object Tracking
  – *Background modeling*
  – Template based tracking
  – *Color based tracking*
  – *Contour based tracking*
  – Tracking by online classification
  – **Tracking-by-detection**

• Bayesian Filtering

• Multi-Object Tracking

• Visual Odometry

• Visual SLAM & 3D Reconstruction
Recap: Tracking as Online Classification

- Tracking as binary classification problem

object vs. background

Image source: Disney/Pixar

Slide credit: Helmut Grabner

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler
Recap: Tracking as Online Classification

• Tracking as binary classification problem

– Handle object and background changes by online updating

Image source: Disney/Pixar

• Main idea
  – Iteratively select an ensemble of classifiers
  – Reweight misclassified training examples after each iteration to focus training on difficult cases.

• Components
  – $h_m(x)$: “weak” or base classifier
    - Condition: <50% training error over any distribution
  – $H(x)$: “strong” or final classifier

• AdaBoost:
  – Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:
    \[
    H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)
    \]
Recap: AdaBoost – Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \ldots, N$.

2. For $m = 1, \ldots, M$ iterations
   a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function
      \[
      J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)
      \]
   b) Estimate the weighted error of this classifier on $X$:
      \[
      \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}
      \]
   c) Calculate a weighting coefficient for $h_m(x)$:
      \[
      \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}
      \]
   d) Update the weighting coefficients:
      \[
      w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(h_m(x_n) \neq t_n)\}
      \]
From Offline to Online Boosting

• Main issue
  – Computing the weight distribution for the samples.
  – We do not know a priori the difficulty of a sample!
    (Could already have seen the same sample before...)

• Idea of Online Boosting
  – Estimate the importance of a sample by propagating it through
    a set of weak classifiers.
  – This can be thought of as modeling the information gain w.r.t. the first $n$
    classifiers and code it by the importance weight $\lambda$ for the $n+1$
    classifier.
  – Proven [Oza]: Given the same training set, Online Boosting converges
    to the same weak classifiers as Offline Boosting in the limit of $N \rightarrow \infty$
    iterations.

N. Oza and S. Russell. Online Bagging and Boosting.
Recap: From Offline to Online Boosting

---

**off-line**

**Given:**
- set of labeled training samples
  \[ \mathcal{X} = \{ \langle x_1, y_1 \rangle, \ldots, \langle x_L, y_L \rangle \mid y_i \pm 1 \} \]
- weight distribution over them
  \[ D_0 = 1/L \]

for \( n = 1 \) to \( N \)
- train a weak classifier using samples and weight dist.
  \[ h_{weak}^n(x) = \mathcal{L}(\mathcal{X}, D_{n-1}) \]
- calculate error \( e_n \)
- calculate weight \( \alpha_n = f(e_n) \)
- update weight dist. \( D_n \)

next

\[ h_{strong}(x) = \text{sign}\left( \sum_{n=1}^{N} \alpha_n \cdot h_{weak}^n(x) \right) \]

---

**on-line**

**Given:**
- ONE labeled training sample
  \[ \langle x, y \rangle \mid y \pm 1 \]
- strong classifier to update
- initial importance \( \lambda = 1 \)

for \( n = 1 \) to \( N \)
- update the weak classifier using samples and importance
  \[ h_{weak}^n(x) = \mathcal{L}(h_{weak}^{n-1}, \langle x, y \rangle, \lambda) \]
- update error estimation \( \tilde{e}_n \)
- update weight \( \alpha_n = f(\tilde{e}_n) \)
- update importance weight \( \lambda \)

next

\[ h_{strong}(x) = \text{sign}\left( \sum_{n=1}^{N} \alpha_n \cdot h_{weak}^n(x) \right) \]
Recap: Online Boosting for Feature Selection

• Introducing “Selector”
  – Selects one feature from its local feature pool

$$\mathcal{H}^{weak} = \{ h_1^{weak}, \ldots, h_M^{weak} \}$$

$$\mathcal{F} = \{ f_1, \ldots, f_M \}$$

$$h^{sel}(x) = h_m^{weak}(x)$$

$$m = \arg \min_{i} e_i$$

On-line boosting is performed on the Selectors and not on the weak classifiers directly.

Recap: Direct Feature Selection

K one training sample

\[ \text{hSelector}_1 \]

\[ \text{estimate errors} \]

\[ \text{select best weak classifier} \]

\[ \text{initial importance } \lambda = 1 \]

\[ \alpha_1 \]

repeat for each training sample

\[ \text{update weight} \]

\[ h_i \]

\[ h_k \]

\[ h_m \]

\[ h_M \]

global weak classifier pool

\[ \text{estimate errors} \]

\[ \text{select best weak classifier} \]

\[ \lambda \]

\[ \alpha_2 \]

\[ \alpha_N \]

\[ \text{update weight} \]

\[ \text{current strong classifier } h_{\text{Strong}} \]

Slide credit: Helmut Grabner
Recap: Tracking by Online Classification

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Helmut Grabner
Image source: Disney/Pixar
When Does It Fail...
Why Does It Fail?

from time $t$ to $t+1$

<table>
<thead>
<tr>
<th>Update classifier (tracker)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluate classifier on sub-patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search region</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyze map and set new object position</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Create confidence map</th>
</tr>
</thead>
</table>

Actual object position

Update classifier (tracker)

Create confidence map

Evaluate classifier on sub-patches

Search region

Analyze map and set new object position

from time $t$ to $t+1$
Why Does It Fail?

- Actual object position
- Search region
- Evaluate classifier on sub-patches

From time \( t \) to \( t+1 \)

- Update classifier (tracker)
- Analyze map and set new object position

Self-learning

Create confidence map

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Helmut Grabner
Image source: Disney/Pixar
Drifting Due to Self-Learning Policy

⇒ Not only does it drift, it also remains confident about it!
Self-Learning and Drift

- Drift
  - Major problem in all adaptive or self-learning trackers.
  - Difficulty: distinguish “allowed” appearance change due to lighting or viewpoint variation from “unwanted” appearance change due to drifting.

  - Cannot be decided based on the tracker confidence!
    - Since the confidence is always dependent on the learned model
    - Model may already be affected by drift when the confidence is measured.

  - Several approaches have been proposed to address this.
Strategy 1: Match Against Initialization

- Used mostly in low-level trackers (e.g., KLT)
  - Advantage: robustly catches drift
  - Disadvantage: cannot follow appearance changes

Strategy 2: Semi-Supervised Learning

Object Detector  Our approach  Object Tracker

Fixed Training set  Fixed Prior for updating an  On-line update
General object detector  Adaptive on-line classifier  Object vs. Background

Prior

Labeled data
Un-labeled data

Strategy 3: Using Additional Cues

- **Tracking-Learning-Detection**
  - Combination of KLT and Tracking-by-Detection
  - Use a KLT tracker as additional cue to generate confident (positive and negative) training examples.
  - Learn an object detector on the fly using Online Random Ferns.

TLD Results
Accumulated Training Examples
Can we use generic object detection to track people?
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Main ideas

- Apply a generic object detector to find objects of a certain class
- Based on the detections, extract object appearance models
  - Even possible to derive figure-ground segmentations from detection results
- Link detections into trajectories
Tracking-by-Detection in 3D

Object detections

Simple f/g model: E.g., elliptical region in detection box

3D Camera path estimation

Spacetime trajectories

Main Issue: Data Association
(We’ll come to that later…)

[Leibe, Cornelis, Schindler, Van Gool, PAMI'08]
Spacetime Trajectory Analysis

Pedestrian detection

Car detections

Own vehicle

[Leibe, Cornelis, Schindler, Van Gool, CVPR'07]
Elements of Tracking

- **Detection**
  - *Where are candidate objects?*

- **Data association**
  - *Which detection corresponds to which object?*

- **Prediction**
  - *Where will the tracked object be in the next time step?*

---

**Today’s topic**
Recap: Sliding-Window Object Detection

- Basic component: a binary classifier

Car/non-car Classifier

No

Yes, car.

Yes, not car.
Recap: Sliding-Window Object Detection

• If object may be in a cluttered scene, slide a window around looking for it.

• Essentially, this is a brute-force approach with many local decisions.
What is a Sliding Window Approach?

- Search over space and scale

- Detection as subwindow classification problem

- “In the absence of a more intelligent strategy, any global image classification approach can be converted into a localization approach by using a sliding-window search.”
Recap: Non-Maximum Suppression

After multi-scale dense scan

Clip detection score

Map each detection to 3D [x, y, scale] space

Apply robust mode detection, e.g., mean shift

Non-maximum suppression

Image source: Navneet Dalal, PhD Thesis
Recap: Sliding-Window Object Detection

- Fleshing out this pipeline a bit more, we need to:
  1. Obtain training data
  2. Define features
  3. Define classifier

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Kristen Grauman
Object Detector Design

• In practice, the classifier often determines the design.
  – Types of features
  – Speedup strategies

• Today, we’ll look at 3 state-of-the-art detector designs
  – Based on SVMs
  – Based on Boosting
  – Based on CNNs
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations

Object/Non-object

Linear SVM

Collect HOGs over detection window

Contrast normalize over overlapping spatial cells

Weighted vote in spatial & orientation cells

Compute gradients

Gamma compression

Image Window

Slide adapted from Navneet Dalal
Recap: Support Vector Machine (SVM)

- **Basic idea**
  - The SVM tries to find a classifier which maximizes the **margin** between pos. and neg. data points.
  - Up to now: consider linear classifiers
    \[
    \mathbf{w}^T \mathbf{x} + b = 0
    \]

- **Formulation as a convex optimization problem**
  - Find the hyperplane satisfying
    \[
    \arg\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2
    \]
    under the constraints
    \[
    t_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n
    \]
    based on training data points \( \mathbf{x}_n \) and target values \( t_n \in \{-1, 1\} \)
Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template

\[ y(x) = w^T x + b \]

HOG feature map

Template

Detector response map

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005
Pedestrian detection with HoGs & SVMs

Incorporating Ground Plane Constraints

- Efficient integration into detector design (**groundHOG**)
  - Idea: only evaluate geometrically valid detection windows
  - Derivation: Region of interest lies between two parabolas…
  - …that can in most cases be approximated by straight lines.
  ⇒ *Only touch pixels inside the ROI for all computations.*
  ⇒ Factor 2-4 speed improvement *on top of all other optimizations*
Real-Time Pedestrian Detection

- Efficient CUDA HOG implementation (equivalent to original HOG code)
- Code made **publicly available as open source under GPL**
- Run-time comparison:

<table>
<thead>
<tr>
<th>run-time</th>
<th>1280 (\times) 960</th>
<th>640 (\times) 480</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cuda</td>
<td>ground</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cuda</td>
</tr>
<tr>
<td>Laptop GTX 285M</td>
<td>1.6 fps</td>
<td>9.6 fps</td>
</tr>
<tr>
<td>Desktop GTX 280</td>
<td>5.5 fps</td>
<td>17.2 fps</td>
</tr>
<tr>
<td>Desktop GTX 580</td>
<td>9.8 fps</td>
<td>27.8 fps</td>
</tr>
</tbody>
</table>

⇒ *Detection at video frame rate possible even on laptops with mobile GPUs!*
You Can Try It At Home…

• groundHOG GPU detector code publicly available
  – Highly optimized for speed
  – Can be used with or without ground plane constraints
  – Supports general ROI processing
  – Supports multi-class detection with feature sharing
  – Published under GPL license (other licensing negotiable)

  – http://www.vision.rwth-aachen.de/projects/groundhog

P. Sudowe, B. Leibe, Efficient Use of Geometric Constraints for Sliding Window Object Detection in Video, ICVS 2011
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Part-Based Models

• Pictorial Structures model
  – [Fischler & Elschlager 1973]

• Model has two components
  – Parts
    (2D image fragments)
  – Structure
    (configuration of parts)

• Use in **Deformable Part-based Model (DPM)**
  – Parts \(\equiv\) 5-7 semantically meaningful parts
  – Probabilistic model enabling efficient inference
Starting Point: HOG Sliding-Window Detector

- Array of weights for features in window of HOG pyramid
- Score is dot product of filter and vector

$$\phi(p, H) = \text{concatenation of HOG features from window specified by } p.$$
Deformable Part-based Models

- Mixture of deformable part models (Pictorial Structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

[Felzenszwalb, McAllister, Ramanan, CVPR'08]
2-Component Bicycle Model

Root filters
coarse resolution

Part filters
finer resolution

Deformation models

[ Felzenszwalb, McAllister, Ramanan, CVPR'08 ]
Object Hypothesis

- Multiscale model captures features at two resolutions

Score of object hypothesis is sum of filter scores minus deformation costs

Score of filter: dot product of filter with HOG features underneath it

Image pyramid

HOG feature pyramid

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Score of a Hypothesis

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)
\]

**“data term”**

**“spatial prior”**

filters

deformation parameters

deformations

\[
\text{score}(z) = \beta \cdot \Psi(H, z)
\]

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Recognition Model

- Difference to standard HOG model
  - Hidden variable $z$: vector of part offsets
  - $\Phi(x, z)$: vector of HOG features (from root filter & appropriate part sub-windows) and part offsets
  $\Rightarrow$ Need to optimize over all possible part positions

\[
f_w(x) = w \cdot \Phi(x) \quad f_w(x) = \max_z w \cdot \Phi(x, z)
\]
Results: Persons

- Results (after non-maximum suppression)
  - ~1s to search all scales
Results: Bicycles
Extensions and Detailed Improvements

- More efficient features
  - Very simplified version of HOG

- Latent part (re-)learning
  - Perform several rounds of training, adapting the annotation bboxes

- Multi-aspect detection
  - Mixture model of different aspects to capture different viewpoints of objects

- Bounding box prediction
  - Infer final detection bounding box from detected part locations

- Multi-resolution models

- Cascaded evaluation
You Can Try It At Home…

- Deformable part-based models have been very successful at several recent evaluations.
  ⇒ One of the state-of-the-art approaches in object detection

- Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
  http://www.cs.uchicago.edu/~pff/latent
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Viola-Jones Face Detector

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade (6061 features in final layer)
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]
Recap: Haar Wavelets

“Rectangular” filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images ⇒ Scale features directly for same cost

Value at \( (x,y) \) is sum of pixels above and to the left of \( (x,y) \)

\[
D = 1 + 4 - (2 + 3) \\
= A + (A + B + C + D) - (A + C + A + B) \\
= D
\]
Recap: Cascading Classifiers for Detection

- Even if the filters are fast to compute, each new image has a lot of possible windows to search...

- Idea: Classifier cascade
  - Observation: most image windows are negative and look very different from the searched object class.
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

[Fleuret & Geman, IJCV’01; Rowley et al., PAMI’98; Viola & Jones, CVPR’01]
Viola-Jones Face Detector: Results
You Can Try It At Home…

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements

- C++ implementation available in OpenCV [Lienhart, 2002]
  - [http://sourceforge.net/projects/opencvlibrary/](http://sourceforge.net/projects/opencvlibrary/)

- Matlab wrappers for OpenCV code available, e.g. here

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Integral Channel Features

- Generalization of Haar Wavelet idea from Viola-Jones
  - Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  - Still efficiently represented as integral images.

Integral Channel Features

- Generalize also block computation
  - 1\textsuperscript{st} order features:
    - Sum of pixels in rectangular region.
  - 2\textsuperscript{nd}-order features:
    - Haar-like difference of sum-over-blocks
  - Generalized Haar:
    - More complex combinations of weighted rectangles
  - Histograms
    - Computed by evaluating local sums on quantized images.
Results: Integral Channel Features

- **fastHOG**
  - ~10 Hz on GPU
  - [Prisacariu 2009]

- **DPM**
  - [Felzenszwalb 2008]

- **ChnFtrs/FPDW**
  - ~5 Hz on CPU
  - [Dollar 2009+2010]

---

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Rodrigo Benenson
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Performance Comparison of Detectors

INRIA dataset

- Shapelet-orig (90.5%)
- PoseInVrSvm (68.6%)
- VJ-OpenCv (53.0%)
- PoseInV (51.4%)
- Shapelet (50.4%)
- VJ (47.5%)
- FtrMine (34.0%)
- Pls (23.4%)
- HOG (23.1%)
- HikSvm (21.9%)
- LatSvm-V1 (17.5%)
- MultiFtr (15.6%)
- MultiFtr+CSS (10.9%)
- LatSvm-V2 (9.9%)
- FPDW (9.3%)
- ChnFtrs (8.7%)

false positives per image vs. miss rate
Performance Comparison of Detectors

VeryFast 50 Hz
Issues for Efficient Detection

- One template cannot detect at multiple scales...
Issues for Efficient Detection

- Typically, features are computed many times

~50 scales
Issues for Efficient Detection

- Typically, features are computed many times

~50 scales
VeryFast Detector

• **Idea 1**: Invert the relation

Practical Considerations

• Training and running 1 model/scale is too expensive
VeryFast Detector

• Idea 2: Reduce training time by feature interpolation

5 models, 1 image scale ≈ 50 models, 1 image scale

• Shown to be possible for Integral Channel features
VeryFast Detector

- Effect: Transfer test time computation to training time

\[ \Rightarrow \text{Result: } 3x \text{ reduction in feature computation} \]
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + \]

Slide credit: Rodrigo Benenson
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + w_2 \cdot h_2 \]
VeryFast: Classifier Construction

• Ensemble of short trees, learned by AdaBoost

\[ score = w_1 \cdot h_1 + w_2 \cdot h_2 + \ldots + w_N \cdot h_N \]
Learned Models

Integral Channel features

Models

Slide credit: Rodrigo Benenson
• Detection without resizing improves quality of results
Multi-Scale Models > Single-Scale Model
Comparison to State-of-the-Art

**INRIA dataset**

**ETH dataset**

- **Extension:** Roerei detector
  - Detailed evaluation of design space
  - Non-regular pooling regions found to work best.

---

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking  
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide adapted from Rodrigo Benenson
Roerei Results

Applications: Mobile Robot Navigation

link to the video
Topics of This Lecture

• Tracking by Detection
  – Motivation
  – Recap: Object detection

• SVM based Detectors
  – Recap: HOG
  – DPM

• AdaBoost based Detectors
  – Recap: Viola-Jones
  – Integral Channel features
  – VeryFast/Roerei

• CNN-based Detectors
  – Recap: CNNs
  – R-CNN
Recap: Convolutional Neural Networks

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Recap: Intuition of CNNs

• Convolutional net
  – Share the same parameters across different locations
  – Convolutions with learned kernels

• Learn *multiple* filters
  – E.g. $1000 \times 1000$ image
    100 filters
    $10 \times 10$ filter size
  ⇒ only 10k parameters

• Result: Response map
  – size: $1000 \times 1000 \times 100$
  – Only memory, not params!
Recap: Convolution Layers

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single \([1 \times 1 \times \text{depth}]\) depth column in output volume.
Recap: Activation Maps

Activation maps

one filter = one depth slice (or activation map)

5×5 filters
Recap: Pooling Layers

- **Effect:**
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations
R-CNN Detector

**R-CNN: Regions with CNN features**

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

- Results on PASCAL VOC Detection benchmark
  - Pre-CNN state of the art: 35.1% mAP [Uijlings et al., 2013]
  - 33.4% mAP DPM
  - R-CNN: 53.7% mAP

You Can Try All of This At Home…

• Detector code is publicly available
  
  ➢ HOG:  
    – Dalal’s original implementation:  
      http://www.navneetdalal.com/software/  
    – Our CUDA-optimized groundHOG code (>80 fps on GTX 580)  
      http://www.mmp.rwth-aachen.de/projects/groundhog
  
  ➢ DPM:  
    – Felzenswalb’s original implementation:  
      http://www.cs.uchicago.edu/~pff/latent
  
  ➢ VeryFast  
    – Benenson’s original implementation:  
      https://bitbucket.org/rodrigob/doppia/
  
  ➢ R-CNN  
    – Girshick’s original implementation:  
      https://github.com/rbgirshick/rcnn