Object Recognition, Detection, Categorization

Archeology?

A single Goal with Variations
Find evidence for familiar objects in a given image
- Recognition - Find evidence ... specific object(s) that you have information about (Model, Library)
- Categorization - Find evidence to objects belonging to specific classes. Say which.
  - Detection - Find evidence ... some object from a specific class. Say where it is.
  - Classification - Find evidence ... some object from a specific class. No need to say where.
- Semantic segmentation - Find all objects in the image and their location. Classify every pixel.
Image Variability

The image depends on many factors
- Object category
- Objects’ specific shape (within category), position vs. camera, color, reflection properties, occlusions
- Illumination sources’ location, light distribution, color

A solution based on synthesis

- Suppose we know how to synthesize an image
  \[ I(\text{Shapes, positions, illumination, ...}) \]
- Then, we can try to solve the inverse problem
  \[ \arg \min_{\text{Shape, position, camera param.,...}} \| f_{\text{given}} - I(\text{Shape, position, camera param.,...}) \| \]

- **Difficulties**
  - Models (shape, color) & imaging process are complex, unknown
  - The scene (and image) contains much irrelevant data, and solving for all and for unimportant variables is difficult
  - The inverse problem is multivalued & hard (non-convex)

- **Not done fully in practice**
**Simple principles/approaches/tricks that help**

**Principle I:** If possible, make the problem easier
- Control relative position and illumination

- Limit variability
  - OCR with known font
  - Train for your own handwriting
  - Ask the person to look directly at the camera

**Simple principle II:**

**Selection:** solve separately for image parts
- Moving window
- Attention
- Segmentation
- Proposals generation
• **Principle III:**
  • Use **Effective Representation** for image comparison:
    - Informative for discrimination
    - Insensitive\Invariant to irrelevant parameters. (illumination, pose, compression effects, ...)
    - Efficient

• Simple representations:
  - Intensities, normalized intensities, PCA, Edges, Gradient directions, Histograms, Bag of words (e.g. textons)

• Modern representations:
  - Features learned specifically for the task.

---

**A sample of typical approaches**

• Comparing intensities using subspaces
• Recognizing binary objects using tailored features
• Compensating for pose change and verifying
• Using parts and relative positions
• Using parts with no relative positions
• Convolutional Neural Networks
How to handle image variability?

**Answer 1:** Approximately enumerate all possibilities

---

**Algorithm 1: Intensity subspaces**

Basic approach (template matching, correlation):
- **Model:** Represent object by its intensity image (size N)
- **Algorithm:** (Nearest Neighbor)
  - for every candidate (subimage), find SSD(model, subimage)
  - Decide “detection” is the SSD is small enough

**Pros:** Simplest and relatively fast $O(SN)$
**Cons:** sensitive to pose, illumination ...

Improvement: Use multiple images representing different pose, illumination
**Con:** Needs more SSDs, $O(MN)$,
**Improvement:** represent model and image compactly - PCA.
Algorithm 1: The subspace idea

- Idea: Use efficient image representation, \( N = \#\text{pixels} \)
  \[
  I_i(x, y) \equiv \phi_0(x, y) + \sum_{j=1}^{N' < N} c_j \phi_j(x, y)
  \]
  \[
  I_i(x, y) \sim \{c_{i1}, \ldots, c_{iN'}\}
  \]
  For many such collections, \( N' \ll N, \|\{I_i(x, y)\}\| \)

Why?

- Reason 1: Images are Smooth & similar
- Reason 2:
  - All images of Lambertian object under different illumination (no shadows) are in a 3D subspace. Proof: done in lecture \( \sim 3 \)
  - With attached shadows, the effective dimension is small.

Which subspace?

Principle components analysis (PCA)

- Approximation of a set of vectors \( \{v_i\}_{i=1}^m \)
  \[
  v_i \approx \hat{v}_i = \phi_0 + \sum_{j=1}^{N' < N} c_j \phi_j
  \]

- Let \( \{\phi_j\} \) be orthonormal.
- Approximation error
  \[
  \|v_i - \hat{v}_i\|^2 = \left\|\sum_{j=N'+1}^{N} c_j \phi_j\right\|^2 = \sum_{j=N'+1}^{N} |c_j|^2
  \]
- Use \( \phi_0 = \overline{v} \) (to get uncorrelated coefficients)
- Minimizing error:
  \[
  \arg \min_{\{\phi_j\}} \sum_{j=N'+1}^{N} |c_j|^2 = \arg \max_{\{\phi_j\}} \sum_{j=1}^{N'} |c_j|^2 = \sum_{j=1}^{N'} \left(\phi_j^T (v_i - \phi_0)\right)^2
  \]
Principle components analysis (cont.)

Optimizing the average error for all images in \( \{ v_i \} \).

\[
\arg \max_{\{ \phi_j \}} \left( \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{N'} |c_{ij}|^2 \right) = \sum_{j=1}^{N'} (\phi_j)^T \frac{1}{m} (v - \bar{v})(v - \bar{v})^T (\phi_j) = \sum_{j=1}^{N'} (\phi_j)^T \Sigma (\phi_j) \quad ; \quad (\phi_j)^T (\phi_j) = 1
\]

For \( N' = 1 \), using Lagrange multipliers,

\[
\max_{\{ \phi_j \}} (\phi_j)^T \Sigma (\phi_j) - \alpha \left( (\phi_j)^T (\phi_j) - 1 \right)
\]

\( \Sigma \phi_j - \alpha \phi_1 = 0 \)

\( \phi_1 \) is an e.vector of \( \Sigma \). Which one? one with largest e.value.

For next one... \( \max \sum_{j=2}^{N'} (\phi_j)^T \Sigma (\phi_j) - \alpha \left( (\phi_j)^T (\phi_j) - 1 \right) - \beta \left( (\phi_2)^T (\phi_1) \right) \)

All the \( \{ \phi_j \} \), denoted “principle components” are e.vectors of \( \Sigma \) associated with largest e.values.

Principle components analysis (cont.)

- The subspace spanned by the \( N' \) components is often a good characterization of the set \( \{ v_i \} \).
- The PCA is the most common dimension reduction tool
- Useful in many domains
  - Compression
  - Search

- If \( \Sigma \) is exact:
  - PCA = Karhunen Loeve transform,
  - coefficients \( \{ c_{ij} \} \) are uncorrelated
- Other subspaces: LDA, DCT
Recognition with PCA

PCA example: Eigenfaces

- **Algorithm a: common subspace**
  - Preprocessing: get model examples \( \{ v_i \} \), find one PCA subspace for all models
  - Represent every model images by PCA coefficients \( \{ c_{ij} \} \)
  - For every sub-image (k-th) find PCA coefficients \( \{ d_{ij} \} \) and find nearest model using approximate SSD
    \[
    SSD_{ik} \approx \sum_j^N (c_{ij} - d_{ij})^2
    \]

- **Algorithm b: Different subspaces**
  - Preprocessing: build different PCA subspace for every model
  - Represent the sub-images \( \{ d_{ij} \} \) in all subspaces
  - Choose the subspace that gives best approximation
Inside the subspace

- **Limitations:** can't handle occlusion or variable background (global), not very effective
- **Conclusions:**
  - Images of the same object are in low dimension space/manifold,
  - PCA is old but is very useful (usually not for recognition).

How to handle image variability?

Answer 1: Approximately enumerate all possibilities
Answer 2: Use Invariants

(Pose) invariant – a property that is measured in an image and does not depend on pose.
Alg 2: Recognizing binarized objects using features/invariants

- **Context:** recognizing binarized objects
- **Evaluate basic features:**
  - Tailored: perimeter, area, # holes, # concavities.
  - Generic: e.g. moments
    \[ m_{pq} = \sum_{(x,y)} x^p y^q I(x,y) \]
  - All can be made invariant to translation, rotation, etc.
- **Global**
  - Use the examples vector to learn (any learning algorithm)
- **Pros:** compact & fast, **Cons:** Need binarization, limited.

Alg 2 (cont): Moment invariants

- **Basic moments**
  \[ m_{pq} = \sum_{(x,y)} x^p y^q I(x,y) \]
  \[ m_{00}, m_{10}, m_{01}, ... \]
- **Invariance to translation**
  \[ \mu = m_{10} / m_{00}, m_{01} / m_{00} \]
  \[ \mu_{pq} = \sum_{(x,y)} (x-x) (y-y) I(x,y) \]
  \[ \mu_{00} = m_{00}, \mu_{10} = \mu_{01} = 0, \mu_{11} = m_{10} - \bar{x} m_{01} \]

- **Invariance to rotation**
  \[ \mu_{ij} = \mu_{ij}' \]
  It is diagonal in the basis of the e.vectors
  \[ \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \]
**More on invariants**

(Pose) invariant - a property that is measured in an image and does not depend on pose.

More general pose changes has more complex invariants
- Euclidean transformation: area, angle
- Affine transformation: length ratio (on same line)
- Perspective Transformation: Cross ratio

**Limitations**
- No invariants for true 3D objects
- Works if change is due to known trans.
- Difficult to measure in general images
- Invariants amplify noise

**How to handle image variability?**

Answer 1: Approximately enumerate all possibilities
Answer 2: Use Invariants
**Answer 3:** hypothesize pose (illumination) and test
**Alg 3: Geometric alignment**

- Construct a model that can be used for synthesizing image related description (e.g. an edge image)

- Recognition - Given an image
  - Use a RANSAC process to generate hypotheses on the transformation between model and image.
  - Verify (evaluate) each hypotheses by image comparison.
    - e.g. grad. direction.

---

**Alg 3: Geometric alignment**

- Past: popular with geometric models
- Modern variations
  - Face recognition applications
  - Iterative alignment of 3D model
  - Needs a model, takes time
  - Captures fine differences
How to handle image variability?

Answer 1: Approximately enumerate all possibilities
Answer 2: Use Invariants
Answer 3: hypothesize pose (illumination) and test
Answer 4: Use learning

Learning – creating classifiers from examples

- Examples – images or descriptions extracted from images (e.g. PCA).
- The examples express object variability due to pose/illumination and due to non-rigid change and change within class

- Recognition is not representation
  - Not everything must be represented
  - Example: PCA is a good representation but PCA based classifier are non-optimal
  - Representing only discriminative information
Alg 4: Histogram of Gradients (HOG+)

Another learning based algorithm.

- Similar to SIFT: In each 8x8 block calculate a histogram of gradient orientations, overlapping blocks, different normalizations, concatenate, use as feature vectors for SVM training.
- Works nice for face, pedestrian detector,

Alg 4: Histogram of Gradients (HOG)

An example:
Part based Algorithms

- A part based alg.
  - Parts of the object are recognized in the image
  - The decision is based on the part recognition scores/locations

- Motivation: Often, the appearance of object’s parts change much less than the appearance of full objects

- How many parts? What kind of parts?
  - A few or many

- Use the spatial relations between parts?
  - No, Yes, Partially, Approximately

Alg. 5: Bag-of-features models

Many slides adapted from Fei-Fei Li, Rob Fergus, Antonio Torralba, and S.Lazebnik
Origin 1: Bag-of-words models


Origin 2: Texture recognition

- Texture is characterized by the repetition of basic elements or textons, not their spatial arrangement, that matters.
- For a large subset ("stochastic") textures, it is the identity of the textons, not their spatial arrangement, that matters.

---

Micha Lindenbaum
Origin 2: Texture recognition

Alg. 5: Bag of features - Basic Algorithm

Training
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images as histogram of “visual words”
5. Train a (multiclass) classifier

Classification: Extract features, quantize them, represent image as distribution, classify.
Bag of features

1. Extract features

Extract small regions
- Regular grid
- Interest point detector
- Random sampling

Represent each window
- Grad direction histograms
  - SIFT etc.
- PCA coefficients

Bag of features

2. Learn Visual vocabulary - Often by K-means

Fei-Fei et al. 2005
Bag of features

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images as histogram of “visual words”
5. Train a (multiclass) classifier
   – Discriminative: e.g. SVM
   – Generative: e.g. Naïve Bayes

...
Performance: confusion matrix (SVM)

<table>
<thead>
<tr>
<th></th>
<th>faces</th>
<th>buildings</th>
<th>trees</th>
<th>cars</th>
<th>phones</th>
<th>bikes</th>
<th>books</th>
</tr>
</thead>
<tbody>
<tr>
<td>faces</td>
<td>98</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>34</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>buildings</td>
<td>1</td>
<td>63</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>trees</td>
<td>1</td>
<td>10</td>
<td>81</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>cars</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>85</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>phones</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>55</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>bikes</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>91</td>
<td>0</td>
</tr>
<tr>
<td>books</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td>error rate</td>
<td>2</td>
<td>27</td>
<td>19</td>
<td>15</td>
<td>45</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>mean rank</td>
<td>1.04</td>
<td>1.77</td>
<td>1.28</td>
<td>1.30</td>
<td>1.83</td>
<td>1.09</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Sparse vs. Dense sampling

Sparse
- Sampling based on key points
- Different interest operator help
- More invariant & compact
- Better for finding specific objects

Dense
- Densely spatial sampling
- patches at all image scales.
- Better for categorization

Jurie and Triggs, Creating Efficient Codebooks for Visual Recognition, ICCV05
Alg. 5': Bag of Features with Spatial considerations

Training
1. Quantize to words
2. For every example, build representation
   1. Divide the ROI into a spatial pyramid
   2. For every word, find histogram for every cell
3. Train some learning algorithm (usually SVM)

Test
1. Build representation
2. Classify

Alg 6: Viola Jones face detection

- A part based alg.
  - Parts of the object are recognized in the image
  - The decision is based on the set of part scores
- The object is described by
  - Appearance - How the part looks like
  - Shape - What is the relative part position
- The VJ alg. is based on the boosting learning process
  + Several computational tricks
- (was) Very fast (15 fps on Pentium3)
- Widely adopted and implemented in openCV, cameras
Alg 6: Viola Jones face detection (cont)

- **Weak classifiers**
  - Each based on a simple feature:
    - brightness difference in a mask
  - (Haar wavelets)
  - Masks are defined in all locations relative to sub-window

- **Faces share appearances that make these feature informative**
  - Dark region around the eyes
  - Bright nose

- 45,396 weak classifier. Which?

Alg 6: Viola Jones face detection (cont)

- **Strong classifier**
  - Trained from positive/negative examples
  - Every feature - fast to evaluate
  - Uses Adaboost to select & combine features

\[ h_{\text{strong}} = \begin{cases} 1 & \text{if } a_1 h_1(x) + \cdots + a_n h_n(x) \geq t \\ 0 & \text{otherwise} \end{cases} \]

- **How many weak classifiers?**
  - More - slow but accurate
  - With 200 weak classifiers
  - Nice ROC, 0.7 sec.

Receiver Operating Curve
Alg 6: Viola Jones face detection (cont)

Making it faster – I

• **Integral image** for faster evaluation of the weak classifiers
  
  - Formal definition: \( I(x,y) = \sum_{x'<x, y'<y} I(x',y') \)
  
  - Preparing the integral image: \( O(N) \)
  
  - Usage: calculating features in constant time
  
  - E.g. \( \sum_{(x,y) \in D} I(x,y) = II(4) + II(1) - II(2) - II(3) \)

Alg 6: Viola Jones face detection (cont)

Making it faster – II

• Observation: in most places, 200 weak classifiers are much more than needed to decide “no”.

• A Cascade of classifiers
  
  - The first uses 2 classifiers. Rejects 40%
  
  - The second uses 5 classifiers. Rejects 60%
  
  ....
  
  - The last ones use 200 classifiers
Alg 6: Viola Jones face detection (cont)

- Experiments
  - Training 4619 (+), 10,000 (-), 24 x 24 images
  - About 92% detection, with 1/image false detection.
  - As accurate as before, and much faster

Alg 7: Discriminatively trained deformable part models

- Combination
  - Tree based spatial model
  - HOG appearance in 2-scales
  - Whole object+Parts

- Location:
  - start model
    - Deformation allowed

- Score = appearance score – deformation cost

- Training by SVM

Felzenszwalb, Mcallester, Ramanan, CVPR 2008
Some results

Convolutional Neural networks
Multilayer perceptron

The main idea:
- Build several perceptrons and use their output as inputs to another set of perceptrons
- Training – adjusting all the weights together so that, given an example’s input $X$, the network’s output is $Y$.
- Training by gradient descent (backpropagation alg.)

What do we optimize?
- Suppose we have $K$ classes
- Then, we build the network to have $K$ outputs $o_k$
- The probability of class (softmax) $P(C_k = 1|X) = \frac{e^{o_k}}{\sum_j^K e^{o_j}}$
- The loss for a single training example
  $$L(X, Y; \theta) = -\sum y_j \log P(C_j|X)$$
- For many training examples we minimize
  $$\theta^* = \arg \min_\theta \sum_{n=1}^N L(X^n, Y^n; \theta) + \lambda \|\theta\|^2$$
A regularizatin term
Fully Connected Layer

Example: 200x200 image
40K hidden units

~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Locally Connected Layer

Example: 200x200 image
40K hidden units

Filter size: 10x10
4M parameters
Assuming image stationarity, we may replicate the weights.

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters
100 parameters!

We can build different features, each with different weights.
Convolution layer

Several components:
1. (Several) convolutions with local filters
2. Each convolution is followed by a nonlinearity
   - ReLU - rectified linear unit - \( f(x) = \max(0,x) \)
3. (Optionally) Values are normalized
4. Pooling - To reduce location sensitivity, and save parameters, resolution is reduced
   - Max value our of 2x2

ConvNets: Typical Stage

One stage (zoom)

- Convol.
- LCN
- Pooling

courtesy of K. Kavukcuoglu

Micha Lindenbaum
The architecture of LeNet5

- LeCun’s net for digit recognition
- Both convolutions and arbitrary (full) connection

Recent changes in favor of CNNs.

Until recently large CNNs were considered inferior
- Too many parameters to train
- Little theory and performance guarantees

Recently however we got ...
- Much more powerful computer and GPUs
- Much larger annotated databases
- Networks for which training works better (ReLU)
- Better training methods (DropOut)

A task that was hard to run before:
- A subset of ImageNet, 1000 classes, 10^6 images.
Spatial and feature resolution

Typically,
- spatial resolution decreases
- Number of features increase
- Note that the convolutions are 3D

Krizhevsky, Sutskever, Hinton, 2012
Results from a data base of 1000 classes, 1000000 images

Why do CNNs work?

The common answer:
- They calculate complex features that are hard to design.
  - The features calculated for one task are often good for others.
- They implement a very complex classifier
- Still a good question

Why do they converge?
- Many minima exist but they all give the same performance.
Why do CNNs work?

Several attempts to understand by visualization:
Which image gives maximal response to features?
Can we invert the network, from feature to image.

Layer 1                      layer 3                        layer 5

Current trends

- Make it deeper (e.g. VGG), add bypass,
- Recurrent networks that model time changes and other related classifications
- Use one network to generate examples for training the other
- A big hammer
CNN: indoor semantic labeling RGBD

Figure 2: Some scene labelings using our Multiscale Convolutional Network trained on RGBD images.

Farabet, 2013

What does a human see in short time?

Some kind of game or fight. Two groups of two men? The foreground pair looked like one was getting a fist in the face. Outdoors seemed like because I have an impression of grass and maybe lines on the grass? That would be why I think perhaps a game, rough game though, more like rugby than football because they pairs weren’t in pads and helmets, though I did get the impression of similar clothing, maybe some trees? in the background. (Subject: SM)

PT = 500ms

PT = 27ms
This was a picture with some dark sploches in it. Yeah. . . that’s about it. (Subject: KM)

PT = 40ms
I think I saw two people on a field. (Subject: RW)

PT = 67ms
Outdoor scene. There were some kind of animals, maybe dogs or horses, in the middle of the picture. It looked like they were running in the middle of a grassy field. (Subject: IV)

PT = 107ms
Two people, whose profile was toward me. looked like they were on a field of some sort and engaged in some sort of sport (their attire suggested soccer, but it looked like there was too much contact for that). (Subject: AI)

Fei-Fei, Iyer, Koch, Perona, JoV, 2007