Object Recognition, Detection, Categorization

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.

In all types learning & classification are the main tool
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
  \[ I(\text{Shapes, positions, illumination, \ldots}) \]
- Then, we can try to solve the inverse problem

\[
\arg \min_{\text{Shape, position, camera param.}} \| I_{\text{given}} - I(\text{Shape, position, camera param.}) \|
\]

- 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

How to find the representation?
- Traditional approach:
  - use general features with good properties

- Modern approach:
  - Features learned together with the classifier.

2D or 3D representations?
- How to represent a 3D object?

  - Option 1: by its image(s)
    - Easy to get
    - Easy to learn
    - Easy to compare for classification

  - Option 2: by the 3D shape
    - Change with pose & illumination in a simpler way
    - Does not require many examples
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 a single intensity image
• **Algorithm**: (Nearest Neighbor)
  - for every candidate (subimage), find SSD(model, subimage)
  - Decide “detection” is the SSD is small enough
Problem: sensitive to pose, illumination ...

Solution: Use multiple images for different pose, illumination
Problem: computationally expensive
Solution: Find a subspace that approximates all images, and represent them using a low dimension coefficient vector.
This is the PCA method or eigenfaces

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**
  \[ m_{pq} = \sum_{(x,y)} x^p y^q I(x,y) \]
  **Meaning:** \( m_{00}, m_{10}, m_{01}, \ldots \)

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

- **Invariance to rotation**
  \[ \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \]
  **Matrix of central moments**
  There is a rotated coordinate system (e.vectors)
  where the matrix diagonal
  \[ \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \]
  \( \rightarrow \) e.values are rotation invariants

How to handle image variability?

**Answer 1:** Approximately enumerate all possibilities
**Answer 2:** Use Invariants
**Answer 3:** hypothesize (pose or 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, occlusion, non-rigid change and due to change within class

- Recognition is not representation
  - Not everything must be represented
  - Example: PCA is the best subspace representation but other subspace (discriminant analysis = Fisher faces) give more accurate recognition.
  - Finding and Representing discriminative information
Alg 4: Histogram of Gradients (HOG)

A simple representation & 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 linear (SVM) training.

Alg 4: Histogram of Gradients (HOG)

An example:

- Works nice for face, pedestrian detector,
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
Origin 1: Bag-of-words models


Origin 2: Texture recognition

- Texture is characterized by the repetition of basic elements or textons
- For a large subset ("stochastic") textures, it is the identity of the textons, not their spatial arrangement, that matters
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 to the vocabulary, represent image as histogram, classify.
Bag of features

1. Extract features

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

Represent each window
- Grad direction histograms
- 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, e.g. SVM

Many questions: sparse or dense, which representation, how to cluster, how many words, ...

An effective generalization: Fisher vectors.
Tells not only the closest word but encodes the location relative to it.
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>

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 example, find histogram for every cell
3. Train a learning algorithm (usually SVM)

Test
1. Build representation
2. Classify
Alg 6: Viola Jones face detection

- **Part based** algorithms.
  - Discriminative parts of the object are selected and 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. face detection 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?
Adaboost starts with a uniform distribution of “weights” over training examples.

- Select the classifier with the lowest weighted error (a “weak” classifier)
- Increase the weights on the training examples that were misclassified.

At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

\[
h_{\text{strong}}(x) = \begin{cases} 
1 & \sum_{i=1}^{n} \alpha_i h_i(x) + \ldots + \alpha_n h_n(x) \geq \frac{1}{2} (\alpha_1 + \ldots + \alpha_n) \\
0 & \text{otherwise}
\end{cases}
\]

Source: Qing Chen, Discover Lab, University of Ottawa

**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

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

- **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: $I(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$

  - 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

• The cascade = a kind of degenerate tree.
Alg 6: Viola Jones face detection (cont)

- Experiments
  - Training 4619 (+), 10,000 (-), 24 x 24 images
  - About 92% detection, with 1 false detection /image.
  - Almost as accurate, 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**

![Images of objects with bounding boxes indicating recognition results.]

**Context**

*Use information outside the object to recognize it*

![Images of segmentation results with labeled objects.]

Figure 8: Images at a resolution of 32x32 pixels and the segmentations provided by the participants. Figure B shows some of the recognized objects cropped. Many of those objects become unrecognizable once they are extracted from the context.
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.)
Multilayer perceptron

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)
  \[
P(C_k = 1 | X) = \frac{e^{o_k}}{\sum_{j} e^{o_j}}
\]
  \((X, Y = (0, \ldots, y_k = 1, \ldots, 0))\)
- 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 regularization 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

Convolution Layer

Assuming image stationarity, we may replicate the weights

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

We can build different features, each with different weights
Convolutional 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 out of 2x2
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.

Architecture for Classification

Total nr. params: 60M  Total nr. flops: 832M

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Spatial and feature resolution

Typically,

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

Results from a data base of 1000 classes, 1,000,000 images

Krizhevsky, Sutskever, Hinton, 2012
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.
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