Recognizing object categories
Kristen Grauman
UT-Austin
Wed Sept 13, 2017

Announcements
• Reminders:
  • Assignment 1 due Sept 22 11:59 pm on Canvas
  • No laptops, phones, tablets, etc. in class
• Thoughts on review sharing?
• Questions about presentations, experiments, discussion proponent/opponent?

Last time: Recognizing instances

• 1. Basics in feature extraction: filtering
• 2. Invariant local features
• 3. Recognizing object instances

Instance recognition: remaining issues
• How to summarize the content of an entire image? And gauge overall similarity?
• How large should the vocabulary be? How to perform quantization efficiently?
• Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Spatial Verification

Both image pairs have many visual words in common.

Kristen Grauman
Spatial Verification

Only some of the matches are mutually consistent

Spatial Verification: two basic strategies

• RANSAC
• Generalized Hough Transform

Outliers affect least squares fit

RANSAC

• RANdom Sample Consensus

• Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

• Intuition: if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.

RANSAC for line fitting

Repeat N times:
• Draw s points uniformly at random
• Fit line to these s points
• Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than t)
• If there are d or more inliers, accept the line and refit using all inliers

Lana Lazebnik
1. Randomly select minimal subset of points

2. Hypothesize a model

3. Compute error function

4. Select points consistent with model
That is an example fitting a **model** (line)... What about fitting a **transformation** (translation, affine...)?
Robust feature-based alignment

- Extract features
- Compute putative matches

RANSAC: General form

- **RANSAC loop:**
  1. Randomly select a seed group of points on which to base transformation estimate
  2. Compute model from seed group
  3. Find inliers to this transformation
  4. If the number of inliers is sufficiently large, re-compute estimate of model on all of the inliers

- Keep the model with the largest number of inliers
RANSAC example: Translation

Putative matches

Source: Rick Szeliski

RANSAC example: Translation

Select one match, count inliers

RANSAC example: Translation

Select one match, count inliers

Find “average” translation vector

RANSAC verification

For matching specific scenes/objects, common to use an affine transformation for spatial verification

Fitting an affine transformation

Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras.

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix} +
\begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]
**RANSAC verification**

**Spatial Verification: two basic strategies**

- **RANSAC**
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., “success” if find an affine transformation with > N inlier correspondences
- **Generalized Hough Transform**
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

**Voting**

- It’s not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.

**Difficulty of line fitting**

**Hough Transform for line fitting**

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- **Hough Transform** is a voting technique that can be used to answer all of these questions.
  - **Main idea:**
    1. Record vote for each possible line on which each edge point lies.
    2. Look for lines that get many votes.
Finding lines in an image: Hough space

Connection between image \((x,y)\) and Hough \((m,b)\) spaces
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points \((x,y)\), find all \((m,b)\) such that \(y = mx + b\)

Finding lines in an image: Hough algorithm

How can we use this to find the most likely parameters \((m,b)\) for the most prominent line in the image space?
- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Voting: Generalized Hough Transform
- If we use scale, rotation, and translation invariant local features, then each feature match gives an alignment hypothesis (for scale, translation, and orientation of model in image).
- A hypothesis generated by a single match may be unreliable,
- So let each match vote for a hypothesis in Hough space
Gen Hough Transform details (Lowe’s system)

- **Training phase**: For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)
- **Test phase**: Let each match between a test SIFT feature and a model feature vote in a 4D Hough space
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension
  - Find all bins with at least three votes and perform geometric verification
    - Estimate least squares affine transformation
    - Search for additional features that agree with the alignment


Example result

Gen Hough vs RANSAC

- **GHT**
  - Single correspondence -> vote for all consistent parameters
  - Represents uncertainty in the model parameter space
  - Linear complexity in number of correspondences and number of voting cells; beyond 4D vote space impractical
  - Can handle high outlier ratio

- **RANSAC**
  - Minimal subset of correspondences to estimate model -> count inliers
  - Represents uncertainty in image space
  - Must search all data points to check for inliers each iteration
  - Scales better to high-d parameter spaces

Video Google System

1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

Sivic & Zisserman, ICCV 2003

- Demo online at: http://www.robots.ox.ac.uk/~vgg/research/vedge/index.html

Recognition via feature matching + spatial verification

**Pros:**
- Effective when we are able to find reliable features within clutter
- Great results for matching specific instances

**Cons:**
- Scaling with number of models
- Spatial verification as post-processing – not seamless, expensive for large-scale problems
- Not suited for category recognition.

Kristen Grauman
Summary: instance recognition

- **Matching local invariant features**
  - Useful not only to provide matches for multi-view geometry, but also to find objects and scenes.
- **Bag of words** representation: quantize feature space to make discrete set of visual words
  - Summarize image by distribution of words
  - Index individual words
- **Inverted index**: pre-compute index to enable faster search at query time
- **[today] Recognition of instances via alignment**: matching local features followed by spatial verification
  - Robust fitting: RANSAC, GHT

Rest of today

- **Intro to categorization problem**
- **Object categorization as discriminative classification**
  a) Boosting + fast face detection example
  b) Nearest neighbors + scene recognition example
  c) Support vector machines + pedestrian detection example
    - Pyramid match kernels, spatial pyramid match
  d) Convolutional neural networks + ImageNet example

What does recognition involve?

Detection: are there people?

Activity: What are they doing?

Object categorization
Instance recognition

- **Potala Palace**
- **A particular sign**

Scene and context categorization

- outdoor
- city
- ...

Attribute recognition

- gray
- flat
- made of fabric
- crowded

Object Categorization

- **Task Description**
  - “Given a small number of training images of a category, recognize a-priori unknown instances of that category and assign the correct category label.”

- Which categories are feasible visually?

Visual Object Categories

- **Basic Level Categories in human categorization** [Rosch 76, Lakoff 87]
  - The highest level at which category members have similar perceived shape
  - The highest level at which a single mental image reflects the entire category
  - The level at which human subjects are usually fastest at identifying category members
  - The first level named and understood by children
  - The highest level at which a person uses similar motor actions for interaction with category members

- **Basic-level categories in humans seem to be defined predominantly visually.**
- There is evidence that humans (usually) start with basic-level categorization before doing identification.
  - How does this transfer to automatic classification algorithms?
How many object categories are there?

~10,000 to 30,000

Source: Fei-Fei Li, Rob Fergus, Antonio Torralba
Biederman 1987

Other Types of Categories
- Functional Categories
  - e.g. chairs = "something you can sit on"

Challenges: robustness
- Illumination
- Object pose
- Clutter
- Occlusions
- Intra-class appearance
- Viewpoint

Challenges: context and human experience
- Context cues
- Function
- Dynamics

Video credit: J. Davis
Challenges: complexity

- Millions of pixels in an image
- 30,000 human recognizable object categories
- 30+ degrees of freedom in the pose of articulated objects (humans)
- Billions of images online
- 300 hours of new video on YouTube per minute
- ...
- About half of the cerebral cortex in primates is devoted to processing visual information [Felleman and van Essen 1991]

Challenges: learning with minimal supervision

- Less
- More

Find the pottopod

What kinds of things work best today?

- Reading license plates, zip codes, checks
- Recognizing flat, textured objects (like books, CD covers, posters)
- Frontal face detection
- Fingerprint recognition

What kinds of things work best today?
Evolution of methods

- Hand-crafted models
- Hand-crafted features
- 3D geometry
- Learned models
- Hypothesize and align
- Data-driven
- "End-to-end” learning of features and models*, **

* Labeled data availability
** Architecture design decisions, parameters.

Generic category recognition: basic framework

- Build/train object model
  - (Choose a representation)
  - Learn or fit parameters of model / classifier
- Generate candidates in new image
- Score the candidates

Window-based models
Generating and scoring candidates

- Window-based object detection
  Training:
  1. Obtain training data
  2. Select/learn features/classifier
  Given new image:
  1. Slide window
  2. Score by classifier

Object proposals: all windows -> probable regions

- How “object-like” is each candidate region?
  - Factors in choosing:
    - Generative or discriminative model?
    - Data resources – how much training data?
    - How is the labeled data prepared?
    - Training time allowance
    - Test time requirements – real-time?
    - Fit with the representation

Object recognition as classification

- What classifier?
  - Factors in choosing:
    - Generative or discriminative model?
    - Data resources – how much training data?
    - How is the labeled data prepared?
    - Training time allowance
    - Test time requirements – real-time?
    - Fit with the representation

* Labeled data availability
** Architecture design decisions, parameters.
Object recognition as classification

- What categories are amenable to window-based classification?
  - Similar to specific object matching, we expect spatial layout to be roughly preserved.
  - Unlike specific object matching, by training classifiers we attempt to capture intra-class variation or determine required discriminative features.

Viola-Jones face detector

Main idea:
- Represent local texture with efficiently computable "rectangular" features within window of interest
- Select discriminative features to be weak classifiers
- Use boosted combination of them as final classifier
- Form a cascade of such classifiers, rejecting clear negatives quickly

Image classification
Three landmark case studies

- Boosting + face detection
  - Viola & Jones
- NN + scene Gist classification
  - e.g., Hays & Efros
- SVM + person detection
  - e.g., Dalal & Triggs

Boosting intuition

Boosting illustration
Boosting: training

• Initially, weight each training example equally
• In each boosting round:
  – Find the weak learner that achieves the lowest weighted training error
  – Raise weights of training examples misclassified by current weak learner
• Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
• Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Boosting: pros and cons

• Advantages of boosting
  • Integrates classification with feature selection
  • Complexity of training is linear in the number of training examples
  • Flexibility in the choice of weak learners, boosting scheme
  • Testing is fast
  • Easy to implement
• Disadvantages
  • Needs many training examples
  • Often found not to work as well as an alternative discriminative classifier, support vector machine (SVM), or CNNs
    – especially for many-class problems
Viola-Jones detector: features

"Rectangular" filters
Feature output is difference between adjacent regions

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

![Integral Image](image)

Kristen Grauman

Computing sum within a rectangle

- Let A, B, C, D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:
  \[ \text{sum} = A - B - C + D \]
- Only 3 additions are required for any size of rectangle!

Kristen Grauman

Viola-Jones detector: features

"Rectangular" filters
Feature output is difference between adjacent regions

Avoid scaling images → scale features directly for same cost

![Integral Image](image)

Kristen Grauman

Viola-Jones detector: AdaBoost

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.

\[
h(x) = \begin{cases} 
  +1 & \text{if } f(x) > \theta \\
  -1 & \text{otherwise}
\end{cases}
\]

Outputs of a possible rectangle feature on faces and non-faces.

Kristen Grauman

Viola-Jones Face Detector: Results

First two features selected

Kristen Grauman
• Even if the filters are fast to compute, each new image has a lot of possible windows to search.
• How to make the detection more efficient?

Cascading classifiers for detection

- Form a cascade with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Kristen Grauman

Viola-Jones detector: summary

- Train cascade of classifiers with AdaBoost
- Select features, thresholds, and weights
- New images

Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

[Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv]

Kristen Grauman

Viola-Jones Face Detector: Results

P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004.
Viola-Jones Face Detector: Results

Detecting profile faces?

Can we use the same detector?

Viola-Jones Face Detector: Results

Example using Viola-Jones detector

Consumer application: iPhoto

Everingham, M., Sivic, J. and Zisserman, A.

http://www.apple.com/ilife/iphoto/
**Consumer application: iPhoto**

*Things iPhoto thinks are faces*

![Image of cookies with a box around one](image1.png)

*Slide credit: Lana Lazebnik*

**Consumer application: iPhoto**

*Can be trained to recognize pets!*

![Image of iPhoto interface with cats](image2.png)

*http://www.maclife.com/article/news/iphotos_faces_recognizes_cats*

---

**Privacy Gift Shop – CV Dazzle**

![Image of woman with black and white patterns on her face](image3.png)

*http://www.wired.com/2015/06/facebook-can-recognize-even-dont-show-face/*

*Wired, June 15, 2015*

**Privacy Visor**

![Image of people wearing visors](image4.png)


---

**Window-based detection: strengths**

- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

**Window-based detection: Limitations**

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
  - With so many windows, false positive rate better be low
Limitations (continued)

• Not all objects are “box” shaped

Limitations (continued)

• Non-rigid, deformable objects not captured well with representations assuming a fixed 2D structure; or must assume fixed viewpoint

• Objects with less-regular textures not captured well with holistic appearance-based descriptions

• If considering windows in isolation, context is lost

• In practice, often entails large, cropped training set (expensive)

• Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

Visual Object Recognition Tutorial

Figure credit: Derek Hoiem

Image credit: Adam, Rivlin, & Shimshoni

Image classification:
Three landmark case studies

- Boosting + face detection
  - Viola & Jones

- NN + scene Gist classification
  - e.g., Hays & Efros

- SVM + person detection
  - e.g., Dalal & Triggs

Nearest Neighbor classification

• Assign label of nearest training data point to each test data point

- Black = negative
- Red = positive

Voronoi partitioning of feature space for 2-category 2D data
K-Nearest Neighbors classification

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify

If query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.

80M Tiny Images [Torralba et al. 2008]

Where in the World?

[Slide credit: James Hays]

6+ million geotagged photos by 109,788 photographers

Annotated by Flickr users [Slide credit: James Hays]
6+ million geotagged photos by 109,788 photographers

Which scene properties are relevant?

- Gist scene descriptor
- Color Histograms - 4x14x14 histograms
- Texton Histograms – 512 entry, filter bank based
- Line Features – Histograms of straight line stats
Scene Matches

The Importance of Data
Nearest neighbors: pros and cons

• **Pros:**
  – Simple to implement
  – Flexible to feature / distance choices
  – Naturally handles multi-class cases
  – Can do well in practice with enough representative data

• **Cons:**
  – Large search problem to find nearest neighbors
  – Storage of data
  – Must know we have a meaningful distance function

Today

• Intro to categorization problem
• Object categorization as discriminative classification
  • Boosting + fast face detection example
  • Nearest neighbors + scene recognition example
    • Support vector machines + pedestrian detection example
    • Pyramid match kernels, spatial pyramid match
  • Convolutional neural networks + ImageNet example

Image classification: Three landmark case studies

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting + face detection</td>
<td>Viola &amp; Jones</td>
<td>e.g., Dalal &amp; Triggs</td>
</tr>
<tr>
<td>NN + scene Gist classification</td>
<td>e.g., Hays &amp; Efros</td>
<td></td>
</tr>
<tr>
<td>SVM + person detection</td>
<td>e.g., Dalal &amp; Triggs</td>
<td></td>
</tr>
</tbody>
</table>

Linear classifiers

• Find linear function to separate positive and negative examples

\[ \mathbf{x}, \text{positive} : \mathbf{w} \cdot \mathbf{x} + b \geq 0 \]
\[ \mathbf{x}, \text{negative} : \mathbf{w} \cdot \mathbf{x} + b < 0 \]

Which line is best?

Support Vector Machines (SVMs)

• Discriminative classifier based on optimal separating hyperplane

• Maximize the margin between the positive and negative training examples
Support vector machines
• Want line that maximizes the margin.

\[ \mathbf{x}_i, \text{positive (} y_i = +1\text{)}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \]
\[ \mathbf{x}_i, \text{negative (} y_i = -1\text{)}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \]

For support vectors, \( \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1 \)

Distance between point and line:
\[ \frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{||\mathbf{w}||} \]

Therefore, the margin is \( \frac{2}{||\mathbf{w}||} \)

Finding the maximum margin line
• Solution:
\[ \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \]

\[ \text{Learned weight \hspace{1cm} Support vector} \]

Finding the maximum margin line
1. Maximize margin \( \frac{2}{||\mathbf{w}||} \)
2. Correctly classify all training data points:
\[ \mathbf{x}_i, \text{positive (} y_i = +1\text{)}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \]
\[ \mathbf{x}_i, \text{negative (} y_i = -1\text{)}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \]

**Quadratic optimization problem:**

Minimize \( \frac{1}{2} \mathbf{w}^T \mathbf{w} \)

Subject to \( y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \)

\[ \mathbf{w} \cdot \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \]

\[ f(x) = \text{sign} \left( \mathbf{w} \cdot \mathbf{x} + b \right) \]

\[ = \text{sign} \left( \sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \right) \]
Person detection with HoG’s & linear SVM’s

- Map each grid cell in the input window to a histogram counting the gradients per orientation.
- Train a linear SVM using training set of pedestrian vs. non-pedestrian windows.

Dalal & Triggs, CVPR 2005

HoG descriptor

- Datasets that are linearly separable with some noise work out great:
- But what are we going to do if the dataset is just too hard?
- How about… mapping data to a higher-dimensional space:

YOLO detector


Question

- What if the data is not linearly separable?
Nonlinear SVMs

- **The kernel trick**: instead of explicitly computing the lifting transformation $\phi(x)$, define a kernel function $K$ such that
  $$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

- This gives a nonlinear decision boundary in the original feature space:
  $$\sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) + b$$

Example

2-dimensional vectors $x = [x_1, x_2]$;
let $K(x_i, x_j) = (1 + x_i^T x_j)^2$

Need to show that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$:
$$K(x_i, x_j) = (1 + x_i^T x_j)^2 = 1 + 2x_i^T x_j + x_i^2 x_j^2 + 2x_i^T x_j + 2x_i x_j^2$$
$$= \begin{bmatrix} 1 & x_i & x_i^T \end{bmatrix} \begin{bmatrix} 1 & x_j & x_j^T \end{bmatrix}$$
where $\phi(x) = \begin{bmatrix} 1 & x_1 & x_1^T \end{bmatrix}$

Examples of kernel functions

- **Linear**: $K(x_i, x_j) = x_i^T x_j$
- **Gaussian RBF**: $K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$
- **Histogram intersection**: $K(x_i, x_j) = \sum_{k} \min(x_i(k), x_j(k))$

SVMs for recognition

1. Define your representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples.
4. Use this “kernel matrix” to solve for SVM support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.

What about a matching kernel?

Local feature correspondence useful similarity measure for generic object categories

Partially matching sets of features

Optimal match: $O(m^2)$
Greedy match: $O(mn \log m)$
Pyramid match: $O(m)$

($m$=num pts)

Previous work: Indyk & Thaper, Bartal, Charikar, Agarwal & Vardarajan, …
Pyramid match: main idea

Feature space partitions serve to "match" the local descriptors within successively wider regions.

Pyramid match kernel

\[ K_\Delta(X, Y) = \sum_{i=0}^{L} 2^{-i} \left( \tilde{H}_X^{(i)}, \tilde{H}_Y^{(i)} \right) \]

measures
difficulty of a
match at level \(i\)

• For similarity, weights inversely proportional to bin size
  (or may be learned)
• Normalize these kernel values to avoid favoring large sets

Pyramid match: main idea

Histogram intersection counts number of possible matches at a given partitioning.

Pyramid match kernel

\[ I(H_X, H_Y) = \sum_{i=1}^{3} \min(H_X(i), H_Y(i)) \]

Optimal partial matching

Optimal match: \(O(m^3)\)

Pyramid match: \(O(mL)\)

Unordered sets of local features:
No spatial layout preserved!

Spatial pyramid match

• Make a pyramid of bag-of-words histograms.
• Provides some loose (global) spatial layout information
Spatial pyramid match

- Make a pyramid of bag-of-words histograms.
- Provides some loose (global) spatial layout information

\[ K^L(X,Y) = \sum_{m=1}^{M} K^L(X_m,Y_m) \]

Sum over PMKs computed in image coordinate space, one per word.

[Szabo, Schmid & Ponce, CVPR 2008]

Spatial pyramid match

- Can capture scene categories well—texture-like patterns but with some variability in the positions of all the local pieces.
- Sensitive to global shifts of the view

SVMs: Pros and cons

**Pros**
- Kernel-based framework is very powerful, flexible
- Often a sparse set of support vectors—compact at test time
- Work very well in practice, even with very small training sample sizes

**Cons**
- No “direct” multi-class SVM, must combine two-class SVMs
- Can be tricky to select best kernel function for a problem
  - During training time, must compute matrix of kernel values for every pair of examples
  - Learning can take a very long time for large-scale problems

Basic recognition models so far

Instances: recognition by alignment

Categories: Holistic appearance models (and sliding window detection)
Today

- Intro to categorization problem
- Object categorization as discriminative classification
  - Boosting + fast face detection example
  - Nearest neighbors + scene recognition example
  - Support vector machines + pedestrian detection example
  - Pyramid match kernels, spatial pyramid match
    - Convolutional neural networks + ImageNet example
- Some new representations along the way
  - Rectangular filters
  - GIST
  - HOG

Evolution of methods

- Hand-crafted models
- 3D geometry
- Hypothesize and align
- Hand-crafted features
- Learned models
- Data-driven
- "End-to-end" learning of features and models*,**

Traditional Image Categorization:

Training phase

- Training Images
- Training Labels
- Image Features
- Classifier Training
- Trained Classifier

Testing phase

- Test Image
- Image Features
- Trained Classifier
- Prediction

Features have been key

- SIFT \([\text{Lowe, IJCV, 04}]\)
- HOG \([\text{Dalal and Triggs, CVPR, 05}]\)
- and many others:
  - SURF, MSER, LBP, GIST, Color-SIFT, Color histogram, GLOH, …

Learning a Hierarchy of Feature Extractors

- Each layer of hierarchy extracts features from output of previous layer
- All the way from pixels \(\rightarrow\) classifier
- Layers have the (nearly) same structure

- Train all layers jointly
Learning Feature Hierarchy

Goal: Learn useful higher-level features from images

Feature representation

1st layer
- "Edges"

2nd layer
- "Object parts"

3rd layer
- "Objects"

Input data

Lee et al., ICML 2009; CACM 2011

Better performance

Other domains (Less clear how to hand engineer?):
- Kinect
- Video
- Multi spectral

Feature computation time
- Dozens of features now regularly used [e.g., MKL]
- Getting prohibitive for large datasets (10’s sec /image)

Learning Feature Hierarchy

Biological neuron and Perceptrons

A biological neuron

An artificial neuron (Perceptron)
- a linear classifier

Simple, Complex and Hypercomplex cells

Suggested a hierarchy of feature detectors in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.

Hubel/Wiesel Architecture and Multi-layer Neural Network

Hubel and Weisel’s architecture

Multi-layer Neural Network
- A non-linear classifier

Neuron: Linear Perceptron

Inputs are feature values
Each feature has a weight
Sum is the activation

activation\(_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)\)

If the activation is:
- Positive, output +1
- Negative, output -1
Learning $w$

- Training examples
  \[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\]

- Objective: a misclassification loss
  \[\min_w \sum_{i=1}^{m} \left(y^{(i)} - h_w(f(x^{(i)}))\right)^2\]

- Procedure:
  - Gradient descent / hill climbing
Two-layer neural network

Neural network properties

- Theorem (Universal function approximators): A two-layer network with a sufficient number of neurons can approximate any continuous function to any desired accuracy

- Practical considerations:
  - Can be seen as learning the features
  - Large number of neurons
  - Danger for overfitting
  - Hill-climbing procedure can get stuck in bad local optima

Significant recent impact on the field

Convolutional Neural Networks (CNN, ConvNet, DCN)

- CNN = a multi-layer neural network with
  - Local connectivity:
    - Neurons in a layer are only connected to a small region of the layer before it
  - Share weight parameters across spatial positions:
    - Learning shift-invariant filter kernels

Neocognitron \cite{Fukushima, Biological Cybernetics 1980}

Deformation-Resistant Recognition

- S-cells: (simple) - extract local features
- C-cells: (complex) - allow for positional errors

LeNet \cite{LeCun et al. 1998}

Gradient-based learning applied to document recognition \cite{LeCun, Bottou, Bengio, Haffner, 1998}

LeNet-1 from 1993
What is a Convolution?

- Weighted moving sum

Convolutional Neural Networks

Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution (Learned)
Input Image

Input
Feature Map
slide credit: S. Lazebnik

Convolutional Neural Networks

Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution (Learned)
Input Image

Input Image
Feature Map
slide credit: S. Lazebnik

Convolutional Neural Networks

Max-pooling: a non-linear down-sampling
Provide translation invariance

Convolutional Neural Networks

Feature maps
Normalization
Spatial pooling
Non-linearity
Convolution (Learned)
Input Image

Input Image
Feature Map
slide credit: S. Lazebnik
Engineered vs. learned features

Convolutional filters are trained in a supervised manner by back-propagating classification error.

Label

Convolution/pool

Pooling

Feature extraction

Image

SIFT Descriptor

Image Pixels

Apply oriented filters

Spatial pool (Sum)

Normalize to unit length

Feature Vector

SIFT Descriptor

Lowe [IJCV 2004]

Spatial Pyramid Matching

SIFT Features

Filter with Visual Words

Max

Multi-scale spatial pool (Sum)

Classifier

Lazebnik, Schmid, Ponce [CVPR 2006]

Visualizing what was learned

What do the learned filters look like?

Typical first layer filters

Applications

Handwritten text/digits
- MNIST (0.17% error [Ciresan et al. 2011])
- Arabic & Chinese [Ciresan et al. 2012]

Simpler recognition benchmarks
- CIFAR-10 (9.3% error [Wan et al. 2013])
- Traffic sign recognition
  - 0.56% error vs. 1.16% for humans [Ciresan et al. 2011]

Google's Artificial Brain Learns to Find Cat Videos

https://www.wired.com/2012/06/google-a-neural-network/
Application: ImageNet

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

[Deng et al. CVPR 2009]

AlexNet

- Similar framework to LeCun’98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data ($10^6$ vs. $10^3$ images)
  - GPU implementation (50x speedup over CPU)
  - Trained on two GPUs for a week


ImageNet Classification Challenge

Industry Deployment

- Used in Facebook, Google, Microsoft
- Image Recognition, Speech Recognition, ....
- Fast at test time

[Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR’14]

Beyond classification

- Detection
- Segmentation
- Regression
- Pose estimation
- Matching patches
- Synthesis

and many more...

[Ja-Bin Huang and Derek Hoiem, UIUC]

Recap

- Neural networks / multi-layer perceptrons
  - View of neural networks as learning hierarchy of features
- Convolutional neural networks
  - Architecture of network accounts for image structure
  - “End-to-end” recognition from pixels
  - Together with big (labeled) data and lots of computation \( \rightarrow \) major success on benchmarks, image classification and beyond

[Ja-Bin Huang and Derek Hoiem, UIUC]