Recap: Part-Based Models

- Fischler & Elschlager 1973
- Model has two components
  - parts (2D image fragments)
  - structure (configuration of parts)

Recap: Implicit Shape Model - Representation

- Learn appearance codebook
  - Extract local features at interest points
  - Clustering appearance codebook
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object

Recap: Deformable Part-Based Model

- Root filters
  - coarse resolution
- Part filters
  - finer resolution
- Deformation models

Recap: Object Hypothesis

- Multiscale model captures features at two resolutions
Recap: Score of a Hypothesis

\[ \text{score}(p_0, \ldots, p_n) = \sum \text{filters} \cdot \phi(H, p_i) - \sum \text{displacements} \cdot d_i \cdot (d x^2, d y^2) \]

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

Topics of This Lecture

- Deep Learning
  - Motivation
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
- Applications

We've finally got there!

Traditional Recognition Approach

- Characteristics
  - Features are not learned, but engineered
  - Trainable classifier is often generic (e.g., SVM)
  - Many successes in 2000-2010.

- Features are key to recent progress in recognition
  - Multitude of hand-designed features currently in use
    - SIFT, HOG, ...........
  - Where next? Better classifiers? Or keep building more features?

What About Learning the Features?

- Learn a feature hierarchy all the way from pixels to classifier
  - Each layer extracts features from the output of previous layer
  - Train all layers jointly
“Shallow” vs. “Deep” Architectures

Traditional recognition: “Shallow” architecture
- Image/Video Pixels
- Hand-designed feature extraction
- Trainable classifier
- Object Class

Deep learning: “Deep” architecture
- Image/Video Pixels
- Layer 1 → … → Layer N
- Simple classifier
- Object Class

Background: Perceptrons

Input
- $x_1$
- $w_1$
- $x_2$
- $w_2$
- $x_3$
- $w_3$
- …
- $x_d$
- $w_d$

Output: $\sigma(w \cdot x + b)$

Sigmoid function

$\sigma(i) = \frac{1}{1 + e^{-i}}$

Inspiration: Neuron Cells

Axonal arborization
Dendrites
Synapse
Axon from another cell
Nucleus
Cell body or Soma

Background: Multi-Layer Neural Networks

- Nonlinear classifier
  - Training: find network weights $w$ to minimize the error between true training labels $t_n$ and estimated labels $f_w(x_n)$:
    $$E(W) = \sum L(t_n, f(x_n; W))$$
  - Minimization can be done by gradient descent, provided $f$ is differentiable
    - Training method: error backpropagation.

Hubel/Wiesel Architecture

  - Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells

Convolutional Neural Networks (CNN, ConvNet)

- 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

Topics of This Lecture

- Deep Learning
  - Motivation
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
- Applications

Convolutional Networks: Structure

- Feed-forward feature extraction
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. (Normalization)
- Supervised training of convolutional filters by back-propagating classification error

Convolutional Networks: Intuition

- Fully connected network
  - E.g. 1000x1000 image
  - 1M hidden units
  - $\approx$ 1T parameters!
- Ideas to improve this
  - Spatial correlation is local

Convolutional Networks: Intuition

- Locally connected net
  - E.g. 1000x1000 image
  - 100 filters
  - 10x10 receptive fields
  - $\approx$ 100M parameters!
- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance
Important Conceptual Shift

• Before

• Now:

Convolution Layers

• Note: Connectivity is
  • Local in space (5 x 5 inside 32 x 32)
  • But full in depth (all 3 depth channels)

Convolution Layers

• All Neural Net activations arranged in 3 dimensions
  • Multiple neurons all looking at the same input region, stacked in depth

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 x 1 x depth] depth column in output volume.

Convolution Layers

• Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

• Replicate this column of hidden neurons across space, with some stride.
Convolution Layers

- Replicate this column of hidden neurons across space, with some stride.

Example:
7 × 7 input
assume 3 × 3 connectivity
stride 1
⇒ 5 × 5 output

What about stride 2?
Convolution Layers

Example:
7 x 7 input
assume 3 x 3 connectivity
stride 1 ⇒ 5 x 5 output

What about stride 2?
⇒ 3 x 3 output

• Replicate this column of hidden neurons across space, with some stride.
• In practice, common to zero-pad the border.
  > Preserves the size of the input spatially.

Activation Maps of Convolutional Filters

38

B. Leibe

Slide credit: FeiFei Li, Andrej Karpathy

Example:
7 £ 7 input
assume 3 £ 3 connectivity
stride 1
⇒ 5 £ 5 output

What about stride 2?
⇒ 3 £ 3 output

Effect of Multiple Convolution Layers

40

B. Leibe

Slide credit: Yann LeCun

Activation maps

Each activation map is a depth slice through the output volume.

Commonly Used Nonlinearities

• Sigmoid
  \[ g(a) = \sigma(a) = \frac{1}{1+\exp(-a)} \]

• Hyperbolic tangent
  \[ g(a) = \tanh(a) = 2\sigma(2a) - 1 \]

• Rectified linear unit (ReLU)
  \[ g(a) = \max\{0, a\} \]

Preferred option for deep networks

Convolutional Networks: Intuition

• Let’s assume the filter is an eye detector
  > How can we make the detection robust to the exact location of the eye?

Convolutional Networks: Intuition

• Let’s assume the filter is an eye detector
  > How can we make the detection robust to the exact location of the eye?

• Solution:
  > By pooling (e.g., max or avg) filter responses at different spatial locations, we gain robustness to the exact spatial location of features.
Max Pooling

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

Note
- Pooling happens independently across each slice, preserving the number of slices.

Compare: SIFT Descriptor

Compare: Spatial Pyramid Matching

Topics of This Lecture
- Deep Learning
  - Motivation
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
- Applications


- Early convolutional architecture
  - 2 Convolutional layers, 2 pooling layers
  - Fully-connected NN layers for classification
  - Successfully used for handwritten digit recognition (MNIST)

**ImageNet Challenge 2012**

- **ImageNet**
  - ~14M labeled internet images
  - 20k classes
  - Human labels via Amazon Mechanical Turk

- **Challenge (ILSVRC)**
  - 1.2 million training images
  - 1000 classes
  - Goal: Predict ground-truth class within top-5 responses
  - Currently one of the top benchmarks in Computer Vision

**CNN Architectures: AlexNet (2012)**

- Similar framework as LeNet, but
  - Bigger model (7 hidden layers, 650k units, 60M parameters)
  - More data ($10^6$ images instead of $10^3$)
  - GPU implementation
  - Better regularization and up-to-date tricks for training (Dropout)

**ILSVRC 2012 Results**

- AlexNet almost halved the error rate
  - 16.4% error (top-5) vs. 26.2% for the next best approach
  - Acquired by Google in Jan ’13, deployed in Google+ in May ’13

**AlexNet Results**

- Test image
- Retrieved images

**CNN Architectures: VGGNet (2014/15)**

CNN Architectures: VGGNet (2014/15)

- **Main ideas**
  - Deeper network
  - Stacked convolutional layers with smaller filters (+ nonlinearity)
  - Detailed evaluation of all components

- **Results**
  - Improved ILSVRC top-5 error rate to 6.7%.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>11 x 11</td>
<td>11 x 11</td>
</tr>
<tr>
<td>conv2</td>
<td>11 x 11</td>
<td>11 x 11</td>
</tr>
<tr>
<td>conv3</td>
<td>11 x 11</td>
<td>11 x 11</td>
</tr>
<tr>
<td>conv4</td>
<td>11 x 11</td>
<td>11 x 11</td>
</tr>
<tr>
<td>conv5</td>
<td>11 x 11</td>
<td>11 x 11</td>
</tr>
</tbody>
</table>

Comparison: AlexNet vs. VGGNet

- **Receptive fields in the first layer**
  - AlexNet: $11 \times 11$, stride 4
  - Zeiler & Fergus: $7 \times 7$, stride 2
  - VGGNet: $3 \times 3$, stride 1

- **Why that?**
  - If you stack three $3 \times 3$ on top of another $3 \times 3$ layer, you effectively get a $5 \times 5$ receptive field.
  - With three $3 \times 3$ layers, the receptive field is already $7 \times 7$.
  - But much fewer parameters: $3^2 \cdot 3^2 = 27$ instead of $7^2 = 49$.
  - In addition, non-linearities in-between $3 \times 3$ layers for additional discriminativity.

CNN Architectures: GoogLeNet (2014)

- **Main ideas**
  - “Inception” module as modular component
  - Learns filters at several scales within each module


GoogLeNet Visualization

Results on ILSVRC

<table>
<thead>
<tr>
<th>Method</th>
<th>top-1 val. error (%)</th>
<th>top-5 val. error (%)</th>
<th>top-5 test. error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-2 (net, multi-cept. &amp; dense eval.)</td>
<td>21.7</td>
<td>6.8</td>
<td>6.8</td>
</tr>
<tr>
<td>VGG (4 net, multi-feat. &amp; dense eval.)</td>
<td>21.7</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td>VGG (ILSVRC submission, 3 net, dense eval.)</td>
<td>21.7</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>GoogleNet (Szegedy et al., 2014)</td>
<td>-</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>MNIST (Fak et al., 2014/11)</td>
<td>27.9</td>
<td>9.1</td>
<td>9.1</td>
</tr>
<tr>
<td>CIFAR-10 (Szegedy et al., 2014)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR-100 (Szegedy et al., 2014)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Zeiler &amp; Fergus (Zeiler &amp; Fergus, 2014)</td>
<td>36.0</td>
<td>15.9</td>
<td>14.8</td>
</tr>
<tr>
<td>Zeiler &amp; Fergus (Zeiler &amp; Fergus, 2013)</td>
<td>37.3</td>
<td>16.0</td>
<td>16.1</td>
</tr>
<tr>
<td>Overfeat (Simonyan et al., 2014)</td>
<td>54.0</td>
<td>15.2</td>
<td>15.2</td>
</tr>
<tr>
<td>Krizhevsky et al. (Krizhevsky et al., 2014)</td>
<td>39.1</td>
<td>16.4</td>
<td>16.4</td>
</tr>
<tr>
<td>Krizhevsky et al. (Krizhevsky et al., 2014)</td>
<td>40.9</td>
<td>15.8</td>
<td>-</td>
</tr>
</tbody>
</table>

- VGGNet and GoogLeNet perform at similar level
  - Comparison: human performance - 5% [Karpathy]

Newest Development: Residual Networks
Newest Development: Residual Networks

- Core component
  - Skip connections bypassing each layer
  - Better propagation of gradients to the deeper layers

ImageNet Performance

- 152 layers

Topics of This Lecture

- Deep Learning
  - Motivation
- Convolutional Neural Networks
  - Convolutional Layers
  - Pooling Layers
  - Nonlinearities
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
- Applications

The Learned Features are Generic

- Experiment: feature transfer
  - Train network on ImageNet
  - Chop off last layer and train classification layer on CalTech256
  - State of the art accuracy already with only 6 training images

Other Tasks: Detection

- R-CNN: Regions with CNN features

R-CNN: Regions with CNN features

- 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

Faster R-CNN (based on ResNets)

Faster R-CNN (based on ResNets)


Semantic Segmentation


Other Tasks: Semantic Segmentation

Farabet et al. ICML 2012, PAMI 2013

Other Tasks: Face Verification

Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014

Commercial Recognition Services

Be careful when taking test images from Google Search. Chances are they may have been seen in the training set...

Image source: clarifai.com

Commercial Recognition Services

Image source: clarifai.com
References and Further Reading

- **LeNet**

- **AlexNet**

- **VGGNet**

- **GoogLeNet**