Computer Vision - Lecture 15

Deep Learning for Object Categorization


Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de
Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Object Categorization I
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features - Detection and Description
  - Recognition with Local Features
  - Indexing & Visual Vocabularies
- Object Categorization II
  - Bag-of-Words Approaches & Part-based Approaches
  - Deep Learning Methods
- 3D Reconstruction
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 \(\Rightarrow\) appearance codebook

- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object

Training images (+reference segmentation)

Appearance codebook

Spatial occurrence distributions
+ local figure-ground labels
Recap: Deformable Part-Based Model

Root filters
coarse resolution

Part filters
finer resolution

Deformation models

Slide credit: Pedro Felzenszwalb
Recap: Object Hypothesis

- Multiscale model captures features at two resolutions

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

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

Slide credit: Pedro Felzenszwalb
Recap: 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

score \((z) = \beta \cdot \Psi(H, z)\)

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

Slide credit: Pedro Felzenszwalb

B. Leibe
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!

Deep Learning
Traditional Recognition Approach

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

- 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?

DPM
[Felzenszwalb et al., PAMI’07]

Dense SIFT+LBP+HOG → BOW → Classifier
[Yan & Huan ‘10] (Winner of PASCAL 2010 Challenge)
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

![Diagram showing a feature hierarchy from Image/Video Pixels to Simple Classifier, with three layers labeled as Layer 1, Layer 2, and Layer 3.](image-url)
“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

Slide credit: Svetlana Lazebnik
Background: Perceptrons

Input

\[ x_1, x_2, x_3, \ldots, x_d \]

Weights

\[ w_1, w_2, w_3, \ldots, w_d \]

Output:

\[ \sigma(w \cdot x + b) \]

Sigmoid function

\[ \sigma(t) = \frac{1}{1 + e^{-t}} \]
Inspiration: Neuron Cells

Dendrite

Synapse

Axon from another cell

Axonal arborization

Nucleus

Cell body or Soma

Synapses

Slide credit: Svetlana Lazebnik, Rob Fergus
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**.

Slide credit: Svetlana Lazebnik
Hubel/Wiesel Architecture

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

Slide credit: Svetlana Lazebnik, Rob Fergus
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


Slide credit: Svetlana Lazebnik
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. $1000 \times 1000$ image
    - 1M hidden units
  $\Rightarrow$ 1T parameters!

- Ideas to improve this
  - Spatial correlation is local

Slide adapted from Marc’Aurelio Ranzato

B. Leibe

Image source: Yann LeCun
Convolutional Networks: Intuition

- Locally connected net
  - E.g. $1000 \times 1000$ image
  - $1M$ hidden units
  - $10 \times 10$ receptive fields
  - $\Rightarrow 100M$ parameters!

- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance
Convolutional Networks: Intuition

- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
Convolutional Networks: Intuition

- **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
  - $\Rightarrow$ 10k parameters

- **Result: Response map**
  - size: $1000 \times 1000 \times 100$
  - Only memory, not params!
Important Conceptual Shift

• **Before**

• **Now:**

Slide credit: FeiFei Li, Andrej Karpathy
**Convolution Layers**

- **Note: Connectivity is**
  - Local in space (5×5 inside 32×32)
  - But full in depth (all 3 depth channels)

Example image: 32×32×3 volume

**Before:** Full connectivity
32×32×3 weights

**Now:** Local connectivity
One neuron connects to, e.g., 5×5×3 region.
⇒ Only 5×5×3 shared weights.
Convolution Layers

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

Slide adapted from FeiFei Li, Andrej Karpathy
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.

Naming convention:

Slide credit: FeiFei Li, Andrej Karpathy
Convolution Layers

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

Example:
7×7 input
assume 3×3 connectivity
stride 1

Example: 7×7 input
assume 3×3 connectivity
stride 1
Convolution Layers

Example:
7×7 input
assume 3×3 connectivity
stride 1

- 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
Convolution Layers

Example:
7×7 input
assume 3×3 connectivity
stride 1

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

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

• 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

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

Example:

7×7 input
assumption 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?
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?
⇒ 3×3 output

Slide credit: FeiFei Li, Andrej Karpathy
Convolution Layers

- 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.

Example:
- 7×7 input
- assume 3×3 connectivity
- stride 1

⇒ 5×5 output

What about stride 2?
⇒ 3×3 output
Activation Maps of Convolutional Filters

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

Activations: 5x5 filters

Activation maps

Slide adapted from FeiFei Li, Andrej Karpathy
Effect of Multiple Convolution Layers

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun
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.

Slide adapted from Marc’Aurelio Ranzato

Image source: Yann LeCun
Max Pooling

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

- Note
  - Pooling happens independently across each slice, preserving the number of slices.
Compare: SIFT Descriptor

Image Pixels → Apply oriented filters → Spatial pool (Sum) → Normalize to unit length → Feature Vector

Lowe [IJCV 2004]

Slide credit: Svetlana Lazebnik
Compare: Spatial Pyramid Matching

- SIFT features
- Filter with Visual Words
- Take max VW response (L-inf normalization)
- Multi-scale spatial pool (Sum)
- Global image descriptor

Lazebnik, Schmid, Ponce
[CVPR 2006]

Slide credit: Svetlana Lazebnik
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)


Slide credit: Svetlana Lazebnik
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

[Deng et al., CVPR’09]
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
  - A revolution in Computer Vision
  - Acquired by Google in Jan ‘13, deployed in Google+ in May ‘13
AlexNet Results

AlexNet Results

Test image

Retrieved images

CNN Architectures: VGGNet (2014/15)


Image source: Hirokatsu Kataoka
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%.

![ConvNet Configuration Table](image)

**Mainly used**

*Image source: Simonyan & Zisserman*
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: $3 \cdot 3^2 = 27$ instead of $7^2 = 49$.
  - In addition, non-linearities in-between $3 \times 3$ layers for additional discriminativity.
CNN Architectures: GoogLeNet (2014)

(a) Inception module, naïve version

(b) Inception module with dimension reductions

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

GoogLeNet Visualization

Inception module + copies

Auxiliary classification outputs for training the lower layers (deprecated)

Convolution Pooling Softmax Other

B. Leibe
## 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 nets, multi-crop &amp; dense eval.)</td>
<td>23.7</td>
<td>6.8</td>
<td>6.8</td>
</tr>
<tr>
<td>VGG (1 net, multi-crop &amp; dense eval.)</td>
<td>24.4</td>
<td>7.1</td>
<td>7.0</td>
</tr>
<tr>
<td>VGG (ILSVRC submission, 7 nets, dense eval.)</td>
<td>24.7</td>
<td>7.5</td>
<td>7.3</td>
</tr>
<tr>
<td>GoogLeNet (Szegedy et al., 2014) (1 net)</td>
<td>-</td>
<td>-</td>
<td>7.9</td>
</tr>
<tr>
<td>GoogLeNet (Szegedy et al., 2014) (7 nets)</td>
<td>-</td>
<td>-</td>
<td>6.7</td>
</tr>
<tr>
<td>MSRA (He et al., 2014) (11 nets)</td>
<td>-</td>
<td>-</td>
<td>8.1</td>
</tr>
<tr>
<td>MSRA (He et al., 2014) (1 net)</td>
<td>27.9</td>
<td>9.1</td>
<td>9.1</td>
</tr>
<tr>
<td>Clarifai (Russakovsky et al., 2014) (multiple nets)</td>
<td>-</td>
<td>-</td>
<td>11.7</td>
</tr>
<tr>
<td>Clarifai (Russakovsky et al., 2014) (1 net)</td>
<td>-</td>
<td>-</td>
<td>12.5</td>
</tr>
<tr>
<td>Zeiler &amp; Fergus (Zeiler &amp; Fergus, 2013) (6 nets)</td>
<td>36.0</td>
<td>14.7</td>
<td>14.8</td>
</tr>
<tr>
<td>Zeiler &amp; Fergus (Zeiler &amp; Fergus, 2013) (1 net)</td>
<td>37.5</td>
<td>16.0</td>
<td>16.1</td>
</tr>
<tr>
<td>OverFeat (Sermanet et al., 2014) (7 nets)</td>
<td>34.0</td>
<td>13.2</td>
<td>13.6</td>
</tr>
<tr>
<td>OverFeat (Sermanet et al., 2014) (1 net)</td>
<td>35.7</td>
<td>14.2</td>
<td>-</td>
</tr>
<tr>
<td>Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)</td>
<td>38.1</td>
<td>16.4</td>
<td>16.4</td>
</tr>
<tr>
<td>Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)</td>
<td>40.7</td>
<td>18.2</td>
<td>-</td>
</tr>
</tbody>
</table>

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

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/
Newest Development: Residual Networks

AlexNet, 8 layers (ILSVRC 2012)
- 11x11 conv, 96, /4, pool/2
- 5x5 conv, 256, pool/2
- 3x3 conv, 384
- 3x3 conv, 384
- 3x3 conv, 256, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

VGG, 19 layers (ILSVRC 2014)
- 3x3 conv, 64
- 3x3 conv, 64, pool/2
- 3x3 conv, 128
- 3x3 conv, 256, pool/2
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

GoogleNet, 22 layers (ILSVRC 2014)
Newest Development: Residual Networks

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

\[ H(x) = F(x) + x \]

- AlexNet, 8 layers (ILSVRC 2012)
- VGG, 19 layers (ILSVRC 2014)
- ResNet, 152 layers (ILSVRC 2015)
ImageNet Performance

- ILSVRC'15: 3.57, ResNet
- ILSVRC'14: 6.7, GoogleNet; 7.3, VGG
- ILSVRC'13: 11.7, 8 layers
- ILSVRC'12: 16.4, AlexNet
- ILSVRC'11: 25.8, shallow
- ILSVRC'10: 28.2

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

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

Faster R-CNN (based on ResNets)

Faster R-CNN (based on ResNets)

Other Tasks: Semantic Segmentation

[Farabet et al. ICML 2012, PAMI 2013]
Semantic Segmentation

- More recent results
  - Based on an extension of ResNets

[Pohlen, Hermans, Mathias, Leibe, arXiv 2016]
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

Slide credit: Svetlana Lazebnik
Commercial Recognition Services

- E.g., clarifai

Try it out with your own media

Upload an image or video file under 100mb or give us a direct link to a file on the web.

*By using the demo you agree to our terms of service

- Be careful when taking test images from Google Search
  - Chances are they may have been seen in the training set...
Commercial Recognition Services
References and Further Reading

- **LeNet**

- **AlexNet**

- **VGGNet**

- **GoogLeNet**