Machine Learning – Lecture 16

Convolutional Neural Networks II

18.12.2019

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

Course Outline

- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
- Classification Approaches
  - Linear Discriminants
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Random Forests
- Deep Learning
  - Foundations
  - Convolutional Neural Networks
  - Recurrent Neural Networks

Topics of This Lecture

- Recap: CNNs
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNets
- Visualizing CNNs
  - Visualizing CNN features
  - Visualizing responses
  - Visualizing learned structures
- Applications

Recap: Convolutional Neural Networks

- 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

Recap: Intuition of CNNs

- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
- Learn multiple filters
  - E.g. 1000 × 1000 image
  - 100 filters
  - 10 × 10 filter size
  - Only 10k parameters
- Result: Response map
  - Size: 1000 × 1000 × 100
  - Only memory, not params!

Recap: 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 × 1 × depth] depth column in output volume.

Naming convention:
Convolution Layers

- Replicate this column of hidden neurons across space, with some \textit{stride}.

Example: 7 \times 7 input
assume 3 \times 3 connectivity
stride 1

\Rightarrow 5 \times 5 output

What about stride 2?
Convolution Layers

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

Example:

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

What about stride 2?

Convolution Layers

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

Example:

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

What about stride 2?

⇒ 3 x 3 output

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

Max Pooling

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

CNNs: Implication for Back-Propagation

- Convolutional layers
  - Filter weights are shared between locations
  - Gradients are added for each filter location.

Topics of This Lecture

- Recap: CNNs
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
- Visualizing CNNs
  - Visualizing CNN features
  - Visualizing responses
  - Visualizing learned structures
- 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
  - ~1.4M 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

ImageNet

\[ \text{ImageNet} \]

\[ \sim 14M \text{ labeled internet images} \]

\[ 20k \text{ classes} \]

\[ \text{Human labels via Amazon Mechanical Turk} \]

ILSVRC 2012

- 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

ILSVRC 2012 Results

[Graph showing error rates for different approaches]

CNN Architectures: AlexNet (2012)

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


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%

Comparison: AlexNet vs. VGGNet

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

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

- 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)

Results on ILSVRC

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

ImageNet Performance

- Core component
  - Skip connections bypassing each layer
  - Better propagation of gradients to the deeper layers
  - We'll analyze this mechanism in more detail later...

Newer Developments: Residual Networks

- Newer Developments: Residual Networks
  - Core component
    - Skip connections bypassing each layer
    - Better propagation of gradients to the deeper layers
    - We’ll analyze this mechanism in more detail later...

ImageNet Performance

- 152 layers
- 35.7
- 22 layers
- 19 layers
- 13.7
- 16.4
- 25.5
- 28.2
Understanding the ILSVRC Challenge

- Imagine the scope of the problem!
  - 1000 categories
  - 1.2M training images
  - 50k validation images

- This means...
  - Speaking out the list of category names at 1 word/s... takes 15mins.
  - Watching a slideshow of the validation images at 2s/image... takes a full day (24h+).
  - Watching a slideshow of the training images at 2s/image... takes a full month.

Quirks and Limitations of the Data Set

- Generated from WordNet ontology
  - Some animal categories are overrepresented
  - E.g., 120 subcategories of dog breeds

⇒ 6.7% top-5 error looks all the more impressive
Topics of This Lecture

• Recap: CNNs
• CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNets
• Visualizing CNNs
  - Visualizing CNN features
  - Visualizing responses
  - Visualizing learned structures
• Applications

Visualizing CNNs

- Visualizing CNN features
- Visualizing responses
- Visualizing learned structures

What Does the Network React To?

• Occlusion Experiment
  - Mask part of the image with an occluding square.
  - Monitor the output

What Does the Network React To?

Input image

p(True class)

Most probable class

Total activation in most active 5th layer feature map

Other activations from the same feature map.

What Does the Network React To?

Input image

Total activation in most active 5th layer feature map

Other activations from the same feature map.

What Does the Network React To?

Input image

Total activation in most active 5th layer feature map

Other activations from the same feature map.

What Does the Network React To?

Input image

Total activation in most active 5th layer feature map

Other activations from the same feature map.
Inceptionism: Dreaming ConvNets

• Idea
  ▶ Start with a random noise image.
  ▶ Enhance the input image such as to enforce a particular response (e.g., banana).
  ▶ Combine with prior constraint that image should have similar statistics as natural images.
  ⇒ Network hallucinates characteristics of the learned class.

http://googleresearch.blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html

Deeper into Neural Networks

Topics of This Lecture

• Recap: CNNs
• CNN Architectures
  ▶ LeNet
  ▶ AlexNet
  ▶ VGGNet
  ▶ GoogleLeNet
  ▶ ResNets
• Visualizing CNNs
  ▶ Visualizing CNN features
  ▶ Visualizing responses
  ▶ Visualizing learned structures
• Applications

Transfer Learning with CNNs

1. Train on ImageNet
2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

I.e., swap the Softmax layer at the end

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

state of the art level (pre-CNN)
Transfer Learning with CNNs

1. Train on ImageNet
2. If you have medium sized dataset, "finetune" instead: use the old weights as initialization, train the full network or only some of the higher layers.
3. If you have a large dataset, "pretrain" instead: use the old weights as initialization, and train the full network.

Other Tasks: Detection

R-CNN: Regions with CNN features
1. Input
2. Extract region image 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]
  - DPM: 33.4% mAP
  - R-CNN: 53.7% mAP

Most Recent Version: Faster R-CNN
- One network, four losses
  - Remove dependence on external region proposal algorithm.
  - Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training
    - Object detection in a single pass becomes possible.
    - mAP improved to >70%

Faster R-CNN (based on ResNets)


Object Detection Performance

- mAP improvement on PASCAL VOC dataset
  - Before deep connections
  - Using deep connections
Mask R-CNN Results

- Detection + Instance segmentation
- Detection + Pose estimation

YOLO / SSD

- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)

YOLO-v3 Results

Semantic Image Segmentation

- Perform pixel-wise prediction task
  - Usually done using Fully Convolutional Networks (FCNs)
    - All operations formulated as convolutions
    - Advantage: can process arbitrarily sized images
CNNs vs. FCNs

- **CNN**
  - Intuition: Think of CNNs as performing a sliding-window classification, producing a heatmap of output scores for each class.

- **FCN**
  - Intuition: Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class.

Semantic Image Segmentation

- **Encoder-Decoder Architecture**
  - Problem: FCN output has low resolution.
  - Solution: Perform upsampling to get back to desired resolution.
  - Use skip connections to preserve higher-resolution information.

Semantic Segmentation

- **Current state-of-the-art**
  - Based on an extension of ResNets.

Other Tasks: Face Verification

- Based on an extension of ResNets.

Commercial Recognition Services

- E.g., clarifai
  - Upload an image or video file and train the model to recognize entities in the image.

References and Further Reading

- **LeNet**

- **AlexNet**

- **VGGNet**

- **GoogLeNet**
References and Further Reading

- **ResNets**

References: Computer Vision Tasks

- **Object Detection**

- **Semantic Segmentation**