CMPSCI 670: Computer Vision
Deep learning

University of Massachusetts, Amherst
November 17, 2014

Instructor: Subhransu Maji
Homework 4 - grades posted
Homework 5 - due on Wednesday
Project
  • Presentations on Dec. 1 and 3
  • Each person (or team) will get 7 (or 10) mins to present
    - Preliminary results, data analysis, etc
  • Final report due on Dec. 13 (hard deadline)
Next lecture is a guest lecture by:
  • “Crafting the Perfect Selfie using Computer Vision”
    Aditya Khosla, MIT
Overview

• Shallow vs. deep architectures
• Background
  • Traditional neural networks
  • Inspiration from neuroscience
• Stages of CNN architecture
• Visualizing CNNs
• State-of-the-art results
• Packages

Many slides are by Rob Fergus and S. Lazebnik.
Traditional Recognition Approach

- Features are not learned
- Trainable classifier is often generic (e.g. SVM)
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?

Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007

Yan & Huang
(Winner of PASCAL 2010 classification competition)
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
Artificial neural networks

- Artificial neural network is a group of interconnected nodes
- Circles here represent artificial “neurons”
- Note the directed arrows (denoting the flow of information)
Inspiration: Neuron cells

• Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells
The basic unit of computation

“Peceptron”, Frank Rosenblatt 1957

Input

Weights

\[ x_1 \quad w_1 \]
\[ x_2 \quad w_2 \]
\[ x_3 \quad w_3 \]
\[ \vdots \]
\[ x_d \quad w_d \]

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

Sigmoid function:

\[ \sigma(t) = \frac{1}{1 + e^{-t}} \]
Non-linearity is important

- Without non-linearity, the whole system is linear
- Unfortunately, neural network research stagnated for decades after the publication by Minsky and Papert, 1969, who showed that a perceptron cannot represent the “xor” function
Training ANNs

“Chain rule” of gradient

\[ \frac{df(g(x))}{dx} = \left( \frac{df}{dg} \right) \left( \frac{dg}{dx} \right) \]

- **Back-propagate** the gradients to match the outputs
- Were too impractical till computers became faster

Issues with ANNs

• In the 1990s, simpler and faster learning methods such as SVMs and boosting were favored over ANNs.

• Why?
  • Need many layers to learn good features — many parameters need to be learned
  • Needs vast amounts of training data (related to the earlier point)
  • Convergence is slow, get stuck in local minima
  • Vanishing gradients for deep models
The **neocognitron**, by Fukushima (1980)
(But he didn’t propose a way to learn these models)
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

Convolutional Neural Networks

- 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
1. Convolution

- Dependencies are local
- Translation invariance
- Few parameters (filter weights)
- Stride can be greater than 1 (faster, less memory)
2. Non-Linearity

- Per-element (independent)
- Options:
  - Tanh
  - Sigmoid: $\frac{1}{1+\exp(-x)}$
  - Rectified linear unit (ReLU)
    - Simplifies backpropagation
    - Makes learning faster
    - Avoids saturation issues
      \[\rightarrow \text{ Preferred option}\]
3. Spatial Pooling

- Sum or max
- Non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (see more of input)
4. Normalization

- Within or across feature maps
- Before or after spatial pooling

Feature Maps

Feature Maps After Contrast Normalization
Compare: SIFT Descriptor

Image Pixels

Apply oriented filters

Spatial pool (Sum)

Normalize to unit length

Feature Vector

Lowe [IJCV 2004]
CNN successes

- 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]

- But until recently, less good at more complex datasets
  - Caltech-101/256 (few training examples)
ImageNet Challenge 2012

[Deng et al. CVPR 2009]

- 14+ million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- The challenge: 1.2 million training images, 1000 classes

ImageNet Challenge 2012

- 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
  - Better regularization for training (DropOut)

Krizhevsky et al. -- **16.4% error** (top-5)

Next best (SIFT + Fisher vectors) – **26.2% error**
Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form (6 · 6 · 256 = 9216 dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.
Layer 1 Filters

Similar to the filter banks used for texture recognition
Layer 1: Top-9 Patches
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
Layer 2: Top-9 Patches
Layer 3: Top-9 Patches
Layer 3: Top-9 Patches
Layer 4: Top-9 Patches
Layer 4: Top-9 Patches
Layer 5: Top-9 Patches
Layer 5: Top-9 Patches
Evolution of Features During Training
Evolution of Features During Training
Occlusion Experiment

- Mask parts of input with occluding square
- Monitor output (class probability)
Total activation in most active 5th layer feature map

Other activations from same feature map

True Label: Pomeranian
p(True class)  Most probable class

True Label: Car Wheel
Total activation in most active 5th layer feature map

Other activations from same feature map

True Label: Car Wheel

![Car Wheel Image]
p(True class)

Most probable class

True Label: Afghan Hound
Total activation in most active 5th layer feature map

Other activations from same feature map
ImageNet Classification 2013 Results


ImageNet 2014 - Test error at 0.07 (Google & Oxford groups)

http://image-net.org/challenges/LSVRC/2014/results
CNNs for small datasets

- Take model trained on ImageNet
- Take outputs of 6th or 7th layer before or after nonlinearity as features
- Train linear SVMs on these features (like retraining the last layer of the network)
- Optionally back-propagate: fine-tune features and/or classifier on new dataset
Tapping off features at each Layer

Plug features from each layer into linear SVM

<table>
<thead>
<tr>
<th></th>
<th>Cal-101 (30/class)</th>
<th>Cal-256 (60/class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (1)</td>
<td>44.8 ± 0.7</td>
<td>24.6 ± 0.4</td>
</tr>
<tr>
<td>SVM (2)</td>
<td>66.2 ± 0.5</td>
<td>39.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (3)</td>
<td>72.3 ± 0.4</td>
<td>46.0 ± 0.3</td>
</tr>
<tr>
<td>SVM (4)</td>
<td>76.6 ± 0.4</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>SVM (5)</td>
<td>86.2 ± 0.8</td>
<td>65.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (7)</td>
<td>85.5 ± 0.4</td>
<td>71.7 ± 0.2</td>
</tr>
</tbody>
</table>

Higher layers are better
## Results on benchmarks

### [1] Caltech-101 (30 samples per class)

<table>
<thead>
<tr>
<th>Method</th>
<th>DeCAF$_5$</th>
<th>DeCAF$_6$</th>
<th>DeCAF$_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogReg</td>
<td>63.29 ± 6.6</td>
<td>84.30 ± 1.6</td>
<td>84.87 ± 0.6</td>
</tr>
<tr>
<td>LogReg with Dropout</td>
<td>-</td>
<td>86.08 ± 0.8</td>
<td>85.68 ± 0.6</td>
</tr>
<tr>
<td>SVM</td>
<td>77.12 ± 1.1</td>
<td>84.77 ± 1.2</td>
<td>83.24 ± 1.2</td>
</tr>
<tr>
<td>SVM with Dropout</td>
<td>-</td>
<td>86.91 ± 0.7</td>
<td>85.51 ± 0.9</td>
</tr>
</tbody>
</table>

Yang et al. (2009) 84.3  
Jarrett et al. (2009) 65.5

### [1] SUN 397 dataset (DeCAF)

<table>
<thead>
<tr>
<th>Method</th>
<th>DeCAF$_6$</th>
<th>DeCAF$_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogReg</td>
<td>40.94 ± 0.3</td>
<td>40.84 ± 0.3</td>
</tr>
<tr>
<td>SVM</td>
<td>39.36 ± 0.3</td>
<td>40.66 ± 0.3</td>
</tr>
</tbody>
</table>

Xiao et al. (2010) 38.0

### [1] Caltech-UCSD Birds (DeCAF)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeCAF$_6$</td>
<td>58.75</td>
</tr>
<tr>
<td>DPD + DeCAF$_6$</td>
<td><strong>64.96</strong></td>
</tr>
<tr>
<td>DPD (Zhang et al., 2013)</td>
<td></td>
</tr>
<tr>
<td>POOF (Berg &amp; Belhumeur, 2013)</td>
<td></td>
</tr>
</tbody>
</table>

### [2] MIT-67 Indoor Scenes dataset (OverFeat)

<table>
<thead>
<tr>
<th>Method</th>
<th>mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROI + Gist[36]</td>
<td>26.05</td>
</tr>
<tr>
<td>DPM[30]</td>
<td>30.40</td>
</tr>
<tr>
<td>Object Bank[25]</td>
<td>37.60</td>
</tr>
<tr>
<td>RBow[31]</td>
<td>37.93</td>
</tr>
<tr>
<td>BoP[22]</td>
<td>46.10</td>
</tr>
<tr>
<td>miSVM[26]</td>
<td>46.40</td>
</tr>
<tr>
<td>D-Parts[40]</td>
<td>51.40</td>
</tr>
<tr>
<td>IFV[22]</td>
<td>60.77</td>
</tr>
<tr>
<td>MLrep[11]</td>
<td><strong>64.03</strong></td>
</tr>
<tr>
<td>CNN-SVM</td>
<td>58.44</td>
</tr>
</tbody>
</table>

---


R-CNN achieves mAP of 53.7% on PASCAL VOC 2010
For comparison, Uijlings et al. (2013) report 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach.
Part-based model with HOG (DPM, Poselets) \(~33.5\%\)

CNN features for face verification

Open-source CNN software

- **Cuda-convnet** (Alex Krizhevsky, Google)
  - High speed convolutions on the GPU
- **Caffe** (Y. Jia, Berkeley)
  - Replacement of deprecated **Decaf**
  - High performance CNNs
  - Flexible CPU/GPU computations
- **Overfeat** (NYU)
- **MatConvNet** (Andrea Vedaldi, Oxford)
  - An easy to use toolbox for CNNs from MATLAB
  - Comparable performance/features with Caffe