CAP 6412 Advanced Computer Vision

Website:

Jan 14, 2016
Today

- Administrivia
- Neural networks & backpropagation (Part I)
- Fundamentals of Convolutional Neural Networks (CNN), by Fareeha
Webcourse vs. Course homepage

- Webcourse: https://webcourses.ucf.edu/
- Announcements
  - Check your UCF email!
- Homework submission

- All the others
  - Lecture notes, papers, links to resources, syllabus, etc.
  - Bookmark and check regularly
Topics you have chosen

- Object recognition
- Low-level CV
- Visual saliency
- Machine learning for CV
- Human-centered CV
- 3D CV
- Vision and language
- Scene understanding
- Retrieval and matching
- Video: action, tracking, ...
## Tentative schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Week 2</strong></td>
<td><strong>CNN visualization &amp; object recognition</strong></td>
</tr>
<tr>
<td>Week 3</td>
<td>CNN &amp; object localization</td>
</tr>
<tr>
<td>Week 4</td>
<td>CNN &amp; transfer learning</td>
</tr>
<tr>
<td>Week 5</td>
<td>CNN &amp; segmentation, super-resolution</td>
</tr>
<tr>
<td>Week 6</td>
<td>CNN &amp; videos (optical flow, pose)</td>
</tr>
<tr>
<td>Week 7</td>
<td>Image captioning &amp; attention model</td>
</tr>
<tr>
<td>Week 8</td>
<td>Visual question answering</td>
</tr>
<tr>
<td>Week 9</td>
<td>Attention model, aligning books with movies</td>
</tr>
<tr>
<td>Week 10--16</td>
<td>Video: tracking, action, surveillance</td>
</tr>
<tr>
<td></td>
<td>Human-centered CV</td>
</tr>
<tr>
<td></td>
<td>3D CV</td>
</tr>
<tr>
<td></td>
<td>Low-level CV, etc.</td>
</tr>
</tbody>
</table>
# Next week:
CNN visualization & object recognition

## Sign up for presentations in CAP 6412

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Topics</td>
<td>Volunteer for class Tuesday</td>
<td>Volunteer for class Thursday</td>
</tr>
<tr>
<td>Week 2</td>
<td>CNN &amp; object recognition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 3</td>
<td>CNN &amp; object localization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 4</td>
<td>CNN &amp; transfer learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 5</td>
<td>CNN &amp; segmentation, super-resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 6</td>
<td>CNN &amp; videos (optical flow, pose)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 7</td>
<td>Image captioning &amp; attention model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 8</td>
<td>Visual question answering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 9</td>
<td>Attention model, aligningbooks with movies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week 10-16</td>
<td>TBD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Link will be sent to your UCF emails]
Today

• Administrivia
• Neural networks & backpropagation (Part I)
• Fundamentals of Convolutional Neural Networks (CNN), by Fareeha
Biological neurons

- Human brains has about 10 billion neurons
- Each connected to 10K other neurons
- A neuron fires if the sum of electrochemical inputs exceeds some threshold

Image credit: cs.stanford.edu/people/eroberts
Artificial neurons --- perceptrons

- Introduced by Rosenblatt in 1958
- The basic building blocks for (not all) neural networks

\[
y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right)
\]

\[
= \varphi\left(\mathbf{w}^T \mathbf{x} + b\right)
\]

\(\varphi(\cdot)\) : activation function

Image credit: www.hiit.fi/u/ahonkela/dippa/node41.html
Popular activation functions

\[ \varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \]

\[ \varphi(x) = \frac{1}{1 + \exp(-x)} \]

\[ \varphi(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \]

\[ \varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \]
Artificial neurons --- perceptrons

- Support Vector Machines
- Logistic regression
- AND
- OR
- NOT
- XOR?

- Linear regression

Image credit: www.hiit.fi/u/ahonkela/dippa/node41.html
Building neural networks from perceptrons

• Next Tuesday
Today

• Administrivia
• Neural networks & backpropagation (Part I)
• Fundamentals of Convolutional Neural Networks (CNN), by Fareeha
Convolutional Neural Networks

Fareeha Irfan
Outline

- Background
- Applications: Convnets for object recognition and language
- How to design convolutional layers
- How to design pooling layers
- How to integrate back-propagation in Convnets
- How to build convnets in torch
- AlexNet
Background

- Complex classification tasks
- Object Recognition in Images:
  - grayscale: $32 \times 32 = 1024$ pixels
  - rgb: $32 \times 32 \times 3 = 3072$ pixels
  - Fully-connected NN becomes computationally intensive

Algorithm that mimics the brain:

- Neural connections
- Neurons activated during learning
Convnet Applications

- **Image/Object Recognition**: Can predict who is in the image, what pose are they in.
- **Natural Language Processing**: Predict sentiments about sentences to classify tweets. Extract summaries by finding sentences that are most predictive.
- **Drug Discovery**: Predicting the interaction between molecules and biological proteins can be used to identify potential treatments.

Some Common Libraries:

- **Caffe**: Supports both CPU & GPU. Developed in C++
- **Torch framework**: Written in C
- **Cuda-convnet**: Implementation in CUDA
A Simple Neural Network

Activation Functions:
- Sigmoid
- Hyperbolic
- Tangent
- ReLU (Rectified Linear Unit)
Neural Network
Convnet Overview
Neural Network
Layer 1 (C1)
parameters:
\[(32*32+1)*(28*28+1)*6\]
\[= 4827750\]

ConvNet
Layer 1 (C1)
parameters:
\[(5*5+1)*6\]
\[= 156\]
Convolutional Layer

\[ y : \text{Output of the convolution} \]
\[ x : \text{Map with K channels} \]
\[ K' : \text{Total filters, generating a K' dimensional map } y \]

\[ y_{i'j'k'} = \sum_{ijk} w_{ijkk'} x_{i+i', j+j', k} \]
Back-propagation
Back-propagation for Conv Layer

\[ \delta^l_{i,j,f} = \sum_{i',j',f'} \delta^{l+1}_{i',j',f', \theta_{i-i'+1,j-j'+1,f,f'}} \]
A pooling operator operates on individual feature channels, coalescing nearby feature values into one by the application of a suitable operator.

Common choices include max-pooling (using the max operator) or sum-pooling (using summation).

Max-pooling is defined as:

\[ y_{i',j'} = \max_{ij \in \Omega(i'j')} x_{ij} \]
Pooling Layer

224x224x64

pool

112x112x64

downsampling

224

224

Single depth slice

1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4

max pool with 2x2 filters and stride 2

6 8
3 4
Convnet

Layer 1

Layer 2
model = nn.Sequential()
model.add(nn.Reshape(1,32,32))
-- layer 1:
model.add(nn.SpatialConvolution(1, 16, 5, 5))
model.add(nn.Tanh())
model.add(nn.SpatialMaxPooling(2, 2, 2, 2))
-- layer 2:
model.add(nn.SpatialConvolution(16, 128, 5, 5))
model.add(nn.Tanh())
model.add(nn.SpatialMaxPooling(2, 2, 2, 2))
-- layer 3, a simple 2-layer neural net:
model.add(nn.Reshape(128*5*5))
model.add(nn.Linear(128*5*5, 200))
model.add(nn.Tanh())
model.add(nn.Linear(200,10))
model.add(nn.LogSoftMax())
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky,
Ilya Sutskever,
Geoffrey E. Hinton
University of Toronto
Paper’s contribution

- Trained one of the largest CNNs on subsets of ImageNet
- Achieved the best reported results (by far)
- Publically available GPU implementation
- Describes unusual techniques to improve performance
Motivation

“Consider for example the problem of object recognition in computer vision: we could be interested in building recognizers for at least several thousand categories of objects. Should we have specialized algorithms for each?”

...“Are we going to have to do this labor-intensive work for all the possible types of [objects]? our system will not be very smart if we have to manually engineer new patches each time ... new types of object category must be processed. If there exist more general-purpose learning models, ... , then searching for them may save us a considerable amount of labor in the long run.”

15 million labeled images with 22,000 categories

Labeled by humans using...

Amazon’s Mechanical Turk crowdsourcing tool pay people to manually label large datasets
ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

Subset of the full ImageNet dataset

1.2 million training images
50,000 validation, 150,000 test images
1000 different classes!

~1000 images in each category (not an unreasonable number for MIA)
amazing Results to motivate why we learn about this approach

On test (unseen) data:

- top-1 error = 37.5% (prev best was 45.7%)
- top-5 error = 15.3% (second-best 26.2%)
Results
Results

test image

smallest Euclidean distance from the feature vector of the test image
Architecture

- 60 million parameters
- 650,000 neurons
- 5 convolutional layers (followed by max-pooling layers)
- 3 fully-connected layers with
- a 1000-way softmax final layer
Layer 1 (Convolutional)

- Images: 227x227x3
- F (receptive field size): 11
- S (stride) = 4
- Conv layer output: 55x55x96
Architecture

RELU Nonlinearity

• Standard way to model a neuron
  \[ f(x) = \tanh(x) \quad \text{or} \quad f(x) = \frac{1}{1 + e^{-x}} \]
  Very slow to train

• Non-saturating nonlinearity (RELU)
  \[ f(x) = \max(0, x) \]
  Quick to train
Architecture

RELU Nonlinearity

A 4 layer CNN with ReLUs (solid line) converges six times faster than an equivalent network with tanh neurons (dashed line) on CIFAR-10 dataset.
Local Response Normalization

After applying a kernel $i$ at position $(x,y)$, we get an activity $a$ for a neuron. We divide the neuron activity $a$ by the other activity of neurons in the other neighbouring kernel maps.

$\sum \text{ over } n \text{ neighbouring kernel maps}$

$$b_{x,y}^i = \frac{a_{x,y}^i}{\left( k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta}$$

- $N =$ total number of kernels per layer
- Example where $N = 3$
- hyper-parameters: $k = 2$, $\alpha = 10^{-4}$, $n = 5$, $\beta = 0.75$
  - found using the validation set
- Done to give a form of lateral inhibition inspired what is found in real neurons

Reduces error by about 1.3%
Local Pooling

Max

Summarizes the outputs of neighbouring groups

Traditionally, neighbourhoods don’t overlap

However, overlapping reduces error by ~0.4%
"The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 x 5 x 48"
Architecture Overview

- Full Connect: 4M
  - FULL 4096/ReLU: 16M
  - FULL 4096/ReLU: 37M
- Max Pooling: 442K
  - Conv 3x3/ReLU 256fm: 74M
- Conv 3x3/ReLU 384fm: 149M
- Max Pooling 2x2sub: 307K
  - Conv 11x11/ReLU 256fm: 223M
- Local Contrast Norm: 35K
  - Conv 11x11/ReLU 96fm: 105M
Reducing Overfitting

Data Augmentation

• 60 million parameters, 650,000 neurons
  → Overfits a lot.

• Crop 224x224 patches (and their horizontal reflections.)
Reducing Overfitting

Data Augmentation

• At test time, average the predictions on the 10 patches.
Reducing Overfitting

Data Augmentation

- Change the intensity of RGB channels

\[
I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T
\]

add multiples of principle components

\[
[p_1, p_2, p_3][\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3]^T
\]

\[
\alpha_i \sim N(0, 0.1)
\]
Dropout

- Set to 0 the output of a neuron with 0.5 probability
- Reduces complex co-adaptation and forces to learn more robust features
- Done in last two layers
Reducing Overfitting

**Dropout**

- With probability 0.5
- last two 4096 fully-connected layers.
Training

Using stochastic gradient descent and the backpropagation algorithm (repeated application of the chain rule)

One output unit per class

\[ x_i = \text{total input to output unit } i \]

\[ f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)} \]

We maximize the log-probability of the correct label, \( \log f(x_t) \)

Start with some initialized weights

Optimize so the correct label is predicted

Propagate errors back, and update weights to take a small step in the direction that minimizes the error

Reducing Overfitting

- Softmax

\[ L = \frac{1}{N} \sum_i -\log \left( \frac{e^{f_{yj}}}{\sum_j e^{f_j}} \right) + \lambda \sum_k \sum_l W_{k,l}^2 \]

\[ j = 1...1000 \]

\[ P(y_i \mid x_i; W) \] Likelihood
Stochastic Gradient Descent Learning

Updating the weights

**Momentum**: Adds a fraction of the previous weight update to the current one (increases speed of convergence)

\[
\begin{align*}
v_{i+1} & := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left( \frac{\partial L}{\partial w} \right)_{D_i}
w_{i+1} & := w_i + v_{i+1}
\end{align*}
\]

**Batch size** = how many examples in an iteration \(i\) to train the model on = 128 examples

**Weight decay**: penalizes large weights (as weight increases, changes less)

Average over the \(i\)th batch \(D_i\) of the derivative of the objective with respect to \(w\), evaluated at \(w_i\)
Tech Details - Preprocessing

An input image (256x256) Minus sign The mean input image
Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse coding [2]</td>
<td>47.1%</td>
<td>28.2%</td>
</tr>
<tr>
<td>SIFT + FVs [24]</td>
<td>45.7%</td>
<td>25.7%</td>
</tr>
<tr>
<td>CNN</td>
<td>37.5%</td>
<td>17.0%</td>
</tr>
</tbody>
</table>

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Also tried on ImageNet with ~10,000 categories and 8.9 million images
Top-1 = 67.4%, Top-5 = 40.9%
(prev best = 78.1% and 60.9%)

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 (val)</th>
<th>Top-5 (val)</th>
<th>Top-5 (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT + FVs [7]</td>
<td>---</td>
<td>---</td>
<td>26.2%</td>
</tr>
<tr>
<td>1 CNN</td>
<td>40.7%</td>
<td>18.2%</td>
<td>---</td>
</tr>
<tr>
<td>5 CNNs</td>
<td>38.1%</td>
<td>16.4%</td>
<td>16.4%</td>
</tr>
<tr>
<td>1 CNN*</td>
<td>39.0%</td>
<td>16.6%</td>
<td>---</td>
</tr>
<tr>
<td>7 CNNs*</td>
<td>36.7%</td>
<td>15.4%</td>
<td>15.3%</td>
</tr>
</tbody>
</table>

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.