Principles of Robot Autonomy II

Machine learning and modern visual recognition techniques
Today’s itinerary

• Stats/ML review

• Neural network basics

• Convolutional neural networks

• Robotic applications
Efficient feature extraction

If we know the input is image data, we can assume some spatial locality

⇒ weight sharing
Convolutional neural networks (CNN)

Traditionally consist of 4 types of layers:
• Convolutional layers (CONV)
• Nonlinearity layers (RELU)
• Pooling layers (POOL)
• Fully-connected layers (FC)

LeNet (1998)
Convolution layer

32x32x3 image

5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”
Convolution layer

32x32x3 image
5x5x3 filter $w$

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. $5*5*3 = 75$-dimensional dot product + bias)

$w^T x + b$
Convolution layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolution layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps
For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolution Layer Visualization

http://cs231n.github.io/convolutional-networks/
Feature hierarchy

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

As we move higher up the feature “food chain” we can save ourselves some computational effort by lowering the resolution.

Types of pooling:
- MAX pooling
- MEAN pooling
Fully connected layer

We’ve seen this one before!

Image “summary vector” with all of the redundant pixel info boiled out

Linear classifier (softmax)
Putting it all together – CNN

http://cs231n.stanford.edu/
Live Demo - Inner Workings of a CNN

http://scs.ryerson.ca/~aharley/vis/conv/

There’s also a 2D version:
http://scs.ryerson.ca/~aharley/vis/conv/flat.html
Classification showdown

\[ \nabla (f \circ g)(x) = ((Dg)(x))^T (\nabla f)(g(x)) \]

Who wins?
End-to-end learning wins!

Results

- **ILSVRC-2012 results**

  ![Error (5 predictions) graph]

  - **SuperVision**: Runner-up Top-5 error rate: 26.172%
  - **Oxford VGG**: Top-5 error rate: 16.422%

  Disclaimer: hand-crafted features may still be the right choice for your niche application

AlexNet (2012)
Modern architectures (deeper and deeper)

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 weight layers</td>
<td>19 weight layers</td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td>FC-4096</td>
</tr>
<tr>
<td>FC-4096</td>
<td>FC-1000</td>
</tr>
<tr>
<td>softmax</td>
<td></td>
</tr>
</tbody>
</table>

Modern architectures (deeper and deeper)

Inception-v3 (2016)
Today’s itinerary

• Stats/ML review
• Neural network basics
• Convolutional neural networks
• Robotic applications
Object localization and detection

Results from Faster R-CNN, Ren et al 2015
Object localization

Instead of outputting only a class (with associated loss function), also regress on 4 numbers defining the edges of a bounding box.

Input: image

Only one object, simpler than detection

Neural Net

Output:
Box coordinates (4 numbers)

Correct output:
box coordinates (4 numbers)

Loss:
L2 distance
Localization and detection

Instead of outputting only a class (with associated loss function), also regress on 4 numbers defining the edges of a bounding box.
Object detection

Sliding window: using a classifier as the basis for a detector

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores: P(cat) = 0.5
Object detection

Sliding window: using a classifier as the basis for a detector

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores: P(cat)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.75</td>
</tr>
</tbody>
</table>
Object detection

Sliding window: using a classifier as the basis for a detector

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores:
P(cat) = 0.5, 0.75, 0.6
Object detection

Sliding window: using a classifier as the basis for a detector

Network input: 3 x 221 x 221

Larger image: 3 x 257 x 257

Classification scores:
P(cat)

\[
\begin{array}{cc}
0.5 & 0.75 \\
0.6 & 0.8 \\
\end{array}
\]
Object detection – sliding window

Overfeat
(Sermanet et al. 2014)
Object detection – more efficient approaches

“Proposal” method to identify “blobby” regions of interest (could be another NN)

Two-headed classifier/bounding box regressor
Object detection – more efficient approaches

YOLO: You Only Look Once
Detection as Regression

Divide image into $S \times S$ grid

Within each grid cell predict:
- B Boxes: 4 coordinates + confidence
- Class scores: C numbers

Regression from image to
$7 \times 7 \times (5 \times B + C)$ tensor

Direct prediction using a CNN

Robotics – need for speed!

<table>
<thead>
<tr>
<th>Model Checkpoint</th>
<th>Million MACs</th>
<th>Million Parameters</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet_v1.0.224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
<td>89.5</td>
</tr>
<tr>
<td>MobileNet_v1.1.0.192</td>
<td>418</td>
<td>4.24</td>
<td>69.3</td>
<td>88.9</td>
</tr>
<tr>
<td>MobileNet_v1.1.0.160</td>
<td>291</td>
<td>4.24</td>
<td>67.2</td>
<td>87.5</td>
</tr>
<tr>
<td>MobileNet_v1.1.0.128</td>
<td>186</td>
<td>4.24</td>
<td>64.1</td>
<td>85.3</td>
</tr>
<tr>
<td>MobileNet_v1.0.75_128</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
<td>88.2</td>
</tr>
<tr>
<td>MobileNet_v1.0.75_192</td>
<td>233</td>
<td>2.59</td>
<td>67.4</td>
<td>87.3</td>
</tr>
<tr>
<td>MobileNet_v1.0.75_224</td>
<td>162</td>
<td>2.59</td>
<td>65.2</td>
<td>86.1</td>
</tr>
<tr>
<td>MobileNet_v1.0.75_256</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
<td>83.6</td>
</tr>
<tr>
<td>MobileNet_v1.0.50_224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
<td>85.4</td>
</tr>
<tr>
<td>MobileNet_v1.0.50_192</td>
<td>110</td>
<td>1.34</td>
<td>62.1</td>
<td>84.0</td>
</tr>
<tr>
<td>MobileNet_v1.0.50_160</td>
<td>77</td>
<td>1.34</td>
<td>59.9</td>
<td>82.5</td>
</tr>
<tr>
<td>MobileNet_v1.0.50_128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
<td>79.6</td>
</tr>
<tr>
<td>MobileNet_v1.0.25_224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
<td>75.0</td>
</tr>
<tr>
<td>MobileNet_v1.0.25_192</td>
<td>34</td>
<td>0.47</td>
<td>49.0</td>
<td>73.6</td>
</tr>
<tr>
<td>MobileNet_v1.0.25_160</td>
<td>21</td>
<td>0.47</td>
<td>46.0</td>
<td>70.7</td>
</tr>
<tr>
<td>MobileNet_v1.0.25_128</td>
<td>14</td>
<td>0.47</td>
<td>41.3</td>
<td>66.2</td>
</tr>
</tbody>
</table>

End-to-end: from pixels to motor commands

DAVE-2 (NVIDIA 2016)

Output: vehicle control
- Fully-connected layer
- Fully-connected layer
- Fully-connected layer

Convolutional feature map 64@1x19
Convolutional feature map 64@3x20
Convolutional feature map 48@5x22
Convolutional feature map 36@14x47
Convolutional feature map 24@31x98
Normalized input planes 3@66x200
Input planes 3@66x200

Somewhat less scary:
https://www.youtube.com/watch?v=HJ58dbd5g8g
Tools of the trade

• Software packages for automatic differentiation/gradient computation
  • Caffe (old)
  • Torch (old)
  • Theano (old)
  • TensorFlow (Google, Heavyweight #1)
  • PyTorch (Facebook, Heavyweight #2)
  • MXNet/Chainer/... (Others, better at some things for specific applications)

• Specify an abstract computation graph (inputs and outputs of NN equations); software does the rest!

TensorFlow: a *lot* of chain rule in this picture
Lots of stuff left out

- Generative vs. discriminative models
- Train/validation/test sets
- Learning rate and other hyperparameter tuning
- Recurrent neural networks for series data (e.g., videos)
- Reinforcement learning and ML outside of purely visual recognition-focused tasks

Consider STATS216, CS229, CS231n, CS224n, CS331b to learn more!
Next time