Convolutional Neural Nets

4-25-16
Reading Quiz

Convolutional neural networks are most commonly used for what sort of task?

a) Recognizing images
b) Transcribing speech
c) Translating documents
d) Playing games
Neural network review

Playground.tensorflow.org

Further reading on convolutional NNs: cs231n.github.io/convolutional-networks
Convolutional neural networks
- Hidden layer units connected to only a small subset of the previous layer.
- Connections have spatial locality (input from several nearby pixels).
- These hidden units “convolve” the input (like a blurring filter).

Deep belief networks
- Unsupervised pre-training of hidden layers (like the encoder example).
- Use weight reduction or smaller layers to avoid exact matching.
- Puts the backprop starting point in a good region of weight space.
Easy image recognition problem
Hard image recognition problem
We don’t want to use fully-connected layers

- 200x200 pixels, 3 color channels → 120,000 input neurons.
- The second layer has 120,000 weights for **EACH** neuron.
- We probably want many neurons in each hidden layer.
- We probably want many hidden layers.

- Recall that backpropagation can be interpreted as local search in the space of connection weights.
- This leads to an **EXTREMELY** high-dimensional search problem.
- The search will probably either perform poorly or overfit the training data.
Convolutional neural network layout

classical neural network

convolutional neural network
The architecture that won the ImageNet challenge in 2012 accepted images of size $[227x227x3]$. On the first Convolutional Layer, it used neurons with receptive field size $F=11$, stride $S=4$ and no zero padding $P=0$. Since $(227 - 11)/4 + 1 = 55$, and since the Conv layer had a depth of $K=96$, the Conv layer output volume had size $[55x55x96]$. Each of the $55*55*96$ neurons in this volume was connected to a region of size $[11x11x3]$ in the input volume. Moreover, all 96 neurons in each depth column are connected to the same $[11x11x3]$ region of the input, but of course with different weights.
Second key idea: shared weights for convolution

“Parameter sharing scheme is used in Convolutional Layers to control the number of parameters. Using the example above, we see that there are $55 \times 55 \times 96 = 290,400$ neurons in the first Conv Layer, and each has $11 \times 11 \times 3 = 363$ weights and 1 bias. Together, this adds up to $290400 \times 364 = 105,705,600$ parameters on the first layer of the ConvNet alone. Clearly, this number is very high.”
Second key idea: shared weights for convolution

“We can dramatically reduce the number of parameters by making one reasonable assumption: That if one patch feature is useful to compute at some spatial position \((x,y)\), then it should also be useful to compute at a different position \((x_2,y_2)\). In other words, denoting a single 2-dimensional slice of depth as a depth slice (e.g. a volume of size \([55 \times 55 \times 96]\) has 96 depth slices, each of size \([55 \times 55]\)), we are going to constrain the neurons in each depth slice to use the same weights and bias. With this parameter sharing scheme, the first Conv Layer in our example would now have only 96 unique set of weights (one for each depth slice), for a total of \(96 \times 11 \times 11 \times 3 = 34,848\) unique weights, or 34,944 parameters (+96 biases). Alternatively, all \(55 \times 55\) neurons in each depth slice will now be using the same parameters. In practice during backpropagation, every neuron in the volume will compute the gradient for its weights, but these gradients will be added up across each depth slice and only update a single set of weights per slice.”
Third key idea: pooling nodes for downsampling

Single depth slice

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

max pool with 2x2 filters and stride 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Example convolutional neural network

Layer 1
64x75x75

Layer 2
64x14x14

Layer 3
256x6x6

Layer 4
256x1x1

Output
101

Input
83x83

9x9 convolution
(64 kernels)

10x10 pooling,
5x5 subsampling

9x9 convolution
(4096 kernels)

6x6 pooling
4x4 subsamp

fully connected