Convnets in TensorFlow

CS 20SI:
TensorFlow for Deep Learning Research
Lecture 7
2/3/2017
Agenda

Playing with convolutions
Convolution support in TF
More MNIST!!!
Autoencoder

(Half) guest workshop by Nishith Khandwala
Understanding convolutions
Convolutions in maths and physics

a function derived from two given functions by integration that expresses how the shape of one is modified by the other
Convolutions in maths and physics
Convolutions in neural networks

a function derived from two given functions by element-wise multiplication that expresses how the value and shape of one is modified by the other
Convolutions in neural networks

We can use one single convolutional layer to modify a certain image
Convolutions in neural networks

We can use one single convolutional layer to modify a certain image

```
tf.nn.conv2d(input, filter, strides, padding,
use_cudnn_on_gpu=None, data_format=None, name=None)
```
Convolutions without training

Kernel for blurring

<table>
<thead>
<tr>
<th>0.0625</th>
<th>0.125</th>
<th>0.0625</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.25</td>
<td>0.125</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.125</td>
<td>0.0625</td>
</tr>
</tbody>
</table>

tf.nn.conv2d

input

output
Some basic kernels

See kernels.py and 07_basic_filters.py on the class GitHub!!!
Convolutions in neural networks

In training, we don’t specify kernels. We learn kernels!
Getting dimensions right

tf.nn.conv2d(input, filter, strides, padding,
use_cudnn_on_gpu=None, data_format=None, name=None)

Input: Batch size x Height x Width x Channels

Filter: Height x Width x Input Channels x Output Channels (e.g. [5, 5, 3, 64])
Strides: 4 element 1-D tensor, strides in each direction (often [1, 1, 1, 1] or [1, 2, 2, 1])

Padding: ‘SAME’ or ‘VALID’

Data_format: default to NHWC
Convnet with MNIST
Getting dimensions right

Original Image
28 x 28 x 1

Conv1
Filter: 5 x 5 x 1 x 32
Stride: 1, 1, 1, 1
Out: 28 x 28 x 32
Relu
Maxpool (2 x 2 x 1)
Out: 14 x 14 x 32

Conv2
Filter: 5 x 5 x 32 x 64
Stride: 1, 1, 1, 1
Out: 14 x 14 x 64
Relu
Maxpool (2 x 2 x 1)
Out: 7 x 7 x 64

Fully connected
W: 7*7*64 x 1024
Out: 1 x 1024
Relu
Out: 1 x 1024
Softmax
W: 1024 x 10
Out: 1 x 10

Softmax
1 x 10
Getting dimensions right

Original Image
28 x 28 x 1

Conv1
Filter: 5 x 5 x 1 x 32
Stride: 1, 1, 1, 1
Out: 28 x 28 x 32
Relu
Maxpool (2 x 2 x 1)
Out: 14 x 14 x 32

Conv2
Filter: 5 x 5 x 32 x 64
Stride: 1, 1, 1, 1
Out: 14 x 14 x 64
Relu
Maxpool (2 x 2 x 1)
Out: 7 x 7 x 64

Fully connected
W: 7*7*64 x 1024
Out: 1 x 1024
Relu
Out: 1 x 1024
Softmax
W: 1024 x 10
Out: 1 x 10

More exciting math in the lecture note!

\[(W−F+2P)/S+ 1\]

W: input width
F: filter width
P: padding
S: stride
TensorFlow support

- Convolution: `tf.nn.conv2d`
- Relu: `tf.nn.relu`
- Maxpool: `tf.nn.max_pool`
- Fully connected: `tf.nn.relu`
- Softmax: `tf.nn.softmax_cross_entropy_with_logits`
Variable scope

with tf.variable_scope('conv1') as scope:

    w = tf.get_variable('weights', [5, 5, 1, 32])
    b = tf.get_variable('biases', [32],
                        initializer=tf.random_normal_initializer())
    conv = tf.nn.conv2d(images, w, strides=[1, 1, 1, 1],
                        padding='SAME')
    conv1 = tf.nn.relu(conv + b, name=scope.name)
Interactive coding

Download 07_convnet_mnist_starter.py from GitHub!
MNIST Convnet graph
## Accuracy

<table>
<thead>
<tr>
<th>Epochs</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9111</td>
</tr>
<tr>
<td>2</td>
<td>0.9401</td>
</tr>
<tr>
<td>3</td>
<td>0.9494</td>
</tr>
<tr>
<td>5</td>
<td>0.9549</td>
</tr>
<tr>
<td>10</td>
<td>0.9692</td>
</tr>
<tr>
<td>25</td>
<td>0.9736</td>
</tr>
<tr>
<td>40</td>
<td>0.9793</td>
</tr>
<tr>
<td>50</td>
<td>0.9804</td>
</tr>
</tbody>
</table>
Autoencoder
Autoencoder

Input → Autoencoder → Reconstructed Input
Autoencoder

- Input and Output dimensions should match.
- Input and Output range should be same.
Autoencoder

Live coding
See autoencoder folder on GitHub
Next class

Guest lecture by Jon Shlens

Convnet

Deep Dream

Feedback: huyenn@stanford.edu

Thanks!