1 Convolutional Networks

By: Jussi Martin

1.1 Preface

This post is written for the AI Helsinki study group *Image and Video Statistics*. It is based on the Chapter 9 of the book *Deep Learning* by Ian Goodfellow, Yoshua Bengio and Aaron Courville, which is under preparation and available online at `http://www.deeplearningbook.org`.

We will assume the reader to be familiar with basic components of artificial neural networks, such as weights and biases. Some mathematical preliminaries are introduced in the text.

Our goal is to introduce basic concepts of the convolutional neural networks, explain where they are used and what kind of benefits the convolutional structure offers.

1.2 Definition

We begin by introducing the mathematical notion of convolution. Let $x(t)$ and $w(t)$ two functions then we define their convolution $x * w$ as

$$(x * w)(t) = \int x(a)w(t-a)da.$$

In practical applications x could be some signal depending on the time t and w a filter applied to it.

Obiously the data for which the convolution is applied usually is obtained only as a discrete input, in these cases we define the convolution by using summation:

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a).$$

Moreover, the data might be two dimensional, which is the case with images, then we use the follwing definition:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

where I and K represent two images and i and j are the pixel coordinates.

In practice the input grid is finite, the filter is defined on a smaller grid and usually only applied when its coordinates are fully in contained in the input grid These technical details will be covered soon.

We say that a neural network is convolutional if it uses convolutional filtering in at least one of its layers.

1.3 Benefits of Convolutional Networks

Convolutional networks are particularlly useful in speech recognition and image classification, since they can learn to be unsensitive to local translations in the input. For example, convolutional network can learn to classify different faces as been an image of a face even though individual faces have slight variation in the positions of different facial features.

Besides the classificational aspescts, convolution is also effective in the sence that it uses fewer connections and shared parameters, which reduces the computational cost and memory requirements. In deep networks this difference is significant.

We will examine sparse connectivity and parameter sharing later, after we have first introduced tensors and shown how differnt type of convolutions are represented with them.

1.4 Tensors

We adopt the Deep Learning book's convention of defining tensors as multidimensional arrays of real numbers.

Namely, 0-D tensors are just real numbers, 1-D tensors are arrays

$$T = (T_1, \dots, T_n)$$

of real numbers, 2-D tensors are arrays

$$T = ((T_{1,1}, \dots, T_{1,n}), \dots (T_{m,1}, \dots, T_{m,n}))$$

of arrays of real numbers (with mutually equal lenght), and so on.

Basically our tensors can be viewed as D-dimensional grids of real numbers. Example in 2-D:

$$((1,2,3),(4,5,6),(7,8,9)) \quad \simeq \quad \begin{array}{c|c|c} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}$$

In general case the sizes of the axes do not need to match.

1.5 Convolution via Tensors

*Stride
*Channels
*Formulas, etc.
(convolution with 1×1)... traditional fully connected layer can be seen as other extreme case, with zero stride and kernel grid size been equal to the input grid size.

1.6 Examples (submerge with previous section?)

(Pic of 2-D convolution, should be earlier?)

1.7 Sparse Connectivity and Parameter Sharing

As mentioned earlier, in a convolutional layer there are fewer connections between the nodes compared to fully connected layers and the same weights are used with multiple connections. We will now illuminate this with some visualizations and estimates.

Namely, only as many input nodes as there are weights in the convolution kernel are connected to every single node in the output nodes and vice versa.

Pic: 1-D, connections from inputs to single node in the outputs

Pic: 1-D, connections from single input to nodes in the outputs

This is in drastic contrast to fully connected layers where there are connections from every input node to a given output node and vice versa.

For example, if we have an image with 28×28 pixels grid, like the images in the MNIST data set have, and we convolve it with 3×3 filter. Then there are only $3 \cdot 3 = 9$ connections from input nodes to single output node, where as in fully connected layer there would be $28 \cdot 28 = 784$ connections to single output node. And similarly with roles of the input and output nodes reversed.

Here is an illustration of how the weights of the convolution kernel are shared between connections:

Pic: illustration of the parameter sharing

Sometimes even the biases are shared in similar manner, but this type of parameter sharing is not often used. Also here the difference to fully connected layer is significant. If fully connected layer has n input nodes and m output nodes, then there are $n \times m$ weight parameters in the layer, where as convolutional layer has only as meny weight parameters as the convolution kernel has.

1.8 Zero Pading

One drawback of using convolutional layers is that the layers shrink, which opposes limit to the depth of the network.

Pic: 1-D network with shrinking layers

This however can be compansated by adding zeros to the boundary of each neuron layer. That way we can prevent the sizes of the layers shrinking and obtain deeper networks.

Pic: 1-D network with zero pading and same size layers

The method discribed above is known as zero padding. One way to do it is to add just enough zeros that the layer sizes stays the same. Other more extreme option is to add so many zeros that the convolution can visit every neuron k times, where k is the size of the convolution kernel.

Pic: 2-D, original layer and layer with minimal pading

Pic: 2-D, original layer and layer with maximal pading

One may notice that zero pading makes the outputs near the boundary to have less information from the inputs, especially if there is maximal amount of pading used. This may be compensated to some extend if the biases are not shared (which usually is the case).

1.9 (Max) Pooling

Typically pooling is used for down sampling, which means that the grid size of the data is reduced.

$$\begin{array}{c|c|c|c} 0 & 1 & 3 & 9 \\ \hline 4 & 8 & 6 & 2 \\ \hline 7 & 4 & 2 & 0 \\ \hline 4 & 3 & 5 & 1 \end{array} \quad \mapsto \quad \begin{array}{c|c} 8 & 9 \\ \hline 7 & 5 \end{array}$$

Example in 2-D: max pooling with 2×2 kernel and stride = 2.

Another benefit of max pooling with down sampling is that it is not so sensitive to small translations. If we take to grids, which are small translations of each other, and compare the pooled grid, we see that many of the values are the same.

Pic: Without Shift, Shifted

1.10 Growing Receptive Fields (optional ?)

For a given node the nodes in the previous layers which are directly connected to it are known as receptive field.

Although the receptive field for single convolutional layer is small due to sparse connections, it grows when there are more layers in the network and it can even cover the whole input grid if the network has enough layers.

1.11 Convolutional Network Components

Typically: (convolution, Relu, pooling) x N_1, followed by FC x N_2

1.12 Example of Learned Invariances

Explanation (no pics)

1.13 Further Topics (Optional ?)

In this last section we will mention briefly some of the topics presented in the Chapter 9 of the Deep Learning book that we have not yet covered.

There are variants of the usual way of doing the convolution...
*Comparison of Local Connections, Convolution, and FC (Optional)
*Tiled Convolution (optional)
*Recurent Convolutional Networks

1.14 Links

* Our main source:
`http://www.deeplearningbook.org`
* Video with very basics of convolutional networks:
`https://www.youtube.com/watch?v=JiN9p5vWHDY`
* Video with slightly more advanced level introduction:
`https://www.youtube.com/watch?v=FmpDIaiMIeA`
* Video of a conference presentation:
`https://www.youtube.com/watch?v=AQirPKrAyDg`