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 basics of convolutional neural networks and explain some of their properties.

Convolutional networks are particularlly usefull in image classification, since they can learn to be unsensitive to local translations in the input. For example, convolutional network can learn to classify image of a face correctly dispite individual faces having slight variation in the positions of 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.

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 following 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 contained in the input grid. These technical details will be covered soon.

We say that a neural network is convolutional if it has at least one convolutional layer in it. This layer first applies convolutional filtering to its input data, after which the data is passed trough a nonlinearity and a pooling stage. Typically the layer can have several channels which apply different filters.

1.3 Tensors and Convolutional Stage

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

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

of real numbers, 2-D tensors are arrays

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

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.

1.4 (submerge to previous, when ready)

(Add to definition convolution without kernel fliping and explanation of why sum can be in the first function instead of last ?)

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

where $K_{i,l,m,n}$ is the weight of the connection from input in channel j to output in channel i, both having row index k and column index l. The output value has channel index i, row index j and column index k.
With stride:

$$Z_{i,j,k} = c(K, V, s) = \sum_{l,m,n} V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}$$

*Tiled (optional?)
*Local connections (optional?)
*Formulas, etc.
... 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.

1.5 Nonlinearities

Some of the typical nonlinearities used in convolutional networks are:

Rectified linear unit also known as *ReLu*, defined by

$$f(x) = \max(x, 0).$$

Logistic funtion also known as *sigmoid*, defined by

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

Hyperbolic tangent function also known as *tanh*, defined by

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}.$$

A nonlinearity is applied after convolutional stage in a convolutional layer. Its output is defined as

$$T_{i,j,k} = f(... + b_{i,j,k})$$

where ... is the output of the convolution and $b_{i,j,k}$ is the bias unit, which is often shared between units in the same channel, that is

$$b_{i,j,k} = b_i \quad \text{for all } j \text{ and } k$$

where j and k are the positional indicies and i is the channel index.

In some case however, for example if objects in a set of images are known to be in the center of the images, it may be more convinient to allow the biases to vary with respect to location.

1.6 (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

1.7 Zero Padding

One drawback of using convolutional layers is that they shrink size of the data grid, 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 padding

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

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

1.8 Receptive Field

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 added to the network and it can even cover the whole input grid if the network has enough layers.

1.9 Convolutional Networks Structure and Output

Typically the first layers in a convolutional network are convolutional. The final layers are fully connected and responsible for the one hot encoding of the output.

Optionally the output can be also...

*Output can be a real number (regression, is this without FC layers?)

*Output can be tensor of propabilities of member ship in a class. (This may need recurent structure in order to work well, check from the book!)

1.10 Learned Invariances and Feature Detectors

Examples, Explanation (no pics)

1.11 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)
*Training of the network
*Recurent Convolutional Networks

1.12 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`