Tutorial on Convolutions

Ronan Collobert

October 1, 2002

1 Sequence Convolutions

Three types of GradientMachine layers are provided to deal with sequences convolutions: TemporalConvolution, TemporalSubSampling and TemporalMean. These layers allow us to construct Time Delay Neural Networks[1]. We will have a look into the details of these layers below.

1.1 TemporalConvolution

Given an input sequence \((x_t)_{t=1...N}\), the output sequence \((y_t)_{t=1...M}\) of a TemporalConvolution layer will be:

\[ y^i_t = b_i + \sum_k \sum_j w_{i,j,k} x^j_{\delta \times (t-1)+k} \]

where \(K\) is the size of the kernel, \(\delta\) is the number of input frames incremented between each application of the kernel, and \(b_i\) and \(w_{i,j,k}\) are the weights of the layer. The size \(M\) of the output sequence is computed as follows:

\[ M = \frac{N - K}{\delta} + 1 \]

Thus, if \(\delta(M-1) + K < N\), the last frames of \((x_t)\) will not be taken in account (See Fig. 1). You could add some “dummy” frames in your input sequence to avoid this. Note that \(\delta \neq 1\) means that you are doing a convolution and sub-sampling. Creating such a convolution is quite easy: use the constructor

\[
\text{TemporalConvolution}(\text{int input\_frame\_size}, \text{int output\_frame\_size}, \text{int k\_w}=5, \text{int d\_t}=1);
\]

where \(\text{input\_frame\_size}\) corresponds to the frame size of \((x_t)\), \(\text{output\_frame\_size}\) corresponds to the frame size of \((y_t)\), \(k\_w\) is \(K\) and \(d\_t\) is \(\delta\).

1.2 TemporalSubSampling

For this layer, we have:

\[ y^i_t = b_i + \sum_k x^i_{\delta \times (t-1)+k} \]

Thus, the frame size of the input sequence is the same as the frame size of the output sequence. \(\delta\) is the “sub-sampling factor”. To create such a layer, just call:

\[
\text{TemporalSubSampling}(\text{int input\_frame\_size}, \text{int k\_w}=2, \text{int d\_t}=2);
\]

Same remarks apply concerning the size of input sequence as for TemporalConvolution.
1.3 TemporalMean

It computes the mean of the input sequence over time. It could be useful to do discrimination of sequences. We have:

\[ y_i = \frac{1}{N} \sum_s x_{s+i} \]

The output sequence of this machine is a sequence with one frame and which has the same frame size as input frames. The constructor is again pretty easy to evoke:

```
TemporalMean(int input_frame_size);
```

2 Image convolutions

Two layers are provided to deal with image convolutions: SpatialConvolution and SpatialSubSampling. With these layers, it’s easy to create networks similar to LeNet[2].

2.1 SpatialConvolution

Given \((x_k)_{k=1..N}\), \(N\) input images of size \(w_x \times h_x\), the output \((y_k)_{k=1..M}\) of this layer is given by \(M\) is the number of output images that you want):

\[ y_k^{(i,j)} = b_k + \sum_s \sum_{l=1}^{K} w_{k,l,s,t} x_{l-1+(\delta_w i + \delta_h j)} \]

where \(K\) is the size of the kernel, \(\delta_w\) and \(\delta_h\) are the number of pixels incremented between each application of the kernel onto the input image (over the width and the height), and \(b_l\) and \(w_{k,l,s,t}\) are the weights of the layer. The size \((w_y \times h_y)\) of all output images is computed as follow:

\[ w_y = \frac{w_x - K}{\delta_w} + 1 \]
\[ h_y = \frac{h_x - K}{\delta_h} + 1 \]
Thus, if $\delta_w(w_y-1) + K < w_y$ or $\delta_h(h_y-1) + K < h_y$, then the last columns or last lines of the input images will not be taken in account. You may add some “dummy” lines or columns in your input images to avoid this. Note that $\delta \neq 1$ means that you are doing a convolution and sub-sampling.

To construct such a layer, just use:

```
SpatialConvolution(int n_input_planes, int n_output_planes, 
                   int width, int height, int k_w=5, int d_x=1, int d_y=1);
```

where `n_input_planes` is $N$, `n_output_planes` is $M$, `width` is $w_x$, `height` is $h_x$, $k_w$ is $K$, $d_x$ is $\delta_w$ and $d_y$ is $\delta_h$.

The input frames given to the `SpatialConvolution`, when doing a `forward()`, is one big vector containing all images. You have to provide all rows of the first image (row after row), then all rows of the second image, and so on (See Fig. 2). Thus, the size of input frames is `n_input_planes*input_width*input_height`. The output format is the same. (And the size of output frames will be `n_output_planes*output_width*output_height`). Even if data is given in one vector, it is considered as 2D-images when doing convolution!

### 2.2 SpatialSubSampling

The output images are computed as follow:

$$y_k^{(i,j)} = b_k + c_f \sum_{s=1}^{K} \sum_{t=1}^{K} x_k^{(s(i-1)+s+\delta_y(j+t))}$$

Here, the number of input and output images are the same. Same remarks apply concerning the size of output images size as for `SpatialConvolution`. Creating such a layer is easily done by:

```
 SpatialSubSampling(int n_input_planes, int width, int height, 
                    int k_w=2, int d_x=2, int d_y=2);
```

When providing images to `SpatialSubSampling` with the `forward()` method, use the same format as for the `SpatialConvolution`.

---

1 As for `TemporalConvolution` where frames can be lost.
References
