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 contruct *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:

$$y_t^i = b_i + \sum_j \sum_{k=1}^{K} w_{i,j,k} x^j_{\delta \times (t-1)+k}$$

where K is the size of the kernel, δ 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

```
TemporalConvolution(int input_frame_size, int output_frame_size,
                                                int k_w_=5, int d_t_=1);
```

where `input_frame_size` corresponds to the frame size of (x_t), `output_frame_size` corresponds to the frame size of (y_t), `k_w` is K and `d_t` is δ.

1.2 TemporalSubSampling

For this layer, we have:

$$y_t^i = b_i + c_i \sum_{k=1}^{K} x^i_{\delta \times (t-1)+k}$$

Thus, the frame size of the input sequence is the same as the frame size of the ouput sequence. δ is the "sub-sampling factor". To create such a layer, just call:

```
TemporalSubSampling(int input_frame_size, int k_w_=2, int d_t_=2);
```

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

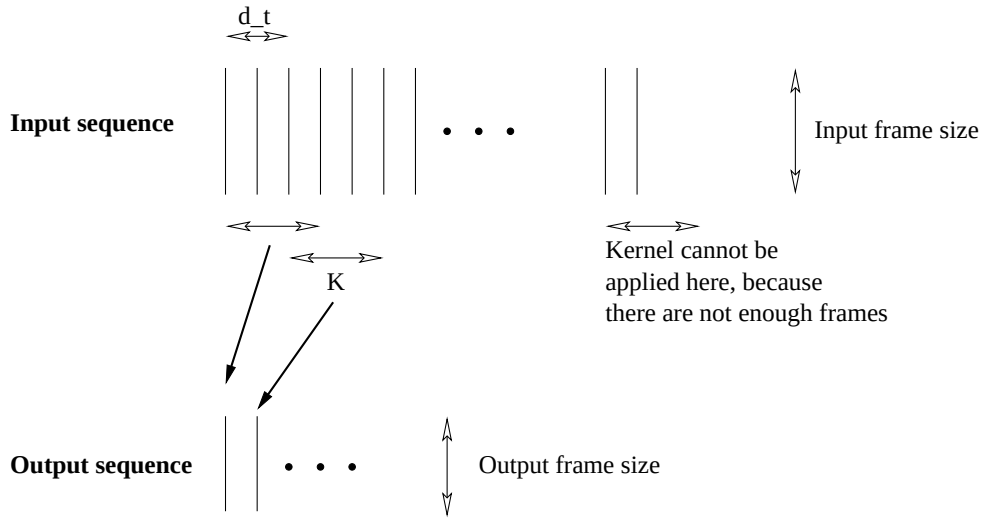

Figure 1: *Sequence convolution*

1.3 TemporalMean

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

$$y_t^i = \frac{1}{N} \sum_s x_s^i$$

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):

$$y_k^{(i,j)} = b_k + \sum_l \sum_{s=1}^{K} \sum_{t=1}^{K} w_{k,l,s,t} x_l^{(\delta_w (i-1)+s, \delta_h (j+t))}$$

where K is the size of the kernel, δ_w and δ_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_i 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 $\delta_w(w_y - 1) + K < w_x$ or $\delta_h(h_y - 1) + K < h_x$, then the last columns or last lines of the input images will not be taken in account[1]. 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 δ_w and `d_y` is δ_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

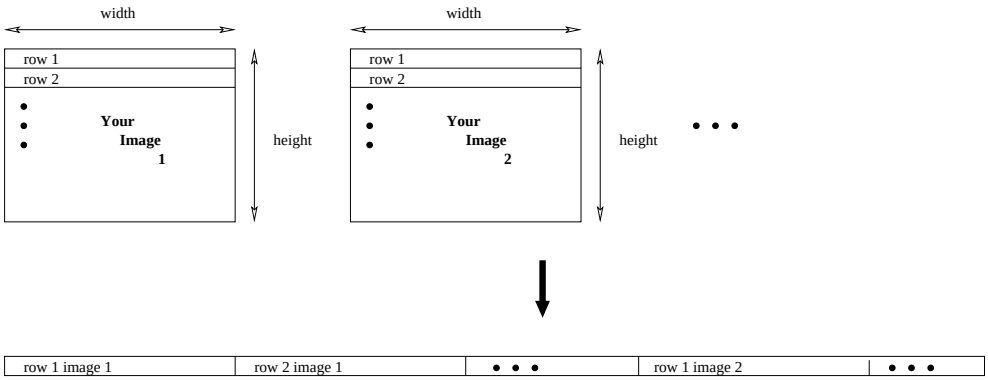

Figure 2: *Format of input and output frames when doing* spatial *convolution or sub-sampling.*

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:

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

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

[1] K. J. Lang and G. E. Hinton. The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University, 1988.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.