ECE 595: Homework 4

Rahul Deshmukh, PUID: 0030004932

(Fall 2019)

1 Convolution Operator

(b) The only difference between convolution and cross-correlation operator is that in convolution the kernel K is flipped. The output of convolution and cross-correlation operator on a 2D Image I is given by:

$$\text{Conv}(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{1}$$

$$= (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2}$$

$$\text{Cross-Correlation}(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{3}$$

From Eq (1) and Eq (2) we can observe that the sum defined by convolution operator remains the same even if we change the order of operands which means convolution is a commutative operator whereas the same is not true for cross-correlation.

(b) The cross-correlation operation is used in DNNs. This choice does not affects the DNNs as in the context of machine learning, the learning algorithm will learn the appropriate values of the kernel in the appropriate place, so an algorithm based on convolution with kernel flipping (ie true convolution) will learn a kernel that is flipped relative to the kernel learned by an algorithm without the flipping (ie cross-correlation). Also, convolution is used simultaneously with other functions and the combination of these functions does not commute regardless of whether the convolution operation flips its kernel or not.

2 Variants of the basic convolution function

(a) The major difference between convolution and locally connected layers is that while in convolution we have parameter sharing of weights of kernel to be the same for the entire image whereas in locally connected layers we still have a small kernel but the weights of the kernel as no longer constrained to be the same ie each kernel can have its own different weight and can learn different features at different spatial position. This is also illustrated in Figure 1.

(b) Pooling over spatial regions for convolution layers with parameter sharing helps in achieving translational equivariance due to convolution and then pooling helps in achieving translation invariance. This means that if we translate the input by a small amount, the values of the detector stage (ie convolution + activation) will also translate by the same amount however the values of

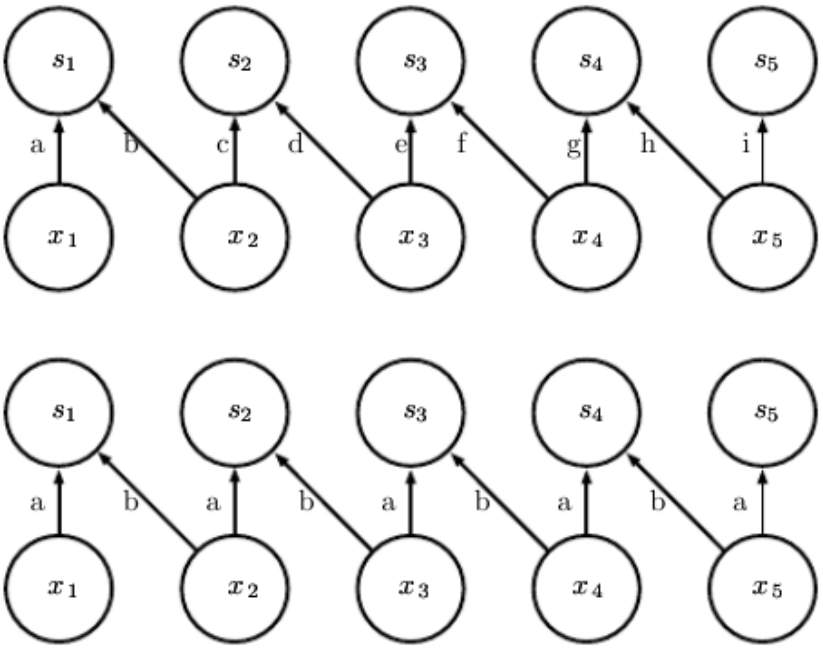

Figure 1: (source: [1]) Difference between convolution and locally connected layers. (Bottom) Regular convolution layer with with width of 2 pixels, notice that the convolution uses the same two weights repeatedly across entire input, as indicated by the repetition of the letters labelling each edge. (Top) Locally connected layers with width of 2 pixels, each edge is labeled with a unique letter to show that each edge is associated with its own weight parameter.

most of the pooled outputs do not change. For example, in the case of max pooling, a translation smaller than the pooling width will correspondingly shift the output of the individual detector units (prior to pooling) but after max pooling the output of majority of the units will remain the same.

3 Pooling

(a) A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. It can be done using max-pooling or average pooling where the summary is obtained by taking a max or an average of neighbouring outputs respectively.

(b) Advantages of pooling:

(i) Computation: Because pooling summarizes the response over a whole neighborhood, it is possible to use fewer pooling units that the detector units (convolution + NL-activation, by reporting the summary statistics for pooling regions spaced k pixels apart rather than 1 pixel apart. This improves the computational efficiency of the network because the next layer has roughly k times fewer inputs to process.

(ii) Generalization: Pooling is only useful when the assumption holds that the function the layer should learn contains only local interactions and is invariant to translation. However, it fails

to generalize when the task relies on preserving the precise spatial information like in the case of image alignment where we need sub-pixel accuracy for the estimate of location of a feature.

(c) Definitions:

(i) Pooling width: This is the number of detector units that the pooling function spans to get the summary statistics.

(ii) Stride: This controls how frequently the summary statistics is reported with respect to the detector units. For example a stride of 1 will report the summary at every detector unit location whereas a stride of 2 will report the stats at every consecutive detector unit.

(d) Pooling over spatial regions introduces translational invariance. This is because pooling summarises the output of a bunch of detector units, therefore if there is a translation smaller than the pooling width in the input data then the output of detector unit will also shift due to translation equivariance but the summary provided by the pooling unit will remain the same.

(e) To make the network learn the type of output invariance that pooling introduces, we can augment the original data with samples generated by translating the images in X and Y directions with small translations (1-3 pixels), this can be done by shifting the row and/or columns of the original image. Then with the augmented dataset we can ensure that the network learns translation invariance.

4 Zero Padding

(a) A down-sampled convolution is an operation which can be thought of as a down-sampling of the full convolution function, this can be achieved if we skip over some positions of the kernel to reduce the computational cost. The stride (s) of a down-sampled convolution defined as the frequency at which we draw the sample in a direction, ie a stride of 2 will draw only consecutive samples. The output of down-sampled convolution is given by:

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

(b) Without the zero-padding, with a input of size $m = 16$ and kernel width of $k = 6$ we will have only $m - k + 1$ units in the convolution layer. Therefore, for the given example we will have only 3 convolution layers of sizes $16 - 5 = 11$, $11 - 5 = 6$, $6 - 5 = 1$ starting from input layer to output layer respectively.

(c) The last convolution layer does not move the kernel and rather just summarizes the 6 units of second layer to single output. This is because our kernel size was also $k = 6$ therefore the input and kernel are of the same size and hence the kernel cannot move on the input.

(d) To increase the number of layers one can use smaller kernels but smaller kernels are less

expressive and also even with smaller kernel the number of layers will still be constrained due to the shrinking effect caused by convolution without zero-padding.

(e) To avoid the shrinking problem, zero-padding is employed where we add enough zeros to keep the size of the output equal to the size of the input. Thus, with zero-padding the network can contain as many convolution layers as available hardware can support.

(f) In MATLAB terminology, the name of the convolution operation that employs the zero-padding solution is called as **same convolution**.

5 Tiled Convolution

(a) Tiled convolution is a compromise between a convolution layer and a locally connected layer where rather than learning a separate set of weights at every spatial location, we learn a set of kernels that we rotate through as we move through space. This means that immediately neighbouring locations will have different filter, as in a locally connected layer, but the memory requirement for storing the parameters will increase only by a factor of the size of this set of kernels, rather than by the size of the entire output feature map.

(b) A tiled convolution with a set of t different choices in kernel stack in each direction becomes equivalent to:

- Normal Convolution: When $t = 1$,ie we have only one kernel in the stack thus the same as normal convolution.
- Locally Connected Layers: When t is equal to the output width.

6 Data Types

(a) Examples of data types for single-channel and multi-channel cases:

1. 1-D data type:
   (a) Single Channel: Audio waveform: we convolve over discretized time
   (b) Multi Channel: Skeleton animation data: each channel gives the information about the angle about one axis of one joint and we again convolve over discretized time.
2. 2-D data type:
   (a) Single Channel: Audi data with Fourier transform, we convert the audio wave into 2D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time.
   (b) Multi Channel: Color image data with one channel for individual color R,G & B. The convolution kernel moves over X & axis of the image

3. 3-D data type:

   (a) Single Channel: Volumetric data such as that obtained by CT scans

   (b) Multi Channel: Color Video data with three channels for colors and three dimensions for time, image X & Y. Another example can be Dual Energy CT scans which has two channels containing the information of materials mass density and atomic number.

(b) The use of convolution for processing variably sized inputs is suitable only for inputs that have variable size because they contain varying amounts of observation of the same kind of thing- for example, different lengths of recordings over time, different widths of observation over space etc.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.