Convolutional Neural Network
Theoretical Exercise

Fernando Labra Caso (ferlab-1)

April 4, 2023

Convolution

Initially we have the function defining the image as a signal and the function defining the kernel as matrices of sizes $n = 4$ and $k = 3$ respectively. When the kernel is shifted horizontally and vertically stays the same.

$$I_n = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & -3 & -4 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad K_k = K'_k = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

We want to calculate the padding for a stride of 2 that would allow a same convolution, resulting on a same size matrix. We consider the first equation for the calculation of the output size and suppose that by adding the padding to the size, the output should be the same as the original image.

$$o_{size} = \frac{n-k}{s} + 1 \qquad o_{size\_padding} = \frac{(n+p)-k}{s} + 1 = n$$

$$p + n - k + s = s * n \qquad p = s * n + k - n - s = 5$$

The resulted padding is an odd value, therefore we cannot distribute it equally around the signal matrix. The decision made was to have two zeros on the left and upper part and three zeros on the right as the kernel would be applied a bigger number of times on the signal matrix. In fact, the number of possible combinations would lead up to 36 when considering left/right up/down number of zeros for the padding (e.g L0/R5 with U0/D5, L0/R5 with U1/D4, etc.). The result of the discretized convolution can be seen on the right matrix where not many values have been captured.

$$I'_{k+p} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -3 & -4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (I'_{k+p} * K'_k) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 3 & 6 & 0 \\ 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Activation Function

The activation function chosen for applying non linearity to our problem is the Rectified Linear Unit or ReLU that gives the maximum between the value and 0, i.e. any negative value will be translated to 0. The result of applying ReLU to the convolution does not affect it as no negative values have been computed.

$$ReLU(X) = \forall i \in X,\ max(i, 0) \qquad ReLU(I'_{k+p} * K'_k) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 3 & 6 & 0 \\ 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Max Pooling

After applying the activation function it is common to apply a pooling method such as minimum, maximum or average in order to extract information (i.e. features) from the signal and reduce the dimensions. In this case, a valid MaxPooling (no padding) is applied with a filter and a stride of 2x2 resulting in a matrix of that same size.

$$MaxPooling_s(ReLU(I'_{k+p} * K'_k)) = \begin{bmatrix} 3 & 6 \\ 1 & 4 \end{bmatrix}$$

Flattening

Convolutional Neural Networks are used for extracting features of input signals such as images. When having a classification task, it is common to flatten the output of the convolutions once the intended complexity has been reached for using it as input to a fully connected ANN.

$$Flatten(MaxPooling_s(ReLU(I'_{k+p} * K'_k))) = \begin{bmatrix} 3 & 6 & 1 & 4 \end{bmatrix}^T$$

Fully Connected Layer

A fully connected layer performs a linear combination with coefficients the weight matrix W indicating the variable which is associated (i.e. the input feature). In this case, we have two input features and a fully connected layer of 4 neurons therefore a weight matrix of 2x4. The result of applying the learnt function of the layer with the result of the CNN is a matrix of size 2x1.

$$W = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix} \quad W * Flatten(MaxPooling_s(ReLU(I'_{k+p} * K'_k))) = \begin{bmatrix} 34 & 90 \end{bmatrix}^T$$

SoftMax

The softmax function is used for obtaining the distributed probability of a set of values that in this case is used on a classification task. The result when applying the function led to a probability in the order of 10^{-25} on the first of the values, so it was assumed to be 0. This means that the convolution that we have performed along with the following procedures has led to a clear decision (absolute probability) on the second class.

$$SoftMax(X) = \frac{e^{X_i}}{\sum_{j=1}^{K} e^{X_j}} \; for\ i = 1, ..., K\ and\ X = (x_1, ..., x_K) \in \Re^K$$

$$SoftMax(W * Flatten(MaxPooling_s(ReLU(I'_{k+p} * K'_k)))) = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$$