Theoretical Tasks

Often enough, Convolutional Neural Networks (CNN) are built by stacking blocks composed of a convolutional layer followed by a nonlinearity and by a max pooling layer. At the end, it is common to have a fully connected layer to classify the extracted features. Finally softmax operation is performed.
In the exercise below you have to calculate by hand each one of the steps.

Task 1.1 Convolution

Theoretical Background Read the following two blog posts:

- What are convolutions?
- Convolutions and Neural Network

Practice

Now consider the following image I, represented as a matrix I:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & -3 & -4 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

And the following kernel k, represented as a matrix k:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Calculate a same convolution $I \star k$ as described in Convolutions and Neural Networks above. Use zero padding for handling the margins. Since its a same convolution, use (1, 1) stride.
DO NOT use a computer to compute this operation, do it by hand - yes, I know, I know. . . but it takes you 5 minutes and it's good to do it at least once in life

Result:

First we add the padding to I:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 2 & 1 & 0 \\ 0 & 1 & -3 & -4 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then we slide k over I:
which give us for the first position:
O[1,1] = 1+1+2+1-3 = 4
and the second:
O[1,2] = 1+2+1+1 = 5
and the third:
O[1,3] = 1+2+1+2 = 6
and the forth:
O[1,4] = 1+2+1 = 4
That's it for the first line, since we have a stride of (1,1) the resultant matrix is of size (4,4), but we still need to continue for all the lines...At the end we get R, the resultant matix:

$$\begin{bmatrix} 4 & 5 & 6 & 4 \\ 5 & 3 & 3 & 6 \\ 1 & -7 & -7 & 0 \\ 4 & 1 & 0 & 4 \end{bmatrix}$$

Task 1.2 Non Linearity

Theoretical Background

In general, Convolutional layers are followed by a nonlinearity i.e. a non linear function is applied to their output. Without the nonlinearity all layers could be collapsed to a single matrix multiplication (non desirable). These functions play a major role affecting the learning (or non learning!) of neural networks. Common ones are sigmoid-like (sigmoid, tanh, softsign, . . .) and ReLU-like (ReLU, LeakyReLU, . . .).

Practice

Apply the ReLU activation function to the result of previous task.

Result:

Using ReLu on the matrix R mean getting rid of the value below 0 and will replace them by 0. Indeed, the ReLu activation function is define by:
f(x) = O if x<0 and f(x) = x if x>= 0
So we have R now equal to:

$$\begin{bmatrix} 4 & 5 & 6 & 4 \\ 5 & 3 & 3 & 6 \\ 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 4 \end{bmatrix}$$

Task 1.3 Max Pooling

Theoretical Background

Max Pooling divides the input image in several sections of a given size. This size is often referred to as subsample size, or filter size. Then, for each section, we only return the biggest value present in that section. For example in Figure 10 here the sections are of size (2, 2). It is easy to notice that the output image will have a smaller size than the input. In this case, because the filter size is (2, 2), the final size of the image half of the original image. If it were (3, 3), then the final size would be one third of the original image.

Practice

Apply valid Max Pooling with a filter size of (2, 2) on the result of previous task. Consider a stride of (2, 2). Notice that often enough if the stride is not provided it means it equal to the filter size (bad practice!)

Result:

Applying max pooling, with a filter of size (2,2), means dividing the matrix into new ones of size 2 by 2 and then keeping the highest of each sub-matrix. In our case we have 4 sub-matrix which are:

$$\begin{bmatrix} 4 & 5 \\ 5 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 4 \\ 3 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$$

Then, as explained, we keep from each sub-matrix the highest value and we end up with:

$$\begin{bmatrix} 5 & 6 \\ 4 & 4 \end{bmatrix}$$

Task 1.4 Flattening

Theoretical Background

Flattening is the process of reshaping a matrix into a one dimensional vector e.g by putting all the rows of the image in one same line. For example, the image below:
I =

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

will become:
I_flattened =

$$\begin{bmatrix} 0 & 1 & 2 & 3 \end{bmatrix}$$

Practice

Flatten the result of previous task.

Result:

When we flatten the matrix R we obtain :

$$\begin{bmatrix} 5 & 6 & 4 & 4 \end{bmatrix}$$

Task 1.5 Fully Connected Layer

Theoretical Background

After extracting the features a fully connected layer is used to perform classification. This is consisting in a simple matrix multiplication where there weight matrix of the layer has are as many rows as classes and as many columns as 2 features. For this reason the flattening operation the we just performed is very handy.

Practice

Calculate the output of a Fully Connected layer. Use the following W:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

Result:

We have:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

x

$$\begin{bmatrix} 5 \\ 6 \\ 4 \\ 4 \end{bmatrix}$$

=

$$\begin{bmatrix} 45 \\ 121 \end{bmatrix}$$

Task 1.6 SoftMax

Theoretical Background

Finally, a softmax operation is performed to transform the raw output of the network into probabilities. Eventually, the highest number after the softmax operation will be selected as output class.

Practice

Apply a softmax to the output of previous task and indicate which is the output class (1st or 2nd).

Result:

Applying softmax on the matrix R give us:

$$\begin{bmatrix} 2.473e-18 \\ 1.000 \end{bmatrix}$$

So the line one is basically 0 and the second line is 1, that means the second line has a higher probability than the first one.