Assignment #3

This assignment is due on Monday, November 14th 23:59:59 via email to christian.wallraven+NNF2022@gmail.com.

Important: You need to name your file properly. If you do not adhere to this naming convention, I may not be able to properly grade you!!!

If you are done with the assignment, make one zip-file of the assignment1 directory and call this zip-file `STUDENTID1_STUDENTID2_STUDENTID3_A3.zip` (e.g.: `2016010000_2017010001_A3.zip` for a team consisting of two students or `2016010000_2017010001_2017010002_A3.zip` for a three-student team if applicable). The order of the IDs does not matter, but the correctness of the IDs does! **Please double-check that the name of the file is correct!!**

Also: Please make sure to comment all code, so that I can understand what it does. Uncommented code will reduce your points!

Finally: please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!

Part1 Convolutions versus Fourier Transform (60 points):

First, make sure that the class notebook works properly. Take this notebook as inspiration and create a new notebook `convVsFT.ipynb`.

Use the following function to create some random data I [square with size N, typically large, added dimensions numimg to simulate different number of inputs] and filter F [square with size M, typically smaller than A, again added a dimension n_M to simulate different filters]. The output will be stored in a pre-allocated array O that has the same dimensions as A, but is extended by the number of inputs and filters.

```
def makeData(N=256, numimg=2, M=8, numfilt=3, seed=42):
    np.random.seed(seed)
    I = np.random.rand(N, N, numimg)
    F = np.random.rand(M, M, numfilt)
    O = np.zeros((N, N, numimg, numfilt))
 return I, F, O
```

The next ingredients we need are two functions that will compare the convolution operation of the filters with their counterpart using the fourier transform. For this, we will use scipy's convolve2d and fftconvolve functions.

Create two helper functions. The first is `c2d`:

```
def c2d(I,F,O):
    for [loop over all images]:
        for [loop over all filters]:
            O[:,:,im,fi]=convolve2d(img,fil,
mode='same', boundary='fill', fillvalue=0)
```

The second is `c2dfft`:

```
def c2dfft(I,F,O):
    for [loop over all images]:
        for [loop over all filters]:
            O[:,:,im,fi]=fftconvolve(img,fil,
mode='same')
```

Use suitable functions from `datetime` to time the execution of these two functions repeating them each at least 20 times!! and find out at which point the fourier method is "faster" as you increase the dimensionality of the filter M.

Make sure to call makeData outside of the timing code, and also make sure to change the seed to 20 different values to create 20 different datasets.

Make a nice plot (including error bars or shading!!) and carefully interpret the results.

Finally, tell me which ordering of the arrays is faster:

`N, N, numimg` and `M, M, numfilt`

or

`numimg, N, N` and `numfilt, M,M`

Although it looks like it shouldn't influence the results perhaps it does - change the code and compare the results, running the codes again with multiple number of comparisons and plotting!