Filtering and Convolution Exercise!

Prof. Alex Berg, bergac@uci.edu

Winter 2024

There are many ways to think about and learn filtering and convolution. Here we will try to focus on computations that are performed on pixel grids and leave some of the more mathematical background for other courses (signal processing, linear algebra, etc). There are seven exercise questions below, 5 to work out on paper and photo/scan/otherwise turn in a pdf on canvas. There is also a short implementation practice that you should do, but no need to turn in, and an extra credit question. If you want to do the extra credit, please include your answer in the pdf.

1 Filtering

As a start, a filter $f(i,j)$ has a value for $i \in \{0 \ldots n\}$ and $j \in \{0 \ldots m\}$. Note it is a little easier on notation to start with 0 for those indexes, but is not necessary. Similarly an image $I(a,b)$ has a value for each location (a,b). We *apply* the filter at a location (a,b) in an image by computing

$$(f \otimes I)(a,b) = \sum_{i,j} f(i,j) I(a+i, b+j)$$

Here the sum is over the range of i and j as stated earlier. Doing this for all the (a,b) gives us the output image of filtering I by f. We also call this applying the filter f to I.

2 Convolution

For convolution we apply the filter flipped left to right and top to bottom so:

$$(f * I)(a,b) = \sum_{i,j} f(i,j) I(a-i, b-j)$$

Note that here we are using the convolution symbol $*$ instead of the filtering symbol $\otimes$.

3 Filtering is linear transform and a dot product!

Looking at the formula above for $(f * I)(a,b)$ we can think of the $f(i,j)$ as coefficients and the entries $I(a+i, b+j)$ as variables. This helps to see that filtering is a linear transformation of the image. We can also think of $(f * I)(a,b)$ as the inner product (dot product) of two vectors. One vector consists of the entries in f (arranged as a vector) and the other vector consists of the corresponding entries of $I(a-i, b-j)$.

4 Separable filter

A separable filter has a special form that lets us write it as the outer product of two vectors. A filter $f(i,j)$ is separable if there are two filters v and w so that $f(i,j) = v(i,1)w(j,1)$. If we think of the filter f as a matrix F with entries $F[i,j] = f(i,j)$, then it is separable if there are vectors v and w so that $F = vw^T$.

5 Questions

1. Verify that if you have an image I with all zeroes except for a single location with value 1, that $(f * I)(a,b) = f$. Try it for a 3x3 filter f and a 5x5 image I with a 1 in the middle and zeros everywhere else. Note that the result $(f * I)(a,b)$ might be a bit offset from $f(a,b)$ depending on your indexing.

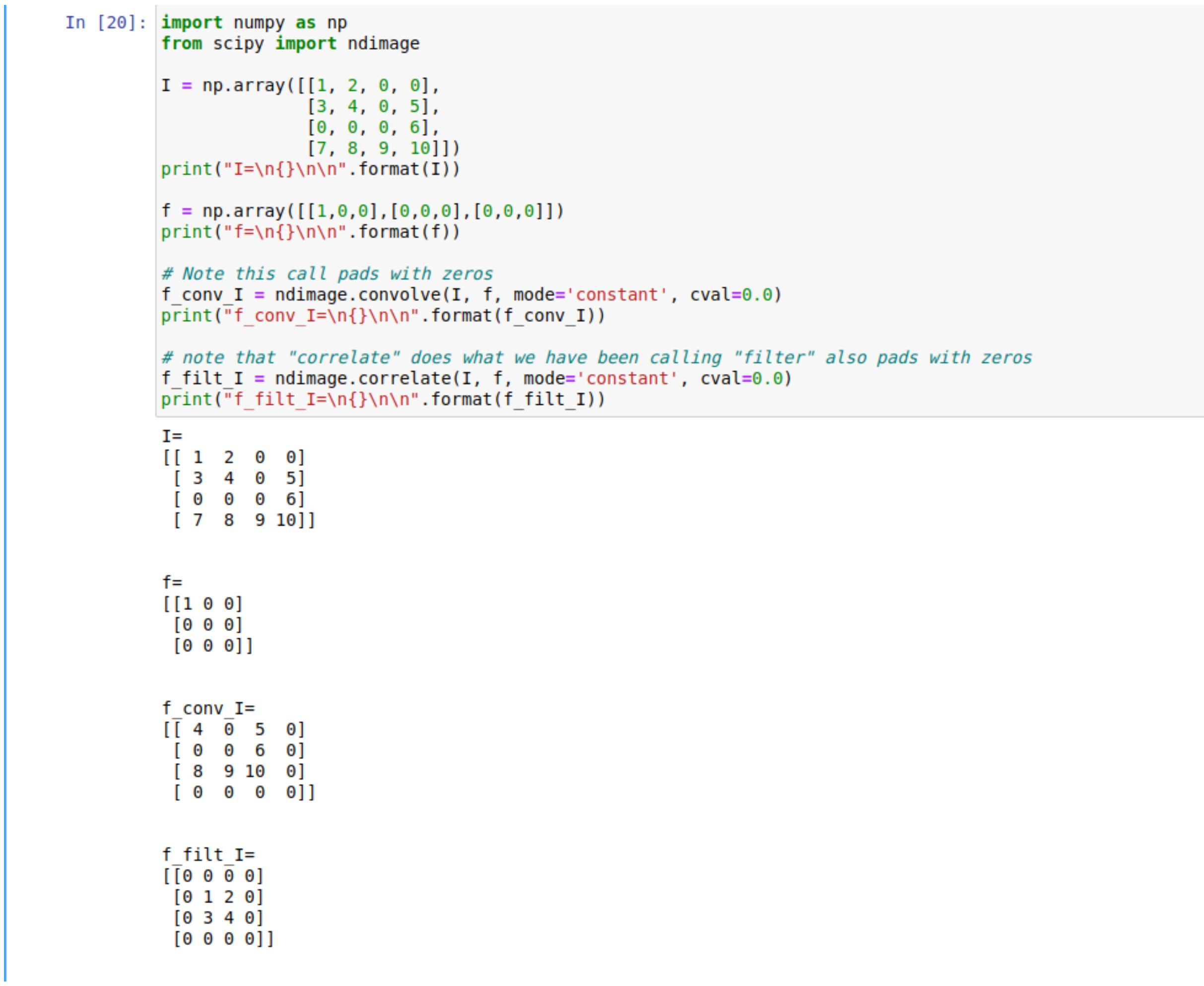
2. Verify that for the same filter and image $(f \otimes I)(a,b)$ is f flipped top to bottom and side to side.

3. (*fixed*) Show that the filter $F = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ is separable by finding s v and w so that $F = vw^T$. Then show that the filter $F = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ is **not** separable by showing that you cannot find s v and w so that $F = vw^T$.

4. Check that $F \otimes I = v \otimes (w^T \otimes I)$.

5. How many operations (multiplications and additions) does it take to apply the filter F in the previous question with a 10x10 image I (how many operations to compute the image $F \otimes I$? How many operations for $v \otimes (w^T \otimes I)$? It should be fewer!

6. Implement filtering in python and see if you can get the same result as the built in function (at least for the cases illustrated here)



```
In [20]: import numpy as np
         from scipy import ndimage

         I = np.array([[1, 2, 0, 0],
                       [3, 4, 0, 5],
                       [0, 0, 0, 6],
                       [7, 8, 9, 10]])
         print("I=\n{}\n\n".format(I))

         f = np.array([[1,0,0],[0,0,0],[0,0,0]])
         print("f=\n{}\n\n".format(f))

         # Note this call pads with zeros
         f_conv_I = ndimage.convolve(I, f, mode='constant', cval=0.0)
         print("f_conv_I=\n{}\n\n".format(f_conv_I))

         # note that "correlate" does what we have been calling "filter" also pads with zeros
         f_filt_I = ndimage.correlate(I, f, mode='constant', cval=0.0)
         print("f_filt_I=\n{}\n\n".format(f_filt_I))
```

```
I=
[[ 1  2  0  0]
 [ 3  4  0  5]
 [ 0  0  0  6]
 [ 7  8  9 10]]


f=
[[1 0 0]
 [0 0 0]
 [0 0 0]]


f_conv_I=
[[ 4  0  5  0]
 [ 0  0  6  0]
 [ 8  9 10  0]
 [ 0  0  0  0]]


f_filt_I=
[[0 0 0 0]
 [0 1 2 0]
 [0 3 4 0]
 [0 0 0 0]]
```

No need to include your code, just answer ”

7. hints online, but cite your source if you do). How can you use convolution to compute normalized inner product between a filter and each window in the image I? This is not linear so there need to be some tricks. As a hint, you can do some operations by applying a filter to I and then make another image $I2(i, j) = I(i, j)^2$ where each entry is the square of the corresponding entry in I and apply some filters to that. Then do some operations to combine the results.