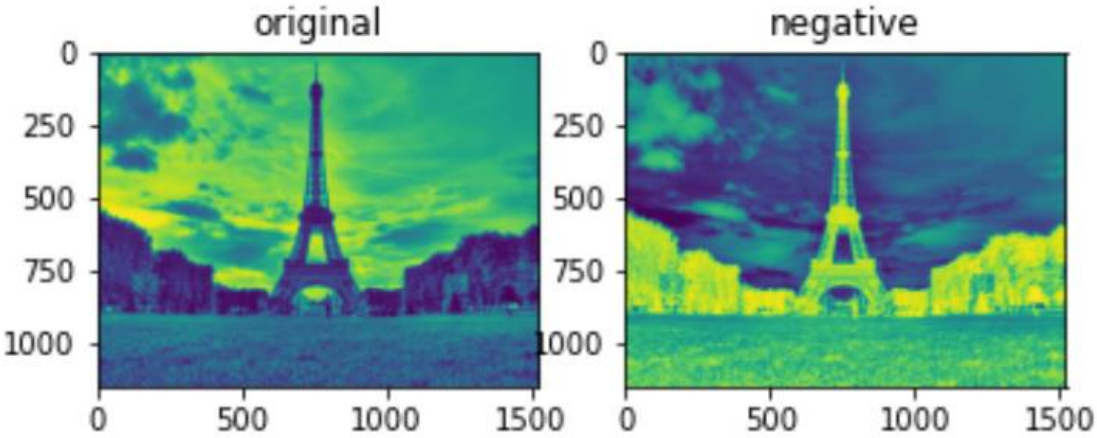
Task-1:

- a = np.array([[2, 3, 4],[5, 2, 200]]) **,** Creates a two dimensional array, 2 rows & 3 columns
- b = a[: , 1] **,** Selects elements from all rows but only from the second column
- f = np.random.randn(400,1)+3**,** generates an aray of the provided shape filled with values sampled from the normal distribution
- g = f[f>0]*3, g stores all the values of f that are greater than 0, after multiplying them by 3
- x = np.zeros(100)+ 0.45 **,** creates an one dimensional array of 100 elements all having value 0, After that 0.45 is added to each element.
- y = 0.5 * np.ones([1, len(x)]) **,** creates an one dimensional array of 100 elements all having value 1, After that 0.5 is multiplied to each element.
- z = x + y **,** element wise addition of two single dimension arrays.
- a = np.linspace(1,499, 250, dtype=int) **,** returns 250 numbers equally spaced between 1 and 499, including both.
- b = a[: :-2] **,** stores the array from last elemnt upto 2nd element including both
- b[b >50]=0 **,** it modifies the elements in b, just the element which is >50 it replaces it by 0.

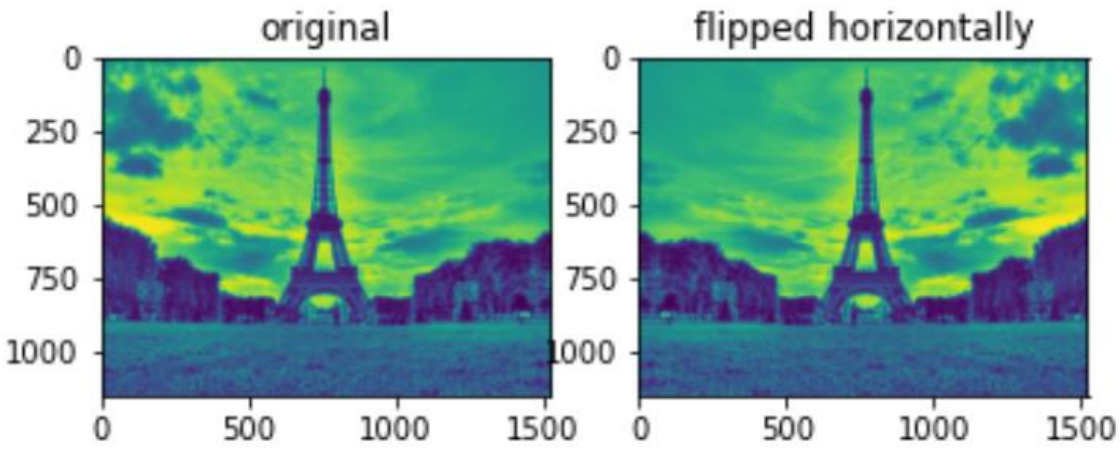
Task-2:

1)



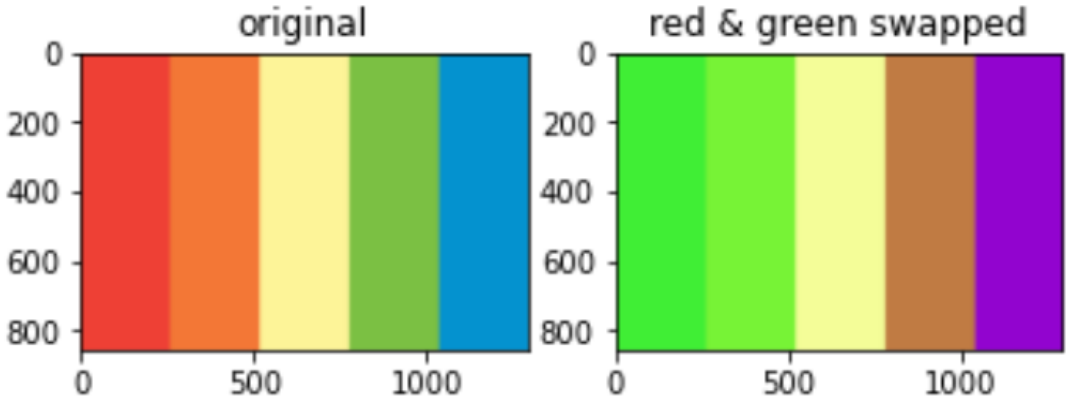
Just by subtracting the grayscale img by 255 the output is obtained

2)



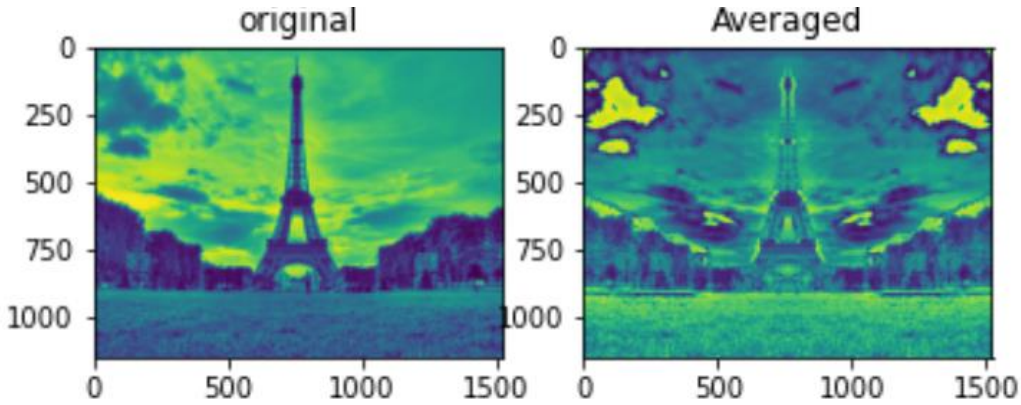
Obtained by interchanging the columns with respect to vertical axis.

3)



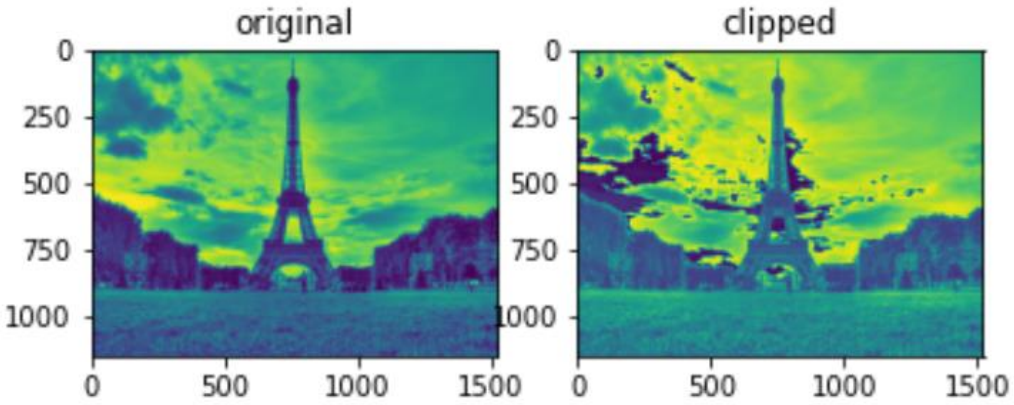
Swapping the first two 3rd dimension and keeping all the 2d values same.

4)



Added original and horizontally flipped and divided by 2.

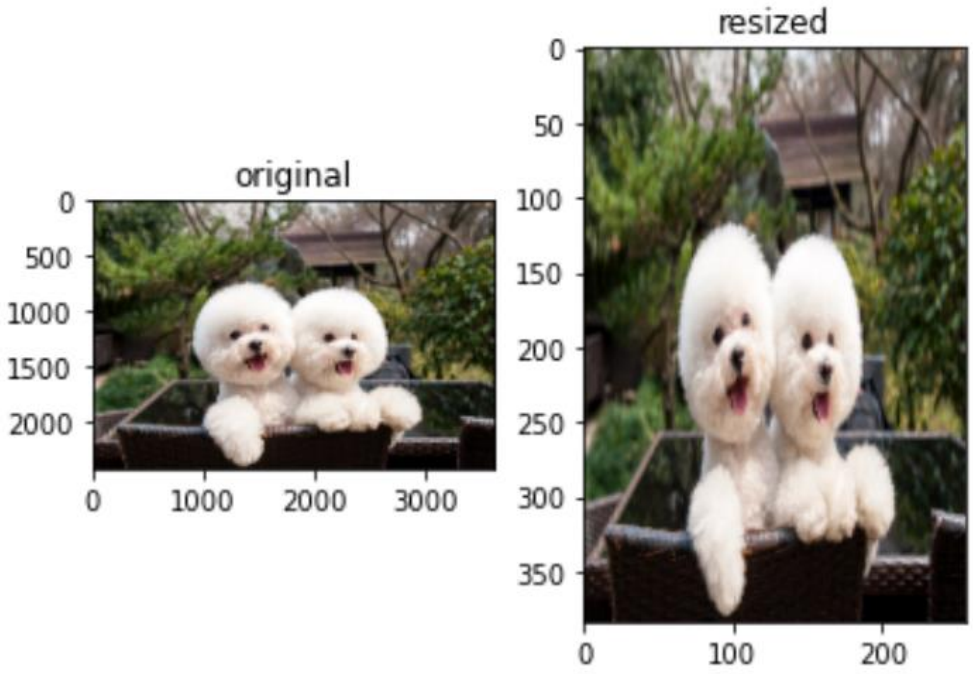
5)



Added a random value and used np.clip

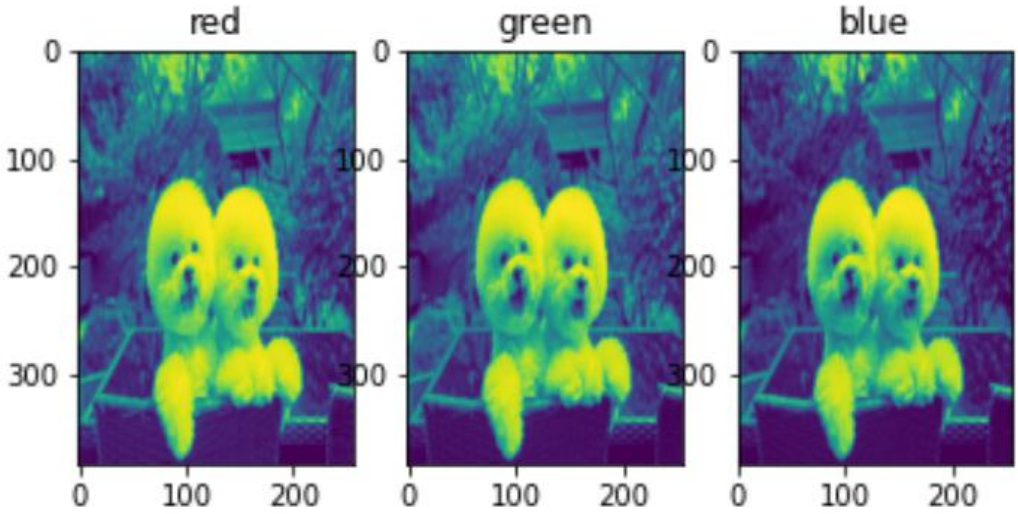
Task-3:

a)



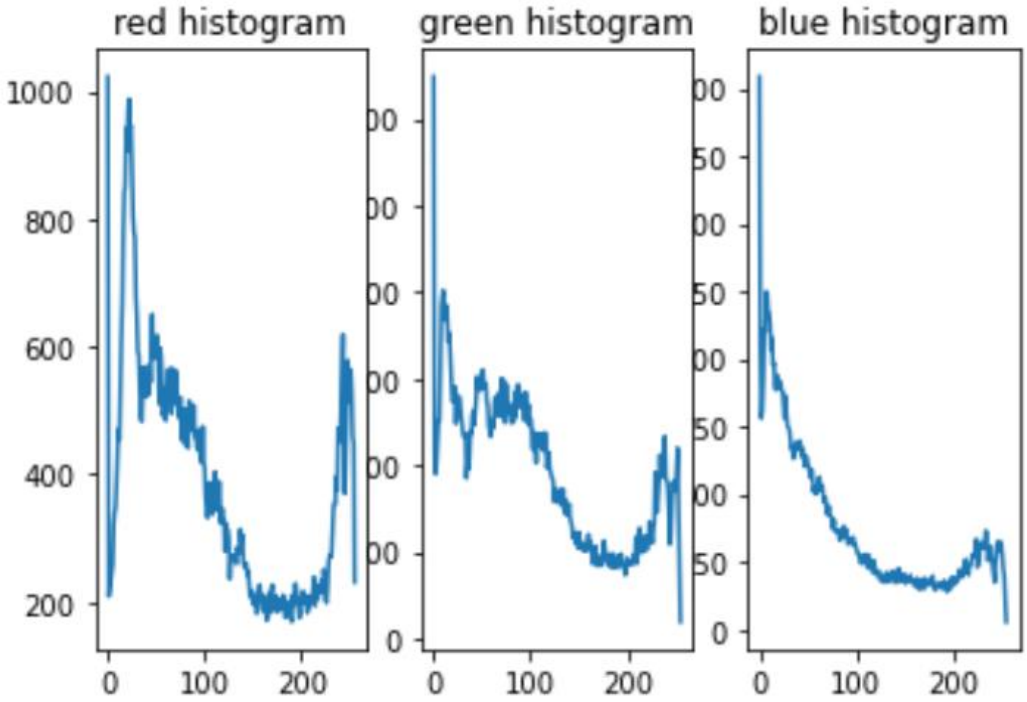
Resized using cv2.resize

b)



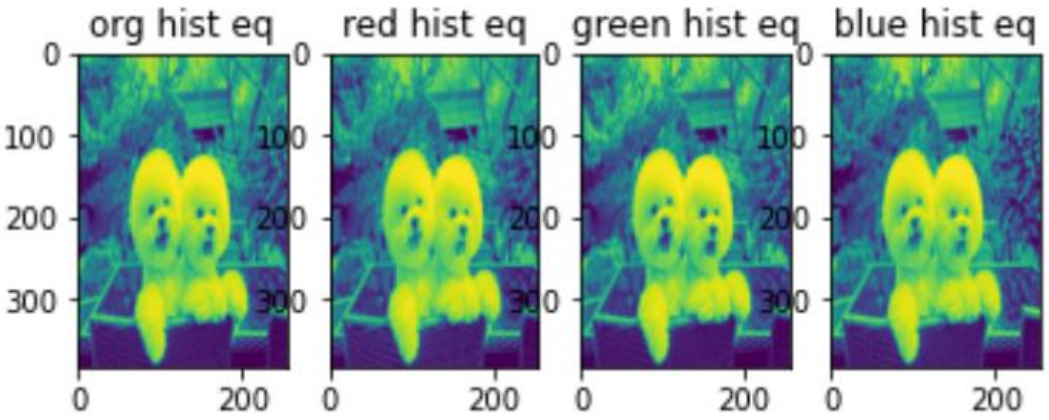
Showing different channels using cv2.split

c)



Showing the histograms using cv2.calcHist and plt.plot

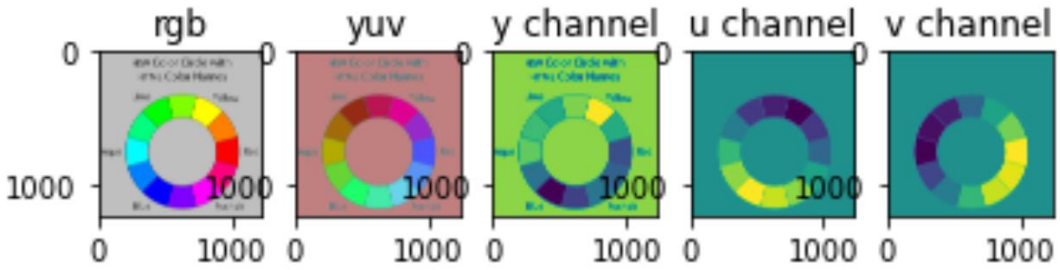
d)



Histogram equalisation calculated using cv2.equalizeHist

Task-4:

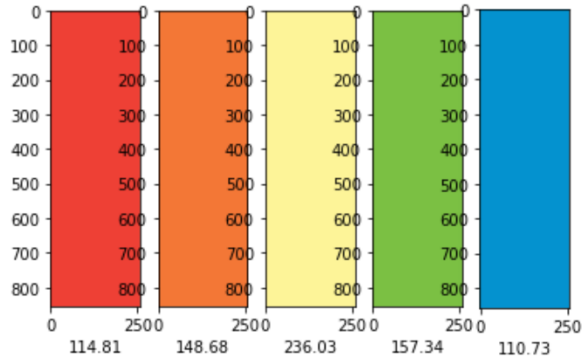
1)



theory referred from SzeliskiBookDraft_20210214

2)

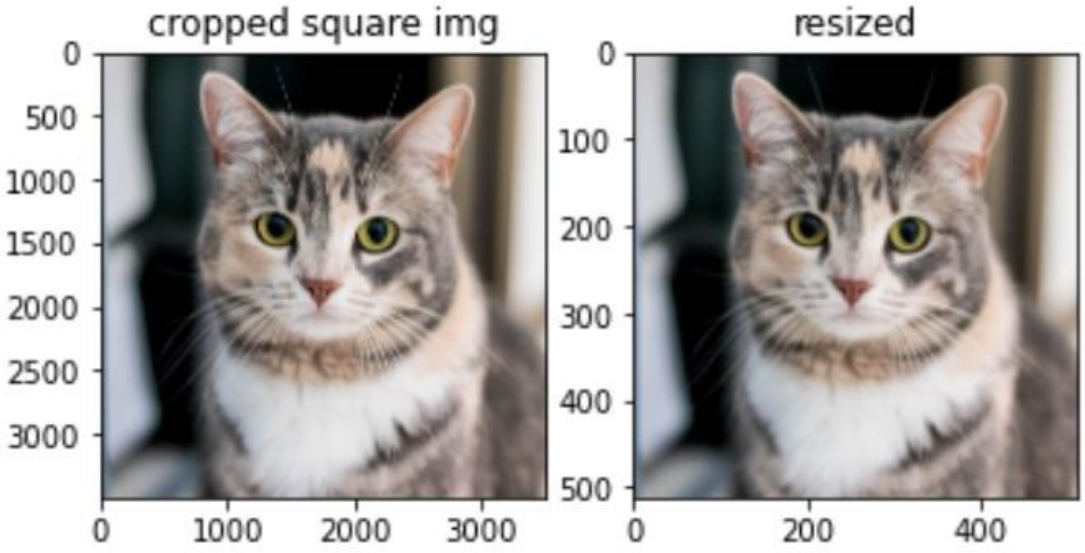
```
Avg. y value for my function (region 1) is 114.80956201550381
Avg. y value for cv2 function (region 1) is 114.80956201550381

Avg. y value for my function (region 2) is 148.68355728926394
Avg. y value for cv2 function (region 2) is 148.68355728926394

Avg. y value for my function (region 3) is 236.0261604388845
Avg. y value for cv2 function (region 3) is 236.0261604388845

Avg. y value for my function (region 4) is 157.34320373938291
Avg. y value for cv2 function (region 4) is 157.34320373938291

Avg. y value for my function (region 5) is 110.73144630395603
Avg. y value for cv2 function (region 5) is 110.73144630395603

Since the average y value of my function and cv2 matche, I am gonna show the avg. to my function only
```



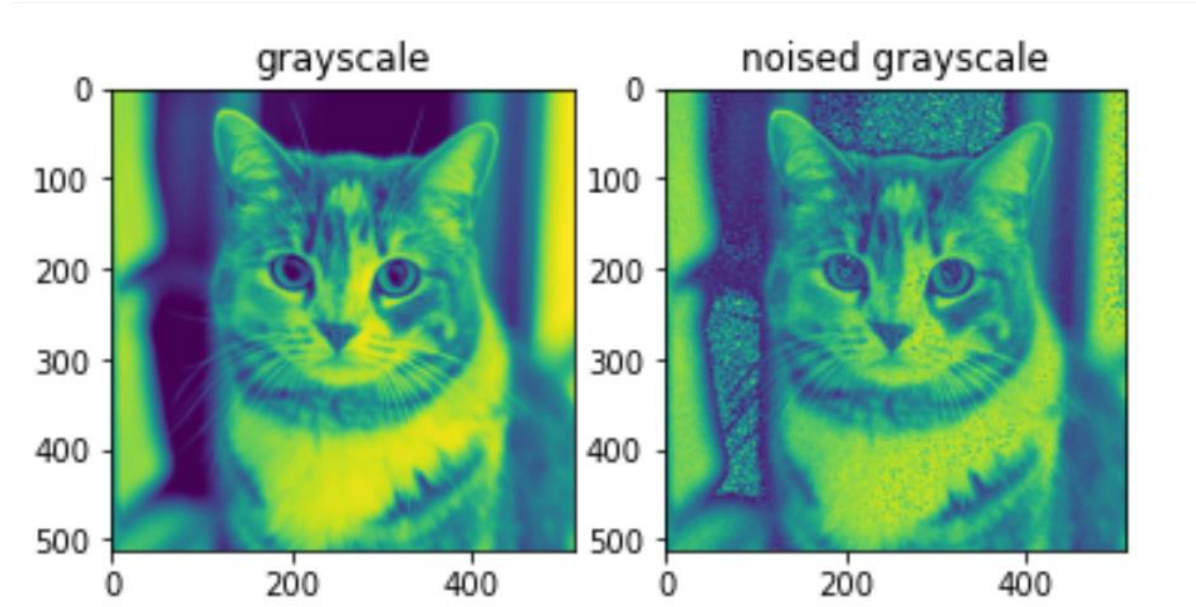
Used concept that the five regions are equally spaced therefore divided the width by 5 and created separate regions accordingly.

Task-5:

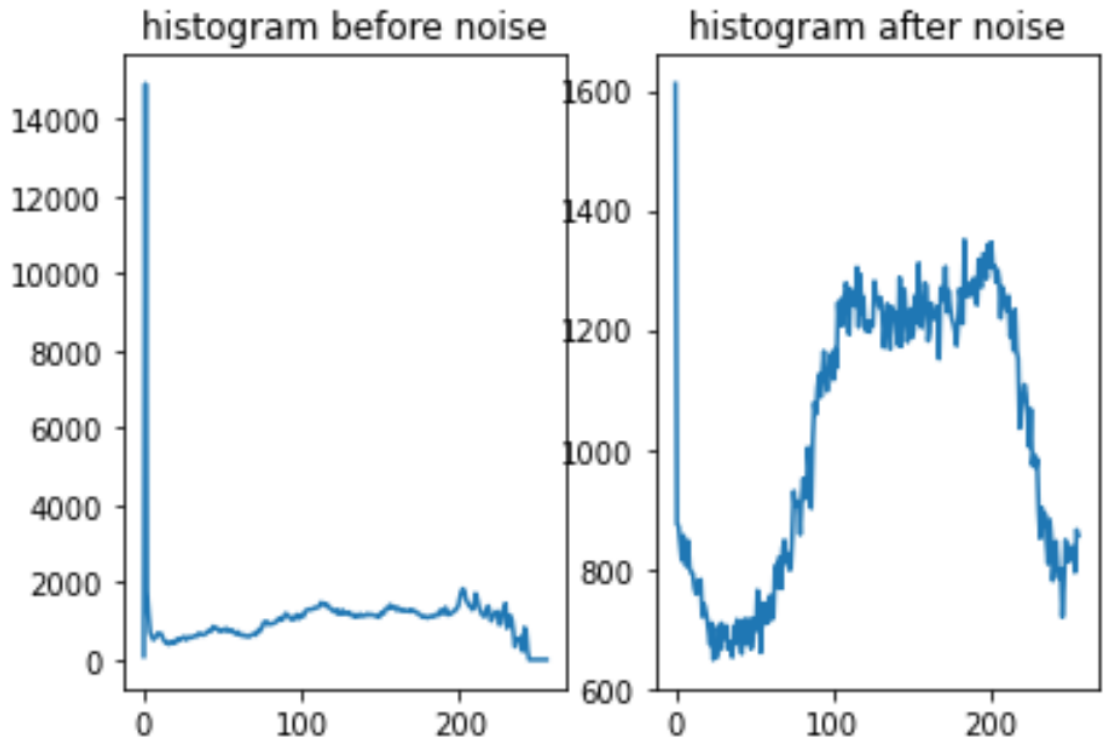
1)



Original image cropped and resized.



Converted to grayscale and added noise



Plotted the histogram of two grayscale images

2)

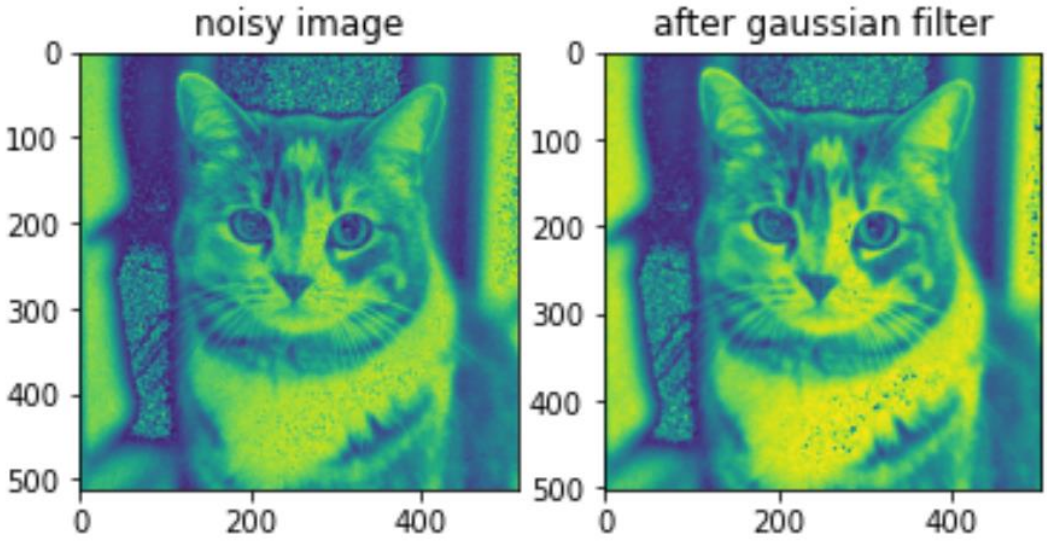
my_Gauss_filter()

input: *noisy_image, my 5x5 gausskernel*

output: *output_image*

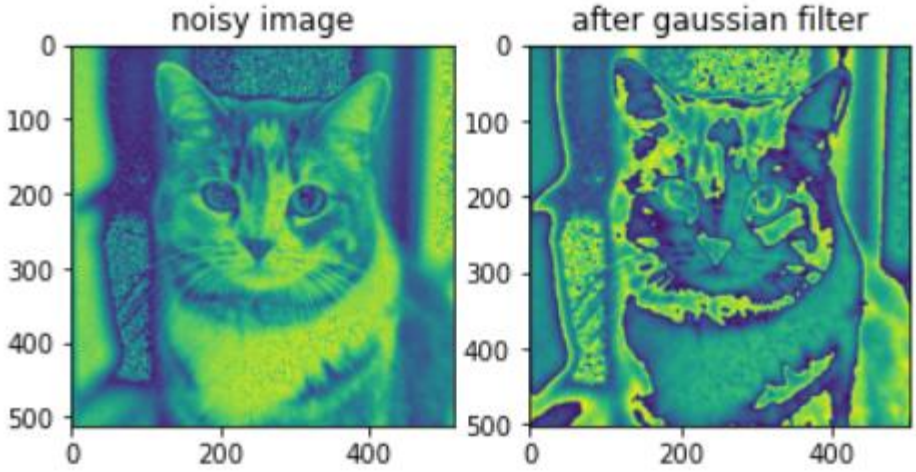
Implemented the gaussian function in the above format. Helper function has also been designed to form the 11*11 matrix. By proper use of formulas from SzeliskiBookDraft_20210214, a gaussian function is designed that functions almost similar to inbuilt gaussian filter functions. The code has been provided with comments that explain the process.

3)

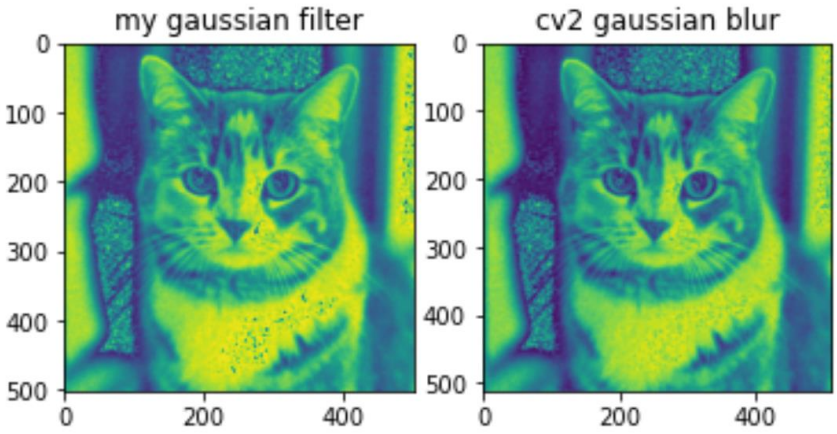


Sigma = 1.1012

On increasing sigma to 1.9 image quality degrades as below.



4)



As evident from above ouput comparision the two images are nearly identical.

Image Denoising via a Bilateral filter

1)

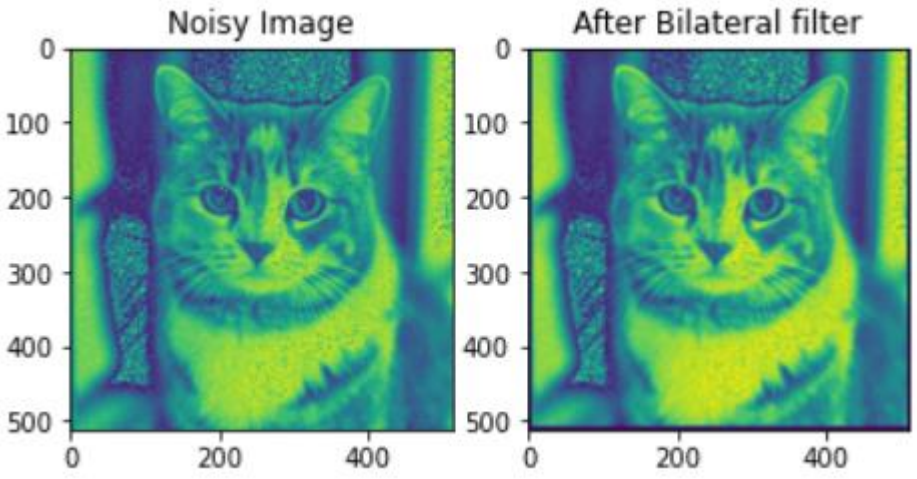
my_Bilateral_filter()

input: *noisy_image, my_5x5_gausskernel, colour_sigma*

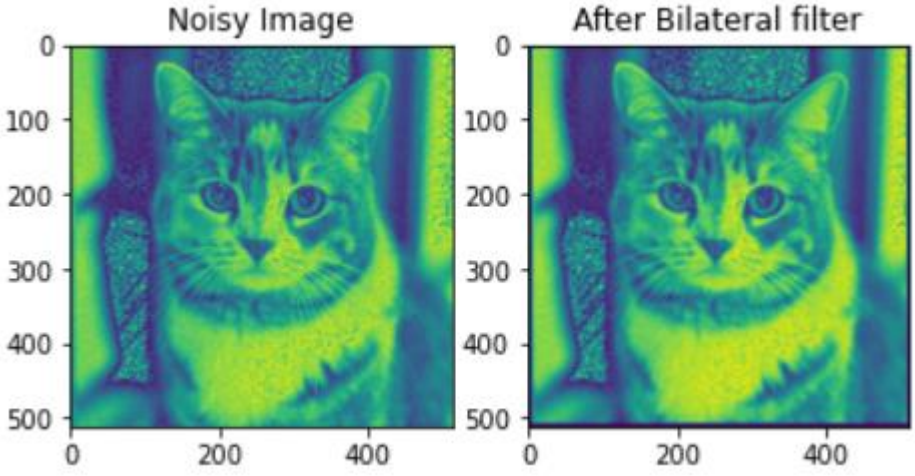
output: *output_image*

A function has been designed as per the above mention prototype. I have used spatial variance and intensity variance instead of single color sigma. If we want to use only colour sigma(intensity variance) then the spatial variance can be kept constant at **1.102**. Relation between variance and sigma is that variance is sigma square. This function uses three helper gunctions and their descriptions has been mentioned in the .ipynb file.

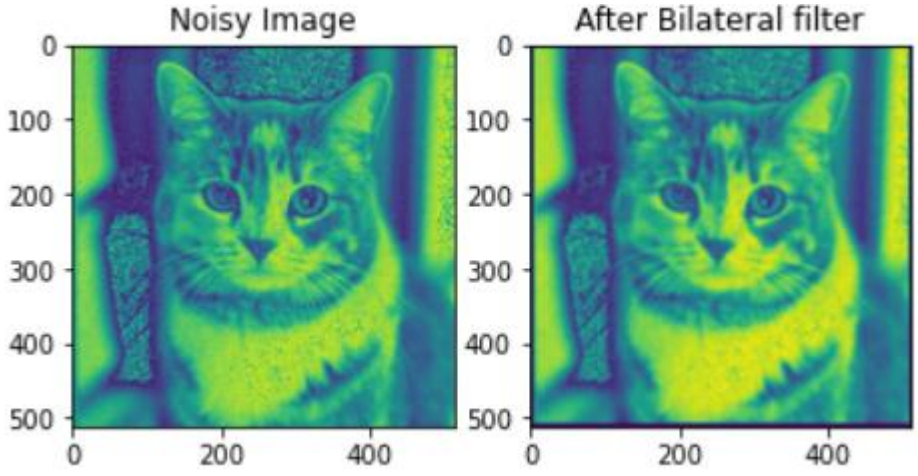
2)



Standard deviation = 1.05 (Intensity_Variance = 1.102)



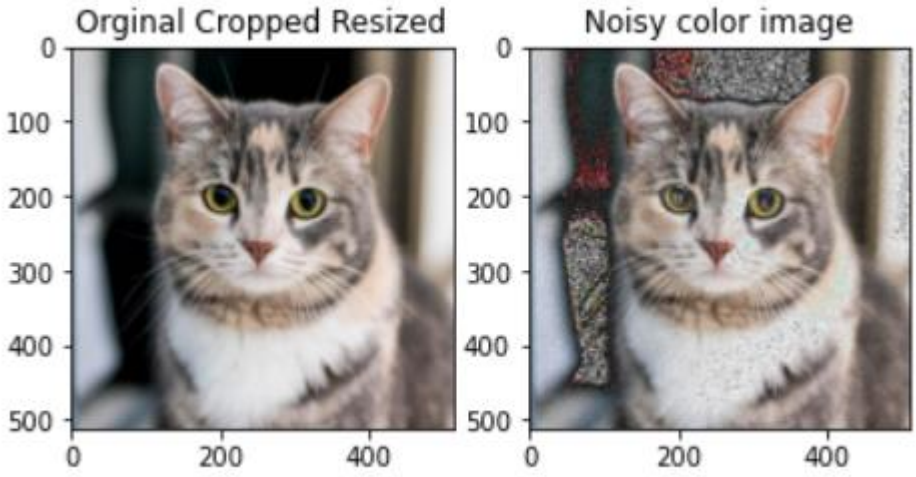
Standard deviation = 1 (Intensity_Variance = 1)



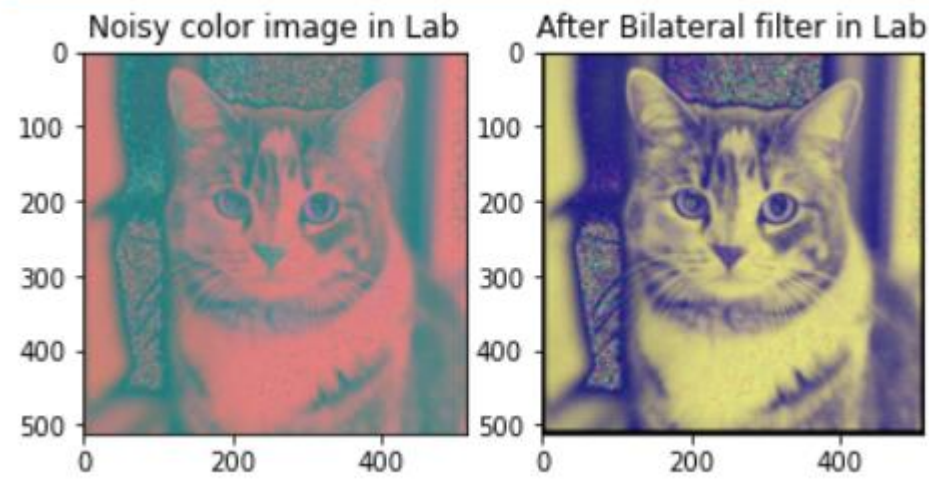
Standard deviation = 2.24 (Intensity_Variance = 5)

If we increase the standard deviation more than 2.3 the spots in the body will become more prominent.

3)

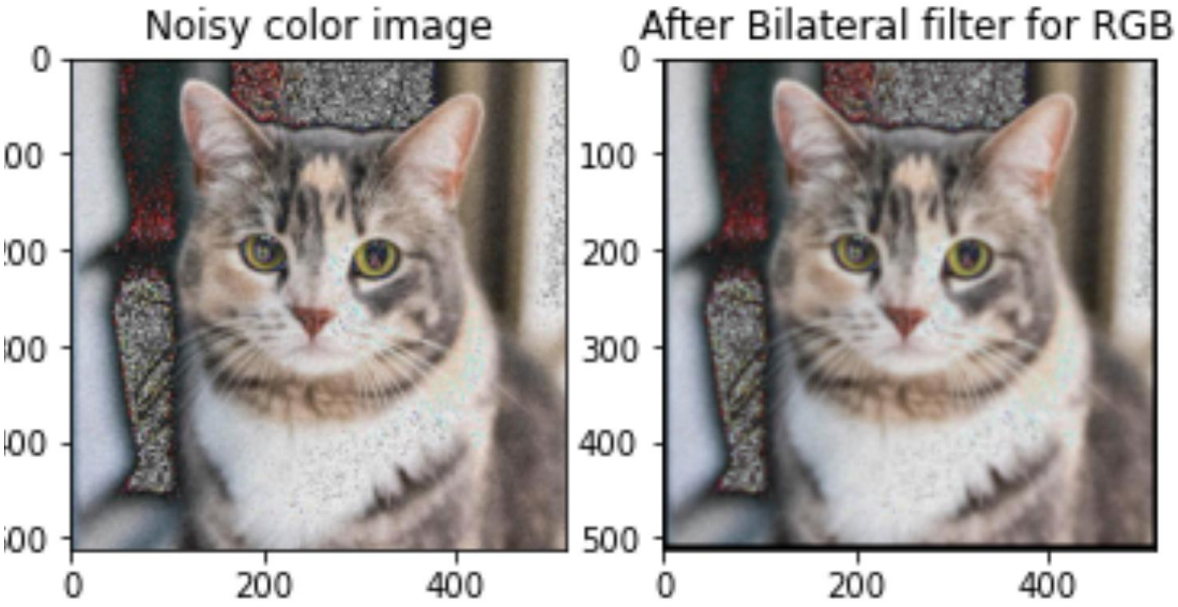


Formed noisy color image at first



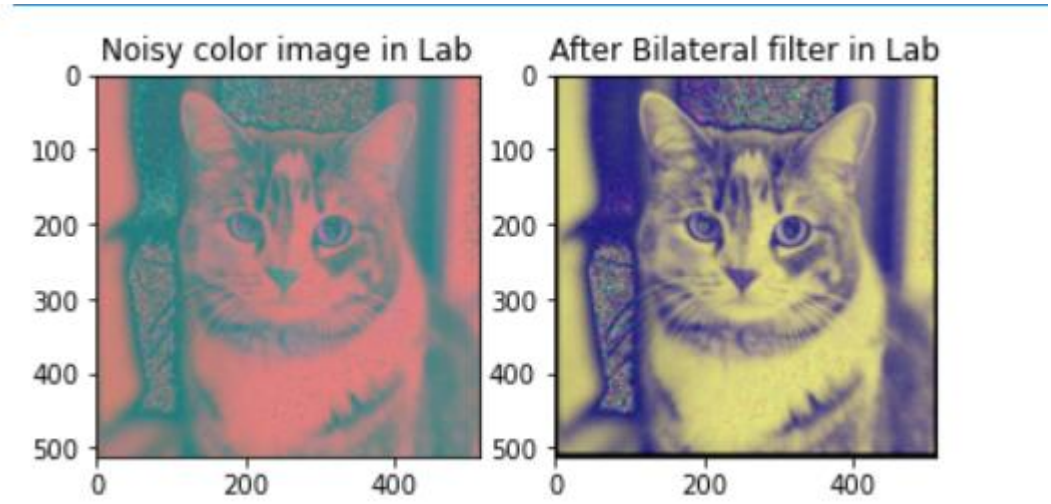
Converted to CIE-Lab color space then applied the bilateral filtering. After, this I obtained the above result.

4)



For RGB applying gaussian filter below.

We can see new colors emerging in the body



In the above image no new color emerged in body or anywhere.

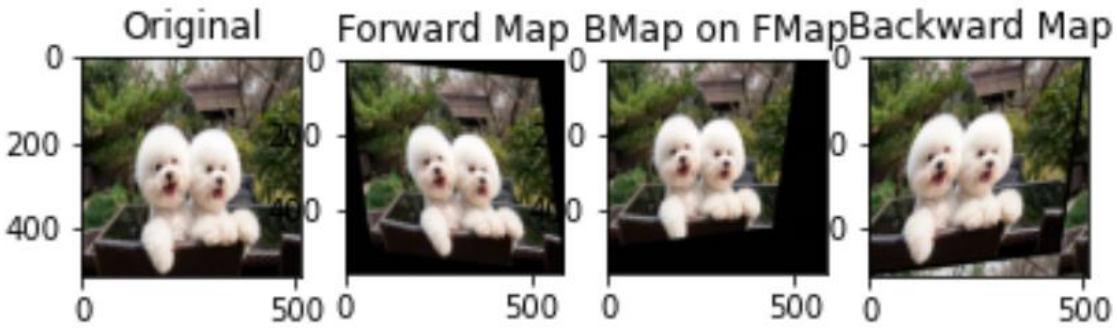
If we use color space Like RGB and we use linear filters like gaussian filters. In linear space, on applying gaussian filters we donot get new color apart from the existing ones. But if we apply non-linear filters like bilateral filters in rgb color space, we may or may not get several new colors apart form existing ones. Since the image contain the face of a cat therefore it is important to preserve the edges, therefore bilateral filtering plays a significant role. Also, for color images bilateral filtering in CIE-Lab color space is the best, since, the colors that are perceptually similar are averaged together, and only important edges are preserved.

Task-6:

1.

Images are original followed by (2.0,4.0),(- 4.0,-6.0),(2.5, 4.5), (-0.9,1.7), (92.0,-91.0) translated.

2.



Forward mapping basically maps the image forward by using a scale factor where as backward mapping maps the image backward. When backward mapping is applied on forward mapped image they almost neutralize each other. Code is included in .ipynb file.

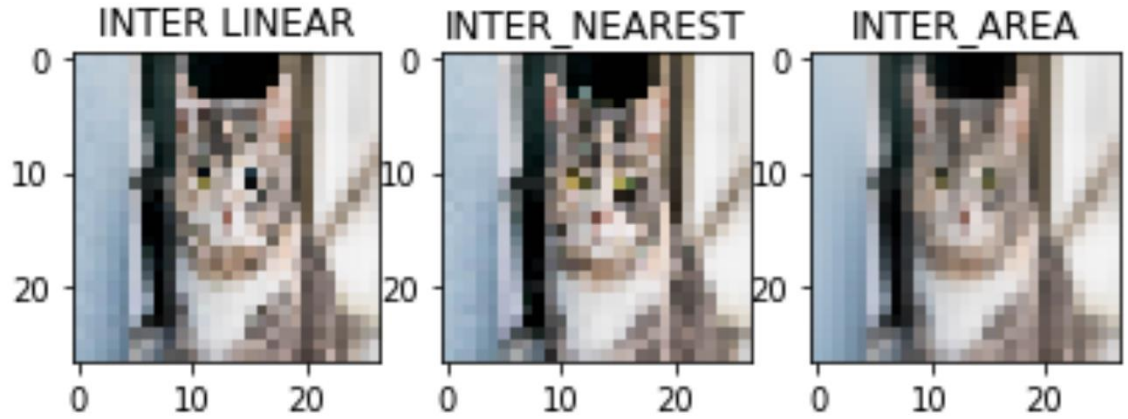
3.

INTER_LINEAR – it is simply bilinear interpolation , which is linear interpolation performed in either direction.

INTER_NEAREST – it is simply earest neighbour interpolation. It simply determines the nearest-neighbouring pixel, and assumes the intensity value of it.

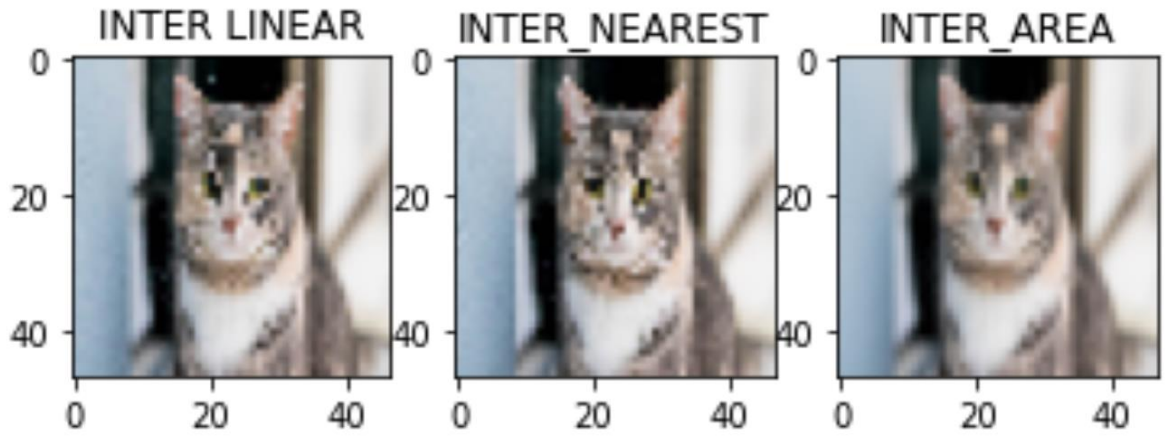
INTER_AREA – It resamples using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method.

Dimensions = 27X27



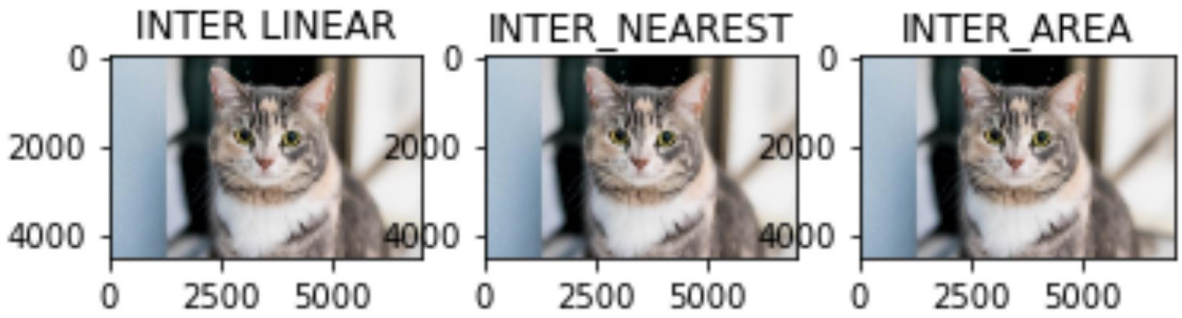
At very low dimensions inter_area is quite good, followed by inter_linear.

Dimensions = 47X47



At intermediate Inter_area performs better than the rest. If we concentrate on eyes of cat we can find inter_area is followed by inter_nearest.

Dimensions = 7000X4500



At larger resolution all of them are similar.