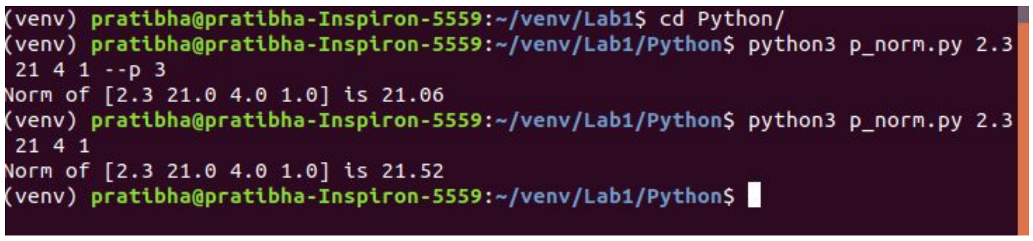
CS 763: LAB1

Pratibha M Varadkar(183079005) S Divakar Bhat(18307R004) Apoorva Agarwal(203050018)

1. PYTHON
   1.1. P-norm

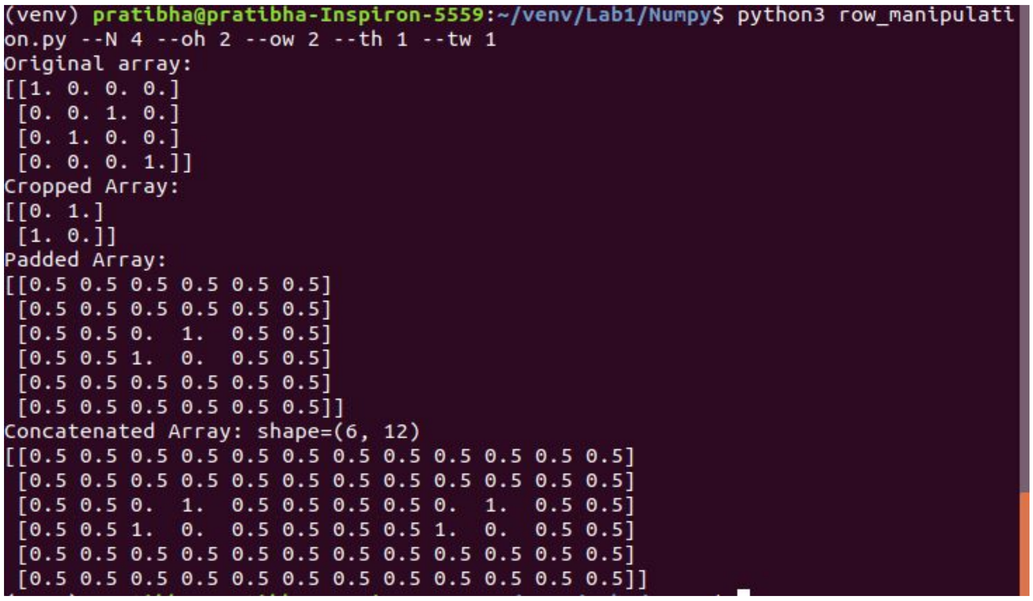
Learned about the different ways in which we can take arguments from the script: required, optional, with specific data types, etc. Additionally how to take a list as an argument to the script (from here).



```
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1$ cd Python/
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1/Python$ python3 p_norm.py 2.3
 21 4 1 --p 3
Norm of [2.3 21.0 4.0 1.0] is 21.06
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1/Python$ python3 p_norm.py 2.3
 21 4 1
Norm of [2.3 21.0 4.0 1.0] is 21.52
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1/Python$ 
```

2. NUMPY
   2.1. Row Manipulation

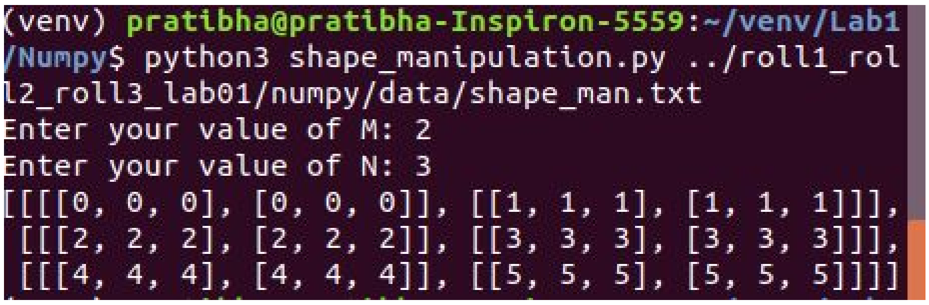
- Given a 1D array we can rearrange it so that the numbers at odd position are transferred at the beginning and the even numbers are placed at the end by using the Permutation matrix (P) by simple matrix multiplication: $1dArray_{(1*N)} * P_{(N*N)}^{T} = oddEvenArr_{(1*N)}$
- To generate the required Permutation matrix, we first formed the shuffled indices array (with odd and even rearranged) and used it to shuffle the rows of the Identity matrix.
- Crop Array function: This was analogous to selecting a subrange of rows and columns of a matrix/2d array.
- Padding the array: Numpy function numpy.pad(array, pad_width, mode='constant', **kwargs) was used with pad_width as ((2,2),(2,2)) and in 'constant' mode (with constant value of 0.5).
- Concatenate arrays: Here Numpy function numpy.concatenate((a1, a2, ...), axis=0, out=None, dtype=None, casting="same_kind") was used, with the padded array as both the arguments and axis = 1 for horizontal concatenation.



```
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1/Numpy$ python3 row_manipulati
on.py --N 4 --oh 2 --ow 2 --th 1 --tw 1
Original array:
[[1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]]
Cropped Array:
[[0. 1.]
 [1. 0.]]
Padded Array:
[[0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.  1.  0.5 0.5]
 [0.5 0.5 1.  0.  0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5]]
Concatenated Array: shape=(6, 12)
[[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.  1.  0.5 0.5 0.5 0.5 0.  1.  0.5 0.5]
 [0.5 0.5 1.  0.  0.5 0.5 0.5 0.5 1.  0.  0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]
 [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]]
```

2.2. Shape Manipulation

One naive approach was to traverse through the array and generate a M*N matrix with all its elements equal to the current element and then append it to form the required 4D array. Another approach was to use the Numpy function numpy.repeat() to generate the N*M copies of every element and then reshape it to the required 4D array.



```
(venv) pratibha@pratibha-Inspiron-5559:~/venv/Lab1
/Numpy$ python3 shape_manipulation.py ../roll1_rol
l2_roll3_lab01/numpy/data/shape_man.txt
Enter your value of M: 2
Enter your value of N: 3
[[[[0, 0, 0], [0, 0, 0]], [[1, 1, 1], [1, 1, 1]]],
 [[[2, 2, 2], [2, 2, 2]], [[3, 3, 3], [3, 3, 3]]],
 [[[4, 4, 4], [4, 4, 4]], [[5, 5, 5], [5, 5, 5]]]]
```

2.3. Principal Component Analysis (PCA)

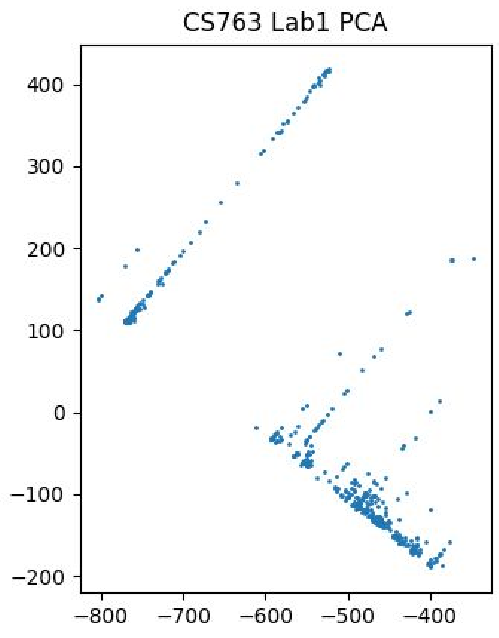
Data used:
The data over which PCA was performed for dimensionality reduction was the Boston Housing Data obtained from here. It consisted of 506 data points with 14 attributes each.

The variables were normalized to have a mean of 0 and a standard deviation of 1, before using PCA. This will orient the data points such that they pass through the origin and have the same variance.

Followed by this we performed eigenvalue decomposition using the SVD technique and projected the data points to the 2-dimensional plane.

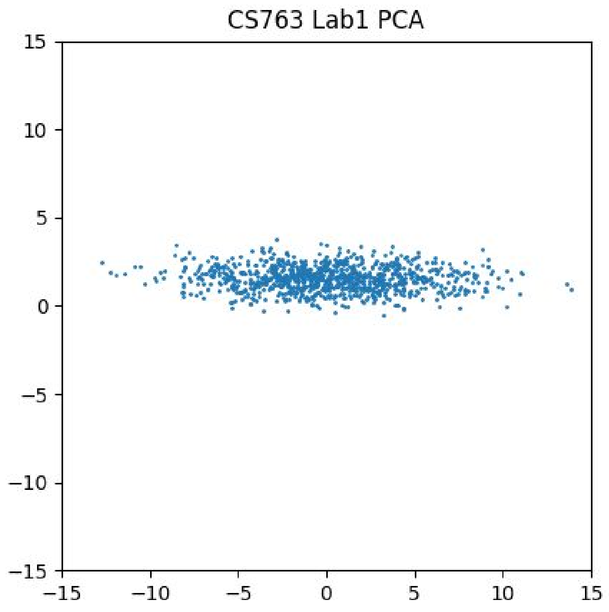
In NumPy eigenvalues and eigenvectors can also be found using the linalg.eig routine. This is different from the SVD technique used here but is closely related in terms of the relation between eigenvalues and singular values obtained.

In fact, most PCA implementations use SVD instead of performing their own covariance matrix decomposition because SVD can be much more efficient and much more economical to calculate.



The above figure depicts the Boston housing data plotted by projecting it to the 2D plane using PCA. (Note that the range of values is not restricted in this case between -15 and 15 as no corresponding values are present in this range).

While the following figure depicts the plot corresponding to the data provided in the pca_data.txt file.



3. OPENCV

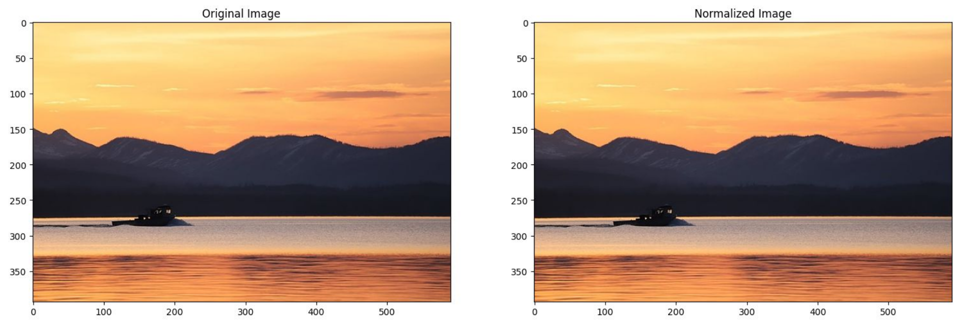
3.1. Image Conversion

In OpenCV color images are read in a numpy array following the BGR channel using cv2.imread(image_path) method. Numpy array for color image is a 3D array where each dimension contains colour(B/G/R) intensity in a pixel. Pixel intensities range from (0,255), before using images in Machine Learning models the image is normalized to a range of pixel intensity to improve the model performance.

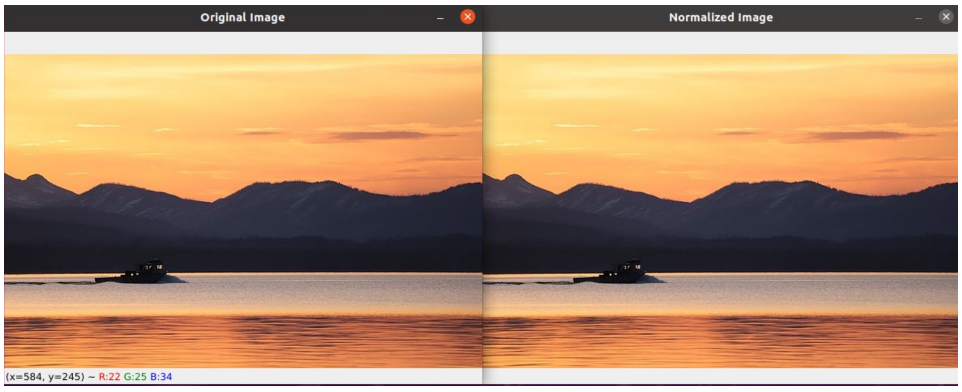
Here image is normalized to (0,1) range using cv2.normalize(img, None, 0, 1, norm_type=cv.NORM_MINMAX, dtype=cv.CV_32F)

Images are plotted using 2 different python modules-

1. Matplotlib
   As matplotlib uses RGB channel and openCV uses BGR channel image read using opencv can be first converted to RGB using cv2.cvtColor(img,cv2.COLOR_BGR2RGB) and then plotted using plt.imshow(img). Following image shows the original and normalized image plotted using Matplotlib.



2. OpenCV
   The OpenCV image is plotted using cv.imshow(Window name",img). Following image shows the original and normalized image plotted using OpenCV.



(x=584, y=245) ~ R:22 G:25 B:34

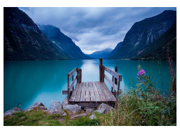
3.2. Display Images

To display the images in a wrap around fashion and give previous & next functionality images are loaded in a numpy array and stored in a list. Index of next image to be displayed is computed using the following code snippet:

```
if k == ord('p'):
      curr_img=(curr_img+1)%len(img_list)
elif k == ord('n'):
      curr_img=(curr_img-1)%len(img_list)
```

To display the images in the same window same window name is used to display all the images

Following is the list of images used to display:

The output of the display_images question can be viewed here.

3.3. Display Video

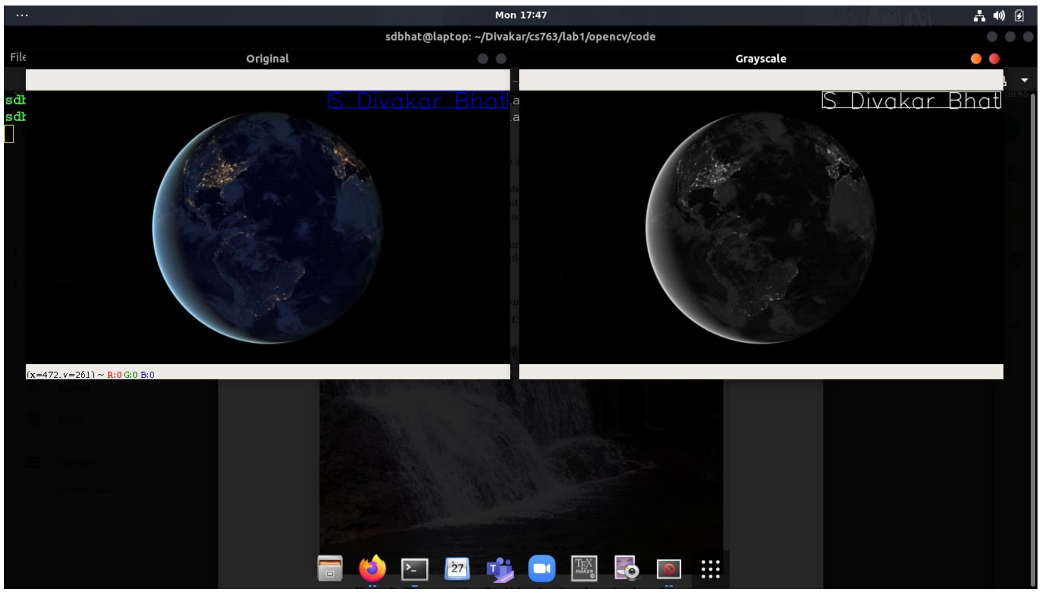
For this, the sample video provided in the document was utilized which can be accessed here.
The video was annotated and displayed by extracting each frame and performing necessary operations on the same. Each frame was also then converted to grayscale so as to be used for displaying the grayscale version of the video under discussion.

For annotating each frame cv2.putText() was used to overlay each frame with a text at the top right corner. The bounding box was also placed enclosing the text using the cv2.rectangle() function. The coordinates for the proper placement of both the text and the bounding box was approximated and supplied to the functions through trial and error.

The approach used here appears to be very straightforward and naive. It will be better if an alternative approach was devised to place and annotate each frame with the required shapes and texts with more flexibility in its placement.

cv2.moveWindow function was employed to display both the original and grayscale videos annotated side by side.

The following figure depicts the snapshot of the implementation.


Mon 17:47
sdbhat@laptop: ~/Divakar/cs763/lab1/opencv/code
Original
Grayscale
S Divakar Bhat
S Divakar Bhat
(x=472, y=261) ~ R:0 G:0 B:0

Acknowledgement

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html
- https://www.geeksforgeeks.org/python-opencv-cv2-rectangle-method/
- https://www.geeksforgeeks.org/python-opencv-cv2-puttext-method/
- https://towardsdatascience.com/pca-with-numpy-58917c1d0391
- https://www.codespeedy.com/normalizing-an-image-in-opencv-python/
- https://stackoverflow.com/questions/30230592/loading-all-images-using-imread-from-a-given-folder
- https://docs.opencv.org/3.4.2/dc/d2e/tutorial_py_image_display.html
- https://unstabnet.in/site/generate-permutation-matrix-in-numpy/
- https://numpy.org/doc/stable/reference/generated/numpy.pad.html
- https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html
- https://numpy.org/doc/stable/reference/generated/numpy.repeat.html