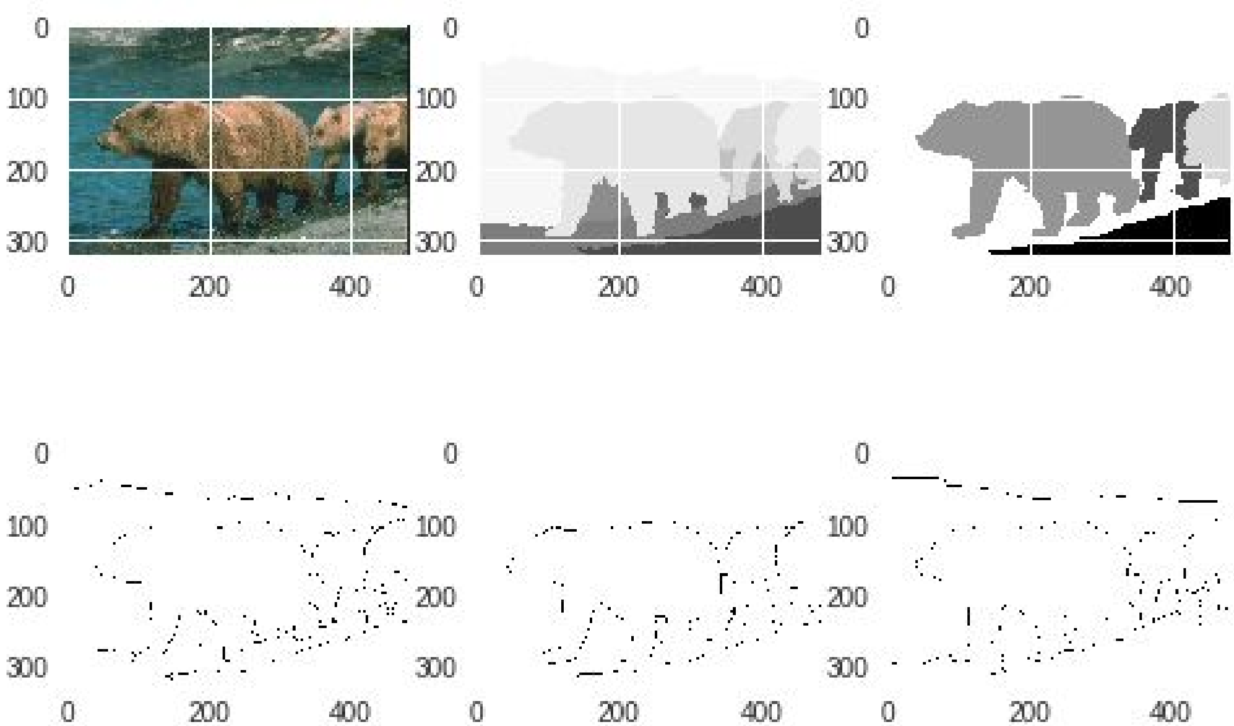
Image Segmentation

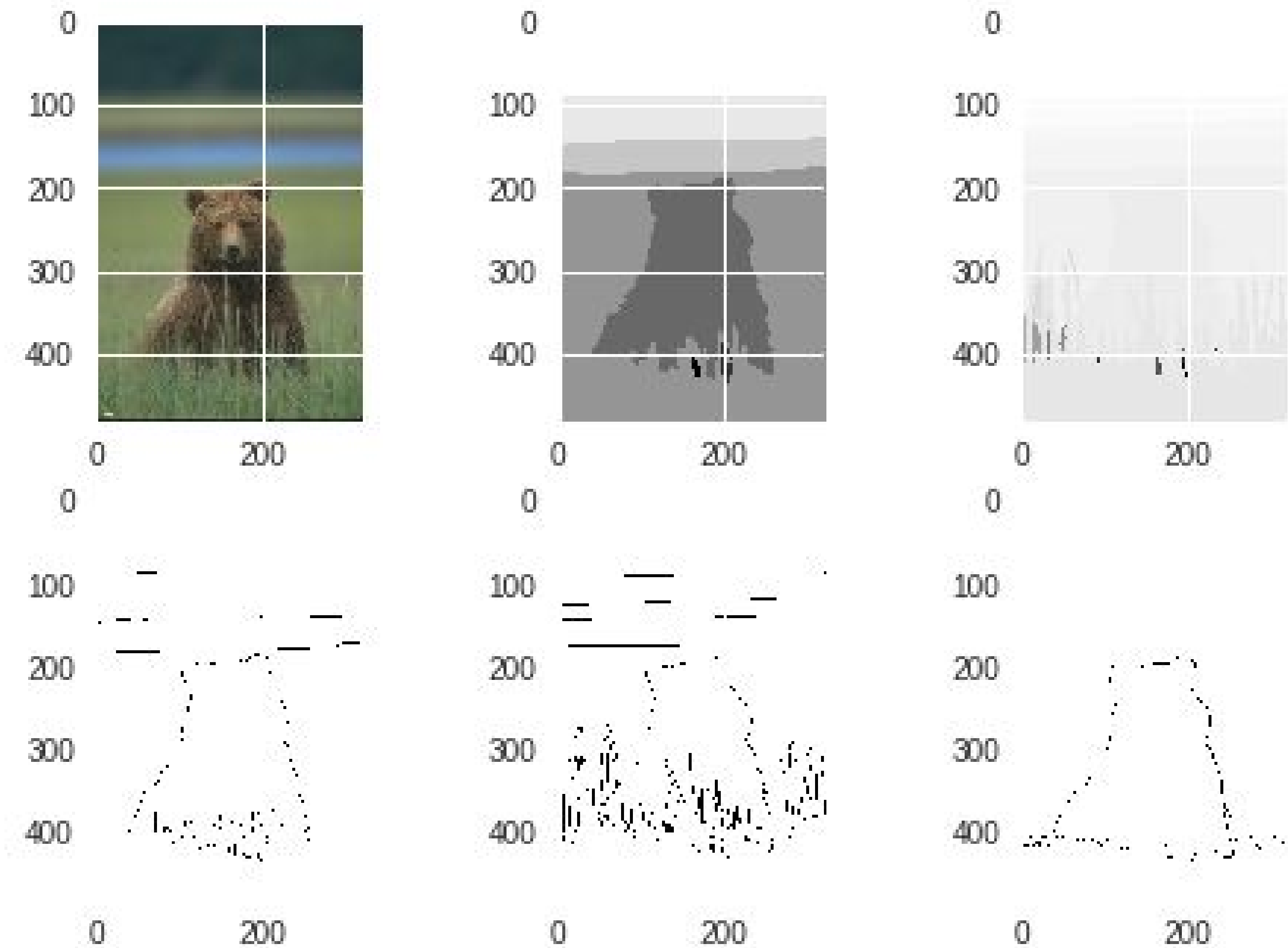
The problem tackled is segmenting similar parts of an image that for a cluster using unsupervised learning, K-means clustering is the main focus.

1- Downloading the dataset and understanding format:

After discovering the data we found that it is composed of 3 subgroups one for training,one for testing and one for validation.
In each group there is a set of images in jpg format and a corresponding mat file containing ground truth images for each image.
The ground truth images are in 2 shapes : either a grayscale segmentation of each cluster or just a white image with black lines showing boundaries of clusters.

2- Visualizing the image and ground truth segmentation:





3- Segmentation using K-means

The code below represents the kmeans implementation of the library **cv2**. It is responsible for creating segmented images using given parameters such as number of clusters **K**.

```
def images_to_seg_colored(imgs,K):

  imgs_=[]
  for img in imgs:
    Z = img.reshape((-1,3))
```

```
    # convert to np.float32
    Z = np.float32(Z)

    # define criteria, number of clusters(K) and apply kmeans()
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10,
1.0)


ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
    #print(label)

    # Now convert back into uint8, and make original image
    center = np.uint8(center)
    res = center[label.flatten()]
    res2 = res.reshape((img.shape))


    imgs_.append(res2)

  return imgs_
```

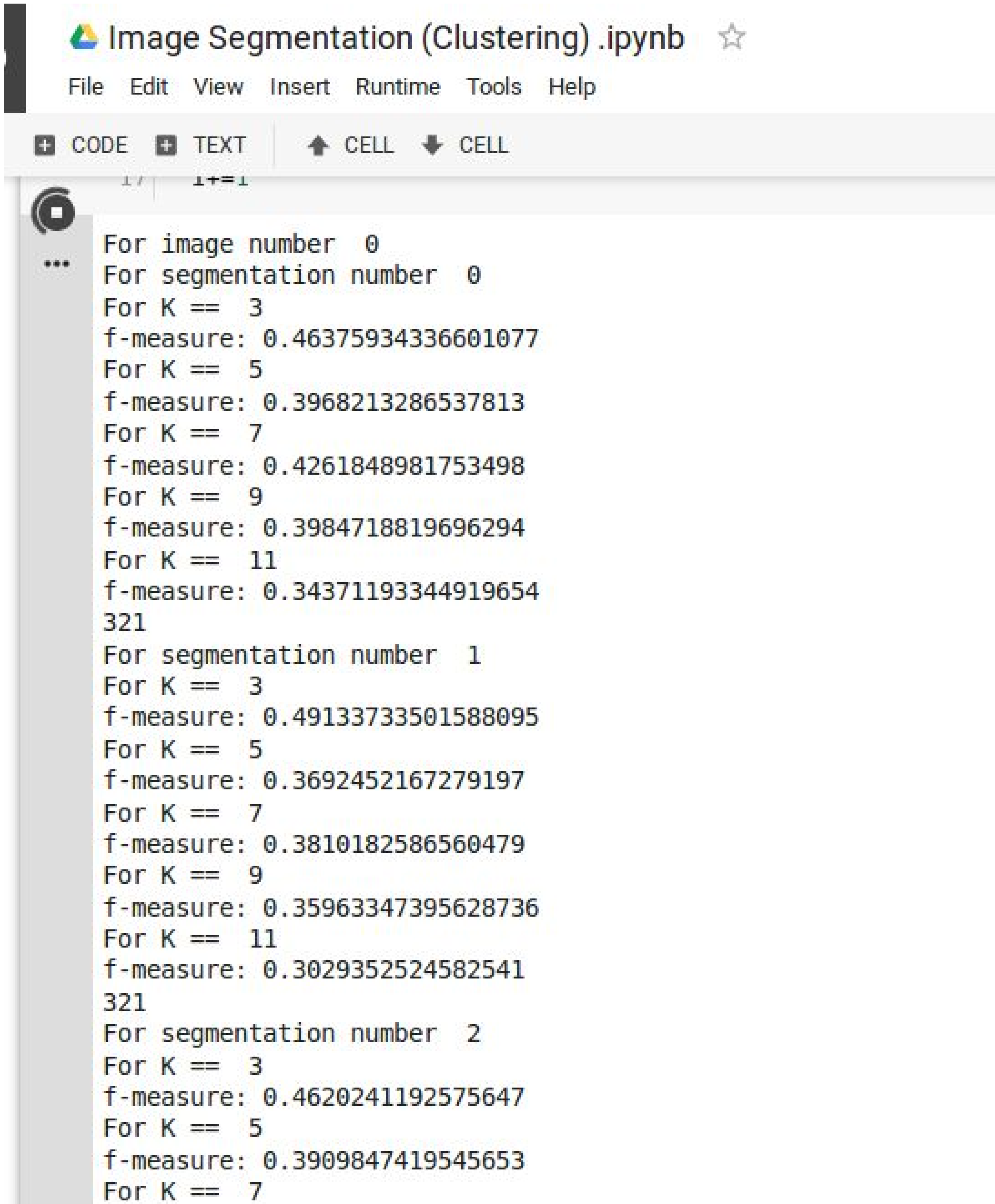


Figure above shows f-measure scores for some images against their segmentations

F-measure

$$\begin{bmatrix} * & * & * & \dots & \\ \dots & * & & \dots & * \\ * & & & & \\ \dots & * & * & \dots & * \end{bmatrix}$$

$$\begin{bmatrix} \$ & \$ & \& & \dots & \\ \dots & \# & & \dots & \$ \\ \& & & & & \\ \dots & \& & \$ & \dots & \# \end{bmatrix}$$

Suppose that the first matrix is that for the image extracted using K-Means for certain number of clusters **K**. Also, say that the second matrix is the one that represents the ground truth segmentation image. Accordingly, to evaluate clustering efficiency, we follow these steps:

1. If "*****" were considered in same cluster in 1st matrix, **we keep track of their indices but in second matrix**
2. After specifying the indices in 2nd matrix, we can now compute purity: for instance, in 2nd matrix, purity here is number of occurence of "**$**", as it represents the maximum frequent sample in cluster, divided by number of samples in cluster.
3. For calculating recall, we divide the frequency of "**$**" in our cluster by the frequency of "**$**" in all clusters
4. Now can compute the f-measure of this cluster $\mathbf{F_i=(2*purity_i*recall_i)/(purity_i+recall_i)}$
5. The overall f-measure **F** is average of sum of $\mathbf{F_i}$ of each cluster

Conditional entropy:

1- as described in lecture we compute conditional entropy by determining conditional entropy of T with respect to each cluster

– Conditional Entropy of T with respect to C_i

$$H(\mathcal{T}|C_i) = -\sum_{j=1}^{k}\left(\frac{n_{ij}}{n_i}\right)\log\left(\frac{n_{ij}}{n_i}\right)$$

The we compute the conditional entropy :

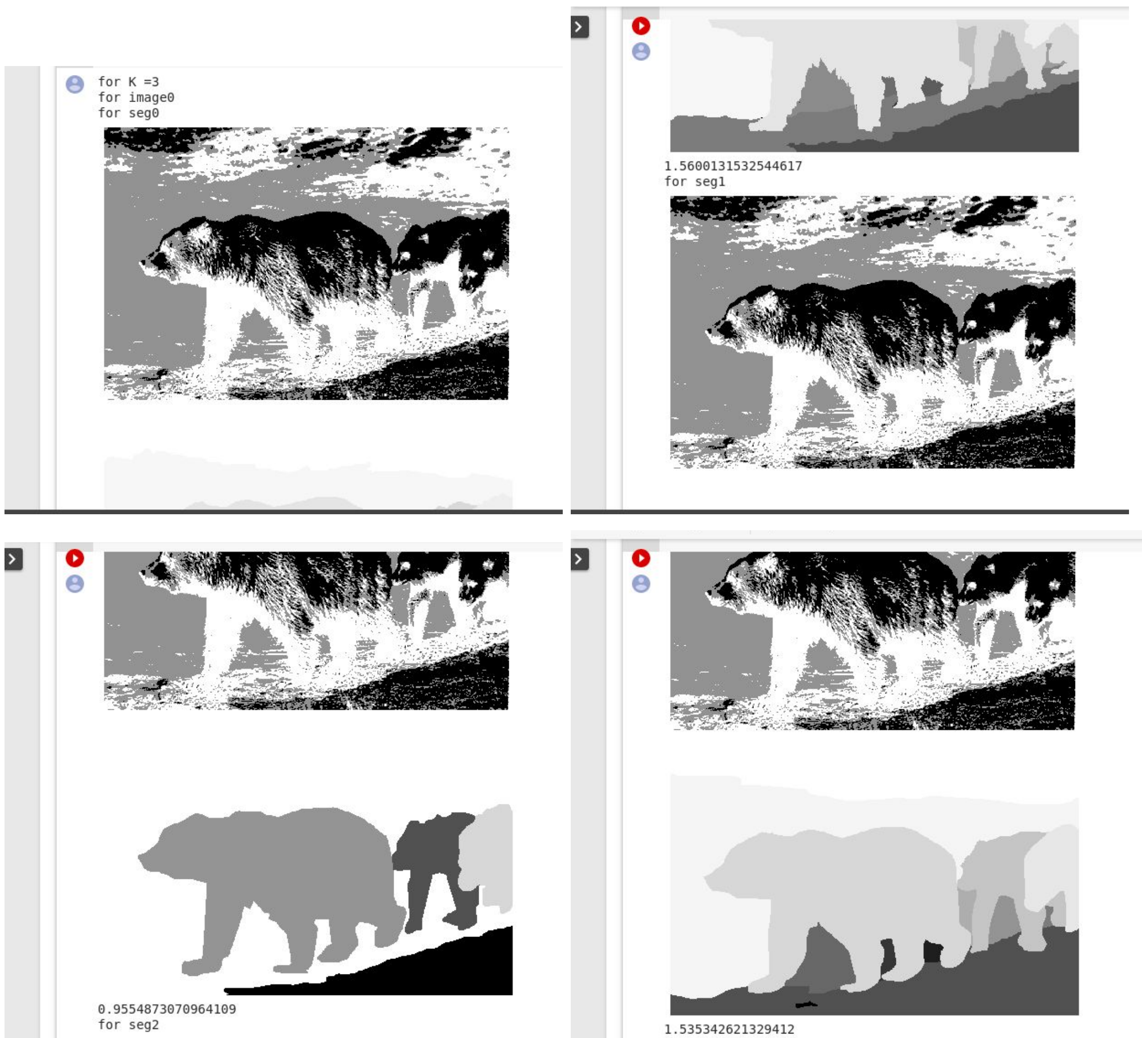
– The conditional entropy of T given clustering C

$$H(\mathcal{T}|\mathcal{C}) = \sum_{i=1}^{r} \frac{n_i}{n} H(\mathcal{T}|C_i)$$

By applying the above rules in the code we compute the conditional entropy for each segmentation

- Best case scenario is having an entropy of zero
- Worst case scenario is having an entropy log(k) where K is number of clusters

After applying all of that we get the below results:

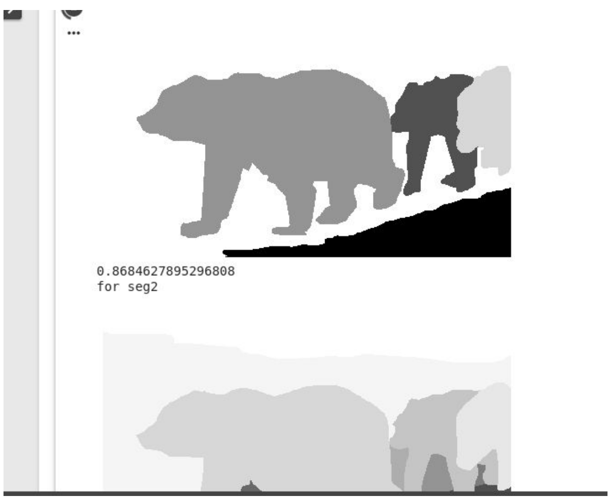


Where the number printed is the number of the conditional entropy for the segment .
Here is a simple table showing the average of 10 images using conditional entropy and the over all data set average

	img1	img2	img3	img4	img5	img6	img7	img8	img9	img10	Average dataset
K=3	1.49	2.22	2.22	1.457	0.266	0.724	1.20	1.514	2.93	0.78	1.735
K=5	1.308	1.924	1.164	1.38	0.251	0.68	1.13	1.177	2.836	0.76	1.5723
K=7	1.25	1.58	1.1412	1.25	0.239	0.68	1.109	1.1031	2.7930	0.73	1.4891
K=9	1.489	1.522	1.126	3.538	0.239	0.62	1.105	1.094	2.697	0.707	1.42821
K=11	1.087	1.29	1.29	1.29	0.234	0.628	1.082	1.071	2.65	0.66	1.382

4-Big picture

For image 1:

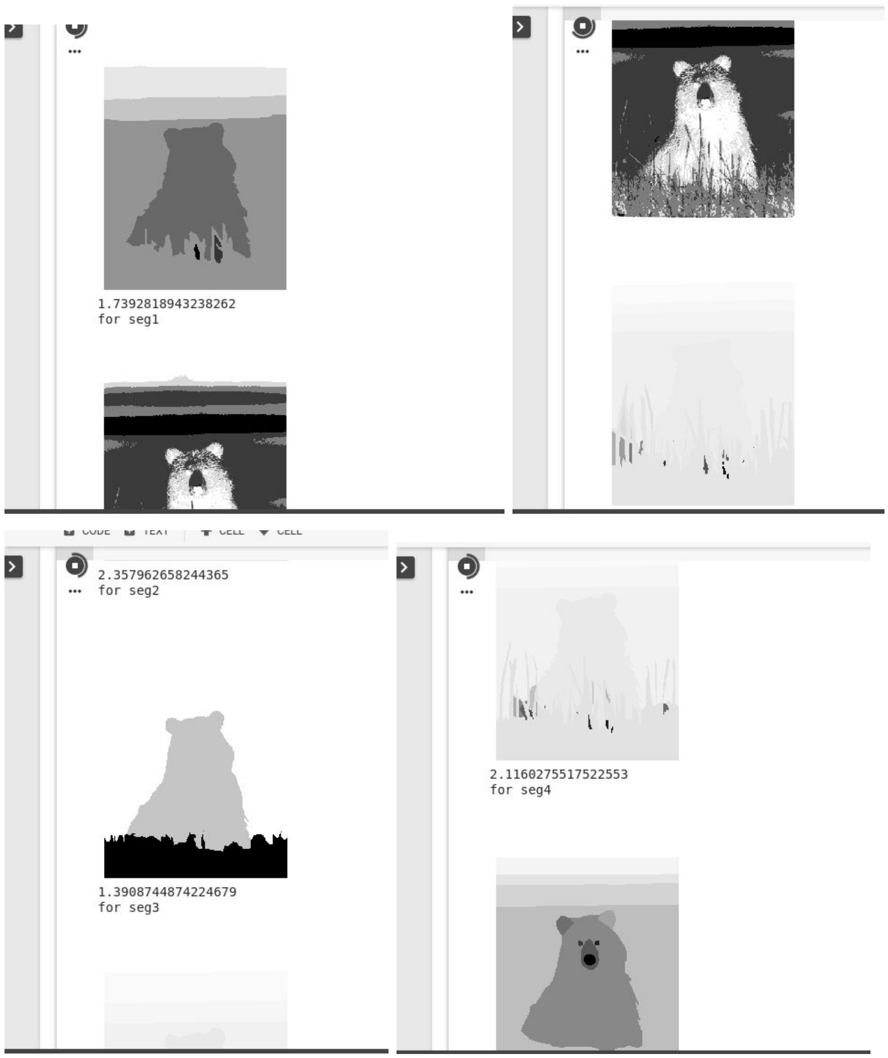

0.8684627895296808
for seg2


1.417150447804223
for seg3
1.4110873448292631
for seg4


1.4110873448292631
for seg4
1.641921746636568
for seg5


1.641921746636568
for seg5
1.4462193996837083
averagee of image 0= 1.3692350743667383
for image1
for seg0

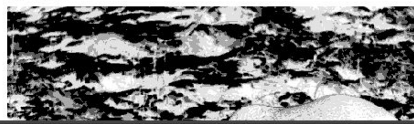
For img2:


1.7392818943238262
for seg1
2.357962658244365
for seg2
1.3908744874224679
for seg3
2.1160275517522553
for seg4

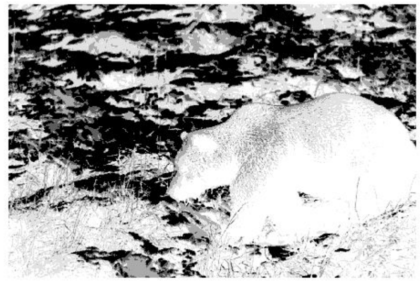
For image3:

averagee of image 1= 1.91050025270929
for image2
for seg0

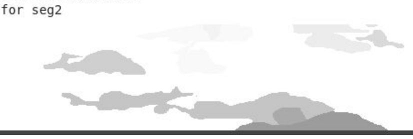
1.3365538207645342
for seg1

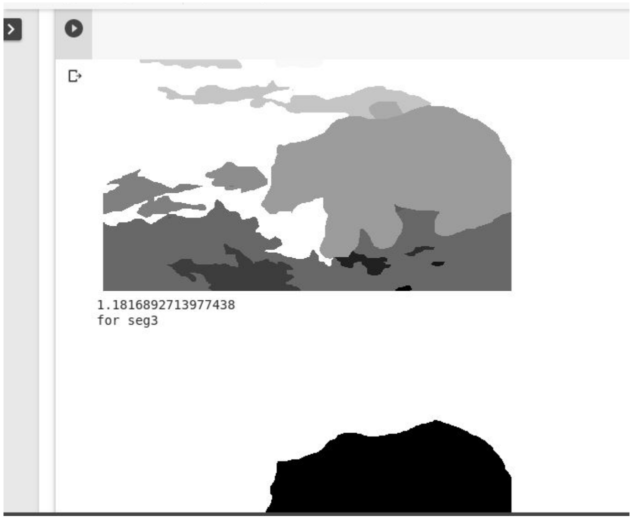


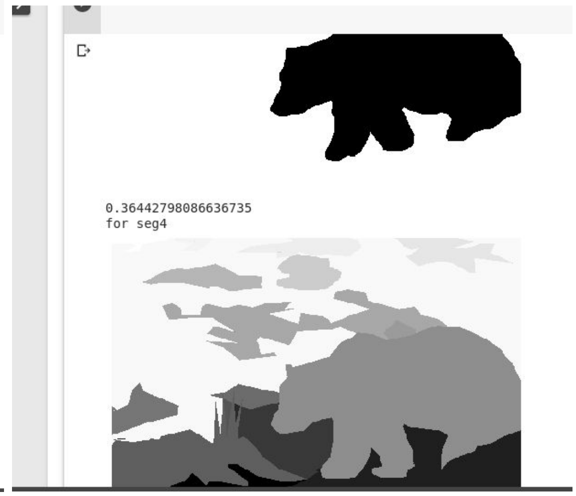
1.3365538207645342
for seg1

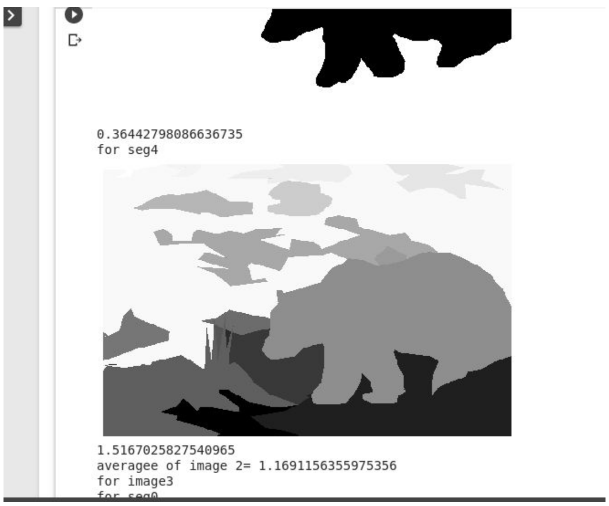


1.4462045222049356
for seg2



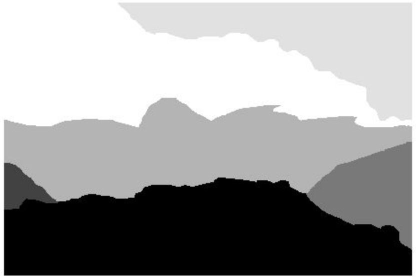

1.1816892713977438
for seg3


0.36442798086636735
for seg4

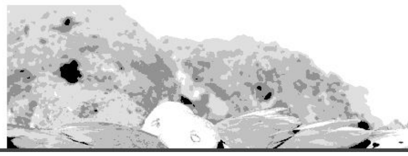

0.36442798086636735
for seg4
1.5167025827540965
averagee of image 2= 1.1691156355975356
for image3

For image 4:

```
for image3
for seg0
```



```
1.1545690230821648
for seg1
```

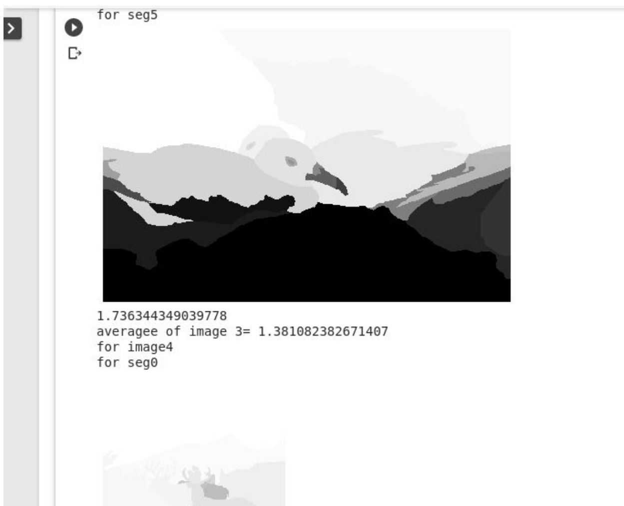


```
1.1637686222574044
for seg2
```

```
1.5646165481783787
for seg3
```


1.5646165481783787
for seg3
1.2978614610716264
for seg4


1.3693342923990894
for seg5


for seg5
1.736344349039778
averagee of image 3= 1.381082382671407
for image4
for seg0

For image 5:

averagee of image 3= 1.381682382671
for image4
for seg0

4.277687242275284
for seg1

4.277687242275284
for seg1

3.9944953551501396
for seg2

2.459284707103287
for seg3

3.3858846307471073
for seg4

3.743460903689229
for seg5

4.319424314132549
for seg6

2.976068602484601
averagee of image 4= 3.593757965083171
for image5
for seg0

Segmentation using Normalized Cut

```
For image number  0
For segmentation number  0
For K ==  5
f-measure: 0.517889426154622
For segmentation number  1
For K ==  5
f-measure: 0.6130584077912961
________________________________
For image number  0
For segmentation number  0
For K ==  5
f-measure: 0.39392547461548866
For segmentation number  1
For K ==  5
f-measure: 0.371604714052863
```

The table above shows a **snippet*** of the comparison using k-means segmentation and normalized cut version. Obviously, and **in general**, results using normalized cut segmentation shows better f-measure indicating better clustering efficiency.

<u>Note</u>: Increasing f-measure indicates better clustering. F-measure of "1" represents ideal clustering efficiency.