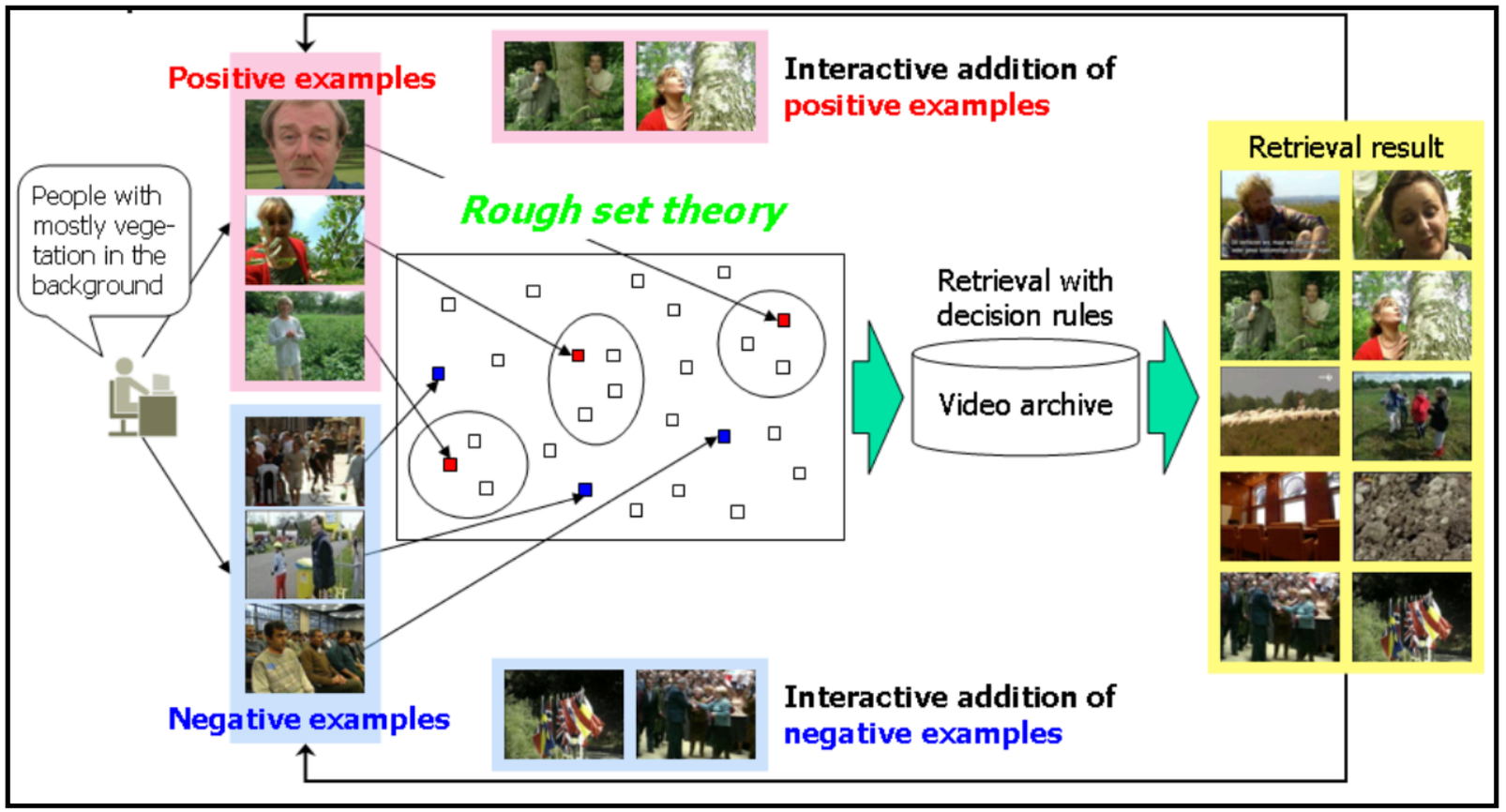
Computer Vision

Assignment 4:

Content Image retrieval systems

Irene Volpe - r0740784



Summary¶

Most of us will acknowledge that it is a challenge to find back a specific picture that we took months ago. Efficient image retrieval is a challenging task in many domains.
In this notebook we build a simple Content Based Image Retrieval system (CBIR, figure 1) by:
1) generating image features using three feature representations of images:

- **Histograms (**page 2-3)
- **Mean** of the RGB Color Space (page 4)
- **Moment** of the RGB Color Space (page 4)

2) Selecting the best feature representation of the training set among the three above and use it to predict unseen images using a deep learning algorithm. Finally, we evaluate the score of the test set prediction.

We will be using the CIFAR-10 dataset for this notebook.
This dataset consists of 60,000 32x32 colour images in 10 classes (Figure 2), with 6,000 images per class.
There are 50,000 training images (figure 3) and 10,000 test images.
The CIFAR-10 dataset can be easily loaded from Keras.

We train (generate image features) the CBIR model using the images in the training set.
We then get 10 random images from the test set (query images).
For each of the images from the test set, we compute how similar they are according to the images in the training set using Euclidian distance between the image features.
We retrieve the images from the training set that are similar to the query image and measure the performance of the CBIR model using Precision and Recall measure.

Figure 3:First 10 images per class from training set

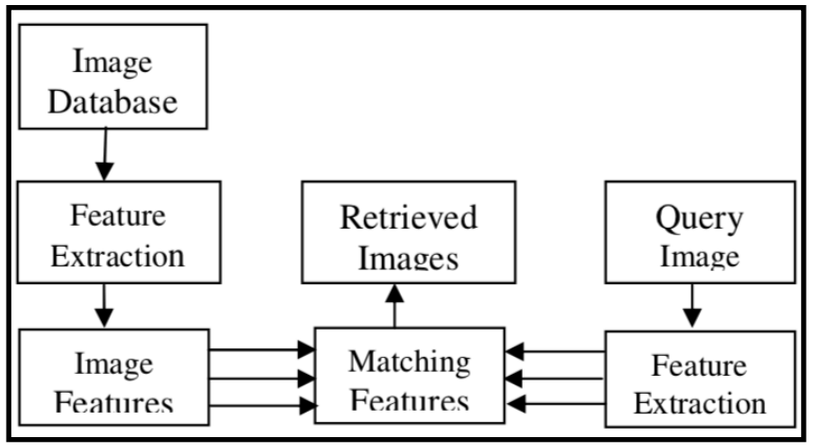


Figure 1: a block diagram of CBIR

Figure 2: First 500 images from training set

CBIR trained by generating image features using Histograms

“Content-based” means that the search will analyze the actual contents of the image, and such content might refer to colors, shapes, textures, or any other information that can be derived from the image itself.
Histogram search algorithms characterize an image by its color distribution or histogram (figure 4).A histogram will represents all the colors and the level of their occurrence in an image irrespective of the type of the image.
Few basic properties about an image can be obtained from using a Histogram: the **shape** and the **concentration of the colors** in the histogram will be the same for similar objects even though they are of different colors. To estimate **color similarity** between a query image and a database image, bins which are the product of H and S are used for histogram intersection.
Identifying objects in a grey scale image is the easiest one as the histogram is almost similar as the objects have the same colors for same objects (figure 5a), yet we’re converting our images from RGB to HSV, so we’ll obtain three histograms one per channel (figure 65b).
Therefore if speeding up the process is required, the color conversion can be done to black and white. In order for identifying the objects in the images or generating the histogram the system has to obtain the array values.

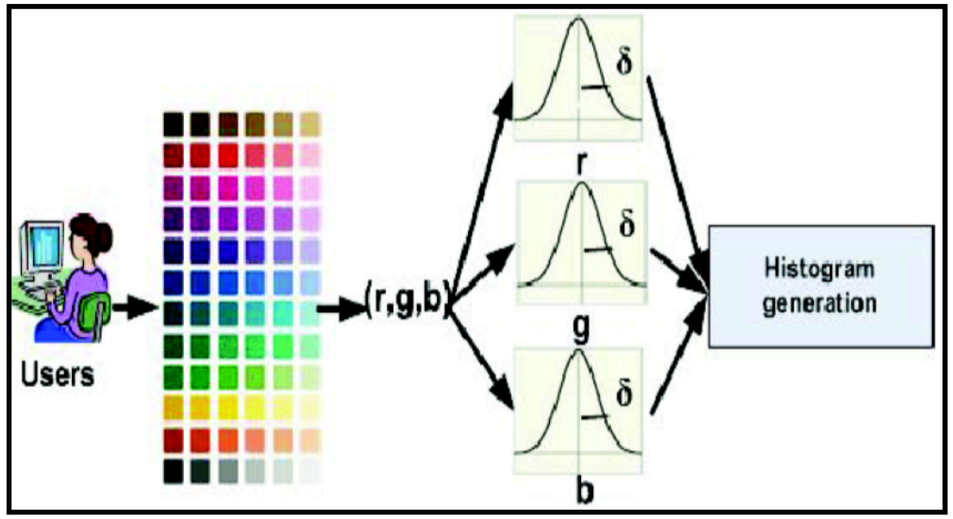
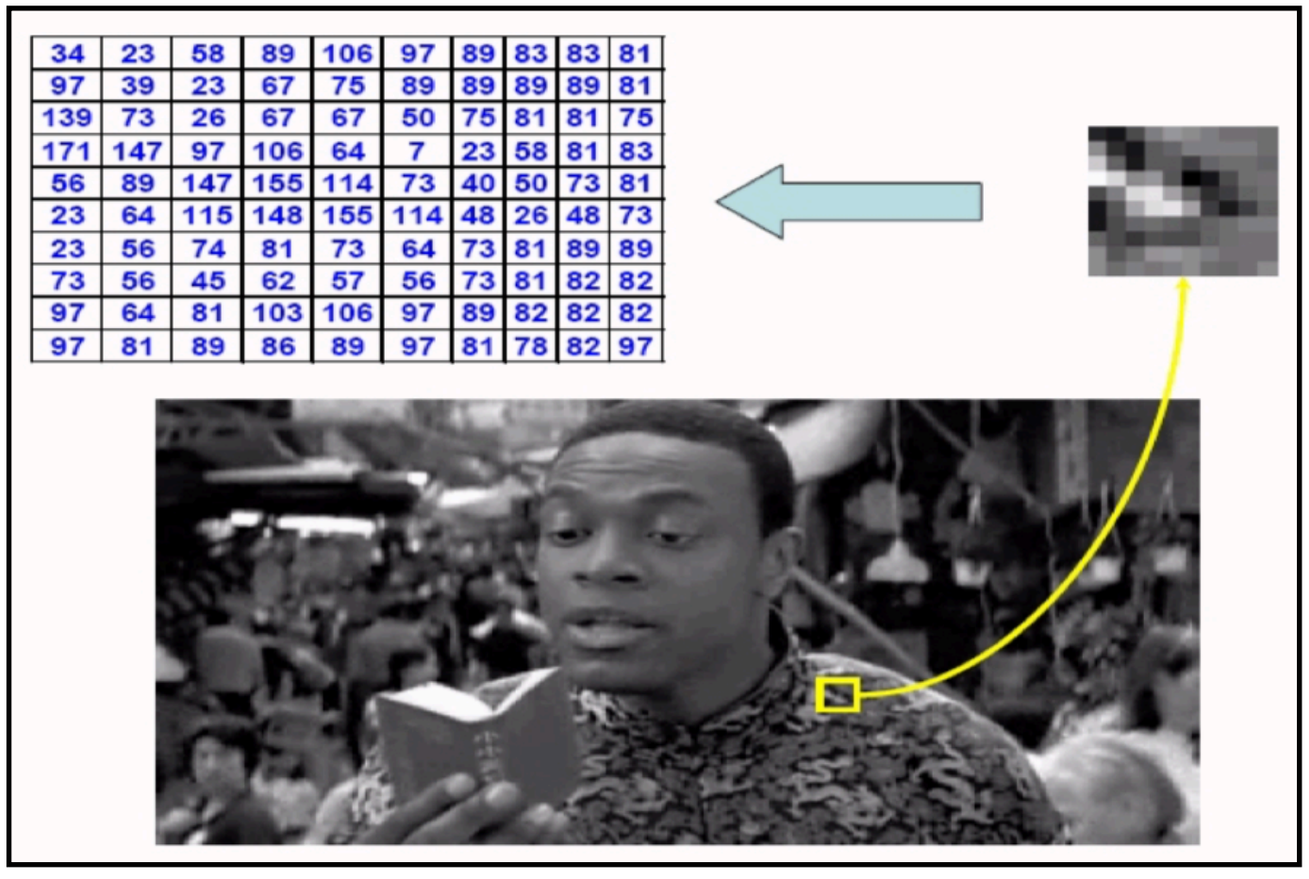


Figure 4: color histogram

The system has to differentiate between useful and unuseful informations.
Consider an image where a person reading a book is the useful information, and the background is the unwanted data.
The system has to group together the repeated pattern to identify the objects in the image. For example in figure 5 is given the array for the part of the shirt and this pattern is repeated again,



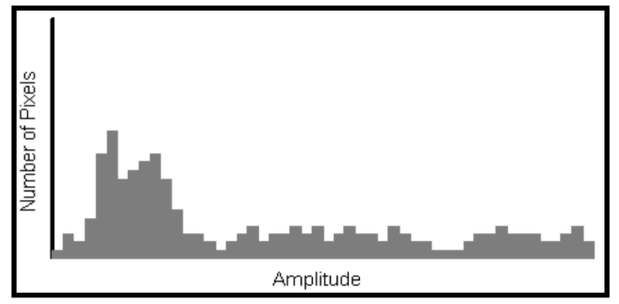


Figure 5a: histogram generation (1 channel)

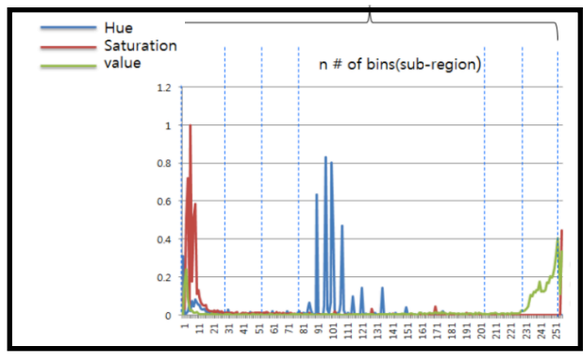


Figure 5b: histogram generation (3 channels)

CBIR trained with Euclidian distance as similarity measure

```
def find_sim(X_train_hist_df,hist_test_flatten):
    for index,row in X_train_hist_df.iterrows():
        sim = np.linalg.norm(hist_test_flatten - row['HIST_TRAIN'])
        X_train_hist_df.at[index,'SIMILARITY']=sim
    return X_train_hist_df
```

	Euclidian
Precision	0,147
Recall	0,139

Tabòe 1

	IMAGES	HIST_TRAIN	SIMILARITY
0	[[[59, 62, 63], [43, 46, 45], [50, 48, 43], [6...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	0.0
1	[[[154, 177, 187], [126, 137, 136], [105, 104,...	[0.0, 0.0, 0.040154897, 0.020077448, 0.0100387...	0.0
2	[[[255, 255, 255], [253, 253, 253], [253, 253,...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.007983666, 0.03193...	0.0
3	[[[28, 25, 10], [37, 34, 19], [38, 35, 20], [4...	[0.0, 0.0, 0.0, 0.0, 0.007296549, 0.0, 0.00729...	0.0
4	[[[170, 180, 198], [168, 178, 196], [177, 185,...	[0.0, 0.053285465, 0.097690016, 0.01776182, 0....	0.0
5	[[[159, 102, 101], [150, 91, 95], [153, 95, 97...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.00782468...	0.0
6	[[[164, 206, 84], [105, 140, 61], [118, 148, 1...	[0.0, 0.0, 0.0, 0.018113831, 0.018113831, 0.04...	0.0
7	[[[28, 35, 39], [30, 34, 44], [33, 44, 47], [6...	[0.0, 0.0, 0.009159676, 0.009159676, 0.0366387...	0.0
8	[[[134, 186, 223], [131, 184, 220], [128, 182,...	[0.0, 0.0, 0.023115939, 0.10016906, 0.02311593...	0.0
9	[[[125, 125, 116], [110, 101, 91], [102, 90, 8...	[0.0, 0.0, 0.0, 0.0, 0.20510148, 0.41020295, 0...	0.0

Figure 6: First 10 histograms flattened pixel list of the training set and corresponding Euclidian distances.

Figure 6:
IMAGES lists the first ten images from training set represented by their RGB pixel values in arrays;
HIST_TRAIN lists all histogram representations of all the colors and the level of their occurrence for the corresponding image.

We define similarity between between the image features of the train and test images using Euclidian distance, and retrieve the images from the training set that are similar to the query image. Finally, we measure the performance of the CBIR model using Precision and Recall measure.

In figure 7 we can see at row 4 that the queried image of an airplane(class 0) from test set is found similar to the image of a bird (class 2) from the train test mainly because of the similar shape of the two images (wigs of the airplane might be confused with a bird wigs); similarly at row 5 such airplane wigs from the test set are confounded with a deer ears (class 4) from the training set. This happens because similar shapes are similar patters, aka pixel intensity distributions. For the same reason, the top least similar images according to the main background colors of images are selected in such a way that for a queried image one least similar image from the training set is one has the opposite level of pixel intensity for the back-ground (eg: at row6 the white background of the image determines the selection of images with almost white background to be the most similar and almost black background to be the least similar.
The hope is that in the distribution of such pixel intensity values resides the key to same class belonging images.

Mean precision is calculated by evaluating the average percentage of the truly similar images retrieved out of the total images retrieved (TP/TP+FP) (eg: in row 1 is 1/4 because only the second image is truly a frog).
We obtain mean precision = 0.147 for such trained model.
Mean recall is calculated by evaluating the average percentage of the truly similar images retrieved out of the total truly similar images (TP/TP+FN);ì (eg: if we queried 10 images from class 0 and we predict 3 belonging to such class, 4 to another class and 3 to a third class, we0d obtain recall value of 3/10);
We obtain mean recall = 0.139 for such trained model. (Table 1)

If we'd wanted to tune the model, a good strategy might be to preprocess the input images to reduce color entropy before feeding the model (polarization). In such manner, the conversion to HSV would lead to more extreme pixel differences for the same pattern.

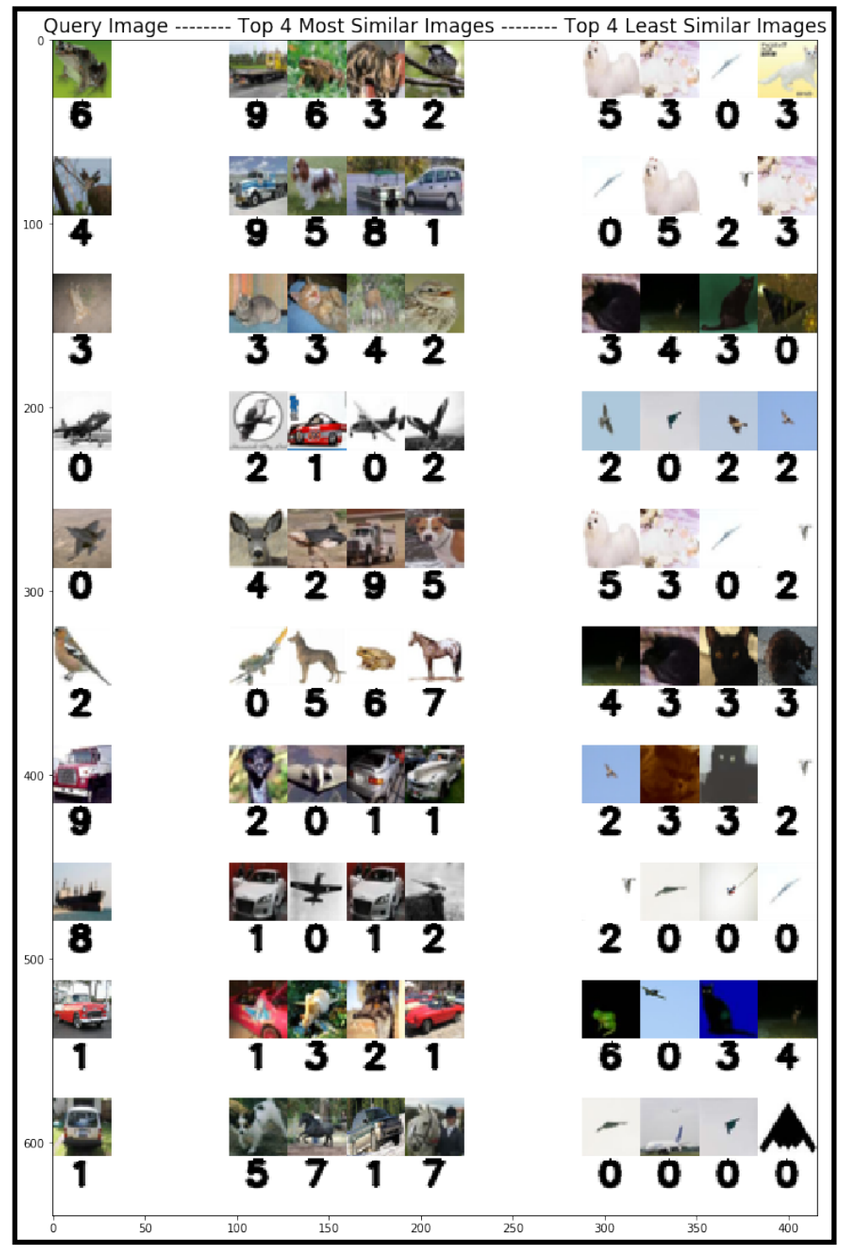


Figure 7: First 4 most similar and least similar images from training set for the first 10 queried images from the test set, using Euclidian distance

CBIR trained with mean and moments distance as similarity measure

```
def get_mean(img):
    mean_red = img[:,:,0].mean()
    mean_blue = img[:,:,1].mean()
    mean_green = img[:,:,2].mean()
    #mn_value = (mean_red, mean_blue, mean_green)
    #x = cv2.normalize(mn_value, mn_value)
    #return (x[0][0],x[1][0],x[2][0])
    return (mean_red, mean_blue, mean_green)

def get_moment(img):
    red = cv2.moments(img[:,:,0])
    blue = cv2.moments(img[:,:,1])
    green = cv2.moments(img[:,:,2])
    moment_red = ((red['m10']/red['m00'],red['m01']/red['m00'])) #(cX,cY)
    moment_blue = ((blue['m10']/blue['m00'],blue['m01']/blue['m00']))
    moment_green = ((green['m10']/green['m00'],green['m01']/green['m00']))
    #mt_value = (moment_red, moment_blue, moment_green)
    #x = cv2.normalize(mt_value, mt_value)
    #return (x[0][0],x[1][0],x[2][0])
    return (moment_red, moment_blue, moment_green)

images_list = [X_train[i] for i in range(X_train.shape[0])]
color_mean = [get_mean(X_train[i]) for i in range(X_train.shape[0])]
color_moment = [get_moment(X_train[i]) for i in range(X_train.shape[0])]

X_train_mm_df = pd.DataFrame()
X_train_mm_df['IMAGES'] = images_list
X_train_mm_df['MEAN'] = color_mean
X_train_mm_df['MOMENT'] = color_moment
X_train_mm_df['SIMILARITY_MEAN'] = 0.0
X_train_mm_df['SIMILARITY_MOMENT'] = 0.0
print(X_train_mm_df.shape)
X_train_mm_df.head(10)
```

	IMAGES	MEAN	MOMENT	SIMILARITY_MEAN	SIMILARITY_MOMENT
0	[[[59, 62, 63], [43, 46, 45], [50, 48, 43], [6...	(141.205078125, 105.099609375, 64.037109375)	((15.40869607314273, 16.358693998367844), (15....	0.0	0.0
1	[[[154, 177, 187], [126, 137, 136], [105, 104,...	(130.19921875, 130.365234375, 130.4775390625)	((15.213697458822118, 14.619655875911313), (14...	0.0	0.0
2	[[[255, 255, 255], [253, 253, 253], [253, 253,...	(133.484375, 135.15234375, 132.5849609375)	((14.608290413203793, 10.831792110499824), (14...	0.0	0.0
3	[[[28, 25, 10], [37, 34, 19], [38, 35, 20], [4...	(99.9794921875, 83.279296875, 57.4140625)	((15.710135867707244, 16.234237490110278), (15...	0.0	0.0
4	[[[170, 180, 198], [168, 178, 196], [177, 185,...	(92.021484375, 102.400390625, 116.0029296875)	((14.6943011779688, 14.140019102196753), (14.8...	0.0	0.0
5	[[[159, 102, 101], [150, 91, 95], [153, 95, 97...	(142.4404296875, 70.7802734375, 57.7783203125)	((14.884854551313255, 14.990957020135884), (15...	0.0	0.0
6	[[[164, 206, 84], [105, 140, 61], [118, 148, 1...	(120.53125, 152.916015625, 85.3779296875)	((15.536103189007001, 14.986680062224528), (15...	0.0	0.0
7	[[[28, 35, 39], [30, 34, 44], [33, 44, 47], [6...	(132.4521484375, 134.7744140625, 97.869140625)	((15.330440680965266, 16.05420589688198), (15....	0.0	0.0
8	[[[134, 186, 223], [131, 184, 220], [128, 182,...	(109.1650390625, 156.2373046875, 188.9833984375)	((14.809026255758823, 16.27113655678311), (15....	0.0	0.0
9	[[[125, 125, 116], [110, 101, 91], [102, 90, 8...	(78.13671875, 71.5634765625, 61.7958984375)	((13.997450382442633, 12.815165225216218), (13...	0.0	0.0

Figure 8: First 10 histograms flattened pixel list of the training set and corresponding mean and moments distances.

We here define similarity between between the image features of the train and test images using mean and moments distance, and retrieve the images from the training set that are similar to the query image. Finally, we measure the performance of the CBIR model using Precision and Recall measure.

Figure 8:
An image moment is a certain particular weighted average (moment) of the image pixels' intensities; these are useful to describe objects after segmentation: **area** (or total intensity), **centroid**, and information about the **orientation** of images.
If the points represent probability density, then the zeroth moment is the total probability (i.e. one), the first moment is the **mean**, the second moment is the **variance**, the third moment is the **skewness**, and the fourth moment (with normalization and shift) is the **kurtosis**.
For an image of pixel intensities I(x,y) the raw image moments Mi,j are calculated by Mi,j = $\sum x \sum y$ I(x,y).
You can imagine similar scenarios where higher order of moments become relevant, for example in tracking cars (figure 9), the orientation of the cars and position could both be easily represented by centroid, variance, and axis of orientation image moments.

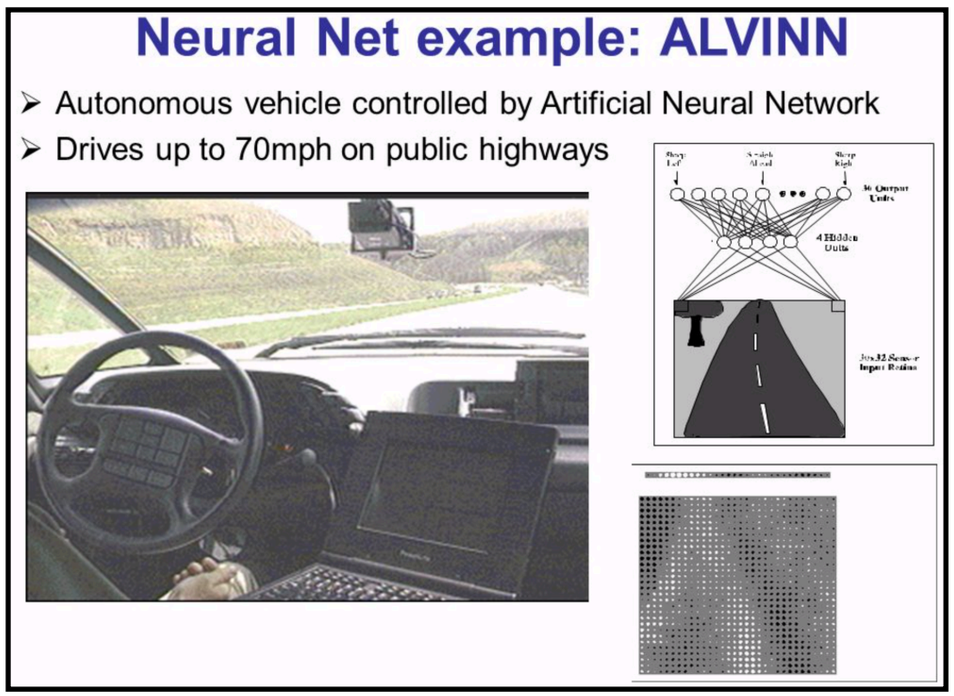


Figure 9: Example of a scenario in which a matrix representation of probability densities pixels with a higher order of moments is required in order to compare the actual image with the recoded images from the training set of images collected recording a human driving the truck.

Figure 10 and 11 show first ten examples of top 4 most and least similar images determined using the mean and moments similarity measures accordingly.

Mean precision score using the former is 0.147 and using the latter is 0.157;

Mean recall score using the former is 0.165 and using the latter is 0.206;

Given the above results (table 2), we conclude that the best feature representation for training our model is given by the moments similarity measure.

Figure 10: First 4 most similar and least similar images from training set for the first 10 queried images from the test set, using mean distance

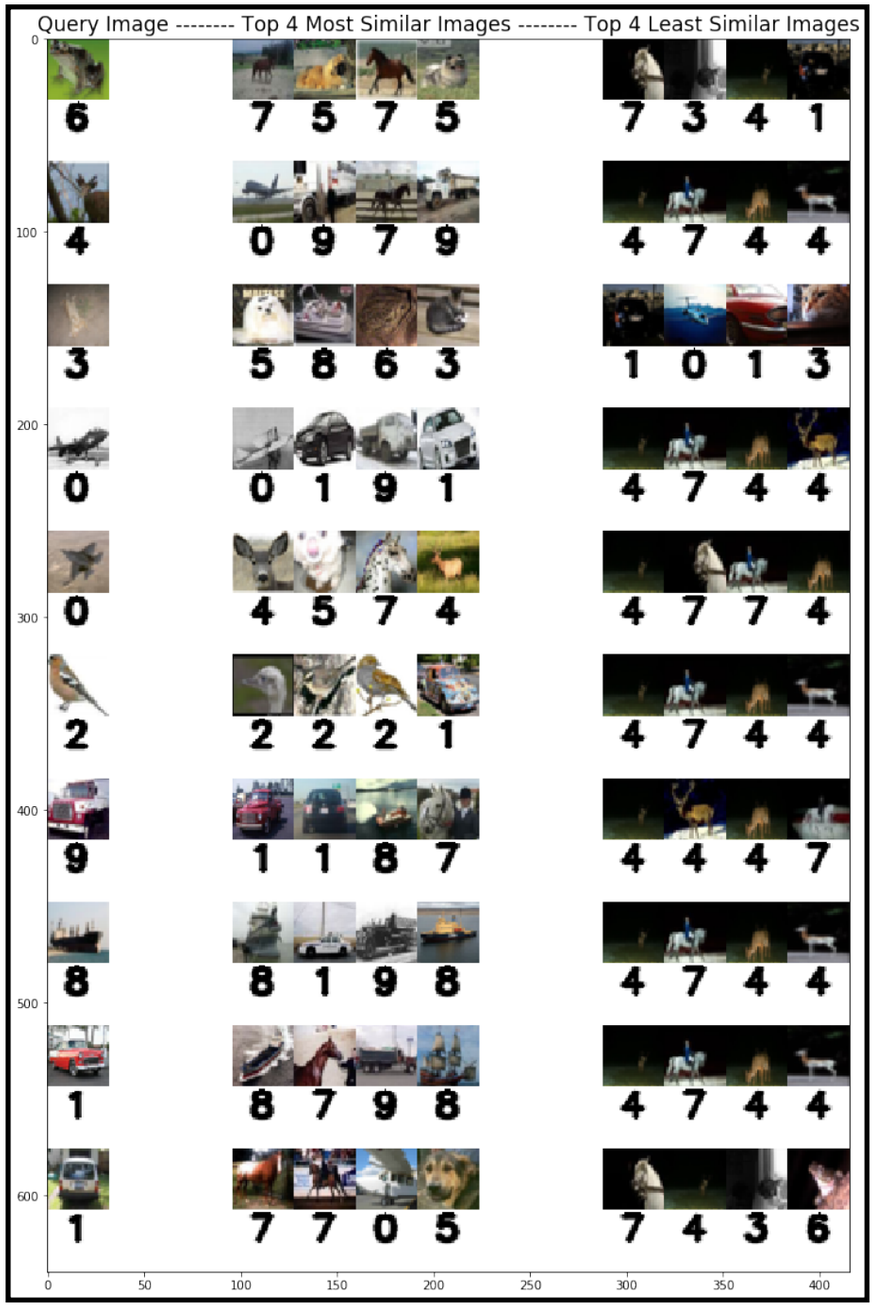


Figure 11: First 4 most similar and least similar images from training set for the first 10 queried images from the test set, using moments distance

	Euclidian	Mean	Moments
Precision	0,147	0,147	0,157
Recall	0,139	0,165	0,206

Tabòe 2

CBIR tested by average precision and recall of the most similar images found, with VGG16 and VGG19 distances as similarity measure

```
def find_sim_vgg16(X_train_dl_df,dl_test_flatten):
    for index,row in X_train_dl_df.iterrows():
        sim = np.linalg.norm(dl_test_flatten - row['VGG16_FEATURES'])
        X_train_dl_df.at[index,'SIMILARITY_VGG16']=sim
    return X_train_dl_df

def find_sim_vgg19(X_train_dl_df,dl_test_flatten):
    for index,row in X_train_dl_df.iterrows():
        sim = np.linalg.norm(dl_test_flatten - row['VGG19_FEATURES'])
        X_train_dl_df.at[index,'SIMILARITY_VGG19']=sim
    return X_train_dl_df
```

Figure 12 and 13 show first ten examples of top 4 most and least similar images determined using the VGG16 and VGG19 similarity measures accordingly.

Mean precision score using the former is 0.282 and using the latter is 0.244;
Mean recall score using the former is 0.194 and using the latter is 0.278;

Given the above results (table 3), we conclude that the best feature representation for training our model is given by:

a) the **VGG19** similarity measure if we're more concern about having a model better in finding all the data points that are relevant in a dataset. (higher **recall**) (eg: more likely error is model says an image is similar while it's not, yet it'll detect more similar images).
b) the **VGG16** similarity measure if we're more concern about having a model that has a better proportion of the truly relevant datapoints selected and the total of the datapoint our model says are relevant (higher **precision**). (eg: more likely error is model detect less similar images, yet among those the ratio of truly similar and actually not simiar is better). Figure 15 shows such trade-off.

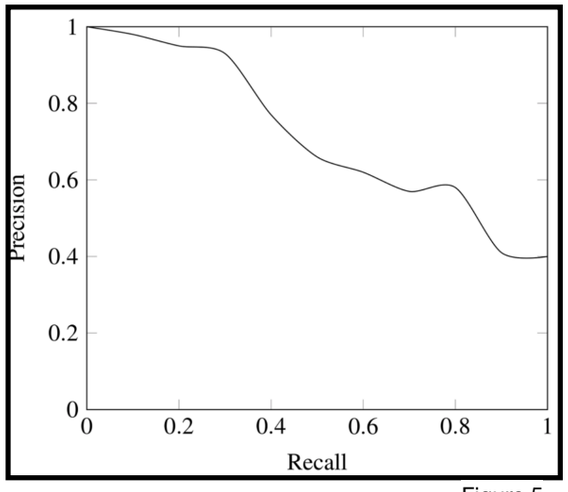


Figure 5

	IMAGES	VGG16_FEATURES	SIMILARITY_VGG16	VGG19_FEATURES	SIMILARITY_VGG19
0	[[[59, 62, 63], [43, 46, 45], [50, 48, 43], [6...	[38.739197, 96.806465, 5.1946206, 0.0, 0.0, 0....	0.0	[0.0, 86.262566, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	0.0
1	[[[154, 177, 187], [126, 137, 136], [105, 104,...	[22.41855, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 30.27544, 0.0, 0.0, ...	0.0
2	[[[255, 255, 255], [253, 253, 253], [253, 253,...	[12.937462, 0.0, 0.0, 0.6051576, 20.543226, 30...	0.0	[0.0, 0.0, 0.0, 0.0, 47.490852, 57.012547, 0.0...	0.0
3	[[[28, 25, 10], [37, 34, 19], [38, 35, 20], [4...	[0.0, 0.0, 0.0, 25.399084, 30.455841, 0.0, 0.0...	0.0	[0.0, 1.5151628, 0.0, 0.0, 51.08864, 0.0, 0.0,...	0.0
4	[[[170, 180, 198], [168, 178, 196], [177, 185,...	[0.0, 0.0, 0.0, 15.977697, 0.0, 29.414213, 19....	0.0	[0.0, 0.0, 0.0, 0.0, 0.0, 44.431396, 0.0, 0.0,...	0.0

Figure 12: First 5 histograms flattened pixel list of the training set and corresponding VGG16 and VGG19 distances.

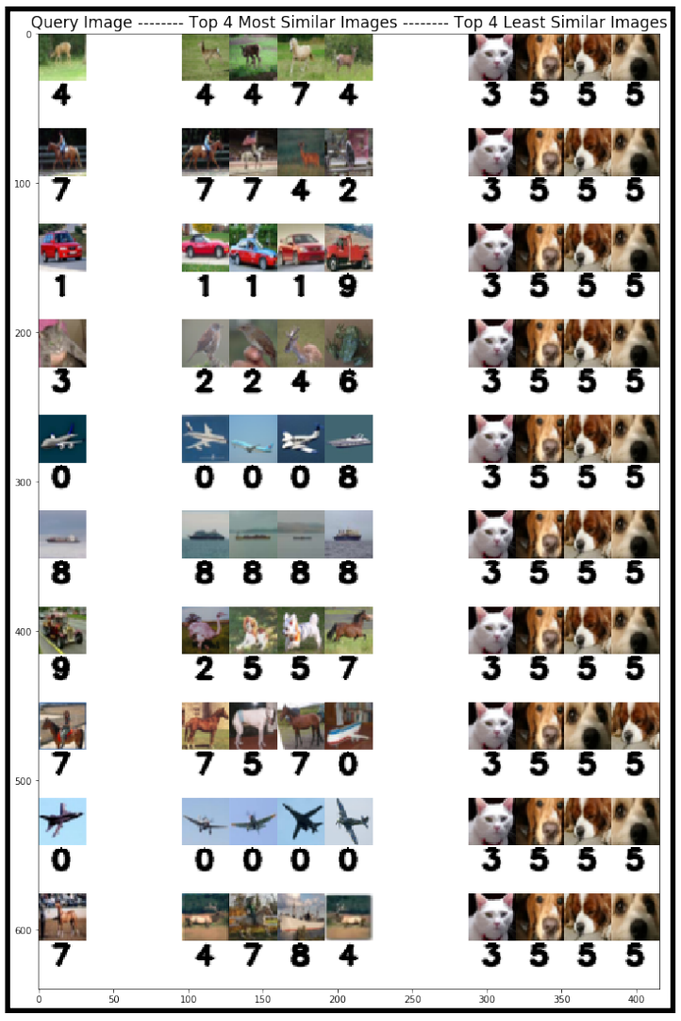


Figure 14: First 4 most similar and least similar images from training set for the first 10 queried images from the test set, using VGG16 distance

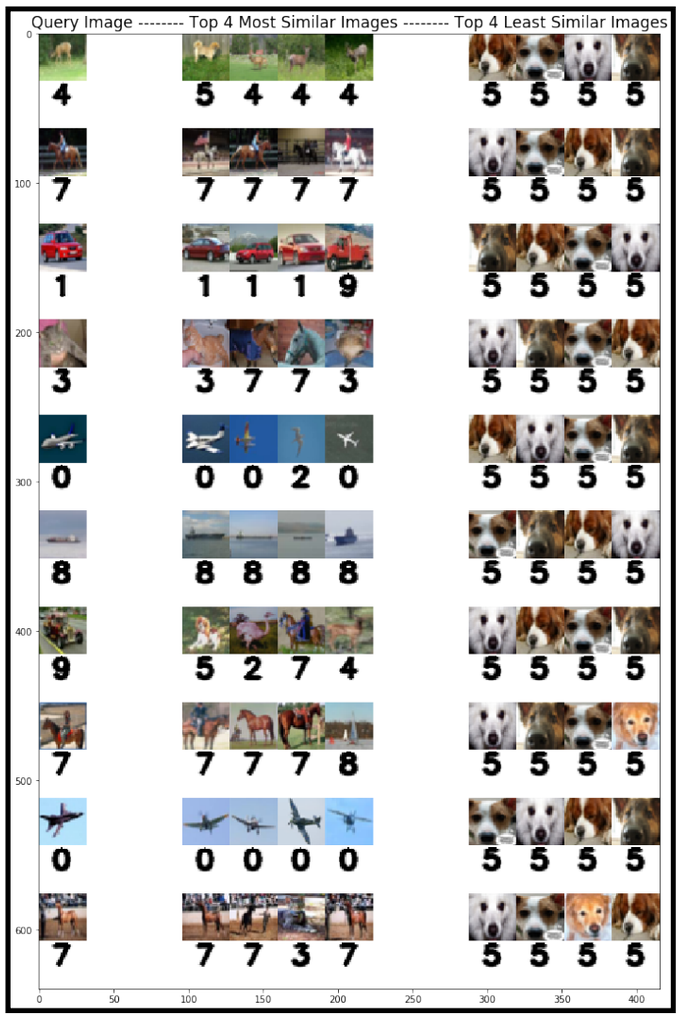


Figure 14: First 4 most similar and least similar images from training set for the first 10 queried images from the test set, using VGG19 distance

	VGG16	VGG19
Precision	0,282	0,244
Recall	0,194	0,278

Tabòe 3

Computer Vision

Assignment 4:

Content Image retrieval systems

Irene Volpe - r0740784

- Thanks for the attention -