Content-Based Image Retrieval

Realized by: **Id Mansour Hassan**

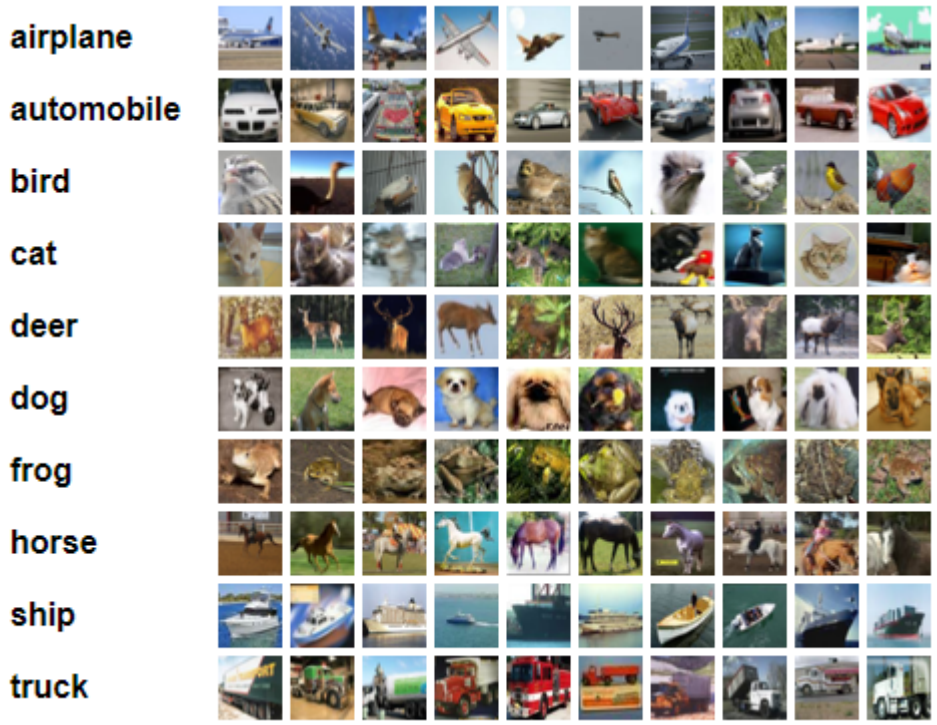
E-mail : h.idmansour000@gmail.com

I. Introduction

In recent times, technology has developed faster and has become one of the important means in our daily life in various fields, such as e-commerce, medicine and browsing websites from a set of images. For example, most search engines (such as Google search engine) rely on CBIR to find images that are very similar to the image of the search. So, when you search for an image that contains a cat, the engine often suggests other images of cats. In this work we will try to improve the search by the content of images by CBIR Content- based image retrieval by exploiting the correlation coefficient and sorting the result in ascending order by the end.

$$r_{xy}=\frac{\operatorname{cov}(x,y)}{\sigma_x\sigma_y}=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\sqrt{n\sum_{i=1}^{n}x_i^2-(\sum_{i=1}^{n}x_i)^2}\sqrt{n\sum_{i=1}^{n}y_i^2-(\sum_{i=1}^{n}y_i)^2}}$$

We need a query image and an image database; in our case we will use **mnist (**database of handwritten digits**)** and **cifar 10(**consists of 60000 32x32 colour images in 10 classes, with 6000 images per class**)** from **keras**.



II. Code and explanation

First of all, we import the necessary libraries.

```
import cv2
import math
import numpy as np
import  panda ace pd
from math import sqrt
import matplotlib.pyplot ace plt
from keras.datasets import mnist, cifar10
```

We create a function which takes as input a list or an array of images, the desired number of images and a list of indices. The function must guarantee a square display form.
Example 1: n_imgs=81 we obtain a 9*9 square.
Example 2: n_imgs=60 we obtain a 7*8 square.

```
def showImgs(imgs, n_imgs, i_imgs):
    not = sqrt(n_imgs)
    m = not
    p = 1
    if not != int(n):
        m = int(n) + 1

    fig = plt. figure()
    for I in i_imgs:
        fig.add_subplot(int(n), int(m), p)
        plt.imshow(imgs[i])
        plt.axis('off')
        p += 1
    plt.show()
```

Create a function for Euclidean distance calculation.

```
def calculateDistance(img1, img2):
    #return sqrt(np.sum(np.square(i1-i2)))
    return np.linalg.norm(img1 - img2)
```

Realize a function for the calculation of the cosine similarity.

```
def cosineSim(img1, img2):
    p_s = np.sum(img1 * img2)
    s1 = np.linalg.norm(img1)
    s2 = np.linalg.norm(img2)
    return p_s / sqrt(s1 * s2)
```

Create a function for calculating the correlation coefficient.

```
def pearsonCorr_coef(img1, img2):
    v1 = img1 - img1.mean()
    v2 = img2 - img2.mean()
    return np.sum(v1 * v2) / (np.linalg.norm(v1) *
np.linalg.norm(v2))
```

This function compares two images, a query image and the other of the database according to the chosen criterion distance, cosine similarity and correlation.

```
def compare(img, imgS, n, t='dist'):
    dict = {}
    if you == 'cossim' :
        for I in range(0, n):
            dic[i] = cosineSim(img, imgS[i])
        return {k: v for k, v in dic.items(sorted(),
key=lambda item: item[1], reverse=True)}

    elif you == 'dist':
        for I in range(0, n):
            dic[i] = calculateDistance(img, imgS[i])
        return {k: v for k, v in dic.items(sorted(),
key=lambda item: item[1])}

    elif you == 'correct':
        for I in range(0, n):
            dic[i] = pearsonCorr_coef(img, imgS[i])
        return {k: v for k, v in dic.items(sorted(),
key=lambda item: item[1], reverse=True)}
```

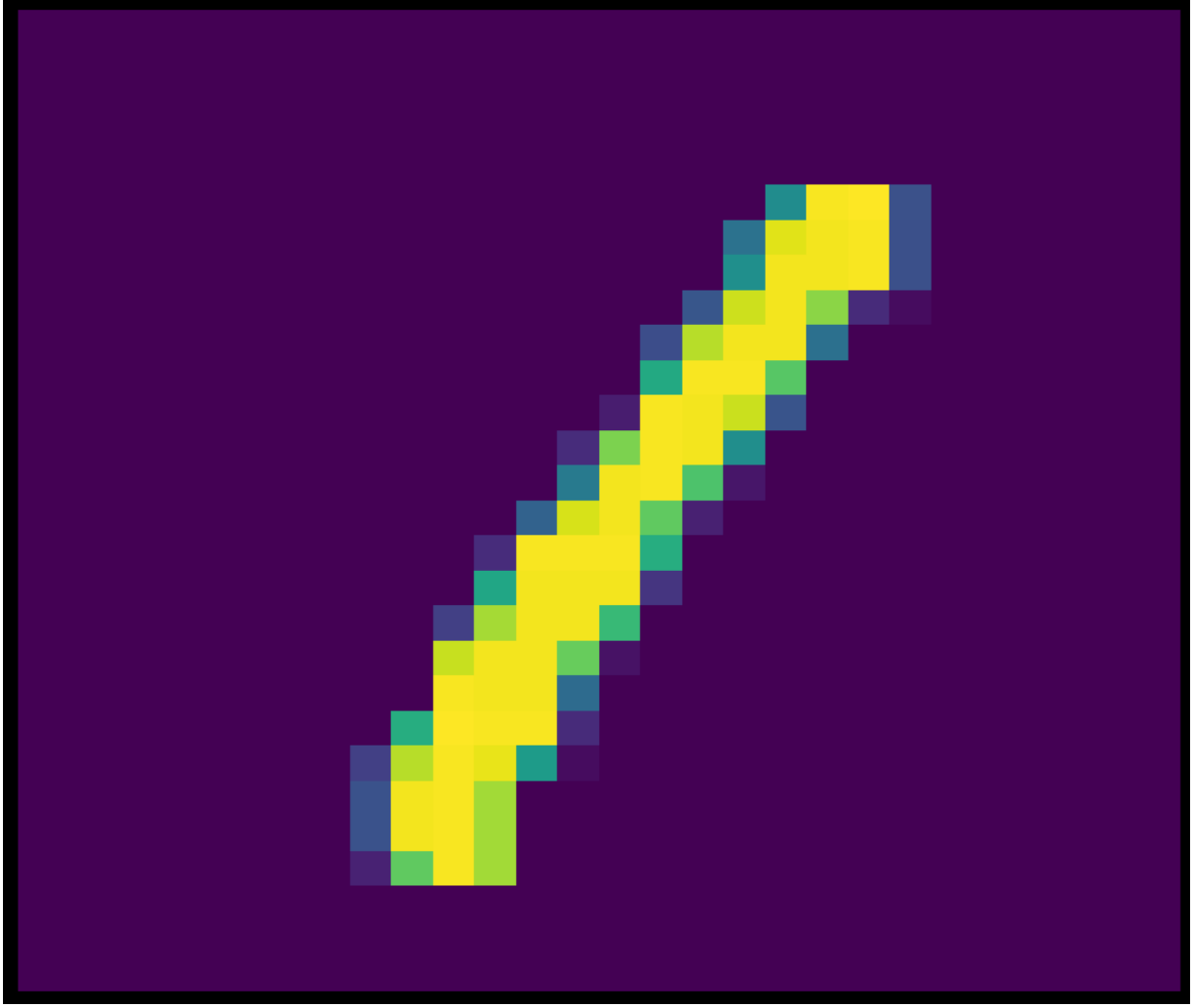
III. Test and result

To test this search, we first use the digits mnist image database. We must load it with this piece of code:

```
(trainX, trainy), (testX, testy) = mnist.load_data()
```

Choose the digit 1 as a query image.

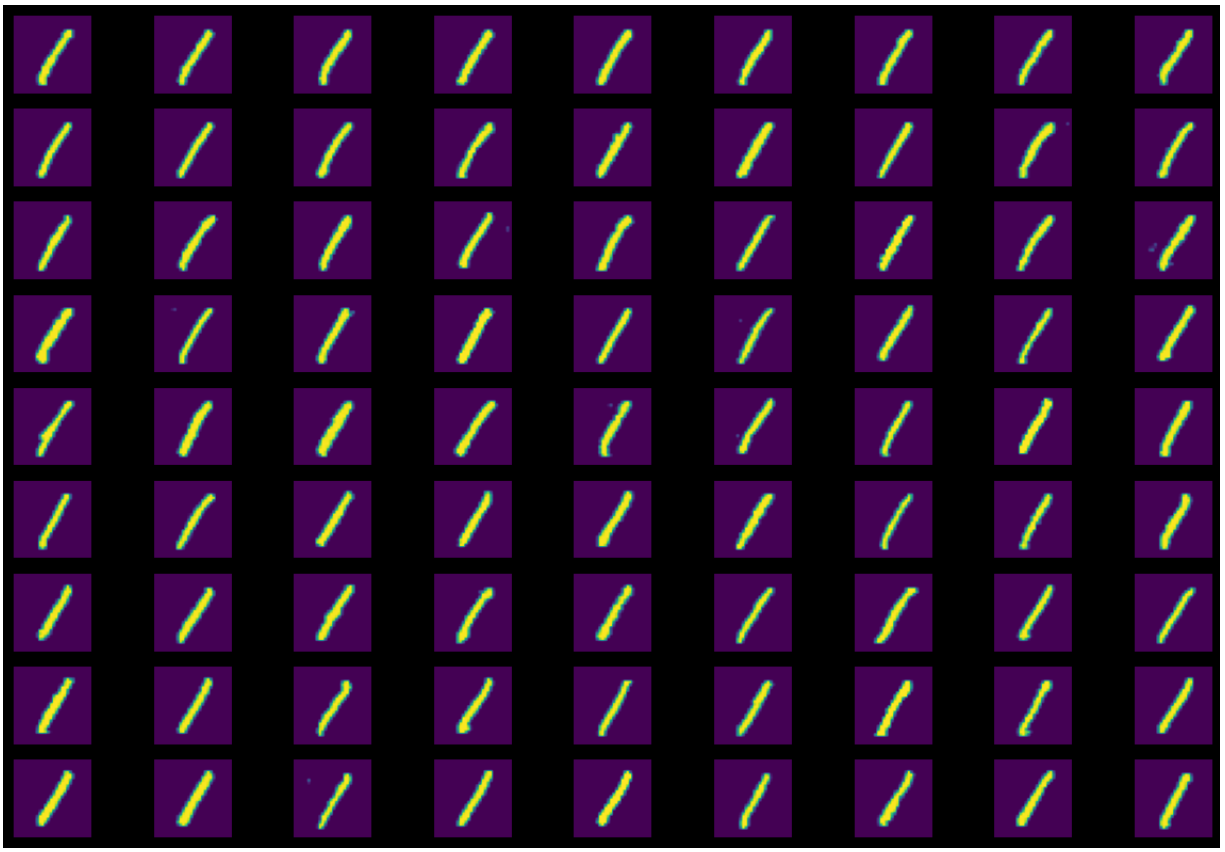
```
img_req = trainX[3]
plt.imshow(img_req)
plt.axis('off')
plt.show()
```



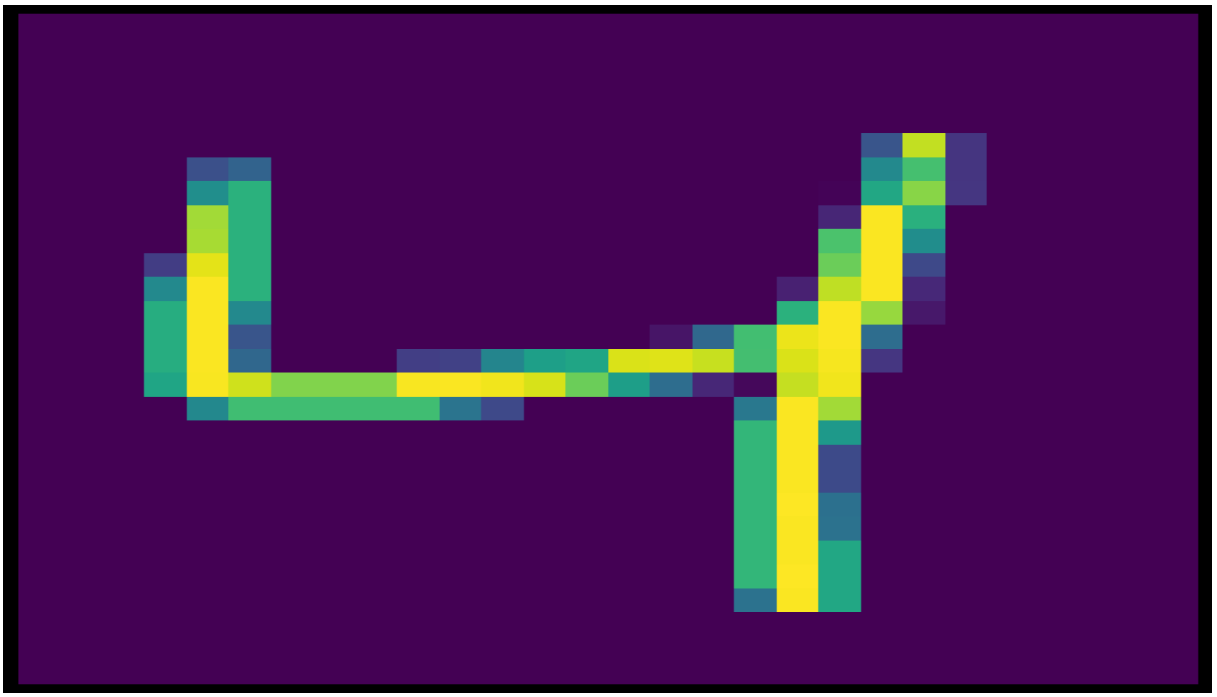
The comparison with the other images in the database.

```
dict = compare(img_req, trainX, len(trainX),
               t='corr')
showImgs(trainX, 81, [k for k in dic][0:81])
```

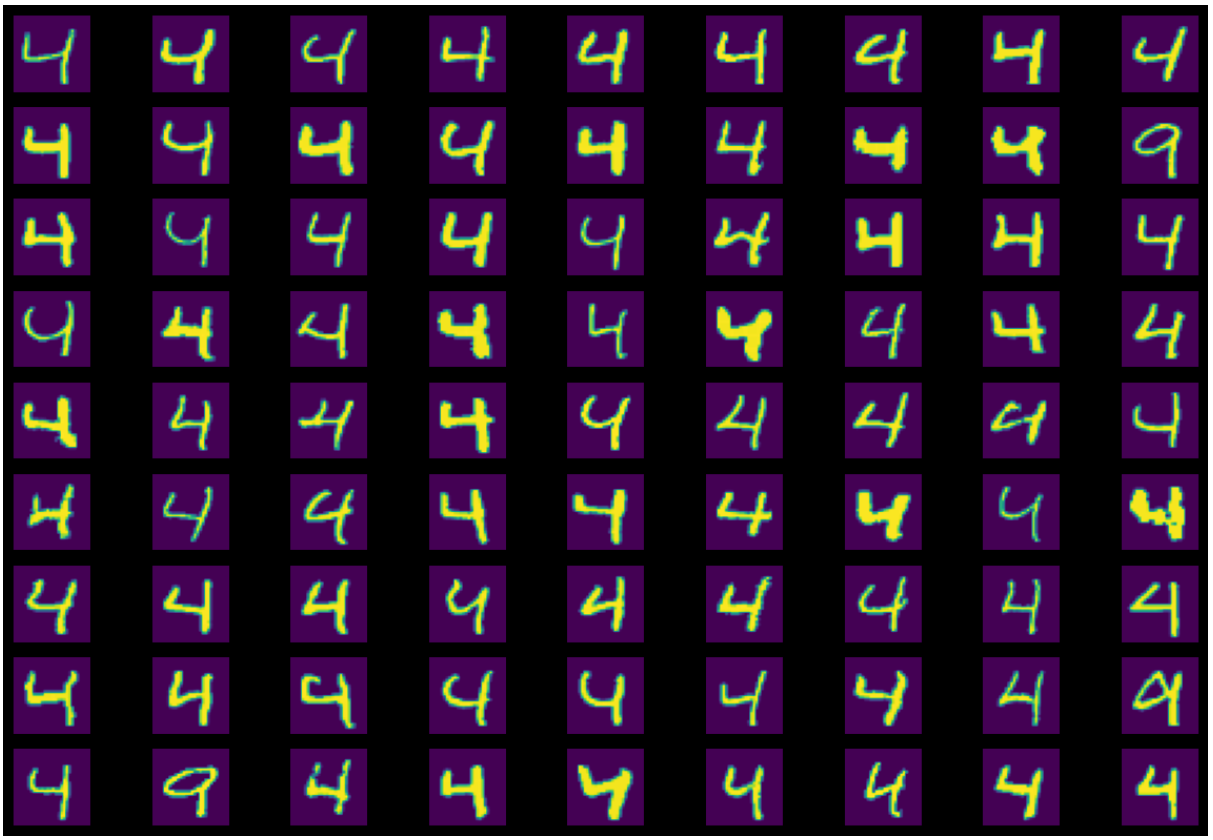
the search result:



Digit 4 as a query image.

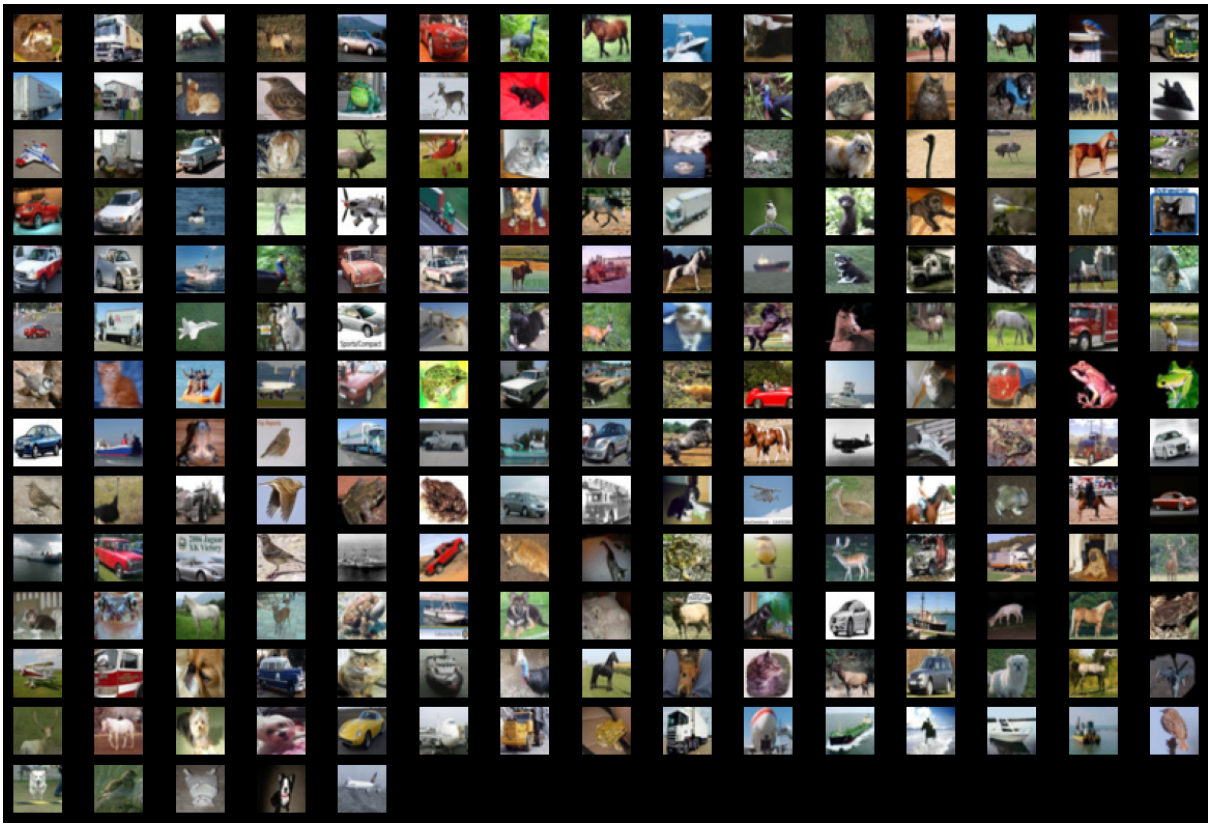


The search result:



Now we will try with the other cifar10 base.

```
(trainX, trainy), (testX, testy) = cifar10.load_data()
showImgs(trainX, 200, [i for I in range(0, 200)])
```



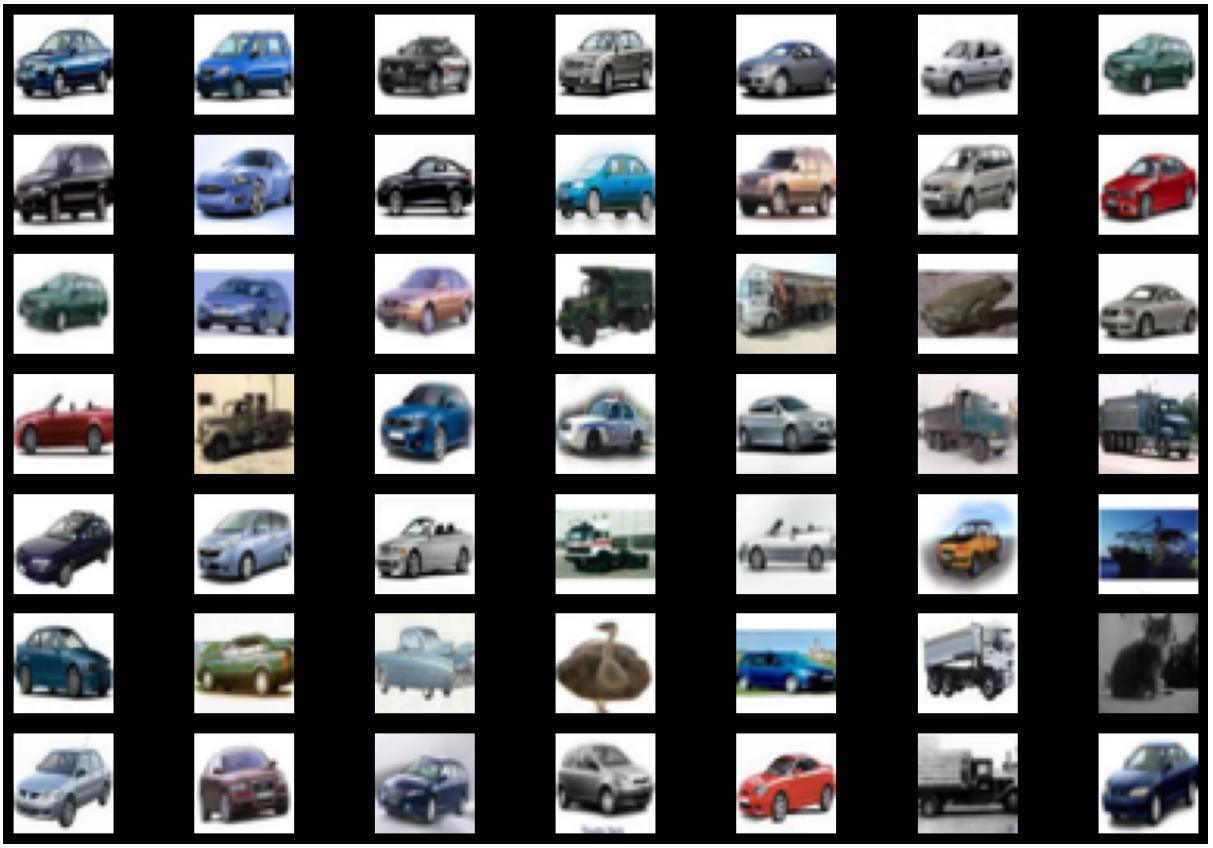
Select the horse query-image of index 7 in the table above.

```
img_req = trainX[7]
plt.imshow(cv2.resize(img_req, (128, 128)))
plt.axis('off')
plt.show()
```

The search result:

Test image-query car:

The search result:



Testing with plane as a query image:

The search result:

VI. Conclusion

During this Article, the correlation coefficient was used as an effective method without using Deep Learning to measure the similarity between the query images and the others in the database.