**DESIGN PROJECT REVIEW**

**Detection of Deep Network Generated Images Using Disparities in Color Components**

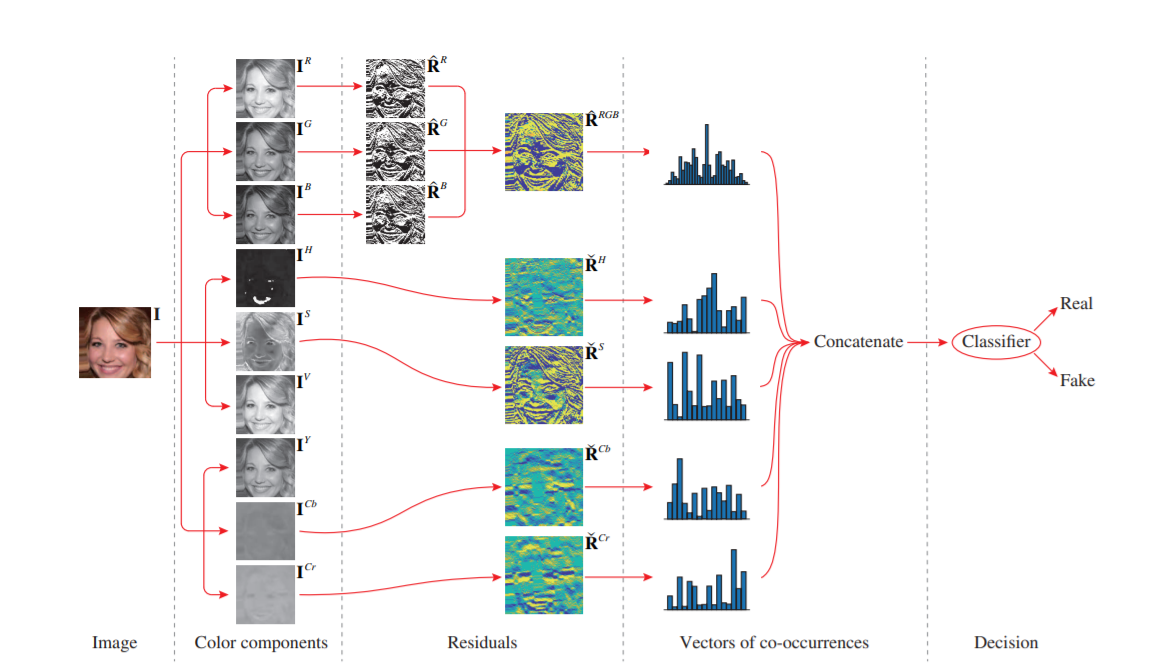
**COE18B025**

In this project we approached the problem of detecting DCGAN images from real images through the evaluation of the color components of an image using a co-occurrence matrix .

**Exacting Features from Color Components :**

The features from the colour components are computed and then they are concatenated into the feature vector. A classifier is trained to predict whether the image is real or generated. The images are passed through the high-pass filter to suppress the image contents. The co-occurrence matrix on the high-pass filtering residuals are computed which is then processed for the truncation or quantization to reduce the complexity of the co-occurrence matrix. The co-occurrence matrices which are extracted are then merged to form a feature set.

From the experiments they conducted in the paper they found visible differences between the RGB , H ,S , Cr, Cb components of the real and fake images. So these 5 components are used to differentiate between the images.

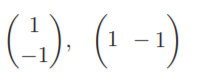
****

1. **Suppressing image components:**

Instead of using the conventional gray level co-occurrence matrix of the image pixels, we compute the co-occurrence matrix on image high-pass filtering residuals. They are then passed through a high-pass filter to enhance high frequency disparitie. The image is denoted in their RGB, HSV, and YCbCr spaces.

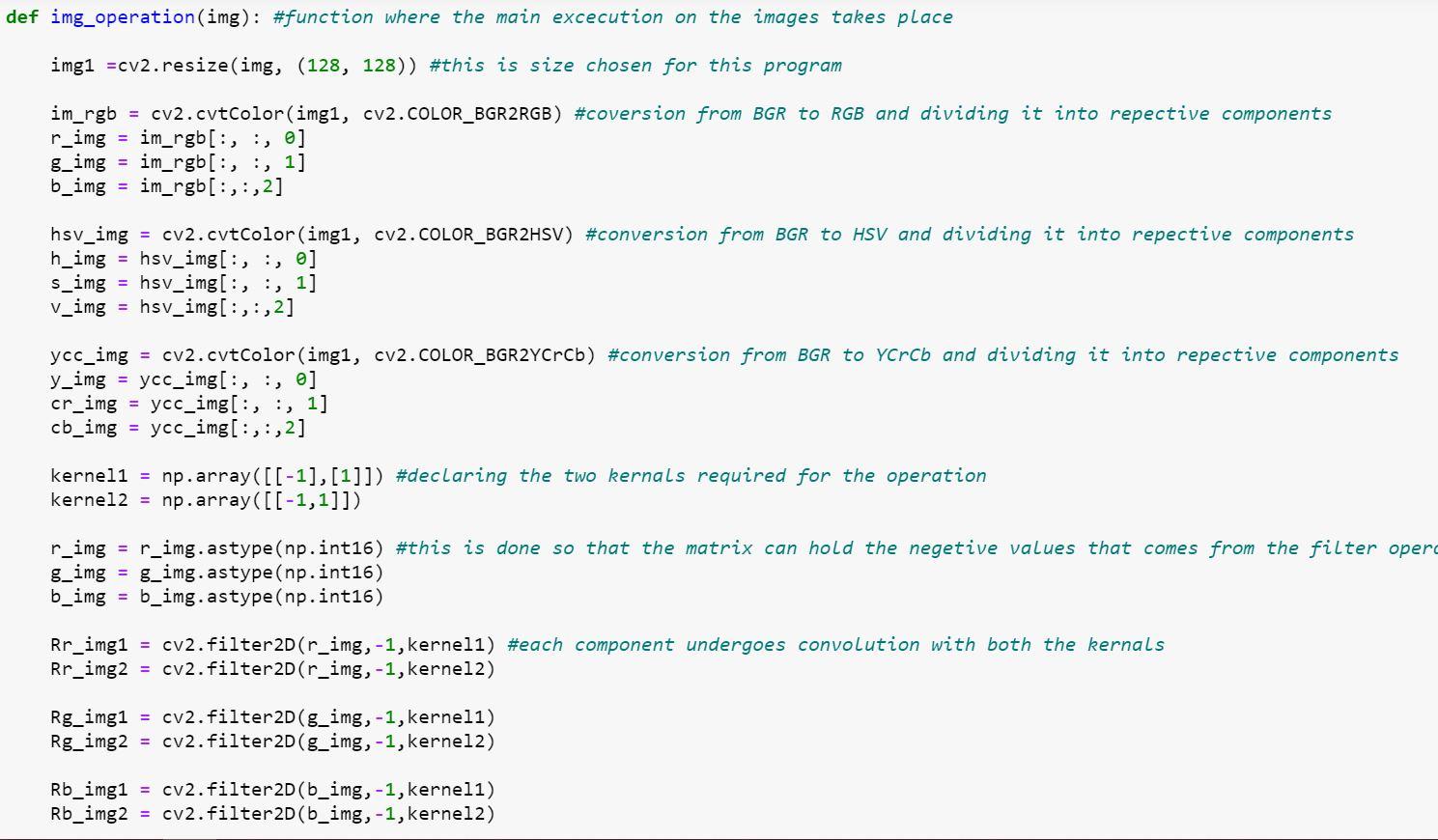


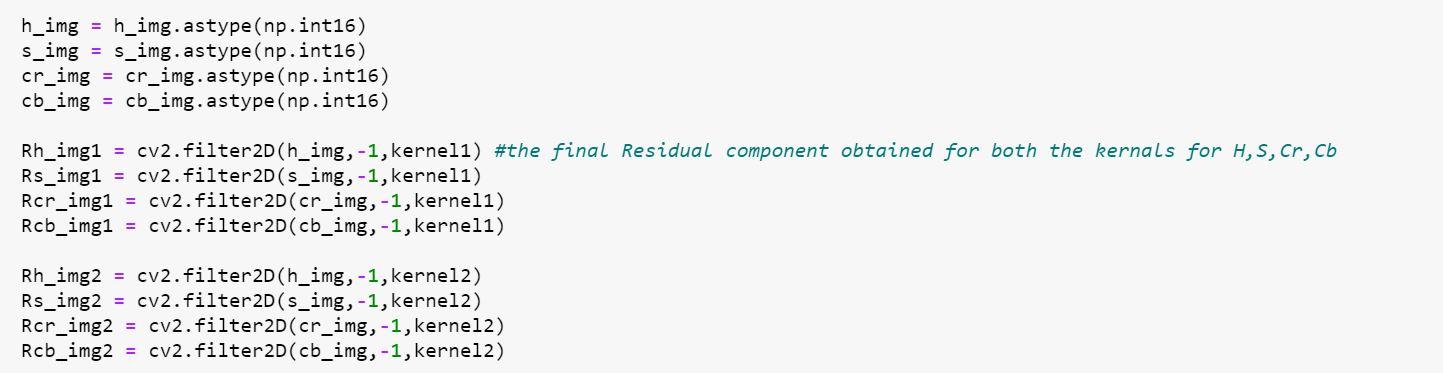
Where, f(·) is a high-pass filtering operation performed in spatial domain and Rc is the residual value. We get two different values of Rc from the two high pass filters.



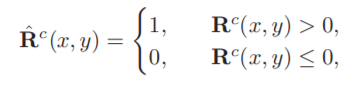
These are the high-pass filters used for computing image residuals.

Implementation:-





1. **Binarization of RGB residuals:**

We then binarize the residual image of each color component to reduce the number of bins in the resultant co-occurrence matrix and obtain an assembled residual image.

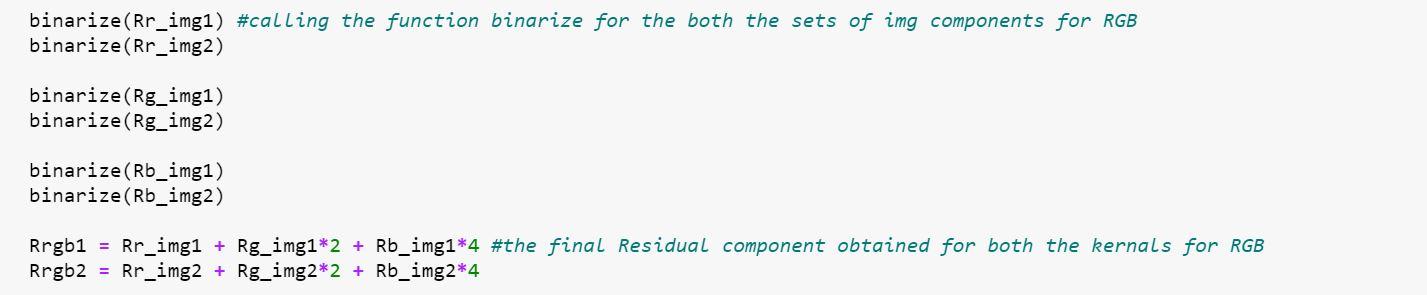


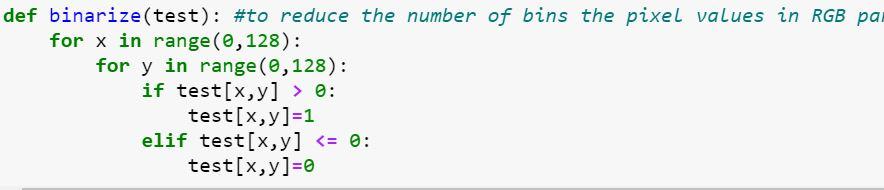
Where,



The final R^rgb matrix will have elements within the ranges of [0,7].

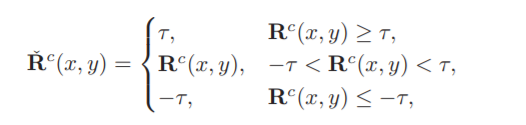
Implementation:-





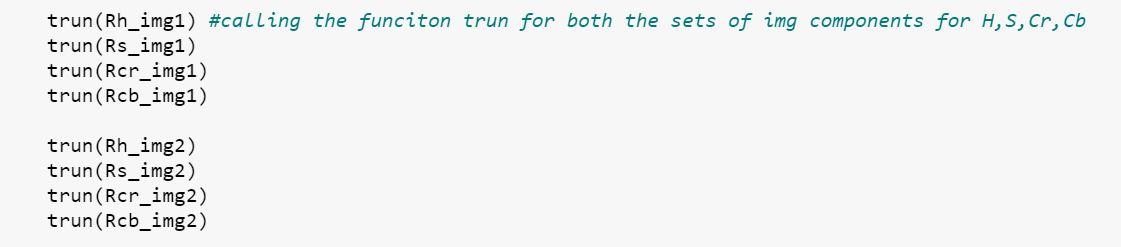
1. **Truncation of chrominance residuals:**

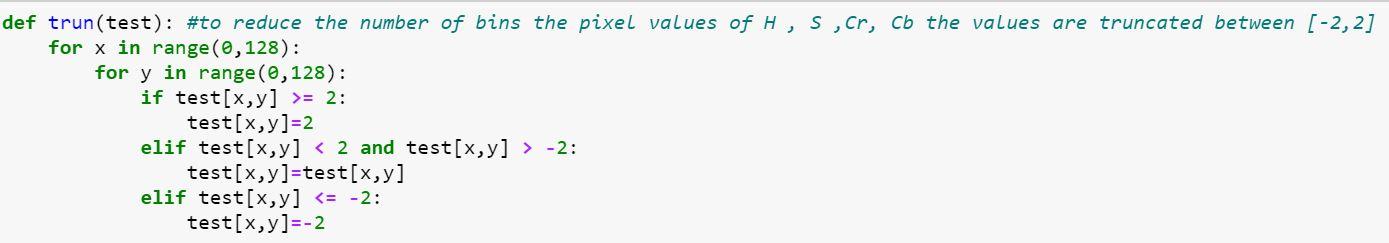
H, S, Cb, Cr components are processed independently as they represent different chrominance information.



Where, c ∈ {H, S, Cb, Cr} and τ is the truncation threshold. For experimental purposes, we set the value of τ = 2. So, the values of R^c ranges from [-2,2]

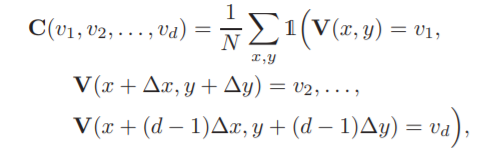
Implementation:-





1. **Formation of co-occurrence matrix:**

The co-occurrence matrix of a 2-D array V is computed by:

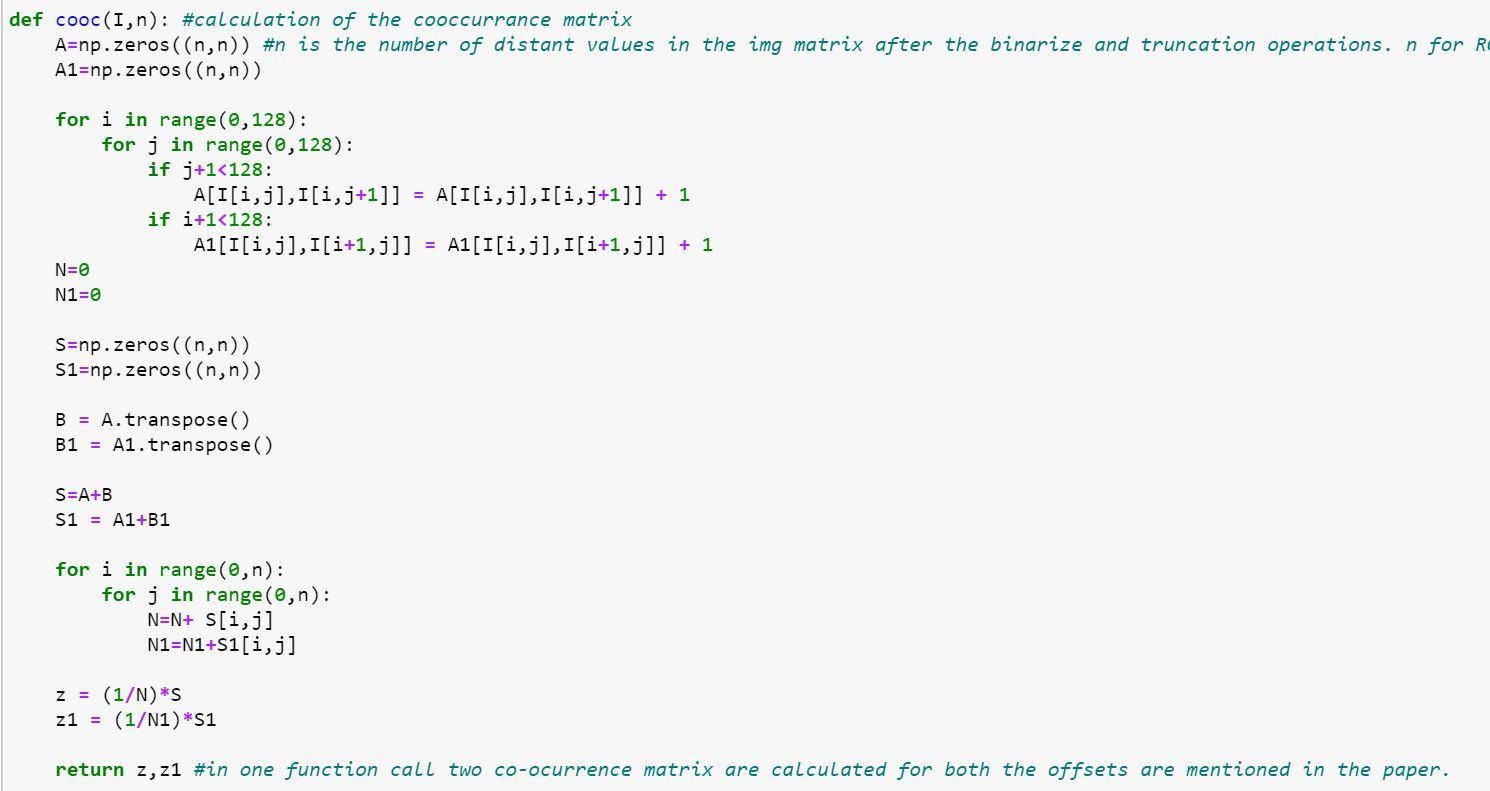


Where **1(·)** is an indicator function, **(v1, v2, . . . , vd)** is the index of co-occurrence matrix, **d** is the order of co-occurrence matrix, N is the normalization factor, and **∆x, ∆y** are the offsets for two neighboring elements.



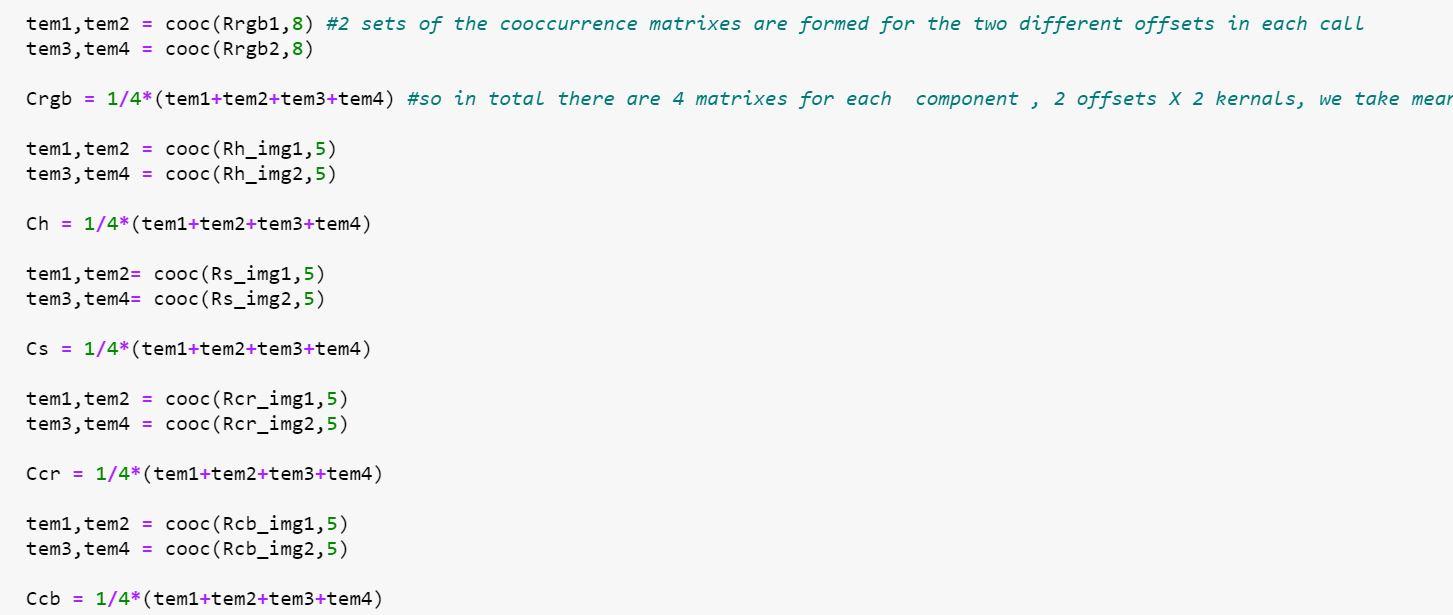
For each component residual 2 co-occurrence matrices are calculated, one along the x offset and the other along the y offset.

Implementation:-



While calculating the co-occurrence matrix we make the matrix symmetric and normalize the values by dividing it with the sum of all the elements. S and S1 show the co-occurrence matrices calculated in the x offset and y offset respectively.

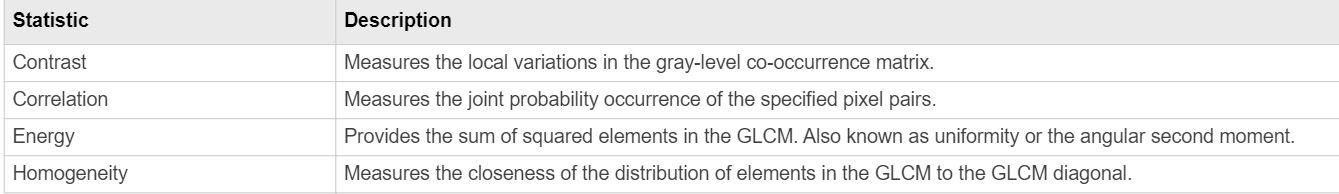
Finally we have 4 matrices for each component, 2 residuals (from the 2 different high pass filters) x 2 offsets. Therefore, we take the mean of the 4 matrices to form a single matrix for every single component.



1. **Feature Extraction :**

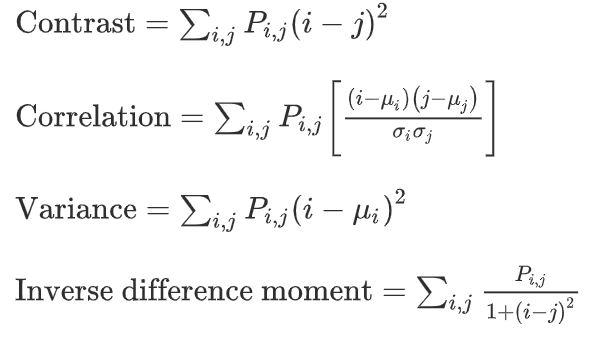
We use the 5 co-occurrence matrices to extract the features of the image. In this code implementation, we primarily use the following 4 features :- Energy, Homogeneity, Contrast and Correlation.

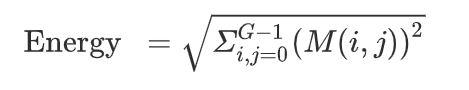
These 4 features are the main features used for the extraction of image properties.



The formulas used were taken from :- <https://www.sciencedirect.com/topics/engineering/cooccurrence-matrix>

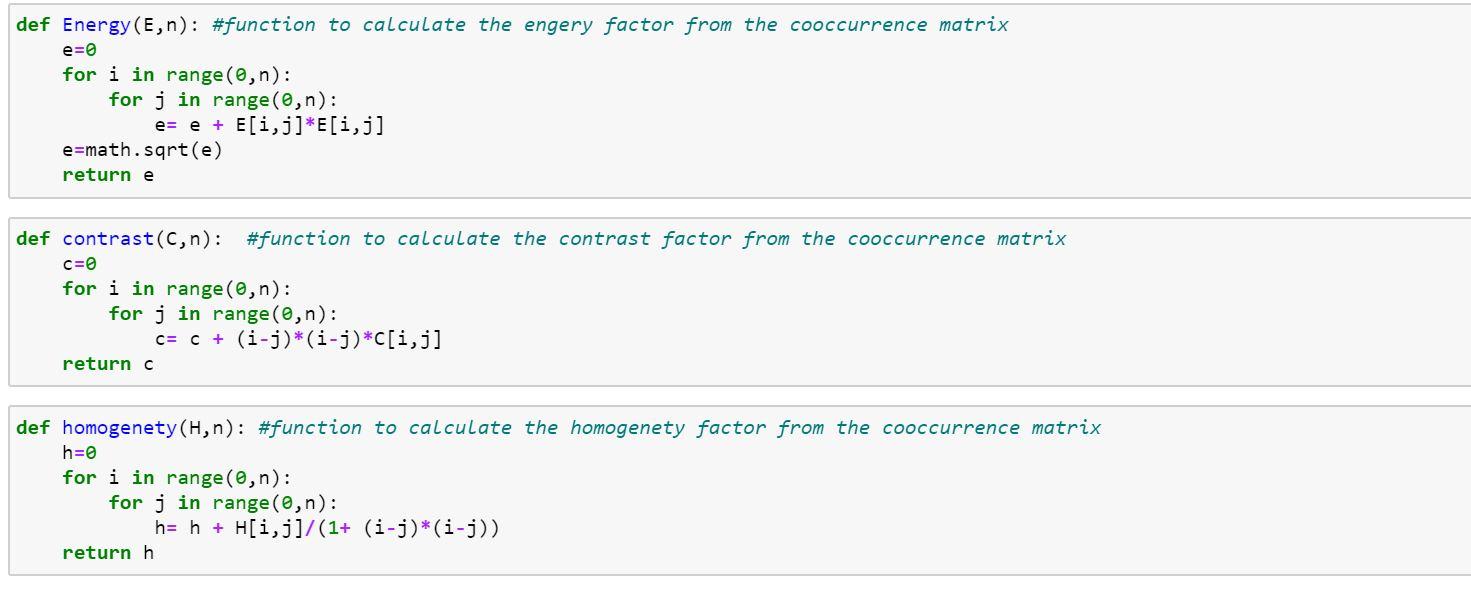
Following are the formulas used to implement the feature extraction:-

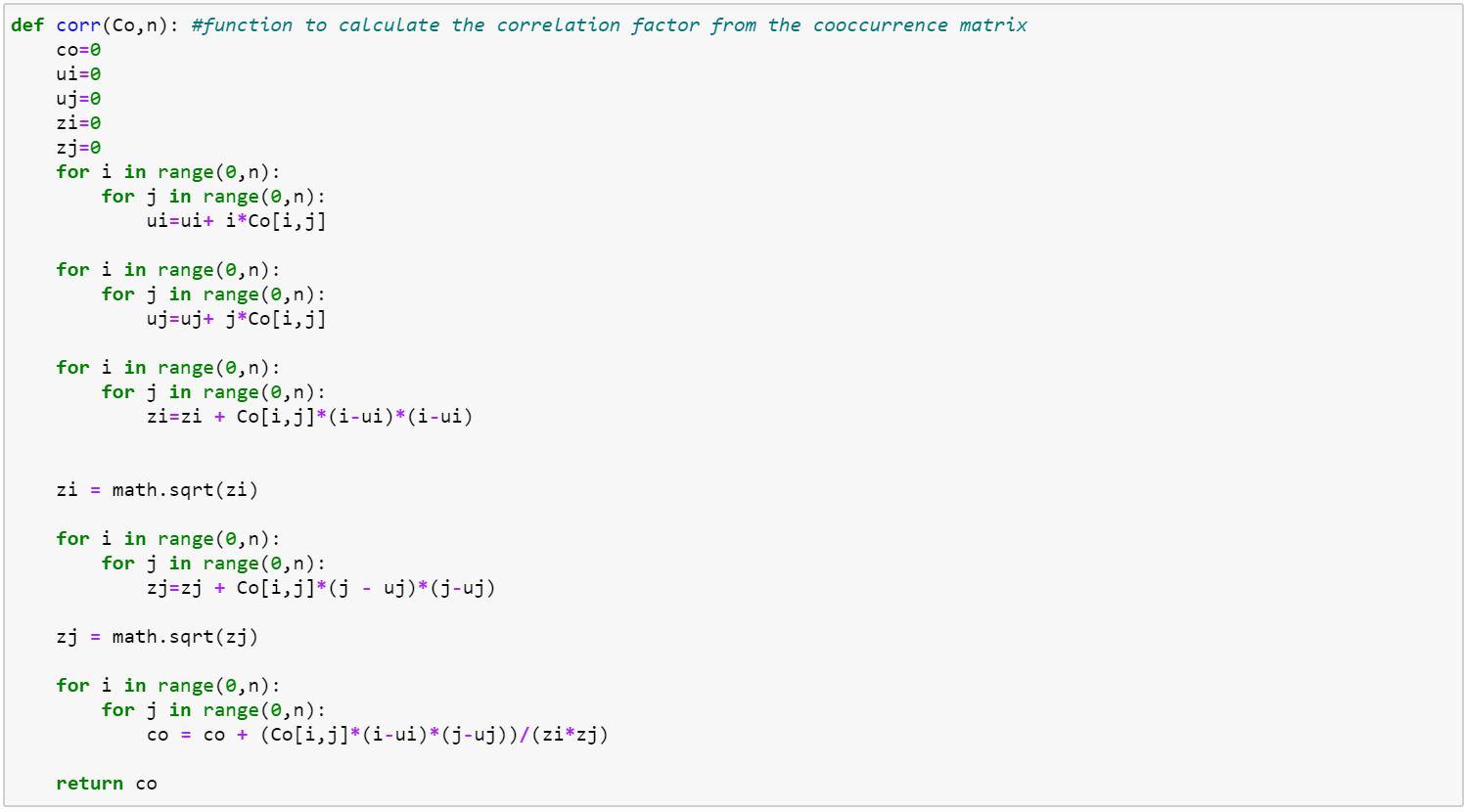




M and P represent the co-occurrence matrix.

Implementation:-





**μi** and **μj** are the means, where as **𝝈i** (zi) and **𝝈j** (zj) variance.

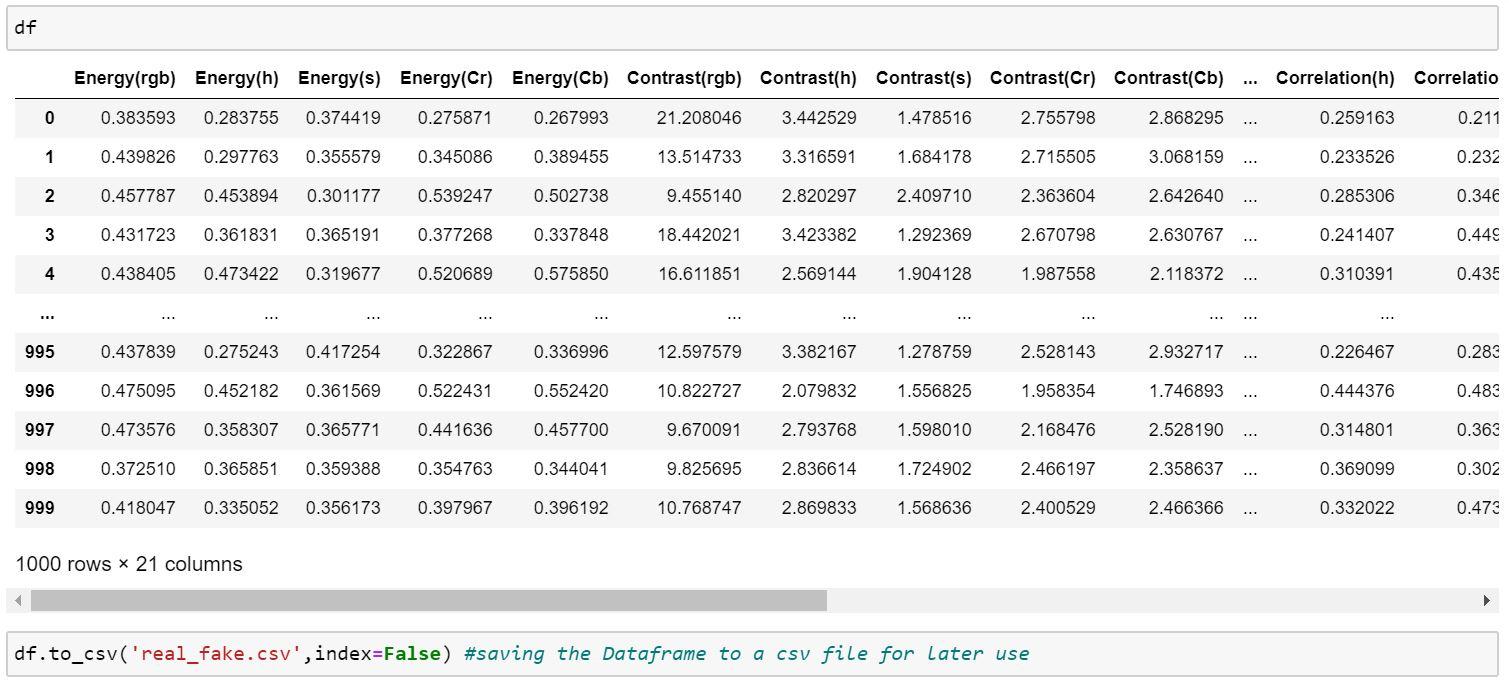
1. **Experiment :**

We used 500 real images from the CelebA dataset and 500 fake DCGAN images.



Data is returned from the func **img\_operation** in the form of a dictionary (when flag is set to 0) . Depending on the type of image, it is labeled as 0 if the image is real, else 1 if the image is fake.

After the images are fed into the code and a data frame is generated, the Data Frame looks like this:-

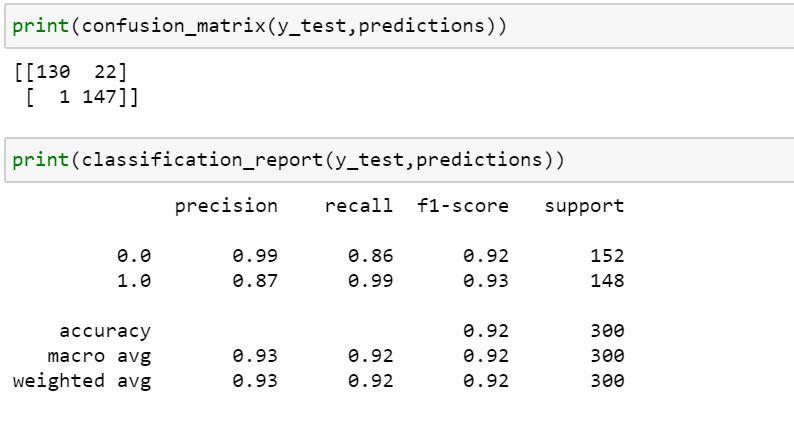


df has 21 columns (5 components x 4 features + Type (0 or 1)) and 1000 rows.

It is saved into a .csv file for later use.

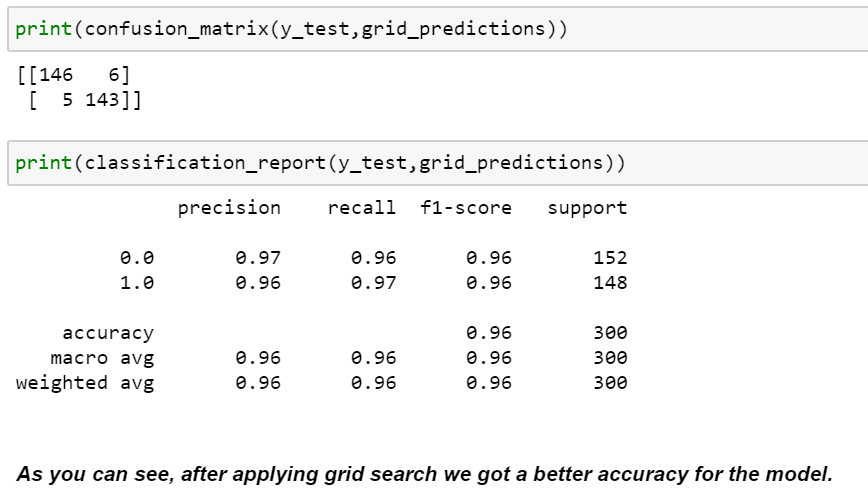
**The classifier used for this implementation is SVM ( or SVC (the linear SVM classifier) ).**

After training the SVC model we get the following output : -



Further to increase the working of the SVC model, we train the model using GridSearchCV, which finds the values of the SVC parameters at which it performs the best.

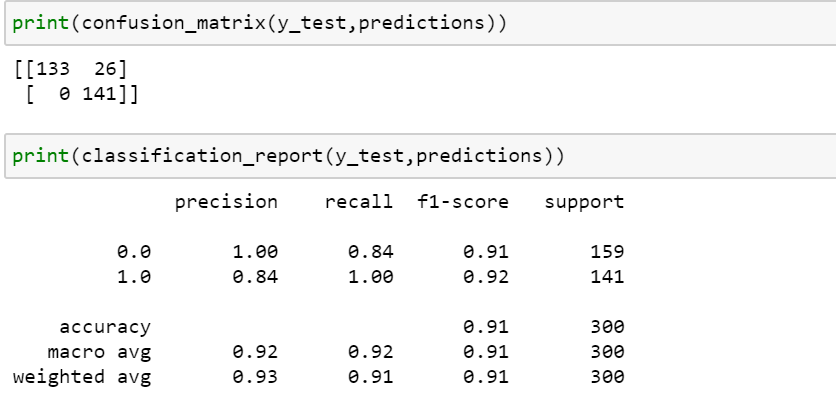
After applying grid search :-



In general ,the more the number of images trained, the higher would be the accuracy.

**If we try to incorporate more number of features , the accuracy decreases due to overfitting of data,**

The following shows the result of the ML model when more features were used :-

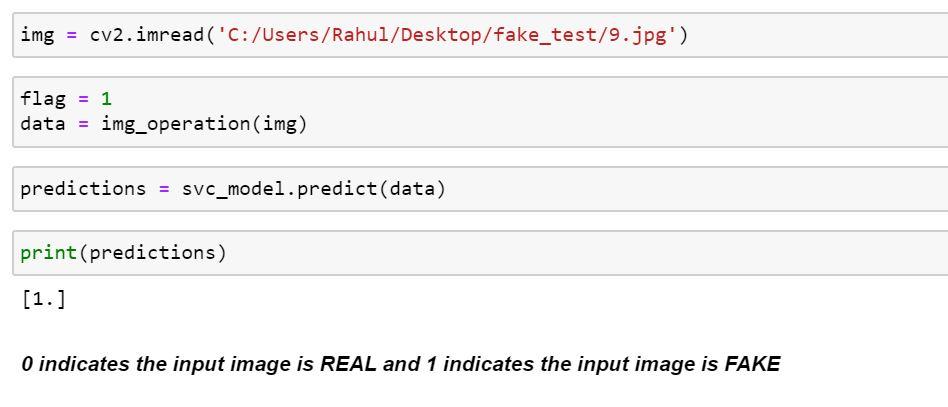


The extra features incorporated were dissimilarity and entropy.

The formulas for which can be found on :-

<https://www.sciencedirect.com/topics/engineering/cooccurrence-matrix>

The last section of the code is for testing the working of the model,



Here , we input an image from the test data set (fake\_test and real\_test) which was not used during training the model.

We apply the image operations and input the values in **svc\_model.predict**, which returns value 0 if the image is detected to be real or returns 1 if the image is detected to be fake.

In the above image, we tested the model with a fake image and it **successfully** detected the image to be a fake, showing the working of our trained model.

**Thank You.**