COMPUTER VISION

ECE 661

HOMEWORK 8

Tanzeel U. Rehman

Email: *trehman@purdue.edu*

1. Introduction:

This homework performs image classification on 4 different classes using the textural features extracted from Gram matrix representation combined with the support vector machines (SVM) classifier. The performance of the classification task on the test data was reported in terms of confusion matrix and overall classification accuracy.

1.1. Description of Methods:

A. Gram matrix feature extraction:

The Gram matrix representation of image texture is actually inspired from the deep learning. In order to construct a Gram matrix of an image, it was convolved with the C different convolutional kernels having a size of 3×3. Each convolutional kernel was generated randomly from uniform distribution having a range of -1 to 1 with weight summing to 0. After convolving the image with C kernels, we reduced the size of each convolved image using linear interpolation to 16×16 pixels. Each image obtained by convolutional operation was considered as an output channel (leading to C output channels), therefore, leading to an output image having a size of $16 \times 16 \times C$ (In case of $C = 10$, the output image will have a size of $16 \times 16 \times 10$). Next, we vectorized, each channel output, therefore representing each output channel by 256 elemental vector. Finally, we took the dot product of all these output channels leaving into a matrix having a size of $C \times C$. This matrix is called as Gram matrix and since it is symmetric in nature, therefore, we only retained the upper triangular part including diagonal elements as the feature vector to train the SVM classifier.

In this homework, we also used image center cropping rather than resizing for Gram matrix extraction. We performed the center cropping having a size of 96×96 pixels. After convolving these images with C convolutional operators, we had an output image having a size of $96 \times 96 \times C$. From here we proceed as indicated in the above paragraph to extract the Gram matrix features.

B. SVM Classifier:

In this homework, we used the sklearn implementation of the SVM classifier. The parameters used for training the SVM classifier are given in Table 1 and the exhaustive list of parameters can be found in the zip file accompanying this report.

1.2. Notes:

The classification task with Gram matrix was performed by varying the number of channels (C) from 1 to 100 for both image resizing and center cropping variants and the best C value was the one which achieved the maximum overall classification accuracy on the validation data. The table below provides a comparison of the different parameters of feature extraction and SVM training for two variants (image resizing and center cropping). The exhaustive parameter list including convolutional kernels, SVM coefficients, support vectors and gamma values yielding best classification accuracies for both variants are provided in the zip file accompanying this report.

Table 1: Parameters used for feature extraction and training the SVM classifier.

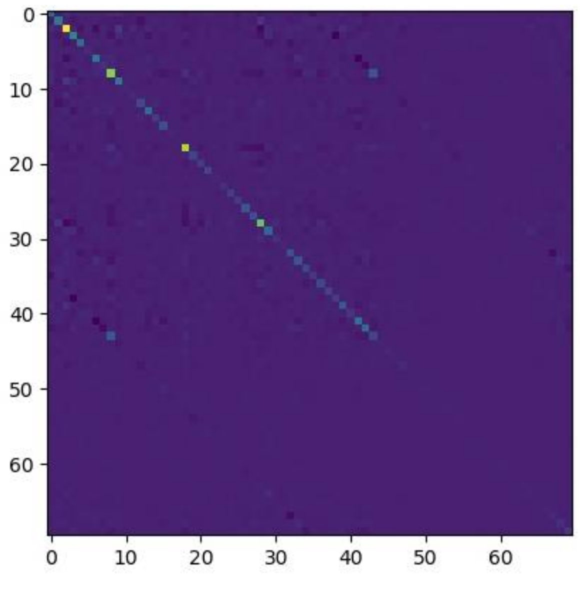
Parameters	**Image resized**	**Image center cropped**
C parameter for Gram Matrix	70	12
Feature vector length	2485 including the diagonal elements	78 including the diagonal elements
Image type for feature extraction	Grayscale	Grayscale

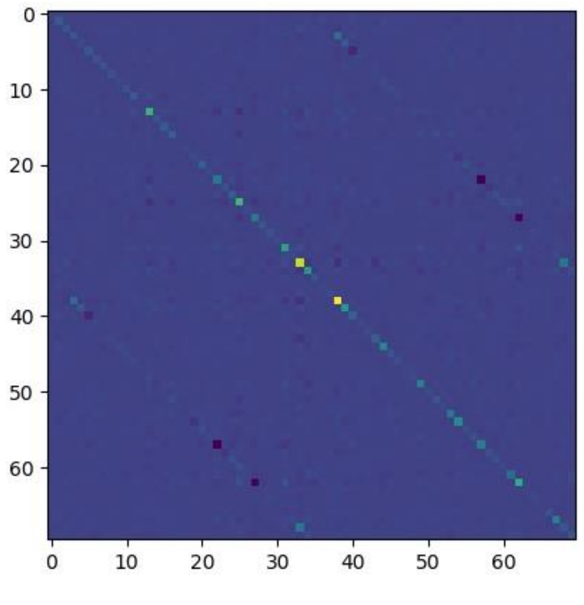
Image spatial size reduction	Original image resized to 256 × 256 initially and then resized to 16× 16 after convolution using linear interpolation.	Image center cropped to a size of 96 × 96 and no further spatial reduction was performed after convolution operation.
Kernel size	Kernels having spatial dimension of 3 × 3 generated randomly from uniform distribution having a range of -1 to 1 with weight summing to 0.	Kernels having spatial dimension of 3 × 3 generated randomly from uniform distribution having a range of -1 to 1 with weight summing to 0.
C parameter for SVM	1.0	1.0
Kernel parameter for SVM	RBF	RBF
Gamma parameter for SVM kernel	Used ‘scale’ method for gamma estimation (0.00018056232501245767)	Used ‘scale’ method for gamma estimation (4.40985564071284e-06)
Tolerance value for SVM	0.001	0.001
Decision function shape	One versus rest	One versus rest

Input samples for training SVM	756 (70% of 1080 total training samples)	756 (70% of 1080 total training samples)
Class Weight	[1, 1, 1, 1]	[1, 1, 1, 1]

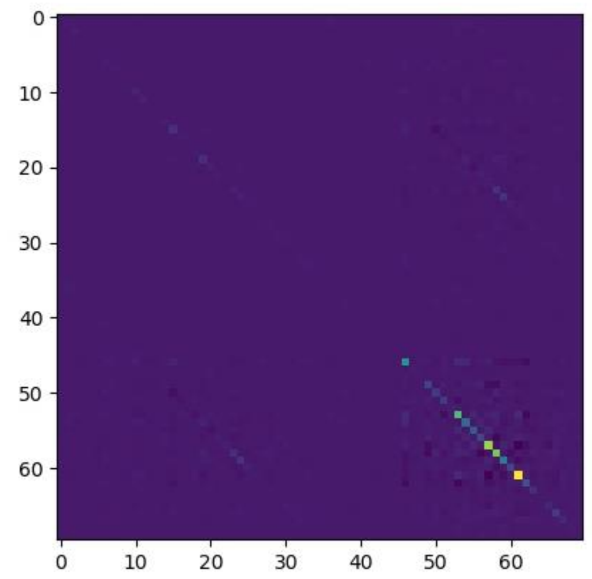
3. Results:

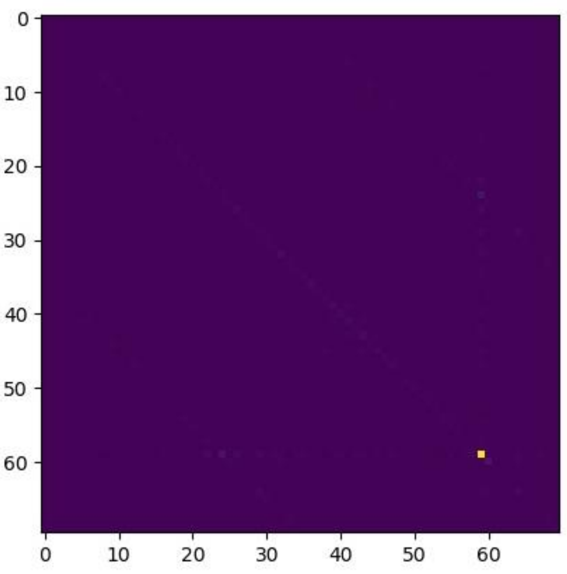
The Gram matrix for the random training images are provided in the figure below for image resizing (Figure 1) and image cropping (Figure 2), respectively. It can be seen that the image cropping generally contained more clustered and distinct textural information, while for the image resizing there is visually less distinction among instances of different classes. compared to the image resizing. This is especially true for the cloudy, shine and sunrise images. This could be due to the fact that in the image resizing some information can be lost due to the blurring effect caused by resizing.





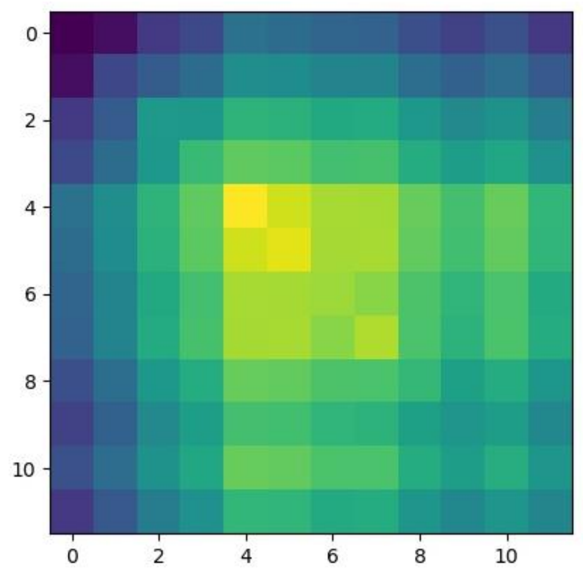
(a) **(b)**

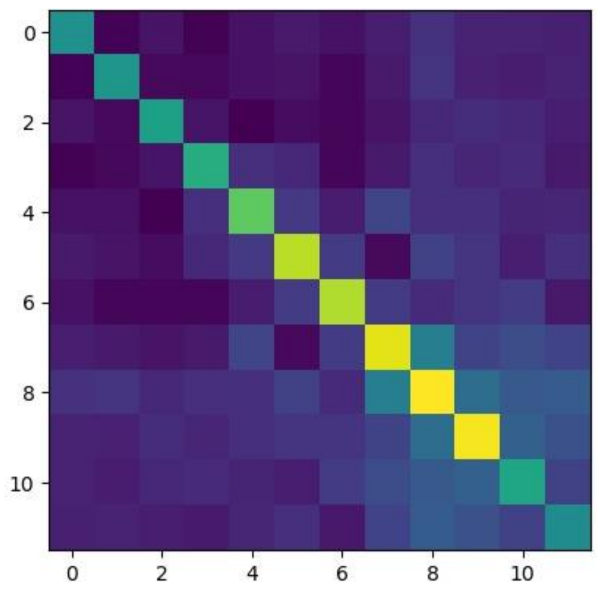




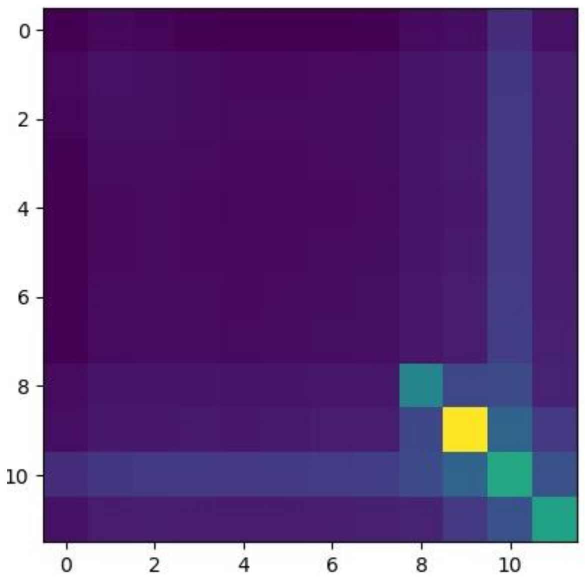
(c) **(d)**

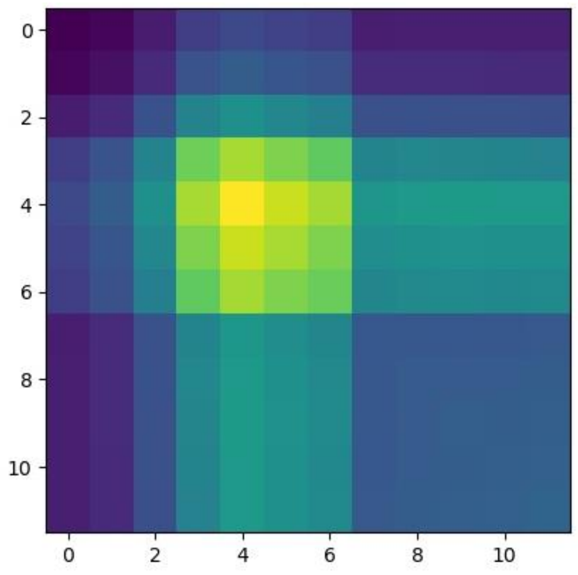
Fig 1: Gram matrices for the randomly picked samples from the training data for resized images at C = 70, (a) cloudy, (b) rain image (c) sunshine image and (d) sunrise image.





(a) **(b)**

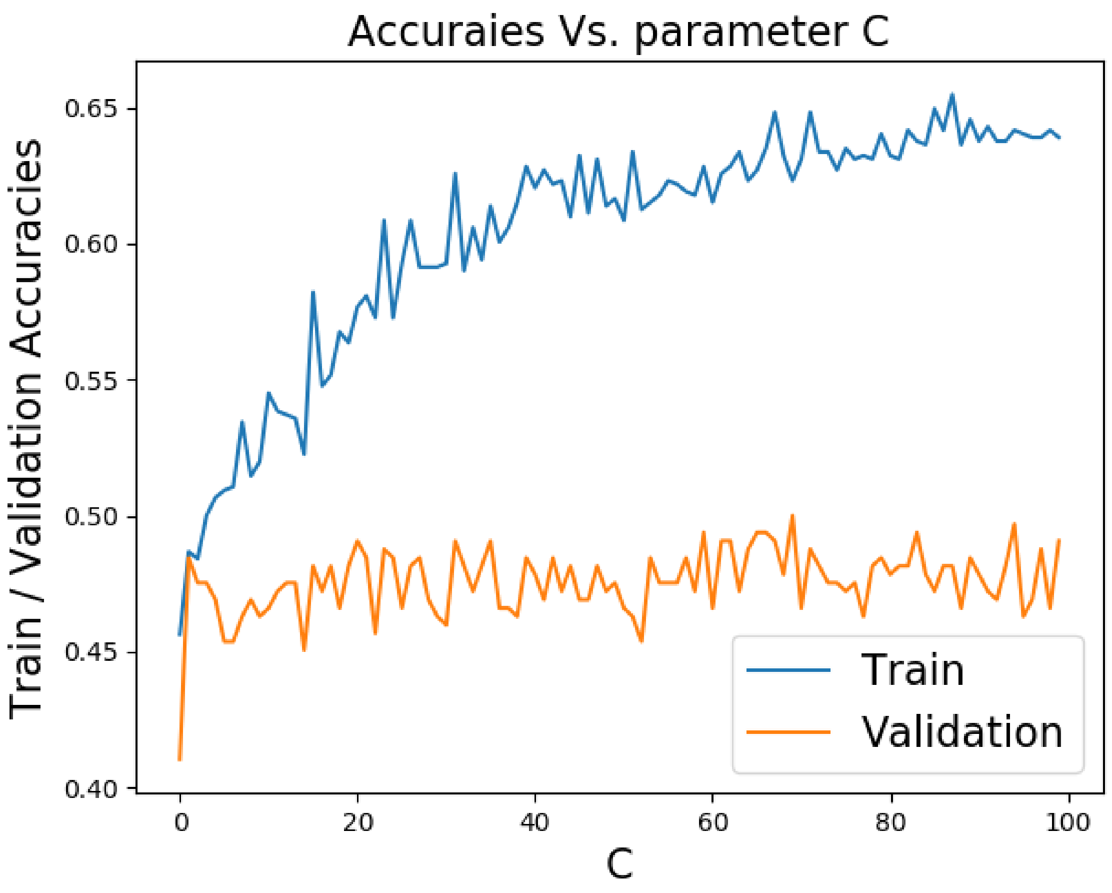




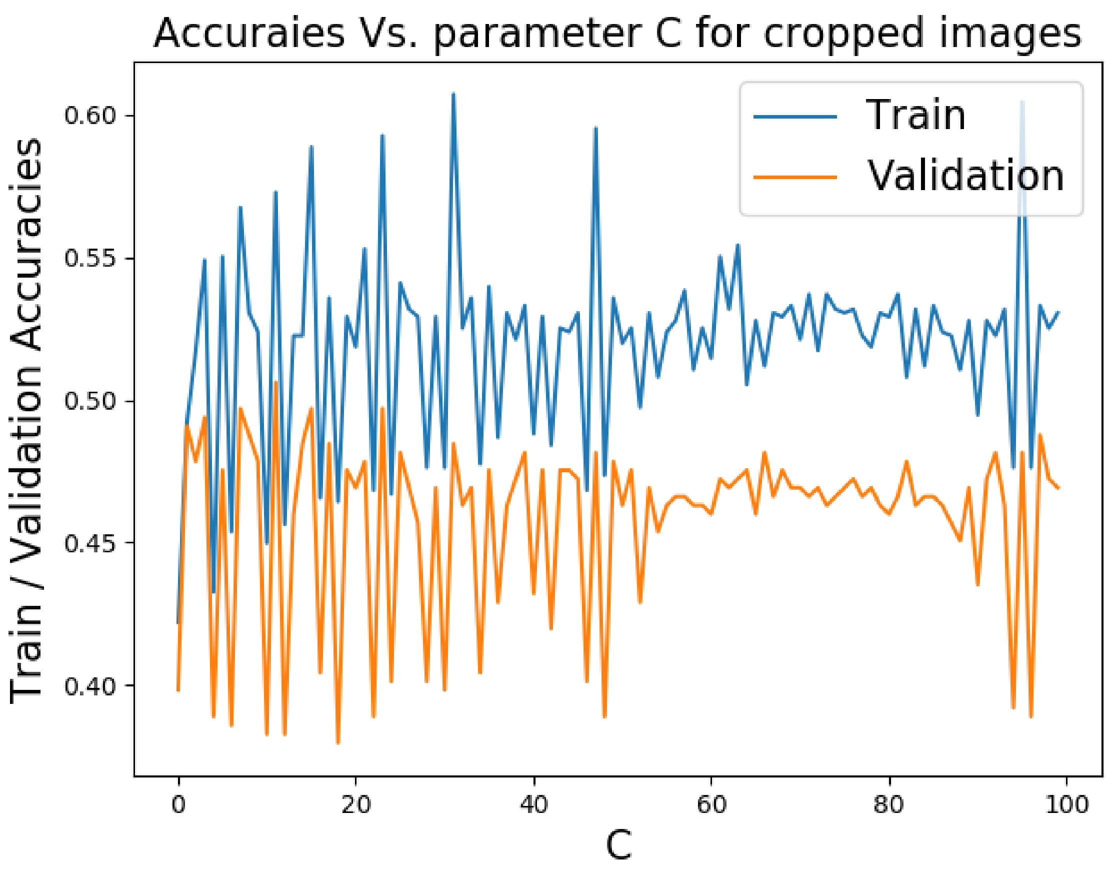
(c) **(d)**

Fig 2: Gram matrices for the randomly picked samples from the training data for center cropped images at C = 12, (a) cloudy, (b) rain image (c) sunshine image and (d) sunrise image.

Figure 3 is showing the impact of parameter value C against training and validation accuracies for both image cropping and resizing variants. It can be seen that for image resizing the training accuracy was consistently increasing with the value of C, while the validation accuracy was fluctuating between 45 and 50% throughout the training (Figure 3a). For the image center crop, we did not observe any specific trend for training and validation data (Figure 3b). The training accuracy fluctuated between 40 to 60%, while the validation accuracy was fluctuating between 35 to 50.61%. The absence of any trend in the training or validation data could be due to the fact that we lost a significant proportion of the image data during cropping as we picked relatively small size of 96 × 96 pixels. We picked to do center crop the images of 96 × 96 pixels due to the fact that the minim image height and width in the given data pool were around 110 pixels and 158 pixels, respectively. The best results for image resizing and cropping were achieved by C of 70 and 12, respectively and feature vector of size 2485 and 78 (including diagonal elements), respectively.



(a)



(b)

Fig 3: Training and test accuracies against the C parameter of the Gram Matrix (a) for resized images and (best validation accuracy of 50% at C = 70) (b) for center cropped images (best validation accuracy of 50.61% at C = 12).

The overall accuracies for the image resizing and image cropping were 40% and 50%, respectively on the test dataset as can be seen from the confusion matrices shown in Figure 4.

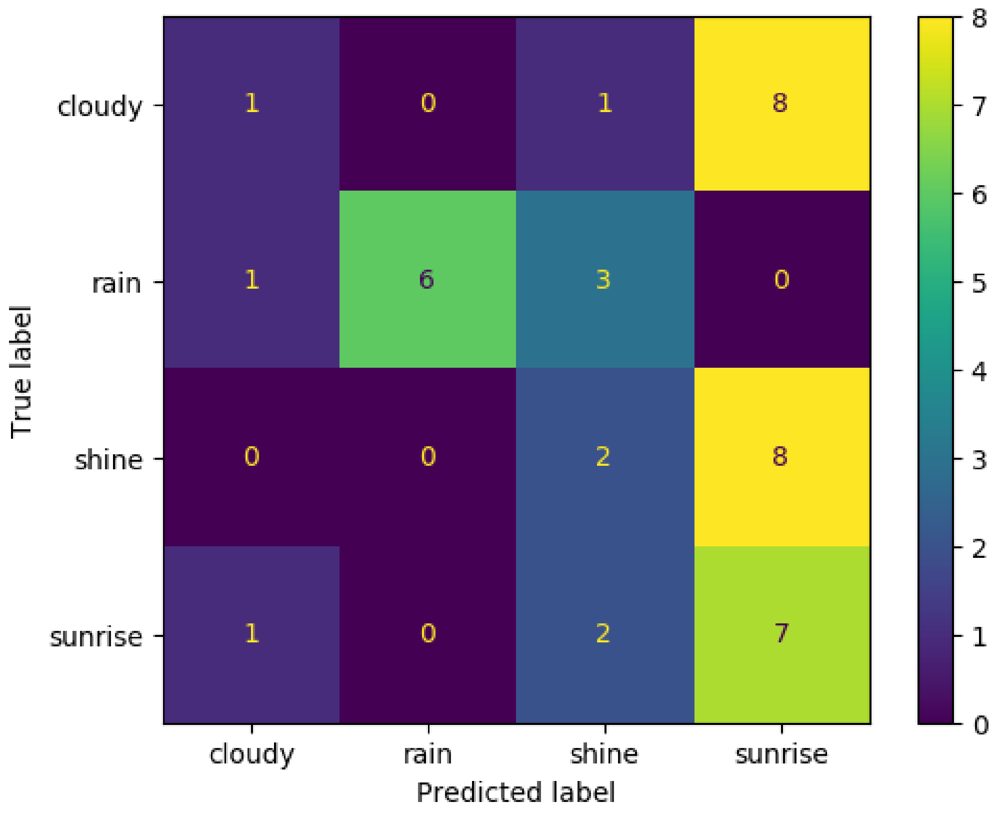
Image Resizing:

The highest misclassification was observed for the cloudy and shine classes. Looking at the confusion matrix, it can be seen that most of the cloudy instances were miscalssfied to the sunrise.

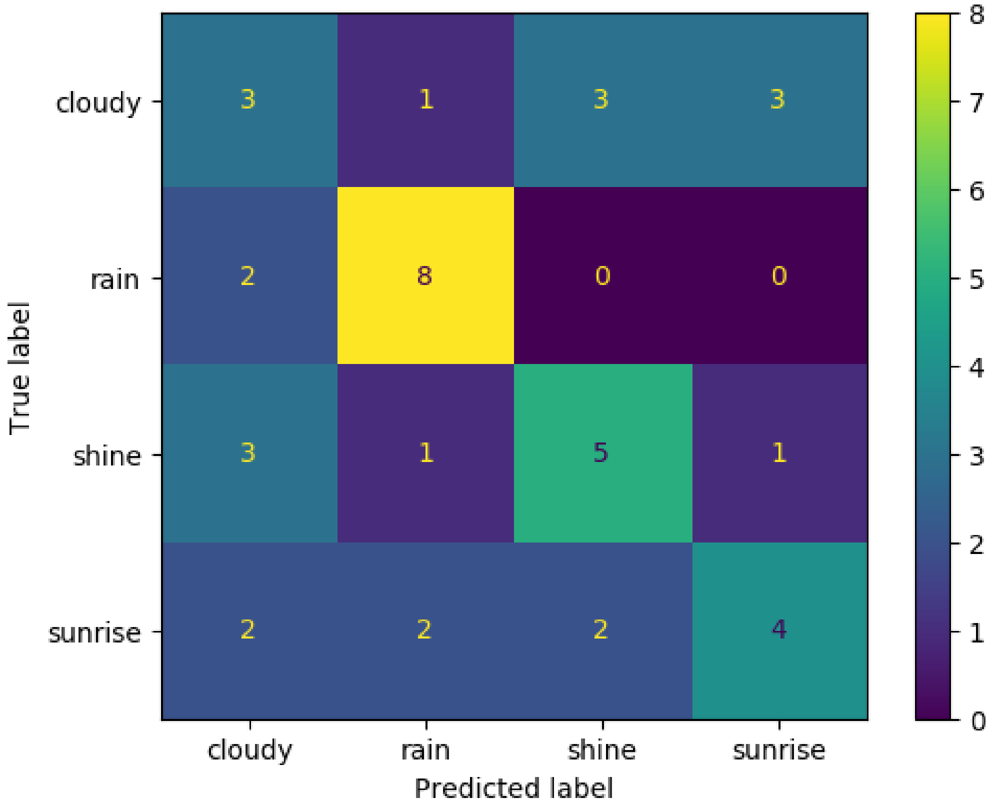
Subjectively visulaizng the test data as shown in Figure 5a indicated that the cloudy instances have some portion of the sunrise effect or viceversa, therefore leading to the miscalssification. For the rain highest miscalssification was towards the shine. Visualzing the rain image (Figure 5b) does not indicate any direct confoudning between these two classes, however, it might be the case that for some of the training rain images, there might be the sunshine present. Sunrise and shine might have some confounding as can be seen from Figure 5c and 5d, thereby might have some similar Gram matrix feature representation.

Image Cropping:

The highest misclassification was observed for the cloudy and sunrise and sunshine classes. Looking at the confusion matrix, it can be seen that most of the cloudy instances were miscalssfied to the sunrise and shine classes. Similarly, for the shine and rise the highest misccalsification is to either these classes or to the cloudy class (Figure 6). This could be due to the fact that using the center crop of only 96 × 96 pixels might have picked up the features that might not be very stable among these classes. Looking at Figure 6a, 6c and 6d, we can see that performing center crop yield to somewhat similar visualization, that might lead to poor feature representation. We believe, picking a relatively large crop size for center would have been helpful to improve the overall accuracy.


True label
cloudy
rain
shine
sunrise
Predicted label
1
0
1
8
1
6
3
0
0
0
2
8
1
0
2
7
0
1
2
3
4
5
6
7
8

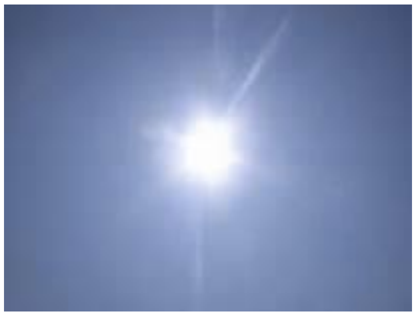
(a)



(b)

Fig 4: Confusion matrix of test dataset obtained with Gram matrix features and SVC classifier (a) for resized images with C = 70 (Total accuracy = 40%) and (b) for center cropped images with C = 12 (Total accuracy = 50%).

(a) (b)



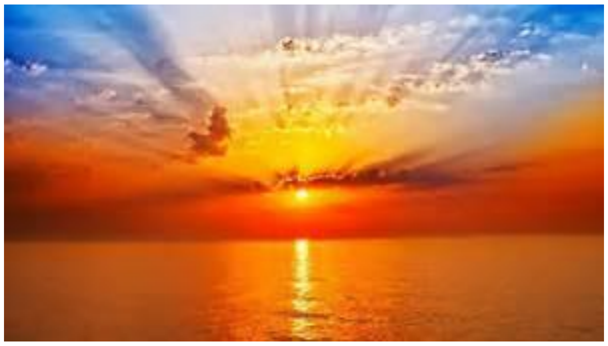
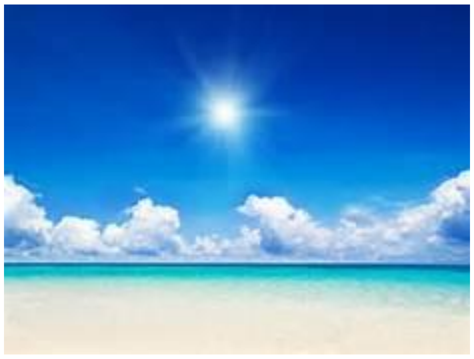
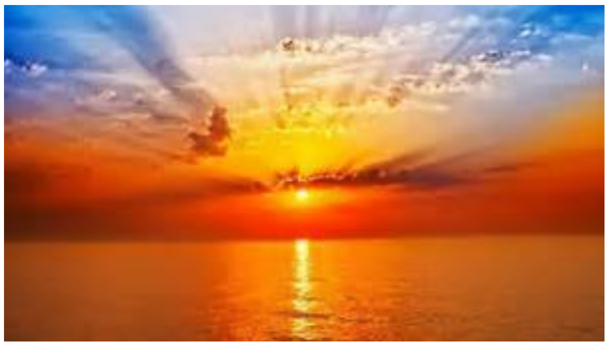


Fig 5: Some results of misclassified observations from the test dataset using Gram matrix features and SVC classifier for resized images with C = 70, (a) cloudy image misclassified as sunrise, (b) rain image misclassified as shine, (c) shine image misclassified as sunrise and (d) sunrise image misclassified as cloudy.

(a) (b)



(c)



(d)

Fig 6: Some results of misclassified observations from the test dataset using Gram matrix features and SVC classifier for center cropped images with C = 12, (a) cloudy image misclassified as shine, (b) rain image misclassified as cloudy, (c) shine image misclassified as sunrise and (d) sunrise image misclassified as cloudy.

The sunrise class generally have more observation, therefore, in future may be assign some weights to perform SVM classification.

4. References:

1) Gbeminiyi Ajayi. Multi-class Weather Dataset for Image Classification. Available at http://dx.doi.org/10.17632/4drtyfjtfy.1.

5. Source Code:

5.1.Function calls for classification:

```
'''----------Main Code for training and testing a classifier based on LBP and KNN-------'''
validation_accuracies = []
training_accuracies = []
best_valid = 0.0
for i in range(100):
    C = i+1
    kernels = GenerateRandom_kernel(-1.0,1.0,3,C)
    # Collect the Gram matrix features of entire training data
    cloudy_features=Class_Gram_features('cloudy',kernels,256,256,1,C,True)
    rain_features=Class_Gram_features('rain',kernels,256,256,1,C,True)
    shine_features=Class_Gram_features('shine',kernels,256,256,1,C,True)
    sunrise_features=Class_Gram_features('sunrise',kernels,256,256,1,C,True)
    #Get the corresponding Y_vector explaining class labels
    Y_C1=np.repeat("cloudy",len(cloudy_features))
    Y_C2=np.repeat("rain",len(rain_features))
    Y_C3=np.repeat("shine",len(shine_features))
    Y_C4=np.repeat("sunrise",len(sunrise_features))

    Train_X                                                                                    =
np.concatenate((cloudy_features,rain_features,shine_features,sunrise_features),axis=0)
    Train_Y = np.concatenate((Y_C1,Y_C2,Y_C3,Y_C4),axis=0)

    # Split the data in training and test
    X_train,  X_val,  y_train,  y_val  =  train_test_split(Train_X,  Train_Y,  test_size=0.3,
random_state=42)
    #Train the SVC model
    clf = SVC(C=1,kernel='rbf',gamma='scale')
    clf.fit(X_train,y_train)
    acc_train = clf.score(X_train,y_train)
    acc_val = clf.score(X_val,y_val)
    validation_accuracies.append(acc_val)
    training_accuracies.append(acc_train)
    predicted=clf.predict(X_val)
    print(f"[C-Vale:  {i+1}]  Training  Accuracy:  {acc_train:.3f}\t  Validation  Accuracy:
{acc_val:.3f}")

    if acc_val > best_valid:
        best_valid = acc_val
        print("New Best model found")
        # save the model to disk
        filename = 'Best_model.pkl'
        pickle.dump([clf,kernels,X_train,y_train,X_val,y_val], open(filename, 'wb'))
        #disp = metrics.plot_confusion_matrix(clf, X_val, y_val)

#Plot the training and test accuracies
plt.figure()
plt.plot(training_accuracies)
plt.plot(validation_accuracies)
plt.legend (['Train', 'Validation'], loc= 'best',fontsize = 16)

plt.xlabel('C',fontsize = 16)
plt.ylabel('Train / Validation Accuracies',fontsize = 16)
plt.title('Accuraies Vs. parameter C for cropped images',fontsize = 16)

print('Training is finished')

#Open the best saved model with kernels and training and validation data
```

```
print('Evaluating the model on test dataset')
filename = 'Best_model.pkl'
loaded_model = pickle.load(open(filename, 'rb'))
best_kernels = loaded_model[1]
C_best=best_kernels.shape[2]
model = loaded_model[0]

cloudy_features_test=Class_Gram_features('cloudy',best_kernels,256,256,16,C_best,False)
rain_features_test=Class_Gram_features('rain',best_kernels,256,256,16,C_best,False)
shine_features_test=Class_Gram_features('shine',best_kernels,256,256,16,C_best,False)
sunrise_features_test=Class_Gram_features('sunrise',best_kernels,256,256,16,C_best,False)
Y_C1_test=np.repeat("cloudy",len(cloudy_features_test))
Y_C2_test=np.repeat("rain",len(rain_features_test))
Y_C3_test=np.repeat("shine",len(shine_features_test))
Y_C4_test=np.repeat("sunrise",len(sunrise_features_test))

Test_X =
np.concatenate((cloudy_features_test,rain_features_test,shine_features_test,sunrise_features_test
),axis=0)
Test_Y = np.concatenate((Y_C1_test,Y_C2_test,Y_C3_test,Y_C4_test),axis=0)
acc_test = model.score(Test_X,Test_Y)
predicted=model.predict(Test_X)
#Make a confusion matrix
disp = metrics.plot_confusion_matrix(model, Test_X, Test_Y)
```

A. Different functions for extracting the Gram matrix features:

```
import numpy as np
import cv2
import glob
from scipy import signal as sig
import matplotlib.pyplot as plt
import re
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import pickle

def
Class_Gram_features(img_class,kernels,Width=256,Height=256,downsample=16,C=10,Trai
n_Flag=True):
   features_class = []
   if Train_Flag ==True:
      st = 'Training'
      print ('Extracting Gram matrix features from training data for class:
{}'.format(img_class))
      images = glob.glob('../imagesDatabaseHW8/training/{}'.format(img_class) + '/*.jpg')
   else:
        st = 'Testing'
        print ('Extracting Gram matrix features from Testing data for class:
{}'.format(img_class))
        images = glob.glob('../imagesDatabaseHW8/testing/{}'.format(img_class) + '_*.jpg')

    images = sort_alphanum(images)
    num_images = len(images)
    newH,newW = 96,96
    for i in range(num_images):
        image_color = cv2.imread(images[i])
        if image_color is not None:
            h,w = image_color.shape[0:2]
            #features_class.append(h,w)    #Check the size of each image
            #print('Currently running Image number:',images[i])
            #image_color=cv2.resize(image_color,(Width,Height),interpolation =
cv2.INTER_LINEAR)
            image_gray = cv2.cvtColor(image_color,cv2.COLOR_BGR2GRAY)
            #Do center cropping of the grayscale image--->Choose this or resizing
            image_gray = image_gray[int(h/2-newH/2):int(h/2+newH/2),int(w/2-
newW/2):int(w/2+newW/2)]
            feat_matrix = Get_FeatureMaps(image_gray,kernels,3,downsample,C)
            G = GramMatrix(feat_matrix)
            img_featur=G[np.triu_indices(C)]
            features_class.append((img_featur))
        else:
            print('Skipping Image number:',images[i])
            continue

    #Make a plot of last image in the class sequence for showing in report
    plt.imshow(G)
    plt.savefig('Gmatrix_Class_{}_'.format(img_class) + 'C_{}_'.format(int(C)) +
'{}_'.format(int(i)) + '{}'.format(st) + '.jpg')
    plt.close()

    return np.array(features_class)

def Get_FeatureMaps(image_gray,kernels,kernel_size=3,downsample=16,C=10):
    # Function for computing the convolving the C kernels with an input grayscale image
    feature_matrix =
np.zeros((int(image_gray.shape[0]/downsample),int(image_gray.shape[1]/downsample),C))
    for i in range(C):
        #Generate a random kernel
        #kernel= GenerateRandom_kernel(-1.0,1.0,(kernel_size,kernel_size))
        Ix = sig.convolve2d(image_gray,kernels[:,:,i],mode='same')
        # Possible downsample options
        #Ix = skimage.measure.block_reduce(image_gray, (downsample,downsample), np.mean)
        #Ix=cv2.resize(Ix,(downsample,downsample),interpolation = cv2.INTER_LINEAR)
        feature_matrix[:,:,i] = Ix
    return feature_matrix

def GenerateRandom_kernel(low = -1.0,high=1.0,size= 3,C=10):
    kernels = np.zeros((size,size,C))
    for i in range(C):
        kernel = np.random.uniform(low,high,size)
        #Normalize the kernel to obtain zero sum
        kernel = kernel - np.mean(kernel)
        kernels[:,:,i] = kernel
    return kernels

def GramMatrix(input_feature_Matrix):
    # Function for generating the gram matrix
    # Get the size f input matrix
    h,w,c = input_feature_Matrix.shape #h=height, w=width, c= num of feature maps
    # Reshape the input matrix by flattening its spatial dimension
    features = input_feature_Matrix.reshape((c, h * w))
    # Get the gram matrix by taking the dot product
    G = np.dot(features, features.T)
    #Normalize the final output by dividing the total num of elements in the matrix
    G = G/(h*w*c)
    return G

''' Functions for sorting the list of images in alphanumeric order '''
def atoi (string):
    try:
        return int(string)
    except ValueError:
        return string

def alphanum_key(string):
    return[atoi(c) for c in re.split('([0-9]+)',string)]

def sort_alphanum(listofImages):
    return (sorted(listofImages,key=alphanum_key))
```