1 Introduction

In this homework we implement a simple image classifier using Local Binary Pattern (LBP) feature extraction algorithm and a Nearest Neighbor (NN) classifier. Specifically,

1. We first implement the LBP feature extraction algorithm to obtain a histogram of feature vector for each image in the training set.
2. Next, using the Euclidean distance metric in the feature vector space, we find the k-nearest neighbors of each image in test set from the training set and assign the label to the test set image which appears maximum number of times.
3. In the last step, we construct a confusion matrix based on the classification results to calculate the overall accuracy of the image classification algorithm.

2 LBP Feature Extraction

The LBP feature extraction algorithm has been implemented in the following manner:

1. For each pixel in the image, we calculate the position of P number of points which lie along a circle of radius R using,

$$(\Delta u, \Delta v) = \left(Rcos\left(\frac{2\pi p}{P}\right), Rsin\left(\frac{2\pi p}{P}\right)\right) \tag{1}$$

   where $p = 0, 1, 2, ...P$ and Δu and Δv indicate the displacement of point p in the X and Y directions respectively.

2. The grayscale levels at the pixel locations of these P points is calculated using bilinear interpolation,

$$grayscale(x) = (1-\Delta u)(1-\Delta v)A + (1-\Delta u)\Delta v B + \Delta u(1-\Delta v)C + \Delta u \Delta v D \tag{2}$$

   where A, B, C, D represent the four corners of the rectangle around the point x. The bilinear interpolation is based on the assumption that the inter-pixel sampling interval is a unit distance along the X and Y direction.

3. Next, we threshold the grayscale levels at the P points with respect to the grayscale level at the center pixel. If the interpolated grayscale level at the neighbor point is greater than the center pixel grayscale level we set the point to 1 and if it is less than the center pixel we set it to 0. As a result we obtain a binary pattern around each pixel.
4. The local binary pattern is then made rotationally invariant by circularly rotating it until it acquires the smallest integer value. In this study we use Prof. Avi Kak's **BitVector** module to get the pattern with the least integer value.
5. In the following step we encode the circularly rotated local binary pattern around each pixel based on the number of runs of 1's and 0's by using the following methodology:
   (a) If there are exactly two runs, a run of 0's followed by a run of 1's then represent the pattern by the number of 1's in the run
   (b) If the pattern has all 0's represent the pattern by 0

(c) If the pattern has all 1's we encode the pattern with the value of P

(d) If the patter has more than two runs, the pattern is encoded with $P + 1$

6. In the final step we generate the LBP histogram for 10 bins corresponding to values from 0 to $P + 1$ for the binary pattern corresponding to each pixel.

The LBP histogram is calculated for each of the 20 images in the training set for the 5 class of images. This is the training step in the image classifier. The next step is the inferencing or prediction step which is done using a Nearest-Neighbor Classifier.

3 Nearest-Neighbor Classifier

Once the LBP histogram is calculated for each image in the training set, we determine the k-nearest training set image neighbors of each image in the test set using Euclidean distance in the feature space. Here we set $k = 5$. Once the top k nearest neighbors (lowest Euclidean distance) are obtained, we determine the label with maximum count among the k neighbors and assign that label to the test image.

4 Implementation in the Code

The image classifier has been implemented in **hw7_TejasPant.py**. Following are the major steps involved:

1. We first read the images using **cv2.imread** and then convert it into its grayscale form using **cv2.COLOR_BGR2GRAY**

2. For each class of image in the training set, the LBP histogram is calculated in the method **LBPHistogramForClass**. The LBP algorithm is called in this method using the method **LBP**. The LBP algorithm implemented in this study is based on Prof. Avi Kak's implementation.

3. Once the LBP histogram is generated for all the images in the training set the model is saved so that the training step does not have to be repeated during inferencing stage. The flag **ImageClassificationStep** is used to switch between training step (**Train**) and inferencing step (**Predict**)

4. For inferencing, we first load the saved model. Then using the method **LBPHistogramForTestSet** we calculated the LBP histogram for each image in the test set.

5. To label the images in the test set, the k-Nearest Neighbor algorithm is implemented in the method **NearestNeighborClassifier**. This method returns predicted label for an image in the test set

6. In the last step we calculate the confusion matrix by comparing the predicted label of the test image with its true label.

5 Observations

1. The images in class beach have much higher resolution than the other classes. An online tool is used to reduce the resolution of images in both the training and test set for the class beach otherwise the computational cost for training the image classifier on a workstation is too high.

2. In general, the image classifier based on the LBP feature extraction algorithm and the k-NN classifier gives reasonably good results with an accuracy of 72% on the test set.

3. The classifier works well for the images in class beach and car where the image textures tend to dominate the majority of the image.

4. For class building and mountain since there are many common textures between the two, the accuracy of prediction of these two classes is relatively less.

6 Results

In this study we define the accuracy of the image classifier as,

$$\text{Accuracy} = \frac{\text{CorrectClassifications}}{\text{TotalNumberofClassifications}}. \tag{3}$$

Based on Table 1, the accuracy of the implemented image classifier is **72 %**.

Table 1: Confusion Matrix for Test Set

	Beach	Building	Car	Mountain	Tree
Beach	4	0	0	1	0
Building	0	3	0	2	0
Car	0	1	4	0	0
Mountain	0	2	0	3	0
Tree	0	1	0	0	4

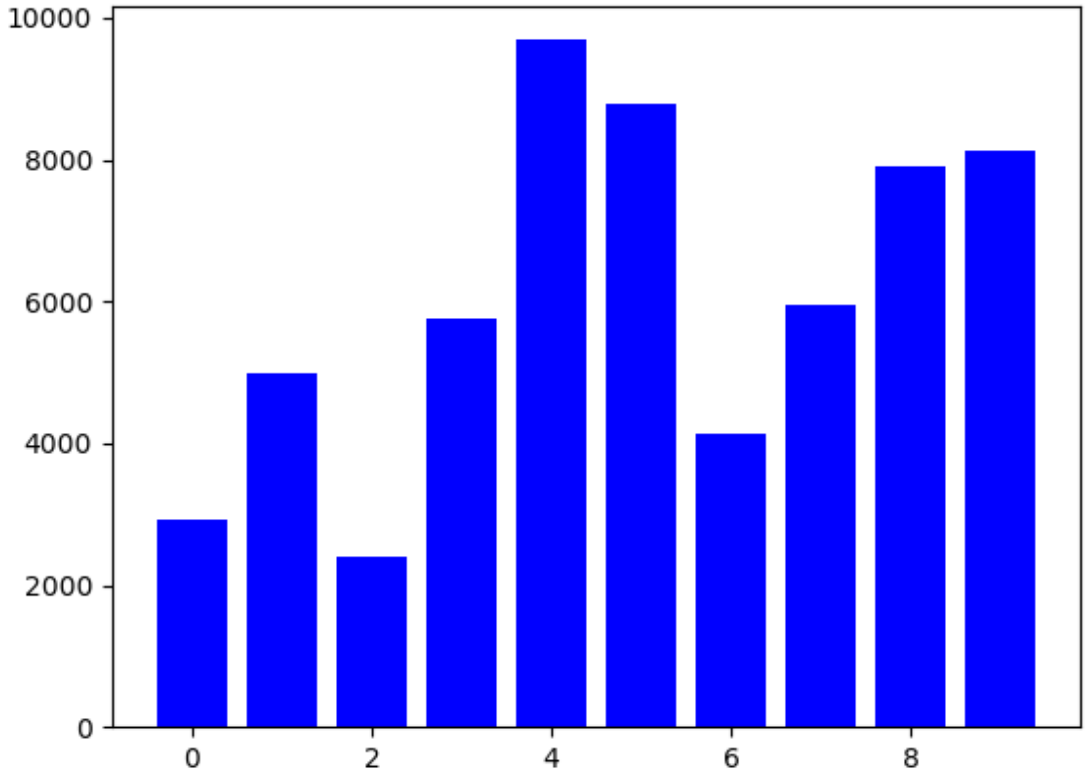

Figure 1: LBP Histogram for Sample 1 in Training Set of class **Beach**. Note: The resolution of the images for class Beach has reduced in the training and test data set to reduce computational cost.

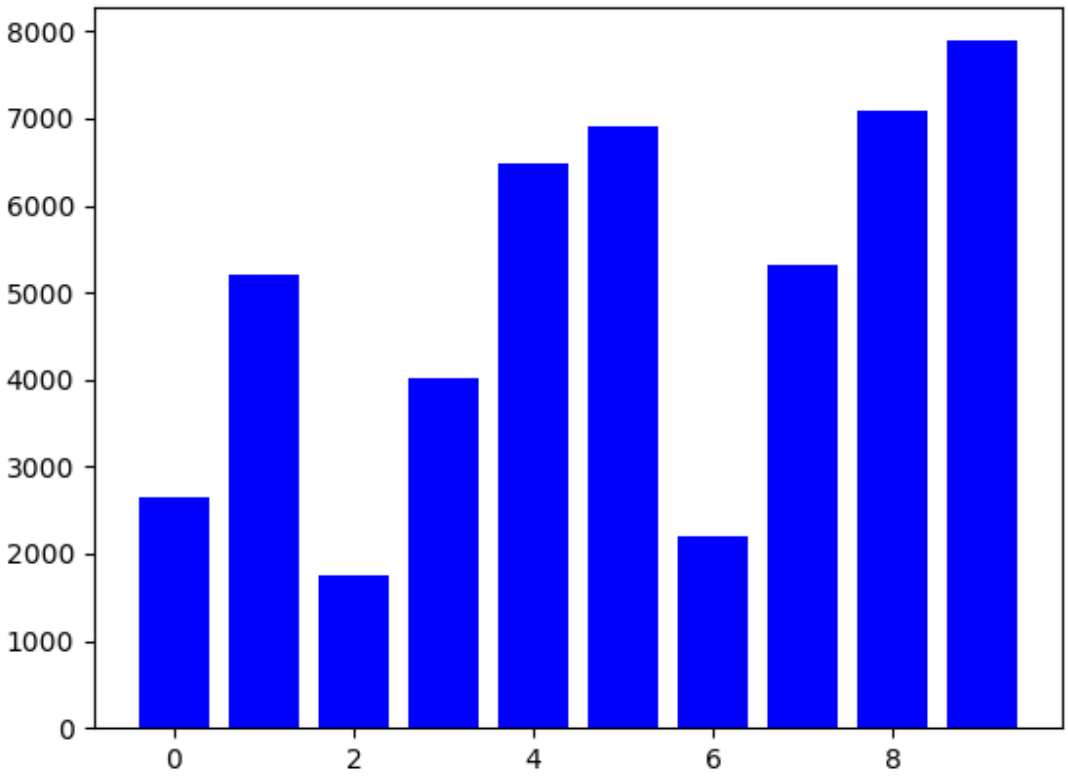

Figure 2: LBP Histogram for Sample 1 in Training Set of class **Building**.

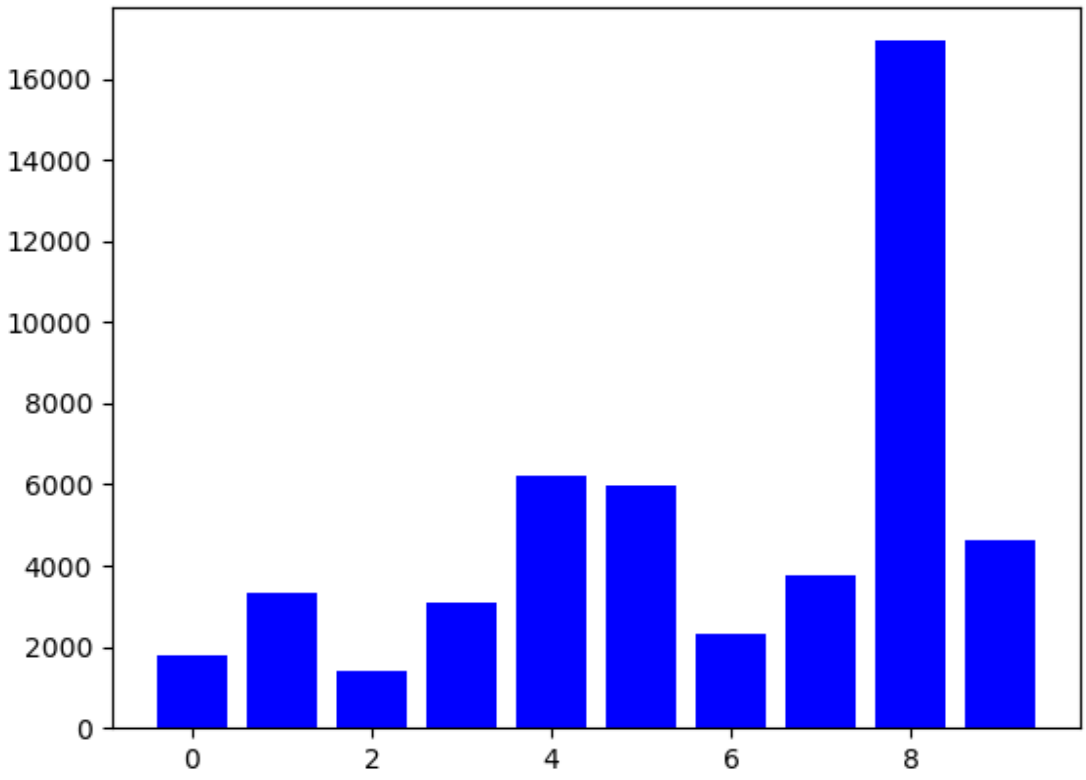

Figure 3: LBP Histogram for Sample 1 in Training Set of class **Car**.

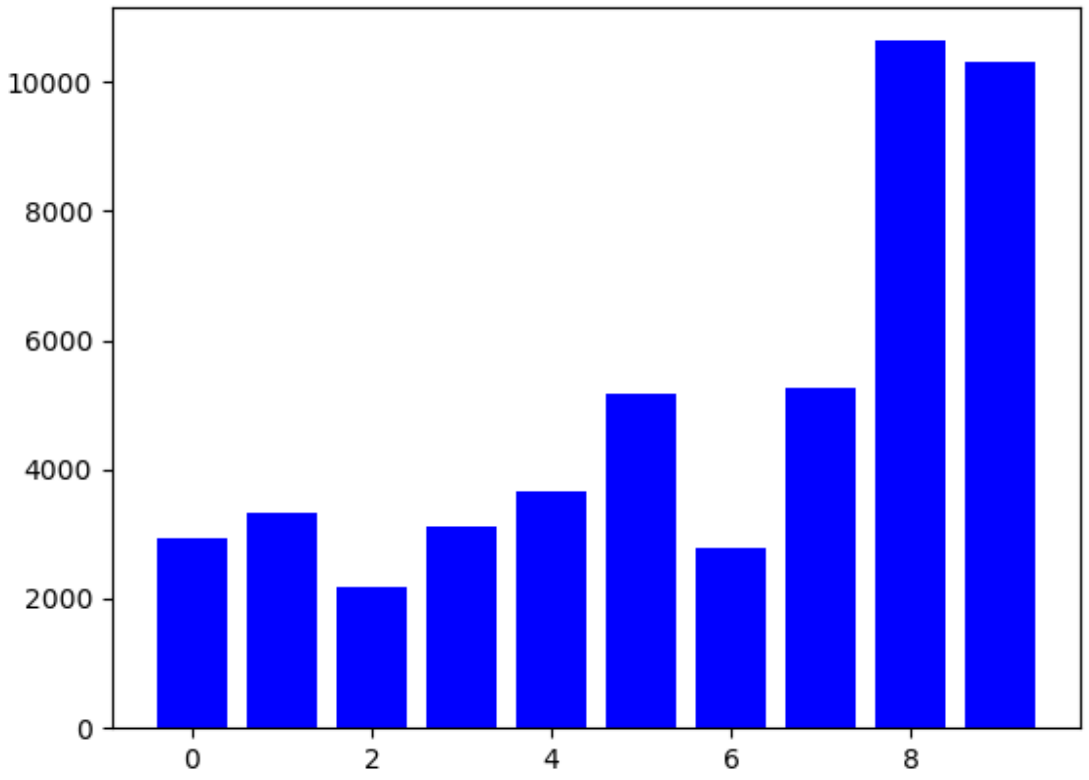

Figure 4: LBP Histogram for Sample 1 in Training Set of class **Mountain**.

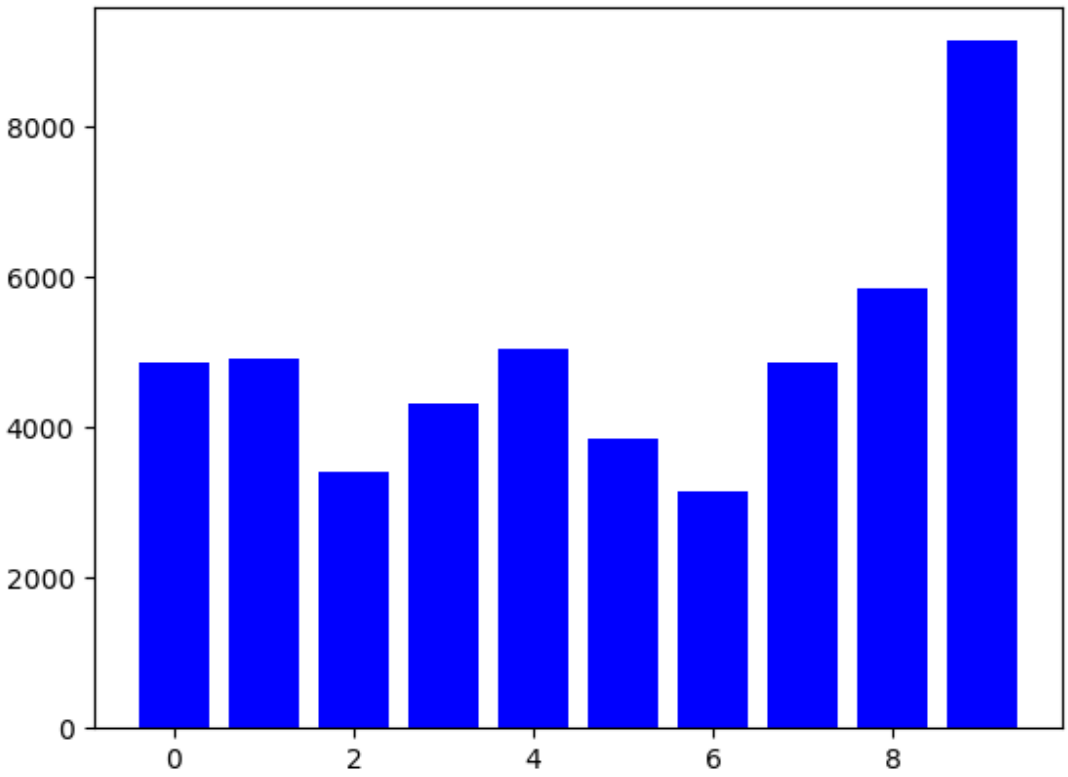

Figure 5: LBP Histogram for Sample 1 in Training Set of class **Tree**.