```
import numpy as np
import skimage
from skimage.feature import daisy
from skimage import data
from skimage.transform import resize
from skimage.transform import rescale
import matplotlib.pyplot as plt
import os
from skimage import io
from natsort import natsorted, ns


from zipfile import ZipFile
fileName = "HW1.zip"

with ZipFile(fileName, 'r') as zip:
  zip.extractall()


# Listing images
buttteryFlyList = os.listdir('/content/train/butterfly')
chairList = os.listdir('/content/train/chair')
laptopList = os.listdir('/content/train/laptop')

buttteryFlyList = natsorted(buttteryFlyList)
chairList = natsorted(chairList)
laptopList = natsorted(laptopList)

nofClasses = 3
classNames = ['butterfly', 'chair', 'laptop']


# Reading images
butterflyImList = []
chairImList = []
laptopImList = []

# Butterfly images
for i in buttteryFlyList:
  directory = '/content/train/butterfly/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = True)
    tempIm = resize(tempIm, (256, 256))
    butterflyImList.append(tempIm)

for i in chairList:
  directory = '/content/train/chair/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = True)
    tempIm = resize(tempIm, (256, 256))
    chairImList.append(tempIm)


for i in laptopList:
```

```
  directory = '/content/train/laptop/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = True)
    tempIm = resize(tempIm, (256, 256))
    laptopImList.append(tempIm)

# DAISY parameters
nofStep = 180
radiusVal = 58
nofRing = 2
nofHistogram = 16
nofOrientations = 8



# Using descriptor to extract features of butterfly images
index = 0;
butterflyFeatures = []
butteryflyLabels = []
for i in butterflyImList:
  descs = daisy(butterflyImList[index], step=nofStep, radius=radiusVal, rings=nofRing, his
  butterflyFeatures.append(descs);
  butteryflyLabels.append('butterfly')
  #print(butterflyFeatures[index].size)
  index = index + 1




# Using descriptor to extract features of chair images
index = 0;
chairFeatures = []
chairLabels = []
for i in chairImList:
  descs = daisy(chairImList[index], step=nofStep, radius=radiusVal, rings=nofRing, histogr
  chairFeatures.append(descs);
  chairLabels.append('chair')
  #print(chairFeatures[index].size)
  index = index + 1


# Using descriptor to extract features of laptop images
index = 0;
laptopFeatures = []
laptopLabels = []
for i in laptopImList:
  descs = daisy(laptopImList[index], step=nofStep, radius=radiusVal, rings=nofRing, histog
  laptopFeatures.append(descs);
  laptopLabels.append('laptop')
  #print(chairFeatures[index].size)
  index = index + 1


# Combining features and labels.
combinedFeatures = [*butterflyFeatures, *chairFeatures, *laptopFeatures]
combineLabels = [*butteryflyLabels, *chairLabels, *laptopLabels]
```

```
# Defining KNN function with minkowski distance.
def KNN(x_train, y_train, sample_test, k ):
  counter = 0
  minVal = 99999;
  for i in x_train:
    if minVal > x_train[counter].size:
      minVal = x_train[counter].size
    counter = counter + 1
  if minVal > sample_test.size:
    minVal = sample_test.size
  distances = []
  indexImg = 0;
  q = 3
  for i in range(len(x_train)- 1):
    totalDist = 0
    indexFeature = 0;
    for j in range(minVal - 1):
      tempDist = 0
      tempDist = abs(x_train[indexImg][0][0][indexFeature] - sample_test[0][0][indexFeatur
      totalDist = totalDist + tempDist
      indexFeature = indexFeature + 1
    distances.append(pow(totalDist, 1/q))
    indexImg = indexImg + 1;

  sortedDistances = sorted(distances)
  index = []
  classifiedLabels = []
  counter = 0
  for i in range(k):
    index.append(distances.index(sortedDistances[counter]))
    classifiedLabels.append(y_train[distances.index(sortedDistances[counter])])
    counter = counter + 1
  # print(index)
  # print(classifiedLabels)

  counter = 0
  classifierVote = np.full((1, nofClasses), 0)
  for i in range(k):
    classifierVote[0][classNames.index(classifiedLabels[counter])] = classifierVote[0][cla
    counter = counter + 1
  # print(classifierVote)
  # result = sorted(classifierVote, reverse=False)
  maxVal = np.max(classifierVote)
  result = np.where(classifierVote == maxVal)
  print(classNames[int(result[1])])


# Reading test image
testIm = io.imread('/content/test/image_0001.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
```

```
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)


# Reading test image
testIm = io.imread('/content/test/image_0007.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
laptop
```

```
# Reading test image
testIm = io.imread('/content/test/image_0008.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
chair
```

```
# Reading test image
testIm = io.imread('/content/test/image_0010.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
butterfly
```

```
# Reading test image
testIm = io.imread('/content/test/image_0013.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
chair
```

```
# Reading test image
testIm = io.imread('/content/test/image_0014.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
laptop
```

```
# Reading test image
testIm = io.imread('/content/test/image_0031.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
butterfly
```

```
# Reading test image
testIm = io.imread('/content/test/image_0073.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
laptop
```

```
# Reading test image
testIm = io.imread('/content/test/image_0074.jpg', as_gray = True)
testIm = resize(testIm, (256, 256))
# Extracting Feactures
testFeatures = daisy(testIm, step=nofStep, radius=radiusVal, rings=nofRing, histograms=nof
# Testing the Function
KNN(combinedFeatures, combineLabels, testFeatures, 7)
```

```
laptop
```