```
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
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
import os
from skimage import io
from natsort import natsorted, ns
import scipy as sp
import heapq
import math


from zipfile import ZipFile
fileName = "HW3.zip"

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


# Listing images
cannonList = os.listdir('/content/CaltechTinySplit/train/cannon')
cellphoneList = os.listdir('/content/CaltechTinySplit/train/cellphone')

cannonList = natsorted(cannonList)
cellphoneList = natsorted(cellphoneList)

nofClasses = 2
# class0 is cannon, class1 is cellphone
classNames = [0, 1]


# Reading images
cannonImList = []
```

```
cellphoneImList = []

# Classes
cannonClassList = []
cellphoneClassList = []

# Cannon images
for i in cannonList:
  directory = '/content/CaltechTinySplit/train/cannon/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = False)
    tempIm = resize(tempIm, (64, 64))
    tempIm = tempIm.ravel()
    cannonImList.append(tempIm)
    cannonClassList.append(0)

# Cellphone images
for i in cellphoneList:
  directory = '/content/CaltechTinySplit/train/cellphone/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = False)
    tempIm = resize(tempIm, (64, 64))
    tempIm = tempIm.ravel()
    cellphoneImList.append(tempIm)
    cellphoneClassList.append(1)



def tanhActivationFunc(x):
  return ((np.exp(x) - np.exp(-1*x)) / (np.exp(x) + np.exp(-1*x)))
def derivativeTanhActFunc(x):
  val = tanhActivationFunc(x)
  return (1 - pow(val, 2))
def calculateDeltaWeight(rho, t, y, dervActVal, x):
  return rho*(t - y)*dervActVal*x
```

```
def standardize(value):
    mean = np.mean(value, axis=0)
    std = np.std(value, axis=0)+0.000001
    X_train = (value - mean) / std
    return X_train

# Concatenating lists
dataSetList = [*cannonImList, *cellphoneImList]
dataClassList = [*cannonClassList, *cellphoneClassList]
```

## Perceptron Training Function

```
def trainPerceptron(inputs, t, weights, rho, iterNo):
  appendedDataSetList = inputs
  dataClassList = t
  for i in range(iterNo):
    sumVector = np.zeros(len(appendedDataSetList))
    # Feed forward
    for j in range(len(appendedDataSetList)):
      if(appendedDataSetList[j].shape[0] == weights.shape[0]):
        arr = np.dot(weights, appendedDataSetList[j])
        sumVector[j] = arr
    stdVal = standardize(sumVector)
    for j in range(len(appendedDataSetList)):
      if(appendedDataSetList[j].shape[0] == weights.shape[0]):
        y = tanhActivationFunc(stdVal[j])
        target = dataClassList[j]
        # Feed backward
        deltaWeight = calculateDeltaWeight(rho, target, y, derivativeTanhActFunc(y), appendedDataSetList[j])
        weights += deltaWeight;
  return weights
```

```
appendedDataSetList = []
appendingVal = 0
for i in range(len(dataSetList)):
  x = dataSetList[i]
  x = np.append(x, [appendingVal])
  appendedDataSetList.append(x)
appendedDataSetList, dataClassList = shuffle(appendedDataSetList, dataClassList) # don't you forget the parameters that enters
                                                                                  # the shuffle function must be function parameter as
```

```
weights = np.random.random((len(appendedDataSetList[0])))
weights = trainPerceptron(appendedDataSetList, dataClassList, weights, 0.001, 1000)
np.save('weights.npy', weights) # save
```

# Testing Phase

```
weights = np.load('weights.npy') # load
```

# Perceptron Testing Function

```
def testPerceptron(sample_test, weights):
  sumVector = np.zeros(len(sample_test))
  for i in range(len(sample_test)):
    sum = np.dot(sample_test[i], weights)
    sumVector[i] = sum
  sumVector = standardize(sumVector)
  y = tanhActivationFunc(sumVector)
  return y
```

```
# Listing images
cannonTestList = os.listdir('/content/CaltechTinySplit/test/cannon')
```

```
cellphoneTestList = os.listdir('/content/CaltechTinySplit/test/cellphone')

cannonTestList = natsorted(cannonTestList)
cellphoneTestList = natsorted(cellphoneTestList)

# Reading Test images
cannonTestImList = []
cellphoneTestImList = []

# Classes
cannonTestClassList = []
cellphoneTestClassList = []

# Accordion test images
for i in cannonTestList:
  directory = '/content/CaltechTinySplit/test/cannon/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = False)
    tempIm = resize(tempIm, (64, 64))
    tempIm = tempIm.ravel()
    cannonTestImList.append(tempIm)
    cannonTestClassList.append(0)

# Airplane test images
for i in cellphoneTestList:
  directory = '/content/CaltechTinySplit/test/cellphone/' + i
  if i != '.ipynb_checkpoints' and i != 'Thumbs.db':
    tempIm = io.imread(directory, as_gray = False)
    tempIm = resize(tempIm, (64, 64))
    tempIm = tempIm.ravel()
    cellphoneTestImList.append(tempIm)
    cellphoneTestClassList.append(1)


appendedCannonTestList = []
for i in range(len(cannonTestImList)):
  x = cannonTestImList[i]
  x = np.append(x, [1]) # bias value
```

```
  appendedCannonTestList.append(x)
```

+ Kod + Metin

```
appendedCellphoneTestList = []
for i in range(len(cellphoneTestList)):
  x = cellphoneTestImList[i]
  x = np.append(x, [1]) # bias value
  appendedCellphoneTestList.append(x)


# Testing canon list
cannonTestOutputs = testPerceptron(appendedCannonTestList, weights)
print(cannonTestOutputs)
```

```
[-0.14932068 -0.47304869  0.94423864 -0.84604322  0.12998825]
```

```
# Testing cellphone list
cellphoneTestOutputs = testPerceptron(appendedCellphoneTestList, weights)
print(cellphoneTestOutputs)
```

```
[-0.96202855  0.20952595  0.3068603   0.23605937  0.64495183  0.85436299
 -0.68400091]
```

✓ 0 sn. tamamlanma zamanı: 17:37

reCAPTCHA hizmetiyle bağlantı kurulamadı. Lütfen internet bağlantınızı kontrol edin ve reCAPTCHA testi almak için sayfayı yeniden yükleyin.