projeto 3

July 15, 2024

Stochastic Processes - class 1/2024 - University of Brasília Computational work 3 - classification

Gabriel Tambara Rabelo - 241106461

O presente projeto visa organizar um conjunto de dados a ser utilizado para classificação, atribuindo a estes, dados estatísticos, bem como organizar em histogramas os seus valores de cores, conforme realizado nos trabalhos computacionais 1 e 2. Os resultados seguem o modelo apresentado a seguir:

expectancy,mode,median,variance,skewness,kurtosis,entropy,class 53.891967984934084,16.0,16.0,2366.49105036432
80.08158192090394,16.0,48.0,5663.520651358167,289773.10140834015,61596894.36590457,2.3554137854502386,0
80.37126177024481,16.0,48.0,6078.010526279875,372971.9596642113,79378789.52637905,2.392316701226323,0

h[0],h[1],h[2],h[3],h[4],h[5],h[6],h[7],class 58419,11150,9625,13219,13300,450,20,17,0
52196,7232,6733,7406,9320,9624,12177,1512,0

A seguir, segue o código equivalente.

[ ]:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from math import log2
import os
import csv

img_size = 300
classes = ['Alzheimer', 'COVID', 'Brazilian_seeds', 'Brazilian_leaves',␣
 ↪'skin_cancer']
class_labels = {0: 'Alzheimer', 1: 'COVID', 2: 'Brazilian_seeds', 3:␣
 ↪'Brazilian_leaves', 4: 'skin_cancer'}
data_dir = './image_dataset/'
images = []

# Control of when to stop showing images and progressing in the processing
def waitKey():
    cv.waitKey(0)
    cv.destroyAllWindows()

# Show images
def printAll(list, subtitle):
    for i in range(len(list)):
        cv.imshow(subtitle + str(i), list[i])

# Save images in the correct folder
def saveAll(list, subtitle, class_label):
    for i in range(len(list)):
        cv.imwrite(f"./images/{class_label}/{subtitle}_{i}.jpg", list[i])

def makeHist(image, bins):
    hist, bin_edges = np.histogram(image.flatten(), bins=bins, range=(0, 256))
    return hist, bin_edges

def normalizeHist(hist):
    total_pixels = np.sum(hist)
    normalized_hist = hist / total_pixels
    return normalized_hist

def bin_centers(bin_borders):
    bin_center_list = (bin_borders[:-1] + bin_borders[1:]) / 2
    bin_center_list[-1] = 255 # Ensure the last bin center is 255
    return bin_center_list

def expectancy(hist, bin_centers):
    return np.sum(hist * bin_centers)

def median(hist, bin_centers):
    cumulative_freq = np.cumsum(hist)
    return bin_centers[np.searchsorted(cumulative_freq, 0.5)]

def mode(hist, bin_centers):
    return bin_centers[np.argmax(hist)]

def moment(hist, bin_centers, order, expectancy):
    return np.sum(((bin_centers - expectancy) ** order) * hist)

def entropy(hist):
    epsilon = 1e-10
    return -np.sum(hist * np.log2(hist + epsilon))

def calculate_statistics(image, bins):
    hist, bin_edges = makeHist(image, bins)
    normalized_hist = normalizeHist(hist)
    centered_bins = bin_centers(bin_edges)

    var_exp = expectancy(normalized_hist, centered_bins)
    var_median = median(normalized_hist, centered_bins)
    var_mode = mode(normalized_hist, centered_bins)
    var_variance = moment(normalized_hist, centered_bins, 2, var_exp)
    var_skewness = moment(normalized_hist, centered_bins, 3, var_exp)
    var_kurtosis = moment(normalized_hist, centered_bins, 4, var_exp)
    var_entropy = entropy(normalized_hist)

    return hist, var_exp, var_mode, var_median, var_variance, var_skewness, var_kurtosis, var_entropy

def process_images_from_folder(folder_path):
    image_files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))]
    images = []

    for image_file in image_files:
        img = cv.imread(os.path.join(folder_path, image_file), cv.IMREAD_GRAYSCALE)
        ratio = img_size / img.shape[1]
        img_resized = cv.resize(img, (img_size, int(img.shape[0] * ratio)), cv.INTER_AREA)
        images.append(img_resized)

    return images

def save_statistics_to_csv(images, class_label, case1_csv_filename, case2_csv_filename):
    with open(case1_csv_filename, mode='a', newline='') as case1_file, open(case2_csv_filename, mode='a', newline='') as case2_file:
        case1_writer = csv.writer(case1_file)
        case2_writer = csv.writer(case2_file)

        for image in images:
            hist, var_exp, var_mode, var_median, var_variance, var_skewness, var_kurtosis, var_entropy = calculate_statistics(image, bins=8)

            case_1_data = list(hist) + [class_label]
            case_2_data = [var_exp, var_mode, var_median, var_variance, var_skewness, var_kurtosis, var_entropy, class_label]

            case1_writer.writerow(case_1_data)
            case2_writer.writerow(case_2_data)

# Initialize CSV files with headers
case1_csv_filename = 'case1_statistics.csv'
case2_csv_filename = 'case2_statistics.csv'

with open(case1_csv_filename, mode='w', newline='') as case1_file, open(case2_csv_filename, mode='w', newline='') as case2_file:
    case1_writer = csv.writer(case1_file)
    case2_writer = csv.writer(case2_file)

    # Write headers
    case1_writer.writerow([f'h[{i}]' for i in range(8)] + ['class'])
    case2_writer.writerow(['expectancy', 'mode', 'median', 'variance',␣
  ↪'skewness', 'kurtosis', 'entropy', 'class'])

# Process each class folder and save statistics
for class_id, class_name in enumerate(classes):
    folder_path = os.path.join(data_dir, class_name)
    images = process_images_from_folder(folder_path)
    save_statistics_to_csv(images, class_id, case1_csv_filename,␣
  ↪case2_csv_filename)
```