Classificação de Imagens no Conjunto de Dados MPEG7 Modificado

1 Introdução

Este trabalho tem como objetivo a classificação de imagens no conjunto de dados MPEG7 modificado utilizando técnicas de aprendizado supervisionado.

2 Importação de Bibliotecas e Configuração Inicial

O primeiro passo foi importar as bibliotecas necessárias para o processamento de imagens e construção do modelo. As bibliotecas utilizadas são:

- `os` - para manipulação de arquivos e diretórios;
- `numpy` - para operações numéricas;
- `skimage` - para processamento de imagens;
- `sklearn` - para construção e avaliação do modelo;
- `matplotlib` e `seaborn` - para visualização de dados.

O código para importação das bibliotecas é o seguinte:

```
import os
import numpy as np
from skimage import util, transform,
    filters, color, measure, morphology
from sklearn import model_selection,
    neighbors, metrics, preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import cv2
```

3 Carregamento e Pré-processamento dos Dados

As imagens foram carregadas a partir de um diretório no Google Drive, e as imagens foram redimensionadas para 1/4 do tamanho original para reduzir o custo computacional.

```
from google.colab import drive
drive.mount('/content/drive')

ds_path = '/content/drive/My Drive/
    mpeg7_mod'
classes_list = os.listdir(ds_path)
image_list = []
label_list = []
filename_list_ = []

for classe in classes_list:
    filename_list = os.listdir(os.path.join
        (ds_path, classe))
    for filename in filename_list:
        img_temp = plt.imread(os.path.join(
            ds_path, classe, filename))
        img_temp = transform.resize(
            img_temp, (img_temp.shape
            [0]//4, img_temp.shape[1]//4),
            anti_aliasing=True)
        image_list.append(img_temp)
        label_list.append(classe)
        filename_list_.append(filename)
```

4 Seleção das Imagens e Rótulos

Foi realizada a seleção das primeiras 6 imagens de cada classe, garantindo um conjunto de dados balanceado para o treinamento do modelo.

```
seg_list_temp = []
filename_list_temp = []

for i, class_name in enumerate(classes_list
    ):
    seg_list_temp += [seg_list[j] for j in
        np.where(label_list==class_name)
        [0][:6]]
    filename_list_temp += [filename_list_[j
        ] for j in np.where(label_list==
        class_name)[0][:6]]
```

5 Cálculo das Características das Imagens

Foi realizada a segmentação binária das imagens e, em seguida, extraídas as características de cada objeto segmentado, como área, maior eixo, menor eixo, solidez e excentricidade.

```
features = ['area', 'major_axis', '
    minor_axis', 'solidity', 'eccentricity'
    ]
seg_list = []
list_label = []
```

```
feature_mat = []

for i, (image, label) in enumerate(zip(image_list, label_list)):
    img_float = util.img_as_float(image)
    if len(img_float.shape) == 2:
        img_gray = img_float
    else:
        img_gray = color.rgb2gray(img_float)
    img_seg = morphology.remove_small_objects(img_gray > 0, 1000)
    seg_list.append(img_seg)
    im_lbl = measure.label(img_seg)
    props = measure.regionprops(im_lbl)

    for prop in props:
        area = prop.area
        major_axis = prop.major_axis_length
        minor_axis = prop.minor_axis_length
        solidity = prop.solidity
        eccentricity = prop.eccentricity
        feature_list = [area, major_axis, minor_axis, solidity, eccentricity]
    feature_mat.append(feature_list)
```

6 Transformação e Normalização das Características

Foi realizada a normalização das características para garantir que todas possuam a mesma escala, utilizando a média e o desvio padrão do conjunto de treinamento.

```
X_train_mean = X_train.mean(0)
X_train_std = X_train.std(0)

X_train_norm = (X_train - X_train_mean) / X_train_std
X_test_norm = (X_test - X_train_mean) / X_train_std
```

7 Construção do Modelo de Classificação

Utilizamos dois modelos de aprendizado supervisionado: o K-Nearest Neighbors (K-NN) e o Perceptron. Para ambos, as imagens foram classificadas com base nas características extraídas.

7.1 Classificador K-NN

Foi treinado um classificador K-NN com $k = 3$.

```
clf = neighbors.KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train_norm, y_train)
pred = clf.predict(X_test_norm)
```

7.2 Classificador Perceptron

Em seguida, um Perceptron foi treinado utilizando os dados normalizados.

```
clf = Perceptron(max_iter=1000, eta0=0.01, random_state=42)
clf.fit(X_train_norm, y_train)
pred = clf.predict(X_test_norm)
```

8 Avaliação do Modelo

A avaliação do modelo foi feita utilizando as métricas de precisão, recall e f1-score, além de uma matriz de confusão para verificar o desempenho do classificador.

```
print(classification_report(y_test, pred))
```

A matriz de confusão foi gerada para visualização dos acertos e erros do classificador:

```
conf_matrix = confusion_matrix(y_test, pred)
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt='g')
```