CURSO: <BÁSICO EM MACHINE LEARNING>

Tema: Resumo sobre pré-processamento de imagens, segmentação de imagens e detecção/classificação de imagens.

Aluna: Talita Dias da Silva

Itens:

1. Pré-processamento de imagens;
2. Segmentação de imagens;
3. Detecção/classificação de imagens;

PRÉ-PROCESSAMENTO DE IMAGENS

INTRODUÇÃO

O pré-processamento de imagens é uma etapa crucial na visão computacional, que visa melhorar a qualidade das imagens e prepará-las para análise subsequente. Esta etapa pode incluir operações como ajuste de brilho e contraste, remoção de ruído, normalização, entre outras, para garantir que os dados fornecidos aos modelos de aprendizado de máquina ou redes neurais sejam otimizados e livres de distorções.

EXEMPLOS DE BIBLIOTECAS/FRAMEWORKS

Existem várias bibliotecas e frameworks disponíveis para o pré-processamento de imagens. Alguns dos mais populares incluem:

1. **OpenCV:** Uma biblioteca open-source amplamente utilizada em visão computacional, que oferece uma vasta gama de funções para manipulação e análise de imagens.
2. **Pillow:** Um fork da biblioteca Python Imaging Library (PIL), fácil de usar, que oferece funções básicas de processamento de imagens.
3. **Scikit-Image:** Parte do ecossistema Scipy, esta biblioteca fornece algoritmos para processamento de imagens em Python.
4. **TensorFlow e Keras:** Embora focadas em aprendizado profundo, essas bibliotecas oferecem funções para pré-processamento de imagens, como redimensionamento, normalização e aumento de dados.

EXEMPLOS DE APLICAÇÕES

A seguir, apresento alguns exemplos simples de código em Python que demonstram o pré-processamento de imagens.

Exemplo 1: Ajuste de Brilho e Contraste usando OpenCV

No exemplo abaixo, usei o OpenCV para ajustar o brilho e o contraste de uma imagem:

```
[10] import cv2
     import numpy as np

[11] # Carregar a imagem
     imagem = cv2.imread('imagem1.jpg')

     # Ajustar brilho e contraste
     brilho = 50
     contraste = 30
     nova_imagem = cv2.convertScaleAbs(imagem, alpha=1 + contraste / 100, beta=brilho)

[13] # Salvar a imagem
     cv2.imwrite('imagem_ajustada.jpg', nova_imagem)

     True
```

Neste exemplo, a função `convertScaleAbs` é utilizada para ajustar o brilho e o contraste da imagem. O parâmetro `alpha` controla o contraste, enquanto `beta` ajusta o brilho. A imagem resultante é salva em um novo arquivo.

Exemplo 2: Redimensionamento e Rotação usando Pillow

O código abaixo mostra como redimensionar e rotacionar uma imagem usando a biblioteca Pillow:

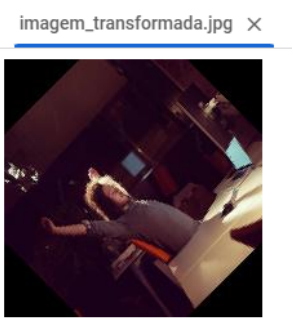
```
[5] from PIL import Image

[6] # Carregar a imagem
    imagem = Image.open('imagem1.jpg')

[7] # Redimensionar a imagem
    imagem_redimensionada = imagem.resize((200, 200))

[8] # Rotacionar a imagem
    imagem_rotacionada = imagem_redimensionada.rotate(45)

    # Salvar a imagem
    imagem_rotacionada.save('imagem_transformada.jpg')
```



Este exemplo demonstra o uso da biblioteca Pillow para redimensionar uma imagem para 200x200 pixels e, em seguida, rotacioná-la em 45 graus. A imagem transformada é então salva em um novo arquivo.

SEGMENTAÇÃO DE IMAGENS

INTRODUÇÃO

A segmentação de imagens é uma técnica fundamental na visão computacional, utilizada para dividir uma imagem em partes ou segmentos distintos, com base em características como cor, textura e intensidade. Cada segmento representa uma região significativa da imagem, como um objeto ou uma área de interesse, facilitando a análise e interpretação subsequente.

EXEMPLOS DE BIBLIOTECAS/FRAMEWORKS

Diversas bibliotecas e frameworks suportam a segmentação de imagens, oferecendo ferramentas para diferentes abordagens. Alguns dos mais notáveis incluem:

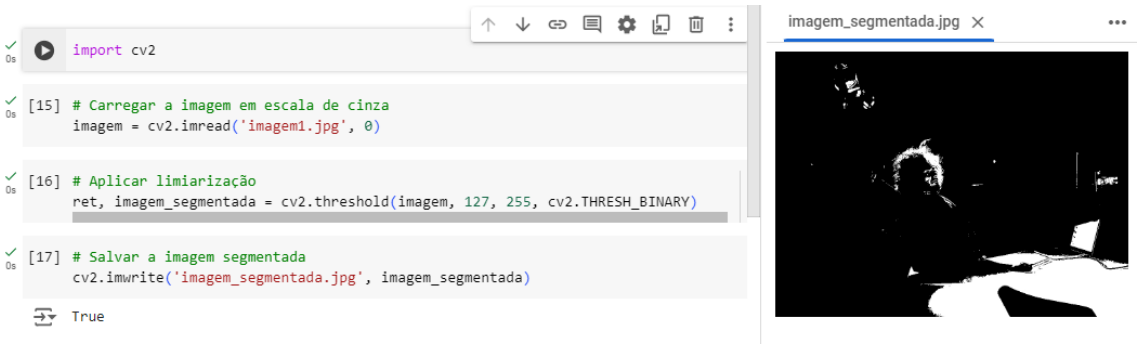
1. **OpenCV:** Oferece métodos para segmentação baseada em bordas, regiões, e limiarização, amplamente utilizada em aplicações de visão computacional.
2. **Scikit-Image:** Fornece algoritmos avançados para segmentação, incluindo Watershed, felzenszwalb, e segmentação por limiarização.
3. **TensorFlow e Keras:** Usadas principalmente para segmentação semântica, essas bibliotecas suportam redes neurais convolucionais (CNNs) para tarefas de segmentação complexas.

EXEMPLOS DE APLICAÇÕES

A seguir, alguns exemplos simples de código em Python que ilustram a segmentação de imagens.

Exemplo 1: Segmentação por Limiarização usando OpenCV

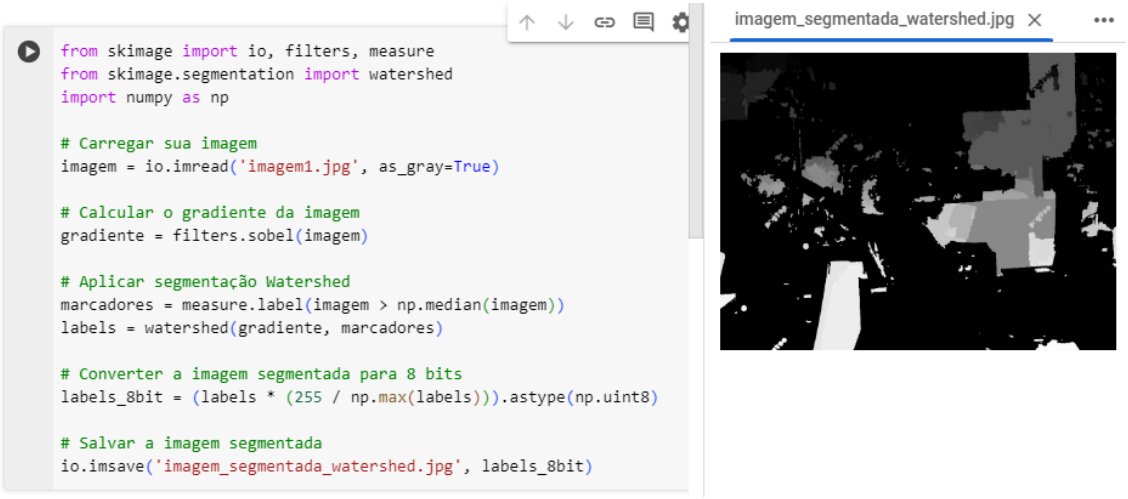
No exemplo abaixo, usamos a função de limiarização do OpenCV para segmentar uma imagem:



Aqui, a função `threshold` converte a imagem em uma representação binária, onde os pixels acima de um certo limiar são definidos como brancos, e os abaixo como pretos. Esta técnica é útil para segmentar objetos de fundo homogêneo.

Exemplo 2: Segmentação usando Watershed com Scikit-Image

O código abaixo ilustra o uso do algoritmo Watershed da biblioteca Scikit-Image para segmentação de uma imagem:



Neste exemplo, a função `watershed` é usada para segmentar a imagem baseada nos marcadores obtidos através do cálculo do gradiente. Este método é eficaz para separar objetos que estão próximos ou se sobrepõem em uma imagem.

DETECÇÃO/CLASSIFICAÇÃO DE IMAGENS

INTRODUÇÃO

A detecção e classificação de imagens são tarefas essenciais na visão computacional, utilizadas para identificar e categorizar objetos ou características específicas dentro de uma imagem. A detecção envolve localizar objetos de interesse, enquanto a classificação atribui rótulos a esses objetos com base em modelos treinados. Essas técnicas são amplamente aplicadas em diversas áreas, como reconhecimento facial, detecção de veículos e classificação de espécies de plantas.

EXEMPLOS DE BIBLIOTECAS/FRAMEWORKS

Diversas bibliotecas e frameworks são amplamente utilizadas para detecção e classificação de imagens, incluindo:

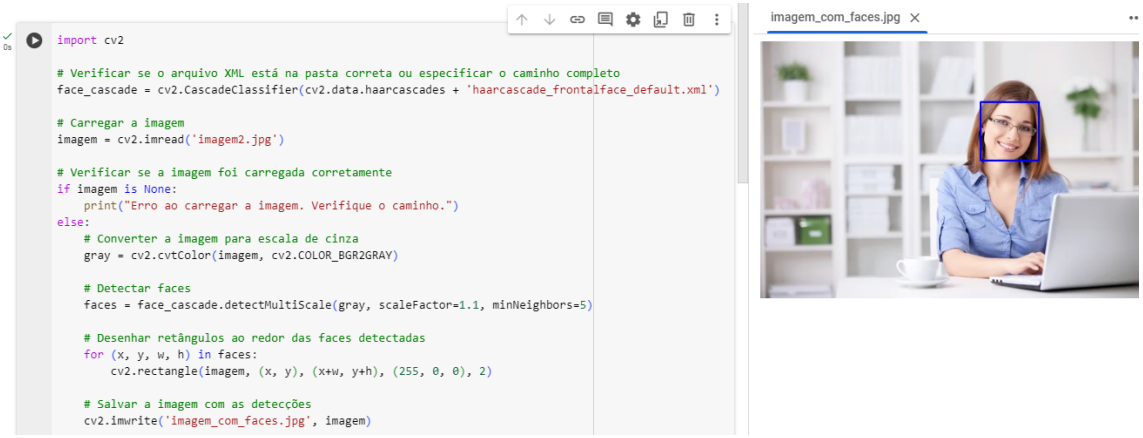
1. **OpenCV:** Fornece ferramentas para detecção de objetos, como o detector de faces Haar Cascade e o método de detecção de bordas Canny.
2. **TensorFlow e Keras:** Suportam a criação e treino de redes neurais profundas para tarefas de classificação e detecção de objetos, incluindo modelos pré-treinados como MobileNet e YOLO.
3. **PyTorch:** Outro framework popular que oferece suporte a redes neurais convolucionais (CNNs) e modelos pré-treinados para detecção e classificação de imagens.
4. **Detectron2:** Um framework desenvolvido pelo Facebook AI Research, especializado em detecção de objetos, segmentação e classificação.

EXEMPLOS DE APLICAÇÕES

A seguir, alguns exemplos simples de código em Python que demonstram a detecção e classificação de imagens.

Exemplo 1: Detecção de Faces usando OpenCV

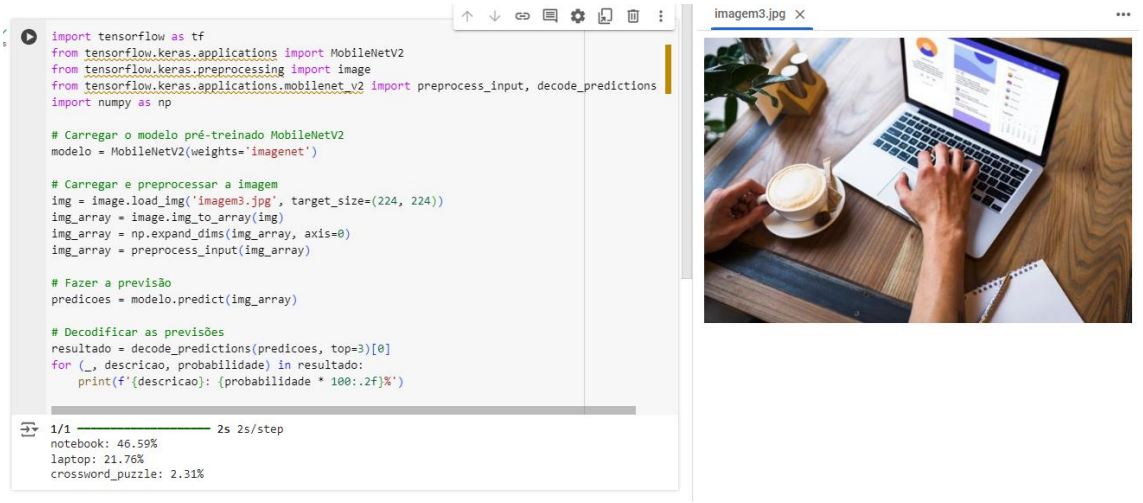
No exemplo abaixo, utilizamos o OpenCV para detectar rostos em uma imagem:



Neste exemplo, o classificador Haar Cascade é utilizado para detectar faces em uma imagem. O método `detectMultiScale` retorna as coordenadas dos retângulos que delimitam as faces detectadas, e `cv2.rectangle` desenha esses retângulos na imagem original.

Exemplo 2: Classificação de Imagens usando TensorFlow/Keras

O exemplo abaixo mostra como usar um modelo pré-treinado do TensorFlow para classificar imagens:



Neste exemplo, um modelo MobileNetV2 pré-treinado é usado para classificar uma imagem. A imagem é carregada e redimensionada para 224x224 pixels, preprocessada, e depois passada pelo modelo. As previsões retornam as três classes mais prováveis, junto com suas respectivas probabilidades.