Bootcamp Atlântico- Machine Learning

Cauan Gomes dos Santos Barbosa

Atividade - 05

1. Pré-processamento de Imagens

Introdução:
O pré-processamento de imagens é uma etapa crucial na análise de imagens, pois garante que os dados estejam em um formato apropriado para serem usados por algoritmos de Machine Learning. O objetivo principal é melhorar a qualidade da imagem, corrigir distorções e destacar características importantes. As técnicas incluem redimensionamento, normalização, remoção de ruídos, ajuste de contraste e conversão de cores.

Técnicas Comuns:

- Normalização: Converte os valores dos pixels para um intervalo padrão (geralmente entre 0 e 1).
- Redimensionamento: Ajusta o tamanho da imagem para garantir que todas as entradas tenham as mesmas dimensões.
- Filtro de ruído: Remove interferências que podem prejudicar o desempenho do modelo.

Exemplo de Normalização com OpenCV:

```python
import cv2
import numpy as np

# Carregar imagem
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Normalizar valores dos pixels
img_normalized = img / 255.0

print(f'Dimensões da imagem: {img_normalized.shape}')

```

O que faz?

Esse código ajusta os valores dos pixels da imagem para ficarem entre 0 e 1. Isso ajuda bastante porque muitos algoritmos de Machine Learning funcionam melhor quando os dados estão nesse formato padronizado

Carregar a imagem: A função cv2.imread lê a imagem em preto e branco.
Normalizar os pixels: Dividir por 255 faz com que os valores da imagem, que vão de 0 a 255, fiquem entre 0 e 1. Isso facilita a vida do modelo.
Mostrar a dimensão: Mostra o tamanho da imagem carregada (altura x largura)

2. Segmentação de Imagens

Introdução:
A segmentação de imagens é o processo de dividir uma imagem em partes ou regiões distintas, cada uma correspondendo a um objeto ou área de interesse. Essa etapa é essencial para tarefas como reconhecimento de objetos e análise de cena. Existem diferentes técnicas de segmentação, incluindo thresholding, segmentação baseada em bordas e segmentação semântica.

Tipos de Segmentação:

- Thresholding: Divide a imagem com base em valores de intensidade.
- Segmentação por contornos: Detecta limites entre diferentes objetos na imagem.
- Segmentação semântica: Classifica cada pixel de uma imagem em categorias específicas.

Exemplo de Segmentação por Thresholding:

```python
import cv2
import matplotlib.pyplot as plt

# Carregar a imagem em escala de cinza
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Aplicar thresholding
_, segmented = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# Exibir a imagem segmentada
plt.imshow(segmented, cmap='gray')
plt.title('Segmentação Thresholding')
plt.show()
```

O que faz?

Aqui, a ideia é separar a imagem em duas cores (preto e branco) usando um valor limite. Tudo que estiver acima do valor definido (127, nesse caso) vira branco, e o que estiver abaixo vira preto.

- Carregar a imagem: Novamente, lê a imagem em preto e branco.
- Aplicar o threshold: O cv2.threshold transforma a imagem em duas cores — qualquer pixel mais claro que 127 vira branco (255), e o resto fica preto (0).
- Exibir a imagem: O plt.imshow mostra o resultado final da segmentação.

3. Detecção e Classificação de Imagens

Introdução:
A detecção e a classificação são tarefas fundamentais em visão computacional.

- Detecção de imagens: Envolve a localização de objetos específicos dentro de uma imagem. Ferramentas como YOLO e Faster R-CNN são amplamente utilizadas para essa tarefa.
- Classificação de imagens: Atribui uma etiqueta ou categoria a uma imagem inteira, utilizando redes neurais convolucionais (CNNs) para identificar padrões.

Exemplo de Classificação usando CNN com Keras:

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

# Definir o modelo
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')  # 10 classes de saída
])

# Compilar o modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

print('Modelo criado com sucesso!')
```

O que faz?

Esse código cria uma rede neural convolucional (CNN), um modelo muito usado para classificar imagens. Aqui, o modelo identifica características da imagem em várias etapas, como bordas e formas, até conseguir decidir a qual classe ela pertence.

- **Camada de Convolução (Conv2D): Detecta padrões na imagem, como bordas ou texturas. Aqui, criamos 32 filtros (pequenos detectores de características).**
- **MaxPooling (MaxPooling2D): Reduz o tamanho da imagem, mantendo as informações mais importantes, o que ajuda a simplificar o processamento.**
- **Flatten: Achata a imagem para transformá-la em uma lista de dados, que o modelo pode entender.**
- **Camadas densas (Dense): Processam as informações extraídas. A última camada usa a função softmax, que atribui a probabilidade de a imagem pertencer a cada uma das 10 classes.**
- **Compilação: Configura o modelo para ser treinado, usando o otimizador adam e uma função de perda adequada para classificação.**

Conclusão

Ao longo deste trabalho, explorei as etapas fundamentais para o uso de Machine Learning na visão computacional: o pré-processamento, a segmentação e a detecção/classificação de imagens. Durante essa jornada, ficou claro como cada uma dessas fases é indispensável para o desenvolvimento de soluções precisas e eficientes, desde a preparação dos dados até a identificação de padrões e objetos em imagens.

O pré-processamento destacou-se como o alicerce para garantir que as imagens estejam em um formato adequado, eliminando ruídos e padronizando as informações. Já a segmentação mostrou-se essencial para dividir as imagens em partes significativas, permitindo uma análise mais detalhada e organizada. Por fim, a detecção e classificação de imagens demonstraram o impacto das redes neurais convolucionais, que transformaram a maneira como os sistemas reconhecem e categorizam elementos visuais.

Foi um processo enriquecedor pesquisar e aplicar essas técnicas, utilizando ferramentas robustas como OpenCV e TensorFlow, além de implementar exemplos práticos que conectam a teoria à prática. Com este estudo, compreendi melhor como esses conceitos são aplicados em áreas como diagnóstico médico, segurança e até mesmo em veículos autônomos.

Acredito que esta experiência me ajudou a solidificar conhecimentos importantes e despertou ainda mais meu interesse por Machine Learning. Espero continuar aprofundando-me nesse campo, enfrentando novos desafios e desenvolvendo soluções ainda mais completas no futuro.