Unidade II

5 OPERAÇÕES MORFOLÓGICAS

As operações morfológicas são técnicas fundamentais no processamento de imagens que envolvem a transformação geométrica de objetos em imagens binárias (preto e branco) com base em suas formas e estruturas. Essas operações são utilizadas para aprimorar e analisar características específicas em imagens, especialmente em aplicações de visão computacional e processamento de imagens. As duas operações morfológicas mais comuns são a erosão e a dilatação, frequentemente seguidas pela abertura e pelo fechamento.

- **Erosão**: a erosão é uma operação que reduz o tamanho dos objetos na imagem. Ela funciona deslocando um elemento estruturante (geralmente uma pequena máscara ou janela) pela imagem e verificando se todos os pixels sob essa janela são iguais a 1 (branco) no caso de imagens binárias. Se todos os pixels forem 1, o pixel central da janela permanece como 1; caso contrário, ele é convertido em 0 (preto). A erosão é usada para remover pequenos ruídos e detalhes de objetos.

- **Dilatação**: a dilatação é o oposto da erosão. Ela aumenta o tamanho dos objetos na imagem. Quando o elemento estruturante é deslocado pela imagem, o pixel central da janela se torna 1 se pelo menos um dos pixels sob a janela for 1. A dilatação é usada para preencher pequenos espaços entre objetos e unir objetos separados.

- **Abertura**: a abertura é uma sequência de erosão seguida de dilatação. Ela é útil para remover ruídos, separar objetos próximos e manter as formas dos objetos.

- **Fechamento**: o fechamento é o oposto da abertura, tratando-se de uma sequência de dilatação seguida de erosão. Ele é útil para fechar pequenas lacunas entre objetos e suavizar as bordas dos objetos.

- **Transformação morfológica de gradiente**: a transformação morfológica de gradiente envolve a diferença entre uma dilatação e uma erosão da imagem original. Ela destaca as bordas dos objetos na imagem.

- **Reconstrução morfológica**: a reconstrução morfológica é uma técnica avançada que combina operações de dilatação e erosão para segmentar objetos com base em suas regiões de interesse. É frequentemente usada em aplicações médicas, como segmentação de órgãos em imagens de ressonância magnética.

As operações morfológicas são valiosas em várias aplicações, incluindo processamento de imagens médicas, análise de documentos, inspeção industrial, visão computacional e muito mais. A escolha da operação morfológica e dos parâmetros depende das características da imagem e dos objetivos da aplicação. Elas são especialmente eficazes em imagens binárias, mas podem ser adaptadas para outras em tons de cinza. As operações morfológicas são partes essenciais do conjunto de ferramentas no campo do processamento de imagens.

5.1 Elemento estruturante

Um elemento estruturante é uma parte fundamental das operações morfológicas no processamento de imagens. Trata-se de uma pequena matriz ou janela, geralmente de forma geométrica simples, usada para aplicar transformações morfológicas, como erosão, dilatação, abertura, fechamento, entre outras, em uma imagem. Ele atua como um modelo que é deslocado pela imagem para realizar operações específicas com base na forma e no tamanho desse modelo.

- **Formato**: o elemento estruturante pode assumir várias formas, como quadrados, retângulos, discos, cruzes, entre outros. A escolha do formato depende da aplicação e das características dos objetos a serem analisados na imagem.

- **Tamanho**: o tamanho do elemento estruturante é determinado pelas dimensões da matriz ou janela. Ele afeta diretamente o resultado da operação morfológica. Elementos estruturantes maiores tendem a fazer com que as operações dilatem ou erodam mais, enquanto elementos menores têm um efeito mais sutil.

- **Posição relativa**: a posição do elemento estruturante em relação ao pixel central é importante. Em muitos casos, o pixel central é posicionado no centro do elemento estruturante, mas em algumas aplicações, ele pode ser deslocado para outros locais, dependendo do efeito desejado.

Elementos estruturantes binários e escalares

Os elementos estruturantes podem ser binários ou escalares. Elementos binários têm valores 0 (preto) ou 1 (branco) e são adequados para imagens binárias. Já os elementos escalares podem ter valores em uma faixa contínua e são mais adequados para imagens em tons de cinza.

Algumas operações morfológicas comuns em que utilizamos os elementos estruturantes:

- **Erosão**: o elemento estruturante é deslocado pela imagem e, se todos os pixels sob o elemento estruturante forem iguais a 1, o pixel central da janela permanece 1. Caso contrário, ele é convertido em 0.

- **Dilatação**: se pelo menos um pixel sob o elemento estruturante for igual a 1, o pixel central da janela se torna 1.

- **Abertura**: é uma sequência de erosão seguida de dilatação. Remove pequenos detalhes, ruídos e mantém as formas dos objetos.
- **Fechamento**: é uma sequência de dilatação seguida de erosão. Fecha pequenas lacunas entre objetos e suaviza as bordas.
- **Gradiente morfológico**: é a diferença entre a dilatação e a erosão de uma imagem. Destaca as bordas dos objetos.
- **Transformação morfológica de Top-Hat**: é a diferença entre a imagem original e a abertura. Pode ser usada para destacar detalhes pequenos.
- **Reconstrução morfológica**: usada em aplicações de segmentação de imagens médicas e análise de documentos.

Os elementos estruturantes são escolhidos com base nas características dos objetos que se deseja realçar ou suprimir na imagem. As operações morfológicas são essenciais para a análise e processamento de imagens em várias aplicações, incluindo visão computacional, inspeção industrial, processamento de imagens médicas e muito mais.

5.2 Erosão e dilatação

Como visto, erosão e dilatação são operações morfológicas fundamentais em processamento de imagem, sendo empregadas para processar e manipular imagens binárias (preto e branco). Essas operações são usadas para realçar características de interesse, como bordas, remover ruído e realizar transformações em imagens para análise de objetos.

Erosão em processamento de imagem

A operação de erosão é usada para diminuir o tamanho dos objetos brancos (pixels brancos) em uma imagem binária. A ideia por trás dela é deslocar um elemento estruturante (também conhecido como kernel) pela imagem e verificar se todos os pixels sob o kernel são brancos. Se todos forem brancos, o pixel central sob o kernel permanece branco; caso contrário, ele é definido como preto. Isso resulta na diminuição do tamanho dos objetos brancos na imagem.

Consta a seguir exemplo de código Python usando a biblioteca OpenCV para realização de erosão em uma imagem binária:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem binária
5 imagem = cv2.imread('sua_imagem.jpg', 0) # Certifique-se de que sua imagem seja binária (preto e branco)
```

```
6 # Defina o elemento estruturante (kernel)
7 kernel_5 = np.ones((5, 5), np.uint8) # Um kernel simples de 5x5 com todos os elementos iguais a 1
8 kernel_9 = np.ones((9, 9), np.uint8) # Um kernel simples de 9x9 com todos os elementos iguais a 1

9 # Realize a erosão
10 imagem_erodida_5 = cv2.erode(imagem, kernel_5, iterations=1)
11 imagem_erodida_9 = cv2.erode(imagem, kernel_9, iterations=1)

12 # Exiba a imagem original e a imagem erodida
13 plt.figure(figsize=(10, 5))

14 plt.subplot(1, 3, 1)
15 plt.imshow(imagem, cmap='gray')
16 plt.title('Imagem Original')
17 plt.axis('off')

18 plt.subplot(1, 3, 2)
19 plt.imshow(imagem_erodida_5, cmap='gray')
20 plt.title('Imagem Erodida (kernel 5)')
21 plt.axis('off')

22 plt.subplot(1, 3, 3)
23 plt.imshow(imagem_erodida_9, cmap='gray')
24 plt.title('Imagem Erodida (kernel 9)')
25 plt.axis('off')

26 plt.show()
```

Saída:

Figura 31

Neste exemplo, podemos observar a diminuição do tamanho da parte branca do olho, pois quando usamos erosão o resultado aparece na diminuição do tamanho dos objetos brancos na imagem.

Dilatação em processamento de imagem

A operação de dilatação é o oposto da erosão. Ela é usada para aumentar o tamanho dos objetos brancos em uma imagem binária. Assim como na erosão, um kernel é usado, mas, neste caso, se pelo menos um pixel sob o kernel for branco, o pixel central sob o kernel também é definido como branco.

Consta a seguir exemplo de código Python para realização de dilatação em uma imagem binária usando o OpenCV:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem binária
5 imagem = cv2.imread('sua_imagem.jpg', 0) # Certifique-se de que sua imagem seja binária (preto e branco)

6 # Defina o elemento estruturante (kernel)
7 kernel_5 = np.ones((5, 5), np.uint8) # Um kernel simples de 5x5 com todos os elementos iguais a 1
8 kernel_9 = np.ones((9, 9), np.uint8) # Um kernel simples de 9x9 com todos os elementos iguais a 1

9 # Realize a dilatação
10 imagem_dilatada_5 = cv2.dilate(imagem, kernel_5, iterations=1)
11 imagem_dilatada_9 = cv2.dilate(imagem, kernel_9, iterations=1)

12 # Exiba a imagem original e a imagem dilatada
13 plt.figure(figsize=(10, 5))

14 plt.subplot(1, 3, 1)
15 plt.imshow(imagem, cmap='gray')
16 plt.title('Imagem Original')
17 plt.axis('off')

18 plt.subplot(1, 3, 2)
19 plt.imshow(imagem_dilatada_5, cmap='gray')
20 plt.title('Imagem Dilatada (kernel 5)')
21 plt.axis('off')

22 plt.subplot(1, 3, 3)
23 plt.imshow(imagem_dilatada_9, cmap='gray')
24 plt.title('Imagem Dilatada (kernel 9)')
25 plt.axis('off')

26 plt.show()
```

Saída:

Figura 32

Neste exemplo, podemos observar o aumento do tamanho da parte branca do olho, pois quando usamos dilatação o resultado aparece no aumento do tamanho dos objetos brancos na imagem. Podemos observar também que perdemos a boca e o contorno do rosto, o que poderia ser um problema dependendo da aplicação.

5.3 Abertura e fechamento

A abertura e o fechamento são duas operações morfológicas importantes em processamento de imagem que envolvem a erosão e a dilatação. Elas são usadas para melhorar a qualidade das imagens, remover ruído, separar objetos e conectar objetos, dependendo do contexto da aplicação.

Abertura em processamento de imagem

A abertura é uma operação morfológica que consiste em aplicar a erosão seguida da dilatação em uma imagem. Ela é usada principalmente para remover pequenos detalhes (ruído) e abrir pequenos espaços entre objetos. A abertura pode ser útil para separar objetos que estão muito próximos.

Consta a seguir exemplo de código Python usando OpenCV para realização de abertura em uma imagem binária:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem binária
5 imagem = cv2.imread('sua_imagem.jpg', 0) # Certifique-se de que sua imagem seja binária (preto e branco)

6 # Defina o elemento estruturante (kernel)
7 kernel_5 = np.ones((5, 5), np.uint8) # Um kernel simples de 5x5 com todos os elementos iguais a 1
8 kernel_9 = np.ones((9, 9), np.uint8) # Um kernel simples de 9x9 com todos os elementos iguais a 1
```

```
# Realize a abertura
imagem_abertura_5 = cv2.morphologyEx(imagem, cv2.MORPH_OPEN, kernel_5)
imagem_abertura_9 = cv2.morphologyEx(imagem, cv2.MORPH_OPEN, kernel_9)

# Exiba a imagem original e a imagem erodida
plt.figure(figsize=(10, 5))

plt.subplot(1, 3, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(imagem_abertura_5, cmap='gray')
plt.title('Imagem Abertura(kernel 5)')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(imagem_abertura_9, cmap='gray')
plt.title('Imagem Abertura (kernel 9)')
plt.axis('off')

plt.show()
```

Saída:

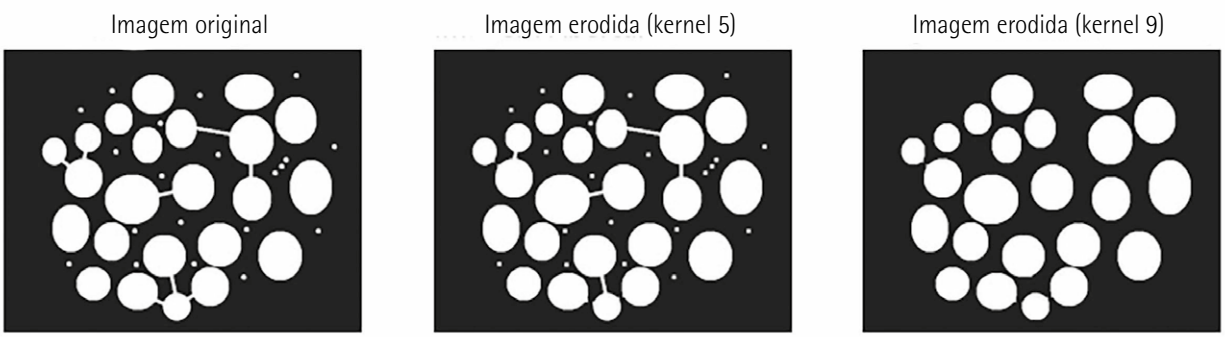

Figura 33

Neste exemplo, podemos observar a separação dos objetos que estão muito próximos.

Fechamento em processamento de imagem

O fechamento é a operação morfológica oposta à abertura. Ela consiste em aplicar a dilatação seguida da erosão em uma imagem. O fechamento é usado para preencher pequenos buracos em objetos e fechar pequenas lacunas entre eles. Isso pode ser útil para conectar objetos que estão quase se tocando.

Consta a seguir exemplo de código Python usando OpenCV para realização de fechamento em uma imagem binária:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem binária
5 imagem = cv2.imread('sua_imagem.jpg', 0) # Certifique-se de que sua imagem seja binária (preto e branco)

6 # Defina o elemento estruturante (kernel)
7.kernel_5 = np.ones((5, 5), np.uint8) # Um kernel simples de 5x5 com todos os elementos iguais a 1
8 kernel_9 = np.ones((9, 9), np.uint8) # Um kernel simples de 9x9 com todos os elementos iguais a 1

9 # Realize a fechamento
10 imagem_fechamento_5 = cv2.morphologyEx(imagem, cv2.MORPH_CLOSE, kernel_5)
11 imagem_fechamento_9 = cv2.morphologyEx(imagem, cv2.MORPH_CLOSE, kernel_9)

12 # Exiba a imagem original e a imagem erodida
13 plt.figure(figsize=(10, 5))

14 plt.subplot(1, 3, 1)
15 plt.imshow(imagem, cmap='gray')
16 plt.title('Imagem Original')
17 plt.axis('off')

18 plt.subplot(1, 3, 2)
19 plt.imshow(imagem_fechamento_5, cmap='gray')
20 plt.title('Imagem Fechamento(kernel 5)')
21 plt.axis('off')

22 plt.subplot(1, 3, 3)
23 plt.imshow(imagem_fechamento_9, cmap='gray')
24 plt.title('Imagem Fechamento (kernel 9)')
25 plt.axis('off')

26 plt.show()
```

Saída:

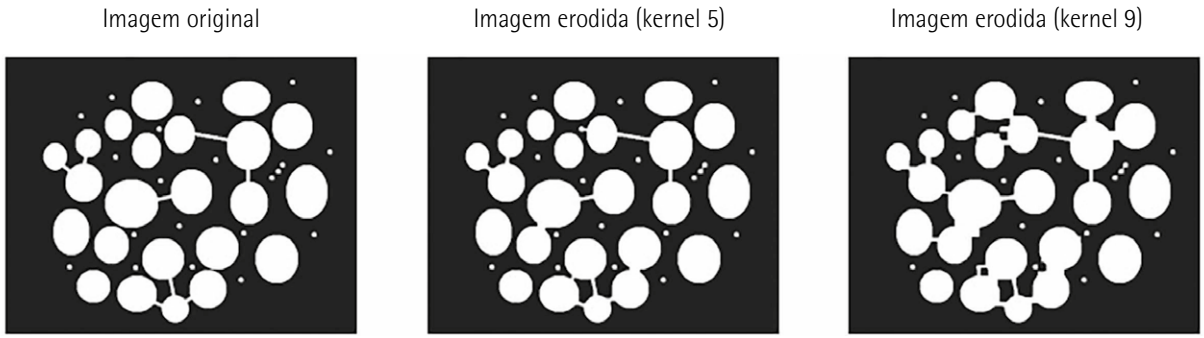

Figura 34

Neste exemplo, podemos observar entre os objetos o fechamento de pequenas lacunas entre eles, muito útil para conectar objetos que estão quase se tocando.

Essas operações morfológicas de abertura e fechamento são valiosas em várias aplicações de processamento de imagem, incluindo segmentação de objetos, remoção de ruído e preparação de imagens para análise de objetos.

5.4 Gradiente morfológico

O gradiente morfológico é uma operação morfológica que calcula a diferença entre a dilatação e a erosão de uma imagem. Essa operação é usada para realçar as bordas dos objetos em uma imagem binária. O seu resultado mostra onde as transições abruptas de valores de pixel ocorrem, o que geralmente corresponde às bordas dos objetos na imagem.

Na sequência está uma explicação mais detalhada de como o gradiente morfológico funciona:

- Primeiramente é realizada uma erosão na imagem original com um kernel estruturante.
- Em seguida, ocorre a dilatação na mesma imagem original usando o mesmo kernel estruturante.
- O resultado da dilatação é subtraído do resultado da erosão para produzir o gradiente morfológico.

Consta a seguir exemplo de código Python usando OpenCV para cálculo do gradiente morfológico em uma imagem binária:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem binária
5 imagem = cv2.imread('sua_imagem.jpg', 0) # Certifique-se de que sua imagem seja binária (preto e branco)

6 # Defina o elemento estruturante (kernel)
7 kernel = np.ones((5, 5), np.uint8) # Um kernel simples de 5x5 com todos os elementos iguais a 1

8 # Realize a erosão
9 imagem_erosao = cv2.erode(imagem, kernel, iterations=1)

10 # Realize a dilatação
11 imagem_dilatacao = cv2.dilate(imagem, kernel, iterations=1)

12 # Calcule o gradiente morfológico
13 gradiente_morfologico = cv2.subtract(imagem_dilatacao, imagem_erosao)

14 # Exiba as imagens
15 plt.rcParams.update({'font.size': 20})
16 plt.figure(figsize=(15, 10))

17 plt.subplot(2, 2, 1)
18 plt.imshow(imagem, cmap='gray')
19 plt.title('Imagem Original')
20 plt.axis('off')

21 plt.subplot(2, 2, 2)
22 plt.imshow(imagem_erosao, cmap='gray')
23 plt.title('Imagem após Erosão (kernel 5)')
24 plt.axis('off')

25 plt.subplot(2, 2, 3)
26 plt.imshow(imagem_dilatacao, cmap='gray')
27 plt.title('Imagem após Dilatação (kernel 5)')
28 plt.axis('off')

29 plt.subplot(2, 2, 4)
30 plt.imshow(gradiente_morfologico, cmap='gray')
31 plt.title('Gradiente Morfológico')
32 plt.axis('off')

33plt.show()
```

Saída:

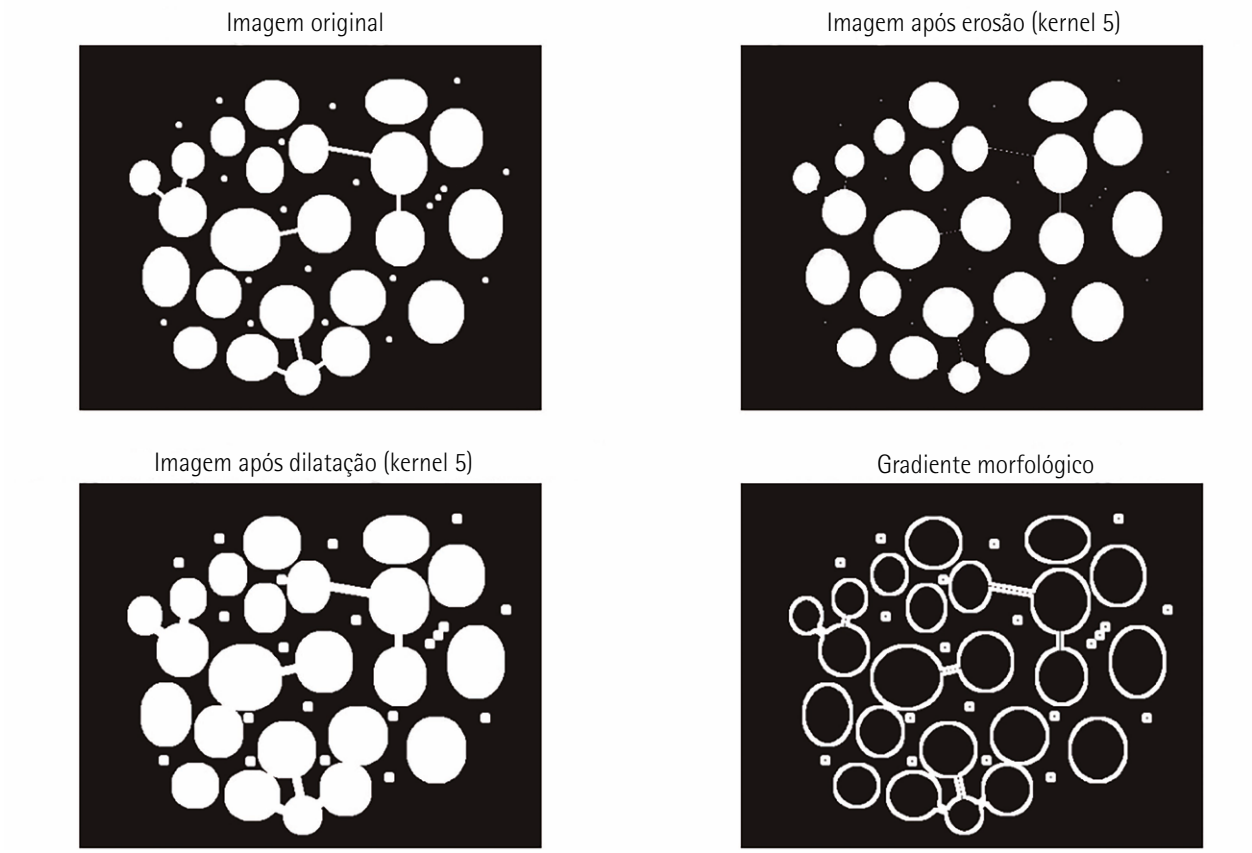

Figura 35

O resultado do gradiente morfológico é uma imagem cujas bordas dos objetos são realçadas, o que pode ser útil em várias aplicações, como detecção de bordas, segmentação de objetos e análise de características em imagens binárias.

5.5 Top Hat

O operador Top Hat, também conhecido como "Top Hat morfológico", é uma operação morfológica usada para realçar os detalhes finos ou pequenos estruturais em uma imagem. Ele é aplicado em imagens para destacar características que são significativamente menores do que o elemento estruturante empregado na operação.

O Top Hat é calculado subtraindo-se a imagem original da operação de abertura morfológica. Aqui está como funciona:

- Primeiramente é aplicada uma abertura morfológica na imagem original. A abertura é uma operação que consiste em uma erosão seguida de dilatação com um elemento estruturante.

- A imagem original é subtraída do resultado da abertura.

- O resultado dessa operação é uma imagem que realça as características finas e pequenas da imagem original, como pequenos objetos ou detalhes.

Consta a seguir exemplo de código Python usando OpenCV para cálculo do operador Top Hat em uma imagem:

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt

4 # Carregue a imagem
5 imagem = cv2.imread('sua_imagem.jpg', 0)

6 # Defina o elemento estruturante (kernel)
7 kernel = np.ones((5, 5), np.uint8)

8 # Realize a operação de abertura morfológica
9 imagem_abertura = cv2.morphologyEx(imagem, cv2.MORPH_OPEN, kernel)

10 # Calcule o operador Top Hat subtraindo a imagem original da abertura
11 top_hat = cv2.subtract(imagem, imagem_abertura)

12 # Exibe as imagens
13 plt.rcParams.update({'font.size': 20})
14 plt.figure(figsize=(15, 10))

15 plt.subplot(2, 2, 1)
16 plt.imshow(imagem, cmap='gray')
17 plt.title('Imagem Original')
18 plt.axis('off')

19 plt.subplot(2, 2, 2)
20 plt.imshow(imagem_abertura, cmap='gray')
21 plt.title('Imagem Após Abertura (kernel 5)')
22 plt.axis('off')

23 plt.subplot(2, 2, 3)
24 plt.imshow(top_hat, cmap='gray')
25 plt.title('Operador Top Hat')
26 plt.axis('off')

27 plt.show()
```

Saída:

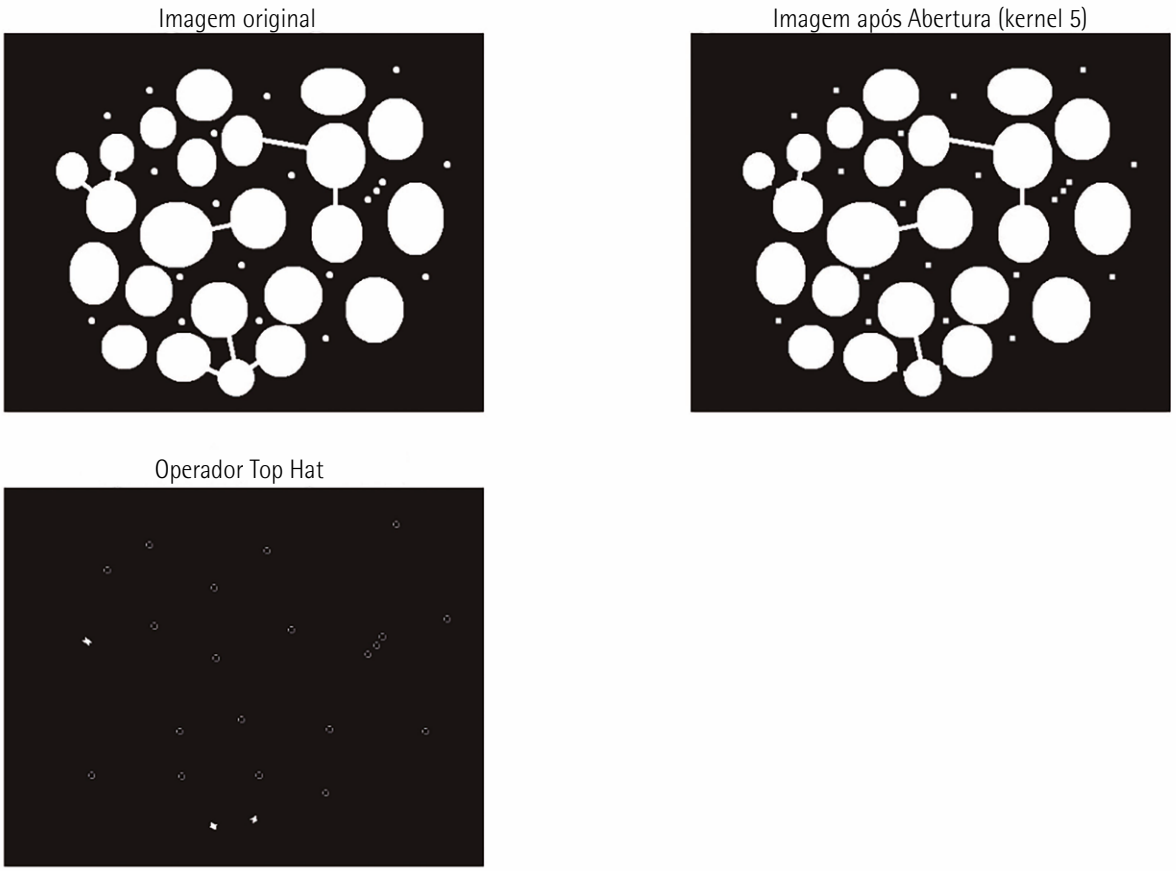

Figura 36

O resultado do operador Top Hat é uma imagem cujos detalhes finos ou pequenos são realçados em relação ao fundo, tornando-o útil em várias aplicações, como detecção de objetos pequenos, remoção de ruído fino e realce de características sutis em imagens.

5.6 Detecção de descontinuidades

A segmentação pode seguir uma estratégia de descontinuidade, quando ocorre a partição da imagem efetuada com base nas alterações bruscas de intensidade (por exemplo, detecção de contornos).

As técnicas de detecção de descontinuidade são usadas para identificar transições abruptas de intensidade, que geralmente correspondem a bordas, limites ou descontinuidades em uma imagem. A detecção de descontinuidade é uma etapa fundamental em várias aplicações de processamento de imagem, como segmentação de objetos, reconhecimento de padrões e análise de texturas. Algumas das técnicas comuns de detecção de descontinuidade incluem:

- **Operadores de gradiente**: os operadores de gradiente, como os operadores de Sobel, Prewitt e Roberts, calculam a magnitude do gradiente da intensidade da imagem em direções horizontal e vertical. As regiões com maiores magnitudes de gradiente são interpretadas como bordas.

- **Operador laplaciano**: o operador laplaciano calcula a segunda derivada da intensidade da imagem em relação às coordenadas x e y. Regiões com valores de laplaciano significativamente diferentes de zero são consideradas bordas.

- **Filtro de máscara de desaguçamento (LoG – Laplacian of Gaussian)**: o filtro LoG é a combinação do operador laplaciano com um filtro gaussiano. Ele suaviza a imagem com um filtro gaussiano e, em seguida, calcula o laplaciano da imagem suavizada para identificar bordas.

- **Detector de bordas de Canny**: o detector de bordas de Canny é um método sofisticado que combina detecção de bordas, supressão de não máximos e histerese para produzir bordas nítidas e bem definidas.

- **Filtro de máscara gradiente**: os filtros gradientes são usados para calcular a magnitude e a orientação do gradiente em diferentes direções. O ângulo do gradiente pode ser usado para identificar a orientação das bordas.

- **Transformada de Hough**: a transformada de Hough é uma técnica que pode ser usada para detectar linhas retas em uma imagem, o que é útil para identificar bordas lineares. Esses filtros são projetados para identificar pontos de interesse e, portanto, são úteis para a detecção de cantos e características distintas.

A escolha da técnica de detecção de descontinuidade depende das características da imagem, do tipo de descontinuidade que se deseja detectar e dos requisitos da aplicação.

5.7 Detecção de similaridades

A segmentação pode seguir uma estratégia de similaridade, com a partição sendo efetuada com base na similaridade entre pixels, seguindo um determinado critério (por exemplo, binarização, crescimento de regiões, divisão e junção de regiões).

Binarização

A binarização é um processo de pré-processamento de imagens que envolve a conversão de uma imagem em tons de cinza em outra binária, cujos pixels são representados como valores binários, geralmente 0 (preto) ou 1 (branco). Essa técnica é comumente usada em visão computacional para destacar objetos de interesse em uma imagem, separando-os do fundo com base na intensidade do pixel. A binarização é uma etapa importante para a segmentação de imagens e a extração de características.

Consta a seguir exemplo de binarização de uma imagem em Python usando a biblioteca OpenCV:

```
1 import cv2
2 import matplotlib.pyplot as plt

3# Carregue a imagem
4 imagem = cv2.imread('sua_imagem.JPG', cv2.IMREAD_GRAYSCALE)

5 # Escolha um valor de limiar (threshold)
6 limiar = 180

7# Aplicar a binarização
8 _, imagem_binaria = cv2.threshold(imagem, limiar, 255, cv2.THRESH_BINARY)

9 # Exibir as imagens
10 plt.figure(figsize=(15, 20))

11 plt.subplot(2, 2, 1)
12 plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
13 plt.title('Imagem Original')
14 plt.axis('off')

15 plt.subplot(2, 2, 2)
16 plt.imshow(cv2.cvtColor(imagem_binaria, cv2.COLOR_BGR2RGB))
17 plt.title('Imagem Binarizada')
18 plt.axis('off')

19 plt.show()
```

Saída:

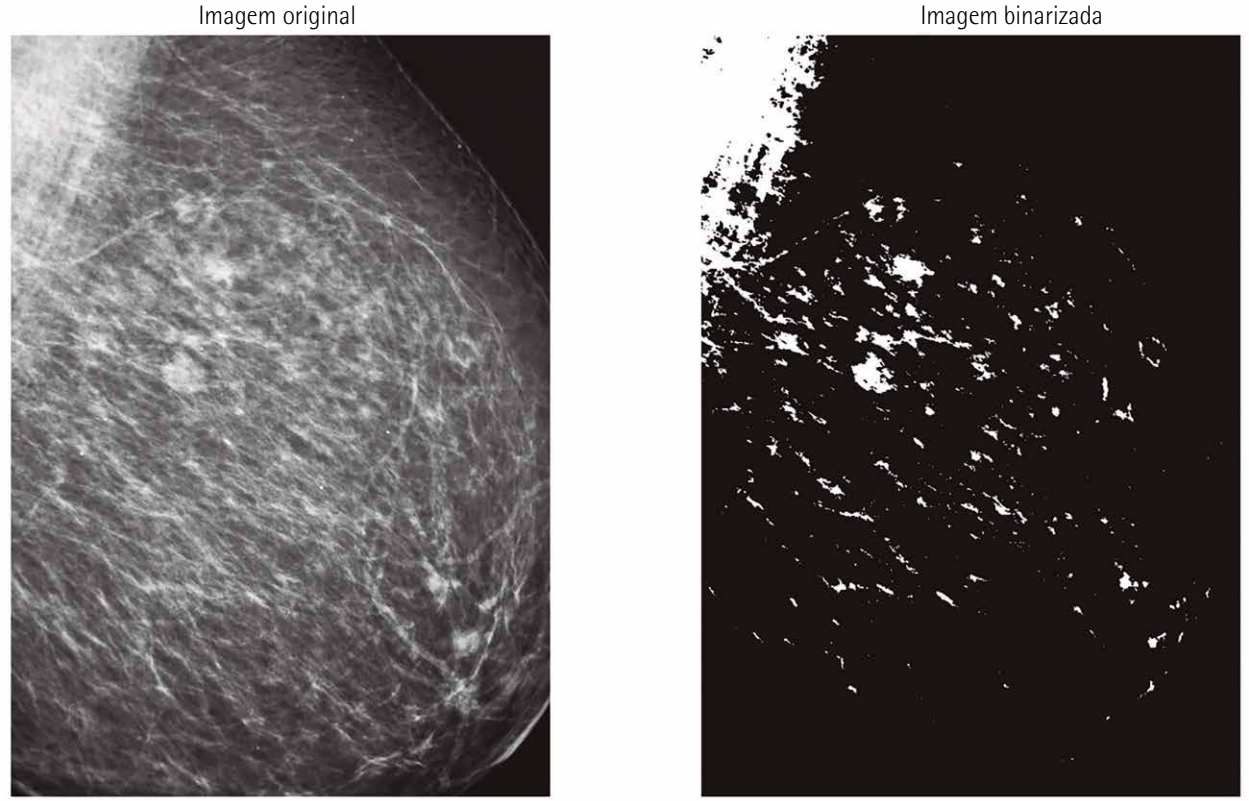

Figura 37 – Mamografia em projeção OML mostrando nódulo de densidade semelhante à do parênquima glandular, com contornos bem definidos, são características de lesão

Disponível em: https://tinyurl.com/9t43hxb8. Acesso em: 24 out. 2023.

As técnicas de detecção de similaridade são usadas para comparar e identificar regiões ou objetos semelhantes em imagens. Isso é útil em várias aplicações, como correspondência de padrões, reconhecimento de objetos e identificação de regiões semelhantes em imagens médicas, entre outros. Algumas das técnicas de detecção de similaridade mais comuns incluem:

- **Correlação cruzada**: a correlação cruzada compara duas imagens deslizando uma sobre a outra e calculando a medida de similaridade entre elas. A posição com a maior medida de similaridade indica uma correspondência.

- **Template matching**: a técnica de template matching envolve o uso de uma imagem de referência (template) que é comparada com regiões da imagem de entrada. A região com a melhor correspondência ao template é identificada como uma correspondência.

- **Descritores de características**: descritores de características, como Sift (scale-invariant feature transform) e Surf (speeded-up robust features), identificam pontos de interesse e descrevem suas características distintas. A correspondência é estabelecida comparando esses descritores entre imagens.

- **Histograma de cores**: a comparação de histogramas de cores é usada para medir a similaridade de distribuições de cores em imagens. É útil em aplicações de classificação e recuperação de imagens.

- **Métodos baseados em textura**: estes métodos comparam características de textura, como o padrão de co-ocorrência de intensidade (GLCM) e características da transformada de Fourier para identificar áreas de textura semelhante.

- **Correspondência de ponto a ponto**: algoritmos de correspondência de ponto a ponto, como o Ransac (random sample consensus), são usados para identificar correspondências entre pontos de interesse em diferentes imagens.

- **Descritores locais**: descritores locais, como o HOG (histogram of oriented gradients), são usados para identificar objetos com base nas características locais, como bordas e gradientes.

- **Aprendizado de máquina**: algoritmos de aprendizado de máquina, como redes neurais convolucionais (CNNs), podem ser treinados para realizar tarefas de correspondência de objetos e reconhecimento de padrões.

A escolha da técnica de detecção de similaridade depende do contexto e dos requisitos específicos da aplicação. Cada uma delas tem suas vantagens e limitações, e a seleção apropriada deve ser baseada nas características das imagens e na natureza da correspondência desejada. Em muitos casos, a combinação pode ser usada para a obtenção de resultados mais robustos e precisos.

Saiba mais

A fim de compreender melhor sobre transformações de intensidade e filtragem espacial, e filtragem no domínio da frequência, recomendamos a leitura dos capítulos 3 e 4 da seguinte obra:

GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens*. 3. ed. São Paulo: Pearson, 2010.

6 EXTRAÇÃO DE CARACTERÍSTICAS

A extração de características em processamento de imagem refere-se ao processo de identificar, isolar e medir características específicas em uma imagem que são relevantes para uma determinada tarefa de análise ou reconhecimento. Essas características podem ser estruturais, estatísticas, texturais, geométricas ou baseadas em intensidade, dependendo dos requisitos da aplicação. A extração de características é fundamental em diversas áreas, podemos listar algumas das categorias comuns extraídas em processamento de imagem:

Características de intensidade

- **Histograma**: representa a distribuição de intensidades de pixel na imagem.
- **Momentos de intensidade**: calcula informações estatísticas sobre a intensidade dos pixels, como média, variância, assimetria e curtose.

Características estruturais

- **Bordas**: detecta transições abruptas de intensidade na imagem.
- **Cantos**: identifica pontos onde as bordas se encontram.
- **Linhas e formas**: detecta linhas retas, curvas e formas geométricas.

Características de textura

- **Matriz de co-ocorrência**: mede a distribuição espacial de padrões de intensidade.
- **Transformada de Fourier**: analisa a textura em termos de frequência espacial.
- **Filtros de textura**: aplica filtros para realçar ou medir a textura.

Características geométricas

- **Área e perímetro**: calcula a área e o perímetro de objetos na imagem.
- **Centroide**: encontra o centro de massa de um objeto.
- **Eixos principais**: identifica os eixos principais de um objeto.

Características baseadas em histograma

- **Histograma de cores**: descreve a distribuição de cores em uma imagem colorida.
- **Histograma de gradientes**: mostra a distribuição de gradientes de intensidade na imagem.

Características de texto

- **Reconhecimento de texto**: extrai texto de imagens para fins de OCR (reconhecimento óptico de caracteres).
- **Descritores de texto**: mede propriedades de texto, como tamanho, orientação e forma.

Características de desempenho de detecção

- **Sensibilidade e especificidade**: mede a capacidade de um detector em identificar verdadeiros positivos e evitar falsos positivos.

- **Curva ROC**: avalia o desempenho do detector em diferentes pontos de corte.

Características de frequência espacial

- **Espectro de frequência**: mostra as frequências presentes na imagem.

- **Filtros de frequência**: aplica filtros para realçar ou suprimir frequências específicas.

A escolha das características a serem extraídas depende da tarefa específica. Após a extração, as características podem ser usadas para treinar algoritmos de aprendizado de máquina, realizar correspondência de padrões, segmentar objetos, reconhecer objetos e muito mais. É importante selecionar as características apropriadas a fim de garantir que elas representem eficazmente as informações necessárias para a tarefa de processamento de imagem em questão.

6.1 Redução de dimensionalidade

A redução de dimensionalidade é o processo de reduzir o número de variáveis ou dimensões em uma imagem, enquanto tenta manter as informações essenciais e relevantes. Isso é feito principalmente para simplificar a análise, a visualização ou o armazenamento de imagens, especialmente quando elas têm alta resolução ou alta dimensionalidade, o que pode ser computacionalmente caro e difícil de lidar. A redução de dimensionalidade em processamento de imagem é usada com técnicas de extração de características para melhorar a eficiência e a eficácia da análise de imagens. Existem duas abordagens comuns para a redução de dimensionalidade em processamento de imagem. São elas:

- Redução de dimensionalidade linear

- **Análise de componentes principais (PCA)**: é uma técnica estatística que encontra as direções (componentes principais) quando os dados têm a maior variabilidade. PCA é usado para projetar os dados em um espaço de dimensão inferior enquanto tenta preservar a maior parte da variância.

- **Análise discriminante linear (LDA)**: ao contrário do PCA, o LDA é usado para encontrar as direções enquanto os dados são mais discriminativos entre diferentes classes. É frequentemente usado em tarefas de classificação.

- **Análise de fatoração de matriz (NMF)**: é uma técnica que fatora uma matriz de dados em dois subconjuntos de matrizes de menor dimensionalidade, geralmente sendo usada para a redução de dimensionalidade e a análise de tópicos em dados de texto ou imagem.

Redução de dimensionalidade não linear

- **T-distributed stochastic neighbor embedding (t-SNE)**: é uma técnica que mapeia os dados de alta dimensão para um espaço de menor dimensão de maneira que objetos semelhantes permaneçam próximos e objetos diferentes fiquem separados. É usado principalmente para visualização e redução de dimensionalidade em tarefas de agrupamento.

- **Isomap**: é uma técnica que cria um grafo de vizinhos próximos com base nas distâncias entre os dados e, em seguida, encontra uma representação de baixa dimensão que preserva essas distâncias geodésicas.

- **Autoencoders**: são redes neurais usadas para aprender representações compactas e eficientes dos dados. Eles consistem em uma rede que codifica (reduz a dimensionalidade) e decodifica (reconstrói a imagem original) os dados, sendo úteis para a redução de dimensionalidade e a geração de imagens.

A escolha entre abordagens lineares e não lineares depende da natureza dos dados e das tarefas a serem realizadas. A redução de dimensionalidade pode ser benéfica para tarefas como reconhecimento de padrões, classificação, agrupamento, recuperação de informações e visualização de dados. No entanto, é importante lembrar que a redução de dimensionalidade pode resultar na perda de informações e, portanto, deve ser usada com cuidado, garantindo que as características essenciais sejam retidas.

6.2 Maldição da dimensionalidade

A maldição da dimensionalidade é um conceito importante em processamento de imagem e em muitos outros campos relacionados ao processamento de dados e aprendizado de máquina. Ela se refere aos desafios e complicações que surgem quando se lida com conjuntos de dados de alta dimensionalidade, quando o número de características ou dimensões é significativamente maior do que o de amostras de dados disponíveis. Este problema afeta não apenas o processamento de imagem, mas áreas como mineração de dados, aprendizado de máquina e estatísticas.

Com o aumento da dimensionalidade, os dados tendem a se tornar esparsos, o que significa que a maioria das combinações de características não é representada nos dados. Isso torna a modelagem mais difícil, pois não há informações suficientes para aprender com precisão as relações entre as características.

O número de combinações possíveis de características cresce exponencialmente com a dimensionalidade. Isso significa que, à medida que adicionamos mais dimensões aos dados, precisamos de uma quantidade exponencialmente maior de dados para representar adequadamente essas dimensões.

Em conjuntos de dados de alta dimensionalidade, os modelos de aprendizado de máquina têm maior probabilidade de se ajustar demais aos dados de treinamento, capturando ruído em vez de padrões reais. Isso pode levar a modelos que não generalizam bem para novos dados (overfitting).

O aumento na dimensionalidade também leva ao crescimento significativo nos custos computacionais. O treinamento e a avaliação de modelos em dados de alta dimensão podem ser muito mais demorados e exigir recursos computacionais substanciais. À medida que a dimensionalidade aumenta, torna-se cada vez mais difícil visualizar os dados. Isso pode dificultar a compreensão das relações entre as características e a interpretação dos resultados.

Em conjuntos de dados de alta dimensão, pode ser desafiador determinar quais características são realmente relevantes para a tarefa em questão. Muitas delas podem ser redundantes ou irrelevantes, o que dificulta a seleção das características mais informativas.

Para lidar com a maldição da dimensionalidade em processamento de imagem e em outras áreas, é importante considerar estratégias como:

- **Seleção de características**: identificar e manter apenas as características mais relevantes e informativas, eliminando as redundantes ou irrelevantes.
- **Extração de características**: usar técnicas de extração de características para reduzir a dimensionalidade, transformando os dados em um espaço de características mais compacto e informativo.
- **Agrupamento e redução de dimensionalidade**: aplicar técnicas de agrupamento para associar características relacionadas e reduzir a dimensionalidade de forma controlada.
- **Regularização**: usar técnicas de regularização, como L1 e L2, para evitar o overfitting em modelos de aprendizado de máquina.
- **Aumento de dados**: se possível, coletar mais dados para lidar com o problema de esparsidade de dados em dimensões elevadas.
- **Visualização de dados**: quando possível, usar técnicas de visualização, como PCA ou t-SNE, a fim de projetar os dados em um espaço de menor dimensão para melhor compreensão e interpretação.

Observação

Técnicas de regularização são métodos usados em aprendizado de máquina e aprendizado profundo para evitar overfitting de modelos. O overfitting ocorre quando um modelo se ajusta demais aos dados de treinamento e não generaliza bem para novos dados. As técnicas de regularização introduzem penalidades nos parâmetros do modelo durante o treinamento, incentivando-os a manter valores menores e, assim, reduzindo a complexidade do modelo.

Duas técnicas de regularização comuns são a L1 (regularização Lasso) e a L2 (regularização Ridge):

Regularização L1 (Lasso): na regularização L1, é adicionada uma penalização à função de custo do modelo, que é proporcional ao valor absoluto dos parâmetros do modelo. Isso significa que a regularização L1 tende a forçar alguns parâmetros do modelo a se tornarem exatamente zero, o que efetivamente seleciona um subconjunto de características importantes, realizando seleção automática de características.

A L1 é útil para a seleção de características, reduzindo a dimensionalidade dos dados e tornando o modelo mais simples e interpretável.

Regularização L2 (Ridge): na regularização L2, a penalização adicionada à função de custo é proporcional ao quadrado dos parâmetros do modelo. A L2 não força os parâmetros a se tornarem exatamente zero, mas os incentiva a serem pequenos, tornando-os mais uniformes e evitando valores extremamente altos.

A L2 é útil para suavizar as variações nos parâmetros, tornando o modelo mais robusto e resistente ao overfitting.

Ambas as técnicas de regularização, L1 e L2, podem ser usadas separadamente ou combinadas em uma técnica chamada regularização Elastic Net, que combina as penalizações da L1 e da L2. A escolha entre L1 e L2 (ou ambas) depende da natureza do problema e da necessidade de seleção de características. A L1 é preferida quando se suspeita que muitos parâmetros não são relevantes, enquanto a L2 é usada para evitar valores extremamente altos e suavizar o modelo.

A maldição da dimensionalidade é um desafio crítico em muitos campos de análise de dados, e abordá-la de maneira adequada é essencial para obter resultados significativos e evitar problemas como overfitting e baixa generalização de modelos.

6.3 Segmentação no espaço de atributos

A segmentação no espaço de atributos é um processo fundamental no processamento de imagem que envolve a divisão de uma imagem em regiões ou objetos com base nas características dos pixels. Essas características podem ser extraídas do espaço de atributos da imagem, que é uma representação multidimensional na qual cada dimensão corresponde a uma característica ou atributo específico dos pixels. A segmentação no espaço de atributos é uma etapa crítica em várias aplicações de processamento de imagem, como reconhecimento de objetos, rastreamento de objetos, análise de cena e muito mais. A seguir constam os principais conceitos e abordagens relacionados à segmentação no espaço de atributos:

- **Espaço de atributos**: o espaço de atributos é uma representação multidimensional que descreve as características dos pixels em uma imagem. Cada dimensão do espaço de atributos corresponde a uma característica específica, como intensidade, cor, textura, gradiente, entre outros. A escolha das características a serem usadas depende da tarefa de segmentação e dos aspectos distintivos dos objetos de interesse.

- **Segmentação por limiarização**: a limiarização é uma técnica simples de segmentação que se baseia na definição de um ou mais limiares no espaço de atributos. Pixels com valores de atributo que atendem aos critérios de limiar são atribuídos a uma classe ou região específica. Por exemplo, na limiarização de intensidade, pixels com intensidade acima de um limiar são considerados como objeto, enquanto aqueles abaixo do limiar são considerados como fundo.

- **Segmentação por regiões**: em vez de usar limiares fixos, a segmentação por regiões busca identificar grupos de pixels com características semelhantes no espaço de atributos. Isso é frequentemente feito usando técnicas de agrupamento, como k-means ou regiões crescentes. Os pixels são agrupados em regiões com base na proximidade das características no espaço de atributos.

- **Segmentação por borda**: a segmentação por borda visa identificar as bordas ou contornos dos objetos na imagem. Isso é feito detectando transições abruptas nas características, como mudanças na intensidade, gradiente ou textura. Algoritmos de detecção de borda, como os operadores Sobel ou Canny, são comumente usados para esta finalidade.

- **Segmentação por crescimento de região**: este método começa com um ou mais pontos de semente na imagem e expande uma região conectada em torno delas com base em critérios de similaridade no espaço de atributos. Isso é útil para segmentar objetos conectados ou áreas homogêneas.

- **Segmentação por aprendizado de máquina**: o aprendizado de máquina, como redes neurais convolucionais (CNNs), também é utilizado para a segmentação no espaço de atributos. Esses modelos podem aprender automaticamente a identificar objetos com base em um grande conjunto de dados de treinamento.

- **Segmentação semântica**: é uma forma avançada de segmentação que atribui uma classe semântica a cada pixel na imagem. Isso é comum em aplicações de visão computacional, como veículos autônomos e processamento de imagens médicas.

A escolha da técnica de segmentação no espaço de atributos depende das características dos objetos de interesse na imagem e dos requisitos da aplicação. Muitas vezes, uma combinação de técnicas é usada para obter resultados mais precisos e robustos. A segmentação no espaço de atributos é uma etapa fundamental no processamento de imagem que permite extrair informações significativas de uma imagem para análise, interpretação e tomada de decisões.

6.4 Análise de componentes principais (PCA)

A análise de componentes principais (PCA) é uma técnica de redução de dimensionalidade utilizada em processamento de imagem e em diversas outras áreas da análise de dados. Ela é usada para extrair as principais informações ou características de um conjunto de dados, reduzindo sua dimensionalidade, mantendo ao mesmo tempo a maior parte da variância dos dados. Em processamento de imagem, a PCA é aplicada para reduzir a quantidade de informações nas imagens, torná-las mais gerenciáveis e representativas e, muitas vezes, para fins de compressão de imagem.

O que é a análise de componentes principais (PCA)?

Trata-se de uma técnica de transformação linear que calcula as direções (componentes principais) ao longo das quais os dados têm a maior variabilidade. Essas direções correspondem aos autovetores da matriz de covariância dos dados. Ao projetar os dados nas direções dos autovetores mais importantes, é possível reduzir a dimensionalidade dos dados, mantendo grande parte da informação original.

Aplicação da PCA em processamento de imagem

Em processamento de imagem, a PCA é aplicada principalmente para:

- **Redução de dimensionalidade**: em imagens digitais, cada pixel pode ser considerado uma dimensão, o que pode resultar em um espaço de alta dimensionalidade. A PCA permite reduzir a dimensionalidade das imagens, mantendo as características mais informativas. Isso é útil a fim de economizar espaço de armazenamento, acelerar algoritmos de processamento e análise de imagem e melhorar a visualização de dados.

- **Compressão de imagem**: a PCA é usada em técnicas de compressão de imagem, como a compressão de imagem JPEG. Ao representar a imagem em um espaço de dimensão reduzida com base nas principais componentes, é possível obter uma representação compacta da imagem que preserva a maioria das informações visuais.

- **Análise de texturas**: a PCA pode ser aplicada a texturas em imagens para reduzir a dimensionalidade e identificar características importantes das texturas, facilitando a classificação e a análise.

- **Remoção de ruído**: a PCA pode ser usada para separar informações de sinal de ruído em imagens, concentrando-se nas principais componentes que representam o sinal.

Como a PCA funciona em processamento de imagem

- **Coleta de dados**: as imagens são coletadas e representadas como matrizes de pixels, uma vez que cada pixel é uma dimensão.

- **Padronização**: os dados são geralmente padronizados (subtraindo a média e dividindo pelo desvio padrão) para garantir que todas as dimensões tenham uma escala comparável.

- **Cálculo da matriz de covariância**: a matriz de covariância dos dados é calculada. Esta matriz descreve como as diferentes dimensões dos dados estão correlacionadas.

- **Cálculo dos autovetores e autovalores**: os autovetores e autovalores da matriz de covariância são calculados. Os autovetores representam as direções principais nos dados, já os autovalores indicam a importância relativa dessas direções.

- **Seleção das componentes principais**: as principais componentes são selecionadas com base na magnitude dos autovalores. Normalmente, elas são ordenadas em ordem decrescente de importância.

- **Projeção dos dados**: os dados são projetados nos principais componentes selecionadas. Isso reduz a dimensionalidade deles, criando uma representação das imagens em um espaço de dimensão reduzida.

Lembrete

A PCA é uma técnica poderosa em processamento de imagem usada em combinação com outras técnicas de análise de dados e aprendizado de máquina para melhorar a eficiência e a eficácia de várias tarefas de processamento de imagem. Ela ajuda a simplificar a representação de imagens enquanto mantém a maior parte das informações importantes.

6.5 Análise discriminante linear (LDA)

A análise discriminante linear (LDA) é uma técnica estatística e de aprendizado de máquina usada em processamento de imagem e em várias outras áreas para a redução de dimensionalidade e a classificação de dados. Ela é valiosa quando o objetivo é encontrar as características que melhor separam ou discriminam diferentes classes de dados. Em processamento de imagem, a LDA é aplicada principalmente para melhorar a eficiência da classificação de imagens, em especial em tarefas de reconhecimento de padrões.

O que é a análise discriminante linear (LDA)?

A LDA é uma técnica que se concentra na maximização da separação entre diferentes classes de dados, ao mesmo tempo em que minimiza a dispersão nas classes. Ela é usada para encontrar combinações lineares das características originais que melhor discriminam ou separam diferentes classes de dados. A principal diferença entre a LDA e a análise de componentes principais (PCA) é que a LDA é supervisionada, enquanto a PCA não.

Aplicação da LDA em processamento de imagem

Em processamento de imagem, a LDA é aplicada principalmente para:

- **Classificação de imagens**: a LDA é usada para melhorar a classificação de imagens, identificando as características mais discriminantes com o objetivo de separar diferentes classes de objetos ou padrões em imagens. Isso é útil em tarefas como reconhecimento de objetos, detecção de faces, classificação de documentos, entre outras.

- **Redução de dimensionalidade supervisionada**: a LDA permite reduzir a dimensionalidade dos dados enquanto mantém as informações mais discriminantes. Isso é útil para acelerar algoritmos de processamento, economizar espaço de armazenamento e melhorar a eficácia da classificação.

- **Reconhecimento de padrões**: a LDA ajuda a encontrar as características mais importantes que distinguem diferentes padrões ou classes de objetos em imagens. Isso é crucial em tarefas de reconhecimento de padrões, quando a identificação de padrões distintos é essencial.

Como a LDA funciona em processamento de imagem

- **Coleta de dados de treinamento**: um conjunto de imagens de treinamento é coletado, cada uma delas é rotulada com sua classe correspondente.

- **Extração de características**: características relevantes são extraídas das imagens, como intensidades de pixel, texturas, gradientes, entre outras.

- **Padronização**: os dados de características são geralmente padronizados (subtraindo a média e dividindo pelo desvio padrão) para garantir que todas as características tenham uma escala comparável.

- **Cálculo das médias e matriz de dispersão**: a média das características é calculada para cada classe, bem como uma matriz de dispersão na classe e uma matriz de dispersão entre as classes.

- **Cálculo dos autovetores e autovalores**: a LDA calcula os autovetores e autovalores da matriz resultante da multiplicação da matriz de dispersão entre classes pela matriz de dispersão nas classes.

- **Seleção das componentes principais**: as componentes principais são selecionadas com base na magnitude dos autovalores. Normalmente, elas são ordenadas em ordem decrescente de importância discriminante.

- **Projeção dos dados**: os dados de características são projetados nas principais componentes selecionadas. Isso reduz a dimensionalidade dos dados, criando uma representação das imagens em um espaço de dimensão reduzida, que é otimizado para a discriminação entre classes.

A LDA é uma técnica poderosa em processamento de imagem, especialmente quando se trata de classificação de imagens e reconhecimento de padrões. Ela ajuda a encontrar as características mais discriminantes para melhorar a precisão da classificação e a eficácia da análise de imagem.

7 RECONHECIMENTO DE PADRÕES

O reconhecimento de padrões em processamento de imagem é uma área fundamental que se concentra na identificação de estruturas, objetos ou características específicas em imagens digitais. Essas estruturas podem incluir objetos, rostos, texto, gestos, ou qualquer padrão visual que seja relevante para uma determinada aplicação. Ele é aplicado em diversas áreas, incluindo visão computacional, aprendizado de máquina, processamento de imagens médicas e muitos outros domínios.

O que é reconhecimento de padrões em processamento de imagem?

O reconhecimento de padrões é a tarefa de identificar automaticamente padrões, objetos ou informações em imagens digitais com base em características específicas extraídas delas. Isso envolve a análise das características de interesse nas imagens e a comparação dessas características com modelos ou padrões predefinidos para tomar decisões sobre a presença ou ausência de um padrão ou objeto específico.

Aplicações do reconhecimento de padrões em processamento de imagem

O reconhecimento de padrões em processamento de imagem tem ampla gama de aplicações, incluindo:

- **Reconhecimento de objetos**: identificação automática de objetos em imagens, como carros em cenas de tráfego, produtos em linhas de produção, ou objetos em aplicações de visão robótica.
- **Reconhecimento de rostos**: detecção e reconhecimento de rostos humanos em imagens e vídeos, com aplicações em autenticação biométrica, vigilância e entretenimento.
- **Reconhecimento de texto**: identificação de texto em imagens, como em documentos digitalizados, imagens de placas de carro, ou em aplicações de OCR (reconhecimento óptico de caracteres).
- **Classificação de imagens**: categorização automática de imagens em classes ou categorias específicas, como classificar imagens de animais em diferentes espécies.
- **Reconhecimento de gestos**: identificação de gestos e movimentos de mãos ou corpo em aplicações de interação humano-computador e realidade virtual.
- **Diagnóstico médico**: auxílio na análise de imagens médicas, como radiografias, ressonâncias magnéticas e tomografias computadorizadas para identificar doenças e anormalidades.
- **Detecção de anomalias**: identificação de padrões anômalos ou suspeitos em imagens para segurança, controle de qualidade e prevenção de falhas.

Abordagens comuns em reconhecimento de padrões

- **Extração de características**: a primeira etapa na maioria dos sistemas de reconhecimento de padrões é a extração de características relevantes das imagens, como texturas, bordas, cores, formas, entre outras.

- **Aprendizado de máquina**: técnicas de aprendizado de máquina, como redes neurais, máquinas de vetores de suporte (SVM) e árvores de decisão, são frequentemente usadas para treinar modelos que podem reconhecer padrões com base nas características extraídas.

- **Redes neurais convolucionais (CNNs)**: as CNNs são especialmente eficazes em tarefas de reconhecimento de padrões em imagens. Elas são projetadas para capturar automaticamente características hierárquicas em imagens.

- **Detecção de objetos**: algoritmos de detecção de objetos, como YOLO e Faster R-CNN, são usados para identificar objetos em imagens e vídeos.

- **Processamento de linguagem natural (NLP)**: em algumas aplicações, como OCR, o reconhecimento de padrões também pode envolver o processamento de texto para extrair informações de documentos.

O reconhecimento de padrões é uma disciplina interdisciplinar que combina elementos de processamento de imagem, aprendizado de máquina, estatísticas e visão computacional. Avanços recentes em deep learning e algoritmos de aprendizado de máquina têm impulsionado significativamente a eficácia do reconhecimento de padrões em ampla gama de aplicações. Trata-se de uma área em constante evolução que desempenha um papel vital em tecnologias como veículos autônomos, assistentes virtuais, diagnóstico médico automatizado, entre outras.

7.1 Classificador de K-vizinhos mais próximos

O classificador de k-vizinhos mais próximos (k-Nearest Neighbors, k-NN) é um algoritmo de aprendizado de máquina supervisionado que pode ser aplicado com sucesso a tarefas de processamento de imagem, incluindo reconhecimento de padrões, classificação de objetos e segmentação de imagens. Ele baseia-se no princípio de que objetos semelhantes estão próximos uns dos outros no espaço de características.

Como funciona o k-NN em processamento de imagem

- **Coleta de dados de treinamento**: um conjunto de imagens de treinamento é coletado e cada imagem é associada a uma classe (rótulo) correspondente. Essas classes podem representar diferentes objetos, categorias ou padrões.

- **Extração de características**: a partir das imagens de treinamento, características relevantes são extraídas. No processamento de imagem, isso pode incluir texturas, histogramas de cores, descritores de formas, gradientes, entre outros.

- **Padronização dos dados**: as características são frequentemente padronizadas (subtraindo a média e dividindo pelo desvio padrão) para garantir que todas tenham uma escala comparável.

- **Treinamento do modelo**: os dados de treinamento são usados para treinar o modelo k-NN. O modelo armazena as características das imagens de treinamento e suas classes correspondentes.

- **Classificação de imagens de teste**: para uma nova imagem que precisa ser classificada, as características também são extraídas. O algoritmo k-NN calcula a distância entre as características da imagem de teste e as das imagens de treinamento. Ele seleciona os k exemplos de treinamento mais próximos (os vizinhos) com base nas menores distâncias.

- **Votação de maioria**: o algoritmo determina a classe mais frequente entre os k vizinhos mais próximos e a atribui à imagem de teste. Por exemplo, se a maioria dos vizinhos pertencer à classe "cachorro", a imagem de teste será classificada como um "cachorro".

Consta a seguir exemplo de código Python que usa a biblioteca scikit-learn para implementação de um classificador k-NN para reconhecimento de dígitos escritos à mão usando o conjunto de dados MNIST:

```
1 import numpy as np
2 from sklearn.datasets import fetch_openml
3 from sklearn.model_selection import train_test_split
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score

6 # Carregando o conjunto de dados MNIST
7 mnist = fetch_openml('mnist_784', version=1)
8 X = mnist.data.astype('float32')
9 y = mnist.target.astype('int')

10 # Dividindo os dados em conjuntos de treinamento e teste
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

12 # Criando e treinando o classificador k-NN
13 knn = KNeighborsClassifier(n_neighbors=3)
14 knn.fit(X_train, y_train)

15# Classificando as imagens de teste
16 y_pred = knn.predict(X_test)

17 # Calculando a precisão do modelo
18 accuracy = accuracy_score(y_test, y_pred)
19 print(f'A precisão do k-NN é: {accuracy * 100:.2f}%')
```

Saída:

A precisão do k-NN é: 97.13%

Saiba mais

A fim de compreender melhor sobre k-NN, recomendamos a leitura do capítulo 11.6 – Regra dos k-vizinhos mais próximos na obra a seguir:

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais*: princípios, algoritmos e aplicações. São Paulo: Cengage Learning Brasil, 2007.

Neste exemplo, o k-NN é treinado no conjunto de dados MNIST, que consiste em imagens de dígitos manuscritos de 0 a 9. O código carrega o conjunto de dados, divide-o em conjuntos de treinamento e teste, cria um modelo k-NN com k=3 vizinhos e avalia a precisão do modelo.

O valor de k deve ser ajustado com base na natureza dos dados e na tarefa de classificação específica. Um valor adequado de k é crucial para obter um desempenho ideal do k-NN em uma aplicação de processamento de imagem.

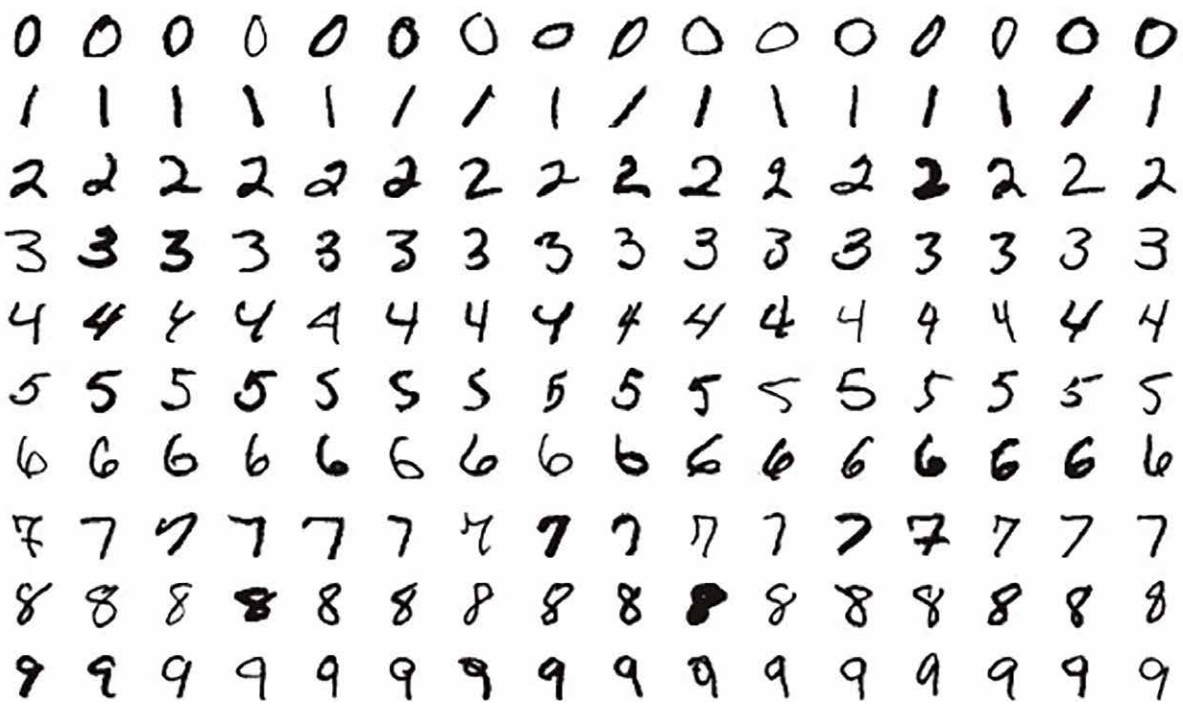

Figura 38 – Um exemplo de conjunto de dados de números MNIST

Disponível em: https://tinyurl.com/ykwrheja. Acesso em: 24 out. 2023.

7.2 Método de classificação em cascata (Haar cascade)

O método de classificação em cascata (Haar cascade) é um algoritmo de detecção de objetos utilizado em processamento de imagem e visão computacional. Foi originalmente proposto por Viola e Jones (2001) e é conhecido por sua eficiência na detecção de objetos em tempo real, como rostos, olhos, carros, placas de veículos e muitos outros. Ele é baseado na técnica de classificação em cascata, que envolve a aplicação sequencial de classificadores para filtrar regiões de interesse em uma imagem.

Como funciona o método de classificação em cascata (Haar cascade)

O Haar cascade é baseado em características de Haar, que simples, retangulares e de valores positivos e negativos. Essas características são semelhantes a filtros e podem ser usadas para representar diferentes padrões em uma imagem. O processo de detecção de objetos usando Haar cascade pode ser resumido em quatro etapas principais:

1. **Treinamento do classificador em cascata**: a primeira etapa envolve o treinamento do classificador em cascata usando um grande conjunto de dados de treinamento que contém exemplos positivos (imagens que contêm o objeto de interesse) e exemplos negativos (imagens que não contêm o objeto de interesse). O classificador é treinado para distinguir entre as duas classes com base nas características de Haar.

2. **Criação de um classificador em cascata**: o classificador em cascata é criado combinando vários classificadores fracos (geralmente classificadores de árvore de decisão simples) em uma cascata. Cada classificador na cascata é treinado para ser rápido e descartar rapidamente as regiões que não contêm o objeto de interesse.

3. **Detecção em cascata**: a imagem de entrada é dividida em várias janelas sobrepostas de diferentes tamanhos. Em cada janela, o classificador em cascata é aplicado sequencialmente. Se uma janela falhar em qualquer estágio da cascata, ela é descartada. Somente se passar por todos os estágios a janela será considerada uma detecção positiva.

4. **Pós-processamento**: as detecções positivas são refinadas e agrupadas para formar regiões de objetos. Isso pode envolver a eliminação de sobreposições, o ajuste de retângulos de detecção e a rejeição de detecções fracas.

Consta a seguir exemplo de código Python usando a biblioteca OpenCV para a detecção de rostos usando um modelo de Haar cascade pré-treinado:

```
1 import cv2
2 import matplotlib.pyplot as plt

3 # Carregar o classificador Haar Cascade pré-treinado para detecção de rostos
4 face_cascade = cv2.CascadeClassifier('/haarcascade_frontalface_default.xml')
```

```
5 # Carregar uma imagem de entrada
6 imagem = cv2.imread('sua_imagem.jpg')
7 img = cv2.imread('sua_imagem.jpg')

8 # Converter a imagem para escala de cinza
9 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

10 # Realizar a detecção de rostos na imagem
11 faces=face_cascade.detectMultiScale(gray,scaleFactor=1.3,minNeighbors=5,minSize=(30,30))

12 # Desenhar retângulos ao redor dos rostos detectados

13 for (x, y, w, h) in faces:
14 cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

15 # Exibir a imagem com as detecções
16 plt.rcParams.update({'font.size': 20})
17 plt.figure(figsize=(15, 20))

18 plt.subplot(2, 2, 1)
19 plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
20 plt.title('Imagem Original')
21 plt.axis('off')

22 plt.subplot(2, 2, 2)
23 plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
24 plt.title('Detecção de Rostos')
25 plt.axis('off')

26 plt.show()
```

Saída:

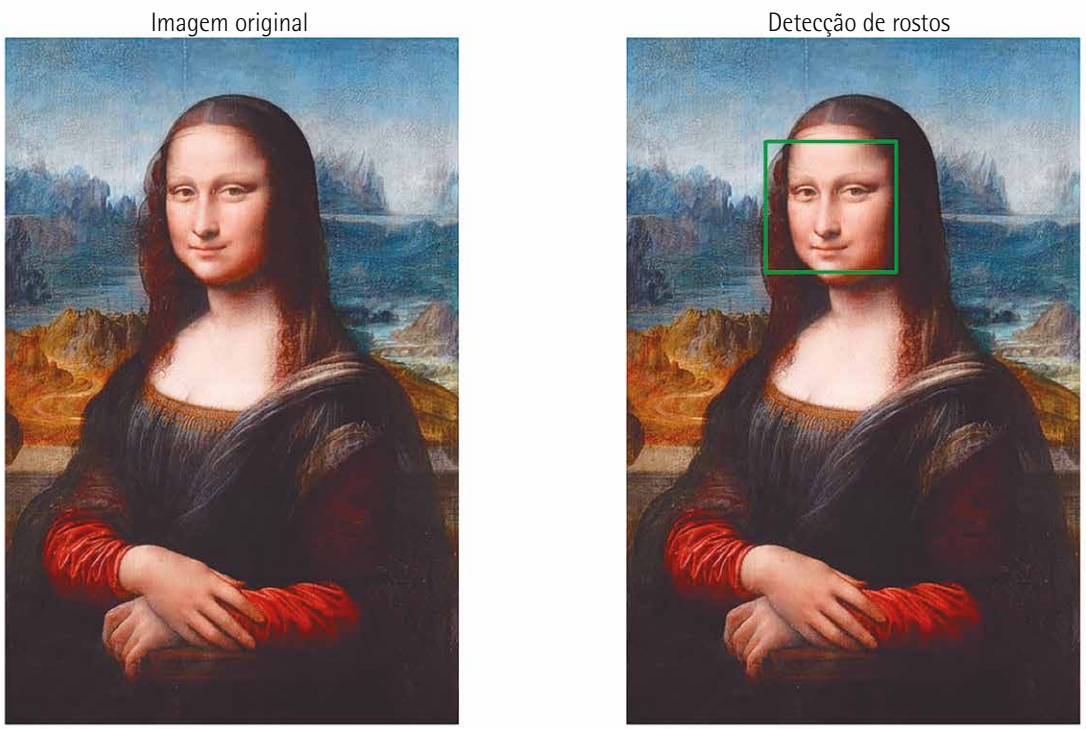

Figura 39 – Leonardo da Vinci: Mona Lisa

Disponível em: https://tinyurl.com/s293x43e. Acesso em: 24 out. 2023.

Neste exemplo, usamos o classificador Haar cascade pré-treinado para detecção de rostos da OpenCV. A imagem de entrada é carregada, convertida para escala de cinza e, em seguida, a detecção de rostos é realizada. Os retângulos são desenhados ao redor dos rostos detectados na imagem.

Saiba mais

Certifique-se de ter o arquivo haarcascade_frontalface_default.xml no mesmo diretório do código ou forneça o caminho correto para o arquivo XML do classificador Haar cascade. É possível fazer o download dele no link a seguir:

LIENHART, R. *Intel License Agreement For Open Source Computer Vision Library*. GitHub, 2000. Disponível em: https://tinyurl.com/2dnjctfe. Acesso em: 24 out. 2023.

O Haar cascade é uma técnica eficiente para a detecção de objetos em imagens e vídeos. É utilizado em aplicações de visão computacional e tem um desempenho notável em tempo real. Para aplicações específicas, é possível treinar seus próprios classificadores Haar cascade usando dados de treinamento personalizados.

7.3 Estado da arte

Certamente o campo do reconhecimento de padrões tem uma longa história, e houve muitos trabalhos importantes ao longo dos anos que contribuíram para o seu desenvolvimento. Alguns dos marcos mais significativos em ordem cronológica são:

- **"Perceptron" de Rosenblatt (1958)**: Frank Rosenblatt introduziu o Perceptron, um dos primeiros modelos de aprendizado de máquina que poderia ser treinado para reconhecer padrões. Embora suas limitações tenham sido posteriormente reveladas, ele foi um marco inicial no campo.

- **"Pattern classification and scene analysis" de Duda e Hart (1974)**: Este livro seminal de Richard O. Duda e Peter E. Hart é um dos primeiros textos abrangentes sobre reconhecimento de padrões, definindo muitos conceitos fundamentais.

- **"Vision" de Marr (1982)**: David Marr é conhecido por seu trabalho pioneiro na visão computacional, que ajudou a estabelecer as bases teóricas para a compreensão da visão por computador e o reconhecimento de padrões em imagens.

- **"Back-propagation" de Rumelhart, Hinton e Williams (1986)**: o algoritmo de retropropagação (back-propagation) para treinamento de redes neurais artificiais foi revolucionário para o reconhecimento de padrões, tornando possível treinar redes profundas.

- **"Face Detection" de Viola e Jones (2001)**: Paul Viola e Michael Jones desenvolveram um dos primeiros sistemas eficazes de detecção de rosto em tempo real, uma aplicação importante do reconhecimento de padrões em visão computacional.

- **ImageNet e AlexNet (2012)**: a competição ImageNet, que começou em 2010, estimulou o avanço do reconhecimento de imagens em larga escala. Em 2012, a rede AlexNet, projetada por Alex Krizhevsky, venceu a competição, demonstrando a eficácia das redes neurais profundas.

- **VGGNet, GoogLeNet, ResNet (2014-2015)**: estas arquiteturas de redes neurais profundas expandiram o campo de visão computacional e estabeleceram novos padrões de desempenho em tarefas de classificação de imagens.

- **Inception-v4 e ResNet-152 (2016)**: estas arquiteturas aprimoradas demonstraram o poder das redes profundas e eficientes na classificação de imagens em grandes conjuntos de dados.

- **AlphaGo (2016)**: a equipe do Google DeepMind desenvolveu o AlphaGo, um sistema de IA que derrotou o campeão mundial de Go. Isso demonstrou a capacidade de máquinas de aprender e reconhecer padrões em jogos complexos.

- **Transformers e GPT-3 (2018-2020)**: a arquitetura Transformer revolucionou o processamento de linguagem natural e a visão computacional. O GPT-3, desenvolvido pela OpenAI, é um exemplo notável que demonstrou habilidades impressionantes em reconhecimento de padrões em texto.

Esses são apenas alguns dos muitos trabalhos importantes ao longo da história do reconhecimento de padrões. À medida que a pesquisa continua e novas tecnologias emergem, o campo continua a evoluir rapidamente. É importante observar que esses avanços muitas vezes resultam de uma colaboração interdisciplinar entre matemáticos, cientistas da computação, engenheiros e especialistas em domínios específicos, e a evolução continua:

- **Redes neurais convolucionais (CNNs)**: as CNNs continuam sendo a espinha dorsal do reconhecimento de padrões em imagens. Arquiteturas como ResNet, Inception, e EfficientNet demonstraram desempenho excepcional em várias tarefas de visão computacional, incluindo classificação de imagens, detecção de objetos e segmentação semântica.

- **Transfer learning**: a transferência do conhecimento de modelos pré-treinados para novas tarefas provou ser uma técnica eficaz. Modelos pré-treinados em grandes conjuntos de dados, como o ImageNet, podem ser ajustados para tarefas específicas com conjuntos de dados menores, economizando tempo e recursos.

- **Transformers:** os modelos Transformers, originalmente desenvolvidos para processamento de linguagem natural, estão sendo cada vez mais aplicados ao processamento de imagem. A arquitetura ViT (Vision Transformer) é um exemplo notável, que mostra resultados competitivos em tarefas de classificação de imagens.

- **Aprendizado profundo multimodal**: integração de informações de várias modalidades, como texto, áudio e vídeo, em sistemas de reconhecimento de padrões para tarefas mais complexas, como compreensão de vídeos e interpretação de dados multimodal.

- **Aprendizado por reforço**: o aprendizado por reforço é usado em aplicações de reconhecimento de padrões, como robótica e jogos, para treinar agentes a tomar decisões sequenciais.

- **Detecção de anomalias**: técnicas avançadas de detecção de anomalias, como autoencoders variacionais e modelos generativos adversariais (GANs), são usadas para identificar padrões incomuns em dados, o que é fundamental em segurança cibernética e detecção de fraudes.

- **Aprendizado não supervisionado**: métodos de aprendizado não supervisionado, como clusterização e redução de dimensionalidade, são utilizados para encontrar padrões em dados sem a necessidade de rótulos explícitos.

- **Reconhecimento de voz e processamento de linguagem natural**: multimodal em termos de reconhecimento de padrões em áudio e linguagem natural, os modelos de linguagem pré-treinados, como GPT-3, têm demonstrado uma compreensão impressionante da linguagem humana e sido aplicados em muitas tarefas.

- **Ética e interpretabilidade**: com o aumento do uso de IA em sistemas críticos, a ética e a interpretabilidade dos modelos de reconhecimento de padrões estão se tornando áreas de foco. A compreensão do funcionamento interno dos modelos e a mitigação de vieses são preocupações crescentes.

- **Aplicações práticas**: o reconhecimento de padrões é usado em uma variedade de campos, incluindo medicina (diagnóstico médico), automotivo (veículos autônomos), manufatura (controle de qualidade), segurança (reconhecimento facial) e muito mais.

O estado da arte no reconhecimento de padrões continua a se expandir à medida que novas técnicas e aplicações surgem. A colaboração entre a pesquisa acadêmica e a indústria é fundamental para impulsionar esses avanços e tornar a tecnologia mais acessível e eficaz em diversas áreas. Recomendamos sempre verificar as últimas pesquisas e avanços em conferências e publicações relevantes para obter as informações mais atualizadas sobre o campo.

7.4 Abordagens baseadas em aprendizado profundo

Abordagens baseadas em aprendizado profundo (deep learning) revolucionaram o campo do reconhecimento de padrões nas últimas décadas. Essas técnicas têm se destacado em várias tarefas de visão computacional, processamento de linguagem natural e muito mais.

As CNNs são uma classe fundamental de redes neurais profundas que têm sido especialmente eficazes em tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação semântica. Elas são projetadas para lidar com dados de grade, como imagens, e empregam camadas de convolução para extrair características hierárquicas. O trabalho seminal da equipe do ImageNet em 2012, liderado por Alex Krizhevsky, introduziu redes neurais convolucionais (CNNs) profundas e demonstrou resultados impressionantes na tarefa de classificação de objetos. Essa conquista desencadeou o uso generalizado de CNNs em tarefas de visão computacional.

Redes neurais, como a Faster R-CNN e a YOLO (You Only Look Once), são usadas para detecção de objetos em imagens, sendo aplicadas em sistemas de vigilância, veículos autônomos e muito mais. Redes neurais profundas, como as redes de segmentação semântica, permitem a identificação precisa de áreas de interesse em imagens, sendo úteis em áreas como medicina, geologia e análise de satélite.

Redes generativas adversariais (GANs) são usadas para gerar imagens realistas. O trabalho do BigGAN e do DALL-E, da OpenAI, são exemplos notáveis de geração de imagens usando GANs. O aprendizado profundo é usado na classificação de imagens médicas, auxiliando no diagnóstico de doenças a partir de imagens de raios-X, ressonâncias magnéticas e tomografias computadorizadas.

O reconhecimento de expressões faciais é uma aplicação popular que utiliza CNNs para identificar emoções em imagens e vídeos. O processamento de imagens em astronomia é aprimorado pelo uso de aprendizado profundo para classificar estrelas, planetas e outros objetos celestes.

Aplicações de realidade aumentada e virtual usam aprendizado profundo para rastreamento de marcadores, detecção de planos, detecção de objetos e muito mais. O processamento de imagens desempenha um papel importante na percepção de veículos autônomos, usando CNNs para identificar pedestres, carros e obstáculos.

A pesquisa em aprendizado profundo está em constante evolução. Novas arquiteturas, algoritmos e técnicas estão sendo desenvolvidos para lidar com desafios emergentes e melhorar o desempenho em tarefas complexas de reconhecimento de padrões. As abordagens baseadas no tema revolucionaram o campo do reconhecimento de padrões, permitindo uma precisão notável em tarefas diversas. À medida que a pesquisa continua avançando, é provável que essas técnicas desempenhem um papel cada vez mais importante em diversas aplicações práticas.

7.5 Redes neurais artificiais convolucionais (CNNs)

Redes neurais artificiais convolucionais (convolutional neural networks – CNNs) são um avanço fundamental no campo do reconhecimento de padrões, especialmente em tarefas relacionadas ao processamento de imagens e visão computacional. Essas redes foram projetadas especificamente para lidar com dados de grade, como imagens, e têm sido utilizadas em uma variedade de aplicações. Consta a seguir a explicação mais detalhada sobre as CNNs e seu papel no reconhecimento de padrões:

- **Estrutura de camadas**: as CNNs consistem em camadas empilhadas, incluindo camadas de convolução, camadas de pooling e camadas totalmente conectadas. As camadas de convolução são o elemento central das CNNs, sendo responsáveis por extrair características das imagens.

- **Camadas de convolução**: essas camadas aplicam filtros (também chamados de kernels) a pequenas regiões da imagem. Esses filtros detectam padrões simples, como bordas, texturas e formas. À medida que você avança nas camadas, os filtros se tornam mais abstratos e capturam características mais complexas.

- **Camadas de pooling**: reduzem a dimensionalidade da imagem, preservando as características mais importantes. Elas fazem isso agrupando informações de regiões próximas e mantendo apenas a informação mais significativa. Isso ajuda a reduzir a quantidade de parâmetros e a evitar o overfitting.

- **Camadas totalmente conectadas**: após as camadas de convolução e pooling, as CNNs geralmente incluem camadas totalmente conectadas para realizar a classificação ou outra tarefa específica. Essas camadas são semelhantes àquelas de uma rede neural tradicional.

- **Aprendizado hierárquico de características**: uma das principais vantagens das CNNs é sua capacidade de aprender características hierárquicas automaticamente. As camadas iniciais capturam detalhes de baixo nível, enquanto as camadas posteriores combinam essas características para formar representações mais abstratas.

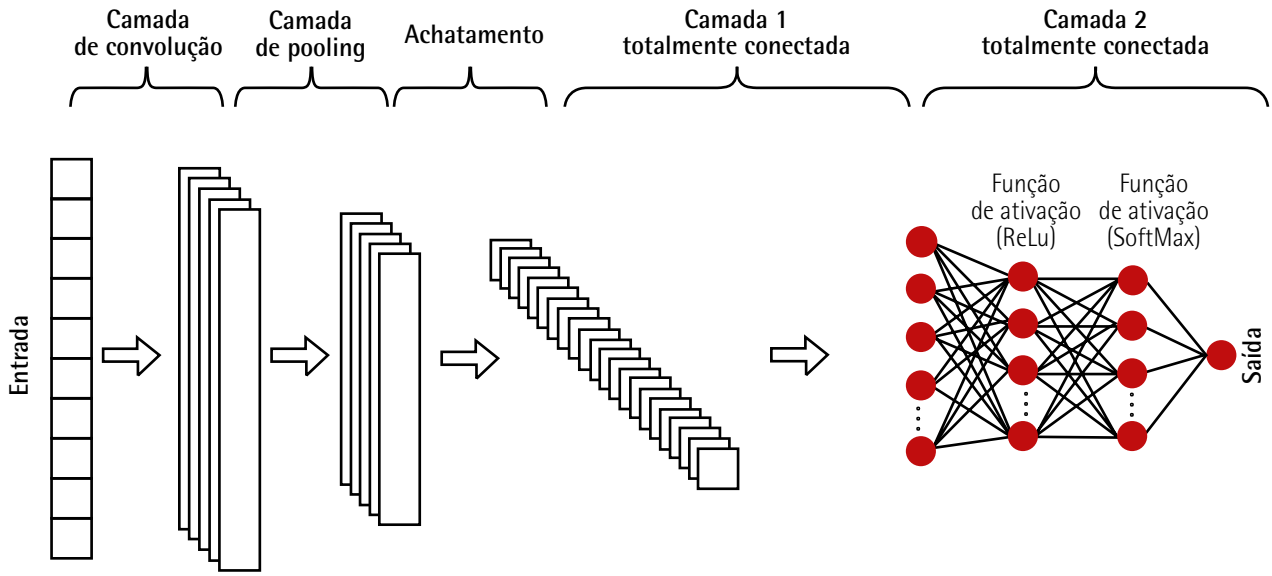

Figura 40 – Redes neurais artificiais convolucionais

- **Transfer learning**: CNNs pré-treinadas em grandes conjuntos de dados, como o ImageNet, provaram ser valiosas. Os pesos aprendidos por esses modelos pré-treinados podem ser usados como ponto de partida para tarefas específicas com conjuntos de dados menores, economizando tempo de treinamento e recursos computacionais.

- **Desafios**: embora as CNNs tenham obtido muito sucesso, elas ainda enfrentam desafios, como a necessidade de grandes conjuntos de dados rotulados e o ajuste fino dos hiperparâmetros. Além disso, podem não ser ideais para tarefas que envolvem dados sequenciais ou estruturados, para os quais outras arquiteturas, como as redes recorrentes, podem ser mais adequadas.

- **Pesquisa contínua**: a pesquisa em CNNs continua avançando, com o desenvolvimento de arquiteturas mais eficientes e modelos mais profundos. Também estão em andamento esforços para tornar as CNNs mais interpretables e robustas.

As redes neurais artificiais convolucionais são essenciais no reconhecimento de padrões em imagens e têm se mostrado altamente eficazes em uma variedade de aplicações. Sua capacidade de aprender hierarquicamente características complexas a partir de dados de grade as torna uma ferramenta poderosa para lidar com desafios de visão computacional e processamento de imagens.

Observação

Uma rede interpretable, ou modelo interpretable, refere-se a um tipo de modelo de aprendizado de máquina ou rede neural cujas decisões e previsões podem ser facilmente compreendidas, explicadas e interpretadas por seres humanos. A interpretabilidade é uma característica crítica, especialmente em domínios nos quais a compreensão do processo de tomada de decisão é importante.

O modelo interpretable é transparente, ou seja, suas operações e decisões podem ser rastreadas e compreendidas passo a passo. Isso geralmente envolve a capacidade de explicar como o modelo chegou a uma determinada previsão. Geralmente são mais simples do que redes profundas ou modelos complexos. Eles tendem a ter menos camadas e parâmetros, o que torna mais fácil entender como funcionam.

Os modelos interpretables geralmente fornecem explicações para suas decisões. Isso pode incluir a identificação das características mais importantes que influenciaram a previsão ou a descrição do raciocínio lógico usado para chegar a uma decisão. Eles permitem a interpretação das características de entrada e como elas afetam a saída do modelo. Isso é particularmente importante para entender por que o modelo fez uma determinada previsão. Frequentemente são acompanhados por visualizações que auxiliam na compreensão das operações e decisões do modelo.

A interpretabilidade é crucial em aplicações nas quais a responsabilidade, a ética e a confiabilidade são preocupações importantes. Por exemplo, em diagnósticos médicos, é crucial entender por que um modelo de aprendizado de máquina recomenda um tratamento específico. Em sistemas de justiça criminal, é necessário entender as razões por trás das decisões de um modelo de pontuação de risco.

Modelos interpretables podem incluir árvores de decisão, regressão linear, regressão logística, modelos lineares generalizados, entre outros. Além disso, há um crescente interesse em tornar modelos de aprendizado profundo mais interpretables, desenvolvendo técnicas que permitam explicar as decisões tomadas por redes neurais profundas.

7.6 Operação de convolução

A operação de convolução é um conceito fundamental no reconhecimento de padrões. Ela desempenha um papel central em CNNs e é utilizada para extrair características relevantes dos dados.

O que é a operação de convolução?

A convolução é uma operação matemática que combina duas funções para criar uma terceira. No contexto do reconhecimento de padrões e processamento de imagens, a convolução envolve uma matriz chamada "filtro" ou "kernel" que desliza sobre uma matriz de dados de entrada (por exemplo, uma imagem), multiplicando e somando elementos correspondentes.

Aplicação da convolução em imagens

- **Filtro (kernel)**: é uma matriz menor que atua como detector de características. Por exemplo, ele pode ser projetado para detectar bordas horizontais em uma imagem.

- **Imagem de entrada**: é representada como uma matriz de pixels. O filtro é aplicado a pequenas regiões da imagem por vez, começando na parte superior esquerda e movendo-se gradualmente para a direita e para baixo.

- **Operação de multiplicação e soma**: em cada etapa, o filtro é sobreposto à região da imagem, e os elementos correspondentes são multiplicados. Os produtos são então somados para produzir um único valor.

- **Mapa de características (feature map)**: o resultado da convolução é chamado de "mapa de características". Ele destaca a presença de características específicas, como bordas, texturas ou padrões, na imagem.

Aprendizado de características hierárquicas

Uma das vantagens da operação de convolução é sua capacidade de aprender características hierárquicas automaticamente. Isso ocorre porque os filtros em camadas mais profundas de uma CNN podem combinar informações de camadas mais rasas para identificar características mais complexas. Por exemplo, filtros nas primeiras camadas podem aprender a detectar bordas, enquanto filtros nas camadas posteriores podem aprender a detectar formas mais complexas que consistem em bordas.

Redução da dimensionalidade

Além de extrair características, a convolução ajuda na redução da dimensionalidade dos dados. Isso é feito usando camadas de pooling após as camadas de convolução. As camadas de pooling resumem informações em regiões maiores, preservando as características mais importantes e reduzindo a quantidade de dados processados nas camadas seguintes.

Aplicações no reconhecimento de padrões

A operação de convolução é usada em tarefas de reconhecimento de padrões que envolvem imagens, como classificação de imagens, detecção de objetos, segmentação semântica e muito mais. Os mapas de características resultantes da convolução contêm informações valiosas para identificar padrões em imagens.

Personalização de filtros

Um dos benefícios das CNNs é que os filtros são aprendidos durante o treinamento da rede. Isso significa que a rede neural pode adaptar seus filtros para serem específicos à tarefa em questão, resultando em melhor desempenho. A operação de convolução permite a extração automática de características relevantes de dados de grade, como imagens.

7.7 Pooling

A operação de pooling é uma etapa essencial no reconhecimento de padrões, especialmente em CNNs. Ela desempenha redução da dimensionalidade e na extração de características discriminantes dos dados. Contam a seguir os principais pontos sobre pooling no contexto do reconhecimento de padrões:

O que é pooling?

Pooling, também conhecido como subamostragem, é uma operação utilizada para reduzir a dimensionalidade dos dados, tornando-os mais gerenciáveis e eficientes computacionalmente. Em particular, é comumente aplicado após as camadas de convolução em CNNs.

Tipos de pooling

Os dois tipos mais comuns pooling são:

- **Max pooling**: esta operação seleciona o valor máximo de um grupo de valores em uma região específica da matriz de entrada. Ele é frequentemente usado para preservar características dominantes em uma região da imagem.

- **Average pooling**: em vez de selecionar o valor máximo, o average pooling calcula a média dos valores em uma região da matriz de entrada. Isso pode ser útil quando se deseja a representação mais suavizada dos dados.

Dando sequência às informações sobre pooling, temos:

- **Redução da dimensionalidade:** uma das principais funções do pooling é reduzir a dimensionalidade dos dados. Por exemplo, após várias camadas de convolução, a matriz resultante pode ser muito grande. Aplicando pooling, é possível reduzir significativamente o número de elementos na matriz, o que economiza recursos computacionais e evita o overfitting.

- **Invariância a translação**: o pooling oferece uma certa invariância à translação. Isso significa que, mesmo que um objeto em uma imagem seja deslocado, o pooling pode ajudar a preservar as informações relevantes. Essa invariância é especialmente útil em tarefas como detecção de objetos, quando a posição exata do objeto na imagem pode variar.

- **Localização aproximada de características**: enquanto a convolução extrai características importantes das imagens, o pooling fornece uma representação mais geral e compacta dessas características. Ele permite que a rede neural se concentre nas características mais discriminantes e ignore detalhes menores.

- **Tamanho do pooling**: o tamanho da janela de pooling (geralmente representado por uma matriz quadrada, como 2 × 2 ou 3 × 3) é um hiperparâmetro ajustável. Um tamanho maior de pooling resultará em maior redução da dimensionalidade, mas poderá levar à perda de informações mais finas. O tamanho adequado depende da tarefa e dos dados específicos.

- **Utilização em CNNs**: em CNNs, geralmente, após várias camadas de convolução, é comum aplicar uma camada de pooling para reduzir a resolução da imagem e as informações desnecessárias. Isso é frequentemente seguido por camadas totalmente conectadas para tarefas de classificação ou detecção de padrões.

- **Limitações**: embora o pooling seja uma técnica valiosa, ele também tem limitações. Em alguns casos, informações detalhadas podem ser perdidas durante a redução da dimensionalidade, o que pode ser indesejável em tarefas sensíveis a detalhes finos.

O pooling é uma operação importante no reconhecimento de padrões, especialmente em CNNs. Ele ajuda a reduzir a dimensionalidade dos dados, tornando-os mais eficientes e contribui para a extração de características relevantes. A escolha entre max pooling e average pooling, bem como o tamanho da janela de pooling, depende das características dos dados e da tarefa específica em mãos.

7.8 Flatten layer

A flatten layer, ou camada de achatamento, é um componente fundamental em redes neurais artificiais, especialmente em CNNs, quando se trata de reconhecimento de padrões. Esta camada desempenha um papel importante na transição entre as camadas de convolução e aquelas totalmente conectadas em uma CNN.

- **O que é a flatten layer?**: a flatten layer é uma camada simples que tem a tarefa de transformar os dados tridimensionais em um vetor unidimensional. Em outras palavras, ela "achata" as representações de dados que vêm de camadas anteriores, como camadas de convolução e pooling, e converte-as em um formato adequado para a entrada das camadas totalmente conectadas, também conhecidas como camadas densas.

- **Necessidade da flatten layer**: antes de entrar nas camadas totalmente conectadas, as CNNs geralmente passam por camadas de convolução e pooling que extraem características das imagens em um formato de matriz tridimensional (altura x largura x canais). No entanto, as camadas totalmente conectadas requerem entradas unidimensionais, cujo cada valor é considerado como um recurso independente. Portanto, a flatten layer é usada para fazer essa transição suave de representações 3D para vetores 1D.

- **Como funciona**: a flatten layer não realiza cálculos complexos. Ela simplesmente reorganiza os valores na matriz 3D em um único vetor, preservando a ordem dos valores. Por exemplo, se a saída de uma camada de convolução for uma matriz de tamanho 4 × 4 × 64, a flatten layer a transformará em um vetor de 1024 elementos (4 × 4 × 64 = 1024).

- **Papel nas camadas totalmente conectadas**: após a flatten layer, o vetor resultante é alimentado nas camadas totalmente conectadas da rede neural. Essas camadas podem conter neurônios que estão conectados a todos os elementos do vetor, permitindo que a rede aprenda relações complexas entre as características extraídas nas camadas de convolução e pooling.

- **Limitações**: a flatten layer não introduz não linearidades ou parâmetros treináveis na rede. Sua função é puramente transformar a forma dos dados. Portanto, a principal limitação é que ela não contribui diretamente para a extração de características complexas ou aprendizado de representações mais avançadas, que são tarefas realizadas pelas camadas de convolução e pooling.

- **Integração com outras camadas**: a flatten layer é frequentemente usada em combinação com outras camadas, como camadas de dropout para regularização ou camadas densas para classificação final. A combinação dessas camadas permite que as CNNs se adaptem e generalizem melhor a partir dos dados de treinamento.

Ela desempenha um papel crítico na arquitetura das CNNs, facilitando a transição das representações 3D extraídas nas camadas de convolução e pooling para vetores 1D que podem ser alimentados nas camadas totalmente conectadas. Essa camada é essencial para redes neurais usadas em reconhecimento de padrões em imagens e outras aplicações de visão computacional.

7.9 Funções de ativação

As funções de ativação são essenciais no reconhecimento de padrões, especialmente em redes neurais artificiais e deep learning. Elas são funções matemáticas aplicadas nas saídas das camadas de uma rede neural e responsáveis por introduzir não linearidades, tornando possível que a rede aprenda representações complexas e não lineares dos dados.

O que são funções de ativação?

São funções matemáticas que determinam se um neurônio artificial deve ser ativado (ou seja, transmitir um sinal adiante) ou não. Elas atuam sobre a saída ponderada de um neurônio e aplicam uma transformação não linear a esse valor. Sem funções de ativação, redes neurais seriam equivalentes a modelos lineares, incapazes de aprender relações complexas entre os dados.

Importância da não linearidade

As funções de ativação introduzem não linearidade nas redes neurais, permitindo que elas capturem e modelem padrões complexos em dados, o que é crucial para tarefas de reconhecimento de padrões. Sem funções de ativação, redes neurais seriam limitadas a representações lineares, incapazes de lidar com problemas mais sofisticados.

Funções de ativação comuns

- **Sigmoid**: a função sigmoid mapeia os valores de entrada para um intervalo entre 0 e 1. Foi muito utilizada em redes neurais mais antigas, mas tem caído em desuso em favor de funções mais eficazes.

$$f(x)=1/\left(1+e^{-x}\right)$$

- **Função de etapa binária:** a função de ativação é um classificador baseado em limiar (threshold), isto é, se o neurônio deve ou não ser ativado.

$$f(x)=1, x>=0$$
$$f(x)=0, x<0$$

- **Tangente hiperbólica (tanh)**: a função tangente hiperbólica mapeia os valores de entrada para um intervalo entre -1 e 1, tornando-a mais centrada na origem do que a sigmoid.

$$\tanh(x)=2/\left(1+e^{-2x}\right)-1$$

- **ReLU (rectified linear unit)**: a função ReLU retorna zero para valores negativos e o próprio valor de entrada para valores não negativos. Ela é usada devido à sua simplicidade e eficácia.

$$f(x)=\max(0,x)$$

- **Leaky ReLU**: o Leaky ReLU é uma variante da função ReLU que permite que uma pequena inclinação seja aplicada à parte negativa dos valores de entrada, evitando o problema do "neurônio morto" (dead neuron) que pode ocorrer com ReLU.

- **Função de ativação linear (linear activation)**: trata-se de uma função de ativação simples que retorna o próprio valor de entrada. É usada em camadas de saída de regressão.

$$f(x)=ax$$

- **Softmax**: a função é usada nas camadas de saída de redes neurais quando a tarefa tem classificação multiclasse. Ela converte as saídas da rede em uma distribuição de probabilidade sobre as classes.

Escolha da função de ativação

A escolha da função de ativação depende da tarefa e do tipo de rede neural. ReLU e suas variantes geralmente funcionam bem na maioria das situações devido à sua simplicidade e eficácia. No entanto, o experimento é muitas vezes necessário para determinar qual função de ativação funciona melhor em uma tarefa específica.

Problemas com funções de ativação

- **Desvanecimento do gradiente**: funções de ativação sigmoid e tanh podem levar ao problema do desvanecimento do gradiente, quando os gradientes durante o treinamento se tornam muito pequenos para atualizar os pesos da rede. Isso pode afetar o treinamento em redes profundas.

- **Neurônios mortos**: funções de ativação ReLU podem levar a "neurônios mortos" que não ativam para nenhum valor de entrada negativo. Isso pode acontecer quando um neurônio sempre retorna zero, impedindo que a rede aprenda.

Funções de ativação são elementos fundamentais em redes neurais utilizadas para reconhecimento de padrões. Elas introduzem não linearidade nas redes, permitindo a modelagem de relações complexas nos dados. A escolha da função de ativação depende da tarefa e do tipo de rede, e uma escolha apropriada pode ser fundamental para o sucesso do modelo.

8 VISÃO COMPUTACIONAL

A história da visão computacional é uma jornada fascinante que se estende por várias décadas e tem suas raízes em diversas disciplinas, desde a fotografia até a inteligência artificial. Consta a seguir um resumo das principais etapas na evolução da visão computacional ao longo do tempo:

- **Década de 1950 – Fundações iniciais**: a visão computacional tem suas raízes em experimentos fotográficos na década de 1950. Pesquisadores exploraram a ideia de digitalizar imagens e processá-las automaticamente em computadores. No entanto, o poder computacional limitado da época restringiu o progresso.

- **Década de 1960 – Primeiras tentativas**: nessa década, os pesquisadores começaram a desenvolver algoritmos para analisar imagens. Um dos primeiros projetos notáveis foi o The Summer Vision Project, que tentou identificar objetos em imagens.

- **Década de 1970 – Primeiras aplicações**: nessa década, a visão computacional começou a encontrar aplicações em áreas como processamento de imagens médicas, reconhecimento de padrões em imagens de satélite e automação industrial.

- **Década de 1980 – crescimento e avanços**: essa década testemunhou o crescimento da visão computacional, impulsionada por avanços em hardware e software. Técnicas de detecção de

bordas e transformações geométricas tornaram-se populares. A rede neural convolucional (CNN) foi proposta nesta década, embora ainda não tivesse alcançado seu potencial total.

- **Década de 1990 – Progressos contínuos**: nessa década houve progressos significativos em rastreamento de objetos, reconhecimento de padrões e análise de vídeo. Surgiram aplicações práticas em sistemas de vigilância e visão por computador móvel.

- **Década de 2000 – A Era das CNNs**: essa década foi um ponto de virada para a visão computacional com a ascensão das CNNs. A CNN LeNet foi uma das primeiras redes neurais profundas bem-sucedidas usadas para reconhecimento de dígitos manuscritos. As CNNs rapidamente se tornaram a espinha dorsal das aplicações de visão computacional.

- **Década de 2010 – Avanços em aprendizado profundo**: essa década testemunhou avanços impressionantes em aprendizado profundo, impulsionando ainda mais a visão computacional. A ImageNet Large Scale Visual Recognition Challenge, em 2012, representou um marco importante com a vitória da CNN AlexNet. A partir daí, as CNNs se tornaram líderes em tarefas de classificação de imagens, detecção de objetos e segmentação semântica.

- **Década de 2020 – Expansão das aplicações**: atualmente a visão computacional está em plena expansão. Ela é usada em várias aplicações, como veículos autônomos, reconhecimento facial, análise de imagens médicas, realidade aumentada, automação industrial e muito mais. Além disso, a pesquisa continua a explorar novos domínios, como visão computacional multimodal (combinação de imagens e texto), visão computacional 3D e interpretabilidade de modelos.

A visão computacional percorreu um longo caminho desde suas origens na digitalização de imagens até se tornar uma disciplina essencial na ciência da computação e na indústria. Ela promete continuar evoluindo e transformando a forma como interagimos com o mundo visualmente, à medida que a tecnologia avança e novos desafios e aplicações emergem.

8.1 Tecnologias e projetos de visão computacional

A visão computacional tem visto um rápido avanço tecnológico nos últimos anos, impulsionado pelo crescimento do aprendizado profundo e pelo aumento da capacidade de processamento de hardware.

As CNNs revolucionaram a visão computacional. Projetos como AlexNet, VGG, Inception (GoogLeNet), ResNet e EfficientNet introduziram arquiteturas de redes profundas que superaram recordes em tarefas de reconhecimento de imagens, detecção de objetos e segmentação semântica.

O OpenCV (Open Source Computer Vision Library) é uma das bibliotecas de visão computacional mais populares. Ela fornece um grande conjunto de funções e algoritmos para processamento de imagens e vídeo, detecção de objetos, calibração de câmera e muito mais. TensorFlow e PyTorch são duas das bibliotecas de aprendizado profundo mais usadas. Elas são empregadas em uma variedade de projetos de visão computacional e oferecem uma estrutura flexível para construir e treinar modelos de CNN e outros tipos de redes neurais.

O YOLO é um projeto que introduziu uma abordagem eficiente para detecção de objetos em tempo real. Ele permite a detecção de objetos em imagens e vídeos com alta precisão e velocidade. O Mask R-CNN é um projeto que estende o Faster R-CNN para realizar também a segmentação semântica em imagens, é utilizado em tarefas que exigem detecção e segmentação de objetos, como em aplicações médicas e de visão computacional em veículos autônomos.

Como visto, o ImageNet Large Scale Visual Recognition Challenge é um projeto que impulsionou avanços significativos na visão computacional. Ele ofereceu uma grande base de dados rotulada com milhões de imagens e definiu benchmarks para tarefas de classificação de imagens.

Projetos de visão computacional são fundamentais em aplicações de robótica e inteligência artificial, incluindo veículos autônomos (como os da Waymo e Tesla), robôs industriais e assistentes de casa inteligente. Eles ainda desempenham um papel importante em aplicações de realidade aumentada (RA) e realidade virtual (VR) como jogos, simulações e treinamento. Projetos de reconhecimento facial, como o FaceNet do Google, têm aplicação em segurança, autenticação biométrica e identificação de pessoas em redes sociais.

A visão computacional é muito utilizada em aplicações médicas, como diagnóstico por imagem, detecção de câncer, segmentação de órgãos em imagens de ressonância magnética e análise de histologia digital. Ela também é usada para monitorar a qualidade do ar, detectar incêndios florestais, analisar safras e prever doenças de plantas.

Esses são apenas alguns exemplos de tecnologias e projetos em visão computacional. À medida que o campo continua a evoluir, podemos esperar novos avanços tecnológicos e aplicações surgindo em áreas tão diversas quanto saúde, transporte, entretenimento, indústria e muito mais. A visão computacional desempenha um papel cada vez mais importante em nosso cotidiano e na resolução de desafios complexos em todo o mundo.

Lembrete

A visão computacional é muito utilizada em aplicações médicas, como diagnóstico por imagem, detecção de câncer, segmentação de órgãos em imagens de ressonância magnética e análise de histologia digital, bem como para monitorar a qualidade do ar, detectar incêndios florestais, analisar safras e prever doenças de plantas.

8.2 Planejamento de projeto de visão computacional

O planejamento de um projeto de visão computacional é uma etapa fundamental para garantir o sucesso na implementação de sistemas e aplicações que envolvem processamento de imagens e vídeos. Os principais passos e considerações no planejamento de um projeto de visão computacional são:

- **Definição de objetivos**: defina os objetivos do projeto. O que você deseja alcançar com a visão computacional? Isso pode incluir tarefas como detecção de objetos, reconhecimento de padrões, segmentação de imagens, rastreamento de movimento, entre outros.

- **Coleta de dados**: identifique as fontes de dados necessárias para o projeto. Isso pode incluir imagens e vídeos capturados por câmeras, dados de sensores, imagens de satélite etc. Certifique-se de que os dados estejam disponíveis e sejam relevantes aos objetivos do projeto.

- **Análise de requisitos de hardware e software**: avalie os requisitos de hardware, como poder de processamento, memória e capacidade de armazenamento, necessários para executar as tarefas de visão computacional. Escolha o software e as bibliotecas adequadas, como TensorFlow, OpenCV ou PyTorch, com base nas necessidades do projeto.

- **Design da arquitetura do sistema**: projete a arquitetura do sistema, incluindo a seleção de câmeras e sensores, configuração de hardware, escolha de algoritmos de visão computacional e estrutura de software. Certifique-se de que o sistema seja escalável e eficiente.

- **Anotação de dados e treinamento de modelos**: anote os dados de treinamento se o projeto envolver aprendizado de máquina, ou seja, rotule manualmente os objetos de interesse nas imagens. Em seguida, treine os modelos de visão computacional usando esses dados anotados.

- **Desenvolvimento de software**: desenvolva o software necessário para executar as tarefas de visão computacional. Isso pode incluir a implementação de algoritmos, a criação de interfaces de usuário, a integração com outros sistemas e a criação de APIs para acessar os resultados.

- **Testes e validação**: realize testes rigorosos para avaliar o desempenho do sistema em uma variedade de condições. Isso inclui testes de precisão, robustez, escalabilidade e tempo de resposta. A validação também envolve comparar os resultados do sistema com dados de referência ou especialistas humanos.

- **Integração e implantação**: integre o sistema de visão computacional em sua aplicação ou sistema maior. Certifique-se de que ele funcione de maneira harmoniosa com outros componentes e implante-o em um ambiente de produção.

- **Manutenção e atualizações**: planeje a manutenção contínua do sistema de visão computacional. Isso inclui atualizações de software, calibração de câmeras e monitoramento regular para garantir que o sistema continue a funcionar de maneira confiável.

- **Considerações éticas e legais**: esteja ciente das considerações éticas e legais relacionadas ao projeto, especialmente se ele envolver captura de dados pessoais, reconhecimento facial ou questões de privacidade.

- **Orçamento e prazos**: estabeleça um orçamento claro para o projeto, incluindo custos de hardware, software, mão de obra e recursos de treinamento de modelo. Defina prazos realistas para cada fase.

- **Documentação e treinamento**: documente todos os aspectos do projeto, incluindo o design da arquitetura, algoritmos implementados, fluxos de trabalho de processamento de dados e procedimentos de manutenção. Forneça treinamento adequado para os usuários finais, se necessário.

- **Monitoramento e melhoria contínua**: após a implantação, estabeleça um sistema de monitoramento contínuo para avaliar o desempenho do sistema em tempo real. Use os dados coletados para fazer melhorias e otimizações conforme necessário.

O planejamento cuidadoso de um projeto de visão computacional é essencial para alcançar resultados bem-sucedidos e evitar problemas ao longo do caminho. Ao seguir essas etapas e considerações, você estará melhor preparado para desenvolver e implementar com sucesso soluções de visão computacional para uma variedade de aplicações.

8.3 Principais desafios

A visão computacional é um campo empolgante, mas também enfrenta diversos desafios técnicos e práticos. A variabilidade de dados, como imagens e vídeos podem variar significativamente em termos de iluminação, perspectiva, resolução, qualidade e escala. Desenvolver algoritmos robustos que funcionem bem em diferentes condições é um desafio constante.

Detectar objetos em imagens ou vídeos é uma tarefa desafiadora, especialmente quando eles estão parcialmente obscurecidos, em poses incomuns ou em cenários ruidosos. Segmentar imagens em regiões semânticas (por exemplo, separar objetos de fundos) é uma tarefa complexa, especialmente naquelas com sobreposição de objetos e texturas complexas.

Analisar informações em sequências de vídeo em tempo real é um desafio, pois requer a detecção e o rastreamento de objetos em movimento constante. A primeira etapa é adquirir o vídeo em tempo real. Isso pode ser feito por meio de câmeras de vigilância, câmeras de smartphones, câmeras de veículos autônomos ou qualquer dispositivo que capture vídeo. A aquisição de vídeo é frequentemente realizada usando bibliotecas como OpenCV, que oferecem suporte para diversas fontes de vídeo. O vídeo bruto geralmente precisa de pré-processamento. Isso inclui ajustes de brilho e contraste, redução de ruído e correção de distorções da lente, entre outros. Essas etapas garantem que os dados de entrada estejam em boas condições para análise. Na sequência precisamos extrair as características, fazer a análise e a tomada de decisão e a integração com sistemas de controle.

Em muitos cenários, as informações extraídas do vídeo são usadas para controlar sistemas em tempo real. Por exemplo, um sistema de segurança pode acionar um alarme em caso de detecção de intrusão, ou um sistema de automação industrial pode ajustar o funcionamento com base nas informações do vídeo. As informações processadas podem ser exibidas em tempo real, normalmente em uma interface gráfica,

permitindo que operadores humanos monitorem o sistema e tomem decisões com base nas informações geradas. Em muitos casos, é importante armazenar dados brutos ou informações processadas para fins de análise posterior ou para cumprir requisitos regulatórios.

Em cenários de tempo real, a otimização de desempenho é crucial. Isso envolve o uso de algoritmos eficientes, paralelização de processamento e uso de hardware especializado, como GPUs (unidades de processamento gráfico).

Figura 41 – Monitores trabalhando no Centro de Operações de Segurança da Universidade de Maryland

Disponível em: https://tinyurl.com/mrxa83ru. Acesso em: 24 out. 2023.

O uso de tecnologias de visão computacional em reconhecimento facial e vigilância levanta preocupações sobre a privacidade e a ética, exigindo regulamentação adequada. Além das preocupações com a privacidade, ela levanta questões éticas sobre o uso da tecnologia, como viés algorítmico e discriminação.

Garantir a segurança de sistemas de visão computacional é fundamental, especialmente em aplicações críticas, como veículos autônomos e sistemas de reconhecimento de segurança.

Há limitações de dados rotulados, isto é, a anotação manual de grandes conjuntos de dados para treinar algoritmos é um processo demorado e caro. A disponibilidade de dados rotulados é um desafio constante.

Modelos de aprendizado profundo são frequentemente caixas-pretas, dificultando a interpretação de suas decisões. Tornar os modelos mais interpretables é uma área ativa de pesquisa.

Escalar algoritmos de visão computacional para lidar com grandes volumes de dados em tempo real, como em aplicações de veículos autônomos, é um desafio significativo. Tais algoritmos, especialmente aqueles baseados em aprendizado profundo, podem ser computacionalmente intensivos. Garantir que o hardware seja adequado e eficiente é essencial.

Assegurar que modelos treinados em um conjunto de dados funcionem bem em novos conjuntos de dados é um desafio. O desenvolvimento de técnicas de transferência de aprendizado é uma solução em evolução. Calibrar câmeras com precisão é essencial para muitas aplicações de visão computacional, como realidade aumentada e mapeamento 3D.

Enfrentar esses desafios exige pesquisa contínua, desenvolvimento de algoritmos mais avançados, coleta de dados diversificados e um compromisso com a ética e a segurança. Apesar dos desafios, a visão computacional continua a avançar e desempenhar um papel fundamental em diversas áreas, desde medicina até automação industrial e entretenimento.

8.4 Organização das etapas de desenvolvimento

A organização das etapas de desenvolvimento de um projeto de visão computacional é fundamental para garantir que o projeto seja bem-sucedido, eficiente e cumpra seus objetivos. Constituem as principais etapas organizadas de maneira sequencial:

- **Definição do problema**: trata-se do ponto de partida fundamental. Nesta etapa, é preciso não apenas identificar o problema, mas entender o contexto, os requisitos e as metas do projeto. Por exemplo, no caso do desenvolvimento de um sistema de reconhecimento facial, é importante definir claramente o que se espera do sistema, como taxa de reconhecimento, tempo de resposta e cenários de uso. Além disso, a definição do problema envolve a coleta de dados de treinamento ou aquisição de fontes de vídeo relevantes.

- **Aquisição de dados**: a qualidade e a quantidade dos dados adquiridos desempenham um papel crítico no sucesso do projeto. Isso envolve a escolha de hardware apropriado, como câmeras e sensores, bem como a criação de um procedimento para capturar dados consistentes e representativos. Além disso, pode ser necessário rotular os dados para treinamento de modelos de aprendizado de máquina, que é um processo trabalhoso e requer atenção aos detalhes.

- **Pré-processamento de dados**: os dados brutos são preparados para análise. Isso inclui ajustes de contraste, equalização de histograma, remoção de ruído e normalização. Em projetos de visão computacional, esta etapa pode incluir também a calibração da câmera para corrigir distorções geométricas. A qualidade do pré-processamento afeta diretamente a qualidade das análises posteriores.

- **Extração de características**: é um componente fundamental em projetos de visão computacional. Envolve a identificação e o isolamento de informações relevantes de uma imagem ou vídeo, que podem ser usadas para descrever e distinguir objetos, padrões ou regiões de interesse. Essas características são cruciais para a tomada de decisões e análises em uma variedade de aplicações, como reconhecimento de objetos, detecção de padrões, classificação de imagens e muito mais. Constituem as técnicas comuns de extração de características usando processamento de imagem:

  – **Detecção de borda**: envolve a identificação de transições bruscas de intensidade de pixel em uma imagem. Isso é feito através da aplicação de operadores, como o operador Sobel ou Canny, que realçam as bordas dos objetos na imagem. As bordas representam mudanças significativas nas características visuais e são frequentemente usadas na segmentação de objetos.

  – **Extração de textura**: refere-se à variação da intensidade dos pixels em pequenas regiões da imagem. A extração de características de textura pode incluir o uso de matrizes de co-ocorrência, histogramas de níveis de cinza ou descritores estatísticos para representar a textura de uma região. Isso é útil em tarefas como identificação de materiais e classificação de superfícies.

  – **Detecção de pontos de interesse**: assim como cantos e keypoints, são locais distintivos na imagem que podem ser usados para rastreamento, alinhamento de imagens e correspondência. Algoritmos como o Harris corner detector e o SIFT (Scale-invariant feature transform) são usados para identificar esses pontos.

  – **Histogramas de cor**: representam a distribuição das cores na imagem. A extração de características de histogramas de cor é útil na classificação de imagens e na detecção de objetos com base em cores específicas.

  – **Momentos de imagem**: são descritores estatísticos que resumem a distribuição de intensidades de pixel na imagem. Eles podem ser usados para caracterizar a forma de objetos na imagem e para tarefas como reconhecimento de caracteres manuscritos.

  – **Transformações geométricas e morfológicas**: técnicas de transformação geométrica, como transformações de afinidade e rotações, podem ser usadas para extrair características relacionadas à geometria dos objetos na imagem. Além disso, operações morfológicas, como erosão e dilatação, podem destacar características estruturais.

  – **Filtros de resposta de frequência**: assim como o filtro de Gabor, podem ser aplicados para destacar características baseadas em padrões de textura, orientação e frequência na imagem.

  – **Segmentação de regiões de interesse**: a segmentação divide a imagem em regiões ou objetos distintos. As características extraídas de regiões segmentadas podem incluir área, perímetro, centroide e outros descritores relacionados à forma.

- **Seleção de algoritmos**: é um passo crítico. A escolha dos algoritmos de processamento de imagem e visão computacional depende das características extraídas e dos objetivos do projeto. Além disso, é importante considerar o desempenho computacional e a eficiência dos algoritmos, especialmente em cenários de tempo real.

- **Treinamento de modelos (se aplicável)**: se o projeto envolver aprendizado de máquina, como redes neurais, é necessário treinar modelos com dados etiquetados. Isso pode envolver a criação de conjuntos de treinamento e validação, além da escolha de arquiteturas de modelo e algoritmos de treinamento. O treinamento é iterativo e requer ajustes para otimizar o desempenho.

- **Implementação do software**: envolve a integração dos algoritmos e modelos em uma aplicação funcional. Isso requer programação, desenvolvimento de interfaces e integração com sistemas existentes, quando aplicável.

- **Avaliação e testes**: essa etapa é crucial para garantir que o sistema atenda aos requisitos. Isso inclui a realização de testes rigorosos, medição de desempenho, validação cruzada e comparação dos resultados com métricas predefinidas. O feedback dos testes é usado para ajustar o sistema e melhorar seu desempenho.

- **Otimização e ajustes**: com base nos resultados dos testes, o sistema é otimizado e ajustado. Isso pode incluir otimizações de código, afinamento de hiperparâmetros de algoritmos e refinamento de técnicas de pré-processamento.

8.5 Implantação do projeto

A implantação de um projeto de visão computacional envolve a instalação e operação prática do sistema em um ambiente real. Certifique-se de que todo o hardware necessário esteja configurado e funcione corretamente. Isso inclui câmeras, sensores, computadores ou dispositivos que executarão o software. Além disso, verifique se o software está instalado e configurado adequadamente.

Antes de implantar o sistema em um ambiente de produção, é aconselhável realizar testes iniciais em um ambiente controlado. Isso permite detectar e corrigir quaisquer problemas antes da implantação completa.

Calibre câmeras e sensores conforme necessário. A calibração é importante para garantir que as medições de distância e perspectiva sejam precisas. Além disso, ajuste os parâmetros do software de visão para otimizar o desempenho.

Se o projeto envolver câmeras ou sensores físicos, instale-os nas posições desejadas. Certifique-se de que eles estejam adequadamente fixados e protegidos contra condições climáticas adversas, vandalismo ou outros danos.

Integre o sistema de visão computacional com os outros componentes do sistema maior, se aplicável. Isso pode envolver a configuração de comunicações, interfaces de usuário e sincronização com outros dispositivos ou sistemas.

Realize testes de integração para garantir que o sistema completo funcione conforme o esperado. Isso envolve a verificação de que todos os componentes interagem corretamente e que os dados são transmitidos e processados sem problemas. Após a integração, realize uma validação completa do sistema em um ambiente real. Isso pode incluir testes sob condições de iluminação, clima e tráfego reais. Verifique se o sistema atende aos requisitos e objetivos do projeto.

Treine a equipe de operação e manutenção do sistema para garantir que eles saibam como operar o sistema de visão computacional corretamente. Isso inclui o uso de interfaces de usuário, resolução de problemas e procedimentos de manutenção. Estabeleça um sistema de monitoramento em tempo real para rastrear o desempenho do sistema após a implantação. Isso pode incluir o monitoramento de alertas, erros ou comportamentos anômalos.

Programe manutenções preventivas regulares para garantir que o sistema continue a funcionar de maneira confiável. Mantenha o software e os modelos de visão atualizados para incorporar melhorias e correções de bugs. Após a implantação, continue a avaliar o desempenho do sistema e colete dados para fazer melhorias adicionais. Ouça o feedback dos usuários e esteja pronto para ajustar o sistema conforme necessário.

A implantação de um projeto de visão computacional é uma etapa crítica que requer cuidado e atenção aos detalhes. É importante garantir que o sistema funcione de maneira confiável e eficaz no ambiente real, atendendo aos objetivos do projeto e às expectativas dos usuários. A manutenção e a avaliação contínua são essenciais para garantir o sucesso em longo prazo.

8.6 Aplicações de reconhecimento facial e segmentação de objetos

O reconhecimento facial e a segmentação de objetos são duas tecnologias poderosas no campo da visão computacional e processamento de imagem. Ambas têm uma grande quantidade de aplicações em diversos setores, desde segurança e saúde até entretenimento e varejo. A seguir exploraremos algumas das principais aplicações de cada uma delas:

Aplicações de reconhecimento facial

- **Segurança e vigilância**: o uso mais comum do reconhecimento facial é em sistemas de segurança, como controle de acesso a edifícios, aeroportos e eventos. Ele pode identificar pessoas autorizadas e detectar criminosos procurados.

- **Autenticação biométrica**: o reconhecimento facial é utilizado para autenticação em dispositivos móveis e sistemas de segurança, substituindo senhas tradicionais.

- **Monitoramento de multidões**: em eventos esportivos, concertos e manifestações públicas, o reconhecimento facial pode ser usado para monitorar multidões em tempo real, ajudando na segurança e na identificação de pessoas desaparecidas.

- **Marketing personalizado**: algumas empresas usam o reconhecimento facial para identificar o perfil demográfico de seus clientes em lojas e shoppings, permitindo direcionar anúncios personalizados.

- **Assistência médica**: em hospitais, o reconhecimento facial pode ajudar a identificar pacientes e acessar seus registros médicos com segurança, melhorando o atendimento ao paciente.

- **Entretenimento**: em jogos e aplicativos de entretenimento, o reconhecimento facial permite aos jogadores controlar personagens e interagir com o ambiente usando gestos faciais.

Aplicações de segmentação de objetos

- **Detecção de veículos autônomos**: a segmentação de objetos é crucial em veículos autônomos para identificar pedestres, outros veículos e obstáculos na estrada.

- **Saúde e medicina**: na medicina, a segmentação de objetos é usada para identificar e rastrear órgãos e estruturas do corpo em imagens médicas, como ressonâncias magnéticas e tomografias.

- **Agricultura de precisão**: agricultores usam a segmentação de objetos para identificar pragas, doenças e o crescimento das culturas, permitindo intervenções mais eficientes.

- **Manufatura inteligente**: a segmentação de objetos é usada em sistemas de inspeção de qualidade para identificar defeitos em produtos e embalagens de forma rápida e precisa.

- **Realidade aumentada**: em aplicativos de realidade aumentada, a segmentação de objetos é usada para sobrepor objetos virtuais ao mundo real de forma precisa.

- **Varejo e logística**: no varejo, a segmentação de objetos é usada para rastrear produtos em estoque, automatizar processos de gerenciamento de inventário e facilitar a automação na logística.

- **Segurança de vídeo**: em sistemas de segurança de vídeo, a segmentação de objetos ajuda a identificar atividades suspeitas, como invasões e movimentos não autorizados.

Ambas as tecnologias estão evoluindo rapidamente com o avanço da inteligência artificial e do aprendizado profundo (deep learning). No entanto, é importante considerar questões éticas e de privacidade ao implementar essas tecnologias, especialmente o reconhecimento facial, para garantir que elas sejam usadas de maneira responsável e respeitem os direitos individuais.

8.7 Arcabouço tecnológico

O campo da visão computacional depende de um arcabouço tecnológico robusto para desenvolver aplicações que envolvem processamento de imagens e vídeos. Esse arcabouço é composto de várias plataformas, bibliotecas e linguagens de programação que fornecem ferramentas e recursos para a análise e manipulação de dados visuais. A seguir exploraremos esses componentes em detalhes:

Plataformas

- **OpenCV (Open Source Computer Vision Library)**: o OpenCV é uma plataforma de código aberto utilizada para visão computacional. Ele oferece ampla variedade de algoritmos e funções para processamento de imagens, detecção de objetos, rastreamento, calibração de câmera e muito mais. O OpenCV é compatível com várias linguagens de programação, incluindo Python, C++, Java e outros.

- **TensorFlow**: desenvolvido pela Google, o TensorFlow é uma plataforma de aprendizado de máquina que também oferece suporte para visão computacional. A biblioteca TensorFlow tem módulos específicos para processamento de imagens, como o TensorFlow Object Detection API.

- **PyTorch**: PyTorch é outra biblioteca de aprendizado de máquina que ganhou popularidade. Ele oferece suporte sólido para visão computacional com módulos como torchvision, que inclui conjuntos de dados, modelos pré-treinados e ferramentas para treinar redes neurais para tarefas de visão.

- **Microsoft Azure Computer Vision**: a plataforma Azure da Microsoft fornece serviços de visão computacional na nuvem. O Azure Computer Vision API permite que os desenvolvedores integrem recursos de reconhecimento de imagem em suas aplicações com facilidade.

Bibliotecas

- **Dlib**: Dlib é uma biblioteca C++ com várias funcionalidades, incluindo detecção facial, detecção de pontos de referência faciais, rastreamento de objetos e muito mais. É utilizada em aplicações de visão computacional e aprendizado profundo.

- **scikit-image**: esta biblioteca Python é baseada no scikit-learn e fornece uma coleção de algoritmos para processamento de imagem, como filtragem, segmentação, transformações e muito mais.

- **SimpleCV**: SimpleCV é uma biblioteca Python que simplifica o desenvolvimento de aplicativos de visão computacional. Ela inclui uma variedade de ferramentas para trabalhar com imagens e vídeos, tornando mais fácil a implementação de tarefas comuns.

Linguagens de programação

Python: Python é a linguagem de programação mais usada na visão computacional devido à sua facilidade de uso, vasta quantidade de bibliotecas de suporte e comunidade ativa de desenvolvedores. Bibliotecas como OpenCV, scikit-image e TensorFlow têm interfaces Python.

C++: C++ é frequentemente usado quando é necessária alta performance, como em sistemas embarcados e aplicações em tempo real. O OpenCV, por exemplo, oferece suporte nativo para C++.

Java: Java é usado principalmente em aplicativos Android que envolvem visão computacional. A OpenCV oferece uma API Java para desenvolvimento em dispositivos Android.

C#: a Microsoft oferece suporte ao desenvolvimento de aplicativos de visão computacional em C# por meio das bibliotecas do Azure Computer Vision e da Emgu.CV, uma interface C# para o OpenCV.

O arcabouço tecnológico em visão computacional está em constante evolução, à medida que novas tecnologias e bibliotecas são desenvolvidas. A escolha de plataformas, bibliotecas e linguagens depende das necessidades específicas do projeto, dos recursos disponíveis e da experiência da equipe de desenvolvimento.

Resumo

Vimos nesta unidade que as operações morfológicas são técnicas essenciais no processamento de imagens, especialmente em visão computacional, pois permitem a manipulação e transformação de regiões específicas de uma imagem com base em seus contornos e estruturas. Essas operações são úteis para tarefas como pré-processamento, segmentação, detecção de bordas, remoção de ruído e muito mais. As operações morfológicas são realizadas usando elementos estruturantes, que são pequenos kernels ou máscaras que deslizam sobre a imagem.

Elas são aplicadas em diferentes contextos, como processamento de imagens médicas, visão computacional, análise de documentos, entre outros. A escolha do elemento estruturante e a sequência de operações dependem da tarefa específica e das características da imagem. Por exemplo, na detecção de bordas, a erosão pode ser usada para afinar os contornos, enquanto a dilatação pode engrossá-los. No entanto, é importante observar que as operações morfológicas são sensíveis à escolha do elemento estruturante, e a seleção adequada desse elemento é fundamental para obter resultados precisos.

Estudamos também sobre a extração de características, um processo muito importante em processamento de imagem e visão computacional, quando informações significativas são identificadas e isoladas em uma imagem para representar objetos, regiões ou padrões específicos. Essas características são cruciais para descrever e distinguir elementos visuais em uma imagem ou vídeo.

Demonstramos que em tarefas de visão computacional uma imagem pode conter grande quantidade de informações, mas nem todas são relevantes para a análise. A extração de características permite reduzir a dimensão dos dados, concentrando-se nas informações mais relevantes. Características bem escolhidas simplificam tarefas de processamento, como classificação, detecção de objetos, reconhecimento de padrões e segmentação.

Ao representar uma imagem por meio de características, ela pode ser comparada e analisada de forma mais eficiente. Isso é essencial em tarefas como recuperação de imagens ou correspondência de objetos.

As características incluem informações básicas, como intensidade de pixel, textura, cor e bordas. Operações como detecção de bordas, histogramas de cores e transformadas de Fourier podem ser usadas para extrair características de baixo nível. Elas representam informações mais complexas, como formas,

estruturas geométricas e padrões. São exemplos: pontos de interesse, descritores SIFT (scale-invariant feature transform) e descritores HOG (histogram of oriented gradients). Textura se refere à repetição de padrões em uma imagem. Características de textura capturam essa repetição e podem ser usadas em tarefas de classificação ou segmentação, como reconhecimento de tecidos em imagens médicas.

As características representam estatísticas da imagem, como média, desvio padrão, assimetria e curtose. São frequentemente usadas para descrever distribuições de intensidade de pixel. Características derivadas de transformações da imagem, como a transformada de Hough, é usada para identificar linhas em imagens.

Observamos que a extração de características geralmente começa com a aquisição da imagem, seguida pelas etapas de pré-processamento, quando a imagem é corrigida, ajustada e filtrada, se necessário. Na etapa de detecção, as características são identificadas. Isso pode envolver a aplicação de algoritmos específicos para identificar bordas, pontos de interesse ou outros elementos relevantes. Na etapa de descrição, as características identificadas são quantificadas, normalmente em forma de vetores de características. Isso envolve calcular valores ou descritores que as representam. Por fim, as características extraídas são usadas em tarefas de análise de imagem, como reconhecimento, classificação, detecção de objetos ou qualquer outro propósito para o qual as informações representadas pelas características sejam relevantes.

Vimos também que o reconhecimento de padrões é a capacidade de identificar regularidades ou padrões em dados e, com base neles, fazer previsões ou tomar decisões. O reconhecimento de padrões tem aplicações em uma variedade de campos, incluindo visão por computador, processamento de linguagem natural, biometria, medicina, finanças e muito mais.

Com o aumento no volume de dados, técnicas de reconhecimento de padrões baseadas em aprendizado profundo têm ganhado destaque, particularmente em visão computacional e processamento de linguagem natural. O reconhecimento de padrões tem aplicações crescentes em campos como veículos autônomos, assistência médica, análise de vídeo, segurança cibernética e muito mais. Desafios incluem o tratamento de dados desequilibrados, a interpretabilidade de modelos complexos e a necessidade de lidar com o viés em algoritmos de reconhecimento.

Apresentamos ainda que a visão computacional é uma disciplina interdisciplinar que envolve o desenvolvimento de algoritmos e sistemas para a interpretação de informações visuais a partir de imagens ou vídeos.

Ela permite que as máquinas entendam, analisem e tomem decisões com base em dados visuais, tornando-a uma parte essencial da inteligência artificial e da automação.

O avanço em técnicas de aprendizado profundo, como redes neurais convolucionais (CNNs), revolucionou a visão computacional, permitindo o reconhecimento de objetos e características complexas. Desafios contínuos incluem o tratamento de variações na iluminação, escala e rotação, bem como a interpretabilidade de modelos de aprendizado profundo. A combinação de visão computacional com outras tecnologias, como Internet das Coisas (IoT) e processamento de linguagem natural, cria oportunidades para aplicações mais abrangentes.

A visão computacional desempenha um papel importante em várias indústrias, proporcionando automação, análise de dados visuais e aprimoramento de sistemas de segurança e assistência médica. Trata-se de uma área em constante evolução, com aplicações cada vez mais amplas à medida que as máquinas se tornam mais capazes de entender o mundo visual ao nosso redor.

Exercícios

Questão 1. (UFSM 2022, adaptada) Sobre o processamento morfológico de imagens, avalie as afirmativas.

I – A morfologia matemática é fundamentada na teoria dos conjuntos e tem como exemplos de operadores morfológicos a erosão, a dilatação, a abertura e o fechamento.

II – As operações morfológicas em imagens binárias ocorrem de forma diferente das operações morfológicas em imagens tons de cinza.

III – A dilatação e a erosão morfológica em imagens tons de cinza podem ser usadas em conjunto com a subtração de imagens para obter o gradiente morfológico da imagem.

É correto o que se afirma em:

A) I, apenas.

B) III, apenas.

C) I e II, apenas.

D) II e III, apenas.

E) I, II e III.

Resposta correta: alternativa E.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: a teoria dos conjuntos fornece uma estrutura matemática que permite a análise de relações entre conjuntos de pixels em uma imagem. Assim, a morfologia matemática utiliza conceitos e técnicas da teoria dos conjuntos para analisar e processar imagens. Os operadores morfológicos incluem a erosão, a dilatação, a abertura e o fechamento.

II – Afirmativa correta.

Justificativa: as operações morfológicas em imagens binárias, em imagens em tons de cinza e em imagens em cores têm estudos e comportamentos diferentes devido à natureza dos pixels em cada tipo de imagem.

III – Afirmativa correta.

Justificativa: o gradiente morfológico de uma imagem é uma técnica que combina as operações de dilatação e de erosão para destacar as transições de intensidade nela. Subtrair a imagem erodida da dilatada é uma maneira de calcular o gradiente morfológico. Isso destaca as bordas e as transições de intensidade na imagem em tons de cinza.

Questão 2. (UFSM 2022, adaptada) Uma CNN (Convolutional Neural Network) representa um tipo particular de arquitetura de uma rede neural profunda que é particularmente adequada para conjuntos de dados de imagem. Em adição, devido ao seu bom desempenho quando comparada às abordagens tradicionais, as CNNs estão sendo utilizadas no desenvolvimento de várias aplicações que envolvem visão computacional.

Considerando as redes neurais convolucionais e suas aplicações, avalie as asserções e a relação proposta entre elas.

I – As camadas de pooling são o elemento central das CNNs e responsáveis por extrair características das imagens.

porque

II – Uma das principais vantagens das CNNs é a sua capacidade de aprender características hierárquicas automaticamente.

Assinale a alternativa correta.

A) As asserções I e II são verdadeiras, e a asserção II justifica a I.

B) As asserções I e II são verdadeiras, e a asserção II não justifica a I.

C) A asserção I é verdadeira, e a II é falsa.

D) A asserção I é falsa, e a II é verdadeira.

E) As asserções I e II são falsas.

Resposta correta: alternativa D.

Análise das asserções

I – Asserção falsa.

Justificativa: o elemento central das CNNs, responsável por extrair características das imagens, são as camadas de convolução. As camadas de pooling reduzem a dimensionalidade da imagem, preservando as características mais importantes.

II – Asserção verdadeira.

Justificativa: à medida que os dados de entrada passam pelas camadas, a rede neural aprende representações progressivamente mais abstratas das características das imagens. Ou seja, as camadas iniciais capturam detalhes de baixo nível, enquanto as camadas posteriores combinam essas características para formar representações mais abstratas.

REFERÊNCIAS

Textuais

ABUALHOUL, M. Y.; MUNOZ, E. T.; NASHASHIBI, F. The Use of Lane-Centering to Ensure the Visible Light Communication Connectivity for a Platoon of Autonomous Vehicles. *In:* IEEE INTERNATIONAL CONFERENCE ON VEHICULAR ELECTRONICS AND SAFETY (ICVES), 2018., Madrid. *Anais* [...]. Madrid: ICVES, 2018. Disponível em: https://tinyurl.com/yzp7fe2a. Acesso em: 24 out. 2023.

AZEVEDO, E.; CONCI, A.; LETA, F. *Computação gráfica*. Teoria e prática: geração de imagens – vol. 2. Rio de Janeiro: Alta Books, 2022.

BARELLI, F. *Introdução à visão computacional*: uma abordagem prática com Python e .OpenCV. São Paulo: Casa do Código, 2018.

DUDA, R. O.; HART, P. E. *Pattern classification and scene analysis*. New York: A Wiley-Interscience Publication, 1974.

FILGUEIRAS, C.; GARROT, J. *Introdução ao processamento digital de imagem*. Lisboa: FCA, 2008.

GOMES, J. M.; VELHO, L. *Image Processing for Computer Graphics*. New York: Springer, 1997.

GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens*. 3. ed. São Paulo: Pearson, 2010.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. *ImageNet Classification with Deep Convolutional Neural Networks*. New York: Curran Associates, Inc., 2012. Disponível em: https://tinyurl.com/274zb4dv. Acesso em: 24 out. 2023.

LAGANIERE, R. *OpenCV 3 Computer Vision Application Programming Cookbook*. 3rd. ed. Birmingham: Packt Publishing, 2017.

LIENHART, R. *Intel License Agreement For Open Source Computer Vision Library*. GitHub, 2000. Disponível em: https://tinyurl.com/2dnjctfe. Acesso em: 24 out. 2023.

LIMA, I.; PINHEIRO, C. A.; SANTOS, F. A. *Inteligência artificial*. São Paulo: Grupo GEN, 2014.

MARR, D. *Vision*: a computational investigation into the human representation and processing of visual information. San Francisco: W. H. Freeman, 1982.

NOAA. GOES *Image Viewer*. [s.d.]. Disponível em: https://tinyurl.com/3bhswuun. Acesso em: 24 out. 2023.

NORVIG, P.; RUSSEL, S. *Inteligência artificial*. 3. ed. São Paulo: Grupo GEN, 2013.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais*: princípios, algoritmos e aplicações. São Paulo: Cengage Learning Brasil, 2007.

PICHETTI, R. F. *et al. Computação gráfica e processamento de imagens*. Porto Alegre: Sagah, 2022.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386-408, 1958. Disponível em: https://tinyurl.com/mwyphy6f. Acesso em: 24 out. 2023.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, n. 323, p. 533-536, 1986.

SÁ, Y. V. *Desenvolvimento de aplicações IA*: robótica, imagem e visão computacional. São Paulo: Saraiva, 2021.

SANTOS, M. H. *Introdução à inteligência artificial*. São Paulo: Saraiva, 2021.

VIEIRA NETO, H.; MARQUES FILHO, O. *Processamento digital de imagens*. Rio de Janeiro: Brasport, 1999.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Kauai: CVPR, 2001. Disponível em: https://tinyurl.com/bdf6t29u. Acesso em: 24 out. 2023.

Informações:
www.sepi.unip.br ou 0800 010 9000