Detecção de Melanoma com Convolução e Correlação

Isaac Alves Schuenck

Engenharia de Computação – Pontifícia Universidade Católica de Minas Gerais (PUCMG)

1. Introdução

A detecção precoce do câncer de pele, especialmente do melanoma, é uma das principais prioridades da dermatologia moderna. Com o avanço da inteligência artificial e do aprendizado de máquina, tornou-se viável aplicar técnicas de visão computacional para auxiliar na triagem automática de lesões cutâneas.

Nesse contexto, o domínio de técnicas fundamentais de processamento digital de imagens torna-se essencial, especialmente com o uso crescente de sistemas inteligentes e análise de imagens em tempo real. Entre essas técnicas, destacam-se a correlação e a convolução, amplamente empregadas na extração de características, realce e filtragem de imagens.

Segundo Gonzalez e Woods (2018), operações como a convolução são a base de métodos aplicados em visão computacional, reconhecimento de padrões, robótica e diagnóstico médico. Em particular, as redes neurais convolucionais (CNNs), que lideram benchmarks em tarefas de classificação e segmentação de imagens, são construídas a partir dessas operações fundamentais.

Este trabalho tem como objetivo aplicar os conceitos de correlação e convolução 2D com o filtro de Sobel já explorados em atividades anteriores no contexto da classificação de imagens médicas. Para isso, utilizamos um conjunto de imagens reais de lesões benignas e malignas, que são processadas com técnicas de realce de bordas e classificadas por meio de CNNs.

2. Fundamentação Teórica

2.1. Correlação

A correlação 2D consiste em aplicar um pequeno bloco de valores, chamado filtro ou kernel, sobre regiões da imagem. A cada posição, realiza-se a multiplicação ponto a ponto entre o kernel e os pixels da região e soma-se o resultado, deslocando o filtro por toda a imagem.

Matematicamente:

$$R(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

Onde:

- I é a imagem.
- K é o kernel.
- R é o resultado da correlação.

2.2. Convolução

A convolução difere da correlação porque o kernel é rotacionado 180° antes da aplicação:

$$R(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(-m, -n)$$

Essa operação preserva certas propriedades matemáticas úteis, como comutatividade e associatividade. É a forma adotada em redes neurais convolucionais, onde os filtros aprendem automaticamente a extrair bordas, texturas e formas.

2.3. Diferenças na Prática

- Correlação é mais intuitiva para aplicação direta de filtros visuais (Sobel, média, etc.).
- Convolução é preferida em sistemas matematicamente rigorosos, como CNNs e transformadas.
- Ambas produzem resultados parecidos, exceto para filtros não simétricos.

2.4. Filtro de Sobel

O filtro de Sobel é um operador de gradiente que enfatiza bordas horizontais e verticais. Ele é composto por dois kernels:

sobel_horizontal e sobel_vertical

```
sobel_h = np.array([[1, 0, -1],
                    [2, 0, -2],
                    [1, 0, -1]])
sobel_v = np.array([[1, 2, 1],
                    [0, 0, 0],
                    [-1, -2, -1]])
```

O uso combinado dos dois permite calcular a magnitude do gradiente, evidenciando as bordas na imagem.

3. Artigos Relacionados

Ouhtit et al. (2023) propõem uma abordagem híbrida que integra filtros de Sobel direcional ao pipeline de detecção de melanoma. Eles aplicam estratégias robustas de pré-processamento (como DullRazor) e extraem características de borda via Sobel antes de treinar uma pilha de Restricted Boltzmann Machines, alcançando acurácias superiores a 95% em bases renomadas .

Essa abordagem valida a ideia de integrar métodos clássicos de processamento de imagem com redes neurais, reforçando a relevância do nosso uso de Sobel com correlação e convolução manual no trabalho, mesmo em um pipeline simplificado comparado ao deles.

4. Exercício Base

A seguir, peguei um exercício da lista solicitada e impleentei manualmente, para fins didáticos.

Exercício 3: Correlação e Convolução

Dado um filtro 3×3:

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Calcular:

- A correlação da imagem com o filtro.
- A convolução da imagem com o filtro.

O script implementa manualmente a correlação e a convolução 2D entre uma imagem e um kernel, com impressões passo a passo úteis para visualização e entendimento do processo, especialmente para fins educacionais ou demonstrativos.

4.1. Correlacao2d(imagem, kernel)

Essa função percorre a imagem ignorando as bordas (com base no tamanho do kernel) e aplica a correlação:

- Para cada pixel válido da imagem (excluindo bordas), é extraído um bloco do mesmo tamanho do kernel.
- Esse bloco é multiplicado elemento a elemento pelo kernel (produto de Hadamard).
- O somatório do resultado é armazenado na posição correspondente da matriz de saída.

```
for i in range(pad, altura - pad):
    for j in range(pad, largura - pad):
        # Extrai o bloco da imagem do mesmo tamanho do kernel
        bloco = imagem[i - pad:i + pad + 1, j - pad:j + pad + 1]
        # Produto escalar (soma dos produtos elemento a elemento)
        produto = bloco * kernel
        soma = np.sum(produto)
        saida[i - pad, j - pad] = soma
```

4.2. Convolucao2d(imagem, kernel)

A convolução é feita reutilizando a função anterior:

```
# Rotaciona o kernel 180 graus
kernel_rotacionado = np.flipud(np.fliplr(kernel))
print("\n===== Convolução =====")
print("Kernel rotacionado:\n", kernel_rotacionado)
return correlacao2d(imagem, kernel_rotacionado)
```

A convolução se diferencia da correlação pela rotação do kernel em 180°, por isso antes de chamar correlacao2d, o kernel é rotacionado com flipud e fliplr.

4.3. Convolucao2d(imagem, kernel)

A imagem usada é uma matriz 5x5 crescente, e o kernel é o Sobel horizontal:

```
# Imagem 5x5 usada no exercício 1: Histograma e Histograma Acumulado
imagem = np.array([
    [0, 1, 2, 2, 3],
    [1, 2, 3, 3, 4],
    [2, 3, 4, 4, 5],
    [3, 4, 5, 5, 6],
    [4, 5, 6, 6, 7]
])
# Filtro 3x3 usado no exercicio 3
kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])
```

Esse kernel realça bordas verticais na imagem. O script calcula e imprime o resultado da correlação e da convolução, permitindo ver as diferenças práticas (que serão mínimas aqui, pois o kernel é simétrico horizontalmente).

4.4. Resultado

- Correlação

```
Resultado da Correlação:
 [[-6. -3. -3.]
 [-6. -3. -3.]
 [-6. -3. -3.]]
```

Esse resultado mostra que:

- A coluna da esquerda da imagem tem valores menores que a da direita (ou seja, está "subindo").
- O filtro não foi rotacionado, então os valores negativos significam que o lado esquerdo da vizinhança está menor do que o direito.
- A borda vertical está sendo detectada no sentido esquerda → direita.

- Convolução

```
Resultado da Convolução:
 [[6. 3. 3.]
 [6. 3. 3.]
 [6. 3. 3.]]
```

Aqui, a única diferença em relação à correlação é que o kernel foi rotacionado 180° antes de aplicar.

- Agora o filtro compara o lado direito contra o esquerdo, invertendo o sinal do resultado.
- Por isso os valores ficaram positivos, com as mesmas magnitudes.
- Esse comportamento mostra claramente a diferença matemática entre convolução e correlação: os valores são espelhados no sinal quando o kernel não é simétrico (como nesse caso).

5. Metodologia

5.1. Mudança de Domínio

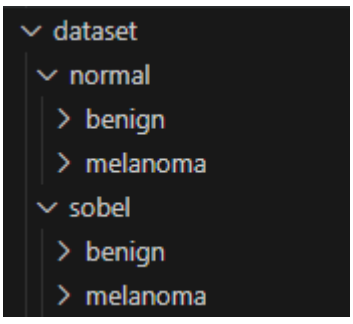
Inicialmente, o projeto focava no reconhecimento de escudos esportivos. Porém, devido à limitação do volume de imagens (apenas 60 exemplos criados manualmente), optou-se por um tema com maior potencial prático e melhor disponibilidade de dados: a detecção de melanoma em imagens médicas. Assim, foi possível aplicar as mesmas técnicas fundamentais (correlação e convolução com filtro de Sobel) em um contexto real e desafiador, ampliando significativamente a utilidade do trabalho:

5.2. Base de Dados

O conjunto de dados utilizado foi o Skin Cancer MNIST: HAM10000, uma base pública amplamente utilizada em pesquisas médicas. Ela contém:

- 10.015 imagens dermatoscópicas
- 7 classes de doenças da pele, das quais foram selecionadas apenas duas para classificação binária:
    - Melanoma (maligno)
    - Lesões benignas

Para simplificar o modelo e manter coerência com o objetivo do trabalho, as imagens foram reorganizadas em duas pastas dentro de duas estruturas.



5.3. Pipeline de Pré-processamento

As imagens passaram por um pipeline clássico de pré-processamento com o objetivo de realçar suas bordas e melhorar a qualidade visual antes da classificação automática. Esse fluxo foi composto pelas seguintes etapas, todas implementadas em Python de forma manual para preservar o vínculo com os conceitos trabalhados no primeiro projeto.

- Conversão para tons de cinza:

Cada imagem foi convertida para o formato em tons de cinza, eliminando os canais de cor e permitindo que o foco recaia unicamente sobre as variações de intensidade, aspecto fundamental na detecção de bordas:

```
def aplicar_filtro_sobel(img_path):
    img = Image.open(img_path).convert("L")
    matriz_img = np.array(img)
```

- Equalização de histograma:

Para melhorar o contraste das imagens, foi aplicada a equalização de histograma com a função *exposure.equalize_hist*, que redistribui os níveis de intensidade da imagem, destacando regiões escuras ou claras:

```
# Equaliza o histograma para melhor contraste
img_eq = exposure.equalize_hist(matriz_img)
matriz_eq = (img_eq * 255).astype(np.uint8)
```

- Suavização com filtro Gaussiano:

A suavização da imagem ajuda a reduzir ruídos que poderiam interferir na aplicação dos filtros de borda. Foi utilizado o filtro Gaussiano da biblioteca scipy:

```
# Suaviza com filtro gaussiano
matriz_suave = gaussian_filter(matriz_eq, sigma=1)
```

- Aplicação dos filtros de Sobel com convolução 2D:

Com a imagem suavizada, foram aplicados os filtros de Sobel nas direções horizontal e vertical. Para isso, foi usada a operação de **convolução**, baseada na função convolucao2d(), que por sua vez reutiliza a função correlacao2d() após a rotação do kernel:

```
def convolucao2d(imagem, kernel):
    kernel_rotacionado = np.flipud(np.fliplr(kernel))
    return correlacao2d(imagem, kernel_rotacionado)
```

A função correlacao2d() é o núcleo que realiza a multiplicação ponto a ponto entre o kernel e a vizinhança da imagem. Essa reutilização entre funções permite observar na prática a diferença entre as duas operações, como explorado no primeiro trabalho, e também traz modularidade ao código.

```
# Suaviza com filtro gaussiano
matriz_suave = gaussian_filter(matriz_eq, sigma=1)

sobel_h = np.array([[1, 0, -1],
                    [2, 0, -2],
                    [1, 0, -1]])
sobel_v = np.array([[1, 2, 1],
                    [0, 0, 0],
                    [-1, -2, -1]])

# Aplicar convolução usando sua função aprimorada
conv_h = convolucao2d(matriz_suave, sobel_h)
conv_v = convolucao2d(matriz_suave, sobel_v)
```

- Cálculo da magnitude e normalização e conversão:

Com as respostas dos filtros horizontal e vertical, foi calculada a **magnitude do gradiente**, combinando ambas as direções para destacar bordas de qualquer orientação.
A imagem foi então normalizada para valores entre 0 e 255 e convertida para RGB, garantindo compatibilidade com o modelo de rede neural:

```
magnitude = normaliza(np.hypot(conv_h, conv_v))

return Image.fromarray(magnitude).convert("RGB")
```

5.4. Modelo de Classificação — CNN

Para a etapa de classificação das imagens, foi empregada uma rede neural convolucional (CNN) de arquitetura simples, porém eficaz para tarefas de visão computacional. O modelo foi construído utilizando a API Sequential do Keras, e suas camadas foram definidas da seguinte forma:

```
modelo = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

- Camadas convolucionais (Conv2D): responsáveis por extrair padrões visuais relevantes da imagem, como formas, bordas e texturas, fundamentais para a diferenciação entre lesões benignas e malignas.

- Camadas de pooling (MaxPooling2D): realizam uma redução da dimensionalidade, preservando os principais padrões detectados pelas convoluções e contribuindo para a eficiência computacional do modelo.

- Camada de Dropout: utilizada para evitar o overfitting durante o treinamento, ao desativar aleatoriamente uma fração dos neurônios da camada anterior em cada época.

- Camadas densas (Dense): realizam a etapa final de classificação. A última camada contém um único neurônio com ativação sigmoide, adequada para problemas de classificação binária — neste caso, distinguir entre lesões benignas e melanoma.

Essa arquitetura permitiu explorar os efeitos do pré-processamento (com e sem o filtro de Sobel) no desempenho da CNN, mantendo o mesmo modelo base para garantir uma comparação justa entre os dois conjuntos de dados.

5.5. Avaliação

O modelo foi treinado por 15 épocas, utilizando:

- 80% das imagens para treino
- 20% para validação

Os gráficos de acurácia e perda serão apresentados na seção de Resultados.

6. Resultados

Após o treinamento do modelo por 15 épocas com o conjunto de imagens processadas com filtro de Sobel, foram obtidos os seguintes resultados de desempenho, apresentados nos gráficos abaixo:

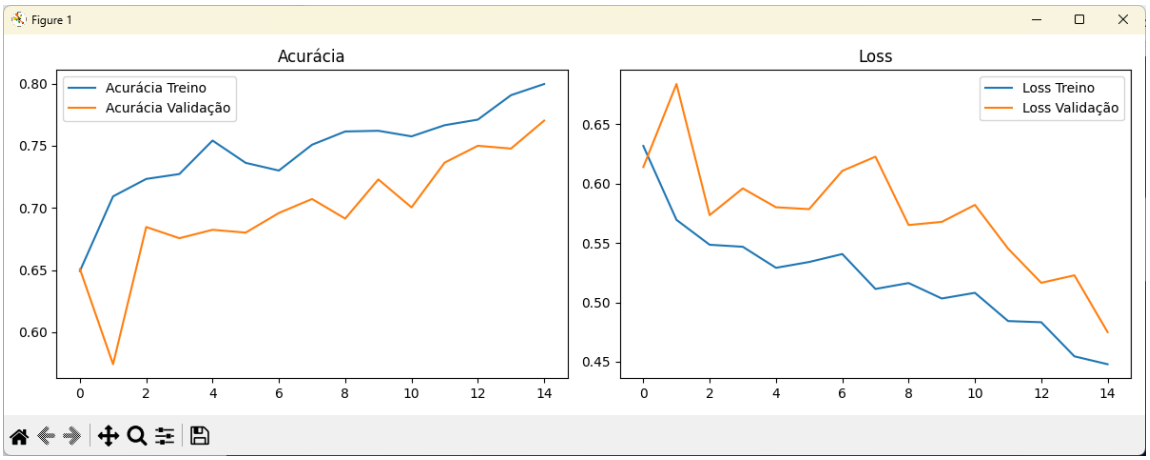


Gráfico da Esquerda – Acurácia:
Mostra a evolução da acurácia do modelo tanto nos dados de treino quanto nos de validação.

- A acurácia de treinamento começou em torno de 65% e chegou a 80% na última época.
- A acurácia de validação acompanhou de forma estável, partindo de cerca de 60% e alcançando aproximadamente 76%.

Gráfico da Direita – Perda (Loss):
Mostra a função de perda (binary crossentropy) ao longo do treinamento.

- A loss de treino caiu de forma contínua, de 0.6 até abaixo de 0.45.
- A loss de validação teve oscilações, mas também apresentou tendência de queda, indicando que o modelo está aprendendo de forma razoável.

Análise:

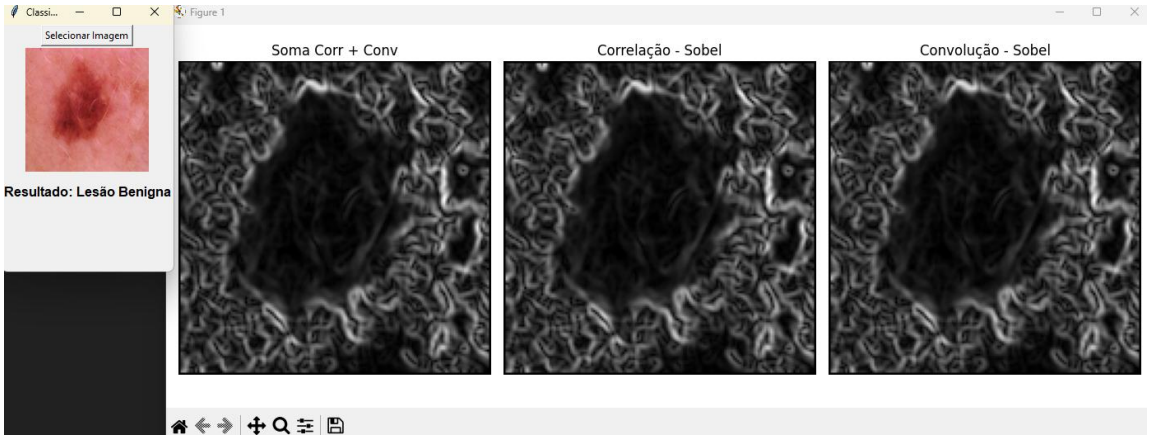
- A diferença entre treino e validação não é muito grande, o que sugere que o modelo não sofreu overfitting acentuado, mesmo com o uso do filtro de Sobel.

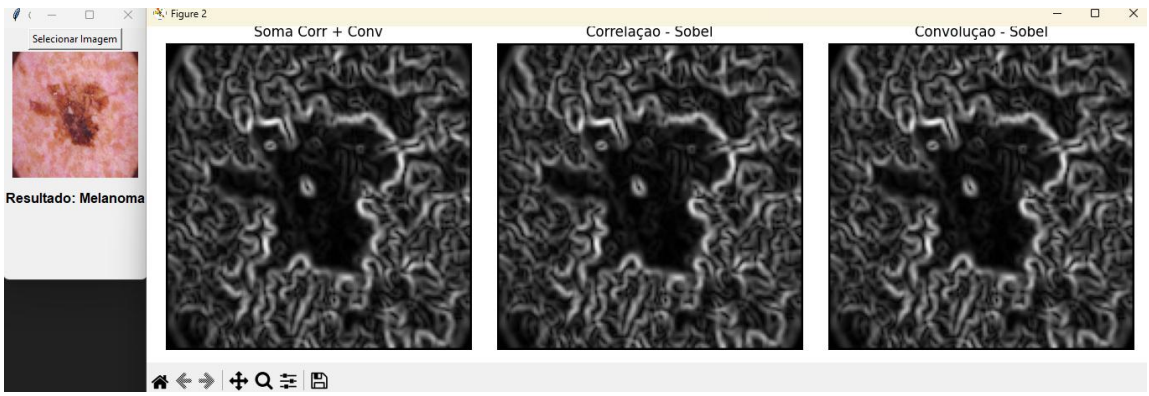
- O uso de técnicas clássicas de pré-processamento, como equalização, suavização e detecção de bordas com Sobel, contribuiu para destacar as regiões mais relevantes da imagem antes da entrada na CNN.
- A acurácia final de validação (~76%) é considerada boa para uma arquitetura simples e sem ajustes de hiperparâmetros avançados.

Visualização do Pipeline de Processamento: As imagens abaixo ilustram as principais etapas do pipeline aplicado a diferentes amostras do conjunto de dados:

- Imagem Original
- Correlação + Convolução
- Correlações com Filtro de Sobel (horizontal e vertical)
- Convolução com Filtro de Sobel (horizontal e vertical)

Essas transformações permitiram realçar bordas e padrões importantes das lesões, facilitando a detecção de características morfológicas relevantes.





Em ambos os exemplos, tanto para lesões benignas quanto para melanomas, é possível observar como as técnicas de realce de contraste**,** remoção de ruído e detecção de bordas contribuem para evidenciar a estrutura da lesão antes da entrada na rede neural.

7. Melhorias Implementadas

Em relação ao trabalho anterior, diversas melhorias foram aplicadas ao pipeline de processamento e análise das imagens:

- Filtro Sobel completo: enquanto a versão anterior aplicava apenas o filtro horizontal, esta versão utiliza ambos os filtros (horizontal e vertical), calculando a magnitude do gradiente.
- Pré-processamento mais robusto:
    - Equalização de histograma para realce de contraste.
    - Suavização com filtro Gaussiano para redução de ruído.
- Normalização das imagens filtradas: garantiu uniformidade nas intensidades e compatibilidade com redes neurais.
- Automatização do pipeline: o código foi modularizado, separando a etapa de pré-processamento da etapa de treinamento, facilitando reuso e manutenção.

8. Conclusão

A reutilização das operações de correlação e convolução com o filtro de Sobel, anteriormente aplicadas em contextos mais simples, como imagens de logotipos, demonstrou sua robustez também em um cenário mais complexo e clínico: a análise de lesões de pele.

Apesar da mudança significativa no domínio do problema, as técnicas fundamentais de processamento de imagem mantiveram sua relevância. Isso reforça não apenas seu valor acadêmico e didático, mas também sua aplicabilidade prática em tarefas reais de visão computacional e diagnóstico assistido.

Com a utilização de uma base de dados mais abrangente e composta por imagens médicas reais, foi possível construir um pipeline completo, da equalização e suavização até a extração de bordas com Sobel e posterior classificação automática com CNNs. Os resultados obtidos, como a acurácia de validação em torno de 76%, evidenciam que mesmo modelos simples, quando combinados com um bom pré-processamento, podem alcançar desempenho satisfatório na detecção de melanoma.

9. References

Gonzalez, R. C. and Woods, R. E. (2018). *Digital Image Processing*. 4th Edition, Pearson.

Tschandl, P., Rosendahl, C., & Kittler, H. (2018). *The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions*. [Kaggle Dataset]. Disponível em: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000. Acesso em: junho de 2025.

Ouhtit, A., Cherif, A., Alsmadi, M., & Al-Qaness, M. A. A. (2023). *A Hybrid Stacked Restricted Boltzmann Machine with Sobel Directional Patterns for Melanoma Prediction in Colored Skin Images*. Computers in Biology and Medicine, 161, 107070. https://doi.org/10.1016/j.compbiomed.2023.107070

OpenCV Documentation. “Image Filtering”. Disponível em: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html. Acesso em: abril de 2025.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). “Going deeper with convolutions”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 1–9.