Uso de uma *Support Vector Machine* para o reconhecimento de padrões por textura em imagens mamográficas

Lucas Santiago de Oliveira[1], **Rafael Amauri Diniz Augusto**[1], **Thiago Henriques Nogueira**[1]

[1]Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais (PUC-MG)

{lucas.oliveira.1201561, thiago.nogueira, radaugusto}@sga.pucminas.br

Abstract. *O presente trabalho descreve as técnicas implementadas para o desenvolvimento de um classificador de imagens de radiografia de mama com o fim de construir um classificador que possa ser utilizado para identificar a qual classe* BI-RADS *uma imagem de exame pertence.*
O trabalho implementa técnicas de pré-processamento de imagens e descritores de textura para encontrar características que auxiliam na identificação bem-sucedida da classe BI-RADS *de uma imagem. Para o fim de classificar a imagem, é implementada uma* Support Vector Machine *de* kernel *linear que recebe as imagens resultantes do pré-processamento. Com a implementação de ambas as técnicas foi utilizado um conjunto de 400 imagens para treinamento e avaliação do modelo, sendo obtida uma acurácia de aproximadamente 65%.*

1. Descrição do problema

O câncer de mama é uma doença que afeta predominantemente as mulheres devido a diversos fatores como a exposição prolongada do corpo aos hormônios femininos. De acordo com o Instituto Nacional de Câncer houveram 66.280 casos anuais de câncer de mama entre 2020 e 2022 (INCA). Por conta disto, é importante o acompanhamento desde cedo para evitar possíveis complicações. *Breast Imaging Reporting and Data System (BI-RADS)* é um sistema padronizado desenvolvido pela *American College of Radiology* a fim de auxiliar radiologistas a analisar exames mamográficos e detectar sinais prematuros de câncer de mama.

No sistema *BI-RADS* existem quatro categorias para a composição da mama: Tecido complemente composto por gordura, tecido fibroglandular difuso, tecido denso heterogêneo e tecido extremamente denso. Os resultados da radiografia da mama retornam imagens diferentes para cada uma das quatro categorias, e essas imagens podem ser utilizadas para treinar um classificador que realiza a identificação da classe *BI-RADS* de um exame a fim de auxiliar radiologistas e profissionais da saúde a terem os resultados do exame mais rapidamente. Tendo isso em vista, o presente trabalho tem como objetivo o desenvolvimento de um classificador capaz de identificar a qual classe *BI-RADS* uma determinada imagem de exame mamográfico pertence por meio de descritores de textura. Para classificar uma imagem já pré-processada foi utilizada uma *Support Vector Machine*. Em conjunto com as descrições das implementações, serão exibidos e avaliados os resultados encontrados na etapa de avaliação do classificador, bem como os seus pontos a serem melhorados.

2. Descrição das técnicas implementadas para a solução, principalmente do classificador

O grande objetivo deste trabalho foi o desenvolvimento de um software capaz de determinar a qual das quatro classes *BI-RADS* uma imagem de exame de mamografia pertence. Para este fim, o funcionamento do software é dividido em duas grandes etapas: a de obtenção e pré-processamento dos dados e a de treinamento.

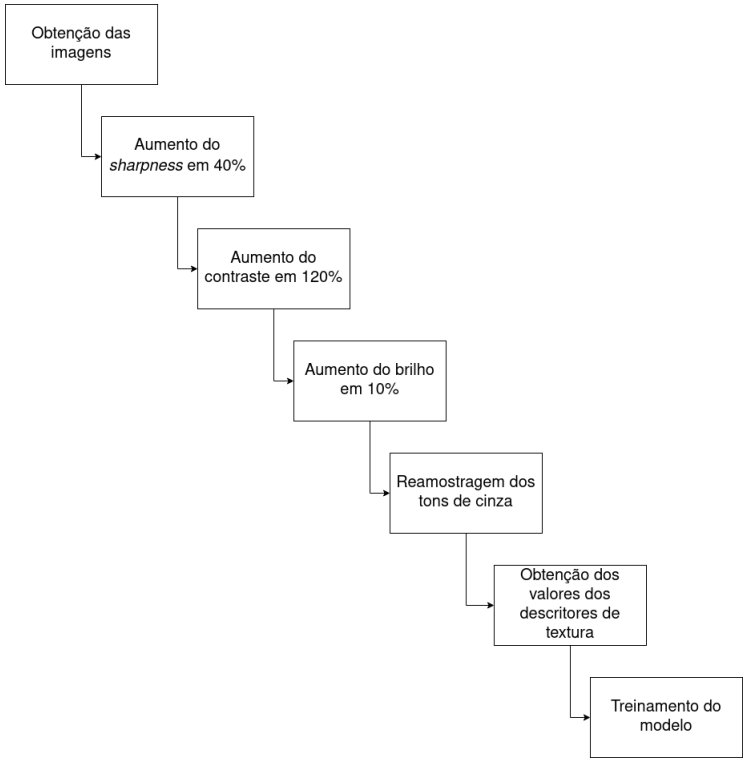

Figura 1. Fluxograma das etapas antes do treinamento do classificador

2.1. Obtenção dos dados e Pré-processamento

Como ilustrado na figura 1, a primeira etapa consiste na obtenção das imagens. Nessa etapa as imagens são separadas de forma aleatória entre um conjunto de treino e um conjunto de teste, mas ainda mantendo um balanceamento equivalente entre as classes *BI-RADS* de acordo com a porcentagem estipulada para treinamento.

A etapa de pré-processamento das imagens foi identificada como imperativa e de extrema relevância para o treinamento do modelo. Essa etapa tem como objetivo realçar características de cada imagem em ambos conjuntos de treino e teste a fim de aumentar a discrepância entre as imagens de classes diferentes, e assim obter um classificador com maior acurácia. Primeiro é utilizada a biblioteca *Pillow* para realçar o *sharpness* das imagens em 40% a fim de realçar mais as bordas e regiões limítrofes nas imagens. Após isso é feito um aumento de 120% no contraste da imagem a fim de alterar as tonalidades de cinza e realçar mais as diferenças de cores entre os pixels. Por fim, é feito um aumento do brilho em 10%. Para a etapa de pré-processamento também foi explorada a possibilidade da implementação de um filtro de suavização gaussiana, mas os testes rapidamente demonstraram que a resolução das imagens se mostrou muito baixa, e a implementação da suavização gaussiana suprimiu grande partes das sutilezas e detalhes finos das imagens. O filtro de utilização gaussiano pode ser habilitado no menu de opções avançadas da interface, mas por padrão ele está desabilitado. O grupo recomenda a sua utilização apenas em imagens que tenham resolução superior a 128x128 pixels.

Também é feita uma reamostragem dos tons de cinza da imagem de acordo com o valor definido pelo usuário na interface. Esta reamostragem é feita após a escolha

pelo usuário de um valor entre 1 e 32 tons de cinza, e também é implementada com a biblioteca *Pillow*. Após isto, são calculadas as matrizes de co-ocorrência circulares C1, C2, C4, C8 e C16 da imagem já pré-processada e obtidos os descritores de textura para cada uma delas. Finalmente, os valores obtidos para os descritores de textura das imagens no conjunto de treinamento são alimentados ao modelo de identificação como parte da etapa de treinamento.

2.2. Treinamento

A fim de obter um modelo de alta acurácia, foi utilizado o *Support Vector Machine* (*SVM*) com kernel linear e parâmetro de regularização (C) com valor 1.4. Tendo em vista as imagens que foram fornecidas e que compõem o dataset entende-se que a escolha de utilizar *SVM* é mais relevante à medida que foi identificada uma variação intra-classe relativamente alta nas imagens, e esta variação poderia facilmente afetar de forma negativa um modelo baseado em redes neurais. Outro grande motivador foi a preocupação na sensibilidade do classificador em relação a outliers. Após diversos testes, a kernel linear se mostrou resistente a outliers no conjunto de treinamento e apresentou os melhores resultados nas métricas de acurácia e especificidade, o que o configurou como a escolha ideal.

Para a etapa de treinamento, primeiro o classificador é criado com uma kernel linear e parâmetro de regularização com valor 1.4. Após isso, o classificador é alimentado com os valores dos descritores de textura obtidos de cada imagem no conjunto de dados de treinamento, bem como a categoria à qual esse grupo de descritores pertence. Após esse ponto, o classificador já é considerado treinado e está pronto para identificar a qual classe *BI-RADS* as imagens no conjunto de teste pertencem. Também foi implementada uma função de classificação de uma única imagem, que pode ser escolhida individualmente pelo usuário.

3. Bibliotecas utilizadas

Para o desenvolvimento do projeto foi utilizada a linguagem Python3, que pode ser instalada pelo site da linguagem. Complementando a linguagem em si, foram utilizadas as bibliotecas PySimpleGUI (PYSIMPLEGUI) Pillow (PILLOW), Numpy (NUMPY), Scikit-Image (SCIKIT-IMAGE), Matplotlib (MATPLOTLIB) e Scikit-learn (SCIKIT-LEARN). A biblioteca PySimpleGUI pode ser instalada com o comando `pip3 install PySimpleGUI` e foi utilizada para o desenvolvimento da interface gráfica e também tem como objetivo permitir o usuário definir certos parâmetros da aplicação como a quantidade de aumento do contraste, brilho, sharpness, quantidade de tons de cinza e a intensidade da suavização gaussiana das imagens no pré-processamento. A biblioteca Pillow pode ser instalada com o comando `pip3 install Pillow` e é responsável pela aplicação de todos os filtros e operações realizadas na etapa de pré-processamento, além de também ser responsável por fazer a leitura das imagens do disco. A biblioteca Numpy é instada com o comando `pip3 install numpy` e foi utilizada principalmente com o objetivo de aumentar a performance da aplicação, uma vez que, para conjuntos massivos de dados, arrays Numpy têm desempenho superior aos arrays normais encontrados no Python3. A biblioteca Scikit-Image pode ser instalada com o comando `pip3 install scikit-image` e ela teve como objetivo gerar as matrizes de co-ocorrência das imagens lidas com a biblioteca Pillow, bem como encontrar os valores para os descritores de textura utilizados.

A biblioteca matplotlib pode ser instalada com o comando `pip3 install matplotlib` e foi utilizada para a geração dos histogramas das imagens e para a criação da imagem da matriz de ocorrência. Por fim, a biblioteca Scikit-Learn pode ser instalada com o comando `pip3 install scikit-learn`. A biblioteca Scikit-Learn foi utilizada para acessar a implementação da SVM, bem como as funções de treino e classificação que a acompanham.

4. Medidas de tempo de execução para diversas imagens, descritores e hiperparâmetros do classificador.

Para realizar as medições de tempo, foi utilizado um computador com *Linux (Arch Linux com a kernel 5.17.7.arch1-2), Python* 3.10.4, um processador *AMD Ryzen 5 5600X* e *32 GB* de RAM. As imagens foram armazenadas em um SSD Adata XPG SX8100.

O tempo medido para a etapa de treinamento do modelo teve uma média de 0.005430 segundos. A etapa de separação das imagens nos conjuntos de treino e teste durou, em média, 1.533053 segundos. Por fim, a etapa de classificação das imagens no conjunto de teste durou, em média, 0.000656 segundos.

As imagens utilizadas para os testes abaixo foram escolhidas com o objetivo de apresentar os tempos observados para operações em imagens pertencentes às 4 diferentes classes *BI-RADS*. As imagens escolhidas foram: "1/p_d_left_cc(12).png", "2/p_e_left_cc(144).png", "3/p_f_left_cc(192).png" e "4/p_g_left_cc(204).png".

Para a imagem "1/p_d_left_cc(12).png" foram observados os seguintes tempos:

- Pré-processamento da imagem: 0.001435 segundos
- Cálculo das matrizes de co-ocorrências e extração dos descritores: 0.004100 segundos
- Classificação da imagem: 0.004216 segundos

Para a imagem "2/p_e_left_cc(144).png" foram observados os seguintes tempos:

- Pré-processamento da imagem: 0.001502 segundos
- Cálculo das matrizes de co-ocorrências e extração dos descritores: 0.004096 segundos
- Classificação da imagem: 0.004209 segundos

Para a imagem "3/p_f_left_cc(192).png" foram observados os seguintes tempos:

- Pré-processamento da imagem: 0.001419 segundos
- Cálculo das matrizes de co-ocorrências e extração dos descritores: 0.003937 segundos
- Classificação da imagem: 0.004052 segundos

Para a imagem "4/p_g_left_cc(204).png" foram observados os seguintes tempos:

- Pré-processamento da imagem: 0.001329 segundos
- Cálculo das matrizes de co-ocorrências e extração dos descritores: 0.003971 segundos
- Classificação da imagem: 0.004139 segundos

A *SVM* utilizou os descritores de textura: entropia, energia, homogeneidade, correlação e dissimilaridade. Testes feitos com outros descritores de textura não afetaram o tempo de execução de forma significativa (aproximadamente 10% a mais de tempo de execução para cada descritor adicionado), mas afetaram negativamente a acurácia do modelo. Por este motivo, foi decidido manter apenas os descritores citados anteriormente.

Os hiperparâmetros utilizados no classificador foram uma kernel linear e o valor do parâmetro de regularização igual a 1.4. Os tempos testados tanto para outros valores de kernel quanto para outros valores do parâmetro de regularização não se mostraram diferentes dos apresentados anteriormente. Os outros parâmetros testados foram a kernel ”rbf”, ”poly”e ”sigmoid”, e valores no intervalo entre 0.5 e 2.5 para o parâmetro de regularização.

De forma geral, os tempos observados para todas as operações que estão sendo feitas são considerados extremamente satisfatórios. Grande parte desse alto desempenho se deve à melhora do parâmetro **levels** da função **graycomatrix** da biblioteca *Scikit-Image*, que está sendo usada para gerar as matrizes de co-ocorrência. Esse parâmetro tem como objetivo realizar diversas otimizações nas matrizes ao saber qual o número máximo de tons de cinza da imagem, e como a imagem foi quantizada para um número específico, é possível saber quantos tons existem na imagem, ganhando desempenho.

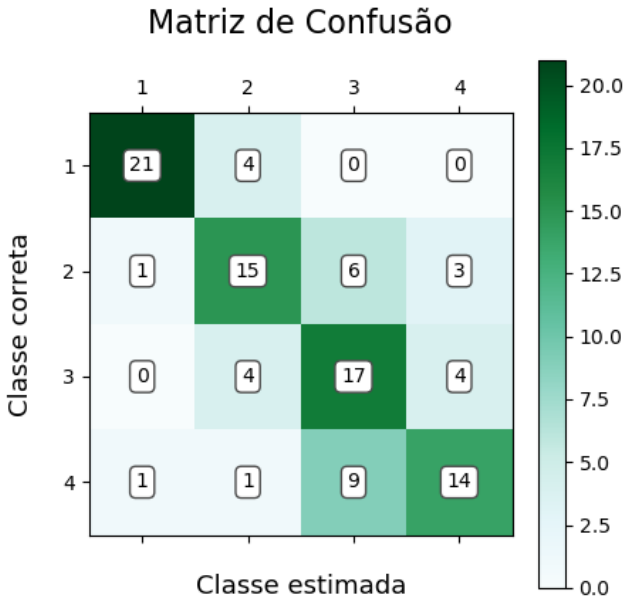

Figura 2. Exemplo de matriz de confusão para o classificador.

5. Resultados obtidos nos testes exemplos de erros e acertos dos métodos

Por conta do menu de opções avançadas, o classificador pode ter diferentes configurações que afetarão o desempenho e a taxa de acerto do modelo. Os resultados encontrados e demonstrados nesta seção são referentes às configurações padrão do classificador: aumento do sharpness das imagens em 40%, aumento do contraste em 120%, aumento do brilho em 10%, reamostragem para 32 cores, suavização gaussiana desligada e 75% das imagens sendo usadas para o treinamento.

A acurácia média para o classificador tem valor 0.65, com os valores oscilando entre 0.6 e 0.7. A especificidade média tem valor 0.33. Como pode ser visto na figura 2, os maiores erros do modelo são originados da identificação de imagens pertencentes às classes *BI-RADS* 2 e 4. Isso se deve à grande semelhança entre as imagens da classe 3 com as do grupo 2 e 4, e isso faz o modelo "acreditar" fortemente que imagens *BI-RADS* pertencentes às classes 2 e 4 pertencem à classe 3. Apesar da etapa de pré-processamento auxiliar significativamente o modelo em diferenciar melhor as classes, ainda existe uma certa dificuldade do classificador em fazer essa distinção. A fim de melhorar o classificador e aumentar a taxa de acerto, seriam necessárias imagens com maior resolução tanto para a etapa de treinamento quanto para a etapa de avaliação do modelo.

Referências

AMERICAN CANCER SOCIETY. About breast cancer. Disponível em: <https://www.cancer.org/cancer/breast-cancer/about/what-is-breast-cancer.html>. Data de acesso: 17 de maio de 2022.

INSTITUTO NACIONAL DE CÂNCER. Neoplasia maligna da mama feminina e colo do útero (taxas ajustadas). Disponível em: <https://www.inca.gov.br/estimativa/taxas-ajustadas/neoplasia-maligna-da-mama-feminina-e-colo-do-utero>. Data de acesso: 16 de maio de 2022.

MATPLOTLIB. Matplotlib API Reference. Disponível em: <https://matplotlib.org/stable/api/index>. Acesso em: 17 de maio de 2022.

NUMPY. NumPy Documentation. Disponível em: <https://numpy.org/doc/stable/ . Acesso em: 17 de maio de 2022.

PYSIMPLEGUI. Call Reference. Disponível em: <https://pysimplegui.readthedocs.io/en/latest/call%20reference/#element-and-function-call-reference>. Acesso em: 16 de maio de 2022.

PILLOW. Pillow Reference. Disponível em: <https://pillow.readthedocs.io/en/stable/ . Acesso em: 17 de maio de 2022.

SCIKIT-IMAGE. Scikit-Image API Reference. Disponível em: <https://scikit-image.org/docs/stable/api/api.html . Acesso em: 17 de maio de 2022.

SCIKIT-LEARN. Scikit-Learn API Reference. Disponível em: <https://scikit-learn.org/stable/modules/classes.html . Acesso em: 17 de maio de 2022