PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
ICEI - Instituto de Ciências Exatas e Informática
Programa de Graduação em Engenharia de Computação

Amanda Fernandes Oliveira
Guilherme Gomes Nogueira Chaves

TRABALHO PRÁTICO
Reconhecimento de padrões por textura em imagens mamográficas

Belo Horizonte
2021

Amanda Fernandes Oliveira
Guilherme Gomes Nogueira Chaves

TRABALHO PRÁTICO
Reconhecimento de padrões por textura em imagens mamográficas

Trabalho apresentado à disciplina de Processamento de Imagens, como requisito parcial de avaliação.

Orientador: Prof. Alexei Machado

Área de concentração: Processamento de Imagens

Belo Horizonte
2021

SUMÁRIO

1 INTRODUÇÃO

De acordo com o INCA (Instituto do Câncer), o câncer de mama é o tipo de câncer com maior incidência entre as mulheres no mundo. Se o tumor for diagnosticado precocemente, mais eficiente vai ser o tratamento.

O risco do desenvolvimento de câncer está relacionado com a densidade da mama, uma vez que mulheres com uma maior densidade mamária podem esconder lesões, levando o câncer a ser detectado tardiamente. A escala de densidade chamada BIRADS informa os radiologistas sobre a diminuição da sensibilidade do exame com o aumento da densidade da mama. BI-RADS definem a densidade como sendo quase inteiramente composta por gordura (densidade I), por tecido fibrobroglandular difuso (densidade II), por tecido denso heterogêneo (III) e por tecido extremamente denso (IV). Uma mama é considerada densa quando ela apresenta pouca gordura e uma quantidade grande de tecido glandular e fibroso. A densidade mamária depende de muitos fatores como número de filhos, peso e idade.

O uso da computação no processamento e análise de imagens na medicina tem sido muito utilizado em diagnósticos médicos e consequentemente contribuindo para a melhoria da saúde de milhares de pessoas.

”A classificação BI-RADS, inicialmente composta por quatro categorias descritivas de composição mamária, em sua quarta edição (Figura 1), associou a abordagem quantitativa com a divisão em quartis da composição percentual por tecido glandular na mama. Embora esta última versão implique alguma subjetividade, ela apresenta a vantagem de ser rotineiramente empregada em laudos mamográficos, o que facilita sua aplicação em estudos epidemiológicos. Comparadas as mamas categorizadas como extremamente densas com mamas lipossubstituídas, observou-se que há um aumento do risco relativo de quatro vezes para desenvolvimento do câncer de mama entre as primeiras.” (D’ORSI, 2003 apud MACCHETTI; MARANA, 2007).

Figura 1 – Tabela BI-RADS quarta edição - categorias finais

Categoria	Definições
1	Negativo
0	Necessita avaliação adicional
2	Achados benignos
3	Achados provavelmente benignos
4A	Baixa suspeição de malignidade
4B	Intermediária suspeição de malignidade
4C	Moderada suspeição de malignidade
5	Altamente sugestivo de malignidade

Fonte: (NASCIMENTO; SILVA; MACIEL, 2010)

Como pode ser visto na figura acima, existem variações das 4 classificações. Porém, no aplicativo desenvolvido, utilizamos as 4 mencionadas nas especificações deste trabalho.

O uso da informática no processamento e análise de imagens na medicina tem sido muito utilizado em diagnósticos médicos e consequentemente contribuindo para uma melhor saúde de milhares de pessoas. Para a análise das imagens resultantes da mamografia, propõe-se a implementação de um aplicativo que leia imagens de exames mamográficos e possibilite o reconhecimento automático da densidade da mama, utilizando técnicas de descrição por textura.

2 TÉCNICAS IMPLEMENTADAS

Algumas técnicas implementadas neste trabalhão serão discutidas a seguir:

- **main** - No arquivo principal do programa, um menu é criado utilizando da biblioteca tkinter. O menu fica localizado no topo do *frame*, abaixo há a região onde a imagem é exibida utilizando a classe implementada no arquivo Zoom.py, ao início é aberta uma imagem totalmente em alpha que serve apenas para dimensionamento da janela. Este arquivo faz todo o gerenciamento chamando as funções dos outros arquivos do sistema além de exibir alertas quando erros são cometidos pelo usuário. Nesta parte também há um controle sobre a imagem que é utilizada nas várias opções, como quantização e equalização, onde uma imagem que foi alterada de alguma forma é salva tanto no programa quanto em um arquivo temporário no diretório do projeto, no qual é apagado quando a opção "Sair"é clicada.

- **Zoom.py** - A parte de exibição e manipulação da imagem também é feito utilizando a biblioteca tkinter. No início, é realizado um processamento que verifica o tamanho da imagem, para o caso de ela ser muito grande a ponto da RAM não aguentá-la. Nesse caso, o programa irá lidar para que seja possível utilizar estas imagens tentando fazer com que as operações ocorram da forma mais suave possível e evitando erros como *DecompressionBombError* e *DecompressionBombWarning*. Também é utilizado nesta parte, uma variável chamada pyramid que suaviza o zoom em imagens muito grandes. Nesta variável é armazenada a imagem com uma grande quantidades de pixels diferentes (há um fator de redução sempre diminuindo a metade da quantidade de pixels na largura e altura até que a imagem tenha no mínimo 512 pixels de largura e altura). Outra implementação neste arquivo é a limitação para que a imagem não ultrapasse o espaço reservado e que não seja possível utilizar o zoom em uma região fora da imagem.

  Para aproveitar as implementações neste arquivo, a operação de selecionar a região de interesse foi feita neste arquivo fazendo a captura das coordenadas do *mouse* quando há um clique duplo no botão esquerdo do *mouse* e após esta captura é realizada a criação do retângulo que limita a região de interesse com este ponto de coordenadas no centro dele. Para realizar o *crop* da região de interesse, é necessário apertar a tecla 'c' ou 'C' do teclado. Uma observação é que é possível criar vários retângulos clicando várias vezes. Porém apenas o último retângulo criado será usado na hora de fazer o recorte da região. Quando há uma grande quantidade de retângulos na tela é possível apertar a tecla 'r' ou 'R' para que eles sejam apagados.

  De forma geral, a manipulação possível na imagem criada neste arquivo foi a seguinte: Zoom-in e Zoom-out utilizando o *scroll* do *mouse*; opção para arrastar a imagem segurando o botão esquerdo do *mouse* em cima da imagem e arrastando-a; e toda a operação para seleção e recorte da região de interesse.

- **Quantization.py** - Na opção de quantização, no início, é feito um tratamento na imagem para que ela tenha 3 canais e que seja em *Grayscale*. Em seguida é extraído o *shape* da imagem (largura, altura, canais). Por fim, é feito um *(*reshape) nesta imagem multiplicando a altura e a largura dela. Para fazer a quantização, foi utilizado o método de clusterização Kmeans da biblioteca Sklearn. Para isto foi feito um treinamento com amostras que ao final criou um modelo que tentava aproximar os tons de cinza parecido para que então fosse obtido a imagem com a quantidade

tons de cinza especificado pelo usuário. No final do processo é plotado o histograma da imagem original e da imagem quantizada mostrando a diferença. Como o usuário que irá entrar com a quantidade de tons de cinza desejado, é possível testar qualquer quantidade de tons de cinza disponíveis.

- **Equalization.py** - Ao inicio da equalização imagens com 3 canais são transformadas em imagens com apenas 2 canais. Neste arquivo foram disponibilizadas 3 opções de equalização, primeiro uma implementada com numpy utilizando das equações para equalização com histograma. a segunda e a terceira são opções já implementadas na biblioteca cv2/opencv, onde a segunda é também utilizando o método do histograma e a terceira utilizando clahe. Foi pensado em ter mais opções para que caso algum método não seja suficientemente satisfatório, seja possível tentar de outra forma para talvez ter um resultado melhor.

- **ResizeRegion.py** - O redimensionamento é bem simples ele apenas irá aplicar um fator de escala na imagem utilizando a biblioteca cv2/opencv. Como o usuário entra com o fator de escala existe a possibilidade de ter imagens com resoluções adicionais às que foram pedidas na especificação do trabalho.

- **Save.py** - O arquivo Save contem um método para salvar a imagem atual de trabalho em algum diretório escolhido pelo usuário.

- **TrainClassifier.py** - Para começar o treinamento, é pedido o diretório onde se encontra o *dataset* de imagens (Obs: As pastas de textura devem ser um subdiretório do diretório escolhido no dialog). Em seguida, todas as imagens são lidas e armazenadas. Após isso, a operação é iniciada passando por cada pasta de texturas, embaralhando-as e extraindo 75% delas. Em seguida, cada imagem destas texturas, diminuem as quantidades de tons de cinza de acordo com o definido pelo usuário (16 ou 32). E então é calculada a matriz de co-ocorrência da imagem atual sendo analisada. A seguir, são analisadas quais características foram selecionadas pelo usuário. Nas características selecionadas, há o cálculo, por exemplo, do contraste da matriz de co-ocorrência de níveis de cinza. Um adicional, é que também foram incluídas duas opções de característica a mais do que as especificadas que são energia e momento de Hu. Após todos os cálculos, utiliza-se uma SVM linear implementada na biblioteca Sklearn com os resultados obtidos. É realizado o treinamento da svm que mais tarde é utilizada para fazer o cálculo da matriz de confusão, acurácia, taxa de erro, sensibilidade e especificidade sobre os 25% de dados restantes do dataset. Uma observação é que tanto os cálculos de sensibilidade, especificidade e taxa de erro, foram calculados utilizando os resultados obtidos na matriz de confusão. Os modelos treinados são armazenados para serem depois utilizados na parte de classificação das imagens. Outra operação é a plotagem da matriz de confusão obtida com um mapa de calor mostrando onde houveram mais acertos, na tela mostrada também aparecem os resultados da acurácia, taxa de erro e os demais já mencionados, com a adição do tempo de execução para realizar este treinamento.

- **Classifier.py** - Na parte da classificação é realizado o mesmo processo para o cálculo da matriz de co-ocorrência e cálculo das características selecionadas em relação a esta matriz, então é feito a previsão do modelo já treinado e então e exibida uma mensagem com a textura prevista pelo modelo, caso tenha várias característica todas as previsões são mostradas e também o tempo de execução da classificação.

3 BIBLIOTECAS UTILIZADAS

Para o desenvolvimento do aplicativo, foi utilizado Pyhton na versão 3.8.5. Abaixo estão listadas as principais bibliotecas utilizadas para o desenvolvimento deste trabalho:

- **scikit-image** - versão: 0.18.1
- **easygui** - versão: 0.98.2
- **matplotlib** - versão: 3.4.1
- **numpy** - versão: 1.19.3
- **opencv-python** - versão: 4.5.1.48
- **tk** - versão: 8.6.9
- **pandas** - versão: 1.2.4
- **scikit-learn** - versão: 0.23.2
- **Pillow** - versão: 8.0.1
- **seaborn** - versão: 0.11.1

Também foram utilizadas bibliotecas nativas do Python como a *time*, *math*, *unicode* e outras.

4 INSTRUÇÕES PARA INSTALAÇÃO, COMPILAÇÃO E USO

O programa foi testado e executado na versão 3.8.5 do Python. Também faz-se necessário a instalação das versões das bibliotecas apresentadas no capítulo anterior (talvez seja possível executar em versões mais recentes destas bibliotecas, porém não foi testado). Após a compilação aparecerá o menu, para a execução dos processamentos é necessário entrar com uma imagem, o único não necessário é a opção de treinamento do classificador.

Para seleção da região de interesse basta dar um duplo clique com o botão esquerdo do *(*mouse) na região separada para a exibição da imagem. Para o recorte desta imagem a tecla 'c' ou 'C' deve ser apertada, caso queira apagar o retângulo é só apertar a tecla 'r' ou 'R'. Como já mencionado, é possível criar vários retângulos, porém apenas as coordenadas do último criado é utilizado.

Na seção de opções, há a quantização que pede a quantidade de tons de cinza que o usuário quer, também a a opção de redimensionamento que é feita com o fator de escala em porcentagem que o usuário entrar e por último a opção de quantização que oferece as opções já mencionadas anteriormente no documento. Na aba *file* estão as opções para abrir uma imagem que pode ser uma região de interesse ou não, a opção de salvamento da imagem em trabalho atual e a opção de sair do programa.

Por último,, há a aba de texturas onde é possível escolher as opções de treinamento. Por padrão as opções com 32 níveis de tons de cinza e a característica "contraste"são selecionadas. Após o treinamento, é possível realizar a classificação da imagem atual que estiver aberta no programa. Caso queira, o usuário pode realizar a operação de seleção da região de interesse mencionada para selecionar uma região específica de uma imagem antes da classificação. Quando estiver tudo como o usuário preferir, é só clicar na opção de classificar na opção de texturas, no qual o cálculo é realizado e o resultado é mostrado na tela.

5 RESULTADOS OBTIDOS

No treinamento do classificador, 3 características tiveram resultados parecidos: as características foram o contraste, entropia e homogeneidade. Elas apresentaram acurácia com uma media entre 55% e 62% em alguns casos essa acurácia pode ser um pouco maior ou menor. Já a característica de entropia teve um resultado mediano, com uma acurácia entre 45% e 50%. Os piores resultado ocorreram utilizando a característica de momentos de Hu que teve uma média de acurácia menor que 40%.

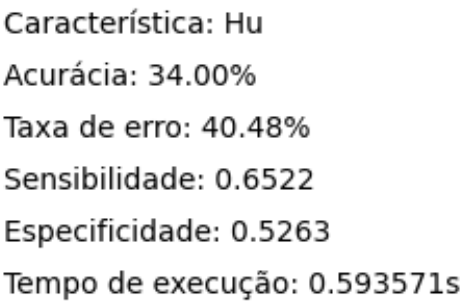
A sensibilidade em todos os casos ficou bem próxima de 1 o que é um ótimo resultado. A especificidade variou bastante, em certos casos ela ficava bem próximo de 0 e em outros ela ficou bem próxima de 1.

O tempo de execução foi bem rápido em todos os casos, no treinamento teve-se em média um tempo em torno de 0,5s quando apenas uma característica era selecionada. O único caso em que o tempo passou um pouco de 1,5s foi quando todas as características foram selecionadas. Na classificação o tempo foi extremamente rápido, tendo a resolução do problema em um tempo muito satisfatório.

Vale ressaltar que o tempo de execução pode variar de acordo com as configurações do computador. Os testes foram realizados em um computador com um processador Intel i7-7500U, uma RAM de 8Gb e uma placa de vídeo Intel Graphics 620. O tempo de execução pode diminuir drasticamente se os testes forem feitos num computador de primeira linha.

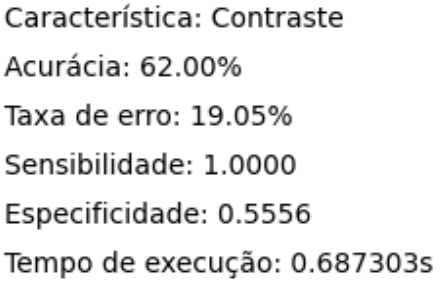
Abaixo estão imagens com os resultados obtidos.

Figura 2 – Métricas do treinamento - Hu



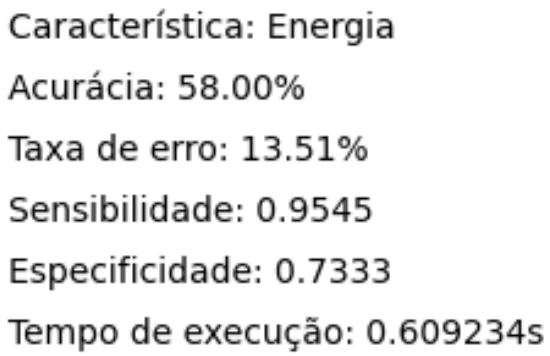
Fonte: Elaborado pelo autor

Figura 3 – Métricas do treinamento - Contraste



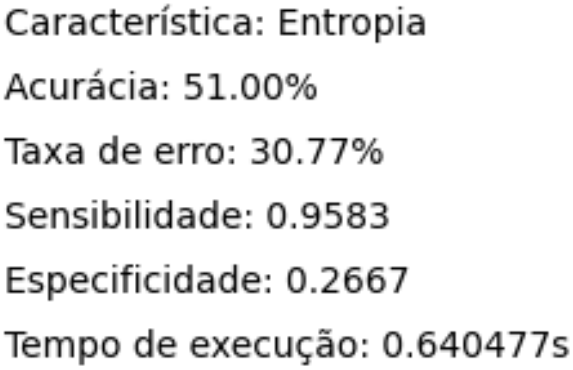
Fonte: Elaborado pelo autor

Figura 4 – Métricas do treinamento - Energia



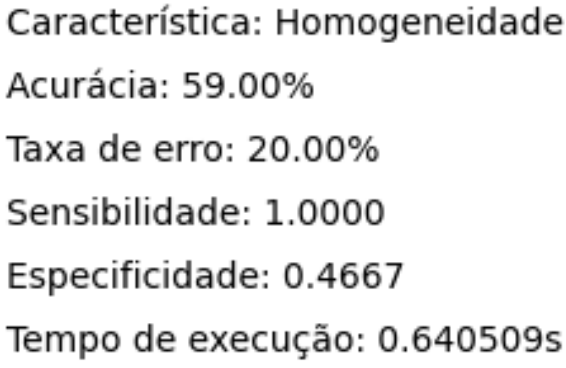
Fonte: Elaborado pelo autor

Figura 5 – Métricas do treinamento - Entropia



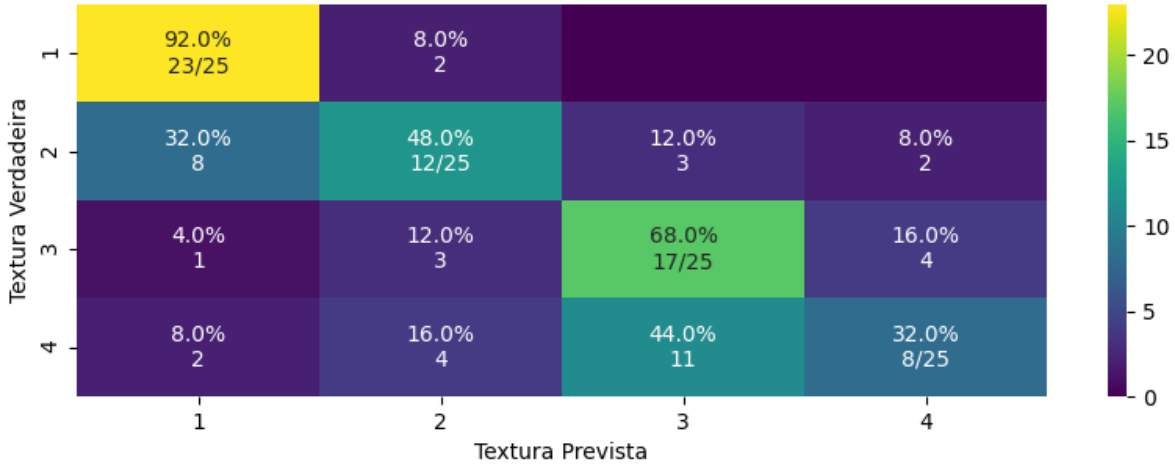
Fonte: Elaborado pelo autor

Figura 6 – Métricas do treinamento - Homogeneidade



Fonte: Elaborado pelo autor

Figura 7 – Exemplo da matriz de confusão da característica contraste



Fonte: Elaborado pelo autor

Figura 8 – Tempo de execução com todas as características selecionadas

Tempo de execução: 1.124776s

Fonte: Elaborado pelo autor

Figura 9 – Tempo de execução da classificação com todas as características

Tempo de execução: 0.010087728500366211s

Fonte: Elaborado pelo autor

6 CONSIDERAÇÕES FINAIS

Ao analisar os resultados, pode-se perceber que o algoritmo teve resultados razoáveis da acurácia para o contraste, energia e homogeneidade. Pode-se perceber também que o tempo de execução permaneceu baixo, o que significa que mesmo utilizando imagens de alta resolução, é possível continuar tendo um bom desempenho. Além disso, a sensibilidade se destacou na homogeneidade obtendo um valor ótimo. Porém, a taxa de erro esteve elevada em quase todas as métricas de treinamento. Seria ideal investigar a possível causa deste problema.

Na matriz de confusão, foi possível ver que há um grande número de acertos na textura BI-RADS 1 em relação às outras, em outros testes ainda foi possível obter resultados melhores onde a quantidade de acertos foi 100%.

De modo geral, houve uma certa dificuldade na implementação do Zoom e da região de interesse. Houveram complicações no momento de obter as coordenadas do ponto correto na imagem. Outra dificuldade foi no momento do recorte da região de interesse que acabava recortando uma parte errada da imagem quando era realizado um zoom ou movimento em sua posição. Todos estes problemas eventualmente foram solucionados e neste trabalho obteve-se um bom resultado.

Referências Bibliográficas

D'ORSI, C. J. *Breast Imaging Reporting and Data System: breast imaging atlas: mammography, breast ultrasound, breast MR imaging*. [S.l.]: ACR, American College of Radiology, 2003. 3

MACCHETTI, A. H.; MARANA, H. R. C. Densidade mamográfica como fator de risco para o câncer de mama. *Revista Brasileira de Ginecologia e Obstetrícia*, SciELO Brasil, v. 29, n. 10, p. 493–496, 2007. 3

NASCIMENTO, J. H. R. d.; SILVA, V. D. d.; MACIEL, A. C. Acurácia dos achados mamográficos do câncer de mama: correlação da classificação bi-rads e achados histológicos. *Radiologia Brasileira*, SciELO Brasil, v. 43, n. 2, p. 91–96, 2010. 3