Segmentação e classificação de imagens mamográficas

Lucas Saliba[1], **Ygor Matheus Lacerda de Melo**[2]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Av. Dom José Gaspar, 500 – 30.535-610 – Belo Horizonte – MG – Brazil

1. Introdução

O câncer de mama é uma das doenças mais prevalentes entre mulheres em todo o mundo, superando o câncer de pulmão em número de casos, de acordo com a Organização Mundial da Saúde (OMS) em 2020. A densidade mamária pode ser um indicador de risco para o desenvolvimento do câncer de mama, pois mulheres com maior densidade podem ter lesões ocultas que podem ser indicativas da doença. Infelizmente, essas lesões geralmente são descobertas tardiamente, resultando em um agravamento do problema. Portanto, é crucial detectar o câncer o mais cedo possível, pois isso aumenta significativamente as chances de cura.

A densidade da mama está diretamente relacionada ao risco de desenvolvimento do câncer. Mulheres com maior densidade mamária podem ter lesões que passam despercebidas, levando a um diagnóstico tardio da doença. O American College of Radiology desenvolveu uma escala de densidade chamada BIRADS, que fornece informações aos radiologistas sobre a diminuição da sensibilidade dos exames à medida que a densidade da mama aumenta.

A motivação deste trabalho é realizar uma análise visual das imagens disponíveis resultantes de exames de mamografia, que é a principal ferramente de rastreamento do câncer de mama. É essencial que a densidade mamária seja levada em consideração durante a avaliação, a fim de garantir uma detecção precoce e um tratamento adequado para o câncer de mama.

O objetivo deste trabalho é o desenvolvimento de um software que seja capaz de treinar uma rede neural através de imagens mamográficas a fim de classificar qual o tipo de BIRADS pertence a imagem. Para obtermos um resultado satisfatório, é necessário a alta precisão de detecção da rede neural, diminuindo as taxas de erro e aumentando as taxas de acerto.

TODO: Conferir numeração abaixo

O restante deste artigo está organizado da seguinte forma: a Seção 2 trata da descrição de implementação do software e informações de bibliotecas utilizadas; a Seção 3 lista as técnicas implementadas para realização da classificação e método de segmentação das imagens; a Seção 4 discute os resultados obtidos; a Seção 5 mostra o relatório do Github e, por fim, a Seção 6 as referências bibliográficas.

2. Descrição do Projeto

O projeto foi desenvolvido utilizando a linguagem Python (versão 3.11.3 disponível para download no site oficial) e o Visual Studio Code como IDE. A seguir, forneceremos uma breve descrição sobre as bibliotecas aplicadas em nosso projeto.

2.1. Bibliotecas utilizadas

A biblioteca Tkinter é uma biblioteca que já vem com a instalação do Python, utilizamos na criação da interface e importamos a função ”filedialog”. Para auxiliar no processamento das imagens e execução de tarefas de visão computacional, utilizamos a biblioteca CV2 do OpenCv, além da função ”ImageTk”da biblioteca Pillow que realiza a junção das bibliotecas Pillow e Tkinter. A biblioteca Numpy também foi importada para auxiliar na manipulação das imagens. Já a biblioteca OS, é um módulo que auxilia na criação de scripts e a biblioteca TQDM utilizamos para adicionar barras de progresso.

```
7   import cv2
8   import PIL
9   import PIL.ImageTk
10  import tkinter
11  import tkinter.filedialog
12  import os
13  import time
14  import tqdm
15  import numpy as np
16  import tensorflow as tf
```

Figura 1. Bibliotecas utilizadas no desenvolvimento da aplicação

A biblioteca “Time” tem o objetivo de medir o tempo de execução do treinamento e da classificação da rede neural. Se tratando da classificação, esta medição é utilizada tanto para medir o tempo de execução da classificação da imagem completa quanto da classificação de uma região específica selecionada pelo usuário. Por fim, utilizamos a biblioteca de rede neural TensorFlow, voltada para o aprendizado de máquina, com o objetivo de criar e treinar a rede neural para detectar os padrões e classificar as imagens no tipo de BIRADS pertencente.

TODO: Conferir se foi utilizada mais alguma biblioteca

3. Técnicas Implementadas

Ao iniciar o software, deve-se selecionar a imagem para ser carregada no projeto. Após realizar a abertura da imagem, é permitido ao usuário que aplique zoom, altere o contraste e realize a segmentação através de uma barra deslizante selecionando o valor de tons de cinza correspondente.

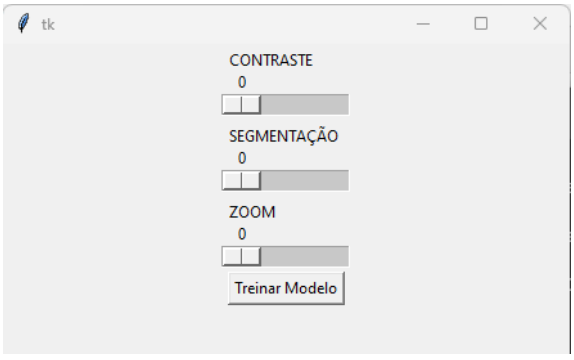

Figura 2. Janela de visualização ao inciar o software

TODO: Conferir se eliminamos a TAG (placas) nas imagens para o texto abaixo

Ao realizar a leitura das imagens, a função 'apply_threshold' aplica a segmentação binária de forma automática nas mesmas, transformando os elementos de fundo e anotações com valor preto.

No diretório 'mamografia' estão presentes as imagens utilizadas para realizar o treinamento dos classificadores de BIRADS da nossa rede neural. Realizamos a leitura da base através da função 'read_data', separando as imagens de treino e de teste. Ao realizamos a leitura de cada imagem, a função 'flip_data' realiza o espelhamento e a função 'equalize_data' a equalização do seu histograma, logo, para cada imagem, é gerado 4 variações: original e espelhada X original e equalizada.

Para realizar a construção do software que seja capaz de identificar os tipos de BIRADS através de uma rede neural treinada, foi necessário a implementação de um descritor de texturas

TODO: Adicionar a descrição de Haralick para rede a rede neural e a descrição das matrizes de coocorrência (Homogeneidade, entropia, energia, contraste, correção, dissimilaridade e tempo de execução) - itens abordados no subtópico DESCRITORES

A construção da nossa rede neural foi baseada nos descritores de textura como parte fundamental do processo. Para o treinamento, empregamos as bibliotecas disponíveis no TensorFlow e classificamos as imagens em quatro classes: BIRADS I, BIRADS II, BIRADS III e BIRADS IV.

TODO: Descrever as funções e adicionar prints da parte do treinamento da rede neural

3.1. Medidas

Para de iniciarmos a medição do tempo de execução do algoritmo, realizamos uma análise do custo na resolução do nosso problema, que consiste no treinamento e classificação de imagens. Com base nessa análise, teremos o número de vezes que cada etapa do algoritmo foi executada, visando obter uma medida final precisa e representativa. Ao compreendermos completamente o tempo e a frequência de execução de cada parte do algoritmo, podemos estabelecer uma medida de referência confiável para avaliar o desempenho do sistema em relação ao tempo de processamento.

TODO: é melhor medir a complexidade de cada método presente no código e somar com a medida da rede neural ou somente colocar o tempo de execução da rede neural? PIOR CASO e MELHOR CASO.

3.2. Descritores

TODO: Verificar na matriz de coocorrência se os descritores batem com os listados abaixo

No nosso projeto, utilizamos descritores baseados em matrizes de coocorrência, que incluem: homogeneidade, entropia, energia, contraste, dissimilaridade e correlação. Esses descritores desempenham um papel crucial no treinamento da rede neural, fornecendo informações importantes sobre as características das imagens. A seguir, descrevemos cada um desses descritores e sua contribuição para o processo de treinamento:

- Contraste: utilizamos para avaliar a variação local dos níveis de cinza em uma imagem. Ele reflete a diferença entre os valores dos pixels vizinhos, fornecendo informações sobre a textura e os detalhes presentes na imagem..
- Homogeneidade: mede a distribuição dos tons de cinza em relação à diagonal da matriz de coocorrência. Ela fornece informações sobre a uniformidade ou regularidade dos elementos na imagem, destacando regiões com padrões semelhantes.
- Entropia: é uma medida da informação contida nas imagens. Ela avalia a complexidade e a imprevisibilidade dos padrões de tons de cinza na imagem, sendo útil para identificar imagens com alto grau de aleatoriedade ou desordem.
- Energia: medida que retorna a soma dos elementos elevados ao quadrado dentro da matriz de coocorrência de tons de cinza. Ela reflete a intensidade geral dos padrões presentes na imagem, destacando áreas com maior concentração de detalhes.
- Correlação: a correlação mede o grau de correlação entre um pixel e seus vizinhos na imagem. Essa medida compara a similaridade dos padrões entre os pixels, permitindo identificar áreas com alta correlação e padrões repetitivos.
- Dissimilaridade: mede a diferença entre os tons de cinza presentes na matriz de coocorrência. Ela fornece informações sobre a diversidade e a variação dos padrões na imagem, destacando áreas com tons de cinza distintos.

Esses descritores, combinados, oferecem uma representação abrangente das características das imagens, auxiliando no processo de treinamento da rede neural e na extração de informações relevantes para a classificação e análise das imagens.

3.3. Hiperparâmetros do classificador

O resultado foi obtido ao considerarmos uma métrica que avalia a precisão do modelo através da acurácia e a perda (loss) do modelo. Durante o treinamento, utilizamos os ”epochs”, que representam os ciclos completos nos quais a rede neural percorre todos os dados de treinamento. TODO: Conferir o texto a seguir Além disso, definimos o tamanho do lote (batch size) da rede neural, e utilizamos 1024 nós ou neurônios para testar as bases de dados.

TODO: EXEMPLO DE TEXTO Para efeitos comparativos, também realizamos medições de tempo em minutos. Essas medidas nos auxiliaram na análise comparativa do desempenho dos modelos em relação ao tempo de processamento. Por fim, dividimos a base de dados em quatro classes, correspondendo aos quatro tipos de BIRADS. Essa divisão nos permitiu uma análise mais precisa e específica durante o treinamento e avaliação do modelo, considerando as diferentes características e padrões presentes em cada classe de BIRADS.

4. Resultados obtidos

Ao realizarmos o treinamento e classificação da base de dados, levamos em consideração os critérios de precisão e eficiência de processamento. O tempo total para rodar a base de dados foi de TODO: Adicionar tempo.

TODO: Conferir o texto abaixo. Durante o treinamento, realizamos diversos testes (medidas de cada epoch?)

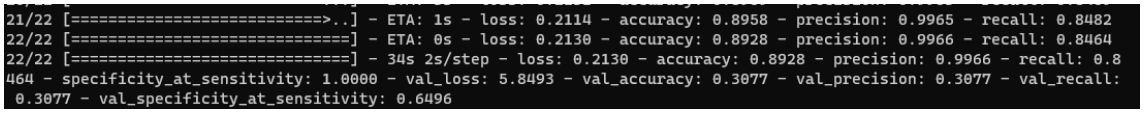

Figura 3. Visualização dos valores durante a execução de cada epoch

TODO: EXEMPLO: Geramos também uma matriz de confusão da base de dados (imagem a seguir) e como visto, os BIRADS 1 e 2 estão sendo mais reconhecidos do que os BIRADS 3 e 4.

Após o processo de treinamento, o algoritmo armazena o modelo encontrado em um arquivo chamado ”conv”. É exibido na janela de execução, os valores referentes a classificação binária (I+II x III+IV) e da classificação de 4 classes (IxIIxIIIxIV) como mostrado na figura abaixo:

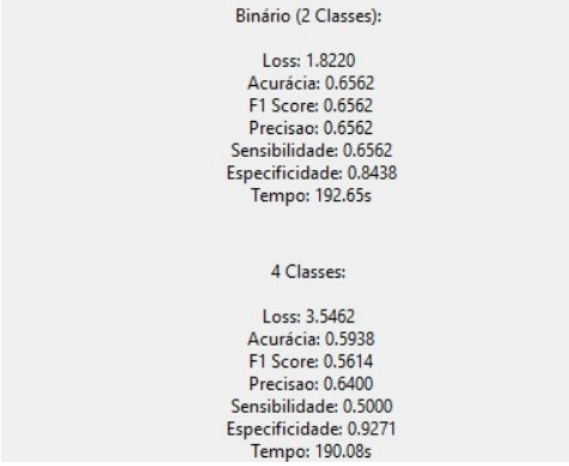

Figura 4. Visualização dos valores do processo de treinamento e classificação

TODO: Descrever as métricas.

5. Relatório

6. Referências Bibliográficas

Referências