Universidade Estadual de Londrina
Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Jean Marcel Faria Tonin

Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais

Londrina
2023

Universidade Estadual de Londrina

Centro de Tecnologia e Urbanismo
Departamento de Engenharia Elétrica

Jean Marcel Faria Tonin

Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais

Trabalho de Conclusão de Curso orientado pelo Prof. Dr. Ernesto Fernando Ferreyra Ramirez intitulado "Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais" e apresentado à Universidade Estadual de Londrina, como parte dos requisitos necessários para a obtenção do Título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Ernesto Fernando Ferreyra Ramirez

Londrina
2023

Ficha Catalográfica

Jean Marcel Faria Tonin
Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais - Londrina, 2023 - 93 p., 30 cm.
Orientador: Prof. Dr. Ernesto Fernando Ferreyra Ramirez
1. Diagnóstico assistido por Computador. 2. Segmentação. 3. Reconhecimento de Padrões. 4. Correlação Linear. 5. Histograma.
I. Universidade Estadual de Londrina. Curso de Engenharia Elétrica. II. Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais.

Jean Marcel Faria Tonin

Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Engenharia Elétrica.

Comissão Examinadora

Prof. Dr. Ernesto Fernando Ferreyra Ramirez
Universidade Estadual de Londrina
Orientador

Prof. Dr. Leonimer Flávio de Melo
Universidade Estadual de Londrina

Prof. Dr. Francisco Granziera Júnior
Universidade Estadual de Londrina

Londrina, 17 de maio de 2023

Dedico este trabalho aos que sempre estiveram ao meu lado e foram minha base durante toda essa jornada. Em especial à minha amada companheira e aos meus pais. Sem o amor e a presença de vocês, essa conquista não seria possível.

Agradecimentos

Gostaria de expressar minha sincera gratidão às pessoas que tornaram possível a realização deste trabalho.

Primeiramente, agradeço à minha querida companheira, Giulia, que esteve presente em todos os momentos, apoiando-me e incentivando-me em todas as etapas. Também não posso deixar de agradecer à minha família: meus pais, meu irmão e meus avós, que sempre estiveram ao meu lado, me dando suporte e motivação.

Aos meus amigos, que sempre me deram força e me proporcionaram inúmeros momentos de descontração e alegria durante essa jornada, mesmo nos dias mais difíceis.

Agradeço ao meu orientador, Prof. Ernesto, pela sua orientação, paciência e contribuições valiosas para o desenvolvimento deste trabalho. Suas sugestões e críticas foram fundamentais para o aprimoramento do conteúdo. Gostaria de estender meu agradecimento à UEL, que me proporcionou um ambiente propício para o aprendizado, além de recursos e ferramentas essenciais para a realização deste trabalho.

Por fim, agradeço a Deus, que me deu forças para enfrentar os desafios e superar as dificuldades. Sem Sua orientação, sabedoria e amor nada disso seria possível.

"Você pode encarar um erro como uma besteira a ser esquecida
ou como um resultado que aponta a uma nova direção"
Steve Jobs

Jean Marcel Faria Tonin. **Aplicação de Métodos de Processamento de Imagens para Diagnóstico e Localização de Tumores Cerebrais**. 2023. 93 p. Trabalho de Conclusão de Curso em Engenharia Elétrica - Universidade Estadual de Londrina, Londrina.

Resumo

O presente trabalho tem como objetivo o desenvolvimento de um método para auxiliar no diagnóstico e localização de tumores cerebrais a partir de imagens de ressonâncias magnéticas. Foi elaborada uma metodologia de diagnóstico assistido por computador a partir da média de correlações lineares entre o histograma da amostra a ser classificada e o histograma de padrões ouro positivos e negativos. Foram alcançados os seguintes valores nas métricas de desempenho: $80,00\%$ de acurácia, $84,21\%$ de sensibilidade e $73,47\%$ de especificidade em experimentos com 250 amostras. A localização é desenvolvida a partir da segmentação da imagem de entrada com a função `imadjust()`, a qual teve que ser adaptada manualmente de Matlab para Python. Também é utilizado um algoritmo iterativo para ajustar o intervalo de segmentação a partir da proporção de pixels brancos. Por fim, é elaborada uma interface gráfica do usuário, para facilitar a utilização do programa, que funciona de maneira autônoma, ou seja, sem a necessidade de outros arquivos.

Palavras-Chave: 1. Diagnóstico assistido por Computador. 2. Segmentação. 3. Reconhecimento de Padrões. 4. Correlação Linear. 5. Histograma.

Jean Marcel Faria Tonin. **Application of Image Processing Methods for Diagnosis and Localization of Brain Tumors**. 2023. 93 p. Monograph in Electrical Engineering - Londrina State University, Londrina.

Abstract

The present work aims to develop a method to aid in the diagnosis and location of brain tumors from magnetic resonance images. A computer-aided diagnosis methodology was created based on the average of linear correlations between the histogram of the sample to be classified and the histogram of positive and negative gold standards. The following values were achieved in the performance metrics: 80.00% of accuracy, 84.21% of sensitivity and 73.47% of specificity in experiments with 250 samples. The localization is developed from the segmentation of the input image using the `imadjust()` function, which had to be manually adapted from Matlab to Python. An iterative algorithm is also used to adjust the segmentation based on the white pixel ratio. Finally, a graphical user interface is designed to facilitate the usage of the program, which works in an autonomous mode.

Key-words: 1. Computer-aided Diagnosis. 2. Segmentation. 3. Pattern Recognition. 4. Linear Correlation. 5. Histogram.

Lista de ilustrações

Lista de tabelas

Lista de quadros

Lista de Siglas e Abreviaturas

ANN	*Artificial Neural Network* (Rede Neural Artificial)
CAD	*Computer-aided Diagnosis* (Diagnóstico Assistido por Computador)
CCL	*Connected Component Labeling* (Rotulagem de Componentes Conectados)
DWT	*Discrete Wavelet Transform* (Transformada Wavelet Discreta)
FN	Falso Negativo
FP	Falso Positivo
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
IA	Inteligência Artificial
ICA	*Independent Component Analysis* (Análise de Componentes Independentes)
INCA	Instituto Nacional do Câncer
IPT	*Image Processing Toolbox*^{TM}
K-NN	*K-Nearest Neighbors* (K-ésimo Vizinho Mais Próximo)
MEC	Memória Endereçável por Conteúdo
OMS	Organização Mundial da Saúde
PCA	*Principal Component Analysis* (Análise de Componentes Principais)
PCC	*Pearson's Correlation Coefficient* (Coeficiente de Correlação de Pearson)
PIL	*Python Imaging Library*
RBFBK	*Radial Basis Function Based Kernel* (Kernel Baseado em Função de Base Radial)
RM	Ressonância Magnética
SNC	Sistema Nervoso Central
SOM	*Self-organizing Maps* (Mapas Auto-organizáveis)
SVM	*Support Vector Machine* (Máquina de Vetores de Suporte)
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

Sumário

1 Introdução

1.1 Motivação

Os tumores cerebrais são um dos tipos mais complexos e desafiadores de neoplasias (que se transformam em câncer quando malignas) e que podem afetar pessoas de todas as idades e origens. Esses tumores se desenvolvem a partir de células anormais no cérebro ou em suas proximidades e podem causar uma variedade de sintomas que afetam a função cerebral, como alterações de comportamento, dificuldade de movimento e perda de memória. Os tumores cerebrais são uma das principais causas de morte relacionadas a câncer em todo o mundo, principalmente entre adolescentes e jovens adultos (entre 15 e 39 anos) (NATIONAL BRAIN TUMOR SOCIETY, 2021).

O estudo dos tumores cerebrais é de extrema importância, pois ajuda a entender as causas, os mecanismos e as possíveis terapias para essa doença grave. Além disso, o conhecimento sobre os tumores cerebrais tem um impacto significativo na qualidade de vida dos pacientes e de suas famílias, bem como no tratamento desses tumores.

O diagnóstico de tumores cerebrais é um processo complexo e delicado, que exige uma avaliação cuidadosa e precisa das imagens cerebrais. Com a evolução da tecnologia, a aplicação de técnicas de processamento de imagens tem se mostrado uma ferramenta promissora para o diagnóstico deste tipo de tumor. Tais técnicas permitem a análise detalhada de imagens cerebrais, possibilitando a identificação de características sutis e importantes para o diagnóstico e acompanhamento do tratamento do tumor cerebral.

A ressonância magnética é uma das principais técnicas de imagem usadas para identificar tumores cerebrais, porém, muitas vezes, identificar e localizar com precisão o tumor são tarefas desafiadoras. O uso de técnicas de processamento de imagens pode auxiliar na localização precisa do tumor cerebral, permitindo que os médicos determinem sua extensão e tomem decisões de tratamento mais adequadas. O processamento de imagens permite a análise de características específicas do tumor, como tamanho e formato para localizá-lo com maior precisão.

O uso de técnicas de processamento de imagens para diagnóstico e localização de tumores cerebrais permite a análise de imagens médicas para identificar as características dos tecidos cerebrais afetados e auxiliar na tomada de decisões clínicas.

A importância deste trabalho reside na possibilidade de melhorar o diagnóstico e a localização de tumores cerebrais, contribuindo para a melhoria da qualidade de vida dos pacientes. Além disso, a metodologia proposta pode ser utilizada como base para o desenvolvimento de sistemas de apoio à decisão clínica para o diagnóstico e localização de tumores em outros órgãos do corpo humano.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo apresentar uma metodologia para o diagnóstico e a localização de tumores cerebrais utilizando técnicas de processamento de imagens. Deseja-se criar uma aplicação para computador que mantenha uma complexidade computacional relativamente baixa e que possa ser acessado por usuários leigos em programação através da interface gráfica. Assim, será possível auxiliar no diagnóstico e localização de tumores cerebrais.

1.2.2 Objetivos Específicos

- Realizar um estudo sobre características de imagens de ressonância magnética de câncer no cérebro;
- Estudar métodos de diagnóstico de tumores cerebrais a partir de imagens;
- Estudar métodos de localização e segmentação de tumores em imagens de cérebros;
- Realizar testes de diagnóstico e localização de tumores cerebrais em Matlab;
- Desenvolver método de diagnóstico de tumores cerebrais em Python;
- Desenvolver método de localização e segmentação de tumores cerebrais em Python;
- Criar interface gráfica em Python.

1.3 Organização do Trabalho

O trabalho realizado está dividido da seguinte forma:

1º Capitulo: Este capítulo contém uma introdução sobre o tema, onde é realizada uma contextualização, são demonstrados os objetivos e a estrutura do trabalho.

2º Capítulo: Trata-se da Fundamentação Teórica para expor a base bibliográfica do desenvolvimento deste trabalho e mostrar aspectos importantes sobre Processamento de Imagens e Sistema Nervoso Central.

3º Capítulo: É demonstrada e explicada como foi realizada cada etapa do desenvolvimento do projeto.

4º Capítulo: Mostra os resultados obtidos após as etapas de desenvolvimento do trabalho, as dificuldades e as soluções encontradas.

5º Capítulo: Aborda as conclusões e discussões do trabalho, além de pontos que poderiam ser desenvolvidos em trabalhos futuros.

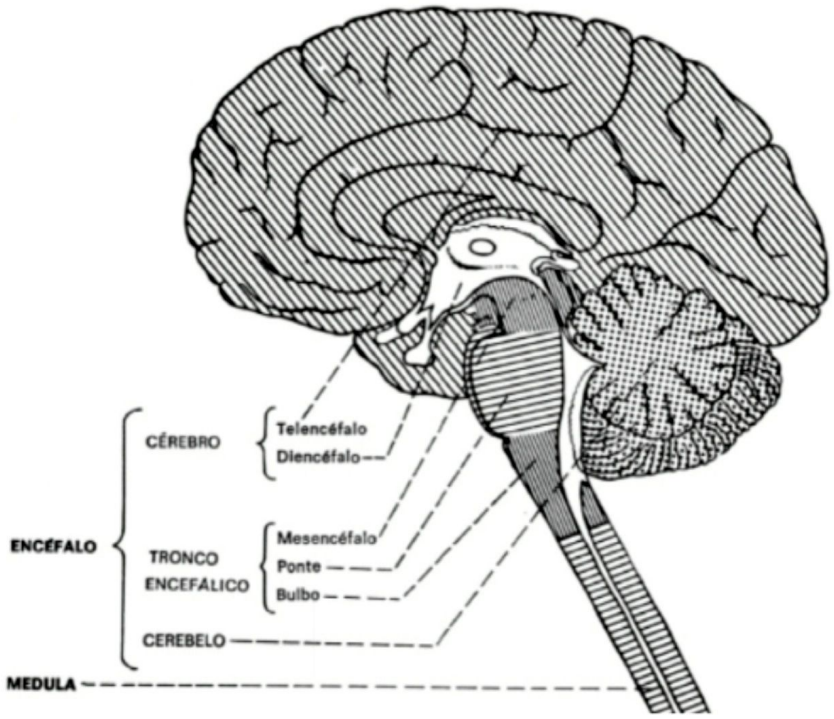
2 Fundamentação Teórica

2.1 Sistema Nervoso Central (SNC)

O SNC é uma divisão do Sistema Nervoso e é composto pelo encéfalo e pela medula espinhal. Ele é responsável pela interpretação de sinais advindos do restante do corpo (PFIZER, 2019) e subsequente envio de comandos. Portanto, as funções relacionadas aos sentidos e ao funcionamento dos órgãos são conduzidas pelo SNC.

Tanto o encéfalo, quanto a medula são protegidos por revestimentos ósseos, sendo o crânio no caso do encéfalo e a coluna vertebral para a medula. Além disso, o encéfalo se divide em três partes principais: o cérebro, o tronco encefálico e o cerebelo. Do SNC saem nervos e gânglios que formam o Sistema Nervoso Periférico. Uma visão geral do SNC é mostrada na figura 1.

Figura 1 – Anatomia do SNC



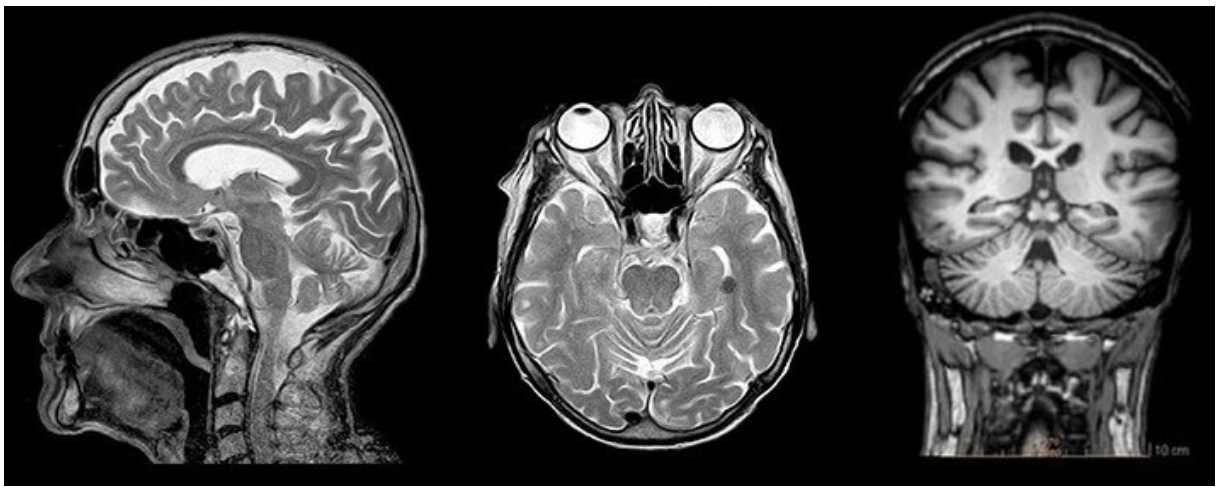
Fonte: Adaptado de Machado e Haertel (2013)

2.1.1 Planos de Observação 3D

O corpo humano pode ser dividido em planos diferentes para facilitar a visualização de certos sistemas, órgãos ou anomalias, como tumores, por exemplo. Os principais planos de visualização são o sagital, axial e coronal, vistos na figura 2 e explicados a seguir:

- Sagital: planos verticais que dividem o corpo em região esquerda e direita;
- Axial: planos horizontais que dividem o corpo em parte superior e inferior;
- Coronal: planos verticais que dividem o corpo em região frontal e dorsal, ou anterior e posterior.

Figura 2 – Plano sagital, axial e coronal, respectivamente



Fonte: BIOMEDICINA PADRÃO (2018)

2.1.2 Tumores Cerebrais

Tumor cerebral pode ser definido de maneira geral como um crescimento descontrolado de massa sólida formada por células indesejadas encontradas em diferentes partes do cérebro (LANG; ZHAO; JIA, 2016). Mais especificamente refere-se a uma coleção de neoplasias, cada uma com sua própria biologia, prognóstico e tratamento. Esses tumores são melhores identificados como neoplasias intracranianas, pois alguns não surgem do tecido cerebral (DEANGELIS, 2001).

Um tumor cerebral é marcado como benigno ou maligno. Os tumores cerebrais benignos não contêm células cancerígenas e crescem gradualmente, eles não se espalham e geralmente ficam em apenas uma região do cérebro, enquanto os tumores malignos contêm células cancerígenas, crescem rapidamente e se espalham para outras regiões do cérebro e, eventualmente, da coluna também. A Organização Mundial da Saúde (OMS) classificou os tumores cerebrais de acordo com o comportamento de saúde do cérebro, em tumores de grau 1 e 2, que são tumores de baixo grau (geralmente os benignos), ou tumores de grau 3 e 4 que são tumores de alto grau (malignos) (KHAN et al., 2020).

No entanto, para a maioria dos tumores intracranianos, a apresentação clínica, a abordagem diagnóstica e o tratamento inicial são semelhantes.

Existem vários tipos de tumores cerebrais, variando em sua grande maioria a célula que o constitui (FUNCIONALITÁ, 2020). Aqui serão citados cinco dos casos mais frequentes atualmente.

- Gliomas: proveniente das células gliais. Possui vários subtipos, entre eles o glioblastoma multiforme, que é o tumor mais comum e agressivo, principalmente em idosos;
- Meningiomas: presentes nas meninges, que são membranas que revestem o interior do crânio. Mais comuns em adultos do sexo feminino. Presente na figura 3;
- Meduloblastoma: proveniente das células neuroectodérmicas do cerebelo e apresentam crescimento acelerado. Mais frequente em crianças;
- Schwannomas: presentes nas células de Schwann. Geralmente apresenta caráter benigno e crescimento lento;
- Tumores de Hipófise: proveniente da glândula hipófise. Pode ter dois subtipos: funcionante e não-funcionante. Costuma ter caráter benigno.

Figura 3 – Exemplo de meningioma benigno, marcado com um círculo. As letras referem-se à orientação espacial do tomo axial da cabeça, a saber: A = anterior (frente); P = posterior (costas); R = lado direito (*right*) e L = lado esquerdo (*left*).

Fonte: Adaptado de McFaline-Figueroa e Lee (2018)

2.1.3 Casos de Tumores Cerebrais no Brasil

O Instituto Nacional do Câncer (INCA) publicou dados em 2022, onde constam mais de 11.100 casos de Câncer em que a localização primária do tumor foi no Sistema Nervoso Central (SNC) entre os sexos masculinos e femininos, colocando-o como o 10° com mais casos no Brasil. (INSTITUTO NACIONAL DO CÂNCER, 2022)

Além disso, houve mais de 9.350 casos de mortalidade com localização primária do tumor no SNC, situando-se em 8º lugar entre ambos os sexos. Estes dados evidenciam que se trata de um tipo muito comum de câncer, e, principalmente, que possui uma alta taxa de mortalidade (em torno de 84% dos casos).

2.1.4 Diagnóstico de Tumores Cerebrais por Imagens

O diagnóstico por imagens de tumores cerebrais é largamente utilizado, principalmente pela utilização de equipamentos de Ressonância Magnética (RM), que, atualmente, apresentam imagens relativamente claras em um exame que se encontra disseminado em várias áreas da medicina.

A ressonância magnética faz uso de ímãs potentes para produzir um forte campo magnético que induz o alinhamento dos prótons, no caso os íons H+ presentes na água, que compõe cerca de 70% do corpo humano (UNIMED, 2020). Em seguida uma corrente de radiofrequência é pulsada através do paciente, fazendo com que os prótons sejam estimulados e desalinhem-se do campo. Quando a corrente é cessada, eles voltam a se alinhar com o campo magnético inicial. Os sensores do equipamento, como o da figura 4, então, medem a energia liberada e o tempo demorado neste processo. A partir da diferença entre os valores destas variáveis é possível separar cada componente do corpo.

Esse procedimento é repetido várias vezes para obter a imagem completa tão nítida quanto possível. Apesar de requerer que o paciente fique imóvel por um certo período, a fim de não borrar a imagem, e às vezes requerer o uso de substâncias que aumentam o contraste, a ressonância magnética é um exame amplamente solicitado por especialistas pela clareza das imagens e por não fazer uso de radiação. (NATIONAL INSTITUTE OF BIOMEDICAL IMAGING AND BIOENGINEERING, 2013)

Além disso, o grande avanço em processamento de imagens computadorizado nas últimas décadas permite que sejam aplicados filtros que realcem certos tipos de elementos na imagem, como bordas, tons escuros e claros. Porém ainda é necessária a supervisão médica, que pode variar de profissional para profissional, além de demandar tempo destes especialistas.

Figura 4 – Exemplo de aparelho de Ressonância Magnética MR 7700

Fonte: PHILLIPS (2023)

2.2 Processamento de Imagens

Uma imagem em escala de cinza pode ser definida como uma função bidimensional, $f(x, y)$, onde x e y são coordenadas espaciais, e a amplitude de f em qualquer par de coordenadas é chamada de intensidade ou nível de cinza da imagem naquele ponto (GONZALEZ; WOODS; EDDINS, 2004). Uma imagem digital é composta por um número finito de elementos, cada um com uma localização e valor específicos. Estes elementos que possuem intensidade e coordenadas próprias são usualmente chamados de *Pixels*.

Já no que se refere ao processamento de imagens, como o nome sugere, pode ser simplesmente definido como o processamento (análise e manipulação) de imagens com algoritmos em um computador (através de código) (DEY, 2018). Dentro do grande conceito de processamento de imagens estão inclusos outros temas de importância para seu entendimento, como armazenamento, representação, extração de informações, manipulação, aprimoramento, restauração e interpretação de imagens.

2.2.1 Reconhecimento de Padrões

Reconhecimento de Padrões é a disciplina científica cujo objetivo é a classificação de um objeto em um número de categorias ou classes (THEODORIDIS; KOUTROUMBAS, 2006). Essa tarefa é, de certa forma, trivial para o cérebro humano na maioria das

aplicações, como no reconhecimento de uma forma geométrica, ou na diferenciação de rostos e vozes. Para computadores, por outro lado, pode ser extremamente custoso, como a maioria das tarefas que envolvem imagens.

De forma geral, alterações como, por exemplo, na iluminação, rotação ou inclinação são facilmente separadas e neutralizadas por seres humanos. Porém, pequenas alterações podem significar cenários completamente diferentes para um sistema artificial de reconhecimento de padrões. Dessa forma a minimização, ou até anulação, de interferências e ruídos são cruciais para uma boa classificação de imagens (DUDA; HART; STORK, 2000).

Por outro lado, é necessário ponderar o custo de se ter um sistema que resolve o problema, mas que não são implementáveis na prática devido a um elevado custo computacional. Problemas como o *overfitting* (no contexto de aprendizado de máquinas) podem surgir, que, além de demandar muito tempo e capacidade de processamento, prejudicam a capacidade de generalizar e classificar corretamente as amostras.

São apresentados, a seguir, alguns dos principais empecilhos e obstáculos que surgem no contexto de reconhecimento de padrões por sistemas computacionais e que podem comprometer seu funcionamento.

- **Extração das Características**

  Seja para um filtro de pré-processamento das entradas (*inputs*) ou o classificador final de um sistema, é necessário equilibrar o quanto exigir de cada camada de processamento. Utilizar uma única camada (*layer*) para extrair toda a informação desejada pode ser infactível, uma vez que ela ficaria sobrecarregada e não conseguiria performar da maneira devida.

  Há então de se separar os modelos a serem analisados e delegar a sistemas específicos de acordo com suas qualidades, sem que haja a perda de informação, o sobrecarregamento ou a subutilização dos respectivos classificadores.

- **Interferência nas Amostras**

  Dentro do contexto de *inputs*, é possível que surjam ruídos (também chamados de artefatos), ou seja, características que não são do objeto de análise em si, mas do ambiente ou da própria transformação da informação. Assim, algumas figuras podem demonstrar alterações na iluminação, desfoques ou até mesmo a falta de alguma parte do objeto analisado.

  Um bom reconhecedor de características deve saber separar os detalhes que importam dos que são meros detalhes e conseguir prosseguir com a análise. No caso de falta de informações, deve conseguir agir e classificar com o que tem em mãos, evidenciando a importância de um modelo que reconheça múltiplas características.

- **Conhecimento Prévio**

  O cérebro humano consegue analisar vários aspectos do contexto temporal e local para realizar uma decisão. Porém, para o contexto computacional é desafiador equilibrar os pontos relevantes e ponderá-los no momento de realizar a decisão.

  De maneira geral, é uma tarefa comum pré-selecionar manualmente as informações que vão chegar até a análise da rede em si. Desta forma, pode-se enfatizar o que é realmente importante e útil para o processo de aprendizagem, filtrando valores atípicos (*outliers*) que poderiam acabar sobrecarregando a rede com informação inútil ou até prejudicial.

  Cobrar que a rede adquira seu próprio conhecimento prévio sobre os modelos e os objetos de análise pode comprometer o desempenho do classificador, já que com mais dados, é mais complexo o trabalho de relacionar e dar sentido a todas essas informações em conjunto.

- **Agrupamento de Evidências**

  Como são muitas características em um objeto e frequentemente se procura analisar vários pontos de vista para chegar a uma decisão mais segura, ou seja, com menos risco de erro de classificação, é comum separar a tarefa em mais de um sistema.

  O caso em que todos os componentes concordam em uma dada classificação pode não ser factível, logo a rede mestre deve saber analisar e solucionar casos de discordância de decisões. A média ou a maioria nem sempre reflete a opção mais provável, já que um dos componentes pode oferecer resultados mais confiáveis que os restantes.

- **Complexidade Computacional**

  Conforme citado acima, todos os critérios para melhorar a performance e a qualidade da classificação devem ser analisados com cautela. A quantidade de características extraídas do modelo pode tornar a análise impraticável, em termos de processamento e/ou de tempo. Sempre há de se otimizar os dados para garantir que cada componente possa se especializar, providenciando melhores resultados com uma quantidade exequível de processamento.

2.2.2 Segmentação de Imagens

Segmentação de imagens, dentro do contexto de imagens digitais, é um processo de decomposição de uma imagem digital em vários segmentos, ou regiões, com base em suas propriedades visuais, como cor, intensidade, textura, borda, forma ou tamanho (SALDANHA; FREITAS, 2009). É um passo crucial no processamento de imagens e é usado para extrair informações relevantes da imagem para posterior análise.

Os objetos que compõem uma imagem possuem duas características básicas, que são: exibir alguma uniformidade interna em relação a uma propriedade da imagem, e contrastar em relação ao seu arredor. Em uma imagem digital, estas características não podem ser deterministicamente modeladas por sofrerem influência do ruído.

A segmentação de imagens pode ter como objetivo decompor a imagem em partes para posterior análise ou realizar uma mudança de representação (SHAPIRO; STOCKMAN, 2001). O resultado final esperado de um processo de segmentação é um número finito de regiões, que individualizam e evidenciam os diferentes segmentos contidos em uma imagem.

Há algumas subdivisões de quais processos podem ser atingidos pelo processo de segmentação a depender da técnica utilizada e de seu objetivo final.

- **Detecção de Descontinuidades**

  As descontinuidades encontradas em uma imagem podem ser pontuais, linhas ou os limites de um objeto. Tais características, destacam-se em uma imagem, seja por possuir tons distintos à região na quais estão inseridas (caso de pontos e linhas) ou por assinalarem mudanças bruscas de tons entre regiões (caso de bordas e limiares).

- **Detecção de Similaridades**

  A detecção de similaridade observa principalmente o interior dos objetos e não as bordas que os delimitam. Para isso, a análise parte da ideia que os *pixels* que compõem um objeto têm características similares enquanto que *pixels* de objetos distintos têm outras propriedades.

- **Segmentação no Espaço de Atributos**

  Agrupa as técnicas de limiarização e agrupamento (ou *Clustering*). Limiarização se baseia em definir um valor de limite para separar as partes de uma imagem em regiões distintas. O valor limiar é um valor numérico que divide as partes da imagem com intensidade de *pixel* acima desse valor de intensidade e abaixo dele, usualmente fazendo uso de histogramas. Já o *Clustering* agrupa regiões de *pixels* que possuem características semelhantes, tentando formar um padrão com grupos de feições similares.

2.2.3 *Image Processing Toolbox*

A *Image Processing Toolbox*^{TM} (IPT) é uma biblioteca de funções e algoritmos do *software* Matlab® desenvolvida pela Mathworks® para realizar tarefas na área de manipulação de imagens. A ferramenta oferece suporte ao processamento de imagens 2D e 3D, estando inclusos o processamento, análise, visualização e desenvolvimento de algoritmos. (MATHWORKS, 2021)

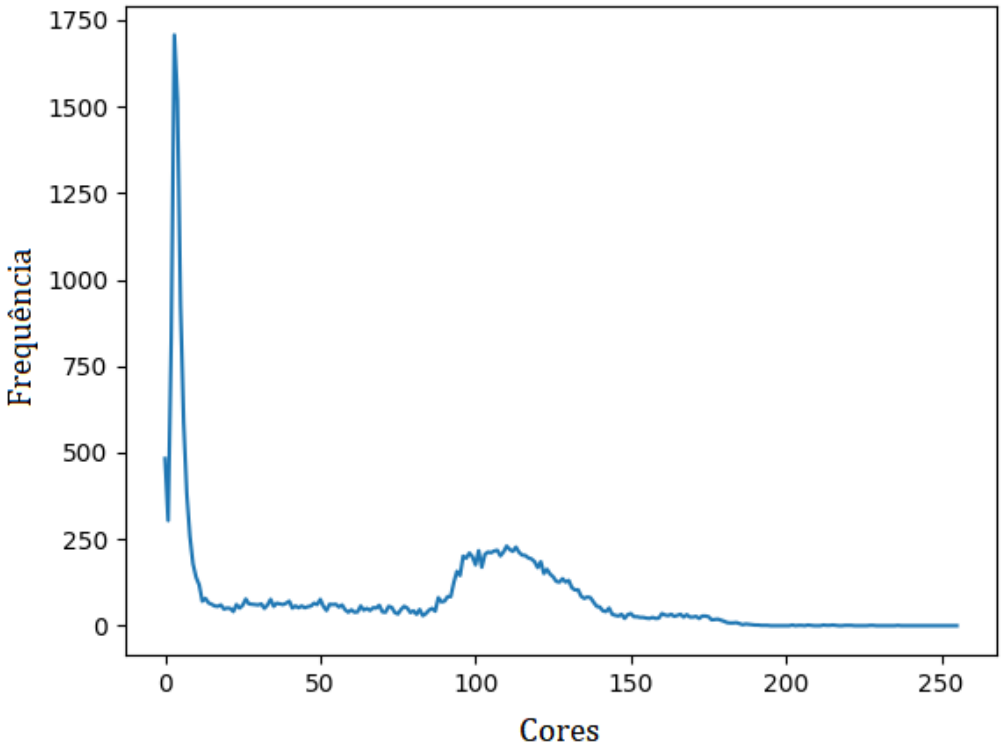
Algumas aplicações são: segmentação de imagens, aprimoramento, redução de ruído, transformações geométricas e utilização de aprendizado profundo e técnicas tradicionais de processamento de imagens. Dentro do contexto de análise de imagens, a IPT possui várias funções de identificação de cores, contagem de elementos e localização de formatos. Já em segmentação, estão presentes funções com diferentes abordagens para determinar os limites de regiões em uma imagem.

2.2.4 Processamento por Histograma

O histograma é uma forma de representação gráfica principalmente utilizada na área de estatística que relaciona uma variável (usualmente quantitativa contínua) com sua frequência de aparição. Pode ser usado também em representações envolvendo diversas variáveis, acompanhadas em vários momentos de tempo (GUIMARÃES, 2018).

Pode fazer uso de barras ou de uma linha contínua (como o da figura 5) para representar tal frequência, importando a altura, ou seja, o valor do eixo y. É possível agrupar os valores da variável, porém é importante manter o mesmo intervalo ao longo de toda a faixa da variável a fim de preservar a confiabilidade do gráfico.

Figura 5 – Exemplo de histograma feito em Python utilizando as bibliotecas OpenCV e Matplotlib



Fonte: O Próprio Autor

Os histogramas também possuem várias aplicações no contexto de processamento de imagens, em áreas como aprimoramento, compactação, segmentação e descrição (GONZALEZ; WOODS; EDDINS, 2004). Neste caso, ele relaciona as cores com o número de *pixels*, servindo como uma métrica qualitativa para analisar a intensidade de cor mais presente em uma imagem.

O histograma (h) de uma imagem digital com níveis de intensidade distribuídos no intervalo $[0, G]$ é definido como a função discreta 2.1:

$$h(r_k) = n_k \tag{2.1}$$

onde r_k é o k-ésimo nível de intensidade no intervalo $[0, G]$ e n_k é o número de *pixels* na imagem cujo nível de intensidade é r_k. O valor de G é 255 para imagens do tipo `uint8`, formato mais usualmente utilizado.

2.2.5 Redes de Hopfield

A rede de Hopfield é uma das formas mais simples de rede neural artificial (*Artificial Neural Network* - ANN), já que se trata de apenas uma camada de neurônios com laços de realimentação entre eles (HEATON, 2015). A saída de cada neurônio é conectada na entrada de todos neurônios, com exceção de si mesmo, ou seja, não acontece a auto-realimentação, como é possível visualizar no diagrama da figura 6.

Este modelo pode ser descrito como uma memória endereçável por conteúdo (MEC), pois a partir de um dado incompleto recebido, é capaz de relacionar a alguma informação armazenada em sua memória. No processo de recuperar as informações, ele também acaba sendo capaz de corrigir a entrada, caso necessário. Redes de Hopfield são muito utilizadas em aplicações com imagens justamente por suas habilidades auxiliarem na recuperação de imagens corrompidas por ruído, por exemplo.

O algoritmo contempla duas etapas: armazenamento e recuperação.

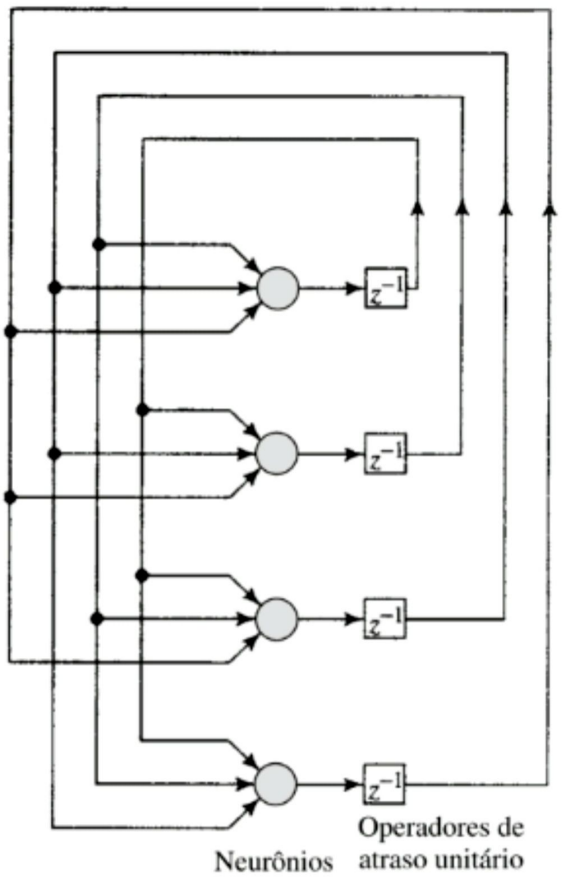
- **Armazenamento**

A fase de armazenamento corresponde ao processo de aprendizado da rede, onde as M memórias fundamentais serão gravadas para posteriormente serem recuperadas. Cada memória ($\boldsymbol{\xi}_\mu | \mu = 1, 2, \cdots, M$) tem o formato de um vetor de dimensão N, e é armazenada através da Aprendizagem de Hebb, gerando uma matriz de pesos $\mathbf{W}$ através da equação 2.2 (HAYKIN, 2001).

$$\mathbf{W} = \frac{1}{N} \sum_{\mu=1}^{M} \boldsymbol{\xi}_\mu \boldsymbol{\xi}_\mu^T - M\mathbf{I} \tag{2.2}$$

$\mathbf{W}$ é simétrica e, como não há auto-realimentação, sua diagonal principal é nula.

Figura 6 – Exemplo de arquitetura de uma rede de Hopfield com quatro neurônios



Fonte: Adaptado de Haykin (2001)

• **Recuperação**

A fase de recuperação recebe um vetor $\mathbf{y}$ com elementos iguais a ± 1. Este vetor é multiplicado pela matriz de pesos e um vetor de *bias* $\mathbf{b}$ pode ser somado, como mostra a equação 2.3.

$$\mathbf{y} = sinal(\mathbf{W}\mathbf{y} + \mathbf{b}) \tag{2.3}$$

O resultado da equação é determinístico porém a escolha da ordem dos neurônios é aleatória. Este processo é denominado atualização assíncrona. A atualização é realizada até que a saída se estabilize.

A rede de Hopfield transforma o vetor de entrada em alguma das memórias fundamentais. Por essa razão, $\mathbf{y}$ é normalmente uma versão ruidosa de uma memória. Caso o ruído seja muito grande, o resultado pode ser uma memória incorreta. Além disso, caso a rede tenha que armazenar muitos padrões, ela pode começar a apresentar resultados errôneos, ou seja, a rede de Hopfield possui um limite de armazenamento.

Vários estudos foram realizados para tentar encontrar a capacidade de armazenamento. (SILVA; SPATTI; FLAUZINO, 2010) cita a equação 2.4 que garante quase 100% de acerto

na fase de recuperação.

$$C^{100\%} = \frac{N}{4 \cdot ln(N)} \tag{2.4}$$

Lembrando que N é a dimensão do vetor de entrada. Em outras palavras, o limite de memórias fundamentais é severamente limitado pela dimensão da entrada, principalmente em aplicações onde precisão é um fator decisivo.

2.3 Correlação Linear

A correlação linear, de forma simplificada, mede quão relacionadas são duas variáveis, não somente pelos valores que elas assumem, mas pela semelhança entre os padrões de suas variações. Em outras palavras, o coeficiente de correlação linear mede quão próximos os pontos em um diagrama de dispersão estão espalhados ao redor da linha de regressão (MANN, 2010).

Dados n pares de valores (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n), o coeficiente de correlação entre as duas variáveis X e Y é definido pela equação 2.5, onde $dp(X)$ é o desvio padrão de X e $\overline{x}$ a média.

$$corr(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{dp(X)} \right) \left(\frac{y_i - \bar{y}}{dp(Y)} \right) \tag{2.5}$$

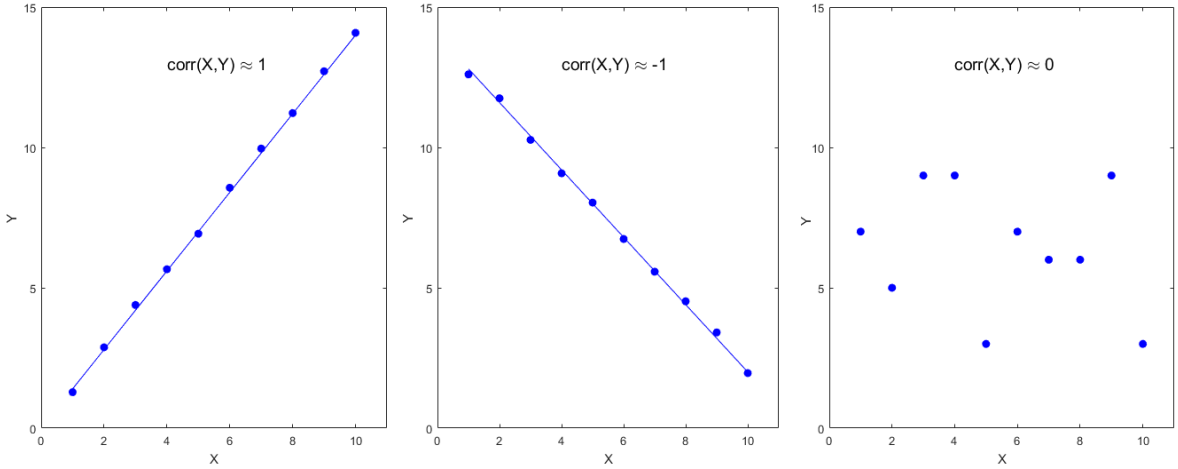
Ou seja, é dado pela média dos produtos dos valores padronizados das variáveis (MORETTIN; BUSSAB, 2017).

Esta equação limita a $corr(X, Y)$ entre $[-1, 1]$, resultando em uma métrica normalizada que possibilita uma visualização lógica da similaridade entre duas variáveis.

Se a correlação entre duas variáveis for positiva e próxima de 1, é dito que as variáveis têm uma forte correlação linear positiva. Se a correlação entre duas variáveis for positiva, mas próxima de zero, elas terão uma correlação linear positiva fraca. Em contraste, se a correlação entre duas variáveis for negativa e próxima de -1, diz-se que as variáveis têm uma forte correlação linear negativa. Se a correlação entre duas variáveis for negativa, mas próxima de zero, existe uma fraca correlação linear negativa entre elas. Agora, se a correlação entre elas é 0, as variáveis são totalmente descorrelacionadas.

Graficamente, uma correlação forte indica que os pontos no diagrama de dispersão estão muito próximos da linha de regressão, e uma correlação fraca indica que os pontos no diagrama de dispersão estão amplamente espalhados ao redor da linha de regressão. Estas características são vistas na figura 7.

Figura 7 – Exemplos de Correlação linear entre variáveis feitos utilizando o Matlab: (a) Correlação forte positiva (b) Correlação forte negativa (c) Correlação fraca



(a) (b) (c)

Fonte: O Próprio Autor

2.4 Python

Python foi criado em 1991 por Guido van Rossum, como um projeto de *hobby* para ser uma linguagem de programação fácil de aprender e usar, com uma sintaxe clara e legível. Atualmente, Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos. É popular entre programadores devido à sua sintaxe clara e concisa, sua ampla gama de bibliotecas e sua capacidade de ser utilizada em diversas áreas.

2.4.1 Processamento de imagens em Python

O processamento de imagens em Python é uma área em constante crescimento devido à popularidade da linguagem e à disponibilidade de várias bibliotecas de processamento de imagens de código aberto. Python oferece uma ampla variedade de ferramentas e bibliotecas para lidar com várias tarefas de processamento de imagens, como aquisição de imagens, manipulação, análise e visualização.

Algumas das bibliotecas mais populares para processamento de imagens em Python incluem:

- **OpenCV**

  Se trata de uma biblioteca de código aberto para processamento de imagens e visão computacional. É usada para uma ampla gama de tarefas, desde detecção de rosto até reconhecimento de objetos em tempo real.

- **Matplotlib**

  Biblioteca de visualização de dados em Python que é principalmente usada para visualização de imagens. É usada para exibir imagens em 2D e 3D, plotar gráficos e gerar animações.

- **Pillow**

  A PIL (*Python Imaging Library*) adiciona muitos recursos de processamento de imagem ao Python. Pillow é um ramo da PIL que adiciona alguns recursos mais amigáveis aos usuários. A Pillow suporta vários formatos de imagem, como JPEG, PNG, BMP e TIFF e é usada para tarefas como redimensionamento de imagens, ajuste de brilho e contraste, e adição de texto e marca d'água.

2.4.2 Interface Gráfica em Python

Interface gráfica, ou GUI (*Graphical User Interface*), é a forma pela qual os usuários interagem com aplicativos ou softwares por meio de elementos visuais, como botões, janelas, caixas de diálogo e outros elementos gráficos. Em Python, existem várias bibliotecas que permitem a criação de interfaces gráficas para *desktop*, e uma delas é a PySimpleGUI.

A PySimpleGUI é uma biblioteca Python de código aberto que fornece uma maneira simples e intuitiva de criar interfaces gráficas para desktop de forma rápida e fácil. É uma biblioteca de alto nível que permite criar GUIs em diversas plataformas, como Windows, macOS e Linux, e oferece suporte para Python 3 (PYSIMPLEGUI, 2023).

A principal ênfase da PySimpleGUI é na simplicidade e facilidade de uso. A biblioteca oferece uma abordagem baseada em eventos, onde você pode definir funções de manipulação de eventos para os elementos da GUI, como botões, caixas de texto, listas, gráficos e muito mais, permitindo criar interfaces gráficas ricas e interativas.

2.5 Trabalhos em Diagnóstico de Tumores a partir de Imagens

Conforme citado anteriormente, o avanço das técnicas de processamento de imagem e a melhoria de sistemas computacionais permitiu um grande aumento em programas de diagnóstico e localização de patologias em imagens como de ressonâncias magnéticas. Porém, logicamente, estes programas estão propensos a erros.

Dentro do contexto de exames e diagnósticos existem nomes específicos para estes erros: um falso positivo (FP) ocorre quando um exame sem o distúrbio procurado é sinalizado como positivo, e um falso negativo (FN) é o oposto, ou seja, há a presença da patologia, porém o programa sinaliza que não há.

Logo, é interessante para as implementações neste contexto manter controle sobre os FPs e FNs. Desta forma, foram criadas métricas para averiguar a presença destes eventos, elas são denominadas:

- **Sensibilidade**: mede a proporção de verdadeiros positivos (VPs) que são identificados corretamente. Neste trabalho também será referido pelo símbolo $\mathcal{S}$.

$$\mathcal{S} = \frac{VP}{VP + FN} \tag{2.6}$$

- **Especificidade**: mede a proporção de verdadeiros negativos (VNs) que são identificados corretamente. Neste trabalho também será referido pelo símbolo $\mathcal{E}$.

$$\mathcal{E} = \frac{VN}{VN + FP} \tag{2.7}$$

- **Acurácia**: mede a proporção de amostras que são identificadas corretamente, juntando as amostras positivas e negativas. Neste trabalho também será referido pelo símbolo $\mathcal{A}$.

$$\mathcal{A} = \frac{VN + VP}{VN + FP + VP + FN} \tag{2.8}$$

Foram analisados alguns estudos de diagnóstico assistido por computador (*Computer-aided Diagnosis* - CAD) aplicado à área de interesse deste trabalho: tumores cerebrais em imagens de ressonância magnética. As seguir são citadas, resumidamente, suas metodologias e os resultados alcançados.

Em Zhang et al. (2011), é feito uso da Transformada Wavelet para extrair dados das imagens de entrada, em seguida, a análise de componentes principais (*Principal Component Analysis* - PCA) atua para diminuir a dimensão dos recursos, desta forma, extraindo e filtrando as informações. O classificador escolhido é formado por uma rede neural com retropropagação com gradiente conjugado escalado para encontrar seus pesos. O trabalho, entretanto, possui somente 66 imagens no banco de dados entre exemplos negativos e positivos.

A técnica de Análise de Componentes Principais é um método matemático que emprega uma transformação ortogonal, conhecida como ortogonalização de vetores, para converter um conjunto de observações de variáveis que podem estar correlacionadas em um conjunto de valores de variáveis lineares não correlacionadas, denominadas de componentes principais.

Já em Jafari e Kasaei (2011), o processo tem seis etapas e visa classificar as imagens em três categorias: negativo, maligno e benigno. Primeiramente, a etapa de pré-processamento promove melhoria e restauração no banco de dados, em seguida é realizada

a segmentação de crescimento de região semeada, que se encaminhará para a etapa de rotulagem de componentes conectados (*Connected Component Labeling* - CCL). A extração de recursos também é feita usando a Transformada Wavelet Discreta (*Discrete Wavelet Transform* - DWT), a redução de dimensão dos recursos faz uso da PCA. Por fim, a classificação é processada por uma rede neural *feed-forward* com retropropagação.

El-Dahshan, Hosny e Salem (2010) também utiliza a DWT e a PCA para extrair e reduzir os recursos, respectivamente. Porém, compara a classificação realizada por dois tipos de ANN: o primeiro é baseado na rede neural *feed-forward* com retropropagação e o segundo classificador é baseado no algoritmo do k-ésimo vizinho mais próximo (*K-Nearest Neighbors* - K-NN).

Em seu trabalho, Chaplot, Patnaik e Jagannathan (2006) aplica a DWT nos dados de entrada para classificadores baseados em métodos de aprendizado de máquinas. No primeiro teste usa mapas auto-organizáveis (*Self-organizing Maps* - SOM), em seguida, testa SOM com kernel baseado em função de base radial (*Radial Basis Function Based Kernel* - RBFBK) e, por fim, máquina de vetores de suporte (*Support Vector Machine* - SVM). O procedimento foi aplicado em 52 imagens de RM.

Ao contrário do feito em Zöllner, Emblem e Schad (2012), que fixa um classificador baseado em SVM, para analisar técnicas de redução de parâmetros. Foram testados três métodos: coeficientes de correlação de Pearson (*Pearson's Correlation Coefficient* - PCC), análise de componentes principais (PCA) e análise de componentes independentes (*Independent Component Analysis* - ICA).

El-Dahshan et al. (2014) propõe um conjunto de técnicas que conta com: uma rede neural retroalimentada acoplada a pulsos para segmentação de imagem, a TWD para extração de recursos e a PCA para reduzir a dimensionalidade dos recursos. Por fim, a rede neural *feed-forward* com retropropagação para classificar entradas em positivas e negativas.

No geral, é possível notar três pontos comuns na maioria dos trabalhos: extração de recursos com a TWD, sua redução com a PCA e algum método de ANN como classificador. Como os métodos são similares, os resultados também ficam dentro de uma faixa parecida: aproximadamente entre 80% a 100% de acurácia. A tabela 1 resume os resultados obtidos pelos trabalhos citados anteriormente.

Tabela 1 – Dados de Acurácia, Sensibilidade e Especificidade atingidos em outros trabalhos, ordenados a partir da acurácia

Trabalho	$\mathcal{A}$ [%]	$\mathcal{S}$ [%]	$\mathcal{E}$ [%]
Zhang et al. (2011)	100	100	100
Jafari e Kasaei (2011)	99,8	100	98
El-Dahshan et al. (2014)	99	100	92,8
Chaplot, Patnaik e Jagannathan (2006) com SVM e RBFBK	98	-	-
El-Dahshan, Hosny e Salem (2010) usando K-NN	98	96	97
El-Dahshan, Hosny e Salem (2010) usando FP-NN	97	95,9	96
Chaplot, Patnaik e Jagannathan (2006) com SVM	96	-	-
Chaplot, Patnaik e Jagannathan (2006) com SOM	94	-	-
Zöllner, Emblem e Schad (2012) com PCA	85	89	84
Zöllner, Emblem e Schad (2012) com PCC	82	89	77
Zöllner, Emblem e Schad (2012) com ICA	79	87	75

Fonte: Adaptado de El-Dahshan et al. (2014)

3 Desenvolvimento

3.1 Banco de Dados

Foi buscado um banco de dados que tivesse várias imagens, ao menos duzentas, para que fosse possível encontrar diferentes tipos, localizações e tamanhos de tumores. Além disso, era importante que houvesse tanto imagens positivas, quanto negativas, para poder comparar e diferenciar os dois casos.

A maioria dos bancos de dados não são muito extensos, já que o acesso a estas imagens é relativamente restrito. Alguns também acabam colocando mais de uma imagem do mesmo paciente, limitando a questão de diversidade das amostras. Assim, foi evitada a utilização destes.

Por fim, encontrou-se o banco de dados disponibilizado em KAGGLE (2019), que conta com 155 imagens positivas (com a presença de tumores) e 98 imagens negativas (sem a presença de tumores). Os pontos negativos de tal fonte é que, aparentemente, foram reunidas informações de diversas origens, então elas não seguem o mesmo padrão.

O quadro 1 apresenta alguns parâmetros que foram adotados como padrões para as imagens que serão testadas.

Quadro 1 – Padrão de referência das imagens do banco de dados

Plano de Observação	Axial
Cor de fundo	Preto
Sentido	Vertical
Presença de Olhos	Negativo

Fonte: O Próprio Autor

Tal referência engloba a grande maioria das imagens presentes do *dataset* e foi adotada para que o algoritmo entenda que a maior alteração de imagem para imagem é a presença do tumor (ou sua ausência) e, no máximo, a iluminação característica de cada amostra.

Abaixo foram citados alguns dos problemas encontrados nos dados e as ações tomadas para minimizar a interferência nos testes:

- **Informação Escrita:** Encontrado em boa parte das amostras do banco de dados, geralmente letras ou informações pequenas. Optou-se por mantê-las pois é é uma característica de várias imagens de RM, não um problema específico do *dataset*. Além disso, eram relativamente poucos *pixels*, como pode ser visto na figura 8.

Figura 8 – Imagem com informação escrita nos cantos

Fonte: KAGGLE (2019)

- **Presença de Olhos:** A presença dos olhos nas imagens acaba alterando significativamente as características das amostras, uma vez que eles possuem formato circular e cor clara, assim como os tumores. Optou-se por retirar três imagens já que não haveria como consertá-las. A figura 9 é uma das que foram excluídas.

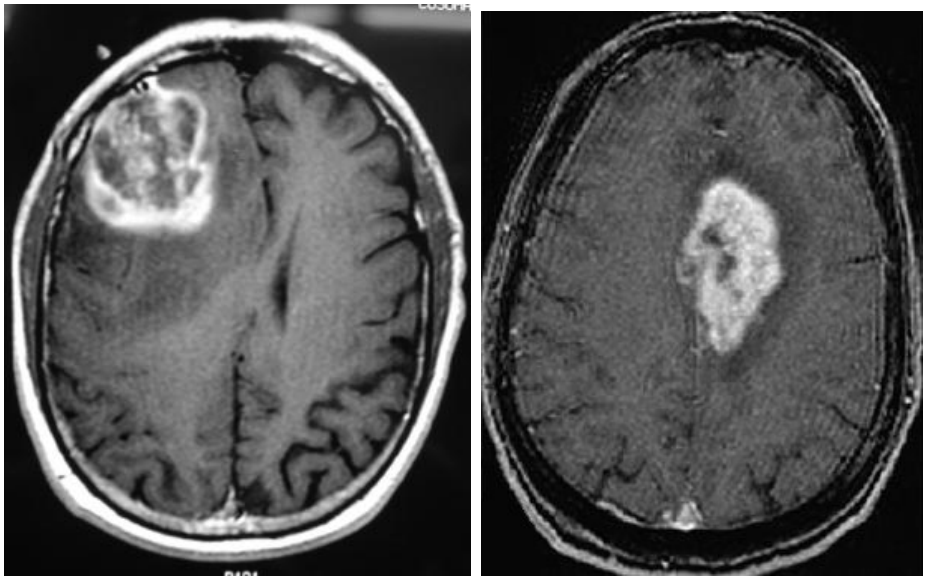
Figura 9 – Imagem do banco de dados retirada por conter olhos

Fonte: KAGGLE (2019)

- **Baixa Resolução:** Algumas imagens apresentavam baixa resolução, como a figura 10 (b), resultando como principal problema a falta de detalhes em regiões de fron-

teira. Foi notado que haviam casos, como o da figura 10 (a), que a falta de nitidez estava presente porque a amostra foi fotografada, ao invés de digitalizada.

Figura 10 – Exemplos de imagens com baixa resolução: (a) Amostra fotografada e (b) imagem embaçada

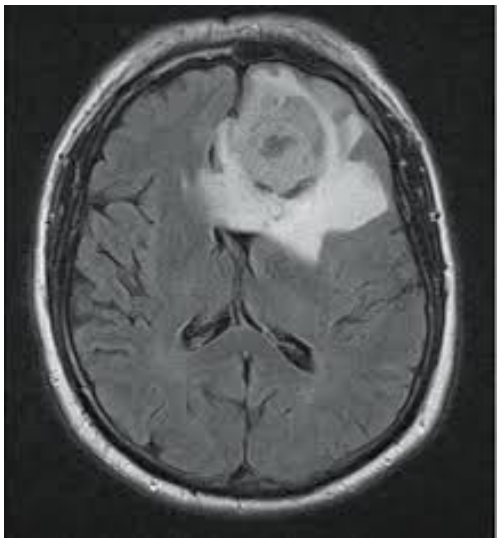


(a) (b)

Fonte: KAGGLE (2019)

- **Fundo mais claro:** Algumas imagens tinham o fundo em um tom mais acinzentado do que propriamente preto, como a figura 11. Porém não chegavam a ser tão claros quanto os tumores ou sequer a massa encefálica, logo foram mantidas. Em casos muito discrepantes, o fundo foi escurecido.

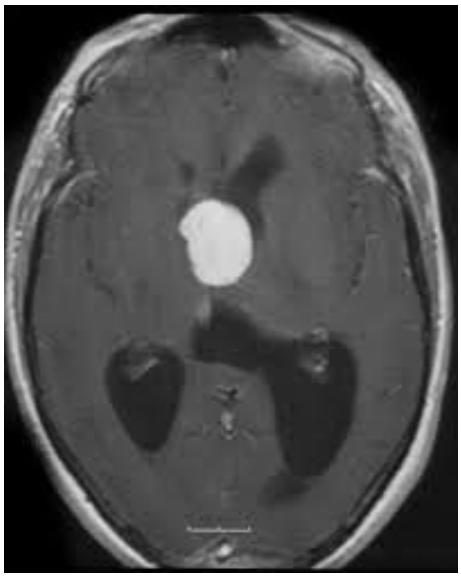
Figura 11 – Imagem com o fundo mais claro



Fonte: KAGGLE (2019)

- **Mal recorte da imagem:** Algumas imagens foram excessivamente recortadas, tirando uma pequena parte do crânio, como pode ser visto na figura 12. Como não houve nenhum caso em que um recorte desfigurava a imagem, elas foram mantidas.

Figura 12 – Imagem do banco de dados mal recortada



Fonte: KAGGLE (2019)

3.2 Testes Iniciais

Tendo o objetivo e o *dataset* em mãos, buscou-se um ambiente que possibilitasse a realização de testes. Seriam necessários algoritmos com imagens, logo programas que tratam com linguagens mais básicas, como C, foram descartados. Ao mesmo tempo, o problema se mostrava multidisciplinar, logo era necessário que o ambiente também fosse capaz de realizar vários tipos de operações numéricas.

Optou-se por realizar os primeiros testes no Matlab, da MathWorks, por questão de familiaridade e por ser um programa de alto nível com várias *toolboxes* que permitem as mais variadas aplicações, seja no contexto de processamento de imagens ou cálculo numérico no geral.

3.2.1 Testes para diagnóstico

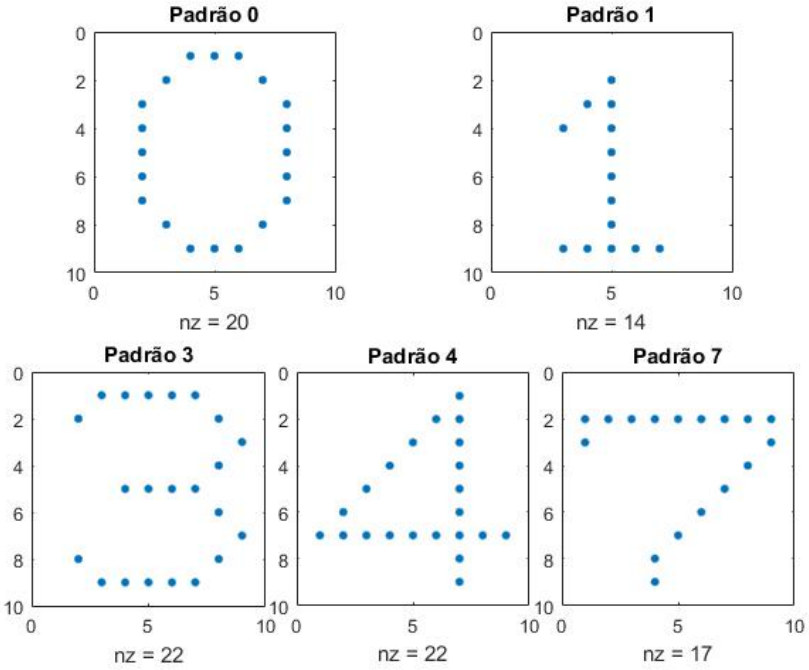
O problema de diagnóstico de tumores é comumente abordado utilizando inteligência artificial (IA), conforme foi notado nos trabalhos citados na tabela 1. Logo, primeiramente este trabalho buscou formas de aplicar alguma das inúmeras técnicas de IA no diagnóstico das amostras disponíveis.

Os estudos se iniciaram por uma das formas mais simples de ANN: as Redes de Hopfield. Sua capacidade de mapear as entradas para alguma de suas memórias fundamentais

foi um dos principais motivos para sua escolha. Isto porque, caso houvessem memórias com diferentes casos positivos (alternando os brilhos e as localizações dos tumores) e casos negativos, a rede poderia classificar a amostra de entrada através do mapeamento para a memória mais parecida. Sua simplicidade e complexidade computacional relativamente reduzida também foram pontos de interesse.

Para aprender sobre a implementação desta rede neural e obter resultados em menor escala, foi proposto o desenvolvimento de um programa para recuperar símbolos corrompidos por ruído.

Figura 13 – Padrões de símbolos a serem armazenados pela Rede de Hopfield



Fonte: O próprio autor

O programa foi implementado respeitando as equações do modelo de Hopfield em especial a equação 2.2, descrita anteriormente. Ela tem grande importância pois armazena as memórias fundamentais através do cálculo da matriz de pesos.

```
function [ W ] = memorizar( ME )
%Funcao implementada para calcular a matriz de pesos
%W atraves das memorias fundamentais contidas em ME

M = length(ME(1,:));
N = length(ME(:,1));
W=zeros(N);
for i=1:M
    W = W + ME(:,i) * transpose(ME(:,i));
end
```

```
11 W = W - M*eye(N);
12 W = (1/N)*W;
13
14 end
```

A implementação objetivava compreender melhor as qualidades e limitações das Redes de Hopfield, e analisar, mesmo que em pequena escala, se ela teria potencial para ser melhor desenvolvida e atuar como classificadora da etapa de diagnóstico. Na tentativa de adiantar os problemas possíveis ao aumentar a escala para imagens reais, foram analisados alguns pontos importantes da aplicação escolhida que podem vir a sobrecarregar tanto as Redes de Hopfield, quanto outras formas de ANN que também poderiam ser utilizadas.

Apesar de ressonâncias magnéticas de cérebros não possuírem uma alta nitidez quando comparadas aos formatos mais modernos de imagem, somente por se tratar de imagens, elas já são mais pesadas que a maioria das informações que são normalmente aplicadas em redes neurais, ou métodos de inteligência artificial no geral.

Uma amostra do banco de dados utilizados tem resolução média de 250×250, ou seja, possuem em média 50.000 *pixels*. Mesmo imagens com resolução mais baixa possuem 25.000 *pixels* e as com maior resolução ultrapassam os 150.000 *pixels*. Considerando que cada *pixel* é representado por uma variável `uint8`, ocupando consequentemente um byte de memória por *pixel*, mesmo imagens mais simples requereriam muita capacidade de processamento.

Por isso, vários trabalhos estudam formas de reduzir as dimensões da informação para que a rede trabalhe somente com o crucial para a tomada de decisão, conforme demonstrado na tabela 1. Neste contexto foram iniciados também testes para realizar o pré-processamento das amostras.

Analisando as figuras do banco de dados haviam algumas informações que não se mantinham constantes de um caso para o outro, como: localização e dimensão do tumor, resolução e brilho. Entretanto, o padrão das cores era quase idêntico em todas: fundo preto, crânio esbranquiçado, massa encefálica cinzenta e, quando estava presente, tumor esbranquiçado.

Portanto, mesmo se tratando de figuras em escala de cinza, as cores conseguiam fazer toda a diferença na diferenciação dos elementos da imagem. Então, por mais que fosse pretendido encontrar a localização de um eventual tumor na imagem, esta etapa não precisava ser realizada no mesmo processo que o diagnóstico.

Caso fosse possível detectar este maior número de *pixels* claros na imagem (que teoricamente seria a região com tumor) seria possível dizer se a imagem é positiva ou negativa. Ou seja, toda a informação da imagem poderia ser substituída pela quantidade de *pixels* com cada cor.

• **Histograma**

Felizmente, os histogramas possuem exatamente esta finalidade. E, por se tratarem de uma ferramenta comum dentro do contexto de imagens, sua aplicação seria relativamente simples. As funções da IPT cuidam da parte de imagens e possuem o padrão de manter o prefixo "im"no nome das funções. Para os primeiros testes foram usadas funções de leitura e obtenção de histograma.

```
1 modelo = imread('8 no.jpg');
2 modelo = modelo(:,:,1);
3 histmodelo = imhist(modelo);
```

O fragmento de código anterior lê a imagem com a função `imread()` e obtém o histograma com a função `imhist()`. A linha 2 se trata de uma precaução caso a imagem não esteja em escala de cinza.

• **Coeficiente de Correlação Linear**

Em seguida, pensando em avaliar se fazia sentido a ideia proposta, buscou-se alguma métrica que diria matematicamente caso os resultados foram promissores. Como o histograma de uma imagem retorna uma curva com o nível de presença de cada cor, era desejado saber o quão parecido eram duas curvas. E é exatamente o que propõe o coeficiente de correlação.

A função de correlação linear, `corr()`, pertence à *toolbox* de estatística e *machine learning*.

```
1 modelo = imread('8 no.jpg');
2 modelo = modelo(:,:,1);
3
4 positivo1 = imread('Y6.jpg');
5 positivo1 = positivo1(:,:,1);
6
7 histmodelo = imhist(modelo);
8 histpos1 = imhist(positivo1);
9 cor1=corr(histpos1,histmodelo);
```

• **Subtração de Histogramas**

Conforme comentado anteriormente, as cores de cada elemento são quase padronizadas nas amostras: fundo escuro, cérebro acinzentado, crânio e tumores claros. Como o que é interessante é a presença ou a ausência do tumor para o diagnóstico, todo o resto poderia ser cancelado.

No próximo passo, procederam-se testes de subtração entre histogramas. No caso de amostras perfeitamente iguais entre si, com exceção de um possível tumor, todo o restante das tonalidades se cancelariam. Porém era desejado saber o efeito de diferentes resoluções, tamanhos de cérebro em relação ao fundo e iluminação intrínseca das amostras.

```
1 teste1 = histpos1 - histmodelo;
```

```
bar(teste1)
```

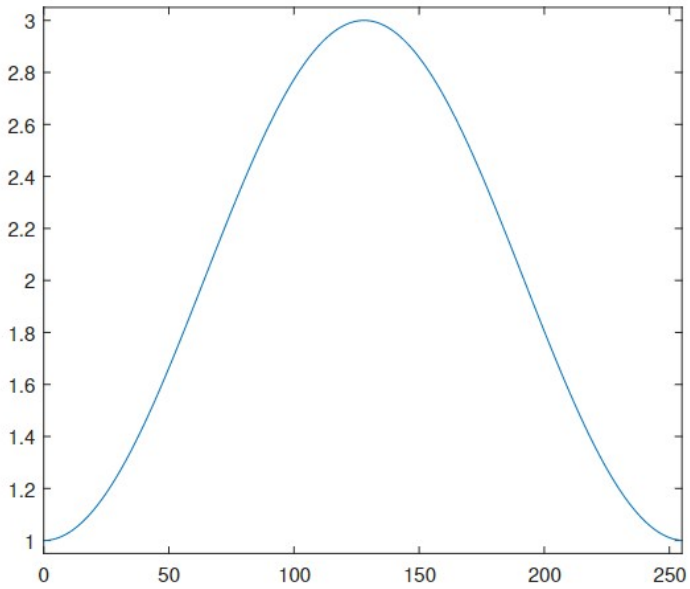
Como a função `imread` converte os valores dos *pixels* para o formato `uint8`, `imhist()` sempre retorna um vetor com 256 posições, logo o Matlab entende que a subtração será realizada índice a índice.

- **Função de Normalização de Histograma**

Conforme foram sendo analisados os histogramas das amostras, notou-se que as faixas de cores intermediárias eram ofuscadas pela quantidade de *pixels* escuros, devido ao plano de fundo das imagens. Para tentar dar um foco maior nas cores do cérebro, foi pensada em uma função que pudesse amplificar somente a zona intermediária dos histogramas. Tal função foi implementada conforme o código a seguir e seu gráfico está disponível na figura 14.

```
a=0:1:255;
norm = (2-cos(a*pi/128))';
plot(a,norm)
```

Figura 14 – Função normalizada plotada pelo Matlab



Fonte: O Próprio Autor

```
histmodelo = imhist(modelo);
hmodelnorm = histmodelo .* norm;

histpos1 = imhist(positivo1);
cor1 = corr(histpos1,histmodelo);
cor1norm = corr(histpos1,hmodelnorm);
```

Neste caso, o Matlab precisa do operador .∗ para entender que a multiplicação é de termo a termo.

• **Função de Equalização de Histograma**

Em outro teste, foi examinado se era possível fazer um pré-processamento nas amostras para tentar manter todas equalizadas, reduzindo o efeito do brilho característico de cada uma. para isto, foi utilizada a função `histeq()`, que busca uniformizar um histograma, ou seja, tenta achatá-lo.

```
pos2=imread('Y13.jpg');
subplot(2,2,1)
imshow(pos2)

teste=histeq(pos2,256); % 256: numero de divisoes do histograma
subplot(2,2,2)
imshow(teste)

subplot(2,2,3)
imhist(pos2)

subplot(2,2,4)
imhist(teste)
```

• **Equalização Guiada de Histograma**

A função `histeq()` não necessariamente tenta achatar o histograma, caso seja passado um vetor, ela tentará deixar o histograma próximo ao vetor. Em outras palavras, na forma padrão, `histeq()` tenta se aproximar a uma função constante. Entretanto, é possível passar outra função para ela tentar se aproximar.

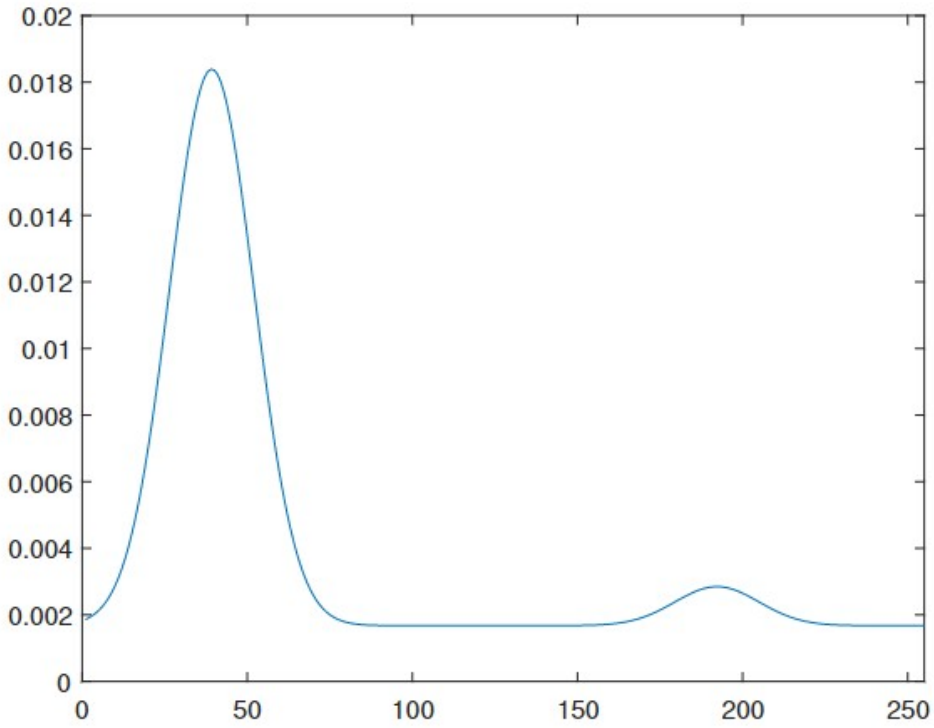
Como o tumor é claro, e quase todo o restante da imagem é mais escuro que ele (com exceção do crânio), foi pensado em acumular tudo o que está cinza e preto em uma tonalidade de cor e todo o resto em outra. Desta forma, teoricamente seria possível equalizar o brilho das amostras ao mesmo tempo em que destaca o tumor.

Para isso ser possível, o vetor a ser passado para a `histeq()`, deve ter dois picos, um próximo de zero e outro próximo de 255, e o restante dos elementos devem ser próximos de zero. Em Gonzalez, Woods e Eddins (2004), é feita a implementação de uma função gaussiana bimodal, ou seja, uma gaussiana com dois picos, totalmente configurável. A figura 15 é um exemplo desta função.

```
function p = twomodegauss(m1, sig1, m2, sig2, A1, A2, k)
    % TWOMODEGAUSS Generates a bimodal Gaussian function.
    % P = TWOMODEGAUSS(M1, SIG1, M2, SIG2, A1, A2, K) generates a bimodal,
    % Gaussian-like function in the interval [0, 1]. P is a 256-element
    % vector normalized so that SUM(P) equals 1. The mean and standard
    % deviation of the modes are (M1, SIG1) and (M2, SIG2), respectively.
    % A1 and A2 are the amplitude values of the two modes. Since the
    % output is normalized, only the relative values of A1 and A2 are
    % important. K is an offset value that raises the "floor" of the
    % function. A good set of values to try is M1 = 0.15, SIG1 = 0.05,
```

```
    % M2 = 0.75, SIG2 = 0.05, A1 = 1, A2 = 0.07, and K = 0.002.
    c1 = A1 * (1 / ((2 * pi) ^ 0.5) * sig1);
    k1 = 2 * (sig1 ^ 2);
    c2 = A2 * (1 / ((2 * pi) ^ 0.5) * sig2);
    k2 = 2 * (sig2 ^ 2);
    z = linspace(0, 1, 256);
    p = k + c1 * exp(-((z - m1) .^ 2) ./ k1) + ...
    c2 * exp(-((z - m2) .^ 2) ./ k2);
    p = p ./ sum(p(:));
    figure, plot(p)
    xlim([0 255])
end
```

Figura 15 – Gaussiana Bimodal implementada no Matlab



Fonte: O próprio autor

A função é exatamente a que era procurada. Foram escolhidos índices para atrair os picos nos locais desejados, além disso, o histograma das amostras sempre tem mais valores próximos a zero, logo o primeiro pico da gaussiana bimodal deve ser maior que o segundo. A função da figura 15 segue este padrão.

```
p = twomodegauss(0.15, 0.05, 0.75, 0.05, 1, 0.07, 0.002);

figure();
subplot(2,2,1)
imshow(pos2)
```

```
6
7 teste=histeq(pos2,p);
8 subplot(2,2,2)
9 imshow(teste)
10
11 subplot(2,2,3)
12 imhist(pos2)
13
14 subplot(2,2,4)
15 imhist(teste)
```

3.2.2 Testes para localização

Como alguns testes do que era para ser um pré-processamento de diagnóstico já haviam sido promissores como classificadores finais, ou seja, sem a necessidade de uma IA atuar posteriormente, foi proposta a tentativa de localizar os tumores também sem a utilização de IA.

Dentro do plano anterior de utilizar, por exemplo, uma ANN, a rede seria treinada para localizar na imagem algum elemento que se assemelharia a um tumor e, a partir disto, tentaria diagnosticar a amostra como positiva ou negativa. Ao sair do contexto de IA, o processo de localização passaria a ser determinístico, o que seria um desafio já que a posição do tumor e sua cor são variáveis.

Dentro do contexto de processamento de imagens, estava sendo buscada alguma função para segmentar a imagem. A segmentação desejada, logicamente, destacaria o tumor em relação à sua vizinhança. Logo, as atenções retornam ao padrão de cor que cada elemento das amostras possui. A própria função `histeq()` com a função gaussiana bimodal, já poderia ter uma aplicação neste contexto. Isso porque ela busca dar destaque ao tumor.

Também foi testada a função `imadjust()`, que recebe uma imagem, uma faixa de cores das imagem, tal que esta faixa será distribuída ou reduzida para outra faixa de cores também especificada na chamada da função. Em outras palavras, ela pode atuar para aumentar o contraste da imagem.

```
1 subplot(1,2,1)
2 imshow(neg2)
3
4 a4=imadjust(pos1,[0.65  0.7],[0  1]);
5 subplot(1,2,2)
6 imshow(a4)
```

No código anterior, a função `imadjust()` recebe uma imagem, a faixa de valores de entrada ([0.65 0.7]) e a de saída ([0 1]). A função retorna uma matriz (imagem) que tem agora a cor respectiva a 65% como $'0'$ (preto) e a cor respectiva a 70% como $'1'$ (branco). Tudo que é menor que 65% agora é zero, ao mesmo tempo, tudo que é maior que 70% é

um. Enquanto isso, a faixa de cores entre 65% e 70% é distribuída entre 0% e 100%, ou seja, entre 0 e 255.

3.3 Diagnóstico

Após o desenvolvimento dos diversos testes citados nos tópicos anteriores, foi tomada a decisão de dar andamento com um programa utilizando somente os métodos que, a princípio, seriam aplicados como pré-processamento das amostras para seguir com uma aplicação de ANN. Ou seja, a solução se basearia em sua grande parte nos histogramas das imagens de RM e das correlações entre eles.

Entretanto, para o prosseguimento dos experimentos, optou-se pela utilização da linguagem de programação Python. Tal escolha se baseou na quantidade de bibliotecas e tutoriais disponíveis, em sua versatilidade, em sua velocidade de processamento e também para adquirir experiência na linguagem em questão, que é muito popular, pois possui diversas aplicações em várias áreas.

Primeiramente, o objetivo era implementar um classificador, juntando todo o conhecimento obtido nos testes, que aproveitasse da melhor maneira possível o potencial de diagnóstico dos métodos utilizados. Para validar este classificador criado, também serão realizados, pela primeira vez, testes em larga escala com todas as amostras do banco de dados, a fim de calcular métricas relevantes no contexto: acurácia, sensibilidade e especificidade.

3.3.1 Leitura das Imagens e Obtenção dos Histogramas

Como se tratava de um *dataset* extenso (152 amostras positivas e 98 negativas), a implementação de operações com eles não poderia ser feita da mesma maneira que nos testes, onde eram copiadas e coladas as mesmas funções alterando o nome do arquivo, já que se tratavam de no máximo cinco amostras utilizadas. O tratamento dos dados teria que passar realizada através de *loops* (laços).

Por mais que trabalhar com variáveis, principalmente com *arrays* (vetores), em laços fosse mais fácil, ainda era possível manipular arquivos através da adaptação de strings. Entretanto, havia outro obstáculo antes de realmente conseguir utilizar as imagens no programa: a nomenclatura dos dados do *dataset* não era completamente padronizado.

No banco de dados negativo haviam amostras nomeadas com o padrão '# no.jpg', sendo # um número referente à contagem, outras com o padrão 'N#.jpg' e algumas com 'No#.jpg'. Já no positivo, todas seguiam o molde 'Y#.jpg', porém havia *gaps*, ou seja, ocasionalmente a contagem parava, pulava alguns números e continuava. Além disso, em ambos *datasets*, algumas amostras não seguiam a extensão padrão '.jpg', algumas estavam

em '.png', outras em '.jpeg' e '.JPG'. Apesar deste último não alterar a leitura, os outros dois geravam erros.

Optou-se por alterar manualmente os nomes das imagens mantendo o padrão: '# no.jpg' para amostras negativas e 'Y#.jpg' para positivas. Mesmo os formatos diferentes podem ser convertidos somente alterando o sufixo. Somente assim foi possível realizar de forma iterativa a leitura das imagens.

As imagens são lidas através da função `imread()` da biblioteca OpenCV e a `imhist()` foi baixada do repositório (PYPI, 2021), ambas possuem funcionamento quase idêntico às suas correspondentes do Matlab. Optou-se por armazenar os histogramas em listas, `pos` e `neg`, também para possibilitar a manipulação em laços.

```
pos=[]
neg=[]
str1 = " no.jpg"
str2 = ".jpg"


for i in range(98):
    file = str(i+1) + str1  #i+1 pois indice comeca em 0
                            #mas a contagem das figuras comeca em 1
    hsv = cv2.imread(file,0) # o 0 implica na leitura em
                            #escala de cinza caso seja colorida
    neg.append(imhist(hsv))


for i in range(152):
    file = 'Y' + str(i+1) + str2
    hsv2 = cv2.imread(file,0)
    pos.append(imhist(hsv2))
```

3.3.2 Classificação e Cálculo das Métricas

Este código realiza uma análise de correlação entre as amostras do banco de dados e oito amostras padrão, também chamadas de padrões ouro, sendo quatro positivas e quatro negativas, a fim de classificar todas as imagens do *dataset* e calcular as métricas de desempenho.

O código começa criando duas listas vazias para armazenar as correlações entre a amostra que será classificada e os modelos positivos e negativos. Em seguida, ele percorre o banco de dados positivo e calcula a correlação entre cada elemento e cada um dos modelos positivos e negativos. A correlação é adicionada à lista de correlações positivas e, em seguida, essa lista é limpa. Em seguida, o mesmo procedimento é repetido para cada elemento do banco de dados negativo.

Depois de calcular as correlações com cada elemento dos modelos positivos e negativos, o código compara as médias das correlações para identificar se a amostra de dados é positiva ou negativa. Se a média da correlação com a amostra negativa for maior que a média da correlação com a amostra positiva, a amostra é considerada negativa, caso contrário, é considerada positiva. Em seguida, realiza a mesma rotina para outra amostra, sempre incrementando a contagem de casos classificados corretamente.

Por fim, o código calcula a porcentagem de casos positivos diagnosticadas no banco de dados positivo, a porcentagem de casos negativos diagnosticadas no banco de dados negativo e a porcentagem de casos certos considerando todas as amostras. A sensibilidade é dada pela porcentagem de casos positivos, a especificidade é dada pela porcentagem de casos negativos e a acurácia é dada pela porcentagem de casos certos, conforme equações 2.6, 2.7 e 2.8.

O algoritmo desenvolvido para a classificação e cálculo das métricas foi disponibilizado no apêndice A.

3.3.3 Escolha de Modelos

Como citado anteriormente cada amostra é comparada com outras oito amostras, as chamadas padrões ouro, quatro positivas e quatro negativas. Portanto, a escolha de tais modelos é crucial para garantir a qualidade da análise e a confiabilidade dos resultados. Eles servem como base para avaliar a precisão do método utilizado na análise de novas amostras. Se os padrões ouro não forem representativos das amostras reais, a análise pode ser prejudicada e os resultados serão menos confiáveis.

Inicialmente, havia sido estabelecido que o programa teria dez padrões ouro, cinco de cada rótulo, a fim de proporcionar um bom senso de generalidade para o classificador. Para selecionar tais amostras, foi iniciada uma análise visual do banco de dados, foram buscadas amostras que englobassem algumas características como: diferentes níveis de brilho, espessuras de crânio variadas e, no caso de positivas, dimensões distintas de neoplasias.

Após alguns testes, o desempenho estava abaixo do desejado, então foram adicionadas no programa funções de impressão das correlações das amostras com os modelos. Foi notado que alguns modelos estavam piorando o resultado ao invés de melhorar, em outras palavras, ele dava correlação maior com o rótulo oposto a si próprio. Logo, foram retirados tais padrões prejudiciais e mantidos somente quatro, dois de cada rótulo que aparentemente eram condizentes com seus rótulos.

Em seguida, foi acrescentada mais uma vaga de modelo positivo e foram sendo testadas amostra por amostra qual aumentaria a acurácia, quando se encontrou uma, a busca se encerrou. Ou seja, não se procurou no banco de dados inteiro qual teria a maior acurácia, somente qual aumentaria primeiro o valor da métrica em questão.

O mesmo procedimento foi feito mais uma vez com os padrões positivos e duas vezes com os padrões negativos, resultando em quatro amostras de cada rótulo. Ao tentar adicionar o quinto modelo, em ambos rótulos, foram testados todas as amostras do banco de dados, porém nenhuma aumentou a acurácia. Logicamente, optou-se por manter somente as oito.

3.4 Localização do Tumor

O desenvolvimento do algoritmo de localização do tumor envolve o programa de segmentação de imagem e o programa de adaptação da faixa de cores da imagem que serão passados como entrada da função de segmentação.

3.4.1 Segmentação

Foram realizados testes na seção 3.2.2 com a função `imadjust()` da IPT no Matlab. Como ela realiza a segmentação desejada nas imagens, optou-se por seguir com sua utilização no programa final. Entretanto, não há uma versão da `imadjust()` implementada nas bibliotecas mais conhecidas de Python, alguns fóruns recomendam a declaração dentro do próprio programa, e assim foi feito.

A ordem dos argumentos de entrada continuam os mesmos, a diferença é que no Matlab a faixa de entrada e saída das cores estava no intervalo $[0, 1]$, enquanto nesta função implementada a faixa está entre $[0, 255]$.

```
1  def imadjust(x,a,b,c,d):
2
3      shape=np.array(x.shape)
4      y = ((x - a) / (b - a)) * (d - c) + c
5
6      for j in range(shape[1]):
7          for i in range(shape[0]):
8              if y[i,j] < 0:
9                  y[i,j] = 0
10             elif y[i,j] > 255:
11                 y[i,j] = 255
12
13     return y
```

A equação na linha 4 é a parte mais importante da função, pois realiza a redistribuição das cores. Os laços e as funções condicionais foram adicionadas pois em alguns casos a função retornava valores fora do intervalo padrão. Isso acontece devido à subtração do *pixel* `x` pelo valor `a`, que pode resultar em um valor negativo, fazendo com que o valor de `y` relativo a este *pixel* também seja negativo.

3.4.2 Adaptação Iterativa do Intervalo de Segmentação

A variação da iluminação e brilho das amostras era capaz de afetar os resultados de segmentação por se tratar de uma função determinística. Porém, na maioria dos casos, ao tentar segmentar e destacar o tumor, a imagem resultante era formada por uma mancha da neoplasia e uma circunferência do crânio em volta. Logo, foi notado que havia uma proporção aproximadamente padronizada entre a quantidade de *pixels* pretos e o restante de *pixels* em uma imagem bem segmentada.

A proporção de *pixels* claros em relação aos *pixels* pretos desejada está por volta de $1,75\%$ (valor obtido experimentalmente). Para conseguir alterar este valor, é necessário mudar o intervalo de das cores de entrada da função `imadjust()`, no código isto é feito pelas variáveis `a` e `b`. Quando `a` e `b` são próximos de $'1'$, a maioria dos *pixels* tem intensidade menor que este valor, então a proporção de *pixels* claros cai. Quando `a` e `b` diminuem de valor, aumenta a quantidade de *pixels* com intensidade maior que eles, então a proporção aumenta.

Resumindo, o algoritmo declara `a = 0,5` e `a = 0,6`, ou seja, valores intermediários, em seguida funciona em *loop*: aplica a função `imadjust()` e calcula a proporção de *pixels* claros, se estiver baixa diminui os valores de `a` e `b`, se estiver alta aumenta `a` e `b`, se estiver aceitável interrompe o laço. O algoritmo completo se encontra no apêndice B.

3.5 Desenvolvimento do Software

Enfim, foi desenvolvido o programa que realmente faria o diagnóstico e a localização (caso positiva) de uma imagem de RM de um cérebro. Porém, o desenvolvimento foi dividido em duas etapas: de *back-end*, que faz os cálculos e chama as funcionalidades de diagnóstico e localização, e a etapa de *front-end*, que implementa a GUI.

3.5.1 *Back-end*

Conforme anteriormente, dentro da implementação do *back-end*, novamente dividiu-se o trabalho nas tarefas de diagnóstico e localização. Já tendo sido desenvolvidos vários pontos deste contexto, boa parte do que foi implementado no programa final acabou sendo igual ou até mesmo uma versão mais simplificada do que já havia sido feito antes.

No que se refere ao diagnóstico, a metodologia de leitura da imagem, obtenção do histograma, cálculo da correlação com os padrões ouro e classificação a partir da média foi implementada novamente. Desta vez, porém, não são lidas as imagens do banco de dados. Logicamente, é o usuário que envia sua amostra a ser analisada. Além disso, os histogramas dos modelos foram salvos como variáveis para não ser necessário ter as amostras em conjunto com o programa.

Já em relação à localização, ela só é realizada quando o classificador indica que a amostra é positiva e funciona exatamente conforme explicado na seção 3.4. A imagem segmentada é salva no mesmo diretório que se encontra o programa.

Todas estas funcionalidades foram aplicadas dentro de uma função que é chamada pela interface gráfica para realizar sua tarefa quando o usuário envia uma imagem. Ela está disponível no apêndice C.

3.5.2 *Front-end*

A implementação da interface gráfica com o usuário foi feita através da biblioteca PySimpleGUI. Através dela é possível gerar aplicações simples por meio de uma metodologia que considera que a janela gráfica é uma matriz, que consequentemente possui linhas e colunas.

Para o caso da aplicação deste trabalho, a janela possui quatro linhas: uma de texto, uma para entrada do arquivo, outra para os botões e a última com a impressão do diagnóstico. A primeira linha simplesmente imprime uma frase pedindo para escolher a imagem. A segunda linha é dividida em duas colunas: um campo de texto com o nome do arquivo e um botão de procura de arquivos, que está especificado para receber somente imagens em extensão `jpg`. A terceira linha tem os botões de enviar imagem e de sair da aplicação, ou seja, também está dividida em duas colunas. Já a última linha, que a princípio não tem nada, posteriormente imprime o resultado do diagnóstico.

Definido o *layout* da janela, o programa entra em *loop* esperando algum evento do usuário, que são basicamente dois: que ele feche a tela, neste caso a aplicação é interrompida, ou que ele envie uma imagem, o que fará com que ele chame a função de diagnóstico e localização passando para ela a imagem e um contador indicando quantas vezes já foi chamada esta função. Tal contador é usado para dar nomes diferentes às imagens segmentadas que serão salvas, de forma que evite sobreposições de arquivos no diretório. Por fim, o programa recebe o diagnóstico, imprime na quarta linha e aguarda outro evento, podendo receber outra imagem ou ser simplesmente fechado. O código implementado está disponível no apêndice D.

Com o programa finalizado, optou-se por passá-lo para um arquivo de extensão `.exe`, assim é possível executá-lo em qualquer computador, sem necessidade de ter Python ou outro *software* instalado. Isto foi realizado por meio da biblioteca PyInstaller, que gera o arquivo ao enviar o comando a seguir no terminal.

```
1 > pyinstaller --onefile --noconsole .\nomearquivo.py
```

Sendo "`-onefile`"usado para gerar um só arquivo e "`-noconsole`"para que não seja aberto o Prompt de Comando enquanto o aplicativo é executado.

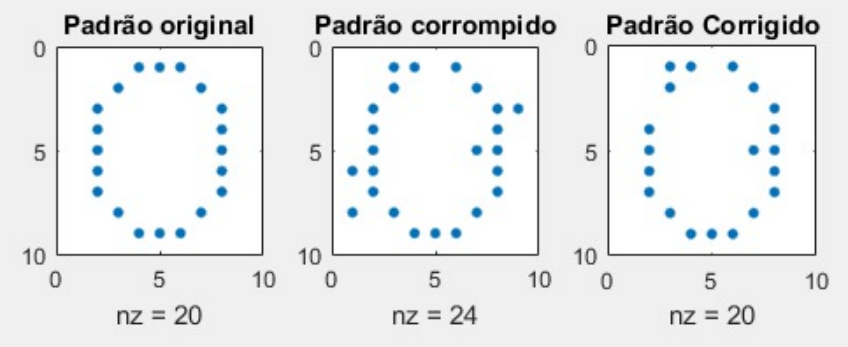
4 Resultados e Discussão

4.1 Testes Iniciais

4.1.1 Testes para diagnóstico

Os testes em pequena escala da Rede de Hopfield mostraram dois pontos: primeiro, a rede era capaz de atuar contra a corrupção dos modelos. Nos testes com adição de ruído aos padrões originais, havia, na maioria dos casos, aproximação da imagem à sua correspondente original.

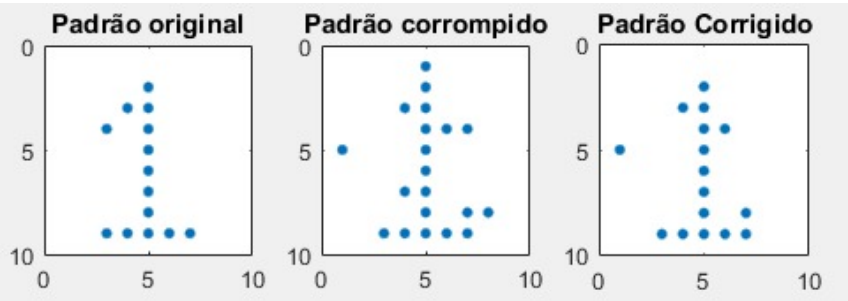
Figura 16 – Recuperação do símbolo '0' corrompido



Fonte: O próprio autor

Entretanto, o segundo ponto é que a rede implementada ainda não estava retornando exatamente aos padrões inicias, conforme é possível notar em ambas as figuras 16 e 17. É evidente que havia alguma inconsistência na implementação feita e não no método em si, já que estava sendo respeitado o limite de memórias fundamentais em relação à suas dimensões (equação 2.4).

Figura 17 – Recuperação do símbolo '1' corrompido



Fonte: O próprio autor

Em primeiro momento, tal problema foi deixado de lado, pois o processo estava apenas iniciando e ainda havia outros pontos a serem implementados, como o pré-processamento das amostras. Porém, é provável que o erro se encontrava na etapa de recuperação e não na de armazenamento, uma vez que o programa conseguia começar a regenerar a entrada, porém não evoluía na sequência.

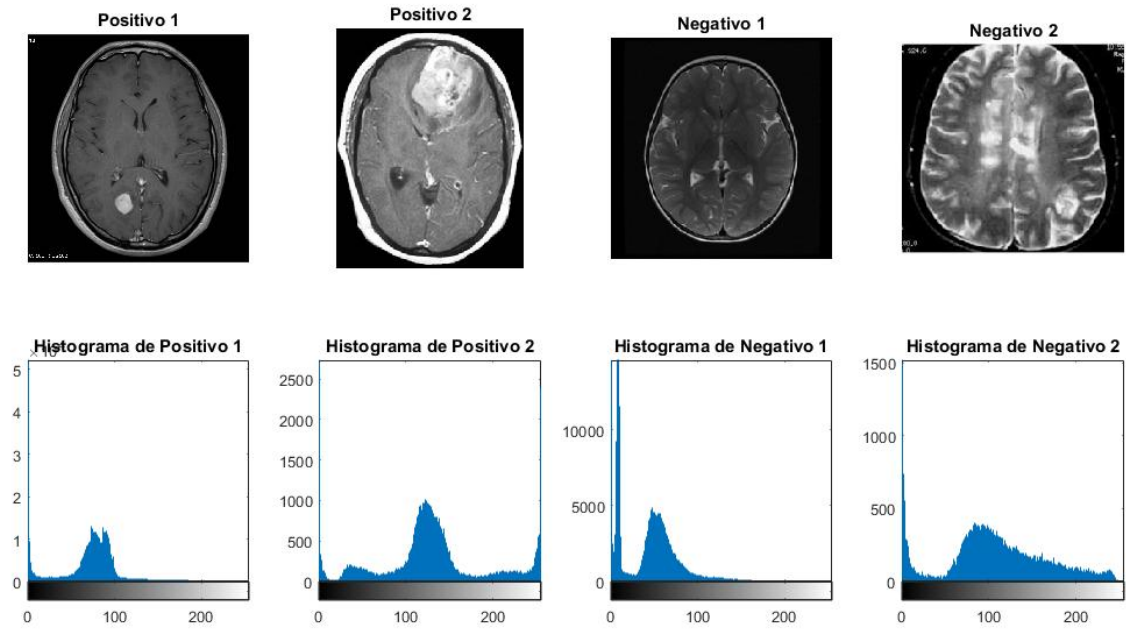
No caso do método voltar a ser desenvolvido, desta vez especificamente para o tratamento das amostras de RM, será necessária uma inspeção para otimizar o funcionamento. Independentemente disto, sua implementação em pequena escala já trouxe reflexões sobre os benefícios e limitações de métodos de ANN em geral.

É inegável que as redes neurais são capazes de se adaptar melhor que os métodos tradicionais e algumas vezes até possuem implementações mais simples por serem relativamente parecidas para várias aplicações e pela vasta quantidade de recursos à disponibilidade *online*. Por outro lado, a alta demanda de processamento e de quantidade de amostras para conseguir "aprender" satisfatoriamente são limitações sérias para diversas aplicações.

- **Histograma**

Ao plotar um histograma, a função `imhist()` também adiciona uma régua abaixo da janela gráfica, facilitando o discernimento das cores. A figura 18 demonstra alguns exemplos de amostras e seus histograma para exemplificar algumas das características mais importantes.

Figura 18 – Comparação entre Amostras e seus Histogramas plotados através da função `imhist()`



Fonte: O próprio autor

O histograma da figura de Negativo 1 representa bem uma amostra padrão negativa: a maior concentração de *pixels* está próxima de zero, pois o fundo é escuro. Além disso, há outro pico menor, referente ao cérebro. Já as outras cores são mais raras e estão mais distribuídas, ficando mais discretas no histograma. A amostra Negativo 2 possui um comportamento similar, porém o segundo pico é consideravelmente mais alargado por haver mais contraste entre as cores da região do cérebro.

Os histogramas das imagens positivas são similares aos negativos, principalmente em relação às aglomerações de *pixels* escuros do fundo e cinzentos da massa encefálica. Normalmente há um terceiro pico, menor ainda que os outros dois, quando um tumor está presente, assim como no final do histograma da amostra Positivo 2.

Vale ressaltar que dois pontos podem dificultar a classificação de amostras positivas: primeiro que o crânio pode ser confundido com um tumor do ponto de vista do histograma e segundo que em imagens escuras e com tumores pequenos, como a amostra Positivo 1, os histogramas podem acabar sendo muito parecidos com os negativos.

• Coeficiente de Correlação Linear

Os valores de correlação foram calculados para serem utilizados como métricas das mudanças testadas e é importante lembrar que a correlação analisa se duas curvas são similares, porém não são importantes os valores absolutos e, sim, se eles estão proporcionalmente relacionados. Logo, o efeito de diferenças entre resoluções de amostras é mitigado por esta métrica.

Tabela 2 – Correlações entre amostras positivas e negativas

Amostra	Positivo 1	Positivo 2	Negativo 1	Negativo 2
Positivo 1	$1,0000$	$0,9347$	$0,0393$	$0,1475$
Positivo 2	$0,9347$	$1,0000$	$0,0506$	$0,2306$
Negativo 1	$0,0393$	$0,0506$	$1,0000$	$0,5045$
Negativo 2	$0,1475$	$0,2306$	$0,5045$	$1,0000$

Fonte: O Próprio Autor

A tabela 2, mostra que, mesmo com características de brilho e contraste consideravelmente diferentes entre as imagens, a função de correlação linear foi capaz de relacionar as amostras do mesmo tipo.

Por mais que, às vezes, as correlações estavam próximas entre si, o programa retornou valores maiores para imagens do mesmo tipo em todos os casos. Ou seja, imagens positivas tinham maior correlação com imagens positivas e negativas com negativas. Vale ressaltar que as imagens já haviam sido escolhidas por serem consideravelmente distintas entre si.

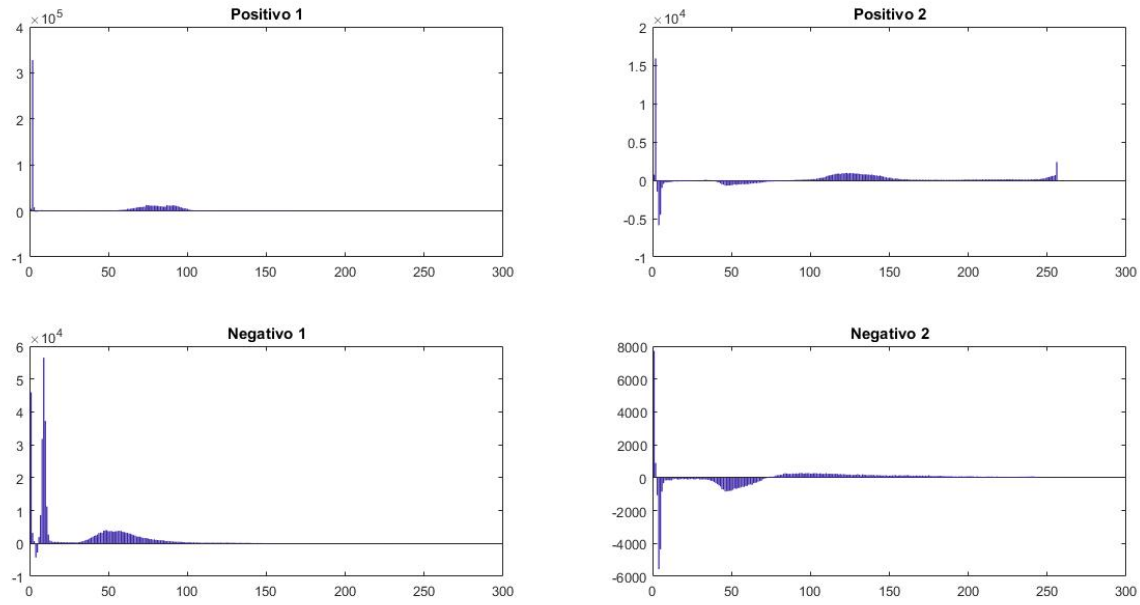
• Subtração de Histogramas

Conforme comentado no capítulo de Desenvolvimento, havia o interesse de manter somente o que se refere a um tumor nos dados passados para um classificador, já que o

restante dos elementos das imagens são comuns a praticamente todas as amostras.

Entretanto, a figura 19 evidencia que os histogramas que resultaram da operação de subtração não oferecem informação útil para o prosseguimento do desenvolvimento do programa de diagnóstico. Os elementos das RMs estão mais discretos, porém continuam visíveis, ou seja, eles não foram mitigados conforme era desejado. Além disso, as amostras de mesmo rótulo não apresentam um padrão visível. Em outras palavras, elas parecem mais diferentes e aleatórias do que antes da subtração.

Figura 19 – Histogramas de amostras subtraídos do modelo pelo Matlab



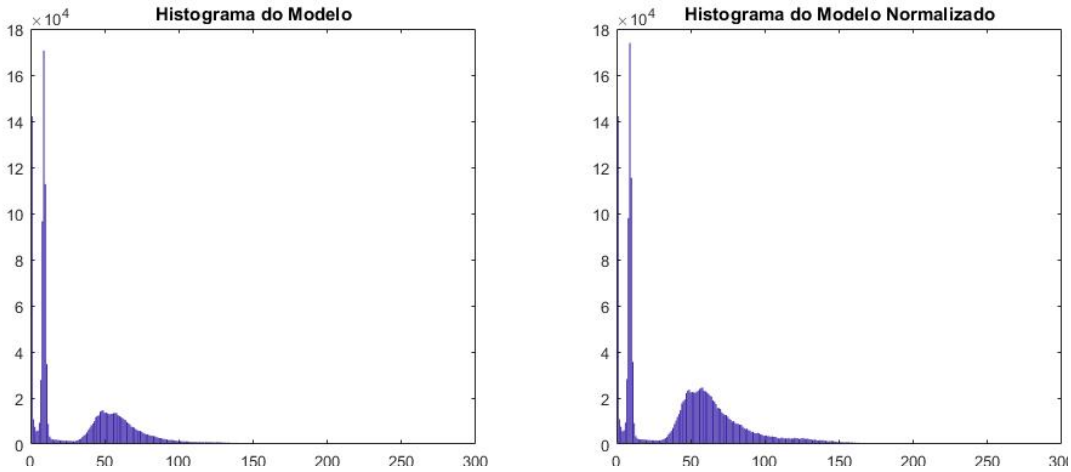
Fonte: O Próprio Autor

- **Função de Normalização do Histograma**

A figura 20 compara os histogramas da mesma amostra antes e depois de ser multiplicada pela função da figura 14. A principio há pouca diferença visual entre os dois gráficos. Isto ocorre pois a maior parte das cores está próxima de zero, enquanto a normalização dá maior ganho para cores intermediárias, o que acaba alterando pouco na visão geral.

A normalização, mesmo com amplitude aumentada, não interferiu significantemente nos resultados conforme pode-se notar nos valores de correlações da tabela 3 que estão próximos aos do modelo sem normalização.

Figura 20 – Histogramas da amostra modelo antes e depois da normalização



Fonte: O Próprio Autor

Além de alterar pouco, quando alterou não se mostrou muito promissora, já que aumentou a correlação com algumas amostras e diminuiu com outras de mesmo rótulo. Ou seja, ela não necessariamente aumentava a correlação ou abaixava para uma amostra positiva, por exemplo. Esta instabilidade e incoerência fizeram com que não fossem realizados mais testes com tal normalização.

Tabela 3 – Comparação entre as correlações lineares de um modelo original (negativo) e um modelo equalizado com algumas amostras do *dataset*

Modelo	Positivo 1	Positivo 2	Negativo 1	Negativo 2
Original	$0,0393$	$0,0506$	$0,2088$	$0,5045$
Normalizado	$0,0455$	$0,0363$	$0,2416$	$0,4807$

Fonte: O Próprio Autor

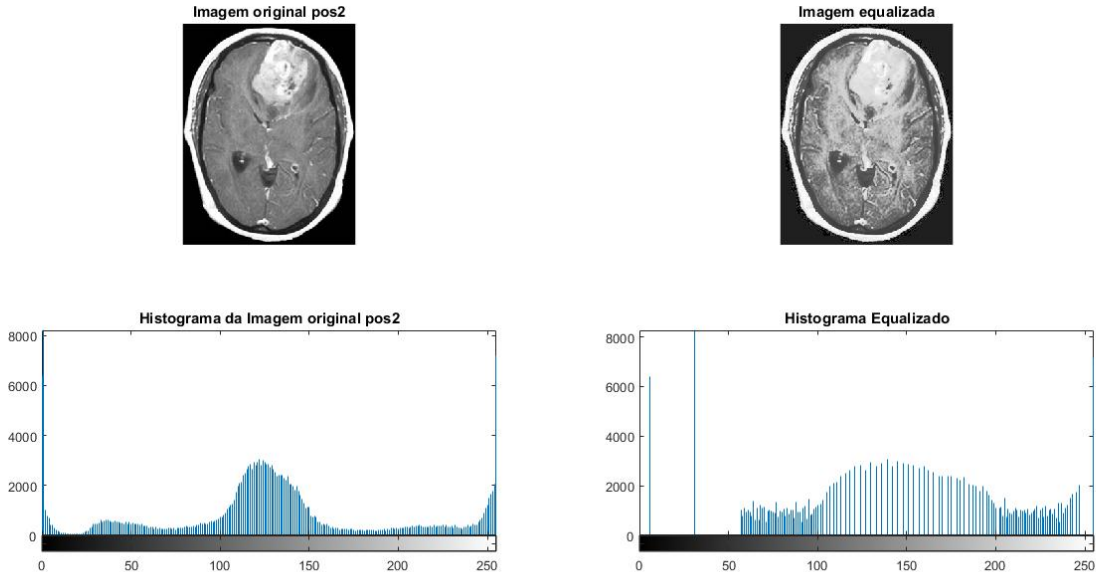
• Função de Equalização do Histograma

Havia uma expectativa considerável acerca da função de equalização de histogramas (`histeq()`). Isto porque, conforme já comentado anteriormente, as amostras seguem o mesmo padrão, porém algumas características individuais das imagens, como a dimensão de cada elemento, suas cores e suas posições, dificultavam a aplicação de métodos determinísticos para diagnóstico.

Logo, a ideia de equalizar os histogramas de todas as amostras poderia auxiliar consideravelmente neste processo, já que, ao menos teoricamente, as diferenças individuais de cores (e iluminação) seriam mitigadas. Tal benefício não seria exclusivo para o diagnóstico, mas também para a localização, para o caso de se utilizar uma metodologia determinística.

A figura 21 compara as imagens e os histogramas da amostra antes e depois de ser aplicada a função de equalização.

Figura 21 – Comparação entre amostra original e amostra após equalização do histograma



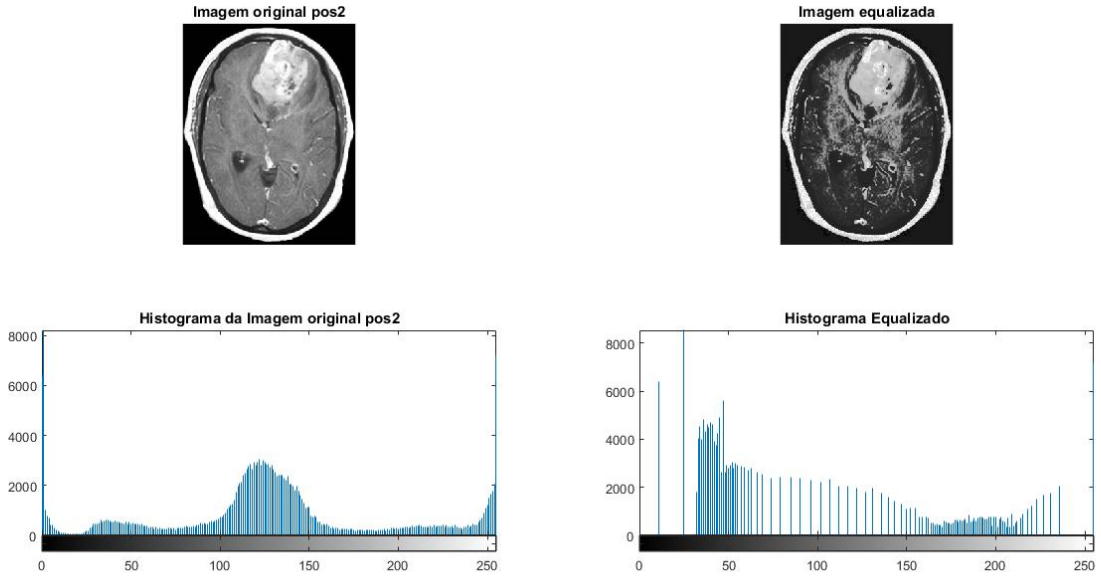
Fonte: O Próprio Autor

Entretanto, através da imagem da amostra equalizada, pode-se perceber que a `histeq()` acabou misturando demais as cores, comprometendo as fronteiras entre diferentes elementos da RM. Principalmente no que se refere à diferenciação entre cérebro e tumor, que foi prejudicada enquanto se buscava exatamente o contrário, que era destacar.

• **Equalização Guiada de Histograma**

Contudo, ainda resta analisar os resultados de `histeq()` guiada por uma Gaussiana Bimodal, presentes na figura 22.

Figura 22 – Comparação entre amostra original e amostra após equalização guiada do histograma a partir da função gaussiana bimodal



Fonte: O Próprio Autor

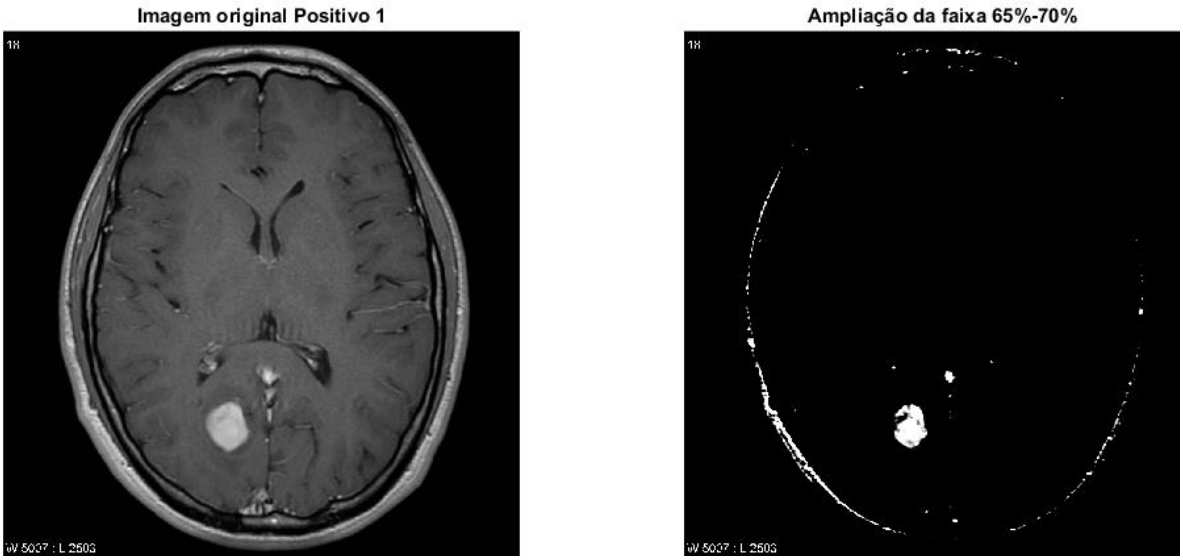
A inserção da função Gaussiana Bimodal como guia do histograma proporcionou uma diferença considerável na imagem equalizada. É visível que a maioria dos *pixels* que representam a massa encefálica tiveram suas cores alteradas para um tom mais escuro, o que era desejado, já que aumentaria o contraste. Todavia, persiste a semelhança entre partes do cérebro e o tumor, o que, novamente, é o oposto do desejado.

A possibilidade de se ficar ajustando os parâmetros da função de guia até encontrar valores que melhorassem a equalização foi descartada, uma vez que sua finalidade seria facilitar a implementação de classificadores e/ou localizadores. Acrescentar mais parâmetros de uma função determinística só aumentaria a complexidade sem garantias concretas de melhoria.

4.1.2 Testes para localização

A figura 23 demonstra a diferença entre a amostra original e o resultado de sua aplicação na função `imadjust()`.

Figura 23 – Aplicação da função `imadjust()` para destacar tumor cerebral em amostra



Fonte: O Próprio Autor

O resultado da ampliação da faixa de cores entre 65% e 70% produziu resultados muito satisfatórios, uma vez que colocou o tumor em posição de destaque ao aumentar o contraste da amostra. Isto acontece pois somente alguns poucos elementos da imagem estavam acima de 70% da intensidade de cor (um deles sendo o tumor), logo todo o resto estava abaixo de 65% e, consequentemente, era levado para a cor preta (zero). O ponto negativo é que nem todas amostras irão apresentar uma segmentação tão boa para o

mesmo intervalo adotado para este exemplo, logo seria necessário um ajuste específico para cada amostra de acordo com seu brilho intrínseco.

4.2 Diagnóstico

No que se refere ao diagnóstico, é valido discutir alguns pontos sobre a escolha dos padrões ouro. É provável que o primeiro teste com dez modelos estava generalizando as amostras. Porém, é possível que as amostras eram mais similares a algum modelo em específico, ou seja, elas não eram uniformemente distribuídas. Logo, a explicação mais plausível é que manter modelos de ocasiões que podem ocorrer, embora raras, acabava atrapalhando a classificação de amostras mais frequentes.

Em relação à quantidade final de padrões ouro, é evidente que cinco (de cada rótulo) proporcionaria mais generalidade, porém isso poderia acabar sendo prejudicial, pois, ao tentar classificar uma amostra, ao mesmo tempo em que há um modelo parecido (correlação alta), há outros quatro que não são tão parecidos (correlação baixa), fazendo a média ficar com um valor baixo. Por outro lado, poucos modelos, como dois em cada rótulo por exemplo, poderia ter o efeito contrário, ou seja, causar uma perda de generalidade. Logo, quatro de cada, oito no total, acaba sendo um valor que balanceia melhor estes pontos desejados.

Ao final dos testes de diagnóstico, foram obtidos os valores da tabela 4 para as métricas de desempenho.

Tabela 4 – Dados de Acurácia, Sensibilidade e Especificidade atingidos neste trabalho

Acurácia	Sensibilidade	Especificidade
$0,8000$	$0,8421$	$0,7347$

Fonte: O Próprio Autor

O valor de acurácia atingiu um patamar satisfatório e está dentro da faixa de resultados em outros trabalhos, conforme listado na tabela 1. Logicamente, o valor está um pouco menor que o da maioria, entretanto é válido relembrar que a abordagem é consideravelmente menos complexa do ponto de vista computacional. Redes Neurais requerem etapas de treinamento, validação e teste o que acaba exigindo mais processamento e tempo, também.

Alguns dos pontos positivos do método empregado neste trabalho são justamente o menor tempo de processamento e a menor complexidade computacional. Além de, claro, os recursos de localização e de interface gráfica com o usuário.

Outro ponto de discussão é que a sensibilidade (média de acerto em amostras positivas) teve resultado mais de 10% superior ao da especificidade (média de acerto em amostras negativas). Não existe uma explicação concreta para isso, porém é possível

que alguns artefatos nas imagens, ou mesmo pontos dissemelhantes nos cérebros tenham ficado similares à tumores quando calculado o histograma.

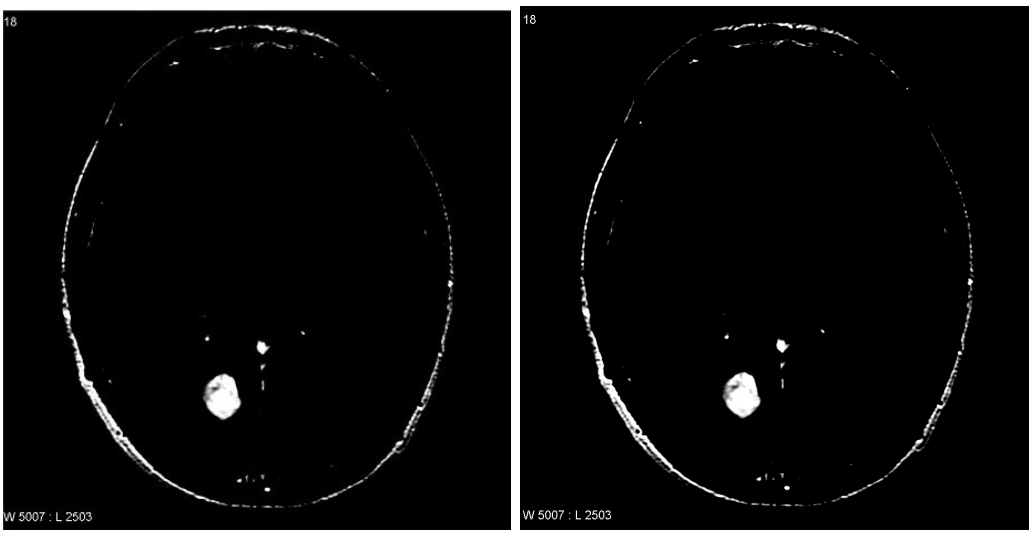
De qualquer forma, esta diferença entre os valores das figuras de mérito também é vista em outros trabalhos citados na tabela 1. Portanto, pode ser algo recorrente em sistemas de CAD em imagens de RM do cérebro.

4.3 Localização do Tumor

4.3.1 Segmentação

Primeiro desejou-se analisar se a função `imadjust()`, utilizada para segmentação, gerava resultados iguais para Python e Matlab, principalmente porque, em Python, ela não é importada de uma biblioteca, mas sim implementada no próprio programa. A figura 24 compara duas imagens segmentadas com ênfase na faixa de cores entre 55% e 70%.

Figura 24 – Comparação entre `imadjust()` original em Matlab (a) e a versão implementada em Python (b)



(a) (b)

Fonte: O Próprio Autor

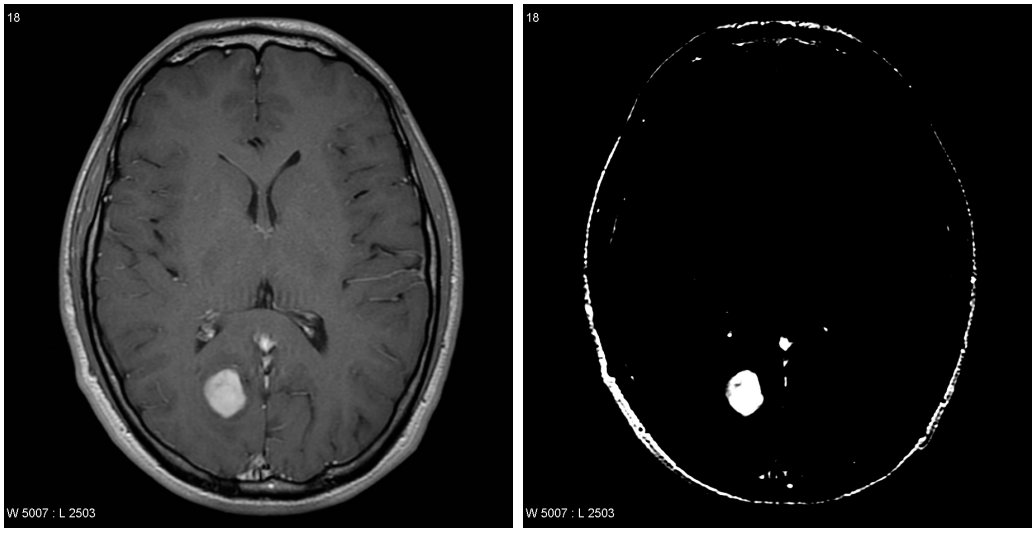
Comparando as figuras é justo dizer que os resultados são aproximadamente iguais, com nenhuma distinção que chame a atenção.

Outro ponto a ser discutido é a presença do crânio na imagem segmentada. A princípio o plano era alterar o algoritmo para excluir este elemento da figura, entretanto quando a circunferência do crânio está presente, ela age como uma referência que facilita o entendimento da localização do tumor e também de sua dimensão. Além disso, sua implementação acrescentaria um passo de dificuldade considerável na etapa de localização, levando em conta que cada imagem possui sua própria característica.

4.3.2 Adaptação Iterativa do Intervalo de Segmentação

A figura 25 mostra um exemplo de segmentação gerada pelo algoritmo implementado.

Figura 25 – Comparação entre amostra original (a) e resultado da segmentação através do algoritmo iterativo (b)

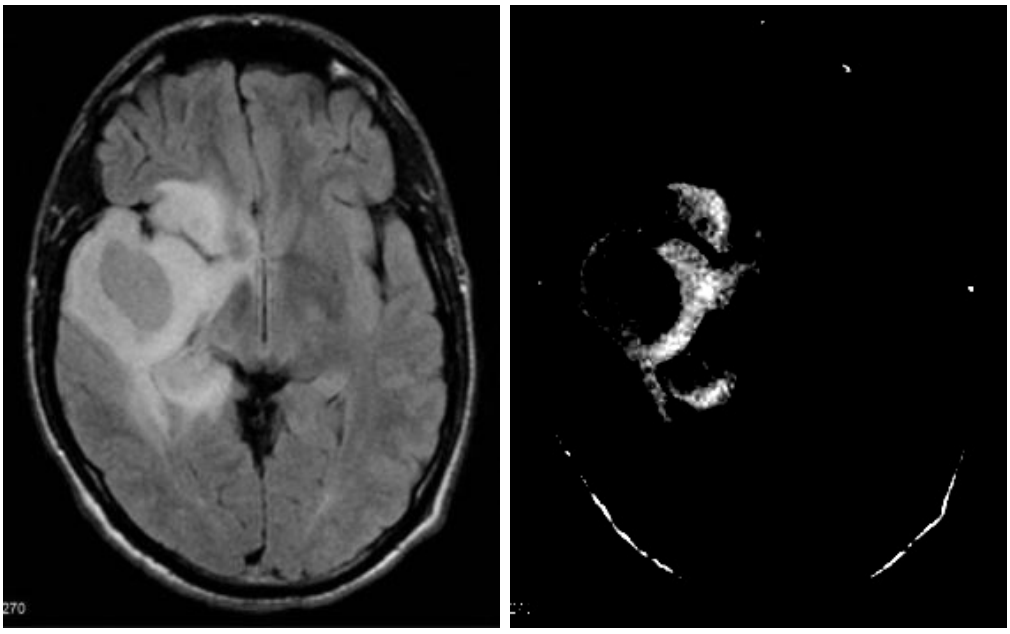


(a) (b)

Fonte: O Próprio Autor

A imagem 25 conta com um tumor pequeno, mas a figura 26 exibe um tumor maior.

Figura 26 – Comparação entre amostra original com tumor mais extenso (a) e resultado da segmentação através do algoritmo iterativo (b)



(a) (b)

Fonte: O Próprio Autor

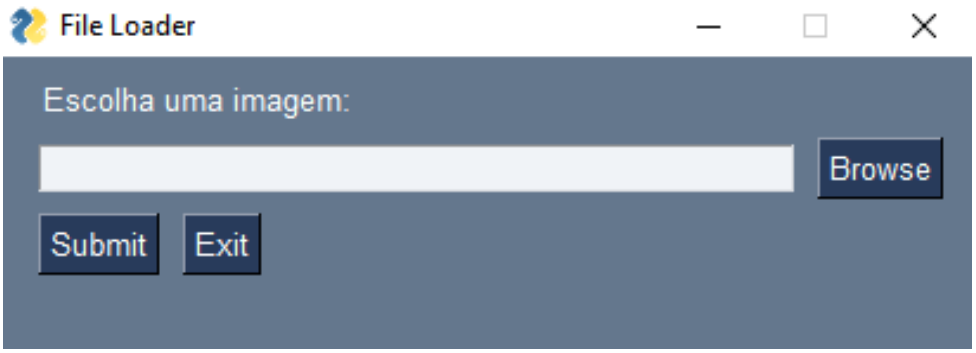
É notável que a qualidade da localização é inversamente proporcional ao tamanho do tumor. Isto porque, a razão de *pixels* brancos, que é utilizada como critério de parada do algoritmo iterativo, está melhor ajustada para imagens de tumores pequenos. Logo, quando a neoplasia é mais extensa, uma boa segmentação teria uma proporção de *pixels* brancos maior, entretanto esta proporção é fixa no programa atual.

O algoritmo de localização foi mantido desta maneira por dois motivos: primeiro, a maioria dos tumores presentes no *dataset* são pequenos. Além disso, é válido relembrar que o objetivo deste trabalho é somente auxiliar no diagnóstico. Então se o tumor já é grande, não há muita dúvida ou necessidade de juntar vários métodos de diagnóstico para se ter certeza. Então conclui-se que ele seja muito mais útil para casos de tumores menores, que estejam no início do seu desenvolvimento, permitindo assim, o seu diagnóstico precoce.

4.4 Desenvolvimento do Software

Finalmente, após integrar todas as etapas citadas anteriormente, o programa foi finalizado. A figura 27 exibe seu *layout* de abertura.

Figura 27 – Abertura do Programa



Fonte: O Próprio Autor

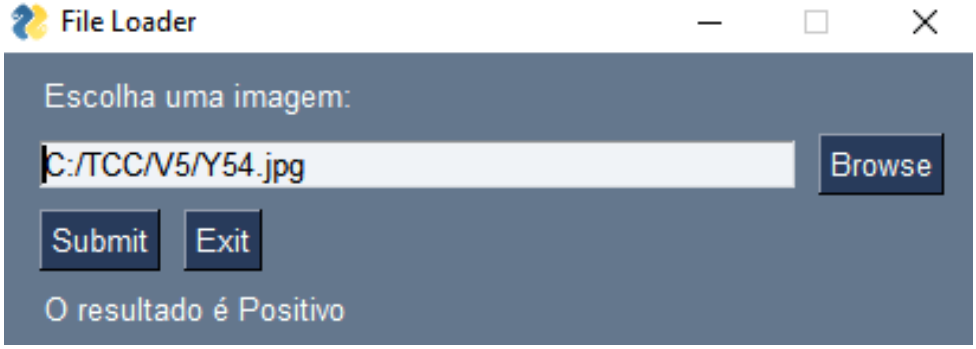
É interessante notar como os conceitos de linha e coluna utilizados no desenvolvimento são exibidos no resultado final. A figura 28 exibe a resposta do programa para uma amostra negativa. Ênfase na quarta linha que passa a ser exibida.

Por fim, a tela do *software* ao receber uma amostra positiva é mostrada na figura 29. Um ponto a ser comentado é que o algoritmo demora mais para imprimir a mensagem de resultado positivo, isso acontece pois toda a etapa de localização é realizada em conjunto neste caso. A imagem segmentada, aliás, é salva no mesmo diretório em que se encontra o executável.

Figura 28 – Resultado Negativo

Fonte: O Próprio Autor

Figura 29 – Resultado Positivo



Fonte: O Próprio Autor

5 Conclusões e Trabalhos Futuros

Com base nos resultados obtidos neste trabalho, é possível concluir que o método proposto de detecção e localização de tumores cerebrais por meio de correlação linear entre histogramas de amostras com padrões ouro apresentou resultados satisfatórios, com uma taxa de acerto de 80% em diagnóstico e segmentação promissora das amostras para facilitação da visualização de neoplasias.

Os testes iniciais em Matlab foram fundamentais para o desenvolvimento da metodologia. Desde os primeiros experimentos, eles possibilitaram o estudo mais aprofundado das amostras, auxiliaram na criação e validação de hipóteses e, mesmo quando não retornaram resultados positivos, eles auxiliavam no entendimento e desenvolvimento da aplicação.

Após diversos testes, foi optado por realizar o diagnóstico através do cálculo da correlação linear entre a amostra e os oito modelos padrões, quatro positivos e outros quatro negativos. Não foram utilizadas operações de equalização, normalização ou subtração, então a metodologia para alcançar maior generalidade e, consequentemente, melhores resultados, foi a escolha de padrões ouro que conseguissem balancear características de ambas as amostras mais frequentes e as mais raras.

Em relação à localização, ficou claro em testes iniciais que era necessário segmentar a imagem e isto foi feito através da função `imadjust()`, primeiro no Matlab através da IPT e, em seguida, em Python implementada no próprio programa. Para contornar as inconstâncias das amostras foi criado um algoritmo que altera o intervalo de segmentação das imagens através da proporção de pixels claros. Tal programa é capaz de destacar tumores pequenos e médios e contrastá-lo dos seus arredores de maneira convincente, porém há espaço para melhora em casos de tumores mais extensos.

É importante destacar que o método proposto não utilizou técnicas de aprendizado de máquina, mas sim técnicas de processamento de imagens e análise estatística. Redes neurais, como o caso testado de Hopfield, possuem grande utilidade neste contexto, porém são consideravelmente mais complexas computacionalmente e já foram extensamente estudadas nesta aplicação.

Além disso, é desenvolvida uma interface gráfica do usuário através da biblioteca PySimpleGUI, para facilitar o acesso e a utilização do programa. Seu *layout* é inspirado em linhas e colunas de uma matriz e conta com um botão de pesquisa de imagens nos documentos, além de caixas de texto. Já seu funcionamento se baseia na espera de eventos do usuário, que pode fechar o programa ou inserir uma imagem, este último inicia a rotina de diagnóstico e, caso seja positiva, a rotina de localização e armazenamento da imagem segmentada. O *software* é simples, porém é eficaz e autônomo, ou seja, não requer a presença de outros arquivos ou a instalação de Python no computador.

Vale ressaltar que o método proposto apresenta algumas limitações, como a dependên-

cia da qualidade das amostras utilizadas e a sensibilidade a muitas variações de iluminação e contraste nas imagens, por se tratarem de funções determinísticas. Ainda assim, os resultados obtidos são encorajadores, indicando que o método pode ser uma alternativa promissora e de baixo custo para a detecção de tumores cerebrais em imagens de ressonância magnética.

5.1 Trabalhos Futuros

A seguir são listadas algumas sugestões de temas e melhorias para trabalhos futuros.

- Desenvolvimento de programa para otimizar e agilizar a seleção de padrões ouro;
- Análise da metodologia de correlação linear de histogramas fazendo uso de um banco de dados mais extenso e padronizado;
- Desenvolvimento de algoritmo mais eficiente de localização do tumor, segmentando melhor imagens com tumores maiores;
- Possibilitar o armazenamento da imagem segmentada em outros formatos mais próximos à aplicação de cirurgia robótica;
- Desenvolvimento de braço robótico que segue dados de localização de tumor, simulando uma cirurgia robótica;
- Elaboração de interface gráfica mais completa, com mais funções para o usuário e mais apelativa visualmente.

Referências

BIOMEDICINA PADRÃO. *Planos e termos de referência utilizados na descrição anatômica*. 2018. Acesso em: 13 de fevereiro de 2023. Disponível em: <https://www.biomedicinapadrao.com.br/2018/10/planos-e-termos-de-referencia.html>. 28

CHAPLOT, S.; PATNAIK, L. M.; JAGANNATHAN, N. R. Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network. *Biomedical signal processing and control*, Elsevier, v. 1, n. 1, p. 86–92, 2006. 42, 43

DEANGELIS, L. M. Brain tumors. *New England journal of medicine*, Mass Medical Soc, v. 344, n. 2, p. 114–123, 2001. 28

DEY, S. *Hands-On Image Processing with Python*. [S.l.]: Packt Publishing, 2018. 31

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. New York: John Wiley & Sons, 2000. 32

EL-DAHSHAN, E.-S. A.; HOSNY, T.; SALEM, A.-B. M. Hybrid intelligent techniques for mri brain images classification. *Digital signal processing*, Elsevier, v. 20, n. 2, p. 433–441, 2010. 42, 43

EL-DAHSHAN, E.-S. A. et al. Computer-aided diagnosis of human brain tumor through mri: A survey and a new algorithm. *Expert systems with Applications*, Elsevier, v. 41, n. 11, p. 5526–5545, 2014. 42, 43

FUNCIONALITÁ. *Saiba o que são tumores cerebrais e quais os principais sintomas*. 2020. Acesso em: 24 de agosto de 2022. Disponível em: <https://www.funcionalita.com.br/o-que-sao-tumores-cerebrais-e-principais-sintomas>. 29

GONZALEZ, R.; WOODS, R.; EDDINS, S. *Digital Image Processing Using MATLAB*. [S.l.]: Pearson Education, 2004. 31, 36, 53

GUIMARÃES, P. R. B. *Métodos quantitativos estatísticos*. 2. ed. Curitiba: IESDE Brasil, 2018. 35

HAYKIN, S. *Redes Neurais: Princípios e prática*. 2. ed. [S.l.]: Bookman, 2001. 36, 37

HEATON, J. *Artificial intelligence for humans, volume 3: Deep learning and neural networks*. [S.l.]: Heaton Research, Inc., 2015. 36

INSTITUTO NACIONAL DO CÂNCER. *Estatísticas de Câncer*. 2022. Acesso em: 24 de agosto de 2022. Disponível em: <https://www.gov.br/inca/pt-br/assuntos/cancer/numeros/>. 30

JAFARI, M.; KASAEI, S. Automatic brain tissue detection in mri images using seeded region growing segmentation and neural network classification. *Australian Journal of Basic and Applied Sciences*, v. 5, n. 8, p. 1066–1079, 2011. 41, 43

KAGGLE. *Brain MRI Images for Brain Tumor Detection*. 2019. Acesso em: 25 de abril de 2022. Disponível em: <https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection?resource=download>. 45, 46, 47, 48

KHAN, H. A. et al. Brain tumor classification in mri image using convolutional neural network. *Math. Biosci. Eng*, v. 17, n. 5, p. 6203–6216, 2020. 28

LANG, R.; ZHAO, L.; JIA, K. Brain tumor image segmentation based on convolution neural network. In: IEEE. *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. [S.l.], 2016. p. 1402–1406. 28

MACHADO, A.; HAERTEL, L. M. *Neuroanatomia funcional*. Brasil: Atheneu, 2013. 27

MANN, P. *Introductory Statistics*. 7. ed. [S.l.]: John Wiley & Sons, 2010. 38

MATHWORKS. *Image Processing Toolbox*. 2021. Acesso em: 20 de fevereiro de 2023. Disponível em: <https://www.mathworks.com/products/image.html>. 34

MCFALINE-FIGUEROA, J. R.; LEE, E. Q. Brain tumors. *The American journal of medicine*, Elsevier, v. 131, n. 8, p. 874–882, 2018. 29

MORETTIN, P.; BUSSAB, W. *Estatística básica*. 9. ed. São Paulo: Saraiva Educação S.A., 2017. 38

NATIONAL BRAIN TUMOR SOCIETY. *Brain Tumor Facts*. 2021. Acesso em: 21 de março de 2023. Disponível em: <https://braintumor.org/brain-tumors/about-brain-tumors/brain-tumor-facts/>. 25

NATIONAL INSTITUTE OF BIOMEDICAL IMAGING AND BIOENGINEERING. *Magnetic Resonance Imaging (MRI)*. 2013. Acesso em: 20 de fevereiro de 2023. Disponível em: <https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri>. 30

PFIZER. *Sistema Nervoso Central*. 2019. Acesso em: 31 de agosto de 2022. Disponível em: <https://www.pfizer.com.br/sua-saude/sistema-nervoso-central>. 27

PHILLIPS. *MR 7700*. 2023. Acesso em: 21 de fevereiro de 2023. Disponível em: <https://www.usa.philips.com/healthcare/product/HCNMRF429/mr-7700>. 31

PYPI. *Imhist*. 2021. Acesso em: 21 de abril de 2023. Disponível em: <https://pypi.org/project/imhist/>. 57

PYSIMPLEGUI. *PySimpleGUI: Python GUIs for Humans*. 2023. Acesso em: 9 de abril de 2023. Disponível em: <https://www.pysimplegui.org/en/latest/>. 40

SALDANHA, M. F.; FREITAS, C. Segmentação de imagens digitais: Uma revisão. *Divisão de Processamento de Imagens-Instituto Nacional de Pesquisas Espaciais (INPE), São Paulo*, 2009. 33

SHAPIRO, L. G.; STOCKMAN, G. C. *Computer Vision*. [S.l.]: Pearson, 2001. 34

SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. *Redes neurais artificiais para engenharia e ciências aplicadas*. [S.l.]: Artliber Editora, 2010. 37

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern recognition*. [S.l.]: Elsevier, 2006. 31

UNIMED. *A importância da água no corpo humano: tire todas suas dúvidas*. 2020. Acesso em: 17 de março de 2023. Disponível em: <https://www.unimed.coop.br/viver-bem/saude-em-pauta/a-importancia-da-agua-no-corpo-humano-tire-todas-as-suas-duvidas>. 30

ZHANG, Y. et al. A hybrid method for mri brain image classification. *Expert Systems with Applications*, Elsevier, v. 38, n. 8, p. 10049–10053, 2011. 41, 43

ZÖLLNER, F. G.; EMBLEM, K. E.; SCHAD, L. R. Svm-based glioma grading: optimization by feature reduction analysis. *Zeitschrift für medizinische Physik*, Elsevier, v. 22, n. 3, p. 205–214, 2012. 42, 43

Apêndice A - Código de Cálculo das Métricas de Desempenho

```
correlacaop = []
correlacaon = []
certo=0
n=0
p=0
print("Banco de dados Positivo")
for j in range(len(pos)):
    mediap=0
    median=0
    #print("Correlacoes com positivos")
    for i in range(len(histp)):
        correlacaop.append(np.corrcoef(pos[j],histp[i])[0,1])
        mediap=mediap + correlacaop[i]

    correlacaop.clear()
    #print("Correlacoes com negativos")
    for i in range(len(histn)):
        correlacaon.append(np.corrcoef(pos[j],histn[i])[0,1])
        median=median + correlacaon[i]

    correlacaon.clear()
    mediap=mediap/len(histp)
    median=median/len(histn)
    #print("Media da Correlacao com modelos positivos:",mediap)
    #print("Media da Correlacao com modelos negativos:",median)

    if median > mediap:
        print(j+1,"e Negativo")
        n=n+1
    else:
        print(j+1,"e Positivo")
        p=p+1
        certo=certo+1

porcentp=p/(p+n) #Sensibilidade
n=0
p=0

print("Banco de dados Negativo")
for j in range(len(neg)):
```

```
41      mediap=0
42      median=0
43      #print("Correlacoes com positivos")
44      for i in range(len(histp)):
45          correlacaop.append(np.corrcoef(neg[j],histp[i])[0,1])
46          mediap=mediap + correlacaop[i]
47
48      correlacaop.clear()
49      #print("Correlacoes com negativos")
50      for i in range(len(histn)):
51          correlacaon.append(np.corrcoef(neg[j],histn[i])[0,1])
52          median=median + correlacaon[i]
53
54      correlacaon.clear()
55      mediap=mediap/len(histp)
56      median=median/len(histn)
57      #print("Media da Correlacao com modelos positivos:",mediap)
58      #print("Media da Correlacao com modelos negativos:",median)
59
60      if median > mediap:
61          print(j+1,"e Negativo")
62          n=n+1
63          certo=certo+1
64      else:
65          print(j+1,"e Positivo")
66          p=p+1
67
68  porcentn = n/(p+n)  #Especificidade
69  porcent = certo/(152+98)  #Acuracia
70  print("Porcentagem de casos positivos:",porcentp)
71  print("Porcentagem de casos negativos:",porcentn)
72
73  print("Porcentagem de casos certos:",porcent)
```

Apêndice B - Algoritmo de Adaptação Iterativa do Intervalo de Segmentação

```
a=0.5
b=0.6
jj=0
while jj<20:
    arr3=imadjust(arr,a*255,b*255,0,255)
    arr2= np.array(arr3)
    soma=0
    shape=np.array(arr.shape)
    for j in range(shape[1]):
        for i in range(shape[0]):
            soma=soma + (arr2.item(i,j)/255)
    razao = soma/arr2.size
    if razao<0.015:
        a=a-0.025
        b=b-0.025
        if a>1:
            a=1
        elif a<0:
            a=0
        if b>1:
            b=1
        elif b<0:
            b=0
    elif razao>0.02:
        a=a+0.025
        b=b+0.025
        if a>1:
            a=1
        elif a<0:
            a=0
        if b>1:
            b=1
        elif b<0:
            b=0
    else:
        break
    jj=jj+1
```

Apêndice C - Função de Diagnóstico e Localização de Tumores do Programa Final

```
def leImagem(imageadd,cont):
    histp = []
    histn = []
    correlacaop = []
    correlacaon = []
    a=0.5
    b=0.6
    ##########  MODELOS POSITIVOS

    hmp1=[252471,7029,4074,2565,1989,1815,1836,1956,2022,2244,2247,
          2523,2442,2661,2814,3030,2874,2796,2544,2505,
          2319,2448,2472,2640,2511,2682,2736,2853,3204,2754,
          3156,2973,2889,3219,3393,3222,3159,3501,3558,3819,
          3822,4215,4305,4314,4761,5019,5355,5883,6300,6882,
          7767,8376,9147,9486,10317,11010,11865,13140,13539,13290,
          12747,12579,12333,12060,12075,12567,12432,12720,12606,13107,
          14205,16551,17307,17652,15288,11997,8409,5562,3726,2721,
          2394,2001,1965,1791,1719,1749,1629,1602,1614,1494,
          1497, 1614, 1623, 1641, 1467, 1452, 1350, 1152, 1269, 1122,
          1176, 1101, 1071,984,957,975,903,795,759,  843,
          837,783,723,753,693,681,681,636,633,606,
          537,582,528,621,552,537,453,516,582,552,
          519,423,486,528,441,459,429,426,474,387,
          420,444,402,450,447,396,381,417,384,303,
          411,438,417,351,399,450,435,435,486,402,
          390,468,405,447,390,423,477,477,474,483,
          507,606,513,480,417,588,519,522,507,564,
          606,558,564,672,660,648,654,726,690,735,
          720,801,837,831,951,888,1017,864,855,927,
          975,873,828,825,888,864,828,876,780,816,
          783,723,693,603,579,654,693,633,540,540,
          570,573,588,447,432,483,375,369,357,312,
          330,222,258,195,219,216,183,132,156,105,
          126, 99,102, 93, 75, 69, 54, 75, 57, 30,
          42, 54, 30, 42,  9, 39]
    histp.append(np.array(hmp1))

```

```
38     hmp2=[40632,22779 ,5772 ,2166 ,2358 ,1062 , 810 , 636 , 555 , 492 , 456
       , 447,
39         342 , 345 , 390 , 288 , 285 , 984 , 828 , 261 ,1578 , 345 , 249 ,
       207,
40         222 , 171 , 192 , 180 , 192 , 192 , 180 , 216 , 177 , 201 , 234 ,
       189,
41         240 , 300 , 252 , 234 , 270 , 303 , 264 , 327 , 294 , 345 , 309 ,
       351,
42         324 , 339 , 342 , 312 , 234 , 345 , 336 , 333 , 369 , 282 , 378 ,
       441,
43         516 , 573 , 555 , 600 , 645 , 582 , 732 , 804 , 789 , 879 , 990 ,
       918,
44         975 ,1125 ,1086 ,1140 ,1188 ,1149 , 993 , 957 ,1098 ,1026 ,1167
       ,1086,
45         1101 ,1134 , 921 ,1029 ,1050 ,1155 ,1086 ,1062 ,1032 ,1131 ,1146
       ,1119,
46         1140 ,1230 ,1044 ,1332 ,1281 ,1257 ,1035 , 966 ,1053 , 993 , 855 ,
       753,
47         735 , 657 , 531 , 453 , 426 , 312 , 309 , 318 , 285 , 294 , 198 ,
       192,
48         195 , 123 , 129 , 144 , 120 , 132 , 153 , 126 , 114 , 174 , 117 ,
       162,
49         78 , 111 , 156 ,  99 , 111 , 102 , 105 , 108 , 111 , 117 ,  87 ,
       114,
50         66 ,108 , 90 , 81 ,102 , 90 , 81 , 63 , 72 , 57 , 63 , 75,
51         69 , 75 , 60 , 48 ,102 , 84 , 60 , 54 , 72 , 75 , 45 , 51,
52         39 , 36 , 33 , 51 , 48 , 30 , 42 , 30 , 33 , 18 , 21 , 39,
53         30 , 24 , 12 , 12 , 15 , 18 , 24 , 18 , 30 ,  9 , 24 , 12,
54         9 , 6 , 9 ,12 , 6 , 3 ,12 , 9 , 9 , 9 , 6 , 9,
55         12 , 9 , 9 , 6 , 3 , 3 , 0 , 0 , 3 , 0 ,12 , 0,
56         0 , 6 , 3 , 0 , 0 , 0 , 0 , 0 , 0 , 3 , 0 , 0,
57         3 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0,
58         0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0,
59         0 , 0 , 0 , 0]
60     histp.append(np.array(hmp2))
61
62     hmp3=[18603, 983667 ,28641 ,13293 , 9627 , 7530 , 6291 , 5793 , 5202 ,
       4521,
63         3684 ,3684 ,3360 ,3537 ,3369 ,3204 ,3135 ,2814 ,2859 ,2826,
64         2823 ,2829 ,2766 ,2655 ,2940 ,3351 ,3261 ,3306 ,3027 ,3090,
65         2874 ,2865 ,3060 ,2760 ,2925 ,2877 ,2913 ,3330 ,3357 ,3375,
66         3387 ,3504 ,3486 ,3414 ,3639 ,3675 ,3927 ,3774 ,4155 ,4881,
67         5037 ,5127 ,5178 ,5664 ,6318 ,7146 ,8226 ,8685 ,9306 ,9336,
68         10578 ,14661 ,15654 ,16101 ,17886 ,20079 ,21687 ,23052 ,25203
       ,26253,
69         27306 ,27198 ,28560 ,39324 ,37761 ,35592 ,35121 ,34752 ,35400
       ,34959,
```

```
70          33438 ,32013 ,30840 ,29406 ,27750 ,27813 ,38031 ,35436 ,34707
    ,35394,
71          36009 ,34848 ,33135 ,30801 ,26781 ,22599 ,19149 ,16314 ,17721
    ,13518,
72          9510 ,7032 ,5604 ,4422 ,4002 ,3591 ,3291 ,2910 ,2841 ,2769,
73          3204 ,2976 ,2760 ,2493 ,2571 ,2211 ,2286 ,2157 ,2037 ,2070,
74          1899 ,1962 ,2097 ,2136 ,1962 ,2025 ,1983 ,2067 ,1833 ,1893,
75          1986 ,1863 ,1716 ,1833 ,1821 ,1809 ,1884 ,1767 ,1767 ,1641,
76          1587 ,1632 ,1461 ,1560 ,1395 ,1464 ,1524 ,1593 ,1638 ,1485,
77          1509 ,1350 ,1545 ,1350 ,1350 ,1332 ,1302 ,1149 ,1287 ,1401,
78          1242 ,1266 ,1155 ,1254 ,1185 ,1071 ,1233 ,1119 ,1149 ,1074,
79          1041 ,1161 ,1113 ,1068 ,1173 ,1026 ,1065 ,1185 ,1164 ,1152,
80          1074 , 999 , 912 ,1068 , 843 , 744 , 717 , 735 , 630 , 588,
81          525 , 531 , 504 , 441 , 366 , 324 , 348 , 384 , 258 , 159,
82          207 ,138 ,135 ,114 , 87 , 93 ,102 ,102 , 69 , 75,
83          90 , 54 , 51 , 54 , 45 , 33 , 33 , 27 , 15 , 21,
84          21 , 9 ,24 ,18 ,18 ,18 , 6 ,18 , 6 , 9,
85          6 ,15 ,15 , 6 , 6 , 3 , 0 , 6 , 9 , 0,
86          0 , 0 , 6 ,12 , 3 ,12 ,12 ,12 ,60 ,72,
87          66 ,144 ,192 ,237 ,231, 1377]
88      histp.append(np.array(hmp3))
89
90      hmp4=[10371,  1404 ,996 ,648 ,627 ,390 ,414 ,330 ,294 ,237 ,357 ,264,
91          297 ,204 ,228 ,240 ,393 ,405 ,564,  1737,  1251 ,789 ,672 ,630,
92          453 ,495 ,432 ,552 ,462 ,414 ,351 ,303 ,333 ,342 ,336 ,390,
93          630 ,1248 ,1380 ,1488 ,1563 ,1686 , 942 , 948 , 840 , 678 , 600 ,
    513,
94          501 ,495 ,513 ,468 ,483 ,477 ,405 ,477 ,540 ,483 ,468 ,504,
95          561 ,558 ,570 ,555 ,594 ,648 ,762 ,822 , 981 ,1035 ,1083 ,1551,
96          1581 ,2097 ,2385 ,2472 ,3318 ,3330 ,3597 ,3747 ,3459 ,3459 ,3600
    ,3444,
97          3294 ,3621 ,3783 ,2961 ,3024 ,2853 ,2586 ,2625 ,2202 ,2109 ,1911
    ,1809,
98          1200 ,1032 , 714 , 624 , 489 , 396 , 480 , 351 , 306 , 360 , 315 ,
    324,
99          306 , 285 , 324 , 273 , 318 , 273 , 285 , 252 , 204 , 249 , 291 ,
    291,
100         231 ,288 ,258 ,240 ,171 ,225 ,213 ,264 ,228 ,228 ,207 ,240,
101         213 ,189 ,159 ,171 ,192 ,219 ,195 ,219 ,186 ,174 ,198 ,174,
102         147 ,159 ,156 ,186 ,174 ,156 ,180 ,174 ,120 ,153 ,183 ,192,
103         180 ,147 ,153 ,156 ,144 ,153 ,201 ,141 ,198 ,147 ,126 ,144,
104         132 ,105 ,108 ,156 ,126 ,177 ,147 ,135 ,105 , 87 ,105 ,117,
105         99 , 87 ,111 ,138 ,105 , 90 ,105 ,111 ,141 , 81 ,102 ,123,
106         117 ,108 ,132 ,123 ,141 , 96 , 87 , 60 ,114 ,120 , 90 ,129,
107         168 ,168 ,156 ,135 ,141 ,192 ,237 ,252 ,183 ,165 ,288 , 96,
108         84 , 39 , 33 , 21 , 15 , 6 , 9 , 24 , 3 , 3 , 0 , 0,
109         3 , 3 , 3 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0,
```

```
110         0 ,  0 ,  0 ,  0 ,  0 ,  0 ,  0 ,  0 ,  0 ,  0 ,  0,
111         0 ,  0 ,  0 ,  0]
112     histp.append(np.array(hmp4))
113
114
115     ##########  MODELOS NEGATIVOS
116
117     hmn1=[65442, 24537 ,2496 ,1203 ,1017 , 930 , 801 , 639 , 567 , 570 ,
    522 , 417,
118         438 ,321 ,339 ,327 ,273 ,285 ,279 ,312 ,240 ,240 ,243 ,261,
119         225 ,228 ,204 ,189 ,195 ,201 ,219 ,207 ,219 ,207 ,168 ,144,
120         144 ,162 ,189 ,153 ,213 ,165 ,156 ,210 ,192 ,177 ,165 ,168,
121         147 ,195 ,144 ,150 ,207 ,183 ,120 ,219 ,171 ,147 ,192 ,177,
122         201 ,198 ,237 ,249 ,246 ,258 ,324 ,312 ,372 ,459 ,486 ,546,
123         663 , 663 , 771 , 783 , 879 , 900 , 987 ,1119 ,1071 ,1086 ,1170
    ,1239,
124         1224 ,1194 ,1080 ,1026 , 981 ,1122 ,1095 , 930 , 993 , 900 , 750 ,
    747,
125         612 ,621 ,522 ,612 ,561 ,432 ,480 ,402 ,372 ,354 ,321 ,261,
126         300 ,234 ,273 ,171 ,201 ,168 ,141 ,174 ,153 ,129 ,135 ,129,
127         141 ,135 , 84 , 96 , 75 , 84 , 99 , 78 , 87 , 96 , 84 , 93,
128         57 , 81 , 90 , 72 , 78 , 63 , 78 , 84 , 84 , 69 , 87 , 54,
129         75 , 75 , 75 , 45 , 45 , 63 , 63 , 39 , 54 , 39 , 48 , 45,
130         57 , 63 , 18 , 51 , 39 , 24 , 33 , 21 , 36 , 42 , 24 , 39,
131         27 , 57 , 27 , 42 , 24 , 45 , 51 , 36 , 45 , 33 , 42 , 51,
132         48 , 36 , 42 , 48 , 54 , 60 , 66 , 33 , 66 , 48 , 45 , 78,
133         60 , 54 , 36 , 54 , 63 , 87 ,108 , 48 , 90 , 81 , 57 , 69,
134         72 ,111 , 63 , 93 , 81 , 69 , 60 , 54 , 69 , 60 , 75 , 45,
135         60 , 69 , 57 , 84 , 54 , 51 , 42 , 42 , 54 , 48 , 30 , 45,
136         48 , 42 , 27 , 48 , 21 , 42 , 27 , 24 ,  9 ,  9 , 15 ,  9,
137         15 ,  0 ,  3 ,  6 ,  0 ,  3 ,  9 ,  0 ,  0 ,  3 ,  3 ,  0,
138         3 ,  0 ,  0 ,  3]
139     histn.append(np.array(hmn1))
140
141     hmn2=[27180 ,28212 ,43548 ,46491 ,43029 ,35361 ,25764 ,18495 ,12456 ,
    9090 ,6516 ,4815,
142         3627 ,3129 ,2733 ,2358 ,2106 ,2004 ,1875 ,1722 ,1788 ,1668 ,1707
    ,1596,
143         1548 ,1545 ,1578 ,1482 ,1590 ,1611 ,1527 ,1509 ,1632 ,1491 ,1635
    ,1539,
144         1581 ,1557 ,1545 ,1497 ,1674 ,1905 ,1971 ,2310 ,2376 ,2778 ,3306
    ,3555,
145         4233 ,4761 ,4989 ,5925 ,6162 ,6543 ,7596 ,7767 ,8082 ,8625 ,8682
    ,8508,
146         9000 ,9162 ,9021 ,9081 ,8892 ,8646 ,7860 ,8013 ,7365 ,6708 ,6093
    ,5619,
147         5088 ,4503 ,3981 ,3639 ,3204 ,2547 ,2331 ,1935 ,1764 ,1545 ,1380
```

```
    ,1221,
148         1050 ,1011 , 789 , 744 , 693 , 489 , 465 , 456 , 384 , 270 , 273 ,
    291,
149         291 ,270 ,198 ,198 ,249 ,216 ,261 ,210 ,207 ,243 ,234 ,192,
150         183 ,207 ,234 ,135 ,192 ,234 ,222 ,147 ,210 ,198 ,177 ,186,
151         174 ,180 ,228 ,132 ,219 ,207 ,180 ,228 ,207 ,168 ,192 ,192,
152         192 ,189 ,174 ,183 ,183 ,165 ,189 ,216 ,204 ,153 ,174 ,207,
153         174 ,192 ,186 ,213 ,219 ,228 ,240 ,234 ,279 ,237 ,249 ,213,
154         234 ,342 ,258 ,276 ,330 ,285 ,321 ,312 ,327 ,345 ,363 ,399,
155         417 ,432 ,390 ,447 ,429 ,447 ,468 ,471 ,414 ,450 ,483 ,453,
156         330 ,414 ,357 ,444 ,366 ,411 ,387 ,357 ,405 ,285 ,306 ,396,
157         300 ,336 ,312 ,249 ,303 ,324 ,303 ,258 ,264 ,267 ,267 ,255,
158         297 ,237 ,183 ,225 ,243 ,192 ,246 ,183 ,183 ,180 ,171 ,156,
159         138 ,129 ,150 ,144 ,135 , 93 ,102 ,132 ,123 , 90 ,120 , 90,
160         105 , 87 , 60 , 48 , 54 , 36 , 63 , 51 , 45 , 24 , 24 , 27,
161         33 ,15 ,15 , 9 ,18 , 6 , 9 , 3 , 0 , 3 ,15 , 3,
162         0 , 0 , 0 , 0]
163     histn.append(np.array(hmn2))
164
165     hmn3=[ 4167 ,1041 ,5412, 18384, 14175 ,3411 ,1845 ,1230 ,1080 , 933 ,
    849 , 762,
166         609 , 489 , 507 , 543 , 477 , 429 , 420 , 438 , 396 , 438 , 402 ,
    450,
167         393 , 402 , 438 , 324 , 330 , 453 , 477 , 459 , 387 , 453 , 501 ,
    480,
168         681 , 681 , 879 ,1056 ,1248 ,1473 ,1713 ,2208 ,2241 ,2658 ,2532
    ,2565,
169         2490 ,2487 ,2334 ,2106 ,2187 ,2172 ,2070 ,2061 ,1872 ,1974 ,1827
    ,1815,
170         1749 ,1632 ,1650 ,1518 ,1356 ,1407 ,1314 ,1230 ,1224 ,1050 , 993 ,
    966,
171         780 ,789 ,768 ,741 ,693 ,612 ,606 ,564 ,558 ,501 ,447 ,414,
172         384 ,492 ,441 ,378 ,369 ,423 ,282 ,327 ,345 ,324 ,342 ,309,
173         306 ,291 ,288 ,378 ,279 ,267 ,273 ,210 ,288 ,267 ,219 ,240,
174         195 ,201 ,234 ,210 ,156 ,234 ,198 ,159 ,168 ,183 ,186 ,141,
175         144 ,117 ,150 ,147 ,132 ,159 ,144 ,153 ,138 , 93 ,141 ,126,
176         120 ,129 ,126 ,147 ,126 ,135 ,114 , 99 ,132 , 84 ,111 ,111,
177         123 ,132 ,111 ,102 ,105 , 90 ,105 , 81 , 78 ,108 , 66 , 99,
178         87 , 72 ,105 , 81 ,72 ,87 ,51 ,90 ,63 ,84 ,57 ,51,
179         57 ,54 ,54 ,63 ,54 ,51 ,48 ,63 ,69 ,57 ,45 ,51,
180         51 ,51 ,54 ,57 ,48 ,69 ,51 ,39 ,45 ,33 ,63 ,54,
181         54 ,33 ,36 ,48 ,36 ,33 ,30 ,42 ,57 ,42 ,45 ,66,
182         42 ,45 ,54 ,33 ,42 ,78 ,24 ,33 ,39 ,51 ,39 ,48,
183         27 ,24 ,45 ,45 ,63 ,51 ,57 ,27 ,27 ,36 ,15 ,24,
184         27 ,30 ,36 ,24 ,27 ,39 ,24 ,30 ,27 ,36 ,24 ,36,
185         15 ,21 ,15 , 3 ,15 ,18 , 9 ,15 , 6 ,12 , 6 , 9,
186         0 , 6 , 6 ,24]
```

```
187     histn.append(np.array(hmn3))
188
189     hmn4=[27030 ,2637 ,1569 , 891 ,771 ,714 ,696 ,609 ,498 ,414 ,417 ,429,
190         372 ,255 ,327 ,342 ,243 ,267 ,243 ,234 ,258 ,243 ,246 ,240,
191         258 ,240 ,309 ,297 ,333 ,324 ,423 ,405 ,396 ,429 ,495 ,579,
192         582 ,636 ,684 ,618 ,780 ,738 ,690 ,693 ,720 ,681 ,663 ,669,
193         645 ,633 ,636 ,714 ,609 ,624 ,621 ,561 ,597 ,531 ,681 ,648,
194         687 ,627 ,648 ,720 ,723 ,696 ,729 ,735 ,699 ,768 ,816 ,729,
195         747 ,804 ,789 ,858 ,882 ,909 ,918 ,954 ,879 ,930 ,897 ,894,
196         813 ,969 ,945 ,909 ,867 ,741 ,759 ,738 ,801 ,651 ,759 ,702,
197         681 ,753 ,729 ,789 ,756 ,807 ,663 ,702 ,756 ,789 ,789 ,717,
198         651 ,528 ,552 ,585 ,573 ,525 ,447 ,384 ,375 ,249 ,273 ,210,
199         306 ,234 ,273 ,177 , 75 ,57 ,36 ,27 ,18 ,18 , 6 , 9,
200         3 , 6 , 3 , 6 , 0 , 0 , 0 , 3 , 0 , 0 , 0 , 0,
201         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
202         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
203         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
204         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
205         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
206         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
207         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
208         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
209         0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0,
210         0 ,0 ,0 ,0]
211     histn.append(np.array(hmn4))
212
213     teste = cv.imread(imageadd,0)
214     testehist = imhist(teste)
215     #print(testehist)
216
217     mediap=0
218     median=0
219
220     #print("Correlacoes com positivos")
221     for i in range(len(histp)):
222         correlacaop.append(np.corrcoef(testehist,histp[i])[0,1])
223         mediap=mediap + correlacaop[i]
224
225     correlacaop.clear()
226
227     #print("Correlacoes com negativos")
228     for i in range(len(histn)):
229         correlacaon.append(np.corrcoef(testehist,histn[i])[0,1])
230         median=median + correlacaon[i]
231
232     correlacaon.clear()
233     mediap=mediap/len(histp)
```

```
234     median=median/len(histn)
235     #print("Media da Correlacao com modelos positivos:",mediap)
236     #print("Media da Correlacao com modelos negativos:",median)
237
238     if median > mediap:
239         st = "Negativo"
240     else:
241         st = "Positivo"
242         arr = np.asarray(teste)
243
244         jj=0
245         while jj<20:
246             arr3=imadjust(arr,a*255,b*255,0,255)
247             arr2= np.array(arr3)
248             soma=0
249             shape=np.array(arr.shape)
250             for j in range(shape[1]):
251                 for i in range(shape[0]):
252                     soma=soma + (arr2.item(i,j)/255)
253
254             razao = soma/arr2.size
255             print("Soma:",soma)
256             print("Razao:",razao)
257             if razao<0.015:
258                 a=a-0.025
259                 b=b-0.025
260                 if a>1:
261                     a=1
262                 elif a<0:
263                     a=0
264
265                 if b>1:
266                     b=1
267                 elif b<0:
268                     b=0
269
270                 #print('a',a)
271                 #print('b',b)
272             elif razao>0.02:
273                 a=a+0.025
274                 b=b+0.025
275                 if a>1:
276                     a=1
277                 elif a<0:
278                     a=0
279
280                 if b>1:
```

```
                    b=1
                elif b<0:
                    b=0



                #print('a',a)
                #print('b',b)
            else:
                break

            jj=jj+1


        nome = "imagem" + str(cont) + ".jpg"
        cv.imwrite(nome, arr3)

    return st
```

Apêndice D - Programa de Criação da Interface Visual

```
working_directory = os.getcwd()

layout = [
    [sg.Text("Escolha uma imagem:")],
    [sg.InputText(key="-FILE_PATH-"),
     sg.FileBrowse(initial_folder=working_directory, file_types=[("JPG
    Files","*.jpg")])],
    [sg.Button("Submit"),sg.Exit()],
    [sg.Text("",key = "resultado")]
]

window = sg.Window("File Loader",layout)

while True:
    event, values = window.read()
    if event in (sg.WIN_CLOSED, 'Exit'):
        break
    elif event == "Submit":
        imagem=values["-FILE_PATH-"]
        contador = contador + 1
        result = leImagem(imagem,contador)
        window["resultado"].update(f"O resultado e {result}")

window.close()
```