UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MESTRADO ACADÊMICO EM ENGENHARIA ELÉTRICA

PEDRO LINO AZEVEDO LANDIM

PLATAFORMA WEB DE ANOTAÇÕES INTERATIVAS PARA SEGMENTAÇÃO DE NEUROIMAGENS POR RESSONÂNCIA MAGNÉTICA

FORTALEZA

2025

PEDRO LINO AZEVEDO LANDIM

PLATAFORMA WEB DE ANOTAÇÕES INTERATIVAS PARA SEGMENTAÇÃO DE NEUROIMAGENS POR RESSONÂNCIA MAGNÉTICA

Proposta de qualificacao apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica da Universidade Federal do Ceará (UFC) como requisito parcial para a defesa de dissertação.
Orientador: Prof. Dr. Victor Hugo Costa de Albuquerque

Coorientador: Prof. Dr. Bruno Riccelli Dos Santos Silva

FORTALEZA

2025

PEDRO LINO AZEVEDO LANDIM

PLATAFORMA WEB DE ANOTAÇÕES INTERATIVAS PARA SEGMENTAÇÃO DE NEUROIMAGENS POR RESSONÂNCIA MAGNÉTICA

Proposta de qualificacao apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica da Universidade Federal do Ceará (UFC) como requisito parcial para a defesa de dissertação.

Aprovada em: 08 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. Victor Hugo Costa de Albuquerque (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bruno Riccelli Dos Santos Silva (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Débora Ferreira De Assis
Instituto Atlântico

Prof. Dr. Rafael Simões Do Carmo
Universidade Estadual Paulista (UNESP)

RESUMO

O aumento no volume e na complexidade das imagens médicas, especialmente as de ressonância magnética, tem sobrecarregado os profissionais de saúde e evidenciado a necessidade de ferramentas computacionais de apoio ao diagnóstico. Nesse contexto, este trabalho desenvolve uma plataforma interativa destinada a segmentação de neuroimagens por ressonância magnética, integrando técnicas de inteligência artificial. A ferramenta foi projetada com arquitetura modular composta por uma interface em Flutter Web, uma API em Flask e modelos de aprendizado profundo implementados em PyTorch. O sistema gera automaticamente máscaras utilizando o algoritmo Flood Fill, que são posteriormente validadas por uma CNN e utilizadas para o treinamento supervisionado da arquitetura U-Net, responsável pela segmentação automática das imagens. A interface oferece recursos para upload de imagens, marcação de pontos de interesse, visualização de resultados e re-treinamento do modelo com dados personalizados. Foram realizados testes de carga e estresse para avaliação de desempenho, bem como análises quantitativas e qualitativas. O modelo de segmentação U-Net alcançou um Dice Score médio de 90.22%, enquanto a CNN responsável pela validação das máscaras obteve 97.49% de acurácia. Tais resultados, somados à alta taxa de sucesso da API sob estresse, demonstram a viabilidade da aplicação em ambientes clínicos e de pesquisa, destacando-se pela flexibilidade e integração com o fluxo de trabalho médico.

Palavras-chave: segmentação de imagens médicas; UNet; Flood Fill; Flutter Web; Flask; diagnóstico assistido por computador (CAD).

ABSTRACT

The growing volume and complexity of medical imaging, especially magnetic resonance imaging, have increasingly overwhelmed healthcare professionals and highlighted the need for computer-aided diagnostic tools. In this context, this work presents the development of an interactive web platform for the segmentation of magnetic resonance neuroimages, integrating artificial intelligence techniques and accessible visualization. The tool was designed with a modular architecture composed of a Flutter Web interface, a Flask-based API, and deep learning models implemented in PyTorch. The system performs automatic segmentation using the U-Net architecture and enables dataset enhancement through mask generation using the Flood Fill algorithm, validated by a CNN. The interface provides features for image upload, region of interest marking, result visualization, and retraining of the model with customized data. Load and stress tests were conducted to assess system performance, along with both quantitative and qualitative analyses of the segmentations. The U-Net segmentation model achieved a mean Dice Score of 90.22% , while the CNN for mask validation obtained 97.49% accuracy. These results, combined with the API's high success rate under stress, demonstrate the feasibility of applying the platform in clinical and research environments, highlighting its flexibility, adaptability, and integration with medical workflows.

Keywords: medical image segmentation; UNet; Flood Fill; Flutter Web; Flask; computer-aided diagnosis (CAD).

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo introdutório, serão apresentadas as informações essenciais para compreender o contexto e a abordagem proposta nesta tese. Exploraremos a crescente demanda por análise eficiente de exames de imagem, os desafios enfrentados pelos profissionais de saúde e o papel das ferramentas de diagnóstico assistido por computador (CAD) na otimização desse processo. Além disso, discutiremos a relevância da solução proposta, seus impactos na prática clínica e na qualidade dos diagnósticos, bem como os objetivos e contribuições esperadas deste estudo, delimitando claramente seu escopo.

1.0.1 A problemática

A evolução das técnicas de imagem tem transformado profundamente a prática da radiologia, trazendo avanços importantes, mas também impondo desafios significativos aos profissionais de saúde. Com o aumento exponencial do número de exames, como tomografias computadorizadas e ressonâncias magnéticas, os médicos se veem diante da necessidade de analisar centenas de imagens por paciente, frequentemente com elevado grau de complexidade. Esse cenário, somado à crescente subespecialização das áreas médicas, torna inviável que um único profissional domine todas as regiões anatômicas e padrões diagnósticos, o que pode comprometer a precisão e a consistência das interpretações. Essa abundância de dados exige um novo perfil de análise, mais integrado e apoiado por ferramentas computacionais inteligentes (Santos *et al.*, 2019).

Complementando esse panorama, McDonald *et al.* (2015) analisaram os impactos provocados pela evolução tecnológica e pelo aumento na utilização de exames de imagem seccionais, como tomografias computadorizadas (CT) e ressonâncias magnéticas (MRI), sobre a carga de trabalho dos radiologistas. Segundo os autores, entre 1999 e 2010, enquanto o número total de exames cresceu 75%, o volume de imagens geradas aumentou mais de 1000%, impulsionado, principalmente, por melhorias na resolução e pela complexificação dos protocolos de aquisição. Com isso, o número médio de imagens por exame aumentou drasticamente, exigindo que os radiologistas interpretassem, em média, uma imagem a cada 3 a 4 segundos ao longo de um turno de 8 horas. Os próprios autores alertam que esse crescimento no volume de informação está associado a maior fadiga, risco de erros e possíveis prejuízos à acurácia diagnóstica.

O crescimento no volume e na complexidade das imagens médicas também expõe os limites da capacidade humana na interpretação diagnóstica. Gao *et al.* (2019) ressaltam que fatores como fadiga, inexperiência e sobrecarga cognitiva podem comprometer o desempenho dos radiologistas, resultando em taxas de erro que variam entre 10% e 30%. Esses índices, segundo os autores, são preocupantes em contextos de alta demanda, onde o tempo disponível para analisar cada exame é restrito e a atenção do profissional é continuamente exigida. A subjetividade da interpretação e a dificuldade em manter precisão diante de centenas de imagens por exame tornam evidente a vulnerabilidade do diagnóstico exclusivamente humano em ambientes cada vez mais digitalizados.

Redfern *et al.* (2000), por sua vez, investigaram os efeitos da transição do sistema convencional de gerenciamento de imagens para o modelo digital PACS (Picture Archiving and Communication System) sobre o fluxo de trabalho em departamentos de radiologia. A adoção do PACS permitiu maior integração e distribuição das imagens, resultando em um aumento expressivo no volume de exames processados por radiologista, um acréscimo de até 258% durante os turnos diurnos. Embora a digitalização tenha inicialmente reduzido o tempo entre o recebimento da imagem e o ditado do laudo, os autores identificaram uma relação curvilinear entre volume de pacientes e tempo de resposta: até certo ponto, o aumento de volume gera maior eficiência, mas, ao ultrapassar um limite, provoca sobrecarga e retardo na produção dos laudos. Esse achado evidencia que, apesar dos benefícios operacionais trazidos pela evolução das técnicas de imagem e digitalização, o crescimento descontrolado do volume de informações pode comprometer o desempenho clínico e exigir estratégias de balanceamento de carga e apoio tecnológico à interpretação médica.

Essa sobrecarga também é observada em contextos nacionais, como exemplificado por Oliveira (2023), que ressalta que, com o aumento do número de exames e a semelhança visual entre imagens normais e patológicas da região pulmonar em radiografias de tórax, os radiologistas enfrentam um desafio cada vez maior de tempo e atenção na análise de grandes volumes de dados. Esse cenário acentua a necessidade de recursos que auxiliem os profissionais na triagem e interpretação dos exames, diante de uma prática cada vez mais exigente em termos de velocidade, precisão e consistência diagnóstica. A crescente demanda, especialmente em contextos com escassez de especialistas, evidencia como a evolução das técnicas de imagem, embora benéfica, também impõe uma nova sobrecarga aos sistemas de saúde.

Por fim, Cristofalo (2024) destaca que o grande volume de dados disponíveis na área

da saúde tem possibilitado a aplicação de técnicas avançadas de *machine learning* na análise de exames médicos digitalizados. Esses exames incluem registros médicos eletrônicos, sinais vitais e exames laboratoriais rotineiros coletados em unidades de internação neonatal. O autor enfatiza a importância da padronização e segmentação desses dados, propondo a criação de grupos específicos, como via de parto, idade materna e idade gestacional, o que contribui para a especialização dos modelos preditivos. A própria necessidade de segmentar os dados para potencializar o uso de ferramentas automáticas evidencia a complexidade enfrentada pelos profissionais da saúde diante de um volume crescente de exames e da diversidade de fatores clínicos a serem considerados. Essa realidade reforça a importância do desenvolvimento de soluções que auxiliem na triagem e otimizem o atendimento médico de forma eficiente.

Assim, embora o avanço das técnicas de imagem tenha ampliado significativamente as possibilidades diagnósticas, ele também impôs uma nova realidade: o desafio de gerenciar volumes crescentes de dados que extrapolam os limites da cognição humana. Nesse contexto, soluções baseadas em automação e inteligência artificial têm se destacado como alternativas promissoras.

Diversos estudos têm demonstrado o potencial da inteligência artificial para lidar com o volume crescente de exames de imagem na medicina, por meio de técnicas como redes neurais convolucionais (CNNs), segmentação automatizada e geração de laudos assistidos por algoritmos. No entanto, apesar dos avanços técnicos, muitas dessas soluções permanecem restritas ao ambiente de pesquisa, carecendo de interfaces acessíveis e integração com o fluxo clínico real. Essa lacuna entre o desempenho algorítmico e a aplicabilidade prática reforça a necessidade de estudos voltados à criação de ferramentas que unam eficiência computacional à usabilidade clínica, tornando a inteligência artificial de fato uma aliada dos profissionais de saúde no enfrentamento da sobrecarga diagnóstica.

1.1 Objetivos

Diante desse cenário, o objetivo principal deste trabalho é desenvolver uma ferramenta de diagnóstico assistido por computador (CAD) baseada em inteligência artificial, focada na segmentação de neuroimagens de ressonância magnética (RMI) de forma automatizada e adaptável. Embora o escopo deste estudo se concentre na análise cerebral, o sistema foi projetado com uma arquitetura flexível, o que abre perspectivas para sua aplicação futura em outras áreas médicas, como o diagnóstico de lesões em articulações e tecidos musculoesqueléticos.

Para atingir esse objetivo geral, foram definidos os seguintes objetivos específicos, que abrangem desde o desenvolvimento de um pipeline eficiente para segmentação de imagens médicas até a validação e aprimoramento do modelo de IA. Esses objetivos visam garantir a adaptabilidade da ferramenta a diferentes estruturas anatômicas, promovendo maior flexibilidade e precisão na análise de exames de ressonância magnética:

- Desenvolver um sistema de segmentação focado em neuroimagens, com uma arquitetura projetada para ser adaptável.
- Utilizar o algoritmo Flood Fill para a geração automática de máscaras com base em pontos indicados pelo usuário.
- Validar as máscaras geradas por meio de uma CNN antes do treinamento da arquitetura U-Net.
- Integrar a ferramenta a uma interface interativa, com foco em usabilidade e acessibilidade para profissionais da saúde.

2 TRABALHOS RELACIONADOS

A fim de compreender como a comunidade científica tem enfrentado os desafios relacionados ao aumento do volume e da complexidade dos exames de imagem, esta seção apresenta uma revisão da literatura recente, cuja seleção de trabalhos seguiu uma abordagem metodológica clara.

A busca foi realizada prioritariamente na plataforma Google Acadêmico, por sua vasta indexação que abrange os principais portais científicos como IEEE Xplore, Elsevier (ScienceDirect), Springer e MDPI. O levantamento concentrou-se em artigos publicados nos últimos cinco anos (2020-2025) para capturar as abordagens mais recentes, com exceção de trabalhos seminais de alto impacto, incluídos por sua relevância fundamental. A seleção final foi guiada pela relevância com os pilares desta dissertação, utilizando-se palavras-chave como *"medical image segmentation"*, *"U-Net neuroimaging"* e *"computer-aided diagnosis usability"*. Foram incluídos artigos com foco em: (i) performance de algoritmos de segmentação; (ii) aplicabilidade clínica e personalização de modelos; e (iii) desenvolvimento de interfaces de usuário para sistemas CAD.

Essa estratégia metodológica permitiu mapear as soluções existentes, identificar lacunas, e posicionar a contribuição deste trabalho frente ao cenário científico atual.

2.1 Foco em desempenho algorítmico

Diversos estudos têm demonstrado o potencial das redes neurais convolucionais (CNNs) e arquiteturas derivadas para a análise automatizada de imagens médicas. Oliveira (2023) investigou o uso de CNNs para a detecção de pneumonia em radiografias torácicas, utilizando modelos pré-treinados e técnicas de transfer learning. Os resultados mostraram alta acurácia na classificação automatizada, destacando o potencial da inteligência artificial no suporte diagnóstico em larga escala. No entanto, o foco exclusivo na performance técnica da rede neural, sem considerar a criação de uma interface voltada ao uso clínico, limita a aplicabilidade prática da solução.

Seguindo essa linha, Zhu *et al.* (2022) propõem a arquitetura Attention-UNet para segmentação de estruturas brônquicas em tomografias de tórax. O modelo incorpora mecanismos de atenção à arquitetura clássica do U-Net, permitindo foco seletivo em regiões relevantes e minimizando erros comuns, como vazamentos de segmentação. Avaliações experimentais

demonstraram ganhos significativos em métricas como Dice e IoU, evidenciando sua eficácia. Ainda assim, o estudo se restringe à avaliação técnica, sem contemplar formas de interação com profissionais da saúde.

Vasa *et al.* (2024) avançam nesse campo com o STA-UNet, arquitetura baseada em U-Net que utiliza o módulo Super Token Attention (STA), inspirado em superpixels. Avaliado em múltiplas bases públicas, o modelo superou abordagens consolidadas como TransUNet e SwinUNet na segmentação de estruturas pequenas, como o pâncreas. Apesar dos avanços, o estudo também se limita ao aspecto técnico, sem propor estratégias de integração clínica.

2.2 Personalização e aplicabilidade clínica

Cristofalo (2024) propõe uma abordagem voltada à especialização de modelos de machine learning na predição de mortalidade neonatal, a partir da segmentação dos dados clínicos por variáveis como via de parto, idade materna e idade gestacional. A proposta reconhece a heterogeneidade dos dados em saúde e busca aumentar a acurácia dos algoritmos adaptando-os a perfis populacionais específicos. No entanto, assim como outras abordagens técnicas, carece de uma interface funcional que conecte os modelos preditivos aos profissionais de saúde, o que pode comprometer sua adoção prática.

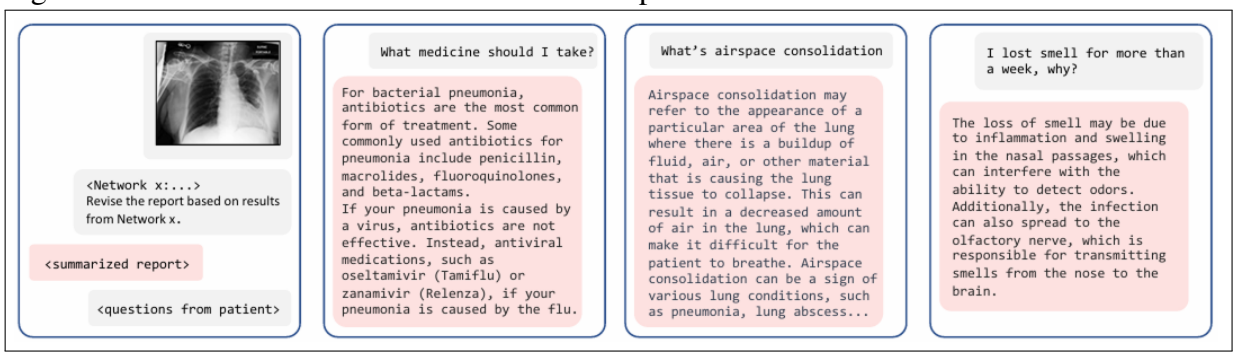
Gao *et al.* (2019) reforçam esse desafio ao propor o uso de CNNs como base para a nova geração de sistemas de diagnóstico assistido por computador (CAD). As redes são capazes de realizar tarefas como segmentação, extração de características e classificação diretamente dos dados brutos, reduzindo vieses e aumentando a acurácia. Os autores reconhecem, contudo, que a integração dessas soluções a sistemas hospitalares existentes e a necessidade de bases rotuladas de qualidade ainda representam barreiras à aplicação em larga escala.

2.3 Interfaces conversacionais

Superando parte dessas limitações, Wang *et al.* (2023) apresentam a arquitetura ChatCAD, que integra redes especializadas de CAD com modelos de linguagem de grande escala (LLMs), como o ChatGPT. O sistema transforma os resultados de análises automatizadas em linguagem natural, organizando-os por meio de um LLM para gerar laudos médicos claros e interpretáveis. Além disso, oferece uma interface conversacional acessível, ilustrada na figura 1, que permite a médicos e pacientes interagir com os dados, tirar dúvidas e compreender

o diagnóstico sem conhecimento técnico prévio. Essa proposta destaca-se não apenas pelo desempenho computacional, mas também pela ênfase na usabilidade clínica, aspecto essencial para a adoção em ambientes reais de saúde.

Figura 1 – CAD interativo com LLMs. Este exemplo usa o ChatGPT como LLM.



Fonte: (Wang *et al.*, 2023).

Essa tendência de integração prática já se manifesta em soluções aplicadas com sucesso. Yang e Yu (2021), em revisão abrangente, destacam sistemas como o ENDOANGEL, baseado em SSD, capaz de detectar lesões em tempo real durante exames endoscópicos, e o GRAIDS, baseado em DeepLab v3+, que alcançou acurácia superior a 90% na detecção de câncer gastrointestinal. Ambos os sistemas demonstram não apenas performance técnica de alto nível, mas também efetiva inserção no fluxo de trabalho clínico, representando um avanço significativo na superação da sobrecarga gerada pelo aumento de exames.

A partir dessa revisão, observa-se que, embora os avanços técnicos em redes neurais e modelos de segmentação tenham promovido melhorias substanciais na acurácia de sistemas de apoio diagnóstico, a efetiva aplicação clínica dessas soluções ainda depende da criação de interfaces acessíveis, interativas e integradas aos fluxos hospitalares. Modelos como o ChatCAD e sistemas já implantados, como o ENDOANGEL, mostram que a aliança entre desempenho computacional e usabilidade é o caminho mais promissor para tornar a inteligência artificial uma aliada real na prática médica.

2.4 Comparativo de abordagens

A partir dessa revisão, observa-se que, embora os avanços técnicos em redes neurais e modelos de segmentação tenham promovido melhorias substanciais na acurácia de sistemas de apoio diagnóstico, a efetiva aplicação clínica dessas soluções ainda depende da criação de interfaces acessíveis, interativas e integradas aos fluxos hospitalares. Modelos como o ChatCAD

e sistemas já implantados, como o ENDOANGEL, mostram que a aliança entre desempenho computacional e usabilidade é o caminho mais promissor para tornar a inteligência artificial uma aliada real na prática médica. Para facilitar a comparação entre as soluções discutidas, o Quadro 1 resume as principais abordagens encontradas na literatura, destacando o foco técnico, a existência de interface com o usuário e o grau de aplicação clínica observado.

Quadro 1 – Comparativo Técnico de Soluções Computacionais em Imagens Médicas

Autor(es)	**Solução Proposta**	**Tarefa Principal do Modelo**	**Métricas de Segmentação (Resultados)**	**Criação das Máscaras**	**Base de Dados**
Oliveira (2023)	UNet e MultiResUNet para segmentação pulmonar em radiografias de tórax.	Segmentação de Imagem	**Sim.** Dice: 97,00% IoU: 94,29%	Utiliza máscaras pré-existentes da base de dados, geradas por um método cooperativo homem-máquina.	COVID-QU-Ex
Cristofalo (2024)	Modelos especializados por grupos para predição de mortalidade neonatal.	Predição de Mortalidade. Divisão dos dados em subgrupos.	**Não aplicável.** Usa métricas de classificação (ROC AUC, etc.).	Não aplicável (utiliza dados tabulares, não imagens).	Maternal Newborn Health Registry (MNHR)
Wang *et al.* (2023)	ChatCAD: integração CAD + LLM com interface conversacional.	Classificação e Segmentação de Imagem (integradas por um LLM).	**Não aplicável.** Avalia o relatório final com F1-score.	Utiliza modelos CAD pré-treinados.	MIMIC-CXR, CheXpert
Gao *et al.* (2019)	Revisão sobre o uso de CNNs como base para sistemas CAD modernos.	Artigo de Revisão (cobre Segmentação e Classificação).	Não aplicável (artigo de revisão).	Não aplicável (artigo de revisão).	LIDC-IDRI, DDSM, MIAS, PROMISE1.
Zhu *et al.* (2022)	Attention-UNet para segmentação brônquica.	Segmentação de Imagem	**Sim.** Dice: 0.721 IoU: 0.653	Utiliza máscaras da base de dados da competição.	EXACT09
Vasa *et al.* (2024)	STA-UNet com Super Token Attention para segmentação eficiente.	Segmentação de Imagem	**Sim.** Dice: 91,03% (no dataset GlaS).	Utiliza máscaras pré-existentes das bases públicas.	Synapse, ACDC, MoNuSeg, GlaS
Yang e Yu (2021)	Revisão de sistemas como ENDOANGEL e GRAIDS.	Artigo de Revisão (cobre Detecção e Segmentação de Imagem).	Não aplicável (artigo de revisão).	Não aplicável (artigo de revisão).	Cita o uso de imagens de endoscopia, CT, MRI, etc.

Fonte: Elaborado pelo autor com base nos trabalhos analisados.

3 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais conceitos, técnicas e tecnologias que fundamentam o desenvolvimento da ferramenta proposta nesta pesquisa. Diferente do capítulo anterior, que analisou trabalhos correlatos, o foco aqui é descrever as bases teóricas que sustentam a implementação detalhada na Metodologia. O capítulo está organizado da seguinte forma: inicialmente, aborda-se o conceito de Diagnóstico Assistido por Computador (CAD). Em seguida, são detalhadas as técnicas de processamento de imagens empregadas, como o algoritmo Flood Fill e o aumento de dados. Posteriormente, são apresentadas as arquiteturas de deep learning utilizadas, com ênfase na U-Net para segmentação. Por fim, descreve-se a arquitetura tecnológica da plataforma, baseada em Flutter e Flask, que viabiliza a integração desses componentes em uma solução web interativa.

3.1 Diagnóstico Assistido por Computador

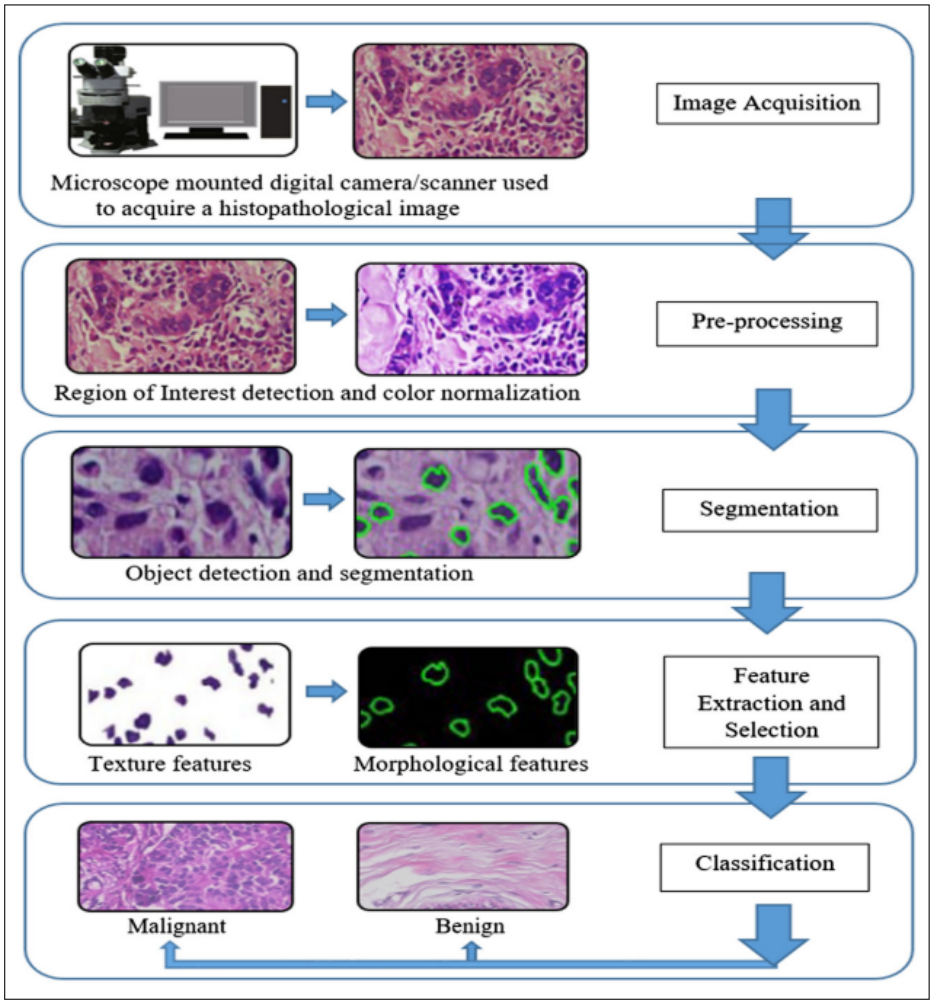
A história das ferramentas de Diagnóstico Assistido por Computador (CAD) remonta ao final da década de 1970 e início dos anos 1980, quando começaram os primeiros estudos com o objetivo de utilizar computadores como apoio à decisão clínica, especialmente em radiologia. Segundo Doi (2007), considerado um dos pioneiros no campo do processamento de imagens médicas, as primeiras aplicações de CAD foram voltadas à detecção de nódulos em radiografias de tórax e à identificação de microcalcificações em mamografias, atuando como uma ferramenta de auxílio ao diagnóstico para os radiologistas. Esses sistemas iniciais não utilizavam aprendizado de máquina, mas sim técnicas determinísticas de processamento de imagem digital, como limiarização para segmentação de regiões suspeitas, realce de contraste, filtros de borda (Sobel, Laplaciano) para detectar contornos, e extração manual de características morfológicas (como área e circularidade). A classificação era feita com base em regras fixas programadas, formuladas em conjunto com especialistas médicos. Apesar de rudimentares, essas abordagens lançaram as bases dos sistemas CAD modernos. Ao longo das décadas seguintes, com o avanço da capacidade computacional e o surgimento de algoritmos de aprendizado de máquina, essas ferramentas passaram a incorporar modelos estatísticos, redes neurais e, mais recentemente, técnicas de deep learning, expandindo seu uso para diversas áreas como oncologia, neurologia, cardiologia e oftalmologia. O trabalho de Doi (2007) foi essencial para consolidar o conceito de CAD como um campo fundamental dentro da medicina computacional, especialmente na área

de processamento digital de imagens médicas, definindo seus objetivos, limitações e potencial de impacto na prática clínica.

O artigo de Kaushal *et al.* (2019) apresenta uma revisão abrangente e sistemática das tecnologias aplicadas em sistemas CAD voltados à detecção do câncer de mama em imagens histopatológicas. O CAD é definido como um sistema de suporte à decisão clínica, que visa reduzir a variabilidade diagnóstica entre observadores humanos e agilizar a detecção de tumores, oferecendo uma segunda opinião objetiva e quantitativa ao patologista. Conforme ilustrado na Figura 2 do artigo, o processo CAD segue um pipeline bem estruturado, composto por quatro etapas fundamentais: (i) pré-processamento, para correção de variações de cor e ruído; (ii) segmentação, que delimita as regiões de interesse (como núcleos celulares), etapa essa que muitas vezes define o sucesso das etapas seguintes; (iii) extração de características, onde se coletam atributos texturais, morfológicos e de intensidade; e (iv) classificação, etapa final em que os tecidos são categorizados como benignos ou malignos com auxílio de algoritmos como SVMs (Support Vector Machines) e redes neurais convolucionais (CNNs). O artigo destaca que essas tecnologias têm alcançado métricas de desempenho elevadas (acurácia > 90% em muitos casos), o que demonstra o potencial clínico dessas ferramentas. No entanto, ainda persistem desafios técnicos e práticos, como a falta de padronização dos dados histológicos, a dependência da magnificação da imagem, além de questões éticas e regulatórias ainda pouco exploradas. Por fim, os autores apontam tendências futuras que envolvem o desenvolvimento de sistemas mais autônomos, independentes de magnificação, com menor tempo de processamento e maior integração aos fluxos clínicos reais.

O trabalho de Pereira *et al.* (2019) amplia substancialmente o escopo tradicional dos sistemas de diagnóstico assistido por computador ao abordar sua aplicação no contexto da Doença de Parkinson (DP). Diferentemente de abordagens focadas exclusivamente em imagens médicas, esse estudo revela que os sistemas CAD voltados para DP fazem uso intensivo de sinais de voz, caligrafia, marcha e eletrofisiologia, muitas vezes adquiridos por meio de sensores vestíveis, smartphones, canetas biométricas e dispositivos de realidade aumentada, o que demanda uma estrutura de entrada de dados bastante heterogênea. Os autores organizaram a revisão com base em uma taxonomia prática, segmentando os estudos por domínio de aplicação (sinais, imagens, sensores, dispositivos móveis, realidade virtual e web). Um dos aspectos técnicos mais relevantes é que quase todos os sistemas CAD descritos incorporam algoritmos de machine learning supervisionado, como SVMs, Random Forests, Redes Neurais Artificiais e modelos

Figura 2 – Visão abrangente de um sistema CAD para câncer de mama utilizando histopatologia.



Fonte: (Kaushal *et al.*, 2019).

evolutivos, adaptados ao tipo de dado de entrada. Além disso, destaca-se o uso crescente de técnicas avançadas de pré-processamento e seleção de atributos, como o mRMR (minimum Redundancy Maximum Relevance) e componentes independentes (ICA), visando extrair as características mais discriminativas. O artigo também aponta a evolução da infraestrutura dessas ferramentas com a integração de plataformas baseadas em IoT, realidade aumentada e telemedicina, permitindo o monitoramento remoto e contínuo de pacientes. Essa transição do CAD tradicional — estático e focado em imagens médicas — para um CAD dinâmico, multimodal e centrado no paciente representa um avanço significativo tanto no diagnóstico precoce quanto no acompanhamento da progressão da doença. Os autores reforçam ainda que, embora promissoras, essas tecnologias precisam ser avaliadas em ambientes clínicos reais, dado que muitos estudos ainda se restringem a contextos experimentais.

O artigo de Eadie *et al.* (2012) realiza uma distinção fundamental entre dois tipos de ferramentas de diagnóstico assistido por computador: o CADe (Computer-Aided Detection) e o

CADx (Computer-Aided Diagnosis). Enquanto o CADe tem como principal objetivo detectar regiões suspeitas nas imagens, funcionando como um "marcador visual" para auxiliar o radiologista a não deixar passar potenciais lesões, o CADx atua de forma mais sofisticada, analisando e classificando as lesões detectadas, oferecendo um diagnóstico auxiliar (por exemplo, benigno vs. maligno). A comparação direta entre os dois sistemas, baseada em 48 estudos clínicos revisados, revelou que o CADx apresenta ganhos significativos em termos de sensibilidade, especificidade e acurácia diagnóstica, especialmente em aplicações de mamografia e ultrassonografia mamária. Já o CADe, apesar de inicialmente promissor, mostrou-se limitado: embora tenha ligeiramente aumentado a sensibilidade, causou uma queda na especificidade, ou seja, aumentou a taxa de falsos positivos. Isso pode ser explicado pelo fato de que o processamento visual humano já é altamente eficaz para detecção, tornando o ganho com o CADe marginal, quando não contraproducente. Em contraste, a tarefa de análise e classificação diagnóstica, que envolve múltiplos fatores sutis e quantitativos, é mais desafiadora para o radiologista e se beneficia da capacidade computacional do CADx de integrar grandes volumes de dados. Os autores sugerem, inclusive, que o futuro das ferramentas CAD está mais alinhado com os sistemas CADx, que otimizam a expertise humana ao fornecer uma segunda opinião baseada em modelos de dados robustos. Portanto, a eficácia clínica dessas tecnologias está diretamente relacionada à sua função no processo diagnóstico: quanto mais analítica e interpretativa a tarefa, maior o potencial de impacto do CADx em comparação ao CADe.

Em síntese, os sistemas CAD representam uma importante evolução no suporte ao diagnóstico médico, especialmente na detecção e classificação de doenças como câncer e Parkinson. A distinção entre CADe e CADx é fundamental para compreender seus impactos clínicos e operacionais, sendo o CADx o modelo mais promissor para integração à prática médica. Apesar dos avanços em inteligência artificial e aprendizado de máquina, os desafios relacionados à padronização, validação clínica e integração nos fluxos hospitalares ainda precisam ser superados para que essas tecnologias alcancem seu pleno potencial.

3.2 Tratamento de Imagens

3.2.1 Pré-Processamento de Imagens

O pré-processamento de imagens é uma etapa essencial para melhorar a qualidade visual e viabilizar a análise automática ou humana de exames médicos. De acordo com Vasuki *et*

al. (2017), as técnicas de pré-processamento podem ser agrupadas em dois domínios principais: o domínio espacial, no qual os pixels da imagem são manipulados diretamente, e o domínio da frequência, que atua sobre os componentes espectrais por meio de transformações, como a de Fourier.

No domínio espacial, destacam-se operações como transformações de intensidade (linear, logarítmica, potência), equalização de histograma e aplicação de filtros de suavização ou nitidez. Além disso, são amplamente utilizados operadores de detecção de bordas, como Sobel, Prewitt e Canny, especialmente úteis na segmentação de estruturas anatômicas. O filtro de Sobel, em particular, é citado pelos autores como eficaz para realçar contornos e transições de intensidade, sendo um recurso comum em métodos baseados em gradiente. Já no domínio da frequência, utilizam-se filtros passa-baixa, que removem ruídos de alta frequência; passa-alta, que realçam detalhes finos; e homomórficos, que separam os componentes de iluminação e refletância da imagem (Vasuki *et al.*, 2017).

Dentre as técnicas de detecção de bordas, destaca-se o filtro de Sobel, um operador baseado em gradiente amplamente utilizado em processamento de imagens. Seu funcionamento consiste na aplicação de uma convolução 2D sobre a imagem utilizando dois kernels (matrizes) de tamanho 3x3. Um kernel é projetado para detectar as variações de intensidade na direção horizontal (gradiente G_x) e o outro, na direção vertical (gradiente G_y). A magnitude total do gradiente, representada por G, é então calculada conforme a Equação 3.1.

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

O mapa de bordas resultante, gerado a partir dos valores de G, realça as transições abruptas de intensidade, correspondentes aos contornos das estruturas na imagem. Esta técnica é particularmente eficaz para destacar os limites de regiões anatômicas em exames médicos (Vasuki *et al.*, 2017).

Além disso, Goel *et al.* (2016) ressaltam que a remoção de ruído é uma das etapas mais críticas do pré-processamento, especialmente em contextos clínicos, nos quais a preservação de detalhes estruturais é fundamental para um diagnóstico preciso. Ruídos como os gaussianos, Poisson, Rician e sal e pimenta comprometem a qualidade das imagens adquiridas por diferentes modalidades, como raios-X, ultrassom e ressonância magnética. O desafio, segundo os autores, está em remover tais ruídos sem comprometer as bordas e os detalhes anatômicos relevantes,

uma vez que a perda dessas informações pode impactar negativamente etapas posteriores, como segmentação, extração de características e classificação. Para isso, discutem-se diversas técnicas eficazes de denoising, como transformadas wavelet, filtros morfológicos, filtragem adaptativa e métodos baseados em redes neurais.

Complementarmente, Perumal e Velmurugan (2018) reforçam a importância do pré-processamento para a preparação de imagens médicas, especialmente de ressonância magnética (MRI), para etapas posteriores como segmentação e detecção de anomalias. Em estudos voltados à análise pulmonar, os autores aplicaram um conjunto de técnicas que incluiu conversão para tons de cinza, remoção de ruído (salt and pepper, gaussiano e speckle) e realce de contraste. Para a filtragem, utilizaram-se os filtros Wiener, mediana e gaussiano, com o objetivo de reduzir diferentes tipos de ruído sem comprometer as bordas. A equalização de histograma adaptativa foi empregada para melhorar a distribuição dos níveis de intensidade, favorecendo a visibilidade de estruturas relevantes. Os autores observaram que o filtro de mediana apresentou melhor desempenho visual e quantitativo nos testes realizados, destacando, assim, a relevância de adaptar as técnicas ao tipo de ruído presente em cada imagem. O estudo evidencia que a qualidade do pré-processamento influencia diretamente o desempenho das etapas subsequentes na análise automatizada de imagens médicas.

3.2.2 Data Augmentation

Data augmentation é uma técnica amplamente utilizada no treinamento de modelos de aprendizado profundo, especialmente em tarefas de classificação de imagens, com o objetivo de ampliar artificialmente a diversidade e a quantidade de dados disponíveis. Isso é feito por meio de transformações geométricas e fotométricas aplicadas sobre as imagens originais, como rotações, espelhamentos, redimensionamentos, e alterações de brilho e contraste. A Figura 3 ilustra a aplicação de algumas dessas transformações.

Essas variações geram novas amostras que preservam a semântica da imagem original, mas oferecem ao modelo diferentes representações do mesmo conteúdo, promovendo um aprendizado mais robusto e generalizável. Essa abordagem contribui significativamente para a redução do sobreajuste (overfitting) e para o aumento da acurácia. A técnica se mostra particularmente eficaz em cenários onde há escassez de dados, como em aplicações médicas ou conjuntos de dados especializados, nos quais a coleta de novos exemplos é limitada por questões éticas, legais ou financeiras (Wang *et al.*, 2017).

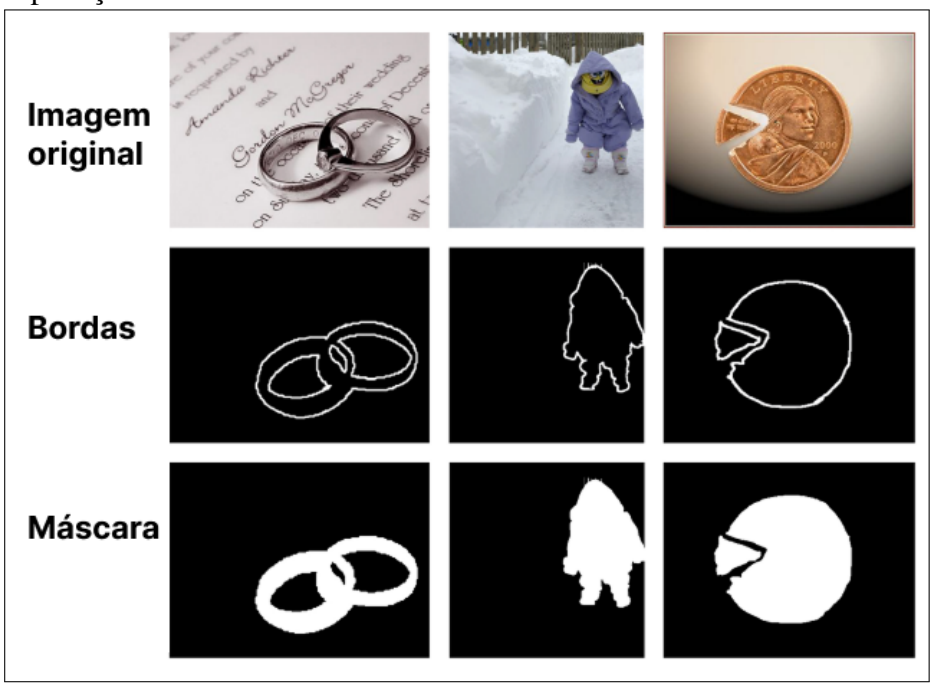
Figura 3 – Aplicação do data augmentation.

Fonte: (Wang *et al.*, 2017).

3.2.3 flood fill

O algoritmo Flood Fill é uma técnica fundamental em processamento de imagens utilizada para o preenchimento de regiões delimitadas dentro de uma imagem digital. Seu funcionamento baseia-se na seleção de um pixel inicial, conhecido como semente, a partir do qual o algoritmo se propaga para os pixels vizinhos que compartilham características semelhantes, como a mesma cor ou valor de intensidade. Essa propagação pode considerar dois tipos de conectividade: a 4-conectividade, que avalia apenas os pixels adjacentes nas direções vertical e horizontal, e a 8-conectividade, que inclui também os vizinhos diagonais. O Flood Fill é amplamente aplicado em tarefas de segmentação, edição de imagens e geração de máscaras para aprendizado de máquina. No entanto, o algoritmo tradicional exige a definição manual do ponto inicial e pode apresentar dificuldades em regiões com contornos complexos ou múltiplas áreas internas. Ainda assim, sua simplicidade e eficiência fazem dele uma escolha viável para aplicações em que a definição da semente é controlada ou conhecida previamente (He *et al.*, 2019).

Figura 4 – Aplicação da técnica flood fill.



Fonte: (He *et al.*, 2019).

3.3 Segmentação de Imagens Médicas com Deep Learning

3.3.1 Redes Neurais Artificiais

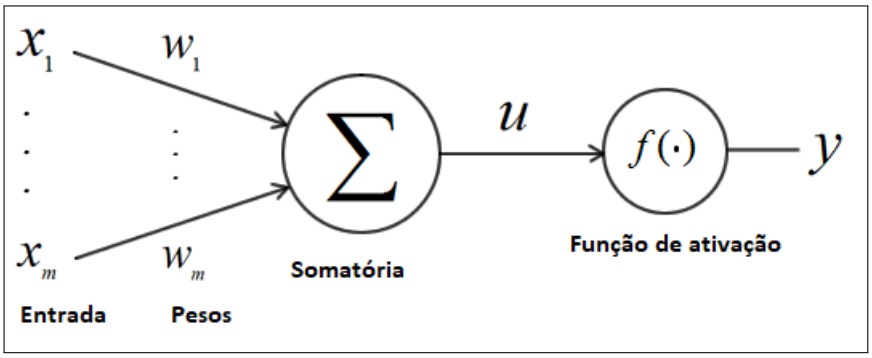
As Redes Neurais Artificiais (RNAs) são modelos de aprendizado de máquina inspirados no funcionamento dos neurônios biológicos e têm como objetivo simular, ainda que de forma simplificada, o comportamento do cérebro humano. Um dos modelos mais elementares de RNA é o perceptron, que representa um único neurônio artificial. Ele recebe valores de entrada X_i, cada um associado a um peso W_i, realiza uma somatória ponderada dessas entradas e, em seguida, aplica uma função de ativação $f(.)$ ao resultado para produzir a saída y (Pacheco, 2015).

A descrição matemática do perceptron é apresentada na Equação 3.2, enquanto sua representação gráfica pode ser observada na Figura 5.

$$y = f\left(\sum_{i=1}^{n} X_i \cdot W_i\right) \tag{3.2}$$

Uma rede neural simples é composta por três camadas principais, que são a camada de entrada, a camada oculta e a camada de saída. A camada de entrada é responsável por receber os dados brutos, como imagens ou informações clínicas. Em seguida, a camada oculta processa esses dados por meio de cálculos e atribuições de pesos. Por fim, a camada de saída

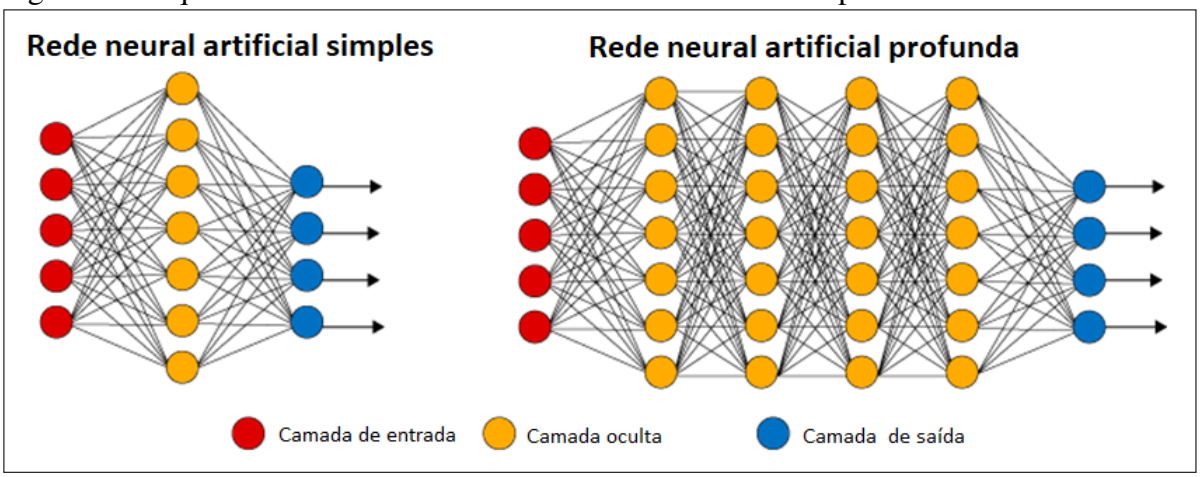
Figura 5 – Modelo de um neurônio artificial.



Fonte: (Pacheco, 2015), modificado pelo autor.

fornece os resultados da inferência, como a classificação ou o diagnóstico. A principal diferença entre uma rede neural simples e uma rede profunda está na quantidade de camadas ocultas. As redes profundas, conhecidas como Deep Neural Networks (DNNs), possuem múltiplas camadas intermediárias. Isso permite a dentificação de padrões mais complexos nos dados de entrada. A Figura 6 ilustra a diferença entre uma arquitetura simples e uma rede profunda (Soares, 2019).

Figura 6 – Esquemático de uma rede neural e de uma rede neural profunda.



Fonte: (Soares, 2019), modificado pelo autor.

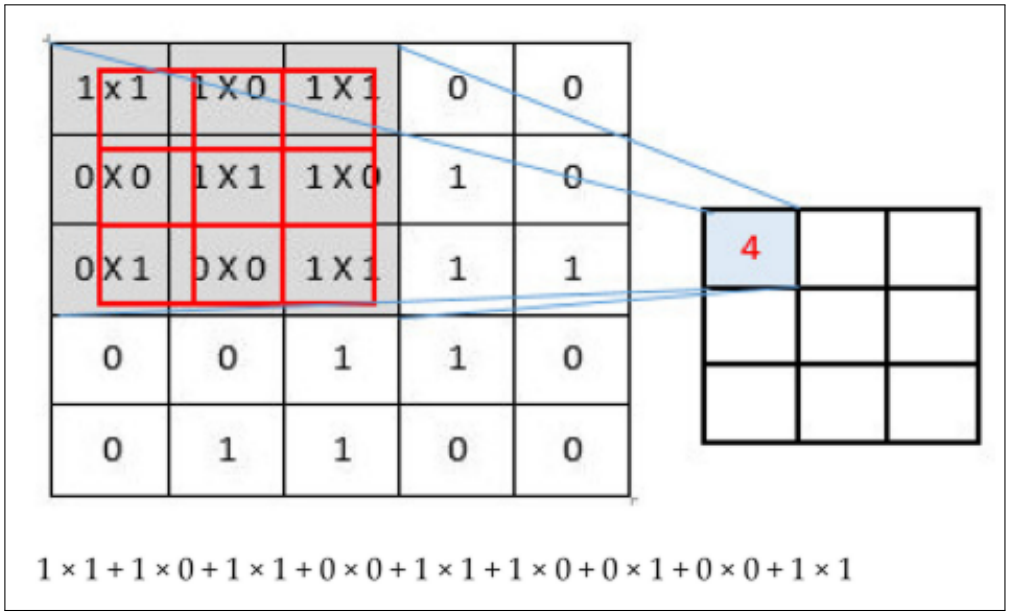
3.3.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) representam uma evolução das RNAs tradicionais e são especialmente projetadas para o processamento de dados com estrutura espacial, como imagens. Diferentemente das abordagens que exigem a extração manual de características (*feature engineering*), as CNNs são capazes de aprender automaticamente, de forma *end-to-end*, os padrões relevantes diretamente a partir dos dados brutos.

O funcionamento das CNNs baseia-se na aplicação de filtros convolucionais que percorrem as imagens para identificar características visuais em diferentes escalas, como bordas, texturas e padrões complexos, acompanhados por operações que reduzem a dimensionalidade dos dados e aumentam a eficiência computacional. Isso permite que a rede aprenda representações cada vez mais abstratas das imagens originais.

Conforme descrito por Bhatt *et al.* (2021), uma Rede Neural Convolucional típica é estruturada em uma sequência de camadas. A primeira etapa fundamental é a camada convolucional, que aplica filtros (kernels) que escaneiam a imagem para detectar padrões locais, gerando mapas de características (*feature maps*). A Figura 7 ilustra esse processo, mostrando a matriz do filtro se deslocando sobre a imagem e produzindo, como resultado, o mapa de características.

Figura 7 – Camada de convolução.



Fonte: (Bhatt *et al.*, 2021).

Matematicamente, a convolução 2D aplicada sobre uma imagem pode ser expressa pela Equação 3.3, onde I representa a imagem de entrada, K é o kernel de convolução de dimensões $(2m+1) \times (2n+1)$, e $S(x,y)$ é o valor resultante da operação na coordenada (x,y) da imagem de saída.

$$S(x,y) = (I * K)(x,y) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} I(x+i, y+j) K(i,j) \tag{3.3}$$

Logo após a convolução, uma camada de ativação introduz não-linearidade no modelo. A mais comum é a função *Rectified Linear Unit* (ReLU), que permite à rede aprender

relações mais complexas e resolver problemas que não seriam modelados apenas com operações lineares. Definida pela Equação 3.4, essa função ativa apenas os valores positivos, anulando os negativos, o que acelera o treinamento e reduz o risco do problema do gradiente desaparecendo.

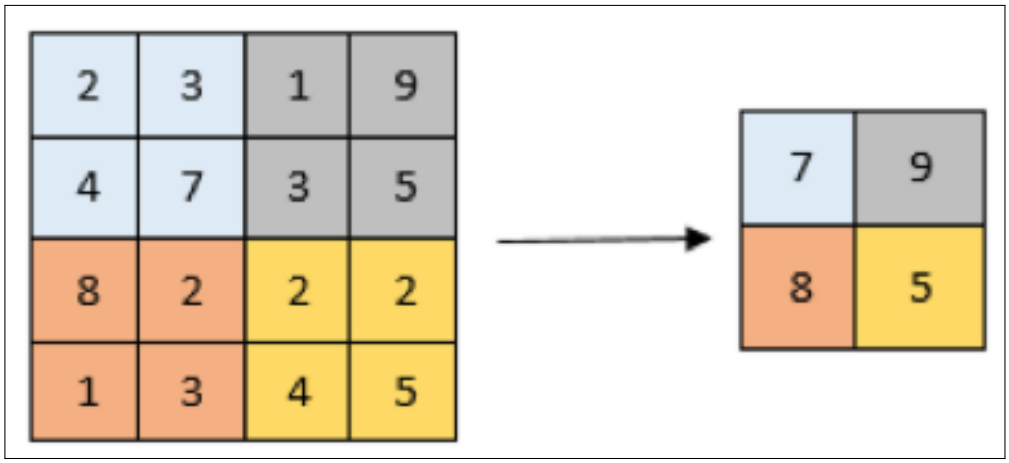
$$f(x) = \begin{cases} 0, & \text{se } x < 0 \\ x, & \text{se } x \geq 0 \end{cases} \tag{3.4}$$

A etapa seguinte é a camada de pooling, que reduz a resolução espacial dos mapas de características, preservando as informações mais relevantes e tornando a rede mais robusta a deslocamentos e variações. A técnica mais comum é o *max pooling*, que seleciona o valor máximo dentro de uma região específica da imagem, conforme descrito na Equação 3.5. Nela, $R(x,y)$ representa a vizinhança local considerada em torno da posição (x,y), e $S(i,j)$ são os valores do mapa de características de entrada.

$$P(x,y) = \max_{(i,j)\in R(x,y)} S(i,j) \tag{3.5}$$

A Figura 8 exemplifica o funcionamento do *max pooling*, evidenciando como, em cada região da imagem, é selecionado o valor máximo para compor um novo mapa de características com resolução reduzida.

Figura 8 – Max Pooling.



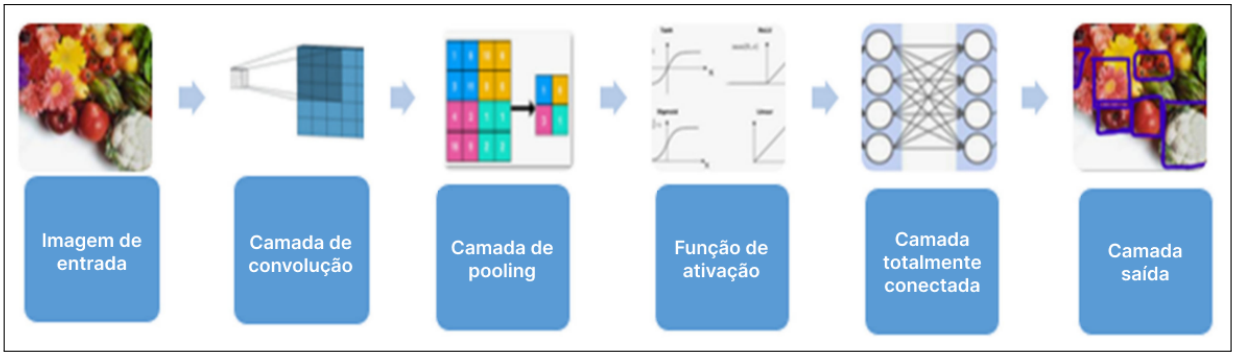
Fonte: (Bhatt *et al.*, 2021).

Após uma ou mais sequências de convolução, ativação e pooling, os mapas de características passam por uma camada de flattening, que os transforma em um vetor unidimensional. Esse vetor serve de entrada para as camadas totalmente conectadas (*Fully Connected*), que

funcionam como classificadores e produzem a saída final com base nas características extraídas. Por fim, a camada de saída gera a predição da rede, geralmente utilizando uma função de ativação como *softmax* ou *sigmoid*, dependendo da tarefa.

A Figura 9 apresenta a estrutura típica de uma Rede Neural Convolucional, destacando as principais camadas que a compõem. Esse esquema auxilia na visualização do fluxo de processamento das informações ao longo da rede.

Figura 9 – Componentes da arquitetura CNN tradicional.



Fonte: (Bhatt *et al.*, 2021).

Durante o treinamento, os pesos dessas camadas são ajustados com o uso de algoritmos de otimização, como o gradiente descendente, que minimizam a diferença entre a predição da rede e os rótulos reais do conjunto de dados.

As CNNs tradicionais destacam-se especialmente em tarefas de classificação de imagens, nas quais o objetivo é associar a imagem a uma categoria predefinida, como a presença ou ausência de uma determinada condição. Essa característica faz com que sejam amplamente utilizadas em aplicações que demandam decisões rápidas e precisas a partir de grandes volumes de imagens, como na triagem automatizada em contextos médicos (Mohammed *et al.*, 2024).

3.3.3 Arquitetura U-Net

A arquitetura U-Net, proposta por Ronneberger *et al.* (2015), foi desenvolvida para realizar segmentações precisas em imagens biomédicas, mesmo com conjuntos de dados reduzidos. Seu nome deriva do formato visual da rede, que lembra a letra “U”, composta por dois caminhos principais: o caminho de contração (encoder) e o caminho de expansão (decoder). Essa estrutura é ilustrada na Figura 10, que destaca o formato simétrico da rede, com os caminhos de compressão e expansão conectados por *skip connections*.

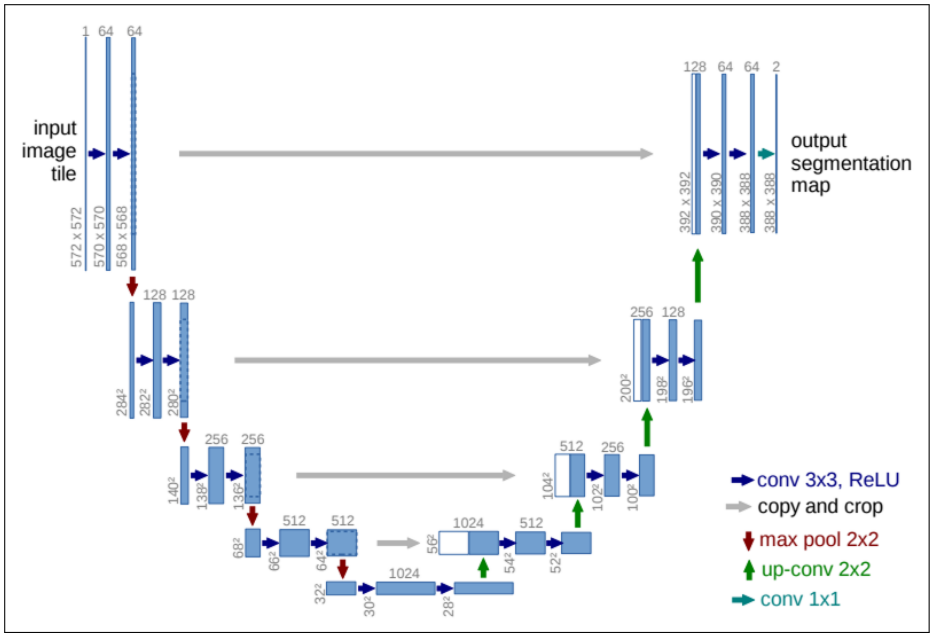
A U-Net distingue-se por ser uma rede convolucional simétrica, que utiliza conexões

de atalho (*skip connections*) entre os blocos equivalentes do encoder e do decoder, permitindo a preservação de informações espaciais perdidas durante o processo de compressão.

A arquitetura é composta por uma sequência de camadas que podem ser descritas em quatro etapas principais. O caminho de contração, ou *encoder*, é responsável por capturar o contexto da imagem por meio da extração progressiva de características em diferentes escalas. Ele consiste em blocos repetidos de duas camadas convolucionais com ativação ReLU, seguidas por uma camada de *pooling* (geralmente *MaxPooling*), que diminui a resolução espacial enquanto aumenta o número de filtros.

Na parte mais profunda da rede, encontra-se o *bottleneck*, o ponto de menor resolução e maior profundidade de filtros, que atua como uma camada de transição entre os processos de codificação e decodificação. Em seguida, o caminho de expansão, ou *decoder*, reconstrói a segmentação realizando o *upsampling* das ativações por meio de convoluções transpostas. Cada etapa do decoder é conectada diretamente ao seu par correspondente no encoder pelas *skip connections*, enriquecendo o processo de reconstrução com informações de borda e detalhes espaciais. Por fim, a camada final de saída aplica uma convolução de 1x1 para gerar o mapa de segmentação, onde cada pixel é classificado.

Figura 10 – Estrutura da arquitetura U-Net.



Fonte: (Ronneberger *et al.*, 2015).

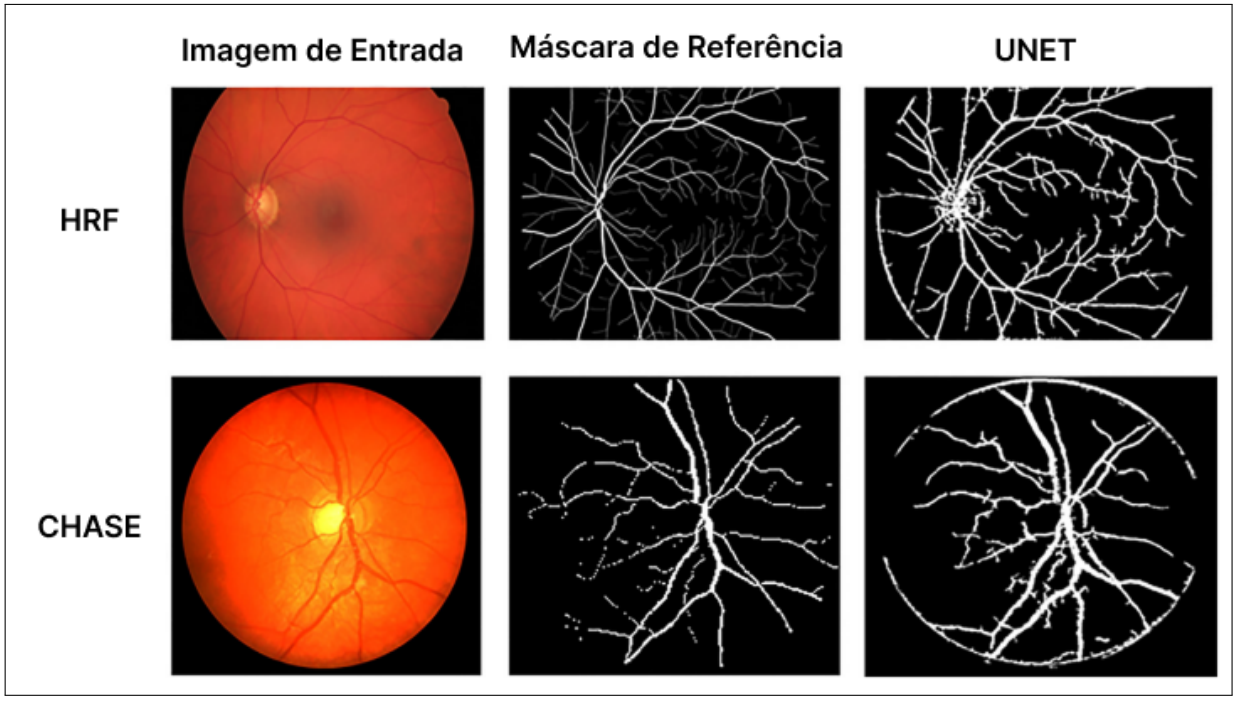
Além disso, a U-Net adota a estratégia de segmentação com blocos sobrepostos (*overlap-tile strategy*), o que permite lidar com imagens de grandes dimensões ao segmentar partes menores com sobreposição e espelhamento nas bordas. Essa abordagem, combinada com

técnicas de *data augmentation* como rotações, deslocamentos e deformações elásticas, contribui para o treinamento eficaz mesmo com conjuntos de dados limitados (Ronneberger *et al.*, 2015).

Em síntese, a U-Net é amplamente reconhecida por sua precisão em tarefas de segmentação semântica, especialmente em aplicações médicas, por permitir a classificação pixel a pixel de estruturas anatômicas com elevado grau de detalhe. Diferencia-se das redes convolucionais tradicionais por seu foco na preservação da informação espacial e por sua capacidade de gerar máscaras segmentadas em alta resolução.

Essa eficácia tem sido amplamente comprovada em diferentes domínios da medicina, como na extração de vasos sanguíneos, identificação de tumores e delineamento de órgãos. A arquitetura simétrica da U-Net permite a reconstrução precisa das regiões segmentadas, mesmo em imagens com baixo contraste ou ruído. A Figura 11 ilustra um exemplo prático da aplicação da U-Net na segmentação de vasos retinianos, evidenciando a separação das estruturas vasculares em imagens de retina (Sathananthavathi e Indumathi, 2021).

Figura 11 – Exemplo de aplicação da U-Net em imagens médicas. Segmentação de vasos sanguíneos em imagens de retina dos bancos de dados HRF e CHASE.



Fonte: Adaptado de (Sathananthavathi e Indumathi, 2021).

Função de perda e métricas de avaliação

A qualidade das segmentações geradas pela arquitetura U-Net é avaliada por meio de métricas específicas amplamente utilizadas na literatura, sendo a função de perda Dice utilizada

durante o treinamento, e as métricas de IoU e acurácia por pixel aplicadas na avaliação dos resultados no conjunto de teste.

A função de perda baseada no coeficiente de Dice é particularmente adequada para tarefas de segmentação, pois mede a sobreposição entre as máscaras previstas e as máscaras reais, sendo robusta a desbalanceamentos entre classes. Sua formulação suavizada é descrita na Equação 3.6, em que p_i representa o valor do pixel previsto, g_i o pixel correspondente da máscara de referência (*ground truth*), e ε é um termo de suavização para evitar divisão por zero (Eelbode *et al.*, 2020).

$$\mathscr{L}_{Dice} = 1 - \frac{2\sum_i p_i g_i}{\sum_i p_i^2 + \sum_i g_i^2 + \varepsilon} \tag{3.6}$$

A métrica IoU (*Intersection over Union*), também conhecida como coeficiente de Jaccard, calcula a razão entre a interseção e a união dos conjuntos de pixels previstos e reais, conforme a Equação 3.7, em que P representa a máscara prevista e G a máscara de referência (Yu *et al.*, 2022).

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} \tag{3.7}$$

A acurácia por pixel avalia a proporção de pixels corretamente classificados, tanto positivos quanto negativos, em relação ao total de pixels. A Equação 3.8 ilustra seu cálculo, onde TP, TN, FP e FN representam, respectivamente, os erdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos calculados pixel a pixel (Silva *et al.*, 2024).

$$\text{Acc}_{pixel} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.8}$$

Intervalo de Confiança

O intervalo de confiança (IC) é uma medida estatística utilizada para estimar, com determinado grau de certeza, a faixa na qual se espera que esteja o valor verdadeiro de um parâmetro populacional, com base em uma amostra. Em contextos de avaliação de modelos de aprendizado de máquina, como a segmentação de imagens, o IC é aplicado para indicar a confiabilidade das métricas calculadas a partir de um conjunto de dados limitado (Ballester *et al.*, 2000).

Um IC de 95%, por exemplo, indica que, se múltiplas amostras fossem extraídas e o mesmo experimento repetido, aproximadamente 95% dos intervalos calculados conteriam o valor verdadeiro da métrica. Essa abordagem é particularmente útil para inferir a estabilidade e a generalização do desempenho do modelo.

A fórmula mais comum para o cálculo do intervalo de confiança é baseada na distribuição normal padrão e pode ser expressa pela Equação 3.9, em que $\bar{x}$ é a média da amostra, s é o desvio padrão da amostra, n é o tamanho da amostra e z é o valor crítico da distribuição normal para o nível de confiança desejado (por exemplo, $z = 1{,}96$ para 95% de confiança).

$$IC = \bar{x} \pm z \cdot \frac{s}{\sqrt{n}} \tag{3.9}$$

Esse cálculo permite quantificar a incerteza associada às estimativas de desempenho do modelo, fornecendo uma faixa estatisticamente confiável para interpretação dos resultados.

3.4 Flutter e interfaces web

Com o crescimento da demanda por aplicações multiplataforma, a utilização de frameworks modernos para desenvolvimento web tem se tornado essencial. Dentre as tecnologias mais populares, destacam-se Flutter, React e Angular, cada uma com características próprias, vantagens e desvantagens. Embora o Flutter tenha sido inicialmente desenvolvido com foco em aplicações mobile, ele passou a oferecer suporte à web, possibilitando o uso de uma base de código única para múltiplas plataformas (Khan *et al.*, 2022).

O Flutter, framework de código aberto desenvolvido pelo Google, tem se destacado como uma alternativa promissora para o desenvolvimento de aplicações multiplataforma, incluindo aplicações web. Seu diferencial reside na capacidade de oferecer interfaces ricas e responsivas a partir de uma única base de código, utilizando a linguagem Dart e a engine gráfica Skia. O Flutter Web diferencia-se das soluções tradicionais por não utilizar o DOM convencional. Em vez disso, renderiza os componentes diretamente com HTML, CSS e WebAssembly, usando o mecanismo gráfico Skia. Essa abordagem proporciona maior controle visual e consistência entre plataformas, embora possa apresentar limitações em termos de SEO e acessibilidade (Müller, 2021).

Diversos estudos têm explorado o potencial e as limitações do Flutter para o desenvolvimento web, especialmente quando comparado a outras abordagens. Tran (2020) analisa o

desempenho do Flutter em dispositivos móveis, destacando que o framework oferece interfaces dinâmicas e desempenho nativo, mesmo em comparação com soluções como o Cordova. Embora seu foco esteja em plataformas móveis, o autor observa que a estrutura de widgets do Flutter, aliada à compilação AOT (Ahead-Of-Time), proporciona uma experiência fluida também para aplicações web, quando bem implementadas.

Complementando essa perspectiva, Piskor e Badurowicz (2023) concluíram que o desempenho do Flutter em navegadores web é inferior ao observado em ambientes nativos, apresentando tempos de carregamento mais longos e uma reconstrução de telas menos eficiente. Essa constatação foi baseada em testes com a mesma aplicação sendo executada em diferentes ambientes, validando as hipóteses iniciais dos autores sobre as limitações de performance do Flutter Web.

Por outro lado, Müller (2021) oferece uma análise comparativa entre Flutter e Electron no contexto de aplicações desktop e web. O autor reconhece o potencial do Flutter ao observar que ele exige cerca de 30% menos código e oferece um fluxo de desenvolvimento mais rápido. No entanto, os testes realizados apontaram que a CPU e a GPU apresentaram uso elevado, acima de 10% e 40% respectivamente, além da ausência de suporte a funcionalidades importantes como múltiplas janelas e menus de contexto personalizáveis. Esses resultados indicam que, apesar do amadurecimento da tecnologia, seu uso em aplicações web e desktop ainda requer aprimoramentos para atingir maturidade plena.

Do ponto de vista arquitetural, Boukhary e Colmenares (2019) propõem a utilização da Clean Architecture adaptada ao Flutter, estruturando a aplicação em camadas independentes (dados, domínio e apresentação). Essa abordagem facilita a manutenção e o reuso de código, inclusive para aplicações web. Os autores destacam que a lógica de negócios escrita em Dart pode ser aproveitada em múltiplas plataformas, incluindo aplicações web, por meio da transpilação do código para JavaScript. A separação de responsabilidades promove um desenvolvimento mais robusto, escalável e compatível com os desafios da web.

Paralelamente, é relevante analisar outras tecnologias amplamente utilizadas. O React, biblioteca JavaScript desenvolvida pelo Facebook, baseia-se em componentes reutilizáveis e utiliza renderização eficiente via Virtual DOM. Sua flexibilidade, aliada a uma comunidade ativa e ao suporte de bibliotecas como Redux, Next.js e React Router, torna-o uma das escolhas mais populares para aplicações web dinâmicas (Khan *et al.*, 2022).

O Angular, por sua vez, é um framework completo desenvolvido pelo Google, que

adota a linguagem TypeScript e oferece soluções integradas para gerenciamento de estado, formulários, rotas e injeção de dependência. Essa robustez o torna ideal para aplicações de larga escala, especialmente em ambientes corporativos, embora sua curva de aprendizado seja mais acentuada (Fentaw, 2020).

Em termos de desempenho, o Flutter Web destaca-se na renderização gráfica, sendo indicado para interfaces ricas e personalizadas. Contudo, seu tempo de carregamento inicial é superior, o que pode impactar a experiência do usuário em conexões mais lentas. React apresenta ótima performance com menor carga inicial, enquanto Angular exige ajustes como lazy loading para otimizar sua eficiência (Müller, 2021).

Em conjunto, os estudos analisados demonstram que o Flutter possui um grande potencial para o desenvolvimento de interfaces web modernas, ricas e interativas. Apesar de ainda enfrentar limitações de desempenho e maturidade em certos aspectos, o framework destaca-se pela unificação da base de código, pela consistência visual entre plataformas e pela agilidade no desenvolvimento. Tais características o tornam uma alternativa promissora frente a outras tecnologias, especialmente em projetos que exigem interfaces personalizadas, responsivas e com alto nível de controle visual.

A fim de sintetizar as principais diferenças entre os frameworks mais utilizados no desenvolvimento web, o Quadro 2 apresenta uma comparação entre Flutter Web, React e Angular, com base em critérios como linguagem, arquitetura, curva de aprendizado, desempenho e casos de uso.

3.5 Flask e integração com IA

O trabalho de Mufid *et al.* (2019) destaca a utilização do framework Flask como uma solução leve e eficiente para o desenvolvimento de backends em aplicações web. Por ser um microframework em Python, o Flask permite construir aplicações de forma rápida, com menor dependência de bibliotecas externas, o que o torna ideal para projetos simples a intermediários. A principal contribuição do estudo está na organização estrutural do código, propondo uma arquitetura baseada no padrão Model-View-Controller (MVC), que não é nativamente imposta pelo Flask. Essa abordagem melhora a separação de responsabilidades entre o acesso a dados (model), a lógica de controle (controller) e a apresentação (view), resultando em maior clareza, manutenibilidade e escalabilidade do backend. Além disso, o trabalho apresenta um gerador automático de projetos que cria a estrutura inicial de pastas e arquivos, facilitando a adoção do

Quadro 2 – Comparação entre Flutter Web, React e Angular

Critério	**Flutter Web**	**React**	**Angular**
Linguagem	Dart	JavaScript / JSX	TypeScript
Tipo	Framework baseado em widgets	Biblioteca de UI	Framework completo
Arquitetura	Canvas + Skia (sem DOM tradicional)	Virtual DOM, baseado em componentes	MVC / MVVM
Curva de aprendizado	Moderada (exige aprender Dart)	Baixa a moderada	Alta (estrutura complexa e rígida)
Performance (Web)	Alta em renderização, carga inicial maior	Muito boa com otimizações	Boa, mas mais pesada sem ajustes
Comunidade	Crescente, com suporte do Google	Muito ativa e consolidada	Estável, com uso em grandes sistemas
SEO e Acessibilidade	Limitado	Bom (melhor com Next.js e SSR)	Bom
Produtividade	Alta (hot reload, widget tree)	Alta (componente + IDEs flexíveis)	Média (configurações extensas)
Integrações	Boas, porém menos plugins para Web	Excelente (ecossistema JS completo)	Completa, com suporte integrado (RxJS, etc.)
Casos de uso ideal	Interfaces ricas, multiplataforma	SPAs, painéis interativos, dashboards	Sistemas empresariais de grande porte

Fonte: Elaborado pelo autor com base em Khan *et al.* (2022), Müller (2021), Fentaw (2020).

Flask por desenvolvedores iniciantes e otimizando o tempo de configuração de novos sistemas.

O trabalho de Ghimire (2020) apresenta uma comparação entre os frameworks Flask e Django, com ênfase prática na construção de aplicações web. No caso do Flask, o autor destaca sua leveza, simplicidade e flexibilidade como grandes vantagens para o desenvolvimento de backends ágeis e personalizados. Utilizando o padrão de projeto MVC, o Flask oferece ao desenvolvedor maior controle sobre a estrutura da aplicação, além de disponibilizar uma ampla variedade de extensões para tarefas comuns, como gerenciamento de banco de dados (com ORMs como Peewee ou SQLAlchemy), autenticação (Flask-Login), proteção contra CSRF (Flask-WTF) e deploy em nuvem com servidores como Gunicorn via Heroku. A documentação clara, a comunidade ativa e a curva de aprendizado mais suave tornam o Flask ideal para protótipos e aplicações de pequeno a médio porte. O estudo evidencia que, apesar de ser um microframework, o Flask oferece os recursos necessários para desenvolver aplicações completas no lado do servidor, com a vantagem de permitir a escolha das ferramentas conforme a necessidade do projeto.

O trabalho de Singh *et al.* (2019) apresenta a implementação de um sistema completo de gestão universitária utilizando o framework Flask em Python, evidenciando seu potencial para

o desenvolvimento de backends robustos e modulares. O sistema abrange funcionalidades como controle de presença, gerenciamento de biblioteca, transporte, resultados acadêmicos e módulo de estágio e emprego, permitindo diferentes níveis de acesso para estudantes e funcionários. O Flask foi adotado como base por sua leveza, flexibilidade e fácil integração com bancos de dados MySQL. O estudo também destaca o uso de extensões como WTForms para criação de formulários e suporte à autenticação. O projeto demonstra como o Flask permite estruturar sistemas administrativos com múltiplos módulos, facilitando tanto a criação quanto a manutenção do sistema, com foco em segurança, escalabilidade e organização de dados.

O artigo de Vyshnavi e Malik (2019) evidencia como o uso combinado de Python e Flask oferece uma forma eficiente e prática para o desenvolvimento de backends em aplicações web. O Flask é apresentado como um microframework leve que fornece ferramentas essenciais, como o mecanismo de templates Jinja2 e o toolkit WSGI Werkzeug, permitindo a construção de aplicações dinâmicas com uma estrutura clara entre arquivos estáticos e templates. O trabalho destaca aspectos técnicos como suporte nativo a testes unitários, cookies seguros e compatibilidade com o Google App Engine. O uso do Flask para organizar páginas com herança de templates e roteamento flexível demonstra sua capacidade de manter o backend modular e reutilizável. Além disso, o estudo exemplifica a integração com bibliotecas externas como OpenCV para reconhecimento de gestos, reforçando o Flask como uma opção robusta para aplicações que combinam backend com funcionalidades computacionais mais avançadas, como visão computacional e aprendizado de máquina.

O artigo de Bonney *et al.* (2022) apresenta o desenvolvimento de uma plataforma operacional de gêmeo digital baseada em navegador, utilizando o framework Flask como base para conectar dispositivos físicos (gêmeos físicos) a interfaces digitais com funcionalidades analíticas e de controle. O Flask foi escolhido por sua leveza, flexibilidade e facilidade de integração com ferramentas científicas em Python, permitindo o desenvolvimento de uma aplicação web modular, escalável e acessível remotamente. A plataforma, chamada DTOP-Cristallo, demonstra como o Flask pode estruturar o backend de aplicações complexas com funcionalidades como simulações personalizadas, leitura/gravação em banco de dados (PostgreSQL com SQLAlchemy), gerenciamento de filas de tarefas (Redis) e integração com softwares externos como ABAQUS. A arquitetura do sistema separa claramente as camadas de interface, IoT e serviços em nuvem, evidenciando como o Flask pode ser utilizado não apenas para criar APIs e rotas web, mas também como núcleo de controle e orquestração de operações científicas e

industriais em ambientes distribuídos.

O artigo de Yaganteeswarudu (2020) apresenta um sistema de predição de múltiplas doenças médicas utilizando algoritmos de machine learning integrados a uma API construída com o Flask, destacando seu potencial como backend leve e eficiente para aplicações baseadas em inteligência artificial. A proposta central consiste em permitir que, a partir de uma única interface, o usuário possa submeter dados clínicos e obter diagnósticos simultâneos para doenças como diabetes, retinopatia diabética, doenças cardíacas e câncer de mama. O Flask é responsável por gerenciar rotas que recebem os dados dos pacientes via requisição HTTP, processam esses dados com modelos previamente treinados (armazenados em formatos como .pkl ou .h5) e retornam as predições ao frontend. A modularidade proporcionada pelo framework facilita a adição de novos modelos à medida que novas patologias são incluídas, mantendo uma arquitetura de baixo custo e fácil manutenção. Essa abordagem demonstra como o Flask pode ser eficaz no desenvolvimento de soluções preditivas de saúde que aliam acessibilidade, centralização e escalabilidade.

Já o artigo de Sri *et al.* (2024) propõe uma solução inovadora para automatizar a geração de casos de teste em APIs REST utilizando modelos de linguagem (LLMs), com o Flask atuando como backend para disponibilizar o modelo via API. O sistema define rotas que recebem descrições de APIs REST em linguagem natural e retornam arquivos JSON com casos de teste formatados para o Postman, permitindo que usuários validem suas APIs de forma automatizada. Essa proposta evidencia a capacidade do Flask de integrar modelos de IA, como os da OpenAI, com aplicações práticas voltadas à engenharia de software, mantendo uma estrutura leve e acessível. O framework também se destaca pela facilidade em implementar validações de entrada, organizar rotas de forma clara e servir modelos em tempo real. A integração com ferramentas como Postman e o suporte a fluxos completos de geração, avaliação e execução de testes reforçam o uso do Flask como componente central no desenvolvimento de soluções inteligentes e escaláveis voltadas para o Quality Assurance automatizado.

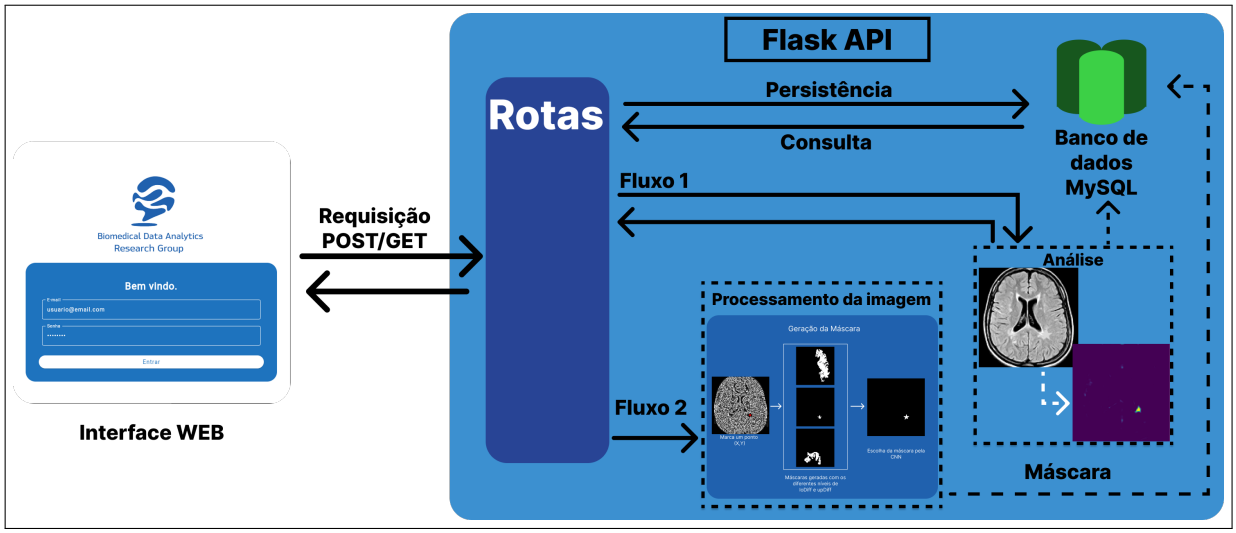
Dessa forma, os estudos analisados evidenciam o papel do Flask como um framework leve, modular e poderoso para a construção de APIs que integram modelos de inteligência artificial. Seja no contexto de aplicações clínicas, industriais ou de automação de testes, o Flask demonstra ser uma escolha estratégica para o desenvolvimento de sistemas inteligentes e escaláveis, oferecendo uma arquitetura flexível e compatível com diversas bibliotecas do ecossistema Python, além de facilitar a implementação de soluções voltadas à predição.

4 METODOLOGIA

A presente metodologia descreve o desenvolvimento e a integração dos componentes da ferramenta de Computer-Aided Diagnosis (CAD) para segmentação de imagens de ressonância magnética (MRI) do cérebro humano. A ferramenta foi projetada de forma modular, integrando uma interface web, uma API em Flask e agentes de inteligência artificial implementados em PyTorch, permitindo tanto a utilização de modelos pré-treinados quanto o re-treinamento com novos dados.

O fluxo geral da ferramenta, ilustrado na Figura 12, pode ser dividido em dois principais processos. O primeiro consiste na análise da imagem, cujo resultado é a segmentação destacada da área de interesse. No segundo processo, o usuário envia para a aplicação a imagem com a marcação de um ponto indicativo da região de interesse. A partir dessa informação, a máscara correspondente é gerada e utilizada para re-treinar o modelo, permitindo sua adaptação a uma nova estrutura de dados.

Figura 12 – Fluxo geral da aplicação



Fonte: elaborado pelo autor (2025).

4.1 Arquitetura da Ferramenta

A ferramenta CAD proposta foi desenvolvida com uma arquitetura modular, estruturada em três camadas principais: a interface do usuário, o backend (API) e os agentes de inteligência artificial. Essa divisão permite escalabilidade, maior adaptabilidade e facilita a manutenção e evolução contínua dos modelos empregados.

A solução é composta por um pipeline robusto que conecta uma interface interativa desenvolvida em Flutter, uma API em Flask e modelos de IA implementados em PyTorch. Essa integração possibilita diagnósticos assistidos por computador por meio de duas abordagens complementares. A primeira, o Processo de Análise, utiliza um modelo UNet pré-treinado para segmentação automática de imagens de ressonância magnética, proporcionando segmentação imediata da área de interesse. A segunda abordagem, o Processo de Geração de Dados, permite o enriquecimento do conjunto de dados por meio de pré-processamento, data augmentation e geração de máscaras via Flood Fill, com validação posterior por uma CNN, garantindo uma melhoria contínua do modelo UNet.

O backend, implementado em Flask, é responsável pelo processamento central da ferramenta. Ele recebe as imagens enviadas pela interface e executa o pré-processamento, aplicando técnicas como normalização, equalização de histograma e filtro SOBEL para realce de bordas. Além disso, ele realiza data augmentation para aumentar a diversidade dos dados de treinamento. Dentro do backend, há um módulo dedicado à geração de máscaras, que utiliza o algoritmo Flood Fill para criar máscaras binárias com diferentes níveis de tolerância a partir dos pontos selecionados pelo usuário. Essas máscaras são avaliadas por um modelo CNN, garantindo que apenas as de alta qualidade sejam utilizadas no re-treinamento do modelo UNet.

A interface do usuário, desenvolvida com Flutter, desempenha um papel central na interação com o sistema. Ela permite o upload e a visualização das imagens originais e pré-processadas, oferece ferramentas interativas como zoom e seleção de pontos de interesse, além de possibilitar o início do treinamento do modelo com os dados disponíveis. Essa camada foi projetada para garantir uma experiência intuitiva e eficiente na segmentação assistida por IA.

O armazenamento de informações na aplicação segue uma abordagem híbrida para garantir eficiência e organização. As imagens utilizadas no treinamento do modelo são armazenadas em diretórios estruturados, otimizando o desempenho no carregamento e processamento. Paralelamente, um banco de dados é utilizado para registrar os metadados associados, incluindo nomes de arquivos, datas de upload, status do processamento, resultados das segmentações e validações realizadas pelo modelo. Essa estratégia permite consultas rápidas e facilita o gerenciamento dos experimentos, garantindo rastreabilidade e controle sobre o fluxo de dados sem comprometer a performance do sistema.

Por fim, a camada de inteligência artificial é composta por dois modelos principais. O primeiro é o Modelo UNet, responsável pela segmentação das imagens de ressonância magnética,

que identifica automaticamente as regiões de interesse com base no treinamento prévio e nos novos dados incorporados ao sistema. O segundo é o Modelo CNN, que avalia a qualidade das máscaras geradas pelo algoritmo Flood Fill. Essa avaliação assegura que apenas as máscaras consideradas adequadas sejam utilizadas no re-treinamento do modelo UNet, aprimorando continuamente sua capacidade de segmentação.

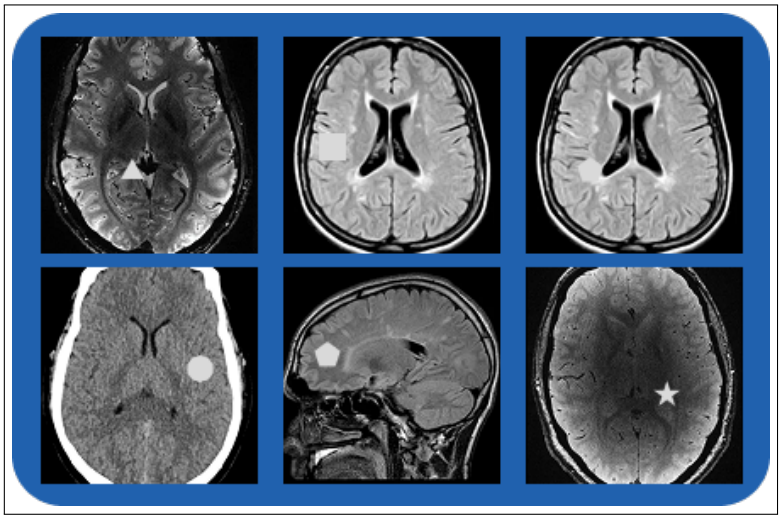
4.2 Banco de dados e Pré-Processamento

O banco de dados foi construído a partir de imagens de ressonância magnética do cérebro humano, extraídas do banco de dados "Psilocybin Precision Functional Mapping", que contém dados sobre como a psilocibina desincroniza as redes neurais do cérebro. As imagens foram padronizadas para ter dimensões de 128×128 pixels, garantindo uniformidade em todo o conjunto. Para simular anomalias e enriquecer os dados para o treinamento, cada imagem foi processada para receber sobreposições de formas geométricas inseridas em posições aleatórias. As formas aplicadas foram triângulo, quadrado, pentágono, decágono e estrela. Elas possuem dimensões de 16×16 pixels e foram renderizadas com cores semelhantes às encontradas nas estruturas cerebrais, criando uma simulação realista de anomalias. As dimensões de 16x16 pixels foram selecionadas para criar um alvo que fosse suficientemente visível na imagem de 128x128 pixels, mas que ainda representasse um desafio de segmentação preciso para o modelo. Na Figura 13 podem ser observadas as formas geométricas utilizadas, e na Figura 14, alguns exemplos de imagens de ressonância magnética com as sobreposições.

Figura 13 – Formas geométricas utilizadas para simular a anomalia

Fonte: Elaborado pelo autor (2025).

Figura 14 – Sobreposição da formas géometricas nas imagens de ressonância magnetica

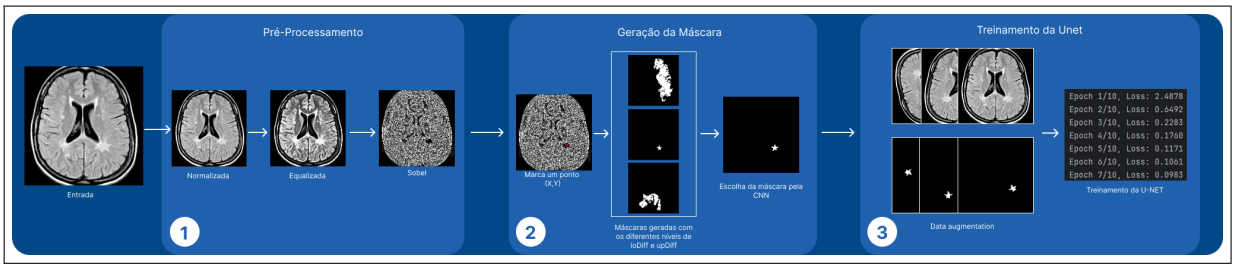


Fonte: elaborado pelo autor (2025).

4.2.1 Estratégias de Pré-Processamento

O pré-processamento das imagens tem como objetivo aprimorar a visualização para o usuário, proporcionando maior segurança no momento da marcação do ponto de interesse. Essa marcação é essencial para que o backend possa gerar a máscara correspondente e re-treinar o modelo UNet. A Figura 15 ilustra em 3 etapas o processo de treinamento da UNet.

Figura 15 – Fluxo do dado para o treinamento do modelo UNet. (1) Pré-processamento da imagem; (2) Indicação da área de interesse e geração da máscara; (3) Data augmentation e treinamento da UNet.



Fonte: elaborado pelo autor (2025).

O pré-processamento, ilustrado na Etapa 1 da Figura 15, inicia-se com a normalização da imagem. Nesse processo, cada imagem é convertida para escala de cinza e seus valores de pixel são ajustados para o intervalo [0,1]. Essa padronização garante que o modelo receba dados com escalas consistentes, minimizando variações decorrentes das diferentes configurações dos equipamentos de ressonância magnética. Esse ajuste é fundamental para melhorar a estabilidade e a convergência do treinamento do modelo.

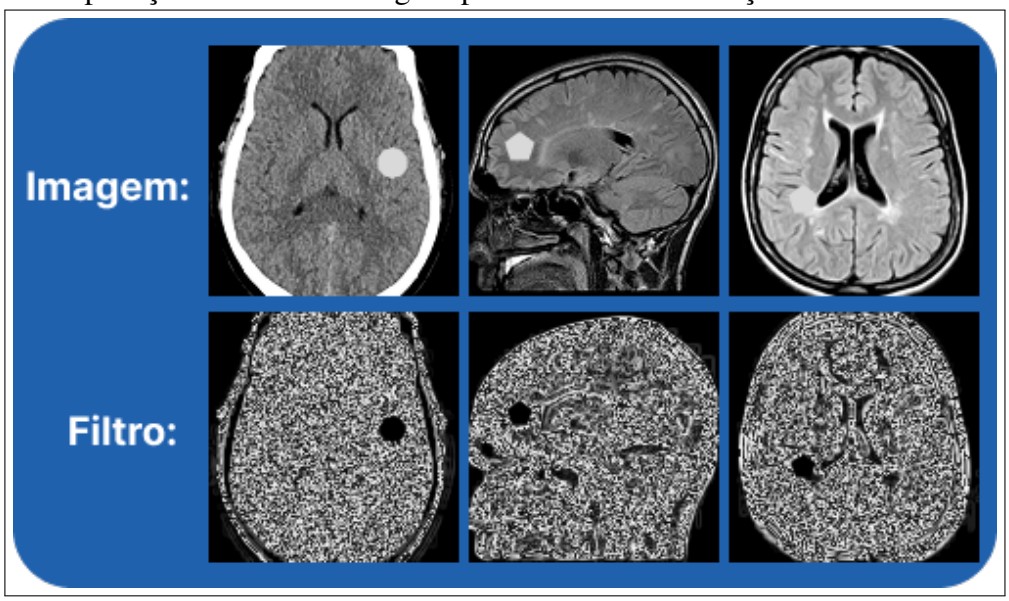
Após a normalização, aplica-se a equalização de histograma para melhorar o con-

traste da imagem. Essa técnica redistribui os valores de intensidade, aumentando a diferenciação entre áreas claras e escuras, facilitando assim a identificação de estruturas anatômicas relevantes.

Além disso, é aplicado o filtro SOBEL para realçar as bordas e transições de intensidade na imagem. O filtro calcula os gradientes de intensidade nas direções horizontal e vertical, permitindo a detecção de contornos que correspondem às estruturas cerebrais. Essa etapa é crucial, pois o filtro SOBEL destaca mudanças abruptas de intensidade, que geralmente correspondem aos limites entre diferentes estruturas anatômicas.

Na Figura 16, apresentam-se exemplos de imagens com a aplicação do filtro SOBEL. Observa-se que a área da simulação da anomalia apresenta um realce maior em comparação à imagem sem nenhum pré-processamento, evidenciando a eficácia do filtro na segmentação dos detalhes anatômicos.

Figura 16 – Aplicação do filtro na imagens para destacar a simulação da anomalia



Fonte: elaborado pelo autor (2025).

4.2.2 Geração da máscara

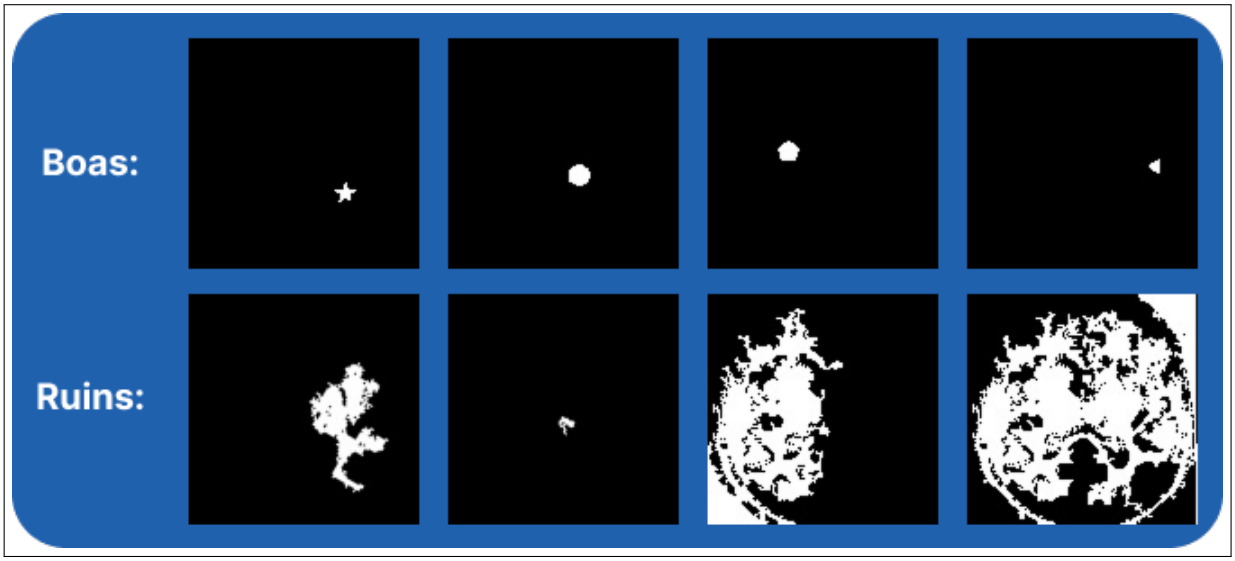
Após o pré-processamento, a região de interesse torna-se mais visível, facilitando a seleção precisa de um ponto no centro da área anômala. Com base nas coordenadas desse ponto, o backend aplica o algoritmo Flood Fill para gerar uma máscara binária que delimita a região conectada. Para aumentar a robustez, o algoritmo é executado com diferentes parâmetros de tolerância (loDiff e upDiff). Se a primeira máscara gerada for classificada como “boa” pelo modelo de validação baseado em CNN, as demais variações não são processadas. Caso contrário,

o sistema gera máscaras adicionais para garantir que apenas dados de alta qualidade sejam utilizados. O processo de geração das máscaras e analise pelo modelo CNN é ilustrado na etapa 2 da figura 15.

O Flood Fill é um algoritmo de preenchimento recursivo amplamente utilizado em segmentação de imagens e gráficos computacionais. Ele opera expandindo a partir de um ponto-semente selecionado, percorrendo os pixels vizinhos e preenchendo aqueles que atendem a um critério de similaridade, definido por um intervalo de tolerância. No contexto deste trabalho, a técnica é aplicada para identificar e segmentar regiões homogêneas na imagem filtrada. A tolerância (loDiff e upDiff) controla o quão semelhantes os pixels vizinhos devem ser em relação ao ponto inicial, permitindo a adaptação a diferentes intensidades e variações na imagem.

Além disso, a imagem filtrada serve como base para o Flood Fill, aproveitando as bordas realçadas para aprimorar a segmentação e garantir a geração precisa das máscaras binárias. Esse processo é essencial para delimitar as regiões de interesse com precisão, reduzindo interferências e garantindo que apenas áreas relevantes sejam analisadas posteriormente. A figura 17 ilustra os 2 grupos de classificação utilizado pelo modelo CNN para analisar as máscaras geradas com os diferentes parâmetros de tolerância.

Figura 17 – Grupo de classificação das máscaras



Fonte: elaborado pelo autor (2025).

É importante destacar que, nesta etapa, as máscaras utilizadas para treinamento foram construídas artificialmente com o uso de formas geométricas simples como estrelas, círculos e quadrados. Essa abordagem foi adotada como uma solução exploratória para simular regiões de interesse segmentadas, permitindo testes preliminares do sistema. No entanto, essa simplificação

compromete a fidelidade morfológica das estruturas reais presentes em exames de ressonância magnética. Como limitação do trabalho, reconhece-se que esse método não é adequado para aplicações clínicas reais, sendo recomendável a substituição por bases de dados rotuladas por especialistas.

4.2.3 Data Augmentation

Após a geração e validação das máscaras pelo modelo CNN, aplica-se a técnica de *Data Augmentation* para expandir a base de dados utilizada no treinamento do modelo U-Net. O processo de aplicação do *data augmentation* e treinamento do modelo UNet é ilustrado na etapa 3 da Figura 15. Cada imagem validada, junto com sua respectiva máscara, é replicada 15 vezes.

Em cada uma das cópias, uma transformação específica é aplicada de forma programática, seguindo critérios baseados no índice da amostra gerada. Para as cópias cujo índice é divisível por 2, é aplicada uma rotação tanto na imagem quanto na máscara, com um ângulo correspondente a 15 graus multiplicado pelo valor do índice. Para as amostras com índice divisível por 3, realiza-se um espelhamento horizontal. Caso o índice da cópia não atenda a nenhum desses critérios, a transformação padrão é manter a imagem original com ruído, garantindo assim a inclusão da amostra sem alterações geométricas no conjunto de treinamento.

Essa abordagem fortalece a robustez do modelo U-Net, permitindo que ele generalize melhor para diferentes variações das imagens médicas. Além disso, ao aumentar a diversidade dos dados de treinamento, reduz-se o risco de sobreajuste (overfitting), garantindo um desempenho mais estável e preciso na segmentação de novas imagens.

4.3 Modelagem e Treinamento

A construção de um modelo de aprendizado profundo eficiente para segmentação e análise de imagens médicas envolve a definição cuidadosa da arquitetura da rede e das estratégias de treinamento. Nesta seção, são detalhados os modelos utilizados na solução proposta, incluindo suas configurações e funcionamento.

Primeiramente, apresenta-se a CNN para verificação de máscaras, uma rede convolucional projetada para validar a segmentação gerada, garantindo a qualidade das previsões. Sua arquitetura e funcionamento são descritos, destacando os componentes responsáveis pela extração de características e pela classificação das máscaras geradas.

Em seguida, é detalhada a U-Net para segmentação, um modelo amplamente utilizado em aplicações médicas devido à sua capacidade de preservar detalhes espaciais enquanto realiza segmentações precisas. São discutidas as etapas de processamento dentro da U-Net, incluindo o caminho de contração, o bottleneck e o caminho de expansão, além da importância das conexões diretas entre camadas para recuperar informações essenciais.

Além das arquiteturas, são abordadas as estratégias utilizadas no treinamento dos modelos, incluindo técnicas de pré-processamento, configuração de hiperparâmetros, escolha de funções de perda e métodos de otimização. O objetivo é garantir que os modelos alcancem um desempenho robusto, evitando problemas como overfitting e garantindo a generalização para novos dados.

Por fim, são discutidos os critérios de avaliação adotados para medir a eficácia dos modelos, permitindo uma análise quantitativa e qualitativa dos resultados obtidos.

4.3.1 Arquitetura da CNN para Verificação de Máscaras

A verificação de máscaras na aplicação desenvolvida foi realizada por meio de uma Rede Neural Convolucional (CNN). Essa rede foi projetada para classificar máscaras de segmentação de imagens médicas em categorias específicas, identificando se a segmentação foi realizada corretamente ou apresenta erros.

A arquitetura da CNN foi definida para processar imagens de entrada de dimensões 32x32 pixels. A estrutura da rede inicia com duas camadas convolucionais: a primeira com um kernel de tamanho 3x3 e 16 filtros, e a segunda com um kernel 3x3 e 32 filtros, ambas utilizando ativação ReLU. Após cada camada convolucional, um operador de MaxPooling 2x2 é aplicado para reduzir as dimensões da imagem pela metade. Na sequência, a arquitetura possui duas camadas totalmente conectadas (*Fully Connected* - FC). A primeira é uma camada oculta com 128 neurônios e ativação ReLU, seguida pela camada de saída, que contém 2 neurônios correspondentes às classes possíveis e não possui função de ativação, para que a função de perda seja aplicada posteriormente.

A função de ativação utilizada nas camadas convolucionais e na camada totalmente conectada oculta é a ReLU (*Rectified Linear Unit*), garantindo uma modelagem eficiente de padrões nas imagens. A camada de saída utiliza ativação *softmax* para conversão em probabilidades.

O modelo foi treinado utilizando o otimizador Adam com taxa de aprendizado de

0.001 e função de perda *CrossEntropyLoss*. O treinamento ocorre em lotes de 32 imagens, utilizando os dados de treinamento e validação organizados em diretórios. Durante o treinamento, o modelo passa por 30 épocas. Em cada uma delas, a rede é treinada utilizando os dados do conjunto de treinamento. Após cada época, o desempenho é avaliado no conjunto de validação, registrando a acurácia e a função de perda. Ao final do treinamento, o modelo é salvo para futuras inferências.

Para verificar a qualidade de uma máscara segmentada, a imagem é pré-processada e enviada pela CNN treinada. O fluxo de inferência inicia com o redimensionamento da imagem para 32x32 pixels e sua conversão para o formato de tensor. Em seguida, a imagem é processada em um *forward pass* pelo modelo treinado, resultando em probabilidades para cada classe. Por fim, realiza-se a determinação da classe mais provável e o cálculo da confiança da predição.

O modelo classifica a imagem em dois grupos, "Boa"e "Ruim". Se a máscara for classificada como "Boa", significa que a segmentação atende aos critérios esperados. Caso contrário, a aplicação gera uma nova máscara ajustando o parâmetro de tolerância do *Flood Fill*.

4.3.2 Arquitetura da UNet para Segmentação

A U-Net implementada nesta ferramenta é uma rede neural convolucional especialmente desenhada para tarefas de segmentação de imagens médicas. O modelo foi configurado para trabalhar com imagens de entrada redimensionadas para 128×128 pixels, garantindo uniformidade e facilitando o processamento ao longo da rede. O fluxo de dados na U-Net pode ser dividido em cinco etapas: entrada, caminho de contração, bottleneck, caminho de expansão e saída.

Na entrada e redimensionamento, todas as imagens são padronizadas para 128×128 pixels. Esse tamanho foi escolhido para equilibrar resolução e eficiência computacional, garantindo que a rede consiga extrair detalhes sem comprometer o desempenho. Esse redimensionamento é realizado durante o pré-processamento, garantindo que todas as imagens tenham as mesmas dimensões antes de serem processadas pela rede. Essa uniformização é crucial para manter a consistência dos dados e facilitar o treinamento do modelo.

No encoder ou caminho de contração, a U-Net aplica uma série de blocos convolucionais. Cada bloco é composto por duas camadas de convolução seguidas de funções de ativação ReLU, que extraem características de diferentes níveis da imagem. Entre os blocos, camadas de pooling (MaxPooling 2×2) reduzem progressivamente a resolução, o que permite capturar

informações globais da imagem enquanto a profundidade dos canais aumenta. Essa etapa é fundamental para aprender representações hierárquicas e abstratas dos dados.

No Bottleneck, após o encoder, a rede atinge o bottleneck, que é a camada mais profunda e com maior número de filtros (por exemplo, 1024 filtros). Essa camada integra as informações extraídas do encoder em uma representação compacta e rica, servindo como um resumo dos recursos mais relevantes da imagem para a segmentação.

O decoder, ou caminho de expansão, é responsável por reconstruir a imagem segmentada a partir das representações extraídas pelo encoder. Utilizando camadas de convolução transposta (upsampling), a rede aumenta gradualmente a resolução espacial dos mapas de características. Além disso, as skip connections (conexões diretas entre camadas correspondentes do encoder e do decoder) permitem a transferência de detalhes finos, garantindo que informações importantes não sejam perdidas durante o processo de redução de dimensionalidade. Essa combinação resulta em uma máscara segmentada com alta precisão.

A camada final da U-Net é uma convolução 1×1 que reduz a profundidade para um único canal, representando a máscara segmentada. A função de ativação sigmoid é aplicada para convertê-los em probabilidades, determinando a presença ou ausência da região de interesse em cada pixel.

Essa arquitetura se destaca por sua capacidade de segmentar com alta precisão mesmo em situações com dados limitados, e sua flexibilidade permite que o modelo seja retreinado continuamente com novas imagens e máscaras, aprimorando a eficácia do diagnóstico assistido por computador.

4.4 Implementação e Integração das Tecnologias

4.4.1 Escolha das Tecnologias Utilizadas

A ferramenta CAD foi desenvolvida utilizando uma arquitetura modular composta por três camadas: Interface Web, API e Agentes de IA. A escolha dessas tecnologias foi fundamentada em critérios de flexibilidade, escalabilidade e facilidade de manutenção.

A Interface Web, desenvolvida em *Flutter*, possibilita o desenvolvimento multiplataforma e oferece uma experiência interativa ao usuário, permitindo o *upload* de imagens, ajustes de zoom, seleção de pontos e visualização dos resultados. A abordagem baseada em *widgets* facilita a criação de interfaces dinâmicas e responsivas.

A API em Flask foi escolhida por sua leveza e simplicidade, permitindo a rápida integração com os módulos de processamento e os agentes de IA. No backend, a API não apenas recebe os dados da interface, mas também executa módulos essenciais de pré-processamento, incluindo normalização, equalização e aplicação de filtros, e utiliza técnicas de *data augmentation* para enriquecer o banco de dados. Além disso, o *backend* implementa o algoritmo *Flood Fill* para a geração de máscaras a partir das coordenadas fornecidas pelo usuário.

Os Agentes de IA, implementados em PyTorch, englobam o modelo *UNet* para segmentação e uma CNN para classificação das máscaras. O modelo *UNet* é responsável por identificar as regiões de interesse nas imagens de MRI, enquanto a CNN valida a qualidade das máscaras geradas, garantindo que somente dados de alta qualidade sejam utilizados para o re-treinamento do modelo. A flexibilidade do PyTorch permite ajustar e re-treinar os modelos conforme a necessidade, beneficiando-se de sua integração com GPUs para acelerar o processamento.

Essa integração de tecnologias modernas permite que a ferramenta seja adaptativa e escalável, possibilitando tanto a utilização de modelos pré-treinados para segmentação automática quanto a reconfiguração contínua do sistema por meio do re-treinamento com novos dados. Dessa forma, a ferramenta não só melhora a precisão dos diagnósticos assistidos, mas também oferece um ambiente robusto para experimentação e evolução dos modelos de segmentação.

4.4.2 Ambiente de Desenvolvimento

Os experimentos e o desenvolvimento da ferramenta foram realizados em um computador com sistema operacional Windows 10, equipado com um processador AMD FX-8300 Eight-Core de 3.30 GHz, 12 GB de memória RAM e uma GPU GeForce RTX 3060 com 12 GB de memória dedicada. Essas configurações permitiram o processamento eficiente das imagens, o treinamento dos modelos de segmentação e classificação, e a execução da interface web em tempo real.

4.5 Interface de usuário

A interface web apresenta uma tela de autenticação de usuário, permitindo o acesso apenas a pessoas cadastradas. Após a validação das credenciais, o usuário é direcionado para a página de menu, onde pode visualizar as opções disponíveis. A Figura 18 ilustra tanto a tela de

login quanto a tela de menu.

A interface foi desenvolvida para atender a dois cenários distintos a segmentação automática de áreas de interesse, utilizando um modelo U-Net pré-treinado, e a geração de novos dados para o re-treinamento do modelo. No primeiro cenário, o fluxo segue pelo caminho de análise, enquanto no segundo, percorre o caminho de geração de dados.

Figura 18 – Página de login e menu de usuário.

(a) Página de Login

(b) Página de menu de usuário.

Fonte: Elaborado pelo autor (2025).

4.5.1 Segmentação Automática

No módulo de segmentação automática, o processo se divide em três etapas principais: upload da imagem, segmentação automática e visualização/interação com o resultado.

Na primeira etapa, o usuário seleciona uma imagem de ressonância magnética (MRI) do cérebro por meio da interface web da aplicação. A imagem original é exibida na interface e armazenada temporariamente, permitindo ao usuário visualizar o dado bruto antes da segmentação.

Em seguida, essa imagem é enviada para a API Flask, que repassa o conteúdo para o modelo U-Net previamente treinado. Esse modelo realiza a segmentação automática, destacando regiões de interesse, como estruturas anatômicas ou possíveis anomalias, com base nos padrões aprendidos durante o processo de treinamento.

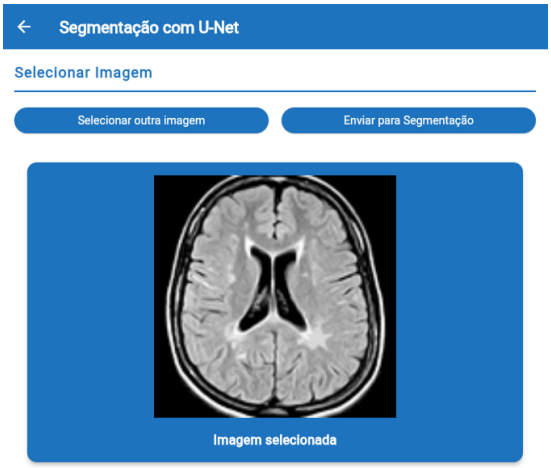
Por fim, o resultado da segmentação é retornado ao frontend e apresentado visualmente ao usuário. A área segmentada é sobreposta à imagem original, facilitando a análise visual.

Além disso, é disponibilizada a opção de realizar o download da imagem segmentada.

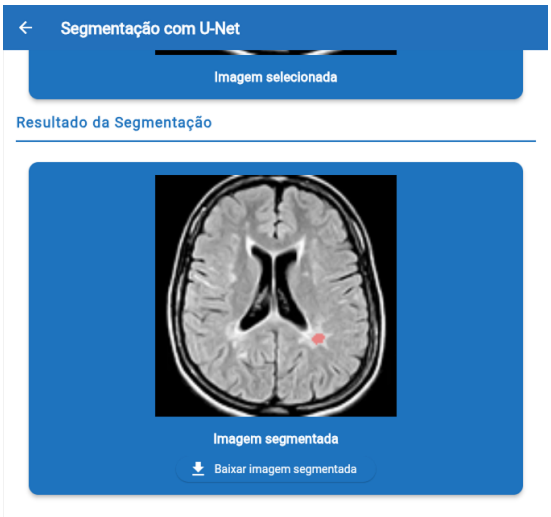
A Figura 19 ilustra a sequência da interface durante esse processo: inicialmente com a interface vazia (sem imagem selecionada), posteriormente com a imagem carregada e, por fim, com a segmentação realizada e o botão para download do resultado ativado.

Figura 19 – Interface da aplicação durante o processo de segmentação automática.

(a) Interface inicial



(b) Imagem carregada



(c) Resultado segmentado com opção de download

Fonte: Elaborado pelo autor (2025).

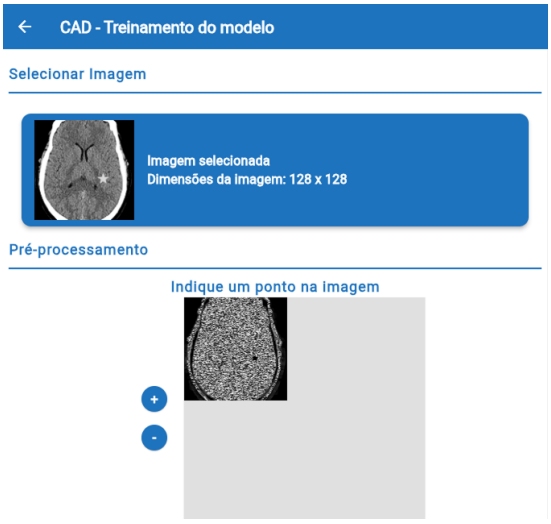
4.5.2 Treinamento do Modelo

No cenário de treinamento do modelo, o fluxo envolve as seguintes etapas: upload e pré-processamento da imagem, exibição e seleção do ponto, geração da máscara por Flood Fill, validação e armazenamento, e, por fim, o re-treinamento do modelo U-Net.

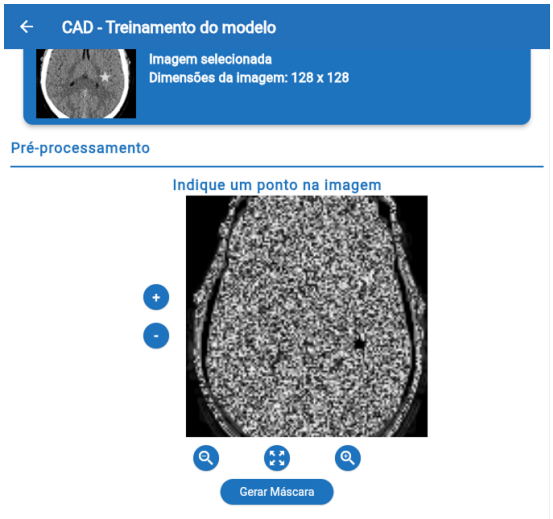
Na etapa de upload e pré-processamento, o usuário seleciona uma imagem de ressonância magnética (MRI) para análise. Em seguida, a imagem passa pelo fluxo de pré-processamento. Após o envio da imagem o usuário pode visualizar a imagem tratada e com a aplicação do filtro o que facilita a identificação da região de interesse e a seleção do ponto para a geração da máscara, otimizando o treinamento do modelo. A figura 20 ilustra a interface web onde o usuário realiza a operação de selecionar e indicar o ponto na imagem. Já a figura 21 ilustra a área de visualização da imagem pré-processada, destacando as funções que auxiliam o usuário, como ampliação da imagem e ajuste do tamanho do ponto de seleção.

Figura 20 – Fluxo da criação da base de dados e treinamento do modelo Unet.

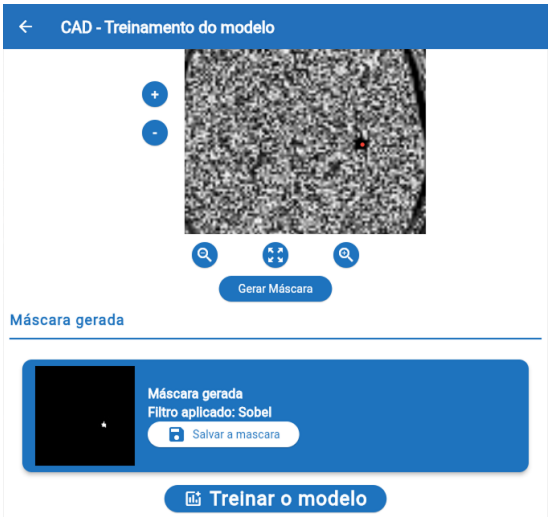
(a) Upload da imagem



(b) Visualização da imagem pré-processada



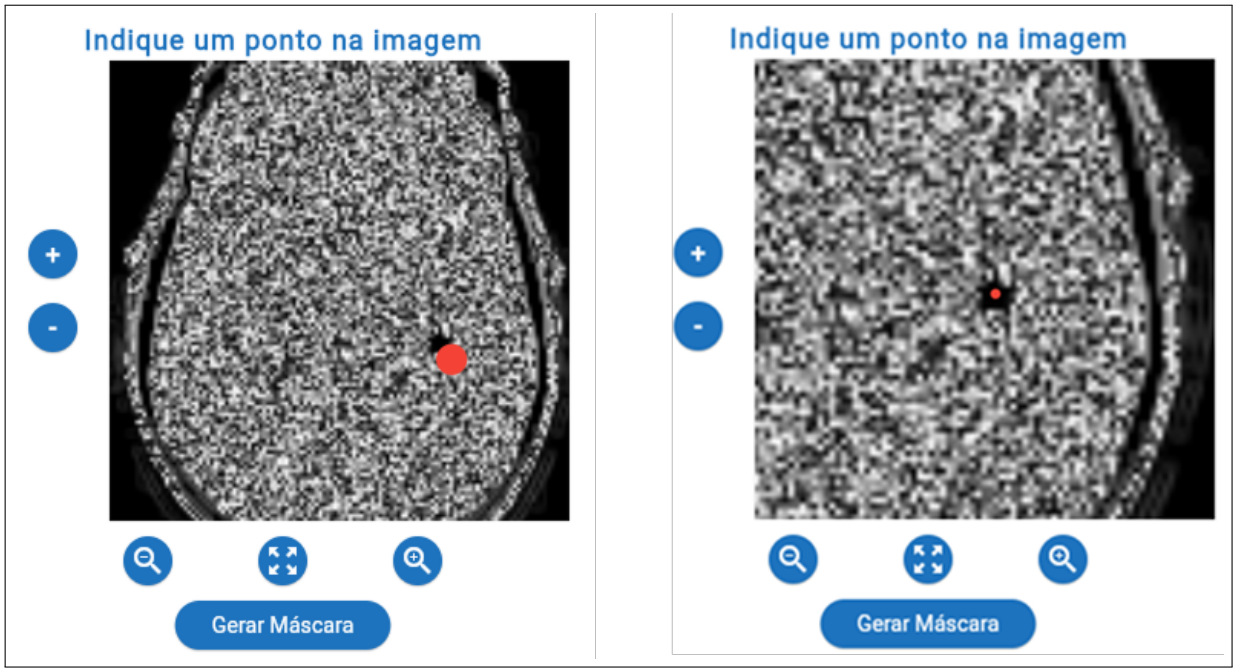
(c) Interação com a imagem pré-processamento



(d) Treinar o modelo

Fonte: Elaborado pelo autor (2025).

Figura 21 – Interação com a imagem pré-processamento.



Fonte: Elaborado pelo autor (2025).

5 RESULTADOS E AVALIAÇÕES

Este capítulo detalha os resultados obtidos e as avaliações realizadas para validar tanto a plataforma web desenvolvida quanto os modelos de inteligência artificial que a compõem. A análise foi estruturada em duas frentes principais: a primeira foca no desempenho técnico e na robustez da aplicação sob diferentes níveis de requisições, enquanto a segunda se concentra na performance e na precisão dos modelos de deep learning nas tarefas de classificação e segmentação de imagens. Inicialmente, serão apresentados os resultados dos testes de carga e estresse da API. Em seguida, serão discutidos os resultados quantitativos e qualitativos dos modelos CNN e U-Net, utilizando métricas como Dice, IoU e acurácia, complementados por uma análise visual das saídas geradas.

5.1 Testes de Desempenho

Para avaliar o desempenho da aplicação desenvolvida, foram conduzidos testes automatizados utilizando a ferramenta Gatling, amplamente empregada para simular o comportamento de usuários em ambientes web. Os testes implementados foram divididos em dois tipos principais: teste de carga e teste de estresse.

O teste de carga tem como objetivo analisar o comportamento da aplicação em situações típicas de uso simultâneo, verificando se o sistema é capaz de responder adequadamente a múltiplas requisições concorrentes sem perda significativa de desempenho. Já o teste de estresse busca explorar os limites da aplicação, submetendo-a a uma quantidade muito maior de requisições com o intuito de identificar o ponto em que começam a ocorrer falhas ou degradações de performance.

Ambos os testes foram conduzidos com a estratégia rampUsers, na qual os usuários virtuais são injetados gradualmente ao longo de um período de 30 segundos, permitindo simular um aumento progressivo da carga sobre o sistema.

Os testes de carga e estresse foram realizados utilizando uma infraestrutura de hardware composta por um processador AMD FX-8300 Eight-Core de 3.30 GHz , 12 GB de memória RAM , uma GPU GeForce RTX 3060 de 12 GB e o sistema operacional Windows 10.

A aplicação foi executada localmente, utilizando o ambiente de desenvolvimento do Flask, sem utilização de servidores de produção otimizados (como Gunicorn ou uWSGI), o que deve ser considerado na análise dos resultados.

5.1.1 Teste de Carga

No teste de carga, foram simulados 5 usuários para o endpoint de autenticação (`/api/login`), 3 usuários para a segmentação de imagens com o modelo U-Net (`/api/segmentar`), 3 para a geração de máscara (`/api/gerar-mascara`), 2 para o upload completo de imagens (/api/upload), 2 para o upload simples (`/api/upload_simples`) e 2 para o pré-processamento de imagens (`/api/pre-processar`).

A Figura 22, que representa a tabela de estatísticas gerais, apresenta indicadores como tempo mínimo, máximo e médio de resposta, desvio padrão e percentis (50º, 75º, 95º e 99º), além da taxa de requisições por segundo.

Figura 22 – Estatísticas gerais das requisições durante o teste de carga.

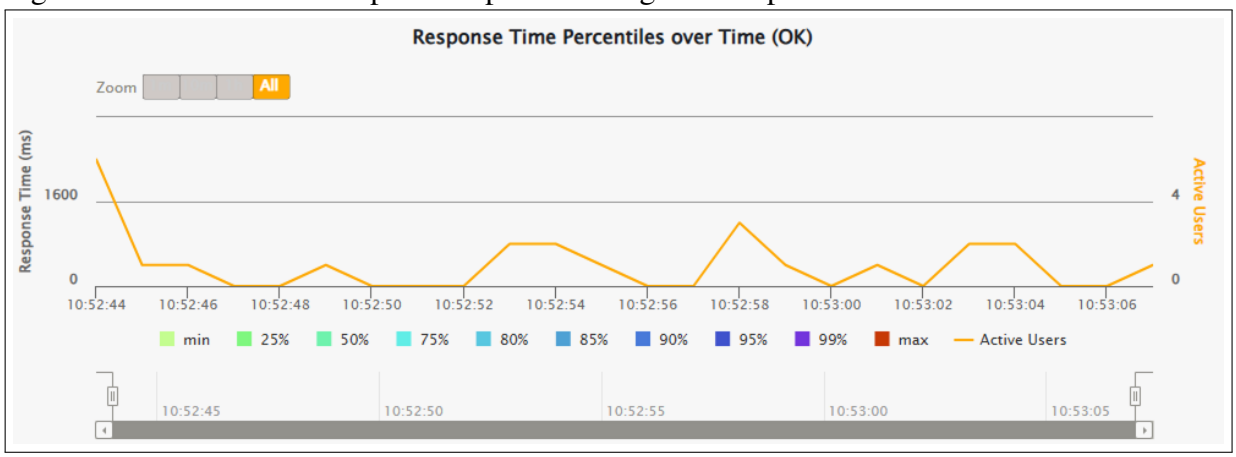
Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	17	17	0	0.00	0.71	7	94	244	1118	2384	2700	303	624
POST Upload Simples	2	2	0	0.00	0.08	26	59	76	89	91	92	59	33
POST Gerar Máscara	3	3	0	0.00	0.12	143	244	260	273	275	276	221	57
POST Autenticação	5	5	0	0.00	0.21	7	17	44	84	92	94	34	33
POST Segmentar com UNet	3	3	0	0.00	0.12	152	376	1538	2468	2654	2700	1076	1152
POST Pré-Processamento	2	2	0	0.00	0.08	34	64	79	91	93	94	64	30
POST Upload de Imagens	2	2	0	0.00	0.08	128	425	574	692	716	722	425	297

Fonte: Elaborado pelo autor (2025).

As rotas com menor complexidade de processamento, como `/api/pre-processar`, `/api/upload_simples` e `/api/login` apresentaram tempos médios de resposta inferiores a 300 ms e tempos máximos abaixo de 700 ms, indicando desempenho estável e eficiente mesmo sob carga simultânea.

A evolução dos percentis de tempo de resposta pode ser observada na Figura 23, correspondente ao gráfico de percentis de tempo de resposta ao longo do tempo. Nota-se que os percentis se mantiveram estáveis durante todo o período, com variações mínimas mesmo com o aumento gradativo de usuários.

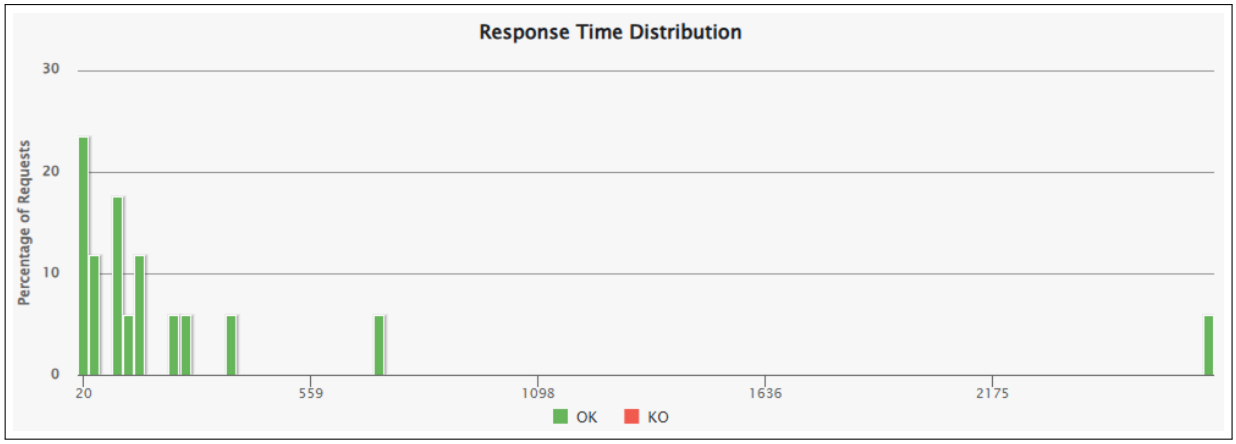
Figura 23 – Percentis de tempo de resposta ao longo do tempo.



Fonte: Elaborado pelo autor (2025).

Já a Figura 24, que mostra o gráfico de distribuição dos tempos de resposta, revela que mais de 90% das requisições foram respondidas em menos de 800 ms. Isso demonstra que a maioria das interações, mesmo em rotas mais exigentes como `/api/upload` e `/api/gerar-mascara`, ocorreram com latência aceitável, reforçando a robustez da aplicação.

Figura 24 – Distribuição dos tempos de resposta durante o teste.

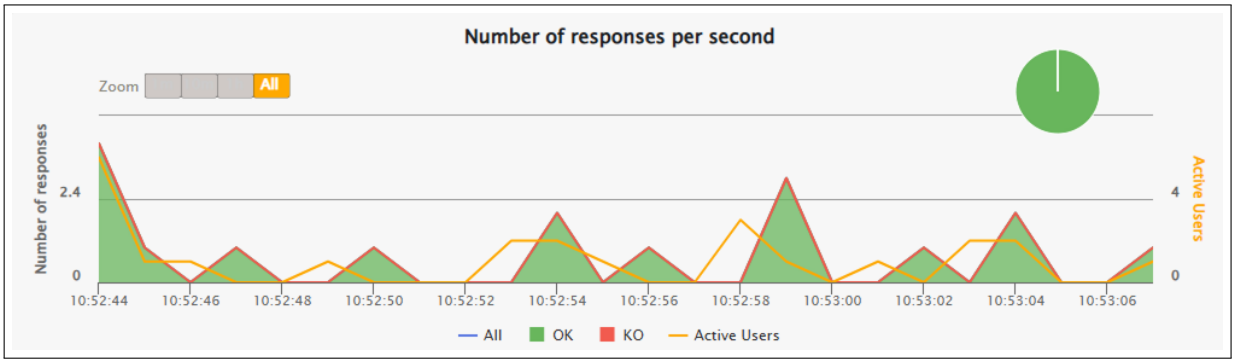


Fonte: Elaborado pelo autor (2025).

A taxa de sucesso das requisições foi de 100% em praticamente todas as rotas, sem ocorrência de erros. O gráfico indica o número de respostas, representado na Figura 25, confirma que a API manteve uma taxa constante de respostas ao longo do teste, sem interrupções ou congestionamentos durante a simulação.

Os resultados obtidos no teste de carga demonstram que a aplicação está apta para operar em ambientes reais, sendo capaz de processar múltiplas requisições simultâneas com baixa latência e comportamento estável, mesmo em rotas que envolvem maior volume de dados e operações computacionalmente intensivas.

Figura 25 – Número de respostas por segundo durante o teste.



Fonte: Elaborado pelo autor (2025).

5.1.2 Teste de Estresse

No teste de estresse, a simulação foi configurada com injeção progressiva de usuários virtuais ao longo de 30 segundos para cada rota. Foram simulados 50 usuários para autenticação (`/api/login`), 75 para segmentação com UNet (`/api/segmentar`), 75 para geração de máscara (`/api/gerar-mascara`), 40 para upload completo (`/api/upload`), 40 para pré-processamento de imagem (`/api/pre-processar`) e 40 para upload simples (`/api/upload_simples`).

Durante os testes, o Gatling gerou relatórios contendo métricas como número total de requisições, tempo médio de resposta, percentis (50º, 75º, 95º, 99º), tempo máximo e desvio padrão. Esses dados estão sintetizados na Figura 26, que apresenta a tabela de estatísticas globais do teste.

Figura 26 – Estatísticas gerais das requisições durante o teste de carga.

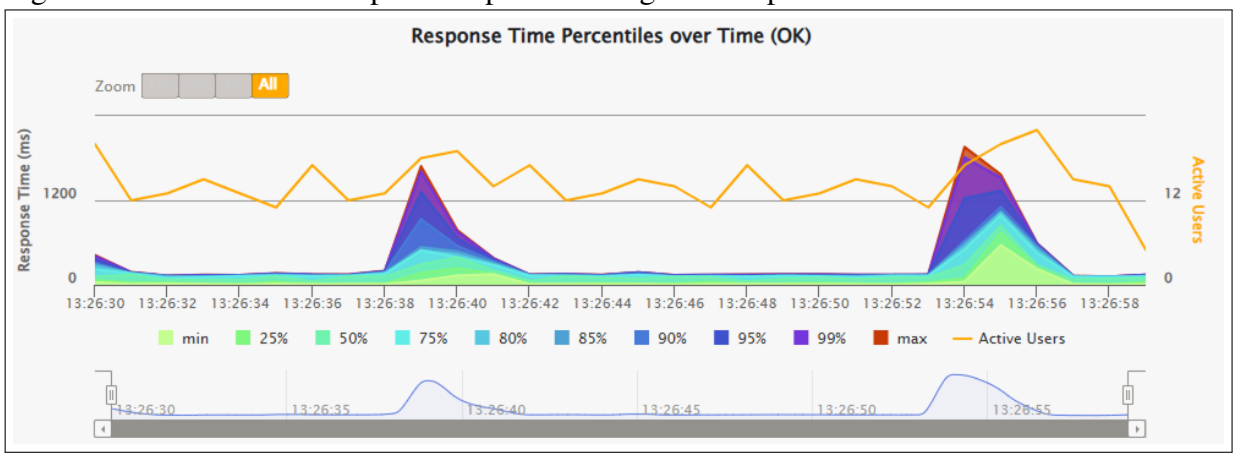
Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	320	319	1	0.31	10.67	5	114	146	566	1106	1959	157	234
POST Pré-processar Imagem	40	40	0	0.00	1.33	13	24	34	399	669	807	76	150
POST Upload Simples	40	40	0	0.00	1.33	7	19	33	302	608	635	70	136
POST Gerar Máscara	75	74	1	1.33	2.5	54	115	145	490	831	1019	170	164
POST Login	50	50	0	0.00	1.67	5	18	38	443	705	729	83	162
POST Segmentar com UNet	75	75	0	0.00	2.5	86	142	165	510	934	1057	207	179
POST Upload de Imagens	40	40	0	0.00	1.33	78	120	149	1574	1853	1959	301	456

Fonte: Elaborado pelo autor (2025).

A Figura 27, que corresponde ao gráfico de percentis de tempo de resposta ao longo do tempo, mostra a evolução da latência para as requisições bem-sucedidas. Observa-se que a maioria das rotas manteve estabilidade, com variações mais acentuadas em `/api/segmentar` e `/api/gerar-mascara`, especialmente após o pico de usuários.

A Figura 28, que apresenta o gráfico de distribuição dos tempos de resposta, evidencia

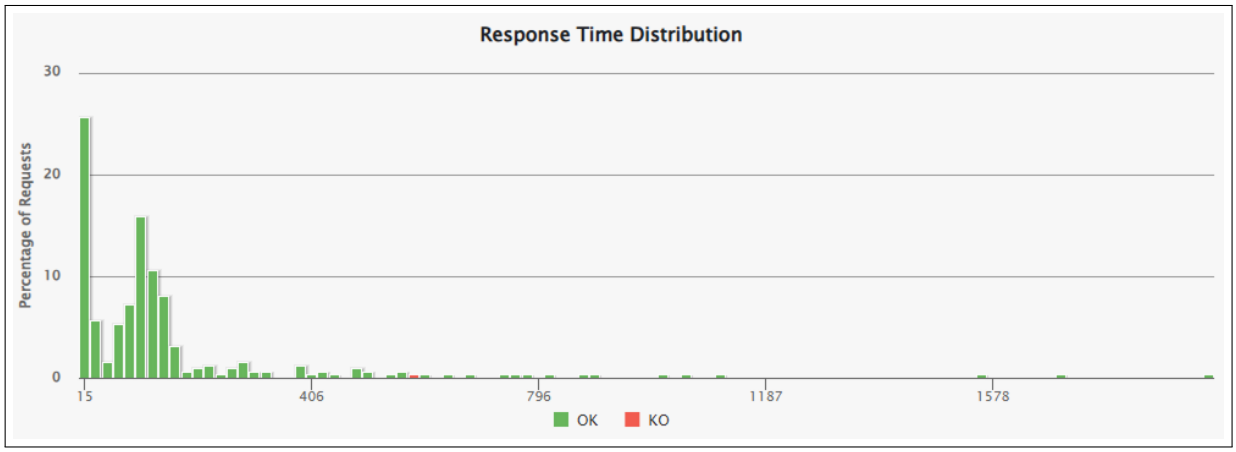
Figura 27 – Percentis de tempo de resposta ao longo do tempo.



Fonte: Elaborado pelo autor (2025).

que a maior parte das requisições foi respondida com tempo inferior a 800 ms em rotas menos exigentes, como `/api/login`, `/api/pre-processar` e `/api/upload_simples`. Já nas rotas mais pesadas, como `/api/segmentar` e `/api/gerar-mascara`, a distribuição se estendeu para faixas entre 800 ms e 1200 ms, com pequenas ocorrências acima desse intervalo, sugerindo aumento de latência, mas sem comprometer gravemente a estabilidade do sistema.

Figura 28 – Distribuição dos tempos de resposta durante o teste de estresse.



Fonte: Elaborado pelo autor (2025).

Apesar da carga elevada, a aplicação demonstrou resiliência: a taxa de sucesso das requisições (respostas “OK”) permaneceu acima de 98% na maior parte das rotas. Foram observados poucos erros HTTP 500, como ilustrado na figura 26 para a rota `/api/gerar-mascara`, indicando que o sistema manteve funcionalidade mesmo sob alto estresse.

Esses resultados indicam que a API consegue lidar com picos significativos de requisições simultâneas, sem degradação severa de desempenho. Contudo, visando garantir escalabilidade em ambientes de produção, recomenda-se a adoção de estratégias adicionais,

como balanceamento de carga, uso de cache em rotas críticas e otimizações no processamento assíncrono.

5.1.3 Análise Crítica dos Resultados

De maneira geral, os testes indicaram que a aplicação é capaz de lidar com múltiplas requisições simultâneas, mantendo tempos de resposta estáveis em grande parte das rotas testadas. A taxa de sucesso permaneceu acima de 98% mesmo sob carga intensa, demonstrando boa resiliência.

Contudo, ao analisar criticamente os resultados, é possível identificar alguns gargalos. O primeiro está relacionado às rotas de processamento intensivo, como `/api/segmentar` e `/api/gerar-mascara`, que apresentaram aumento significativo de latência durante o teste de estresse. Adicionalmente, a presença de erros HTTP 500, embora em pequena quantidade, sugere que o sistema atinge limites de processamento em cenários de alta demanda. Por fim, a própria infraestrutura de testes limitada, com um processador de arquitetura relativamente antiga e um ambiente de execução local sem otimizações de produção, também pode ter contribuído para o aumento da latência.

Portanto, para garantir a escalabilidade em ambientes de produção, são sugeridas diversas melhorias. Recomenda-se a otimização da infraestrutura, como a migração para servidores mais robustos e a utilização de servidores de aplicação como Gunicorn ou uWSGI em modo multiprocessado. Outras estratégias importantes incluem a implementação de cache em rotas críticas, o balanceamento de carga entre múltiplas instâncias da API e a otimização do processamento assíncrono para operações computacionalmente intensivas.

Essas melhorias visam não apenas reduzir a latência nas rotas mais exigentes, mas também aumentar a tolerância a picos de requisições, assegurando a estabilidade e a escalabilidade da aplicação em ambientes reais.

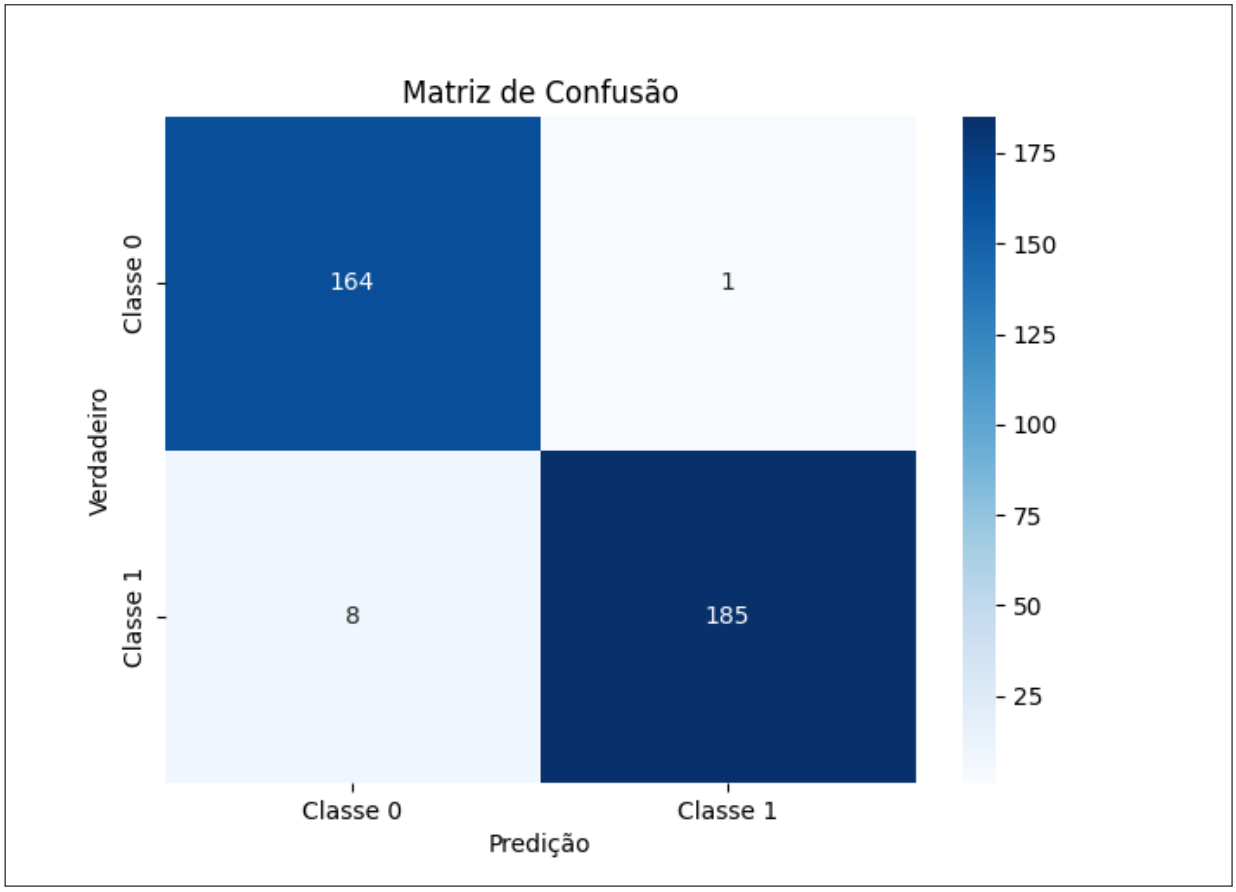
5.2 Desempenho dos Modelos de IA

5.2.1 Classificação com CNN

O modelo de classificação baseado em redes neurais convolucionais (CNN) foi avaliado utilizando um conjunto de teste balanceado, composto por imagens de máscaras geradas automaticamente. Essas imagens foram previamente rotuladas em duas categorias: "Boas"(classe

0), representando máscaras que se adequam corretamente às estruturas da imagem original, e "Ruins"(classe 1), representando máscaras com falhas de cobertura ou inconsistência em relação à região anatômica real. A matriz de confusão obtida está apresentada na figura 29, onde observa-se que o modelo classificou corretamente 164 imagens da classe 0 e 185 imagens da classe 1, cometendo apenas 9 erros de classificação no total.

Figura 29 – Matriz de confusão da CNN no conjunto de teste.



Fonte: Elaborado pelo autor (2025).

As principais métricas de desempenho do modelo, apresentadas na tabela 1, indicam uma acurácia de 97,49%, precisão de 99,46%, recall de 95,85% e F1-score de 97,63%. Esses resultados demonstram que a CNN foi altamente eficaz tanto na identificação correta das imagens classificadas como "Ruins"quanto na redução de falsos positivos, evidenciando a robustez do modelo no processo de avaliação das máscaras geradas automaticamente.

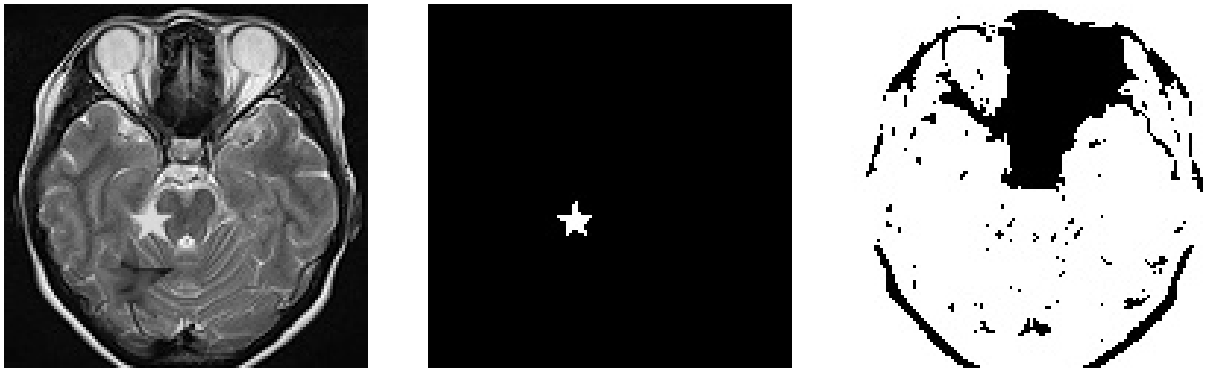
A leve diferença entre precisão e recall revela uma tendência do modelo a ser ligeiramente mais conservador, priorizando a exatidão nas previsões positivas (classe 1), ainda que à custa de algumas classificações incorretas da classe 1 como classe 0. No entanto, o equilíbrio geral entre as métricas demonstra a robustez da CNN aplicada ao problema.

Tabela 1 – Métricas de desempenho do modelo CNN

Métrica	**Valor**
Acurácia	97,49%
Precisão	99,46%
Recall	95,85%
F1-Score	97,63%

A Figura 30a apresenta um exemplo de imagem original utilizada no processo de avaliação. Para ilustrar como máscaras de qualidades distintas podem ser geradas para a mesma imagem, as Figuras 30c e 30b demonstram o impacto da variação dos parâmetros de tolerância (*loDiff* e *upDiff*) do algoritmo *Flood Fill*. Uma máscara considerada "Ruim"(Figura 30c) pode ser o resultado de parâmetros que levam a um preenchimento excessivo ou insuficiente da região de interesse. Em contrapartida, ao ajustar esses parâmetros, o sistema consegue gerar uma máscara "Boa"(Figura 30b), com melhor aderência à anatomia real. Esse comparativo visualiza os critérios que o modelo CNN aprende a diferenciar para validar a qualidade das máscaras que serão usadas no treinamento.

Figura 30 – Exemplos de imagens utilizadas na avaliação das máscaras segmentadas.



(a) Imagem original enviada (b) Classificada como "Boa" (c) Classificada como "Ruim"

Fonte: Elaborado pelo autor (2025).

5.2.2 Segmentação com U-Net

O desempenho da arquitetura U-Net foi avaliado por meio de um conjunto de teste composto por 414 imagens de ressonância magnética do cérebro e suas respectivas máscaras binárias de referência. A avaliação foi realizada com base na comparação direta entre as máscaras geradas automaticamente pelo modelo e as máscaras reais, utilizando três métricas amplamente adotadas em tarefas de segmentação: Dice Score, IoU (Intersection over Union) e acurácia por pixel. Os valores médios obtidos por essas métricas estão apresentados na tabela 2.

Tabela 2 – Métricas médias obtidas pelo modelo U-Net no conjunto de teste.

Métrica	**Valor Médio**
Dice Score	0,9022
IoU	0,8396
Acurácia Pixel	0,9994

Fonte: Elaborado pelo autor (2025).

Com o objetivo de reforçar a robustez estatística dos resultados, foram também calculados os intervalos de confiança de 95% para cada métrica, utilizando a distribuição normal padrão. Esses intervalos fornecem uma estimativa da faixa na qual se espera que esteja o valor real da métrica para a população geral de imagens, com 95% de confiança. Os valores obtidos estão apresentados na Tabela 3.

Tabela 3 – Intervalos de confiança de 95% para as métricas da U-Net.

Métrica	**IC 95%**
Dice Score	[0,8909 ; 0,9135]
IoU	[0,8234 ; 0,8558]
Acurácia Pixel	[0,9993 ; 0,9995]

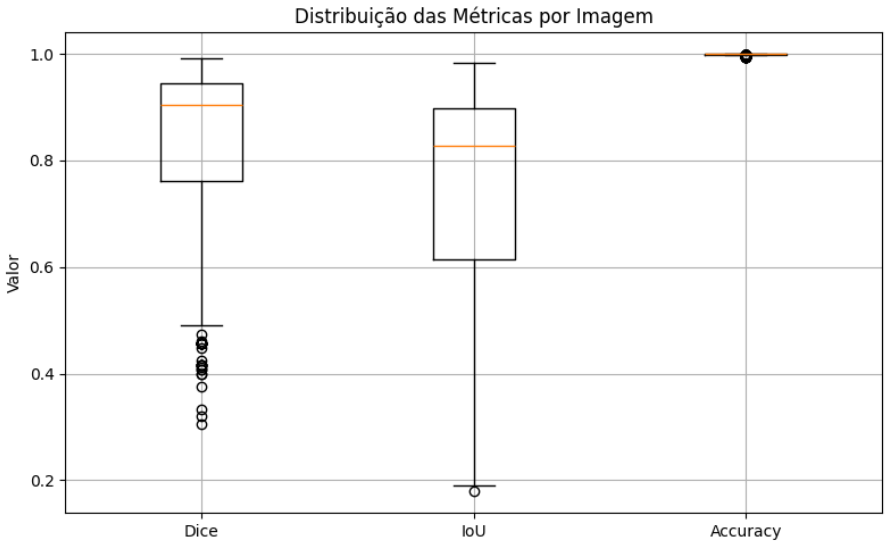
Fonte: Elaborado pelo autor (2025).

Esses valores indicam uma excelente sobreposição entre as regiões segmentadas automaticamente e as regiões reais de interesse, além de um elevado índice de precisão pixel a pixel. A figura 31 apresenta a distribuição dessas métricas por imagem, permitindo uma análise mais robusta da variabilidade nos resultados. A utilização do boxplot evidencia a dispersão, os quartis e potenciais outliers, revelando que, embora a maioria das amostras tenha apresentado resultados elevados, houve maior variabilidade nas métricas Dice e IoU. Isso sugere que o modelo enfrentou dificuldades em alguns casos específicos, possivelmente relacionados à complexidade anatômica ou à qualidade das imagens analisadas.

É importante notar a diferença de variabilidade entre as métricas apresentadas na Figura 31. A acurácia por pixel se mantém consistentemente alta e com variação mínima devido ao desbalanceamento de classes inerente à tarefa de segmentação. Como as regiões de interesse, anomalias simuladas, ocupam uma área muito pequena em relação ao fundo da imagem, o número de Verdadeiros Negativos, pixels de fundo corretamente classificados, é extremamente elevado e domina o cálculo da acurácia, tornando-a menos sensível a pequenos erros na delimitação da anomalia.

Em contrapartida, as métricas Dice e IoU são mais adequadas para avaliar a segmentação em cenários desbalanceados, pois ignoram os Verdadeiros Negativos e focam exclusivamente

Figura 31 – Boxplot das métricas Dice, IoU e Acurácia por pixel.



Fonte: Elaborado pelo autor (2025).

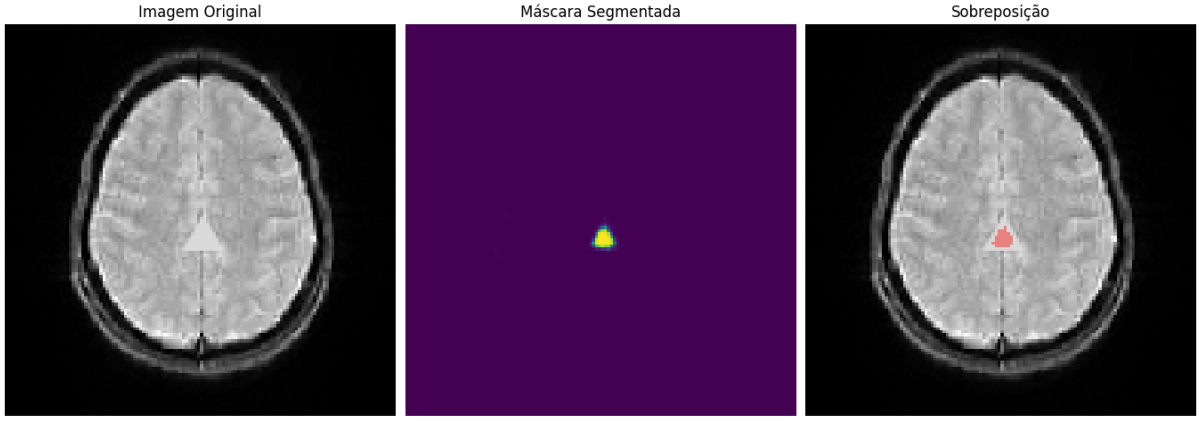
na qualidade da sobreposição entre a área prevista e a área real da região de interesse. Por isso, elas são mais sensíveis a erros de segmentação e refletem com maior precisão a variabilidade no desempenho do modelo, justificando a maior dispersão observada em seus resultados.

Complementando a avaliação quantitativa, foi realizada uma **análise qualitativa** dos resultados de segmentação. A Figura 32 apresenta dois exemplos representativos. Na Figura 32a, observa-se uma imagem com *uma única região de interesse*, segmentada de forma precisa e com boa correspondência morfológica. Já na Figura 32b, o modelo demonstrou competência em lidar com *múltiplas regiões simultâneas*, segmentando três estruturas distintas com precisão, mesmo em um cenário visualmente mais complexo. Essas evidências visuais reforçam a capacidade do modelo em generalizar para diferentes padrões estruturais, contribuindo para sua aplicabilidade em contextos reais de apoio ao diagnóstico médico.

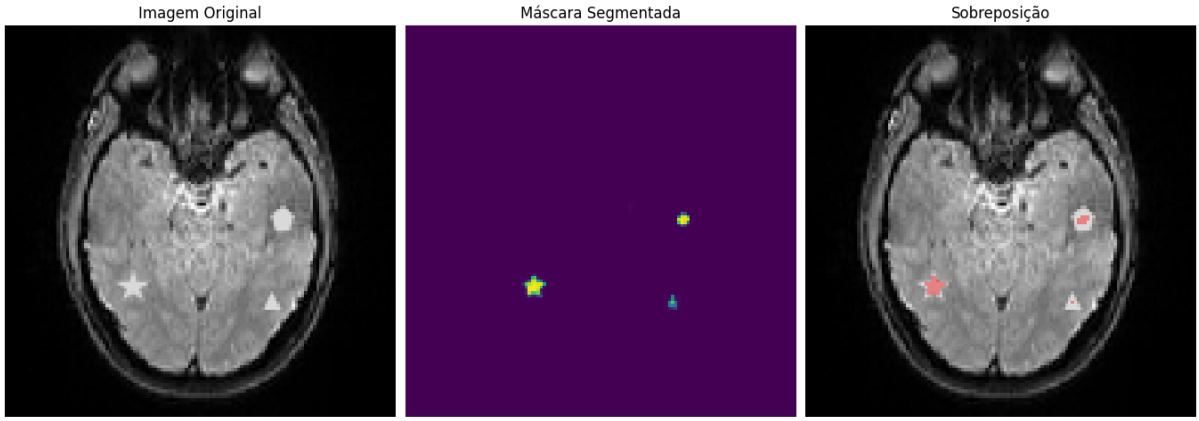
5.3 Limitações do Estudo

Embora os resultados obtidos demonstrem o potencial da plataforma desenvolvida, é importante destacar que o sistema ainda se encontra em estágio experimental. O uso do algoritmo Flood Fill para a geração de máscaras, apesar de funcional, apresenta limitações relevantes quanto à precisão e à generalização das segmentações, especialmente pela dependência de intervenção manual e pela ausência de validação com especialistas da área médica. Além disso, a geração artificial de anomalias com formas geométricas configura uma simplificação da realidade clínica e deverá ser substituída, em trabalhos futuros, pelo uso de conjuntos de dados reais e rotulados.

Figura 32 – Visualização qualitativa de imagens segmentadas com o modelo UNet.



(a) Imagem com uma região de interesse.



(b) Imagem com três regiões de interesse.

Fonte: Elaborado pelo autor (2025).

Outro aspecto crítico é que questões fundamentais, como a privacidade dos dados médicos e a conformidade com a Lei Geral de Proteção de Dados (LGPD), bem como estratégias de segurança da informação, não foram abordadas nesta primeira versão, sendo indispensáveis para a aplicação prática em ambientes hospitalares. Assim, o presente trabalho deve ser compreendido como uma prova de conceito, com resultados promissores, mas ainda distante de uma aplicação direta no contexto clínico.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento de uma plataforma web interativa para segmentação de neuroimagens por ressonância magnética, integrando técnicas de pré-processamento de imagens, algoritmos de segmentação automática com U-Net e mecanismos de validação baseados em redes neurais convolucionais (CNN). A solução proposta demonstrou ser eficaz tanto do ponto de vista técnico quanto prático, oferecendo uma interface acessível, intuitiva e com potencial de aplicação clínica.

A partir dos testes realizados, observou-se que o modelo U-Net atingiu métricas de desempenho elevadas, com Dice Score médio de 90,22%, IoU de 83,96% e acurácia pixel a pixel de 99,94%. Esses resultados indicam uma sobreposição satisfatória entre as máscaras segmentadas automaticamente e as referências manuais, evidenciando a capacidade do modelo em identificar regiões anatômicas relevantes mesmo em cenários visuais complexos. A análise qualitativa reforçou esses achados ao demonstrar a robustez da plataforma frente a diferentes padrões estruturais presentes nas imagens.

No que se refere ao backend da aplicação, os testes de carga e estresse mostraram que a API foi capaz de lidar eficientemente com múltiplas requisições simultâneas, mantendo uma taxa de sucesso superior a 98% e tempos de resposta adequados mesmo em rotas computacionalmente intensivas, como as de segmentação e geração de máscara. Tais resultados evidenciam a escalabilidade e a estabilidade da solução desenvolvida, aspectos essenciais para sua adoção em ambientes clínicos reais.

Para o aprimoramento da plataforma, propõe-se como trabalhos futuros:

- Melhorar a etapa de geração de máscaras, substituindo o algoritmo Flood Fill pelo método Watershed, que oferece maior precisão na segmentação de regiões complexas;
- Implementar mecanismos de conformidade com a LGPD, garantindo a privacidade e a proteção dos dados médicos armazenados e processados;
- Substituir a geração artificial de anomalias pela utilização de bases de dados reais e rotuladas, promovendo maior realismo e validade clínica nas avaliações;
- Implementar uma fila de tarefas para o reprocessamento assíncrono do modelo U-Net, otimizando o uso de recursos computacionais;
- Desenvolver uma tela dedicada à visualização e ao gerenciamento dos dados armazenados no repositório, facilitando o controle e a operação pelo usuário;
- Exibir dinamicamente a quantidade de dados presentes no repositório de imagens, permi-

tindo o acompanhamento da evolução da base de dados;

- Implementar o controle de versões dos modelos U-Net treinados, promovendo maior segurança e rastreabilidade dos resultados;
- Adicionar suporte à visualização de imagens no formato NIfTI (.nii), possibilitando a análise de exames volumétricos e alinhando a plataforma aos padrões mais utilizados em aplicações clínicas de neuroimagem;

Em síntese, a plataforma desenvolvida representa um avanço inicial na interface entre inteligência artificial e diagnóstico por imagem, oferecendo uma base sólida para evoluções futuras que visem à sua aplicação prática em contextos hospitalares.

REFERÊNCIAS

BALLESTER, M. A. G.; ZISSERMAN, A.; BRADY, M. Segmentation and measurement of brain structures in mri including confidence bounds. **Medical Image Analysis**, Amsterdã, v. 4, n. 3, p. 189–200, 2000.

BHATT, D.; PATEL, C.; TALSANIA, H.; PATEL, J.; VAGHELA, R.; PANDYA, S.; MODI, K.; GHAYVAT, H. Cnn variants for computer vision: History, architecture, application, challenges and future scope. **Electronics**, Basileia, v. 10, n. 20, p. 2470, 2021.

BONNEY, M. S.; ANGELIS, M. D.; BORGO, M. D.; ANDRADE, L.; BEREGI, S.; JAMIA, N.; WAGG, D. J. Development of a digital twin operational platform using python flask. **Data-Centric Engineering**, Cambridge, v. 3, p. e1, 2022.

BOUKHARY, S.; COLMENARES, E. A clean approach to flutter development through the flutter clean architecture package. In: **Anais da 2019 international conference on computational science and computational intelligence (CSCI)**. Las Vegas: IEEE, 2019. p. 1115–1120.

CRISTOFALO, R. M. **Segmentação e especialização de modelos de machine learning para a predição de mortalidade neonatal**. Monografia (Trabalho de Conclusão de Curso) — Universidade de São Paulo, São Paulo, 2024.

DOI, K. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. **Computerized medical imaging and graphics**, Amsterdã, v. 31, n. 4-5, p. 198–211, 2007.

EADIE, L. H.; TAYLOR, P.; GIBSON, A. P. A systematic review of computer-assisted diagnosis in diagnostic cancer imaging. **European journal of radiology**, Amsterdã, v. 81, n. 1, p. e70–e76, 2012.

EELBODE, T.; BERTELS, J.; BERMAN, M.; VANDERMEULEN, D.; MAES, F.; BISSCHOPS, R.; BLASCHKO, M. B. Optimization for medical image segmentation: theory and practice when evaluating with dice score or jaccard index. **IEEE transactions on medical imaging**, Nova York, v. 39, n. 11, p. 3679–3690, 2020.

FENTAW, A. E. **Cross platform mobile application development: a comparison study of React Native Vs Flutter**. Monografia (Trabalho de Conclusão de Curso) — Metropolia University of Applied Sciences, Helsinque, 2020.

GAO, J.; JIANG, Q.; ZHOU, B.; CHEN, D. Convolutional neural networks for computer-aided detection or diagnosis in medical image analysis: An overview. **Mathematical Biosciences and Engineering**, [S. l.], v. 16, n. 6, p. 6536–6561, 2019.

GHIMIRE, D. **Comparative study on Python web frameworks: Flask and Django**. Monografia (Trabalho de Conclusão de Curso) — Arcada University of Applied Sciences, Helsinque, 2020.

GOEL, N.; YADAV, A.; SINGH, B. M. Medical image processing: A review. In: **Anais do 2016 Second International Innovative Applications of Computational Intelligence on Power, Energy and Controls with their Impact on Humanity (CIPECH)**. [S. l.]: IEEE, 2016. p. 57–62.

HE, Y.; HU, T.; ZENG, D. Scan-flood fill (scaff): An efficient automatic precise region filling algorithm for complicated regions. In: **Anais do Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops**. Long Beach: IEEE, 2019. p. 0–0.

KAUSHAL, C.; BHAT, S.; KOUNDAL, D.; SINGLA, A. Recent trends in computer assisted diagnosis (cad) system for breast cancer diagnosis using histopathological images. **Irbm**, Amsterdã, v. 40, n. 4, p. 211–227, 2019.

KHAN, S. M.; NABI, A. ul; BHANBHRO, T. H. Comparative analysis between flutter and react native. **International Journal of Artificial Intelligence & Mathematical Sciences**, [S. l.], v. 1, n. 1, p. 16–29, 2022.

MCDONALD, R. J.; SCHWARTZ, K. M.; ECKEL, L. J.; DIEHN, F. E.; HUNT, C. H.; BARTHOLMAI, B. J.; ERICKSON, B. J.; KALLMES, D. F. The effects of changes in utilization and technological advancements of cross-sectional imaging on radiologist workload. **Academic radiology**, Amsterdã, v. 22, n. 9, p. 1191–1198, 2015.

MOHAMMED, F. A.; TUNE, K. K.; ASSEFA, B. G.; JETT, M.; MUHIE, S. Medical image classifications using convolutional neural networks: a survey of current methods and statistical modeling of the literature. **Machine learning and knowledge extraction**, Basileia, v. 6, n. 1, p. 699–735, 2024.

MUFID, M. R.; BASOFI, A.; RASYID, M. U. H. A.; ROCHIMANSYAH, I. F. *et al.* Design an mvc model using python for flask framework development. In: **Anais do 2019 International Electronics Symposium (IES)**. Surabaya: IEEE, 2019. p. 214–219.

MÜLLER, E. **Web technologies on the desktop: an early look at Flutter**. Monografia (Trabalho de Conclusão de Curso (Bacharelado)) — Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, 2021.

OLIVEIRA, K. R. S. **Segmentação pulmonar em imagens de radiografias de tórax utilizando redes neurais convolucionais**. Dissertação (Dissertação (Mestrado)) — Universidade Federal de Uberlândia, Uberlândia, 2023.

PACHECO, A. **Introdução a Redes Neurais Artificiais**. [S. l.]: [s.n.], 2015. Disponível em: http://computacaointeligente.com.br/artigos/redes-neurais-artificiais/. Acesso em: 30 nov. 2020.

PEREIRA, C. R.; PEREIRA, D. R.; WEBER, S. A.; HOOK, C.; ALBUQUERQUE, V. H. C. D.; PAPA, J. P. A survey on computer-assisted parkinson's disease diagnosis. **Artificial intelligence in medicine**, Amsterdã, v. 95, p. 48–63, 2019.

PERUMAL, S.; VELMURUGAN, T. Preprocessing by contrast enhancement techniques for medical images. **International Journal of Pure and Applied Mathematics**, [S. l.], v. 118, n. 18, p. 3681–3688, 2018.

PISKOR, J.; BADUROWICZ, M. Performance comparison of flutter platform gui in web and native environments. **Journal of Computer Sciences Institute**, [S. l.], v. 28, p. 217–222, 2023.

REDFERN, R.; HORII, S.; FEINGOLD, E.; KUNDEL, H. Radiology workflow and patient volume: effect of picture archiving and communication systems on technologists and radiologists. **Journal of Digital Imaging**, Berlim, v. 13, n. Suppl 1, p. 97–100, 2000.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: **Anais do Medical image computing and computer-assisted intervention–MICCAI 2015**. Munique: Springer, 2015. p. 234–241.

SANTOS, M. K.; JÚNIOR, J. R. F.; WADA, D. T.; TENÓRIO, A. P. M.; NOGUEIRA-BARBOSA, M. H.; MARQUES, P. M. d. A. Inteligência artificial, aprendizado de máquina, diagnóstico auxiliado por computador e radiômica: avanços da imagem rumo à medicina de precisão. **Radiologia brasileira**, São Paulo, v. 52, p. 387–396, 2019.

SATHANANTHAVATHI, V.; INDUMATHI, G. Encoder enhanced atrous (eea) unet architecture for retinal blood vessel segmentation. **Cognitive Systems Research**, Amsterdã, v. 67, p. 84–95, 2021.

SILVA, G. F. L. *et al.* **Avaliação Comparativa de Modelos de Segmentação Semântica de Nuvens em Imagens de Satélite**. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal da Paraíba, João Pessoa, 2024.

SINGH, M.; VERMA, A.; PARASHER, A.; CHAUHAN, N.; BUDHIRAJA, G. Implementation of database using python flask framework. **International Journal of Engineering and Computer Science**, [S. l.], v. 8, n. 12, p. 24890–24893, 2019.

SOARES, V. **"Nobel da Computação" vai para os pais do Deep Learning.** [S. l.]: [s.n.], 2019. Disponível em: https://www.institutodeengenharia.org.br/site/2019/04/01/nobel-da-computacao-vai-para-os-pais-do-deep-learning/. Acesso em: 16 abr. 2020.

SRI, S. D.; RAMAN, R. C.; RAJAGOPAL, G.; CHAN, S. T. *et al.* Automating rest api postman test cases using llm. **arXiv preprint**, [S. l.], 2024. Disponível em: https://arxiv.org/abs/2404.10678. Acesso em: 25 fev. 2025.

TRAN, T. **Flutter native performance and expressive ui/ux**. Monografia (Trabalho de Conclusão de Curso) — Centria University of Applied Sciences, Kokkola, 2020.

VASA, V. K.; ZHU, W.; CHEN, X.; QIU, P.; DONG, X.; WANG, Y. Sta-unet: Rethink the semantic redundant for medical imaging segmentation. **arXiv preprint**, [S. l.], 2024. Disponível em: https://arxiv.org/abs/2410.11578. Acesso em: 25 fev. 2025.

VASUKI, P.; KANIMOZHI, J.; DEVI, M. B. A survey on image preprocessing techniques for diverse fields of medical imagery. In: **Anais da 2017 IEEE international conference on electrical, instrumentation and communication engineering (ICEICE)**. Karur: IEEE, 2017. p. 1–6.

VYSHNAVI, V. R.; MALIK, A. Efficient way of web development using python and flask. **Int. J. Recent Res. Asp**, [S. l.], v. 6, n. 2, p. 16–19, 2019.

WANG, J.; PEREZ, L. *et al.* The effectiveness of data augmentation in image classification using deep learning. **Convolutional Neural Networks Vis. Recognit**, [S. l.], v. 11, n. 2017, p. 1–8, 2017.

WANG, S.; ZHAO, Z.; OUYANG, X.; WANG, Q.; SHEN, D. Chatcad: Interactive computer-aided diagnosis on medical image using large language models. **arXiv preprint**, [S. l.], 2023. Disponível em: https://arxiv.org/abs/2302.07257. Acesso em: 25 fev. 2025.

YAGANTEESWARUDU, A. Multi disease prediction model by using machine learning and flask api. In: **Anais da 2020 5th International conference on communication and electronics systems (ICCES)**. Coimbatore: IEEE, 2020. p. 1242–1246.

YANG, R.; YU, Y. Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. **Frontiers in oncology**, [S. l.], v. 11, p. 638182, 2021.

YU, L.; LI, Z.; XU, M.; GAO, Y.; LUO, J.; ZHANG, J. Distribution-aware margin calibration for semantic segmentation in images. **International Journal of Computer Vision**, Berlim, v. 130, p. 95–110, 2022.

ZHU, Z.; YAN, Y.; XU, R.; ZI, Y.; WANG, J. Attention-unet: A deep learning approach for fast and accurate segmentation in medical imaging. **Journal of Computer Science and Software Applications**, [S. l.], v. 2, n. 4, p. 24–31, 2022.