UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

IMPLEMENTAÇÃO DE UM PIPELINE PARA PARCELAMENTO CEREBRAL EM IMAGENS DE RESSONÂNCIA MAGNÉTICA UTILIZANDO O FRAMEWORK MONAI

Autor: Pedro Malandrin Klesse
Orientador: Ricardo José Ferrari
São Carlos
2025

Resumo

Neste projeto, foi implementada uma solução de parcelamento cerebral (*brain parcellation*) utilizando o *framework* MONAI. A abordagem desenvolvida abrange todas as etapas do *pipeline*, desde o pré-processamento das imagens médicas até o treinamento, validação e teste de duas arquiteuras distintas de redes neurais. O objetivo é segmentar e rotular diferentes regiões do cérebro a partir de imagens de ressonância magnética ponderadas em T1.

Para garantir a padronização das imagens antes do treinamento, foram aplicadas diversas técnicas de pré-processamento, incluindo correção de *bias*, normalização de intensidade, reorientação para o sistema RAS e centralização das estruturas cerebrais. Essas etapas asseguram que os dados de entrada estejam em um formato consistente, melhorando a generalização do modelo.

A rede escolhida para a segmentação foi a HighResNet, selecionada com base em um artigo que demonstrou sua eficácia para a tarefa de *brain parcellation*. O treinamento foi realizado utilizando um conjunto de 1200 imagens rotuladas, enquanto a validação e o teste foram conduzidos em 200 e 200 imagens, respectivamente, comparando tal arquitetura com a U-Net, uma das principais redes utilizadas para segmentação de imagens 2D e 3D. O desempenho da rede foi avaliado por meio do coeficiente de Dice, permitindo medir a sobreposição entre as segmentações preditas e os rótulos reais.

Após o treinamento, os pesos do modelo foram salvos, e, para facilitar a utilização da solução, foi desenvolvida uma aplicação em FastAPI integrada ao uvicorn. Essa aplicação permite o *deploy* da redes, permitindo o uso de ambas como um serviço. A API possibilita o envio de imagens por meio de requisições POST, retornando como resposta a imagem volumétrica devidamente parcelada. Dessa forma, os modelos podem ser acessados e utilizados de forma prática e eficiente.

Sumário

Capítulo 1

Introdução

1.1 Contextualização

Segmentação de imagens é um problema clássico e desafiador no campo da visão computacional, caracterizado pela dificuldade de determinar com precisão quais pixels ou voxels pertencem a um objeto em uma imagem [1]. Dentro da área médica, a segmentação pode ser utilizada para auxiliar profissionais da saúde a identificar enfermidades por meio do reconhecimento de padrões [1]. Atualmente, essa segmentação pode ser realizada com o auxílio de redes neurais convolucionais (CNNs) [2], consideradas a solução estado da arte para esse problema.

No entanto, apesar do seu avanço, ainda há desafios na construção de arquiteturas de CNNs capazes de realizar segmentação com alta acurácia em imagens médicas volumétricas, tanto pela necessidade de grandes conjuntos de dados de treinamento quanto pela alta complexidade inerente a essas imagens. Para superar essas dificuldades, são utilizadas algumas técnicas, como o data augmentation [3] (aumento do conjunto de treinamento a partir das imagens já obtidas) e métodos de pré-processamento, que modificam as imagens para otimizar seu aproveitamento durante o treinamento, resultando em maior acurácia dos modelos. Exemplos dessas técnicas incluem normalização, remoção de ruído, reescalonamento, entre outras [4].

Um dos desafios na segmentação de imagens médicas é o *brain parcellation* [5][6], que consiste em dividir uma imagem do cérebro humano em estruturas anatômicas definidas pela nomenclatura médica. Devido à alta complexidade em distinguir diferentes regiões cerebrais — já que o cérebro não apresenta diferenças visuais claras entre muitas dessas regiões —, a técnica de brain parcellation é considerada uma tarefa altamente desafiadora para a visão computacional moderna."

Antes das CNNs se tornarem viáveis computacionalmente, o processo de *brain parcellation* era, e ainda é em muitos casos, realizado principalmente utilizando um atlas — uma imagem volumétrica de um cérebro rotulada manualmente que orienta a divisão das regiões cerebrais. O processo de segmentação utilizando um atlas consiste, basicamente, em ajustar a imagem alvo por meio de transformações para que ela possua as mesmas proporções e orientações do atlas. Em seguida, sobrepõe-se a imagem ao atlas (processo de registro), permitindo que a imagem alvo receba os rótulos do atlas de maneira coerente e simétrica [7].

Existem abordagens que ampliam a ideia do atlas utilizando múltiplos atlases, conhecidas como técnicas de Multi-Atlas. Esses algoritmos combinam vários atlases de diferentes indivíduos para realizar uma fusão ou votação dos resultados [8]. Por exemplo, se a maioria dos atlases, após o registro com a imagem de entrada, determinar que uma determinada região é o hipocampo, então essa região será rotulada como tal na imagem de entrada, com base na maioria dos votos.

Outras técnicas buscam combinar o conceito de multi-atlas com CNNs [9], extraindo características tanto por meio do registro e votação dos múltiplos atlases quanto da análise feita pelas redes neurais. Essa combinação visa melhorar a acurácia na definição dos rótulos. No entanto, há um esforço contínuo para desenvolver técnicas que ofereçam uma segmentação eficiente e rápida do cérebro, em contraste com as técnicas baseadas em atlas, que podem ser computacionalmente dispendiosas e levar a tempos de execução impraticáveis em certos contextos. O objetivo é encontrar um equilíbrio entre técnicas de pré-processamento e arquiteturas de CNNs que, em conjunto, proporcionam resultados eficazes para esse problema.

Portanto, considerando o impacto significativo das técnicas de pré processamento na qualidade do parcelamento cerebral, existem frameworks em Python [10] que oferecem funções padrão para essas operações de maneira padronizada e estruturada, como o Nifty [12] e o MONAI [13]. Esses frameworks fornecem componentes para a construção de pipelines de execução estruturados que abrangem todo o processo de aprendizado da rede, desde o pré-processamento e treinamento da arquitetura até a implementação do modelo.

No entanto, mesmo com o uso dessas técnicas, ainda há dificuldades substanciais em obter modelos de segmentação de imagens que atuem com alta precisão em imagens médicas. Isso se deve à complexidade intrínseca dessas imagens, que não apresentam padrões óbvios ou bem definidos, exigindo modelos com uma alta capacidade de reconhecimento de padrões complexos.

Neste trabalho, será analisada a CNN HighResNet, proposta por Wenqi Li et al. [6]. Essa rede foi originalmente desenvolvida utilizando o *framework*

Nifty [12], e será comparada com a U-Net, uma arquitetura amplamente utilizada para segmentação de imagens. Tal comparação será feita como estudo de caso para validar a execução do *pipeline* construído.

O objetivo deste estudo é compreender os componentes da HighResNet, que se destaca por ser mais compacta em termos de número de parâmetros, mas ainda assim competitiva em relação a outras CNNs de alto desempenho. Além de analisar sua arquitetura, pretende-se reproduzir os experimentos realizados pelos autores, com algumas modificações, como a quantidade de imagens e suas dimensões.

A comparação entre a HighResNet e a U-Net será realizada utilizando o framework MONAI, em substituição ao Nifty. Para isso, será construído um *pipeline* que englobará todas as etapas do *brain parcellation*, desde o pré-processamento das imagens até o *deploy* do modelo.

1.2 Motivação

Nos últimos anos, o uso de algoritmos de inteligência artificial para o diagnóstico de imagens médicas tem ganhado destaque, proporcionando suporte aos profissionais da saúde e acelerando o processo de diagnóstico clínico. Essas abordagens não apenas aumentam a precisão, como também reduzem o tempo necessário para a identificação de condições médicas, beneficiando tanto médicos quanto pacientes.

Dentro desse contexto, a técnica de *brain parcellation* surgiu como uma solução essencial para segmentar imagens volumétricas do cérebro em diferentes estruturas anatômicas. Essa segmentação permite que especialistas analisem regiões específicas com maior precisão, auxiliando na identificação de padrões associados a diversas doenças neurológicas. Por exemplo, no diagnóstico precoce do Alzheimer, o hipocampo é uma das primeiras áreas afetadas, resultando em perda de memória recente. Com a segmentação dessa estrutura na imagem, o profissional pode direcionar sua análise, facilitando a detecção da doença em estágios iniciais.

A complexidade da tarefa de *brain parcellation* exige o uso de modelos avançados de aprendizado profundo. Para essa finalidade, optou-se pela HighResNet, uma arquitetura já aplicada com sucesso nessa tarefa. Além do modelo, foi necessário desenvolver um pipeline de pré-processamento personalizado, garantindo que os dados de entrada estivessem padronizados e otimizados para o treinamento da rede neural.

Outro grande desafio desse tipo de abordagem está na estruturação de um *pipeline* completo e eficiente, que vá desde o pré-processamento das imagens até a disponibilização do modelo treinado em um ambiente de produção. A

construção de um fluxo *end-to-end*, que seja replicável, escalável e organizado, demanda ferramentas robustas que simplifiquem a implementação.

Para atender a esses requisitos, foi escolhido o *framework* MONAI, projetado especificamente para aplicações de aprendizado profundo em imagens médicas. O MONAI fornece suporte para todo o pipeline de desenvolvimento, permitindo a criação de soluções flexíveis e adaptáveis às necessidades do projeto.

Por fim, um modelo treinado não pode ficar apenas em um ambiente de pesquisa, sendo essencial que ele seja disponibilizado para uso prático e acessível. Assim, o projeto inclui uma abordagem para deploy do modelo, utilizando FastAPI e uvicorn, permitindo que usuários façam requisições e obtenham previsões de forma simples e eficiente. Dessa maneira, além de englobar todas as etapas do desenvolvimento da solução, o projeto também busca demonstrar como um modelo pode ser transformado em um serviço funcional e aplicável ao mundo real.

1.3 Objetivo Geral

Como objetivo principal deste trabalho, tem-se a construção de um *pipeline* para a realização de *brain parcellation* utilizando o *framework* MONAI, comparando as arquiteuras U-Net e HighResNet.

A ideia da utilização do *framework* é estudar e aplicar em um projeto real um biblioteca que está sendo amplamente utilizada e vem ganhando bastante força na comunidade de aplicação computacional na área da saúde e implementar através de seus recursos um *pipeline* para a execução da tarefa de *brain parcellation*.

1.4 Objetivos Específicos

O objetivo geral deste projeto é dividido nos seguintes objetivos específicos:

- Aprender sobre o framework MONAI e seus componentes.
- Aprender sobre a arquitetura HighResNet e sua utilização para segmentação de imagens médicas, a fim de evidenciar as suas qualidades e vantagens.
- Comparar os resultados obtidos com a HighResNet com resultados providos dos mesmos experimentos aplicados com a rede U-Net, como estudo de caso para validar o pipeline.

- Criar um *pipeline* utilizando MONAI para a treinamento, validação e teste da arquitetura HighResNet para realizar a tarefa de *brain parcellation*.
- Realizar o deploy do modelos treinados (HighResNet e U-Net) para serem utilizados via API como serviço.

Capítulo 2

Fundamentação Teórica

2.1 Brain Parcellation

Brain parcellation é a tarefa ou processo de divisão de uma imagem de um cérebro em regiões distintas, as quais são chamadas de parcelas ou regiões de interesse (ROI), com base em critérios anatômicos, funcionais ou estruturais. Esse método é amplamente utilizado na neurociência para investigar funções específicas de diferentes regiões do cérebro, e também para identificar alterações cerebrais e suas relações com doenças.

O parcelamento pode então ser anatômico ou funcional. O anatômico é baseado em características estruturais visíveis, como sulcos e giros do cérebro. Já o funcional é baseado em padrões de atividade cerebral detectados por técnicas como o fMRI.

A técnica de *brain parcellation* pode ser utilizada em diferentes contextos, tendo como principais aplicações:

- **Neurocirurgia**: Identificar regiões críticas antes de uma operação cirúrgica
- **Estudo de conectividade**: Mapear redes funcionais, como a *default mode network* (DMN)
- **Modelagem Computacional**: Criar modelos do cérebro para simulações de atividades neurais
- **Diagnóstico de Doenças**: Auxílio na detecção de padrões cerebrais associados a doenças neurodegenerativas, como Alzheimer, AVC, esclerose mútilpla, entre outros distúrbios.

Para realizar o parcelamento cerebral, tem-se diferentes abordagens e técnicas:

- Baseado em atlas: Essas técnicas utilizam atlas cerebrais predefinidos, que são mapas anatômicos ou funcionais do cérebro criados a partir de dados de um grande número de indivíduos. As regiões do cérebro são definidas com base em características anatômicas ou funcionais.

- Baseado em dados: Essas técnicas utilizam algoritmos para dividir o cérebro com base em características extraídas diretamente dos dados de imagem, sem depender de atlas predefinidos. Podem ser supervisionadas (com rótulos) ou não supervisionadas (sem rótulos). Tal abordagem tem como vantagens a adaptabilidade a variações individuais e capacidade de capturar padrões complexos nos dados, porém requer grandes quantidades de dados rotulados (no caso de abordagens supervisionadas) e requer grande poder computacional para conseguir treinar modelos supervisionados ou para executar modelos não supervisionados.

No caso deste projeto, a abordagem escolhida para o parcelamento cerebral foi a baseada em dados.

2.2 Pyrobex e Antspy

Apesar do objetivo do projeto ser criar o pipeline end-to-end utilizando o *framework* MONAI, outras bibliotecas tiveram que ser utilizadas para complementar em funções que o MONAI não possui nativamente.

Uma das bibliotecas que foram utilizadas foi a biblioteca Pyrobex. Tal biblioteca fornece uma interface para o ROBEX (Robust Brain Extraction), uma ferramenta utilizada para extração de cérebro de MRIs do tipo T1. A biblioteca facilita a integração do Robex em pipelines de neuroimagem, permitindo que a operação de *skull stripping* fosse inserida no pipeline MONAI construído.

Por fim, também foram utilizadas funcionalidades da biblioteca ANTsPy. Ela é otimizada e validada para imagens médicas, que serve como uma interface para o framework ANTs (Advanced Normalization Tools) escrito em C++. Ela oferece funcionalidades avançadas para leitura e esrita de imagens médicas, além de algoritmos para registro, segmentação e aprendizado estatístico. Dessa biblioteca, foram utilizadas as funcionalidades de correção de bias e mudança de origem da imagem no pré-processamento, e a operação de corregistro de imagens para criação dos rótulos das imagens. Por mais que a biblioteca MONAI possua funcionalidades de corregistro baseadas em *machine learning*, foi escolhida a função de corregistro da ANTsPy por sua robustez e facilidade de invocação e implementação.

2.3 MONAI

O MONAI é um framework de código aberto projetado para acelerar pesquisa e desenvolvimento em IA relacionados a área médica, principalmente na utilização de visão computacional voltada para imagens médicas [11]. Este framework tem como principais características:

1. **Foco em imagens médicas**

   Como Dito anteriormente, MONAI é especializado em dados de imagens médicas, como ressonância magnética (MRI), tomografia computadorizada (CT), ultrassom e outras modalidades de imagens. Há suporte para formatos populares, como NifTI, DICOM e outros.

2. **Integração com PyTorch**

   Este framework é construído utilizando o pytorch como base. Permite a flexibilidade e o poder do PyTorch para criar redes neurais especializadas para diferentes problemas e contextos.

3. **Modelos e arquiteturas otimizadas**

   O MONAI inclui implementações de modelos de deep learning comumente usados em aplicações médicas, como UNet, SegResNet, e DenseNet. Eles vêm pré-configurados para tarefas como segmentação, classificação e registro de imagens.

4. **Suporte para transferência de aprendizado**

   O MONAI facilita o uso de modelos pré-treinados para ajustar redes a tarefas específicas com conjuntos de dados menores, uma característica comum em ambientes médicos.

5. **Ferramentas para métricas e validação**

   Inclui métricas avançadas para avaliação de desempenho em tarefas médicas, como coeficiente de Dice, Sensibilidade, Especificidade e distância de Hausdorff.

6. **Modularidade e flexibilidade** O framework é altamente modular, permitindo a personalização de cada etapa do pipeline, desde o pré-processamento até a construção de modelos e treinamento.

7. **Compatibilidade com o NVIDIA Clara**

   Como parte do ecossistema NVIDIA Clara, o MONAI oferece suporte otimizado para hardware NVIDIA, incluindo GPUs e soluções como mixed precision training.

Por muito tempo, o framework dominante foi o Nifty, o qual é um framework para processamento de imagens médicas baseado na biblioteca tensorflow. Porém, atualmente, o uso da biblioteca PyTorch encontra-se mais relevante em estudos e desenvolvimentos relacionados à IA médica. Além disso, a comunidade e o andar do desenvolvimento do framework MONAI [13] mostra-se mais acentuado em relação ao Nifty, o qual teve uma diminuição na contribuição de seu desenvolvimento.

Como vantagem, o Monai foi desenvolvido pensando em sua aplicação em clínicas e ambientes de pesquisa, tendo suporte nativo a diferentes tipos de dados de imagem médica, como DICOM, NifTI e outros formatos, além de ferramentas para a integração em workflows clínicos [13]. Por fim, uma das maiores vantagens do Monai hoje em cima do Nifty é sua vasta e ampla rede de conhecimento, composta por documentações detalhadas e diversos tutoriais, exemplos práticos e casos de uso disponíveis para consulta na internet.

2.4 U-Net

A rede neural U-Net é uma arquitetura de segmentação de imagens, primeiramente citada em [15]. Tal rede teve como primeiro propósito contornar o desafio da escassez de imagens rotuladas na área médica.

A arquitetura U-Net é composta, basicamente, por dois componentes: o caminho de contração e o caminho de expansão.

- **Caminho de contração:** visa diminuir as dimensões espaciais da imagem, enquanto captura informações relevantes da mesma, composta por camadas de *encoders*.
- **Caminho de expansão:** visa realizar um *upsampling* do mapa de características da imagem e produzir um mapa de segmentações utilizando os padrões aprendidos no caminho de contração, composta por camadas de *decoders*.

A representação da U-Net pode ser conferida na figura 2.1:

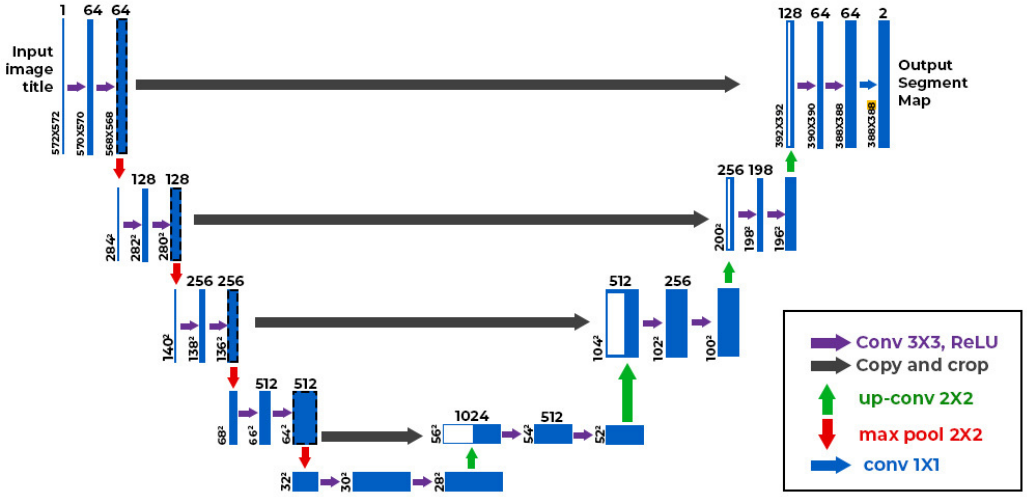

Figura 2.1: Arquitetura U-Net

Como a U-Net foi criada especialmente para aplicações da medicina e é uma rede amplamente utilizada e reconhecida, foi escolhida para comparar com a rede estudada neste projeto, a HighResNet.

2.5 HighResNet

A Highresnet, como a arquitetura U-Net, é projetada para tarefas de segmentação de imagens médicas, porém esta foi proposta para capturar informações de longo alcance preservando detalhes locais, os quais podem ser perdidos por conta de operações de *downsampling* agressivas, como ocorre no caso da U-Net no caminho de contração. Dessa forma, a HighResNet mantém a alta resolução da imagem ao longo da rede. Para isso, a rede utiliza-se de blocos residuais com *receptive field* aumentado.

Além disso, tal arquitetura é multiescala, pois utiliza-se de blocos de convolução de diferentes tamanhos, a fim de extrair detalhes finos mas também padrões globais das imagens.

Por fim, tal rede é utilizada para tarefas que exigem precisão espacial, como é o caso de *brain parcellation*, segmentação de órgãos e estruturas anatômicas. A rede da HighResNet é possível ser identificada na figura 2.2.

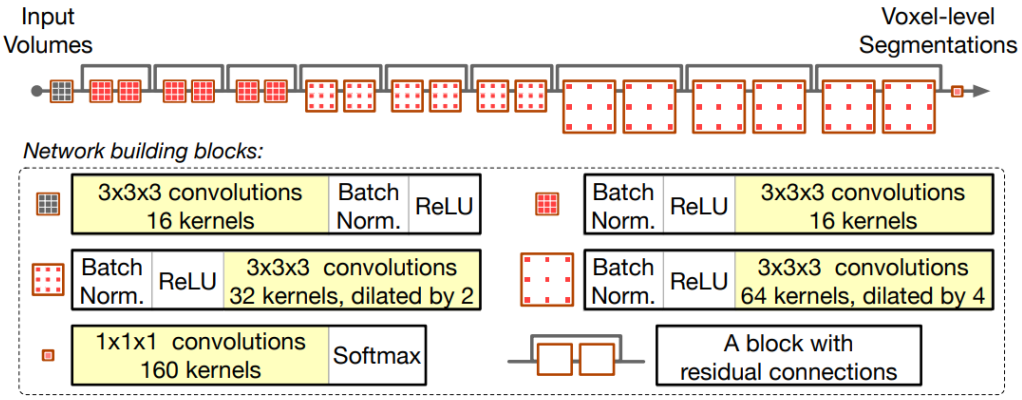

Figura 2.2: Arquitetura Highresnet

2.6 Artigo Base

O artigo "On the compactness, efficiency, and representation of 3D convolutional networks: brain parcellation as a pretext task"foi utilizado como base para o projeto. Tal artigo tem como objetivo mostrar a eficácia da rede HighResNet realizando a tarefa de *brain parcellation* como desafio de desempenho. Dessa forma, tal rede foi treinada utilizando o dataset ADNI, composto por 543 imagens de MRI do tipo T1 ponderada de tamanho 182x244x246 (443 para treinamento, 50 para validação e 50 para teste). O tamanho médio dos voxels presentes em tais imagens foi de 1.18 mm x 1.05 mm x 1.05 mm.

A partir de tais imagens, a rede foi treinada para segmentar as imagens em 155 rótulos diferentes. Em seguida, a fim de avaliar a rede, tal arquitetura foi comparada com outras redes do estado da arte (NoRes-entropy, NoRes-dice, Deepmedic, 3D U-net e V-net). Tais comparações foram feitas utilizando a métrica *Dice Coefficient Similarity* (DCS), mas especificamente a sua média entre todas as imagens de teste e o desvio padrão dessa média. Essa métrica representa uma porcentagem de acerto dos rótulos segmentados, variando entre 0 e 1.

Além da média do DCS, o autor também representou em um box plot a distribuição dos valores de DCS para cada uma das imagens, comparando algumas partes do cérebro (rótulos), conseguindo demonstrar de forma visual como cada uma das redes selecionadas desempenhou em segmentar diferentes regiões, as quais foram escolhidas arbitrariamente.

Ao final da pesquisa e dos experimentos, o autor concluiu que a rede proposta é mais compacta estruturalmente que as outras abordagens, trazendo níveis de desempenho superiores. A partir da aplicação da rede na tarefa

de brain parcellation, ele foi capaz de provar essa evolução em desempenho em segmentação de imagens que a HighResNet proporcionou em relação às outras arquiteturas comparadas.

A partir do artigo, foi escolhido replicar o mesmo experimento com a HighResNet implementada no framework MONAI, porém com algumas alterações. Por conta de inviabilidade de hardware, foi escolhido utilizar imagens de dimensão menor, definido abitrariamente para 91x122x123 (metade em cada dimensão do que utilizado no artigo baso) e voxels de tamanho 1x1x1. Para contornar possíveis perdas de desempenho no treinamento da rede pela diminuição da escala da imagem, foi escolhido utilizar 1200 imagens de treinamento ao invés de 443. Além disso, para fazer a validação e teste do mesmo, foram utilizadas 200 imagens ao invés de 50, para tentar manter uma proporção próxima a 80/10/10.

Capítulo 3

Materiais e Métodos

3.1 Visão Geral

O *pipeline* construído para implementar a tarefa de *brain parcellation* pode ser observado na imagem 3.1.

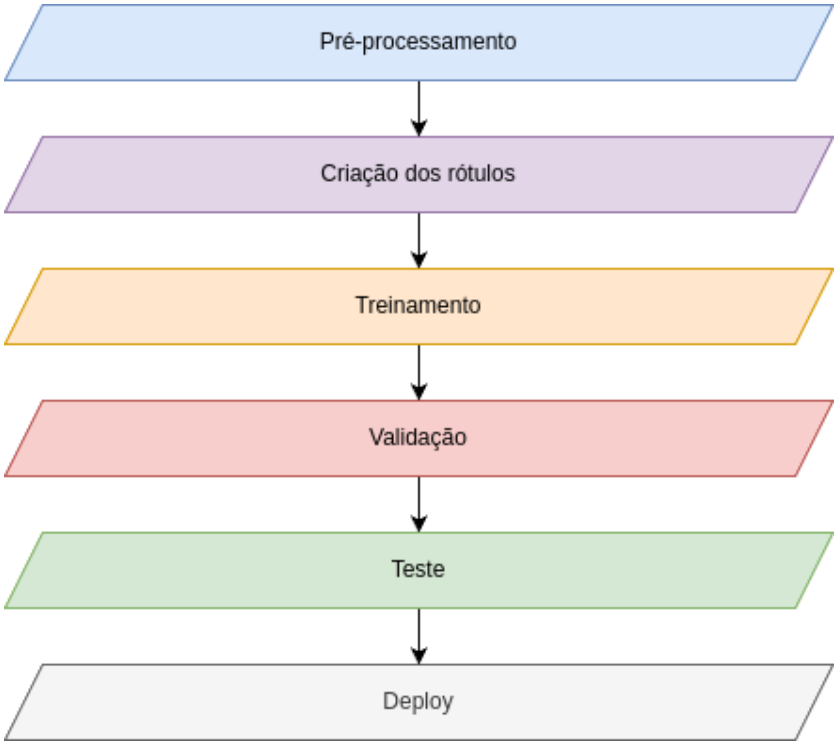

Figura 3.1: Pipeline do projeto

1. **Pré-processamento**

   Na primeira etapa do *pipeline*, as imagens são submetidas a uma certa gama de transformações para apresentarem um mesmo padrão entre si, para garantir consistência no processo de treinamento da rede neural.

2. **Criação dos rótulos**

   Nesta etapa da *pipeline*, todas as imagens são alinhadas com um atlas através da transformação de corregistro, a fim de criar um par de rótulo para cada uma das imagens. Dessa forma, com cada imagem tendo seu rótulo, é possível treinar a rede com estes pares de exemplo.

3. **Treinamento**

   Após possuir todos os pares de imagens e anotações/rótulos, as duas redes (U-Net e HighResNet) são treinadas com o mesmo conjunto de imagens.

4. **Validação**

   Durante o treinamento de ambas as redes, há uma etapa de validação, para avaliar o treinamento da rede e ocasionar um *early stoppying* caso necessário, para evitar *overfit* nos modelos.

5. **Teste**

   A partir de um conjunto de imagens para teste, ambas as redes foram comparadas em desempenho na predição dos rótulos para este conjunto.

6. **Deploy**

   A fim de disponibilizar os modelos, foi criada uma API, utilizando a biblioteca FastAPI, capaz de enviar imagens para o modelo e retornar ao usuário as imagens devidamente parceladas.

3.2 Banco de Imagens

Para treinar a rede neural, o *dataset* a ser utilizado será o banco de imagens da ADNI (Alzheimer's Disease Neuroimaging Iniative). O estudo ADNI dá suporte a investigação e tratamento para diminuir ou parar a progressão da doença de alzheimer. Tal iniciativa possui imagens de diferentes tipos, como T1, T2, Flair, GRE, fMRI, entre outras. Porém, para este projeto, serão utilizadas imagens do tipo T1 ponderadas.

Para realizar o treinamento adequado das redes, foram escolhidas, aleatoriamente, 1200 imagens para treinamento e 200 para validação. Foram

escolhidas mais imagens do que o autor da HighResNet utilizou HighResNet, para compensar a diminuição da escala das imagens. Além disso, foram utilizadas 200 imagens para testar os modelos, a fim de compará-los em desempenho de predição e tempo de execução.

3.3 Configurações de hardware

Segue abaixo as configurações de *hardware* da máquina utilizada para treinar a rede HighResNet e executar os códigos desenvolvidos no projeto:

- **CPU**: AMD Ryzen Threadripper PRO 5975WX 32-Cores
- **Memória RAM**: 256 GB
- **GPU**: NVIDIA GeForce RTX 4090

3.4 Pré-processamento

Para garantir que todas as imagens tenham a qualidade e o padrão de formato necessário para a rede, é necessário realizar transformações de pré-processamento e de *standartization*.

O pré-processamento se refere a processos que garantam que os dados estejam limpos, consistentes e estruturados para serem utilizados para análises ou para servirem como entrada para modelos de aprendizado de máquina.

Em uma imagem, temos dois tipos de coordenadas: coordenadas da imagem e coordenadas de mundo.

Coordenadas de imagem se referem as posições do voxels em uma grade discreta com valores indexados. Já coordenadas de mundo se referem à posição física dos voxels no mundo real. Para garantir que uma imagem tenha as mesmas coordenadas de mundo que outra, é necessário realizar operação de rescale, resize e reorientação.

Como há essa distinção para as imagens em relação a orientação e dimensionamento, é necessário ajustá-las para usar em um treinamento de uma rede neural. Imagens com padrões diferentes podem resultar em um viés na hora de treinamento da rede. As diferenças de escala podem fazer com que o modelo atribua importância desproporcional a certas regiões ou características da imagem.

Para este projeto, nove operações foram realizadas em cima das imagens de treinamento do modelo:

- **Skull-stripping**: transformação responsável por realizar a remoção de estruturas não cerebrias da MRI, como o crânio, couro cabeludo e outros tecidos. Para este projeto, foi escolhido o algoritmo Robex da biblioteca pyrobex para realizar tal operação, incorporando-o com a classe Transform da biblioteca MONAI ao *pipeline*. O ROBEX utiliza modelos de forma probabilísticos para estimar a localização e os contornos do cérebro na imagem. Esses modelos são pré-treinados em conjuntos de dados de MRI e ajudam o algoritmo a fazer previsões robustas mesmo em imagens com variações anatômicas ou ruído.

- **Correção de bias**: remove variações de intensidade de baixa frequência causadas por inhomogeneidades do campo magnético em MRI. Essas variações podem criar regiões com brilho ou escuridão indesejada, dificultando a análise ou segmentação da imagem. A função de correção de bias utilizada foi via antspyx, porém também incorporada como uma transformação do MONAI utilizando a classe *Transform*.

- **Spacing**: Para garantir que todas as imagens possuam um mesmo tamanho de voxel padrão, foi utilizada a operação de spacing para padronizar os voxels das imagens para 1x1x1 mm, garantindo uma imagem isotrópica, diferente do que foi feito em HighResNet. A escolha da padronização das imagens para isotrópicas se dá ao melhor funcionamento e treinamento das redes com tal tipo de imagem. Tal operação foi aplicada utilizando a transformação *Spacing* do pacote MONAI, mostrada em exemplo na figura 3.2.

Figura 3.2: Exemplo da aplicação da operação de spacing

- **Reorientação**: o ajuste de orientação faz com que as imagens tenham a mesma orientação média RAS+, alinhando-as em orientação. A orientação RAS+ é uma orientação médica utilizada para se referir às diferentes faces que compõem uma imagem médica em três dimensões. Tais nomenclaturas que compõem a RAS são essenciais para analisar cortes de uma imagem médica tridimensional. O acrônimo RAS referentes às seguintes direções anatômicas: R (Direita), A (Anterior), S (Superior) e + (Esquerda, Posterior, Inferior). Tal operação foi realizada utilizando a transformação Reorientation do próprio *framework* MONAI, com exemplo na figura 3.3.

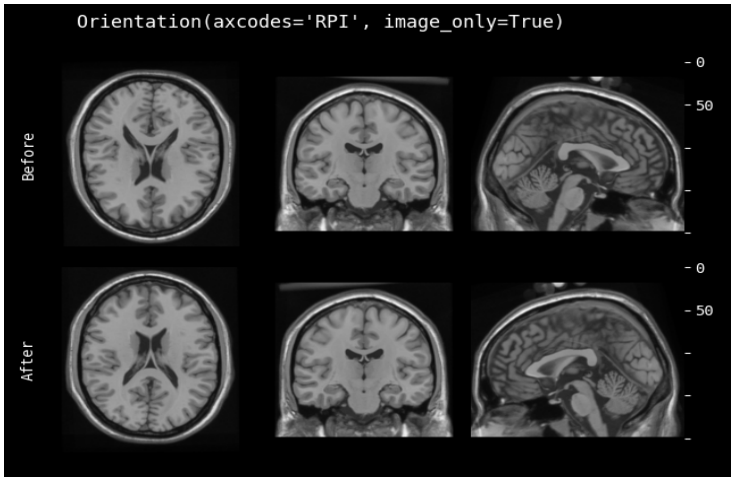

Figura 3.3: Exemplo da aplicação da operação de orientation

- **Normalização de Histograma**: essa técnica de processamento ajusta a distribuição dos valores de intensidade dos pixels para melhorar o contraste e a uniformidade da imagem. Isso é necessário por conta da diferença de intensidade das imagens gerados pelos *scanners* que as geraram. Essa transformação foi realizada utilizando a classe HistogramNormalize do MONAI, com exemplo na figura 3.4.

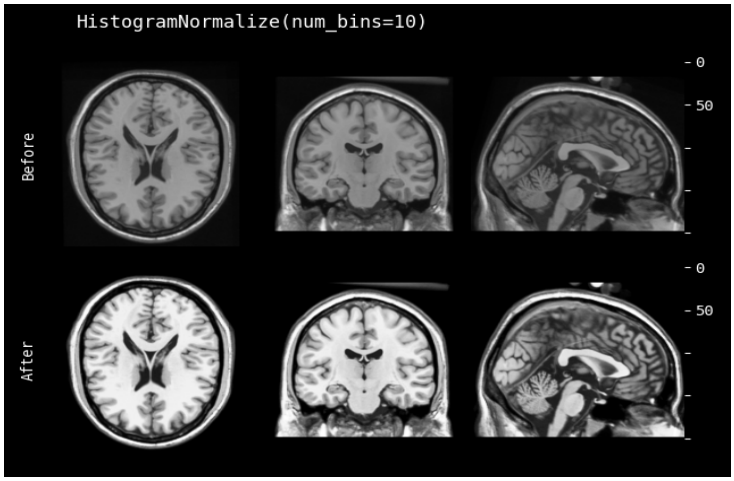

Figura 3.4: Exemplo da aplicação da operação de normalização de histograma

- **Normalização de Intensidade**: tal operação garante que as imagens tenham faixas de intensidade num mesmo intervalo padronizado, não enviesando o modelo por conta de grandes variações de intensidade em regiões diferentes da imagem. Além disso, a normalização ajuda o treinamento acontecer de forma mais rápida e também de forma mais estável. Isso ocorre porque os gradientes usados para ajustar os pesos da rede têm magnitudes mais consistentes, evitando oscilações grandes ou passos pequenos durante a otimização. Essa operação foi realizada nas imagens através da transformação ScaleInsensity do MONAI, exemplificada na figura 3.5. Tal operação garante que os valores dos voxels fiquem num intervalo escolhido, neste caso entre 0 e 1.

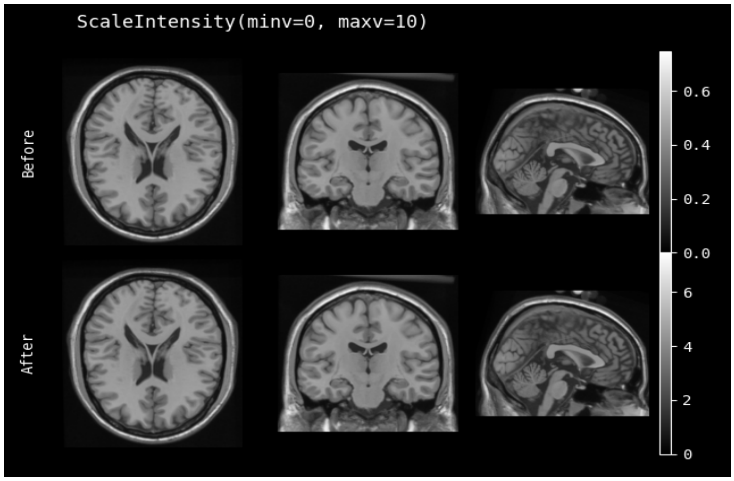

Figura 3.5: Exemplo da aplicação da operação de normalização

- **Crop Foreground**: essa transformação detecta e recorta automaticamente a menor região retangular ou cúbica que contém todos os valores acima de um certo limiar em um canal específico da imagem. Neste projeto, foram utilizados os parâmetros padrões da função. Tal operação é realizada pela transformação CropForeground do MONAI, exemplificada na figura 3.6.

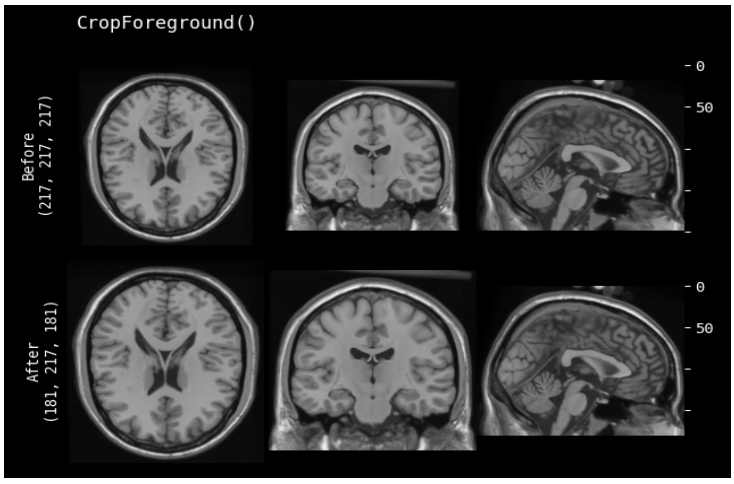

Figura 3.6: Exemplo da aplicação da operação de crop foreground

- **Atualização da origem**: A fim de garantir que as imagens possuam um padrão de sistemas de coordenadas de mundo e de imagem, foi utilizada a função "set_origin"da biblioteca ANTsPyX para definir a

origem da imagem como (0,0,0), padronizando o sistema de coordenadas entre todas as imagens. Foi utilizada a metaclasse Transform do MONAI a fim de criar uma transformação personalizada com a função set_origin que não é nativa do framework. A transformação criada que realiza tal função foi denominada de "SetOriginToZeroWithANTs".

- **Redimensionamento**: ajusta os valores dos voxels para uma dimensão determinada. Em primeira instância, foi escolhida a dimensão (182, 244, 246) por conta do artigo base do projeto [6]. Porém, com as especificações de hardware utilizadas houve problemas de falta de memória de GPU. Dessa forma, para conseguir realizar o experimento, as dimensões foram reduzidas pela metade, então redimensionando as imagens para (91, 122, 123). Tal operação foi realizada pela transformação Resize do MONAI, com exemplo na figura 3.7.

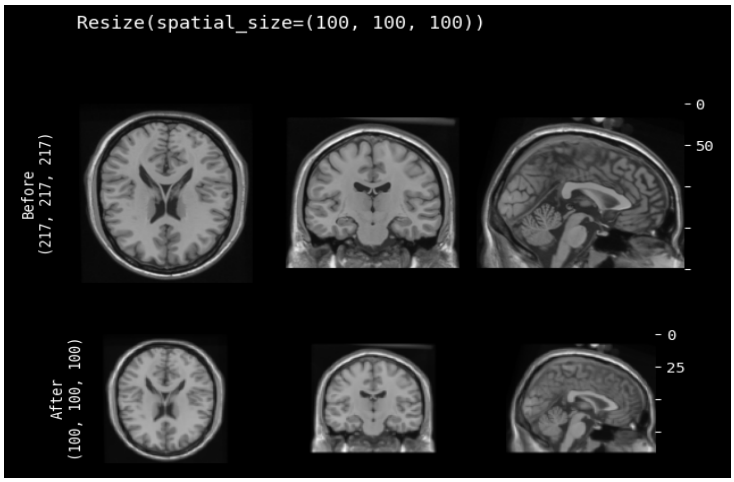

Figura 3.7: Exemplo da aplicação da operação de redimensionamento

Após realizar as operações de pré-processamento, as imagens estão prontas para entrarem na etapa de rotulação. Um exemplo do pré-processamento completamente aplicado a uma imagem pode ser visto a seguir. A imagem 3.8 é a imagem antes do processamento, e a imagem processa é apresentada na figura 3.9.

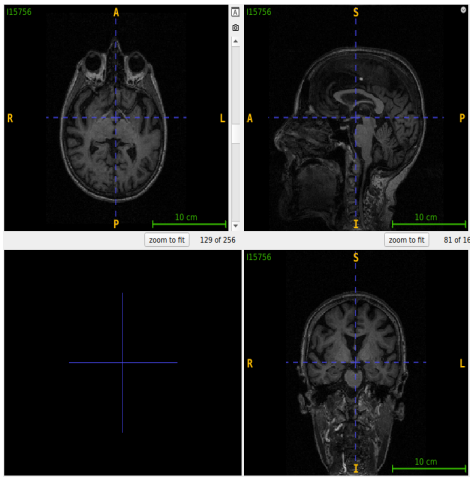

Figura 3.8: Imagem antes do pré-processamento

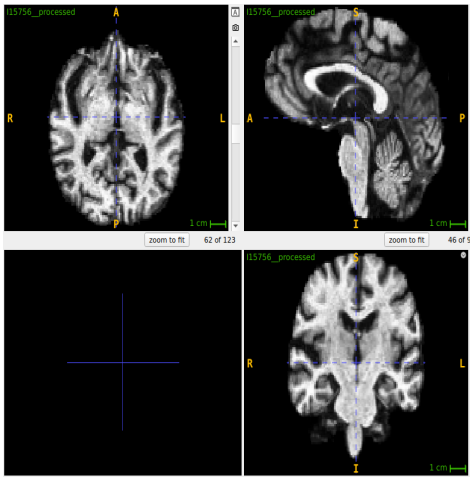

Figura 3.9: Images após o pré-processamento

3.5 Rotulação das Imagens

Num cenário ideal, cada imagem deveria ser minuciosamente rotulada por um especialista da área da saúde, no caso um médico. Essas anotações representam o nível mais alto de confiabilidade, e por isso são denomiadas de padrão ouro, do inglês *gold standard* [16]. Porém, como a hora de um médico para realizar este trabalho é extremamente cara, muitas aplicações, como é o caso desta também, acabam trabalhando com o padrão prata ou bronze. No caso do padrão prata, as imagens de treinamento são previamente rotuladas por um modelo ou algoritmo de segmentação eficiente e validado por profissionais qualificados.

No caso deste projeto, por questões de viabilidade, foi escolhido trabalhar com o padrão bronze. Neste caso, os rótulos das imagens são gerados através do corregistro de um atlas com a imagem. Como o atlas possui os rótulos, faz-se o corregistro do mesmo na imagem e assim é gerado o rótulo para a respectiva imagem.

O corregistro nada mais é do que o processo de alinhar diferentes imagens de um mesmo paciente ou entre pacientes para comparação, fusão ou análise. Nesse caso, o corregistro é feito para alinhar os rótulos do atlas nas imagens, criando assim um rótulo para cada uma delas.

O atlas utilizado para gerar os rótulos foi o ALL (Automated Anatomical Labeling), o qual é amplamente utilizado em neuroimagem. Ele fornece uma segmentação anatômica do cérebro em diferentes regiões de interesse (ROIs)

e é frequentemente aplicado em estudos de fMRI e MRI. Tal atlas divide o cérebro em 116 ROIs.

Dessa forma, foi construído um código capaz de pegar o rótulo do atlas e realizar um corregistro nas imagens de treinamento, para assim criar um par imagem e rótulo para cada imagem de treinamento. Existem diferentes tipos de corregistro:

- **Corregistro Rígido**: utilizada apenas de deslocamento e rotação (6 graus de liberdade). Útil quando não há deformação entre as imagens.
- **Corregistro Afim**: Além de deslocamento e rotação, inclui escalamento e cisalhamento (12 graus de liberdade). Aplicado em casos onde há pequenas diferenças de escala entre as imagens
- **Corregistro Não Rídigo**: Permite transformações locais, adaptando estruturas de forma não linear. Útil para alinhar imagens de diferentes pacientes e modalidades.
- **Corregistro baseado em superfície**: usa pontos ou superfícies extraídas das imagens para realizar o alinhamento. Aplicado quando as estruturas de interesse são bem definidas.
- **Corregistro baseado em intensidade**: usa diferenças de intensidade para alinhar as imagens.

No caso desse projeto, foi utilizada a função *registration* da biblioteca ANTsPyX para realizar o corregistro dos labels nas imagens. Tal função aplica o método de corregistro não rídigo. Como o atlas pertence a um paciente diferente e tem características de espaçamento e dimensão diferentes das imagens pré-processadas, tal abordagem se deu como a mais adequada.

Após o corregistro, o rótulo agora alinhado passada ainda por um processo de pós processamento utilizando operadores morfológicos. Tal função utiliza do operador de erosão para remover pequenos artefatos e suavizar os contornos da imagem, e posteriormente é aplicada a operação de dilatação a fim de recuperar a estrutura original das regiões pós erosão, eliminando pequenas irregularidades sem comprometer a segmentação. O elementro estruturante (kernel) utilizado por padrão foi um de dimensões (3,3,3).

Vale ressaltar que as imagens do dataset ADNI podem variar na questão do grau da doença de Alzheimer que o paciente possui. Imagens de cérebros com alto grau de Alzheimer apresentam hipertrofia dos sulcos cerebrais. Dessa forma, o corregistro pode não ser exato e apresentar imperfeições.

Um exemplo de rótulo criado para uma das imagens de treinamento pode ser observado nas imagens 3.10 e 3.11:

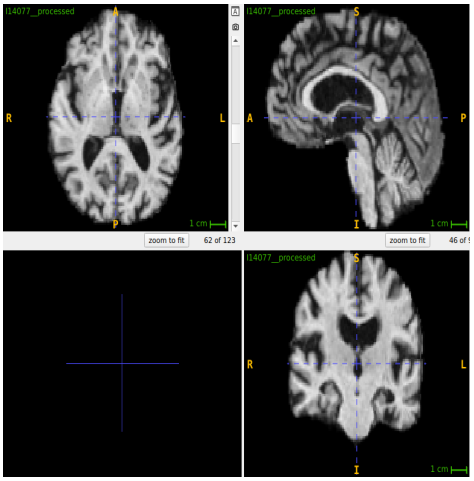

Figura 3.10: Imagem pré-processada

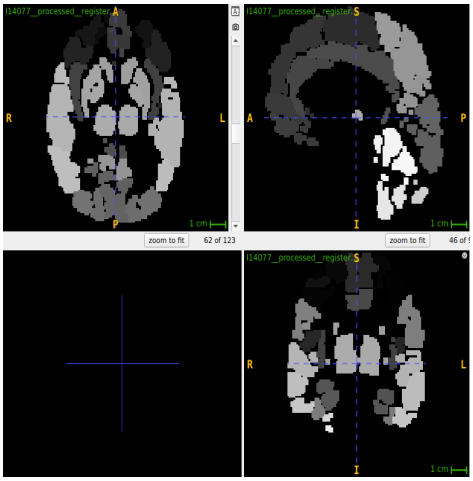

Figura 3.11: Rotulado criado para a imagem pré-processada

O rótulo sobreposto na imagem pode também ser observado na figura 3.12:

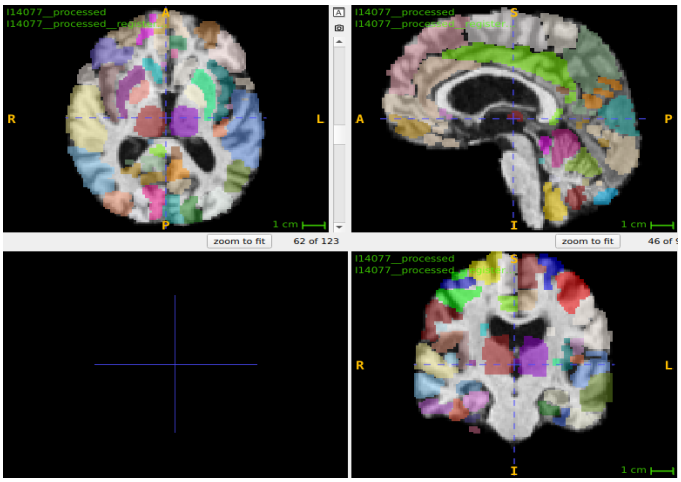

Figura 3.12: Imagem e rótulo sobrepostos

Em média, o processo de rotulação demorou entre 10 a 15 segundos para ser feito para cada uma das imagens separadamente.

3.6 Treinamento da Rede

Para então realizar a operação de *brain parcellation*, fora escolhidas as redes neurais U-Net e HighResNet.

Os parâmetros da rede neural U-Net definidos foram os seguintes:

- **spatial_dims**: 3 (número de dimensões das imagens de entrada)
- **in_channels**: 1 (número de canais de entrada - 1 imagem)
- **out_channels**: 117 (número de canais de saída - 116 rótulos + fundo)
- **channels**: (16, 32, 64, 128, 256) (quantidade de filtros nas camadas de contração e expansão)
- **strides**: (2,2,2,2) (definição do fator de *downsampling*)
- **num_res_units**: 2 (número de blocos residuais em cada nível - ajudam a lidar com o problema de *gradient vanishing*)

Os parâmetros definidos para a HighResNet foram:

- **spatial_dims**: 3 (número de dimensões das imagens de entrada)
- **in_channels**: 1 (número de canais de entrada - 1 imagem)
- **output_channels**: 117 (número de canais de saída - 116 rótulos + fundo)

Para ambas as redes, a função de *loss* foi escolhida a DiceLoss [17]. Tal função é baseada no coeficiente de Dice, que mede a sobreposição entre a predição da rede e o rótulo verdadeiro (*ground truth*).

A função de Dice Loss é definida como:

$$\text{Dice}(P, G) = \frac{2 \sum (P \cdot G)}{\sum P + \sum G} \tag{3.1}$$

onde:

- P é o mapa de probabilidades previsto pelo modelo;
- G é o rótulo verdadeiro (*ground truth*);
- O numerador representa a interseção entre a predição e o rótulo;
- O denominador é a soma das intensidades das regiões preditas e verdadeiras.

A função de perda baseada no coeficiente de Dice é dada por:

$$\mathcal{L}_{\text{Dice}} = 1 - \text{Dice}(P, G) \tag{3.2}$$

A função 3.2 é foi escolhida por ter uma melhora convergência em regiões pequenas, com o denominador garantindo que mesmo rótulos pequenos contribuam para o cálculo da perda, tornando o aprendizado mais eficiente para estruturas pequenas no cérebro, o que pode ser muito conveniente para o *brain parcellation*. Além disso, ela reduz o impacto de falsos positivos e falsos negativos. Já que a DiceLoss trabalha com a interseção entre os conjuntos, ela penaliza tanto falsos positivos (FP) quanto falsos negativos (FN), ajudando a refinar os contornos da segmentação.

Como otimizador, foi escolhido o Adam (*Adaptive Moment Estimation*) [18], o qual tem como vantangens:

- **Adaptativo**: ajusta a taxa de aprendizado para cada parâmetro, o que pode auxiliar na velocidade de aprendizado
- **Momentum**: Consegue navegar em superfícies complexas como ravinas e pontos de sela, quando em busca do máximo local/global
- **Correção de Bias**: Com termos de correção de bias, consegue performar bem nos estágios iniciais do treinamento
- **Eficiente computacionalmente**: Mais eficiente e requer menos memória do que outras técnicas de otimização
- **Robusto em relação a hiperparâmetros**: É menos sensível a escolha de hiperparâmetros em relação a outros otimizadores

Para otimizar o processo de treinamento, foi utilizado o GrandScaler, o qual é uma ferramenta do PyTorch que faz parte do módulo torch.cuda.amp (Automatic Mixed Precision, ou AMP). Ele é usado para escalar os gradientes durante o treinamento de modelos com precisão mista, que combina cálculos em ponto flutuante de 16 bits (FP16) e 32 bits (FP32) [19].

Em cálculos de deep learning, muitas operações não precisam da alta precisão de FP32 (32 bits). Usar FP16 (16 bits) pode acelerar o treinamento e reduzir o uso de memória, pois operações em FP16 são mais rápidas e ocupam menos espaço na GPU.

Basicamente, o GrandScaler gera saídas em FP16 durante o *forward pass*, e os gradientes são calculados no *backward pass* utilizando FP16. O GrandScaler multiplica os gradientes por um fator de escala (scale) para evitar *underflow*. Posteriormente, os gradientes escalados são usados para atualizar os

parâmetros do modelo, mas antes de aplicar os gradientes aos parâmetros do modelo, o GradScaler desescala os gradientes de volta ao seu valor original.

Por fim, com os modelos e suas funções de suporte definidas, foram carregados os pares de imagens (imagem e seu rótulo) utilizado o LoadImage da biblioteca MONAI, convertendo as imagens para um tensor do Pytorch. Com as imagens devidamente carregadas e transformadas em tensores, o treinamento foi iniciado com um número de 150 épocas de treinamento.

3.7 Validação

Durante o treinamento da rede, após a conclusão de cada época, foi realizada uma validação num conjunto a par de imagens, com um total de 200 imagens.

Através do coeficiente de Dice é calculado o coeficiente de loss do modelo em cima do conjunto de validação. Isso é feito para identificar se em algum momento durante o treinamento o modelo está entrando em *overfitting* e perdendo a capacidade de generalizar sua predição em um conjunto a parte que não é utilizado para treinamento, no caso o de validação.

Dessa forma, foi estabelecido um parâmetro de paciência de 10 épocas, ou seja, caso o modelo identifique algum valor de loss de validação maior do que o melhor que ele ja registrou, ele começa a acrescentar um contador. Caso esse contador ultrapasse 10 épocas, o algoritmo conclue que o modelo não será capaz de diminuir a loss de validação com mais épocas de treinamento, e daquele momento em diante a rede apenas entrará no processo de *overfitting*.

Ambos os modelos foram treinados com o parâmetro de máximo de número de épocas sendo 150.

Na tabela 3.1, é capaz de analisar o tempo necessário de treinamento de cada um dos modelos, e qual época eles pararam o treinamento para não entrar em *overfitting*.

Arquitetura	**Tempo de treinamento [h]**	**Época atingida**
HighResNet	37	150
U-Net	0.31	43

Tabela 3.1: Resultados do treinamento

Por não aplicar *downsampling* nas imagens, a HighResNet demora um maior tempo para treinamento. Além disso, vale ressaltar que a rede atingiu o número de épocas máximas estipulada e ainda não indicou convergência, assim podendo continuar o treinamento por mais tempo.

Capítulo 4

Resultados e Discussão

4.1 Experimentos

A fim de realizar a validação da rede treinada, foi escolhido o índice de Dice para verificar o quão próximo é a predição do modelo em relação aos rótulos (*ground truth*).

O coeficiente de dice é uma ferramenta estatística que permite medir a similaridade entre dois conjuntos de dados. Ele é tipicamente usado em visão computacional para analisar a similaridade entre duas imagens volumétricas. Ele é medido pela seguinte expressão:

$$\text{Coeficiente de Dice} = \frac{2 \cdot |A \cap B|}{|A| + |B|} \tag{4.1}$$

Na equação 4.1, A é a imagem de referência e a imagem B é a imagem a ser comparada com A. E o valor do coeficiente de dice varia entre 0 e 1, sendo 0 uma total dissemelhança entre os conjuntos e 1 uma total semelhança.

Com essa métrica, é possível identificar quão próximo a técnica de *brain parcellation* do modelo treinado foi capaz de segmentar a imagem em relação ao *ground truth*, ou seja, o quão próxima está a predição do desejado.

Para ter uma comparação significante, foram escolhidas 200 imagens aleatórias para serem calculados o dice entre a predição do modelo para cada uma dessas imagens em comparação ao seu rótulo definido pelo algoritmo do ANTsPyX.

Como a imagem possui 117 rótulos (contando o fundo), foi necessário calcular, para cada imagem, o dice referente a cada um dós rótulos e, posteriormente, fazer a média entre os 117 dices calculados. Para isso, foram criadas máscaras que representassem apenas os voxels contendo o rótulo desejado e calculado o dice para cada uma dessas másscaras. Posteriormente,

após obter o dice de cada uma das 200 imagens, foi calculado o dice médio e o desvio padrão de todas as imagens.

Como resultado, obteve-se os seguintes números:

Arquitetura	**Função Loss**	**Otimizador**	**DCS (%)**
HighResNet	Dice Loss	Adam	81.64 ±0.09
U-Net	Dice Loss	Adam	83.13 ±0.08

Tabela 4.1: Resultados do teste

Percebe-se na tabela 4.1 que a rede U-Net obteve melhores resultados, com quase 1.5 p.p. a mais que a HighResNet, além de ter um desvio padrão menor.

A fim de exemplificação, as imagens 4.1 e 4.2 representam a predição da HighResNet e da U-Net, respectivamente, para uma mesma imagem:

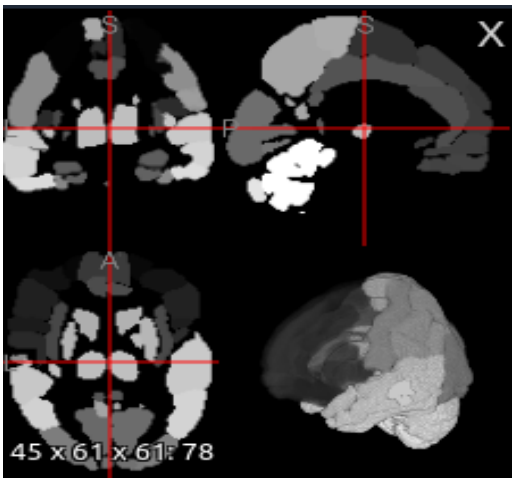

Figura 4.1: Predição da HighResNet

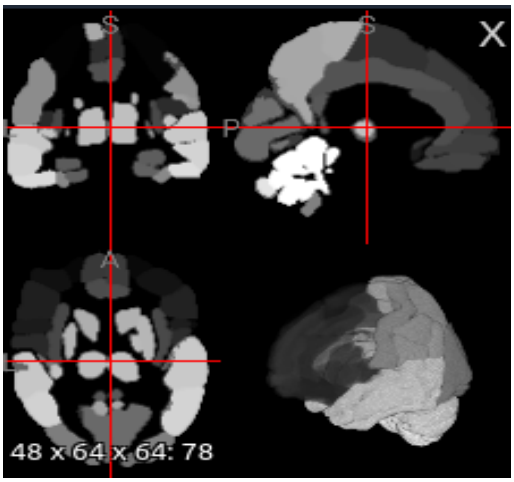

Figura 4.2: Predição da U-Net

Além da média e do desvio padrão, foram calculados o intervalo de confiança de 95% sobre a média do conjunto de dices das predições de cada uma das redes. Os resultados estão presentes na tabela 4.2.

Arquitetura	**Intervalo de confiança**
HighResNet	[80.40, 82.88]
U-Net	[82.04, 84.23]

Tabela 4.2: Intervalo de confiança de predição

É possível observar que a U-Net obteve um intervalo de confiança maior, sendo mais estatisticamente confiável a obter predições melhores que a HighResNet no experimento em questão.

Além do cálculo do dice por imagens, foi realizado o cálculo do dice por *label* para cada um dos modelos, e representados os dices de cada uma das imagens por *label* em um *boxplot*. Primeiramente, na figura 4.3, foram mostrados os cinco *labels* com maiores índices de dice:

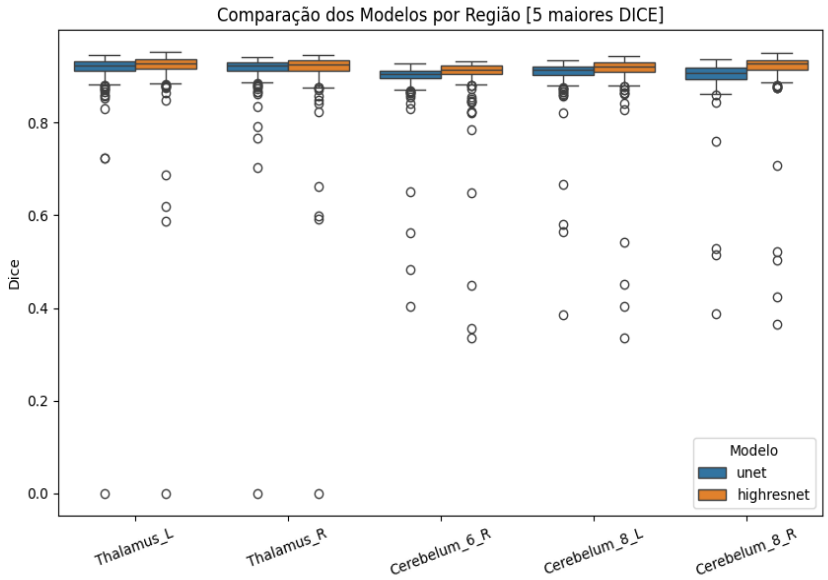

Figura 4.3: Regiões com maiores DICE

Na imagem é possível analisarmos que a rede neural HighResNet possuia uma mediana dos índices de dice maior do que a U-Net, porém esta apresenta mais *outliers* em suas predições. Vale ressaltar houve uma que não se obtou quase nenhuma predição para nenhum rótulo. Isso se deve ao fato de ser uma imagem de um cérebro com quadro clinico avançado de alzheimer, o que fez com que o método de rotulação bronze não fosse capaz de criar um rótulo adequado para ela.

Em seguida, foi evidenciado também os cinco *labels* com menores índices de dice na figura 4.4:

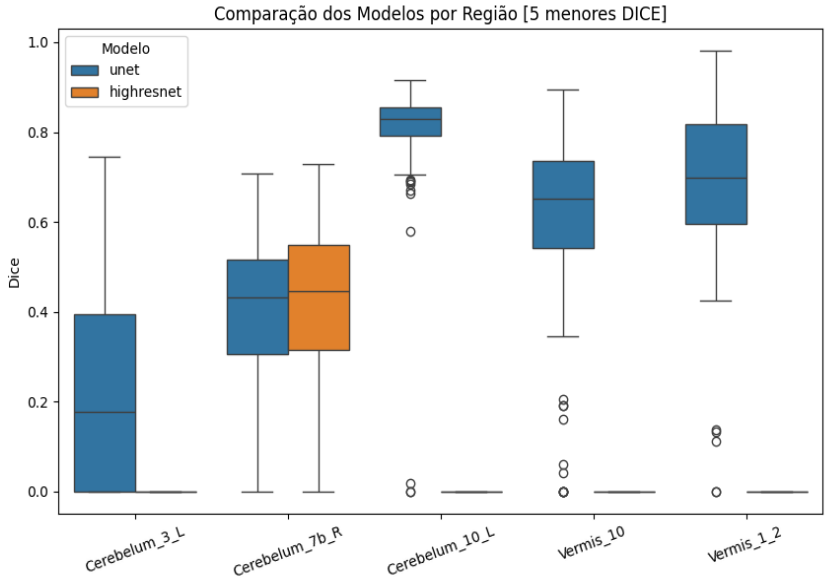

Figura 4.4: Regiões com menores DICE

Para as regiões com menor índice de predição, é possível notar que a U-Net foi capaz de realizar a predição e a HighResNet em quatro dos cinco casos não teve nenhuma predição para as 200 imagens. Isso se deve, possivelmente, ao fato da HighResNet não lidar bem com predição em imagens de pequena escala em comparação com a rede U-Net, a priori.

Por fim, foi calculado o tempo médio de predição médio para cada uma das redes, apresentados na tabela 4.3:

Arquitetura	**Predição média [s]**
HighResNet	0.051 $\pm$0.001
UNet	0.079 $\pm$0.013

Tabela 4.3: Tempos de predição

É notável que a HighResNet possui uma pequena diferença no tempo de predição de imagens, porém, por ser relativamente pequena, seria necessário fazer uma análise mais aprofundada para tirar qualquer conclusão.

4.2 Deploy e utilização do modelo

Após o treinamento e a devida validação do modelo, o mesmo foi disponibilizado de forma fácil e acessível para que possa ser utilizado. Para isso, foi necessário criar uma API para que esse modelo pudesse ser utilizado via requisições.

Para este projeto, foi criada uma API utilizando a biblioteca em python chamada FastAPI. Com ela, é possível criar endpoints com suporte a métodos REST.

No caso de um serviço de parcelamento cerebral, é interessante que a API receba uma imagem (pré-processada ou não) e seja capaz de devolver ao usuário a imagem parcelada, ou seja, a máscara com os rótulos. Para isso, foi desenvolvido então uma API com suporte ao método POST, o qual permite enviar via requisição uma imagem.

A API espera, além da imagem, a indicação de qual modelo ela utilizará para realizar o *brain parcellation*, sendo "highresnet"ou "unet". Além disso, ela pode receber um parâmetro opcional chamado "preprocess", o qual indica que a imagem envia precisa ser pré-processada antes de ser parcelada.

Por fim, quando a API recebe a imagem, ela manda a imagem para o modelo realizar a predição, e devolve posteriormente utilizando a função FileResponse da biblioteca FastAPI.

Com a API devidamente criada e com suporte ao método POST, foi utilizado a aplicação uvicorn para conseguir subir a API e deixar ela disponível para ser consultada. O uvicorn é um servidor ASGI (Asynchronous Server Gateway Interface) de alto desempenho para aplicações web Python, projetado especificamente para frameworks assíncronos como FastAPI, Starlette e Quart. O Uvicorn atua como um servidor web que recebe requisições HTTP, as processa de forma assíncrona e as encaminha para a aplicação ASGI que ele sustenta. A aplicação (neste caso a API), por sua vez, processa a requisição e retorna uma resposta, que é enviada de volta ao cliente.

Com o uvicorn sustentando a API e a deixando disponível para consulta, basta enviar via curl ou postman as requisições para ela e esperar a resposta. Para facilitar o envio de requisições, foi criado um script em shell para fazer requisições estruturas utilizando curl.

```
#!/bin/bash

INPUT_IMAGE=""
OUTPUT_IMAGE=""
PREPROCESS=false

while [[ "$#" -gt 0 ]]; do
    case $1 in
        -i|--input) INPUT_IMAGE="$2"; shift ;;
        -o|--output) OUTPUT_IMAGE="$2"; shift ;;
        -model) MODEL="$2"; shift ;;
        --preprocess) PREPROCESS=true ;;
        *) echo "Op   o desconhecida: $1"; exit 1 ;;
    esac
    shift
done

if [[ -z "$INPUT_IMAGE" || -z "$OUTPUT_IMAGE" ]]; then
    echo "Uso: $0 -i <input_image> -o <output_image> [--
       preprocess]"
    exit 1
fi

curl -X 'POST'  \
  "http://127.0.0.1:8000/predict/?preprocessing=$PREPROCESS&
     model=$MODEL" \
  -H 'accept: application/gzip'  \
  -H 'Content-Type: multipart/form-data' \
  -F "file=@$INPUT_IMAGE" \
  --output "$OUTPUT_IMAGE"

# Exemplo de uso >>>>
# ./predict.sh -i imagem.nii.gz -o resultado.nii.gz --
   preprocess --model highresnet
```

Tal código é executado com "./predict.sh", e possui três tipos de parâmetros:

- **-i**: indica qual o caminho e nome imagem que será enviada para a API
- **-o**: indicada qual será o caminho e o nome da imagem retornada da API
- **-model**: indica qual modelo será utilizado para parcelar a image, U-Net ou HighResnet (unet ou highresnet são os valores que podem ser passados).
- –**preprocess**: este parâmetro é opcional. Caso seja informado, indica à API que a imagem necessita de pré-processamento.

Então, a fim de exemplo, se eu desejo enviar a imagem "image.nii.gz"para o modelo "highresnet"e quero que o retorno tenha como nome "output.nii.gz", e a imagem enviada precisa ser pré-processada, tem-se o seguinte comando para executar o script:

```
./predict.sh -i image.nii.gz -o output.nii.gz --preprocess
                    -model highresnet
```

Assim, é possível disponilizar o modelo como serviço e utilizá-lo de forma fácil e estruturada. Facilmente, tal API poderia ser utilizada em um contêiner docker, por exemplo, e ficar disponível em um servidor aberto à rede, ou em uma plataforma em nuvem.

Capítulo 5

Conclusões

Foi possível executar o objetivo principal do projeto, o qual era utilizar o framework MONAI para criar um pipeline consistente de pré-processamento, treinamento, validação e teste de redes neurais para parcelamento cerebral. Porém, necessitou-se de outras bibliotecas auxiliares para complementar transformações de pré-processamento, as quais foram incorporadas no *pipeline* utilizando a classe *Transform* do MONAI.

Além disso, o *pipeline* deu suporte para que fosse feita a comparação das redes, HighResNet e U-Net, a qual apontou uma leve vantagem em relação a qualidade de predição por parte da U-Net, de aproximadamente 1.5 p.p., possivelmente pelo fato da HighResNet ser uma rede construída para trabalhar com imagens de alta resolução. Para regiões com altos valores de predição, a HighResNet demonstrou possuir uma maior mediana, porém a média caiu por conta de um maior número de *outliers* nestes casos, evidenciado pela figura 4.3.

Apesar da U-Net demorar muito menos em treinamento em relação a HighResNet, aproximadamente 36.69 horas a mais, ambas apresentaram tempos de predição razoavelmente, em média, próximos, sendo necessária uma análise mais aprofundada para tirar uma conclusão mais precisa.

E, além disso, ambas as técnicas acabam sendo muito mais rápidas para realizar o parcelamento cerebral do que o algoritmo de corregistro da biblioteca usada para rotulação das imagens em padrão bronze, sendo pelo menos 196 vezes mais rápidas.

Por fim, foi possível finalizar os objetivos do projeto construindo uma API que fosse capaz de receber requisições e realizar a tarefa de *brain parcellation* como um serviço, simulando um ambiente de utilização de segmentação cerebral de forma eficiente e acessível, capaz de pré-processar imagens se preciso e utilizar dois diferentes tipos de modelo.

Referências Bibliográficas

[1] PATIL, Dinesh D.; DEORE, Sonal G. Medical image segmentation: a review. International Journal of Computer Science and Mobile Computing, v. 2, n. 1, p. 22-27, 2013.

[2] SARRAF, Arman et al. A comprehensive review of deep learning architectures for computer vision applications. American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), v. 77, n. 1, p. 1-29, 2021.

[3] CHLAP, Phillip et al. A review of medical image data augmentation techniques for deep learning applications. Journal of Medical Imaging and Radiation Oncology, v. 65, n. 5, p. 545-563, 2021.

[4] VASUKI, P.; KANIMOZHI, J.; DEVI, M. Balkis. A survey on image preprocessing techniques for diverse fields of medical imagery. In: 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE). IEEE, 2017. p. 1-6.

[5] LEE, Noah; LAINE, Andrew F.; KLEIN, Arno. Towards a deep learning approach to brain parcellation. In: 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro. IEEE, 2011. p. 321-324.

[6] LI, Wenqi et al. On the compactness, efficiency, and representation of 3D convolutional networks: brain parcellation as a pretext task. In: Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings 25. Springer International Publishing, 2017. p. 348-360.

[7] GLASSER, Matthew F. et al. A multi-modal parcellation of human cerebral cortex. Nature, v. 536, n. 7615, p. 171-178, 2016.

[8] OTSUKA, Yoshihisa et al. A multi-atlas label fusion tool for neonatal brain MRI Parcellation and quantification. Journal of Neuroimaging, v. 29, n. 4, p. 431-439, 2019.

[9] TANG, Zhenyu et al. Multi-atlas brain parcellation using squeeze-and-excitation fully convolutional networks. IEEE transactions on image processing, v. 29, p. 6864-6872, 2020.

[10] Associação Python Software Foundation. (2023). Python (versão 3.11) [Software]. Disponível em https://www.python.org/.

[11] MONAI Consortium. MONAI: Medical Open Network for AI. Disponível em: ¡https://monai.io¿. Acesso em: 11 fev. 2025.

[12] SUBRAMANIAN, Adithya et al. NiftyTorch: A Deep Learning framework for NeuroImaging. bioRxiv, p. 2021.02. 26.433116, 2021.

[13] CARDOSO, M. Jorge et al. Monai: An open-source framework for deep learning in healthcare. arXiv preprint arXiv:2211.02701, 2022.

[14] ZOU, Kelly H. et al. Statistical validation of image segmentation quality based on a spatial overlap index1: scientific reports. Academic radiology, v. 11, n. 2, p. 178-189, 2004.

[15] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015. p. 234-241.

[16] MENDRIK, Annemarie M. et al. MRBrainS Challenge: Online evaluation framework for brain image segmentation in 3T MRI scans. *Computers in Biology and Medicine*, v. 70, p. 60-75, 2015.

[17] MILLETARI, Fausto; NAVAB, Nassir; AHMADI, Seyed-Ahmad. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, p. 565-571, 2016.

[18] KINGMA, Diederik P.; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] MICIKEVICIUS, Paulius et al. Mixed Precision Training. arXiv preprint arXiv:1710.03740, 2017. Disponível em: ¡https://arxiv.org/abs/1710.03740¿. Acesso em: 11 fev. 2025.