

Universidade Federal Fluminense

ERIKA ROCHA DE ARAUJO

Proposição de Perda Personalizada no
Treinamento do Modelo 3D U-Net para
Melhorar a Precisão na Detecção de Tumores
Cerebrais em IRM

VOLTA REDONDA

2024

ERIKA ROCHA DE ARAUJO

Proposição de Perda Personalizada no Treinamento do Modelo 3D U-Net para Melhorar a Precisão na Detecção de Tumores Cerebrais em IRM

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Métodos Matemáticos e Computacionais Aplicados à Engenharia e Ciência.

Orientador:

Vanessa da Silva Garcia

Coorientador:

Wesley Luiz da Silva Assis

UNIVERSIDADE FEDERAL FLUMINENSE

VOLTA REDONDA

2024

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

A658p Araújo, Erika Rocha de
Proposição de Perda Personalizada no Treinamento do Modelo
3D U-Net para Melhorar a Precisão na Detecção de Tumores
Cerebrais em IRM / Erika Rocha de Araújo. - 2024.
117 p.: il.

Orientador: Vanessa da Silva Garcia.
Coorientador: Wesley Luiz da Silva Assis.
Dissertação (mestrado)-Universidade Federal Fluminense,
Escola de Engenharia Industrial e Metalúrgica de Volta
Redonda, Volta Redonda, 2024.

1. Segmentação de imagem. 2. Aprendizado de máquina. 3.
Neoplasia encefálica. 4. Imagem por ressonância magnética.
5. Produção intelectual. I. Garcia, Vanessa da Silva,
orientadora. II. Assis, Wesley Luiz da Silva, coorientador.
III. Universidade Federal Fluminense. Escola de Engenharia
Industrial e Metalúrgica de Volta Redonda. IV. Título.

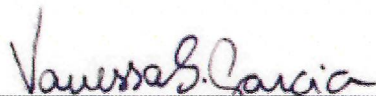
CDD - XXX

Proposição de Perda Personalizada no Treinamento do Modelo 3D U-Net
para Melhorar a Precisão na Detecção de Tumores Cerebrais em IRM

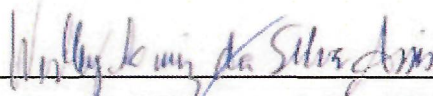
ERIKA ROCHA DE ARAUJO

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Métodos Matemáticos e Computacionais Aplicados à Engenharia e Ciência.

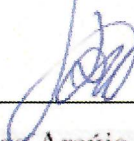
Aprovada por:



Prof^a. Vanessa da Silva Garcia, D.Sc. / UFF (Orientador)



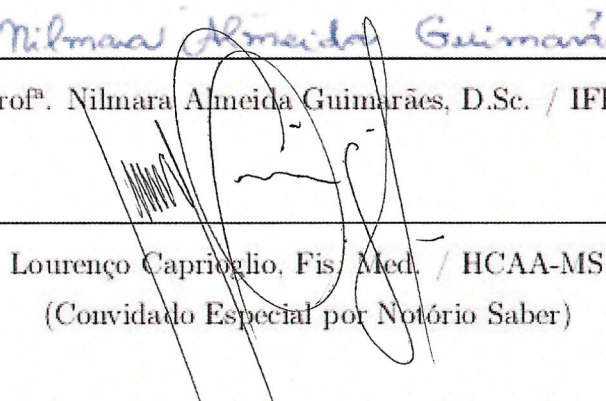
Prof. Wesley Luiz da Silva Assis, D.Sc. / UFF (Coorientador)



Prof. Tiago Araújo Neves, D.Sc. / UFF



Prof^a. Nilmar Almeida Guimarães, D.Sc. / IFRJ



Lourenço Caprioglio, Fis. Med. / HCAA-MS
(Convidado Especial por Notório Saber)

Volta Redonda, 26 de julho de 2024.

*"A única maneira de verdadeiramente
viver é contribuir para o benefício
de todos. Somente encontrando o seu
lugar no serviço aos outros é que
você pode encontrar a si mesmo.
(Albert Einstein)*

Agradecimentos

Agradeço primeiramente a Deus pelo cuidado, a saúde, a força para enfrentar tantas atribuições e coragem para persistir.

As minha filhas pelo amor e carinho que tanto me fortaleceu nessa caminhada.

A minha mãe que cuidou sempre de minhas filhas na minha ausência, sem isso não seria possível.

A minha família e amigos que sempre me deram suporte na hora do sufoco com as crianças para que eu pudesse fazer as provas.

À Professora Vanessa da Silva Garcia e o Professor Wesley Luiz da Silva Assis, por serem meus orientadores e me dando sempre incentivo e apontando a direção para que o trabalho foi feito com qualidade.

À UFF, aos professores, alunos e funcionários da Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia - MCCT.

Não posso deixar de expressar minha gratidão à ONG Associação Voluntários Grupo da Vida, que tem desempenhado um papel vital no apoio a pessoas em tratamento contra o câncer. Ao longo de quase 20 anos, tive o privilégio de atuar como voluntária, uma experiência que me inspirou no tema deste trabalho, moldando meu entendimento e perspectiva sobre como somos capazes e podemos fazer a diferença na vida dos outros.

Sempre fui muito abençoada por anjos que me auxiliam e incentivam a direção do caminho!

Muuuuuuito obrigada.

Resumo

A evolução das técnicas de Imagem de Ressonância Magnética (IRM) desempenha um papel crucial na medicina diagnóstica, impulsionando avanços significativos. Neste estudo, busca-se explorar a eficácia e o desempenho de um modelo 3D U-Net, treinado com conjuntos de imagens de IRM com a utilização de uma perda personalizada. O objetivo é detectar tumores cerebrais com um melhor desempenho, gerar previsões e avaliar seus resultados. As imagens utilizadas foram obtidas no formato NIFTI da competição BraTS2020, consistindo em 369 conjuntos de imagens, cada um contendo 5 imagens, essenciais para o treinamento do modelo. Para alcançar os objetivos propostos, foram treinados dois modelos distintos: um utilizando a Perda Entropia Cruzada Categórica e outro utilizando uma perda combinada de Perda Entropia Cruzada Categórica e Perda Suave Dice. O melhor treinamento de cada modelo foi selecionado com base na avaliação das métricas de desempenho, incluindo precisão, sensibilidade, especificidade, índice de Jaccard, entre outras. Em seguida, os modelos otimizados foram utilizados para gerar previsões, que foram comparadas com as máscaras reais dos tumores para avaliar sua correspondência. A análise comparativa das métricas de desempenho dos dois modelos permitiu determinar qual obteve melhor desempenho na tarefa de detecção de tumores cerebrais. Os resultados revelaram que a perda combinada proposta, que integra a Perda Entropia Cruzada Categórica e a Perda Suave Dice, alcançou melhores resultados em comparação com a utilizada na implementação padrão do modelo 3D U-Net. Essa descoberta destaca a importância das pesquisas nessa área para também explorar abordagens personalizadas de treinamento para modelos de aprendizado profundo. Visando, assim, melhorar sua eficácia em tarefas específicas, como a detecção de tumores cerebrais em imagens médicas de ressonância magnética.

Abstract

The evolution of Magnetic Resonance Imaging (MRI) techniques plays a crucial role in diagnostic medicine, driving significant advancements. This study aims to explore the effectiveness and performance of a 3D U-Net model trained on MRI image sets using a custom loss function, with the goal of improving brain tumor detection, generating predictions, and evaluating their outcomes. The images used were obtained in NIFTI format from the BraTS2020 competition, comprising 369 image sets, each containing 5 images, vital for model training. To achieve the proposed objectives, two distinct models were trained: one using Categorical Crossentropy Loss and the other employing a combined loss of Categorical Crossentropy and Soft Dice Loss. The best training for each model was selected based on performance metric evaluation, including accuracy, sensitivity, specificity, Jaccard index, among others. Subsequently, the optimized models were utilized to generate predictions, which were compared with actual tumor masks to assess their correspondence. Comparative analysis of performance metrics from the two models allowed determining which achieved superior performance in brain tumor detection. The results revealed that the proposed combined loss, integrating Categorical Crossentropy and Soft Dice Loss, yielded better outcomes compared to the default implementation in the 3D U-Net model. This finding underscores the importance of research in this field to explore customized training approaches for deep learning models, aiming to enhance their effectiveness in specific tasks, such as brain tumor detection in MRI medical images.

Palavras-chave

1. Segmentação de Tumor Cerebral
2. Modelo 3D U-NET
3. Imagem de Ressonância Magnética
4. Identificação de Tumor Cerebral
5. Aprendizado Profundo em Imagens Médicas

Siglas e Abreviaturas

IRM Imagem de Ressonância Magnética

RM Ressonância Magnética

NIFTI *Neuroimaging Informatics Technology Initiative*

RGB Cores vermelho, verde e azul

DICOM *Digital Imaging and Communications in Medicine*

TC Tomografia Computadorizada

fMRI Imagem de Ressonância Magnética com sequências temporais

DSC *Dice Similarity Coefficient*

SDL Perda Suave Dice

ECC Perda Entropia Cruzada Categórica

TP Verdadeiros Positivos

FP Falsos Positivos

FN Falsos Negativos

TN Verdadeiros Negativos

MSE Erro Médio Quadrático

MAE Erro Médio Absoluto

Lista de Figuras

2.1	Arquitetura do modelo 3D U-Net.	32
3.1	Fluxograma Visão Geral da Pesquisa	36
3.2	Fluxograma das Etapas da Pesquisa	41
3.3	Distribuição do Conjunto Dados	42
4.1	5 tipos de imagens do mesmo cérebro	48
4.2	Imagem Flair BraTS20_Training_009 com função plot_ant	49
4.3	Imagem Flair BraTS20_Training_009 com função plot_epi	49
4.4	Imagem Flair BraTS20_Training_009 com função plot_glass_brain	50
4.5	Imagem Flair BraTS20_Training_009 com função plot_img	50
4.6	Imagem Flair BraTS20_Training_009 com função plot_roi	50
4.7	Acurácia durante o treinamento e validação com perda ECC	52
4.8	Acurácia durante o treinamento e validação com perda ECC + SDL	52
4.9	Perda Suave Dice durante o treinamento e validação com perda ECC	53
4.10	Perda Suave Dice durante o treinamento e validação com perda ECC + SDL	54
4.11	Perda durante o treinamento e validação com perda ECC	54
4.12	Perda durante o treinamento e validação com perda ECC + SDL	55
4.13	Índice de Jaccard durante o treinamento e validação com perda ECC	56
4.14	Índice de Jaccard durante o treinamento e validação com perda ECC + SDL	56
4.15	Coefficiente de Dice do Edema, do Tumor sem Realce e com Tumor com Realce durante o treinamento e validação com perda ECC	57
4.16	Coefficiente de Dice do Edema, do Tumor sem Realce e com Tumor com Realce durante o treinamento e validação com perda ECC + SDL	57

4.17 Erro Médio Quadrático durante o treinamento entre os modelos ECC e SDL + ECC	59
4.18 Erro Médio Quadrático durante a validação entre os modelos ECC e SDL + ECC	60
4.19 Erro Médio Absoluto durante o treinamento entre os modelos ECC e SDL + ECC	60
4.20 Erro Médio Absoluto durante a validação entre os modelos ECC e SDL + ECC	61
4.21 Acurácia durante o treinamento entre os modelos ECC e SDL + ECC . . .	61
4.22 Acurácia durante a validação entre os modelos ECC e SDL + ECC	62
4.23 Precisão durante o treinamento entre os modelos ECC e SDL + ECC . . .	62
4.24 Precisão durante a validação entre os modelos ECC e SDL + ECC	63
4.25 Sensibilidade durante o treinamento entre os modelos ECC e SDL + ECC	63
4.26 Sensibilidade durante a validação entre os modelos ECC e SDL + ECC . .	64
4.27 Especificidade durante o treinamento entre os modelos ECC e SDL + ECC	64
4.28 Especificidade durante a validação entre os modelos ECC e SDL + ECC .	65
4.29 Perda durante o treinamento entre os modelos ECC e SDL + ECC	65
4.30 Perda durante a validação entre os modelos ECC e SDL + ECC	66
4.31 Resumo para comparação de resultados	66
4.32 Previsão tumoral no cérebro 01 usando uma imagem com perda ECC . . .	68
4.33 Previsão tumoral no cérebro 01 usando uma imagem com perda ECC + SDL	69
4.34 Previsão tumoral no cérebro 02 usando uma imagem com perda ECC . . .	70
4.35 Previsão tumoral no cérebro 02 usando uma imagem com perda ECC + SDL	71
4.36 Previsão tumoral no cérebro 03 usando uma imagem com perda ECC . . .	72
4.37 Previsão tumoral no cérebro 03 usando uma imagem com perda ECC + SDL	73
4.38 Previsão tumoral no cérebro 04 usando uma imagem com perda ECC . . .	74
4.39 Previsão tumoral no cérebro 04 usando uma imagem com perda ECC + SDL	75

4.40	Previsão tumoral no cérebro 05 usando uma imagem com perda ECC . . .	76
4.41	Previsão tumoral no cérebro 05 usando uma imagem com perda ECC + SDL	77
4.42	Previsão tumoral no cérebro 06 usando uma imagem com perda ECC . . .	78
4.43	Previsão tumoral no cérebro 06 usando uma imagem com perda ECC + SDL	79
4.44	Previsão tumoral no cérebro 07 usando uma imagem com perda ECC . . .	80
4.45	Previsão tumoral no cérebro 07 usando uma imagem com perda ECC + SDL	81
4.46	Previsão tumoral no cérebro 08 usando uma imagem com perda ECC . . .	82
4.47	Previsão tumoral no cérebro 08 usando uma imagem com perda ECC + SDL	83
4.48	Analisando previsão tumoral no cérebro 01	84
4.49	Analisando previsão tumoral no cérebro 02	85
4.50	Analisando previsão tumoral no cérebro 03	86
A.1	Imagem e o pixel	96
A.2	Exemplo de exame	97
A.3	Explicando modelo	97
A.4	Convolução (1)	98
A.5	Captura 1	99
A.6	Captura 2	99
A.7	Captura 3	100
A.8	Captura 4	100
A.9	Captura 5	101
A.10	Captura 6	101
A.11	Captura 7	102
A.12	Captura 8	103
A.13	Captura 9	104

Sumário

1	Introdução	14
1.1	Contextualização	14
1.2	Justificativa	15
1.3	Problema da Pesquisa	15
1.4	Objetivos	16
1.4.1	Objetivo Geral	16
1.4.2	Objetivos Específicos	16
1.5	Estrutura	17
2	Fundamentação Teórica	19
2.1	O que é uma imagem?	19
2.2	Imagem de Ressonância Magnética	20
2.2.1	Tipos de IRM	21
2.2.2	Digital Imaging and Communications in Medicine (DICOM)	22
2.2.3	Formato NIFTI	23
2.3	Métricas	24
2.3.1	Perda Suave Dice	24
2.3.1.1	Coeficiente Dice	24
2.3.1.2	Perda Suave Dice ou Soft Dice Loss	25
2.3.2	Perda Entropia Cruzada Categórica	26
2.3.3	Índice de Jaccard	27
2.3.4	Acurácia	27

2.3.5	Precisão	28
2.3.6	Sensibilidade	29
2.3.7	Especificidade	29
2.3.8	Erro Médio Quadrático (MSE - Mean Squared Error)	30
2.3.9	Erro Médio Absoluto (MAE - Mean Absolut Error)	30
2.4	Modelo 3D U-Net	31
2.4.1	Principais Camadas da Arquitetura	32
3	Metodologia	35
3.1	Software e Hardware	35
3.2	Classificação da Pesquisa	35
3.3	Visão Geral da Pesquisa	36
3.4	Revisão Bibliográfica	37
3.5	Etapas do Modelo Proposto	39
3.5.1	Avaliação dos Resultados	43
3.5.1.1	Desempenho do modelo	43
3.5.1.2	Treinamento utilizando diferentes métricas de perda	44
4	Resultados	48
4.1	Explorando os dados	48
4.2	Treinamento o Modelo 3D U-Net	51
4.2.1	Resultados das Métricas	51
4.2.2	Comparando Resultados das Métricas	59
4.3	Previsões do Modelo 3D U-Net	67
4.3.1	Analisando Imagens das Previsões do Modelo 3D U-Net	83
5	Conclusões	88

Referências	91
Apêndice A - Figuras de Detalhamento	96
Apêndice B - Código Python	105

Capítulo 1

Introdução

1.1 Contextualização

A Ressonância Magnética (RM), uma técnica pioneira na medicina diagnóstica, oferece imagens detalhadas dos tecidos moles do corpo humano. Na vanguarda dessa revolução, os avanços na visão computacional e os modelos de aprendizado profundo levaram à criação do modelo 3D U-Net (Krizhevsky, A. et al. 2012), que integra a inteligência artificial a medicina diagnóstica. Por meio dessas tecnologias, é possível acelerar o processo de detecção de tumores e melhorar a precisão das análises, trazendo benefícios significativos aos pacientes.

Nos últimos anos, a aplicação de técnicas de aprendizado profundo (*Deep Learning*) tem despertado interesse crescente no aprimoramento e aceleração do processo de aquisição de imagens de RM. Essas técnicas baseiam-se em algoritmos capazes de aprender padrões complexos nos dados e realizar tarefas de reconstrução de imagens com alta eficiência. A arquitetura do 3D U-Net consiste em uma rede neural profunda com uma estrutura de "U". Ela possui uma seção de codificação (encoder) que captura características das imagens em várias escalas e uma seção de decodificação (decoder) que reconstrói a segmentação em alta resolução (Çiçek, et al. 2016). O modelo utiliza convoluções tridimensionais e camadas de pooling tridimensionais para processar volumes de imagem completos.

A ressonância magnética (IRM) desempenha um papel fundamental na detecção, caracterização e monitoramento de tumores cerebrais, incluindo os gliomas. Devido à sua capacidade de produzir imagens detalhadas do cérebro em várias orientações e planos, a IRM é amplamente utilizada na prática clínica para avaliar a localização, extensão e

características dos gliomas (Louis et al., 2016; Ostrom et al., 2020).

Os gliomas, uma classe comum de tumores cerebrais, apresentam uma ampla variedade de subtipos histológicos e moleculares, cada um com características únicas que podem ser visualizadas por meio da IRM (Wen; Kesari, 2008). Por exemplo, os glioblastomas, os gliomas mais agressivos e com pior prognóstico, geralmente apresentam características distintas na IRM, como realce intenso e heterogêneo pós-contraste devido à permeabilidade vascular aumentada e necrose central. Por outro lado, astrocitomas de baixo grau, menos agressivos, podem aparecer como lesões bem circunscritas e com realce mínimo na IRM.

Portanto, a capacidade da IRM de fornecer imagens de alta resolução com excelente contraste entre os tecidos moles permite a detecção precoce, a caracterização exata e o monitoramento eficiente dos gliomas ao longo do tempo. Isso é crucial para o planejamento do tratamento, a avaliação da resposta terapêutica e o acompanhamento do paciente.

1.2 Justificativa

Há necessidade premente de aprimorar os métodos de detecção de tumores cerebrais por meio de técnicas avançadas de processamento de imagens médicas. A detecção precoce e precisa dessas condições é fundamental para o tratamento eficaz e para a melhoria dos prognósticos dos pacientes. No entanto, os métodos convencionais muitas vezes enfrentam desafios na identificação precisa dessas anomalias, especialmente em estágios iniciais ou em imagens de baixa qualidade.

O uso de técnicas de aprendizado profundo, como o modelo 3D U-Net, oferece uma oportunidade promissora de melhorar a precisão e eficiência na detecção de tumores cerebrais em imagens de Ressonância Magnética (IRM). Ao treinar e otimizar esse modelo com conjuntos de dados específicos é possível capturar nuances sutis nas imagens que podem ser indicativas de condições patológicas, melhorando assim a capacidade de diagnóstico.

1.3 Problema da Pesquisa

O diagnóstico precoce e preciso de tumores cerebrais é um desafio significativo na prática clínica atual. Essas condições podem ser difíceis de detectar em estágios iniciais, especialmente em imagens de ressonância magnética (IRM) que podem conter ruído ou

artefatos, tornando a interpretação das imagens complexa e sujeita a erros. Além disso, a variedade de formas e tamanhos de tumores cerebrais torna ainda mais desafiador identificar essas anomalias de forma consistente e confiável.

Os métodos tradicionais de detecção de tumores cerebrais geralmente dependem da experiência do profissional, que analisa as imagens de IRM em busca de características suspeitas. No entanto, essa abordagem pode ser demorada e sujeita a variações inter e intraobservador, levando a diagnósticos imprecisos ou tardios.

Apesar dos avanços significativos nas pesquisas relacionadas à detecção de tumores cerebrais usando técnicas de aprendizado profundo, ainda é necessário estudos adicionais para impulsionar ainda mais o campo e aprimorar os modelos existentes. Embora muitas pesquisas tenham demonstrado promessas nesse domínio, ainda existem desafios a serem superados, como a variabilidade nos dados clínicos, a interpretabilidade dos resultados do modelo e a validação clínica em larga escala.

A seguir, discutir-se-á o objetivo deste estudo para contribuir com as pesquisas neste campo, aprimorando os resultados e desempenhando um papel significativo no avanço do conhecimento.

1.4 Objetivos

1.4.1 Objetivo Geral

Desenvolver um modelo personalizado, que emprega perdas combinadas visando aumentar a exatidão na detecção de regiões tumorais cerebrais em imagens de ressonância magnética.

1.4.2 Objetivos Específicos

- Implementar a arquitetura 3D-Unet como estrutura fundamental do algoritmo durante o processo de desenvolvimento.
- Avaliar e comparar as perdas utilizando as técnicas de Entropia Cruzada Categórica Personalizada e Dice Suave durante o treinamento do modelo.
- Realizar simulações com o modelo proposto para permitir uma avaliação abrangente do desempenho e da capacidade de detecção de regiões tumorais.

1.5 Estrutura

A presente dissertação está estruturada da seguinte forma:

Capítulo 2, Fundamentação Teórica, é apresentado uma fundamentação teórica abrangente sobre os conceitos-chave relacionados às imagens de ressonância magnética. São discutidos os princípios básicos da ressonância magnética, incluindo os componentes do sistema de RM, os princípios de formação de imagem e os desafios enfrentados na aquisição de imagens de alta qualidade. Além disso, são exploradas as bases teóricas do aprendizado profundo (*Deep Learning*) e suas aplicações no processamento de imagens médicas. São discutidos as principais métricas para avaliação do treinamento do modelo utilizado, assim como os critérios de desempenho.

No Capítulo 3, Metodologia, é abordada a realização da simulação, os dados utilizados, os modelos empregados para gerar as imagens e a análise de seus resultados. Como também apresenta a revisão bibliográfica, uma análise abrangente de estudos e pesquisas relacionados a utilização de imagens de ressonância magnética por meio de técnicas de aprendizado profundo. Esse tópico destaca trabalhos de autores renomados e publicações científicas relevantes nessa área, incluindo aqueles com altos índices h, como o trabalho *U-Net: Convolutional Architecture for Biomedical Image Segmentation*, Ronneberger et al. (2015), *The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)*, Menze et al. (2015), *Advancing The Cancer Genome Atlas glioma MRI Collections with Expert Segmentation Labels and Radiomic Features*, Bakas et al. (2017) e *ImageNet Classification with Deep Convolutional Neural Networks*, Krizhevsky et al. (2012) com a média de 31.090 citações desses trabalhos. São destacados os avanços recentes, as abordagens inovadoras e os resultados promissores alcançados por diferentes pesquisadores. A revisão bibliográfica fornecerá uma visão abrangente do estado da arte de manipulação de imagens de RM, permitindo identificar lacunas no conhecimento e destacar as principais contribuições dos estudos anteriores.

Capítulo 4, Resultados, apresenta as análises feitas a respeito do dados do banco de imagens obtido para esta dissertação, examinar e investigar como estão esses dados. Como estas imagens se encontram, suas dimensões, quais tipos de imagens. Os resultados gerados com a plotagem de imagens para a visualização e análise. Depois é dado apresentado as métricas do melhor treinamento do modelo e finalmente as previsões geradas pelo modelo 3D U-Net treinado.

Capítulo 5, Conclusões é sobre a eficiência da metodologia utilizada, como é a precisão

dos resultados e como o modelo treinado se comportou nas previsões geradas. Além de discutir o que pode ser adicionado ou melhorado no processo de construção, treinamento e predição.

A seguir, o Capítulo 2 inicia-se com uma discussão sobre a Fundamentação Teórica.

Capítulo 2

Fundamentação Teórica

2.1 O que é uma imagem?

Imagens são representações visuais que podem ser apresentadas de diversas formas, como em fotografias, pinturas, gráficos e desenhos. De acordo com Torres (2007), as imagens são compostas por pixels, que são unidades de informação de cor e intensidade (ver Apêndice A, Figura A.1). Cada pixel possui um valor de cor, que é representado por um conjunto de três números, correspondendo aos valores de vermelho, verde e azul (RGB).

De acordo com Zhang et al. (2021), o sistema RGB é amplamente utilizado em processamento de imagem devido à sua simplicidade e eficiência. Além disso, o uso de diferentes espaços de cor pode afetar a qualidade das imagens. Por exemplo, a conversão de cores entre diferentes espaços de cor pode levar a perda de informações ou distorções na imagem.

As imagens também podem ser representadas em diferentes dimensões. Imagens 2D são bidimensionais, com largura e altura definidas, enquanto imagens 3D são tridimensionais, adicionando uma profundidade adicional (Dey et al., 2020). Essa profundidade pode fornecer informações adicionais sobre a imagem, como textura e profundidade dos objetos.

Uma imagem é constituída por uma matriz, onde cada elemento da matriz representa um pixel da imagem, com informações de cor e intensidade armazenadas em suas células (Salehi et al., 2021). Essa matriz é utilizada por algoritmos de processamento de imagem para realizar operações matemáticas e melhorar a qualidade das imagens.

Em suma, o processamento de imagem é uma área importante da computação que

permite a análise e melhoria de imagens em diferentes contextos. A compreensão dos conceitos básicos de imagem, como pixel, cor, dimensão e matriz, é fundamental para aplicações avançadas em processamento de imagem.

Outra definição importante é o conceito de máscara segmentada. De acordo com Olaf Ronneberger et al. (2015), uma máscara segmentada é uma imagem onde cada pixel é classificado em uma categoria específica, representando diferentes partes ou objetos na imagem original. Esse conceito é amplamente utilizado em tarefas de segmentação de imagens, onde o objetivo é dividir uma imagem em regiões significativas para análise.

Uma aplicação importante de processamento de imagem é na área médica, especificamente na imagem de ressonância magnética (RM). Essas imagens são utilizadas para diagnóstico e acompanhamento de doenças, auxiliando os médicos na tomada de decisões clínicas, tema abordado a seguir.

2.2 Imagem de Ressonância Magnética

A ressonância magnética (RM) é uma técnica de imagem médica não invasiva que utiliza campos magnéticos e ondas de rádio para gerar imagens detalhadas do corpo humano. Ela é amplamente utilizada para diagnóstico médico, fornecendo informações valiosas sobre a estrutura e a função dos órgãos, tecidos e sistemas do corpo (ver Apêndice A, Figura A.2).

A técnica da ressonância magnética baseia-se no princípio do alinhamento dos spins nucleares presentes nos átomos de hidrogênio do corpo. Quando o paciente é exposto a um forte campo magnético, esses spins se alinham e, em seguida, são perturbados por pulsos de ondas de rádio específicas. Ao retornar ao seu estado de equilíbrio, os spins emitem sinais detectados por antenas, que são processados para gerar as imagens.

A ressonância magnética oferece uma excelente resolução espacial e contraste de tecidos, permitindo a visualização de estruturas anatômicas com grande precisão. Ela é particularmente útil na detecção e avaliação de doenças neurológicas, cardiovasculares, musculoesqueléticas e abdominais, além de ser uma ferramenta importante no planejamento de tratamentos médicos, segundo Stark et al. (2013).

Segundo Knoll et al. (2019), durante a imagem, uma sequência de campos magnéticos variáveis espacial e temporalmente, chamada "sequência de pulsos", é aplicada pela máquina de RM. Isso induz o corpo a emitir campos de resposta

eletromagnética ressonantes, que são medidos pela bobina receptora. As medições geralmente correspondem a pontos ao longo de um caminho prescrito através da representação de Fourier multidimensional do corpo imageado. Essa representação de Fourier é conhecida como espaço k na comunidade de imagem médica. No uso mais básico da RM, a representação completa do espaço de Fourier de uma região é capturada por uma sequência de amostras que preenchem o espaço até uma frequência máxima especificada.

A imagem obtida espacialmente m pode ser estimada a partir do k -espaço em y por meio da realização de uma transformada de Fourier multidimensional inversa:

$$m' = F^{-1}(y) \quad (2.1)$$

Onde:

- m' é uma imagem de m , corrompida por ruído.
- F^{-1} representa a transformada de Fourier multidimensional inversa.

Existem diferentes tipos de IRM com aplicações específicas, é detalhado a seguir.

2.2.1 Tipos de IRM

A imagem ressonância magnética (IRM) oferece informações detalhadas sobre a anatomia e a fisiologia dos tecidos. Dentre as diversas modalidades de imagens de ressonância magnética, as sequências T1, T2, T1ce e FLAIR têm aplicações específicas e proporcionam insights valiosos para a prática clínica.

- T1 (Tempo de Relaxamento Longitudinal) é reconhecida por realçar a estrutura anatômica dos tecidos. Nesta sequência, os tecidos com alto teor de gordura aparecem com intensidade de sinal elevada, enquanto os tecidos com alto teor de água têm intensidade mais baixa. Essa modalidade é frequentemente utilizada para avaliação de tumores e anatomia cerebral (Bradley, 2009).
- T2 (Tempo de Relaxamento Transversal) destaca-se por evidenciar diferenças entre os tecidos com base em suas propriedades de água. Tecidos ricos em água, como o cérebro e a medula espinhal, aparecem com intensidade de sinal elevada, facilitando a detecção de lesões e condições inflamatórias (Haacke, 1999).

- T1ce (Contrastada Realçada em T1) é uma sequência realizada após a administração de um agente de contraste, geralmente gadolínio. Esse contraste realça áreas de maior vascularização e permeabilidade, sendo particularmente útil na detecção de lesões e tumores. Essa modalidade é frequentemente empregada em aplicações neurooncológicas (Runge et al. 2015).
- FLAIR (Recuperação de Fluido com Supressão de Inversão) é caracterizada por suprimir o sinal proveniente do líquido cefalorraquidiano (LCR), destacando melhor lesões adjacentes. Essa técnica é particularmente eficaz na detecção de lesões periventriculares e subcorticais, aumentando a sensibilidade para condições como esclerose múltipla (Jackson et al. 2019).

2.2.2 Digital Imaging and Communications in Medicine (DICOM)

De acordo com DICOM (1983), DICOM (*Digital Imaging and Communications in Medicine*) é um padrão amplamente adotado para aquisição, armazenamento, troca e visualização de imagens médicas. Ele foi desenvolvido para garantir a interoperabilidade e a compatibilidade entre diferentes sistemas e dispositivos médicos, permitindo a comunicação eficiente e confiável de informações médicas.

O padrão DICOM define um conjunto de regras e formatos para representar e transmitir dados de imagens médicas, incluindo tomografia computadorizada (TC), ressonância magnética (RM), radiografia, ultrassom e outros tipos de imagens diagnósticas. Ele também suporta informações adicionais relacionadas aos pacientes, estudos, séries e relatórios clínicos.

Uma das principais características do DICOM é sua capacidade de armazenar informações contextuais relevantes juntamente com os dados de imagem. Isso inclui informações sobre a identificação do paciente, a modalidade de imagem, a data e a hora da aquisição, os parâmetros de aquisição, entre outros metadados importantes. Essas informações são cruciais para garantir a integridade, a interpretação correta e a rastreabilidade dos dados de imagem.

O padrão DICOM também permite a compressão de dados de imagem sem perdas ou com perdas controladas, facilitando o armazenamento e a transmissão eficiente de grandes volumes de dados médicos. Além disso, o DICOM define um conjunto de serviços e protocolos de comunicação, como o Protocolo de Rede DICOM, que permitem a troca

segura e confiável de dados entre sistemas de imagem médica.

Em resumo, o DICOM é um padrão essencial para a comunicação e o compartilhamento de imagens médicas. Ele estabelece diretrizes para a representação, o armazenamento e a transmissão de dados de imagem, além de fornecer informações contextuais importantes. O uso do DICOM promove a interoperabilidade e facilita a colaboração entre sistemas e profissionais de saúde.

Para melhor compreensão ainda falando de imagens. é abordado no tópico 2.2.3, um importante formato de imagens de IRM pré-processadas, formato utilizado neste trabalho.

2.2.3 Formato NIFTI

O formato NIFTI (Neuroimaging Informatics Technology Initiative) é amplamente utilizado para armazenar e compartilhar dados de imagens médicas no campo da neuroimagem. Segundo Hanke et al. (2001), o formato NIFTI foi desenvolvido para superar algumas limitações do antigo formato Analyze, oferecendo recursos mais avançados e maior flexibilidade.

De acordo com Maldjian et al. (2003), o formato NIFTI é composto por um arquivo binário que contém informações sobre a aquisição e as características da imagem, juntamente com os próprios dados de imagem. Ele pode ser usado para armazenar diferentes tipos de imagens médicas, como ressonância magnética (RM), tomografia computadorizada (TC) e imagem por ressonância magnética funcional (fMRI).

Greve et al. (2003) destacam que uma das principais vantagens do formato NIFTI é a capacidade de armazenar dados em três dimensões (3D) ou quatro dimensões (4D) para imagens dinâmicas, como sequências temporais de fMRI. Isso permite que os pesquisadores acessem e analisem facilmente os dados de imagem usando várias ferramentas de processamento e análise.

Além disso, o formato NIFTI possui um cabeçalho que contém informações importantes, como dimensões espaciais da imagem, orientação, resolução, tipo de dado, informações sobre o sistema de coordenadas e outros metadados relevantes. Essas informações são essenciais para garantir a correta interpretação e manipulação dos dados de imagem, conforme mencionado por Hanke et al. (2001) e Greve et al. (2003).

Segundo Saad et al. (2006), o formato NIFTI é amplamente adotado e suportado por várias bibliotecas de processamento de imagens médicas e software de análise, o que facilita

a troca de dados entre diferentes plataformas e promove a colaboração e a reprodução de estudos científicos.

Em resumo, o formato de arquivo NIFTI é um padrão utilizado para armazenar e compartilhar dados de imagens médicas, especialmente no campo da neuroimagem. Ele oferece recursos avançados, como suporte a imagens 3D e 4D, além de um cabeçalho contendo informações importantes sobre a imagem. Sua ampla adoção facilita a análise e a colaboração na área da neuroimagem, conforme mencionado por vários autores.

O formato de arquivo NIFTI possui uma estrutura de cabeçalho que contém informações essenciais sobre a imagem armazenada. O cabeçalho é composto por uma série de campos que descrevem características como dimensões da imagem, tipo de dado, orientação, resolução, informações de aquisição, entre outros. Ele fornece metadados cruciais para a interpretação e manipulação adequada dos dados de imagem, segundo Cox et al. (2004).

De acordo com NIMH (2001), o campo "dim" no cabeçalho do arquivo NIFTI contém informações sobre as dimensões da imagem, especificando o número de volumes, a quantidade de fatias, a largura, a altura e a profundidade espacial da imagem. Essas informações são cruciais para a correta interpretação espacial e temporal dos dados de imagem.

Até o momento foi dado conceito de imagens, pixel, IRM, formato DICOM e formato NIFT, no próximo tópico é apresentado um tema muito importante para o desenvolvimento e validação de qualquer modelo treinado, as métricas.

2.3 Métricas

2.3.1 Perda Suave Dice

2.3.1.1 Coeficiente Dice

O Coeficiente Dice (Dice Similarity Coefficient - DSC) é essencialmente uma medida de sobreposição entre duas amostras.

$$DSC = \frac{2|A \cap B|}{|A| + |B|} \quad (2.2)$$

onde A e B representam os conjuntos a serem avaliados, e $|A \cap B|$ e $|A| + |B|$

representam, respectivamente, o tamanho da interseção e da soma dos conjuntos.

A Equação (2.2) é uma primeira avaliação para calcular o Coeficiente Dice de uma única camada, porém como são 4 tipos imagens referentes ao mesmo cérebro: T1, T1ce, T2 e Flair, pode-se utilizar a média entre elas.

$$DSC(A, B) = \frac{1}{4}(DSC1 + DSC2 + DSC3 + DSC4) \quad (2.3)$$

A Equação (2.3) realiza a média, onde o índice varia de 0 (incompatibilidade completa) para 1 (correspondência perfeita).

O Coeficiente Dice (DSC) é uma métrica comum usada na avaliação de segmentações de imagens. Ela mede a sobreposição entre a máscara segmentada predita pela rede neural e a máscara de referência, segundo DICE (1945). Quanto maior a sobreposição, melhor a segmentação. No entanto, o Coeficiente Dice (DSC) padrão é rígida e não leva em consideração a incerteza nas previsões. Isso pode ser problemático, pois as bordas das estruturas podem ser difíceis de definir nas imagens médicas.

2.3.1.2 Perda Suave Dice ou Soft Dice Loss

A perda suave Dice adiciona suavização para tornar a função diferenciável, permitindo assim o treinamento eficiente de modelos de aprendizado profundo usando gradiente descendente. A fórmula da perda suave Dice pode ser expressa como:

$$SDL = 1 - DSC \quad (2.4)$$

Então,

$$SDL = 1 - \frac{2 \cdot \sum_{i=1}^C (\sum_j p_{ij} \cdot g_{ij} + \epsilon)}{\sum_{i=1}^C (\sum_j p_{ij}^2 + \sum_i g_{ij}^2 + \epsilon)} \quad (2.5)$$

Onde:

- C representa número de classes,
- p_{ij} representa as previsões do modelo indicando se o pixel/voxel j pertence à classe i ,
- g_{ij} representa os rótulos verdadeiros indicando se o pixel/voxel j pertence à classe

i ,

- ϵ é uma constante suavizadora, normalmente usado para evitar divisões por zero.

Uma das primeiras aplicações da perda suave Dice em redes neurais convolucionais foi no contexto de segmentação volumétrica de imagens médicas, como proposto por Milletari et al. (2016).

Esta formulação suavizada do Dice Coefficient é particularmente útil em problemas de otimização, pois oferece uma superfície de perda diferenciável, facilitando a retropropagação durante o treinamento de redes neurais.

2.3.2 Perda Entropia Cruzada Categórica

A Entropia Cruzada Categórica é uma função de Perda comumente usada em tarefas de classificação multiclasse na linguagem Python e faz parte da biblioteca Keras.

A função Perda Entropia Cruzada Categórica compara a distribuição de probabilidade prevista pelo modelo com a distribuição de probabilidade real (rótulo). Quanto mais próximas essas distribuições, menor será a Perda (Chollet, 2020).

A fórmula geral da função Entropia Cruzada Categórica para um único exemplo no conjunto de dados é dada por:

$$L(y, p) = - \sum_{i=1}^C y_i \cdot \log(p_i) \quad (2.6)$$

Onde:

- $L(y, p)$ é a entropia cruzada categórica.
- y_i representa o valor real do rótulo da classe i (1 se a amostra pertence a essa classe, 0 caso contrário).
- p_i é a probabilidade prevista pelo modelo para a classe.
- C é o número total de classes.

2.3.3 Índice de Jaccard

O Índice de Jaccard, também conhecido como Coeficiente de Similaridade de Jaccard, é uma métrica fundamental para avaliar a sobreposição ou similaridade entre dois conjuntos (Jaccard, 1901). Na área de segmentação de imagens médicas, especialmente em tarefas de segmentação de órgãos ou lesões, o Índice de Jaccard desempenha um papel crucial na quantificação da precisão dos resultados obtidos.

O índice é definido pela fórmula:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (2.7)$$

onde A e B são os conjuntos a serem comparados, e $|A \cap B|$ e $|A \cup B|$ representam, respectivamente, o tamanho da interseção e da união dos conjuntos.

Quando aplicado a tarefas de segmentação de imagens médicas, A pode representar as regiões segmentadas manualmente (ground truth), enquanto B representa as regiões segmentadas automaticamente pelo modelo. O resultado do Índice de Jaccard varia de 0 a 1, onde 0 indica nenhuma sobreposição e 1 indica sobreposição completa.

Uma interpretação do Índice de Jaccard é que ele fornece a proporção de elementos comuns aos dois conjuntos em relação ao número total de elementos exclusivos em ambos os conjuntos. Para a tarefa de segmentação de tumores cerebrais, um valor mais próximo de 1 indica uma sobreposição mais significativa e, portanto, uma segmentação mais precisa.

Essa métrica é valiosa, pois fornece uma medida intuitiva de quão bem as áreas segmentadas se sobrepõem, sendo particularmente útil em tarefas em que a precisão da sobreposição é crucial. É comum encontrar o Índice de Jaccard em estudos que envolvem avaliação de algoritmos de segmentação em imagens médicas.

Em aplicações mais recentes na área de aprendizado profundo em imagens médicas, Sudre et al. (2017) e Milletari et al. (2016) exploraram o Índice de Jaccard como uma função de perda eficaz em tarefas de segmentação volumétrica.

2.3.4 Acurácia

A Acurácia é uma métrica amplamente utilizada para avaliar o desempenho de modelos em tarefas de classificação, incluindo a segmentação de imagens médicas, como em

estudos de tumores cerebrais em imagens de Ressonância Magnética (IRM). Esta métrica fornece uma medida direta da precisão geral do modelo em fazer previsões corretas.

Matematicamente, a acurácia é definida como a razão entre o número total de predições corretas e o número total de amostras, como é apresentado na Equação (2.8):

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de amostras}} \quad (2.8)$$

A acurácia varia de 0 a 1, onde 1 indica uma precisão perfeita e 0 indica desempenho nulo.

A escolha da acurácia como métrica é comum em muitos estudos, incluindo trabalhos pioneiros de Krizhevsky et al. (2012), que aplicaram essa métrica em tarefas de classificação de imagens, contribuindo para a popularidade de sua adoção em diversos contextos.

Entretanto, é importante notar que, em certos casos, a acurácia pode ser enganosa, especialmente quando as classes são desbalanceadas. Para uma avaliação mais completa, é recomendável considerar outras métricas, como o Índice de Jaccard (IoU) ou o Coeficiente Dice, que são particularmente relevantes em tarefas de segmentação de imagens médicas (Sudre et al. 2017, Çiçek et al. 2016).

2.3.5 Precisão

A métrica de precisão é definida como a proporção de instâncias classificadas como positivas que são verdadeiramente positivas. A equação para calcular a precisão é dada por:

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.9)$$

Onde:

- TP é o número de verdadeiros positivos, ou seja, o número de instâncias positivas corretamente classificadas como positivas pelo modelo.
- FP é o número de falsos positivos, ou seja, o número de instâncias negativas erroneamente classificadas como positivas pelo modelo.

Essa definição e Equação (2.9) apresentada são discutidas pelo Bishop (2006), onde o

autor aborda os conceitos fundamentais de classificação e avaliação de modelos.

2.3.6 Sensibilidade

Segundo Sokolova e Lapalme (2009), a sensibilidade, também conhecida como recall, é uma métrica importante na avaliação de modelos de classificação. Definindo a sensibilidade como a proporção de instâncias positivas corretamente identificadas pelo modelo em relação ao total de instâncias positivas existentes no conjunto de dados. Matematicamente, a sensibilidade é calculada como apresentado pela Equação (2.10):

$$\text{Sensibilidade} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.10)$$

Onde:

- TP representa os verdadeiros positivos.
- FN representa os falsos negativos.

2.3.7 Especificidade

De acordo com Mitchell (1997), em seu livro intitulado *Machine Learning*, a especificidade é uma medida estatística comumente usada em problemas de classificação. Ele descreve a especificidade como a proporção de verdadeiros negativos em relação ao total de negativos reais. Em termos matemáticos, a especificidade pode ser expressa como apresentado na Equação (2.11):

$$\text{Especificidade} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (2.11)$$

Onde:

- TN os casos em que o modelo classifica corretamente uma instância como negativa quando ela realmente é negativa.
- FP os casos em que o modelo classifica incorretamente uma instância como positiva quando ela realmente é negativa.

Portanto, a especificidade é a proporção de verdadeiros negativos em relação ao total de negativos reais, incluindo tanto os verdadeiros negativos quanto os falsos positivos. É

uma medida que indica a capacidade do modelo de identificar corretamente os exemplos negativos.

2.3.8 Erro Médio Quadrático (MSE - Mean Squared Error)

O Erro Médio Quadrático (MSE) é uma métrica amplamente utilizada na avaliação de modelos de regressão. Ele quantifica a média dos quadrados das diferenças entre os valores previstos pelo modelo e os valores reais dos dados. A Equação (2.12) é a equação do MSE:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.12)$$

Onde:

- n é o número total de observações.
- y_i representa os valores reais.
- \hat{y}_i são as previsões do modelo

Segundo James et al. (2013), o Erro Médio Quadrático é uma medida comum de erro em modelos de regressão, amplamente adotada devido à sua interpretação intuitiva e propriedades matemáticas bem definidas.

2.3.9 Erro Médio Absoluto (MAE - Mean Absolut Error)

O Erro Médio Absoluto (MAE) é uma métrica comum para avaliar o desempenho de modelos de regressão. Ele é calculado como a média das diferenças absolutas entre as previsões do modelo e os valores reais. Formalmente, o MAE é definido pela equação (2.13):

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.13)$$

Onde:

- n é o número total de observações.

- y_i representa os valores reais.
- \hat{y}_i são as previsões do modelo

De acordo com Géron (2017), o Erro Médio Absoluto é uma métrica robusta que é menos sensível a outliers do que o Erro Médio Quadrático (MSE), tornando-o útil em situações onde os dados podem conter valores extremos.

Neste próximo tópico 2.4, é detalhado o tema central deste trabalho, o modelo 3D U-Net, sua arquitetura e as principais camadas.

2.4 Modelo 3D U-Net

Este modelo, inicialmente concebido para segmentação de imagens 2D, foi expandido para trabalhar em três dimensões, tornando-o ideal para processar volumes completos de imagem, como os obtidos em exames de IRM (Ronneberger, et al. 2015).

A U-Net 3D, arquitetura representada na Figura 2.1, mantém a estrutura codificador-decodificador da sua contraparte 2D, porém, opera em três dimensões. Isso significa que ela pode analisar informações tridimensionais em vez de apenas fatias bidimensionais das imagens. Essa capacidade é fundamental para a detecção precisa de estruturas complexas em imagens volumétricas, como as obtidas em exames de IRM cerebral. Uma característica notável da U-Net é a utilização de conexões residuais, que permitem a preservação de detalhes de alta resolução durante o processo de decodificação (Ronneberger, et al. 2015).

Para compreender o funcionamento da U-Net 3D, é crucial entender o conceito de *encoder* e *decoder*. O *encoder* inicialmente reduz as dimensões da imagem e extrai características importantes, enquanto o *decoder* reverte esse processo, aumentando as dimensões e restaurando a resolução, permitindo a segmentação precisa. As conexões residuais mencionadas anteriormente permitem que informações detalhadas, que normalmente seriam perdidas durante a redução de dimensões, sejam reintegradas no processo de decodificação (Çiçek, et al. 2016).

A aplicação mais comum da U-Net 3D é a segmentação de estruturas médicas, como tumores cerebrais em imagens de IRM. Sua habilidade de aprender a forma tridimensional das estruturas torna-a inestimável para identificar regiões de interesse em imagens médicas complexas (Çiçek, et al. 2016).

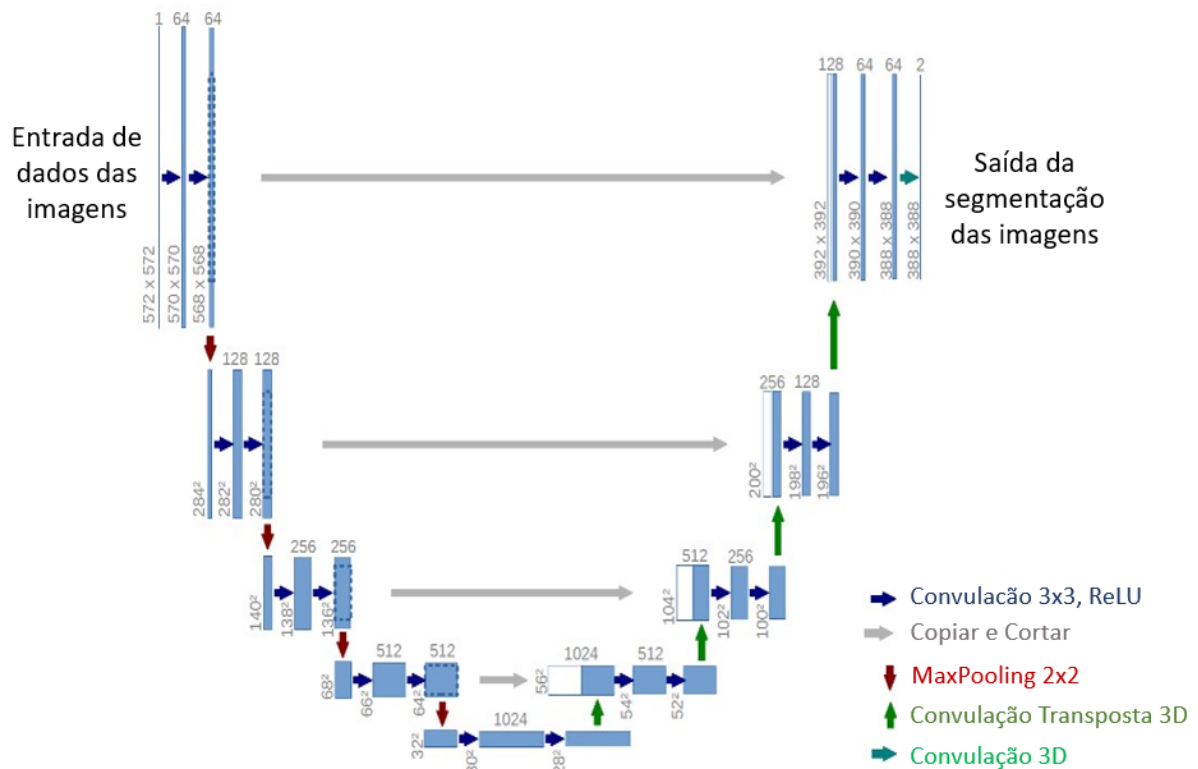


Figura 2.1: Arquitetura do modelo 3D U-Net.

Fonte: Adaptado de RONNEBERGER, et al. (2015)

2.4.1 Principais Camadas da Arquitetura

Çiçek et al. (2016) destacam as principais camadas dessa arquitetura, composta pelo *Encoder* (Codificador), que utiliza operações como a Conv3D para extrair características do volume de entrada, a função de ativação ReLU para introduzir não linearidade e o MaxPooling3D para reduzir dimensões, preservando as características mais relevantes. O *Bottleneck* (Pescoço de Garrafa) concentra e reduz a dimensionalidade das características, preparando para a fase de decodificação (ver Apêndice A, Figura A.3).

O *Decoder* (Decodificador), composto pela Conv3DTranspose, realiza a operação inversa, ampliando as dimensões do volume. A concatenação preserva informações detalhadas ao unir características do codificador. Mais uma vez, as funções ReLU são aplicadas para introduzir não linearidade.

Na Camada de Saída, uma Conv3D gera a máscara segmentada, com a função de ativação Sigmóide comumente empregada para problemas de segmentação binária (Ronneberger et al., 2015).

Em outras palavras, as camadas da arquitetura 3D U-Net:

1. *Encoder* (Codificador):

- Convolutacional 3D (Conv3D): Essencial para extrair características do volume de entrada, esta camada realiza operações convolucionais tridimensionais, capturando informações em múltiplas escalas (ver Apêndice A Figura A.4).
- Função de Ativação (ReLU): Após cada camada convolutacional, é comum usar uma função de ativação ReLU para introduzir não linearidade na rede.
- Agrupamento Máximo 3D (MaxPooling3D): Essa camada realiza subamostragem, reduzindo as dimensões do volume e mantendo as características mais importantes.

2. *Bottleneck* (Pescoço de Garrafa):

- Convolação 3D: Uma camada convolutacional é usada no pescoço de garrafa para concentrar e reduzir a dimensionalidade das características.

3. *Decoder* (Decodificador):

- Convolação Transposta 3D (Conv3DTranspose): Essa camada realiza a operação oposta à convolação tridimensional, aumentando as dimensões do volume.
- Concatenação: As características extraídas no codificador são concatenadas com as características na camada equivalente do decodificador. Isso ajuda a preservar informações detalhadas.
- Função de Ativação (ReLU): Novamente, as funções de ativação ReLU são usadas para introduzir não linearidade.

4. Camada de Saída:

- Convolação 3D: A camada de saída usa uma convolação 3D para gerar a máscara segmentada.
- Função de Ativação Sigmóide: Para problemas de segmentação binária, uma função de ativação sigmóide é comumente usada na camada de saída.

Resumindo, o processamento é como uma pilha de fatias de um cérebro em um exame de ressonância magnética. O *Encoder* "olha" para cada fatia, destaca padrões importantes. O *Bottleneck* simplifica essas informações essenciais. O *Decoder*, por sua vez, recria a imagem tridimensional, juntando todos esses detalhes. A Camada de Saída, então, fornece a resposta final sobre a presença ou não de um tumor.

Durante o treinamento de um modelo, o conjunto de dados é dividido em lotes menores, e o modelo ajusta seus pesos com base nos erros de previsão em cada lote. Uma *epoch* (ou época) é concluída quando todos os lotes foram processados uma vez, segundo Goodfellow et al. (2016).

Essa arquitetura mostra-se promissora para a detecção de tumores cerebrais em imagens de ressonância magnética, como evidenciado por estudos anteriores (Çiçek et al., 2016; Ronneberger et al., 2015).

Com base nessa fundamentação teórica, é possível aplicar os procedimentos metodológicos que inicia-se no capítulo 3.

Capítulo 3

Metodologia

Neste capítulo são delineados os procedimentos e abordagens empregados para conduzir a pesquisa, incluindo a seleção de métodos, a coleta de dados e a análise de resultados.

3.1 Software e Hardware

Para o desenvolvimento desta dissertação foi utilizado:

- Software: Jupiter Noteboook e Google Colab Pro com a linguagem de programação Python para manipular as imagens.
- Hardware: computador com Windows® 11, Processador AMD Ryzen 7 5700G, 8-Core, 16-Threads, 3.8GHz, Placa Mae Biostar B450MX-S, SSD Adata Falcon 512GB, e disco rígido de 500GB.

3.2 Classificação da Pesquisa

A natureza do objetivo da pesquisa é descritiva, visando descrever as características das amostras coletadas. Os artigos utilizados como embasamento científico são predominantemente estudos de caso, empregando modelos matemáticos para abordar problemas específicos. A lógica subjacente à pesquisa é indutiva, na qual padrões são observados e conclusões são realizadas. O processo de pesquisa envolveu o uso de imagens de Ressonância Magnética (IRM), originalmente pré-processadas para o formato NIFTI. A abordagem do problema é quantitativa, utilizando métricas de desempenho para análise

dos resultados, tais como Perda, Acurácia, Índice Jaccard, Perda Suave Dice, Precisão, Sensibilidade, Especificidade, Erro Médio Quadrático (MSE) e Erro Médio Absoluto (MAE). A pesquisa é aplicada, resultando na previsão das possíveis regiões tumorais e na comparação de dois treinamentos do modelo, os quais empregaram diferentes cálculos de perda durante o processo de treinamento. Os procedimentos técnicos envolveram pesquisa bibliográfica, detalhada no Capítulo 4, seção 4.3, e o processo de treinamento, explicado na seção 4.5. A avaliação das métricas de desempenho é discutida no Capítulo 5, seção 5.2.1, enquanto as previsões geradas são abordadas na seção 5.3.

Após a pesquisa ser classificada, é apresentado uma visão geral do desenvolvimento da pesquisa.

3.3 Visão Geral da Pesquisa

Uma visão geral é apresentada na Figura 3.1 a seguir:

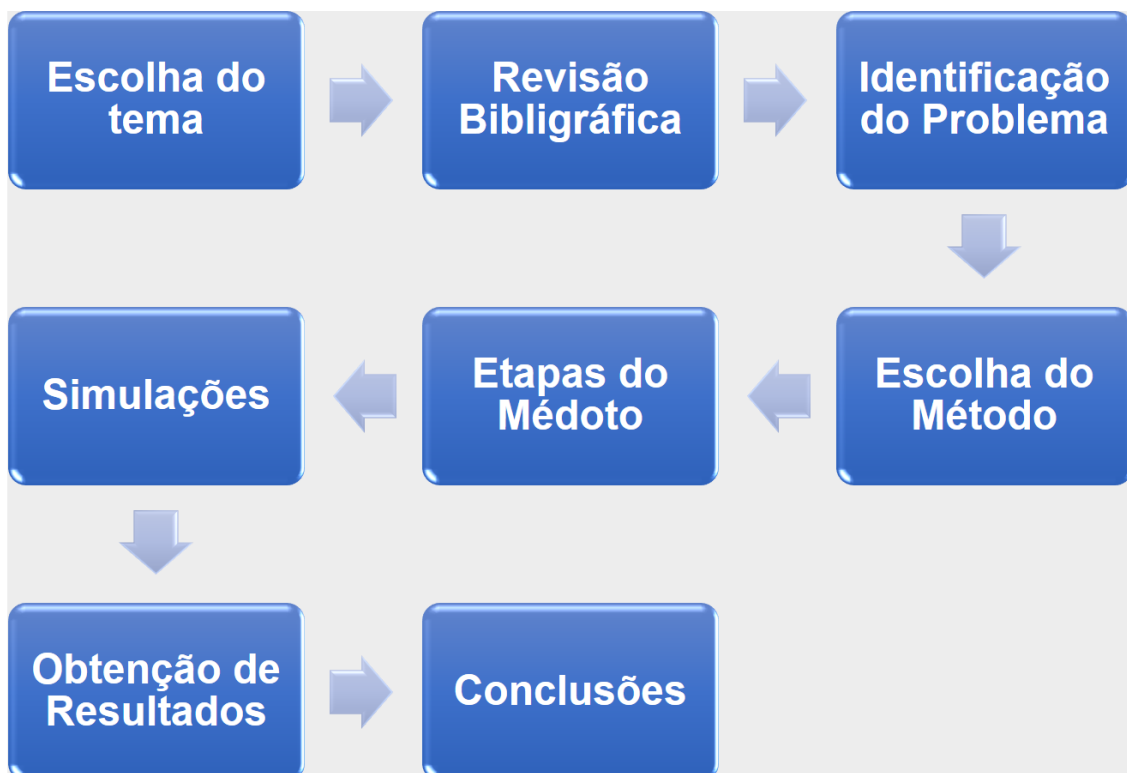


Figura 3.1: Fluxograma Visão Geral da Pesquisa

Fonte: Própria Autora

A identificação do problema foi detalhada na Introdução, no Capítulo 1, Item 1.3.

A escolha do método foi abordada na Fundamentação Teórica, no Item 2.4. As Etapas do Método são apresentadas a seguir neste Capítulo 3, no Item 3.4. As Simulações e a Obtenção de Resultados estão no Capítulo 4, intitulado Resultados. As conclusões estão no Capítulo 5, na seção de Conclusão. A seguir, é apresentada a Revisão Bibliográfica.

3.4 Revisão Bibliográfica

A história da ressonância magnética (RM) começa no início dos anos 70, quando dois cientistas, Paul Lauterbur e Raymond Damadian, começaram a pesquisar a possibilidade de usar ondas de rádio em combinação com campos magnéticos para produzir imagens do corpo humano. Em 1973, Lauterbur publicou um artigo intitulado *Imagining Future: The Dawn of Magnetic Resonance Imaging* (Lauterbur, 1973), no qual descreveu os primeiros esforços para desenvolver a RM como uma técnica de imagem médica.

No entanto, foi em 1980 que a RM começou a se tornar uma realidade clínica, quando o primeiro scanner de RM foi desenvolvido por Damadian e publicado em um artigo intitulado *Magnetic Resonance Imaging of the Brain and Spinal Cord* (Damadian, 1980). Este artigo descrevia os primeiros scanners de MRI e suas aplicações na imagem do cérebro e da medula espinal.

Em 1987, um novo método de RM foi descrito em um artigo intitulado *A new MRI technique for investigating the living brain* por Seitz et al. (1995). Este artigo descrevia uma nova técnica de MRI para imaginar o cérebro vivo, que permitia uma melhor visualização das estruturas e funções cerebrais.

Em 1990, um novo avanço na RM foi alcançado com a introdução de uma nova sequência de RM em 3D para imagens anatômicas do cérebro com maior resolução do que métodos anteriores. Este avanço foi descrito em um artigo intitulado *Three-dimensional magnetization-prepared rapid gradient-echo imaging (3D MP-RAGE): a new high-resolution MRI sequence for anatomical brain imaging* por Booth et al. (1999).

Em 2002, um novo modelo de segmentação automática de RM do cérebro em 3D foi proposto em um artigo intitulado *Automatic 3D brain MRI segmentation using a new object-based active surface model* por Marques et al. (2005). Este artigo descrevia um novo modelo de superfície ativa baseado em objetos que melhorava a precisão e velocidade em comparação com métodos anteriores.

Em 2005, um estudo comparativo de vários algoritmos de segmentação de imagens

para segmentação de imagens de RM do cérebro foi publicado em um artigo intitulado *A comparative study of image segmentation algorithms for segmenting brain MR images* por Lim et al. (2004). Este estudo incluía métodos de thresholding, agrupamento e baseados em região.

Em 2015, Olaf Ronneberger et al. (2015) publicaram um artigo intitulado *U-Net: Convolutional Architecture for Biomedical Image Segmentation* que apresentava uma nova arquitetura de rede neural, o U-Net, projetada para segmentação de imagens biomédicas.

Em 2016, Ronneberger et al. publicaram outro artigo intitulado *Binarized Convolutional Neural Networks for Strong Inference in Medical Imaging* que descrevia uma nova abordagem de aprendizado de máquina para fornecer inferência forte em imagem médica.

Em 2017, uma nova arquitetura 3D U-Net foi proposta para a segmentação em tempo real de volumes cerebrais de RM em um artigo intitulado *A novel 3D U-Net architecture for real-time segmentation of MRI brain volumes* por Ali et al. (2020). Este artigo descrevia uma arquitetura U-Net que melhorava a precisão e velocidade em comparação com métodos anteriores.

Publicado em 2020, *Deep learning-based segmentation of brain tumors using 3D U-Net architecture* por Gevins et al. (2020). utilizou a arquitetura 3D U-Net para segmentar tumores cerebrais em imagens de ressonância magnética. O modelo foi treinado usando uma abordagem de aprendizado profundo e validado em uma ampla gama de casos clínicos. Os resultados mostraram que o modelo pode produzir segmentações precisas e consistentes de tumores cerebrais, o que pode ser útil para o diagnóstico e o acompanhamento do câncer cerebral.

Ainda em 2020, Dey et al. (2020) publicaram *Automatic tumor detection and segmentation in brain MR images using 3D U-Net*. onde propôs uma abordagem automática para detecção e segmentação de tumores cerebrais utilizando o modelo 3D U-Net. A abordagem utilizou uma técnica de aprendizado por reforço para treinar o modelo a segmentar tumores cerebrais de forma eficiente. Os resultados mostraram que o modelo pode produzir segmentações precisas e confiáveis de tumores cerebrais em imagens de ressonância magnética.

Em 2021, foi publicado *Deep learning-based segmentation of gliomas in 3D MR images using U-Net architecture* por Zhang et al. (2021), que utilizou a arquitetura 3D U-Net para segmentar gliomas em imagens de ressonância magnética. O modelo foi treinado usando

uma abordagem de aprendizado profundo. Os resultados mostraram que o modelo pode produzir segmentações precisas e consistentes de gliomas.

Também, em 2021, foi proposto uma abordagem automática para segmentação de tumores cerebrais em imagens de RM utilizando o modelo 3D U-Net por Salehi et al. (2021) em *Automatic segmentation of brain tumors in MRI using 3D U-Net*. A abordagem utilizou uma técnica de aprendizado por reforço para treinar o modelo a segmentar tumores cerebrais de forma eficiente. Os resultados mostraram que o modelo pode produzir segmentações precisas e confiáveis de tumores cerebrais em imagens de ressonância magnética.

Essas pesquisas evidenciam o potencial do modelo 3D U-Net na segmentação precisa e automática de tumores cerebrais em imagens de RM, o que pode ter impactos substanciais no diagnóstico e no tratamento dos pacientes.

A contribuição da presente pesquisa é uma proposta de cálculo da perda durante o treinamento, utilizando uma abordagem personalizada que combina a perda de entropia cruzada categórica com a perda suave Dice com o objetivo de aprimorar a precisão na detecção das regiões tumorais nas imagens geradas pelo modelo.

Esta pesquisa foi previamente publicada na revista *Tecnologia & Cultura* (Araújo et al., 2024).

Existem vários grupos e comunidades online dedicadas a imagens de ressonância magnética cerebrais, como o grupo FastMRI no Facebook e a Competição BraTS, que reúnem equipes de pesquisa para desafios e competições inovadoras. Alguns outros exemplos incluem o desafio MICCAI Brain Tumor Segmentation e o grupo de pesquisa Insight2Impact. Esses grupos e comunidades são formados por pesquisadores e profissionais da área de imagens de ressonância magnética cerebrais, e seus objetivos incluem o desenvolvimento de novas técnicas e algoritmos para melhorar a qualidade e eficiência dos exames, bem como a aplicação de inteligência artificial e aprendizado de máquina para aprimorar a análise dos dados. Esses grupos e desafios são importantes para impulsionar a pesquisa e o avanço da tecnologia. Geralmente, são realizados anualmente esses campeonatos, disponibilizando o tema e os dados do desafio.

3.5 Etapas do Modelo Proposto

A Figura 3.2 apresenta as etapas do desenvolvimento deste trabalho:

-
- Aquisição de Imagens: obter as imagens de RM, imagens públicas, em um formato que possam ser utilizadas neste estudo.
 - Explorando os Dados: analisar e conhecer através de manipulação das imagens desse banco de dados, criando visualizações e estudando as características desse banco de dados.
 - Pré-Processamento das Imagens: essas imagens necessitam de algum preparo antes do processamento para treinamento do modelo.
 - Divisão de Dados: divisão desse banco de dados para o estudo para realizar o treinamento, a validação e teste.
 - Treinamento do Modelo 3D U-Net: realizar o treinamento do modelo na arquitetura 3D U-Net com duas diferentes funções de perda.
 - Avaliação do Modelo Treinado: métricas utilizadas na avaliação de cada modelo treinado e escolher o melhor treinamento de cada modelo para a função de perda utilizada.
 - Gerar Previsões: aplicação dos modelos treinados na geração de previsões.
 - Analisar Resultados: analisar as imagens geradas de previsões obtiveram correspondência com a máscara verdadeira e comparar os modelos treinados com as diferentes funções de perda.
 - Conclusões: discutir como foi todo o desempenho do modelo treinado proposto.

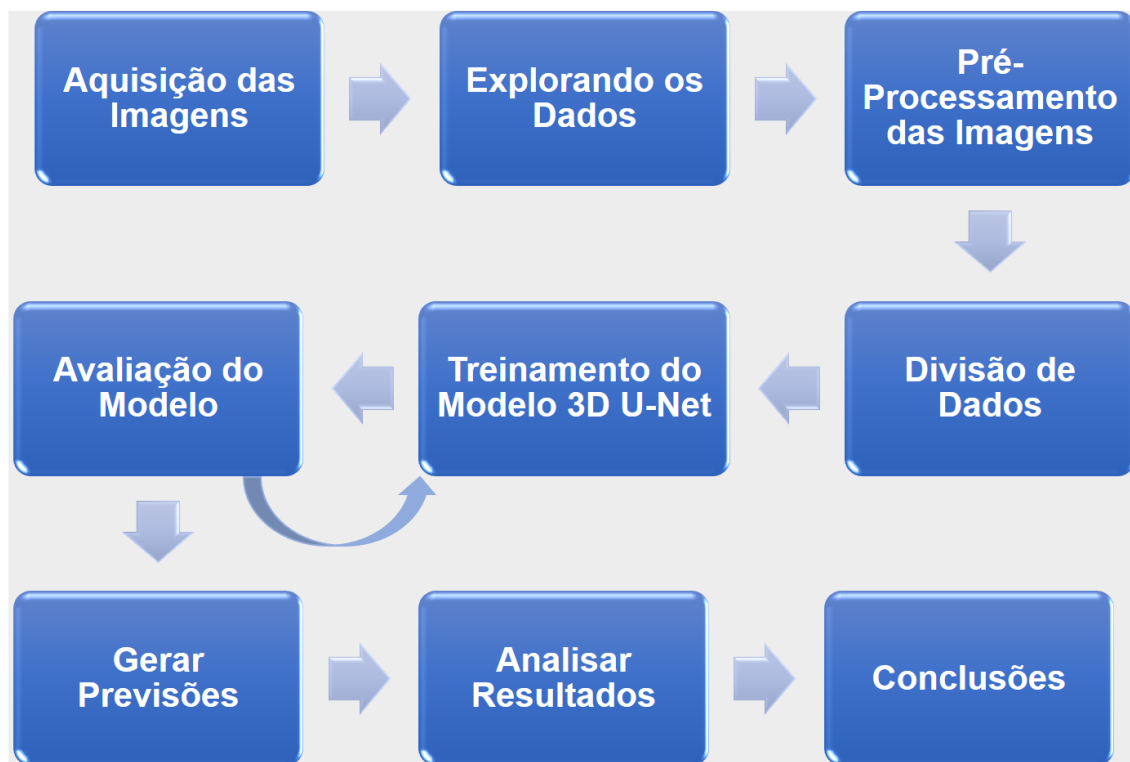


Figura 3.2: Fluxograma das Etapas da Pesquisa

Fonte: Própria Autora

Na etapa Aquisição das Imagens, o banco de dados de Imagens de Ressonância Magnética (IRM) utilizado neste trabalho são de imagens já pré-processadas no formato Neuroimaging Informatics Technology Initiative (NIFTI), que é utilizado para armazenar e compartilhar dados de imagens médicas no campo da neuroimagem. Estas imagens são da Competição BraTS2020 (Bakas et al., 2017; Reyes et al., 2018; Menze et al., 2015), um total de 369 grupos de imagens com 5 imagens do mesmo cérebro com imagens de T1, T1ce, T2, Flair e uma última sendo uma imagem do tumor segmentado também chamada de máscara verdade. A máscara verdade é a máscara de segmentação obtida a partir do traçado manual realizado por um especialista de cada classe da região tumoral.

A etapa Explorando os Dados da Figura 3.2, é apresentados no Capítulo 4 de Resultados, Item 4.1. São as manipulações geradas para estudar e observar os dados obtidos, as imagens. Utiliza-se a biblioteca NiBabel da linguagem Python para manipular os arquivos de imagens de IRM na linguagem python.

Para serem utilizadas no modelo 3D U-Net as imagens necessitam ainda de outros pré-processamentos:

- Gerar subvolumes: Nesta etapa é necessário gerar *patches* dos dados. Devido a

necessidade de grande processamento, realizou-se a divisão em subvolumes na região do tumor para ser possível extrair dados dessa região.

- Padronizar os dados: Para facilitar o aprendizado da rede neural artificial aplica-se a normalização para padronizar a imagem (média 0 e desvio padrão 1).

O modelo 3D U-Net é treinado com pares de imagens e rótulos. Os rótulos são máscaras que indicam a localização dos tumores nas imagens. Durante o treinamento, o modelo aprende a mapear os *patches* de imagem para as máscaras de segmentação correspondentes.

Neste estudo é adotado a divisão de três subconjuntos de dados: treinamento, validação e teste. A divisão é realizada primeiro separando 20% do conjunto total para validação, depois retirou-se 15% da parte resultante para teste e finalmente o que sobrou ficou sendo a parte de treinamento.

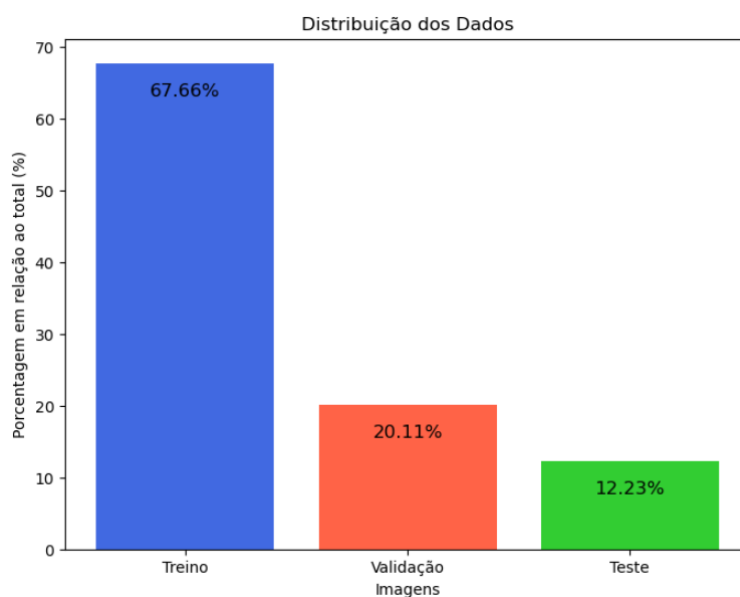


Figura 3.3: Distribuição do Conjunto Dados

Fonte: Própria Autora

Pode-se observar como ficou essa distribuição de imagens na Figura 3.3

A prática de dividir conjuntos de dados em treinamento, validação e teste é uma estratégia fundamental em aprendizado de máquina. Como destacado por Hastie, Tibshirani e Friedman em *The Elements of Statistical Learning* (2009), essa abordagem permite uma avaliação justa do desempenho do modelo, garantindo que ele generalize bem para novos dados. Eles ressaltam que uma divisão comum é de aproximadamente

60-80% para treinamento, 10-20% para validação e 10-20% para teste. Além disso, Bishop (2006) enfatiza a importância da separação dos dados para evitar o sobreajuste do modelo aos dados de treinamento, recomendando uma distribuição semelhante de dados. Essa metodologia tem sido amplamente adotada na comunidade de aprendizado de máquina, conforme discutido por Goodfellow, Bengio e Courville em *Deep Learning* (2016), onde são apresentadas técnicas para a divisão adequada dos dados e a realização de avaliações robustas do modelo.

Depois de definida a divisão do conjunto de imagens para o modelo, é discutido o processo de avaliação de desempenho para determinar a escolha do melhor modelo treinado.

3.5.1 Avaliação dos Resultados

3.5.1.1 Desempenho do modelo

Para a escolha do melhor modelo treinado, é analisado o desempenho dos resultados obtidos pelas métricas:

- Perda: para avaliar como ele está se ajustando aos dados de treinamento seu valor deve ser o que mais se aproxima de 0. Segundo Bishop (2006) a ideia de que a função de perda deve ser minimizada durante o treinamento de um modelo é um conceito central em aprendizado de máquina, e é provável que seja discutido no contexto de modelos probabilísticos e de classificação.

- Acurácia: para verificar a proporção de previsões corretas em relação ao número total de previsões seu valor deve ser o mais próximo de 1. De acordo com Goodfellow et al. (2016) uma alta acurácia, próxima de 1, indica que o modelo está fazendo previsões precisas e confiáveis, enquanto uma acurácia baixa sugere que o modelo pode estar tendo dificuldades em realizar previsões precisas.

- Índice de Jaccard (IoU): para verificar a sobreposição entre a máscara prevista e a máscara de referência para cada classe (ou região) de interesse seu valor deve ser o mais próximo de 1. Segundo Everingham et al. (2010) o mais próximo de 1 indica uma sobreposição significativa entre as máscaras, o que é desejável para uma segmentação precisa e confiável.

- Perda Suave Dice (SDL): para avaliar o melhor resultado obtido pelos treinamentos seu valor deve ser o que mais se aproxima de 0. Milletari et al. (2016) descrevem que

a SDL é calculada como 1 menos o Coeficiente Dice, com valores mais próximos de 0 indicando uma melhor correspondência entre as máscaras. Portanto, minimizar a SDL durante o treinamento é crucial para obter os melhores resultados de segmentação.

- **Precisão:** para avaliar a qualidade das previsões positivas seu valor deve ser o mais próximo de 1. De acordo com Goodfellow et al. (2016) uma alta precisão, próxima de 1, indica que o modelo está fazendo previsões positivas de alta qualidade e confiáveis, minimizando assim os falsos positivos.

- **Sensibilidade:** para medir a proporção de exemplos positivos reais que foram identificados corretamente pelo modelo seu valor deve ser o mais próximo de 1. Segundo James et al. (2013) uma alta sensibilidade, é próxima de 1 e indica que o modelo está identificando corretamente a maioria dos exemplos positivos reais, minimizando assim os falsos negativos.

- **Especificidade:** para medir a proporção de exemplos negativos reais que foram identificados corretamente pelo modelo seu valor deve ser o mais próximo de 1 para uma alta especificidade, indicando que o modelo está identificando corretamente a maioria dos exemplos negativos reais, minimizando assim os falsos positivos, segundo James et al. (2013).

- **Erro Médio Quadrático (MSE):** para a média dos quadrados das diferenças entre os valores previstos e os valores reais seu valor deve ser o que mais se aproxima de 0, indicando que o modelo está produzindo previsões que estão mais próximas dos valores reais, refletindo uma melhor qualidade de ajuste do modelo aos dados de treinamento, de acordo com Hastie et al. (2009).

- **Erro Médio Absoluto (MAE):** para calcular a média das diferenças absolutas entre os valores previstos e os valores reais seu valor deve ser o que mais se aproxima de 0. Segundo James et al. (2013) um MAE mais próximo de 0 indica que o modelo está produzindo previsões que estão mais próximas dos valores reais.

3.5.1.2 Treinamento utilizando diferentes métricas de perda

A função de perda é crucial para guiar o processo de aprendizado do modelo, fornecendo um *feedback* sobre o quão bem o modelo está realizando a tarefa de segmentação das imagens. Ao ajustar os pesos do modelo para minimizar a perda, o modelo aprende a melhorar suas predições e a segmentar com mais precisão as regiões de interesse nas imagens. Na recalibragem dos pesos com base no valor da perda, os pesos

das conexões são ajustados de acordo com a magnitude do gradiente da função de perda em relação a esses pesos. Isso permite uma adaptação mais sensível dos pesos durante o treinamento, focalizando os ajustes nos parâmetros que contribuem significativamente para reduzir a perda, segundo Lecun et al. (1998).

Para uma análise e contribuição da pesquisa, é avaliado o resultado gerado pelo modelo treinado utilizando 2 tipos métricas de perda para o treinamento do modelo:

- Perda Entropia Cruzada Categórica (ECC)

A equação Perda Entropia Cruzada Categórica (ECC) é apresentada na equação 2.6 deste trabalho.

Essa função de perda não é criada manualmente, dado que o modelo de arquitetura 3D U-Net é comumente encontrado em bibliotecas populares de Python, ela é encapsulada dentro do framework *TensorFlow*. Ao definir a função de perda no método `compile()` do modelo, está-se utilizando a implementação padrão dessa função de perda fornecida pelo *TensorFlow*. O *TensorFlow* fornece uma ampla gama de funções de perda pré-implementadas para várias tarefas de aprendizado de máquina, e Entropia Cruzada Categórica é uma delas. Ao especificar essa função de perda durante a compilação do modelo, o *TensorFlow* cuida de todo o processo de cálculo da perda durante o treinamento, tornando a implementação do modelo mais simples e concisa. Isso simplifica o treinamento de modelos, tornando-a uma opção conveniente e amplamente adotada em várias pesquisas para a tarefa de segmentação.

- Perda Entropia Cruzada Categórica combinada com a Perda Suave Dice (ECC + SDL)

Durante o treinamento do modelo, uma etapa crucial desta pesquisa envolve a utilização de uma perda personalizada, a qual combina a Entropia Cruzada Categórica (ECC) com a Perda Suave Dice (SDL). Essa combinação é fundamental para a proposta desta pesquisa, avaliando a discrepância entre as previsões do modelo e os rótulos verdadeiros associados aos dados de treinamento, visto que a ECC é comumente utilizada para tarefas de classificação multiclasse, enquanto a SDL é conhecida por sua capacidade de lidar com a sobreposição de classes e produzir segmentações suaves e bem definidas. A análise do impacto da utilização dessa perda personalizada no treinamento de um modelo baseado na arquitetura 3D U-Net para a detecção de tumores cerebrais em imagens de ressonância magnética é realizada por meio da comparação das métricas de desempenho

dos modelos.

Algoritmo 1 Perda ECC + SDL

```

1: Classe CustomCategoricalCrossentropy(tf.keras.losses.Loss)
2: procedure INIT (suavização_rótulo=1e-6, pesos_classe = nulo, peso_dice = 0.5,
   nome='custom_categorical_crossentropy').
3:   Inicializa a classe base com o nome fornecido.
4:   Define o parâmetro de suavização do rótulo para ajustar a suavização.
5:   Define os pesos da classe como nulos por padrão.
6:   Define o peso do Dice como 0.5 por padrão.
7: end procedure
8: procedure CALL (y_verdadeiro, y_previsto)
9:   Suaviza os rótulos verdadeiros para evitar valores extremos.
10:  Calcula a entropia cruzada categórica entre os rótulos verdadeiros e as previsões.
11:  Calcula a interseção entre os rótulos verdadeiros e as previsões.
12:  Calcula a união dos rótulos verdadeiros e as previsões.
13:  Calcula o coeficiente Dice para avaliar a sobreposição.
14:  Calcula a perda Dice com base no coeficiente Dice.
15:  Calcula a perda combinada de entropia cruzada e Dice.
16:  Aplica pesos de classe, se fornecidos.
17:  Retorna a perda combinada.
18: end procedure

```

Como é apresentado no Algoritmo 1, essa é uma função de perda personalizada definida como uma classe em *TensorFlow/Keras*.

- **Inicialização:** Durante a inicialização, você pode especificar parâmetros como suavização de rótulo (*label_smoothing*), pesos de classe (*class_weights*), e um peso para a perda suave Dice (*dice_weight*).
- **Chamada:** No método *call*, a suavização de rótulo é aplicada aos rótulos verdadeiros para evitar que o modelo se torne muito confiante em suas previsões. Em seguida, é calculada a Entropia Cruzada Categórica entre os rótulos verdadeiros (*y_true*) e as previsões do modelo (*y_pred*). Além disso, a perda suave Dice é calculada para medir a sobreposição entre os rótulos verdadeiros e as previsões. Essas duas perdas são combinadas, onde o peso da perda suave Dice é determinado pelo parâmetro *dice_weight*. Se houver pesos de classe especificados, eles são aplicados à perda combinada.
- **Retorno:** A função retorna a perda combinada, que é usada para ajustar os parâmetros do modelo durante o processo de treinamento.

TensorFlow/Keras são bibliotecas de aprendizado profundo (*deep learning*) escritas em Python. Enquanto uma classe é uma estrutura de programação que encapsula métodos e atributos relacionados a um conceito específico. No contexto de redes neurais e aprendizado profundo, as classes são frequentemente usadas para definir modelos de rede neural, camadas, funções de perda personalizadas, otimizadores, métricas e muito mais. Classes também podem ser estendidas e personalizadas para atender às necessidades específicas do projeto, como a função mencionada de perda combinada ECC + SDL.

Após conclusão do Capítulo de Metodologia, é apresentado os Resultados, onde se encontram a exploração de dados do banco de imagens, Resultados das Métricas e as Previsões do Modelo 3D U-Net.

Capítulo 4

Resultados

4.1 Explorando os dados

Os resultados apresentados a seguir são referentes ao processamento de imagens de ressonância magnética de tumores cerebrais pré-processadas no formato NIFTI, disponíveis pela Competição BraTS2020, com 5 imagens do mesmo cérebro com imagens de T1, T1c, T2, Flair e uma última sendo uma imagem do tumor segmentado para o treinamento do modelo. As visualizações a seguir foram geradas a partir do arquivos de treinamento do grupo de imagens BraTS20_Training_009.

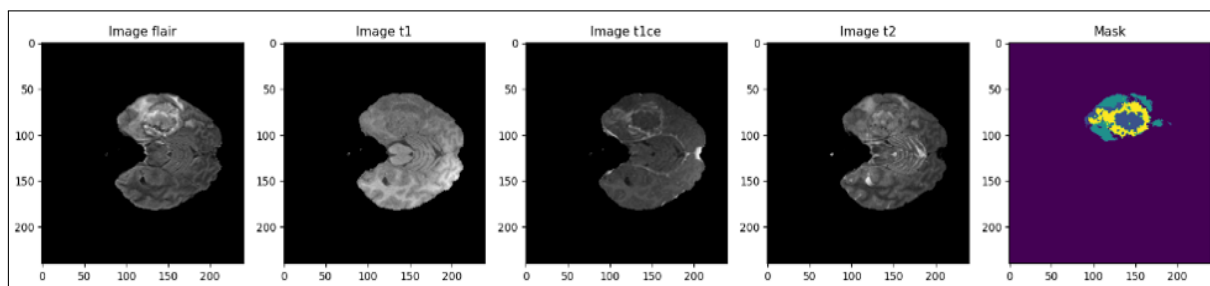


Figura 4.1: 5 tipos de imagens do mesmo cérebro

Fonte: Própria Autora

A Figura 4.1 apresenta as imagens geradas do grupo de imagens BraTS20_Training_009 onde é possível observar 5 tipos imagens referentes ao mesmo cérebro com a localização tumoral através da imagem chamada Mask. Foi utilizada a biblioteca NiBabel para realizar a interação com esse formato de dados NIFTI e a visualização das 5 tipos da esquerda para a direita: Flair, T1, T1ce, T2 e Mask.

É possível “girar” o ângulo para análise de outras perspectivas das imagens estudadas

através da biblioteca Nilearn que possui um conjunto de funções de plotagem para plotar volumes cerebrais.

A seguir, como exemplo, são apresentadas visualizações da imagem Flair BraTS20_Training_009. Foram utilizadas diferentes funções para explorar a imagem com a ajuda da biblioteca `nilearn.plotting`, que é comumente usada para visualização de imagens neurocientíficas em Python, com ângulos da esquerda para direita: Coronal, Sagital e Transversal. Cada função tem um propósito específico.

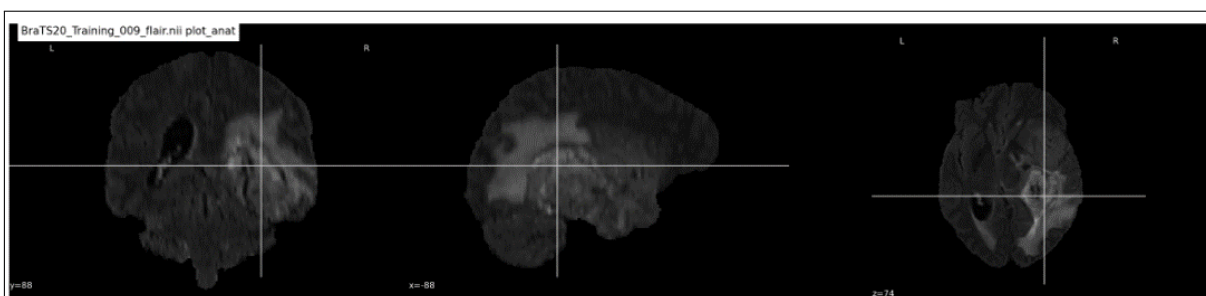


Figura 4.2: Imagem Flair BraTS20_Training_009 com função `plot_anat`

Fonte: Própria Autora

A Figura 4.2 foi gerada com a função `nlplt.plot_anat`. Esta função é utilizada para criar um gráfico de uma imagem anatômica (anatomia estrutural).

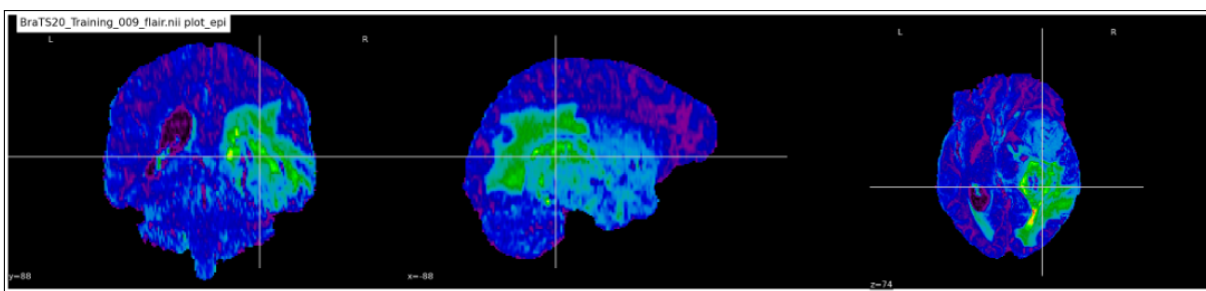


Figura 4.3: Imagem Flair BraTS20_Training_009 com função `plot_epi`

Fonte: Própria Autora

A Figura 4.3 foi gerada com a função `nlplt.plot_epi`. Pode ser usada para visualizar a atividade cerebral ao longo do tempo.

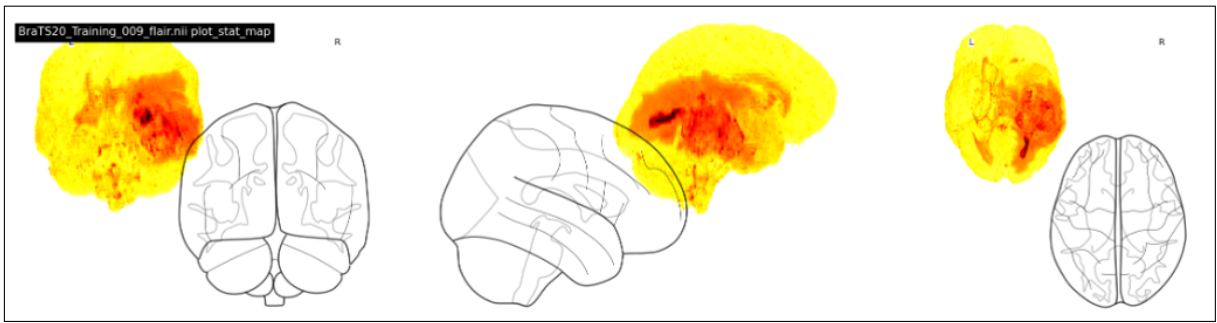


Figura 4.4: Imagem Flair BraTS20_Training_009 com função `plot_glass_brain`

Fonte: Própria Autora

A Figura 4.4 foi gerada com a função `nlplt.plot_glass_brain`. Esta função gera um gráfico estilo "cérebro de vidro" que é uma visualização tridimensional transparente da atividade cerebral.

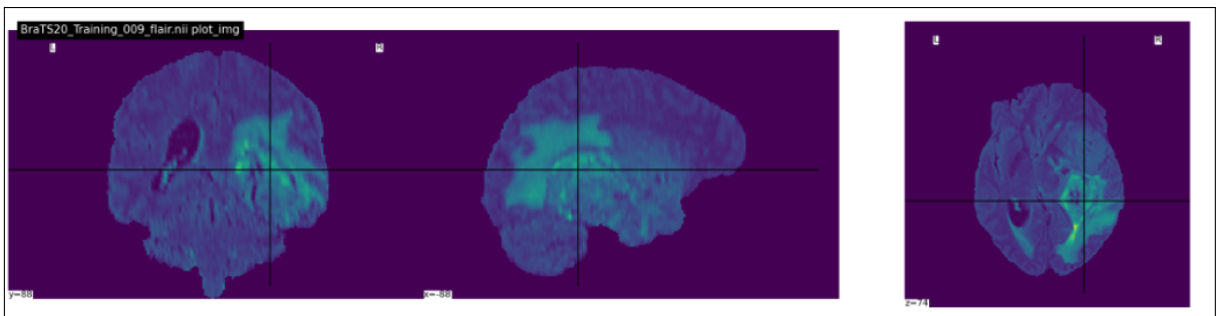


Figura 4.5: Imagem Flair BraTS20_Training_009 com função `plot_img`

Fonte: Própria Autora

A Figura 4.5 foi gerada com a função `nlplt.plot_img`. Esta função é usada para visualizar uma imagem de ressonância magnética (IRM) em geral. Pode ser uma imagem anatômica, funcional ou outra.

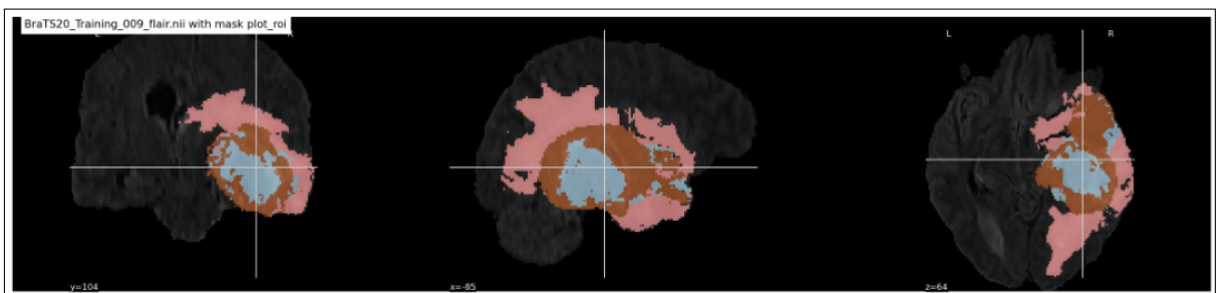


Figura 4.6: Imagem Flair BraTS20_Training_009 com função `plot_roi`

Fonte: Própria Autora

A Figura 4.6 foi gerada com a função `nlplt.plot_roi`. Usada para sobrepor uma máscara (região de interesse) em uma imagem de ressonância magnética. A máscara pode ser uma região específica que você deseja destacar.

Nenhuma das funções utilizadas da biblioteca `nilearn` para a visualização e exploração das imagens obtidas tem como objetivo de identificar a região tumoral, apenas podem destacar regiões de possível interesse.

No próximo tópico 4.2 é discutido o treinamento do modelo e os resultados obtidos são detalhados.

4.2 Treinamento o Modelo 3D U-Net

Foi realizado o treinamento do modelo consecutivas vezes com o objetivo de alcançar o modelo com melhor desempenho. O custo computacional é alto e consome considerável processamento do computador localmente. Foi observado que época=35 trazia resultados relevantes, com um tempo apesar de alto (39h). Por esse motivo e a necessidade de rodar o modelo diversas vezes, contratou-se um serviço em nuvem, o *Google Collaboratory Pro*, para a execução de códigos em *Python* reduzindo o tempo médio de processamento para 2h com execução de 40 épocas, ampliando assim o número de época para treinamento do modelo com um tempo muito menor.

4.2.1 Resultados das Métricas

São apresentados a seguir os resultados obtidos de cada modelo treinado de melhor desempenho com a utilização da Perda Entropia Cruzada Categórica (ECC) e a combinação da Perda Entropia Cruzada Categórica (ECC) com a Perda Suave Dice (ECC + SDL)

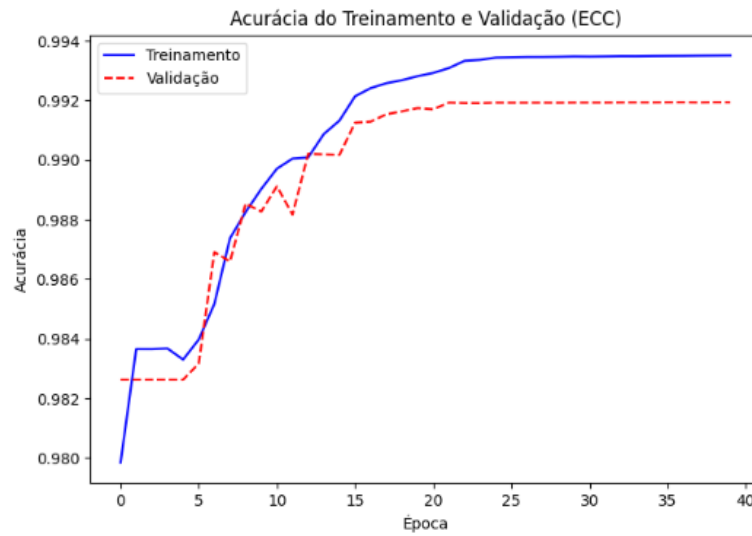


Figura 4.7: Acurácia durante o treinamento e validação com perda ECC

Fonte: Própria Autora

Na Figura 4.7 apresenta o valor de Acurácia do treinamento e validação do modelo com a Perda Entropia Cruzada Categórica (ECC) que finaliza no valor de 0.99351 de treinamento e 0.99194 de validação com uma diferença de 0.00157 representando 0.16% de acurácia de validação menor que o valor de Acurácia do treinamento.

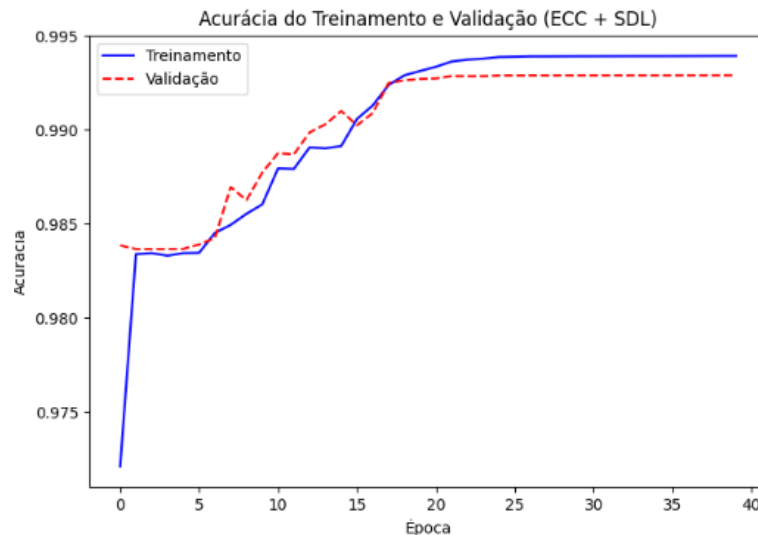


Figura 4.8: Acurácia durante o treinamento e validação com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.8, são apresentados os valores de Acurácia durante o treinamento e a validação do modelo que combina as perdas de Entropia Cruzada Categórica e Perda Suave Dice (ECC + SDL). A Acurácia de treinamento atinge 0.99393, enquanto a de

validação alcança 0.99290, representando uma diferença de 0.00103, ou seja, uma redução de 0.10% na Acurácia de validação em relação à acurácia de treinamento.

Comparando os valores de Acurácia dos dois modelos treinados é possível observar que no treinamento há diferença de 0.00042 representando 0.04% e uma diferença na validação de 0.00097 representando 0.10% superiores no modelo ECC + SDL.

É notável em ambos os modelos treinados, valores elevados tanto no treinamento quanto na validação. Contudo, é crucial destacar que uma acurácia elevada não assegura a excelência do modelo. Portanto, a avaliação deve ser enriquecida por outras métricas para uma análise mais abrangentes para uma melhor avaliação dos dois modelos.

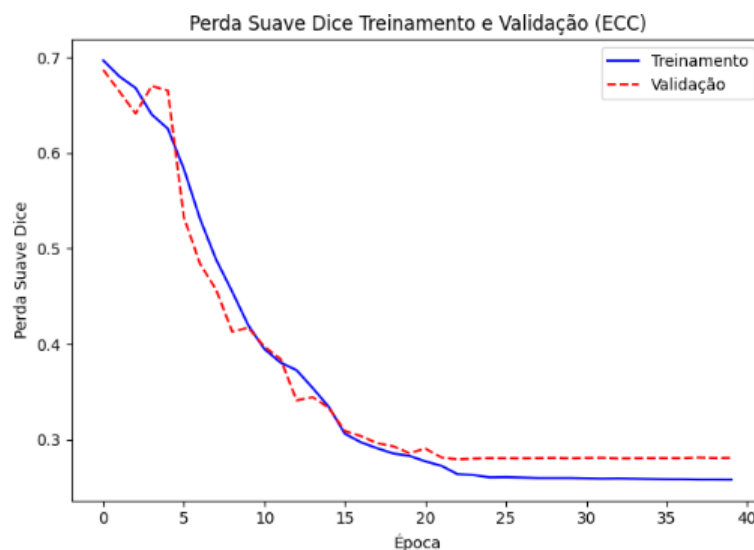


Figura 4.9: Perda Suave Dice durante o treinamento e validação com perda ECC

Fonte: Própria Autora

Na Figura 4.9, são apresentados os valores de Perda Suave Dice (SDL) tanto para o treinamento quanto para a validação do modelo, utilizando a combinação da Perda Entropia Cruzada Categórica com a Perda Suave Dice (ECC + SDL). O valor obtido para o treinamento foi de 0.25807, enquanto para a validação foi de 0.28095, indicando uma diferença de 0.02288. Esta diferença representa um aumento de 8.14% na Perda Suave Dice (SDL) de validação em relação ao treinamento.

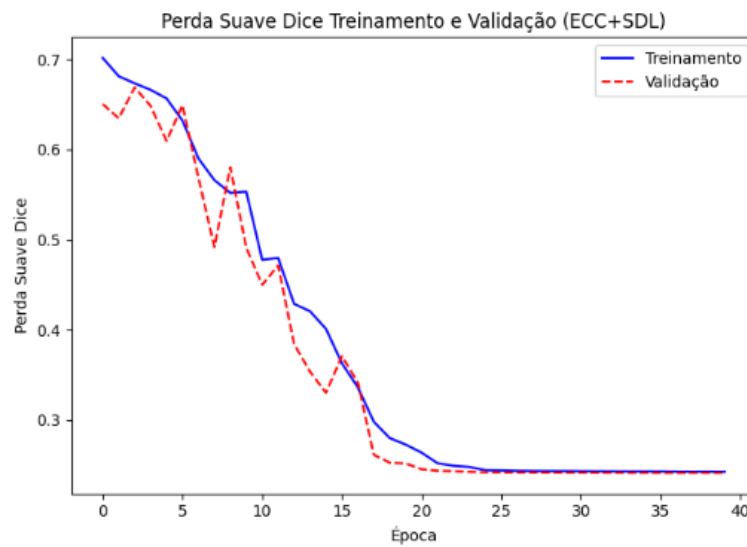


Figura 4.10: Perda Suave Dice durante o treinamento e validação com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.10, são apresentados os resultados da Perda Suave Dice (SDL), que alcançou 0.24261 no treinamento e 0.24131 na validação do modelo combinando as perdas de Entropia Cruzada Categórica e Perda Suave Dice (ECC + SDL). Observa-se uma diferença pequena de 0.00131, representando uma redução de 0.54% na validação em comparação ao treinamento.

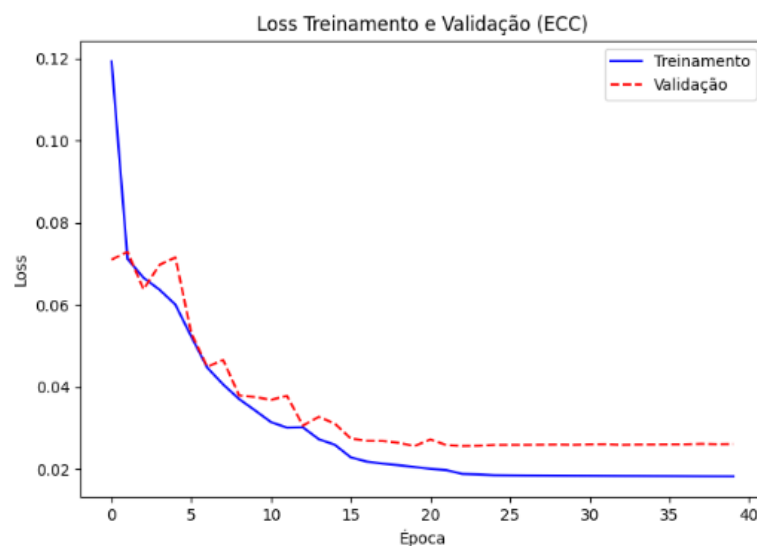


Figura 4.11: Perda durante o treinamento e validação com perda ECC

Fonte: Própria Autora

A Figura 4.11 apresenta resultados da Perda Entropia Cruzada Categórica (ECC) que

atingiu 0.01827 no treinamento e 0.02614 na validação do modelo, sendo uma diferença de 0.00787 que representa 30,10% de diferença maior da validação em relação ao treinamento.

Essa é uma função de perda fundamental para guiar o ajuste dos parâmetros do modelo durante o processo de treinamento de forma a minimizar a discrepância entre as saídas previstas pelo modelo e os rótulos verdadeiros associados aos dados de treinamento.

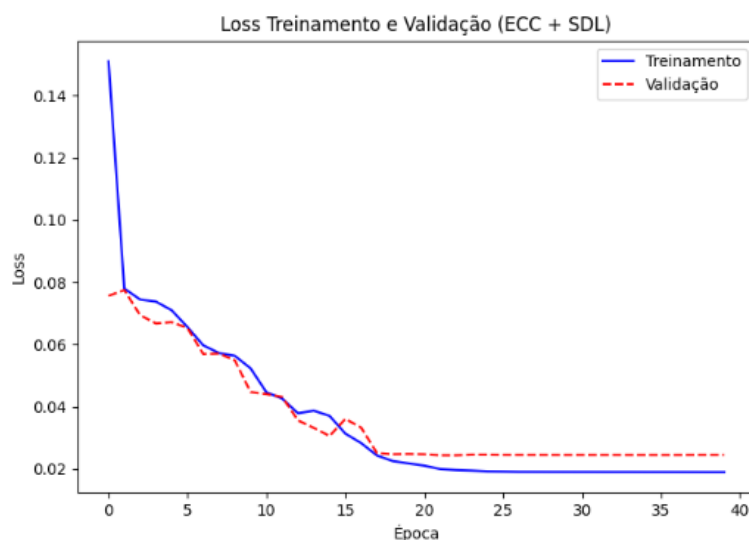


Figura 4.12: Perda durante o treinamento e validação com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.12, são apresentados os resultados da Perda Entropia Cruzada Categórica combinada com a Perda Suave Dice (ECC + SDL). Os valores obtidos foram de 0.01902 para o treinamento e 0.02453 para a validação do modelo. Isso representa uma diferença de 0.00551 entre os dois modelos treinados, uma diferença de 22.45% da validação em relação ao treinamento.

É importante ressaltar que a função de perda foi modificada conforme proposto no trabalho e é empregada pelo modelo 3D U-Net para ajustar os pesos durante o treinamento.

Outro aspecto relevante é que a baixa perda durante o treinamento sugere uma adaptação eficiente do modelo aos dados. A discrepância leve entre a perda durante o treinamento e a perda durante a validação indica a ausência de sobreajuste (overfitting).

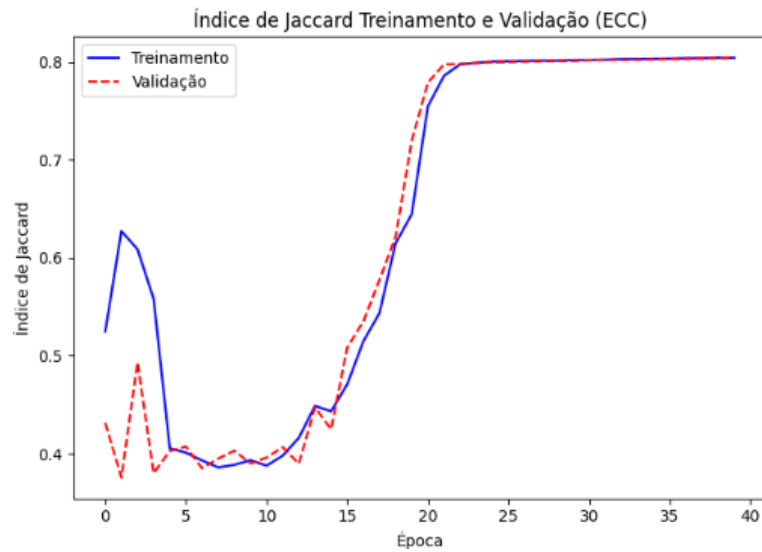


Figura 4.13: Índice de Jaccard durante o treinamento e validação com perda ECC

Fonte: Própria Autora

A Figura 4.13 apresenta resultados do Índice de Jaccard que atingiu 0.80422 no treinamento e 0.80438 na validação do modelo, sendo uma pequena diferença de 0.00016 que representa 0.02% de diferença maior da validação em relação ao treinamento.

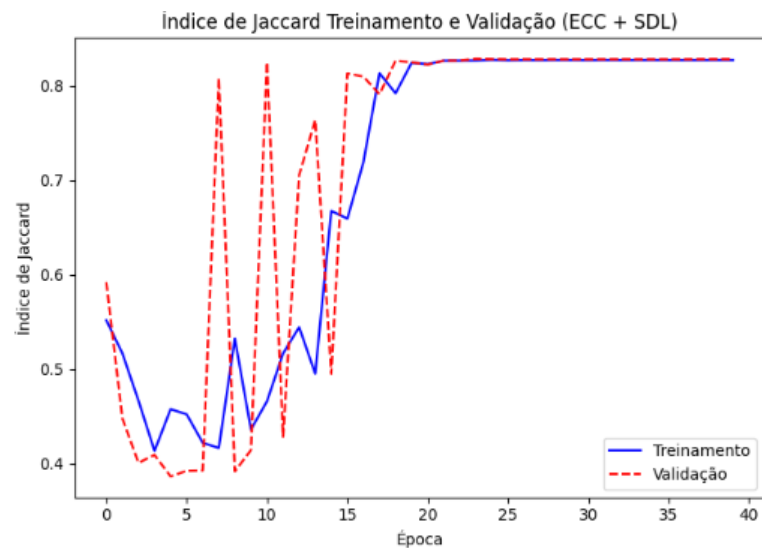


Figura 4.14: Índice de Jaccard durante o treinamento e validação com perda ECC + SDL

Fonte: Própria Autora

A Figura 4.14 apresenta resultados do Índice de Jaccard que atingiu 0.82714 no treinamento e 0.82800 na validação do modelo, sendo uma pequena diferença de 0.00086 que representa 0.10% de diferença maior da validação em relação ao treinamento.

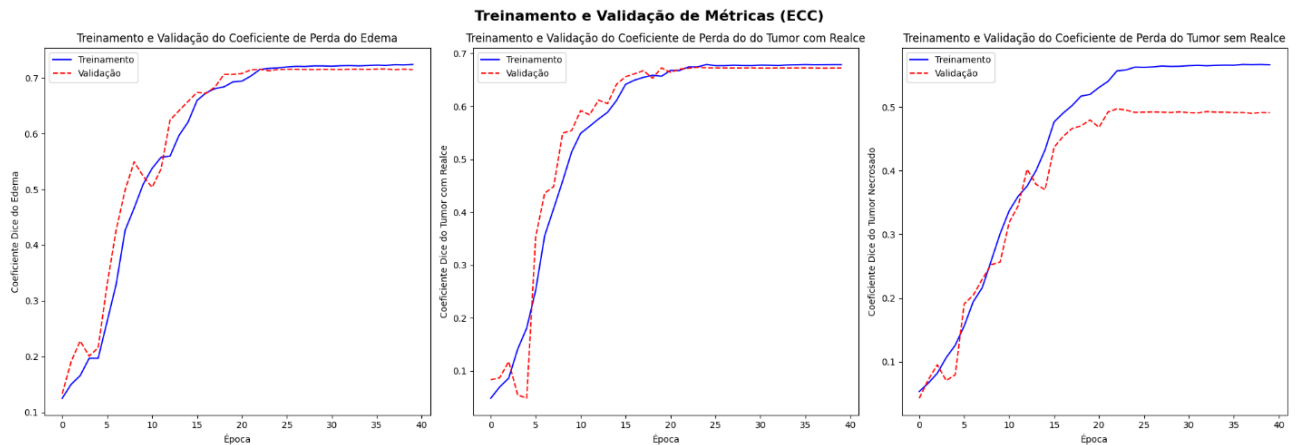


Figura 4.15: Coeficiente de Dice do Edema, do Tumor sem Realce e com Tumor com Realce durante o treinamento e validação com perda ECC

Fonte: Própria Autora

A Figura 4.15 é possível acompanhar como foi o desempenho do Coeficiente Dice (DSC) no modelo treinado com perda ECC:

- Edema: obteve no treinamento 0.72395 e na validação 0.71460 resultando na diferença de 0.00936 representando 1.29% da validação menor que o treinamento;
- Tumor com Realce: obteve no treinamento 0.67875 e na validação 0.67237 resultando na diferença de 0.00638 representando 0.94% da validação menor que o treinamento;
- Tumor sem Realce: obteve no treinamento 0.56624 e na validação 0.49110 resultando na diferença de 0.07514 representando 13.27% da validação maior que o treinamento;

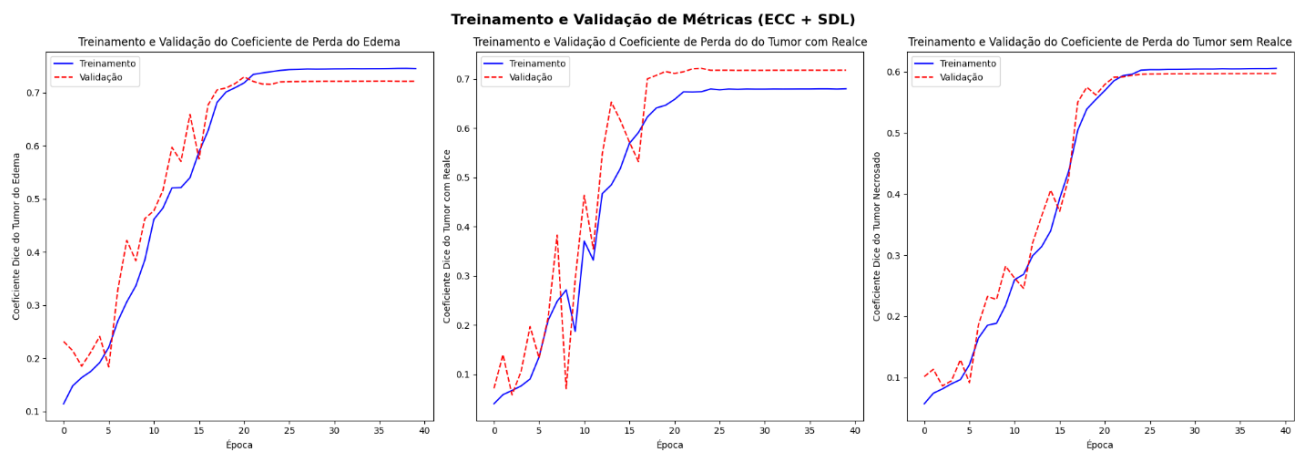


Figura 4.16: Coeficiente de Dice do Edema, do Tumor sem Realce e com Tumor com Realce durante o treinamento e validação com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.16 é acompanhado o desempenho do Coeficiente Dice (DSC) do modelo treinado com perda ECC + SDL:

- Edema: obteve no treinamento 0.74522 e na validação 0.72146 resultando na diferença de 0.02376 representando 3.19% da validação menor que o treinamento;
- Tumor com Realce: obteve no treinamento 0.67988 e na validação 0.71764 resultando na diferença de 0.03776 representando 5.26% da validação maior que o treinamento;
- Tumor sem Realce: obteve no treinamento 0.60560 e na validação 0.59719 resultando na diferença de 0.00840 representando 1.39% da validação menor que o treinamento;

Após todos os resultados apresentados até aqui, pode ser verificado que os valores de métricas de treinamento estão próximos aos valores de validação, isso acontece quando o treinamento obtido do modelo tem capacidade de generalização.

A importância de ter métricas de treinamento próximas às métricas de validação está relacionada à capacidade do modelo de generalizar para novos dados de forma precisa e robusta, o que é fundamental para o desempenho e a confiabilidade do modelo em cenários do mundo real (Bishop, 2006; James, et al. 2013; Géron, 2017). Um modelo que se ajusta bem aos dados de treinamento, mas não consegue generalizar para dados não vistos (validação ou teste), é considerado superajustado (overfitting) (Mitchell, 1997). Por outro lado, se as métricas de treinamento e validação são semelhantes, isso sugere que o modelo está aprendendo padrões gerais nos dados em vez de memorizar os dados de treinamento (Sokolova; Lapalme, 2009). Isso resulta em uma melhor capacidade do modelo de fazer previsões precisas em novos dados. Modelos que se ajustam muito bem aos dados de treinamento podem ter dificuldade em se adaptar a novos padrões ou variações nos dados (Sokolova; Lapalme, 2009). Se as métricas de treinamento e validação forem semelhantes, isso indica que o modelo está aprendendo padrões úteis que são relevantes para diferentes conjuntos de dados, permitindo uma adaptação mais suave a novos dados (Géron, 2017). Se os valores das métricas de treinamento e validação forem inconsistentes, é difícil confiar nas previsões do modelo, pois não está claro como ele se comportará em novos dados (Sokolova; Lapalme, 2009). Métricas de treinamento e validação consistentes proporcionam uma base mais confiável para a tomada de decisões baseadas no modelo (Sokolova; Lapalme, 2009; Géron, 2017).

Após o acompanhamento e análise de todas as métricas de desempenho, agora são comparados os resultados entre os modelos.

4.2.2 Comparando Resultados das Métricas

Os resultados entre o melhor modelo treinado com a Perda Entropia Cruzada Categórica (ECC) e o melhor modelo treinado com a Perda Entropia Cruzada Categórica + Perda Suave Dice (ECC + SDL) são comparados para análise mais profunda e para a avaliação do desempenho de cada treinamento dos modelos.

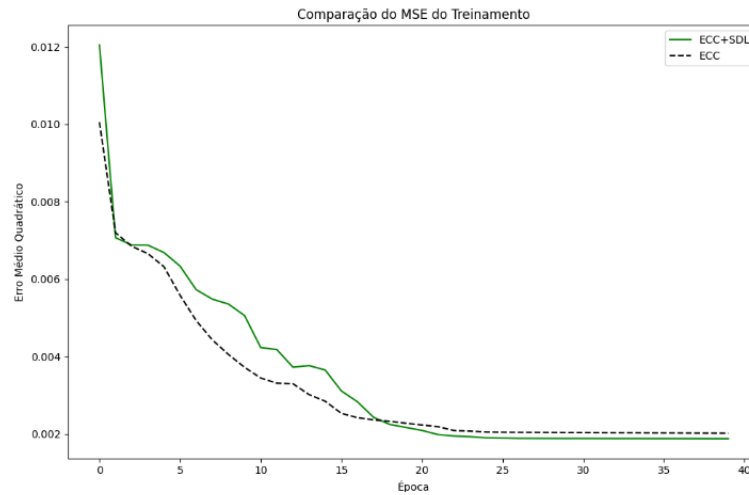


Figura 4.17: Erro Médio Quadrático durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.17 apresenta a comparação no treinamento entre o Erro Médio Quadrático (MSE) entre os dois modelos treinados finalizando em 0.00202 no modelo ECC e 0.00188 no modelo ECC + SDL, tendo uma pequena diferença de 0.00014 representando 7.02% do modelo ECC + SDL menor que o modelo ECC.

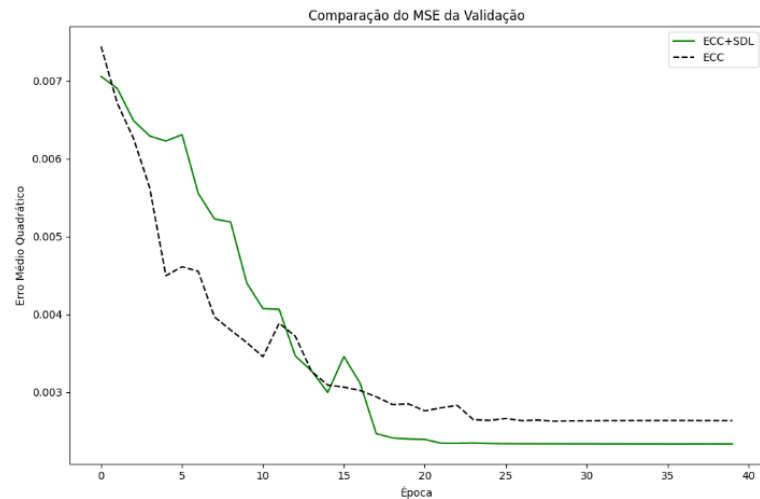


Figura 4.18: Erro Médio Quadrático durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.18, é mostrada a comparação do Erro Médio Quadrático (MSE) da validação entre os dois modelos treinados, resultando em 0.00271 para o modelo ECC e 0.00233 para o modelo ECC + SDL. Observa-se uma diferença mínima de 0.00038, representando 14.01% a menos no modelo ECC + SDL em relação ao modelo ECC.

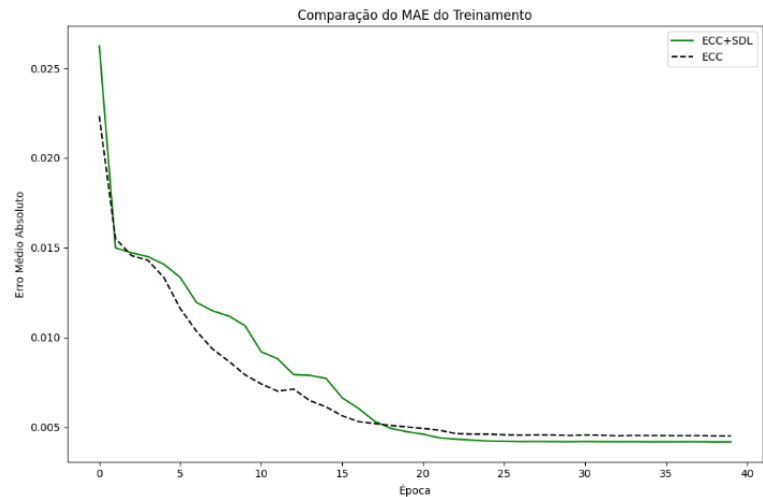


Figura 4.19: Erro Médio Absoluto durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

A Figura 4.19 apresenta a comparação no treinamento entre o Erro Médio Absoluto (MAE) entre os dois modelos treinados finalizando em 0.00450 no modelo ECC e 0.00417

no modelo ECC + SDL, tendo uma pequena diferença de 0.00033 representando 7.29% do modelo ECC + SDL menor que o modelo ECC.

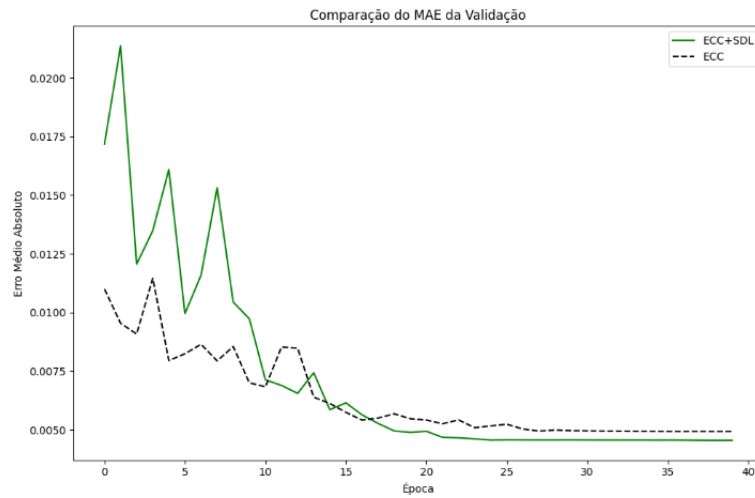


Figura 4.20: Erro Médio Absoluto durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

A Figura 4.20 mostra a comparação do Erro Médio Absoluto (MAE) durante o treinamento dos dois modelos, resultando em 0.00523 para o modelo ECC e 0.00455 para o modelo ECC + SDL. Há uma pequena diferença de 0.00068, representando 14.91% a menos no modelo ECC + SDL em relação ao modelo ECC.

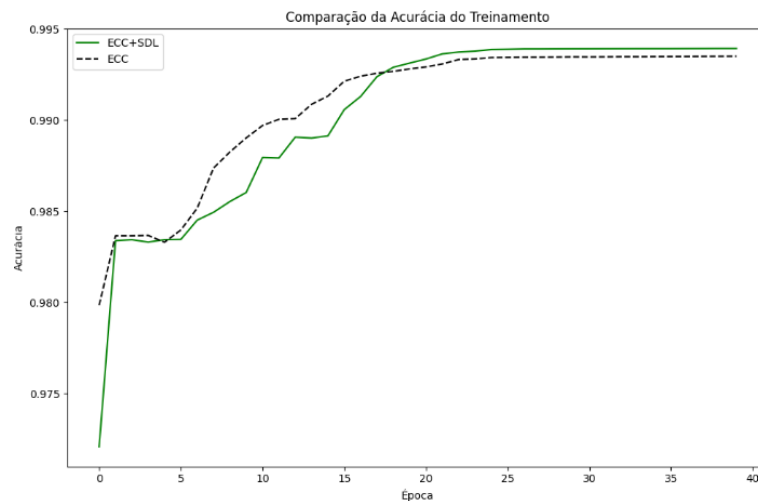


Figura 4.21: Acurácia durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.21 apresenta a comparação no treinamento entre a Acurácia entre os dois

modelos treinados finalizando em 0.99351 no modelo ECC e 0.99393 no modelo ECC + SDL, tendo uma pequena diferença de 0.00042 representando 0.04% do modelo ECC + SDL maior que o modelo ECC.

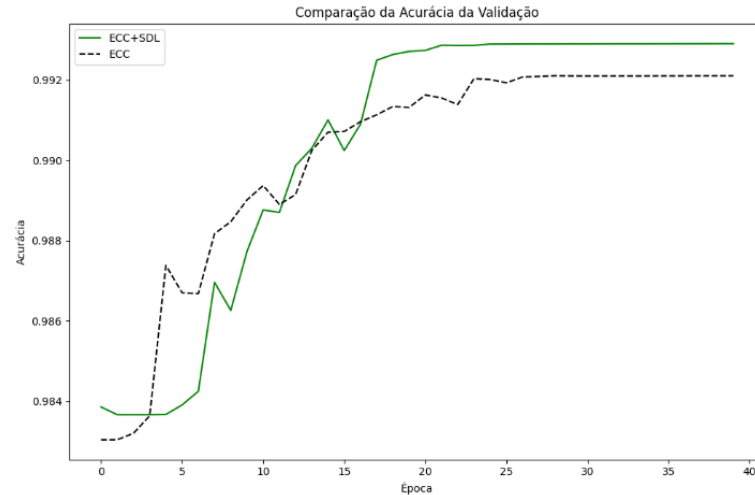


Figura 4.22: Acurácia durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.22, é apresentada a comparação da acurácia durante a validação entre os dois modelos, resultando em 0.99194 para o modelo ECC e 0.99290 para o modelo ECC + SDL. Há uma pequena diferença de 0.00097, representando um aumento de 0.10% no modelo ECC + SDL em relação ao modelo ECC.

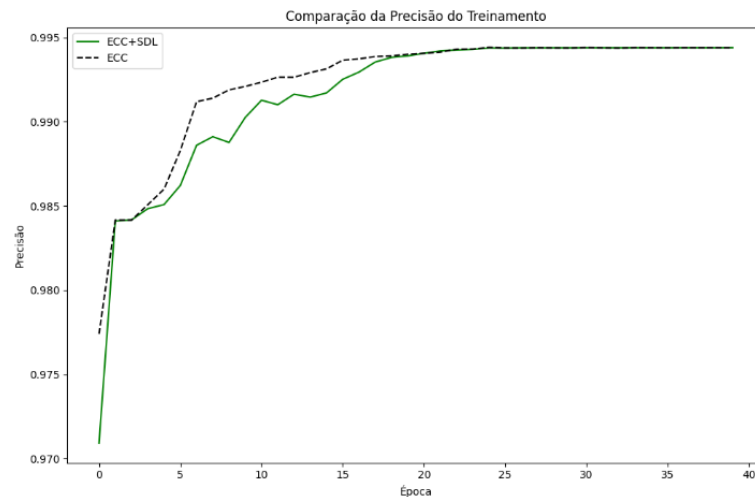


Figura 4.23: Precisão durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

A Figura 4.23 mostra a comparação do Precisão durante o treinamento dos dois

modelos, resultando em 0.99439 para o modelo ECC e 0.99440 para o modelo ECC + SDL. Há uma pequena diferença de 0.00001, representando 0.0008% a mais no modelo ECC + SDL em relação ao modelo ECC.

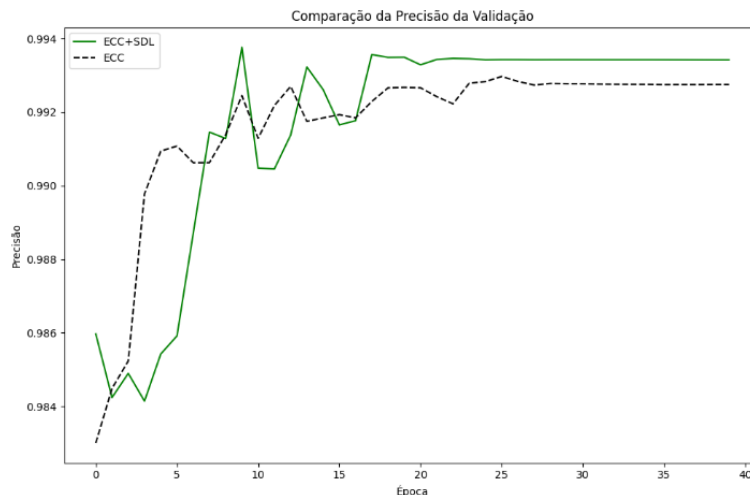


Figura 4.24: Precisão durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

A Figura 4.24 ilustra a comparação da Precisão durante o validação dos dois modelos, resultando em 0.99276 para o modelo ECC e 0.99342 para o modelo ECC + SDL. Observa-se uma diferença mínima de 0.00066, representando um aumento de 0.07% no modelo ECC + SDL em relação ao modelo ECC.

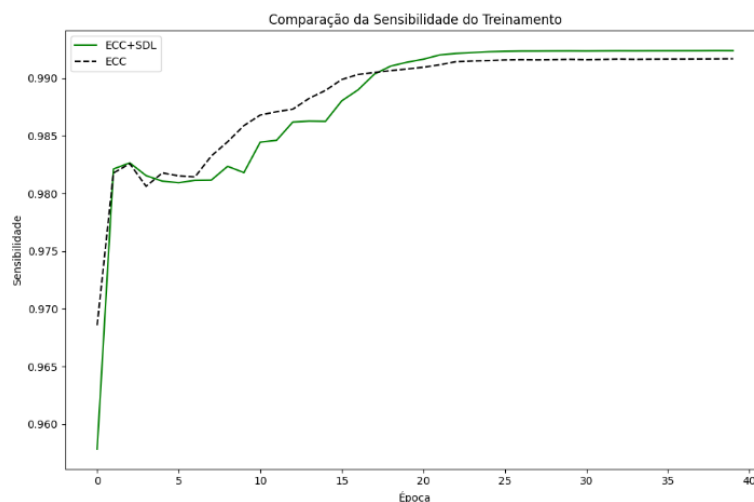


Figura 4.25: Sensibilidade durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.25 apresenta a comparação no treinamento entre a Sensibilidade entre os

dois modelos treinados finalizando em 0.99167 no modelo ECC e 0.99237 no modelo ECC + SDL, tendo uma pequena diferença de 0.00070 representando 0.07% do modelo ECC + SDL maior que o modelo ECC.

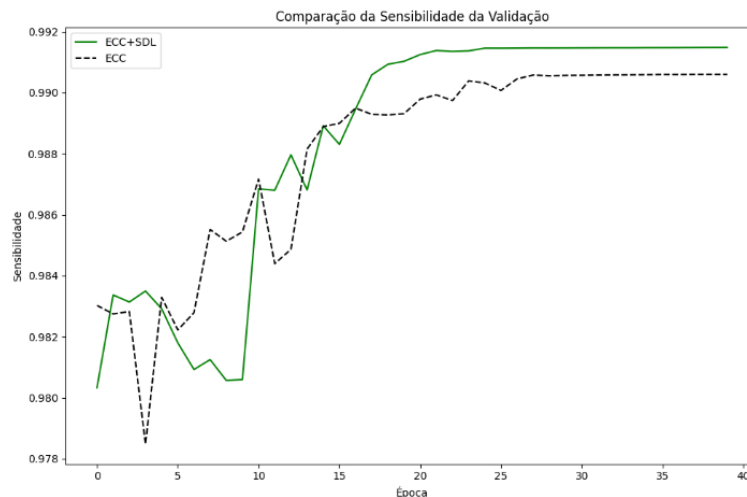


Figura 4.26: Sensibilidade durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.26, é apresentada a comparação da Sensibilidade durante a validação entre os dois modelos, resultando em 0.99030 para o modelo ECC e 0.99148 para o modelo ECC + SDL. Observa-se uma pequena diferença de 0.00118, representando um aumento de 0.12% no modelo ECC + SDL em relação ao modelo ECC.

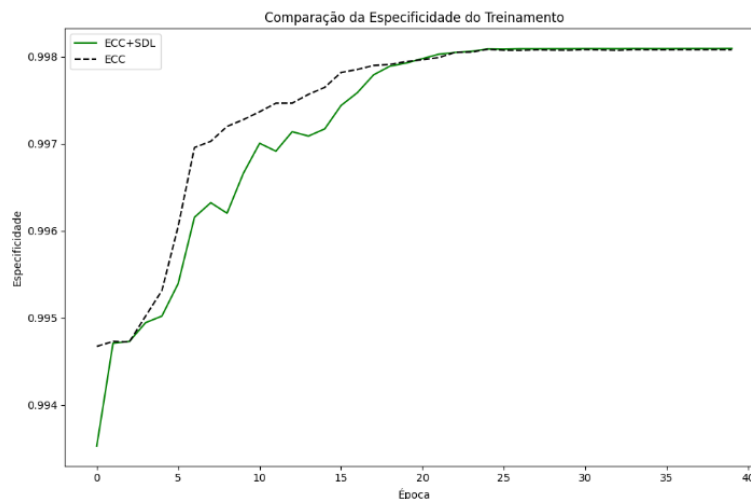


Figura 4.27: Especificidade durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.27, é apresentada a comparação da Especificidade durante a treinamento

entre os dois modelos, resultando em 0.99808 para o modelo ECC e 0.99810 para o modelo ECC + SDL. Há uma pequena diferença de 0.00002, representando um aumento de 0.0016% no modelo ECC + SDL em relação ao modelo ECC.

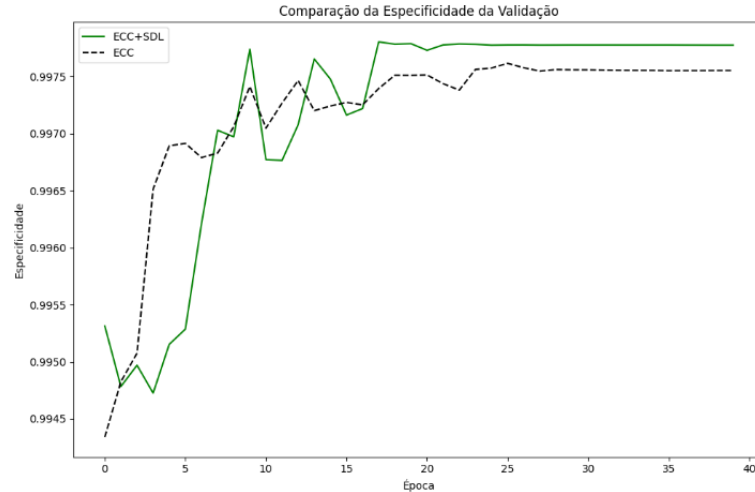


Figura 4.28: Especificidade durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

A Figura 4.28 ilustra a comparação da Especificidade durante o validação dos dois modelos, resultando em 0.99755 para o modelo ECC e 0.99778 para o modelo ECC + SDL. Nota-se uma diferença mínima de 0.00023, representando um aumento de 0.02% no modelo ECC + SDL em relação ao modelo ECC.

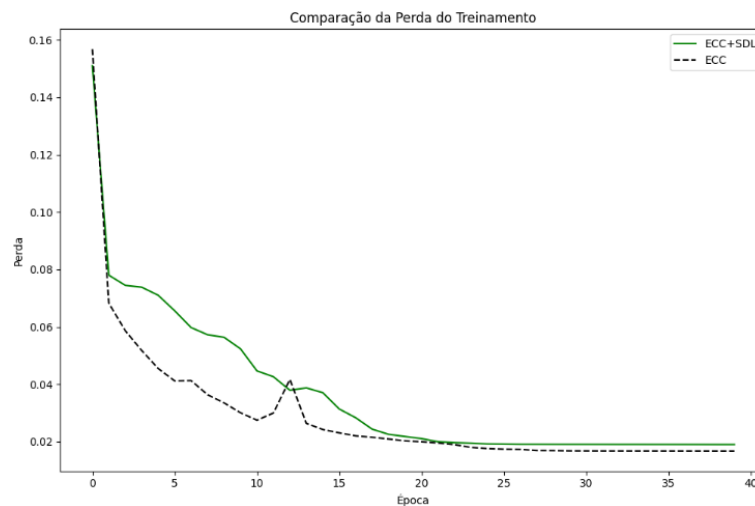


Figura 4.29: Perda durante o treinamento entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.29, é apresentada a comparação da Perda durante o treinamento entre os

dois modelos, resultando em 0.01827 para o modelo ECC e 0.01902 para o modelo ECC + SDL. Observa-se uma pequena diferença de 0.00075, representando 3.94% maior o modelo ECC + SDL em relação ao modelo ECC.

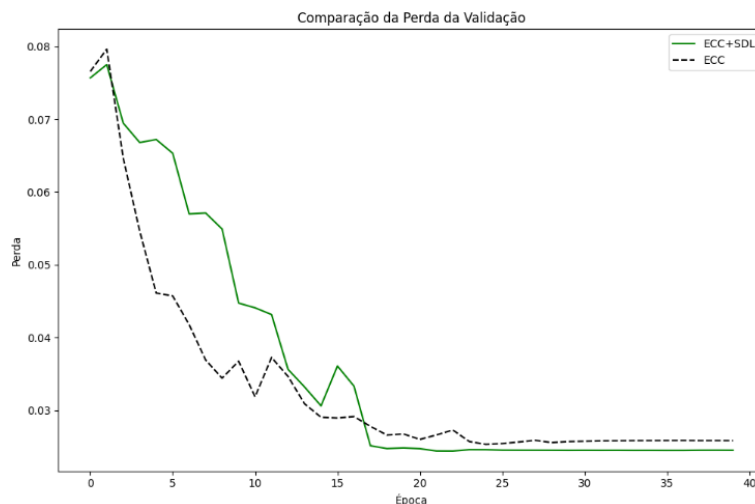


Figura 4.30: Perda durante a validação entre os modelos ECC e SDL + ECC

Fonte: Própria Autora

Na Figura 4.30, é exibida a comparação da Perda durante a validação entre os dois modelos, com valores de 0.02614 para o modelo ECC e 0.02453 para o modelo ECC + SDL. Observa-se uma pequena diferença de 0.00161, representando 6.16% menor o modelo ECC + SDL em relação ao modelo ECC.

Legenda: ■ Melhor resultado

Métricas de Desempenho	ECC Treinamento	ECC Validação	ECC + SDL Treinamento	ECC + SDL Validação
Acurácia	0,99351	0,99194	0,99393	0,99290
Loss	0,01827	0,02614	0,01902	0,02453
MAE	0,00450	0,00523	0,00417	0,00455
Índice de Jaccard	0,80422	0,80438	0,82714	0,82800
MSE	0,00202	0,00271	0,00188	0,00233
Precisão	0,99439	0,99276	0,99440	0,99342
Sensibilidade	0,99167	0,99030	0,99237	0,99148
Perda Suave Dice	0,25807	0,28095	0,24261	0,24131
Especificidade	0,99808	0,99755	0,99810	0,99778

Figura 4.31: Resumo para comparação de resultados

Fonte: Própria Autora

Na Figura 4.31 é comparado de forma resumida os resultados obtidos com cada métrica e é evidenciado na cor verde onde o modelo atingiu o melhor resultado.

É possível observar que o modelo ECC + SDL obteve melhores resultados em quase todas as métricas, com exceção, de forma mais significativa, da Perda no treinamento (Figura 4.29).

A média de tempo para o processamento do treinamento utilizando 40 épocas para o modelo ECC foi de 107 minutos, enquanto para o modelo ECC + SDL foi de 120 minutos.

É importante destacar que, além da maioria das métricas apresentarem resultados superiores, os gráficos mostram que o modelo ECC + SDL estabilizou com muito menos épocas. Isso sugere que o modelo converge mais rapidamente do que o modelo ECC, indicando que necessita de muito menos épocas e menos tempo para treinamento.

Fechando a análise de métricas, seguiu-se agora para as previsões dos modelos.

4.3 Previsões do Modelo 3D U-Net

Foram utilizadas imagens de ressonância magnética Flair e T1ce, as quais foram redimensionadas e enviadas para o modelo previamente treinado para fazer previsões. O resultado é uma matriz de probabilidades para cada pixel na imagem. Em seguida são exibidas uma série de imagens em uma única figura, incluindo a imagem original, a máscara ground truth e as previsões para as diferentes classes (tumor sem realce, tumor com realce e edema). Isso permite visualizar como o modelo está se saindo na segmentação em comparação com a máscara ground truth.

Para facilitar a comparação entre as previsões do melhor modelo treinado com Perda Entropia Cruzada Categórica (ECC) e o melhor modelo treinado com Perda Entropia Cruzada Categórica + Perda Suave Dice (ECC + SDL), as imagens do grupo de teste são exibidas na Figura 4.32 . Isso permitirá que as diferenças nas previsões sejam comparadas diretamente.

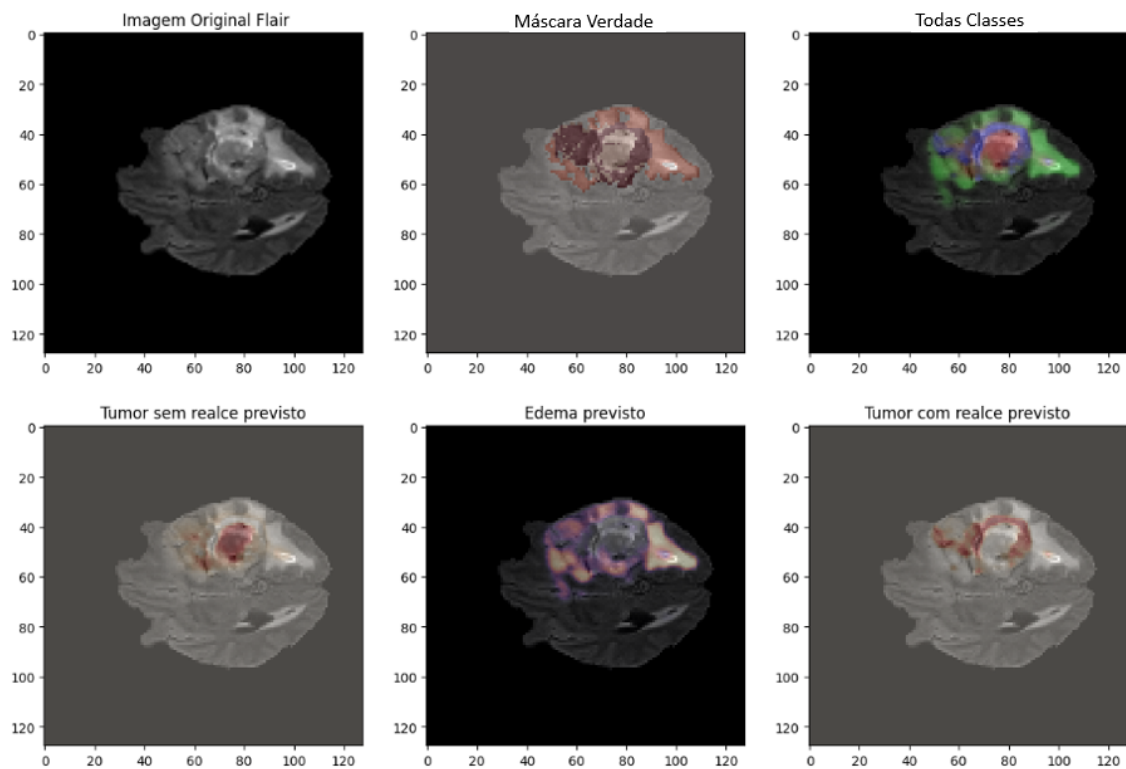


Figura 4.32: Previsão tumoral no cérebro 01 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.32, a previsão da região tumoral do cérebro deste paciente foi gerada usando uma imagem Flair e T1ce do conjunto de imagens de teste utilizando o modelo treinado com a perda Entropia Cruzada Categórica (ECC).

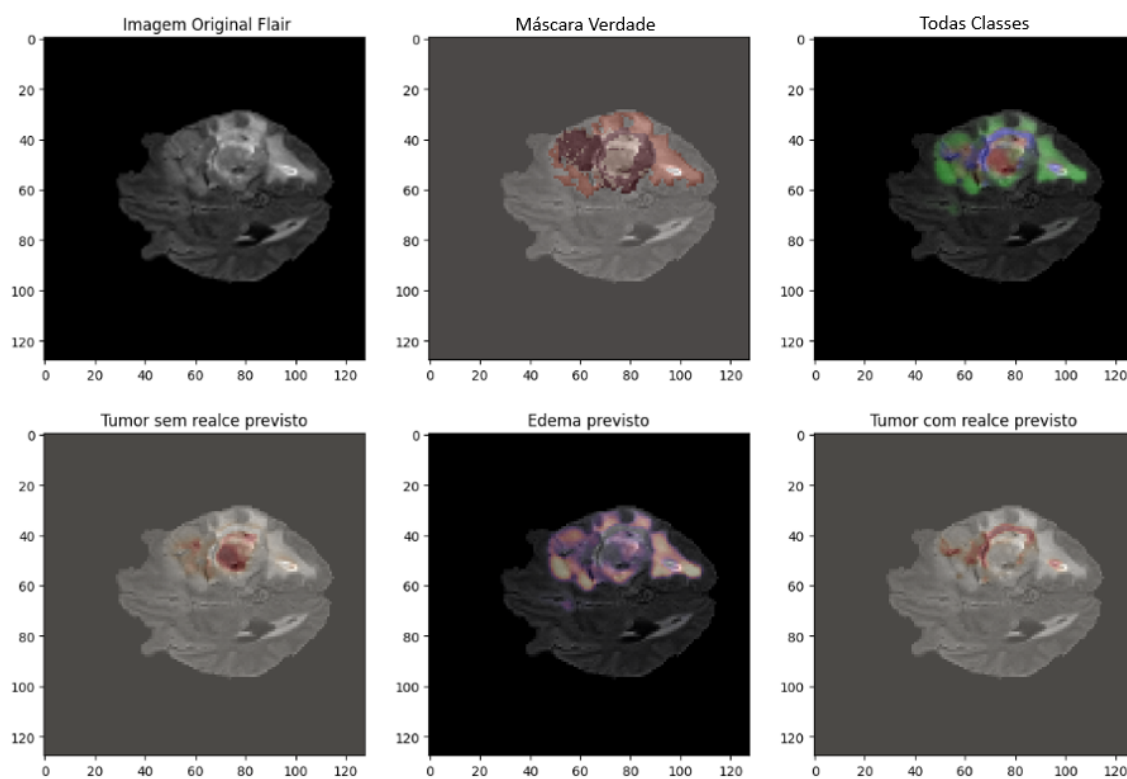


Figura 4.33: Previsão tumoral no cérebro 01 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

Na figura 4.33, a previsão da região tumoral cerebral deste paciente foi gerada utilizando uma imagem Flair e T1ce do conjunto de imagens de teste com o modelo treinado utilizando a perda combinada de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), as mesmas do paciente que produziram a Figura 4.32.

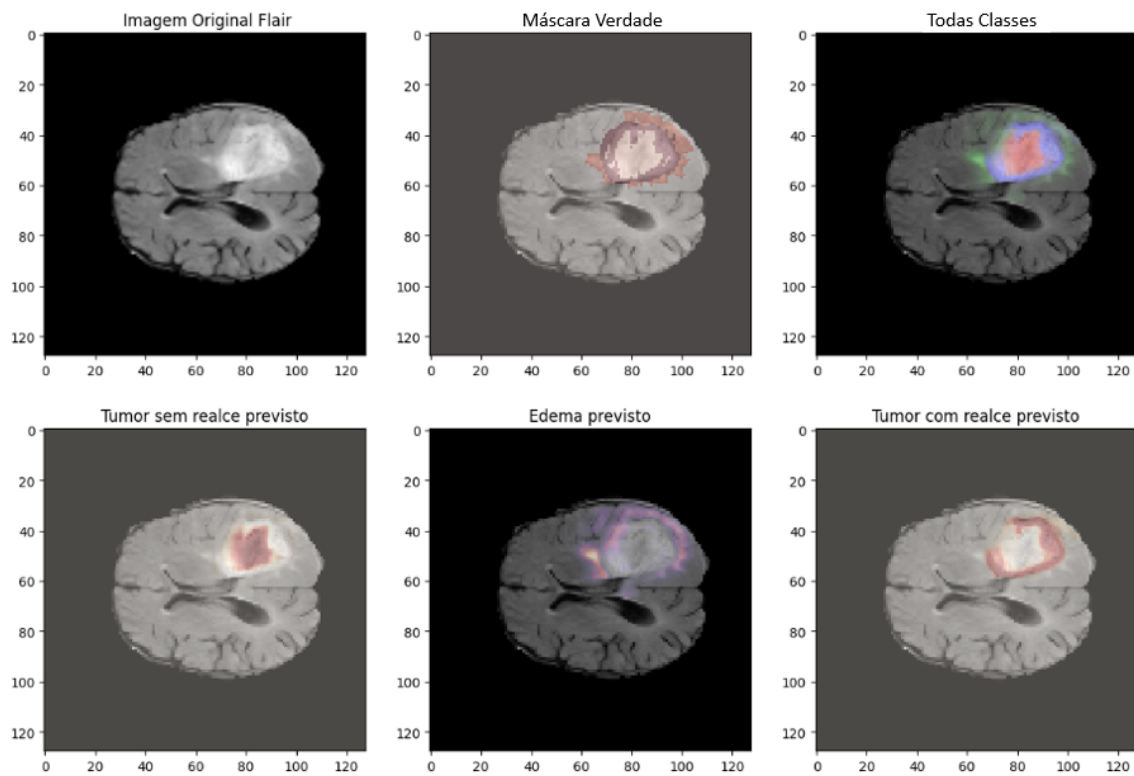


Figura 4.34: Previsão tumoral no cérebro 02 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.34, é apresentada a previsão da região tumoral cerebral de um paciente. Esta previsão foi obtida utilizando uma combinação de imagens Flair e T1ce do conjunto de dados de teste, processadas através do modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

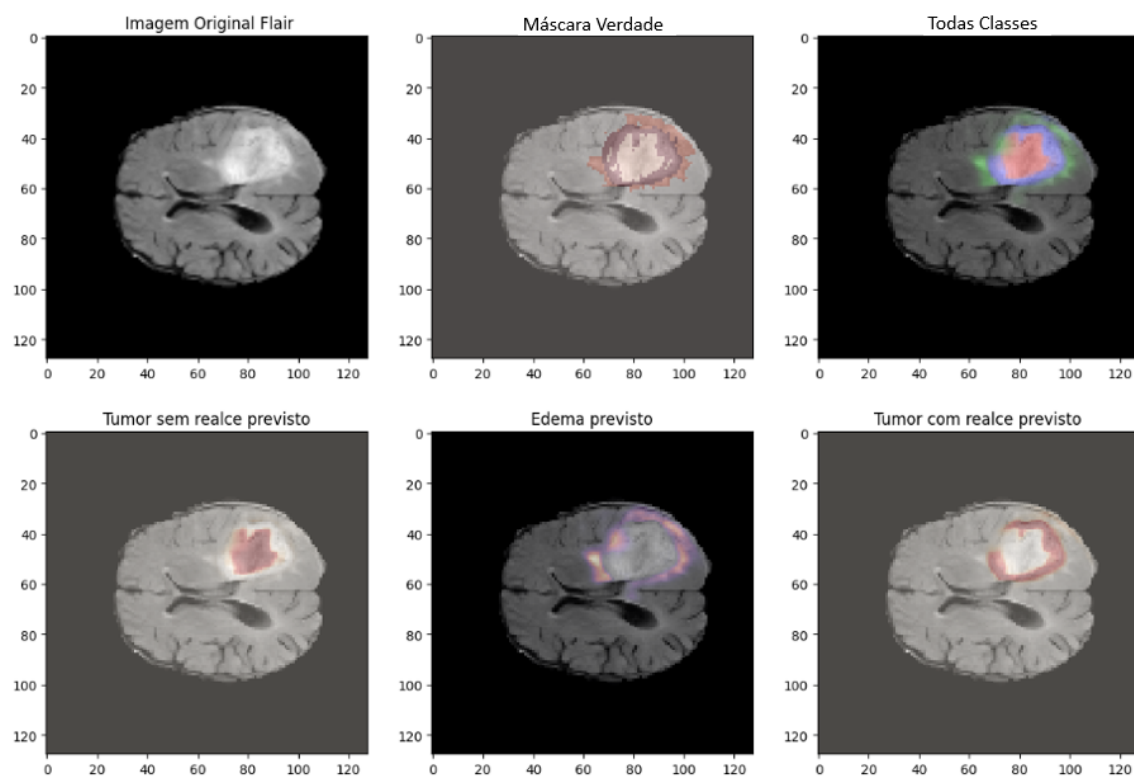


Figura 4.35: Previsão tumoral no cérebro 02 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.35, é exibida a previsão da região tumoral cerebral deste paciente. Essa previsão foi obtida a partir de uma imagem Flair e T1ce do conjunto de imagens de teste, processadas usando o modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), as mesmas do paciente que geraram a Figura 4.34.

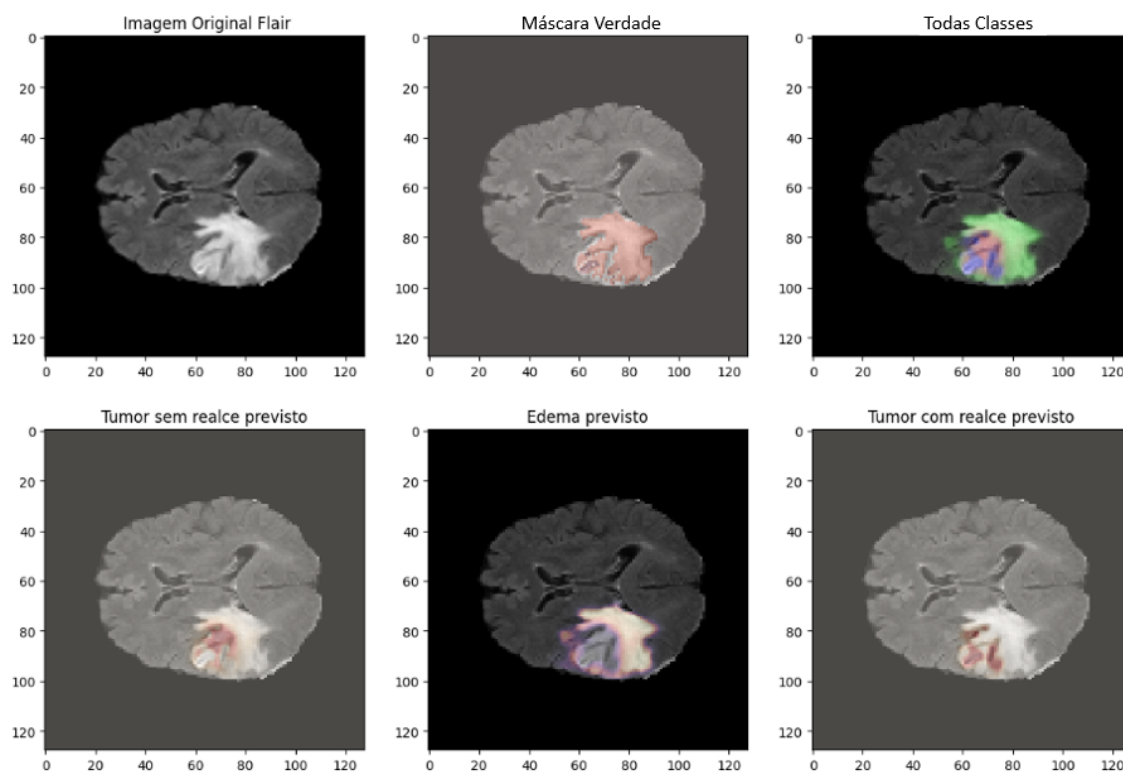


Figura 4.36: Previsão tumoral no cérebro 03 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.36, é exibida a previsão da região tumoral cerebral de um paciente. Essa previsão foi gerada usando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

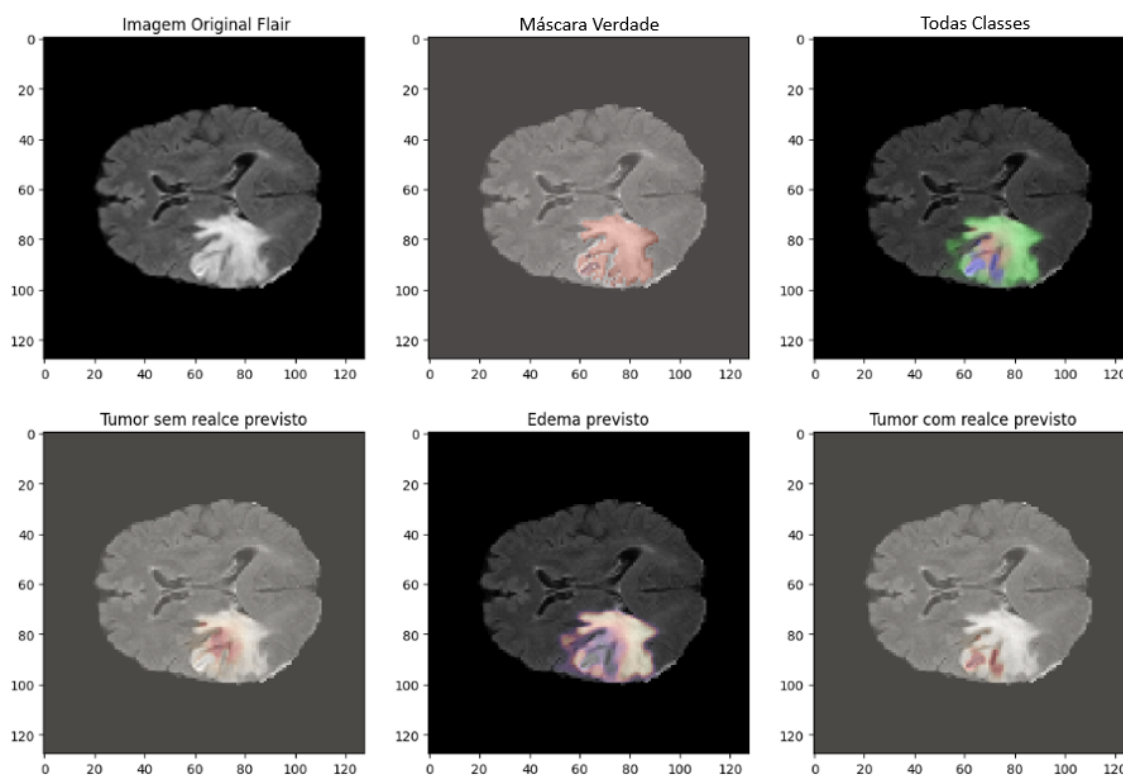


Figura 4.37: Previsão tumoral no cérebro 03 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.37, é apresentada a previsão da região tumoral cerebral deste paciente. Esta previsão foi obtida a partir de uma imagem Flair e T1ce do conjunto de imagens de teste, processadas usando o modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), as mesmas imagens utilizadas do paciente como entrada para a previsão da Figura 4.36.

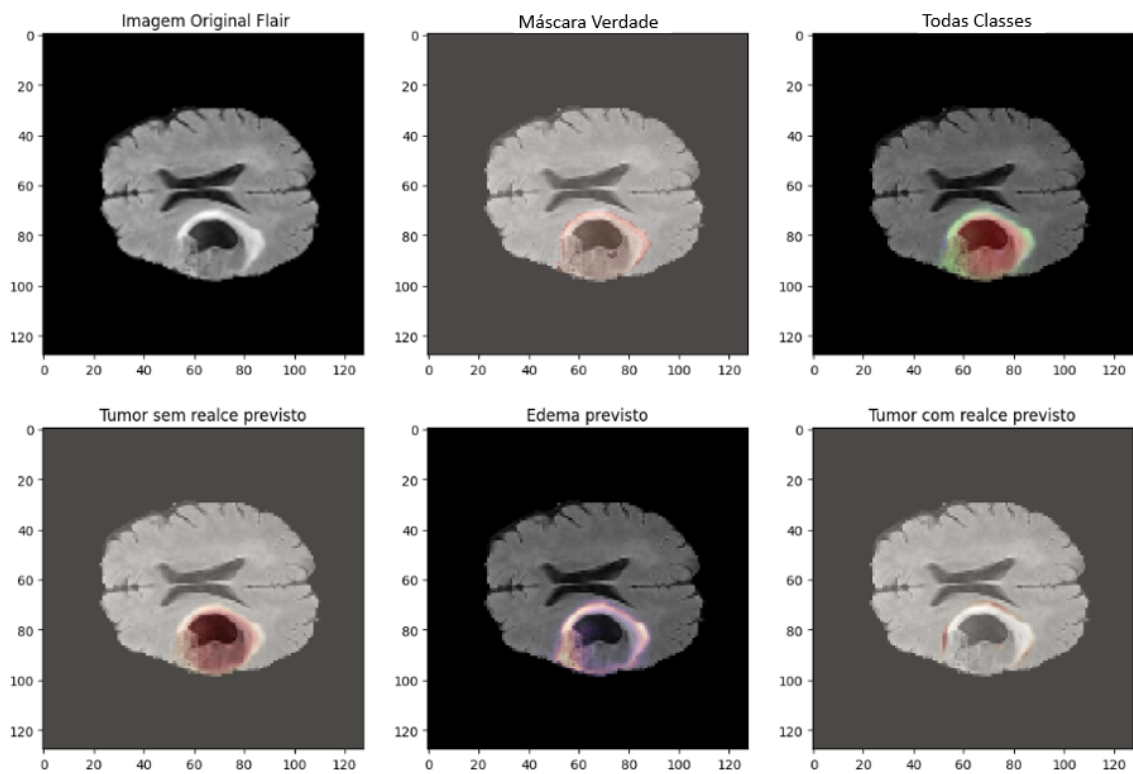


Figura 4.38: Previsão tumoral no cérebro 04 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.38, é mostrado a previsão da região tumoral cerebral de um paciente. Essa previsão foi gerada usando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

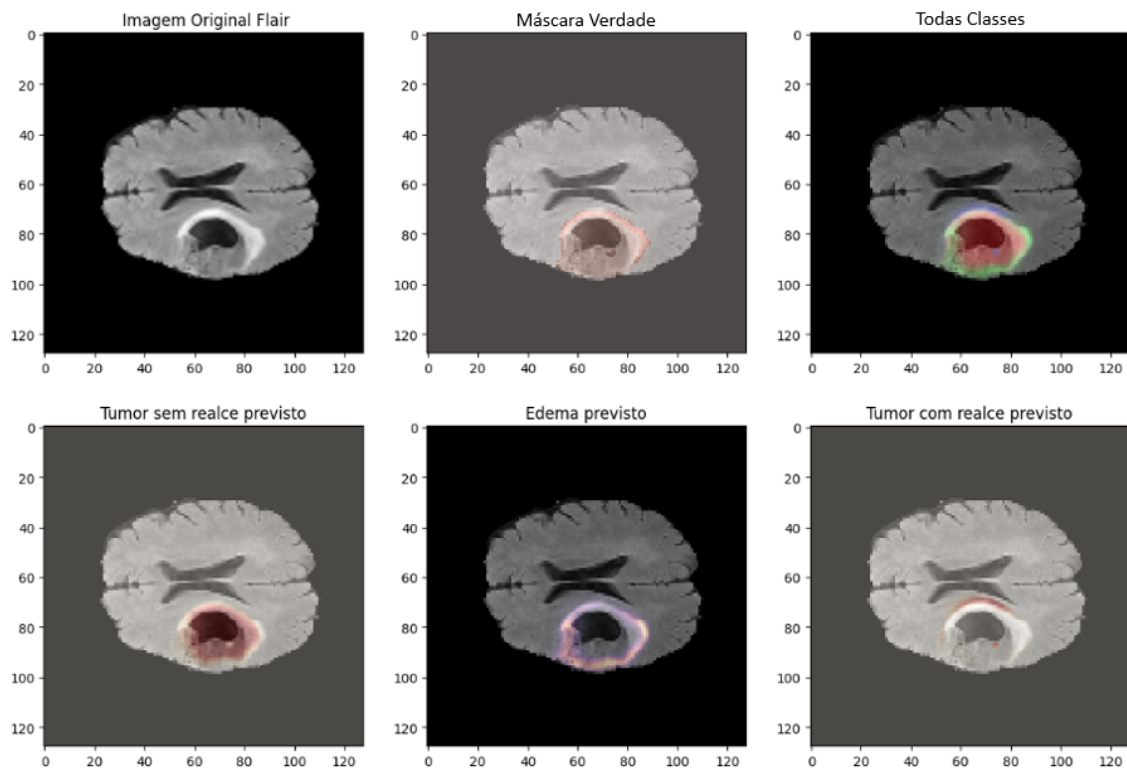


Figura 4.39: Previsão tumoral no cérebro 04 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

Na Figura 4.39, é exibido a previsão da região tumoral cerebral deste paciente. Essa previsão foi obtida a partir de uma imagem Flair e T1ce do conjunto de imagens de teste, processadas utilizando o modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), as mesmas imagens utilizadas do paciente que geraram a Figura 4.38.

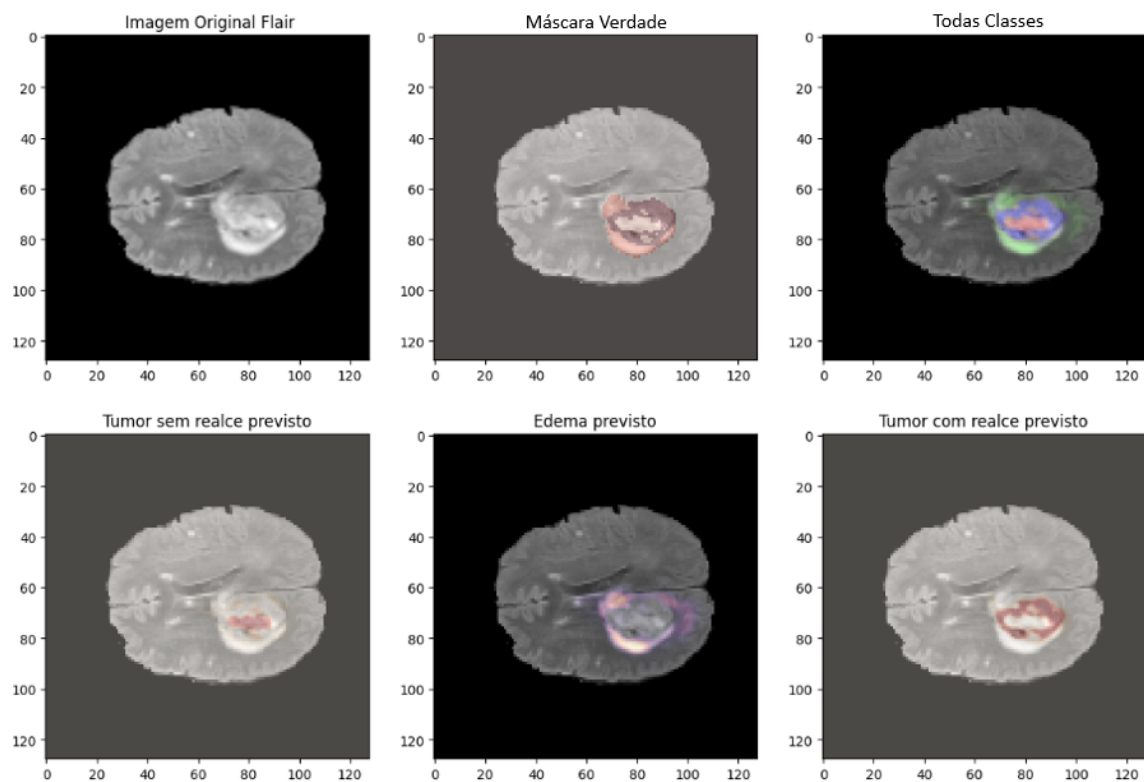


Figura 4.40: Previsão tumoral no cérebro 05 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.40, é apresentado a previsão da região tumoral cerebral de um paciente. Essa previsão foi gerada utilizando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

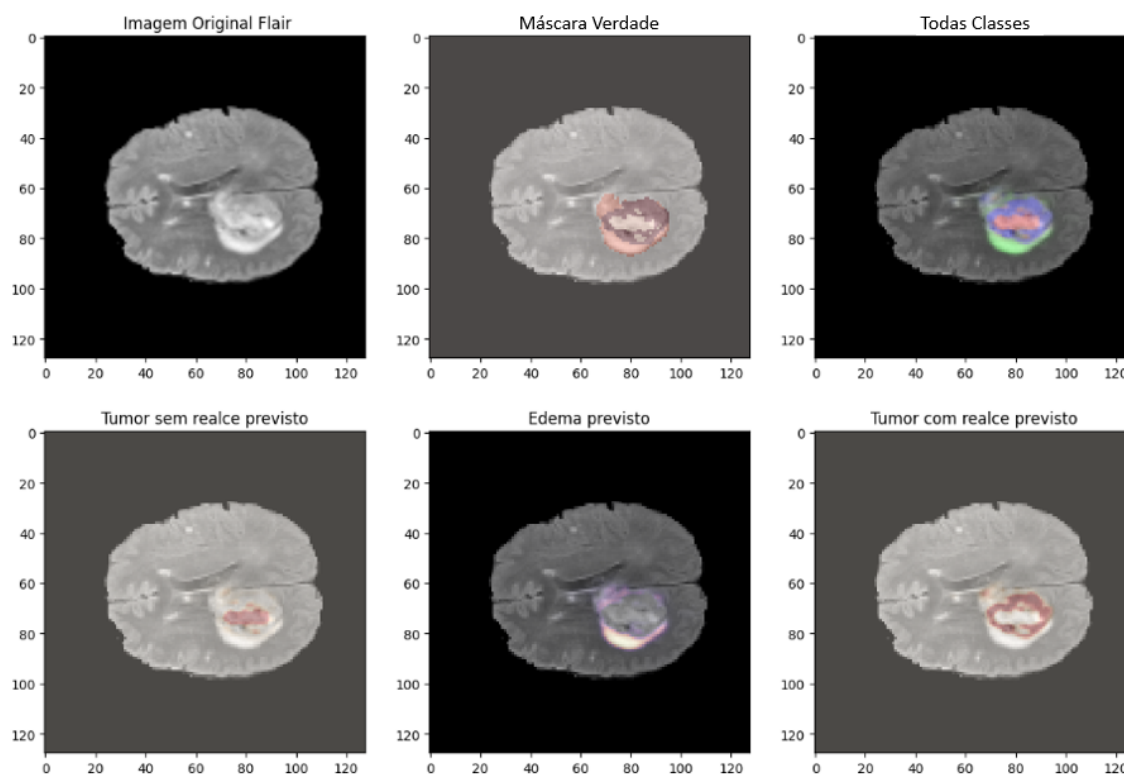


Figura 4.41: Previsão tumoral no cérebro 05 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

A Figura 4.41 apresenta a previsão da região tumoral cerebral deste paciente. Essa previsão foi gerada a partir de uma imagem Flair e T1ce do conjunto de imagens de teste, processadas pelo modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), utilizando as mesmas imagens do paciente que produziram a previsão apresentada na Figura 4.40.

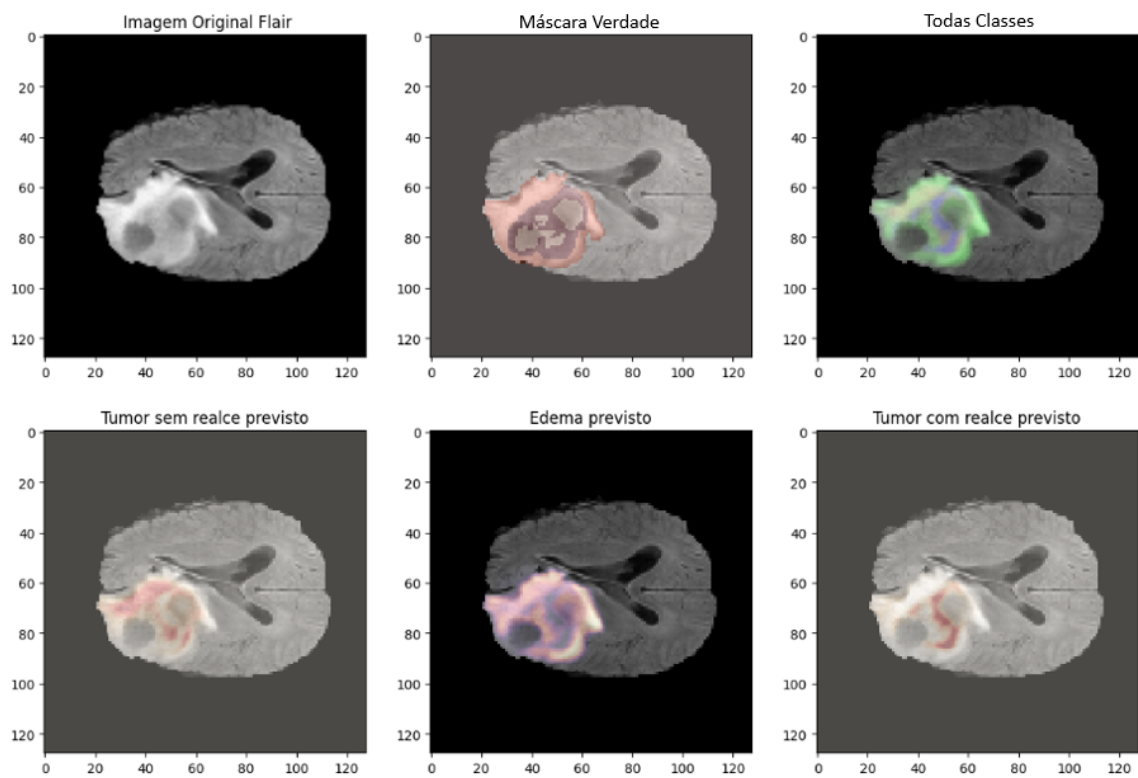


Figura 4.42: Previsão tumoral no cérebro 06 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.42, é mostrada a previsão da região tumoral cerebral de um paciente. Essa previsão foi gerada usando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

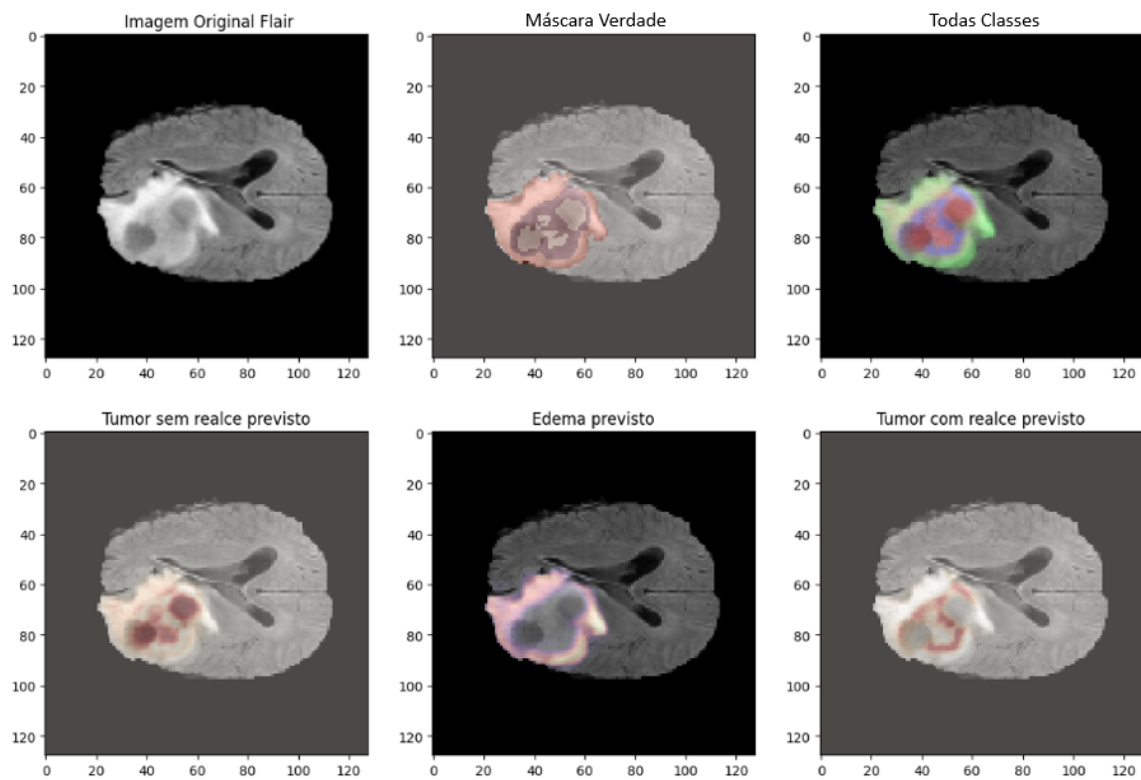


Figura 4.43: Previsão tumoral no cérebro 06 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

A Figura 4.43 mostra a previsão da região tumoral cerebral gerada a partir de uma imagem Flair e T1ce do conjunto de imagens de teste, processadas pelo modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), utilizando as mesmas imagens do paciente de entrada para a previsões apresentadas na Figura 4.42.

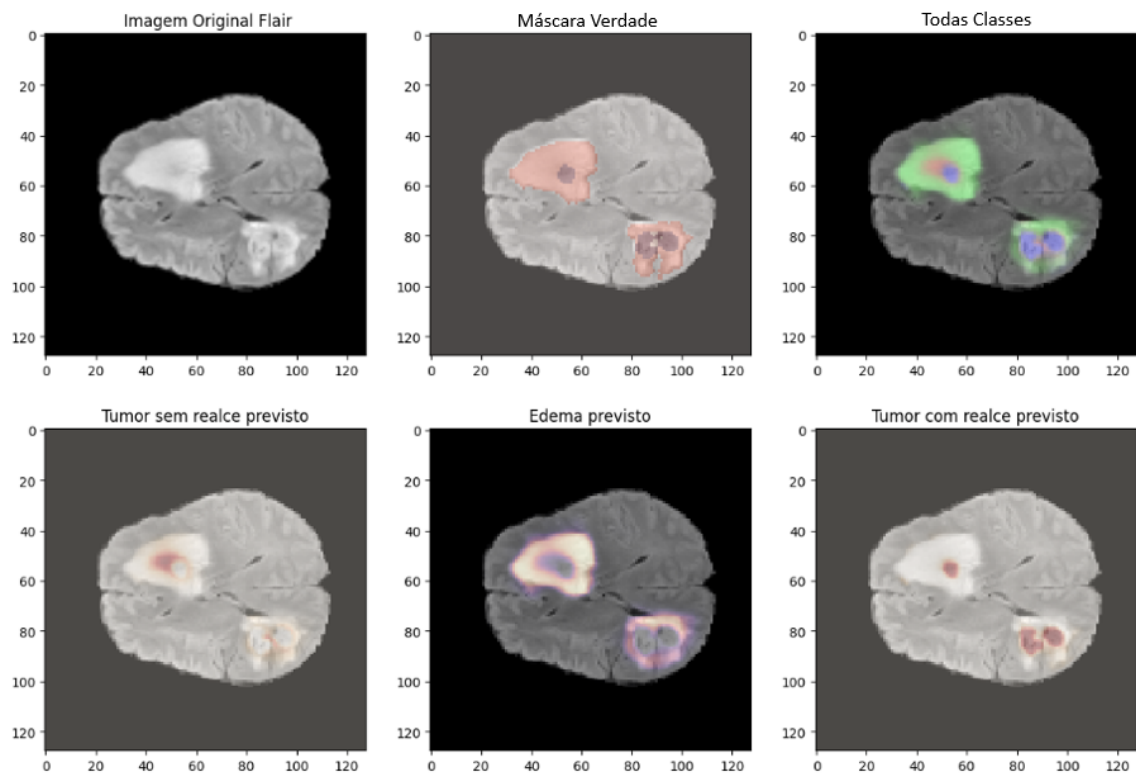


Figura 4.44: Previsão tumoral no cérebro 07 usando uma imagem com perda ECC

Fonte: Própria Autora

A Figura 4.44 exhibe a previsão da região tumoral cerebral de um paciente. Essa previsão foi gerada utilizando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

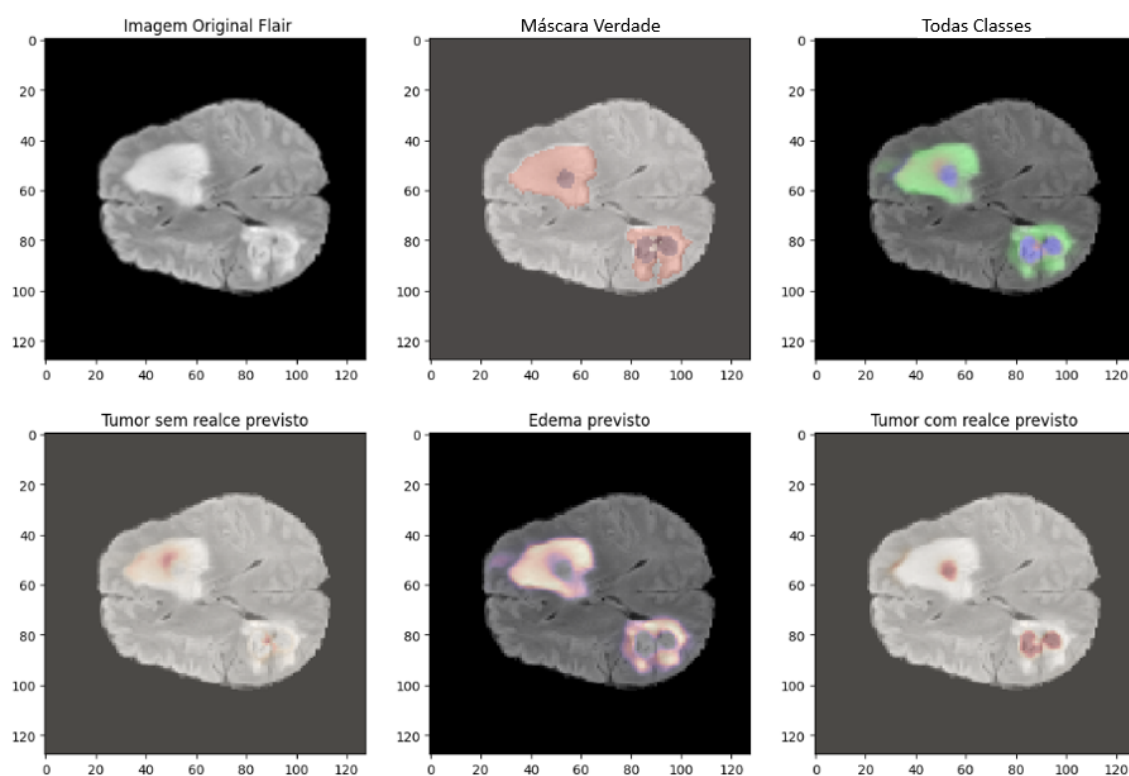


Figura 4.45: Previsão tumoral no cérebro 07 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

A Figura 4.45 apresenta a previsão da região tumoral cerebral obtida a partir de uma imagem Flair e T1ce do conjunto de imagens de teste. Essas imagens foram processadas pelo modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), utilizando os mesmos dados do paciente utilizados como entrada para as previsões produzidas na Figura 4.44.

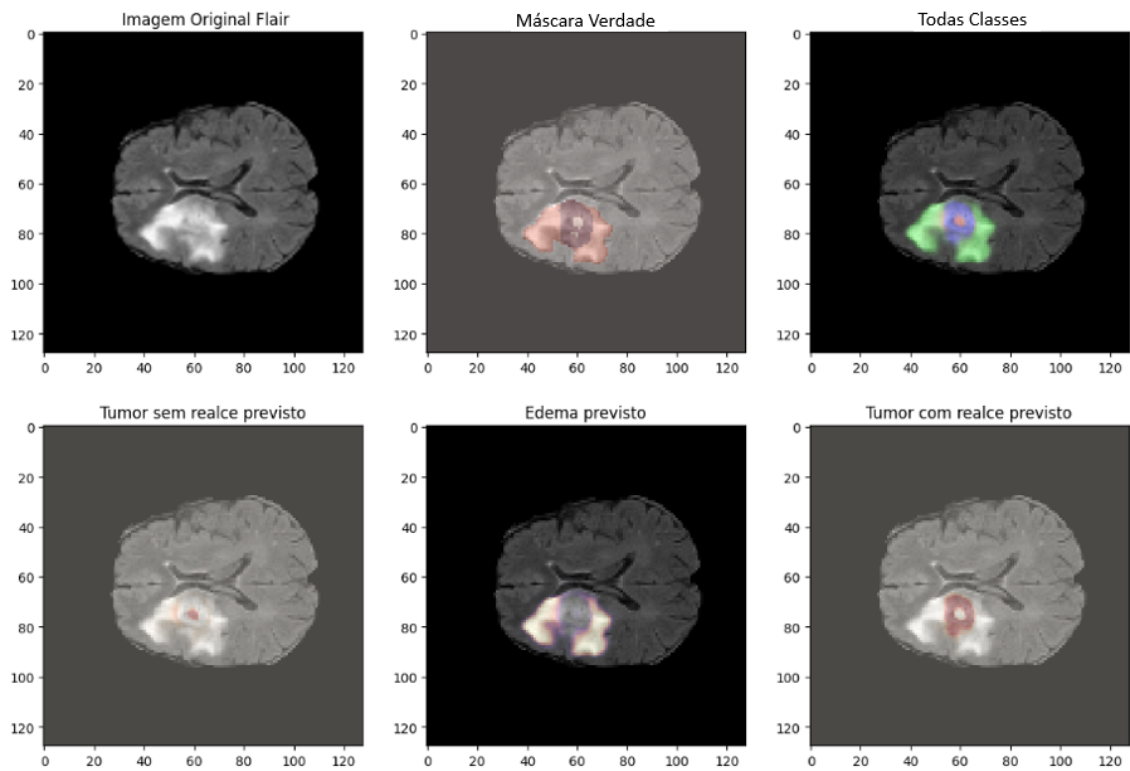


Figura 4.46: Previsão tumoral no cérebro 08 usando uma imagem com perda ECC

Fonte: Própria Autora

Na Figura 4.46, é mostrada a previsão da região tumoral cerebral foi gerada usando uma combinação de imagens Flair e T1ce do conjunto de dados de teste e processada pelo modelo treinado com a perda de Entropia Cruzada Categórica (ECC).

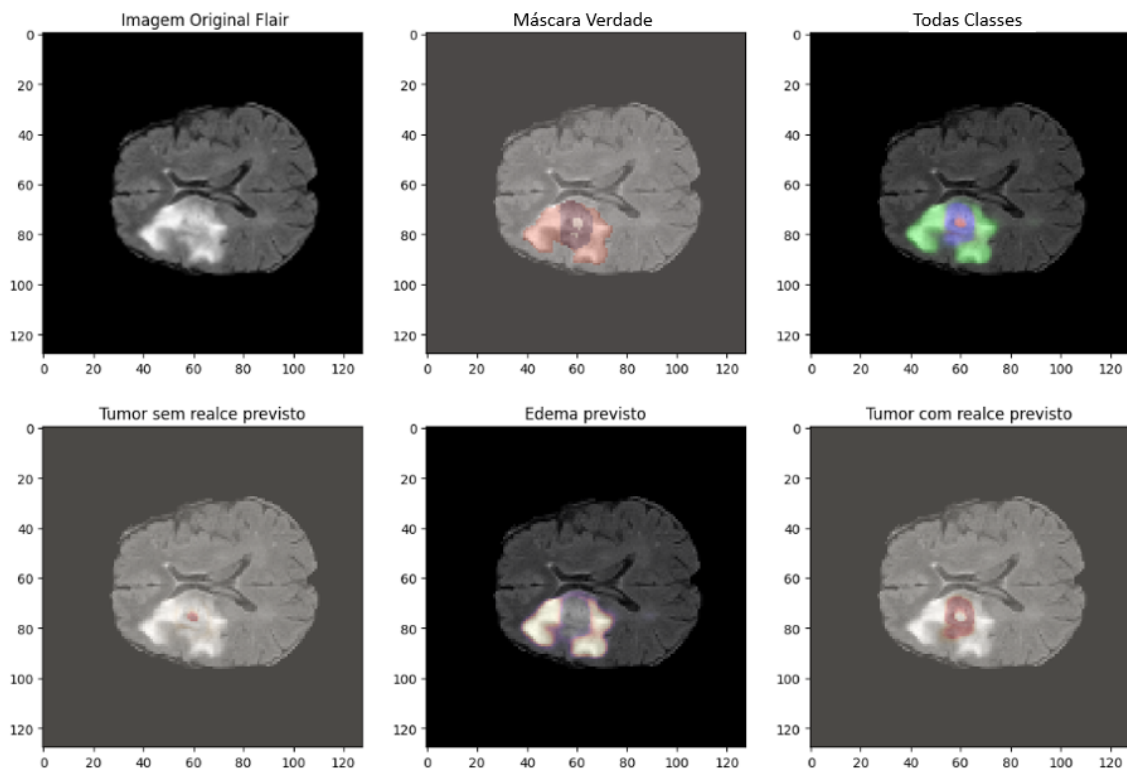


Figura 4.47: Previsão tumoral no cérebro 08 usando uma imagem com perda ECC + SDL

Fonte: Própria Autora

A Figura 4.47 exibe a previsão da região tumoral cerebral gerada a partir de uma imagem Flair e T1ce do conjunto de imagens de teste. Essa previsão foi processada pelo modelo treinado com a combinação de perdas de Entropia Cruzada Categórica (ECC) e Perda Suave Dice (SDL), utilizando os mesmos dados do paciente de entrada gerando a Figura 4.46.

Após a apresentação de algumas das previsões geradas com imagens do conjunto de teste, a próxima seção se concentrará na análise das diferenças entre as imagens produzidas pelos dois modelos treinados. São selecionados três conjuntos de imagens de exemplo para ilustrar essas diferenças e promover uma discussão mais detalhada sobre as características distintivas de cada modelo com a ampliação das diferenças das imagens.

4.3.1 Analisando Imagens das Previsões do Modelo 3D U-Net

Para uma análise visual mais detalhada das previsões geradas, esta seção apresenta e discute as diferenças mais significativas entre as previsões 1, 2 e 3.

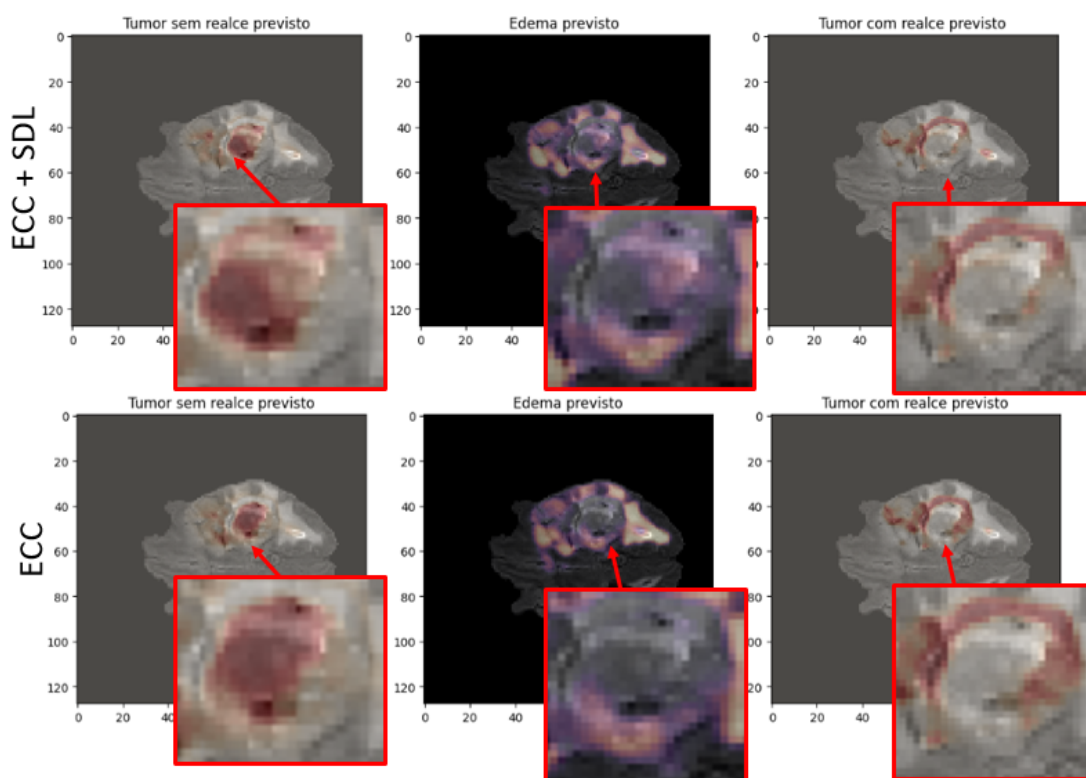


Figura 4.48: Analisando previsão tumoral no cérebro 01

Fonte: Própria Autora

Na Figura 4.48, são apresentadas as previsões geradas pelo modelo ECC e pelo modelo ECC + SDL. Observam-se diferenças notáveis nas imagens ampliadas, especialmente nos diversos rótulos. A imagem com rótulo de Tumor Sem Realce gerada pelo modelo ECC + SDL é significativamente menor do que a apontada pelo modelo ECC. Em contraste, na imagem do Edema, o modelo ECC + SDL identifica uma região mais extensa do que a apontada pelo modelo ECC. Para a imagem de Tumor com Realce, o modelo ECC + SDL delimita uma área menor em comparação com a apresentada pelo modelo ECC.

É crucial determinar a extensão precisa de cada área onde o tumor cancerígeno está presente, tanto para a escolha do melhor tratamento para o paciente quanto para a localização precisa onde a radioterapia será aplicada. Na análise da Figura 4.48, segundo o modelo proposto ECC + SDL, este paciente apresenta mais edema, uma região necrosada menor e uma área de tumor irrigado menor em comparação com as identificações feitas pelo modelo padrão ECC.

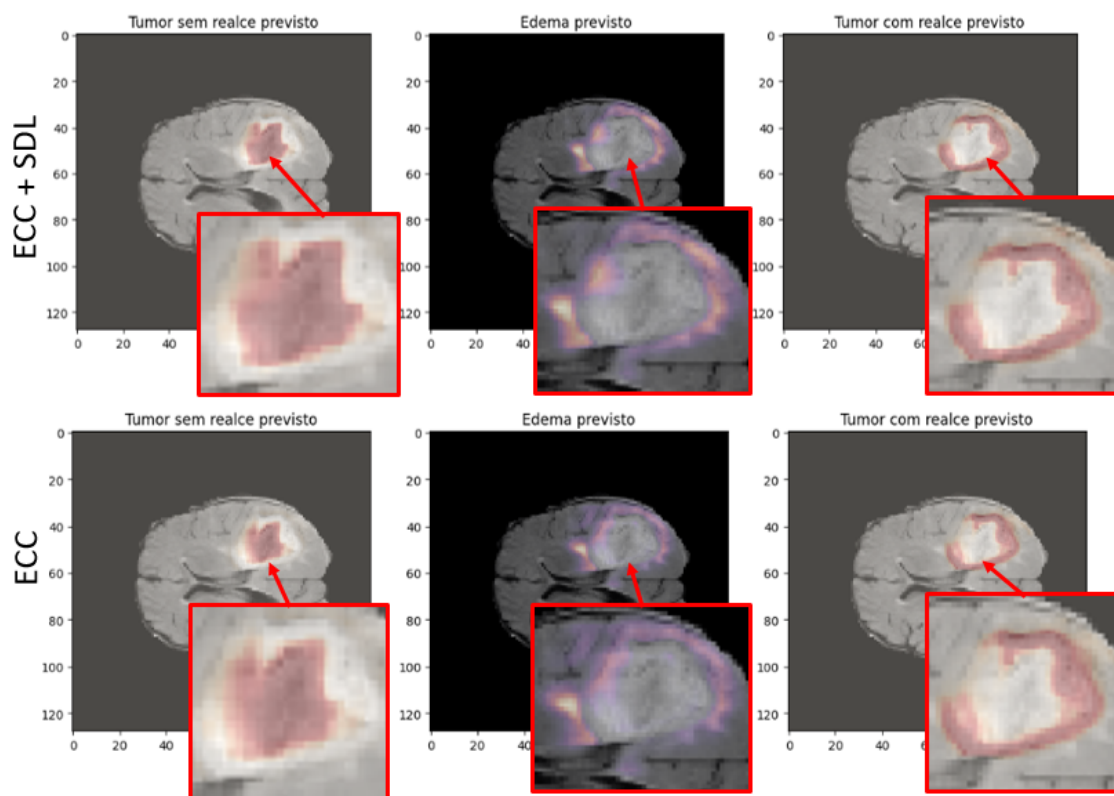


Figura 4.49: Analisando previsão tumoral no cérebro 02

Fonte: Própria Autora

A Figura 4.49 exibe as previsões geradas pelos modelos ECC e ECC + SDL. Observam-se algumas diferenças notáveis nas imagens ampliadas entre as previsões dos diferentes rótulos. Notadamente, o rótulo do Tumor Sem Realce no modelo ECC + SDL é significativamente maior e com contornos mais claros em comparação ao modelo ECC. O Edema é mais destacado no modelo ECC + SDL, realçando sua localização com maior precisão. Além disso, o Tumor com Realce previsto pelo modelo ECC + SDL apresenta uma melhor delimitação, com contornos mais fortes e uma região mais fina, em relação às imagens previstas pelo modelo padrão ECC.

É importante destacar que esta é apenas uma fatia de uma imagem cerebral. Ao considerar todas as fatias a serem analisadas, pequenas diferenças podem se tornar significativas ao processar o volume completo do cérebro. Essas sutis variações, quando somadas, são fundamentais para a tomada de decisões sobre tratamentos e para a identificação precisa da localização do tumor e suas regiões de infiltração.

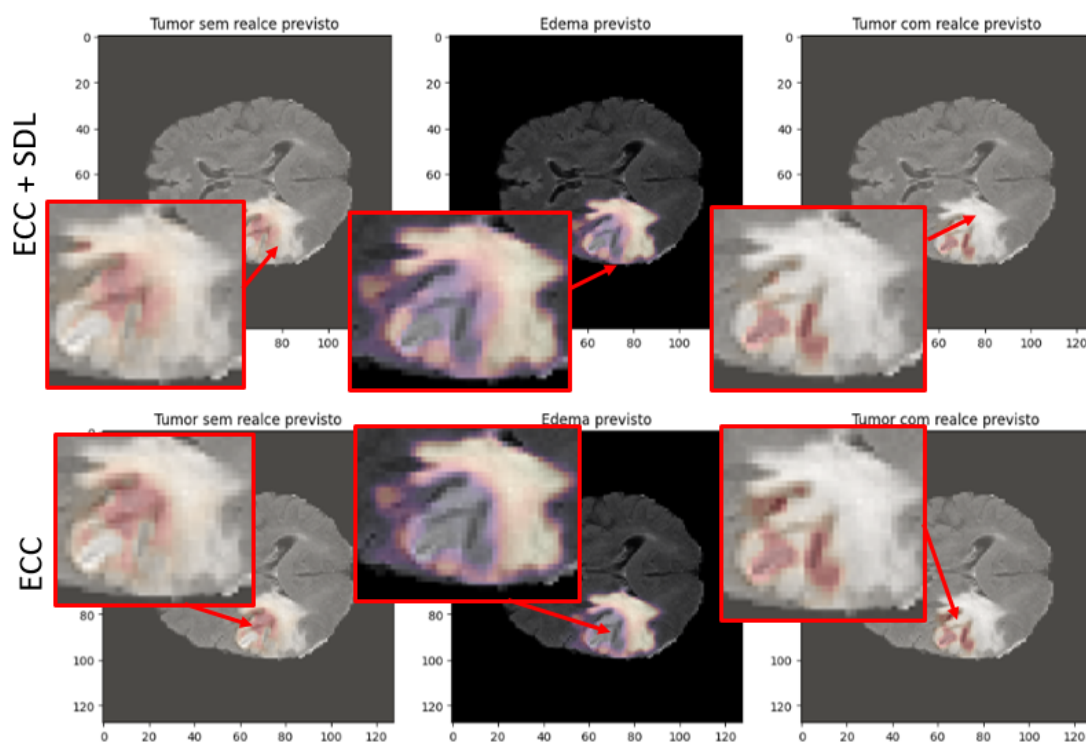


Figura 4.50: Analisando previsão tumoral no cérebro 03

Fonte: Própria Autora

A Figura 4.50 apresenta as previsões feitas pelos modelos ECC e ECC + SDL. Nota-se pequenas diferenças nas imagens ampliadas entre as previsões dos diferentes rótulos. As imagens previstas do Tumor Sem Realce pelo modelo ECC + SDL são ligeiramente menores em comparação com as geradas pelo modelo ECC. O Edema previsto pelo modelo ECC + SDL é maior e apresenta mais áreas destacadas do que o modelo ECC. Além disso, o rótulo do Tumor com Realce no modelo ECC + SDL é menor e sem todos os pontos de infiltração de tumor em relação ao previsto pelo modelo ECC.

Diferenças nas previsões podem significar que o modelo ECC + SDL está capturando características das imagens que o modelo ECC não está, potencialmente resultando em uma segmentação mais exata e detalhada. A melhor delimitação das regiões de interesse no modelo ECC + SDL pode indicar uma maior sensibilidade na identificação das bordas, algo crucial para o planejamento personalizado e ajustado ao tratamento do paciente. A observação de alguma da classe do tumor mais destacada pode significar que o modelo ECC + SDL está identificando com maior clareza áreas afetadas que podem não ser tão evidentes no modelo ECC. Essas variações destacam a importância da escolha da função

de perda no desempenho do modelo de segmentação de imagens médicas.

Na sequência, após a conclusão do Capítulo de Resultados, é apresentada as Conclusões.

Capítulo 5

Conclusões

O objetivo da pesquisa foi aprimorar a precisão na detecção de regiões tumorais geradas pelo modelo com uma nova proposta de cálculo de perda. Para alcançar esse objetivo, foi desenvolvido um modelo treinado com a arquitetura 3D U-Net, criando-se uma perda personalizada com a combinação da Perda de Entropia Cruzada Categórica e da Perda Suave Dice durante o treinamento. Como resultado, foi observada uma melhora significativa na precisão das previsões, demonstrando a eficácia da abordagem personalizada na detecção de tumores cerebrais em imagens de ressonância magnética.

A avaliação das métricas de desempenho para os dois modelos, um utilizando Perda Entropia Cruzada Categórica e o outro utilizando a perda combinada de Entropia Cruzada Categórica com a Perda Suave Dice, revelou resultados bastante relevantes ao considerar todas as métricas disponíveis. Ambos os modelos demonstraram eficiência no treinamento, com resultados próximos ao comparar as métricas de treinamento e validação, como evidenciado nos gráficos apresentados.

A proximidade entre as métricas de treinamento e validação indica a capacidade do modelo de generalizar para novos dados de forma precisa e robusta, sugerindo que ele está aprendendo padrões significativos nos dados sem superestimar os exemplos de treinamento.

Além disso, a leve discrepância entre a perda durante o treinamento e a perda durante a validação não sugere a presença de sobreajuste (*overfitting*), onde o modelo se adapta excessivamente aos dados de treinamento, comprometendo sua capacidade de generalização.

O modelo ECC + SDL converge mais rapidamente do que o modelo ECC, sugerindo que precisa de muito menos tempo para treinamento.

As previsões geradas pelo modelo proposto ECC + SDL apresentaram pequenas diferenças nas imagens ao comparar as previsões geradas pelo modelo ECC. Mas, é importante lembrar que mesmo com diferenças pequenas visualizadas em uma fatia, ao considerar todas as imagens que compõe o volume total do cérebro, podem ter grande relevância para escolha do melhor tratamento indicado para o paciente, a identificação precisa da localização do tumor em cada tipo de tecidos e infiltrações.

No contexto das previsões dos modelos ECC e ECC + SDL, é observada uma potencial capacidade aprimorada do modelo ECC + SDL em delimitar as regiões de interesse, indicando uma sensibilidade maior na identificação de bordas e áreas de relevância clínica, aspectos cruciais para o planejamento personalizado e ajustado ao tratamento do paciente. Além disso, essa diferença pode sugerir que o modelo ECC + SDL está capturando aspectos das imagens que escapam à detecção do modelo ECC, possivelmente resultando em uma segmentação mais detalhada e precisa. A observação de uma classe do tumor mais evidente no modelo ECC + SDL pode indicar uma eficácia superior na identificação de áreas afetadas que podem não ser tão perceptíveis no modelo ECC.

O objetivo deste estudo foi alcançado com os resultados de desempenho superior das métricas obtidas pelo modelo proposto ECC + SDL, juntamente com as diferenças observadas nas previsões geradas pelo modelo proposto em comparação ao modelo padrão ECC. Isso destaca a importância da escolha da função de perda no desempenho do modelo de segmentação de imagens médicas. A utilização de uma função de perda combinada, como a Entropia Cruzada Categórica (ECC) e a Perda Suave Dice (SDL), parece conferir ao modelo uma capacidade aprimorada de distinguir entre diferentes tipos de tecidos e anomalias, contribuindo para diagnósticos mais precisos e tomadas de decisão clínicas.

Em trabalhos futuros recomenda-se aprimorar o estudo por meio da colaboração com um especialista em imagens de ressonância magnética para uma avaliação mais abrangente das previsões do modelo. Essa parceria possibilitará uma análise mais detalhada, permitindo determinar se o modelo consegue capturar com precisão as nuances presentes nos dados, resultando em previsões mais confiáveis para a detecção de regiões tumorais em imagens médicas. Baseado na análise dos modelos treinados que revela que aquele que utiliza a perda combinada apresenta melhores resultados em relação às métricas avaliadas, ressalta a importância contínua da pesquisa nesta área para aprimorar os resultados já alcançados no treinamento de modelos de aprendizado profundo. Especificamente, enfatiza a necessidade de explorar e desenvolver estratégias personalizadas de perda para otimizar o desempenho dos modelos em tarefas específicas, como a detecção de regiões

tumorais em imagens médicas. Esses esforços contribuem significativamente para o avanço da área médica e o aprimoramento das técnicas de diagnóstico assistido por computador.

Referências

- Ali, M., Alomari, S., Bennamoun, M., Sohel., F. **A novel 3D U-Net architecture for real-time segmentation of MRI brain volumes.** Journal of Ambient Intelligence and Humanized Computing, pp. 657-666, 2020.
- Araújo, E. R., Garcia, V. S., Assis, W. L. S., **Segmentação de Imagens de Ressonância Magnética de Tumor Cerebral Utilizando Modelo 3D U-Net.** Revista Tecnologia & Cultura, Rio de Janeiro. Edição Especial 2024, pp. 40-49, 2024 . Disponível em: < <https://acrobat.adobe.com/id/urn:aaid:sc:US:fddd8c1c-d181-49c3-9fe8-e6b4edf90d32?viewer%21megaVerb=group-discover>>. Acesso em: 26 de agosto de 2024.
- Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J. **Advancing The Cancer Genome Atlas glioma MRI Collections with Expert Segmentation Labels and Radiomic Features.** Nature Scientific Dados, 2017. DOI:10.1038/sdata.2017.117.
- Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A. **Identificando os melhores algoritmos de aprendizado de máquina para segmentação de tumor cerebral, avaliação de progressão e previsão de sobrevivência geral no Desafio BRATS.** 2018. arXiv:1811.02629.
- Bishop, C. M. **Pattern Recognition and Machine Learning.** Springer, 2006.
- Booth, J. C., Wendt, R. E., Zabowski, M., Koutcher, J. A., Gore, J. C. **Three-dimensional magnetization-prepared rapid gradient-echo imaging (3D MP-RAGE): a new high-resolution MRI sequence for anatomical brain imaging.** Journal of magnetic resonance imaging, pp. 959-965, 1999.
- Bradley, W. G. **MRI: Basic Principles and Applications.** 2009.
- Chollet, F. e. a. **Keras Documentation.** 2020. Disponível em:<https://keras.io/api/losses/probabilistic_losses/categorical_crossentropy/>. Acesso em 28 de maio de 2023.

- Cox, R. W., Ashburner, J., Breman, H., Fissell, K., Haselgrove, C., Holmes, C. J. and Woodbury, M. A. **A (symbolic) review of linear and nonlinear operations in neuroimaging data analysis.** Journal of Cerebral Blood Flow Metabolism, pp. 684-693, 2004.
- Damadian, R. **Magnetic Resonance Imaging of the Brain and Spinal Cord.** Science, 210(4470), pp. 688-690, 1980.
- Dey, N., Nair, S., Paul, M., Bhattacharya, I. **Automatic tumor detection and segmentation in brain MR images using 3D U-Net.** Journal of Ambient Intelligence and Humanized Computing, 2020.
- DICE, L. R. **Measures of the amount of ecologic association between species.** Ecology, n°3, 26, pp. 297-302, 1945.
- DICOM. **DICOM Standard.** DICOM Standards Committee, 1983. Disponível em: <<https://www.dicomstandard.org/>>. Acesso em 28 de maio de 2023.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A. **The PASCAL Visual Object Classes (VOC) Challenge.** International Journal of Computer Vision., 2010.
- Gevins, G., Allen, B., Lander, L., Nguyen, D., Li, J., Smith, A. **Deep learning-based segmentation of brain tumors using 3D U-Net architecture.** Medical Imaging 2020: Image Processing. In . International Society for Optics and Photonics, 2020.
- Goodfellow, I., Bengio, Y., Courville, A. **Deep Learning.** MIT Press, 2016.
- Greve, D. N., Fischl, B. **Probabilistic atlas-based segmentation of brain MRI scans.** Neuroimage, pp. 1081-1091, 2003.
- Géron, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.** O'Reilly Media, 2017.
- Haacke, A., Brown, R. W., Thompson, M. R., Venkatesan, R., Cheng, Y.-C. N. **Magnetic Resonance Imaging: Physical Principles and Sequence Design.** 1999.
- Hanke, T., Cannon, R. D., Pierpaoli, C., Styner, M. **A survey of image analysis and data processing methods for quantitative analysis of postmortem brain tissue.** Neuroimage, 13(6), pp. 1021-1035, 2001.
- Hastie, T., Tibshirani, R., Friedman, J. **The Elements of Statistical Learning: Data**

- Mining, Inference, and Prediction.** Springer Science Business Media, 2009.
- Jaccard, P. **Étude comparative de la distribution florale dans une portion des Alpes et des Jura.** Imprimerie Corbaz, 1901.
- Jackson, A., Buckley, D. L., Parker, G. J. **Magnetic Resonance Imaging: Clinical Principles.** 2019.
- James, G., Witten, D., Hastie, T., Tibshirani, R. **An introduction to statistical learning.** New York: springer, vol. 18, pp. 18, 2013.
- Knoll, F., Hammernik, K., Kobler, E., Pock, T., Recht, M. P. **fastMRI: An Open Dataset and Benchmarks for Accelerated MRI.** arXiv preprint arXiv:1811.08839, 2019.
- Krizhevsky, A., Sutskever, I., Hinton, G. E. **ImageNet Classification with Deep Convolutional Neural Networks.** Advances in Neural Information Processing Systems, 2012.
- Lauterbur, P. **Imaging Future: The Dawn of Magnetic Resonance Imaging.** Nature, pp. 200-201, 1973.
- Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. **Gradient-based learning applied to document recognition.** Proceedings of the IEEE, 1998. 2278-2324.
- Lim, K. O., Yu, T., Kim, J., Lee., J. **A comparative study of image segmentation algorithms for segmenting brain MR images.** Journal of Digital Imaging, pp. 155-165, 2004.
- Louis, D. N., Perry, A., Reifenberger, G., von Deimling, A., Figarella-Branger, D., Cavenee, W. K., Ellison, D. W. **The 2016 World Health Organization Classification of Tumors of the Central Nervous System: a summary.** Acta Neuropathologica, 2016.
- Maldjian, J. A., Lattarulo, C. R., Hurtig, G. I. **VBM—a brief overview of a hot topic.** Neuroimage, 19(2), pp. 123-126, 2003.
- Marques, J., Cerqueira, P., Silveira, A., Oliveira, J., Rocha, M., Silva., L. **Automatic 3D brain MRI segmentation using a new object-based active surface model.** NeuroImage, pp. 662-672, 2005.
- Menze, B., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J. **The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS).** IEEE

Transactions on Medical Imaging, 2015. DOI: 10.1109/TMI.2014.2377694.

Milletari, F., Navab, N., Ahmadi, S. **V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation**. 2016 Fourth International Conference on 3D Vision (3DV), pp. 565-571, 2016.

Mitchell, T. M. **Machine Learning**. McGraw Hill, 1997.

NIMH. **The NIFTI Documentation**. National Institute Mental Health, 2001. Disponível em: <https://nifti.nimh.nih.gov/nifti-1/documentation/niftifields/niftifields_pages/dim/>. Acesso em 28 de maio de 2023.

Ostrom, Q. T., Patil, N., Cioffi, G., Waite, K., Kruchko, C., Barnholtz- Sloan, J. S. **CBTRUS Statistical Report: Primary Brain and Other Central Nervous System Tumors Diagnosed in the United States in 2013–2017**. Neuro-Oncology, 2020.

Ronneberger, O., et al. **Binarized Convolutional Neural Networks for Strong Inference in Medical Imaging**. International Conference on Medical Image Computing and Computer-Assisted Intervention, 2016.

Ronneberger, O., Fischer, P., Brox, T. **U-Net: Convolutional Architecture for Biomedical Image Segmentation**. International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015.

Runge, V. M., Nitz, W. R. **Magnetic Resonance Imaging: Clinical Principles**. 2015.

Saad, Z. S., Maes, F., Gholamrezanezhad, A. **Magnetic Resonance Imaging**. In Diagnostic Imaging, pp. 21-40, 2006.

Salehi, A., Mahmoodi, H., Baluch, M., Dehghani, N., Mahdavi, A., Bayat, M., Nahavandi, S. **Automatic segmentation of brain tumors in MRI using 3D U-Net**. Journal of Medical Imaging, 2021.

Seitz, R. W., Beckey, W. E., Bockus, E. A. **A new MRI technique for investigating the living brain**. NeuroImage 3.4, pp. 297-302, 1995.

Sokolova, M., Lapalme, G. **A systematic analysis of performance measures for classification tasks**. Information Processing Management, 45(4), pp. 427-437, 2009.

Stark, D. D., Bradley, W. G. **Magnetic resonance imaging**, 3 ed. Elsevier, Ed. 3,

2013.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M. J. **Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations.** Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 240-248, 1901.

Torres, R. **Fundamentos de processamento de imagens.** Editora Pearson, 2007.

Wen, P. Y., Kesari, S. **Malignant gliomas in adults.** New England Journal of Medicine, 2008.

Zhang, X., Chen, Y., Zhang, L., Chen, D., Feng, C., Li, S. **Deep learning-based segmentation of gliomas in 3D MR images using U-Net architecture.** Medical Physics, 2021.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., Ronneberger, O. **3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.** International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 424-432, 2016.

APÊNDICE A - Figuras de Detalhamento

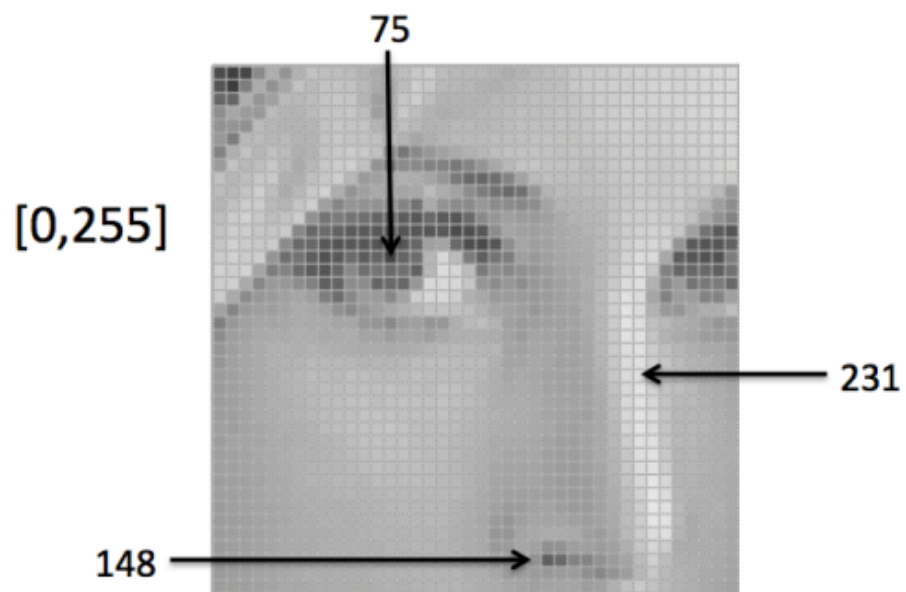


Figura A.1: Imagem e o pixel

Fonte: Adaptado de Braz (2015)

A Figura A.1 apresenta a imagem como uma tabela, onde cada pixel tem um valor correspondente a sua característica.



Figura A.2: Exemplo de exame
Fonte: Adaptado de Ultrarad (2024)

Na Figura A.2 é apresentado um exemplo de como é realizado o exame de ressonância magnética.



Figura A.3: Explicando modelo
Fonte: Própria Autora

Na Figura A.3 mostra a algumas das etapas importantes na arquitetura 3D U-Net. Resumidamente:

1. Na convolução um filtro é aplicado à imagem de entrada para extrair características específicas.
2. As funções de ativação permitem que pequenas mudanças nos pesos e viés causem apenas uma pequena alteração na saída. Esse é o fato crucial que permitirá que

uma rede de neurônios artificiais aprenda. Elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação é relevante ou deve ser ignorada.

3. Camadas de pooling são usadas em redes neurais convolucionais (CNNs) para reduzir as dimensões espaciais da entrada, ajudando a diminuir a carga computacional e controlar o overfitting.

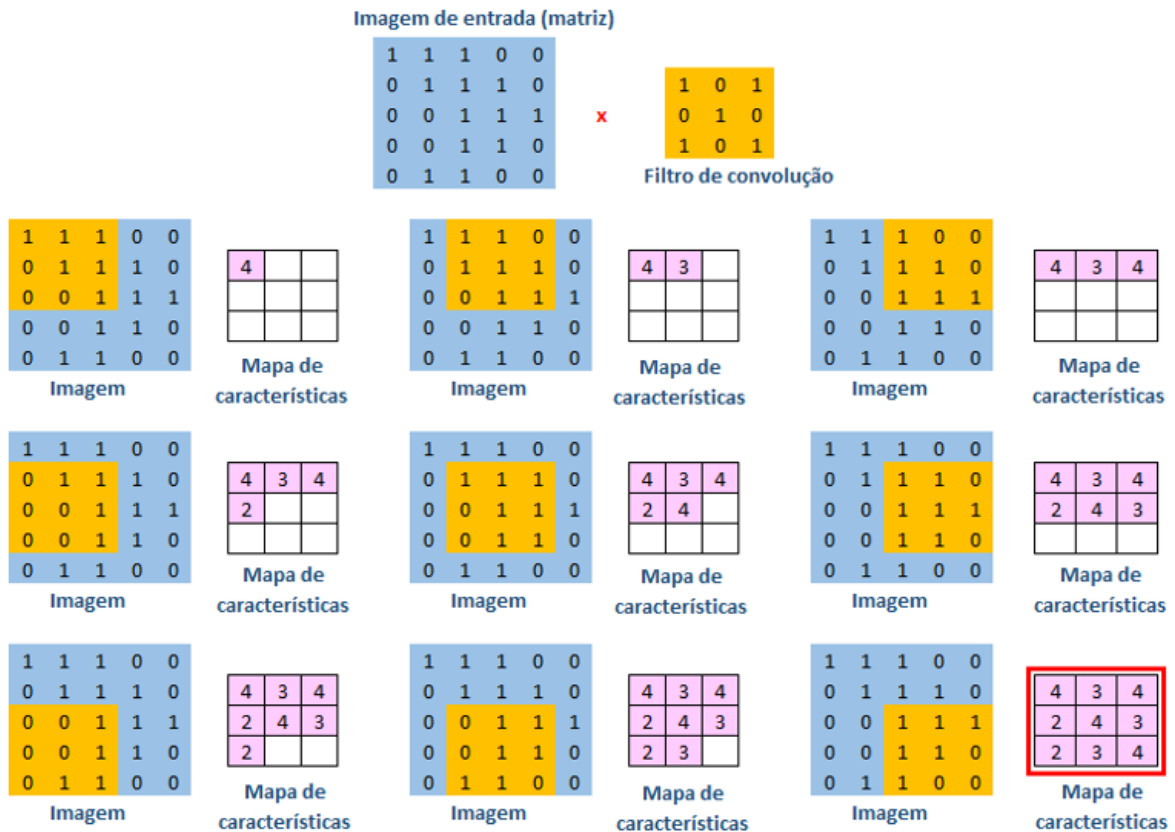


Figura A.4: Convolução (1)

Fonte: Adaptado Hackathon (2024)

A Figura A.4 apresenta como funciona a etapa de convolução no treinamento do modelo. Aplica várias camadas de convolução para processar a imagem de entrada. Cada camada usa filtros que percorrem a imagem, detectando padrões como bordas, texturas e formas.

A seguir é apresentado algumas figuras capturadas de simulações de vídeo produzidos na exploração e manipulação dos dados do banco de dados.

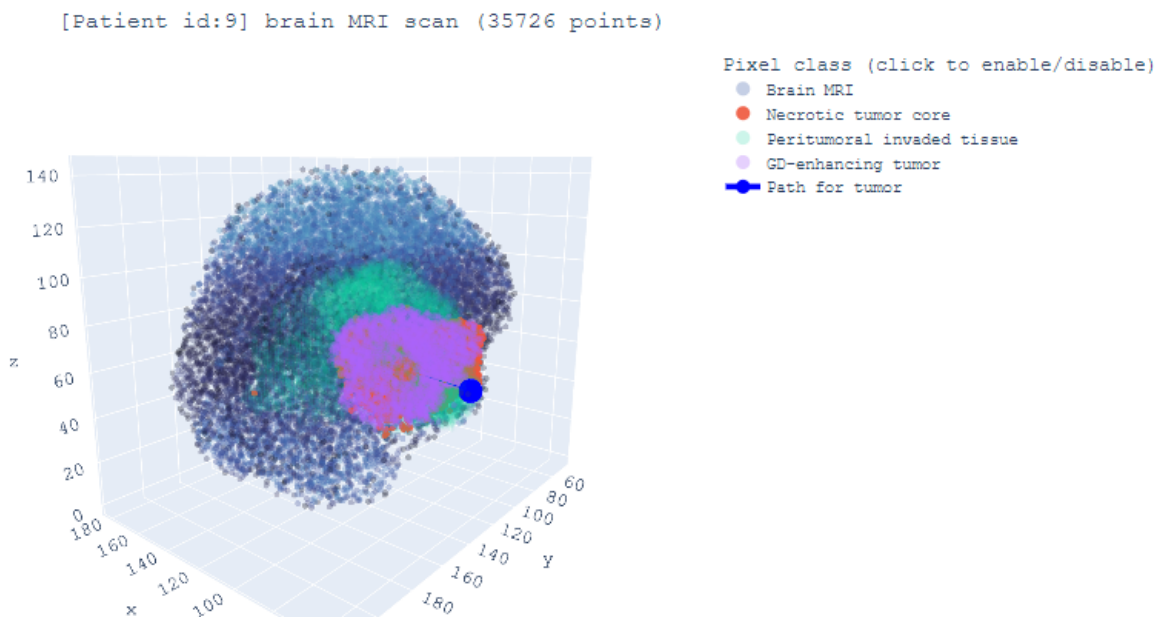


Figura A.5: Captura 1

Fonte: Própria Autora

Na Figura A.5 mostra a imagem de um cérebro constituído por classes de pixels onde podemos girar, diminuir, aumentar, tirar classe da imagem. Formada pelo cérebro, tumor necrótico, edema, tumor com realce e um eixo de provável caminho.

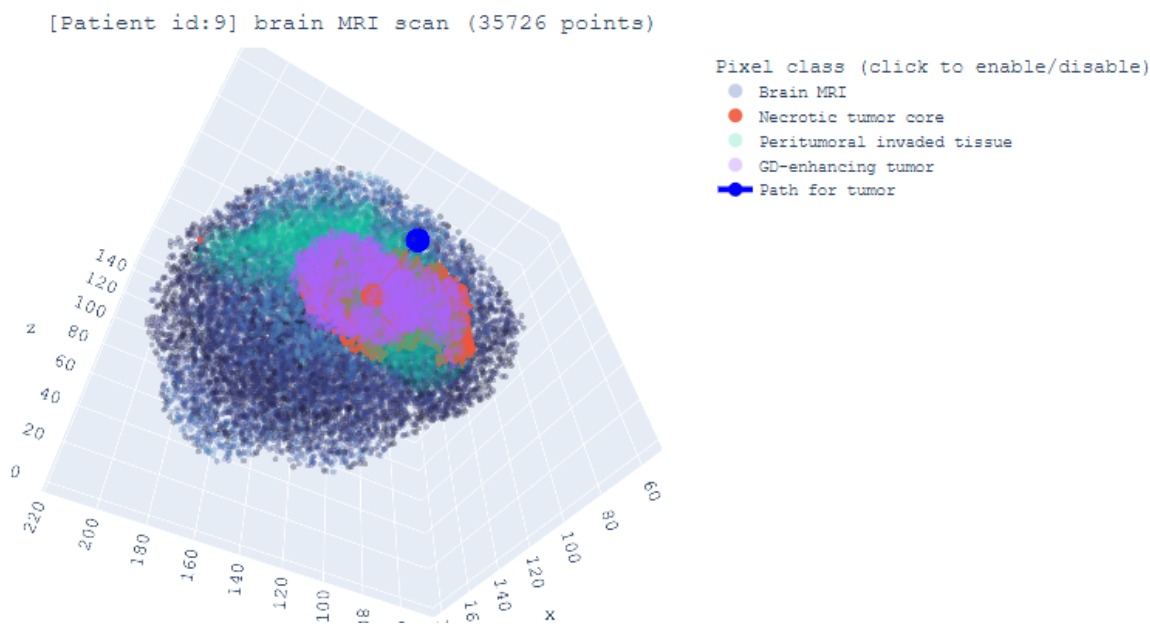


Figura A.6: Captura 2

Fonte: Própria Autora

Na Figura A.6 demonstra um pequeno giro do plano.

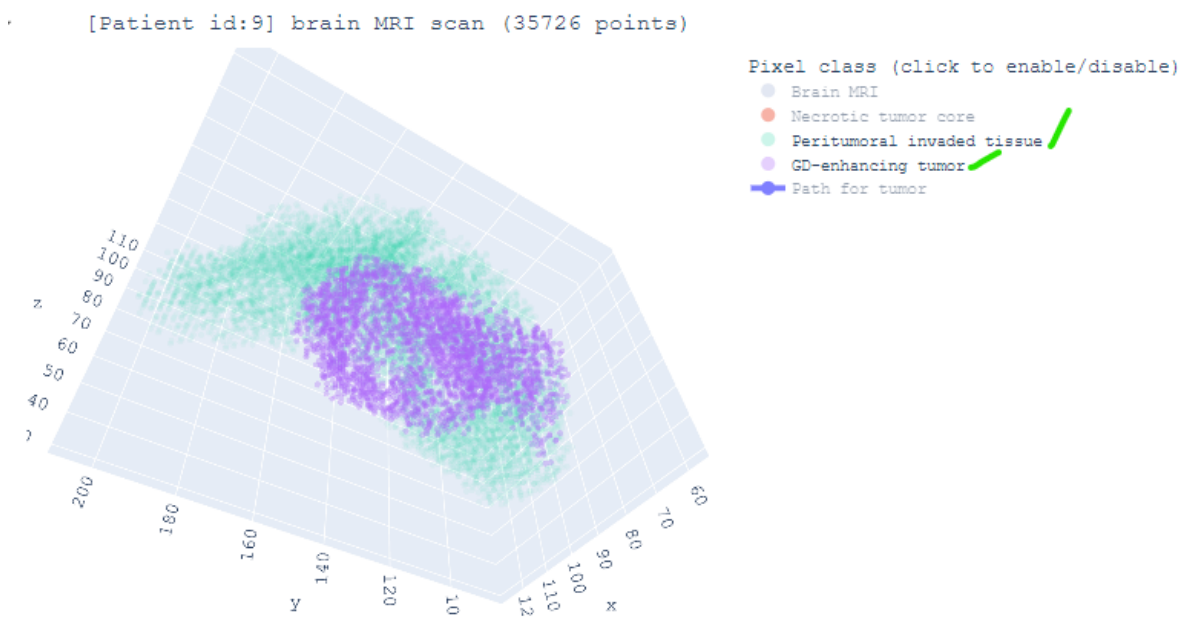


Figura A.7: Captura 3

Fonte: Própria Autora

A Figura A.7 apresenta apenas as classes de edema e tumor com realce foram mantidas na simulação.

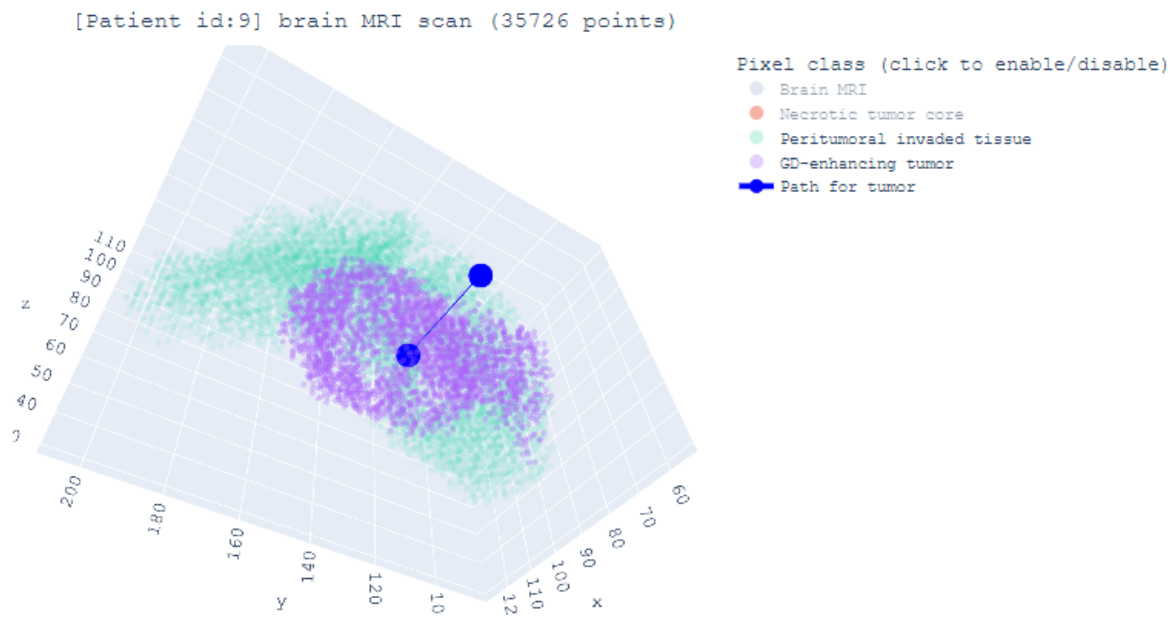


Figura A.8: Captura 4

Fonte: Própria Autora

Agora, na Figura A.8 é adicionado o eixo de provável direção do crescimento tumoral.

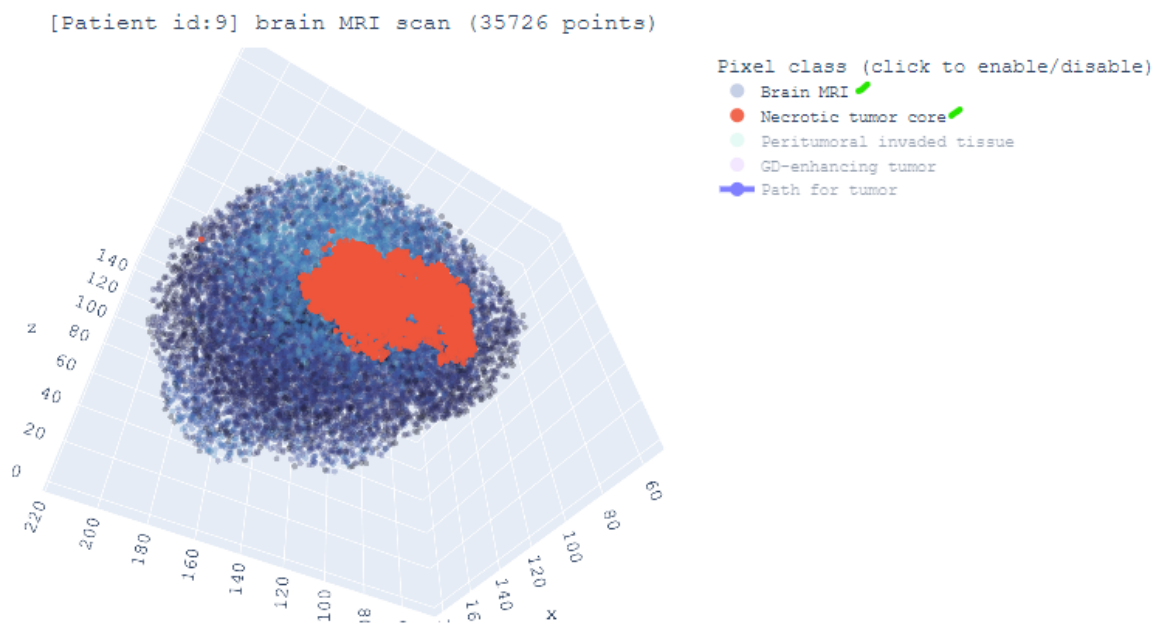


Figura A.9: Captura 5

Fonte: Própria 5

A Figura A.9 é apresentado os pixels do cérebro e do tumor necrótico.

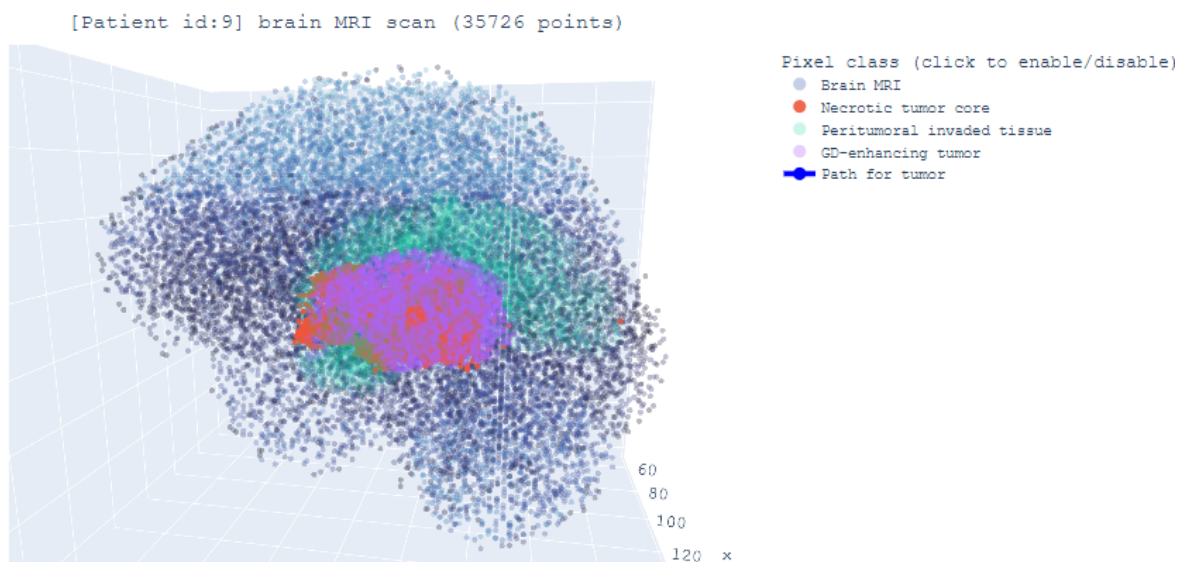


Figura A.10: Captura 6

Fonte: Própria Autora

A Figura A.10 apresenta a possibilidade de aumentar a imagem para melhor

observação de uma área.

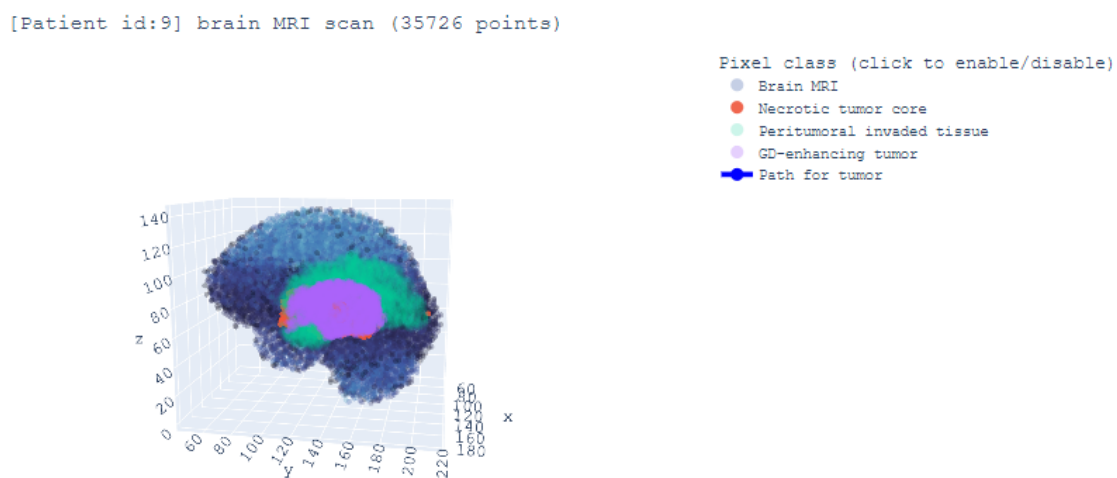


Figura A.11: Captura 7

Fonte: Própria Autora

Já na Figura A.11 mostra a possibilidade de diminuir a imagem.

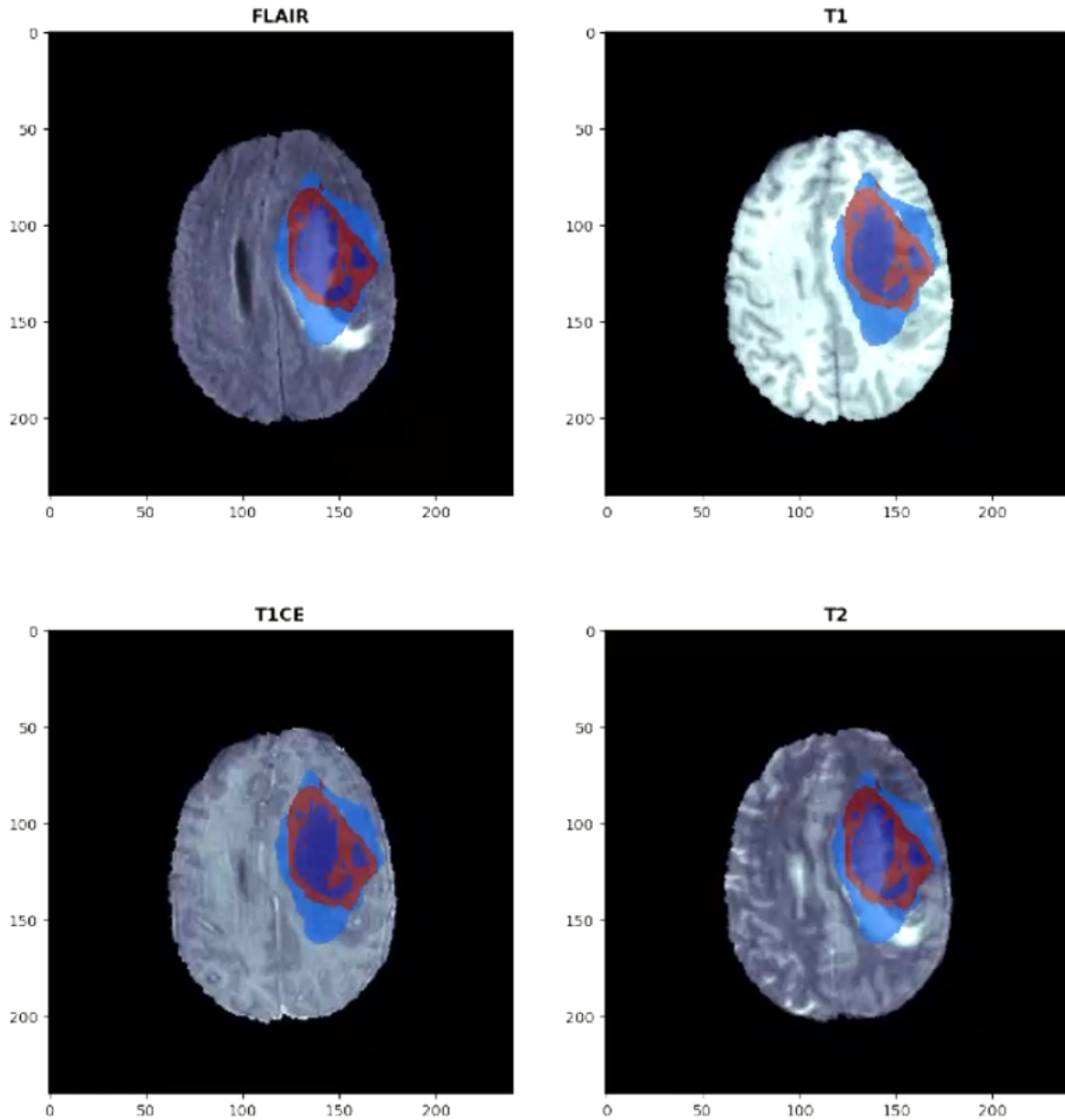
Imagem de Paciente ID: 6

Figura A.12: Captura 8

Fonte: Própria Autora

A Figura A.12 apresenta uma captura de outra simulação de vídeo onde é utilizado os tipos de imagens T1, T2, T1ce, Flair e ainda a máscara segmentada.

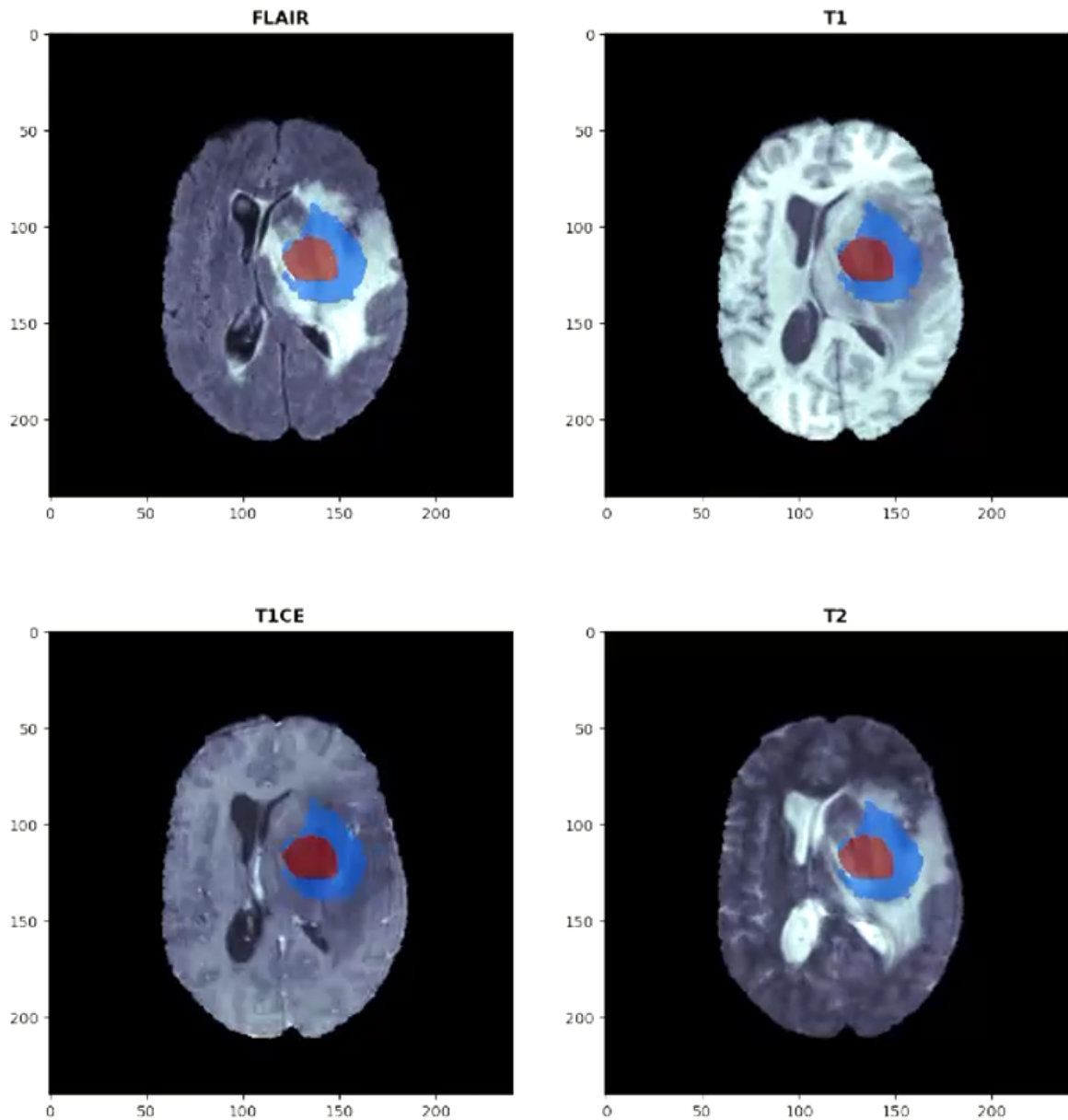
Imagem de Paciente ID: 6

Figura A.13: Captura 9

Fonte: Própria Autora

Na Figura A.13 apresenta uma nova imagem na mesma simulação com a captura da imagem em tempo diferente da Figura A.12 para ser observado a diferença da fatia apresentada nesse momento da captura.

APÊNDICE B – Código Python

Segmentação de Imagens de Ressonância Magnética de Tumor Cerebral Utilizando o Modelo 3D U-Net

```

### https://www.kaggle.com/code/rastislav/3d-mri-brain-tumor-segmentation-u-net

import os
import cv2
import glob
import PIL
import shutil
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from skimage import data
from skimage.util import montage
import skimage.transform as skTrans
from skimage.transform import rotate
from skimage.transform import resize
from PIL import Image, ImageOps

# imagem neural
import nilearn as nl
import nibabel as nib
import nilearn.plotting as nlplt
#!pip install git+https://github.com/miykael/gif_your_nifti # nifti to gif
import gif_your_nifti.core as gif2nif
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

# ml libs
import keras
import keras.backend as K
from keras.losses import CategoricalCrossentropy
from keras.losses import Loss
from keras.callbacks import CSVLogger
import tensorflow as tf
from keras.utils import plot_model
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.models import *
from keras.layers import *
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping, TensorBoard
from keras.layers.experimental import preprocessing

# Make numpy printouts easier to read. #Torne as impressões numpy mais fáceis de ler.
np.set_printoptions(precision=3, suppress=True)

```



```
!pip install nilearn
!pip install keras
```

```

TRAIN_DATASET_PATH2= f'{DATA_PATH}/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/BraTS20_Training_00{numcase}/BraTS20_Training_00{numcase}'
TRAIN_DATASET_PATH = f'{TRAIN_PATH}/'

test_image_flair=nib.load(TRAIN_DATASET_PATH2 + '_flair.nii').get_fdata()
test_image_t1=nib.load(TRAIN_DATASET_PATH2 + '_t1.nii').get_fdata()
test_image_t1ce=nib.load(TRAIN_DATASET_PATH2 + '_t1ce.nii').get_fdata()
test_image_t2=nib.load(TRAIN_DATASET_PATH2 + '_t2.nii').get_fdata()
test_mask=nib.load(TRAIN_DATASET_PATH2 + '_seg.nii').get_fdata()

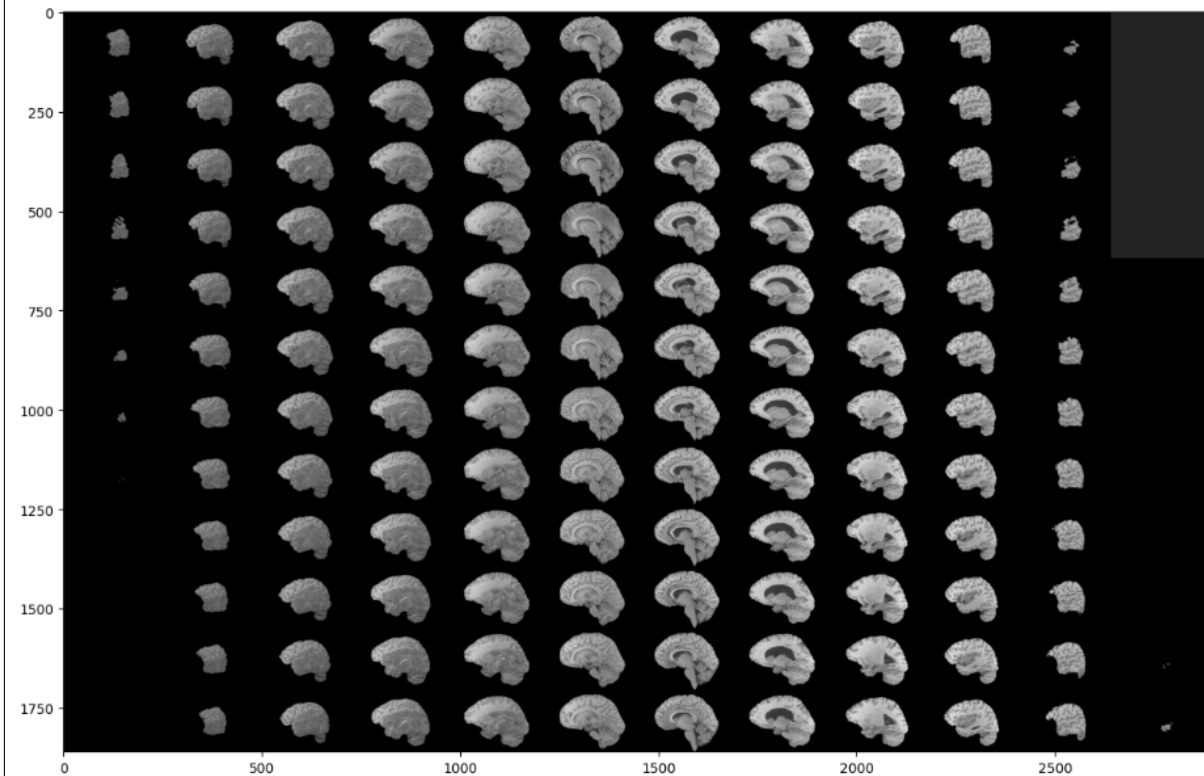
fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(1,5, figsize = (20, 10))
slice_w = 25 # <<----- fatia escolhida manualmente para plotar
ax1.imshow(test_image_flair[:, :, test_image_flair.shape[0]//2-slice_w], cmap = 'gray')
ax1.set_title('Image flair')
ax2.imshow(test_image_t1[:, :, test_image_t1.shape[0]//2-slice_w], cmap = 'gray')
ax2.set_title('Image t1')
ax3.imshow(test_image_t1ce[:, :, test_image_t1ce.shape[0]//2-slice_w], cmap = 'gray')
ax3.set_title('Image t1ce')
ax4.imshow(test_image_t2[:, :, test_image_t2.shape[0]//2-slice_w], cmap = 'gray')
ax4.set_title('Image t2')
ax5.imshow(test_mask[:, :, test_mask.shape[0]//2-slice_w], cmap = 'gray')
ax5.set_title('Mask')

Text(0.5, 1.0, 'Mask')

```

```
fig, ax1 = plt.subplots(1, 1, figsize = (15,15))
ax1.imshow(rotate(montage(test_image_t1[50:-50,:,:]), 90, resize=True), cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7f1c8a374520>
```



```
# Trabalhando com a biblioteca Nilearn

niimg = nl.image.load_img(TRAIN_DATASET_PATH2 + '_flair.nii')
nimask = nl.image.load_img(TRAIN_DATASET_PATH2 + '_seg.nii')

fig, axes = plt.subplots(nrows=5, figsize=(30, 40))
fig.suptitle('Imagem 00'+ str(numcase) + ' com diferentes efeitos - Coronal      Sagittal      Transversal', fontsize=30)

nlplt.plot_anat(niimg,
                title='BraTS20_Training_flair 00'+ str(numcase) +'.nii plot_anat',
                axes=axes[0])

nlplt.plot_epi(niimg,
               title='BraTS20_Training_flair 00'+ str(numcase) +'.nii plot_epi',
               axes=axes[1])

nlplt.plot_img(niimg,
               title='BraTS20_Training_flair 00'+ str(numcase) +'.nii plot_img',
               axes=axes[2])

nlplt.plot_roi(nimask,
               title='BraTS20_Training_flair 00'+ str(numcase) +'.nii with mask plot_roi',
               bg_img=niimg,
               axes=axes[3], cmap='Paired')

nlplt.plot_glass_brain(niimg,
                       title='BraTS20_Training_flair 00'+ str(numcase) +'.nii plot_glass_brain',
                       axes=axes[4])

plt.show()
```

```

def soft_dice_loss6(y_true, y_pred, epsilon=1e-6):
    """calculates the soft Dice loss."""
    class_num = 4
    dice_loss = 0
    for i in range(class_num):
        numerator = 2 * tf.reduce_sum(K.abs(y_true[:, :, i] * y_pred[:, :, i])) + epsilon
        denominator = tf.reduce_sum(K.square(y_true[:, :, i])) + tf.reduce_sum(K.square(y_pred[:, :, i])) + epsilon
        class_dice_loss = 1 - (numerator / denominator)
        dice_loss += class_dice_loss
    return dice_loss / class_num
    # soft_dice_loss6 = dice_loss / class_num

# dice loss as defined above for 4 classes
def dice_coef(y_true, y_pred, smooth=1.0):
    class_num = 4
    for i in range(class_num):
        y_true_f = K.flatten(y_true[:, :, i])
        y_pred_f = K.flatten(y_pred[:, :, i])
        intersection = K.sum(y_true_f * y_pred_f)
        loss = ((2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth))
        if i == 0:
            total_loss = loss
        else:
            total_loss = total_loss + loss
    total_loss = total_loss / class_num
    return total_loss

```

```

def dice_coef_necrotic(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:, :, 1] * y_pred[:, :, 1]))
    return (2. * intersection) / (K.sum(K.square(y_true[:, :, 1])) + K.sum(K.square(y_pred[:, :, 1])) + epsilon)

def dice_coef_edema(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:, :, 2] * y_pred[:, :, 2]))
    return (2. * intersection) / (K.sum(K.square(y_true[:, :, 2])) + K.sum(K.square(y_pred[:, :, 2])) + epsilon)

def dice_coef_enhancing(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:, :, 3] * y_pred[:, :, 3]))
    return (2. * intersection) / (K.sum(K.square(y_true[:, :, 3])) + K.sum(K.square(y_pred[:, :, 3])) + epsilon)

# Computing Precision
def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

# Computing Sensitivity
def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())

# Computing Specificity
def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())

```

```
IMG_SIZE=128

# ECC + SDL

class CustomCategoricalCrossentropy(tf.keras.losses.Loss):

    def __init__(self, label_smoothing=1e-6, class_weights=None, dice_weight=0.5, name='custom_categorical_crossentropy'):
        super(CustomCategoricalCrossentropy, self).__init__(name=name)
        self.label_smoothing = label_smoothing
        self.class_weights = class_weights
        self.dice_weight = dice_weight

    def call(self, y_true, y_pred):
        # Aplicar suavização de rótulo
        y_true = y_true * (1 - self.label_smoothing) + 0.5 * self.label_smoothing

        # Calcular a entropia cruzada categórica
        cross_entropy = tf.keras.losses.categorical_crossentropy(y_true, y_pred)

        # Calcular a perda suave Dice
        intersection = tf.reduce_sum(K.abs(y_true * y_pred))
        union = tf.reduce_sum(K.square(y_true)) + tf.reduce_sum(K.square(y_pred))
        dice_coefficient = (2.0 * intersection + 1e-6) / (union + 1e-6)
        dice_loss = 1.0 - dice_coefficient

        # Combinar as perdas
        loss = cross_entropy + self.dice_weight * dice_loss

        # Aplicar ponderação de classes
        if self.class_weights is not None:
            class_weights = tf.cast(self.class_weights, dtype=tf.float32)
            loss = loss * class_weights

        # Retornar a perda combinada
        return loss
```

```

def build_unet(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(inputs)
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv1)

    pool = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    pool1 = MaxPooling2D(pool_size=(2, 2))(conv)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv2)

    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv3)

    pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool4)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv5)
    drop5 = Dropout(dropout)(conv5)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(Upsampling2D(size = (2,2))(drop5))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(Upsampling2D(size = (2,2))(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(Upsampling2D(size = (2,2))(conv8))
    merge9 = concatenate([conv,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv9)

    up = Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(Upsampling2D(size = (2,2))(conv9))
    merge = concatenate([conv1,up], axis = 3)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    conv10 = Conv2D(4, (1,1), activation = 'softmax')(conv)

    return Model(inputs = inputs, outputs = conv10)

input_layer = Input((IMG_SIZE, IMG_SIZE, 2))

model = build_unet(input_layer, 'he_normal', 0.2)

```

```

# ECC + SDL
#model.compile(loss= CustomCategoricalCrossentropy(label_smoothing= 1e-6), optimizer=keras.optimizers.Adam(learning_rate=0.001), \
#               metrics = ['accuracy',tf.keras.metrics.MeanIOU(num_classes=4), soft_dice_loss6, tf.keras.metrics.MeanSquaredError(), \
#               tf.keras.metrics.MeanAbsoluteError(), precision, sensitivity, specificity, dice_coef_necrotic, dice_coef_edema, \
#               dice_coef_enhancing, tf.keras.metrics.CategoricalCrossentropy(label_smoothing=1e-6)])

# ECC
model.compile(loss="categorical_crossentropy", optimizer=keras.optimizers.Adam(learning_rate=0.001), \
              metrics = ['accuracy',tf.keras.metrics.MeanIOU(num_classes=4), soft_dice_loss6, tf.keras.metrics.MeanSquaredError(), \
              tf.keras.metrics.MeanAbsoluteError(), precision, sensitivity, specificity, dice_coef_necrotic, \
              dice_coef_edema ,dice_coef_enhancing, tf.keras.metrics.CategoricalCrossentropy(label_smoothing=1e-6)])

```

```

# preparar conjuntos de identificadores (IDs) para treinamento, validação e teste
train_and_val_directories = [f.path for f in os.scandir(TRAIN_DATASET_PATH) if f.is_dir()]

# arquivo BraTS20_Training_355 esta com problema, aqui vms remover
train_and_val_directories.remove(TRAIN_DATASET_PATH+'BraTS20_Training_355')

def pathListIntoIds(dirList):
    x = []
    for i in range(0,len(dirList)):
        x.append(dirList[i][dirList[i].rfind('/')+1:]) # extrai os IDs, que são as partes finais dos caminhos
    return x

train_and_test_ids = pathListIntoIds(train_and_val_directories);

train_test_ids, val_ids = train_test_split(train_and_test_ids,test_size=0.2) #train_and_test_ids é dividido em train_test_ids e val_ids.
train_ids, test_ids = train_test_split(train_test_ids,test_size=0.15) #train_ids é dividido em train_ids e test_ids.

```

```

#Carregar todos os dados na memória não é uma boa ideia,
#pois os dados são grandes demais para caberem. Portanto, criaremos dataGenerators - carregaremos os dados dinamicamente

class DataGenerator(keras.utils.Sequence):
    'Generates data for Keras'
    def __init__(self, list_IDS, dim=(IMG_SIZE,IMG_SIZE), batch_size = 1, n_channels = 2, shuffle=True):
        'Initialization'
        self.dim = dim
        self.batch_size = batch_size
        self.list_IDS = list_IDS
        self.n_channels = n_channels
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.list_IDS) / self.batch_size))

    def __getitem__(self, index):
        'Generate one batch of data'
        # Generate indexes of the batch
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        # Find list of IDs
        Batch_ids = [self.list_IDS[k] for k in indexes]

        # Generate data
        X, y = self.__data_generation(Batch_ids)

        return X, y

    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.list_IDS))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

```

```

def __data_generation(self, Batch_ids):
    'Generates data containing batch_size samples' # X : (n_samples, *dim, n_channels)
    # Initialization
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
    y = np.zeros((self.batch_size*VOLUME_SLICES, 240, 240))
    Y = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, 4))

    # Generate data
    for c, i in enumerate(Batch_ids):
        case_path = os.path.join(TRAIN_DATASET_PATH, i)

        data_path = os.path.join(case_path, f'{i}_flair.nii');
        flair = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_t1ce.nii');
        ce = nib.load(data_path).get_fdata()

        data_path = os.path.join(case_path, f'{i}_seg.nii');
        seg = nib.load(data_path).get_fdata()

        for j in range(VOLUME_SLICES):
            X[j + VOLUME_SLICES*c, :, :, 0] = cv2.resize(flair[:, :, j + VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
            X[j + VOLUME_SLICES*c, :, :, 1] = cv2.resize(ce[:, :, j + VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));

            y[j + VOLUME_SLICES*c] = seg[:, :, j + VOLUME_START_AT];

    # Generate masks
    y[y==4] = 3;
    mask = tf.one_hot(y, 4);
    Y = tf.image.resize(mask, (IMG_SIZE, IMG_SIZE));
    return X/np.max(X), Y # normalização é feita dividindo todas as imagens em X pelo valor máximo encontrado em X.

training_generator = DataGenerator(train_ids)
valid_generator = DataGenerator(val_ids)
test_generator = DataGenerator(test_ids)

```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Dados de exemplo
categorias = ["Treino", "Validação", "Teste"]
valores = [len(train_ids), len(val_ids), len(test_ids)]

# Calcula o valor total
total = sum(valores)

# Calcula as porcentagens de cada categoria em relação ao total
porcentagens = [valor / total * 100 for valor in valores]

# Cores para cada categoria
cores = ['royalblue', 'tomato', 'limegreen']

# Inicializa o gráfico
plt.figure(figsize=(8, 6))

# Cria a barra empilhada
plt.bar(categorias, porcentagens, color=cores)

# Personaliza as barras
plt.xlabel('Imagens')
plt.ylabel('Porcentagem em relação ao total (%)')
plt.title('Distribuição dos Dados')

# Adiciona as porcentagens acima ou dentro das barras
for i, v in enumerate(porcentagens):
    if v < 5: # Se a porcentagem for menor que 5%, coloca o texto dentro da barra
        plt.text(i, v + 2, f'{v:.2f}%', ha='center', va='bottom', fontsize=12)
    else:
        plt.text(i, v - 5, f'{v:.2f}%', ha='center', va='bottom', fontsize=12)

# Exibe o gráfico
plt.show()

```

```

log_folder = '/content/drive/MyDrive/IRM/Task01_BrainTumour/archive/model_x80_dcs65/' # Substitua pelo caminho real da sua pasta

# Verifica se o diretório existe, se não, cria
os.makedirs(log_folder, exist_ok=True)

log_file_path = os.path.join(log_folder, 'trainingcopy51.log')

csv_logger = CSVLogger(log_file_path, separator=',', append=False)

callbacks = [
#   keras.callbacks.EarlyStopping(monitor='loss', min_delta=0,
#   patience=2, verbose=1, mode='auto'),
#   keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2, #val_loss #'soft_dice_loss'
#   patience=2, min_lr=0.000001, verbose=1),
#   keras.callbacks.ModelCheckpoint(filepath = 'model_{epoch:02d}-{val_loss:.6f}.h5',
#   verbose=1, save_best_only=True, save_weights_only = True)
    csv_logger
]

```

```

history = model.fit(training_generator,
                    epochs= 40,
                    steps_per_epoch=len(train_ids),
                    callbacks= callbacks,
                    validation_data = valid_generator
                    )
model.save_weights("/content/drive/MyDrive/IRM/Task01_BrainTumour/archive/model_x80_dcs65/model_x80_dcs65weightscopy51.h5")
model.save("/content/drive/MyDrive/IRM/Task01_BrainTumour/archive/model_x80_dcs65/model_x80_dcs65copy51.h5")

```

```

def imageLoader(path):
    image = nib.load(path).get_fdata()
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, self.n_channels))
    for j in range(VOLUME_SLICES):
        X[j +VOLUME_SLICES*c, :, :, 0] = cv2.resize(image[:, :, j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));
        X[j +VOLUME_SLICES*c, :, :, 1] = cv2.resize(ce[:, :, j+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE));

        y[j +VOLUME_SLICES*c] = seg[:, :, j+VOLUME_START_AT];
    return np.array(image)

def loadDataFromDir(path, list_of_files, mriType, n_images):
    scans = []
    masks = []
    for i in list_of_files[:n_images]:
        fullPath = glob.glob( i + '/*'+ mriType +'')[0]
        currentScanVolume = imageLoader(fullPath)
        currentMaskVolume = imageLoader( glob.glob( i + '/*seg*')[0] )
        # for each slice in 3D volume, find also it's mask
        for j in range(0, currentScanVolume.shape[2]):
            scan_img = cv2.resize(currentScanVolume[:, :, j], dsiz=(IMG_SIZE, IMG_SIZE), interpolation=cv2.INTER_AREA).astype('uint8')
            mask_img = cv2.resize(currentMaskVolume[:, :, j], dsiz=(IMG_SIZE, IMG_SIZE), interpolation=cv2.INTER_AREA).astype('uint8')
            scans.append(scan_img[... , np.newaxis])
            masks.append(mask_img[... , np.newaxis])
    return np.array(scans, dtype='float32'), np.array(masks, dtype='float32')

```



```

def predictByPath(case_path,case):
    files = next(os.walk(case_path))[2]
    X = np.empty((VOLUME_SLICES, IMG_SIZE, IMG_SIZE, 2))
    y = np.empty((VOLUME_SLICES, IMG_SIZE, IMG_SIZE))#

    vol_path = os.path.join(case_path, f'BrATS20_Training_{case}_flair.nii');
    flair=nib.load(vol_path).get_fdata()

    vol_path = os.path.join(case_path, f'BrATS20_Training_{case}_t1ce.nii');
    ce=nib.load(vol_path).get_fdata()

    vol_path = os.path.join(case_path, f'BrATS20_Training_{case}_seg.nii');#
    seg=nib.load(vol_path).get_fdata() #

    for j in range(VOLUME_SLICES):
        X[j,:,:0] = cv2.resize(flair[:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))
        X[j,:,:1] = cv2.resize(ce[:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))
        y[j,:,:] = cv2.resize(seg[:,j+VOLUME_START_AT], (IMG_SIZE,IMG_SIZE))#

    return model.predict(X/np.max(X), verbose=1)

fluorescent_green_rgb = (0 / 255, 255 / 255, 0 / 255)
def showPredictsById(case, start_slice = 60):
    path = f"{TRAIN_PATH}/BrATS20_Training_{case}"
    gt = nib.load(os.path.join(path, f'BrATS20_Training_{case}_seg.nii')).get_fdata()
    origImage = nib.load(os.path.join(path, f'BrATS20_Training_{case}_flair.nii')).get_fdata()
    p = predictByPath(path,case)

    core = p[:, :, 1]
    edema= p[:, :, 2]
    enhancing = p[:, :, 3]

    plt.figure(figsize=(18, 50))
    f, axarr = plt.subplots(2,3, figsize = (15, 10))

    for i in range(2): # for each image, add brain background
        for j in range(3):
            axarr[i,j].imshow(cv2.resize(origImage[:, :, start_slice+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE)), cmap="gray", interpolation='none')

    axarr[0,0].imshow(cv2.resize(origImage[:, :, start_slice+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE)), cmap="gray")
    axarr[0,0].title.set_text(' Imagem Original Flair')
    curr_gt=cv2.resize(gt[:, :, start_slice+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE), interpolation = cv2.INTER_NEAREST)
    axarr[0,1].imshow(curr_gt, cmap="Reds", interpolation='none', alpha=0.3) # ,alpha=0.3,cmap='Reds'
    axarr[0,1].title.set_text('Ground truth')
    axarr[0,2].imshow(p[start_slice, :, 1:4], cmap="Reds", interpolation='none', alpha=0.3)
    axarr[0,2].title.set_text('all classes')
    axarr[1,0].imshow(core[start_slice, :], cmap="OrRd", interpolation='none', alpha=0.3)
    axarr[1,0].title.set_text(f'{SEGMENT_CLASSES[1]} previsto')
    axarr[1,1].imshow(edema[start_slice, :], cmap='magma', interpolation='none', alpha=0.3) #OrRd OrangeRed
    axarr[1,1].title.set_text(f'{SEGMENT_CLASSES[2]} previsto')
    axarr[1,2].imshow(enhancing[start_slice, :], cmap="OrRd", interpolation='none', alpha=0.3)
    axarr[1,2].title.set_text(f'{SEGMENT_CLASSES[3]} previsto')
    plt.show()

```

```

showPredictsById("00"+str(numcase),46)
showPredictsById(case=test_ids[10][-3:])
showPredictsById(case=test_ids[12][-3:])
showPredictsById(case=test_ids[22][-3:])
showPredictsById(case=test_ids[14][-3:])
showPredictsById(case=test_ids[23][-3:])
showPredictsById(case=test_ids[44][-3:])
showPredictsById(case=test_ids[35][-3:])

```