



Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE

Luis Alberto Souto Maior Neto

**Redes Neurais Convolucionais para Segmentação de
Tumores Cerebrais em Imagens de Ressonância
Magnética**

Campina Grande, Paraíba

Abril de 2017

Luis Alberto Souto Maior Neto

Redes Neurais Convolucionais para Segmentação de Tumores Cerebrais em Imagens de Ressonância Magnética

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Prof. Dra. Luciana Ribeiro Veloso

Campina Grande, Paraíba

Abril de 2017

Luis Alberto Souto Maior Neto

Redes Neurais Convolucionais para Segmentação de Tumores Cerebrais em Imagens de Ressonância Magnética/ Luis Alberto Souto Maior Neto. – Campina Grande, Paraíba, Abril de 2017

83 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dra. Luciana Ribeiro Veloso

Trabalho de Conclusão de Curso

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE , Abril de 2017.

1. Redes Neurais. 2. Imagem Médica. 2. Processamento de Imagem. I. Universidade Federal de Campina Grande. II. Departamento de Engenharia Elétrica. III. Redes Neurais Convolucionais para Segmentação de Tumores Cerebrais em Imagens de Ressonância Magnética

Luis Alberto Souto Maior Neto

Redes Neurais Convolucionais para Segmentação de Tumores Cerebrais em Imagens de Ressonância Magnética

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado em ____ / ____ / _____

Prof. Dra. Luciana Ribeiro Veloso
Universidade Federal de Campina Grande

Prof. Dr. Edmar Candeia Gurjão
Universidade Federal de Campina Grande

Campina Grande, Paraíba
Abril de 2017

Agradecimentos

À minha família, meu pai Roberto, minha mãe Maria Esther (*in memoriam*), e minha irmã Marília, por sempre terem me apoiado e dado educação em toda a minha vida, dádivas imensuráveis.

À minha namorada Ingrid, por ter sempre ficado ao meu lado e me dado forças para ficar em pé em todos os momentos de fraqueza.

Aos meus amigos Luiz Neto, Matheus Augusto, Arthur, Fábio, Germano e Danilo pela ajuda em abrir minha mente para a vida, o universo e tudo mais.

Aos professores que tive durante minha vida, por me darem a oportunidade de aprender.

Aos funcionários Tchaikowsky e Adail que, sem eles, teria uma família a menos.

À música, por ser capaz de me compreender.

E à oportunidade de ser e de existir.

*“ We live together, we act on, and react to, one another; but always and in all circumstances we are by ourselves. The martyrs go hand in hand into the arena; they are crucified alone. Embraced, the lovers desperately try to fuse their insulated ecstasies into a single self-transcendence; in vain. By its very nature every embodied spirit is doomed to suffer and enjoy in solitude. Sensations, feelings, insights, fancies - all these are private and, except through symbols and at second hand, incommunicable. We can pool information about experiences, but never the experiences themselves.
(Aldous Huxley, "The Doors of Perception")*

Resumo

Câncer cerebral é um dos tipos de câncer com maior taxa de mortalidade na atualidade, e uma das suas formas mais comuns são os gliomas. Gliomas são facilmente observáveis em imagens de ressonância magnética (MRI), porém a sua medição e avaliação é uma atividade complexa e demorada graças à grande quantidade e variedade de dados de imagens médicas que um profissional deve lidar. Este fato torna a análise a partir de imagens médicas uma tarefa árdua e susceptível a grande taxa de erro inter- e intra-operador. Para resolver estes problemas, técnicas de diagnose assistida por computador (CAD) foram desenvolvidas. Neste trabalho, todos estes conceitos são abordados de forma minuciosa e metodologias e arquiteturas baseadas em redes neurais convolucionais profundas são propostas para resolver o problema de segmentação de tumores cerebrais em imagens de MRI. Os resultados obtidos indicam uma taxa de acerto *Dice score* de 88% para segmentação destes tipos de anomalias no banco de dados BRATS 2016.

Palavras-chave: processamento de imagens. engenharia biomédica. aprendizado profundo. tumores cerebrais. redes neurais convolucionais.

Abstract

Cerebral cancer is one of the cancer types with bigger mortality rates in the present days. The most common form of brain cancer is the glioma, which is easily visualized in magnetic resonance images. However, the measurement and evaluation of gliomas is complex and time-consuming due to the great variety and number of medical image data the medical professional has to deal with day-to-day. This increases intra- and inter-operator error variability, making it necessary to develop computer assisted diagnosis (CAD) tools to overcome such problems. In this work, these concepts are approached thoroughly in order to propose and present a method for segmentation of brain tumours on MRI images. The proposed method achieved a Dice score of 88% for anomaly segmentation, evaluated on the BRATS 2016 database.

Keywords: image processing. biomedical engineering. deep learning. brain tumours. convolutional neural networks.

Lista de ilustrações

Figura 1 – Representação artística e fotografia real de gliomas.	19
Figura 2 – Edema em ressonância magnética.	20
Figura 3 – Exame de MRI.	21
Figura 4 – Funcionamento de uma máquina de MRI.	22
Figura 5 – Geração de campo gradiente.	22
Figura 6 – Ilustração planos de MRI.	23
Figura 7 – Exemplos reais planos de MRI.	23
Figura 8 – Diferentes modalidades de MRI.	24
Figura 9 – Como imagens são representadas em um computador.	25
Figura 10 – Modelo matemático.	26
Figura 11 – Esquema de treinamento supervisionado.	28
Figura 12 – Classificação ImageNet.	30
Figura 13 – Segmentação semântica.	31
Figura 14 – Segmentação de glioma e edema cerebral em MRI.	31
Figura 15 – Neurônio biológico e neurônio artificial.	32
Figura 16 – Esquema de uma rede MLP.	33
Figura 17 – Exemplos de função de ativação.	34
Figura 18 – Ilustração de rede neural " rasa " e " profunda ".	37
Figura 19 – Capacidade de abstração das redes neurais profundas.	38
Figura 20 – Ilustração de uma CNN típica.	39
Figura 21 – Esquema de uma convolução bidimensional.	40
Figura 22 – Exemplo de diferentes núcleos na mesma imagem.	41
Figura 23 – Funcionamento da operação de <i>max-pooling</i>	42
Figura 24 – Funcionamento do dropout.	43
Figura 25 – Esquema de processamento e treinamento implementado.	45
Figura 26 – Arquitetura A: Segmentação Multiclases	51
Figura 27 – Arquitetura B: Segmentação de Anomalias	52
Figura 28 – Exemplo de convolução em imagem RGB.	61
Figura 29 – Exemplo de amostra do BRATS 2016 de um único paciente.	62
Figura 30 – Exemplo de amostra redimensionada para 60×60	63
Figura 31 – Exemplo de amostra nula.	64
Figura 32 – Exemplo 1 de resultado de teste da arquitetura A	65
Figura 33 – Exemplo 2 de resultado de teste da arquitetura A	66
Figura 34 – Exemplo 3 de resultado de teste da arquitetura A	67
Figura 35 – Exemplo 4 de resultado de teste da arquitetura A	68
Figura 36 – Exemplo 5 de resultado de teste da arquitetura A	69

Figura 37 – Exemplo 1 de resultado de teste da arquitetura B	70
Figura 38 – Exemplo 2 de resultado de teste da arquitetura B	71
Figura 39 – Exemplo 3 de resultado de teste da arquitetura B	72
Figura 40 – Exemplo 4 de resultado de teste da arquitetura B	73
Figura 41 – Exemplo 5 de resultado de teste da arquitetura B	74

Lista de tabelas

Tabela 1 – Tabela de codificação <i>one-hot</i>	29
Tabela 2 – Tamanhos dos tensores de dados assim que importados ao ambiente Python	47
Tabela 3 – Tamanhos dos tensores após empilhamento.	47
Tabela 4 – Tamanhos dos tensores após redimensionamento.	48
Tabela 5 – Tamanhos dos tensores após redimensionamento.	49
Tabela 6 – Característica do conjunto de dados após pré-processamento.	50
Tabela 7 – Característica do conjunto de treino utilizado na rede neural.	53
Tabela 8 – Característica do conjunto de testes utilizado na rede neural.	53
Tabela 9 – <i>Dice scores</i> obtidos da arquitetura A	54
Tabela 10 – <i>Dice scores</i> obtidos da arquitetura B	54

Lista de abreviaturas e siglas

MRI	Imageamento por Ressonância Magnética (<i>magnetic resonance imaging</i>)
CT	Tomografia Computadorizada (<i>computed tomography</i>)
PET	Tomografia por Emissão de Prótons (<i>positron emission tomography</i>)
CAD	Diagnose Assistida por Computador (<i>computer assisted diagnosis</i>)
BRATS	<i>Multimodal Brain Tumour Image Segmentation Challenge</i>
GPU	Placa de Processamento Gráfico (<i>graphical processing unit</i>)
ML	Aprendizado de Máquina (<i>machine learning</i>)
DL	Aprendizado Profundo (<i>deep learning</i>)
CNN	Rede Neural Convolutacional (<i>convolutional neural network</i>)
MSE	Erro médio quadrático (<i>mean squared error</i>)

Lista de símbolos

\mathcal{L}	Função de erro ou perda (<i>loss function</i>)
Θ	Parâmetros internos do modelo
Γ	Hiperarâmetros internos não-treináveis do modelo
W	Matriz de pesos de uma rede neural
X	Matriz de entradas de uma rede neural ou modelo
Y	Matriz de saídas resultante de uma rede neural ou modelo
T	Matriz de saída alvos de uma rede neural ou modelo
x_i	Entrada i de uma rede neural ou de um modelo
y_i	Saída i resultante de uma rede neural ou modelo
t_i	Saída alvo i resultante de uma rede neural ou modelo
θ_i	Parâmetro interno i de um modelo
w_i	Peso i de uma rede neural
∇	Operador gradiente
∂	Operador derivada parcial
α	Taxa de aprendizado (<i>learning rate</i>)
\cap	Operador intersecção

Sumário

1	INTRODUÇÃO	16
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	Tumores cerebrais	18
2.1.1	Estatísticas	18
2.1.2	Gliomas	19
2.1.3	Edemas cerebrais	19
2.2	Imageamento médico	20
2.2.1	Ressonância magnética	20
2.2.2	Princípio de funcionamento	21
2.2.3	Características e Modalidades	23
2.2.4	Processamento digital de imagens biomédicas	24
2.3	Aprendizado de máquina	25
2.3.1	Modelagem	25
2.3.2	Treinamento supervisionado	27
2.3.2.1	Classificação	28
2.3.2.2	Segmentação	29
2.3.3	Redes neurais artificiais	30
2.3.3.1	Perceptron simples	32
2.3.3.2	Perceptron de múltiplas camadas	33
2.3.3.3	Funções de ativação	34
2.3.3.4	Treinamento via Retropropagação do Erro e Gradiente Descendente	34
2.3.3.5	Funções de erro	35
2.3.3.5.1	Mean Squared Error (MSE)	36
2.3.3.5.2	Crossentropy	36
2.3.3.5.3	<i>Dice score</i>	36
2.3.4	Aprendizagem profunda	37
2.3.4.1	Redes Neurais Convolucionais	39
2.3.4.1.1	Camadas convolucionais	40
2.3.4.1.2	Camadas <i>pooling</i>	41
2.3.4.2	Regularização	42
2.3.4.2.1	Dropout	42
2.3.4.2.2	Batch normalization	43
3	METODOLOGIA	45
3.1	Banco de dados BRATS 2016	46

3.1.1	Estrutura dos dados	47
3.2	Pré-processamento	47
3.2.1	Organização	47
3.2.2	Redimensionamento	48
3.2.3	Eliminação de elementos nulos	48
3.2.4	Normalização	49
3.2.5	Conjunto de treinamento e de teste	50
3.3	Arquiteturas propostas das redes neurais	51
3.3.1	Arquitetura A: Segmentação Multiclasses	51
3.3.2	Arquitetura B: Segmentação de Anomalias	52
3.4	Treinamento da rede neural	53
3.4.1	Separação do conjunto de dados	53
3.4.2	Testes realizados	53
4	RESULTADOS	54
5	CONCLUSÃO	55
	REFERÊNCIAS	57
	ANEXOS	60
	ANEXO A – DEMONSTRAÇÃO DA OPERAÇÃO DE CONVUL-	
	ÇÃO.	61
	ANEXO B – EXEMPLOS DO BRATS 2016	62
	ANEXO C – EXEMPLOS PÓS-TREINAMENTO (CONJUNTO DE	
	TESTE): ARQUITETURA A	65
	ANEXO D – EXEMPLOS PÓS-TREINAMENTO: ARQUITETURA	
	B	70
	ANEXO E – CÓDIGOS	75
E.1	Pipeline de Pré-processamento	75
E.2	Arquitetura da Rede Neural	79
E.3	Treinamento da Rede Neural	81
E.4	Visualização dos Resultados	82

1 Introdução

Imageamento biomédico envolve o estudo da aquisição, processamento e análise à posteriori de imagens bi ou tridimensionais que representam espacialmente partes internas do corpo humano ou animal, obtidas a partir de dispositivos de imageamento médico, como máquinas de ressonância magnética nuclear (MRI), tomografia computadorizada (CT), tomografia por emissão de prótons (PET), ultrassonografia, dentre outros. Esta área provê várias ferramentas extremamente úteis para análise clínica, estudo do funcionamento dos órgãos e sistemas biológicos, bem como para diagnóstico e acompanhamento do tratamento de doenças. Muitas vezes, faz-se uso de distintos tipos de imageamento simultaneamente (JAMES; DASARATHY, 2014) para se obter a maior quantidade de informações de um paciente. A área de processamento de imagens médicas utiliza ferramentas matemáticas, como análise espacial, temporal e espectral, métodos estocásticos e aprendizado de máquina para desenvolver softwares que auxiliem a resolução de problemas da medicina como os citados anteriormente.

Tumores cerebrais, também conhecidos como neoplasias intracranianas, são fenômenos patológicos que podem ser caracterizados pelo crescimento volumoso de células anormais no cérebro. De acordo com a American Cancer Society (ACS, 2016), a chance de sobrevivência em caso de tumor na região nervosa central é extremamente baixa, variando de 27% a 33% dependendo do tipo e de quão avançada está a doença. A forma mais comum de câncer cerebral é o câncer no tecido glial, chamado de glioma (WHO, 2017). Gliomas são geralmente detectados em imagens médicas a partir da descoberta de uma região afetada pelo câncer que é preenchida por líquidos, chamada de edema cerebral. Imagens de ressonância magnética (MRI) são consideradas o tipo de imageamento mais apropriado para a análise de anormalidades no cérebro, pois MRI apresenta alto contraste entre tecidos moles, alta resolução espacial, com técnicas modernas conseguindo contraste espacial na escala de milímetros. No entanto, graças ao formato imprevisível e aparência difusa de gliomas e edemas, a detecção de câncer cerebral se torna um desafio. Para solucionar este problema, sistemas de detecção, classificação e segmentação automática de imagens médicas, conhecidos como sistemas de diagnóstico assistido por computador, do inglês *computer-assisted diagnosis* (CAD), têm sido desenvolvidos (APPENZELLER et al., 2008; VERNOOIJ et al., 2007). Em 2016, a World Health Organization publicou um novo banco de classes de diferentes tipos de tumores cerebrais baseado em metodologias de classificação modernas (LOUIS et al., 2016). Também em 2016, foi demonstrado fortes tendências positivas no uso de técnicas avançadas de aprendizado de máquina, como redes neurais convolucionais para solucionar problemas de detecção e segmentação de patologias cerebrais focalizadas (HVAEI et al., 2016).

Aprendizado de máquina, área que estuda o desenvolvimento de algoritmos computacionais iterativos e adaptativos baseados na análise de grandes conjuntos de dados, tem gerado resultados surpreendentes em diversas aplicações como reconhecimento de padrões, processamento de linguagem natural, classificação de dados e segmentação de imagens. Com o advento de placas de processamento gráfico (GPUs) potentes, a possibilidade de treinar redes neurais profundas (área conhecida como aprendizado profundo ou "*deep learning*") gerou grandes impactos positivos na performance dos algoritmos, particularmente com o uso de redes neurais convolucionais (GU et al., 2015) profundas, nas mais diversas áreas: da detecção de patologias em imagens médicas ao controle de carros autônomos; da tradução de texto automática entre línguas diferentes à descoberta de novas drogas medicinais e seu estudo toxicológico. O estado da arte neste campo de pesquisa é anualmente ultrapassado pode ser observado nos grandes desafios internacionais, como o Multimodal Brain Tumor Image Segmentation Challenge (BRATS) (MENZE et al., 2014) e o ImageNet Large Scale Visual Recognition Challenge (RUSSAKOVSKY et al., 2015). Redes neurais convolucionais têm sido empregadas na área de imageamento médico para solucionar problemas de segmentação, classificação, e têm obtido excelentes resultados nestas tarefas com o crescimento de bancos de imagens médicas em larga escala (DESPOTOVIĆ; GOOSSENS; PHILIPS, 2015; SHIN et al., 2016; TAJBAKHSH et al., 2016).

Este projeto de conclusão de curso teve como objetivo realizar um estudo em técnicas de aprendizado profundo, especificamente redes neurais convolucionais, com o intuito de informar estudantes interessados na área de imageamento médico e de aprendizado de máquina e lhes guiar para darem um passo inicial no estudo destas áreas. Adicionalmente, este trabalho propõe uma metodologia de aprendizado de máquina para solucionar e validar algoritmos utilizando redes neurais convolucionais para segmentação de tumores cerebrais, com base em sua notória performance nesta tarefa (HAVAELI et al., 2016), a partir da análise de imagens biomédicas de MRI obtidas no banco de dados (MENZE et al., 2014).

Estrutura do documento

Este documento está organizado em 5 capítulos. No [Capítulo 2](#), é abordada a fundamentação teórica de temas importantes para o trabalho, como a definição de tumores cerebrais, imageamento médico e aprendizado de máquina. Já no [Capítulo 3](#), são apresentadas as técnicas e procedimentos utilizados na metodologia e desenvolvimento do software desenvolvido. No [Capítulo 4](#) são mostrados os resultados obtidos. Finalmente, no [Capítulo 5](#), são dadas as considerações finais sobre os resultados obtidos, além de serem discutidas possíveis melhorias aos métodos utilizados. No final deste documento estão as referências bibliográficas e apêndices com imagens que possam exemplificar os resultados.

2 Fundamentação teórica

Nesta parte do trabalho, é exposta toda a fundamentação teórica necessária para o bom entendimento do problema estudado, bem como a solução aqui proposta.

2.1 Tumores cerebrais

Tumores acontecem quando há reprodução de células de forma não-controlada, desenfreada ou anormal em alguma parte do corpo. Quando este fenômeno ocorre na região cerebral, é denominado de tumor cerebral ou neoplasia intracraniana. Tumores cerebrais são classificados em uma escala de 1 a 4, de acordo com características, como taxa de crescimento e probabilidade de voltar a crescer após tratamentos. No geral, tumores cerebrais são considerados benignos quando estão entre 1 e 2 nesta escala, e malignos quando estão entre 3 e 4 (NHS, 2017a).

Tumores benignos, por terem baixa taxa de crescimento e menores chances de crescerem após tratamentos, possuem altas chances de serem tratados com sucesso. No entanto, estes tumores ainda podem ser um alto risco para a vida do indivíduo, dependendo do tamanho do tumor e da localização no cérebro na qual ele se encontra. Os tumores considerados malignos, por sua vez, além de crescerem rapidamente, podem ou surgir diretamente no cérebro (tumores primários), ou aparecer no cérebro após ser espalhado ou deslocado de outra região (tumores secundários). Esta incerteza da fonte de onde o tumor origina torna os tumores malignos inerentemente mais perigosos à saúde, fazendo com que necessite de tratamento mais recorrente e agressivo.

2.1.1 Estatísticas

Tumores cerebrais podem advir de diversas fontes, desde mutações genéticas hereditárias até mutações aleatórias ou advindas de radiação ionizante externa, porém a razão mais geral pela qual pessoas desenvolvem tumores cerebrais primários é atualmente desconhecida. Adicionalmente, tumores cerebrais podem afetar indivíduos de qualquer idade, com maiores tendências para acontecer em pessoas de idade mais elevada (WHO, 2017).

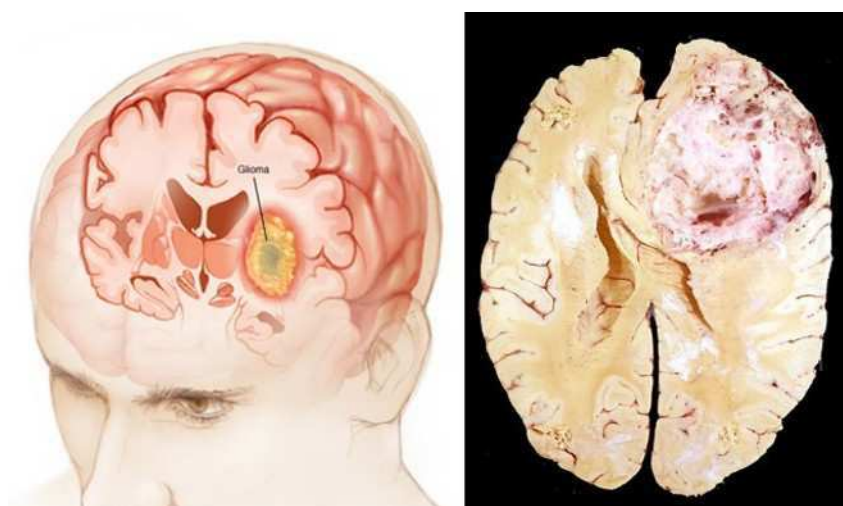
No Brasil, estima-se, em média, uma taxa de aparecimento de neoplasias intracranianas malignas em 9 a cada 200.000 pessoas. No ano de 2012, foram estimados 9270 casos de câncer cerebral, 4820 destes casos ocorrendo em homens e 4450 em mulheres. Mundialmente, neoplasias malignas no sistema nervoso central representa aproximadamente 2% de todos os casos de neoplasias malignas. Além disso, tem sido observado um

aumento na incidência desta patologia nas últimas décadas (ONCOMED, 2017). As taxas de mortalidade de pacientes que desenvolveram câncer no cérebro são extremamente altas: apenas 15 em cada 100 pacientes sobrevivem mais do que 5 anos após o diagnóstico inicial da doença.

2.1.2 Gliomas

Tumores malignos se desenvolvem, em sua maior parte, a partir do tecido glial, que é composto por células chamadas glias, responsáveis pelo suporte mecânico das células nervosas do cérebro. Assim, os tumores malignos mais conhecidos são chamados de gliomas (NHS, 2017b). Na Figura 1, à esquerda, é ilustrado um glioma infiltrado no cérebro de um indivíduo. À direita, é mostrado uma imagem de corte transversal de um cérebro afetado por glioma.

Figura 1 – Representação artística e fotografia real de gliomas.



Fonte: (ZORZI, 2017)

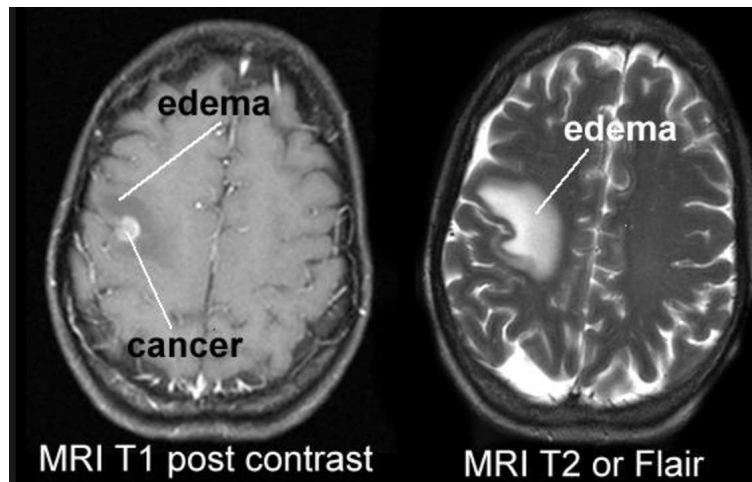
2.1.3 Edemas cerebrais

Geralmente associado aos gliomas estão os edemas cerebrais, que são inchaços causados por descontrolado acúmulo de líquidos no cérebro. Quando ocorrem associados a gliomas, os edemas contribuem significativamente para a mortalidade, pois os líquidos promovem invasão de células gliais afetadas pelo câncer. Adicionalmente, os inchaços causados pelos edemas comprometem mecanicamente o cérebro ao apertar e deslocar as células nervosas na vizinhança, chegando à possibilidade de colapso das mesmas (LIN, 2013).

Edemas são facilmente visualizados em exames de imageamento médico como em certas modalidades de ressonância magnética, e sua descoberta é comumente utilizada

como prognóstico para gliomas. Um edema em uma imagem de ressonância magnética é ilustrado na [Figura 2](#).

Figura 2 – Edema em ressonância magnética.



Fonte: ([ABOUTCANCER, 2017](#))

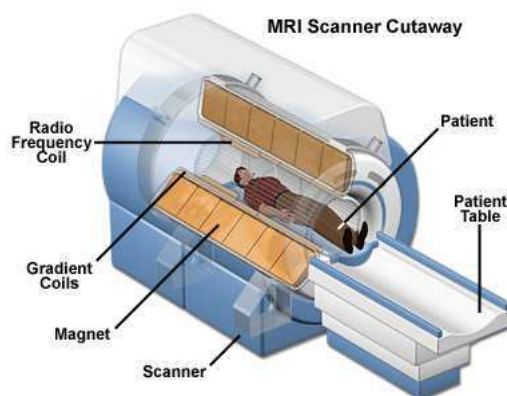
2.2 Imageamento médico

Imageamento médico consiste na aplicação de técnicas de aquisição de imagens de partes internas do corpo humano ou animal, baseadas em tecnologias que foram desenvolvidas para facilitar a diagnose, monitoração e acompanhamento de tratamento de diversas doenças. Existe uma diversidade de técnicas de imageamento médico, como tomografia computadorizada, ressonância magnética, raio-x e ultrassom ([DEVICES; HEALTH, 2016](#)).

2.2.1 Ressonância magnética

Imageamento por ressonância magnética (MRI) é uma técnica de imageamento médico que faz uso de fortes campos magnéticos e emissões eletromagnéticas não-ionizantes para se detectar a presença de átomos de hidrogênio no corpo humano. Com informação da posição e densidade de átomos dos hidrogênio, uma imagem pode ser reconstruída de forma a se ter, com precisão, informação visual das partes internas do corpo, já que esta é composta em maior parte por elementos que contém átomos de hidrogênio. Como os diferentes tecidos moles do corpo humano são composto por materiais diferentes, com distintas proporções de átomos de hidrogênio, é possível reconfigurar a máquina de MRI para visualizar diferenças entre os tecidos, acentuando ou inibindo tecidos específicos.

Figura 3 – Exame de MRI.



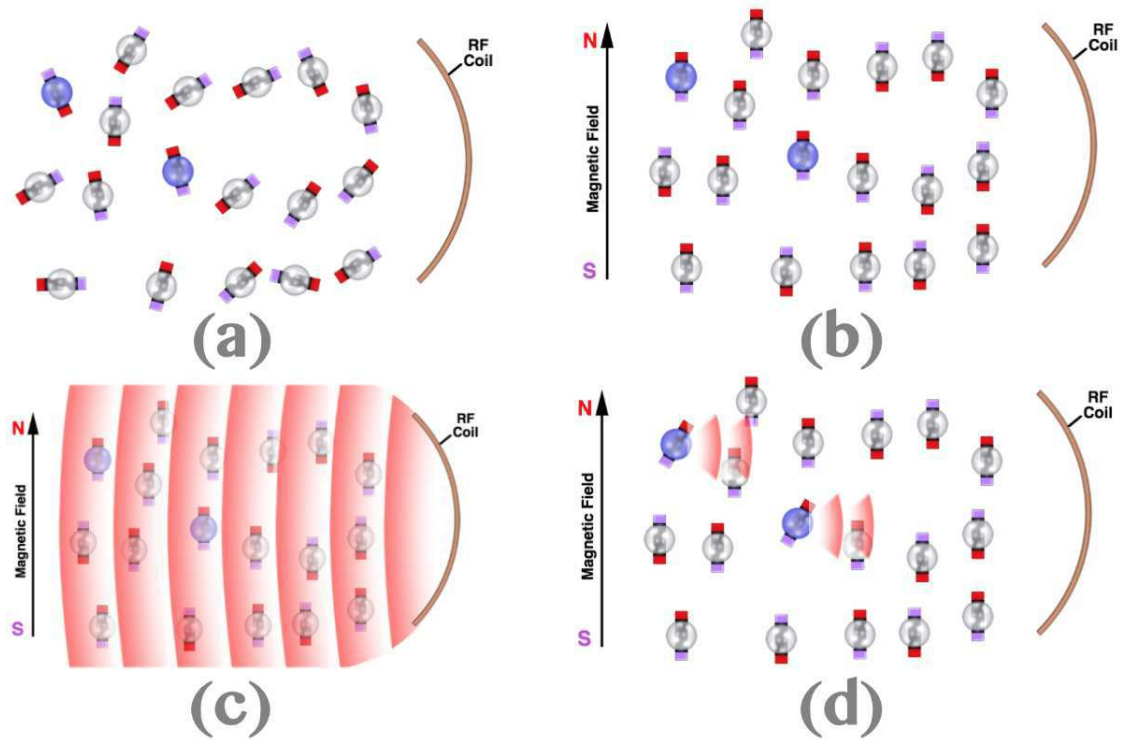
Fonte: (LABORATORY, 2016)

2.2.2 Princípio de funcionamento

Em um exame de MRI (Figura 3), um paciente se coloca no centro de uma máquina que contém uma série de bobinas, capaz de gerar campos magnéticos da ordem de 3 tesla. Estes campos uniformes são ativados, fazendo com que os átomos de hidrogênio, que podem ser modelados por dipólos magnéticos, alinhem-se ao campo. Em seguida, pulsos eletromagnéticos são emitidos sobre o corpo do paciente, a partir de bobinas de radiofrequência presentes no entorno da máquina. Estes pulsos excitam os átomos de hidrogênio presentes no corpo do paciente, que, por sua vez, re-emitem parte da energia absorvida, até voltarem ao seu estado inicial (alinhados com o campo magnético). Esta energia emitida pelos átomos de hidrogênio do corpo do paciente é, então, captada pelas mesmas bobinas de radiofrequência, e o tempo de reação dos átomos é capturado. Simultaneamente, a energia eletromagnética emitida pelos átomos é localizada por meio de um campo magnético gradiente, gerado por um trio de bobinas que geram campos magnéticos auxiliares, apontados nas direções dos eixos X, Y e Z da máquina.

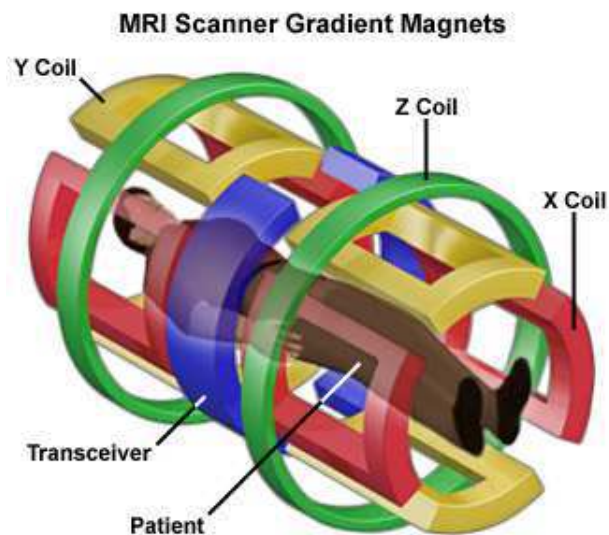
Este processo é ilustrado na Figura 4. Em (a), os átomos de hidrogênio (dipolos magnéticos) estão em seu estado inicial, com suas polaridades distribuídas de forma aleatória ao longo da região. Em (b), um campo magnético de alta intensidade é colocado sobre a região, alinhando todos os átomos de hidrogênio de acordo com a direção do campo magnético. Em (c), uma fonte emite ondas eletromagnéticas que são absorvidas e re-emitidas pelos átomos de hidrogênio em (d). Estas ondas re-emitidas são, então, captadas pela bobina auxiliada por um campo gradiente, gerado por bobinas auxiliares X, Y e Z, como mostrado na Figura 5.

Figura 4 – Funcionamento de uma máquina de MRI.



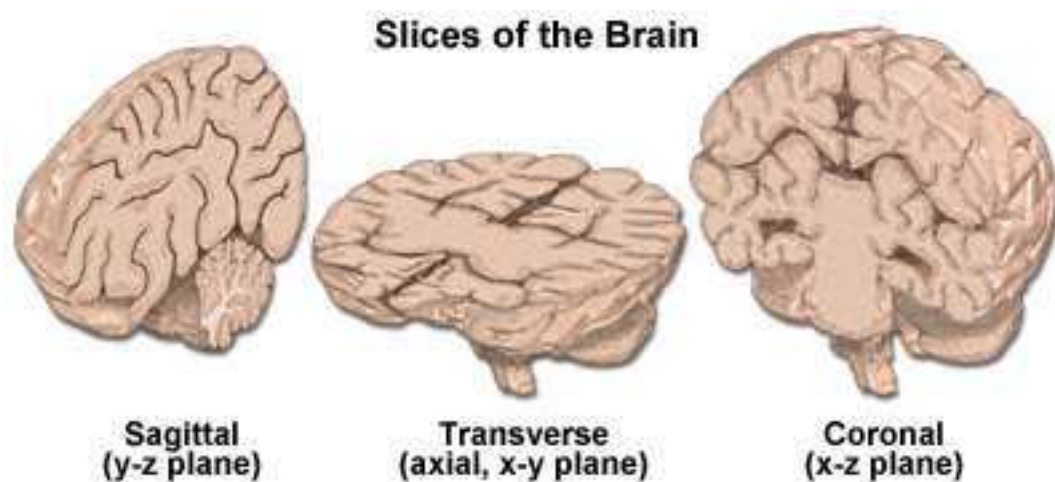
Fonte: (LABORATORY, 2016)

Figura 5 – Geração de campo gradiente.



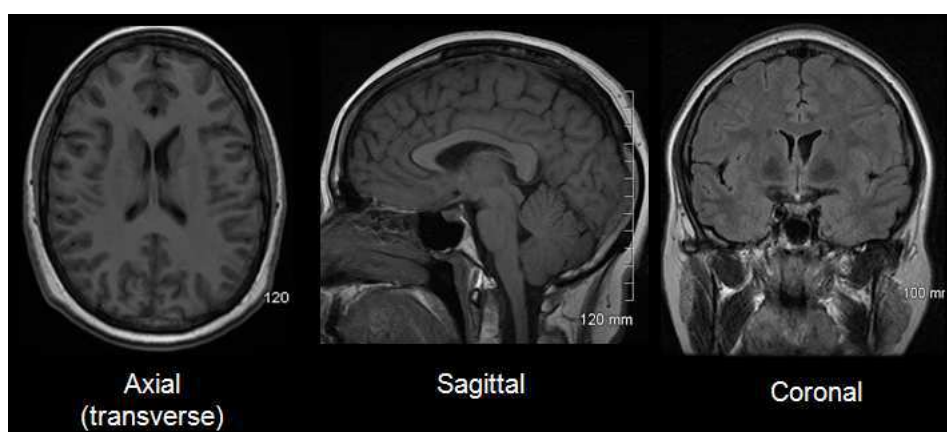
Fonte: (LABORATORY, 2016)

Figura 6 – Ilustração planos de MRI.



Fonte: (LABORATORY, 2016)

Figura 7 – Exemplos reais planos de MRI.



Fonte: (SYSTEM, 2017)

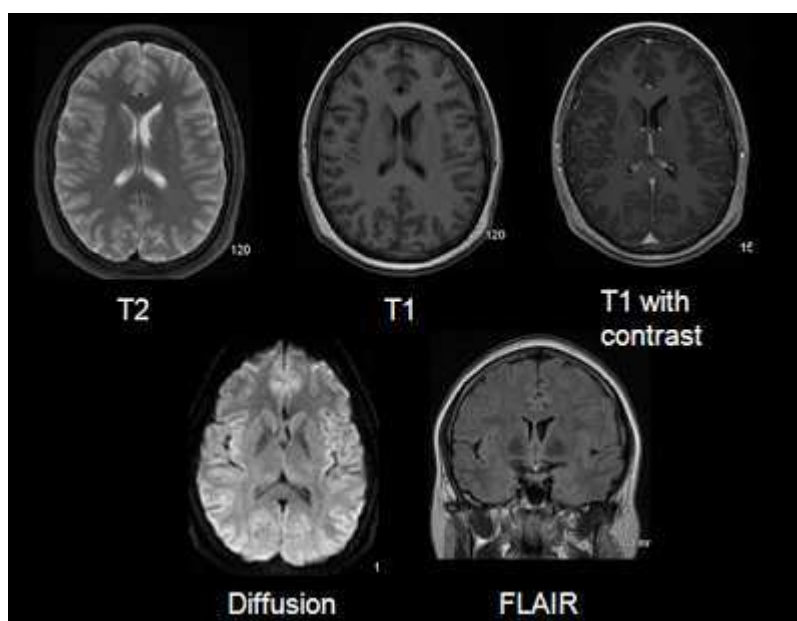
2.2.3 Características e Modalidades

Ressonância magnética possui diversas vantagens, como a possibilidade de realizar exames sem o uso de radiação ionizante como raios-x, além de haver a possibilidade de se obter imagens de diferentes planos ou seções (axial, sagital e coronal) sem a necessidade de reposicionar o paciente, como exemplificado nas figuras [Figura 6](#) e [Figura 7](#).

Além disso, diferentes formas de se aplicar exames de MRI torna possível a exibição ou inibição de diferentes tecidos celulares, como mostrado previamente na [Figura 2](#) e na [Figura 8](#). Com o estudo das interações da radiação utilizadas no MRI sobre diferentes tecidos do corpo humano, pode-se enfatizar determinados tecidos e atenuar outros,

consequentemente tornando este método de imageamento ideal para o diagnóstico de doenças como glioma. Cada tipo de ênfase ou de estudo é chamado de modalidade de MRI. Os pixels contidos nas imagens resultantes de exames de MRI possuem informações que podem ser correlacionadas com difusão de água, fluxo sanguíneo, dentre outros dados, dependendo da modalidade utilizada. Por estes motivos, é comum que se realize uma série de exames em diferentes modalidades de MRI no mesmo paciente, de forma a se obter a maior quantidade de informações possível para otimizar a diagnose.

Figura 8 – Diferentes modalidades de MRI.



Fonte: (SYSTEM, 2017)

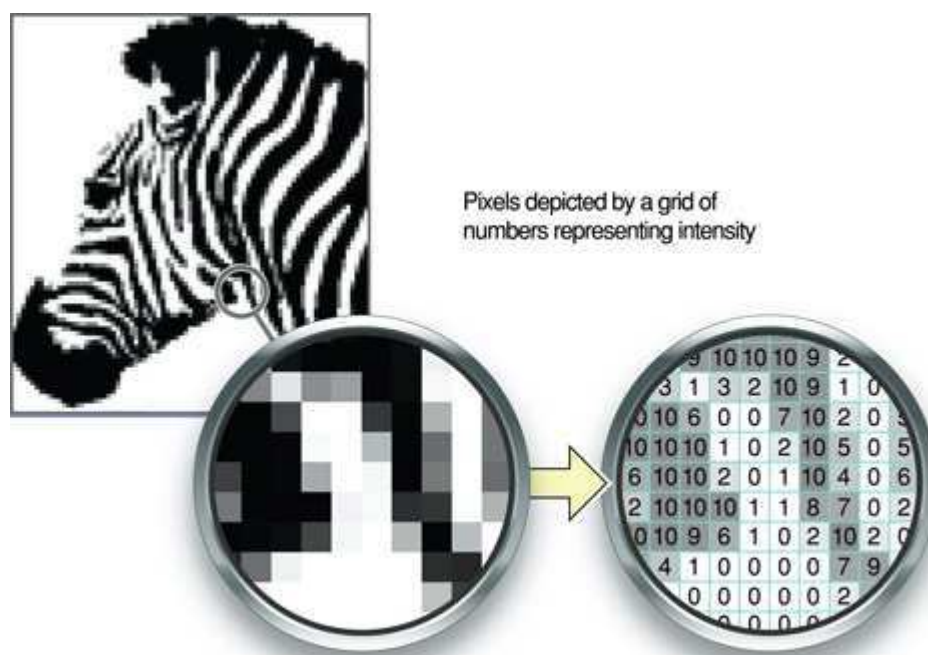
2.2.4 Processamento digital de imagens biomédicas

Imagens de ressonância magnética atuais são extremamente ricas em detalhes e em informações sobre os tecidos moles imageados, fazendo com que técnicas de processamento de sinais e imagens, como filtragem, segmentação e extração de características sejam comumente empregadas em sistemas diagnóstico assistido por computador (CAD).

Após a aquisição e reconstrução das imagens médicas no domínio digital, as imagens passam a existirem virtualmente representadas como matrizes bidimensionais como mostrado na [Figura 9](#). Cada posição dos elementos da matriz correspondem às posições dos pixel da imagem, os valores de cada elemento corresponde à intensidade do sinal de luz ou informação visual que foi digitalizado.

Neste trabalho, são utilizados conceitos como segmentação de imagens, normalização de dados e aprendizado de máquina para segmentação de regiões cancerosas e de tecido

Figura 9 – Como imagens são representadas em um computador.



Fonte: (APPLE, 2016)

edemático em imagens de ressonância magnética.

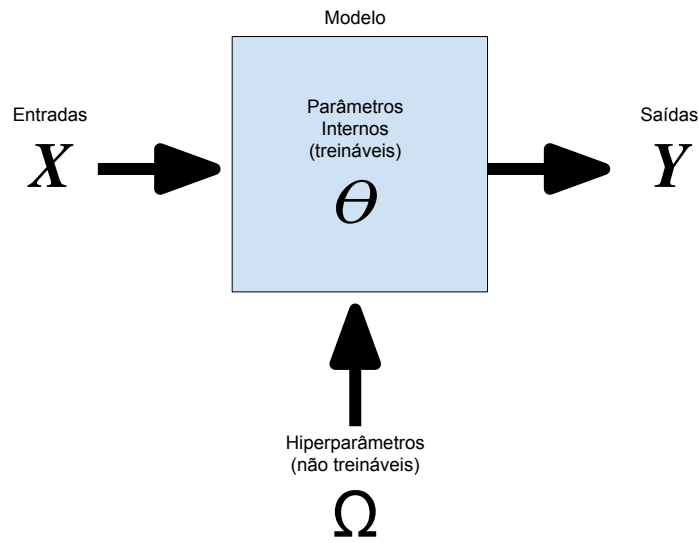
2.3 Aprendizado de máquina

Aprendizado de máquina, do inglês *machine learning* (ML), é uma área da ciência da computação que lida com algoritmos computacionais que resolvem problemas de reconhecimento de padrões. Diversos problemas de processamento, segmentação, regressão, redução de dimensionalidade, classificação e reconhecimento de imagens e sinais dos mais variados podem ser reduzidos a problemas de reconhecimento de padrões solucionáveis via algoritmos estudados na área de aprendizado de máquina. Algoritmos de ML baseiam-se nos preceitos de modelagem, treinamento e inferência, descritos nas seções à seguir.

2.3.1 Modelagem

O primeiro passo em uma *pipeline* tradicional de aprendizado de máquina é a elaboração de um modelo matemático capaz de ser treinado com um conjunto de dados. Redes neurais artificiais, máquinas de vetor de suporte, redes bayesianas, cadeias de markov são alguns exemplos de algoritmos de aprendizado de máquina. Todos estes modelos matemáticos baseiam-se no esquema apresentado na [Figura 10](#).

Figura 10 – Modelo matemático.



Fonte: Autor

onde $Y = f(X, \theta, \Omega)$ representa as saídas $Y = \{y_1, y_2, y_3, \dots, y_m\}$ dadas as entradas $X = \{x_1, x_2, x_3, \dots, x_n\}$, os parâmetros internos $\theta = \{\theta_1, \theta_2, \theta_3, \dots, \theta_k\}$ e os denominados hiperparâmetros $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \dots, \Omega_l\}$ não-treináveis do modelo, e m, n, k, l , respectivamente, o número de saídas, entradas, parâmetros internos e hiperparâmetros do modelo.

As entradas X representam os dados dos quais se quer retirar informações. Estas entradas podem ser, por exemplo, sinais em forma de vetores unidimensionais de comprimento n , imagens em escala de cinza em forma de matrizes bidimensionais $i \times j \times 1$, ou podem ser tensores multidimensionais, como uma série de imagens RGB $i \times j \times 3 \times n$, formando quadros de um vídeo. No caso de imagens médicas, as entradas corresponderiam, por exemplo, a uma imagem de ressonância magnética, representada por uma matriz bidimensional.

As saídas Y representam as saídas resultantes ao passar as entradas X pela função $f(X, \theta, \Omega)$, que representa a atuação do modelo sobre as entradas. Por exemplo, pode-se desejar que o modelo execute uma função de filtragem de ruídos. Os parâmetros θ são variáveis que devem ser ajustadas para que as saídas Y sejam ideais conforme a aplicação. O processo de ajustar estas variáveis é chamado de treinamento.

É importante notar a presença das variáveis denominadas de hiperparâmetros. Hiperparâmetros são os parâmetros que não são passíveis de serem ajustados com base nos dados, pois contêm informações de mais alto nível, como complexidade do modelo e precisão do algoritmo de treinamento. Estes parâmetros são geralmente definidos heurísticamente, e é comum testar diferentes arranjos destes parâmetros para fins de comparação e de

obtenção dos melhores valores para cada aplicação. Este costume, porém, pode gerar consequências negativas, chamadas de *overfitting*, que serão discutidas mais à frente neste trabalho.

2.3.2 Treinamento supervisionado

Uma das etapas mais importantes na construção de modelos com aprendizagem de máquina é a etapa de treinamento. Os dois tipos de treinamento mais comum em algoritmos de ML são os treinamentos supervisionado e não-supervisionado. Para o propósito deste trabalho, trataremos apenas do treinamento supervisionado, pois ele será usado nos métodos aqui implementados.

O treinamento supervisionado começa com um conjunto de dados

$$\{X = \{x_1, x_2, x_3, \dots, x_n\}, T = \{t_1, t_2, t_3, \dots, t_n\}\} \quad (2.1)$$

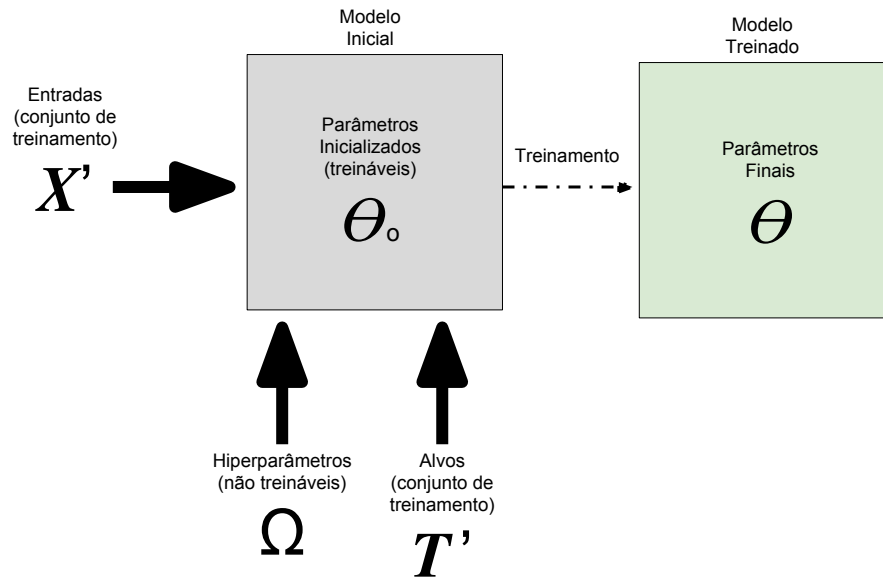
denominado de conjunto de treinamento, onde X são os exemplos (ou amostras) de entradas que irão alimentar o modelo de ML, e T são os exemplos de saídas desejadas. Treinar consiste na busca pelo conjunto de parâmetros internos θ ideais para que, dadas as entradas X , parâmetros θ e saídas desejadas T , as saídas obtidas Y sejam iguais às saídas desejadas T . Estas saídas T são também chamadas de alvos ou rótulos (*targets* e *labels*, respectivamente), e podem ser vetores, matrizes ou qualquer tipo de estrutura de dados. Alguns algoritmos de treinamento são o método do máximo declive, método da regressão linear, método dos mínimos quadrados, dentre outros. O esquema de treinamento supervisionado é ilustrado na [Figura 11](#).

O algoritmo de treinamento começa com um modelo inicial em que todos os parâmetros zerados ou inicializados de forma aleatória. O primeiro passo do treinamento é alimentar o modelo com uma entrada e observar a saída gerada. Esta saída é, então, comparada através de uma função de erro (melhor descrita na [subseção 2.3.3.5](#)) $\mathcal{L}(\theta, \mathbf{x}, \mathbf{t})$, onde \mathbf{x} e \mathbf{t} são, respectivamente, as entradas e saídas desejadas atuais. O resultado da função de erro é utilizado para alterar os parâmetros θ de forma que a função de erro seja minimizada. Após alterar os parâmetros θ , o processo reinicia, desta vez com as próximas entradas e saídas desejadas do conjunto de treinamento. Em suma, o processo de treinamento pode ser resumido como um processo de otimização, na qual se deseja achar o conjunto de parâmetros θ tais que a função \mathcal{L} seja minimizada, ou seja:

$$\Theta^* = \operatorname{argmin}_{\theta} \mathcal{L}(\theta, X, T) \quad (2.2)$$

onde Θ^* é o conjunto de parâmetros ótimos para os quais a função \mathcal{L} obtém seu menor valor, dado o conjunto de treinamento $\{X, T\}$.

Figura 11 – Esquema de treinamento supervisionado.



Fonte: Autor

Algoritmos de treinamento variam de acordo com o modelo de ML utilizado, e seu tempo de convergência depende da complexidade do modelo, como também da qualidade do conjunto de treinamento. Conjuntos de treinamento excessivamente ruidosos, heterogêneos demais, homogêneos demais, pequenos, dados incorretos, pontos fora da curva, dentre outros, são fatores que influenciam tanto na possibilidade de algoritmos de treinamento não convergirem, quanto no tempo necessário para que estes convirjam.

2.3.2.1 Classificação

Nas tarefas de classificação de dados tem-se um conjunto de sinais ou imagens de entrada e se quer treinar um modelo para identificar cada sinal ou imagem pertencente a uma classe específica. Para a realização do treinamento supervisionado é necessário utilizar um conjunto de dados rotulados, na qual associado a cada elemento de entrada tem-se um rótulo associado. Estes dados geralmente são codificados em *one-hot*, como mostrado na [Tabela 1](#), para n classes diferentes:

Por exemplo, pode-se dispor de um conjunto de imagens de animais com suas respectivas espécies pré-classificadas e desejar treinar um modelo para reconhecer uma nova imagem fora deste conjunto como sendo um animal de alguma destas espécies. Um exemplo prático de classificação de dados seria um algoritmo que fosse capaz de rotular um conjunto de sinais de batimentos cardíacos de eletrocardiogramas como batimentos saudáveis ou doentes e desejar treinar um modelo de aprendizado de máquina com este conjunto de dados para que ele possa, depois, ser embarcado diretamente em um eletrocardiógrafo

Tabela 1 – Tabela de codificação *one-hot*.

Classe	Codificação
Classe 1	1 0 0 0 0 ... 0 0
Classe 2	0 1 0 0 0 ... 0 0
Classe 3	0 0 1 0 0 ... 0 0
⋮	⋮
Classe n-1	0 0 0 0 0 ... 1 0
Classe n	0 0 0 0 0 0 ... 1

possibilitando que ele exiba a condição do paciente no ato do exame.

Classificação de dados também tem grande importância na área de imageamento médico. Como existem grandes quantidades de imagens médicas classificadas por especialistas, pode-se utilizar de algoritmos de aprendizado de máquina para, por exemplo, treinar modelos que possam determinar se há ou não alguma anormalidade em uma imagem de ressonância magnética, como câncer ou outras doenças. No entanto, nestes casos, os algoritmos de ML poderão apenas evidenciar a existência ou não de anormalidades, e não poderão localizar na imagem os pixels onde estão estas anomalias.

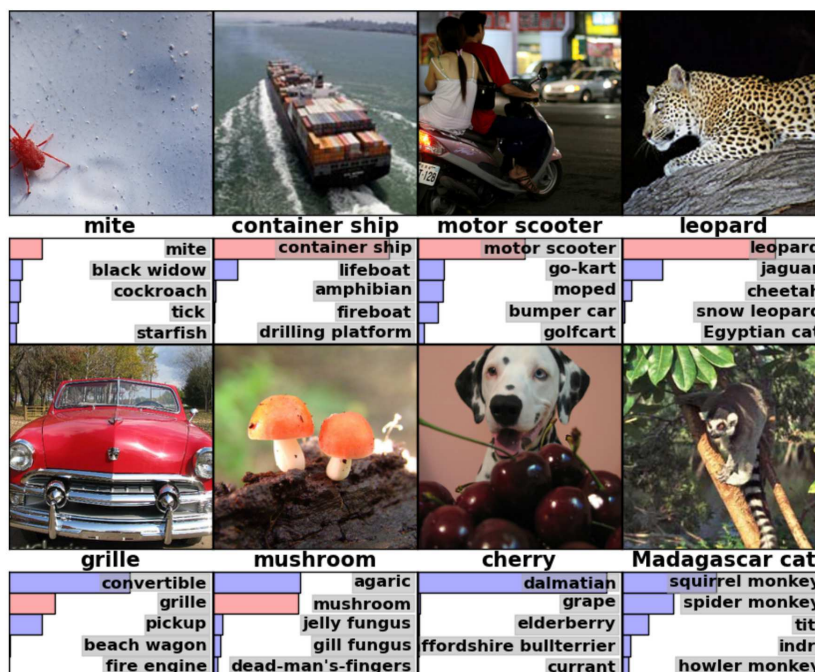
A [Figura 12](#) ilustra um caso de classificação de dados, onde foi utilizado o banco de dados ImageNet, que contém 1.200.000 imagens de paisagens, animais, objetos, plantas, dentre outras classes. São 1000 classes ao todo e cada imagem está pré-classificada como pertencente a uma destas 1000 classes. Os modelos foram treinados com o banco ImageNet de forma a gerar em sua saída um conjunto de probabilidades referentes à chance da imagem em questão ser cada uma das 1000 classes. A imagem mostra apenas as 5 classes de maior probabilidade determinada pelo modelo após o seu treinamento.

2.3.2.2 Segmentação

De forma similar à classificação, segmentação busca identificar e classificar informações contidas em um conjunto de dados. O que difere segmentação de classificação é o fato de que segmentar significa localizar estas informações dentro dos dados de entrada. Por exemplo, em uma imagem de uma rua, pode-se procurar segmentar objetos, como carros, placas e pedestres. Um algoritmo de segmentação, então, precisa ser capaz de identificar quais pixels na imagem de entrada correspondem a quais objeto.

Em algoritmos de segmentação supervisionada de imagem, utiliza-se de um conjunto de dados de entrada X e de saídas desejadas T . As saídas T são imagens, geralmente de mesma resolução das imagens de entrada, porém que representam as regiões da imagem original a serem segmentadas. Por exemplo, caso queira dizer se tal pixel é um pixel cuja localização representa um dado anormal na imagem original, ele deve estar preenchido com valor 1, e todos os outros pixels que não são anormais com 0. Estas saídas são chamadas

Figura 12 – Classificação ImageNet.



Fonte: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

de *ground truth*, ou de máscaras de segmentação. Elas representam as regiões equivalentes às da imagem original nas quais os objetos procurados estão presentes. Quando as classes de objetos são múltiplas, é necessário existir uma imagem máscara para cada classe, de forma análoga à codificação one-hot previamente explicada. A Figura 13 ilustra melhor este conceito. Segmentação de múltiplas classes é conhecida também como segmentação semântica ou multi-classes.

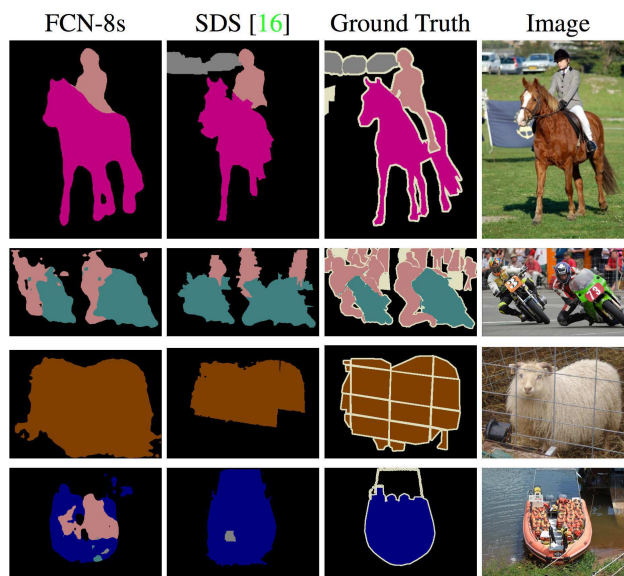
Para segmentação de regiões no cérebro afetadas por câncer ou edema, existem bancos de dados como o BRATS 2016 (MENZE et al., 2014), melhor descrito na seção 3.1. A Figura 14 mostra exemplos de imagens deste banco dados. Algo interessante a ser notado é que a segmentação proposta pelo BRATS é não somente multi-classes, mas também multi-modal. Isto quer dizer que são múltiplas imagens de entradas, cada uma obtida a partir de um método de imageamento diferente, e múltiplas máscaras de segmentação, ou seja, várias classes, cada uma contendo regiões e características distintas a serem separadas.

2.3.3 Redes neurais artificiais

Redes neurais artificiais são um dos algoritmos ou modelos de ML mais utilizados para tarefas de classificação e segmentação de imagens médicas nos dias de hoje, graças à sua marcante performance nestas tarefas (SHIN et al., 2016).

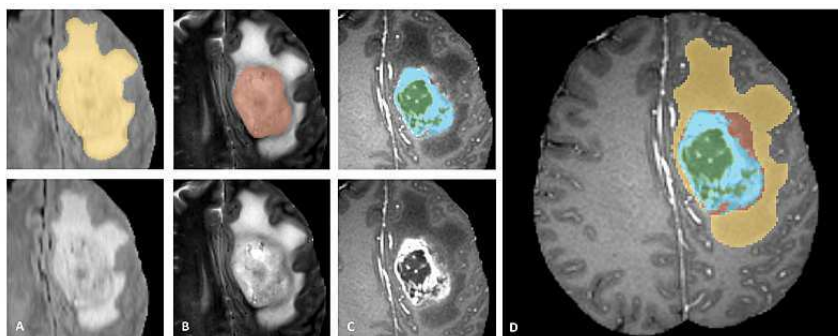
O elemento mais básico de uma rede neural é um "neurônio artificial", que é

Figura 13 – Segmentação semântica.



Nesta imagem, dois algoritmos de segmentação (duas primeiras colunas à esquerda) são comparados com as máscaras (*ground truth*), representadas por cores diferentes para cada classe de objeto e com imagem original. Fonte: (LONG; SHELHAMER; DARRELL, 2015)

Figura 14 – Segmentação de glioma e edema cerebral em MRI.

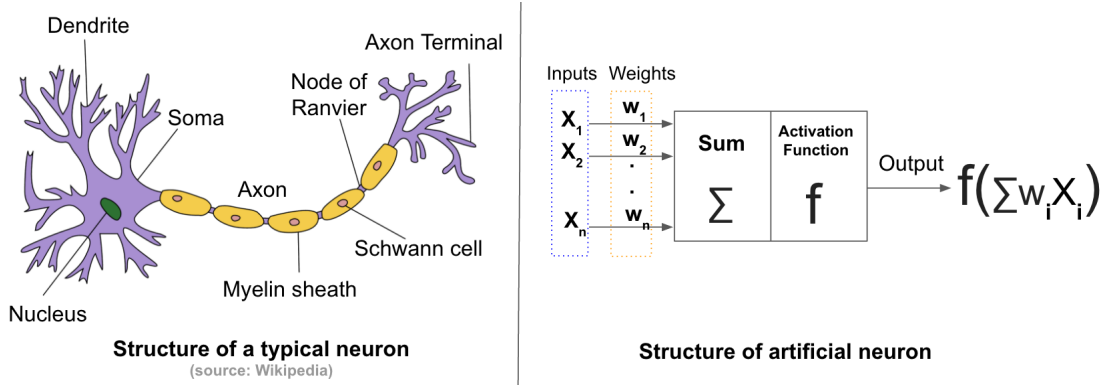


Nesta imagem, diferentes máscaras de segmentação são mostradas em cores distintas, bem como as imagens de entrada obtidas por diferentes modalidades de MRI (a, b e c). Em (d), as máscaras são sobrepostas sobre uma das imagens de entrada. Fonte: (MENZE et al., 2014)

modelado como na Figura 15. Inspirado superficialmente pelo funcionamento dos neurônios biológicos presentes no sistema nervoso central humano, o neurônio artificial possui uma série de entradas $x_1 \dots x_n$ (análogos aos dendritos dos neurônios humanos), que se somam de forma ponderada com respectivos pesos $w_1 \dots w_n$ na região de "processamento" central (análogo ao soma do neurônio humano), que, por sua vez, passa por um canal (análogo ao axônio), modelado pela função de ativação $f()$, e uma saída (terminal do axônio)

$y = f(\sum w_i x_i)$. Para entender o funcionamento de redes neurais mais complexas, é necessário primeiramente entender o modelo mais simples de rede neural, o perceptron simples.

Figura 15 – Neurônio biológico e neurônio artificial.



Assim como os neurônios biológicos, neurônios artificiais recebem sinais de entrada, os processam, e emitem uma saída proporcional à soma ponderada das suas entradas. Fonte: (MAGIZBOX, 2017)

2.3.3.1 Perceptron simples

A rede neural artificial mais simples compreende apenas um neurônio por entrada, e é chamada de perceptron simples (ROSENBLATT, 1961). O perceptron pode ser considerado como um simples classificador binário e possui uma função de ativação do tipo limiar, que pode ser definida como na equação Equação 2.3.

$$y = f(\mathbf{w}^T \cdot \mathbf{x}) = f(\sum w_i x_i) = \begin{cases} 1 & \text{se } y \geq 0 \\ 0 & \text{c.c.} \end{cases} \quad (2.3)$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad (2.4)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (2.5)$$

onde y é a saída, \mathbf{x} é o vetor de entradas e \mathbf{w} é o vetor de pesos.

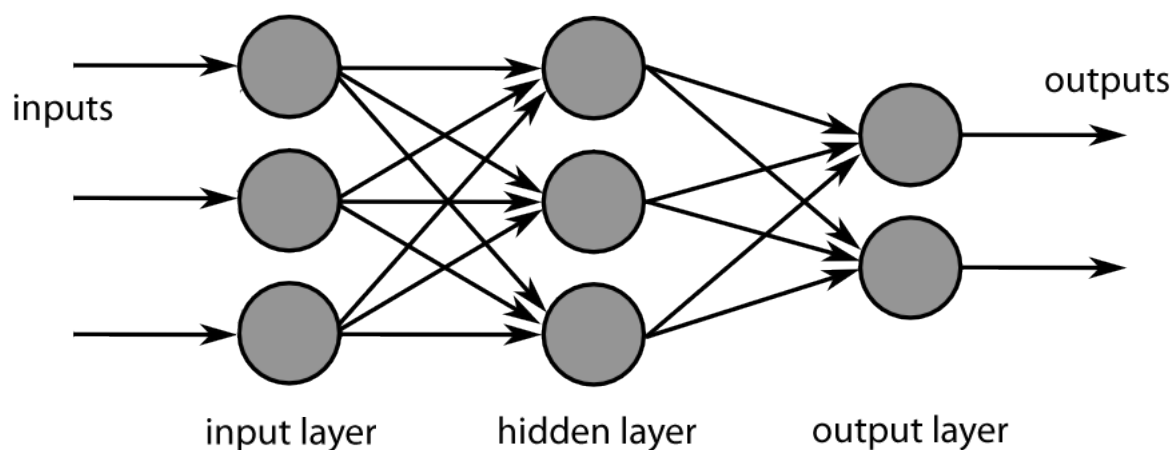
O perceptron pode ser usado para realizar tarefas de classificação simples, porém possui grandes limitações graças à sua simplicidade e natureza binária. Para resolver problemas mais complexos como segmentação ou classificação de imagens, se usa um

arranjo de perceptrons, ordenados em várias camadas, e com funções de ativação variadas. Estes são chamados de perceptron de múltiplas camadas.

2.3.3.2 Perceptron de múltiplas camadas

O perceptron de múltiplas camadas, ou *multilayer perceptron* (MLP), é comumente considerado como sendo a primeira "rede" neural propriamente dita, por ser um arranjo de camadas de neurônios. A Figura 16 ilustra como uma rede MLP é organizada. Esse tipo de rede é chamado de redes *feedforward*, pois a informação segue um caminho direto da entrada até a saída, sem recursividade ou retroalimentação.

Figura 16 – Esquema de uma rede MLP.



Fonte: (MAGIZBOX, 2017)

Cada nó corresponde a um neurônio na Figura 16, e cada ligação entre os nós correspondem a saídas das camadas anteriores, que tornam-se entradas das camadas posteriores. A primeira camada de uma rede MLP é chamada de camada de entrada, a última camada é chamada de camada de saída, e as camadas intermediárias são chamadas de camadas escondidas. Os parâmetros internos treináveis das redes MLP são seus pesos W , já os seus hiperparâmetros não treináveis são o número de camadas, número de neurônios nas camadas escondidas e suas funções de ativação, descritas na subseção 2.3.3.3.

Neste esquema, podemos definir uma matriz de pesos W para cada camada da rede como:










$$W = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_n \end{bmatrix} \quad (2.6)$$

onde $w_1 \dots w_n$ são os pesos de cada neurônio com relação ao vetor de entradas.

2.3.3.3 Funções de ativação

Funções de ativação, nas redes neurais artificiais, são as funções para as quais são passadas as somas ponderadas das entradas de cada neurônio. Na literatura, diversos tipos de funções de ativação são usadas, alguns exemplos são mostrados na [Figura 17](#)

Figura 17 – Exemplos de função de ativação.

Identity		$f(x) = x$	Rectifier (ReLU) ^[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	Parametric Rectified Linear Unit (PReLU) ^[10]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	Exponential Linear Unit (ELU) ^[11]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	SoftPlus ^[12]		$f(x) = \log_e(1 + e^x)$
ArcTan		$f(x) = \tan^{-1}(x)$			

Fonte: Wikipedia

A escolha da função de ativação depende da aplicação para a qual o modelo será treinado. Caso deseje-se obter saídas com valores entre -1 e 1, é necessário escolher funções como a identidade, arcotangente, tangente hiperbólica. Caso queira resultados entre 0 e 1, deve-se utilizar funções como a ReLU (linear retificada), logística, softplus, entre outras. No geral, se usa funções logarítmicas ao invés de lineares para aplicações que envolvem dados naturais, graças à característica não-linear das informações naturais, porém alguns estudos indicam que usar determinadas funções de ativação não-lineares pode fazer com que as saídas tendam a saturar próximas de 1 ou de 0, o que pode afetar o resultado final do treinamento, bem como o tempo necessário para o algoritmo de treinamento convergir ([LECUN et al., 2012](#)). Para evitar este tipo de problema, alguns métodos como *Batch Normalization* e *Dropout* foram desenvolvidos. Estes são explicados na [subseção 2.3.4.2](#).

2.3.3.4 Treinamento via Retropropagação do Erro e Gradiente Descendente

Um método tradicional de treinar redes neurais artificiais é chamado de treinamento via retropropagação do erro (*error backpropagation*) através do máximo declive, ou método do gradiente descendente (*gradient descent*). Este método consiste em alterar os pesos da rede neural na direção na qual a função de erro definida se minimiza, para determinadas entradas e saídas desejadas (conjunto de treinamento $\{X, T\}$).

O primeiro passo do método de retropropagação do erro é a inferência, ou propagação. Primeiro, um vetor de entradas \mathbf{x} é aplicado à rede neural inicializada, fazendo com

que cada camada da rede emita suas saídas que, por sua vez, são utilizadas como entrada à camada seguinte, até que se obtenha uma saída no final da rede neural.

No segundo passo, calcula-se o erro da saída obtida \mathbf{y} com relação a uma saída desejada \mathbf{t} , à partir de uma função de erro. O objetivo do treinamento é minimizar a função de erro para todas as entradas e saídas simultaneamente. Para isso, é necessário modificar os pesos, dados por uma matriz chamada de matriz de pesos atuais no instante n , $W(n) = (w_1, w_2, \dots, w_n)$. Os pesos da rede neural devem ser modificados de forma a obter um erro o mais próximo do zero possível, ou seja, os pesos devem ser movidos na direção na qual esta função minimiza. Para calcular esta direção, utiliza-se do conceito de gradiente.

Gradientes são operações matemáticas que resultam em um vetor que indica a direção na qual uma função $\mathcal{L}(w_1, w_2, \dots, w_n)$ cresce. Então, utiliza-se este operador para atualizar a matriz de pesos atual $W(n)$ na direção oposta à do gradiente da função de erro, ou seja, na direção na qual a função de erro diminui, resultando na matriz de pesos atualizada $W(n + 1)$:

$$W(n + 1) = W(n) - \alpha \cdot \nabla \mathcal{L}(W(n), \mathbf{x}, \mathbf{t}) \quad (2.7)$$

em que α , chamada de taxa de aprendizado (*learning rate*), é um hiperparâmetro ajustável que serve para definir a taxa de convergência do algoritmo, e $\nabla \mathcal{L}(W(n), \mathbf{x}, \mathbf{t})$ é o gradiente definido pela [Equação 2.8](#).

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \dots, \frac{\partial \mathcal{L}}{\partial w_n} \right) \quad (2.8)$$

O algoritmo procede recursivamente com a inserção de novas entradas, até que a função de erro seja minimizada para todas as entradas existentes no banco de dados.

Algumas considerações devem ser levadas em conta na escolha da função de erro, como explicado na seção seguinte.

2.3.3.5 Funções de erro

As funções de erro desempenham um papel extremamente importante nos algoritmos baseados em redes neurais. Isto se dá pois o objetivo final dos algoritmos é minimizar a função de erro que, caso seja mal definida, resultará em minimização de erro sem muita significância para a resolução do problema.

Por exemplo, em problemas de regressão, nos quais se deseja modelar uma função qualquer, não é desejável utilizar o erro médio absoluto como a função de erro, pois ao se somar erros negativos e positivos, pode resultar em um erro nulo, quando, na verdade, os

erros positivos e negativos se somaram, resultando em zero. Para este caso, o erro médio quadrático é mais apropriado.

2.3.3.5.1 Mean Squared Error (MSE)

O erro médio quadrático é definido na [Equação 2.9](#).

$$\mathcal{L} = MSE = \frac{1}{n} \sum_{i=1}^n (t_i - y_i)^2 \quad (2.9)$$

em que t_i e y_i são as i -ésimas amostras dos vetores alvo \mathbf{t} e saídas \mathbf{y} , de comprimento \mathbf{n} .

O erro médio quadrático tem vantagem de eliminar as interferências entre os erros negativos e positivos que poderiam causar, de forma equivocada, valores nulos de erro.

2.3.3.5.2 Crossentropy

Em problemas de classificação, é comum definir saídas utilizando a codificação *one-hot* como discutido anteriormente. Utilizando esta codificação, pode-se fazer uso de funções de ativação como a função softmax ([Equação 2.10](#)), que permite que as saídas da rede neural tenham uma interpretação probabilística

$$f_j(z) = \frac{e^{y_j}}{\sum_k e^{y_k}} \quad (2.10)$$

para $j = 1, \dots, k$, onde \mathbf{y}_j são as saídas obtidas, y_j , seus elementos, e y_k , as saídas de cada classe possível ([UNIVERSITY, 2016](#)). A função de erro equivalente é dada abaixo.

$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right) \quad \text{ou, equivalentemente,} \quad L_i = -f_{y_i} + \log \sum_j e^{f_j} \quad (2.11)$$

A entropia cruzada, ou *cross entropy* é uma função de erro comumente utilizada com a função de ativação softmax, pois ela computa a entropia entre a saída desejada e a obtida, determinando um grau logarítmico de casamento entre elas, resultando em valores entre 0 e 1 para cada elemento da saída ([UNIVERSITY, 2016](#)). Assim, a entropia cruzada é comumente utilizada em modelos de ML para classificação e segmentação de dados.

2.3.3.5.3 Dice score

O *Dice score*, ou *Sørensen-Dice coefficient*, é uma forma de fácil interpretação de comparação de conjuntos. Em problemas de segmentação de imagens, por exemplo, a prática mais comum é utilizar máscaras binárias de segmentação, como discutido anteriormente. Estas máscaras são representadas no computador por matrizes contendo apenas valores 0s

e 1s. Os valores 1s estão presentes nas posições da matriz correspondentes às posições de pixel de uma imagem nas quais o elemento a ser segmentado está presente.

O escore Dice é uma métrica de comparação entre a máscara ideal e a máscara obtida pelo algoritmo de segmentação, e funciona a partir da contagem de elementos iguais entre as duas máscaras ou matrizes de segmentação, normalizados pelo número total de elementos das matrizes. O escore Dice é, logo, calculado da seguinte forma:

$$Dice = \frac{2|T \cap Y|}{|T| + |Y|} \quad (2.12)$$

onde T e Y são, respectivamente, as matrizes das saídas ideais e obtidas e o operador $||$ corresponde ao número de elementos. Normalizado de acordo com o número total de elementos, o valor obtido do Dice score está entre 0 e 1. O oposto do Dice score pode ser utilizado como uma função de perda:

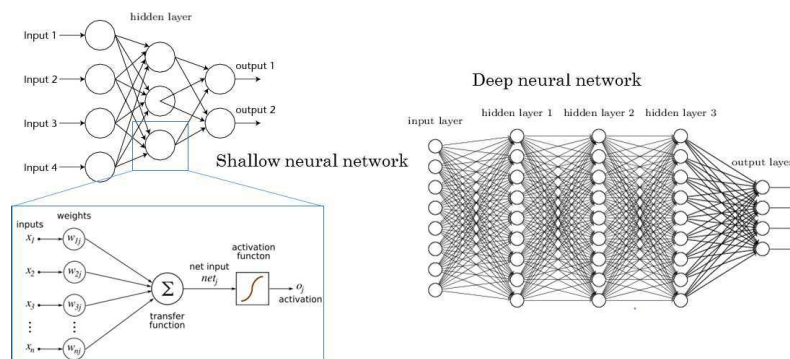
$$Diceloss = -\frac{2|T \cap Y|}{|T| + |Y|} \quad (2.13)$$

O Dice score possui grandes vantagens em algoritmos de segmentação de imagens, por prover um único parâmetro da qualidade da segmentação geral, e é o *benchmark* mais utilizado nos problemas de segmentação da literatura.

2.3.4 Aprendizagem profunda

Aprendizagem profunda é um subcampo da inteligência artificial que estuda o projeto, treinamento e aplicações de modelos de ML de alta profundidade e complexidade, como redes neurais com múltiplas camadas (Figura 18).

Figura 18 – Ilustração de rede neural "rasa" e "profunda".

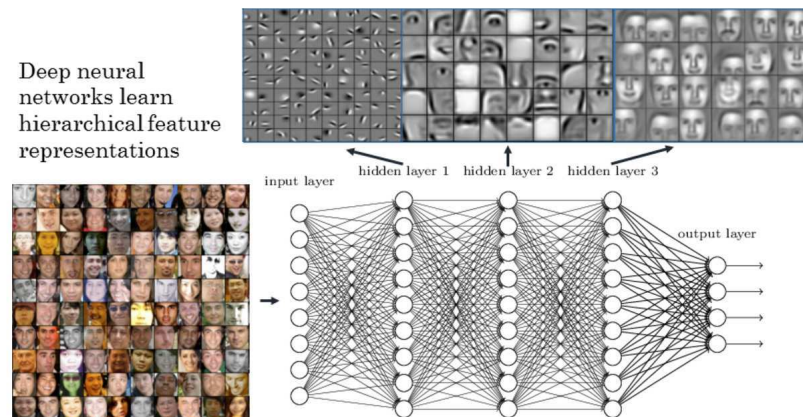


Fonte: (RSIP, 2017)

Uma das vantagens de desenvolver redes neurais com várias camadas é que, à medida em que mais camadas são adicionadas, a rede se torna cada vez mais capaz

de "aprender" conceitos mais abstratos no conjunto de treinamento. Isto se dá pois cada camada acrescenta um nível a mais de não-linearidade à camada anterior. A [Figura 19](#) ilustra este processo.

Figura 19 – Capacidade de abstração das redes neurais profundas.



Fonte: ([RSIP, 2017](#))

O maior problema decorrente de acrescentar camadas a redes neurais é que a quantidade de dados necessária para a convergência do treinamento aumenta conforme a complexidade da rede neural. Ou seja, para treinar uma rede mais profunda, é necessário muito mais dados. Além disso, o processamento computacional para executar uma iteração do algoritmo de treinamento como a retropropagação do erro também aumenta ([CHO et al., 2015](#)).

A necessidade de uma grande quantidade de dados aliada ao maior tempo de treinamento por iteração resulta treinamentos prolongados e grandes chances do algoritmo não convergir ou convergir para mínimos locais e não gerais da função de erro ([DAUPHIN et al., 2014](#)).

Apesar destas desvantagens, fatores como o crescente aumento no poder de processamento dos computadores modernos, o advento de técnicas avançadas de paralelização de dados e placas de vídeos cada vez mais acessíveis, além do grande número de dados disponíveis na internet dos mais diversos tipos, tornam factíveis o uso das técnicas de aprendizado profundo. Modelos de DL têm obtido resultados surpreendentes nas mais diversas áreas, desde reconhecimento de imagens, controle de carros autônomos e segmentação de imagens médicas, até reconhecimento de voz, tradução de texto e geração de voz a partir de texto.

Nas próximas subseções, são tratadas as redes neurais do tipo convolucional, que são as redes neurais mais utilizadas em algoritmos de classificação e segmentação de imagens médicas.

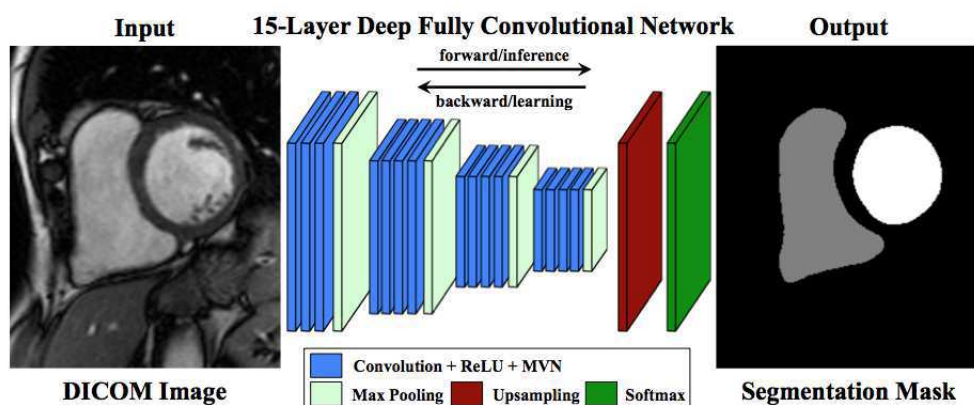
2.3.4.1 Redes Neurais Convolucionais

Um grande problema que ocorre ao se utilizar redes neurais do tipo perceptron de múltiplas camadas para processar informações de imagens é que a quantidade de parâmetros treináveis é proporcional ao número de pixels da imagem. Como exemplo, utilizar redes neurais para serem treinadas em imagens com resolução de 1000x1000 pixels resultaria em uma rede com $1000^2 = 1000000$ neurônios de entrada. Para uma rede com apenas 10 neurônios na camada escondida e 10 classes de saída, um total de $1000000 \times 10 \times 10 = 100.000.000$ de parâmetros precisariam ser treinados. Este é um número excessivo de parâmetros, o que resultaria em um treinamento prolongado.

Redes neurais do tipo convolucional, ou *convolutional neural networks* (CNNs), são similares a redes neurais do tipo perceptron de múltiplas camadas, pois também são compostas por neurônios com pesos e funções de ativação. O que as difere das redes neurais tradicionais é o fato de que os seus pesos não são valores escalares, mas matrizes ou núcleos convolucionais, e a operação realizada entre os pesos e as entradas deixa de ser uma simples multiplicação e passa a ser uma convolução espacial. Ao invés de colocar a sequência de pixels como um vetor unidimensional na entrada da rede, uma CNN pode receber como entrada imagens bidimensionais, nas quais as convoluções espaciais serão executadas.

A [Figura 20](#) ilustra um exemplo típico de arquitetura de rede neural convolucional para aplicação em imageamento médico. Redes neurais convolucionais possuem camadas convolucionais e camadas de reamostragem (chamadas de *pooling*). Convoluções espaciais são descritas na [subseção 2.3.4.1.1](#) e a operação de pooling, na [subseção 2.3.4.1.2](#).

Figura 20 – Ilustração de uma CNN típica.



Fonte: (TRAN, 2016)

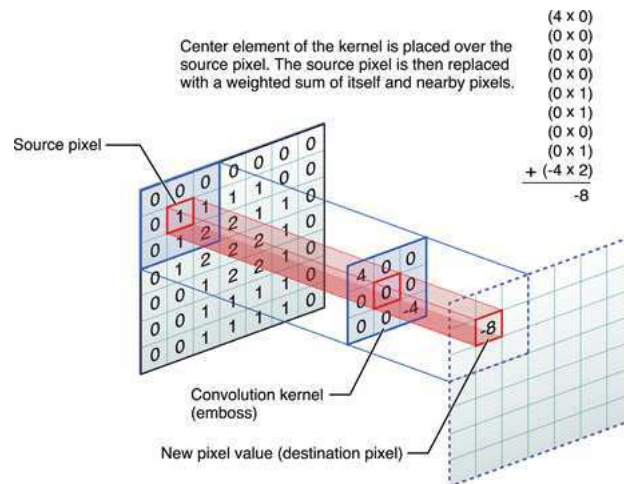
2.3.4.1.1 Camadas convolucionais

Convoluções espaciais são operações matemáticas entre duas matrizes, a matriz a ser operada e a matriz núcleo. A matriz denominada de núcleo ou kernel percorre se sobrepondo à matriz a ser operada, multiplicando seus elementos um-a-um com os elementos da matriz a ser operada. No final, as multiplicações são somadas, resultando em um valor escalar. Este valor escalar é, então, guardado em uma terceira matriz, a matriz resultante. Esta operação é ilustrada na [Figura 21](#), e pode ser definida como:

$$S(i, j) = \sum_{u=-2k-1}^{2k+1} \sum_{v=-2k-1}^{2k+1} N(u, v)E(i - u, j - v) \quad (2.14)$$

onde S , N e E , são as matrizes de saída, núcleo, e entrada, respectivamente.

Figura 21 – Esquema de uma convolução bidimensional.



Fonte: ([APPLE, 2016](#))

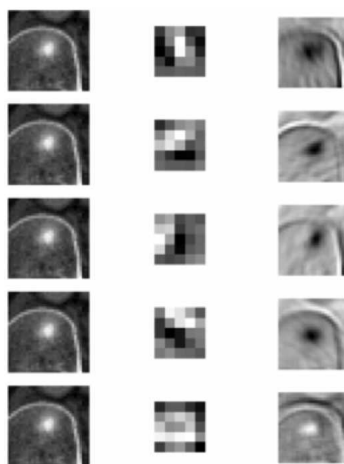
O uso de convoluções no lugares de pesos das redes neurais faz com que o número de parâmetros treináveis necessários seja drasticamente reduzido. Desta forma, redes neurais convolucionais possuem a vantagem de serem menos exigentes computacionalmente, possibilitando a expansão do número de camadas. Além disso, convoluções espaciais retém informação da imagem por completo, não somente de um pixel ou região da imagem. Isto se dá pelo fato de que a convolução é uma operação sobre todos os pixels da imagem. Adicionalmente, o número de núcleos convolucionais por camada pode ser definido pelo projetista, bem como o tamanho dos núcleos e o passo da operação de convolução. Estes, no entanto, se tornam hiperparâmetros, ou parâmetros não treináveis, passíveis de serem determinados heurísticamente pelo usuário.

A operação de convolução resulta na modificação da imagem original de diversas formas diferentes, por isso a convolução também é chamada de filtragem. A [Figura 22](#) mostra

alguns exemplos de como núcleos de convolução diferentes podem afetar distintamente a mesma imagem. No [Apêndice A](#), uma operação de convolução em imagem RGB é demonstrada a partir de uma ilustração mais intuitiva.

É importante salientar que, após realizadas as convoluções, os resultados de todas elas são somadas em um único valor, que é levado a uma função de ativação, assim como nas redes neurais tradicionais. O valor resultante é, então, atribuído ao respectivo pixel da imagem de saída. Também é comum considerar a função de ativação como uma camada à parte da camada convolucional.

Figura 22 – Exemplo de diferentes núcleos na mesma imagem.



Diferentes núcleos, representados na forma de imagens (coluna central), realizam diferentes operações sobre a mesma imagem, como realce de borda horizontal, vertical, suavização, etc. Fonte: ([SUMMERS, 2015](#))

2.3.4.1.2 Camadas *pooling*

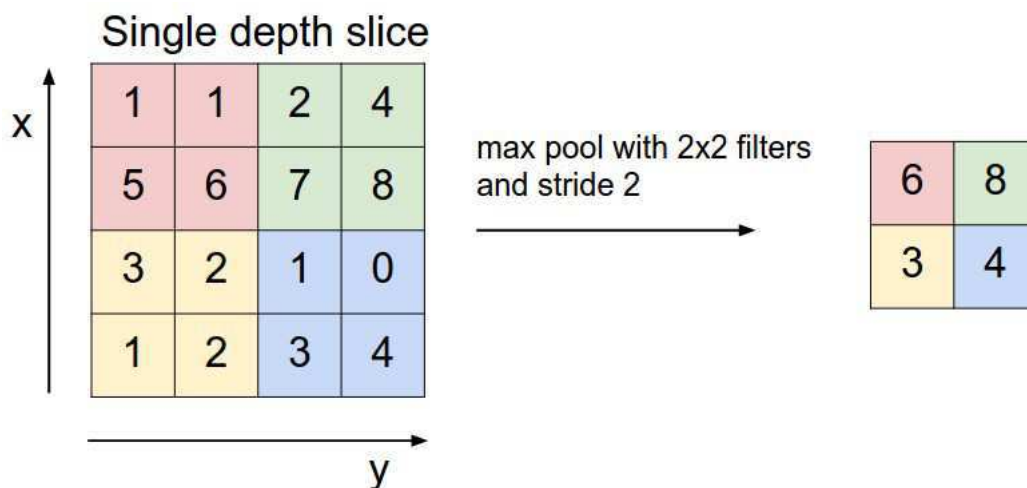
Além das camadas convolucionais, também é comum a utilização de camadas de subamostragem ou superamostragem nas CNNs, chamadas de camadas de *pooling*. Estas camadas realizam operações bem mais simples do que as convoluções. Fundamentalmente, as camadas pooling realizam operações que resultam na redução das dimensões das imagens de entrada. O processo de *pooling* é mostrado na [Figura 23](#).

Pooling é semelhante à convolução no sentido em que uma matriz é colocada sobreposta e é comparada a outra. No entanto, a operação de pooling gera uma matriz com valores médios (*mean-pooling*), mínimos (*min-pooling*) ou máximos (*max-pooling*), para cada posição da matriz ou imagem original.

Pooling serve para reduzir a dimensionalidade das matrizes de forma simplificada, além de adicionar ao modelo invariância ao escalonamento, pois cada camada subsequente

irá ser treinada com uma imagem em uma escala reduzida. Técnicas como *max-pooling* permitem reter apenas as informações de maior intensidade das camadas anteriores, eliminando redundâncias e acelerando o processamento e convergência do treinamento da rede neural.

Figura 23 – Funcionamento da operação de *max-pooling*.



Exemplo de max-pooling 2x2. A operação obtém os valores máximos de cada trecho da matriz original, resultando numa matriz de dimensão reduzida em 1/2 no eixo x e 1/2 no eixo y. Fonte: (UNIVERSITY, 2016)

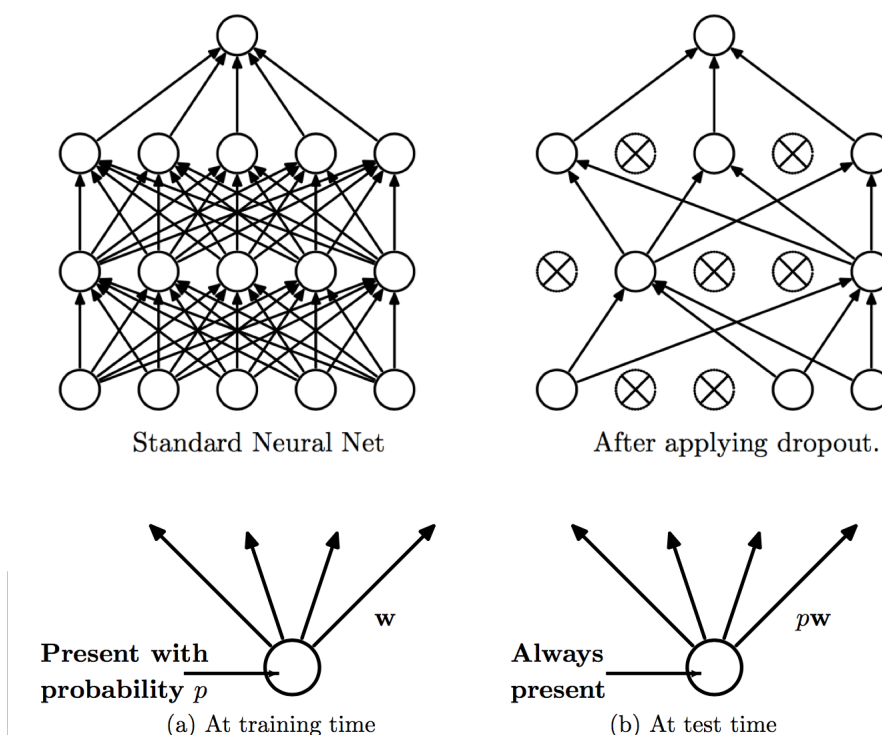
2.3.4.2 Regularização

Regularização é o procedimento realizado no treinamento de redes neurais que serve para evitar o '*overfitting*'. Nesta seção, duas técnicas de regularização são brevemente explicadas.

2.3.4.2.1 Dropout

A técnica de *dropout*, que significa, em tradução livre, "cair fora", é uma forma bastante simples de evitar com que as redes neurais "decorem" o banco de dados no qual ela está sendo treinada. Dropout consiste em eliminar, ou zerar, os pesos de uma rede neural a cada iteração do treinamento, com uma probabilidade de ocorrência p . Durante a fase de inferência, após a rede ser treinada, os pesos que a rede neural aprendeu são então multiplicados por p . Este algoritmo parece ser não intuitivo, porém provou-se melhorar a performance de convergência do treinamento de redes neurais profundas, além de tornar as características aprendidas menos ruidosas nas camadas intermediárias, facilitando a sua visualização.

Figura 24 – Funcionamento do dropout.



No dropout, os pesos da rede neural são eliminados com probabilidade de p no treinamento, mas são amplificados com ganho de p na fase de inferência. Fonte: (SRIVASTAVA et al., 2014)

2.3.4.2.2 Batch normalization

Batch normalization, ou normalização do lote, em tradução livre, é outra técnica de regularização de algoritmos de DL. Esta técnica faz uso do conceito de treinamento por lote, onde, ao invés de enviar uma amostra de entrada por vez no treinamento da rede neural, uma porção do conjunto de treinamento é enviado. Isto possibilita que haja uma normalização por lote do conjunto de treinamento.

A técnica *batch normalization* surgiu ao perceber que as saídas de cada camada são, para a camada subsequente, simples entradas como quaisquer outras. O que ocorre é que estas entradas das camadas subsequentes podem não estar normalizadas dentro de todo o banco de dados, e elas geralmente não estão. Isso significa que as camadas escondidas da rede neural vão estar recebendo entradas não-normalizadas, o que dificulta o treinamento e tende a gerar "overfitting". (IOFFE; SZEGEDY, 2015)

Então, o princípio básico de batch normalization é resolver esse problema a partir do envio de um conjunto de dados por vez, normalizando as saídas de cada camada ao redor de suas médias e desvios unitários.

Como vantagens, além de regularizar e prevenir o overfitting da rede, foi provado

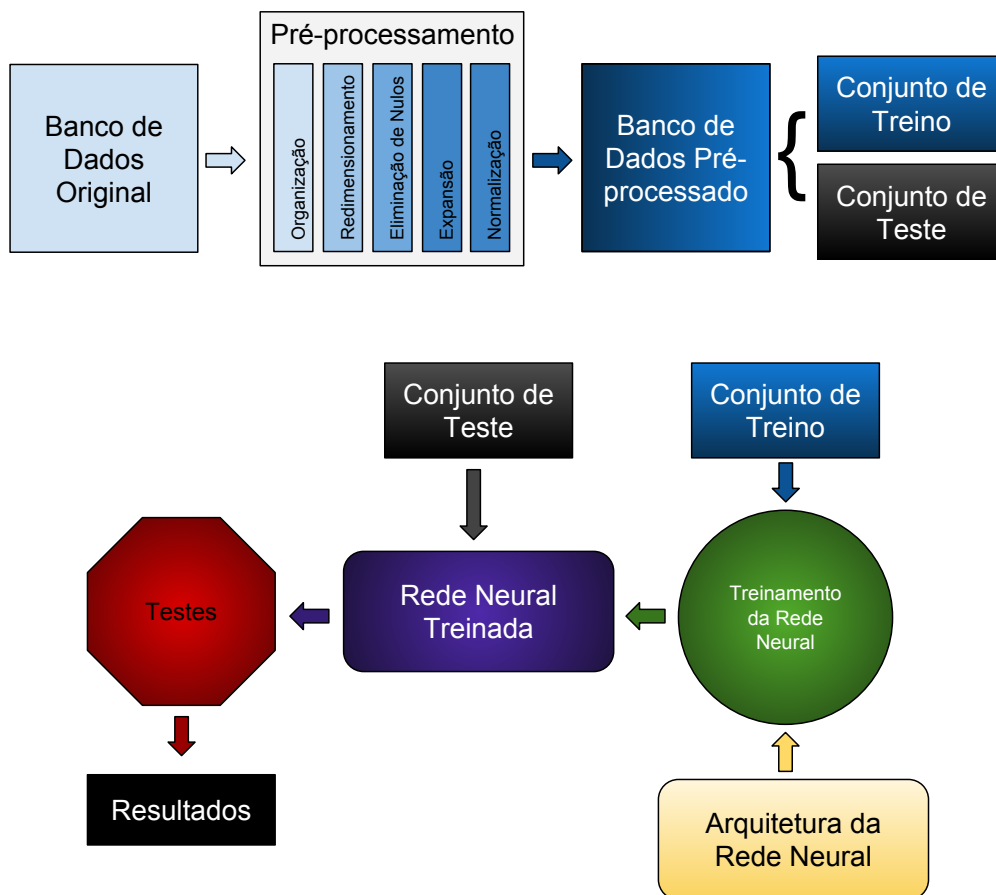
que o uso de *batch normalization* reduz a dependência dos valores iniciais dos pesos de uma rede neural, reduz a tendência a saturação das funções de ativação, e também possibilita o uso de taxas de aprendizado mais elevadas, efetivamente diminuindo o tempo necessário para treinamento.

3 Metodologia

Neste capítulo, é apresentada a metodologia utilizada para o desenvolvimento de um algoritmo segmentador de tumores cerebrais em imagens de ressonância magnética, como proposto neste trabalho de conclusão de curso.

A [Figura 25](#) representa um sumário do processo implementado neste trabalho, para referência.

Figura 25 – Esquema de processamento e treinamento implementado.



Fonte: Autor

A metodologia utilizada neste trabalho foi a seguinte:

- Revisão bibliográfica sobre os problemas enfrentados no imageamento médico moderno;
- Revisão bibliográfica das técnicas de utilizadas para resolver tais problemas e dos conjuntos de dados abertos disponíveis;

- Retirada dos dados do banco BRATS 2016;
- Estudo das diferentes arquiteturas de redes neurais utilizadas nas tarefas de segmentação médica;
- Pré-processamento do conjunto de dados para tornar possível o treinamento da rede em tempo prático;
- Divisão do conjunto pré-processado em dois subconjuntos: treino e teste
- Treinamento da rede neural e aquisição dos resultados
- Análise dos resultados
- Preparação do relatório

O programa foi todo desenvolvido utilizando Python 2.7, com ajuda de bibliotecas *open source* como *VTK*, *SimpleITK* e *OpenCV* para o pré-processamento dos dados, *Keras* para o desenvolvimento da rede neural, e *scipy*, *numpy* e *matplotlib* para visualização e organização dos dados, além de *Jupyter* para o desenvolvimento e *debugging* interativo do *software*.

3.1 Banco de dados BRATS 2016

O banco de dados *Multimodal Brain Tumor Segmentation Challenge* - BRATS foi criado para facilitar a validação e comparação entre estudos e estratégias de segmentação de tumores cerebrais a partir de dados de imagens multimodais, ou seja, provenientes de várias modalidades de imageamento médico simultâneas. A edição de 2016 conta com conjuntos de dados obtidos nos anos de 2012 e 2013 pelo próprio BRATS, além de imagens cedidas pelo NIH Cancer Imaging Archive, um centro virtual e global de imagens médicas de câncer. (MENZE et al., 2014)

Este banco de dados foi escolhido por diversos motivos. Primeiramente, todas as imagens das distintas modalidades são registradas por paciente, ou seja, imagens do cérebro obtidas de cada paciente estão alinhadas, não havendo a necessidade de estudar métodos de alinhamento, o que iria consumir tempo e desviar do tema central deste trabalho. Além disso, o banco contém imagens de quatro modalidades de MRI: T2 FLAIR, T2, T1 com contraste e T1. Todas as imagens, obtidas durante as mesmas sessões de imageamento de cada paciente, foram interpoladas para a resolução de voxel de 1mm^3 , e segmentadas manualmente por até quatro profissionais médicos. As segmentações foram feitas de forma a separar o núcleo do tumor, a região afetada diretamente pelo tumor, a região necrótica (morta) pelo tumor e a região afetada por edema. A Figura 2 mostra um exemplo da segmentação realizada (MENZE et al., 2014).

3.1.1 Estrutura dos dados

O banco de dados está organizado em pastas separadas inicialmente pelo grau do tumor, baixo grau (1 a 2, ou benigno), ou alto grau (3 a 4, maligno). Dados de 220 indivíduos com tumores malignos e 54 com tumores benignos estão presentes. Para cada indivíduo, os dados de imagens de cada modalidade estão separados em sua subpasta. As imagens têm resolução de 240 por 240 pixels, e são 155 imagens (fatias, eixo z) por modalidade. São 4 modalidades ao todo. Aliado às imagens de MRI, estão as máscaras segmentadas de cada característica: núcleo do câncer, região afetada pelo câncer, região morta, região edemática e uma última máscara marcando apenas a região saudável, ou não afetada por edema nem câncer. Estas máscaras estão em apenas uma imagem por fatia, sendo que cada característica corresponde a um valor específico: 1, 2, 3, 4 e 5. Um simples algoritmo foi criado para separar estas máscaras em 5 imagens, sendo uma a mais para identificar a região não afetada por tumor nem edema. A [Figura 29](#) ilustra um exemplo deste banco resultante.

3.2 Pré-processamento

Nesta seção, serão tratados as etapas de tratamento dos dados para utilizá-los no treinamento da rede neural de forma eficiente.

3.2.1 Organização

A primeira etapa é a organização do banco de dados. Todos os dados foram importados ao ambiente Python e organizados na forma de tensores, ou matrizes multidimensionais, de tamanhos dados pela [Tabela 2](#).

Tabela 2 – Tamanhos dos tensores de dados assim que importados ao ambiente Python

Dado	Tamanho
Imagens MRI	$274 \times 4 \times 155 \times 240 \times 240$
Máscaras de segmentação	$274 \times 5 \times 155 \times 240 \times 240$

Como neste trabalho os dados foram tratados bidimensionalmente, o tensor foi reformatado de forma a empilhar todas as fatias de todos os pacientes em uma só pilha, eliminando um dos eixos do tensor. O tamanho dos tensores resultante é dado na [Tabela 3](#).

Tabela 3 – Tamanhos dos tensores após empilhamento.

Dado	Tamanho
Imagens MRI	$42470 \times 4 \times 240 \times 240$
Máscaras de segmentação	$42470 \times 5 \times 240 \times 240$

Desta forma, é possível acessar qualquer imagem a partir de uma especificação numérica no primeiro eixo, por exemplo, acessando o elemento 50 seria dado da seguinte forma: $\text{matriz}[50]$, retornando uma matriz $k \times 240 \times 240$ ($k = 4$ para as imagens MRI e $k = 5$ para as máscaras de segmentação), semelhante a o que ocorreria em uma imagem RGB ($k = 3$ matrizes, uma para cada canal de cor).

3.2.2 Redimensionamento

Na segunda etapa, todas as imagens foram redimensionadas da resolução de 240×240 para 60×60 , ou seja, um subdimensionamento de 4x. O algoritmo para decimação das imagens escolhido foi a decimação bilinear, por ser rápida e não causar artefatos de alta frequência como na decimação por vizinho mais próximo (*nearest neighbour*). A [Figura 30](#) mostra um exemplo de imagem após o redimensionamento. Notar que ambas as máscaras e as imagens de MRI foram redimensionadas.

Este processo foi realizado para tornar viável o treinamento da rede neural em tempo prático. Apesar desta vantagem, inevitavelmente o processo de subdimensionamento resultará em uma redução da performance da rede neural por causa da perda de dados. Sem este procedimento, no entanto, o treinamento da rede neural poderia durar até 5 dias no computador aqui utilizado, de acordo com testes iniciais. Reduzindo as imagens, os treinamentos puderam ser feito em cerca de 8 horas, o que tornou praticável a possibilidade de realizar múltiplos testes com arquiteturas de redes neurais diferentes. A [Tabela 4](#) mostra como ficou o conjunto de dados resultante do redimensionamento.

Nas imagens de máscara, que originalmente contiam apenas valores 0 e 1, o processo de decimação bilinear gerou valores intermediários entre 0 e 1, que poderiam ser arredondados, gerando o equivalente a uma decimação *nearest neighbour*. Isto não foi realizado, pois os valores intermediários podem ser úteis para o treinamento da rede neural, já que há informação contida neles, e eliminá-los equivaleria a eliminar estas informações.

Tabela 4 – Tamanhos dos tensores após redimensionamento.

Dado	Tamanho
Imagens MRI	$42470 \times 4 \times 60 \times 60$
Máscaras de segmentação	$42470 \times 5 \times 60 \times 60$

3.2.3 Eliminação de elementos nulos

A terceira etapa do préprocessamento envolveu a eliminação do excesso de elementos nulos nos dados. O banco de dados, por compreender exames de ressonância magnética do cérebro humano ao longo de um eixo vertical, possui uma quantidade de imagens nas quais não há máscara de segmentação, por não haver câncer nestas imagens. A [Figura 31](#) mostra um exemplo de imagem nula.

Esta etapa serve para homogeneizar o banco de dados, evitando que a rede neural seja treinada com dados que em sua grande maioria sejam de apenas um tipo, ou seja, sem máscara de segmentação. Isso tornaria o treinamento prolongado e poderia gerar mínimos locais nulos no treinamento da rede.

O algoritmo de eliminação consiste em descobrir aleatoriamente no banco de dados as imagens cujos valores das máscaras são todos nulos. Uma contagem é realizada para registrar o número de elementos e quais elementos são nulos, e uma porcentagem pré-definida destes elementos é eliminada.

É importante frisar que é uma boa prática no treinamento de redes neurais é utilizar um conjunto de dados semelhante aos do mundo real. Na vida real, quando a rede fosse ser utilizada, muitas imagens poderiam ser imagens sem algum tipo de anomalia como câncer ou edema, isto quer dizer que não é útil simplesmente eliminar todos as imagens que não contém câncer, mas apenas uma parte delas. Desta forma, foi definido um valor arbitrário de porcentagem de eliminação, apenas para fins de testes. O valor escolhido foi de eliminar, aproximadamente, 50% dos dados nulos. Este valor não foi alterado em momento algum do treinamento, pois a prática de escolher um valor que maximize a performance da rede poderia significar o desenvolvimento de *overfitting* da rede.

As imagens nulas compreendem cerca de 50% do banco de dados total, isto quer dizer que o banco de dados final ficou com, aproximadamente, 50% (imagens não-nulas) + $50\% \times 50\%$ (parcela não eliminada das imagens nulas) = 75% do tamanho dos dados original. Como o algoritmo decide aleatoriamente com uma probabilidade de 50% se cada imagem nula é eliminada ou não, o número final de amostras do banco de dados varia a cada vez que o programa é rodado. A [Tabela 5](#) mostra uma estimativa média do tamanho do banco de dados após a eliminação dos dados nulos.

Tabela 5 – Tamanhos dos tensores após redimensionamento.

Dado	Tamanho
Imagens MRI	$31852 \times 4 \times 60 \times 60$
Máscaras de segmentação	$31852 \times 5 \times 60 \times 60$

3.2.4 Normalização

A próxima etapa no pré-processamento é a normalização dos dados. Normalizar torna os dados padronizados em escala e valor médio, de forma que os resultados das operações realizadas na rede neural façam com que os valores enviados às funções de ativação sempre estejam na mesma faixa.

As imagens de MRI foram normalizados em escala de 0 a 1, dividindo todos os valores das imagens em escala de cinza (0 a 255) por 256.

As imagens de máscara, por já terem valores entre 0 e 1, não foram modificadas.

3.2.5 Conjunto de treinamento e de teste

O conjunto de dados resultante tem as características descritas pela [Tabela 6](#).

Tabela 6 – Característica do conjunto de dados após pré-processamento.

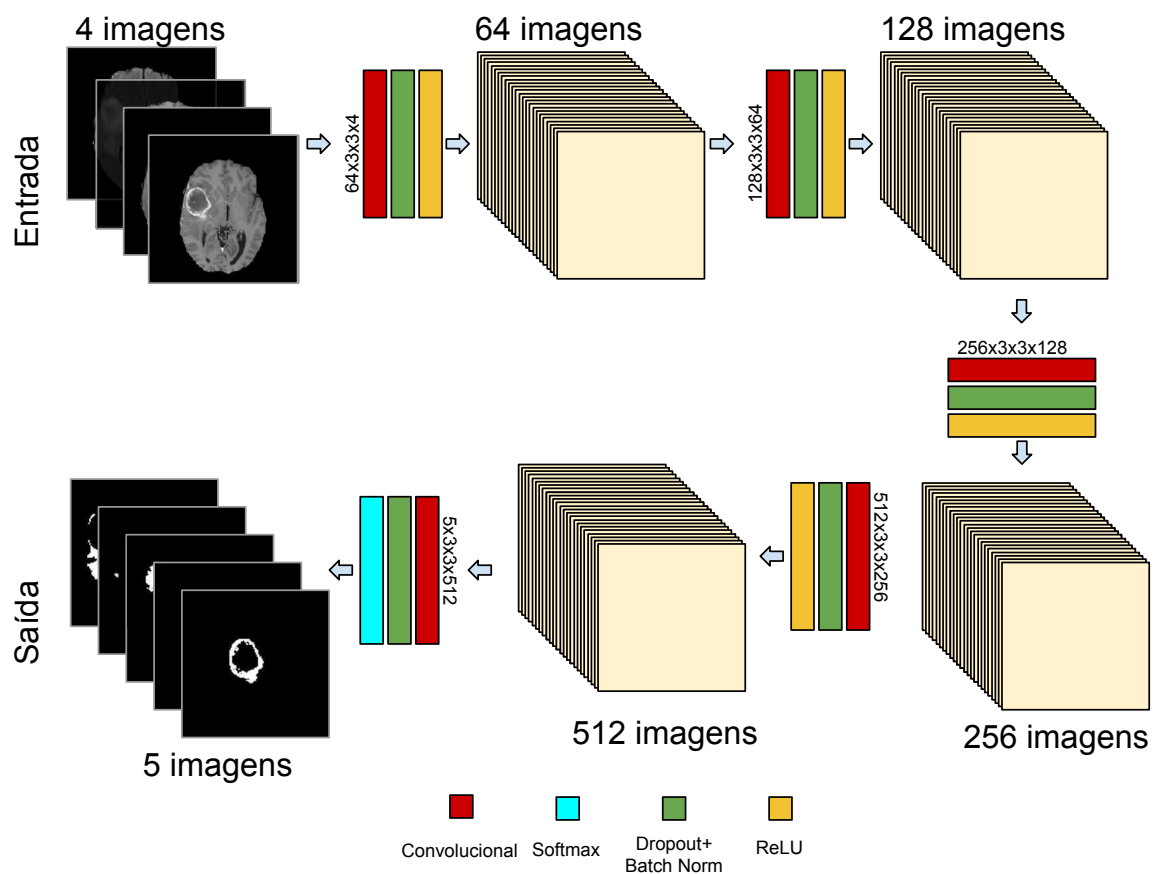
Dado	Tamanho	Faixa de Valores
Imagens MRI	$31852 \times 4 \times 60 \times 60$	0 a 1
Máscaras de segmentação	$31852 \times 5 \times 60 \times 60$	0 a 1

3.3 Arquiteturas propostas das redes neurais

3.3.1 Arquitetura A: Segmentação Multiclasses

A arquitetura de CNN proposta possui 5 camadas convolucionais, com núcleos convolucionais 3×3 . As camadas possuem números de filtros crescentes a cada camada, duplicando o número de filtros da camada anterior. A primeira camada possui 64 filtros como base, e as camadas subsequentes duplicam o número de filtros da camada anterior. Na quinta camada, a camada de saída, apenas 5 filtros são utilizados para unir e filtrar informação das 512 saídas da camada anterior, unindo-as em 5 imagens de saída finais, cada uma representando uma das máscaras de cada classe de saída. A arquitetura é ilustrada na [Figura 26](#). Este esquema deve incentivar o aprendizado de filtros básicos como detecção de borda, suavização e aumento no contraste, nas camadas iniciais, enquanto que as camadas subsequentes podem utilizar as imagens das camadas anteriores para detectar características mais aprimoradas ou complexas. Todas as camadas convolucionais são acompanhadas de *batch normalization* e *dropout*.

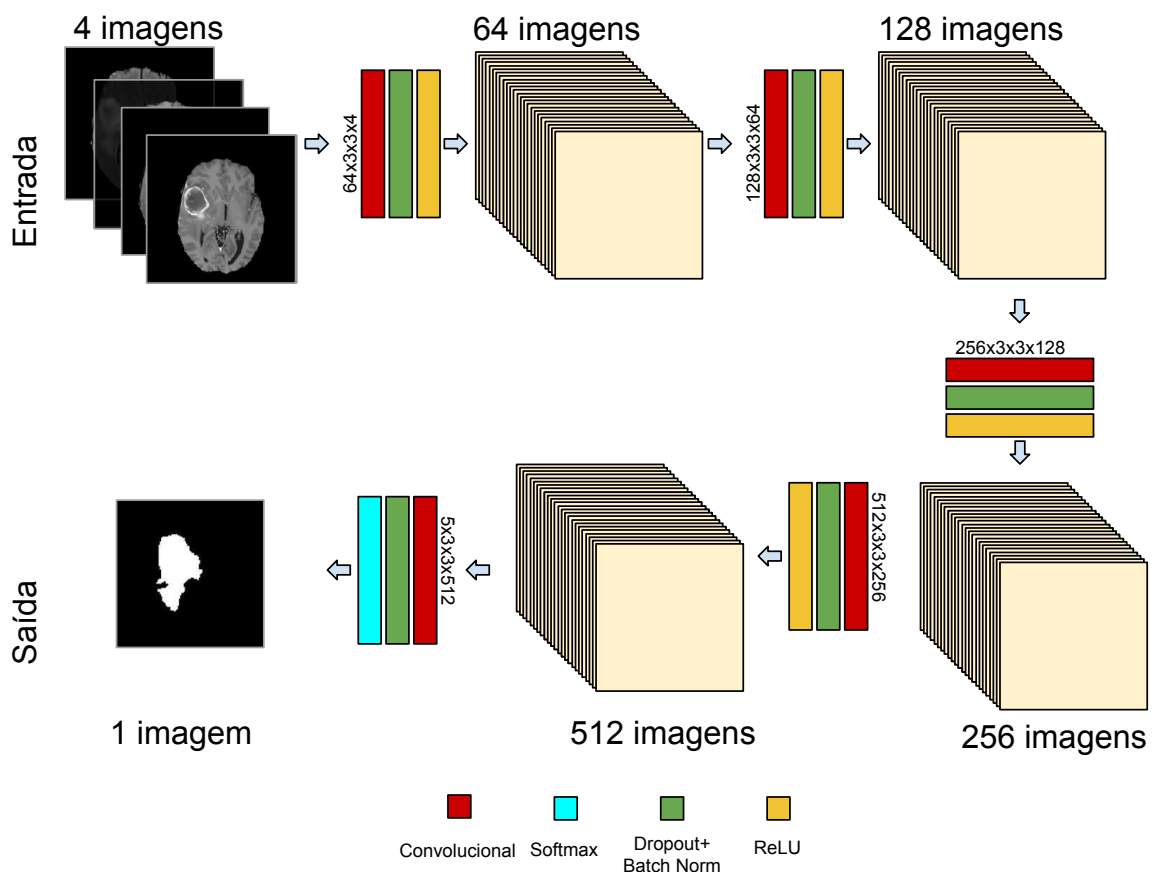
Figura 26 – Arquitetura A: Segmentação Multiclasses



3.3.2 Arquitetura B: Segmentação de Anomalias

Para uma tarefa mais simples, uma arquitetura secundária foi projetada de forma a utilizar apenas a primeira imagem do conjunto BRATS. Esta imagem contém pixels com valor igual a 0 para os correspondentes às posições na imagem original que não possuem anomalia cerebral como edema ou câncer, e pixels iguais a 1 para os que contém estas anomalias. Desta forma, a rede neural pode ser treinada apenas para reconhecer dano ou anomalia cerebral, sem distinguir detalhes internos da anomalia, como proposto na arquitetura anterior. Isto diminui o grau de complexidade da tarefa a ser aprendida pela rede neural, possibilitando maior precisão e menor tempo de treinamento ao todo. A [Figura 27](#) ilustra a arquitetura aqui proposta, que diferencia-se apenas no número de imagens de saída.

Figura 27 – Arquitetura B: Segmentação de Anomalias



3.4 Treinamento da rede neural

3.4.1 Separação do conjunto de dados

Para o treinamento da rede neural, o conjunto de dados resultante foi separado em dois: o conjunto de testes e de treinamento. A distribuição foi feita de forma aleatória para que 70% dos dados fossem para o conjunto de treino e os outros 30%, para o conjunto de teste. Assim, a rede neural poderá ser treinada com um conjunto e testada com um conjunto diferente, de forma a se obter métricas de performance sem viés.

Tabela 7 – Característica do conjunto de treino utilizado na rede neural.

Dado	Tamanho	Faixa de Valores
Entradas (Imagens MRI)	$22296 \times 4 \times 60 \times 60$	0 a 1
Alvos (máscaras de segmentação)	$22296 \times 5 \times 60 \times 60$	0 a 1

Tabela 8 – Característica do conjunto de testes utilizado na rede neural.

Dado	Tamanho	Faixa de Valores
Entradas (Imagens MRI)	$9557 \times 4 \times 60 \times 60$	0 a 1
Alvos (máscaras de segmentação)	$9557 \times 5 \times 60 \times 60$	0 a 1

3.4.2 Testes realizados

Os testes realizados foram feitos com base na premissa de que cada função de perda está sujeita a resultar em redes que realizam tarefas distintas. Assim, as arquiteturas A e B propostas foram treinadas usando duas funções de perda discutidas neste trabalho: o *Dice score* e a *cross entropy*.

A arquitetura A foi treinada com a função de perda *cross entropy*, enquanto que a arquitetura B foi treinada com o *Dice score*. Isto se deu pois o *Dice score* em sua forma original não permite discernimento entre classes diferentes, ele serve apenas para problemas de classificação binária. Este problema não existe no caso da *cross entropy*, por isso este parâmetro foi utilizado no treinamento da arquitetura de segmentação multiclasse A.

Para análise dos resultados, no entanto, as máscaras de segmentação foram consideradas como uma só e comparadas às máscaras ideais de forma a se obter o *Dice score* de cada resultado. Esta análise foi feita para as duas arquiteturas propostas, independente da função de perda usada no treinamento. Ou seja, mesmo na rede treinada com a *cross entropy*, utilizou-se o *Dice score* como métrica quantitativa para a performance do algoritmo de segmentação. Isto possibilitará comparar os resultados entre si de uma forma mais objetiva, com a mesma medida.

4 Resultados

As arquiteturas de rede neural foram treinadas ao decorrer de 3 semanas. O computador utilizado para o treinamento possuía um processador Core i7 (2012, 3,3 GHz) de quatro núcleos, 8 GB de memória RAM 1600MHz embarcada e um disco do tipo SSD.

Após o treinamento das redes propostas, os conjuntos de testes foram aplicados às entradas das redes e, com as saídas desejadas de cada amostra, os *Dice score* foram calculados. Para cada arquitetura, as médias e o desvio padrão de todos os *Dice score* obtidos foram calculados. As tabelas [Tabela 9](#) e [Tabela 10](#) sumariza os resultados.

Tabela 9 – *Dice scores* obtidos da arquitetura A

Média	Desvio padrão
43,33%	11,94%

Tabela 10 – *Dice scores* obtidos da arquitetura B

Média	Desvio padrão
89,61%	18,22%

No [Apêndice C](#), podemos ver exemplos de imagens resultantes do treinamento da rede neural da arquitetura A: multiclases. As imagens [Figura 32](#), [Figura 33](#) e [Figura 34](#) são exemplos dos melhores resultados obtidos com o treino desta rede. Pode-se observar que a rede foi capaz de aprender não somente sobre a área afetada, mas também características mais abstratas da região afetada, sendo capaz de discernir entre os diversos tipos de tecido e danos como edema e região central do tumor. As [Figura 35](#) e [Figura 36](#) mostram exemplos de segmentações menos bem-sucedidas, que envolvem geralmente as partes mais externas do cérebro, onde se possui menos informações no conjunto de treinamento pelo fato de que os tumores geralmente se desenvolvem mais próximos da região mais interna do cérebro. Além disso, estas imagens mostram um ponto fraco da rede que é o de discernir entre os detalhes de maior contraste das bordas do cérebro, e os tumores ou anomalias cerebrais.

A [Apêndice D](#), por sua vez, possui exemplos de resultados do treinamento da arquitetura B. Estes exemplos, particularmente os das imagens [Figura 37](#), [Figura 38](#) e [Figura 39](#), mostram como a rede treinada otimizando o *Dice score* no problema de apenas uma classe é capaz de obter resultados mais robustos do que a do problema de múltiplas classes da arquitetura B. As imagens [Figura 40](#) e [Figura 41](#) mostram exemplos com menor *Dice score*, que coincidem com as regiões mais próximas da borda do cérebro. A [Figura 41](#), em particular, mostra que a falta de detalhes nas imagens pode resultar em uma segmentação difundida, aumentando a região e diminuindo a capacidade da rede de localizar detalhes finos.

5 Conclusão

Neste trabalho, foi realizado um estudo sobre o processamento de imagens biomédicas através de paradigmas que circundam a engenharia elétrica, como aprendizado de máquina e processamento de sinais. Para isso, foi apresentado o problema de detecção de câncer cerebral a partir de imagens médicas como ressonância magnética, um problema relevante na medicina moderna. Para abordar este problema, algoritmos de aprendizado profundo baseados em redes neurais convolucionais foram propostos, e duas arquiteturas de redes neurais profundas foram implementadas e treinadas.

As arquiteturas A e B foram desenvolvidas para segmentação das regiões cerebrais que contém tumores e/ou edemas cerebrais. A arquitetura A foi treinada para localizar não somente os pixels das imagens que contém ou não contém estas anomalias, mas também para caracterizar o tipo de anomalia dentre 5 classes: núcleo do câncer, região afetada pelo câncer, região afetada por edema e região saudável. Uma arquitetura extra (B) foi treinada apenas com o dado de onde está a região saudável, possibilitando a segmentação de anomalias no cérebro.

As arquiteturas de redes neurais treinadas obtiveram resultados promissores, porém a necessidade de redução da resolução das imagens usadas no treinamento pode ter comprometido a performance das redes, levando a altos valores de desvio padrão dos resultados, além de resultados de segmentação com *Dice score* abaixo de 50% de média para segmentação multi-classes (arquitetura A). Para segmentação de anomalias, ou seja, de uma classe apenas (arquitetura B), no entanto, os resultados foram muito positivos, chegando próximo aos 90% de taxa de acerto.

Com estes resultados, pode-se perceber que a tarefa de segmentação de apenas uma classe é uma tarefa de mais simples que a de segmentação multi-classe. Percebe-se que a rede B consegue facilmente localizar danos cerebrais, porém os detalhes mais finos tendem a ser mesclados com a região como um todo. Isto significa que esta rede seria melhor utilizada apenas para aplicações de localização e detecção propriamente dita de anomalias cerebrais, mas não para segmentação fina.

Pôde-se observar em diversos casos de imagem com menos textura e contraste, a região afetada pelo tumor foi incluída na segmentação, porém uma região que a circunda também foi, sugerindo que a rede necessita de maior contraste e textura para ser capaz de segmentar detalhes finos. Esta situação pode ter sido amplificada pela redução da resolução das imagens originais, fato que resulta diretamente na diminuição nos detalhes finos das imagens.

São diversas as melhorias propostas para eventuais estudos continuados deste

trabalho. Algumas delas são: utilização do conceito de *transfer learning*, evitando a necessidade de se treinar redes "do zero"; utilizar computadores mais potentes para tornar desnecessária a redução da resolução das imagens originais; propôr arquiteturas mais profundas, tornando-as capazes de aprender características mais complexas; dentre outras.

Em conclusão, este trabalho mostra-se intrigante ao demonstrar que técnicas computacionais baseadas nos conceitos aprendidos durante a graduação do curso de engenharia elétrica são mais que suficientes para executar tarefas de alta complexidade e relevância de áreas como medicina e biologia, tão longínquas às tradicionalmente abordadas neste curso de graduação.

Referências

- ABOUTCANCER. *MRI Images of Brain Mets*. 2017. Disponível em: <http://www.aboutcancer.com/brain_met_mri_2.htm>. Acesso em: 27 mar 2017. Citado na página 20.
- ACS. *What are the key statistics about brain and spinal cord tumors?* American Cancer Society, 2016. Disponível em: <<http://www.cancer.org/cancer/braincnstumorsinadults/detailedguide/brain-and-spinal-cord-tumors-in-adults-key-statistics>>. Citado na página 16.
- APPENZELLER, S. et al. Quantitative magnetic resonance imaging analyses and clinical significance of hyperintense white matter lesions in systemic lupus erythematosus patients. *Annals of neurology*, Wiley Online Library, v. 64, n. 6, p. 635–643, 2008. Citado na página 16.
- APPLE. *vImage Programming Guide*. 2016. Disponível em: <<https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>>. Acesso em: 27 mar 2017. Citado 2 vezes nas páginas 25 e 40.
- CHO, J. et al. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv preprint arXiv:1511.06348*, 2015. Citado na página 38.
- DAUPHIN, Y. N. et al. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2933–2941. Citado na página 38.
- DESPOTOVIĆ, I.; GOOSSENS, B.; PHILIPS, W. Mri segmentation of the human brain: challenges, methods, and applications. *Computational and mathematical methods in medicine*, Hindawi Publishing Corporation, v. 2015, 2015. Citado na página 17.
- DEVICES, C. for; HEALTH, R. *Medical Imaging*. Center for Devices and Radiological Health, 2016. Disponível em: <<https://www.fda.gov/Radiation-EmittingProducts/RadiationEmittingProductsandProcedures/MedicalImaging/ucm2005914.htm>>. Acesso em: 27 mar 2017. Citado na página 20.
- GU, J. et al. Recent Advances in Convolutional Neural Networks. *ArXiv e-prints*, dez. 2015. Citado na página 17.
- HAVAEI, M. et al. Deep learning trends for focal brain pathology segmentation in MRI. *CoRR*, abs/1607.05258, 2016. Disponível em: <<http://arxiv.org/abs/1607.05258>>. Citado 2 vezes nas páginas 16 e 17.
- IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. Citado na página 43.

JAMES, A. P.; DASARATHY, B. V. Medical image fusion: A survey of the state of the art. *CoRR*, abs/1401.0166, 2014. Disponível em: <<http://arxiv.org/abs/1401.0166>>. Citado na página 16.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>>. Citado na página 30.

LABORATORY, T. N. H. M. F. *MRI: A Guided Tour*. 2016. Disponível em: <<https://nationalmaglab.org/education/magnet-academy/learn-the-basics/stories/mri-a-guided-tour>>. Acesso em: 27 mar 2017. Citado 3 vezes nas páginas 21, 22 e 23.

LECUN, Y. A. et al. Efficient backprop. In: *Neural networks: Tricks of the trade*. [S.l.]: Springer Berlin Heidelberg, 2012. p. 9–48. Citado na página 34.

LIN, Z.-X. *Glioma-related edema: new insight into molecular mechanisms and their clinical implications*. Sun Yat-sen University Cancer Center, 2013. Disponível em: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3845586/>>. Citado na página 19.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 3431–3440. Citado na página 31.

LOUIS, D. N. et al. The 2016 world health organization classification of tumors of the central nervous system: a summary. *Acta Neuropathologica*, v. 131, n. 6, p. 803–820, 2016. ISSN 1432-0533. Disponível em: <<http://dx.doi.org/10.1007/s00401-016-1545-1>>. Citado na página 16.

MAGIZBOX. *Intro to Deep Learning*. 2017. Disponível em: <http://magizbox.com/training/deep_learning/site/introduction/>. Acesso em: 27 mar 2017. Citado 2 vezes nas páginas 32 e 33.

MENZE, B. et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, Institute of Electrical and Electronics Engineers, p. 33, 2014. Disponível em: <<https://hal.inria.fr/hal-00935640>>. Citado 4 vezes nas páginas 17, 30, 31 e 46.

NHS. *Brain tumours*. NHS, 2017. Disponível em: <<http://www.nhs.uk/conditions/brain-tumours/Pages/Introduction.aspx>>. Acesso em: 27 mar 2017. Citado na página 18.

NHS. *Malignant brain tumour*. NHS, 2017. Disponível em: <<http://www.nhs.uk/conditions/brain-tumour-malignant/Pages/Introduction.aspx>>. Acesso em: 27 mar 2017. Citado na página 19.

ONCOMED. *A ONCOMED*. 2017. Disponível em: <<http://www.oncomedbh.com.br/site/?menu=TiposdeCE2ncer&submenu=CE2ncerCerebral>>. Acesso em: 27 mar 2017. Citado na página 19.

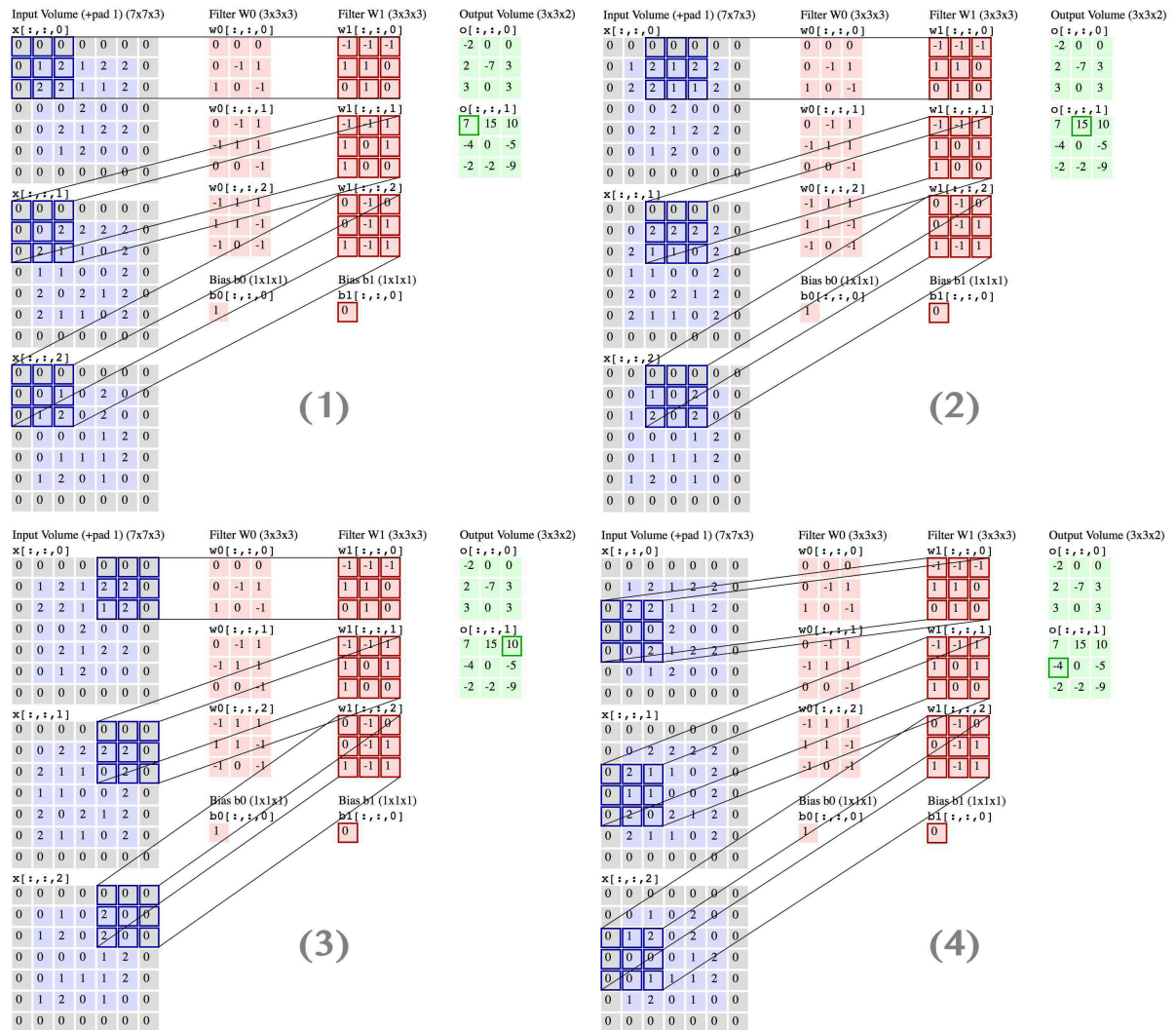
ROSENBLATT, F. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. [S.l.], 1961. Citado na página 32.

- RSIP. *Exploring Deep Learning CNNs*. 2017. Disponível em: <<http://www.rsipvision.com/exploring-deep-learning/>>. Acesso em: 27 mar 2017. Citado 2 vezes nas páginas 37 e 38.
- RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 17.
- SHIN, H.-C. et al. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, IEEE, v. 35, n. 5, p. 1285–1298, 2016. Citado 2 vezes nas páginas 17 e 30.
- SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 43.
- SUMMERS, R. M. Improving computer-aided detection using convolutional neural networks and random view aggregation. 2015. Citado na página 41.
- SYSTEM, B. of Regents of the University of W. *The Basics - Neuroradiology*. 2017. Disponível em: <<https://sites.google.com/a/wisc.edu/neuroradiology/image-acquisition/the-basics>>. Acesso em: 27 mar 2017. Citado 2 vezes nas páginas 23 e 24.
- TAJBAKHSI, N. et al. Convolutional neural networks for medical image analysis: full training or fine tuning? *IEEE transactions on medical imaging*, IEEE, v. 35, n. 5, p. 1299–1312, 2016. Citado na página 17.
- TRAN, P. V. A fully convolutional neural network for cardiac segmentation in short-axis mri. *arXiv preprint arXiv:1604.00494*, 2016. Citado na página 39.
- UNIVERSITY, S. *CS231n Convolutional Neural Networks for Visual Recognition*. 2016. Disponível em: <<http://cs231n.github.io/linear-classify/#softmax>>. Acesso em: 27 mar 2017. Citado 3 vezes nas páginas 36, 42 e 61.
- VERNOOIJ, M. W. et al. Incidental findings on brain mri in the general population. *New England Journal of Medicine*, Mass Medical Soc, v. 357, n. 18, p. 1821–1828, 2007. Citado na página 16.
- WHO. *Tumor Types*. 2017. Disponível em: <<http://braintumor.org/brain-tumor-information/understanding-brain-tumors/tumor-types/>>. Acesso em: 27 mar 2017. Citado 2 vezes nas páginas 16 e 18.
- ZORZI, R. *GLIOMAS*. 2017. Disponível em: <<http://www.raquelzorzi.com.br/gliomas>>. Acesso em: 27 mar 2017. Citado na página 19.

Anexos

ANEXO A – Demonstração da operação de convolução.

Figura 28 – Exemplo de convolução em imagem RGB.



Nesta ilustração, uma imagem RGB, com cada canal representado por uma das três matrizes à esquerda, é operada por dois filtros, representado pelos dois conjuntos de 3 núcleos convolucionais, um para cada canal de cor, no centro, em vermelho. A matriz de saída de cada operação é mostrada à direita, com a saída específica da operação atual sendo acentuada com um quadrado verde. Esta imagem mostra 4 etapas, na ordem correta, da operação de convolução. Notar que esta imagem foi retirada de uma animação e o primeiro conjunto de filtros já foi passado pela imagem e seu resultado está sendo mostrado como a matriz 3x3 verde de cima, à direita. Fonte: (UNIVERSITY, 2016)

ANEXO B – Exemplos do BRATS 2016

Figura 29 – Exemplo de amostra do BRATS 2016 de um único paciente.

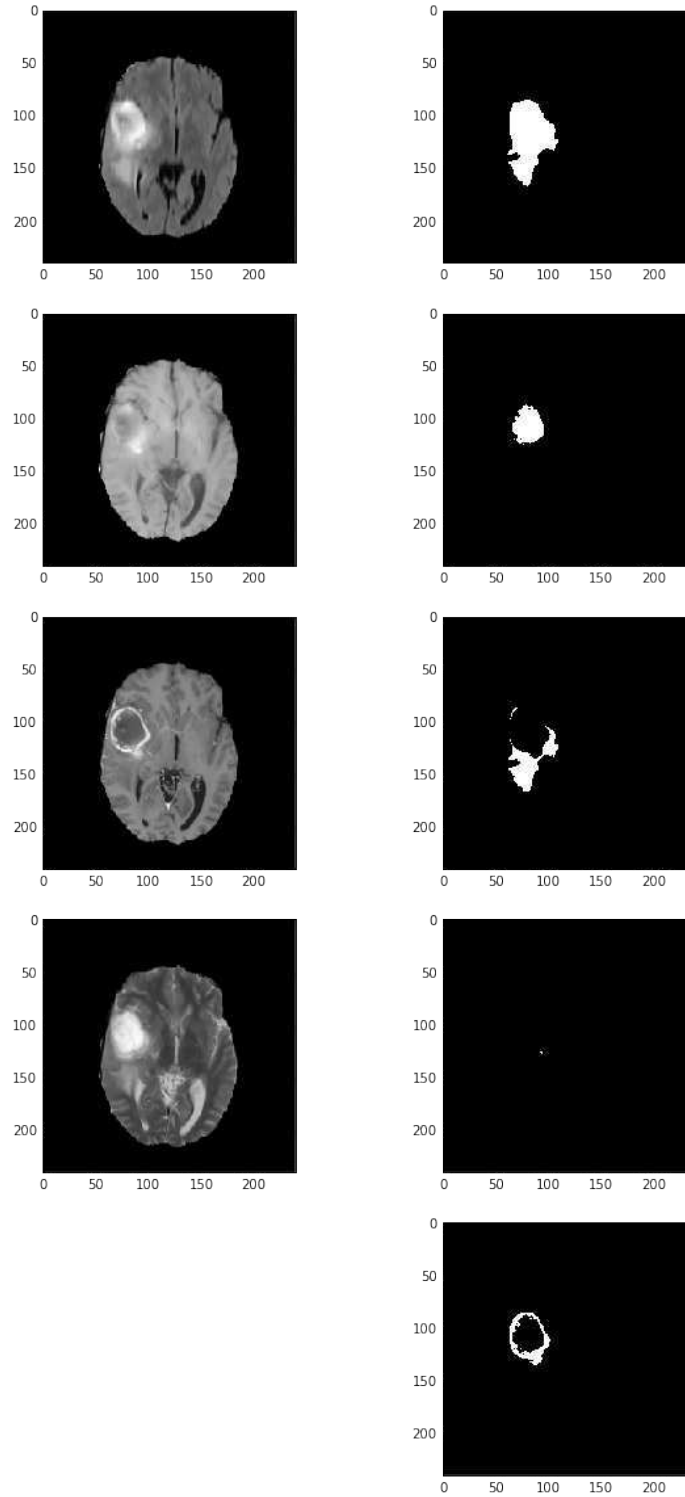


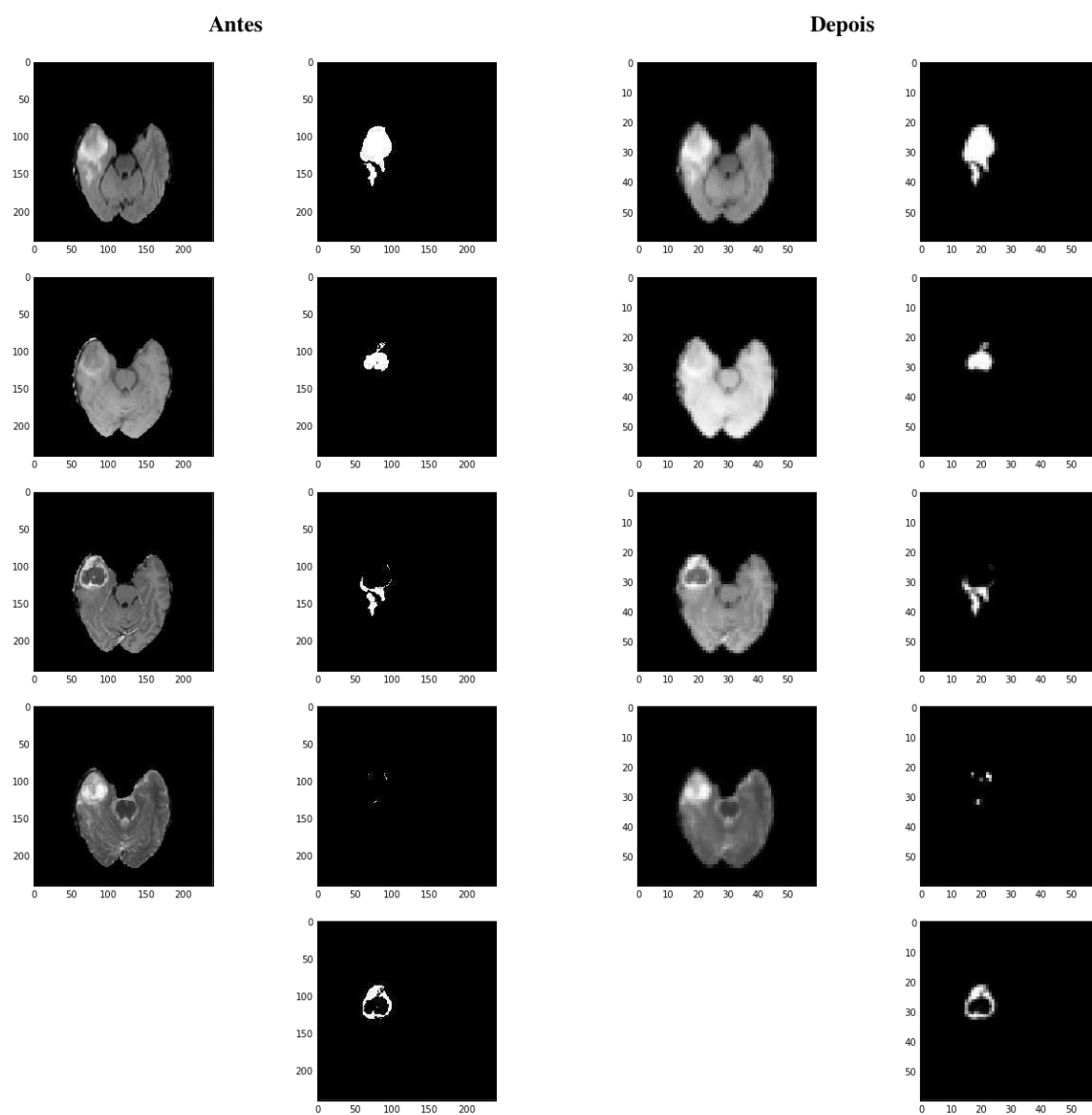
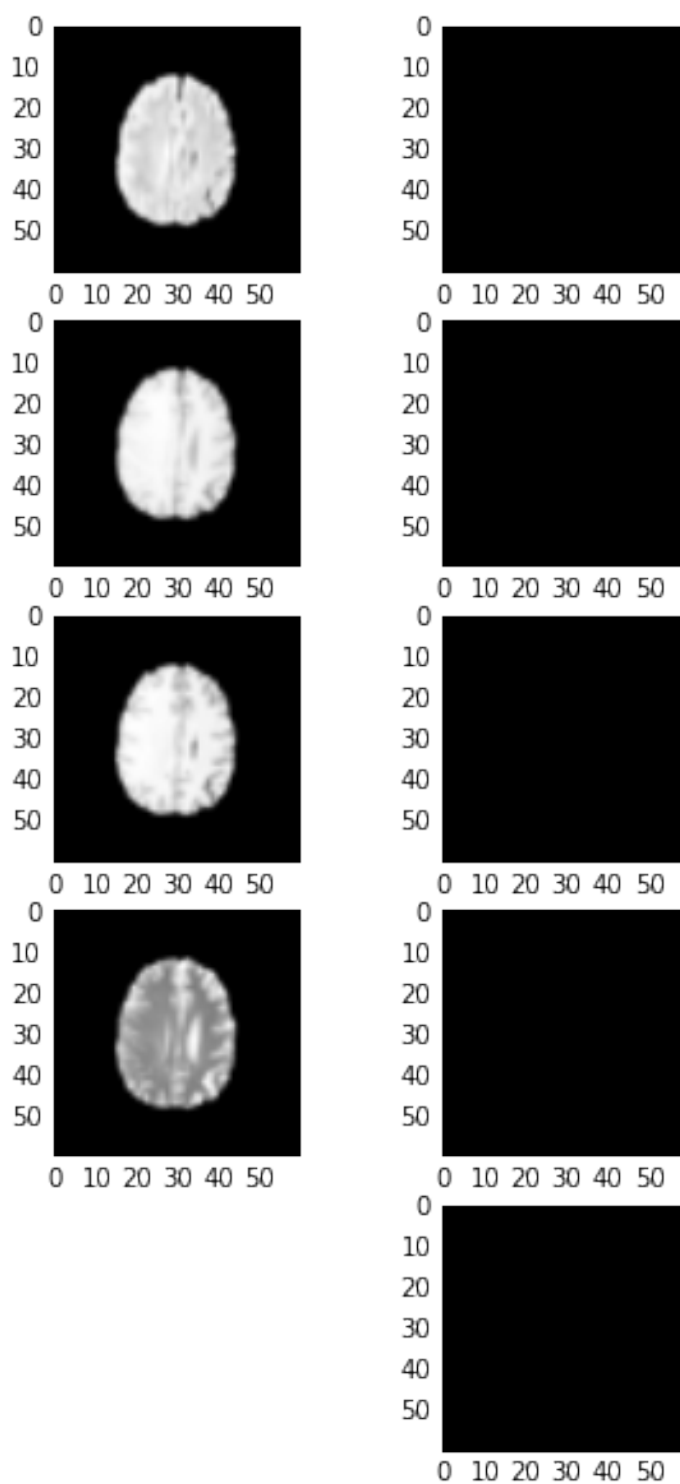
Figura 30 – Exemplo de amostra redimensionada para 60×60 .

Figura 31 – Exemplo de amostra nula.



ANEXO C – Exemplos pós-treinamento (conjunto de teste): Arquitetura A

Figura 32 – Exemplo 1 de resultado de teste da arquitetura A

Dice = 58.3561466584 %

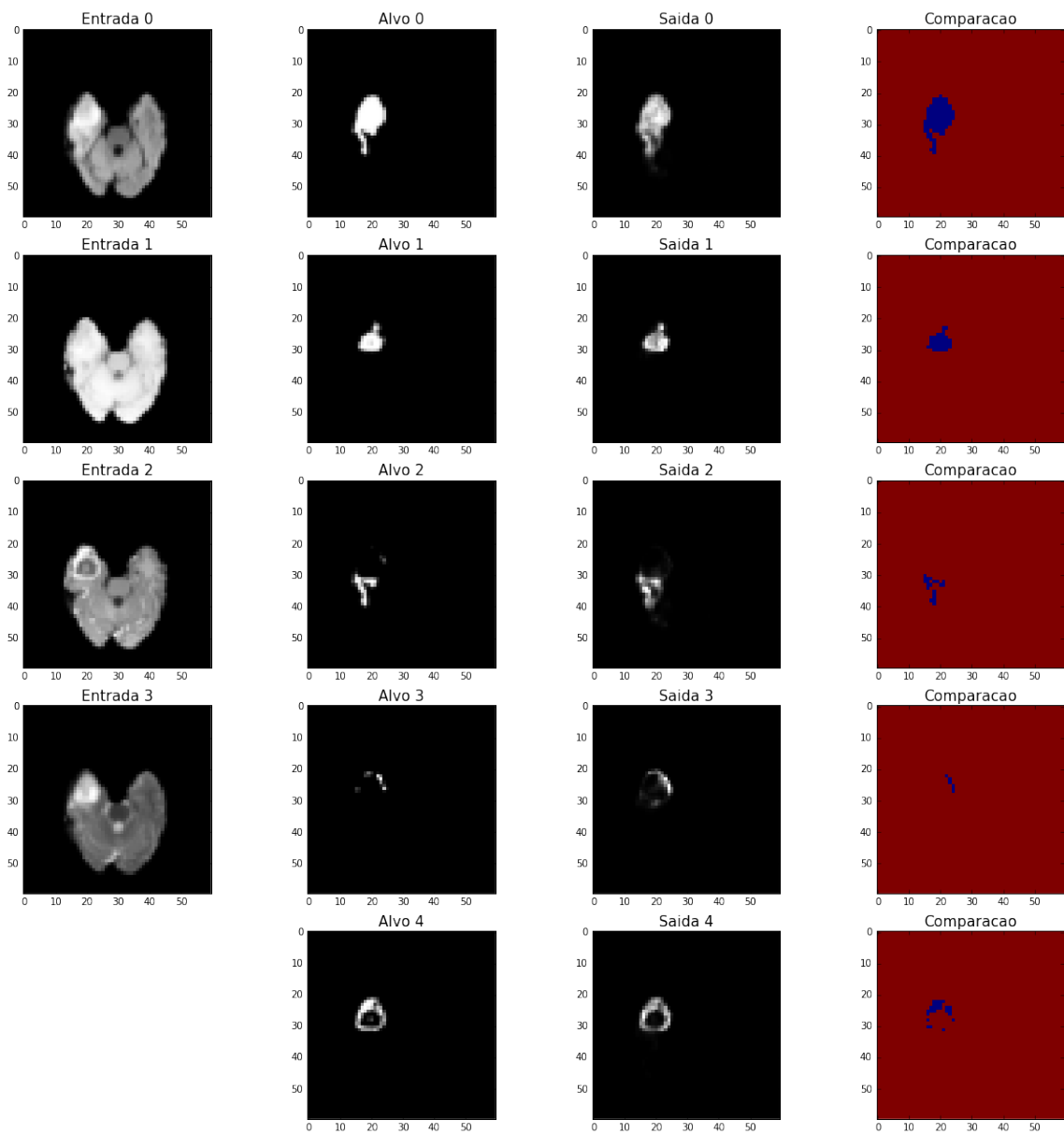


Figura 33 – Exemplo 2 de resultado de teste da arquitetura A

Dice = 68.5839447686 %

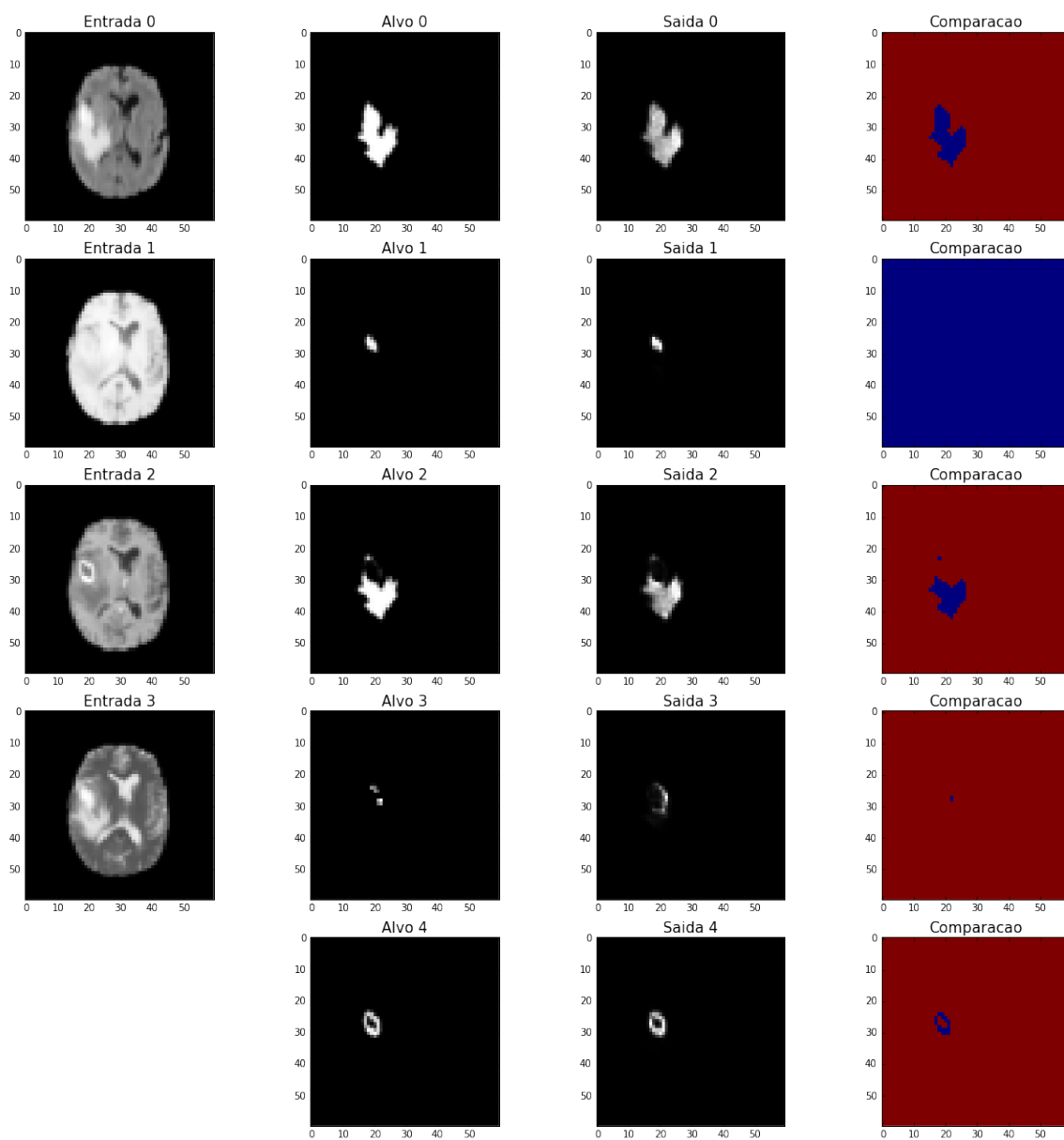


Figura 34 – Exemplo 3 de resultado de teste da arquitetura A

Dice = 45.9654025635 %

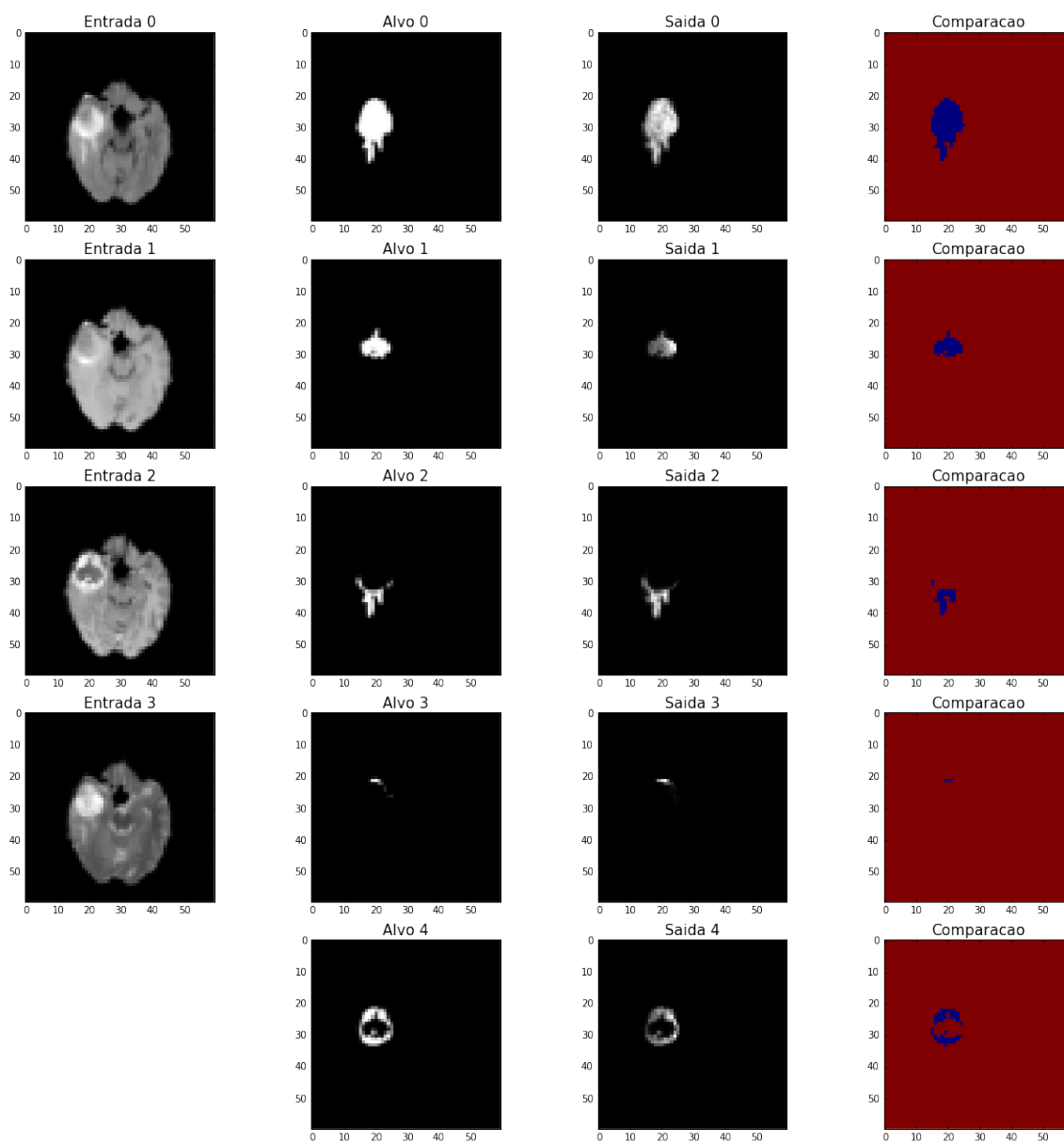


Figura 35 – Exemplo 4 de resultado de teste da arquitetura A

Dice = 21.5042444459 %

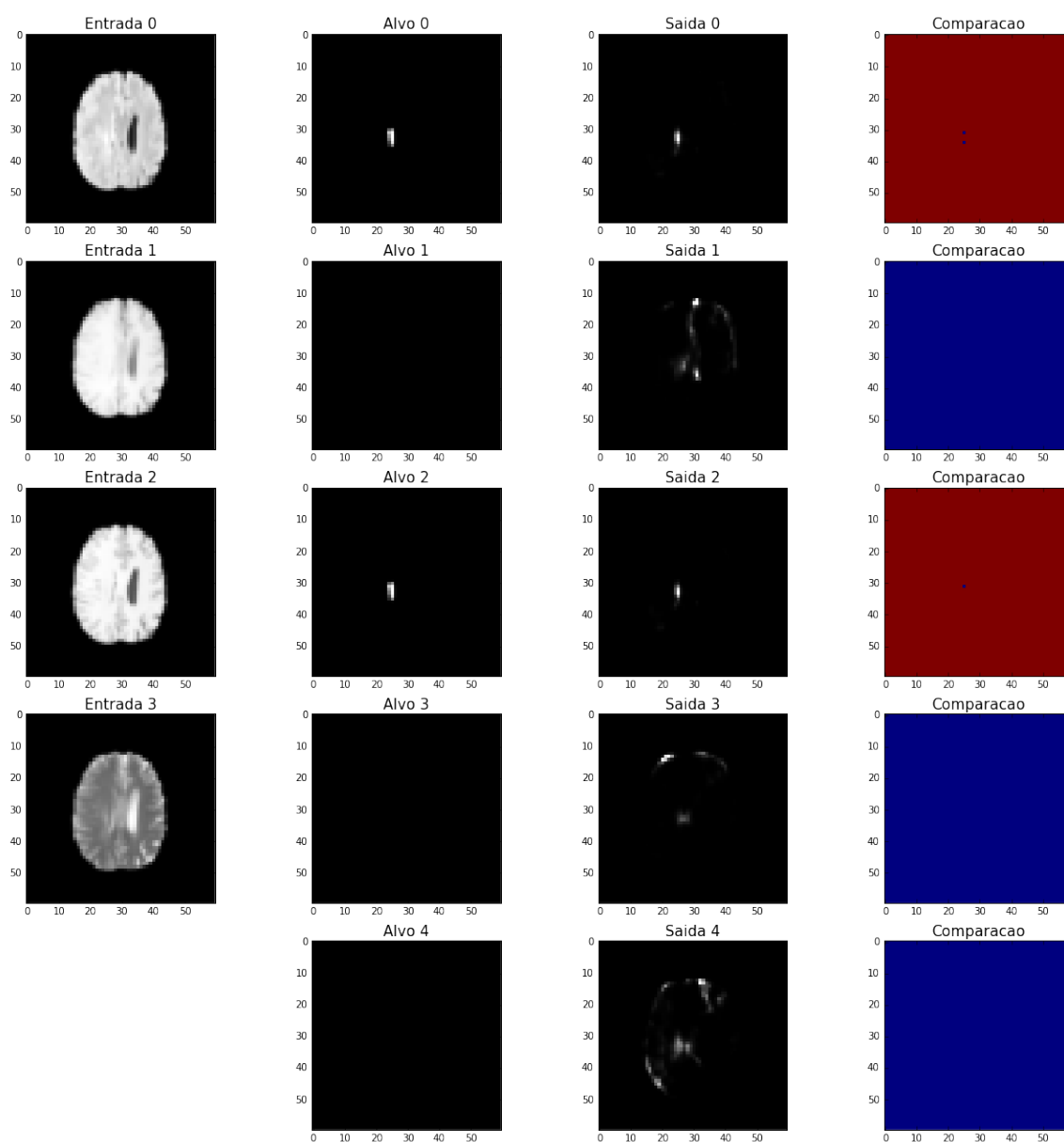
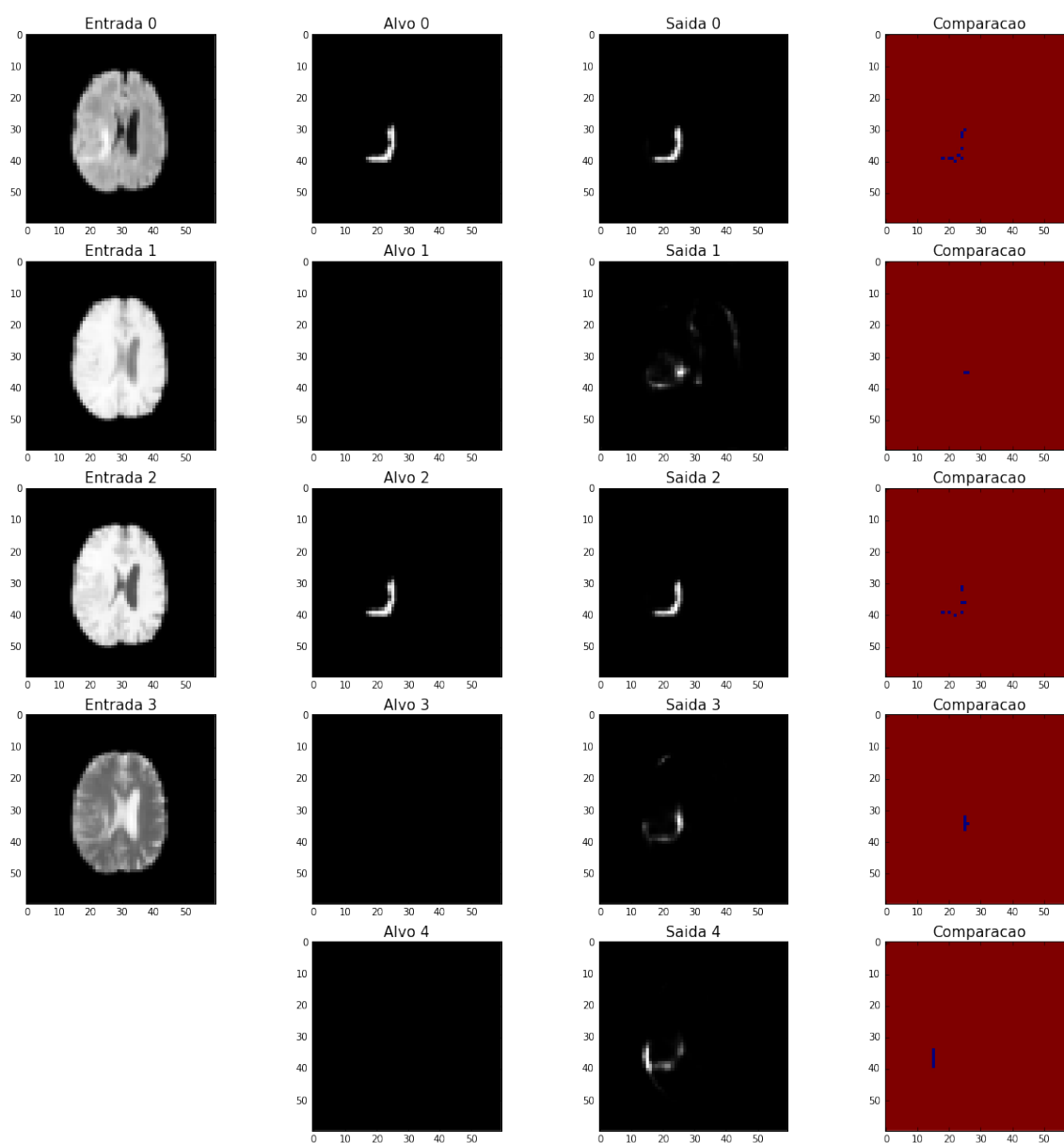


Figura 36 – Exemplo 5 de resultado de teste da arquitetura A

Dice = 33.3838420031 %



ANEXO D – Exemplos pós-treinamento: Arquitetura B

Figura 37 – Exemplo 1 de resultado de teste da arquitetura B

Dice = 92.6933742002 %

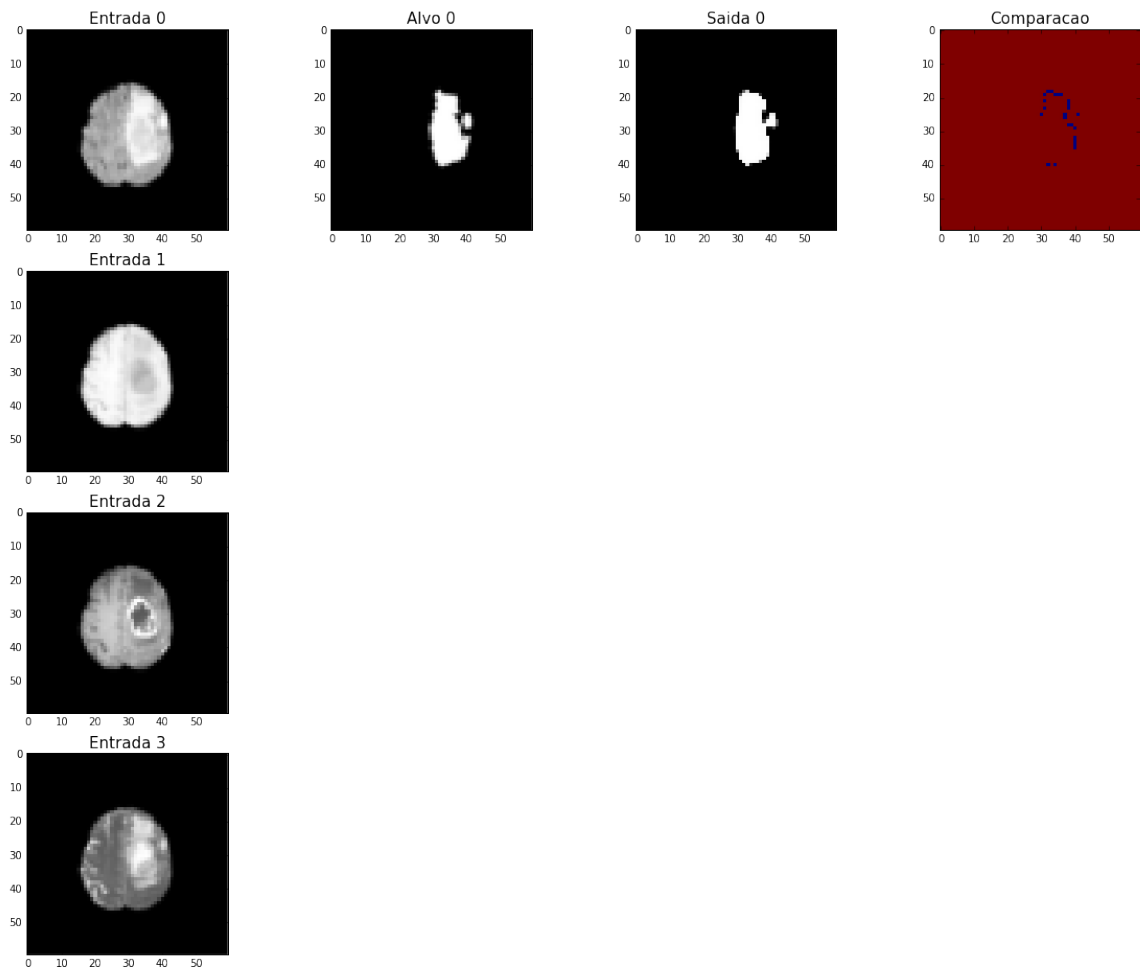


Figura 38 – Exemplo 2 de resultado de teste da arquitetura B

Dice = 94.5647605267 %

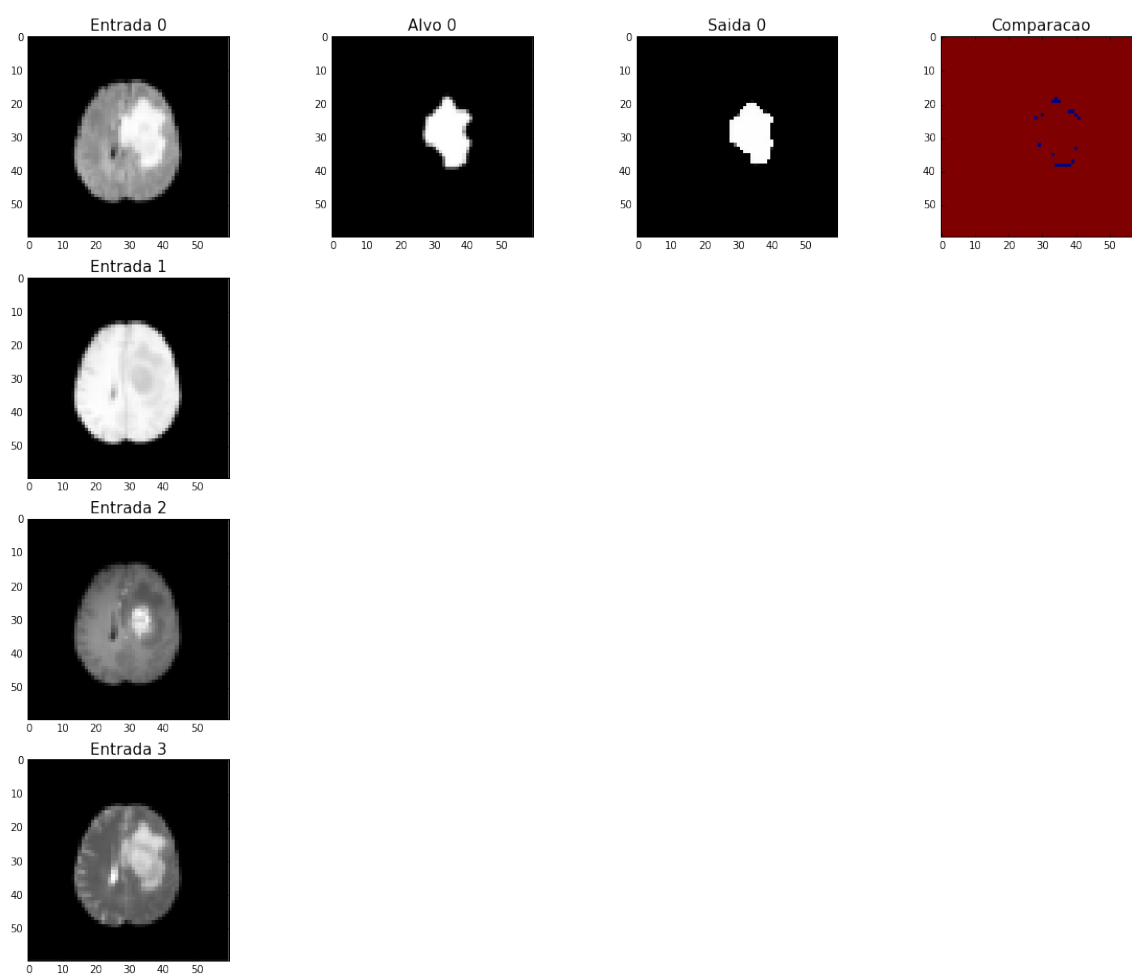


Figura 39 – Exemplo 3 de resultado de teste da arquitetura B

Dice = 90.2491822915 %

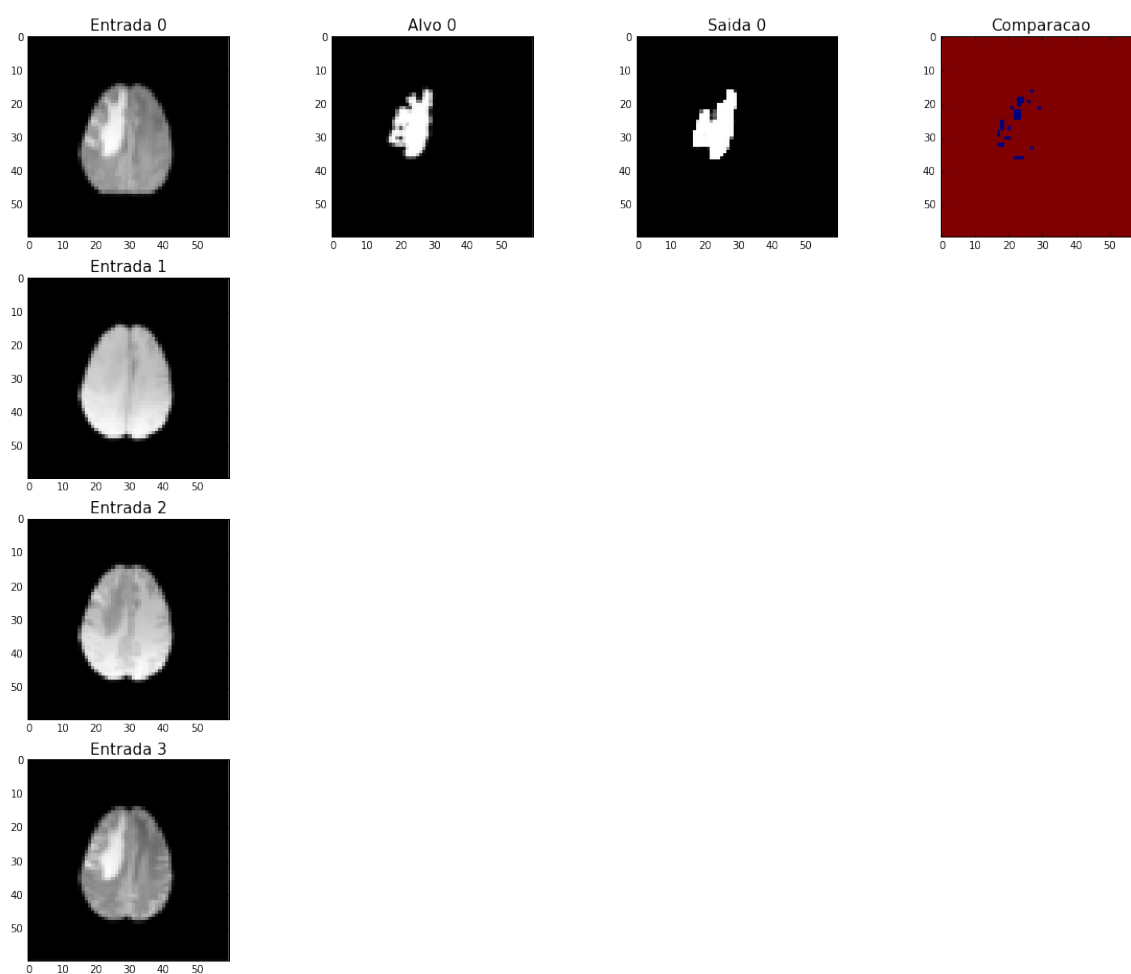


Figura 40 – Exemplo 4 de resultado de teste da arquitetura B

Dice = 58.7476878449 %

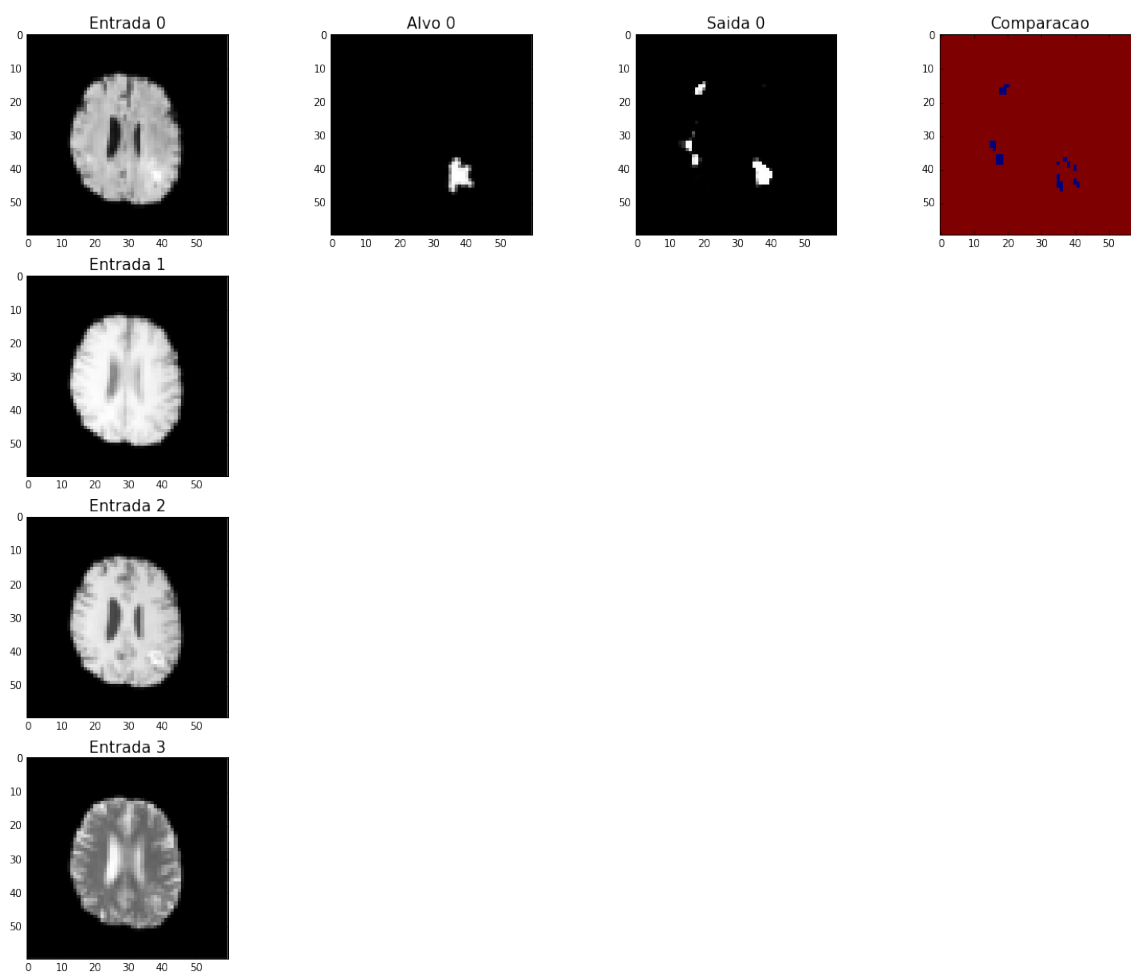
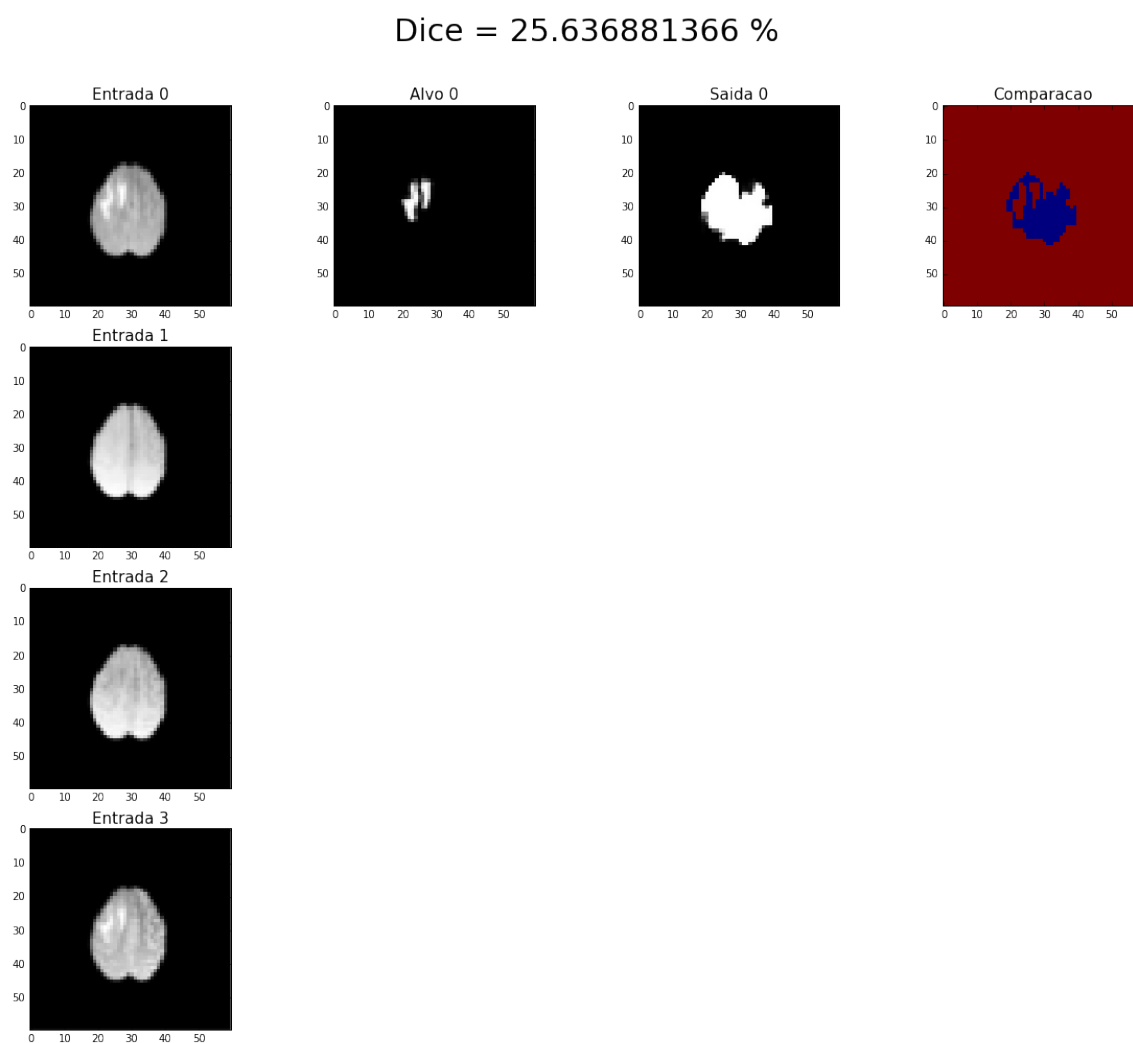


Figura 41 – Exemplo 5 de resultado de teste da arquitetura B



ANEXO E – Códigos

E.1 Pipeline de Pré-processamento

```

1 import SimpleITK as sitk
2 from numpy import *
3 from pylab import *
4 %matplotlib inline
5 import glob
6 from scipy import misc
7 from tqdm import tqdm
8
9 pasta = '/Users/luisoutomaior/Downloads/BRATS2015_Training'
10
11
12 tags = glob.glob(pasta + '/*/*/*mor*/*mor*')
13
14 num_pacientes = int(len(tags) * 1)
15
16 print num_pacientes
17
18 def pipeline(primeiro=0, ultimo = num_pacientes):
19     pasta = '/Users/luisoutomaior/Downloads/BRATS2015_Training'
20
21     tags = glob.glob(pasta + '/*/*/*mor*/*mor*')
22
23     num_pacientes = int(len(tags) * 1.0)
24
25     flair = []
26     t1 = []
27     t1c = []
28     t2 = []
29     data_paths = []
30
31     for n in range(num_pacientes):
32         flair.append(''.join(glob.glob(tags[n][:61] + '*Flair*/*Flair*')))
33         t1.append(''.join(glob.glob(tags[n][:61] + '*T1.*/*T1.*')))
34         t1c.append(''.join(glob.glob(tags[n][:61] + '*T1c*/*T1c*')))
35         t2.append(''.join(glob.glob(tags[n][:61] + '*T2*/*T2*')))
36
37     data_paths = zip(tags,flair,t1,t1c,t2)
38
39     num_pacientes = ultimo;
40     data = []
41     x_ = []

```

```

42     y_ = []
43     tipo = []
44     for z in (range(primeiro,ultimo)):
45
46         img1 = sitk.ReadImage(tags[z],sitk.sitkInt16)
47
48         img2 = sitk.ReadImage(flair[z],sitk.sitkInt16)
49         img3 = sitk.ReadImage(t1[z],sitk.sitkInt16)
50         img4 = sitk.ReadImage(t1c[z],sitk.sitkInt16)
51         img5 = sitk.ReadImage(t2[z],sitk.sitkInt16)
52
53
54
55         slice1 = sitk.GetArrayFromImage(img1)
56         slice2 = sitk.GetArrayFromImage(img2)
57         slice3 = sitk.GetArrayFromImage(img3)
58         slice4 = sitk.GetArrayFromImage(img4)
59         slice5 = sitk.GetArrayFromImage(img5)
60
61         x_.append(asarray([slice2, slice3, slice4, slice5]))
62         y_.append(asarray([slice1 == 0,slice1 == 1,slice1 == 2,slice1 == 3,slice1 ==
        ↪ 4]))
63         benigno_0_maligno_1 = [1]*slice1.shape[0] if flair[1][50:53] == "HGG" else
        ↪ [0]*slice1.shape[0]
64         tipo.append(benigno_0_maligno_1)
65
66     resx = slice1.shape[0]
67     resy = slice1.shape[1]
68     resz = slice1.shape[2]
69
70     x = asarray(x_)
71     y = asarray(y_)
72     tipo = asarray(tipo)
73
74
75     num_y = 5
76
77     x = reshape(transpose(asarray(x_), (0,2,3,4,1)), (-1,resy,resz,4))
78     y = reshape(transpose(asarray(y_), (0,2,3,4,1)), (-1,resy,resz,num_y))
79     tipo = squeeze(reshape(tipo, (x.shape[0],-1)))
80
81     del x_,y_
82
83
84     dwnsmp = .25;
85
86     img_size = int(resy * dwnsmp)
87     x_train_ = np.empty((resx*num_pacientes,img_size,img_size,4))
88     y_train_ = np.empty((resx*num_pacientes,img_size,img_size,num_y))

```

```
89
90     for n in (range(len(x))):
91         for k in range(4):
92             x_train_[n,:,:k] = misc.imresize(x[n, :, :, k], dwnsmp)
93
94         for k in range(5):
95             y_train_[n,:,:k] = misc.imresize(y[n, :, :, k], dwnsmp)
96
97
98     j = 0
99     ion()
100
101     del x, y
102
103
104     x_train__ = transpose(asarray(x_train_), (0,3,1,2))
105     y_train__ = transpose(asarray(y_train_), (0,3,1,2))
106
107     numero = randint(len(x_train__))
108
109     y_train__[:,0] = (y_train__[:,0])*-1 + 255.0
110
111     mask = (y_train__[:] == 0).all(1).all(1).any(1)
112
113     for n in range(len(mask)):
114         if mask[n] == False:
115             if rand(1) < 0:
116                 mask[n] = True
117
118     y_train = y_train__[mask][:]
119     x_train = x_train__[mask][:]
120     tipo = tipo[mask]
121
122
123     def shuffnorm(x,y):
124         from numpy import random
125         ind = arange(len(x))
126         random.shuffle(ind)
127         x_ = x[ind]
128         y_ = y[ind]
129         return (x_)/255.0, (y_)/255.0
130
131     #x, y = shuffnorm(x_train[:,y_train[:]])
132     #x = []
133     #y = []
134     x = x_train[:]/255.0
135     y = y_train[:]/255.0
136     print x.shape
137     print y.shape
```

```
138     print tipo.shape
139     return x, y, tipo
```

```
46     model.add(BatchNormalization(mode=2))
47     model.add(Activation('relu'))
48
49     model.add(Convolution2D(filt_num*4, filt_size, filt_size,
50                             border_mode='same',
51                             input_shape=(x.shape[1:])))
52     model.add(BatchNormalization())
53     model.add(Activation('relu'))
54
55     model.add(Convolution2D(filt_num*8, filt_size, filt_size,
56                             border_mode='same',
57                             input_shape=(x.shape[1:])))
58     model.add(BatchNormalization())
59     model.add(Activation('relu'))
60
61     if multiclasse:
62         tamanho_saida = 5
63     else:
64         tamanho_saida = 1
65     model.add(Convolution2D(tamanho_saida, 3, 3, border_mode='same'))
66     model.add(Reshape((tamanho_saida, y.shape[2]**2)))
67     model.add(Activation('softmax'))
68     model.add(Reshape((tamanho_saida, y.shape[2], y.shape[2])))
69
70     #print model.layers[-1].output_shape
71
72     if func_perda == 0:
73         ## Usando Dice Score
74         model.compile(optimizer='adadelta',
75                       loss= [dice_coef_loss],
76                       metrics = [dice_coef])
77
78         ## Usando Crossentropy
79
80     if func_perda == 1:
81         model.compile(optimizer='adadelta',
82                       loss= 'binary_crossentropy',
83                       metrics = [dice_coef])
84
85         ## Usando MSE
86
87     if func_perda == 2:
88         model.compile(optimizer='adadelta',
89                       loss= 'mse',
90                       metrics = [dice_coef])
91     model.summary()
92     return model
93
```

E.3 Treinamento da Rede Neural

```
1 pasta = '/Users/luisoutomaior/Downloads/BRATS2015_Training'
2
3
4 tags = glob.glob(pasta + '/*/*/*mor*/*mor*')
5
6 num_pacientes = int(len(tags) * 1)
7
8 print num_pacientes
9
10 fim = int(len(tags))*13/13 # 273
11
12 batches = range(0,fim,13)
13
14
15 #modelo_dice = define_arquitetura(1,1)
16 hist_dice = []
17 hist_dice_ = []
18 num_epocas = 100
19 for n in range(num_epocas):
20     print 'Epoca ', n+1, 'de', num_epocas, '#####'
21     for atual in batches:
22         print 'Batch ', 1+atual/13, 'de', 1+batches[-1]/13
23         x, y, tipo = pipeline(atual,atual+13)
24
25         parcela = int(len(x)*.1)
26
27         x_train = x[:parcela]
28         y_train = y[:parcela]
29         tipo_train = tipo[:parcela]
30
31         x_val = x[parcela:]
32         y_val = y[parcela:]
33         tipo_val = tipo[parcela:]
34         hist_num = int(atual/13)
35         hist_dice.append(modelo_dice.fit(x_train,
36                                         y_train,
37                                         validation_data = [x_val, y_val],
38                                         nb_epoch = 1,
39                                         batch_size = 1,
40                                         verbose = 1) )
41         visualiza_resultados(3, modelo_cross,x_val,y_val)
42     hist_dice_.append(hist_dice)
```

E.4 Visualização dos Resultados

```

1  #def dice_coeffff(y_true, y_pred):
2  #
3  #   y_true_f = reshape(y_true, (-1))
4  #   y_pred_f = reshape((y_pred)/amax((y_pred)), (-1))
5  #   intersection = sum(y_true_f * y_pred_f)
6  #   return (2. * intersection) / (sum(y_true_f) + sum(y_pred_f))
7
8  def dice_coeffff(y_true, y_pred):
9      dice = 0
10     for n in range(y_true.shape[1]):
11         y_true_f = reshape(y_true[:,n], (-1))
12         y_pred_f = reshape((y_pred[:,n])/amax((y_pred[:,n])), (-1))
13         intersection = sum(matrix.round(y_true_f) * matrix.round(y_pred_f))
14         dice += ((2. * intersection) / (sum(y_true_f) + sum(y_pred_f)))
15     dice = dice/(n+1)
16     return (dice)
17
18 def visualiza_resultados(num = 3, modelo = [], x_val = [], y_val=[]):
19
20     plt.rcParams['figure.figsize'] = (20,20)
21
22     for n in range(num):
23         numero = randint(len(x_val))
24         #numero = 564
25         #print numero
26         fig3 = plt.figure()
27         axis('off')
28
29         entrada = expand_dims(x_val[numero], axis=0)
30         saida = modelo.predict(entrada)
31         target = expand_dims(y_val[numero], axis=0)
32
33         #print amax(entrada)
34         #print amax(saida)
35         #print amax(target)
36         texto = 'Dice = {0} %'.format(dice_coeffff(target,saida)*100)
37         saida = saida/amax(saida)
38         plt.title(texto,fontsize = 30,y=1.05)
39         plt.suptitle('')
40
41
42         for n in range(entrada.shape[1]):
43             ax = fig3.add_subplot(5,4,4*n+1)
44             imshow(entrada[0,n], cmap=cm.Greys_r, interpolation='none')
45             ax.set_title('Entrada {0}'.format(n), fontsize = 15)
46
47
48         for n in range(saida.shape[1]):
49
50             ax = fig3.add_subplot(5,4,4*n+2)
51             imshow(target[0,n], cmap=cm.Greys_r, interpolation='none')
52             ax.set_title('Alvo {0}'.format(n), fontsize = 15)
53             ax = fig3.add_subplot(5,4,4*n+3)
54             imshow(saida[0,n], cmap=cm.Greys_r, interpolation='none')
55             ax.set_title('Saida {0}'.format(n), fontsize = 15)
56             ax = fig3.add_subplot(5,4,4*n+4)

```

```
57         imshow(matrix.round(target[0,n])==matrix.round(saida[0,n]),
58                interpolation='none')
59         ax.set_title('Comparacao', fontsize = 15)
60
61
62         show()
63
64     visualiza_resultados(10, modelo_cross,x_val,y_val)
```