

**Daniel Weil**

**Uso da Unet para Segmentação de  
Lesões na Substância Branca do  
Cérebro**

**PROJETO FINAL**

**DEPARTAMENTO DE INFORMÁTICA**  
**Programa de Graduação em Engenharia da**  
**Computação**

Rio de Janeiro  
Junho de 2021



**Daniel Weil**

**Uso da Unet para Segmentação de Lesões na  
Substância Branca do Cérebro**

**Relatório de Projeto Final**

Relatório de Projeto Final, apresentado ao Programa de Engenharia da Computação, do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador : Prof. Marcelo Gattass  
Co-orientador: Luiz Fernando Trindade Santos

Rio de Janeiro  
Junho de 2021

Todos os direitos reservados. A reprodução, total ou parcial do trabalho, é proibida sem a autorização da universidade, do autor e do orientador.

**Daniel Weil**

Ficha Catalográfica

Weil,Daniel

Uso da Unet para Segmentação de Lesões na Substância Branca do Cérebro / Daniel Weil; orientador: Marcelo Gattass; co-orientador: Luiz Fernando Trindade Santos. – 2021.

57 f: il. color. ; 30 cm

Projeto Final - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Informática – Teses. 2. Lesões na Substância Branca do Cérebro. 3. Rede Neural Convolucional. 4. U-NET. 5. Apoio a Detecção Apoiado por Computador. I. Gattass, Marcelo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

## **Agradecimentos**

Dedico este Projeto de Conclusão de Curso para o meu orientador, Professor Marcelo Gattass, por ter me aceito e confiado em mim para ser seu orientando, bem como o co-orientador Luiz Fernando Trindade Santos, por me instruir e sempre se mostrar disponível para tirar quaisquer dúvidas e orientar-me da melhor forma.

Para minha família, por me fornecer os subsídios necessários para obtenção de uma ótima formação universitária, tal como a instituição de excelência que é a Pontifícia Universidade Católica do Rio de Janeiro.

Aos meus amigos que estiveram comigo desde o início do curso e me proporcionaram inúmeros momentos de felicidade ao longo dos últimos anos.

À minha namorada, por me fazer companhia ao longo dos anos e compartilhar vários momentos de alegria.

## **Resumo**

Weil,Daniel; Gattass, Marcelo; . **Uso da Unet para Segmentação de Lesões na Substância Branca do Cérebro.** Rio de Janeiro, 2021. 57p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O cérebro humano ainda é um grande mistério, porém muito já se sabe sobre como lesões na substância branca do cérebro são responsáveis por doenças desmielinizantes, tal como a Esclerose Múltipla e a Gliose. A detecção dessas lesões por meio da análise manual de exames de Ressonância Magnética pode ser uma tarefa exaustiva e demorada. Neste trabalho é implementado um processo de detecção auxiliado por computador que se utiliza de uma Rede Neural Convolucional chamada U-NET, reduzindo o número de exames que necessitam de análise manual dos especialistas médicos.

## **Palavras-chave**

Lesões na Substância Branca do Cérebro; Rede Neural Convolucional; U-NET; Apoio a Detecção Apoiado por Computador.

## **Abstract**

Weil,Daniel; Gattass, Marcelo (Advisor); (Co-Advisor). **Detection of White Matter Lesions in the Brain.** Rio de Janeiro, 2021. 57p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The human brain is still a great mystery, but much is already known about how white matter lesions in the brain are responsible for demyelinating diseases such as Multiple Sclerosis and Gliosis. Detecting these injuries through manual analysis of MRI scans can be an exhausting and time-consuming task. In this work, a computer-aided detection process is implemented using a U-NET Convolutional Neural Network to support this task, reducing the number of tests that require manual analysis by medical specialists.

## **Keywords**

White Matter Lesions in the Brain; Convolucional Neural Network; U-NET; Computer Aided Detection.

# **Sumário**

|          |                                  |           |
|----------|----------------------------------|-----------|
| <b>1</b> | <b>Introdução</b>                | <b>13</b> |
| <b>2</b> | <b>Visão Geral</b>               | <b>15</b> |
| <b>3</b> | <b>Pré-processamento</b>         | <b>17</b> |
| 3.1      | Extração do Crânio               | 17        |
| 3.2      | Correspondência de Histogramas   | 18        |
| <b>4</b> | <b>Rede Neural Artificial</b>    | <b>20</b> |
| 4.1      | Perceptron Multicamadas          | 21        |
| 4.2      | Overfitting                      | 23        |
| 4.3      | Rede Neural Convolucional        | 23        |
| 4.3.1    | Camada de Convolução             | 24        |
| 4.3.2    | Camada de <i>Pooling</i>         | 25        |
| 4.3.3    | Camada Totalmente Conectada      | 26        |
| 4.4      | Arquitetura U-NET                | 26        |
| 4.5      | Função de Perda                  | 28        |
| 4.6      | Função de Ativação               | 29        |
| <b>5</b> | <b>Preparação das Entradas</b>   | <b>33</b> |
| 5.1      | Imagens do Cérebro               | 33        |
| 5.2      | Máscaras das Lesões              | 34        |
| <b>6</b> | <b>Técnicas de Regularização</b> | <b>36</b> |
| 6.1      | Parada Antecipada                | 36        |
| 6.2      | Data Augmentation                | 36        |
| 6.2.1    | Rotação                          | 37        |
| 6.2.2    | Inversão                         | 38        |
| <b>7</b> | <b>Métricas</b>                  | <b>40</b> |
| 7.1      | Perda                            | 40        |
| 7.2      | Acurácia                         | 40        |
| 7.3      | Recall                           | 41        |
| 7.4      | Precisão                         | 41        |
| 7.5      | Especificidade                   | 42        |
| 7.6      | F1 Score                         | 42        |
| <b>8</b> | <b>Resultados</b>                | <b>43</b> |
| 8.1      | Uso das Metricas                 | 43        |
| 8.2      | Análise de Casos                 | 45        |
| 8.3      | Comparação                       | 47        |
| 8.3.1    | Trabalho Base                    | 47        |
| 8.3.2    | Trabalho Atual                   | 48        |
| <b>9</b> | <b>Conclusões</b>                | <b>50</b> |

|           |                                   |           |
|-----------|-----------------------------------|-----------|
| 9.1       | Trabalhos Futuros                 | 51        |
| <b>10</b> | <b>Referências bibliográficas</b> | <b>53</b> |

## **Lista de figuras**

|  |    |
|--|----|
| Figura 3.1 Imagens referente a uma fatia da base de dados do trabalho antes e depois da segmentação do crânio durante o pré-processamento.   | 18 |
| Figura 3.2 Imagens acerca dos histogramas da fatia utilizada como referência e de uma fatia do cérebro antes e depois da técnica de Correspondência de Histogramas Global. (a) Referência (b) Antes (c) Depois         | 19 |
| Figura 4.1 Cálculo realizado por um neurônio, utilizando a função de ativação 'degrau' (WEISSTEIN, 2002). (YUNG, )   | 21 |
| Figura 4.2 Imagem de uma rede MLP, mostrando a camada de entrada, a camada escondida com a função de ativação e a camada de saída. (AILEPHANT, )   | 21 |
| Figura 4.3 Tabela mostrando o efeito do ' <i>overfitting</i> ' em diversos modelos de redes neurais. (SHRIVASTAVA, )   | 23 |
| Figura 4.4 Imagem mostrando a arquitetura e organização de uma CNN. (CORNELISSE, )   | 24 |
| Figura 4.5 Imagem mostrando a operação de convolução. (CONVOLUTIONAL..., )   | 25 |
| Figura 4.6 Imagem mostrando a operação de Pooling, tanto na modalidade de média como na modalidade de máximo. (POKHREL, )  | 26 |
| Figura 4.7 Imagem mostrando a arquitetura U-NET. Na metade esquerda pode-se visualizar as camadas pertencentes ao codificador, e na metade direita, é possível ver as camadas pertencentes ao decodificador. (LAPIX, ) | 27 |
| Figura 4.8 Também conhecida como função rampa, é análoga a retificação de meia onda para os conceitos de engenharia elétrica. (BECKER, )   | 30 |
| Figura 5.1 Imagens do cérebro divididos em segmentos em menores, também chamados de 'Patches'.   | 34 |
| Figura 5.2 Imagens de um segmento do cérebro e sua respectiva máscara da lesão. (a) Tecido do cérebro (b) Máscara da lesão (c) Sobreposição da máscara com o tecido do cérebro   | 35 |
| Figura 6.1 Imagem mostrando o ponto ótimo de término do treinamento de uma rede neural, com objetivo de evitar que ocorra o ' <i>overfitting</i> '. (AP, )   | 36 |
| Figura 6.2 Segmentos do cérebro e sua máscara, antes e depois da transformação de rotação.   | 38 |
| Figura 6.3 Segmentos do cérebro e sua máscara, antes e depois da transformação de inversão horizontal.   | 38 |
| Figura 6.4 Segmentos do cérebro e sua máscara, antes e depois da transformação de inversão vertical.   | 39 |
| Figura 8.1 Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão   | 46 |
| Figura 8.2 Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão   | 46 |

|   |    |
|---|----|
| Figura 8.3    Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão | 47 |
| Figura 8.4    Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão | 47 |

## **Lista de tabelas**

|            |  |    |
|------------|--|----|
| Tabela 8.1 | Valor das Métricas sobre a Base de Treinamento no Fim do Aprendizado                                   | 43 |
| Tabela 8.2 | Valor das Métricas sobre a Base de Validação no Fim do Treinamento                                     | 44 |
| Tabela 8.3 | Valor das Métricas ao Fim do Aprendizado utilizando a Base de Treinamento Misturada com a de Validação | 44 |
| Tabela 8.4 | Valor das Métricas sobre a Base de Teste   | 45 |
| Tabela 8.5 | Resultado do método da tese de Doutorado com o melhor resultado na base de teste                       | 48 |

## **Lista de Códigos**

Código 1      Trecho de Arquivo de Texto de Marcações      34

## **Lista de Abreviaturas**

CAD – Computer Aided Detection

CNN – Convolutional Neural Network

ITK – Insight Toolkit

DICOM – Digital Imaging and Communications in Medicine

NIFTI – Neuroimaging Informatics Technology Initiative

ANN – Artifical Neural Network

MLP – Multilayer Perceptron

GAN – Generative Adversarial Network

# 1

## Introdução

No ano de 1965, o engenheiro norte-americano Gordon Moore escreveu um artigo sobre um estudo que previa o aumento de 100% na quantidade existente de transistores nos chips dentro de um computador, pelo mesmo preço, a cada 18 meses (BRITANNICA et al., ). Com a crescente evolução na capacidade de processamento dos computadores, estes equipamentos estão sendo utilizados nas mais diversas áreas de atuação. Em síntese, podemos dizer que o impacto causado por esse novo tipo de tecnologia está permitindo com que outras áreas de pesquisa consigam crescer de maneira mais rápida, sem dispor de tantos recursos. Em virtude disto, todo o conhecimento tecnológico do ser humano vem crescendo exponencialmente.

A indústria médico-farmacêutica é uma das grandes beneficiadas por este crescente tecnológico advindo do uso de computadores (MEHTA; DEB; SUBBA, 1994). Suas aplicações vão de armazenamento e compartilhamento de dados médicos até o processamento de exames radiográficos e a utilização dos resultados para o auxílio a detecção e ao diagnóstico (SHIRAI SHI et al., 2011) (HALALLI; MAKANDAR, 2018).

Um tipo de aplicação oriunda dessa expansão tecnológica é o processo de detecção auxiliada por computador (sigla CAD do inglês '*Computer Aided Detection*') (CASTELLINO, 2005). Essas ferramentas auxiliam os profissionais da saúde na interpretação de dados médicos, que precisam analisar uma vasta quantidade de exames em um curto período de tempo. As ferramentas CAD processam as imagens digitais dos exames, e após terem sido expostos a uma grande base de dados, conseguem encontrar e destacar padrões desejados nos exames submetidos, caso eles existam.

Esse processamento auxilia os médicos a reduzir o espectro de análise necessário sobre os exames, para que eles, consequentemente, analisem o

padrão destacado e consigam confirmar ou descartar o problema encontrado.

Para a produção desse projeto, foram utilizados a linguagem de programação *Python* (ROSSUM; JR, 1995), em combinação com diversas bibliotecas, sendo as principais *Numpy* (HARRIS et al., 2020), *Tensorflow* (ABADI et al., 2015), *Keras* (CHOLLET et al., 2015) e *Matplotlib* (HUNTER, 2007). Além disso, foi utilizado o ambiente computacional *Jupyter Notebook* (KLUYVER et al., 2016) por meio do produto *Google Colaboratory* (BISONG, 2019), que permite acesso gratuito a recursos de processamento gráfico, fundamentais para o desenvolvimento.

O objetivo desse trabalho computacional é implementar um processo CAD que utilize imagens médicas de exames de ressonância magnética para detectar lesões na substância branca do cérebro, responsáveis por causar déficits funcionais significativos aos seres humanos, e são observadas conjuntamente a diversos transtornos neurológicos e psiquiátricos.

Este trabalho será dividido em oito capítulos. O capítulo II fornece uma visão geral sobre os capítulos subsequentes. O capítulo III trata sobre as técnicas de pré-processamento e porque são fundamentais para o desenvolvimento do trabalho. O capítulo IV trata sobre os diversos tipos diferentes de Redes Neurais e o modelo escolhido para o trabalho. O capítulo V entra em detalhe sobre o formato esperado para as entradas do modelo. O capítulo VI trata sobre técnicas de regularização. O capítulo VII explora em detalhes sobre os resultados adquiridos nesse trabalho. Por fim, o capítulo VIII apresenta a conclusão do trabalho e possibilidades de trabalhos futuros.

## 2

### Visão Geral

A implementação do processo CAD desenvolvido nesse trabalho é um modelo inteligente de segmentação. O objetivo dele é conseguir destacar as lesões nos exames médicos a partir do treinamento com uma base de dados constituída com arquivos de imagens do cérebro, e arquivos com o mapa das lesões para esses cérebros, também chamadas de máscaras (SYDORENKO, 2020).

Inicialmente, é necessário fazer o tratamento da base de dados utilizada pelo modelo a partir de procedimentos que adequam os dados para evidenciar os atributos que tem um impacto maior no resultado, e também reduzir possíveis ruídos que tem menor importância para a segmentação. Essa etapa é nomeada de pré-processamento (PANDEY, 2020), e é muito utilizada em modelos de aprendizado de máquina (MITCHELL et al., 1997).

A base de dados do trabalho disponível para treinamento é relativamente pequena se comparada a outras áreas, por possuir apenas 47 exames com as respectivas marcações das lesões. O modelo apresentado neste trabalho consiste de uma Rede Neural Convolucional (CNN do inglês '*Convolutional Neural Network*') (ALBAWI; MOHAMMED; AL-ZAWI, 2017), que tem sua performance diretamente relacionada com a quantidade de dados disponíveis para o treinamento (CHO et al., 2016). Para melhorar o resultado do modelo, e também evitar o que é chamado de '*overfitting*', serão realizadas técnicas chamadas de regularização (YING, 2019).

Um modelo de aprendizado de máquina é constituído de múltiplos componentes, e a segunda metade do trabalho é fazer a melhor escolha dessa arquitetura e a configuração dos parâmetros para resolver o problema em questão. Será utilizado nessa etapa uma arquitetura de CNNs chamada de U-NET (RONNEBERGER; FISCHER; BROX, 2015), especializada em

segmentação de imagens biomédicas.

Depois de ter feito um tratamento inicial na base de dados, e escolhido a configuração inicial para a rede, é necessário separar a base de dados em conjuntos distintos que serão utilizados ao longo do processo, que são os conjuntos de treinamento, validação e teste. Além disso, será necessário reorganizar as informações da base no formato desejado pelo modelo, dividindo as imagens do cérebro em imagens menores. E ainda, será necessário criar os arquivos contendo as máscaras das lesões a partir de arquivos de texto contendo suas coordenadas.

Com a base de dados adequada ao uso do modelo, é realizado o treinamento do modelo com o primeiro conjunto dividido da base, utilizando a base de validação durante o treinamento para melhorar a performance. Depois de ter a rede treinada, é possível fazer a previsão das lesões utilizando como entrada o conjunto de teste. Com isso, podemos comparar a previsão dada pelo modelo com os arquivos de máscara contendo a posição real das lesões.

Utilizam-se métricas distintas para classificar a performance do modelo na solução do problema. A partir dessas métricas, analisa-se se há a necessidade de revisar ou realizar alguma alteração nos procedimentos feitos para progressivamente evoluir o resultado final.

### **3**

## **Pré-processamento**

O pré-processamento (PANDEY, 2020) é um conjunto de técnicas que envolvem preparação, organização e estruturação dos dados. Tratam-se de etapas fundamentais para modelos de aprendizado de máquina, que antecedem a realização de análises e predições. Essa etapa é necessária para melhorar a qualidade dos dados utilizados, e por conta disso, impacta diretamente na capacidade do modelo de aprendizagem. Exemplos de pré-processamento podem ser limpezas dos dados que serão utilizados ou transformações das informações em formatos mais apropriados para o processo, porém existe uma ampla variedade de técnicas que podem ser utilizadas.

### **3.1**

#### **Extração do Crânio**

O primeiro processo realizado é a extração do crânio das imagens, de forma a extrair parte dos dados considerados como ruído para o problema em questão, pois as imagens de um exame de ressonância não apresentam apenas tecidos do cérebro.

Nessa etapa foi utilizado o algoritmo de segmentação (BAUER; NOLTE; REYES, 2012), por meio de sua implementação dentro da biblioteca '*Insight Toolkit*' (ITK) (JOHNSON; MCCORMICK; IBANEZ, ), já na linguagem *Python*. O maior ponto positivo do uso dessa implementação foi conseguir fazer a segmentação do cérebro utilizando a linguagem *Python*, sem haver necessidade de utilizar um software a parte. Os arquivos da base de dados que contém as informações e imagens dos cérebros estão em formato DICOM ('*Digital Imaging and Communications in Medicine*') (PIANYKH, 2010), e para utilizar o algoritmo citado acima, é necessário convertê-los para o formato NIFTI ('*Neuroimaging Informatics Technology Initiative*') (LAROBINA; MURINO, 2014) (LI et al., 2016), que possui informação volumétrica em 3D.

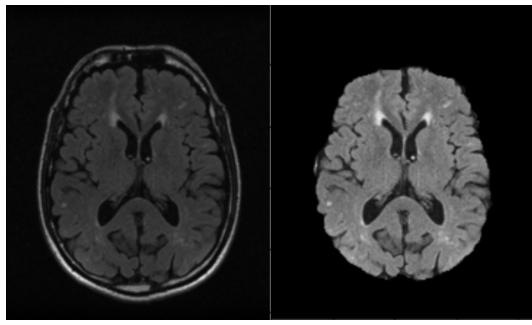


Figura 3.1: Imagens referente a uma fatia da base de dados do trabalho antes e depois da segmentação do crânio durante o pré-processamento.

### 3.2 Correspondência de Histogramas

A base de dados utilizada pode apresentar pequenas diferenças de contraste de um exame para o outro. Com isso, é necessário realizar o procedimento que se chama “Correspondência de Histogramas” (TU; DONG, 2013), para normalizar os diferentes exames entre si.

Um histograma, ou diagrama de dispersão de frequências, consiste em uma representação gráfica de dados que são divididos em classes. Essa visualização nos permite analisar como se comportam os dados em questão. Quando é feito o histograma dos diversos exames de ressonância na base, é possível detectar que o formato do gráfico de um exame pode ser diferente de outro, representando distinção no contraste das imagens.

Dessa forma, é possível ver que a concentração de pixels está diferente nos diversos exames, e isso se dá por conta do uso de múltiplos equipamentos distintos de ressonância magnética utilizados na obtenção da base. Por conseguinte, a Correspondência de Histograma usa um exame de referência para tentar equalizar os demais, tornando os seus histogramas parecidos com a referência.

Com a base equalizada, o modelo tem uma performance melhor em detectar padrões, o que será feito durante a detecção das lesões a partir de suas máscaras.

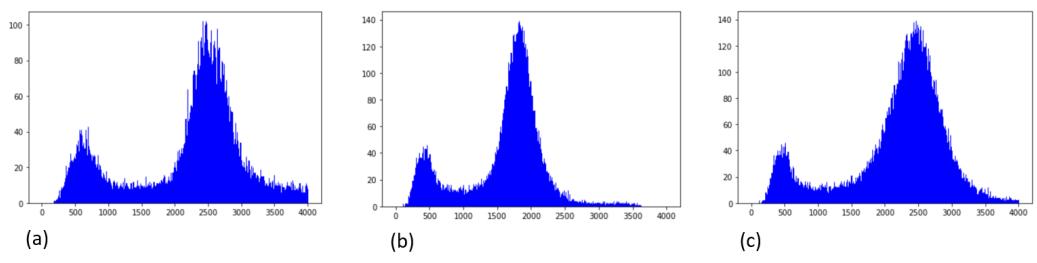


Figura 3.2: Imagens acerca dos histogramas da fatia utilizada como referência e de uma fatia do cérebro antes e depois da técnica de Correspondência de Histogramas Global. (a) Referência (b) Antes (c) Depois

## 4

### Rede Neural Artificial

Uma Rede Neural Artificial (ANN do inglês '*Artifical Neural Network*') (GROSSI; BUSCEMA, 2008) é um modelo computacional inspirado no sistema nervoso central de um animal, que busca simular o comportamento do cérebro e são capazes de realizar o aprendizado de máquina, do inglês '*Machine Learning*' (MITCHELL et al., 1997). Esse termo foi definido por Arthur Samuel em 1959 como o 'campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados' (SAMUEL, 1959).

Na natureza, o cérebro humano é composto em média por cem bilhões de neurônios (HERCULANO-HOUZEL, 2012), que são a unidade básica do sistema nervoso. O neurônio recebe impulsos nervosos por meio de regiões chamadas de sinapses, e os envia para o próximo. Esse impulso nervoso é uma corrente elétrica com o objetivo de transmitir uma informação.

As ANNs também são compostas por unidades básicas que levam o mesmo nome. O neurônio artificial recebe sinais, ou entradas, que possuem pesos atribuídos a elas. Esses termos são somados e passam por uma função não linear chamada de função de ativação (SHARMA; SHARMA, 2017), para produzir uma saída. A saída de um neurônio pode ser definida pela Equação 4-1.

$$y_k = \varphi\left(\sum_{j=0}^m w_{kj}x_j\right) \quad (4-1)$$

Para um determinado neurônio  $k$ , existem  $m + 1$  entradas, com sinais  $x_0$  a  $x_m$ , e pesos  $w_0$  a  $w_m$ , além da função de ativação  $\varphi$ . Usualmente, é atribuído ao sinal  $x_0$  o valor 1, o que o torna um bias, com peso  $w_{k0} = b_k$ . O bias tem a função de transladar a função de ativação no eixo.

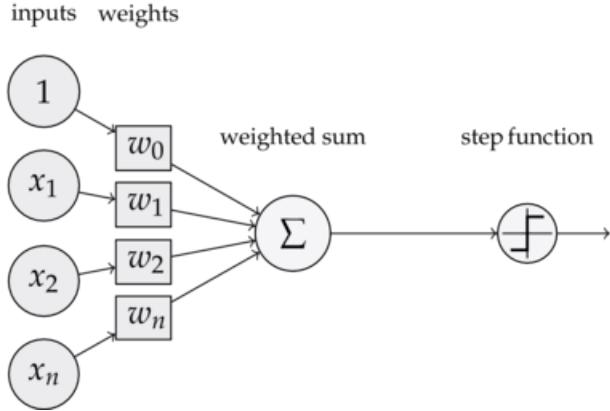


Figura 4.1: Cálculo realizado por um neurônio, utilizando a função de ativação 'degrau' (WEISSTEIN, 2002). (YUNG, )

## 4.1

### Perceptron Multicamadas

A Perceptron Multicamadas (sigla MLP do inglês '*Multilayer Perceptron*') (GARDNER; DORLING, 1998) é uma arquitetura de ANN composta de uma camada com as entradas da rede, uma ou mais camadas escondidas com múltiplos neurônios, e uma camada com as saídas da rede. As camadas são totalmente conectadas, ou seja, todos as saídas dos neurônios de uma camada são submetidas como entradas para a próxima camada. Além disso, as camadas escondidas e a camada de saída possuem funções de ativação que realizam transformações não lineares nas entradas.

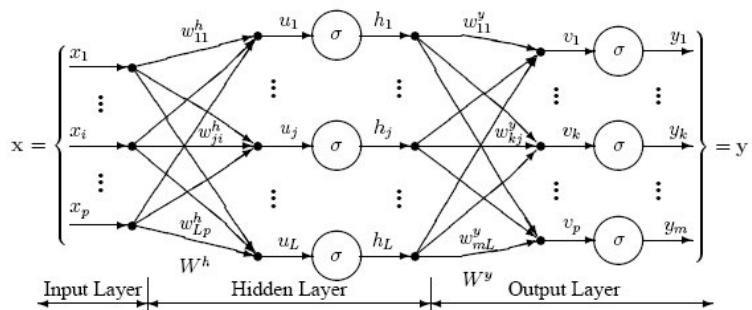


Figura 4.2: Imagem de uma rede MLP, mostrando a camada de entrada, a camada escondida com a função de ativação e a camada de saída. (AILEPHANT, )

A MLP utiliza uma técnica de aprendizado supervisionado (ALLOGHANI et al., 2020) chamada de retro propagação de erros (CHAUVIN;

RUMELHART, 1995), do inglês '*backpropagation*'. É possível representar o erro em um neurônio de saída  $j$  para entrada  $n$  pela equação 4-2.

$$e_j(n) = d_j(n) - y_j(n) \quad (4-2)$$

Aonde  $d$  é o valor desejado para a saída, e  $y$  é o valor produzido pelo neurônio.

Os pesos atribuídos às entradas dos neurônios podem então ser ajustados para tentar minimizar o erro acumulado da rede dado pela equação 4-3.

$$\mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n) \quad (4-3)$$

É preciso definir uma função de perda, que será utilizada para medir o erro geral do modelo. Ao utilizar o método dos mínimos quadrados como função de perda, e realizar o método da descida de gradiente (HOCHREITER; YOUNGER; CONWELL, 2001), podemos definir a variação de cada peso de acordo com a equação 4-4.

$$\delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n) \quad (4-4)$$

Aonde  $y_i$  é a saída do neurônio anterior,  $\eta$  é a taxa de aprendizado (YU; CHEN, 1997) do modelo e  $v_j$  é o campo local induzido, que é a soma das entradas multiplicadas pelos seus pesos, antes de passar pela função de ativação.

Com isso, a cada ciclo de aprendizado, também chamado de '*epoch*', a rede vai ajustando os seus pesos, até um ponto ótimo.

## 4.2 Overfitting

'Overfitting' é um fenômeno que acontece quando a rede se ajusta muito bem a um conjunto de dados, ou seja, tem o seu erro muito próximo de zero, e com isso, ajusta minimamente os seus pesos, pois já alcançou um ponto ótimo. Porém, a rede é incapaz de generalizar seu conhecimento para ser utilizada com novos dados de entrada. Existem diversos métodos para evitar que o 'overfitting' ocorra, chamadas de técnicas de regularização (YING, 2019).

|                             | Underfitting   | Just right  | Overfitting   |
|-----------------------------|--|---|---|
| Symptoms                    | <ul style="list-style-type: none"> <li>• High training error</li> <li>• Training error close to test error</li> <li>• High bias</li> </ul> | <ul style="list-style-type: none"> <li>• Training error slightly lower than test error</li> </ul> | <ul style="list-style-type: none"> <li>• Very low training error</li> <li>• Training error much lower than test error</li> <li>• High variance</li> </ul> |
| Regression illustration     |  |   |   |
| Classification illustration |  |   |   |
| Deep learning illustration  |  |   |   |
| Possible remedies           | <ul style="list-style-type: none"> <li>• Complexify model</li> <li>• Add more features</li> <li>• Train longer</li> </ul>                  |   | <ul style="list-style-type: none"> <li>• Perform regularization</li> <li>• Get more data</li> </ul>   |

Figura 4.3: Tabela mostrando o efeito do 'overfitting' em diversos modelos de redes neurais. (SHRIVASTAVA, )

## 4.3 Rede Neural Convolucional

Uma Rede Neural Convolucional (CNN do inglês '*Convolutional Neural Network*') (ALBAWI; MOHAMMED; AL-ZAWI, 2017) é uma variação de uma rede MLP que realiza uma operação matemática da convolução em pelo

menos uma das camadas, e utiliza técnicas de regularização para lidar com o '*overfitting*'.

A CNN, como a ANN, também é inspirada em um processo biológico representado pelo padrão de conectividade entre os neurônios dentro do córtex visual de um animal (FU et al., 2016). Certos neurônios corticais respondem a estímulos apenas em uma região restrita do campo visual conhecida como campo receptivo. Porém, os campos receptivos de diferentes neurônios se sobrepõem parcialmente, de forma que eles consigam cobrir todo o campo visual.

Diferentemente da rede MLP, em que todas as camadas são totalmente conectadas, a CNN pode apresentar múltiplos padrões de camadas, responsáveis por uma determinada parte do processo por meio de diferentes operações. Juntas, as camadas tornam a rede capaz de ser aplicável em várias classes de problemas, como classificação de imagens, reconhecimento de imagem e vídeo, segmentação de imagens, e muitos outros.

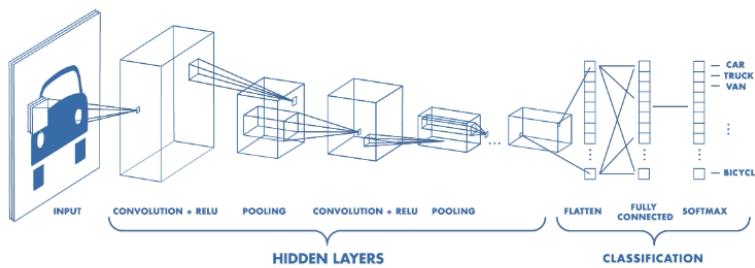


Figura 4.4: Imagem mostrando a arquitetura e organização de uma CNN. (CORNELISSE, )

#### 4.3.1 Camada de Convolução

A camada de convolução é o principal elemento da CNN. A convolução é uma operação matemática que dado duas funções, produz uma terceira função que expressa a sobreposição dessas. Na CNN, a primeira função é representada pelas entradas da camada de convolução, e a segunda função é o

chamado de *kernel*, representado por uma matriz. A concatenação de múltiplos '*kerneis*', tendo cada *kernel* atribuído a uma dimensão da entrada, é chamada de filtro (GANESH, 2019). Na camada de convolução, o filtro é deslocado pela dimensão da altura e largura da entrada, realizando uma multiplicação de matriz entre o filtro e a parte da entrada sobreposta. Essa operação produz um mapa de características daquela entrada, que contém um traço importante a ser aprendido, e é utilizada ao longo do treinamento para melhorar os filtros utilizados, e consequentemente aprender.

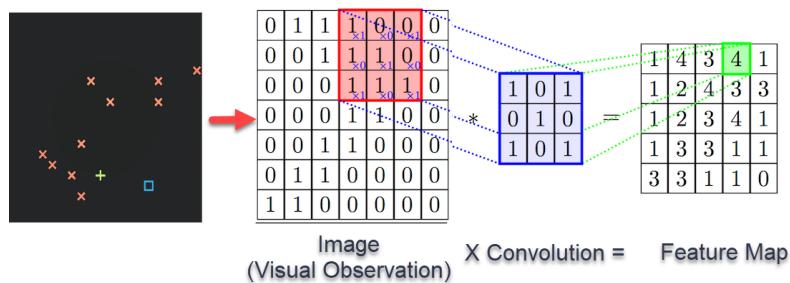


Figura 4.5: Imagem mostrando a operação de convolução. (CONVOLUTIONAL... , )

### 4.3.2

#### Camada de *Pooling*

A camada de *Pooling* é utilizada para reduzir a variância da entrada, por meio da redução do tamanho espacial, enquanto extrai as características dominantes, além de reduzir o ruído presente. O fato de reduzir o número de dimensões faz com que a rede necessite de menor poder computacional para processar as informações, o que é benéfico. Assim como a camada de convolução, a camada de Pooling também utiliza filtros, porém estes não possuem parâmetros que podem ser treinados. Existem múltiplas modalidades de *Pooling*, porém duas delas são as mais utilizadas.

No '*MaxPooling*', o filtro é deslocado sobre a entrada e é realizado a operação de máximo entre os elementos sobrepostos pelo filtro. Já no '*AveragePooling*', o procedimento é semelhante, porém é utilizado a operação

de média. O uso da média faz com que a extração das características seja mais suave, e como com isso, o uso da operação de máximo é mais utilizado.

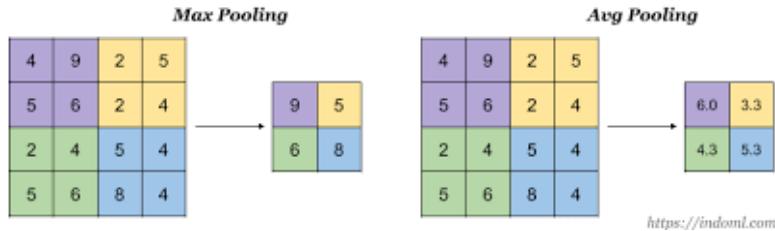


Figura 4.6: Imagem mostrando a operação de Pooling, tanto na modalidade de média como na modalidade de máximo. (POKHREL, )

#### 4.3.3

#### Camada Totalmente Conectada

A Camada Totalmente Conectada possui todos os seus neurônios conectados com os neurônios da camada seguinte. É a mesma configuração utilizada em todas as camadas escondidas da rede MLP. Igualmente, pode ser utilizada para aprender informações oriundas de combinações não lineares.

### 4.4

### Arquitetura U-NET

A arquitetura U-NET (RONNEBERGER; FISCHER; BROX, 2015) é uma CNN especializada em segmentação semântica de imagens biomédicas desenvolvida pela Universidade de Freiburg. Em um problema de classificação, a imagem inserida como entrada é convertida para um vetor. Durante esse processo, a rede aprende o mapa de características dessa imagem. Já para um problema de segmentação, é necessário reconstruir a imagem com base nesse vetor, o que pode ser custoso. A U-NET utiliza o mesmo mapa de características utilizadas para realizar a reconstrução. Dessa forma, consegue apresentar ótimos resultados.

Como o desafio a ser solucionado dentro deste trabalho é um problema de segmentação, a rede U-NET foi a escolhida para ser utilizada. A arquitetura do modelo consiste de duas fluxos, o primeiro sendo o fluxo de contração (também chamado de codificador), e o segundo o fluxo de expansão (também chamado

de decodificador). Essa organização faz com que a arquitetura tenha formato em letra U, de onde origina o nome U-NET.

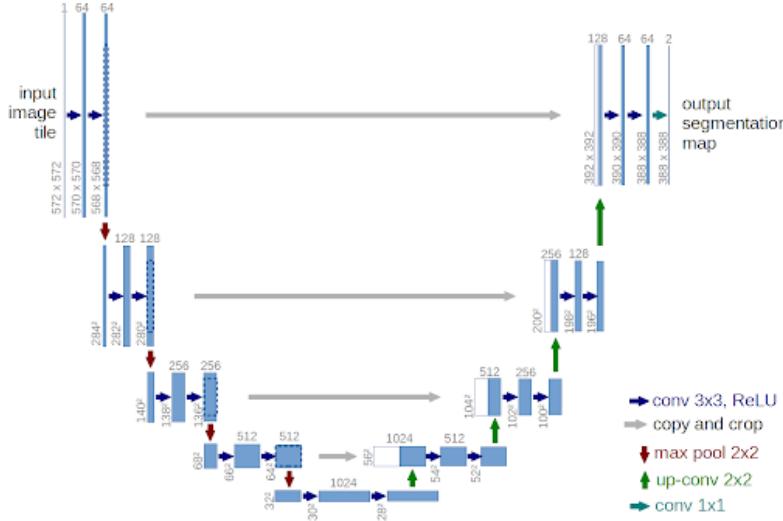


Figura 4.7: Imagem mostrando a arquitetura U-NET. Na metade esquerda pode-se visualizar as camadas pertencentes ao codificador, e na metade direita, é possível ver as camadas pertencentes ao decodificador. (LAPIX, )

O fluxo de contração é formado por uma série de camadas de convolução e camadas de Pooling com a operação de máximo ('*MaxPooling*'). Por meio dessas transformações, a informação espacial da entrada é reduzida enquanto as informações presentes no mapa de características são ampliadas, por meio do aumento do número de dimensões. Basicamente, se mantém a informação presente na entrada, porém perde-se a informação posicional. Por exemplo, uma entrada de dimensões 128x128x3 é transformada em dimensões 8x8x256.

O fluxo de expansão é formado de convoluções transpostas (ODENA; DUMOULIN; OLAH, 2016), seguido por convoluções padrões. Por meio dessas transformações, o tamanho da entrada gradualmente aumenta enquanto o seu número de dimensões diminui. Com isso, o decodificador é capaz de restaurar a posição de cada informação aprendida. A cada nível do decodificador, é utilizado o padrão '*skip connections*' (ADALOGLOU, 2020) para que o modelo consiga obter posicionamentos mais precisos. Isso é feito por meio da concatenação da saída da camada de convolução transposta com o mapa de características do mesmo nível no codificador (LAMBA, 2019).

A U-NET utilizada neste trabalho recebe segmento de imagens do cérebro como primeira entrada e segmento de imagens das máscaras das lesões como segunda entrada, ambas com dimensões 64x64x1. A cada nível no fluxo de contração, são utilizadas duas camadas de convolução seguidas de uma camada de *MaxPooling*. Esse padrão se repete quatro vezes até que se inicie o fluxo de expansão. Nesta etapa, a cada nível são utilizadas duas camadas de convolução seguidas por uma camada de convolução transporta. Esse padrão se repete quatro vezes. Depois disso, existem duas camadas de convolução e uma terceira camada de convolução que produz a saída.

#### **4.5**

#### **Função de Perda**

A função de perda é utilizada para calcular o erro geral do modelo para realizar ajustes nos parâmetros da rede ao longo do treinamento. Existem múltiplas funções de perda diferentes, e cada uma delas possui sua particularidade (WANG et al., 2020). Foi escolhida a função de perda '*Cross-Entropy*' para este trabalho, pois tem um ótimo desempenho com problemas de segmentação, e com isso, funciona muito bem junto a U-NET. Essa função é definida pela equação 4-5.

$$CE = - \sum_i^C t_i \log(f(s_i)) \quad (4-5)$$

Aonde  $t_i$  é o valor da máscara e  $s_i$  é o valor obtido como resposta na saída da CNN, que passa pela função de ativação da camada de saída, dado uma classe  $i$  de  $C$

Inicialmente, foi adotado o número de classes igual a dois, sendo elas lesão e não lesão. Com isso, foi utilizado a variação da função descrita acima, '*Binary Cross-Entropy*', que é definida pela equação 4-6. Essa função de perda exige que a função de ativação da camada de saída seja a função '*Sigmóide*'. (NWANKPA et al., 2018)

$$CE = - \sum_i^{C=2} t_i \log(f(s_i)) = -t_1 \log(f(s_1)) - (1 - t_1) \log(1 - f(s_1)) \quad (4-6)$$

Aonde o valor da máscara pode ser apenas 1 (lesão) ou 0 (não lesão).

Porém, foi possível perceber que os exames da base de dados apresentavam ser dados não balanceados, ou seja, cada exame possui um número muito maior de elementos da classe de não lesão do que da classe com lesão. Sendo assim, foi necessário balancear os dados.

Para tanto, alterou-se a função de perda para outra variação chamada '*Weighted Categorical Cross-Entropy*', definida pela equação 4-7. Essa função de perda aplica pesos aos elementos para conseguir balancear as classes, multiplicando um peso elevado a classe com poucas ocorrências, e um peso menor para a classe em abundância. Para realizar tal tarefa, a implementação utilizada da função exige que a entrada contendo as máscaras esteja codificada em vetores '*one-hot*' (POTDAR; PARDAWALA; PAI, 2017). Assim, é necessário realizar a mudança da função binária para a categórica, que é utilizada quando há mais de duas classes. Essa função de perda exige que a função de ativação da camada de saída seja a função '*Softmax*' (NWANKPA et al., 2018).

$$CE = - \sum_i^C w_i t_i \log(f(s_i)) \quad (4-7)$$

Dado que  $w_i$  seja o peso atribuído a classe  $i$ .

## 4.6 Função de Ativação

A função de ativação (SHARMA; SHARMA, 2017) está presente em todas as camadas escondidas e na camada de saída, e é utilizada para realizar transformações na soma das entradas ponderadas por seus pesos. Essas transformações permitem que o modelo saiba decidir se a informação

presente na entrada do neurônio é importante para o aprendizado ou se deve ser ignorada.

Neste trabalho, é utilizado uma única função de ativação em todas as camadas escondidas do modelo. A função escolhida é a chamada 'ReLU' (do inglês '*Rectified Linear Unit*') (ECKLE; SCHMIDT-HIEBER, 2019). Essa função é definida pela equação 4-8.

$$f(x) = \max(0, x) \quad (4-8)$$

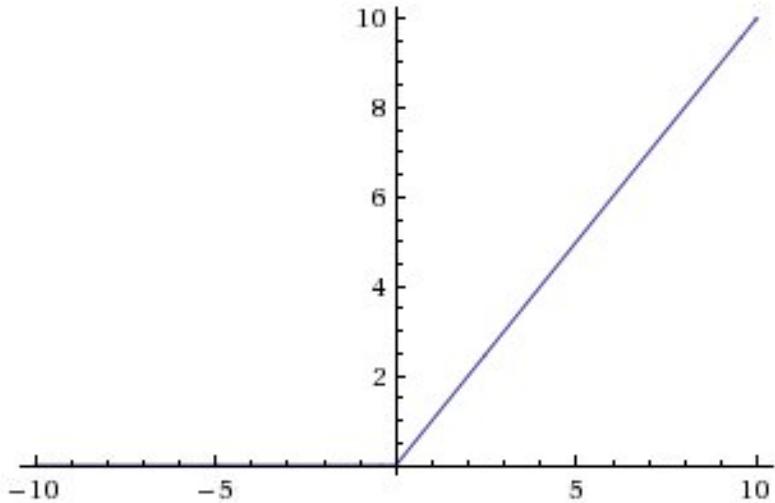


Figura 4.8: Também conhecida como função rampa, é análoga a retificação de meia onda para os conceitos de engenharia elétrica. (BECKER, )

Atualmente, a função 'ReLU' é a função de ativação mais popular utilizada em modelos de '*deep learning*' (EMMERT-STREIB et al., 2020). Entre suas vantagens, a função possui uma ativação esparsa (GALE, 2020), diminuindo o número de neurônios ativados, que ajuda a diminuir o '*overfitting*'. E ainda possui uma ótima propagação do gradiente, reduzindo a ocorrência de problemas de dissipação do gradiente (HOCHREITER, 1998). Tais fatores fazem com que tal função tenha uma ótima eficiência computacional, por ser formada apenas de comparações, adições e multiplicações. Por fim, é invariante na escala (XU et al., 2014), o que pode ser visto na equação 4-9.

$$\max(0, ax) = a \max(0, x) \text{ para } a \geq 0 \quad (4-9)$$

Além disso, há a necessidade de utilizar uma função de ativação na camada de saída para adequar os dados a um formato específico. No modelo utilizado no trabalho foram utilizadas duas funções em momentos distintos. A escolha dessa função depende de como deseja o formato da saída da rede, que será a imagens das previsões das lesões.

Na seção anterior, foi explicado que quando se utiliza a função de perda '*Binary Cross-Entropy*', a entrada que contém as máscaras possui o formato 64x64x1 contendo zero e uns. Com isso, a saída do modelo contendo as previsões das lesões também estará nesse formato. Para isso, é utilizado a função de ativação '*Sigmóide*', definida pela equação 4-10.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (4-10)$$

Esta é uma função suave e continuamente individualizada. Além disso, é uma função não linear e seu valor varia entre 0 e 1. Isso importa dizer que o modelo terá uma saída no mesmo formato da entrada contendo as máscaras, 64x64x1, e possuirá valor entre 0 e 1, correspondente a probabilidade daquele pixel conter uma lesão.

Quando é utilizado a função de perda '*Weighted Categorical CrossEntropy*', para tentar resolver o problema de balanceamento de classes, é necessário utilizar a codificação '*one-hot*' para a entrada contendo as máscaras. Tal ato faz com que seja necessário que a saída contendo as máscaras também possua essa codificação. Para lidar com isso, é necessário substituir a função de ativação '*Sigmóide*' pela '*Softmax*', definida na equação 4-11.

$$\sigma(x)_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} \quad (4-11)$$

Assim, o modelo terá uma saída no formato 64x64x1, e em vez de possuir cada valor entre zero e um, será um vetor com as probabilidades de não ter lesão e de possuir lesão, no formato '*one-hot*'.

# 5

## Preparação das Entradas

Após realizadas as técnicas de pré-processamento, inicia-se a preparação e construção das entradas que serão utilizadas no modelo. O modelo terá duas entradas: a primeira será uma lista com as imagens do cérebro e a segunda será uma lista das imagens das máscaras (SYDORENKO, 2020) indicando o posicionamento das lesões.

### 5.1

#### Imagens do Cérebro

O modelo utiliza uma lista de imagens 2D como primeira entrada, porém a base de dados apresenta formato NIFTI (LAROBINA; MURINO, 2014) (LI et al., 2016), divididos em volumes tridimensionais, após o pré-processamento. Destarte, é necessário dividir esses volumes em arquivos que contenham cada uma das fatias.

Após fazer esse procedimento, poder-se-ia utilizar os arquivos que continham as fatias como primeira entrada da rede. Porém, com o objetivo de realçar as lesões, e obter um melhor resultado, decidiu-se por dividir essas imagens em segmentos menores, chamado na comunidade por '*Patches*' (SHARMA et al., 2017).

Além disso, modelos de aprendizado de máquina como CNN apresentam resultados melhores de forma proporcional a quantidade de dados disponíveis para treinamento (CHO et al., 2016). A base de dados deste trabalho apresenta 47 exames com marcações, e cada exame apresenta 30 fatias. Segmentar as fatias em pedaços menores não gera novas informações, porém aumenta a quantidade de entradas submetidas ao modelo durante o treinamento, o que pode melhorar a performance da rede.

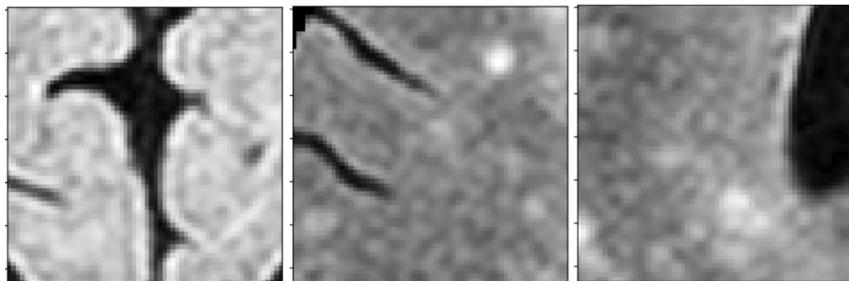


Figura 5.1: Imagens do cérebro divididos em segmentos em menores, também chamados de 'Patches'.

## 5.2 Máscaras das Lesões

A base de dados é originada da Empresa Dasa, que contém as imagens dos exames de ressonância do tipo FLAIR e um arquivo de texto que contém informações sobre as lesões para cada um dos exames. Esses arquivos contém as coordenadas x e y em pixels de cada uma das lesões, bem como o número da fatia em que ela está presente.

Para a segunda entrada do modelo, é necessário submeter uma lista de imagens binárias com as mesmas dimensões da primeira entrada, no entanto, ao invés de possuir os seus bits com os valores da imagem do cérebro, terão valor de 1 (um) caso a sua correspondência na imagem do cérebro apresente uma área de lesão, e valor 0 (zero) caso seja tecido saudável. Essa representação equivale a um mapa das lesões para cada uma das fatias. Foi necessário iterar cada um dos arquivos contendo as coordenadas dos pixeis com lesões para construir essas imagens.

---

### Código 1: Trecho de Arquivo de Texto de Marcações

---

```
1 Segmentation 7
2 333 380 16
3 334 379 16
4 335 378 16
5 336 378 16
6 337 378 16
7 338 377 16
8 339 376 16
```

9 339 375 16  
10 340 374 16  
11 340 373 16  
12 341 372 16  
13 341 371 16  
14 340 370 16  
15 339 369 16  
16 338 369 16  
17 337 369 16  
18 336 369 16  
19 335 369 16  
20 334 370 16  
21 333 371 16  
22 333 372 16  
23 333 373 16  
24 333 374 16  
25 332 375 16  
26 332 376 16  
27 331 377 16  
28 332 378 16  
29 332 379 16  
30 333 380 16  
31 V

---

Ato contínuo, em virtude das coordenadas presentes nos arquivos de texto, é possível criar o mapa das lesões para cada uma das fatias, e utilizá-los como entrada para o modelo.

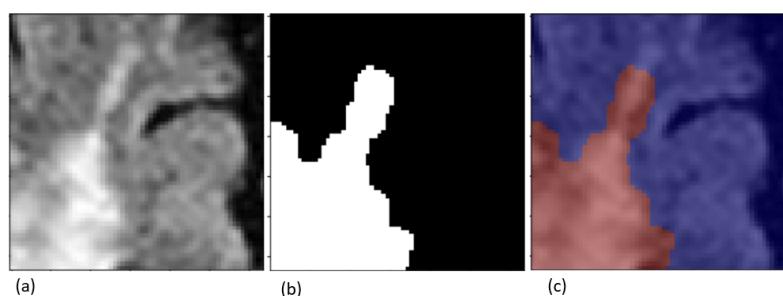


Figura 5.2: Imagens de um segmento do cérebro e sua respectiva máscara da lesão.  
(a) Tecido do cérebro (b) Máscara da lesão (c) Sobreposição da máscara com o tecido do cérebro

## 6

# Técnicas de Regularização

### 6.1

#### Parada Antecipada

Uma das técnicas de regularização (AP, ) (YING, 2019) mais utilizadas é a chamada 'Parada Antecipada' (do inglês '*Early Stopping*'). A base de dados, ao invés de ser dividida em duas partes, é dividida em três, sendo elas o treinamento, validação e teste. A parte de validação é então introduzida ao modelo junto com a base de treinamento.

Com isso, a cada ciclo de treinamento, além da rede ajustar os pesos com base no erro geral dos parâmetros, ela verifica como a rede se comporta utilizando os mesmos pesos juntos com a base de validação, que não é utilizada durante o treinamento para o ajuste dos pesos. Desse modo, é possível ver se a rede ainda está melhorando seu resultado sobre uma base desconhecida. Assim, quando o erro sobre a base de validação parar de diminuir, o modelo pode interromper o seu treinamento de forma antecipada.

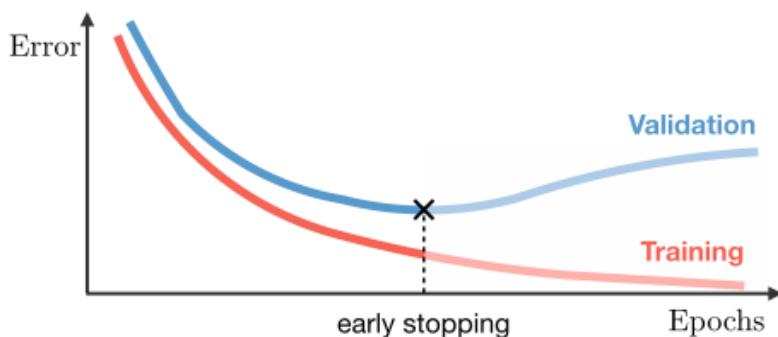


Figura 6.1: Imagem mostrando o ponto ótimo de término do treinamento de uma rede neural, com objetivo de evitar que ocorra o '*overfitting*'. (AP, )

### 6.2

#### Data Augmentation

As técnicas chamadas de *Data Augmentation* (DYK; MENG, 2001) aumentam o número de dados existentes e podem ser utilizadas para combater

a pouca disponibilidade de dados para o treinamento. A técnica utilizada nesse trabalho faz do uso de transformações nos dados existentes para criar novos exemplares. Essa mesma transformação é aplicada à máscara de lesões, para que seja possível que o modelo consiga inferir corretamente o posicionamento das lesões.

Existem duas maneiras de utilizar o *Data Augmentation* (ROSEBROCK, 2021), a primeira é fazer a expansão da base de dados com a criação de novos exemplares e depois utilizar essa nova base de dados no treinamento do modelo.

A segunda maneira é aplicar as transformações ao longo do treinamento do modelo, e com isso, a cada ciclo de aprendizado, são gerados novos exemplares de dados. Dessa forma, a rede nunca será submetida a um mesmo dado mais de uma vez, e assim, consegue melhorar sua generalização.

### **6.2.1 Rotação**

A rotação é uma transformação geométrica de um sistema de coordenadas que faz com que todos os pixels de uma imagem sejam deslocados por um determinado ângulo, em torno do centro dado pela coordenada (0, 0).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (6-1)$$

Os pixeis originais situados nas coordenadas (x,y) são mapeados nas novas coordenadas (x',y'), deslocados pelo ângulo  $\theta$

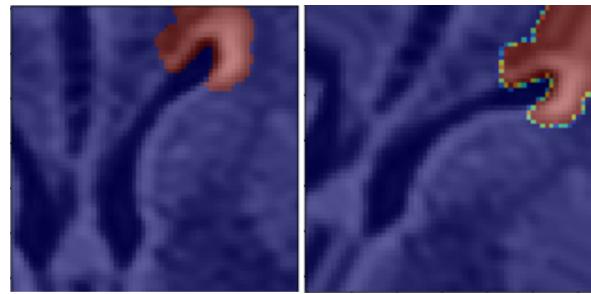


Figura 6.2: Segmentos do cérebro e sua máscara, antes e depois da transformação de rotação.

### 6.2.2 Inversão

A inversão, também chamada de efeito espelho, é uma transformação que produz uma nova imagem de mesmo tamanho e formato, porém que está espelhada tanto no eixo x como possivelmente no eixo y.

$$(x', y') = (\alpha - x - 1, y) \quad (6-2)$$

Os pixeis originais situados nas coordenadas  $(x, y)$  são mapeados nas novas coordenadas  $(x', y')$  depois da inversão horizontal, sendo  $\alpha$  o comprimento da imagem em pixeis.

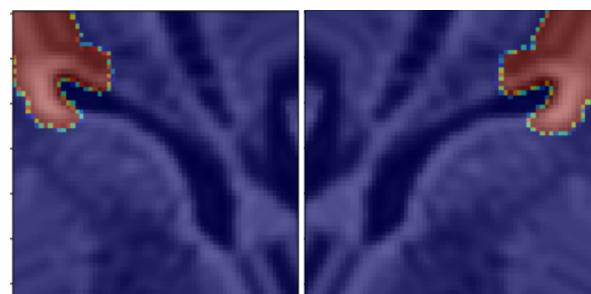


Figura 6.3: Segmentos do cérebro e sua máscara, antes e depois da transformação de inversão horizontal.

$$(x', y') = (x, \beta - y - 1) \quad (6-3)$$

Os pixeis originais situados nas coordenadas  $(x, y)$  são mapeados nas novas coordenadas  $(x', y')$  depois da inversão vertical, sendo  $\beta$  a altura da imagem em pixeis.

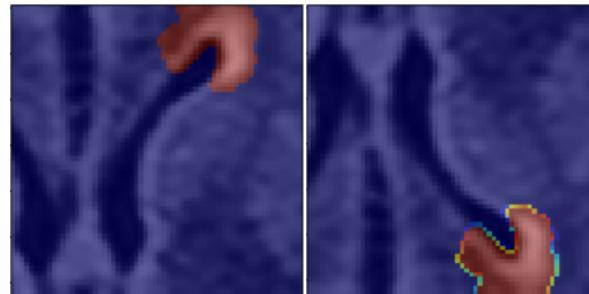


Figura 6.4: Segmentos do cérebro e sua máscara, antes e depois da transformação de inversão vertical.

# 7

## Métricas

Abaixo será explicado cada uma das métricas, e será utilizado alguns termos nas equações. O número de previsões verdadeiro-positivas é representado pela sigla VP (amostras com previsões positivas em que o elemento tem lesão). O número de previsões verdadeiro-negativas é representado pela sigla VN (amostras com previsões negativas em que o elemento não tem lesão). O número de previsões falso-positivas é representado pela sigla FP (amostras com previsões positivas em que o elemento não tem lesão). Por fim, o número de previsões falso-negativas é representado pela sigla FN (amostras com previsões negativas em que o elemento tem lesão).

### 7.1

#### Perda

A perda é o somatório da diferença entre o valor previsto por cada neurônio da rede, e o valor real que deveria ter sido obtido. A função de perda definida na arquitetura da rede utiliza esse somatório como parâmetro, e retorna o valor da métrica. A perda é utilizada para ter uma visão geral de como o modelo está performando, além de ser utilizada para ajustar os parâmetros e pesos presentes na rede.

### 7.2

#### Acurácia

Acurácia é a medição percentual do número de previsões que possuem o valor real. Esta métrica é dada pela equação 7-1.

$$acc = \frac{VP + VN}{VP + VN + FP + FN} \quad (7-1)$$

A acurácia, por conta própria, não consegue capturar se o modelo tem

uma ótima performance quando a base de dados utilizada é desbalanceada. Isso pode ser visto com o exemplo hipotético a seguir. Em uma imagem com 100 (cem) pixels, três deles apresentaram lesão e noventa e sete não apresentaram lesão. Caso o modelo previr que todos os pixels são saudáveis, ela apresentará uma acurácia de 97%, porém não conseguirá detectar os pixels com lesão. Neste caso, isso se dá por conta do número de falso-negativos, porém em outras situações o responsável pelo problema são os elementos falso-positivos. Para lidar com esse problema, são utilizadas as métricas de '*recall*' e precisão (KOEHRSEN, 2018).

### **7.3 Recall**

O '*recall*' mede a capacidade do modelo de encontrar todos os casos relevantes dentro da base de dados, e com isso, mensura a proporção de amostras positivas que foram identificadas corretamente. Para este trabalho, seriam os casos de lesão. Essa métrica é dada pela equação 7-2, e quanto maior o número de falsos negativos, menor o valor da métrica.

$$rec = \frac{VP}{VP + FN} \quad (7-2)$$

### **7.4 Precisão**

A precisão mede a capacidade do modelo de identificar somente elementos relevantes dentro da base de dados. Em outras palavras, a precisão expressa a proporção de casos realmente relevantes dentre os casos classificados nessas condições pelo modelo. Essa métrica é dada pela equação 7-3, e quanto maior o número de falsos positivos, menor o valor da métrica.

$$prec = \frac{VP}{VP + FP} \quad (7-3)$$

## 7.5 Especificidade

A Especificidade, de forma semelhante ao '*recall*', mede a capacidade do modelo de encontrar os casos não relevantes dentro da base de dados, e com isso, mensura a proporção de amostras negativas que foram identificadas corretamente. Para este trabalho, esses casos seriam os casos de não lesão. Essa métrica é dada pela equação 7-4, e quanto maior o número de falsos positivos, menor o valor da métrica.

$$esp = \frac{VN}{VN + FP} \quad (7-4)$$

## 7.6 F1 Score

A métrica *F1 Score* é uma média harmônica da precisão e '*recall*'. O fato dessa métrica utilizar a média harmônica em vez da média aritmética faz com que haja uma maior penalidade sobre os valores extremos, com isso, apresentará um F1 Score muito baixo, caso tanto a precisão quanto o '*recall*' estejam baixos. A métrica é dada pela equação 7-5.

$$f1 = 2 * \frac{prec * rec}{prec + rec} \quad (7-5)$$

# 8

## Resultados

O método proposto neste trabalho apresentou bom desempenho na tarefa de identificar e marcar as lesões na substância branca do cérebro. Para analisar este resultado, foram utilizadas as métricas explicadas anteriormente durante o treinamento para mensurar o aprendizado da rede; bem como foram utilizadas as mesmas métricas na base de validação para mensurar a capacidade de generalização, e na base de teste para avaliar se a rede foi capaz de aprender a um nível desejado.

### 8.1

#### Uso das Metricas

As temáticas explicitadas acima são utilizadas diversas vezes durante a pesquisa deste trabalho. Primeiramente, elas são mensuradas durante o treinamento, e analisadas para detectar se o modelo está sendo capaz de aprender a solucionar o problema em questão.

Tabela 8.1: Valor das Métricas sobre a Base de Treinamento no Fim do Aprendizado

| Métrica        | Valor  |
|----------------|--------|
| Perda          | 0.0266 |
| Acurácia       | 0.9797 |
| Recall         | 0.7113 |
| Especificidade | 0.9846 |
| Precisão       | 0.8281 |
| F1 Score       | 0.7624 |

A métrica de perda sobre a base de validação é parâmetro para a parada antecipada com o objetivo de evitar o '*overfitting*' do modelo. As outras métricas também são analisadas utilizando esta base de dados, para entender se o modelo apresenta bom resultado com dados não vistos durante o treinamento. Em síntese, estas métricas medem de forma antecipada a capacidade do modelo de lidar com dados novos.

Tabela 8.2: Valor das Métricas sobre a Base de Validação no Fim do Treinamento

| Métrica        | Valor  |
|----------------|--------|
| Perda          | 0.0466 |
| Acurácia       | 0.9934 |
| Recall         | 0.7296 |
| Especificidade | 0.9867 |
| Precisão       | 0.6815 |
| F1 Score       | 0.7024 |

Ao fim do aprendizado, as bases de treinamento e validação são misturadas e o modelo é submetido a mais alguns ciclos de aprendizado. Isto faz com que o modelo aumente a sua capacidade de aprendizado, e consiga absorver conhecimento da base de validação, já que esta base não foi conhecida pela rede durante o treinamento.

Tabela 8.3: Valor das Métricas ao Fim do Aprendizado utilizando a Base de Treinamento Misturada com a de Validação

| Métrica        | Valor  |
|----------------|--------|
| Perda          | 0.0266 |
| Acurácia       | 0.9918 |
| Recall         | 0.7705 |
| Especificidade | 0.9851 |
| Precisão       | 0.6689 |
| F1 Score       | 0.7114 |

Depois de realizado o treinamento, a base de teste foi submetida ao modelo para avaliar o resultado nos exames que não foram utilizadas durante o treinamento. Nestes exames, as métrica de '*recall*' e precisão apresentaram valores menores e com isso, em determinados exames, houve a ocorrência de um número maior de falsos positivos e falsos negativos. Essa questão pode ser melhorada ao realizar outras variações de métodos de pré-processamento.

Tabela 8.4: Valor das Métricas sobre a Base de Teste

| Métrica        | Valor  |
|----------------|--------|
| Perda          | 0.0183 |
| Acurácia       | 0.9943 |
| Recall         | 0.5582 |
| Especificidade | 0.9902 |
| Precisão       | 0.5213 |
| F1 Score       | 0.5391 |

## 8.2

### Análise de Casos

Nessa seção serão analisados alguns exames com as marcações das lesões, e também o resultado da previsão do modelo para estes mesmos exames. Serão utilizadas imagens de fatias de diferentes exames de ressonância magnética.

A Figura 7-1 mostra que o modelo foi capaz de detectar e segmentar a maioria das lesões na substância branca do cérebro, com as regiões marcadas apresentando elevada acurácia.

A Figura 7-2 também revela que a rede conseguiu identificar e marcar a grande maioria das lesões. Dessa vez, é possível visualizar casos de falsos positivos na detecção do modelo. A utilização de mais exemplares durante o treinamento faria com que o modelo apresentasse uma métrica de precisão maior, consequentemente, reduziria o número de falsos positivos apresentados.

A Figura 7-3 expõe outra comparação entre exame e detecção do modelo muito próximos. É possível perceber que agora existe um caso de falso negativo no resultado da previsão. Como esse projeto implementa um CAD, e o resultado dele será analisado por um profissional de saúde para confirmar as lesões, é preferível que o modelo tenha menos casos de falsos negativos do que de falsos positivos. Contudo, como o '*recall*' não é perfeito, é possível que estes casos ocorram. Esse problema também pode ser combatido adicionando outros exemplares de exames para treinar o modelo.

A Figura 7-4 mostra outro caso em que a rede identifica a maioria

das lesões com perfeição. Além disso, é possível notar que há áreas de hiperintensidade que não são consideradas lesões pelas marcações, porém o modelo as identificará como potenciais lesões. Como não podemos garantir que há áreas lesionadas não marcadas, essa detecção do modelo é outro caso de falso positivo.

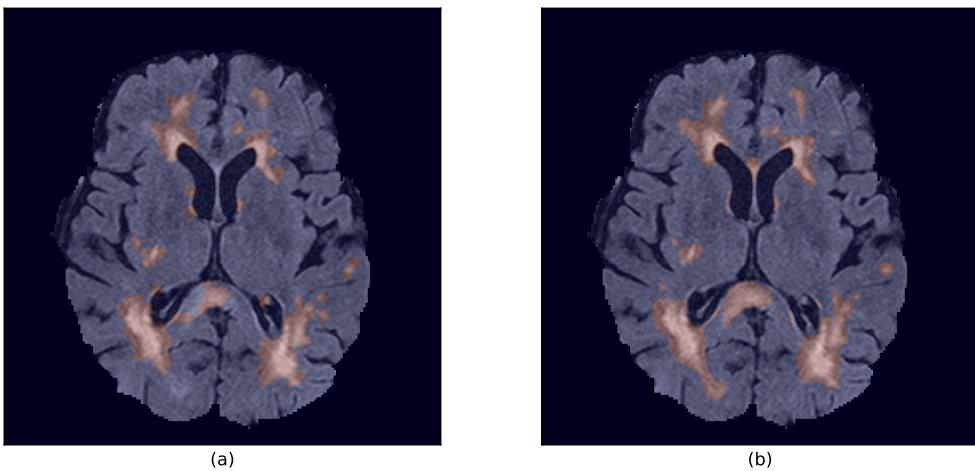


Figura 8.1: Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão

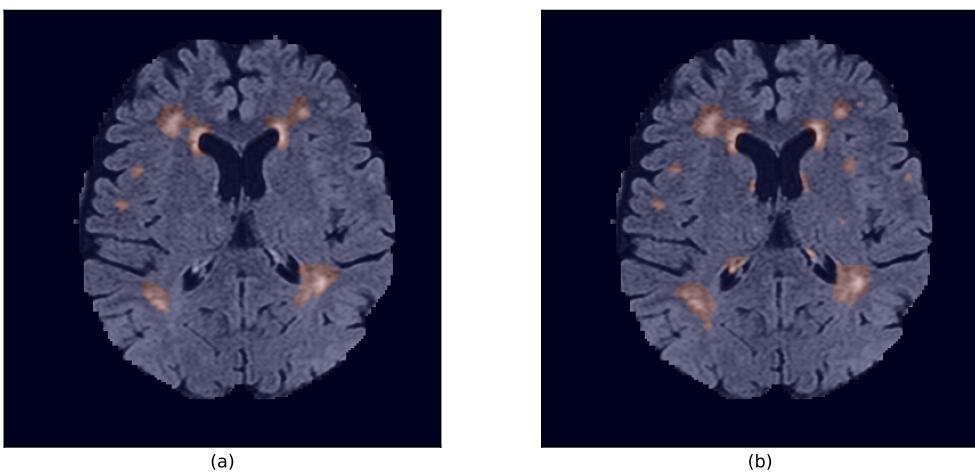


Figura 8.2: Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão

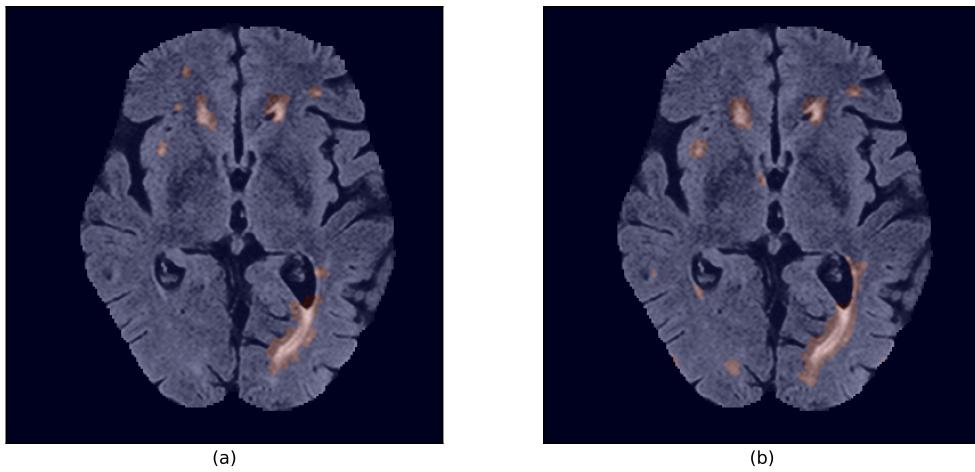


Figura 8.3: Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão

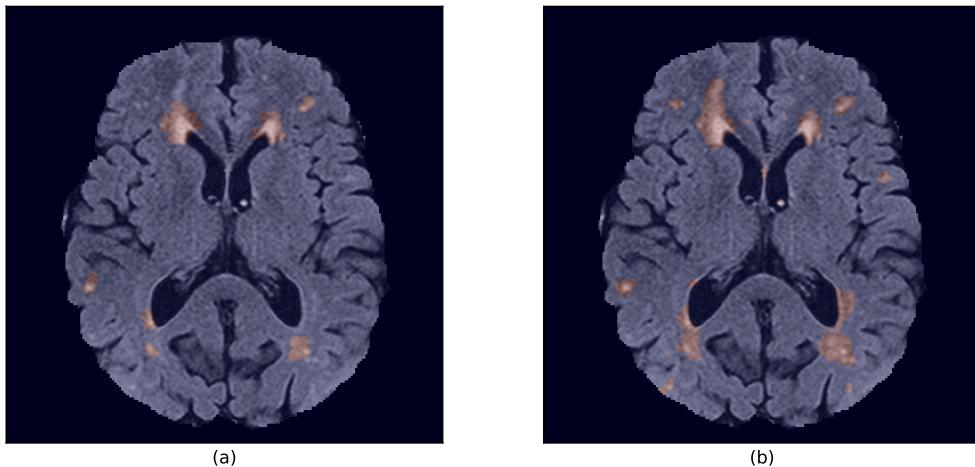


Figura 8.4: Imagens do cérebro com a máscara das lesões e a previsão das lesões feitas pelo modelo (a) Máscara (b) Previsão

## 8.3 Comparação

### 8.3.1 Trabalho Base

Como dito no capítulo de introdução, o presente trabalho é baseado em uma Tese de Doutorado sobre a Detecção de Lesões na Substância Branca do Cérebro, utilizando Imagens T1 e *FLAIR*. A tese utiliza a mesma base de dados do Dasa, que apresenta poucos exemplares de exames com as lesões marcadas. Nessa seção serão feitas comparações entre ambos os trabalhos.

A tese de Doutorado apresenta quatro métodos distintos de fazer a

detecção das lesões na substância branca do cérebro. Todos os métodos utilizam o algoritmo de segmentação (BAUER; NOLTE; REYES, 2012) para realizar a extração do crânio, porém o utiliza por meio de um software em linguagem C. Além disso, os quatro métodos utilizam o algoritmo SLIC0 para segmentar as imagens do cérebro, a fim de usar os superpixels marcados como lesão na criação das entradas da Rede.

A pesquisa da tese de Doutorado fez o uso de marcações manuais feitos pelo autor para lidar com o pequeno número de exemplares, da base de dados da Dasa, que possuíam marcações das lesões por especialistas, e poderiam ser utilizadas para o aprendizado do modelo.

Tabela 8.5: Resultado do método da tese de Doutorado com o melhor resultado na base de teste

| Métrica        | Valor  |
|----------------|--------|
| Acurácia       | 0.9793 |
| Recall         | 0.9012 |
| Especificidade | 0.9802 |

### 8.3.2 Trabalho Atual

Traçando uma comparação entre este trabalho de conclusão de curso e o trabalho base, pode-se destacar algumas diferenças.

Ambos os trabalhos utilizam o mesmo algoritmo para extração do crânio das imagens do cérebro, porém o presente trabalho de conclusão de curso faz o uso de uma implementação dentro da linguagem de programação *Python*, uma vez que a referida linguagem é utilizada em todo o resto da pesquisa e scripts, o que faz com que tenha um ganho de usabilidade.

O presente trabalho faz o uso da rede U-NET para realizar a segmentação das lesões. É uma rede neural especializada neste tipo de problema. Dessa forma, não é necessário utilizar o algoritmo SLIC0 utilizado no trabalho base.

Isto é benéfico pois além de obter um ganho de performance, é necessário menos uma etapa no processo.

Ademais, o trabalho em comento utiliza técnicas de regularização para aumentar o número de exemplares que possam ser utilizados no treinamento do modelo. Com isso, não há a necessidade de um esforço manual para marcar novas lesões, que apresentam baixa confiabilidade.

Lado outro, o trabalho base utiliza apenas três métricas para avaliar o resultado, sendo elas acurácia, '*recall*' e especificidade. Já o presente trabalho apresenta acurácia e especificidade mais evoluídos do que o melhor método do trabalho base, e um '*recall*' pior. No entanto, não é possível comparar ambos os trabalhos com um alto grau de confiabilidade. O trabalho base não apresenta métrica para precisão, e por conseguinte, não se sabe se há um número alto de falsos positivos. A especificidade não é suficiente para analisar este problema, pois como a base de dados é desbalanceada e possui um número maior de elementos que não são lesões, pode haver um alto número de falsos positivos sem que afete a métrica da especificidade.

## 9

# Conclusões

O presente trabalho apresenta um modelo de auxílio a detecção que pode ser utilizado por profissionais médicos para reduzir o número de exames que necessitam de análise. Contudo, é importante ter o diagnóstico feito por um especialista, para confirmar ou descartar as lesões identificadas pelo modelo, e esclarecer e elucidar detalhes sobre as lesões que constituem o aprendizado de um profissional de saúde enquanto não fazem parte do conhecimento adquirido pelo modelo.

A utilização de técnicas de pré-processamento é fundamental para que o modelo consiga obter resultados satisfatórios. A extração do crânio reduz a área de interesse dos dados, permitindo que o modelo entregue resultados melhores e reduza a quantidade do poder computacional necessário para processar as informações. As técnicas de equalização de histogramas são importantes para que o modelo consiga identificar, com precisão, o tecido com lesão em diferentes exames.

O uso de técnicas de regularização reduz a ocorrência de '*overfitting*' do modelo. A parada antecipada durante o treinamento permite que o modelo atinja um ponto de excelência de aprendizado quando utilizado com dados nunca por ele vistos, o que é o objetivo. Já o *Data Augmentation* permite que o modelo tenha resultados satisfatórios mesmo que apresentem um número baixo de exemplares de exames de ressonância magnética do cérebro, com marcações das lesões.

A escolha da arquitetura U-NET para o modelo de Rede Neural Convolutional foi uma decisão acertada, pois apresentou bons resultados sem a necessidade de realizar uma segmentação, a priori, do tecido cerebral. É importante lembrar que essa arquitetura apresenta um grande múltiplo de configurações e parâmetros. Isto faz com que seja necessário um tempo maior para identificar

a configuração que traga os melhores resultados possíveis.

O resultado deste trabalho foi satisfatório na identificação das lesões na substância branca do cérebro. O resultado encontrado não pode ser comparado com o trabalho base diretamente, por falta de algumas métricas no trabalho anterior. Porém, quando são comparadas as métricas disponíveis, o desempenho do modelo proposto obteve uma perda menor e teve mais acurácia. A análise é inconclusiva quando comparado as métricas sobre o número de falsos positivos e número de falsos negativos.

## 9.1

### Trabalhos Futuros

Devido à variedade de possibilidades e diferentes métodos existentes para a área de aprendizado de máquina e redes neurais, há diversas propostas para evoluir o trabalho de conclusão de curso em questão, de modo a melhorar o resultado obtido.

O baixo número de dados marcados é um fator mais impactante no momento de aprendizado do modelo. Dito isso, o uso de técnicas de *Data Augmentation* ajuda a resolver o referido problema, porém pode-se apresentar uma solução limitada. Destarte, é possível recorrer a criação sintética de novos dados. Uma das possibilidades é realizar a mesclagem de exames saudáveis com exames que apresentem lesão. Igualmente é possível utilizar Redes Adversárias Generativas (sigla GAN do inglês "*Generative Adversarial Network*") (YI; WALIA; BABYN, 2019) (LEDIG et al., 2017) para realizar a criação de novos dados com a mesma estatística e características dos dados utilizados no treinamento.

Outro método de pré-processamento válido é a técnica chamada de Equalização de Histograma Adaptativo. Esta é uma técnica de processamento digital que avalia múltiplos histogramas em uma mesma imagem, cada um referente a uma seção, e os utiliza para equalizar o brilho da imagem. É possível que utilizá-la pode melhorar o contraste local das imagens, sendo muito efetivo

para realçar a definição das bordas em cada segmento da imagem.

A técnica AHE é efetiva na acentuação das bordas para cada segmento da imagem, porém tem a tendência de aumentar excessivamente o ruído presente em regiões relativamente homogêneas. Para resolver essa questão, existe uma variação que limita à amplificação do ruído, chamado de Equalização de Histograma Adaptativo com Contraste Limitado. (REZA, 2004).

## Referências bibliográficas

- ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <<https://www.tensorflow.org/>>.
- ADALOGLOU, N. **Intuitive Explanation of Skip Connections in Deep Learning**. Sergios Karagiannakos, 2020. Disponível em: <<https://theaisummer.com/skip-connections/>>.
- AILEPHANT. **Multilayer Perceptron**. <<https://ailephant.com/glossary/multilayer-perceptron/>>. Accessed: 2021-05-25.
- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. **2017 International Conference on Engineering and Technology (ICET)**. [S.I.], 2017. p. 1–6.
- ALLOGHANI, M. et al. A systematic review on supervised and unsupervised machine learning algorithms for data science. In: \_\_\_\_\_. [S.I.: s.n.], 2020. p. 3–21. ISBN 978-3-030-22474-5.
- AP, P. **Regularization**. <<https://pranav-ap.netlify.app/posts/regularization/>>. Accessed: 2021-05-25.
- BAUER, S.; NOLTE, L.; REYES, M. Skull-stripping for tumor-bearing brain images. **Annual Meeting of the Swiss Society for Biomedical Engineering**, 04 2012.
- BECKER, D. **Rectified Linear Units (ReLU) in Deep Learning**. <<https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning>>. Accessed: 2021-05-25.
- BISONG, E. Google colaboratory. In: \_\_\_\_\_. [S.I.: s.n.], 2019. p. 59–64. ISBN 978-1-4842-4469-2.
- BRITANNICA, T. E. of E. et al. Moore's law. Encyclopædia Britannica, Inc.
- CASTELLINO, R. A. Computer aided detection (cad): an overview. **Cancer Imaging**, BioMed Central, v. 5, n. 1, p. 17, 2005.
- CHAUVIN, Y.; RUMELHART, D. E. **Backpropagation: theory, architectures, and applications**. [S.I.]: Psychology press, 1995.
- CHO, J. et al. **How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?** 2016.
- CHOLLET, F. et al. **Keras**. 2015. <<https://keras.io>>.
- CONVOLUTIONAL neural networks. <[https://subscription.packtpub.com/book/game\\_development/9781789138139/4/ch04lvl1sec31/convolutional-neural-networks](https://subscription.packtpub.com/book/game_development/9781789138139/4/ch04lvl1sec31/convolutional-neural-networks)>. Accessed: 2021-05-25.

CORNELISSE, D. An intuitive guide to Convolutional Neural Networks. <<https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>>. Accessed: 2021-05-25.

DYK, D. A. van; MENG, X.-L. The art of data augmentation. **Journal of Computational and Graphical Statistics**, Taylor Francis, v. 10, n. 1, p. 1–50, 2001. Disponível em: <<https://doi.org/10.1198/10618600152418584>>.

ECKLE, K.; SCHMIDT-HIEBER, J. A comparison of deep networks with relu activation function and linear spline-type methods. **Neural Networks**, v. 110, p. 232–242, 2019. ISSN 0893-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0893608018303277>>.

EMMERT-STREIB, F. et al. An introductory review of deep learning for prediction models with big data. **Frontiers in Artificial Intelligence**, Frontiers, v. 3, p. 4, 2020.

FU, H. et al. Visual cortex inspired cnn model for feature construction in text analysis. **Frontiers in computational neuroscience**, Frontiers, v. 10, p. 64, 2016.

GALE, T. **The Future of Sparsity in Deep Neural Networks**. 2020. Disponível em: <<https://www.sigarch.org/the-future-of-sparsity-in-deep-neural-networks/>>.

GANESH, P. **Types of Convolution Kernelsnbsp;; Simplified**. Towards Data Science, 2019. Disponível em: <<https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>>.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, v. 32, n. 14, p. 2627–2636, 1998. ISSN 1352-2310. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1352231097004470>>.

GROSSI, E.; BUSCEMA, M. Introduction to artificial neural networks. **European journal of gastroenterology hepatology**, v. 19, p. 1046–54, 01 2008.

HALALLI, B.; MAKANDAR, A. Computer aided diagnosis-medical image analysis techniques. **Breast Imaging**, BoD—Books on Demand, v. 85, 2018.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <<https://doi.org/10.1038/s41586-020-2649-2>>.

HERCULANO-HOUZEL, S. The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 109, n. Supplement 1, p. 10661–10668, 2012.

HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, World Scientific, v. 6, n. 02, p. 107–116, 1998.

HOCHREITER, S.; YOUNGER, A. S.; CONWELL, P. R. Learning to learn using gradient descent. In: DORFFNER, G.; BISCHOF, H.; HORNIK, K. (Ed.). **Artificial Neural Networks — ICANN 2001**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 87–94. ISBN 978-3-540-44668-2.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

JOHNSON, H. J.; MCCORMICK, M. M.; IBANEZ, L. [S.I.].

KLUYVER, T. et al. Jupyter notebooks ? a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCIMIDT, B. (Ed.). **Positioning and Power in Academic Publishing: Players, Agents and Agendas**. IOS Press, 2016. p. 87–90. Disponível em: <<https://eprints.soton.ac.uk/403913/>>.

KOEHRSEN, W. **Beyond Accuracy: Precision and Recall**. Towards Data Science, 2018. Disponível em: <<https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>>.

LAMBA, H. **Understanding Semantic Segmentation with UNET**. Towards Data Science, 2019. Disponível em: <<https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>>.

LAPIX. **Deep Learning::Segmentação con CNNs**. <<http://www.lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningsegmentacao-semantica/>>. Accessed: 2021-05-25.

LAROBINA, M.; MURINO, L. Medical image file formats. **Journal of Digital Imaging**, 2014.

LEDIG, C. et al. Photo-realistic single image super-resolution using a generative adversarial network. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.I.: s.n.], 2017. p. 4681–4690.

LI, X. et al. The first step for neuroimaging data analysis: Dicom to nifti conversion. **Journal of Neuroscience Methods**, v. 264, p. 47–56, 2016. ISSN 0165-0270. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0165027016300073>>.

MEHTA, V.; DEB, P.; SUBBA, R. D. Application of computer techniques in medicine. **Medical Journal Armed Forces India**, Elsevier, v. 50, n. 3, p. 215–218, 1994.

MITCHELL, T. M. et al. Machine learning. McGraw-hill New York, 1997.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. **arXiv preprint arXiv:1811.03378**, 2018.

ODENA, A.; DUMOULIN, V.; OLAH, C. Deconvolution and checkerboard artifacts. **Distill**, 2016. Disponível em: <<http://distill.pub/2016/deconv-checkerboard>>.

PANDEY, P. **Data Preprocessing: Concepts.** Towards Data Science, 2020. Disponível em: <<https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825>>.

PIANYKH, O. S. **Digital Imaging and Communications in Medicine (DI-COM): A Practical Introduction and Survival Guide.** 1st. ed. [S.I.]: Springer Publishing Company, Incorporated, 2010. ISBN 3642094007.

POKHREL, S. **Beginners Guide to Convolutional Neural Networks.** <<https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d>>. Accessed: 2021-05-25.

POTDAR, K.; PARDAWALA, T.; PAI, C. A comparative study of categorical variable encoding techniques for neural network classifiers. **International Journal of Computer Applications**, v. 175, p. 7–9, 10 2017.

REZA, A. M. Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. **Journal of VLSI signal processing systems for signal, image and video technology**, Springer, v. 38, n. 1, p. 35–44, 2004.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation.** 2015.

ROSEBROCK, A. **Keras ImageDataGenerator and Data Augmentation.** 2021. Disponível em: <<https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/>>.

ROSSUM, G. V.; JR, F. L. D. **Python tutorial.** [S.I.]: Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

SHARMA, A. et al. A patch-based convolutional neural network for remote sensing image classification. **Neural Networks**, v. 95, p. 19–28, 2017. ISSN 0893-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0893608017301806>>.

SHARMA, S.; SHARMA, S. Activation functions in neural networks. **Towards Data Science**, v. 6, n. 12, p. 310–316, 2017.

SHIRAISHI, J. et al. Computer-aided diagnosis and artificial intelligence in clinical imaging. In: ELSEVIER. **Seminars in nuclear medicine.** [S.I.], 2011. v. 41, n. 6, p. 449–462.

SHRIVASTAVA, A. **Underfitting Vs Just right Vs Overfitting in Machine learning.** <<https://www.kaggle.com/getting-started/166897>>. Accessed: 2021-05-25.

SYDORENKO, I. **What Is Data Labeling in Machine Learning? Label Your Data,** 2020. Disponível em: <<https://labelyourdata.com/articles/what-is-data-labeling-in-machine-learning/>>.

TU, L.; DONG, C. Histogram equalization and image feature matching. In: **2013 6th International Congress on Image and Signal Processing (CISP)**. [S.l.: s.n.], 2013. v. 01, p. 443–447.

WANG, Q. et al. A comprehensive survey of loss functions in machine learning. **Annals of Data Science**, Springer, p. 1–26, 2020.

WEISSTEIN, E. W. Heaviside step function. <https://mathworld.wolfram.com/>, Wolfram Research, Inc., 2002.

XU, Y. et al. **Scale-Invariant Convolutional Neural Networks**. 2014.

YI, X.; WALIA, E.; BABYN, P. Generative adversarial network in medical imaging: A review. **Medical image analysis**, Elsevier, v. 58, p. 101552, 2019.

YING, X. An overview of overfitting and its solutions. **Journal of Physics: Conference Series**, IOP Publishing, v. 1168, p. 022022, feb 2019. Disponível em: <<https://doi.org/10.1088/1742-6596/1168/2/022022>>.

YU, X.-H.; CHEN, G.-A. Efficient backpropagation learning using optimal learning rate and momentum. **Neural Networks**, v. 10, n. 3, p. 517–527, 1997. ISSN 0893-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0893608096001025>>.

YUNG, J. **Explaining Tensorflow Code for a Multilayer Perceptron**. <<https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/>>. Accessed: 2021-05-25.