|  |
| --- |
| uNIVERSIDADE REGIONAL DE BLUMENAU  CENTRO DE CIÊNCIAS EXATAS E NATURAIS  CURsO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO |
| modelo de aprendizado de máquina para a contagem e identificação de neurÔnios atípicos  gabriel krzizanowski  Lucas Eduardo de carvalho |
| bLUMENAU  2024 |

|  |
| --- |
| gabriel krzizanowski  Lucas Eduardo de carvalho  Modelo de aprendizado de máquina para a contagem e identificação de neurÔnios atípicos  Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.  Prof. Aurélio Faustino Hoppe - Orientador |
| bLUMENAU  2024 |
|  |
|  |

Dedicamos este trabalho à nossas famílias, e a todos os nossos colegas e amigos que nos apoiaram e auxiliaram em todo o processo de graduação.

AGRADECIMENTOS

Às nossas famílias que sempre nos apoiaram em nossas decisões.

Aos nossos amigos, que de certa forma nos ajudaram nesses últimos anos.

Às nossas namoradas, pelo carinho, paciência e compreensão em todos os momentos.

Ao nosso orientador, professor Aurélio Faustino Hoppe, pelo apoio e contribuição na confecção desse trabalho.

À professora Cláudia Almeida Coelho de Albuquerque e aos acadêmicos do curso de medicina, Gabriel e Tales, pelo apoio e dedicação na obtenção das amostras para análise.

A grande conquista é o resultado de pequenas vitórias que passam despercebidas.

Paulo Coelho

RESUMO

A identificação e contagem de neurônios atípicos representam desafios significativos para pesquisadores da neurociência, especialmente em análises manuais realizadas por meio de microscópios. Esses procedimentos são altamente dependentes da habilidade humana, sujeitos a erros e demandam tempo considerável, o que dificulta a obtenção de resultados consistentes e que pode acarretar diagnósticos falhos sobre doenças mentais, como a doença de Alzheimer. A partir disso, este trabalho apresenta o desenvolvimento de um modelo de aprendizado de máquina para a identificação e contagem de neurônios atípicos, visando automatizar e otimizar o processo que, tradicionalmente, demanda um tempo elevado e está sujeito a erros humanos. Para tanto, utilizou-se a rede neural Faster R-CNN, empregando técnicas de processamento de imagens e aprendizado de máquina. As imagens utilizadas foram obtidas por meio de microscópios e celulares, organizadas em um *dataset* criado na plataforma LabelBox, onde realizou-se o processo de anotação manual. Durante o processo de treinamento, aplicaram-se técnicas de aumento de dados para ampliar a diversidade do *dataset* e melhorar a generalização do modelo. A validação foi realizada por meio de métricas como F1-Score, precisão e revocação, mostrando desempenho consistente. Como resultado, o modelo alcançou uma acurácia média satisfatória para imagens de teste, evidenciando a viabilidade da solução proposta. O modelo também foi integrado a uma aplicação móvel, desenvolvida em Flutter, que permite ao usuário realizar análises de imagens, salvar resultados em repositórios e gerenciar dados de forma intuitiva. Concluiu-se que a aplicação do aprendizado de máquina para a identificação de neurônios atípicos representa uma contribuição relevante para a neurociência, com potencial para aprimorar o entendimento das alterações neuronais e facilitar a rotina de pesquisadores e estudantes.

Palavras-chave: Neurônios atípicos. Identificação. Aprendizado de máquina. Faster R-CNN. Aplicativo móvel.

ABSTRACT

The identification and counting of atypical neurons pose significant challenges for neuroscience researchers, particularly in manual analyses performed via microscopes. These procedures are highly dependent on human skill, prone to errors, and time-consuming, which hinders the consistency of results and may lead to inaccurate diagnoses of mental disorders, such as Alzheimer's disease. To address these issues, this study presents the development of a machine learning model for the identification and counting of atypical neurons, aiming to automate and optimize a traditionally time-intensive and error-prone process. The Faster R-CNN neural network was employed, leveraging image processing and machine learning techniques. The images were obtained through microscopes and mobile cameras, organized into a dataset created on the LabelBox platform, where manual annotation was conducted. During training, data augmentation techniques were applied to enhance dataset diversity and improve model generalization. Validation was performed using metrics such as F1-Score, precision, and recall, demonstrating consistent performance. The model achieved satisfactory accuracy on test images, highlighting the feasibility of the proposed solution. Additionally, the model was integrated into a mobile application developed in Flutter, enabling users to analyze images, save results in repositories, and manage data intuitively. It is concluded that applying machine learning to the identification of atypical neurons represents a significant contribution to neuroscience, with the potential to enhance understanding of neuronal alterations and streamline the routines of researchers and students.

Key-words: Atypical neurons. Identification. Machine learning. Faster R-CNN. Mobile application.

LISTA DE Figuras

[Figura 1 – Localização, estrutura e divisão do hipocampo no cérebro. 15](#_Toc184283845)

[Figura 2 – Representação do neurônio 17](#_Toc184283846)

[Figura 3 – Estrutura interna de um neurônio típico. 18](#_Toc184283847)

[Figura 4 – Neurônio sofrendo morte celular devido a doença de Alzheimer. 19](#_Toc184283848)

[Figura 5 – Arquitetura da Mask-RCNN 20](#_Toc184283849)

[Figura 6 – Fast R-CNN 20](#_Toc184283850)

[Figura 7 – Esquema simplificado do método Faster R-CNN. 21](#_Toc184283851)

[Figura 8 – Etapas definidas para detecção de neurônios atípicos. 27](#_Toc184283852)

[Figura 9 – Processo de captura das imagens. 28](#_Toc184283853)

[Figura 10 – Exemplo do resultado de *labeling* da ferramenta Labelbox 28](#_Toc184283854)

[Figura 11 – Visualização do arquivo JSON. 29](#_Toc184283855)

[Figura 12 – Imagens com *data* *augmentation* com suas rotações. 30](#_Toc184283856)

[Figura 13 – Gráfico de F1-Score, precisão e revocação do modelo. 36](#_Toc184283857)

[Figura 14 – Evolução da *Loss* ao longo dos *Steps* por *Epoch*. 36](#_Toc184283858)

[Figura 15 – Acurácia de Treinamento e Validação do modelo. 37](#_Toc184283859)

[Figura 16 – Resultado da predição do modelo. 40](#_Toc184283860)

[Figura 17 – Porcentagem de acerto por imagem. 41](#_Toc184283861)

[Figura 18 – Diagrama de caso de uso da aplicação móvel 42](#_Toc184283862)

[Figura 19 – Arquitetura da aplicação móvel 44](#_Toc184283863)

[Figura 20 – Imagem original e imagem retornada pelo método save\_images. 49](#_Toc184283864)

[Figura 21 – Telas iniciais da aplicação móvel 50](#_Toc184283865)

[Figura 22 – Tela com análise de uma imagem dos neurônios do hipocampo 51](#_Toc184283866)

[Figura 23 – Telas para a salvar imagem 52](#_Toc184283867)

[Figura 24 – Telas de listagem de informações 52](#_Toc184283868)

LISTA DE Quadros

[Quadro 1 – Resultado de buscas por termos contidos em artigos 23](#_Toc184283869)

[Quadro 2 – Comparação de arquiteturas de Deep Learning para segmentação de imagens dermatoscópicas de melanoma. 23](#_Toc184283870)

[Quadro 3 – Análise ultra-estrutural na morte celular da área CA1 do hipocampo após isquemia cerebral global transitória em ratos Wistar 23](#_Toc184283871)

[Quadro 4 – Diagnóstico da doença de Alzheimer usando autoencoders aplicados a imagens de ressonância magnética 24](#_Toc184283872)

[Quadro 5 – Dinâmica da Plasticidade Sináptica em neurônios do Hipocampo durante ciclos de sono: um estudo computacional 24](#_Toc184283873)

[Quadro 6 – Trecho de código para criação das pastas de salvamento dos dados 29](#_Toc184283874)

[Quadro 7 – Código de multiplicação das imagens com suas anotações 30](#_Toc184283875)

[Quadro 8 – Código de separação de anotação de imagem 31](#_Toc184283876)

[Quadro 9 – Função main do *script* TrainingScript 32](#_Toc184283877)

[Quadro 10 – Divisão de conjuntos e inicialização do modelo 32](#_Toc184283878)

[Quadro 11 – Função get\_model do *script* 33](#_Toc184283879)

[Quadro 12 – Função get\_tranform sendo utilizado para a parte do *dataset*. 33](#_Toc184283880)

[Quadro 13 – Função collate\_fn com a finalidade de empacotar os dados 33](#_Toc184283881)

[Quadro 14 – Código da função calculate\_metrics. 34](#_Toc184283882)

[Quadro 15 – Salvamento do modelo em arquivo .pth. 35](#_Toc184283883)

[Quadro 16 – Demonstração de resultados pela Faster R-CNN. 37](#_Toc184283884)

[Quadro 17 – Importação do modelo. 38](#_Toc184283885)

[Quadro 18 – Função get\_model do script. 39](#_Toc184283886)

[Quadro 19 – Trecho de código com o envio da imagem para predição. 39](#_Toc184283887)

[Quadro 20 – Aplicação do limiar de confiança 40](#_Toc184283888)

[Quadro 21 – Dados após análise das dez imagens 41](#_Toc184283889)

[Quadro 22 – Código fonte da rota /process\_image 47](#_Toc184283890)

[Quadro 23 – Código fonte responsável pela predição 48](#_Toc184283891)

[Quadro 24 – Perfil dos usuários do teste de usabilidade 53](#_Toc184283892)

Lista de tabelas

[Tabela 1 – Respostas da execução da lista de tarefas 54](#_Toc184283893)

[Tabela 2 – Resultados do questionário de avalição da aplicação 55](#_Toc184283894)

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

CPU – Central Processing Unit

DA – Doença de Alzheimer

FURB – Fundação Universidade Regional de Blumenau

GPU – Graphics Processing Unit

JSON – JavaScript Object Notation

M – Molecular

NoSQL – Not Only SQL

REST – REpresentational State Transfer

RF – Requisito Funcional

RNF – Requisito Não Funcional

ROI - Region Of Interest

SNC – Sistema Nervoso Central

SNP – Sistema Nervoso Periférico

P – Piramidal

PL – PoLimórfica

UC – Use case

SUMÁRIO

[1 Introdução 13](#_Toc184283895)

[1.1 OBJETIVOS 14](#_Toc184283896)

[1.2 estrutura 14](#_Toc184283897)

[2 FUNDAMENTAÇÃO TEÓRICA 15](#_Toc184283898)

[2.1 Hipocampo 15](#_Toc184283899)

[2.2 Neurônios 16](#_Toc184283900)

[2.3 r-cnn, fast r-cnn e faster r-cnn 19](#_Toc184283901)

[2.4 trabalhos correlatos 22](#_Toc184283902)

[3 DESENVOLVIMENTO 26](#_Toc184283903)

[3.1 MODELO de identificação de neurônios atípicos 26](#_Toc184283904)

[3.1.1 Requisitos 26](#_Toc184283905)

[3.1.2 Descrição do modelo 26](#_Toc184283906)

[3.2 Aplicativo Mobile 42](#_Toc184283907)

[3.2.1 Diagrama de caso de uso 42](#_Toc184283908)

[3.2.2 Requisitos 43](#_Toc184283909)

[3.2.3 Arquitetura da solução 43](#_Toc184283910)

[3.2.4 Testes de usabilidade 53](#_Toc184283911)

[4 CONCLUSÕES 57](#_Toc184283912)

[Referências 60](#_Toc184283913)

# Introdução

Segundo Amaral e Lavenex (2007), o hipocampo desempenha um papel crucial na formação de memórias episódicas, que são aquelas associadas a eventos específicos e contextualmente ricas. Além disso, estudos têm demonstrado sua importância na memória espacial, permitindo que os indivíduos se localizem em seu ambiente e naveguem por ele de forma eficaz (Maguire *et al*., 2000).

Os neurônios do hipocampo compreendem uma variedade de tipos celulares que desempenham papéis essenciais na função cognitiva, especialmente na formação e recuperação da memória. Entre os principais tipos de neurônios no hipocampo, destacam-se os neurônios piramidais, os neurônios granulares e os interneurônios (Amaral; Lavenex, 2007).

Entre os desafios que a neuroanatomia tem enfrentado desde seu início, está a identificação de características que permitam distinguir diversos tipos de células neuronais do ponto de vista morfológico, tem atraído atenção de pesquisadores pertencentes a diferentes áreas, os quais tentam caracterizar quantitativamente os neurônios. A caracterização morfológica de neurônios, portanto, permite fazer análises com a finalidade de atingir um dos principais objetivos da neurociência: entender o funcionamento do cérebro (Cervantes, 2019).

Entre os desafios inerentes ao procedimento de contagem manual de neurônios hipocampais, destaca-se a considerável exigência de tempo demandada para a realização da quantificação neuronal em uma única imagem, requerendo um investimento substancial de várias horas para a sua conclusão. Além disso, há a necessidade de um nível significativo de familiaridade com as características distintivas dos neurônios tanto típicos quanto atípicos, a fim de sua correta identificação durante o processo de contagem. Outro aspecto relevante diz respeito a confiabilidade dos resultados obtidos, uma vez que a contagem manual está sujeita a possibilidade de variações decorrentes de potenciais erros humanos, os quais podem influenciar a precisão e a consistência dos dados obtidos.

Neste contexto, alunos dos cursos de Biologia, Biomedicina e Medicina, da Universidade Regional de Blumenau (FURB), realizam pesquisas e análises dos neurônios hipocampais, verificando suas alterações após seus indivíduos sofrerem algum tipo de estresse fisiológico ou patológico através do meio em que estão inseridos. A contagem desses neurônios é feita de forma manual, no qual se utiliza a coloração de Nissl, uma técnica de histologia, e, com a ajuda de um microscópio, é realizada a análise da imagem. A contagem é feita utilizando uma metodologia chamada de “método estereológico”, na qual a imagem é dividida em quadrantes ou áreas de interesse, o observador registra o número de neurônios em cada quadrante, garantindo que cada neurônio seja contado apenas uma vez, sendo atribuído ao quadrante apropriado.

A partir desse contexto, este trabalho apresenta o desenvolvimento de um aplicativo móvel para a identificação e marcação de neurônios atípicos utilizando métodos e técnicas de processamento da imagem e aprendizado de máquina.

## OBJETIVOS

O objetivo principal deste trabalho consiste em disponibilizar um aplicativo que seja capaz de realizar a contagem de neurônios atípicos de forma automática, por meio de uma foto tirada pelo celular, utilizando técnicas de processamento de imagens e aprendizado de máquina.

Os objetivos específicos são:

1. efetuar a segmentação dos neurônios existentes em uma lâmina;
2. realizar a extração de características morfológicas dos neurônios;
3. classificar os neurônios atípicos utilizando redes neurais convolucionais;
4. validar e analisar o tempo de resposta da segmentação e extração de medidas morfométricas dos neurônicos assim como sua assertividade e quantidade em relação ao processo manual.

## estrutura

A estrutura do trabalho está divido em quatro capítulos. No primeiro capítulo está descrita a introdução do trabalho juntamente com os objetivos. No segundo capítulo está detalhada a fundamentação teórica, contendo os principais tópicos e os trabalhos correlatos encontrados. O terceiro capítulo descreve o desenvolvimento da ferramenta, contendo as especificações funcionais e não funcionais da aplicação, técnicas e bibliotecas utilizadas em cada etapa, implementação do protótipo e apresentação dos resultados encontrados. Por fim, o quarto e último capítulo descreve a conclusão do trabalho, alinhando as expectativas e resultados, juntamente com as limitações e sugestões de trabalhos futuros.

# FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma visão geral dos tópicos que servem como base para o estudo a ser desenvolvido. A seção 2.1 discute o hipocampo, enquanto a seção 2.2 aborda os neurônios. A seção 2.3 trata dos modelos R-CNN e Faster R-CNN. Por último, a seção 2.4 apresenta uma revisão dos trabalhos correlatos, que são considerados similares ao projeto proposto.

## Hipocampo

De acordo com Burgess, Maguire e O`Keefe (2002), o hipocampo é uma das poucas áreas do cérebro adulto na qual ocorre a neurogenese, surgimento de novos neurônios, também chamados de células nervosas por estarem relacionados ao impulso nervoso, são células excitáveis, capazes de gerar, conduzir e receber estímulos como uma unidade funcional e morfológica nervosa.

Os neurônios do hipocampo (também chamado de arquicórtex) estão dispostos em três camadas celulares, as quais são denominadas: Molecular (M), Piramidal (P) e PoLimórfica (PL) (Aljarari, 2023). A Figura 1 apresenta a estrutura e divisão do hipocampo no cérebro.

Figura – Localização, estrutura e divisão do hipocampo no cérebro.Diagrama

Descrição gerada automaticamenteFonte: Akter *et al*. (2024).

Adicionalmente, Yu *et al.* (2020) ainda destaca que estudos recentes têm demonstrado que o hipocampo não está apenas envolvido na formação e recuperação de memórias, mas também desempenha um papel significativo na regulação do humor e das emoções. A interação complexa entre os neurônios do hipocampo e outras regiões do cérebro, como a amígdala e o córtex pré-frontal, é essencial para a modulação adequada das respostas emocionais e a adaptação a diferentes situações. Essa conexão entre processamento cognitivo e emocional reafirma a importância do hipocampo não apenas na cognição, mas também na regulação do bem-estar emocional e mental.

## Neurônios

Moreira (2013) descreve um neurônio como uma célula nervosa, estrutura básica do sistema nervoso, comum à maioria dos vertebrados. Os neurônios são altamente estimuláveis, que processam e transmitem informação através de sinais eletroquímicos. Uma de suas características é a capacidade das suas membranas plasmáticas gerarem impulsos nervosos.

Lent (2010) destaca que, os neurônios são células especializadas em processar informações do ambiente externo e interno, assim como nossos pensamentos internos. Ainda segundo o autor, tipicamente, existem quatro tipos básicos de células funcionais: (i) os neurônios sensoriais, que informam o cérebro sobre os eventos percebidos no ambiente interno e externo; (ii) os neurônios motores, que contratam os músculos e são responsáveis pelos comportamentos motores; (iii) os neurônios de comunicação, que transmitem sinais de uma área para outra do cérebro; e (iv) os neurônios de associação (interneurônios), que extraem e processam informações sensoriais, comparando-as com as armazenadas na memória, utilizadas no planejamento e execução de comportamentos destacando neurônios como típicos.

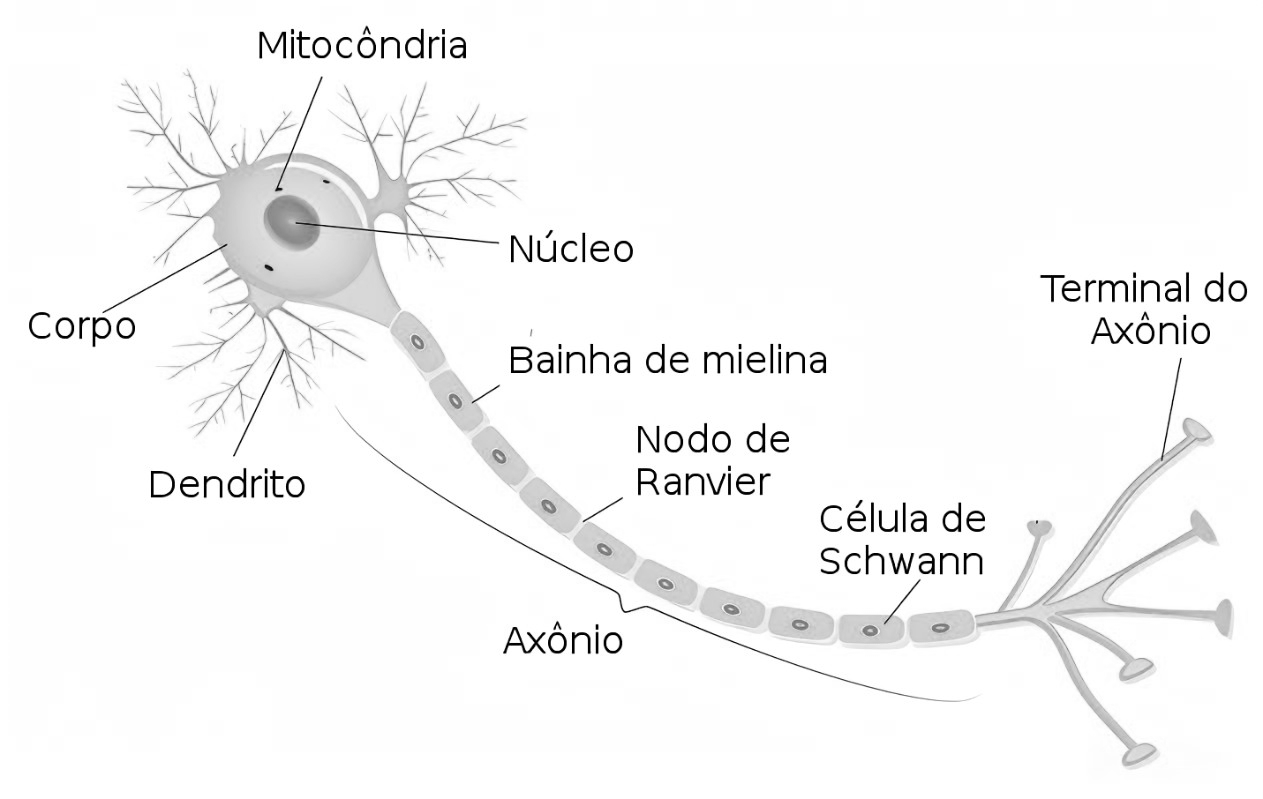
De acordo com Yu *et al.* (2020), além dos quatro tipos básicos de neurônios destacados por Lent (2010), é importante ressaltar também a presença de células da glia, que desempenham um papel fundamental no suporte e na manutenção do ambiente neural. Essas células auxiliam na nutrição, na remoção de resíduos e na modulação da atividade neuronal, contribuindo para a saúde e o funcionamento adequado do sistema nervoso. Entre os diferentes tipos de células da glia, destacam-se os astrócitos, que desempenham um papel crucial na regulação da neurotransmissão e na manutenção da homeostase cerebral.

Para Machado (2013), os neurônios são células altamente excitáveis que se comunicam entre si ou com células efetuadoras (células musculares e secretoras), usando basicamente uma linguagem elétrica, qual seja, modificações do potencial de membrana. A maioria dos neurônios possui três regiões responsáveis por funções especializadas: corpo celular, dendritos e axônio, conforme ilustra a Figura 2.

Moreira (2013) descreve cada região do neurônio como:

1. Corpo celular: contém o núcleo e a maior parte das organelas. É nesta parte onde ocorre a síntese proteica;
2. Dendritos: são prolongamentos finos, geralmente ramificados, que recebem e conduzem os estímulos provenientes de outros neurônios ou de células sensoriais;
3. Axónio: é o prolongamento, geralmente, mais longo que transmite os impulsos nervosos provenientes do corpo celular. O comprimento axónio varia muito entre os diferentes tipos de neurônios. Nos vertebrados e em alguns invertebrados os axónios são cobertos por uma bainha isolante de mielina, tomando a designação de fibra nervosa;
4. Terminações do axónio: contêm sinapses, estruturas especializadas onde são liberadas substâncias químicas, neurotransmissores, que estabelecem a comunicação com as dendritos ou corpo celular de outros neurônios.

Figura – Representação do neurônio



Fonte: Valeso (2021).

Segundo Montanari (2006), o núcleo é grande, esférico ou ovoide e claro, por causa da cromatina frouxa, com um e, às vezes, dois ou três núcleos.

O corpo de um neurônio típico (saudável) possui aproximadamente 20μm de diâmetro. O fluido aquoso no interior da célula, chamado de citosol, é uma solução salina rica em potássio, separada no meio externo pela membrana plasmática. O corpo celular de um neurônio contém as mesmas organelas encontradas nas demais células animais. As mais importantes são o reticulo endoplasmático rugoso, Golgi e as mitocôndrias, representados na Figura 3.

Figura – Estrutura interna de um neurônio típico.

Diagrama

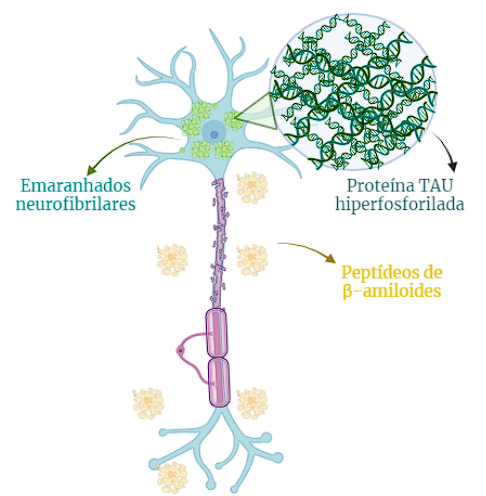
Descrição gerada automaticamente

Fonte: Machado (2013).

De acordo com Ma *et al*. (2011), as células, após sofrerem estresses, os quais seus sistemas de reparo não conseguem corrigir e propiciar manutenção da contiguidade celular, podem induzir apoptose, isto é, sua morte programada. Assim, neurônios em apoptose podem apresentar-se nas lâminas com núcleo picnótico (com sua cromatina mais condensada) em forma de corpúsculo heterocromático. As organelas podem ser abarcadas pelo sistema de endomembranas da célula, formando vacuolizações citoplasmáticas (Aljarari, 2023). Em células nas quais a morte é iminente, há a formação de um espaço pericelular, ou seja, um disco branco ao redor da unidade celular.

Segundo Alves (2022), a Doença de Alzheimer (DA), principal causa de demência em idosos, tem sua patogênese associada ao acúmulo de placas de beta-amiloide e emaranhados neurofibrilares de proteína Tau hiperfosforilada. Esses agregados resultam da clivagem da proteína precursora amiloide pelas enzimas BACE e γ-secretase, desencadeando uma resposta inflamatória mediada pela micróglia e astrócitos reativos, que promove neuroinflamação, disfunção sináptica, morte neuronal, perda de volume cerebral e alterações estruturais no hipocampo e córtex, culminando em declínio cognitivo (Figura 4).

Figura – Neurônio sofrendo morte celular devido a doença de Alzheimer.



Fonte: Matano, Pacheco e Zemlenoi (2023).

A partir dessas análises, torna-se possível estabelecer uma distinção clara entre neurônios saudáveis (típicos) e aqueles que apresentam sinais de comprometimento ou morte celular (atípicos). Essa diferenciação é fundamental, pois permite inferir de maneira mais precisa o impacto dessas alterações na saúde do indivíduo. Neurônios saudáveis exibem uma morfologia preservada, incluindo membranas plasmáticas intactas, núcleo claro com cromatina frouxa, e organelas funcionais distribuídas no citoplasma. Em contraste, neurônios atípicos, frequentemente associados a condições patológicas, apresentam características como núcleo picnótico com cromatina condensada, vacuolização citoplasmática e sinais de apoptose, como espaços pericelulares evidentes. Dessa forma, a identificação dessas diferenças estruturais e funcionais fornece subsídios essenciais para a compreensão do estado de saúde neuronal e sua correlação com doenças neurodegenerativas ou outros distúrbios.

## r-cnn, fast r-cnn e faster r-cnn

Segundo He *et al*. (2017) a Mask R-CNN é uma rede neural convolucional generalista para detectar objetos em uma imagem e gerar máscaras de segmentação de alta qualidade para cada objeto encontrado. O método, chamado de Mask R-CNN estende da Faster R-CNN adicionando uma camada para predição de uma máscara em cada *Region Of Interest* (ROI), trabalhando em paralelo com as camadas já existentes da Faster R-CNN, que são para a classificação e regressão da *boundig box*. Essa nova camada de máscara é uma pequena rede convolucional que gera uma máscara de segmentação para cada ROI. Essa arquitetura pode ser visualizada na Figura 5.

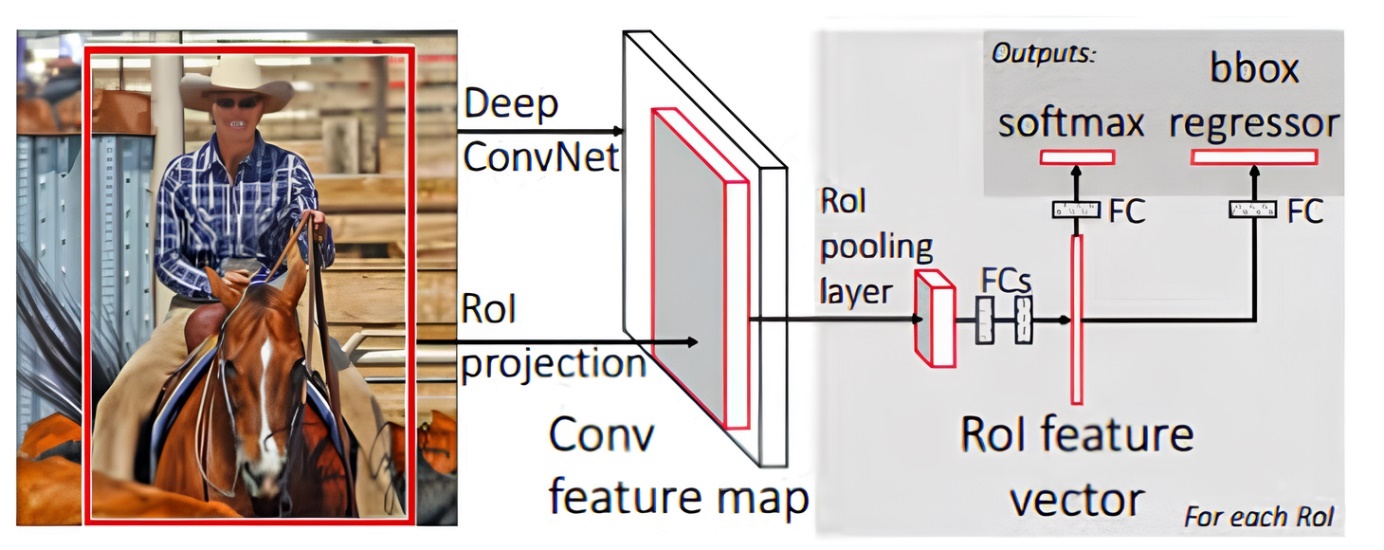
Figura – Arquitetura da Mask-RCNN

|  |
| --- |
| Diagrama  Descrição gerada automaticamente |

Fonte: Girshick et al. (2020).

A ideia por trás dessa abordagem é similar àquela que originou a R-CNN. A diferença é que, em vez de fornecer diretamente as regiões à CNN, agora a CNN recebe a imagem inteira como entrada e gera um mapa convolucional de características. É nesse mapa que as regiões propostas são identificadas e transformadas, utilizando o método de Region of Interest (ROI), o que permite padronizar essas regiões para um tamanho fixo antes de passá-las para a camada totalmente conectada. Ao final do processo, uma camada *softmax* é usada para prever a classe de cada região proposta, bem como os valores de ajuste da caixa delimitadora. A Figura 6 ilustra essa nova abordagem.

Figura – Fast R-CNN



Fonte: Girshick et al. (2020).

Esta abordagem é mais rápida que a abordagem anterior, uma vez que não é passado um input de 2000 regiões, mas sim apenas uma imagem que irá gerar um mapa de atributos. Além disso, ao reutilizar as características extraídas pela CNN para todas as regiões propostas, a Fast R-CNN reduz significativamente a redundância computacional, aumentando a eficiência sem comprometer a precisão. Essa evolução representa um avanço importante no desempenho de modelos de detecção de objetos, especialmente em aplicações que exigem alta velocidade de processamento.

Conforme destacado por Timóteo *et al*. (2020), há uma evolução do modelo Fast R-CNN, a Faster R-CNN, entre outros modelos baseados em CNN, vieram a substituir os modelos tradicionais na área da visão computacional. Trata-se de um modelo do estado da arte de detecção de objetos e tem como base as redes neuronais convolucionais. O modelo tem a versatilidade de detectar múltiplos objetos, bem como informar suas localizações na imagem de entrada. Os modelos de reconhecimento de objetos do esta do-da-arte, no qual inclui a rede Faster R-CNN, utilizam como “espinha dorsal” uma rede CNN. Diferentemente das redes Neuronais tradicioNais (NN), onde a quantidade de camadas e de neurónios (estrutura) são definidos de acordo com o tipo de problema a ser solucionado, as CNNs geralmente têm as suas estruturas fixas e são aplicadas para resolver diversos problemas diferentes, apesar de as estruturas dessas redes puderem ser modificadas (customizadas). Existem diversos tipos de CNNs, como por exemplo a VGG16 Liu. (2015), Resnet He et al. (2016), Inception Szegedy et al. (2016), entre outras. A Figura 7 mostra um esquema simplificado do método Faster R-CNN.

Figura – Esquema simplificado do método Faster R-CNN.

Diagrama, Desenho técnico

Descrição gerada automaticamente

Fonte: adaptado de Ren *et al*. (2016).

No modelo Faster R-CNN, as saídas das camadas de convolução são denominadas de “mapas de características”. Numa segunda fase, utiliza-se a última camada de convolução como entrada da “Rede de proposta de regiões”, no qual produzem as regiões de interesses (em inglês Region Of Interest – ROIs) que contém os possíveis objetos. Essas ROIs são utilizadas na última fase do modelo de deteção para classificação dos objetos, bem como para um possível ajuste de posição do enquadramento da deteção do objeto na imagem de entrada.

## trabalhos correlatos

Conforme destacado por Casarin *et al*. (2020), a revisão de literatura constitui uma abordagem analítica que visa sintetizar resultados previamente encontrados ou mapear o estado da arte acerca do tema em questão. Esse processo pode ser classificado em duas categorias principais: Revisão Sistemática da Literatura (RSL) e Revisão Tradicional da Literatura (RTL). A revisão sistemática, frequentemente utilizada em práticas baseadas em evidências, é um método estruturado que busca sintetizar os achados de pesquisas relacionadas a uma problemática específica (Galvão *et al*., 2004). Por sua vez, a revisão bibliográfica tradicional, também denominada revisão narrativa, utiliza métodos específicos para explorar tópicos relevantes em acervos acadêmicos (Botelho; Cunha; Macedo, 2011).

Para elaboração dessa pesquisa, adotou-se a abordagem de Revisão Sistemática da Literatura (RSL), conduzida em plataformas de acesso acadêmico com o foco em análises relacionadas a neurônios do hipocampo, típicos e atípicos, e a aplicação de técnicas de *deep learning*. Para isso, o Google Acadêmico foi selecionado como principal ferramenta de busca, e optou-se por priorizar publicações realizadas a partir do ano de 2003.

O filtro escolhido para pesquisa foi montado da seguinte maneira: "Deep learning" AND ("neurônios típicos" OR "neurônios atípicos") AND "hipocampo", "Identificação " AND ("neurônios normais" OR "neurônios não convencionais") AND "hipocampo", "Redes neurais" AND "região hipocampal", "Neurônios hipocampais" AND ("ratos" OR "contagem") AND "hipocampo", "artificial" AND ("contagem" OR "separação") AND "hipocampo" AND "neurônio". O Quadro 1 apresenta os resultados obtidos a partir da aplicação desses filtros no Google Acadêmico, demonstrando a relevância e a abrangência dos trabalhos encontrados.

Quadro – Resultado de buscas por termos contidos em artigos

|  |  |
| --- | --- |
| **Termos de busca** | **Google Acadêmico** |
| "Deep learning" AND ("neurônios típicos" OR "neurônios atípicos") AND "hipocampo" | 12 |
| "Identificação " AND ("neurônios normais" OR "neurônios não convencionais") AND "hipocampo" | 14 |
| "Redes neurais" AND "região hipocampal" | 73 |
| "Neurônios hipocampais" AND ("ratos" OR "contagem") AND "hipocampo" | 605 |
| "artificial" AND ("contagem" OR "separação") AND "hipocampo" AND "neurônio" | 322 |
| **Total** | **1.026** |

Fonte: elaborado pelos autores.

Inicialmente, realizou-se uma triagem para reduzir o número de artigos de mais de 1.000 para cerca de 50, utilizando critérios como a relevância das palavras-chave em relação ao foco deste trabalho. Após essa etapa, observou-se um volume conjunto de 10 artigos que demonstram maior pertinência para a análise. Foram priorizados trabalhos que exploravam o uso de técnicas de aprendizado de máquina, em especial Deep Learning, para a análise de neurônios do hipocampo, típicos e atípicos, com explicações detalhadas sobre as metodologias aplicadas. Além disso, buscou-se selecionar artigos que enfatizassem o uso dessas técnicas no contexto de dados experimentais, como contagem de neurônios, separação entre padrões normais e não convencionais, ou análises relacionadas à região hipocampal de modelos experimentais. Os Quadros 2, 3, 4 e 5 exibem os artigos selecionados e seus detalhamentos.

Quadro – Comparação de arquiteturas de Deep Learning para segmentação de imagens dermatoscópicas de melanoma

|  |  |
| --- | --- |
| Referência | Silva Júnior, Bezerra e Andrade (2020) |
| Objetivos | Comparar técnicas de aprendizado profundo para segmentação de melanoma em imagens dermatoscópicas. |
| Principais funcionalidades | Implementação de arquiteturas U-Net com DenseNet-121, ResNet-50 e VGG-19, avaliação com aumento de dados em datasets ISIC 2017 e PH2. |
| Ferramentas de desenvolvimento | Google Colab Pro, TensorFlow, Keras, bibliotecas de Python para processamento de imagens. |
| Resultados e conclusões | A U-Net com DenseNet-121 obteve o melhor desempenho (Índice Jaccard de 83,64%) no dataset ISIC 2017 com aumento de dados, demonstrando a eficácia do aumento de dados e de arquiteturas modernas para segmentação de imagens médicas. |

Fonte: elaborado pelos autores.

Quadro – Análise ultra-estrutural na morte celular da área CA1 do hipocampo após isquemia cerebral global transitória em ratos Wistar

|  |  |
| --- | --- |
| Referência | Winkelmann (2003). |
| Objetivos | Realizar uma análise ultra-estrutural da morte celular na área CA1 do hipocampo após isquemia cerebral global transitória em ratos Wistar machos adultos. |
| Principais funcionalidades | Utilização de microscopia eletrônica para observar as alterações celulares ultra-estruturais na área CA1 do hipocampo. Identificação e classificação das fases de degeneração celular em três fases distintas: inicial, intermediária e final. |
| Ferramentas de desenvolvimento | Vibrátomo e ultra-micrótomo para preparação das amostras. Glutaraldeído, paraformaldeído, tetróxido de ósmio, acetato de uranila e citrato de chumbo foram utilizados para fixação e contraste das amostras. |
| Resultados e conclusões | A morte celular na área CA1 do hipocampo após isquemia cerebral global transitória segue predominantemente um padrão de necrose oncótica, com algumas características iniciais de apoptose, mas sem evidências de apoptose avançada. |

Fonte: elaborado pelos autores.

Quadro – Diagnóstico da doença de Alzheimer usando autoencoders aplicados a imagens de ressonância magnética

|  |  |
| --- | --- |
| Referência | Grivol (2022). |
| Objetivos | A classificação de imagens bidimensionais de ressonância magnética (RM) para diagnóstico da doença de Alzheimer (DA) utilizando *autoencoders* para extração de características e o classificador XGBoost. O estudo visou diferenciar entre classes Cognitivamente Normal (CN) e DA. |
| Principais funcionalidades | Utilização do classificador XGBoost para a classificação das imagens nas classes CN e DA. Especialização dos *autoencoders* por faixa etária para melhorar a precisão do diagnóstico em diferentes grupos etários. |
| Ferramentas de desenvolvimento | Python, TensorFlow, Keras, XGBoost. Algoritmos Non-Local Means (NLM) e N4ITK para redução de ruído e correção de *bias field.* técnicas de aumento de dados, incluindo espelhamento, adição de ruído térmico, rotação, translação e escala. |
| Resultados e conclusões | *Autoencoders* especializados na classe CN são eficazes na extração de características relevantes para a classificação de imagens de RM nas classes CN e DA. A estratificação por idade é fundamental para melhorar a precisão do diagnóstico, considerando as mudanças estruturais naturais do cérebro com o envelhecimento. |

Fonte: elaborado pelos autores.

Quadro – Dinâmica da Plasticidade Sináptica em neurônios do Hipocampo durante ciclos de sono: um estudo computacional

|  |  |
| --- | --- |
| Referência | Figuerola (2012). |
| Objetivos | Desenvolver uma simulação computacional para quantificar a evolução da rede neural biológica durante os ciclos de sono em ratos. |
| Principais funcionalidades | Implementação de uma ANN para simular a dinâmica sináptica durante os ciclos de sono. Simulação de ciclos de sono-vigília, incluindo transições entre SWS e REM. Utilização de dados reais de disparos neurais registrados no hipocampo de ratos para alimentar a ANN. |
| Ferramentas de desenvolvimento | MATLAB para desenvolvimento do código, utilizando programação orientada a objetos. Processador de aquisição múltiplos-neurônios (MNAP) e software Plexon para registro e análise de dados neurais. |
| Resultados e conclusões | Presença de plasticidade de longo prazo durante o sono pode alterar significativamente a configuração sináptica, sugerindo que o sono não apenas previne o esquecimento, mas também facilita a formação de novas memórias e comportamentos. |

Fonte: elaborado pelos autores.

Os trabalhos apresentados nos quadros foram selecionados pois abordam uma variedade de aspectos relacionados ao hipocampo, desde sua estrutura celular até sua função em processos patológicos e cognitivos, bem como por utilizarem ou descreverem a aplicação de técnicas de aprendizado de máquina em imagens médicas. O estudo de Silva Júnior, Bezerra e Andrade (2020) investiga a neurogênese e a morfologia dos neurônios hipocampais em diferentes condições experimentais, enquanto Winkelmann (2003) foca na análise ultraestrutural da morte celular após isquemia cerebral. Em contraste, Grivol (2022) explora o uso de redes neurais na identificação da doença de Alzheimer por meio de imagens de ressonância magnética, evidenciando uma aplicação clínica. Por fim, Figuerola (2012) adota uma abordagem computacional para estudar a dinâmica da plasticidade sináptica durante o sono no hipocampo, utilizando técnicas de inteligência artificial. Embora cada trabalho tenha sua própria ênfase e metodologia, todos contribuem para uma compreensão mais abrangente da função e da plasticidade do hipocampo em contextos variados, desde o desenvolvimento neural até doenças neurodegenerativas.

# DESENVOLVIMENTO

Para um melhor entendimento do desenvolvimento do protótipo, este capítulo foi subdividido em duas seções principais. Na seção 3.1, será abordado o desenvolvimento do modelo de identificação de neurônios atípicos, com foco nos detalhes relacionados ao processo de treinamento, testes realizados e os resultados obtidos. Já a seção 3.2 apresentará a aplicação móvel desenvolvida para exibir os resultados gerados pelo modelo, detalhando sua arquitetura, funcionalidades e os principais aspectos de implementação.

## MODELO de identificação de neurônios atípicos

A etapa de desenvolvimento do modelo de processamento de imagem inicialmente se dá pelo desenvolvimento e levamento de requisitos voltados ao modelo de aprendizado de máquina.

### Requisitos

O modelo foi desenvolvido atendendo os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

1. permitir ao usuário carregar imagens a serem analisadas (RF);
2. realizar o realce e melhoramento de ruídos, distorções e problemas de iluminação utilizando técnicas de processamento de imagens (RF);
3. segmentar a Region Of Interest (ROI) das imagens coletadas no LabelBox (RF);
4. utilizar a rede neural convolucional Fast R-CNN para identificar os neurônios atípicos (RF);
5. aumentar o *dataset* utilizando técnicas de *preprocessing* e *data augmentation* (RNF);
6. utilizar a biblioteca PyTorch e a linguagem de programação Python (RNF);
7. utilizar a biblioteca OpenCV para realizar o processamento de imagens (RNF);
8. ser capaz de realizar a análise em um tempo máximo de um minuto (RNF);
9. disponibilizar as marcações dos neurônios atípicos identificações pelo modelo através de um WebService REST (RNF).

### Descrição do modelo

O diagrama de atividades, apresentado na Figura 8, indica as etapas realizadas para o desenvolvimento do modelo.

Figura – Etapas definidas para detecção de neurônios atípicos.

|  |
| --- |
| ***Data augmentation***  **Pré-processamento**  Hue entre -60º e +60º  Mosaico  Redimensionamento  Marcação das imagens com seus respectivos neurônios atípico  Catalogação de imagens de neurônios  Auto-orientação  Treinamento  Teste  Validação  Área de estuddo |

Fonte: elaborado pelos autores.

Primeiramente, definiu-se o tipo e a forma de captura das imagens. Em seguida, iniciou-se a coleta das imagens dos neurônios atípicos por parte dos alunos do curso de medicina da FURB, separando-as no intuito de facilitar a catalogação e o processamento dessas imagens através de algoritmo de visão computacional. Posteriormente, realizou-se a marcação (*data labeling*), no qual os neurônios a serem detectados são identificados na imagem e demarcados manualmente. Definiu-se uma classe principal: Neurônios atípicos. Depois disso, efetuou-se a etapa de redimensionamento (*resizing*), normalizando o tamanho das imagens, e de *auto-orientation*, garantindo que todas as imagens possuam uma orientação padrão. A partir disso, efetuou-se o *data augmentation* gerando *mosaicos* para as imagens. Além disso, também alterou o valor do Hue, gerando imagens com variação -60° e +60°, visando aumentar a variedade e riqueza do *dataset*. Com isso, torna-se possível efetuar o treinamento e validação do modelo Faster R-CNN. Por fim, são efetuados testes manuais para verificar a eficácia do modelo proposto com imagens teste.

#### Montagem do *Dataset*

O processo de coleta de dados foi realizado com a colaboração dos estudantes do curso de Medicina da FURB. Esses estudantes, com experiência na marcação e visualização de neurônios, utilizaram microscópios da universidade com lentes axiocam 105, para analisar amostras de uma região específica do hipocampo de roedores. A partir dessa análise, os estudantes capturaram imagens das lâminas usando celulares (Samsung e Xiaomi), posicionados para fotografar diretamente através do microscópio. As fotos obtidas foram armazenadas em um *dataset* na internet, compondo o banco de dados utilizado no estudo. A Figura 9 ilustra o processo de captura das imagens dos neurônios atípicos.

Figura – Processo de captura das imagens.





Fonte: elaborado pelos autores.

A criação do *dataset* foi realizada na plataforma LabelBox. Nela, criou-se um projeto chamado TCC – Furb Neurônios Atípicos e um dataset chamado neurônios, onde foram adicionadas as imagens. Posteriormente, realizou-se a marcação dos neurônios que foram avaliados atípicos pela equipe de alunos do curso de Medicina. O processo de captura e análise da imagem durou aproximadamente 6 meses desde o início do projeto (de maio a outubro de 2024) contendo, no final um total de 595 imagens capturadas. A Figura 10 mostra um exemplo de segmentação feita através da ferramenta Labelbox, sendo que as imagens foram rotuladas com o mesmo nome, formando-se pares de imagem (origem e sua segmentação).

Figura – Exemplo do resultado de *labeling* da ferramenta Labelbox

Tela de computador com jogo

Descrição gerada automaticamente

Fonte: elaborado pelos autores.

Com a base de dados montada, realizou-se o *download* de toda a estrutura criada, contendo as imagens e suas respectivas informações, como marcações, posições e *links* das imagens. Optou-se por utilizar a função de exportação de dados v2, ao qual forneceu um arquivo JSON com todos os nós de cada imagem. Nesse arquivo, cada nó continha as informações detalhadas de cada imagem do projeto (Figura 11).

Figura – Visualização do arquivo JSON.

Texto

Descrição gerada automaticamente

Fonte: elaborado pelos autores.

Com a base de dados montada, elaborou-se um *script*, denominado “DownloadLocalImages.py”, que recebe como *input* o arquivo JSON, criando duas pastas: (i) imagens\_aumentadas\_sem\_marcadores e (ii) anotacoes\_aumentadas. O Quadro 6 apresenta o trecho de código para criação das pastas e salvamento dos dados.

Quadro – Trecho de código para criação das pastas de salvamento dos dados.

|  |
| --- |
| # Diretórios de saída  output\_dir = 'C:/temp/imagens\_aumentadas\_sem\_marcadores'  annotations\_dir = 'C:/temp/anotacoes\_aumentadas'  if not os.path.exists(output\_dir):      os.makedirs(output\_dir)  if not os.path.exists(annotations\_dir):      os.makedirs(annotations\_dir)  # Caminho do arquivo NDJSON  ndjson\_file = ndjson\_file = 'C:/temp/jsonBaixado.ndjson'  # Definir as transformações  transform = A.Compose([      A.HorizontalFlip(p=0.5),      A.VerticalFlip(p=0.5),      A.RandomRotate90(p=0.5),      A.Rotate(limit=45, p=0.5),      # Adicione outras transformações conforme necessário  ], bbox\_params=A.BboxParams(format='pascal\_voc', label\_fields=['category\_ids'])) |

Fonte: elaborado pelos autores.

A primeira pasta contém todas as imagens do *dataset* aumentadas por meio de técnicas de *data augmentation*. Essas técnicas, incluem rotações, redimensionamento, ajustes de escala e reflexões horizontais e verticais (Figura 12).

Figura – Imagens com *data* *augmentation* com suas rotações.

|  |  |
| --- | --- |
|  |  |
| Imagem original | Rotação invertida |
|  |  |
| Reflexão horizontal | Reflexão vertical |

Fonte: elaborado pelos autores.

Conforme demonstrado na Figura 12, essa abordagem permite ampliar significativamente a quantidade e a diversidade das informações disponíveis para o treinamento do algoritmo, melhorando sua capacidade de generalização (Quadro 7). No caso, cada imagem foi multiplicada por 5, criando variações que simulam diferentes condições visuais. Isso ajuda o modelo a lidar melhor com situações como mudanças de ângulo, variação de escala e orientações diferentes, tornando o treinamento mais robusto e eficiente. Dessa forma, o *dataset* passou a ter 2080 imagens e marcadores.

Quadro – Código de multiplicação das imagens com suas anotações.

|  |
| --- |
| # Aplicar transformações  for i in range(5): # Número de variações que você deseja criar por imagem  transformed = transform(image=image, bboxes=bboxes, category\_ids=category\_ids)  transformed\_image = transformed['image']  transformed\_bboxes = transformed['bboxes']  transformed\_category\_ids = transformed['category\_ids']  # Salvar imagem transformada sem desenhar os marcadores  output\_image\_name = f'{os.path.splitext(external\_id)[0]}\_aug\_{i}.jpg'  cv2.imwrite(os.path.join(output\_dir, output\_image\_name), transformed\_image) |

Fonte: elaborado pelos autores.

Na segunda pasta, armazenou-se os marcadores de cada imagem criada. Com isso, posteriormente realizou-se o vínculo de cada imagem para o treinamento, incluindo a quantidade de marcações e a posição de cada uma delas (Quadro 8). Esses dados foram salvos no formato .JSON, facilitando a aplicação e o uso nas próximas etapas do projeto.

Quadro – Código de separação de anotação de imagem.

|  |
| --- |
| For bbox, category\_id in zip(transformed\_bboxes, transformed\_category\_ids):  x\_min, y\_min, x\_max, y\_max = bbox  transformed\_annotations['objects'].append({  'annotation\_kind': 'ImageBoundingBox',  'bounding\_box': {  'top': y\_min, 'left': x\_min, 'height': y\_max - y\_min,  'width': x\_max - x\_min  }, 'name': category\_id })  # Salvar as anotações em um arquivo JSON  with open(os.path.join(annotations\_dir, f'{output\_image\_name}.json'), 'w') as f:  json.dump(transformed\_annotations, f)  print(f"Imagem e anotações salvas: {output\_image\_name}") |

Fonte: elaborado pelos autores.

Por fim, o *dataset* passou a conter 2080 imagens e seus respectivos marcadores. Assim, cada imagem específica apresentou seu vínculo individual, o que facilitou, posteriormente, a etapa de treinamento do modelo.

#### Treinamento do modelo

A rede neural escolhida para realizar a identificação de neurônios atípicos foi a Faster R-CNN por ser o estado da arte em termos de reconhecimento e detecção de objetos (ver seção 2.3). Para implementar a Fast R-CNN utilizou-se a plataforma Google Colab devido a possibilidade de ampliação do processamento computacional. Essa solução permitiu treinar o modelo com maior rapidez e facilidade, considerando que as máquinas locais não possuem o mesmo poder de processamento das fornecidas gratuitamente pelo Google Colab. Na plataforma, executou-se o *script* denominado “TrainingScript.py”, ao qual é dividido em cinco funções principais: (i) get\_model; (ii) get\_transform; (iii) collate\_fn; (iv) calculate\_metrics e (v) main.

A função main é o ponto de entrada do programa, responsável por realizar todo o fluxo de treinamento e avaliação do modelo de detecção de objetos. Inicialmente, ela identifica se há uma GPU disponível e define o dispositivo apropriado (GPU ou CPU) para as operações. Em seguida, configura os caminhos das pastas onde estão armazenadas as imagens e suas anotações, carregando o *dataset* customizado através da classe NeuronDataset. Durante esse processo, esta função também verifica se o *dataset* não está vazio, alertando o usuário caso não existam imagens com anotações válidas (Quadro 9).

Quadro – Função main do *script* TrainingScript

|  |
| --- |
| def main():      # Configurar dispositivo      device = torch.device('cuda') if torch.cuda.is\_available() else torch.device('cpu')      # Caminhos das pastas      images\_dir = '/MyDrive/images/imagens\_aumentadas\_sem\_marcadores'      annotations\_dir = '/MyDrive/images/anotacoes\_aumentadas'      # Carregar o Dataset      dataset = NeuronDataset(images\_dir, annotations\_dir, transforms=get\_transform())      # Verificar se o dataset não está vazio      if len(dataset) == 0:          print("Nenhuma imagem com anotações foi encontrada.")          return |

Fonte: elaborado pelos autores.

Após carregar os dados, o *dataset* é dividido em dois conjuntos: um para treinamento (80%) e outro para validação (20%). O DataLoader é configurado para carregar os dados em lotes, otimizando o desempenho durante o treinamento e a validação. Em seguida, a função inicializa o modelo Faster R-CNN, ajustado para a detecção de apenas duas classes (fundo e neurônios atípicos), além do otimizador SGD para ajustar os pesos do modelo. O

Quadro 10 demonstra o processo de inicialização e divisão dos conjuntos de treino e validação.

Quadro – Divisão de conjuntos e inicialização do modelo.

|  |
| --- |
| # Dividir em treino e validação  torch.manual\_seed(1)  indices = torch.randperm(len(dataset)).tolist()  train\_size = int(0.8 \* len(dataset))  dataset\_train = torch.utils.data.Subset(dataset, indices[:train\_size])  dataset\_val = torch.utils.data.Subset(dataset, indices[train\_size:])  data\_loader = DataLoader(    dataset\_train, batch\_size=2, shuffle=True, num\_workers=2,    collate\_fn=collate\_fn  )  data\_loader\_val = DataLoader(    dataset\_val, batch\_size=2, shuffle=False, num\_workers=2,      collate\_fn=collate\_fn  )  # Obter o modelo  num\_classes = 2  # Fundo e neurônio atípico  model = get\_model(num\_classes)  model.to(device)  # Otimizador  params = [p for p in model.parameters() if p.requires\_grad]  optimizer = optim.SGD(params, lr=0.005, momentum=0.9, weight\_decay=0.0005) |

Fonte: elaborado pelos autores.

A função get\_model é responsável por modificar o cabeçalho do modelo Faster R-CNN, adaptando-o ao número de classes. Dessa maneira é possível reutilizar as informações do modelo pré-treinado, ajustando-as para atender ao nosso cenário específico e acelerar o processo de aprendizado (Quadro 11).

Quadro – Função get\_model do *script*.

|  |
| --- |
| def get\_model(num\_classes):    # Carregar o modelo pré-treinado    model = torchvision.models.detection.fasterrcnn\_resnet50\_fpn(weights='DEFAULT')    # Obter o número de entradas do classificador    in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features    # Substituir o cabeçalho preditor    model.roi\_heads.box\_predictor = FastRCNNPredictor(in\_features,num\_classes)    return model |

Fonte: elaborado pelos autores.

Após o carregamento do modelo, executa-se a função get\_transform que é responsável por aplicar as transformações nas imagens, convertendo-as para o formato Python Imaging Library (PIL) e, em seguida, para tensores PyTorch, além de normalizar os valores dos pixels entre 0 e 1.

Quadro – Função get\_tranform sendo utilizado para a parte do *dataset*.

|  |
| --- |
| # Carregar o Dataset  dataset = NeuronDataset(images\_dir, annotations\_dir,  transforms=get\_transform()) |

Fonte: elaborado pelos autores.

Em seguida, utilizou-se a função collate\_fn, cuja principal finalidade é empacotar os dados em lotes no DataLoader (Quadro 13). Ela organiza as imagens em tuplas, facilitando as manipulações e o processamento de múltiplas entradas durante as etapas de treinamento e avaliação.

Quadro – Função collate\_fn com a finalidade de empacotar os dados.

|  |
| --- |
| def collate\_fn(batch):      return tuple(zip(\*batch)) |

Fonte: elaborado pelos autores.

A função calculate\_metrics, apresentada no Quadro 14, é responsável por avaliar o desempenho do modelo de detecção de objetos, utilizando métricas como precisão, revocação e F1-Score. Essa função compara as predições do modelo com as caixas reais (*bounding boxes*) nos dados de validação, com base na métrica Intersection over Union (IoU). Para cada par de imagem e predição, a função verifica quais caixas preditas possuem uma sobreposição suficiente com as caixas reais (com limiar padrão de 0,5) para serem classificadas como verdadeiros positivos (True Positives - TP). As caixas preditas que não possuem correspondência são consideradas falsos positivos (False Positives - FP), enquanto as caixas reais sem correspondência são classificadas como falsos negativos (False Negatives - FN).

Quadro – Código da função calculate\_metrics.

|  |
| --- |
| def calculate\_metrics(outputs, targets, iou\_threshold=0.5):      all\_tp = 0  # Verdadeiros Positivos      all\_fp = 0  # Falsos Positivos      all\_fn = 0  # Falsos Negativos      for output, target in zip(outputs, targets):          pred\_boxes = output['boxes']          pred\_scores = output['scores']          pred\_labels = output['labels']          gt\_boxes = target['boxes']          gt\_labels = target['labels']          # Filtrar predições com base em um score\_threshold (opcional)          score\_threshold = 0.5          keep = pred\_scores >= score\_threshold          pred\_boxes = pred\_boxes[keep]          pred\_labels = pred\_labels[keep]          if len(pred\_boxes) == 0:              all\_fn += len(gt\_boxes)              continue          # Calcular IoU entre predições e ground truth          iou\_matrix = box\_iou(pred\_boxes, gt\_boxes)          # Para cada predição, verificar se há um IoU acima do limiar          ious, indices = iou\_matrix.max(dim=1)          tp = (ious >= iou\_threshold).sum().item()          fp = (ious < iou\_threshold).sum().item()          fn = len(gt\_boxes) - tp          all\_tp += tp          all\_fp += fp          all\_fn += fn      precision = all\_tp / (all\_tp + all\_fp) if (all\_tp + all\_fp) > 0 else 0      recall = all\_tp / (all\_tp + all\_fn) if (all\_tp + all\_fn) > 0 else 0      f1\_score = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) > 0 else 0      return precision, recall, f1\_score |

Fonte: elaborado pelos autores.

Durante o treinamento, que ocorre por um número definido de épocas (padrão: 10), o modelo alterna entre os modos de treino e avaliação. No modo de treino, o modelo processa os dados, calcula as perdas (como erro das caixas preditas em relação às reais), e ajusta seus parâmetros utilizando gradientes descendentes. Já no modo de avaliação, o modelo é testado no conjunto de validação, e métricas como precisão, revocação e F1-Score são calculadas pela função calculate\_metrics para monitorar o desempenho. Ao final do treinamento, o modelo treinado é salvo em um arquivo com extensão .pth (Quadro 15), permitindo que seja utilizado posteriormente para a validação com imagens de teste.

Quadro – Salvamento do modelo em arquivo .pth.

|  |
| --- |
| if i % 10 == 0:                  print(f"Epoch [{epoch+1}/{num\_epochs}], Step [{i}/{len(data\_loader)}], Loss: {losses.item():.4f}")          # Avaliação no conjunto de validação          model.eval()          total\_precision = 0          total\_recall = 0          total\_f1 = 0          num\_batches = 0          with torch.no\_grad():              for images, targets in data\_loader\_val:                  images = list(img.to(device) for img in images)                  targets = [{k: v.to(device) for k, v in t.items()} for t in targets]                  outputs = model(images)                  precision, recall, f1\_score = calculate\_metrics(outputs, targets)                  total\_precision += precision                  total\_recall += recall                  total\_f1 += f1\_score                  num\_batches += 1          avg\_precision = total\_precision / num\_batches if num\_batches > 0 else 0          avg\_recall = total\_recall / num\_batches if num\_batches > 0 else 0          avg\_f1 = total\_f1 / num\_batches if num\_batches > 0 else 0          print(f"Epoch [{epoch+1}/{num\_epochs}] - Precisão: {avg\_precision:.4f}, Revocação: {avg\_recall:.4f}, F1-Score: {avg\_f1:.4f}")      # Salvar o modelo      torch.save(model.state\_dict(), '/content/drive/MyDrive/modelo\_neuronios.pth')      print("Treinamento concluído e modelo salvo em 'modelo\_neuronios.pth'.") |

Fonte: elaborado pelos autores.

Através da função calculate\_metrics, é possível acessar informações sobre o F1-Score, o que permite a análise do gráfico de desempenho, incluindo precisão e revocação do modelo ao longo de suas 10 épocas (Figura 13). Essa função fornece uma visão abrangente do desempenho do modelo, facilitando a identificação de possíveis melhorias e ajustes necessários.

Figura – Gráfico de F1-Score, precisão e revocação do modelo.

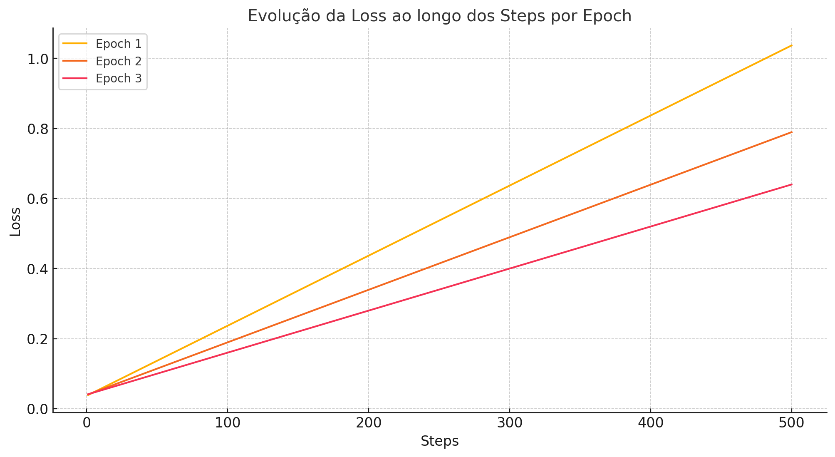
Gráfico, Gráfico de linhas

Descrição gerada automaticamente

Fonte: elaborado pelos autores.

Logo no início do treinamento, conforme mostra a Figura 14, a revocação apresentou um valor inicial mais elevado em relação às demais métricas, indicando que o modelo é capaz de identificar uma maior proporção dos objetos reais presentes nas imagens. No entanto, a precisão iniciou com um valor mais baixo, o que sugere que o modelo ainda estava realizando muitas predições incorretas. Esse comportamento é esperado nas primeiras épocas, pois o modelo ainda está aprendendo os padrões do *dataset*.

Figura – Evolução da *Loss* ao longo dos *Steps* por *Epoch*.



Fonte: elaborado pelos autores.

O gráfico da Figura 15 aponta que a perda (*loss*) do modelo evolui durante os passos de treinamento em cada uma das 10 épocas. Em um treinamento bem-sucedido, espera-se que a perda diminua consistentemente, indicando que o modelo está ajustando seus pesos para melhorar a predição. A perda inicial é relativamente alta, mas diminui gradualmente com o avanço dos steps em cada época. Essa redução reflete o aprendizado do modelo a partir dos dados fornecidos.

Figura – Acurácia de Treinamento e Validação do modelo.

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

Fonte: elaborado pelos autores.

A partir da 5 época, observa-se uma melhoria gradual na precisão, embora seu crescimento tenha sido mais moderado em comparação à revocação. Isso indica que o modelo passou a fazer predições mais assertivas, reduzindo a quantidade de falsos positivos. Por outro lado, conforme apresentado na Figura 13, a revocação apresentou um pico significativo em torno da terceira época, seguido por uma oscilação nas épocas seguintes. Essa oscilação pode ser resultado de ajustes no modelo e do balanceamento entre verdadeiros positivos e falsos negativos.

A métrica F1-Score, que combina precisão e revocação, mostrou uma evolução mais estável ao longo das épocas, refletindo um equilíbrio progressivo entre essas duas métricas. Esse comportamento é um indicativo positivo de que o modelo está aprendendo a generalizar melhor os dados, com um aumento gradual da eficiência global na detecção de objetos.

Nas últimas épocas, todas as métricas apresentaram maior consistência, especialmente o F1-Score presente na Figura 13, que atingiu valores mais elevados e estáveis, evidenciando que o modelo está se aproximando de um bom ponto de convergência. Apesar disso, o desempenho ainda apresenta espaço para melhorias, principalmente em termos de precisão, que pode ser aprimorada com ajustes no treinamento ou *dataset*. Essa análise é crucial para identificar o progresso do modelo e orientar possíveis otimizações futuras, como ajustes nos hiperparâmetros, aumento do *dataset* ou refinamento das transformações aplicadas às imagens.

Quadro – Demonstração de resultados pela Faster R-CNN.

|  |  |  |  |
| --- | --- | --- | --- |
| Rede Neural | Saída | Posições do vetor gerado | Exemplo de características geradas |
| Faster R-CNN | 7x7x1024 | 50176 | 0.154, 0.305, 0.078, 0.561 |

Fonte: elaborado pelos autores.

O Quadro 16 mostra que o modelo Faster R-CNN gera uma saída com resolução 7x7x1024, o que indica que ele extrai características detalhadas de cada região da imagem, representando informações complexas em 1024 canais. O vetor gerado possui 50176 elementos, sugerindo uma alta capacidade de discriminação e captura de detalhes espaciais, o que é útil para detectar objetos pequenos ou complexos, como neurônios atípicos. As características extraídas, como os exemplos numéricos fornecidos (0.154, 0.305, 0.078, 0.561), representam descritores de propriedades específicas dos objetos, como forma e textura, permitindo ao modelo realizar a detecção com precisão. Esse processo reflete uma boa combinação entre resolução detalhada e uma quantidade robusta de informações extraídas da imagem.

#### Execução do modelo

Após o processo de treinamento e geração do arquivo “.pth” contendo todo o treinamento e métricas do modelo com seus respectivos pesos é possível realizar a inferência das imagens de teste. Para isso, torna-se necessário importar as bibliotecas torch e torchvision (para carregar e usar o modelo), warnings (para suprimir mensagens de aviso), e matplotlib (para visualização gráfica das detecções). Também são importadas funções auxiliares para transformação de imagens e manipulação de diretórios

Quadro 17).

Quadro – Importação do modelo.

|  |
| --- |
| from torchvision.models.detection.faster\_rcnn import FastRCNNPredictor  import torchvision.transforms as T  from PIL import Image  def get\_model(num\_classes):      # Carregar o modelo pré-treinado      model = torchvision.models.detection.fasterrcnn\_resnet50\_fpn(weights=None)      # Obter o número de entradas do classificador      in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features      # Substituir o cabeçalho preditor      model.roi\_heads.box\_predictor = FastRCNNPredictor(in\_features,  num\_classes)      return model  # Configurar dispositivo  device = torch.device('cuda') if torch.cuda.is\_available() else torch.device('cpu')  # Definir o número de classes  num\_classes = 2  # Fundo e neurônio atípico  # Carregar o modelo suprimindo o FutureWarning  with warnings.catch\_warnings():      warnings.simplefilter("ignore", category=FutureWarning)      model = get\_model(num\_classes)      model.load\_state\_dict(torch.load('C:/temp/modelo\_neuronios.pth', map\_location=device))  model.to(device)  model.eval() |

Fonte: elaborado pelos autores.

A função get\_model (Quadro 18) configura o modelo Faster R-CNN, carregando uma versão pré-treinada e substituindo o preditor para ajustar o número de classes do problema (nesse caso, 2: fundo e neurônios atípicos). Essa configuração é essencial para personalizar o modelo a uma tarefa específica. Em seguida, o modelo treinado é carregado a partir de um arquivo salvo anteriormente (modelo\_neuronios.pth), utilizando o dispositivo disponível (GPU ou CPU).

Quadro – Função get\_model do *script*.

|  |
| --- |
| def get\_model(num\_classes):      # Carregar o modelo pré-treinado      model = torchvision.models.detection.fasterrcnn\_resnet50\_fpn(weights=None)      # Obter o número de entradas do classificador      in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features      # Substituir o cabeçalho preditor      model.roi\_heads.box\_predictor = FastRCNNPredictor(in\_features, num\_classes)      return model |

Fonte: elaborado pelos autores.

Após o carregamento do modelo treinado, o usuário precisa informar a imagem que será analisada. O *script* verifica a existência do arquivo. A partir disso, a imagem é aberta e convertida para o formato RGB, pois o modelo exige essa configuração (Quadro 19).

Quadro – Trecho de código com o envio da imagem para predição.

|  |
| --- |
| # Solicitar o caminho da imagem ao usuário  image\_path = input("Digite o caminho completo da imagem a ser avaliada: ")  # Verificar se o caminho existe  if not os.path.exists(image\_path):      print(f"O arquivo {image\_path} não foi encontrado.")      exit()  # Carregar a imagem original  img\_original = Image.open(image\_path).convert("RGB")  original\_width, original\_height = img\_original.size  # Aplicar as transformações de treinamento (apenas ToTensor)  transform = get\_transform()  img\_tensor = transform(img\_original).to(device)  # Realizar a predição  with torch.no\_grad():      outputs = model([img\_tensor]) |

Fonte: elaborado pelos autores.

Com a imagem processada, o *script* usa o modelo para realizar a inferência. Isso resulta em uma lista de detecções, contendo caixas delimitadoras, rótulos das classes, e os níveis de confiança (probabilidades associadas às detecções). Para filtrar as predições relevantes, é utilizado um limiar de confiança (0.7, nesse caso). Apenas as detecções com confiança acima desse valor são mantidas (Quadro 20).

Quadro – Aplicação do limiar de confiança.

|  |
| --- |
| # Definir o limiar de confiança  score\_threshold = 0.7  # Obter as boxes, labels e scores  boxes = outputs['boxes'].cpu().numpy()  labels = outputs['labels'].cpu().numpy()  scores = outputs['scores'].cpu().numpy()  # Filtrar boxes com score acima do limiar  indices = np.where(scores >= score\_threshold)[0]  boxes\_filtered = boxes[indices]  scores\_filtered = scores[indices]  labels\_filtered = labels[indices] |

Fonte: elaborado pelos autores.

O número total de detecções e o nível de confiança de cada uma delas são exibidos ao usuário. Caso existam detecções, o código também calcula e exibe a confiança média juntamente com a imagem analisada pelo modelo. Por fim, é retornado duas imagens, uma contendo apenas as marcações dos neurônios com seu nível de confiança, e a segunda imagem informa o nível de confiança em cada neurônio da imagem para ser visualizado na própria imagem em questão (Figura 16).

Figura – Resultado da predição do modelo.

Tela de computador

Descrição gerada automaticamente com confiança média

**(b)**

(a)

Fonte: elaborado pelos autores.

Na Figura 16 item (b) é possível observar as marcações e a acurácia, sendo uma porcentagem média e individual por neurônio identificado como atípico (Figura 16 item a). Caso não seja identificado nenhum neurônio na imagem, o modelo não retorna nada.

Para definir a média de acertos por imagem, capturou-se dez imagens inéditas, ou seja, que não haviam sido submetidas ao *dataset* de treinamento. Essas imagens foram selecionadas pela equipe do curso de medicina, que também produziu uma cópia de cada uma contendo as marcações esperadas. Em seguida, essas dez imagens foram submetiddas ao modelo de identificação de neurônios atípicos, o que permitiu a obtenção de dados necessários para calcular a média de acertos do modelo (Quadro 21).

Quadro – Dados após análise das dez imagens.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exemplo | Marcações\_Academicos | Identicações\_Modelo | Correspondências | % Acerto |
| 1 | 0 | 0 | 0 | - |
| 2 | 6 | 11 | 4 | 66,67 |
| 3 | 4 | 2 | 2 | 50,00 |
| 4 | 6 | 19 | 5 | 83,33 |
| 5 | 2 | 6 | 0 | 00,00 |
| 6 | 7 | 15 | 5 | 71,43 |
| 7 | 5 | 9 | 2 | 40,00 |
| 8 | 2 | 1 | 1 | 50,00 |
| 9 | 5 | 5 | 2 | 40,00 |
| 10 | 3 | 1 | 1 | 33,00 |

Fonte: elaborado pelos autores.

Ao analisar os dados, constatou-se que a média geral de acertos do modelo é de 51,11%. Observando as imagens processadas e os dados coletados, é possível notar que o modelo apresenta uma tendência a identificar um número maior de neurônios atípicos do que realmente está presente nas imagens. No entanto, em grande parte dos casos, o modelo consegue apontar corretamente a maioria dos neurônios indicados pelos acadêmicos do curso de medicina.

Uma análise positiva do modelo pode ser observada na primeira linha dos resultados. Nessa linha, analisou-se uma imagem que, originalmente, não possuía neurônios atípicos, e o modelo conseguiu identificar corretamente esse cenário, não realizando nenhuma marcação.

Figura – Porcentagem de acerto por imagem.

Gráfico, Gráfico de barras

Descrição gerada automaticamente

Fonte: elaborado pelos autores.

A linha vermelha pontilhada no gráfico da Figura 17 representa a média geral de acurácia para os dez exemplos de imagens analisadas. A diferença entre a porcentagem média apresentada na Quadro 21 e no gráfico da Figura 17 ocorre devido ao critério de consideração dos exemplos inválidos. No Quadro 21, desconsidera-se esses casos, como a Figura 16, que não contém marcações feitas pelos acadêmicos. Por outro lado, o gráfico inclui todos os exemplos, mesmo aqueles com 0 marcações, tratando-os como 0% de acerto. Isso faz com que a média apresentada no gráfico seja menor, já que os exemplos inválidos reduzem o valor geral, enquanto o Quadro 21 calcula a média exclusivamente com base nos casos válidos.

Com base nos resultados obtidos, considerando as 595 imagens previamente informadas no *dataset* de treinamento, observa-se uma acurácia média bastante satisfatória por imagem 51,11%. Acredita-se que, com um maior número de imagens disponíveis no *dataset*, seja possível otimizar ainda mais os hiperparâmetros do modelo, tornando-o mais preciso e eficiente. Essa melhoria contribuiria significativamente para o processo de avaliação das imagens por meio do aplicativo móvel.

## Aplicativo Móvel

O desenvolvimento do aplicativo móvel se dividiu em três módulos: o primeiro sendo o desenvolvimento do banco de dados e as regras de negócio, o segundo sendo o desenvolvimento do *backend* e, por último, o desenvolvimento do aplicativo móvel.

### Diagrama de caso de uso

O aplicativo desenvolvido para a identificação de neurônios atípicos oferecerá as funcionalidades representadas no diagrama de casos de uso apresentado na Figura 18.

Figura – Diagrama de caso de uso da aplicação móvel

A diagram of a diagram

Description automatically generated

Fonte: elaborado pelos autores.

Conforme ilustrado na Figura 18, o ator Usuário, após realizar o processo de cadastro (UC01) e autenticação no sistema (UC02), terá acesso a diversas funcionalidades. Entre elas, destaca-se a análise de imagens de neurônios (UC03), na qual o sistema enviará uma requisição ao servidor da API REST (UC04). Em resposta, o servidor retornará a imagem processada, contendo as marcações dos neurônios identificados como atípicos, a quantidade desses elementos e o índice de confiabilidade do resultado (UC05). Adicionalmente, o usuário poderá optar por salvar as imagens processadas para consultas futuras (UC06), bem como acessar e visualizar imagens previamente armazenadas (UC07).

### Requisitos

O aplicativo foi desenvolvido atendendo os seguintes Requisitos Funcionais (RFs) e Requisitos Não Funcionais (RNFs):

1. permitir o carregamento de imagens através do celular para análise (RF);
2. retornar a imagem já analisada e com os neurônios atípicos demarcados (RF);
3. permitir ao usuário salvar a imagem (RF);
4. permitir ao usuário criar repositórios (RF);
5. permitir visualizar imagens já salvas (RF);
6. possibilitar a exclusão de imagens e repositórios específicos (RNF);
7. utilizar a linguagem de programação Flutter para o desenvolvimento da aplicação móvel (RNF);
8. utilizar o banco de dados Firebase para persistir os dados da aplicação móvel (RNF);
9. utilizar *JavaScript Object Notation* (JSON), como canal de comunicação entre a aplicação móvel e a API de identificação de neurônios atípicos (RNF);
10. disponibilizar a identificação de neurônios atípicos através de uma API *Representational State Transfer* (REST) (RNF);
11. ser desenvolvido para Android e para iOS (RNF).

### Arquitetura da solução

A Figura 19 apresenta a arquitetura e o fluxo de funcionamento da aplicação móvel, estruturados em três módulos principais: (i) Aplicação Cliente: representa a interface com o usuário, desenvolvida utilizando o framework Flutter, sendo responsável por oferecer as principais funcionalidades da aplicação móvel; (ii) Servidor: atua como intermediário, gerenciando a comunicação com o banco de dados e executando os processos de análise de imagens por meio de um modelo baseado em aprendizado de máquina; (iii) Banco de Dados: implementado com o uso do Firebase, é responsável pelo armazenamento seguro das informações dos usuários e dos dados associados às imagens processadas.

Figura – Arquitetura da aplicação móvel

A diagram of a cell phone

Description automatically generated

Fonte: elaborado pelos autores.

Nas próximas subseções, serão detalhadas as especificações e os detalhes técnicos de cada módulo de forma individualizada.

#### Banco de dados

Optou-se por utilizar o Firebase como solução de banco de dados para o armazenamento das informações relacionadas às imagens analisadas e aos dados dos usuários. O Firebase é uma tecnologia NoSQL (não relacional) fornecida pelo Google, amplamente reconhecida pela sua flexibilidade e escalabilidade. A escolha por um banco de dados NoSQL justifica-se pelo modelo dinâmico de armazenamento de dados, que dispensa esquemas fixos e facilita o uso de formatos mais flexíveis, como *strings* codificadas em Base64. Essa característica é especialmente relevante para aplicações que demandam constante adaptação e evolução.

Além de sua flexibilidade, o Firebase se destaca pelo alto desempenho em sistemas que lidam com grandes volumes de dados, como imagens. A ferramenta oferece recursos como consultas em tempo real e replicação eficiente, que garantem alta disponibilidade e tornam a plataforma ideal para sistemas escaláveis. No contexto deste projeto, essas funcionalidades foram fundamentais para atender às demandas de armazenamento, processamento e recuperação de dados de maneira eficiente. A Figura 20 exibe a estrutura do banco de dados criado para gerenciar os cadastros de usuários, imagens e repositórios.

Figura 20 – Estrutura do banco de dados.

A screenshot of a computer

Description automatically generated

Fonte: elaborado pelos autores.

No nível mais alto da hierarquia, o banco de dados é organizado em um nó principal denominado users, que agrupa os dados de cada usuário por meio de identificadores únicos gerados automaticamente pelo sistema. Cada registro de usuário contém os seguintes atributos principais:

1. created\_at: *timestamp* que registra o momento da criação do perfil no sistema;
2. email: endereço eletrônico associado ao usuário;
3. password: *hash* criptográfico da senha, garantindo segurança e conformidade com boas práticas de proteção de dados sensíveis;
4. username: nome de usuário para identificação dentro do sistema.

Dentro do nó de cada usuário, existe um nó aninhado chamado repositories, destinado à organização dos repositórios criados. Cada repositório é identificado por um ID único e armazena as seguintes informações:

1. created\_at: *timestamp* que indica a data de criação do repositório;
2. name: nome que identifica o repositório;
3. description: texto descritivo que contextualiza o propósito ou conteúdo do repositório.

Em cada repositório, há um nó específico chamado images, que organiza as imagens relacionadas por identificadores únicos. As informações de cada imagem analisada incluem:

1. accuracy: valor numérico que representa a precisão da análise conduzida;
2. atypical\_neurons: quantidade de neurônios considerados atípicos detectados na imagem;
3. created\_at: *timestamp* correspondente ao momento em que a imagem foi inserida no sistema;
4. description: descrição textual que fornece informações contextuais sobre a imagem;
5. image\_base64: dados da imagem codificados no formato Base64, viabilizando o armazenamento e a recuperação de forma eficiente;
6. name: nome que identifica a imagem.

Essa modelagem orientada por nós e identificadores únicos reflete as melhores práticas para bancos de dados NoSQL, permitindo flexibilidade no armazenamento e na recuperação de dados complexos. Esta estrutura hierárquica favorece consultas direcionadas e a expansão futura do sistema, enquanto as características nativas do Firebase, como sincronização em tempo real e suporte escalável, garantem uma solução robusta para o gerenciamento de informações em um ambiente dinâmico e de alta demanda.

#### *Backend*

O *Backend* foi desenvolvido em Python, utilizando o *framework* Flask, com o objetivo de fornecer uma solução para o gerenciamento de usuários, repositórios e processamento de imagens de neurônios atípicos. Utilizado uma arquitetura REST para organizar diversas funcionalidades que incluem autenticação de usuários, gerenciamento de repositórios e processamento de imagens com o auxílio de um modelo de visão computacional.

A funcionalidade principal do *backend* é o processamento de imagens para detecção de **neurônios atípicos**, utilizando a classe FindImage, que implementa um modelo de visão computacional baseado no **Faster R-CNN.** O Quadro 22 apresenta o trecho de código responsável pela rota/process\_image.

Quadro – Código fonte da rota /process\_image

|  |
| --- |
| @app.route('/process\_image', methods=['POST'])  def process\_image():  try:  image\_base64 = request.json.get('image\_base64')  if not image\_base64:  return jsonify({"error": "No image data provided"}), 400  num\_deteccoes, confianca\_media, boxes\_filtered, scores\_filtered = find\_image\_instance.predict(image\_base64)    image\_predict = find\_image\_instance.save\_images(image\_base64, boxes\_filtered, scores\_filtered)  confianca\_media = float(confianca\_media)  boxes\_filtered = boxes\_filtered.tolist()  scores\_filtered = scores\_filtered.tolist()  response = {  "num\_deteccoes": num\_deteccoes,  "confianca\_media": confianca\_media,  "image\_predict": image\_predict  }  return jsonify(response), 200  except Exception as e:  return jsonify({"error": str(e)}), 500 |

Fonte: elaborado pelos autores.

A rota*/*process\_image, recebe uma imagem em formato Base64, processa-a com o modelo treinado e retorna:

1. o número de detecções encontradas;
2. a confiança média das detecções;
3. a imagem com as marcações de cada neurônio atípico encontrado e a confiança de cada detecção.

A classe FindImage implementa funcionalidades específicas para processar imagens utilizando um modelo modelo\_neuronios.pth. Esse modelo identifica regiões de interesse (neurônios atípicos) em imagens enviadas pelos usuários e retorna informações detalhadas. Os métodos responsáveis por isso são o predict e o save\_images(Quadro 23).

Quadro – Código fonte responsável pela predição

|  |
| --- |
| def predict(self, image\_base64, score\_threshold=0.7):  img\_original = Image.open(BytesIO(base64.b64decode(image\_base64))).convert("RGB")  transform = self.get\_transform()  img\_tensor = transform(img\_original).to(self.device)  with torch.no\_grad():  outputs = self.model([img\_tensor])  outputs = outputs[0]  boxes = outputs['boxes'].cpu().numpy()  labels = outputs['labels'].cpu().numpy()  scores = outputs['scores'].cpu().numpy()  indices = np.where(scores >= score\_threshold)[0]  boxes\_filtered = boxes[indices]  scores\_filtered = scores[indices]  num\_deteccoes = len(boxes\_filtered)  confianca\_media = np.mean(scores\_filtered) if num\_deteccoes > 0 else 0  return num\_deteccoes, confianca\_media, boxes\_filtered, scores\_filtered |

Fonte: elaborado pelos autores.

A função predict,chamada pela rota */*process\_image, é responsável por realizar a inferência do modelo de aprendizado de máquina em uma imagem fornecida como entrada, codificada no formato Base64. Inicialmente, a imagem é decodificada usando a biblioteca *Pillow*, convertendo os dados codificados em um objeto de imagem no formato RGB. Essa conversão é importante para garantir que a imagem tenha três canais de cor, independentemente do formato original.

Depois disso, a imagem é submetida a transformações específicas definidas pela função auxiliar get\_transform. Essas transformações preparam a imagem para ser compatível com o modelo de aprendizado de máquina, realizando etapas como redimensionamento, normalização e conversão para tensor, que é o formato exigido pelos modelos do *PyTorch*. O tensor resultante é então movido para o dispositivo configurado, seja uma GPU ou CPU, otimizando o processamento.

Com a imagem pronta, o modelo realiza a predição. Durante esse processo, a imagem transformada é passada ao modelo dentro de um contexto torch.no\_grad(), que desativa o cálculo de gradientes, melhorando a eficiência da inferência. O modelo retorna os resultados, que incluem as coordenadas das caixas delimitadoras (*boxes*), os rótulos das classes detectadas (*labels*) e as pontuações de confiança (*scores*). Esses valores são extraídos e convertidos para o formato NumPy para facilitar o processamento posterior.

Em seguida, a função filtra os resultados para manter apenas as detecções cuja pontuação de confiança seja maior ou igual a um limiar definido (score\_threshold). Essa filtragem resulta em um subconjunto de caixas (boxes\_filtered) e pontuações (scores\_filtered) que representam as detecções mais relevantes. A função calcula então o número total de detecções e a confiança média das detecções filtradas. Se nenhuma detecção relevante for encontrada, o valor da confiança média será zero.

Por fim, a função retorna o número de detecções, a confiança média, as caixas filtradas e as pontuações associadas. Esses resultados são utilizados posteriormente para análise, visualização ou outras etapas do *pipeline* de processamento de imagens. Essa implementação é essencial em aplicações como detecção de objetos ou análise de imagens, permitindo identificar regiões de interesse com precisão e eficiência (Figura 20).

Figura – Imagem original e imagem retornada pelo método save\_images

|  |  |
| --- | --- |
|  |  |
| (a) Imagem original | (b) Imagem retornada pelo método |

Fonte: elaborado pelos autores.

O método save\_images é responsável por processar e retornar as imagens que contenham detecções feitas pelo modelo de aprendizado de máquina. Primeiramente, a imagem original é recuperada a partir de uma String codificada em Base64. Essa String é decodificada em dados binários, que são convertidos em uma imagem no formato RGB utilizando a biblioteca *Python Imaging Library* (PIL). Essa etapa transforma a entrada em um formato que pode ser manipulado e exibido. A seguir, o método cria uma figura utilizando a biblioteca *matplotlib*. Essa figura é configurada para exibir a imagem original e para desenhar sobre ela as detecções realizadas pelo modelo. Para cada detecção (representada por uma caixa delimitadora), o método desenha um retângulo que corresponde às coordenadas da caixa filtrada. Cada retângulo é configurado com bordas vermelhas e não possui preenchimento, destacando as áreas detectadas na imagem. Além disso, próximo ao canto superior de cada caixa, o método insere um texto com o nível de confiança da detecção (*score*). Esse texto é exibido em amarelo com um fundo semitransparente para facilitar a leitura.

Depois de desenhar todas as caixas e anotações, os eixos da imagem são ocultados para criar uma apresentação mais limpa. A imagem gerada é salva em um *buffer* de memória no formato PNG, usando a biblioteca *BytesIO*. Esse *buffer* armazena a imagem processada temporariamente e é convertido de volta em uma string String Base64.

Por fim, o método retorna a String Base64 correspondente à nova imagem com as detecções marcadas. Esse retorno é utilizado para exibir a imagem processada, garantindo que o resultado da análise seja compreensível e visualmente acessível para os usuários.

#### Operacionalidade do Aplicativo

O desenvolvimento do aplicativo foi realizado utilizando o *framework* Flutter. Para utilizar a aplicação, o usuário precisa primeiramente realizar seu cadastro. Uma vez cadastrado, o usuário deve inserir suas credenciais na tela de login para acessar as funcionalidades do aplicativo. A Figura 21 ilustra as telas de login e de cadastro de usuários da aplicação móvel.

Figura – Telas iniciais da aplicação móvel

Screens screenshot of a login form

Description automatically generated

Fonte: elaborado pelos autores.

Após a autenticação, o usuário será redirecionado à tela inicial da aplicação, que está organizada em duas seções principais:

1. Página Inicial (Home):nesta seção, o usuário pode selecionar uma imagem armazenada em seu dispositivo móvel para análise;
2. Página de Listagem de Repositórios:exibe uma relação de repositórios previamente salvos, juntamente com as respectivas imagens associadas a cada repositório.

Na Página Inicial, é imprescindível que o usuário selecione uma imagem específica de neurônios do hipocampo, pois, do contrário, a análise não será realizada. A imagem é enviada para o servidor, que hospeda a API REST e o modelo de análise. No servidor, a imagem é analisada, para encontrar todos os neurônios atípicos presentes. A aplicação, então, retorna a imagem com as marcações dos neurônios encontrados, a quantidade de neurônios encontrados e uma acurácia média das detecções. Ressalta-se que a qualidade da imagem desempenha um papel crucial no desempenho do modelo, uma vez que imagens de baixa qualidade podem comprometer a acurácia da análise. Isso ocorre porque a baixa qualidade pode dificultar a identificação de neurônios atípicos, resultando na detecção de neurônios presentes na imagem ou na classificação incorreta de neurônios típicos como atípicos. A Figura 22 apresenta um exemplo ilustrativo de uma análise realizada em uma imagem de neurônios do hipocampo.

Figura – Tela com análise de uma imagem dos neurônios do hipocampo

A screenshot of a cell phone

Description automatically generated

Fonte: elaborado pelos autores.

Após a análise da imagem, o usuário terá a opção de excluir as informações exibidas ou salvá-las. Caso opte por salvar, será necessário preencher os campos obrigatórios de nome da imagem e descrição (Figura 23). Além disso, o usuário deverá associar a imagem a um repositório. Para isso, será apresentada uma lista com todos os repositórios previamente cadastrados, permitindo também a criação de um novo repositório, se necessário. Ao optar por criar um novo repositório, o usuário deverá fornecer o nome e uma descrição para o mesmo. Após o preenchimento de todas as informações obrigatórias, será possível finalizar o processo de salvamento da imagem no sistema. Caso opte por excluir a imagem, o usuário será redirecionado para a tela inicial, onde poderá escolher outra imagem para analisar.

Figura – Telas para a salvar imagem

Screens screenshot of a screenshot of a screenshot

Description automatically generated

Fonte: elaborado pelos autores.

Após salvar a imagem, o usuário é redirecionado a tela inicial do aplicativo ao qual é possível visualizar todos os repositórios previamente salvos, que pertençam ao seu usuário, no aplicativo (Figura 24). Ao selecionar um repositório da sublista, as imagens associadas a ele são exibidas. Ao selecionar uma imagem específica, o aplicativo exibe uma tela com todas as informações relacionadas àquela imagem. Adicionalmente, o usuário tem a opção de excluir um repositório completo, o que resultará na remoção de todas as imagens associadas a esse repositório. Alternativamente, é possível excluir individualmente uma imagem específica dentro de qualquer repositório listado.

Figura – Telas de listagem de informações

Screenshots of a screenshot of a cell phone

Description automatically generated

Fonte: elaborado pelos autores.

### Testes de usabilidade

Para avaliar as funcionalidades do aplicativo, realizou-se um teste de usabilidade com dois alunos do curso de medicina da FURB, a fim de avaliar a aceitação do protótipo. Para realização do teste, foi solicitado que cada usuário instalasse em seu dispositivo o executável do aplicativo. Junto ao executável, foi enviado um formulário contendo perguntas relacionadas à identificação do seu perfil, uma lista de tarefas e um questionário de usabilidade. A primeira análise realizada foi em relação ao perfil dos usuários. O Quadro 24 apresenta os resultados obtidos nessa etapa do formulário.

Quadro – Perfil dos usuários do teste de usabilidade

|  |  |
| --- | --- |
| Idade | 100% entre 18 e 25 anos |
| Profissão | Ambos colocaram estudante |
| São estudantes | 100% sim |
| Curso | 100% medicina |
| Frequência de uso do celular | 100% todos os dias |
| Quais funções mais usam do celular | 50% Áudio  100% Foto  50% Google maps  100% Whatsapp, Instagram |
| Modelo usado de celular usado para testar o aplicativo | Xiomi redmi10  Samsung Galaxy S23+ |

Fonte: elaborado pelos autores.

Conforme os resultados obtidos no levantamento do perfil dos usuários, verifica-se que os participantes possuem perfis bem semelhantes, o que confere homogeneidade à amostra. Todos os usuários têm idades entre 18 e 25 anos, sendo estudantes de medicina, o que pode indicar que estão habituados com rotinas exigentes e o uso frequente de tecnologia para estudos e organização pessoal. Além disso, o fato de utilizarem o celular diariamente reflete a integração desse dispositivo em suas atividades cotidianas.

No que diz respeito às funções mais utilizadas, destaca-se que 100% dos participantes relataram usar aplicativos como alarme, foto, WhatsApp e Instagram, evidenciando que essas funções desempenham um papel relevante em suas rotinas. Por outro lado, apenas 50% relataram utilizar o Google Maps e o áudio, o que pode indicar um uso mais específico desses recursos, dependendo das necessidades individuais.

Outro aspecto relevante foi o modelo de celular usado durante os testes, que incluiu dispositivos com diferentes configurações e capacidades, como o Xiaomi Redmi 10 e o Samsung Galaxy S23+. Essa diversidade garante que a avaliação da aplicação seja mais representativa em termos de desempenho e adaptabilidade, considerando dispositivos de gamas diferentes. A análise seguinte foi realizada a partir dos dados obtidos da lista de tarefas executadas pelos usuários (Tabela 1).

Tabela – Respostas da execução da lista de tarefas

|  |  |  |
| --- | --- | --- |
| **Tarefas/Respostas** | **Sim** | **Não** |
| Cadastrar-se no aplicativo | 100,00% | - |
| Fazer login | 100,00% | - |
| Sair do aplicativo | 100,00% | - |
| Escolher uma imagem para ser processada | 100,00% | - |
| Habilitar o acesso a câmera | 100,00% | - |
| Escolher uma foto na galeria | 100,00% | - |
| Visualizar a imagem processada | 100,00% | - |
| Salvar a imagem processada | 100,00% | - |
| Validar os projetos criados | 100,00% | - |
| Abrir imagem processada | 100,00% | - |
| Excluir imagem processada | 100,00% | - |

Fonte: elaborado pelos autores.

A partir dos dados apresentados na Tabela 1, é possível observar que todas as tarefas relacionadas à execução da lista foram concluídas com sucesso por 100% dos usuários. Entre as atividades realizadas estão: cadastrar-se no aplicativo, fazer login, sair do aplicativo, escolher uma imagem para ser processada, habilitar o acesso à câmera, selecionar uma foto na galeria, visualizar e salvar a imagem processada, validar projetos criados, abrir e excluir imagens processadas.

Isso demonstra que o aplicativo é altamente intuitivo e eficiente, permitindo aos usuários realizarem todas as operações necessárias sem dificuldades ou relatos de problemas. Tal desempenho sugere que a aplicação está bem adaptada às necessidades e habilidades dos usuários, proporcionando uma experiência fluida e sem obstáculos. Por fim, realizou-se a análise dos dados obtidos das respostas do questionário de avaliação da aplicação, no qual os usuários podiam responder utilizando uma escala de 1 a 5, sendo 1 discordo plenamente e 5 concordo plenamente. A Tabela 2 apresenta os dados obtidos desta etapa.

Tabela – Resultados do questionário de avalição da aplicação

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pergunta** | **1** | **2** | **3** | **4** | **5** |
| O design da interface do aplicativo é atraente | - | - | - | - | 100% |
| É fácil navegar pelo aplicativo | - | - | - | - | 100% |
| Os símbolos e ícones são claros e intuitivos | - | - | - | - | 100% |
| A interface é semelhante dos demais aplicativos | - | - | 50% | 50% | - |
| É fácil realizar os cadastros de imagens | - | - | - | - | 100% |
| As informações armazenadas no aplicativo são facilmente encontradas | - | - | - | - | 100% |
| Você precisaria do apoio de uma pessoa para utilizar o aplicativo? | 50% | - | - | - | 50% |
| Você acha que exibição a marcação/identificação dos neurônios atípicos encontrados é de fácil interpretação? | - | - | - | - | 100% |
| O aplicativo realiza a análise de neurônios atípicos em um tempo adequado | - | - | - | - | 100% |
| Às vezes, não sei o que fazer no aplicativo | 100% | - | - | - | - |
| O aplicativo tem uma boa fluidez entre as telas (não trava) | - | - | - | 50% | 50% |
| Recomendaria este aplicativo para outra pessoa | - | - | - | - | 100% |
| Acredito que este aplicativo pode ajudar o processo de análise de neurônios atípicos | - | - | - | - | 100% |

Fonte: elaborado pelos autores.

A partir da Tabela 2, é possível dividir a análise entre os aspectos relacionados à usabilidade e as funcionalidades do aplicativo. Em relação à usabilidade, todos os usuários (100%) consideraram o design da interface atraente, a navegação fácil e os símbolos e ícones claros e intuitivos. Além disso, a realização de cadastros de imagens e a busca por informações no aplicativo foram avaliadas positivamente, com todos os usuários afirmando que essas tarefas são fáceis de realizar. No entanto, metade dos usuários relatou necessitar de apoio para utilizar o aplicativo, e todos afirmaram, em algum momento, não saber o que fazer, evidenciando a necessidade de melhorias na orientação ao usuário. Já a fluidez entre as telas foi avaliada como satisfatória por 50% dos participantes, com o restante classificando-a como muito boa.

Sobre as funcionalidades, a marcação e identificação de neurônios atípicos foi considerada de fácil interpretação por todos os usuários, assim como o tempo de resposta do aplicativo para realizar análises. Adicionalmente, todos os participantes recomendariam o aplicativo a outras pessoas e acreditam que ele pode contribuir significativamente para o processo de análise de neurônios atípicos. Em resumo, o aplicativo foi bem avaliado em termos de design, clareza e funcionalidades, mas requer melhorias na usabilidade para minimizar as dificuldades relatadas por parte dos usuários.

Ao final do questionário, os usuários tiveram a oportunidade de compartilhar sugestões e opiniões sobre o aplicativo. Dentre as respostas, destacaram-se comentários positivos, como o reconhecimento do potencial do aplicativo em contribuir para questões educacionais e científicas. Como sugestão de melhoria, foi apontada a necessidade de incluir a funcionalidade de dar zoom nas imagens marcadas, o que facilitaria a revisão humana das marcações. No geral, os usuários elogiaram a otimização, rapidez, e a interface bonita e intuitiva do aplicativo.

# CONCLUSÕES

A identificação e contagem de neurônios atípicos são desafios complexos para os pesquisadores da neurociência, especialmente em análises manuais realizadas com microscópios. Esses procedimentos dependem amplamente da habilidade humana, estão sujeitos a erros e consomem muito tempo, o que dificulta a obtenção de resultados consistentes e pode levar a diagnósticos imprecisos de doenças neurodegenerativas, como a doença de Alzheimer e depressão. Nesse contexto, este trabalho apresentou o desenvolvimento de um modelo de aprendizado de máquina para automatizar e otimizar a identificação e contagem de neurônios atípicos, visando superar as limitações do processo tradicional, que é demorado e suscetível a falhas humanas.

Quanto aos objetivos específicos, pode-se afirmar que foram atendidos de maneira consistente. A segmentação dos neurônios foi realizada utilizando a rede neural Faster R-CNN, que demonstrou bom desempenho em identificar regiões de interesse nas imagens. A extração de características morfológicas permitiu a classificação dos neurônios atípicos, enquanto a validação do modelo incluiu análises detalhadas do tempo de resposta e da precisão em relação aos resultados manuais. A criação de um aplicativo móvel complementou o projeto, fornecendo uma interface amigável para realização de análises, armazenamento e gerenciamento dos dados obtidos.

As decisões tomadas ao longo do trabalho, como a escolha da região do hipocampo para coleta das imagens e o uso da rede Faster R-CNN, mostraram-se adequadas para os objetivos propostos. O uso da plataforma Google Colab permitiu viabilizar o treinamento do modelo com eficiência computacional, enquanto as técnicas de aumento de dados aumentaram a diversidade do *dataset* e contribuíram para a generalização do modelo. As ferramentas empregadas, como LabelBox, PyTorch e Flutter, demonstraram-se eficazes para atender às demandas do projeto.

Os resultados alcançados foram considerados positivos, com o modelo apresentando uma acurácia satisfatória na identificação de neurônios atípicos. Entretanto, foi observado que o modelo ainda possui limitações, como a tendência a identificar um número maior de neurônios atípicos do que o real, o que sugere a necessidade de ajustes adicionais no treinamento e na configuração dos hiperparâmetros. Apesar disso, o simulador utilizado demonstrou-se adequado para o contexto proposto, permitindo avanços significativos em relação à abordagem manual.

Ao comparar este trabalho com estudos correlatos, observou-se que os resultados atingidos estão alinhados às práticas descritas na literatura, como o uso de redes neurais convolucionais para a detecção de padrões em imagens biomédicas. Trabalhos como os de Silva Júnior, Bezerra e Andrade (2020) destacaram o uso de arquiteturas U-Net e obtiveram êxito na segmentação de melanomas em imagens médicas, com índice Jaccard de 83,64%, demonstrando que as técnicas de aprendizado profundo são eficazes para aplicações biomédicas. Embora este estudo tenha utilizado a Faster R-CNN, e não a U-Net, observa-se que os princípios de processamento de imagens são semelhantes e bem-sucedidos. Além disso, Grivol (2022) evidenciou a utilização de *autoencoders* para classificar imagens de ressonância magnética com foco na doença de Alzheimer, com ênfase em métodos de estratificação por idade, corroborando a importância de abordagens personalizadas no aprendizado de máquina. A especificidade deste trabalho para a região do hipocampo e a integração com um aplicativo móvel, contudo, representam diferenciais relevantes, permitindo uma solução acessível e útil para a neurociência experimental e aplicada.

Também cabe ressaltar que foram identificadas limitações que incluem a dependência de um *dataset* mais abrangente para aumentar a precisão do modelo e a necessidade de recursos computacionais mais robustos para reduzir o tempo de processamento em contextos mais complexos. Como extensões futuras sugere-se a ampliação do *dataset* com um maior número de imagens de neurônios atípicos é essencial para melhorar a precisão do modelo de análise. A inclusão de uma variedade de imagens, com diferentes níveis de qualidade – algumas de alta resolução e outras com qualidade inferior – ajudaria a treinar o sistema para lidar com condições adversas, como imagens com ruídos ou baixa definição. Dessa forma, o aumento da diversidade e qualidade das fotos no *dataset* contribui para um modelo mais robusto, capaz de identificar padrões com maior acuracidade. Além disso, pode ser desenvolvido um processo automatizado de coleta e processamento de novas imagens para complementar continuamente o *dataset*, garantindo que o aplicativo se torne mais confiável e preciso à medida que é utilizado por mais profissionais na área de análise de neurônios atípicos. No caso do aplicativo, a implementação de uma funcionalidade de *zoom* nas imagens seria extremamente valiosa, permitindo que os usuários visualizem detalhes mais precisos das células nervosas e suas características, facilitando a identificação de anomalias ou padrões atípicos. Além disso, o compartilhamento de imagens e resultados de análises entre os usuários dentro do próprio aplicativo poderia ser uma adição importante.

Por fim, conclui-se que este trabalho apresenta um grande potencial para transformar a maneira como a identificação e contagem de neurônios atípicos são realizadas, automatizando processos que, tradicionalmente, dependem de análises manuais e intensivas. Dessa forma, a aplicação desenvolvida representa um avanço importante, proporcionando maior consistência e precisão na análise morfológica dos neurônios. Além disso, ela não apenas otimiza a rotina dos pesquisadores, mas também contribui para uma abordagem mais detalhada e quantitativa no estudo das alterações patológicas, possibilitando uma aplicação prática no campo da neurociência.

Referências

AKTER, F. *et al.* **Neuroscience for Neurosurgeons**. [S.l.]: Cambridge University Press, 2024. Disponível em: https://www.cambridge.org/core/books/neuroscience-for-neurosurgeons/neuroanatomy/5FB59BE3638C44ADB97880905A743A4C?utm\_campaign=shareaholic&utm\_medium=copy\_link&utm\_source=bookmark. Acesso em: 03 dez. 2024.

ALJARARI, R. M. **Neuroprotective effects of a combination of Boswellia papyrifera and Syzygium aromaticum on AlCl3 induced Alzheimer’s disease in male albino rat**. Brazilian Journal of Biology, v. 83, 01 jan. 2023.

ALVES, L. F. **Doença de Alzheimer**: uma revisão sistemática da literatura. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano. 07, Ed. 05, Vol. 01, pp. 153-173. Maio de 2022.

AMARAL, D. G.; LAVENEX, P. **Hippocampal neuroanatomy**: The hippocampus book. Oxford: Oxford University Press, 2007. Disponível em: https://academic.oup.com/book/25965/chapter-abstract/193768286?redirectedFrom=fulltext. Acesso em: 20 abr. 2024.

BOTELHO, L. L. R; CUNHA, C. C. A.; MACEDO, M. **O método da revisão integrativa nos estudos organizacionais**. Gestão e sociedade 5.11 (2011): 121-136.

BURGESS, N; MAGUIRE, E. A; O'KEEFE, J. **The human hippocampus and spatial and episodic memory.** Neuron. University College London, 2002. Disponível em: https://www.mendeley.com/catalogue/3b611492-9ae6-3ad1-86a6-0b769ea5beb5/. Acesso em: 15 dez. 2025.

CASARIN, S. T. *et al*. **Tipos de revisão de literatura**: considerações das editoras do Journal of Nursing and Health. J. nurs. health. 2020. Disponível em: https://periodicos.ufpel.edu.br/ojs2/index.php/enfermagem/article/view/19924/11995. Acesso em: 20 abr. 2024.

CERVANTES, E. P. **Classificação morfológica de neurônios baseada na hierarquia das arvores dendríticas**. 2019. Tese (Doutorado em Ciências). Instituto de Matematica e Estatística, Universidade de São Paulo, São Paulo.

FIGUEROLA, W. B. **Dinâmica da Plasticidade Sináptica em neurônios do Hipocampo durante ciclos de sono**: um estudo computacional. 2012. Tese (Doutorado em Ciências) – Curso de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal.

GALVÃO, C. M. *et al*. **Revisão sistemática**: recurso que proporciona a incorporação das evidências na prática da enfermagem. Revista Latino-americana de enfermagem 12 (2004): 549-556.

GRIVOL, M. A. B. **Diagnóstico da doença de Alzheimer usando autoencoders aplicados a imagens de ressonância magnética**. (2022). Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Curso de Ciência da Computação, Universidade Federal de São Carlos, São Carlos.

HE, K. *et al*. Mask R-CNN. **Facebook AI Research (FAIR)**, Cornell University and Cornell Tech, p. 1703-1715, 20 mar. 2017. Disponível em: https://arxiv.org/abs/1703.06870. Acesso em 01 jun. 2020.

HE, K. *et al*. J. 2016. **Deep residual learning for image recognition**. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770-778.

LENT, R. **Cem bilhões de Neurônios? Conceitos Fundamentais de Neurociências**. (2. ed.), 2010. Disponível em: https://edisciplinas.usp.br/pluginfile.php/7626310/mod\_resource/content/1/Roberto%20Lent%20-%20Cem%20Bilh%C3%B5es%20de%20Neur%C3%B4nios%3F.pdf. Acesso em: 20 abr. 2024.

LIU, S.; DENG, W. 2015. **Very deep convolutional neural network based image classification using small training sample size**. IAPR Asian Conference on Pattern Recognition (ACPR) 730-734.

MA, F. *et al*. **Metallothionein 3 attenuated the apoptosis of neurons in the CA1 region of the hippocampus in the senescence-accelerated mouse/PRONE8 (SAMP8)**. Arquivos de Neuro-Psiquiatria, [S.L.], v. 69, n. 1, p. 105-111, fev. 2011. FapUNIFESP (SciELO).

MACHADO, A. **Neuroanatomia funcional.** 2. ed. São Paulo: Atheneu, 2013.

MAGUIRE, E. A. et al. **Navigation-related structural change in the hippocampi of taxi drivers**. Proceedings of the National Academy of Sciences. Montreal, 2000. Disponível em: https://www.pnas.org/doi/epdf/10.1073/pnas.070039597. Acesso em: 20 abr. 2024.

MATANO, B. R. S; PACHECO, T.; ZEMLENOI, L. G. **Prognóstico da Doença de Alzheimer a Partir da Identificação de Biomarcadores.** São Paulo, v 7, 2023. Disponível em: https://www.researchgate.net/publication/376641114\_PROGNOSTICO\_DA\_DOENCA\_DE\_ALZHEIMER\_A\_PARTIR\_DA\_IDENTIFICACAO\_DE\_BIOMARCADORES. Acesso em: 03 dez. 2024.

MONTANARI, T.; **Histologia – Texto, Atlas e Roteiro de Aulas Práticas** – Série Graduação – 2 Ed. Rio Grande do Sul: Ufrgs, 2006. 155p. Disponível em: https://www.nucleodoconhecimento.com.br/saude/sistema-nervoso. Acesso em: 30 nov. 2024.

MOREIRA, C.; **Neurônio.** Revista de Ciência Elementar, dez. 2013. Disponível em: https://rce.casadasciencias.org/rceapp/pdf/2013/008/. Acesso em: 30 nov. 2024.

SILVA JÚNIOR, A. P.; BEZERRA, D. G. S.; ANDRADE, Y. S. **Comparação de arquiteturas de Deep Learning para segmentação de imagens dermatoscópicas de melanoma**. 2020. 84 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) —Universidade de Brasília, Brasília, 2020.

SZEGEDY, C. *et al*. 2016. **Rethinking the inception architecture for computer vision**. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2818-2826.

VALESO, V. **Neurônios:** Unidade Funcional do Tecido Nervoso. Aracaju, [2021]. Disponível em: https://www.passeidireto.com/arquivo/89714279/neuronios-unidade-funcional-do-tecido-nervoso. Acesso em: 03 dez. 2024.

WINKELMANN, E. R. **Análise ultra-estrutural na morte celular da área CA1 do hipocampo após isquemia cerebral global transitória em ratos Wistar**. 2003. Tese de Mestrado (Bacharelado em Biologia) – Instituto de Ciências Biológicas. Universidade Federal do Rio Grande do Sul, Porto Alegre.

WU, Q. et al. **The application of deep learning in computer vision**. 2017 Chinese Automation Congress (Cac), 2017, Jinan, China. Proceedings, IEEE, 2017. p. 6522-6527.

YU, Y. et al. **Estruturas Cerebrais Atípicas como uma Função do Volume de Matéria Cinzenta (VMC) e Densidade de Matéria Cinzenta (DMC) em Jovens Adultos Relacionadas a Traços do Espectro do Autismo**. Frontiers in Psychology, 11. 2020. Disponível em: https://doi.org/10.3389/fpsyg.2020.00523. Acesso em: 20 abr. 2024.

REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster R-CNN: Towards Real Time Object Detection with Region Proposal Networks. **Facebook AI Research (FAIR), Cornell University and Cornell Tech, Cornell University and Cornell Tech**, p. 1506-1520. 6 jan. 2016. Disponível em: https://arxiv.org/pdf/1506.01497.pdf. Acesso em 01 dez. 2024.

He, K., Zhang, X., Ren, S., & Sun, J. 2016. **Deep residual learning for image recognition. The IEEE Conference on Computer Vision and Pattern Recognition** (CVPR) 770-778.

Assunção, Eduardo Timóteo, et al. **"Resultados preliminares de deteção de imagens de pêssegos aplicando o método Faster R-CNN."** *Revista da Associação Portuguesa de Horticultura-março de 2020* 136 (2020): 30-33.