LUÍS OTÁVIO SANTOS

TRIAGEM DE COVID-19:

UM SISTEMA COM MÚLTIPLA INTERPRETABILIDADE BASEADO EM DEEP LEARNING CAPAZ DE DETERMINAR A SEVERIDADE DA LESÃO PULMONAR

LAVRAS - MG

2022

LUÍS OTÁVIO SANTOS

TRIAGEM DE COVID-19:
UM SISTEMA COM MÚLTIPLA INTERPRETABILIDADE BASEADO EM DEEP LEARNING CAPAZ DE DETERMINAR A SEVERIDADE DA LESÃO PULMONAR

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas, para a obtenção do título de Mestre.

Prof. DSc. Demóstenes Zegarra Rodríguez
Orientador

Prof. DSc. Danton Diego Ferreira
Coorientador

LAVRAS - MG

2022

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Santos, Luís Otávio
Triagem de Covid-19 :Um Sistema Com Múltipla Interpretabilidade baseado em Deep Learning capaz de determinar a severidade da lesão pulmonar / Luís Otávio Santos. –2022.
169 p. : il.

Orientador(a): Demóstenes Zegarra Rodríguez.
Coorientador(a): Danton Diego Ferreira.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2022–Universidade Federal de Lavras, 2022.
Bibliografia.

1. COVID-19. 2. Triagem. 3. Aprendizado Profundo. I. Rodríguez, Demóstenes Zegarra. II Ferreira, Danton Diego. III. Título.

LUÍS OTÁVIO SANTOS

TRIAGEM DE COVID-19: UM SISTEMA COM MÚLTIPLA INTERPRETABILIDADE BASEADO EM DEEP LEARNING CAPAZ DE DETERMINAR A SEVERIDADE DA LESÃO PULMONAR

COVID-19 SCREENING: A MULTI-INTERPRETABLE SYSTEM BASED ON DEEP LEARNING CAPABLE OF DETERMINING THE SEVERITY OF LUNG INJURY

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas, para a obtenção do título de Mestre.

APROVADA em 24 de Junho de 2022.

Prof. DSc. Demóstenes Zegarra Rodríguez	UFLA
Prof. DSc. Danton Diego Ferreira	UFLA
Prof. DSc. Wilian Soares Lacerda	UFLA
Prof. DSc. Miguel Arjona Ramírez	USP

Prof. DSc. Demóstenes Zegarra Rodríguez
Orientador

Prof. DSc. Danton Diego Ferreira
Co-Orientador

LAVRAS - MG
2022

Dedico o presente trabalho a minha mãe, Marli, pelo apoio nos meus estudos desde que iniciei minha jornada, e a meus falecidos avôs, Mário e Braz.

AGRADECIMENTOS

Primeiramente agradeço a Deus, por todas as conquistas e pelos fracassos, ambos me fizeram crescer e aprender a cada etapa. Por me proporcionar força de vontade e não desanimar da minha jornada. A minha mãe Marli Silveira Santos pelo suporte em todos os momentos, pela confiança, carinho e ao meu pai Luís Carlos, pelos conselhos e cobranças. A todos meus familiares que confiaram na minha dedicação, em especial para os primos Raphaela, Luana, Géssica e Gustavo, pela ajuda e dicas sobre o texto. A minha namorada Bruna, pelo apoio durante todo o processo.

Aos orientadores Demóstenes Zeguarra Rodríguez e Danton Diego Ferreira pelo apoio e suporte em todas as etapas do trabalho. A todos meus amigos que comigo compartilharam bons momentos juntos.

Ao programa CAPES-EPIDEMIAS, um Programa Estratégico Emergencial de Prevenção e Combate a Surtos, Endemias Epidemias e Pandemia. Ao coordenador do projeto José Manuel de Seixas, pela convocação dos integrantes da Universidade Federal de Lavras, por todo suporte e apoio no trabalho, bem como os demais integrantes do projeto. A UFLA e seus funcionários pela disponibilidade e suporte ao desenvolvimento deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

RESUMO

A COVID–19 é uma doença de nível pandêmico que já conseguiu ceifar milhares de vidas ao redor do mundo. Desde o início da pandemia, pesquisas vêm sendo realizadas em busca da imunização (vacina), triagem e diagnóstico por parte de pesquisadores da área Médica, e também pelo diagnóstico automático por parte das áreas de Engenharia, Ciência da Computação e Estatística, utilizando a inteligência computacional para identificar pacientes infectados. Neste trabalho são reunidos materiais (dados) e informações teóricas sobre as principais técnicas disponíveis para construção de um sistema completo de triagem de pacientes, com base na análise automática por meio de modelos de inteligência computacional. Como material foram utilizadas imagens de tomografia pulmonar, as quais foram segmentadas buscando a região de interesse (os parênquimas pulmonares), utilizando modelos de segmentação semântica. Em seguida, estas imagens foram utilizadas para treinar o modelo *Vision Transformer*, que consiste em um modelo baseado em mecanismos de atenção que permite a explicabilidade de suas classificações. Por fim, foram utilizados modelos de segmentação para identificar duas lesões que são comumente geradas pela ação do vírus no pulmão, a opacidade em vidro fosco e a consolidação. Os resultados experimentais para os modelos de segmentação dos pulmões atingiram um índice *Dice* de 97% aproximadamente. Enquanto o modelo de segmentação de lesões atingiu um índice *Dice* de aproximadamente 85% para consolidação, e um índice *Dice* de aproximadamente 77% para opacidade em vidro fosco. O Modelo de classificação atingiu uma sensibilidade de 91% aproximadamente, precisão e especificidade de 98% aproximadamente.

Palavras-chave: COVID-19, Aprendizado Profundo, Visão Computacional, Segmentação, Interpretabilidade, Diagnóstico, Triagem.

ABSTRACT

COVID-19 is a pandemic-level disease that has taken thousands of lives around the world. Since the beginning of the pandemic, research has been carried out seeking for immunization (vaccine), screening and diagnosis by researchers in the Medical area, and also for automatic diagnosis by the areas of Engineering, Computer Science and Statistics, using computational intelligence methods to identify infected patients. In this work, materials (data) and theoretical information are gathered on the main techniques available for building a complete patient triage system, based on automatic analysis through computational intelligence models. As material, lung tomography images were used, which were segmented looking for the region of interest (lung parenchyma), using semantic segmentation models. Then these images were used to train the *Vision Transformer* model, which is based on attention mechanisms that allows the explanation of its classifications. Finally, segmentation models were used to identify the two types of lesions that are commonly generated by the action of the virus in the lung, ground glass opacity and consolidation. The experimental results for the lung segmentation models reached a *Dice* index of approximately 97%. While the lesion segmentation model achieved a *Dice* index of approximately 85% for consolidation, and a *Dice* index of approximately 77% for ground glass opacity. The classification model reached a recall score of approximately 91%, precision and specificity of approximately 98%.

Keywords: COVID-19, Diagnosis, Screening, Deep Learning, Computer Vision.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Em dezembro de 2019 na cidade Wuhan, localizado na China, foi detectada um novo tipo de doença, a COVID-19, acometida por um vírus da família *coronaviridae*, capaz de causar desde sintomas leves como tosse, febre, dor de cabeça, tontura, e outros acometimentos, até sintomas críticos como a hipóxia silenciosa, falência respiratória e até mesmo levar a óbito. Devido a capacidade de sobrevivência do vírus em ambientes diversificados e à facilidade de contágio entre os seres humanos, os índices de transmissão entre pessoas por contato direto e indireto aumentaram rapidamente (KAUR *et al.*, 2020; WANG *et al.*, 2020).

Além disso, mesmo com esforços do governo Chinês e de outros países, adotando as medidas sanitárias sugeridas pela Organização Mundial da Saúde (OMS), o vírus progrediu para o estágio de pandemia, causando um colapso nos sistemas hospitalares de diversos países (principalmente dos Estados Unidos, Brasil e Índia), atingindo milhares de vítimas ao redor do mundo, que já somam mais de 6 milhões de mortes até então (W.H.O, 2021; ANDERSEN *et al.*, 2020; WU *et al.*, 2020; WILKERSON *et al.*, 2020).

Algumas teorias foram levantadas sobre a contaminação do *SARS-COV-2* ao ser humano, e basicamente, essas teorias seguem duas linhas distintas em relação à forma em que o vírus foi propagado, ambas com o consenso do morcego como origem. Na primeira linha, supõe-se que o vírus tenha sido transmitido pelo consumo direto do morcego pelo ser humano, que se alimenta do animal em algumas regiões da China, principalmente na cidade em que o surto se iniciou (SHEREEN *et al.*, 2020; HUANG *et al.*, 2020). A segunda linha propõe que o vírus tenha sido transmitido por uma cadeia intra e inter espécies, possivelmente passando do morcego ao pangolim (hospedeiro intermediário), até o ser humano (RABI *et al.*, 2020; WACHARAPLUESADEE *et al.*, 2021).

Com o propósito de auxiliar o avanço da cura, melhorar a velocidade do diagnóstico, prevenir novos casos e dar suporte ao sistema de saúde, inúmeros cientistas vêm trabalhando para enfrentar o problema ao redor do mundo. Rapidamente foram desenvolvidas diversas formas de diagnosticar pacientes suspeitos, dentre os quais destacam-se os métodos RT-PCR (do inglês, *Reverse Transcription Polymerase Chain Reaction*) e exames de tomografia computadorizada (do inglês, *Computed Tomography* - CT) (GIRI *et al.*, 2021). Ambos os métodos possuem vantagens e limitações que necessitam ser exploradas (AI *et al.*, 2020). Logo, são necessários esforços de múltiplas frentes de pesquisa, para construção de um

sistema de triagem que contemple a maior quantidade possível de informações sobre os pacientes, evitando ao máximo a perda de vidas.

No campo de tecnologia, diversos dispositivos foram e estão sendo adaptados para que possam ser utilizados no monitoramento e a assistência médica à distância (telemedicina), como termômetros inteligentes, drones (para vigilância, entregas, higienização, etc.), robôs para aplicar testes, aplicativos de monitoramento, trajes inteligentes com capacidade de monitorar sinais vitais, entre outros, todos com o objetivo de facilitar o diagnóstico das pessoas sem gerar novos casos e contribuindo com a redução da proliferação do vírus (CHAMOLA *et al.*, 2020; NASAJPOUR *et al.*, 2020; IMRAN *et al.*, 2020; ROSA *et al.*, 2020; KHUNDAQJI *et al.*, 2020; CONNECT, 2019). No campo de Inteligência Computacional , bem como suas subáreas e outros campos de estudo afins, inúmeras pesquisas se tornaram factíveis do ponto de vista de processamento em tempo real ou tempo hábil, o que vêm contribuindo para o avanço em diversas subáreas como o Aprendizado de Máquina (do inglês, *Machine Learning* - ML) e o Aprendizado Profundo (do inglês, *Deep Learning* - DL), em especial, na área de visão computacional, na qual se concentram as aplicações relacionadas ao processamento de imagens (HEMANTH, 2019; SHIH *et al.*, 2020; MINAEE; ABDOLRASHIDI, 2019).

Em relação ao diagnóstico da COVID-19 por meio de imagens de exames médicos, a maioria dos estudos concentram-se em trabalhar com bases de imagens de tomografia computadorizada, que são imagens geradas por exames radiológicos não invasivos (FARHAT; SAKR; KILANY, 2020). As imagens de tomografia conseguem entregar um maior nível de detalhes no mapeamento das infecções, o que torna a análise um pouco mais completa. Imagens de radiografia também podem ser utilizadas no suporte ao diagnóstico, elas são financeiramente mais acessíveis que as tomografias, e os equipamentos utilizados possuem um processo de higienização menos complexo (GARG *et al.*, 2020; MAJEED *et al.*, 2020; WANG; LIN; WONG, 2020; MAHMUD *et al.*, 2020). Por outro lado, existem algumas divergências nos estudos sobre imagens de ultrasonografia, devido a presença de características semelhantes entre as imagens de ultrasonografia de pacientes diagnosticados com COVID-19 e imagens de pneumonia de origem viral e bacteriana. Alguns pesquisadores apontam a ultrassonografia como não relevante ao estudo do COVID-19; sendo o tipo de imagem mais indicada para o estudo de doenças pulmonares a imagem de tomografia, por apresentar maior nível de detalhes (KHALILI; HASELI; IRANPOUR, 2020). Outros estudos já

sugerem o contrário, ou seja, que a ultrassonografia é relevante pois não expõe os pacientes à radiação e pode até mesmo substituir as imagens de tomografia comparando as lesões nos dois tipos de imagens (VIEILLARD-BARON; GOFFI; MAYO, 2020; ZIELESKIEWICZ *et al.*, 2020; OTTAVIANI *et al.*, 2020). Imagens de ressonância magnética são menos utilizadas pelo sua baixa acessibilidade e falta de bancos de dados extensos, mas também existe a possibilidade de analisá-las (GARCIA *et al.*, 2020).

Neste contexto, os modelos de DL são os mais aplicados quando se trata de diagnóstico automático por imagens. No início da pandemia, predominavam os modelos baseados nas redes neurais convolucionais profundas (do inglês, *Deep Convolutional Neural Networks* - DCNN). São modelos com maior quantidade de parâmetros e que são capazes de lidar com problemas complexos e repetitivos com excelência, obtendo resultados tão bons quanto especialistas no assunto (GUNRAJ; WANG; WONG, 2020). No decorrer da pandemia, os modelos baseados em mecanismos de atenção – até então usados para processamento de linguagem natural (do inglês, *Natural Language Processing* - NLP) – foram adaptados para processamento de imagens (DOSOVITSKIY *et al.*, 2020). Estes modelos começaram a conquistar espaço e atualmente estão sendo amplamente utilizados em pesquisas relacionadas (TEODORO *et al.*, 2022). Entretanto, grande parte destes modelos necessitam de uma quantidade abundante de amostras (imagens) e parâmetros de configuração do modelo para que consigam absorver as características e padrões dos dados, e assim conseguir classificar automaticamente amostras desconhecidas.

Se tratando de um problemas relacionados à saúde, particularmente de imagens médicas, existe uma busca crescente por algoritmos e técnicas capazes de descobrir quais regiões de uma imagem são utilizados pelos modelos de classificação. Pois a simples rotulação dos dados não demonstra quais foram os padrões determinantes que o modelo encontrou na amostra, e que levaram à classificação da mesma. Estas técnicas utilizam basicamente a derivada como forma de encontrar quais parâmetros possuem maior relevância na obtenção do resultado em cada camada, da última camada até a entrada (BRIDGE *et al.*, 2020; CHEFER; GUR; WOLF, 2020).

Além disso, é possível buscar formas alternativas de garantir com que o modelo aprenda a encontrar padrões somente nas regiões de interesse, inserindo técnicas de segmentação semântica no pré-processamento (ZHANG *et al.*, 2020a). Assim, a combinação

da segmentação com modelos baseados em mecanismos de atenção, pode ser uma grande aliada no treinamento dos modelos.

As lesões geradas pela COVID-19 no pulmão, podem ter diferentes características, como por exemplo, opacidade em vidro fosco e consolidação, as quais podem ser segmentadas individualmente e classificadas. Assim, seria possível determinar as áreas de cada lesão e consequentemente determinar o grau de severidade do paciente, servindo esta informação como suporte ao especialista que realiza a triagem.

Enquanto a maioria dos modelos de aprendizado profundo busca classificar imagens, tomando como base somente as informações contidas nas imagens e no banco de dados em geral, o radiologista entende a doença e as características relacionadas. Ao se tratar de um assunto delicado como a saúde, pode ser importante providenciar informações contundentes sobre o resultado. Por exemplo em Mondal *et al.* (2021), são informadas as regiões da imagem que o modelo utiliza para classificar a imagem. Ou também em Zhang *et al.* (2020b), proposta na qual é a realizada a análise dos pacientes por meio das lesões desenvolvidas no pulmão do paciente (por meio de modelos de segmentação), juntamente com os sintomas, com intuito de realizar o diagnóstico dos pacientes. Além da possibilidade de extrair informações dos modelos com a explicabilidade e informações de lesão, também é possível informar a região infectada (AFSHAR *et al.*, 2021; CARVALHO *et al.*, 2021). Com a informação de localização e volume de infecção, é possível identificar o estágio que o paciente se encontra.

1.0.1 Objetivo Geral

O objetivo deste trabalho de forma geral, consiste em propor um sistema base para suporte à triagem e tomada de decisão acerca da doença COVID-19, utilizando a análise automática de imagens de tomografia. Para realizar tal sistema, um conjunto de modelos de inteligência computacional foi proposto, partindo desde a segmentação das imagens buscando a região de interesse ou RoI (do inglês, Region of Interest) até determinar um diagnóstico sugerido.

1.0.2 Objetivo Específico

De forma específica, este trabalho consiste na busca pelos seguintes objetivos:

1. Realizar um sistema de treinamento eficiente, capaz de realizar a seleção de imagens minimamente relevantes ao problema, com base na proporção de parênquima pulmonar presente na imagem;

2. Gerar um sistema com múltiplos modelos de segmentação e diagnóstico, enfatizando as regiões do parênquima pulmonar pertinentes ao problema, conferindo um grau de explicabilidade ao resultado;

3. Fornecer uma base de segmentação de imagens de tomografia, com anotações (máscaras) tanto do parênquima pulmonar quanto das lesões opacidade em vidro fosco e consolidação;

4. Proporcionar uma informação de severidade de lesão presente em cada recorte do paciente, indicando a porcentagem de lesão sobre o pulmão;

5. Realizar uma comparação entre os resultados dos modelos de interpretabilidade e de segmentação de lesões. Proporcionando ao profissional na linha de frente, uma forma de identificação utilizando duas fontes de informação.

1.1 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

1) **Capítulo 1:** O Primeiro capítulo introduz o tema, identificando o problema, discutindo desde as noções básicas até os assuntos mais elaborados relacionados ao tema;

2) **Capítulo 2:** No Segundo capítulo serão discutidas algumas noções básicas sobre os dados utilizados no projeto, e uma base teórica sobre os métodos empregados no trabalho;

3) **Capítulo 3:** O Terceiro capítulo abrange os principais trabalhos e pesquisas relacionadas ao assunto e aos objetivos;

4) **Capítulo 4:** O Quarto capítulo propõe um método para alcançar os objetivos propostos;

5) **Capítulo 5:** O Quinto capítulo mostra os resultados e respectivas discussões;

6) **Capítulo 6:** O Sexto capítulo apresenta as conclusões do estudo, apontando as principais dificuldades do cenário atual sobre o assunto e indica possíveis melhorias.

2 REFERENCIAL TEÓRICO

Neste capítulo serão discutidos os tipos de dados que serão trabalhados, peculiaridades e desafios inerentes. Também serão discutidos as principais técnicas e algoritmos meta-heurísticos utilizados que foram utilizados de forma direta ou indireta no trabalho. Algoritmos que compreendem desde as anotações manuais realizadas via *software*, passando pela segmentação de imagens na área médica, até os tipos de redes neurais utilizados na detecção da COVID-19.

2.1 Bases de dados e suas peculiaridades

De acordo com Peters (2018) pixel é uma representação da resposta de um fenômeno físico em que um sensor óptico é submetido a uma partícula de luz (fóton) que foi refletido por um objeto dentro de um campo de visão. Neste contexto, todos os pixels de uma imagem são valores representativos da intensidade dos fótons em forma de pixels, armazenados em um determinado instante, e dispostos de forma tabular na memória de um computador. Para o cientista de dados, cada tipo de dado deve ser tratado de forma diferente, e seu papel é configurar o algoritmo para que o computador possa fazer sua tarefa de forma automática.

Modelos gerados por redes neurais podem ser treinados de forma supervisionada (com alvos específicos guiando o treinamento) ou não supervisionada (sem alvos, treinam por agrupamento por similaridade), em ambos os casos, eles treinam com base em um conjunto de informações (banco de dados), e dependem fortemente deste conjunto, do formato como o dado foi coletado, metodologia utilizada, organização, das características que este conjunto de dados carrega, entre vários outros aspectos que podem ser detalhados dentro de um banco de dados.

Um problema que têm se tornado preocupante por alguns autores é o desbalanceamento da base (do inglês, *dataset shift*), que ocorre quando uma base de dados está desbalanceada de alguma forma, e leva o modelo para um viés de seleção, apresentando respostas tendenciosas para um subconjunto com maior proporção nesta base de dados (DAVID; De Abajo, 2010; ASIM; AHMED; HAND, 2019). Com relação aos algoritmos, a base de dados cujo conteúdo seja majoritariamente composto de imagens, requer uma capacidade de processamento maior, em geral, pois terá grande quantidade de parâmetros para ajustar, devido a alta dimensionalidade necessária. Como por exemplo, as imagens do tipo RGB, que são compostas por um conjunto de 3 matrizes ou canais no formato (n_h , n_w,

n_c), em que n_h corresponde a quantidade de pixels na vertical (altura), n_w à quantidade na horizontal (largura) e n_c é a quantidade de canais (PLANCHE; ANDRES, 2019b) (SNYDER; FEDOROVSKAYA, 2011).

Um problema que também pode ocorrer em processamento de imagens é a mudança nos dados (do inglês, *data drift*) que é uma mudança nas características das imagens que compõem o processo modelado. Por exemplo, dado um modelo que foi treinado somente com imagens médicas de pacientes com COVID-19 (infectados pelo SARS-COV-2), com o surgimento de novas variantes, existe a possibilidade de o modelo não identificar lesões por serem menos ou mais severas e aparecerem na imagem menos nítidas que as lesões geradas pelo SARS-COV-2 (BENTLEY *et al.*, 2021), e, ao tentar treinar o modelo com novos dados, o modelo pode perder o poder de generalização para os dados antigos. Logo, ele se torna suscetível ao esquecimento catastrófico. Uma forma de explorar ou mitigar esse problema, é dado pelo treinamento continuado, que é uma técnica que busca treinar continuamente o modelo, a medida que novas amostras com novas características vão surgindo, sem perder muito da estatística que já foi absorvida pelo modelo. Este dilema entre assimilar um conjunto novo ao problema, sem perder desempenho no problema original, é denominado dilema estabilidade-plasticidade (LEONILDO COSTA E SILVA, 2019; SEGRELLES-CALVO; De Saraújo; FRASES, 2020).

2.2 Tomografia Pulmonar Computadorizada

A tomografia é um teste não invasivo que permite gerar imagens precisas em 2 ou 3 dimensões do pulmão, ela consiste em uma varredura da região torácica no formato de cortes por meio de raios-x, em que múltiplos feixes de raio-x são utilizados para mapear a região que se pretende analisar, e, por este motivo, possui um nível de detalhamento superior à radiografia. O resultado final do exame consiste em um conjunto de fatias ou *slices* que são imagens da região torácica (NAIDICH *et al.*, 2007). As anormalidades presentes na maioria das doenças pulmonares são mais visíveis e facilitam a detecção por parte dos profissionais especialistas (BHALLA *et al.*, 2020).

2.2.1 Lesões e Suas Características

Quando o exame de tomografia computadorizada (do inglês, *Computed Tomography* - CT) é realizado em um paciente infectado pelo SARS-COV-2, ele possivelmente apresentará lesões em seus pulmões, dependendo do estágio e gravidade dos acometimentos. De acordo com a pesquisa realizada por Ai *et al.* (2020), dependendo do tempo de infecção, as anormalidades podem aparecer no pulmão sem que seja detectado pelo principal teste utilizado, o RT-PCR no caso (WALLER *et al.*, 2020; GOMES *et al.*, 2020). Da mesma forma, o exame de tomografia pode não apresentar lesões, e o teste RT-PCR acusar a presença do vírus no organismo do paciente. Por isso, todo caso deve ser acompanhado de um médico especialista, para que ele possa avaliar concomitantemente os sintomas do paciente, as imagens e severidade associada ao paciente, e demais exames associados, para então tomar as providencias necessárias.

Um levantamento sobre os estudos realizados com pacientes infectados por SARS-COV-2, mostraram as principais lesões que se desenvolvem nos parênquimas pulmonares dos pacientes e são visíveis nos exames de tomografia:

1) Na China, um estudo realizado por Li *et al.* (2020) averiguou a incidência de diferentes tipos de lesões pulmonares em 83 pacientes diagnosticados com COVID-19, tais lesões foram verificadas por radiologistas experientes, as que verificaram uma maior frequência as seguintes lesões (na ordem): Opacidade em Vidro Fosco, Opacidades Lineares e Consolidação;

2) Um outro estudo também realizado na China, desta vez com 307 pacientes, verificou as seguintes lesões com mais incidência: Opacidade em Vidro Fosco, Consolidação e Fibrosi (SHANG *et al.*, 2020);

3) Outros estudos, baseados em ensaios e pesquisas clínicas também fizeram suas análises sobre a frequência das lesões em pacientes, e a Opacidade em Vidro Fosco aparece na primeira colocação de incidência com unanimidade, seguida pela consolidação na maioria dos casos (PAREKH *et al.*, 2020; CAROTTI *et al.*, 2020; WANG *et al.*, 2020).

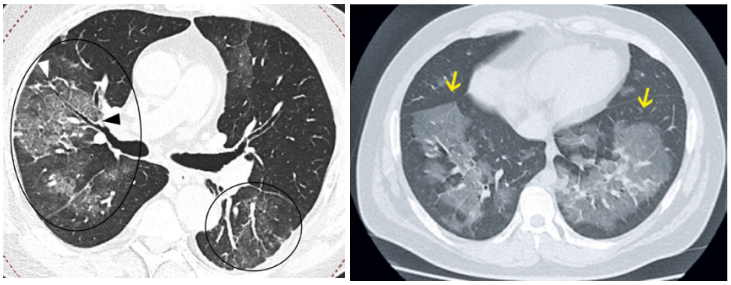
A seguir serão detalhados e exemplificados dois tipos de lesão comumente observadas em pacientes infectados com COVID-19.

2.2.1.1 Opacidade em Vidro Fosco

De acordo com Hansell *et al.* (2008), a Opacidade em Vidro Fosco pode ser definida como uma região nebulosa, na qual pode ocorrer um preenchimento parcial dos espaços aéreos alveolares, porém ela não é patognomônica da COVID-19, ou seja, ela não é uma lesão exclusiva da COVID-19, podendo se manifestar em outros tipos de doenças, como mostrado na Figura 2.1.

Figura 2.1 – Exemplos de Opacidade em Vidro Fosco.

(a) (b)



Fontes: a: (CAROTTI *et al.*, 2020), b:(MATOS *et al.*, 2021).

Como pode ser notado na figura 2.1, a lesão gerada pela ação do vírus nos pulmões, correspondem a região com a tonalidade mais "cinza" (circulada na Figura 2.1.(a) e apontada por setas na Figura 2.1.(b)), e ela pode aparecer em diferentes níveis de intensidade. A partir do acúmulo do vírus nos parênquimas pulmonares, a densidade das células são alteradas pelo acúmulo do exsudato[1], com o aumento da densidade a tonalidade vai aumentando,

[1] Corresponde a um fluído gerado no interior do pulmão quando se tem algum tipo de infecção ou lesão (ALLAM *et al.*, 2020).

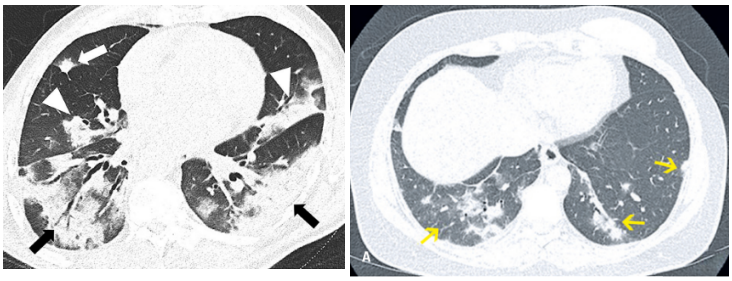
até o ponto em que se observa uma região homogênea, com tonalidade branca, na qual a lesão já passa a ser considerada, consolidação.

2.2.1.2 Consolidação

De acordo com Hansell *et al.* (2008), a consolidação se trata da formação e acúmulo do exsudato ou outro produto de uma doença que venha a substituir o ar alveolar presente no pulmão, tornando-o mais "sólido"(denso) nas regiões infectadas. Observe na Figura 2.2 os dois exemplos:

Figura 2.2 – Exemplos de Consolidação.

(a) (b)



Fontes: a: (CAROTTI *et al.*, 2020)

, b: (MATOS *et al.*, 2021).

A Consolidação por sua vez, possui a tonalidade branca, e sobrepõe uma região completamente. Geralmente ela aparece junto com a Opacidade. Um detalhe importante a se destacar é região na Figura 2.2.(a), na parte inferior direita (apontada por uma seta preta), é que a consolidação compromete a visualização do pulmão, e consequentemente a sua anotação, que será melhor discutida no apêndice A.

2.3 Introdução ao Aprendizado de Máquina

Desde a invenção dos primeiros computadores digitais, desenvolvidos a décadas atrás, o ser humano vem buscando desenvolver programas, algoritmos e sistemas capazes de realizar tarefas de forma automática, e de maneira tão excepcional quanto o ser humano. No início, para desenvolver tarefas simples como cálculos matemáticos, eram utilizadas estruturas lógicas condicionais simples, do tipo "Se"-"Outro"-"Então" (do inglês, *If-Else-Then*, e também para tomada de decisões mais básicas com algoritmos baseados nos conhecimentos de especialistas. Hoje em dia, com o avanço tanto da tecnologia empregada nos computadores (*hardware*) quanto nos programas (*software*) e linguagens de programação, fez com que essa busca por algoritmos capazes de realizarem tarefas complexas tomasse um patamar mais elevado.

Culminando até os algoritmos mais atuais que se baseiam em modelos de Inteligência Computacional. Este é um campo de estudo que surgiu a algumas décadas e vem se desenvolvendo juntamente com o avanço da tecnologia. Consiste em uma área que se concentra no estudo de técnicas que possam fazer um modelo aprender a realizar o reconhecimento dos padrões presentes nos dados por meio de processos estatísticos, e por meio de uma função objetivo, absorver tais padrões relacionados ao problema, e assim gerar um resultado (BISHOP, 2016). Para mais informações sobre o contexto histórico, conferir o apêndice A.1.

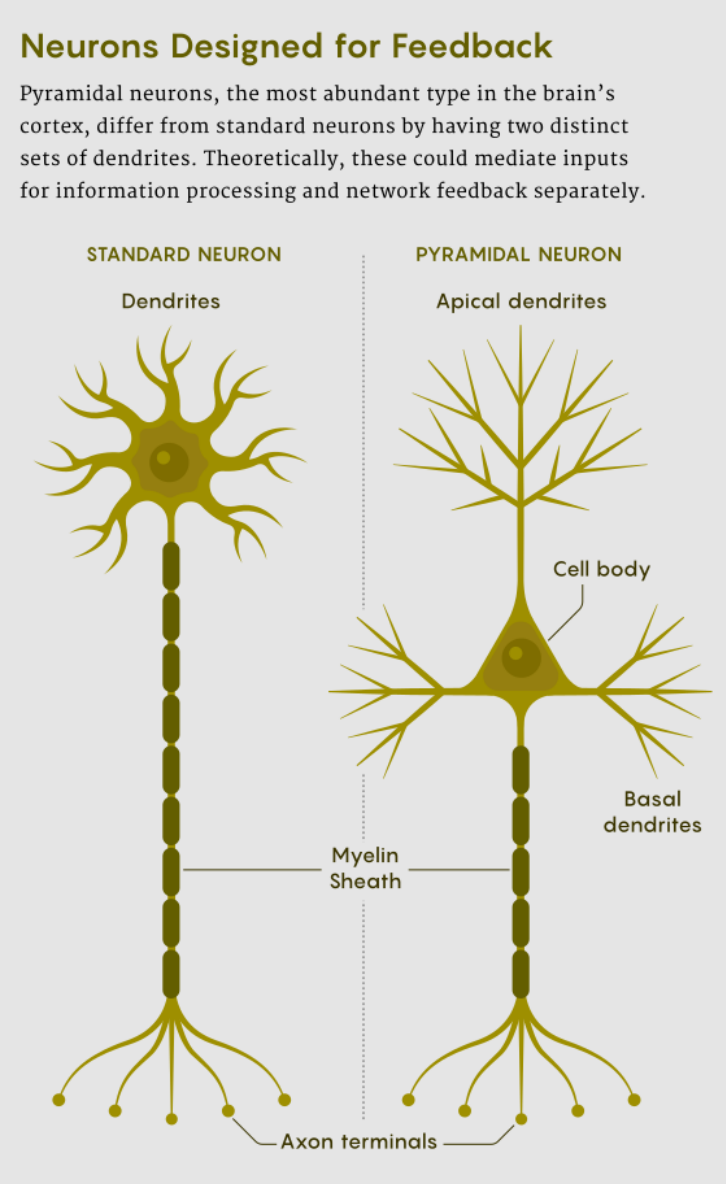
2.3.1 Conceitos de Aprendizado de Máquina

Os modelos de Redes Neurais Artificiais (RNA) partem do comportamento de um neurônio biológico e do princípio psicológico de como o ser humano aprende. Sabe-se que existem diversas situações cotidianas que o ser humano pode aprender, tanto pela experiência prática quanto pelo conhecimento teórico, por tentativa e erro, por um *feedback* positivo ou negativo, entre outros (ARBEL; MURPHY; DONCHIN, 2014; ORMROD, 2017; ROSENBLATT, 1962).

Apesar da premissa do neurônio biológico ser utilizada em muitos estudos, sabe-se que o neurônio e o sistema nervoso em si, é muito mais complexo do que realmente era imaginado até então. Os estudos indicam que existem diferentes tipos de neurônios, tanto biologicamente, quanto computacionalmente. De forma biológica o neurônio mais conhecido (neurônio padrão), por muitas vezes é utilizado para realizar a analogia para

se referir ao neurônio artificial, como mostrado na Figura 2.3 (lado esquerdo). Porém alguns estudos mostram que o neurônio mais abundante no sistema nervoso é o neurônio piramidal, na Figura 2.3 (lado direito) (Anil Ananthaswamy, 2021; GOH *et al.*, 2021).

Figura 2.3 – Diferentes tipos de Neurônios.

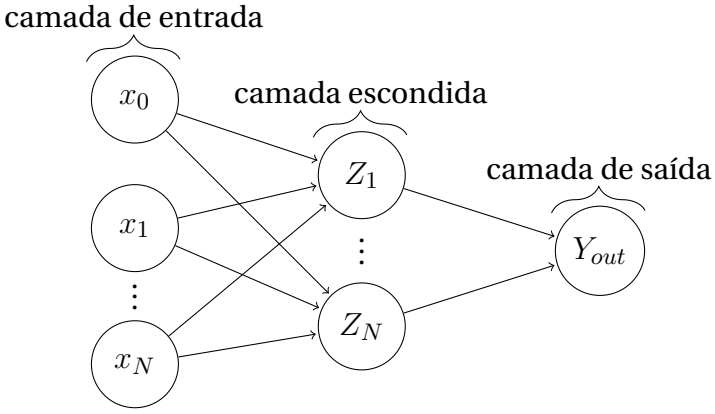


Fonte: (Anil Ananthaswamy, 2021)

Do ponto de vista da aproximação computadorizada, o princípio do neurônio artificial, possui o objetivo de ajustar um conjunto de parâmetros (pesos e *biases*), para que ao final do treinamento, eles sejam capazes de automatizar uma tarefa. Existem diversas tentativas de se modelar o neurônio e seu processo cognitivo, um deles é o neurônio multimodal, observado em (GOH *et al.*, 2021).

Para um melhor entendimento do funcionamento dos algoritmos e modelos descritos no presente trabalho, será apresentado inicialmente o modelo *Perceptron* Multicamadas - MLP. Um conjunto de neurônios artificiais podem ser definidos pela rede multicamadas, representada pela Figura 2.4.

Figura 2.4 – Exemplo MLP



Fonte: Arquivo Pessoal do Autor.

No diagrama da Figura 2.4 cada círculo corresponde a um "neurônio" e eles se relacionam entre si pelos pesos, que são as setas. Dependendo de como a entrada interage (matematicamente) com cada neurônio, ele poderá ser ativado e produzir um resultado que irá rotular ou aproximar um ou mais valores de saída, dependendo da aplicação ou problema, ou seja, Y_{out} pode ser um único valor gerado por um neurônio de saída, ou vários valores de saídas de neurônios distintos. De forma simplificada, as equações de cada camada podem ser equacionadas de forma genérica por (2.1).

$$\begin{cases} Z_1(z) = X_0 \cdot W_{(1,1,1)} + X_1 \cdot W_{(1,2,1)} + \cdots + X_N \cdot W_{(1,i,j)} + b_{1,1} \\ Z_2(z) = X_0 \cdot W_{(1,1,2)} + X_1 \cdot W_{(1,2,2)} + \cdots + X_N \cdot W_{(2,i,j)} + b_{1,2} \\ \vdots \\ Z_N(z) = X_0 \cdot W_{(1,1,j)} + X_1 \cdot W_{(1,2,j)} + \cdots + X_N \cdot W_{(1,i,j)} + b_{l,j} \end{cases} \tag{2.1}$$

Em que $W_{(l,i,j)}$ são os pesos de cada camada em que l indica o índice da camada, i indica o índice do neurônio (ou da entrada), e j indica o índice do neurônio conectado na camada de saída, e $b_{l,j}$ indica o fator *bias* que faz o ajuste de desvio ou erro sistemático da rede neural. Nota-se que b não possui o índice i, uma vez que ele não se conecta entre camadas e sim a um neurônio (ou camada no caso geral) exclusivamente. Como pode ser observado, se tratam de equações lineares simples, capazes de solucionar problemas

simples. A partir dos avanços nas pesquisas, foram adicionadas funções não lineares, para permitir o mapeamento de problemas complexos, principalmente de problemas reais, permeados com relações estatísticas ordem mais elevada (BISHOP, 2016).

No algoritmo de *backpropagation* a rede passa por duas fazes que se repetem até que o modelo tenha erro mínimo. Na primeira etapa são calculadas a(s) saída(s) e na segunda são ajustados os parâmetros da rede, em um processo iterativo. Na primeira etapa, a de propagação (*forward*), a saída pode ser determinada resolvendo uma equação do tipo (2.2)

$$Y_{out} = f\left(\sum_{k=1}^{M} Z_{(l,i,j)} \cdot W_{(l,i,j)} + b_{l,j}\right), \tag{2.2}$$

em que f representa a função de ativação não linear, que pode ser aplicada na saída de cada camada, até a camada de saída. Após calculada a saída da rede neural artificial, o "aprendizado" será efetuado pelo ajuste dos parâmetros. Este ajuste é realizado pela retropropagação do erro, que consiste em determinar as derivadas parciais da função de custo (aplicada na camada de saída) com respeito a cada parâmetro. As derivadas parciais informam qual a "contribuição" cada parâmetro tem sobre o erro. A derivada parcial para uma saída com respeito a um determinado parâmetro, de forma genérica, fica de acordo com a equação (2.3),

$$\frac{\partial \alpha}{\partial W_1} = \frac{\partial \alpha}{\partial Y_1} \cdot \frac{\partial Y_1}{\partial f_1} \cdot \frac{\partial f_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial W_1} + \frac{\partial \alpha}{\partial Y_2} \cdot \frac{\partial Y_2}{\partial f_2} \cdot \frac{\partial f_2}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial W_1} + \cdots + \frac{\partial \alpha}{\partial Y_N} \cdot \frac{\partial Y_N}{\partial f_N} \cdot \frac{\partial f_N}{\partial Z_N} \cdot \frac{\partial Z_N}{\partial W_1}, \tag{2.3}$$

na qual α corresponde a uma função de erro/custo, aplicada à saída Y_{out}, e ela foi omitida para simplificação do exemplo. A atualização dos pesos e *biases* serão dados pelas equações 2.4 e 2.5, respectivamente.

$$W^{(t)}_{(l,i,j)} = W^{t-1}_{(l,i,j)} - \eta \frac{\partial \alpha_{(l,i,j)}}{\partial W_{(l,i,j)}}, \tag{2.4}$$

$$b^{(t)}_{(l,j)} = b^{t-1}_{(l,j)} - \eta \frac{\partial \alpha_{(l,i,j)}}{\partial b_{(l,j)}}. \tag{2.5}$$

Em que t representa a nova atualização da iteração, logo $(t-1)$[2] representa uma iteração anterior; η é a taxa de aprendizado ou atenuação, que funciona como um "freio" para o gradiente, à medida que se aproxima da região de mínimo. Este processo descrito até

[2] Esta notação foi modificada em relação à original (BISHOP, 2016)

aqui é repetido até que sejam satisfeitas uma ou mais condições, como por exemplo até que o erro seja menor que um limiar, ou até que as iterações cheguem a uma quantidade pré-determinada. É importante ressaltar que esta é a forma mais simples de se fazer o ajuste dos pesos (pelo gradiente descendente). Atualmente existem diversos algoritmos que utilizam outras técnicas. Além disso, caso sejam utilizadas funções de ativação não lineares nas camadas, pode ser necessário utilizar a regra da cadeia caso a função seja composta.

2.3.2 Otimizadores

Os otimizadores são os diferentes algoritmos utilizados para retro-propagação dos erros pelos parâmetros do modelo. De acordo com a técnica empregada a convergência será efetuada de diferentes formas. A seguir, serão discutidos os métodos de primeira ordem, ou seja, que se baseiam na primeira derivada (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.2.1 Gradiente Descendente

Este algoritmo é um dos mais conhecidos e utilizados para realizar o processo de ajuste e procura de mínimos de funções, uma vez que ele busca a direção contrária ao máximo crescimento de funções/superfícies de parâmetros[3].

O gradiente descendente e outros algoritmos otimizadores possuem três formatos de operação, que se diferem em relação à quantidade de dados utilizados para determinar o gradiente (RUDER, 2016; BISHOP, 2016; ZHANG *et al.*, 2020a).

1) Gradiente descendente em lotes (do inglês, *Batch Gradient Descendent*):

   Para um dado conjunto de treinamento, o gradiente será computado para todas as amostras e somente depois será realizado somente uma atualização a cada época. Este método pode ser demasiadamente lento ou até mesmo infactível para bases de dados que não cabem na memória de uma só vez, a atualização dos parâmetros pode ser generalizada pela equação 2.6,

$$\theta_{(t+1)} = \theta_t - \eta \nabla_\theta E(\theta_{(t)}), \tag{2.6}$$

[3] Quando a dimensionalidade do banco de dados é de alta ordem, a fronteira de separação e os parâmetros internos do modelo também tendem a ser, podendo ser uma superfície ou hiper-espaço

em que θ representa os parâmetros; η representa a taxa de aprendizado do algoritmo, ela fará o papel de ponderação sobre a atualização dos pesos;∇E representa o conjunto das derivadas parciais matricial (matriz Hessiana) e t representa a iteração.

2) Gradiente descendente estocástico (do inglês, *Stochastic Gradient Descendent*):

Neste formato a retro-propagação dos erros é realizada a cada nova amostra do conjunto de treino. É uma alternativa para aumentar a velocidade de processamento a cada iteração e pode ser utilizado para atualização online (à medida que novas amostras são coletadas). Porém este algoritmo pode realizar atualizações redundantes caso as amostras sejam muito semelhantes, ou realizar ajustes bruscos nos pesos, casos as amostras sejam muito distintas, por isso a curva de decaimento do erro associado, muitas vezes apresenta flutuações. A equação de atualização para este algoritmo pode ser escrita da conforme a equação 2.7,

$$\theta_{(t+1)} = \theta_t - \eta \nabla_\theta E(\theta_{(t)}; x_{(i)}; y_{(i)}), \tag{2.7}$$

em que x representa a amostra e y representa o rótulo a ela associado;

3) Gradiente descendente em mini lotes (do inglês, *Mini-Batch Gradient Descendent*):

Este é o formato mais comumente encontrado em aplicações práticas e publicações científicas, nele, uma quantidade (geralmente baixa) de subconjuntos é predefinida, o treinamento ocorre e o gradiente é atualizado até que a quantidade seja alcançada, somente após este requisito satisfeito que o algoritmo irá realizar a atualização dos pesos. A equação de atualização dos pesos fica conforme a equação 2.8,

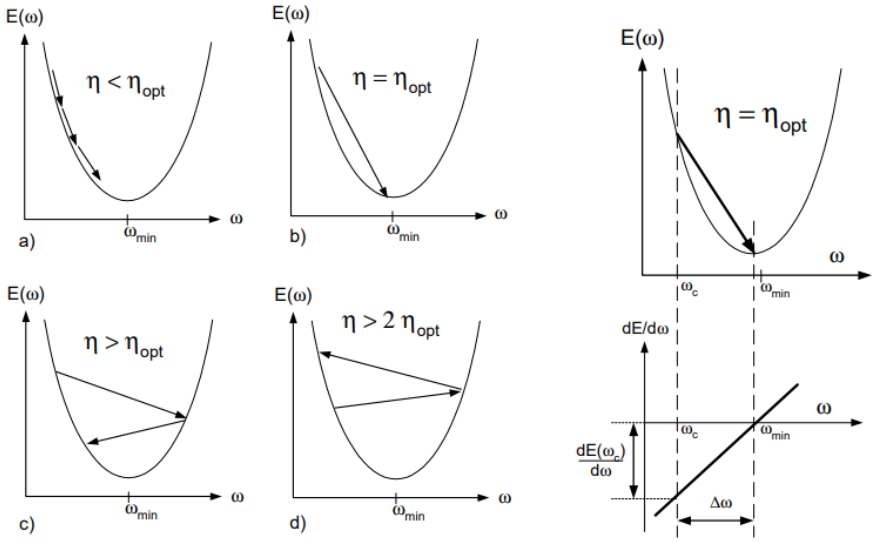
$$\theta_{(t+1)} = \theta_t - \eta \nabla_\theta E(\theta_{(t)}; x_{(i:i+n)}; y_{(i:i+n)}), \tag{2.8}$$

em que x representa a amostra e y representa o rótulo a ela associado; n representa a quantidade de amostras no lote.

Apesar de resolver muitos problemas de otimização e ser amplamente utilizado, o algoritmo gradiente descendente possui algumas desvantagens, como por exemplo, o efeito zigue-zague que ocorre devido à perpendicularidade entre vetor gradiente e o contorno do hiper-espaço de parâmetros. Logo é importante enfatizar o termo η (taxa de aprendizado), que pondera a ação do gradiente sobre os pesos, quanto menor for este valor, menor será

a tendencia de “zigue-zague” sobre o gradiente, porém a quantidade de iterações para o algoritmo convergir se torna muito maior, dependendo do valor de η. Na Figura 2.5 pode ser observado a ação da taxa de aprendizado em relação ao erro para diferentes valores de η. Na qual $E(\omega)$ representa o erro associado na saída no processo de aprendizado de

Figura 2.5 – Diferentes valores de taxa de aprendizado (η) e suas implicações.



Fonte: (LECUN *et al.*, 1998).

retropropagação; η_{opt} é a taxa de aprendizado ótima; ω representa os hiper-parâmetros do modelo. A implicação na Figura 2.5.a) é que se for atribuído uma taxa com valor menor que o ótimo, a tendência é que a convergência seja mais lenta, com a vantagem de mitigar ou até eliminar o efeito “zigue-zague” (Observado na Figura 2.5.c) e Figura 2.6.a)). Porém, caso o valor escolhido seja muito baixo, a convergência será muito lenta. O caso ideal seria escolher a taxa de aprendizado ótima, (que não é conhecida a priori) pois ela leva a convergência em uma iteração (Figura 2.5.b)); E o pior caso é utilizar uma taxa de aprendizado alta, que leva à divergência do algoritmo, Figura 2.5.d).

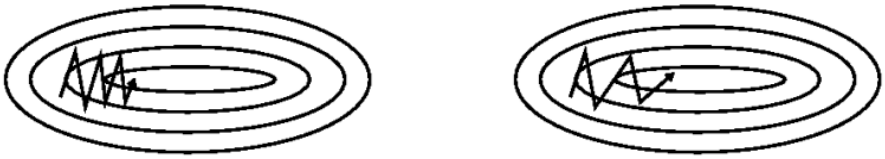
Existem diversos algoritmos e estratégias propostas para contornar estes desafios que foram abordados (tratar do problema “zigue-zague” e da taxa de aprendizado), alguns deles serão discutidos a seguir (RUDER, 2016).

2.3.2.2 Gradiente Descendente com Momento

O algoritmo gradiente descendente tem problemas para operar em regiões em que a superfície é mais ingrime em uma dimensão do que nas outras (o que é comum em torno de ótimos locais). Nesses cenários, o algoritmo oscila pelo hiper-plano enquanto

faz progressão em direção ao ponto de ótimo. Uma alternativa encontrada é a técnica de momento, que advém do conceito físico de que o algoritmo pode ganhar velocidade à medida que converge para o ponto de ótimo. Isto pode ser realizado pela adição de uma fração do vetor de atualizações da iteração passada na iteração atual. Logo, a medida que as iterações vão passando, o vetor gradiente pode obtém maior momento nos sentidos em que os gradientes apontam, amortecendo as oscilações e reduzindo a quantidade de iterações necessárias para convergência, como mostrado na Figura 2.6.

Figura 2.6 – Exemplo de aplicação de momento no gradiente descendente sem momento a), e com momento b).



(a) SGD without momentum (b) SGD with momentum

Fonte: (RUDER, 2016)

Como é possível observar na Figura 2.6, quando se aplica o momento, a convergência tende a se tornar mais suave. Porém em alguns casos a convergência é demorada, mesmo com adição de momento. Para isto, algumas estratégias foram propostas. A equação para o gradiente descendente pode ser dada conforme (RUDER, 2016).

$$\begin{aligned} v_{t+1} &= \gamma v_{(t)} + \eta \nabla_\theta E(\theta_{(t)}), \\ \theta_{(t+1)} &= \theta_{(t)} - v_{t+1}, \end{aligned} \tag{2.9}$$

em que v_{t+1} corresponde ao novo vetor de momento, que é ponderado pelo vetor gradiente; γ corresponde à fração do vetor de atualização anterior (v_t) para o próximo vetor (v_{t+1}); (θ_t) corresponde aos parâmetros do modelo e ∇E ao gradiente do erro, tomado sobre uma função adequada ao problema (função objetivo).

2.3.2.3 AdaGrad

O *AdaGrad* é um algoritmo para gerar um gradiente adaptativo, e isto é feito por meio da modificação na equação original do gradiente descendente (equação (2.6)). Seja a

equação (2.10) (RUDER, 2016).

$$\theta_{(t+1,i)} = \theta_{(t,i)} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \odot \nabla_\theta E(\theta_{(t,i)}), \tag{2.10}$$

substituindo $g = \nabla_{\theta_t} E(\theta)$, equação fica como mostra a equação 2.11,

$$\theta_{(t+1,i)} = \theta_{(t,i)} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \odot g_{t,i}, \tag{2.11}$$

na qual, nota-se que a taxa de aprendizado agora está sendo dividida pelo termo $G_{t,ii}$, sendo que $G_{t,i}$ é uma matriz diagonal em que cada elemento t,i, corresponde a soma dos quadrados dos gradientes em relação aos parâmetros θ_i. E ϵ é um termo atenuante que evita a divisão por zero (geralmente $\epsilon \approx 1e-8$). O símbolo $\odot$ se trata do produto elemento a elemento de *Hadamard*. É importante salientar que o momento não é utilizado no *AdaGrad* originalmente, somente em versões modificadas. Além disso, principal desvantagem do *AdaGrad* é o acúmulo do gradiente elevado ao quadrado no denominador: a soma acumulada continua crescendo durante o treinamento e, dependendo do seu valor, pode fazer com que a taxa de aprendizado diminua demasiadamente, fazendo com que o modelo pare de aprender[4]. Os algoritmos a seguir visam resolver essa falha.

2.3.2.4 Algoritmo RMSPROP

O algoritmo RMSPROP (do inglês, *Root Mean Squared Propagation*), tem a proposta de modificar o algoritmo *AdaGrad* para obter melhor performance em configurações não convexas. Quando aplicado a uma função não convexa para treinar uma rede neural, a trajetória de aprendizado pode passar por diversas estruturas e eventualmente chegar a uma região localmente convexa. A atualização fica conforme a equação 2.12 (RUDER, 2016).

$$\begin{aligned} &E[g^2]_t = 0,9E[g^2]_t + 0,1g_t^2, \\ &\theta_{(t+1)} = \theta_{(t)} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t, \\ &\text{Em que: } g = \nabla_{\theta_t} E(\theta) \end{aligned} \tag{2.12}$$

[4] Este problema é conhecido como esvaecimento do gradiente (do inglês, *Vanishing Gradient*).

2.3.2.5 ADAM

Entre diversos outros otimizadores existentes, um método que se destacou e é muito utilizado, é o método de Estimação de Momento Adaptativo (do inglês, *Adaptive Moment Estimation* - ADAM). Este método está entre os métodos mais utilizados para suavizar o gradiente descendente, que por si só, tem o efeito de maximizar o aprendizado da rede via retro-propagação. Basicamente, este método é união do método RMSPROP com cálculo do momento. O equacionamento pode ser dado conforme a equação 2.13 (RUDER, 2016; KINGMA; BA, 2015).

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned} \tag{2.13}$$

em que m_t é uma estimativa do primeiro momento estatístico (a média) e v_t é a estimativa do segundo momento (a variância); β_1 e β_2 são constantes próximas de 1, os autores recomendam $0,9$ para β_1 e $0,999$ para β_2. Como estes vetores são inicializados com valores nulos, os autores observaram que se tornam enviesados ao entorno de zero; e para contornar este problema de viés, pode ser realizado o seguinte ajuste, dado pela equação 2.14,

$$\begin{aligned} \hat{m}_t &= \tfrac{m_t}{1-\beta_1^t}, \\ \hat{v}_t &= \tfrac{v_t}{1-\beta_2^t}, \end{aligned} \tag{2.14}$$

estes termos são então utilizados para atualizar os parâmetros, que fica conforme a equação 2.15.

$$\theta^{(t)} = \theta^{(t-1)} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} m_{t-1}, \tag{2.15}$$

desta forma, o algoritmo proposto em Kingma e Ba (2015) tem a capacidade de agregar a técnica de RMSProp que confere um direcionamento para o gradiente, juntamente com a técnica de momento, que faz com que o algoritmo ganhe velocidade na direção do ponto de mínimo. Além disso, o algoritmo também tem gerado diversos trabalhos promissores como o ADAMW e SGDW, que consiste na proposta de aplicar a regularização de forma desacoplada nos otimizadores (LOSHCHILOV; HUTTER, 2018). Ou também os métodos adaptativos, como o método proposto em Luo *et al.* (2019), em que as taxas de aprendizado possuem uma zona adaptativa dinâmica. E por fim, algumas propostas estão surgindo na tentativa de agregar os bons resultados que geralmente são obtidos somente com gradiente

descendente (porém com maior lentidão), com a rapidez na convergência de métodos como ADAM, ADAMAX, etc; (ZHUANG *et al.*, 2020; LANDRO; GALLO; GRASSA, 2020).

Além dos algoritmos otimizadores aqui mostrados (de primeira ordem), vale a pena mencionar algoritmos interessantes propostos na literatura como os algoritmos de segunda ordem, *Levenberg Marquadt*, *Curve Ball*, Método de *Newton*, L-BFGS (*Lower Memory - Broyden Fletcher Golfarb Shanno*), entre outros. Os algoritmos de segunda ordem são mais sensíveis à curvatura de otimização que os algoritmos de primeira ordem, por isso em muitos casos conseguem convergir mais rápido, porém possuem uma atualização de segunda ordem, o que limita sua aplicação em bancos de dados de alta dimensionalidade, já que a arquitetura necessita de uma maior quantidade de parâmetros, e consequentemente aumenta-se a quantidade de cálculos por iteração (HENRIQUES *et al.*, 2018; TAN; LIM, 2019; Soham Pal, 2021).

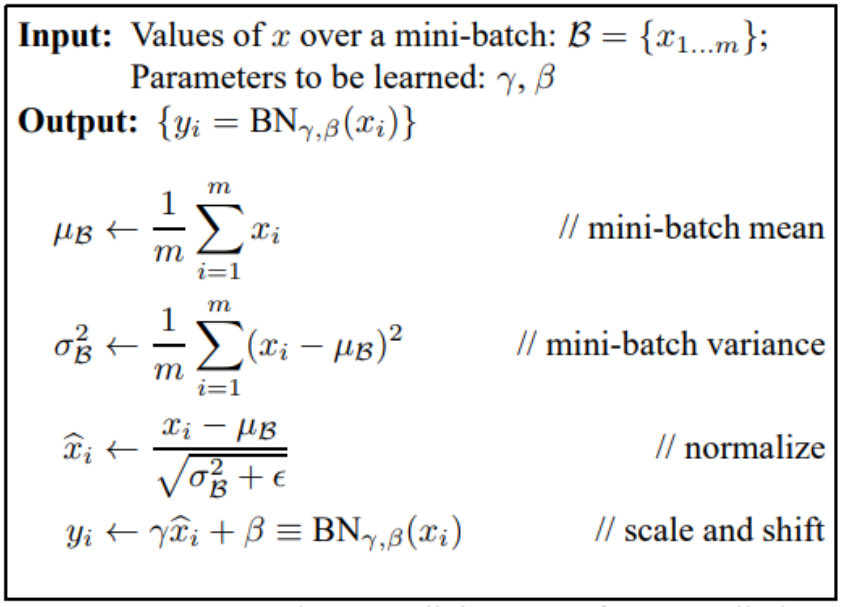
2.3.3 Estratégias Adicionais

Além das estratégias mencionadas acima em relação ao funcionamento do algoritmo, existem estratégias para otimizar os parâmetros ao longo do processo de treinamento *Batch Normalization*; e também métodos para evitar com que o modelo "decore" os dados, como é o caso do *early stopping*.

2.3.3.1 Normalização em Lotes

A normalização em lotes ou *Batch Normalization* configura uma estratégia muito utilizada e que pode auxiliar a mitigar o problema de deslocamento da covariância interna (SANTURKAR *et al.*, 2018). Ela pode ser realizada a cada n amostras, subtraindo a média e dividindo pela variância, e realizando a normalização de cada saída após a função de ativação. o processo faz com que as camadas subsequentes recebam uma amostra normalizada. Segundo os autores Santurkar *et al.* (2018), o algoritmo *Batch Normalization* (Figura 2.7) reduz o deslocamento da covariância interna do modelo. Além disso, não só a covariância interna é reduzida, ajudando a aumentar a eficiência do treinamento, como também a torna superfície de otimização do problema significativamente mais suave.

Figura 2.7 – Algoritmo *Batch Normalization.*



Fonte: (IOFFE; SZEGEDY, 2015).

2.3.3.2 *Early Stopping*

Esta técnica de parada antecipada é muito utilizada para não deixar que o modelo decore os dados, e por muitas vezes é mencionada como técnica de regularização. Quando o modelo está em fase de treinamento, ao final de cada época é realizada uma etapa de validação, para avaliar a generalização do modelo. Caso o erro de validação (ou algum outro parâmetro de escolha) fique N épocas (pré-definidas pelo desenvolvedor) sem um decaimento tanto quanto considerável, ou melhor, caso a taxa de decaimento do erro de validação for menor que um patamar ϵ, o treinamento pode ser interrompido ou modelo armazenado, para evitar o sobre-ajuste; em muitos bancos de dados reais, com a evolução do treinamento, pode ocorrer o aumento no erro de validação do modelo (RUDER, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.3.3 Taxa de Aprendizado Variável

Em alguns casos, quando se deseja evitar que algum parâmetro (erro de validação, acurácia, entre outros) de treinamento do modelo fique estagnado ou flutuando sobre um valor, devido a uma taxa de aprendizado alta; para estes casos, algum erro ou métrica pode ser monitorado para reduzir a taxa de aprendizado do modelo durante o treinamento.

2.4 Redes Neurais "Convolucionais"

As redes neurais convencionais totalmente conectadas descritas na seção 2.3, também são chamadas de densas por tratarem de uma topologia na qual todas as entradas de cada camada possuem todos os neurônios conectados à camada seguinte. Em aplicações nas quais o banco de dados possui uma maior dimensionalidade, como é o caso da maioria dos bancos de dados de imagens. Nestes casos, uma rede densa convencional (em sua forma mais genérica) se torna uma solução não tão eficiente quanto outras alternativas desenvolvidas, como é o caso das redes neurais convolucionais.

Estas redes foram propostas pela inspiração biológica no córtex visual dos animais, que são sensíveis a sub-regiões do campo visual (HUBEL; WIESEL, 1962; FUKUSHIMA; MIYAKE, 1982). O método consiste em utilizar filtros (matriz de pesos nxn) também chamados de *kernels*, porém ao invés de serem fixos, os valores são atualizados de forma iterativa, à medida em que as épocas vão se passando. A forma de aprendizado deste tipo de rede ocorre da mesma maneira como nas redes de multicamadas densas, pela retro-propagação do erro gerado no processamento das amostras. Como o filtro não é fixo, muitos autores e programadores preferem utilizar a operação de correlação cruzada 2.16, computacionalmente mais simples por não ter a necessidade de inversão do filtro a cada operação. As equações para determinar as saídas das operações de correlação cruzada e convolução 2.17 espacial são muito similares, como mostrado em (ZHANG *et al.*, 2020a).

$$\text{Correlação}(f*g)(i,j) = \sum_a \sum_b f(a,b)g(i+a,j+b) \tag{2.16}$$

$$\text{Convolução}(f*g)(i,j) = \sum_a \sum_b f(a,b)g(i-a,j-b). \tag{2.17}$$

Como pode ser observado, as operações são muito semelhantes, a única diferença matemática é o sinal invertido no somatório, que na prática provoca uma mudança na disposição do filtro para operação de convolução, tanto na horizontal quanto na vertical. Em alguns casos simples para detecção de borda por exemplo, utilizar o procedimento de convolução espacial, somente fará a inversão do *kernel* como é provado por Rachel Draelos (2019). Mas em aplicações de DL e ML, a maioria das bibliotecas utilizam a correlação cruzada, eliminando a necessidade de alternar a disposição dos filtros/*kernels* a cada iteração, o que reduz a complexidade computacional do modelo, e, ao fim do treinamento não se tem filtros dispostos de forma invertida.

2.4.1 Operações Elementares

Dentro das aplicações de redes neurais convolucionais, existem algumas operações fundamentais, como a operação de preenchimento (do inglês, *padding*), a operação de passo (do inglês, *stride*), a operação de mapeamento ou agrupamento (do inglês, *pooling*), entre outras. Estas operações e outras mais elaboradas estão detalhadas no apêndice A.2.

Após definidos os principais conceitos necessários para entendimento dos modelos de aprendizado profundo, definem-se então os conceitos sobre mecanismos de atenção, que são operações fundamentais para arquitetura *Vision Transformer*, que vem se destacando em tarefas de aprendizado profundo, e grande concorrente das redes neurais convolucionais (DOSOVITSKIY *et al.*, 2020). Elas serão melhor detalhadas na seção seguinte 2.5.

2.5 Mecanismos de Atenção

Os mecanismos de atenção foram primeiramente propostos para o processamento de linguagem natural (NIU; ZHONG; YU, 2021). Neste contexto as palavras recebem codificações (*tokens*) que são valores exclusivos para cada palavra, tornando possível seu processamento[5] (ZHANG *et al.*, 2020a; BAHDANAU; CHO; BENGIO, 2014).

Do ponto de vista psicológico, a relação dos objetos em um ambiente visual e sua subjetiva[6] chamada de atenção poder ser classificada em duas categorias: os objetos que causam alerta voluntário e involuntário de atenção.

Os objetos no campo receptivo que causam interesse imediato ao primeiro contato visual são os alertas involuntários de atenção (do inglês, *Non-Volitional Cues*), enquanto que os objetos que dependem do processo cognitivo consciente para receber atenção, são denominados alertas voluntários de atenção (do inglês, *Volitional Cues*) (ZHANG *et al.*, 2020a). Uma forma encontrada para relacionar os alertas de atenção com as redes neurais foi pelas consultas, chaves e valores:

1. Consultas: No contexto dos mecanismos de atenção, os alertas voluntários são relacionados às consultas.

2. Chaves e Valores: Os alertas involuntários são definidos como chaves e valores.

2.5.1 Atenção em Visão Computacional

Do ponto de vista de visão computacional, os mecanismos de atenção compõem uma abordagem diferente do que foi proposto nas operações de "convolução", isto porque os mecanismos de atenção se baseiam em diferentes tipos de operações matemáticas (ZHANG *et al.*, 2020a). Enquanto nas arquiteturas CNNs são utilizados filtros de diferentes tamanhos para extrair mapas de características, com o produto escalar entre filtro e entrada; nos mecanismos de atenção são utilizadas três matrizes de pesos (*queries, keys and values*), as quais são correlacionadas para gerar mapas de atenção.

[5] Neste tipo de processamento são aplicados algoritmos como *bag of words*, TF-IDF, *Word2Vec*, entre outros.

[6] Como a atenção que cada indivíduo dá para um determinado objeto ou assunto é diferente, então o alerta de atenção se torna subjetivo. Em DL o alerta depende do objetivo do problema.

Seja uma imagem de dimensão (n_h , n_w, n_c)[7], se fosse criar um mapa de atenção para cada pixel em relação aos demais, (com somente uma camada de atenção) seriam necessárias $(n_h * n_w)^2 * n_c$ parâmetros, o que na prática tornaria a aplicação inviável para imagens de maior dimensão, dada uma arquitetura com maior quantidade de camadas (DOSOVITSKIY *et al.*, 2020). Algumas estratégias foram tomadas para contornar este problema. Como por exemplo o *Vision Transformer* que sub-divide a imagem em sub-recortes e acopla uma codificação de posição ao subconjunto de pixels (DOSOVITSKIY *et al.*, 2020). Ou também o modelo *Swim-Transformer* que cria mapas de atenção deslizantes pela imagem (LIU *et al.*, 2021). Ou também o modelo *Transformer in Transformer* proposto em Han *et al.* (2021), que assim como o *Vision Transformer*, subdivide a imagem, mas trata das posições pixel a pixel, o que gera uma maior quantidade de operações e maior consumo de memória.

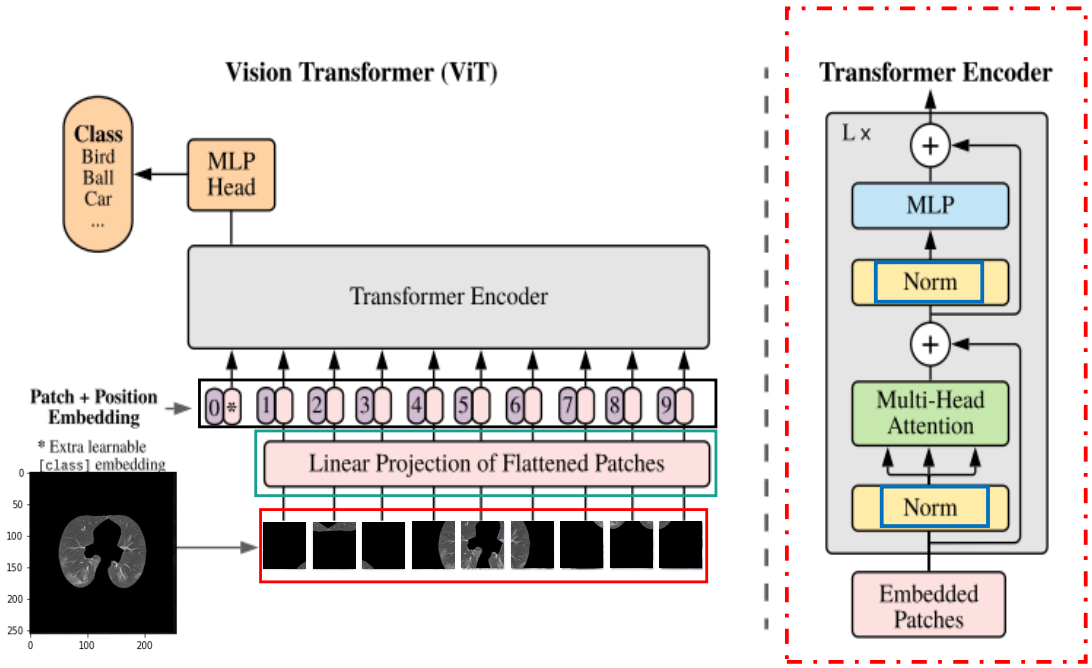
2.5.2 *Vision Transformers*

A arquitetura *Transformer* vem tomando destaque em tarefas de Visão Computacional, como a classificação de objetos. Conforme Dosovitskiy *et al.* (2020) as imagens são subdivididas em partes menores, denominadas remendos ou recortes (destacado em vermelho na Figura 2.8). Em seguida cada recorte é reorganizado de forma linear (destacado em verde na Figura 2.8) e agregado a sua respectiva codificação de *patch*[8], de posição, e codificação de classe (destacado em preto na Figura 2.8). A Figura 2.8, exibe a ideia geral do fluxo de processamento de uma imagem sobre o modelo *Vision Transformer* adaptado para imagens segmentadas de exames de tomografia.

[7] Na qual n_h corresponde a quantidade de pixels na vertical (altura), n_w à quantidade na horizontal (largura) e n_c é a quantidade de canais

[8] Termo em inglês para se referir a cada sub-recorte da imagem ou "remendos".

Figura 2.8 – Diagrama da arquitetura *Vision Transformer*.



Adaptado de: (DOSOVITSKIY *et al.*, 2020)

Após realizada a subdivisão da imagem de entrada e cada um dos recortes for concatenado com suas respectivas codificações, os dados passam por múltiplas camadas *Transformer Encoder* (destacado em vermelho tracejado na Figura 2.8). Cada camada possui operações intercaladas de normalização ("*Norm*"), (destacadas em azul na Figura 2.8), estas etapas de normalização evitam que os valores "explodam" para infinito. Após o bloco de normalização, tem-se o bloco de *Multi Head Attention*, este bloco faz o cálculo dos mapas de atenção de cada recorte da imagem, também chamados de vetores de contexto. Por fim, a saída do *Encoder* é classificada por uma MLP (detalhada na seção 2.3). A seta que parte da entrada para o somatório (região demarcada em vermelho tracejado), e do somatório ao final, correspondem a conexões de salto (do inglês, *skip connection*) são conexões que adicionam a informação de uma camada anterior em uma camada seguinte. O equacionamento do *Vision Transformer* são definidas conforme as equações (2.18, 2.19, 2.20, 2.21).

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \cdots; x_p^N E] + E_{pos}, \quad E \in \mathbb{R}^{(P^2 \cdot C) \times D} \quad E_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{2.18}$$

$$z'_l = MSA(LN(z_{l-1})) + z_{l-1}, \quad l = 1 \cdots L \tag{2.19}$$

$$z'_l = MLP(LN(z'_l)) + z'_l, \quad l = 1 \cdots L \tag{2.20}$$

$$y = LN(z_L^0). \tag{2.21}$$

Em que z_0 corresponde à junção de todos os subconjuntos da imagem codificados; x_{class} corresponde a uma codificação de aprendizado de classe, se trata de um parâmetro que também é atualizado durante o treinamento; x_p representa cada um dos subconjuntos redimensionados em formato de vetores[9]; E corresponde às codificações de cada subconjunto/*patch*; E_{pos} se refere à codificação de posição; P é a dimensão de cada *patch*; C se trata da quantidade de canais que a entrada possui; D corresponde a um valor constante que corresponde à dimensão do vetor latente; N corresponde à quantidade de *patches* gerados e pode ser determinado por $N = HW/P^2$ (Em que H é altura e W é a largura da imagem). LN significa *LayerNorm* que é a normalização intercalada entre as etapas; z'_l corresponde à saída de cada bloco MSA que significa *Multiheaded Self Attention*, que corresponde a múltiplos blocos de auto atenção; MLP se refere à rede *perceptron* multicamadas. As equações para MSA são definidas conforme equações (2.22, 2.23, 2.24):

$$[q, k, v] = zU_{qkv} \qquad U_{qkv} \in \mathbb{R}^D \times D_h, \tag{2.22}$$

$$A = softmax(qk^T/\sqrt{D_h}) \qquad A \in \mathbb{R}^{N \times N}, \tag{2.23}$$

$$SA(z) = Av. \tag{2.24}$$

Para cada elemento de entrada $z \in \mathbb{R}^{N \times D}$, é realizada a soma sobre todos os valores v em sequência. Os pesos de atenção $A_{i,j}$ são baseados na semelhança par a par entre dois elementos da sequência de entrada e suas respectivas representações de consulta q^i e chave k^j; D_h é geralmente definido por D/k em que k é a quantidade de *heads* utilizadas, ou blocos de *self attention*.

2.5.3 Interpretabilidade

Os modelos de DL são modelos complexos e quando utilizados em bases de dados reais, podem sofrer com algum tipo de viés. Além disso os modelos funcionam como uma "caixa preta", ou seja, na maioria dos casos não se sabe o porquê da resposta. Nesse contexto, a área que desenvolve algoritmos e métodos para interpretabilidade de modelos tem tomado maior atenção por parte de pesquisadores, principalmente que trabalham com modelos

[9] Como a imagem tem seus pixels distribuídos de forma tabular, o algoritmo percorre a matriz em cada i-linha, e elas são concatenadas em sequência

de DL. A ideia central é solucionar o mistério e descobrir quais são os parâmetros mais relevantes de um modelo que, quando submetido a uma determinada entrada e fornece uma saída ou rótulo correspondente, por exemplo. Dado um modelo que foi treinado com imagens de gatos e cachorros, e recebe ao final do treinamento, imagens sem nenhum dos dois tipos de animais, e classifica como sendo cachorro fracamente, por exemplo. Os métodos de interpretabilidade vêm justamente buscar explicar quais pixels da imagem são utilizados pelo modelo para classificação final. E em seguida "estressar" o modelo com imagens sem os tais pixels para avaliar o método.

Para os modelos baseados em mecanismos de atenção, pode ser utilizado, por exemplo, o método de Propagação de Relevância em Camadas (do inglês, *Layer-Wise Relevance Propagation* - LRP). O método LRP cria mapas de saliência que representa a relevância de cada pixel de entrada para a saída da rede. Seja uma saída $f(x)$ de uma rede profunda f, sobre uma entrada $x = (x_1, x_2, \cdots, x_N)$; de tal modo que $f(x) = \sum_i r_i$ (na qual r_i representa a pontuação da entrada x_i). Uma das variações do método LRP, é conhecida como decomposição profunda de *Taylor*, ele se baseia no fato de f ser diferenciavel, logo ele pode ser aproximado por uma expansão de *Taylor* de f em alguma raiz de $\hat{x}$ para qual $f(\hat{x}) = 0$, A expansão de *Taylor* pode ser dada conforme a equação (2.25) (XIE *et al.*, 2020).

$$\begin{aligned} f(x) &= f(\hat{x}) + \nabla_{\hat{x}} f \cdot (x - \hat{x}) + \epsilon \\ f(x) &= \sum_{i}^{N} \frac{\partial f}{\partial x_i}(\hat{x}_i) \cdot (x - \hat{x}) + \epsilon \end{aligned} \tag{2.25}$$

em que ϵ encapsula todos os termos de segunda ordem e superiores na expansão de *Taylor*. A pontuação de relevância para entradas pode ser determinada pela equação (2.26).

$$r_i = \frac{\partial f}{\partial x_i}(\hat{x}_i) \cdot (x - \hat{x}) \tag{2.26}$$

Este resultado define uma relevância de uma única camada, para extrapolar para redes profundas, é necessário utilizar o somatório das relevâncias, dado pela equação (2.27).

$$r_i^l = \sum_{j}^{M} r_{i,j}^l, \tag{2.27}$$

na qual l representa a camada, i e j representam iteradores em cada um dos nós e camadas, respectivamente.

2.5.4 Interpretabilidade em *Transformers*

Em Chefer, Gur e Wolf (2020) foi proposto um método baseado no método *Deep Taylor Decomposition* visto anteriormente, o método consiste em unir a relevância do método LRP com direcionamento por gradiente, este direcionamento é necessário para ponderar através das "cabeças" das camadas MSA, quais parâmetros possuem maior relevância global. Seja C a quantidade de classes em uma cabeça e $t \in 1, \cdots |C|$ a classe a ser visualizada. O método inicialmente consiste em propagar a relevância e os gradientes com relação à classe t. Então os gradientes ficam conforme a equação (2.28).

$$\nabla x_j^{(n)} := \frac{\partial y_t}{\partial x_j^{(n)}} = \sum_i \frac{\partial y_t}{\partial x_j^{(n-1)}} \frac{\partial x_i^{(n-1)}}{\partial x_j^{(n)}}, \tag{2.28}$$

em que $x^{(n)}$ é a entrada de cada camada $L^{(n)}$; em que $n \in [1, \cdots, N]$ é o índice da camada; $x^{(N)}$ é a entrada da rede; $x^{(1)}$ é a saída da rede; j corresponde aos elementos em $x^{(n)}$, e i corresponde a cada elemento em $x^{(n-1)}$.

Considerando $L^{(n)}(X,Y)$ as operações de uma camada sobre dois tensores X e Y, a relevância entre eles pode ser dada pela equação (2.29),

$$R_j^{(n)} = \sum_i X_j \frac{\partial L_i^{(n)}(X,Y)}{\partial X_j} \frac{R_i^{(n-1)}}{L_i^{(n)}(X,Y)}, \tag{2.29}$$

na qual j corresponde aos elementos em $R^{(n)}$; i corresponde aos elementos de $R^{(n-1)}$. Em alguns blocos da arquitetura, é utilizada a função de ativação RELU (2.8.3) Chefer, Gur e Wolf (2021), que possui resultados não negativos, para este caso a regra de propagação de relevância pode ser definida pela equação (2.30),

$$R_j^{(n)} = \sum_i \frac{x_j^+ w_{ji}^+}{\sum_{j'} x_{j'}^+ w_{j'i}^+} R_i^{(n-1)}, \tag{2.30}$$

em que x e w são as entradas das camadas e os pesos, respectivamente; o texto sobrescrito "$+$" denota a operação $max(0,v)$ como $v+$. Nos blocos que utilizam a GELU 2.8.5 ou outra função de ativação que possui derivada e saídas tanto positivas quanto negativas, a propagação de relevância pode ser dada pela equação (2.31),

$$R_j^{(n)} = \sum_{\{i|(i,j)\in q\}} \frac{x_j^+ w_{ji}^+}{\sum_{\{j'|(i,j)\in q\}} x_{j'}^+ w_{j'i}^+} R_i^{(n-1)}, \tag{2.31}$$

para inicializar a propagação da relevância, é definido um vetor *One-hot* que é basicamente um vetor binário que possui 1 na classe que foi definida de acordo com a imagem de entrada e 0 para o restante.

Para propagar a relevância entre operações de salto e matriciais, é necessário realizar esta propagação pelos dois tensores. De acordo com os autores, também foi necessário resolver a falta de conservação no mecanismo de atenção devido à multiplicação de matrizes e os problemas numéricos das conexões de salto (*skip connections*) (CHEFER; GUR; WOLF, 2021). Para resolver foram aplicadas normalizações, ficando com as equações (2.32, 2.33),

$$\overline{R}_j^{u^{(n)}} = R_j^{u^{(n)}} \frac{\left|\sum_j R_j^{u^{(n)}}\right|}{\left|\sum_j R_j^{u^{(n)}}\right| + \left|\sum_k R_k^{v^{(n)}}\right|} \cdot \frac{\sum_i R_i^{(n-1)}}{\sum_j R_j^{u^{(n)}}} \tag{2.32}$$

$$\overline{R}_k^{v^{(n)}} = R_k^{v^{(n)}} \frac{\left|\sum_k R_k^{v^{(n)}}\right|}{\left|\sum_j R_j^{u^{(n)}}\right| + \left|\sum_k R_k^{v^{(n)}}\right|} \cdot \frac{\sum_i R_i^{(n-1)}}{\sum_k R_k^{v^{(n)}}} \tag{2.33}$$

Em que $R_j^{u^{(n)}}$ e $R_k^{v^{(n)}}$ são os respectivos resultados da propagação de relevância dado pelas equações (2.31, 2.30).

2.6 Segmentação de Imagens Médicas

No contexto de imagens de exames clínicos, sejam a nível microscópico ou macroscópico, a extração da região de interesse (do inglês, *Region Of Interest - ROI*) pode ser um fator importante no diagnóstico assistido por computador (ZHANG *et al.*, 2020a; VUOLA; AKRAM; KANNALA, 2019). Quando bem planejada e organizada, a segmentação pode ter um papel substancial para análises clínicas, tanto no emprego de diagnóstico quanto de prognóstico da doença COVID-19 (SHI *et al.*, 2020). Ela é providencial para extrair ROI. Existem diversas categorias nas quais a segmentação pode vir a se dividir de acordo com Fan *et al.* (2020).

i) Manual, semiautomático e automático;

ii) Baseados em pixels ou baseados em regiões;

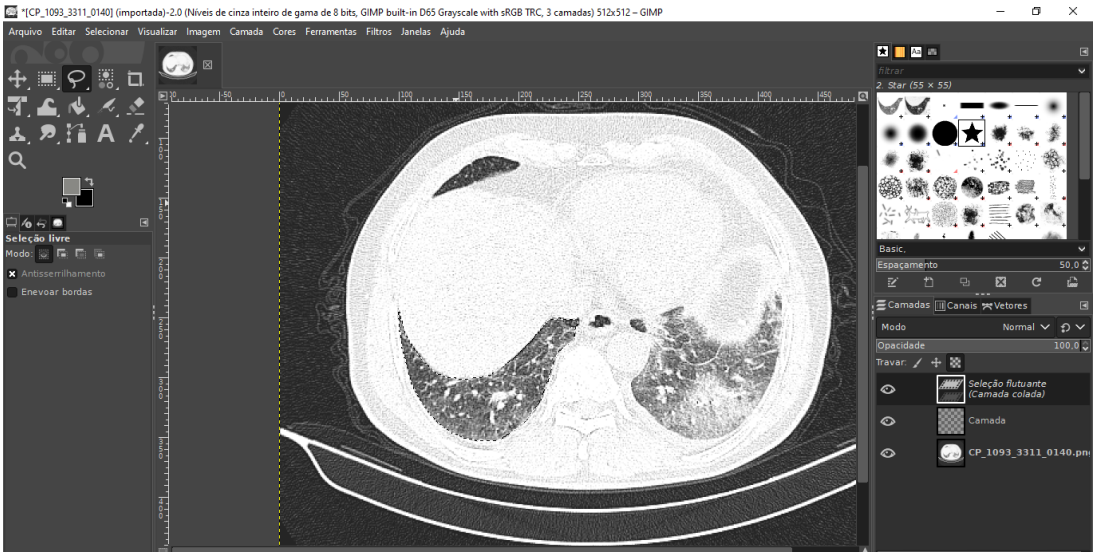
iii) Delineação Manual, segmentação em baixo nível (pelo limiar, crescimento das regiões, etc).

Em relação ao método de extração da ROI em geral, as categorias presentes no item **i)** são descritas conforme proposto em (BANKMAN, 2008).

2.6.1 Segmentação Manual

O tipo de segmentação mais simples de ser efetuada é a segmentação manual. Ela é realizada a partir de um programa de computador ou interface gráfica, capaz de selecionar pixels de uma imagem a partir de entradas (cliques com o *mouse*). Quanto maior for a habilidade do usuário em reconhecer a ROI, melhor será o resultado final. Ao atribuir N cliques que contorne a ROI, o *software* está pronto para segmentar a imagem. Como exemplo, pode ser utilizado o editor de imagens GIMP® (The GIMP Development Team (2019)). Primeiramente, a região de interesse deve ser selecionada, como mostra a Figura 2.9.

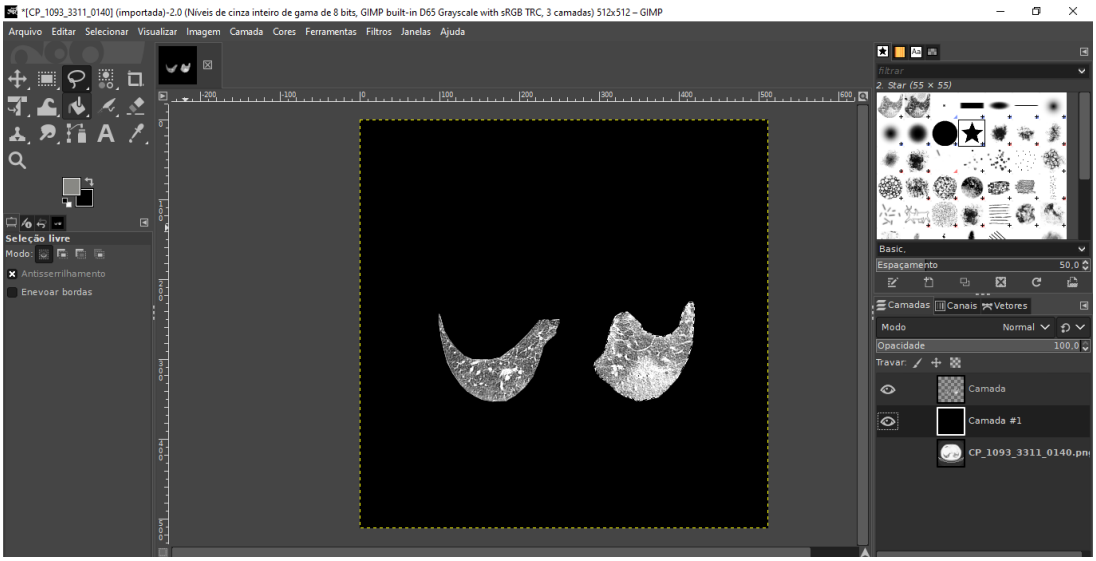
Figura 2.9 – Interface do Editor de imagens GIMP (The GIMP Development Team, 2019).



Fonte: Arquivo Pessoal do Autor.

Após selecionar toda a região de interesse, existem diversas maneiras de se extrair a seleção, uma delas é copiando para uma nova camada, fazendo com que a região de interesse fique isolada do restante da imagem. Como pode ser observado na Figura 2.10.

Figura 2.10 – Exemplo de segmentação Manual (The GIMP Development Team, 2019).



Fonte: Arquivo Pessoal do Autor.

Como pode ser observado, a segmentação manual é bem simples, sendo necessário apenas uma boa sensibilidade ao selecionar o pulmão. Porém, para casos práticos, em que se busca segmentar todo um banco de dados, a segmentação manual se torna inviável. Para

contornar esta dificuldade da segmentação manual, existem os algoritmos semiautomáticos e automáticos, que serão discutidos na seções seguintes (2.6.2, 2.6.3).

2.6.2 Segmentação semiautomática

Apesar da fama e dos algoritmos modernos baseados em aprendizado profundo terem tomado o cenário acadêmico e industrial, os métodos semiautomáticos ainda desempenham um papel muito importante para segmentação de imagens médicas, principalmente no suporte às anotações. Pois alguns dos *softwares* utilizados para gerar bancos de dados, podem fazer uso de algoritmos baseados em segmentação semiautomática para aumentar a rapidez e eficiência dos resultados.

Os métodos clássicos utilizam algoritmos de programação criados especificamente para detecção de algum aspecto da imagem, (geralmente se utilizando de morfologia matemática) e assim segmentar a mesma. Esta detecção de aspectos pode ser subdividida em duas categorias:

i) Baseada em agrupamento de regiões da imagem por homogeneidade de *pixels*: Compreende métodos como crescimento de regiões, *watershed*, contornos ativos, entre outros.

ii) Baseada na detecção de fronteiras através da descontinuidade da informação, utilizando aproximações para derivadas: Compreende os operadores de *Roberts, Prewitt* e *Sobel*, que são utilizados para detecção de bordas (CHAPLE; DARUWALA; GOFANE, 2015).

Para conferir explicações e exemplos de operações morfológicas, veja o apêndice A.3.

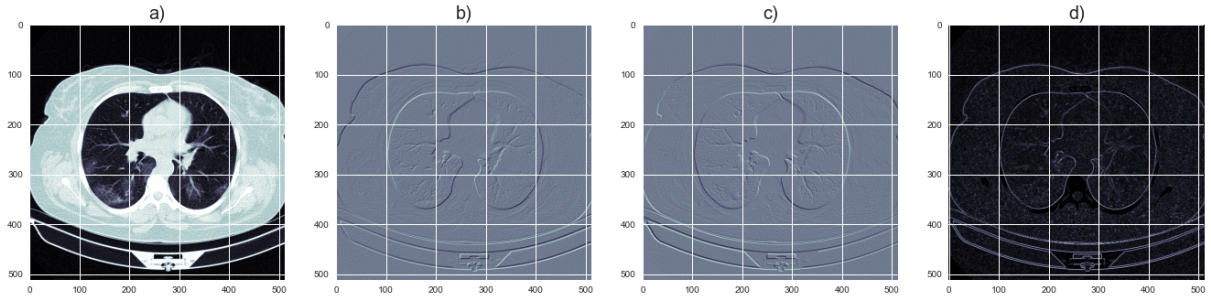
2.6.2.1 Operador *Roberts*

Um dos primeiros algoritmos desenvolvidos para detecção de bordas foi proposto em 1963 por *Lawrence Roberts* (ROBERTS, 1963), baseado na aproximação da primeira derivada. Como exemplo, para uma função $f(x,y)$ o gradiente de f nas coordenadas (x,y)

$$\textbf{a)} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \textbf{,b)} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{2.34}$$

Como pode ser observado a máscara (2.34.b) corresponde à máscara (2.34.a) rotacionada em 90º, e correspondem a 2 versões do operador de *Roberts* para detecção de bordas, sendo cada um capaz de dar ênfase nas bordas em um sentido, na horizontal a máscara (2.34.a) e na vertical a máscara (2.34.b). Aplicando ambas sobre uma imagem, com a operação de convolução espacial, e fazendo a soma, o resultado pode ser observado na Figura 2.11.

Figura 2.11 – Exemplo operador *Roberts* aplicado a uma imagem de tomografia, imagem original a), operador horizontal b), operador vertical c), e soma absoluta d).



Fonte: Arquivo Pessoal do Autor.

A Figura 2.34.d) consiste na soma absoluta das Figuras 2.34.b) e 2.34.c).

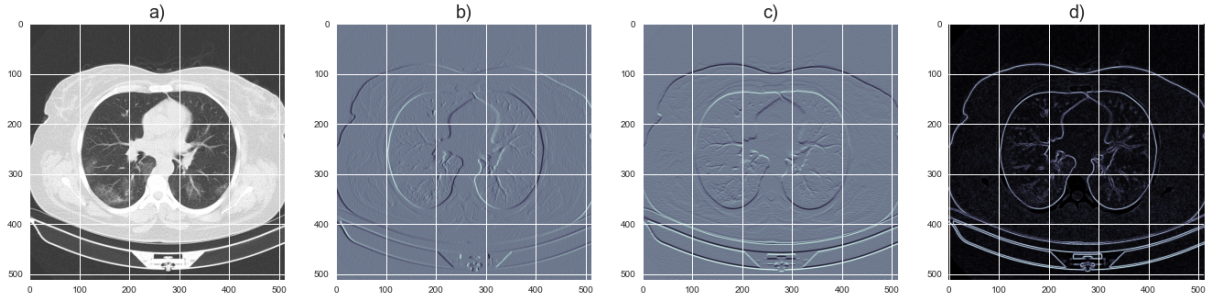
2.6.2.2 Operador *Prewitt*

O operador de Prewitt, foi desenvolvido como aproximação da derivada discreta para obter os pontos de máximo e mínimo nos *pixels* da imagem. A máscara, que por sua vez é uma matriz numérica, quando operada por convolução espacial com uma imagem, evidencia as bordas entre regiões. Como pode ser observados as máscaras (2.35), existem duas máscaras:

$$\textbf{a)}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \textbf{b)}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \tag{2.35}$$

sendo um para a vertical (máscara (2.35.a)), e outra na horizontal (máscara (2.35.b)) (CHAPLE; DARUWALA; GOFANE, 2015). Aplicando ambas as máscaras na imagem e fazendo a raiz da soma, tem-se os seguintes resultados, observados na Figura 2.12.

Figura 2.12 – Exemplo operador *Prewitt* aplicado a uma imagem de tomografia, imagem original a), resultado do operador horizontal b), resultado do operador vertical c), e soma absoluta d).



Fonte: Arquivo Pessoal do Autor.

Em que a Figura 2.12.a) se trata da imagem original, as Figuras (2.12.b) e (2.12.c) referenciam-se aos resultados dos filtros na vertical e horizontal, respectivamente. Por fim, a Figura 2.12.d) é a raiz da soma das Figuras (2.12b e c). Pelo resultado do operador de *Roberts* (na Figura 2.11.d)) comparado ao resultado obtido com operador de *Prewitt* (na Figura 2.12.d)), nota-se uma "melhora" nas linhas de contorno obtidas.

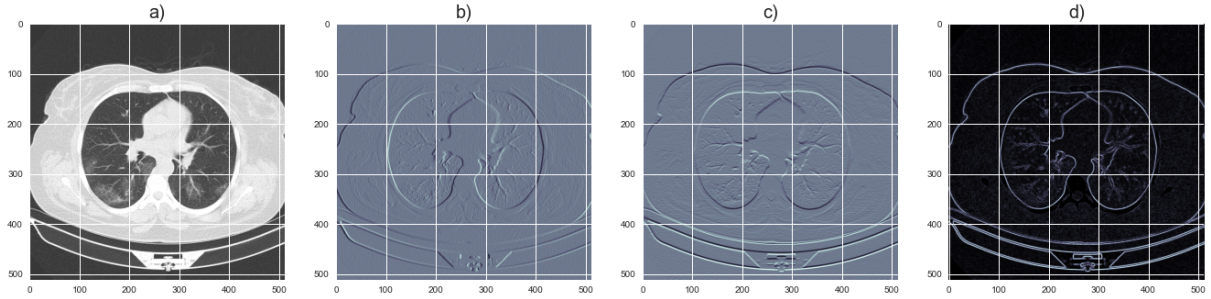
2.6.2.3 Operador *Sobel*

Em 1968, *Irwin Sobel* (SOBEL; FELDMAN, 2015), propôs uma aproximação do gradiente por meio da convolução entre uma "máscara" (matriz numérica) e a imagem. Os operadores de *Sobel* funcionam de maneira semelhante aos operadores de *Prewitt*, com a diferença do termo 2 que aparece no centro da primeira e última componente da máscara. De forma análoga, tem-se um operador para a vertical (máscara (2.36.a)), e outro na horizontal (máscara (2.36.b)).

$$\textbf{a)}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \textbf{b)}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{2.36}$$

O resultado leva a uma imagem com as bordas destacadas, tanto na vertical como na horizontal, e como o operador de *Sobel* possui o termo 2, espera-se que as bordas fiquem mais evidentes (CHAPLE; DARUWALA; GOFANE, 2015). Um exemplo do resultado pode ser visto na Figura 2.13.d).

Figura 2.13 – Exemplo operador *Sobel* aplicado a uma imagem de tomografia, imagem original a), resultado do operador horizontal b), resultado do operador vertical c), e soma absoluta d).



Fonte: Arquivo Pessoal do Autor.

Visualmente não se nota muitas diferenças entre os resultados obtidos pelos operadores *Sobel* e *Prewitt*, porém, em alguns casos quando usado iterativamente, o operador tende a ser mais sensível às bordas dos objetos presentes na imagem (CHAPLE; DARUWALA; GOFANE, 2015).

Os operadores discutidos nesta subseção (2.6.2), geralmente são utilizados em algoritmos mais elaborados, mas que em sua maioria, buscam detectar as bordas ou contornos de um determinado objeto na imagem. Em alguns casos específicos, eles funcionam para maioria das imagens de um banco de dados, e nestes casos podem ser classificados como algoritmos de segmentação automáticos.

Além disso, eles são essenciais para criação de bases de segmentação por modelos de *Deep Learning*. Na maioria dos casos são utilizadas anotações manuais, mas alguns *softwares* oferecem a possibilidade de se trabalhar com algoritmos semiautomáticos para auxiliar no processo de anotação. No apêndice A é mostrado um exemplo, ilustrando a utilização dos algoritmos semiautomáticos como ferramentas de auxílio nas anotações e geração de máscaras para criar bases de dados para segmentação.

2.6.2.4 *WaterShed* ou Linha de Partição de Águas

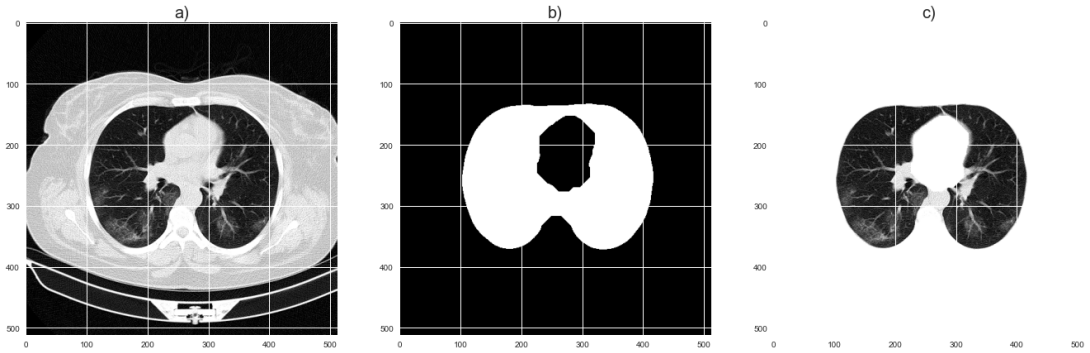
O método *watershed* é definido para imagens binárias[10] ou escala de cinza[11], que podem ser interpretadas como uma superfície topográfica, na qual os pixels mais próximos do valor máximo são os picos, e os valores próximos do valor mínimo são os vales (NAJMAN; SCHMITT, 1994).

[10] A imagem binária, também conhecida como imagem preto e branco, possui somente dois valores (e.g. 0 ou 1, 0 ou 255).

[11] A imagem em escala de cinza possui somente 1 canal com valores entre 0 e 1, ou 0 e 255

Os vales são preenchidos iterativamente como uma região homogênea, como se uma região de vale fosse preenchida por água (conjunto de pixels sob um limiar), e à medida que o nível de "água"aumenta, uma região pode emergir para outra, por isso, é necessário criar uma barreira, que delimita o nível máximo que será preenchido. O processo é repetido até que os picos estejam sob nível de “água”. Por fim, as barreiras serão o resultado da segmentação (NAJMAN; SCHMITT, 1994). Para imagens de tomografia especificamente, foi utilizado o algoritmo proposto em Meyer (1992), que consiste em uma versão modificada para operar com marcadores, ou seja, são contornos previamente detectados e utilizados como sementes iniciais para otimizar o processamento do algoritmo, ele pode ser encontrado e facilmente utilizado na biblioteca OpenCV Bradski (2000) cujo algoritmo foi desenvolvido por (SUZUKI *et al.*, 1985). Exemplifica-se a utilização do algoritmo *watershed* na Figura 2.14.

Figura 2.14 – Imagem da segmentação por *watershed* aplicado a uma imagem de tomografia, a) imagem original; b) resultado do algoritmo; e c) resultado da segmentação.



Fonte: Algoritmo adaptado de (BRADSKI, 2000).

Na Figura 2.14.c) a segmentação é efetuada sobre a imagem, existem diversas formas de se aplicar a máscara gerada na Figura 2.14.b), entre elas, pode-se realizar a multiplicação da máscara pela imagem, anulando os pixels que não correspondem à região de interesse.

Vale destacar que, após aplicar o algoritmo *watershed* sobre a imagem, também é interessante eliminar do resultado, pequenos vales que ficam nas regiões em que existem lesões ou nódulos, que ficam aparentes na região do parênquima. Esta operação pode ser realizada manualmente em um *software* de edição de imagens, ou automaticamente pela aplicação da transformação do tipo “topo de cartola preto” (do inglês, *Black Top Hat Transform*), que se trata de uma operação relacionada à morfologia matemática, tal operação consiste na diferença entre um fechamento aplicado na imagem e a própria imagem. que é definido pela equação (2.37).

$$T_b(A) = (A \bullet_b B) - A \tag{2.37}$$

Em que A corresponde ao conjunto de pixels da imagem, B representa o elemento estruturante, e b indica que a operação é binária. O elemento estruturante utilizado é definido da seguinte forma (2.38).

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}. \tag{2.38}$$

2.6.2.5 Crescimento de Regiões

Enquanto os métodos baseados em detecção de borda focam em detectar pixels com diferentes intensidades, o método de crescimento de regiões (do inglês, *Region Growing*) procura por grupos de pixels com intensidades similares. O método se baseia nas seguintes etapas:

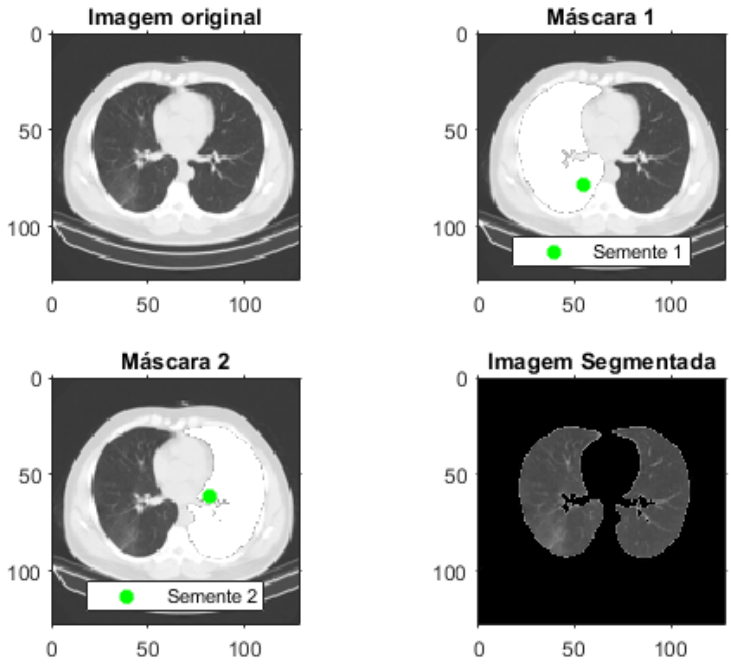
1) Escolha do critério de similaridade pelo qual os pixels serão agrupados;

2) Escolha dos pontos "sementes", que são os pontos iniciais no qual o algoritmo irá se basear para aplicar o critério de similaridade;

3) Definição do critério de parada, o qual corresponde ao limite que o algoritmo deve incorporar em homogeneidade e conectividade.

É possível obter um algoritmo automático com este método, selecionando automaticamente as "sementes" do algoritmo, como proposto em El Hassani, Skourt e Majda (2019), no qual utilizaram este método para segmentação. Um algoritmo semelhante foi adaptado de Kroon (2008), e testado. O resultado é mostrado na Figura 2.15.

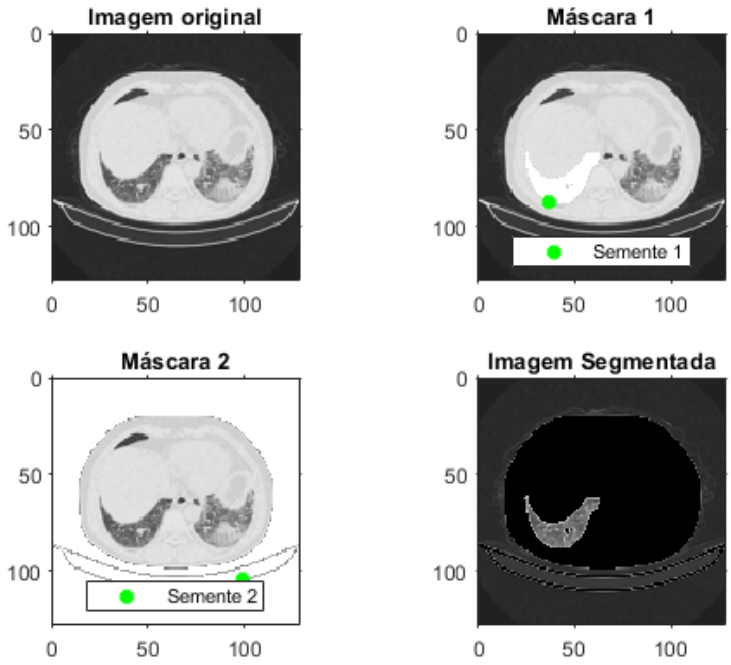
Figura 2.15 – Exemplo 1: Segmentação por Crescimento de Regiões com pulmões totalmente visíveis.



Fonte: Arquivo Pessoal do Autor.

Como pode ser observado, o processamento do algoritmo pode ser dividido em três etapas, as duas primeiras possuem a função de encontrar a semente em cada um dos parênquimas pulmonares. Estas etapas utilizam o cálculo dos centroides das imagens, que são pixels individuais que representam uma região uniforme na imagem. Estes pixels, por sua vez, são obtidos por meio de diagramas de *Voronoi* Peters (2018). Apesar de se obter um bom resultado com este método, em alguns casos as centróides não são encontradas de forma precisa, e a segmentação de um ou dos dois parênquimas não é realizada corretamente, como pode ser observado na Figura 2.16.

Figura 2.16 – Exemplo 2: Segmentação por Crescimento de Regiões com pulmões parcialmente visíveis.



Fonte: Arquivo Pessoal do Autor.

Os algoritmos semiautomáticos baseados em morfologia matemática, conseguem atingir uma qualidade razoável para uma aproximação inicial, por este motivo são muito utilizados para o auxílio nas anotações de imagens.

2.6.3 Segmentação Automática

Os algoritmos de segmentação automática mais utilizados atualmente são os algoritmos da área de aprendizado profundo, cujo objetivo é tentar absorver as principais características da imagem relacionadas às anotações da base de dados. Para entender melhor estes algoritmos, primeiramente é necessário entender como uma base de dados para segmentação é montada.

Uma base de segmentação é composta por um conjunto de imagens e suas respectivas anotações, que podem ser um conjunto de coordenadas cartesianas (vetor que delimita as bordas de um objeto de interesse), ou uma máscara binária que corresponde a todos os pixels presentes no corpo do objeto. O primeiro caso é utilizado em modelos de segmentação por instância[12]. E o segundo caso, quando a base é composta por imagens e suas respectivas

[12] É um tipo de segmentação que trabalha com a identificação de múltiplos objetos em uma mesma imagem, e por este motivo, a informação de formato e tamanho são utilizadas e importam para

máscaras binárias, são utilizadas para segmentação semântica. Neste tipo de segmentação o modelo é treinado para gerar uma máscara semelhante àquela contida na base de dados, ou seja, o modelo tenta gerar um mapa de pixels semelhante às anotações das respectivas imagens. Entre os modelos de segmentação semântica, estão o *U-Net*, *DeepLab*, *PSP-Net*, *Parse-Net*, *EcNet*, entre outros.

Para gerar uma base de dados de segmentação semântica, é necessário um *software* ou API (do inglês, Application Programming Interface), que será utilizada para delimitar a região de interesse da qual se quer extrair da imagem.

Existem *softwares* de livre acesso como o VGG (*Visual Geometry Group*) desenvolvidos nos trabalhos (DUTTA; GUPTA; ZISSERMANN, 2016; DUTTA; ZISSERMAN, 2019), bem como a ferramenta *Labelme* desenvolvida por Wada (2016), que também possui recursos interessantes para anotações de objetos em geral, mas ambas se limitam a ferramentas mais simples. Porém, para segmentação de imagens médicas, que geralmente requerem um nível maior de precisão, existem alguns softwares de livre acesso (para fins de pesquisa), como o *SLICER*-3D® (FEDOROV *et al.*, 2012). Se trata de o *software* que possui diversas ferramentas e extensões para auxiliar nas anotações; Ferramentas das quais utilizam algoritmos descritos na última subseção (2.6.2.5), como crescimento de regiões, em que o usuário delimita uma porção da imagem referente à região de interesse, e o algoritmo utiliza estas delimitações iniciais como sementes para gerar as máscaras; Ou também o *watershed*, que de acordo com delimitações feitas pelo usuário, preenche as regiões de interesse e também gera uma máscara para segmentação (FEDOROV *et al.*, 2012).

2.6.3.1 Segmentação Semântica

Como foi mencionado, a segmentação semântica tem como propósito rotular cada pixel ou conjunto de pixels de uma imagem por uma respectiva máscara (ou máscaras). Este tipo de segmentação faz uma classificação pixel a pixel para cada tipo de objeto que se deseja classificar/segmentar, e ao final do treinamento informa um grau de confidência para cada pixel em relação à determinada classe. O erro do modelo é calculado por uma máscara manualmente anotada por um especialista. À medida que o algoritmo faz o ajuste

o treinamento dos modelos, na segmentação por instância geralmente são utilizados modelos como Mask-R CNN, YOLO, Mobilenet, entre outros

dos filtros[13], com o passar das iterações, o modelo vai se tornando capaz de captar as características presentes nas imagens a nível de pixel a pixel, e ao final do processo, ele é capaz de fornecer uma máscara para cada conjunto de pixels que correspondem a uma ou mais regiões de interesse.

Geralmente são utilizadas arquiteturas do tipo auto-codificantes (do inglês, *auto-encoders* - AE), que são modelos apropriados para absorver os padrões dos dados através da codificação (ou contração) e decodificação (ou expansão) do espaço de características da imagem. Na etapa codificadora, a imagem é comprimida por camadas que realizam operações de "convolução" intercaladas por camadas que realizam operações de *Max Pooling*[14] (representadas por setas vermelhas na Figura 2.17), com filtros nxn. Na versão original proposta por (RONNEBERGER; FISCHER; BROX, 2015), utilizava-se filtros com $n = 3$ para convolução e $n = 2$ para o *Max Pooling*, porém atualmente existem diferentes variações do modelo como base para segmentação, e consequentemente diferentes filtros com dimensões diferentes podem ser encontrados.

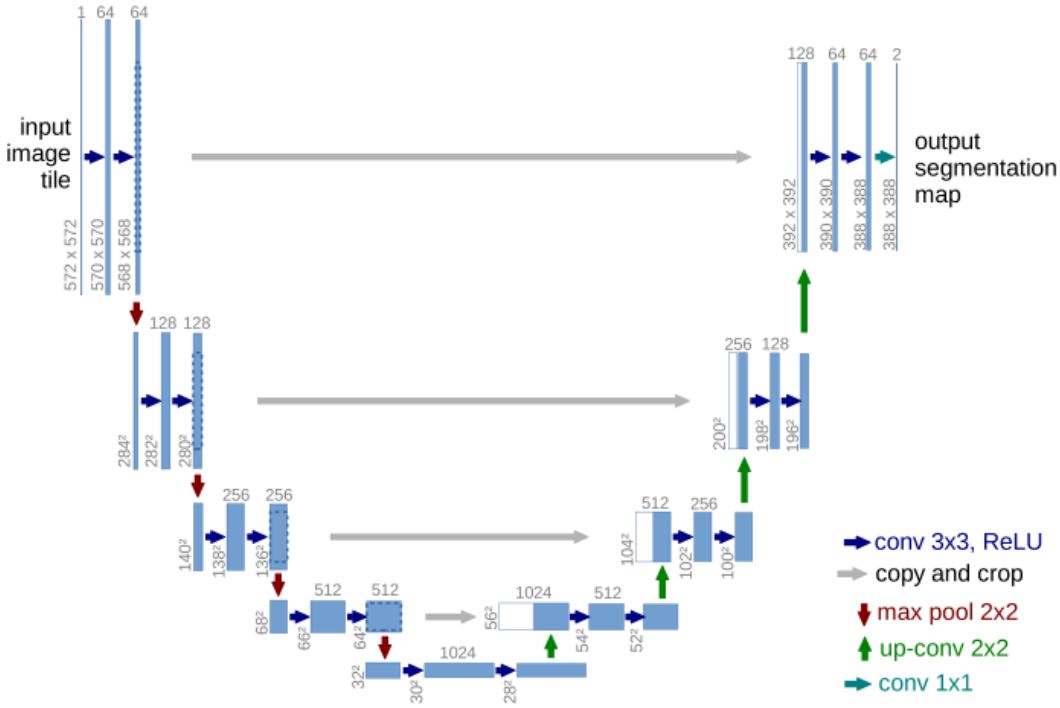
Entre a primeira e a segunda etapa, existe uma estrutura denominada gargalo (do inglês, *BootleNeck*), ela altera o formato na qual as características extraídas estão dispostas, e funcionam como um seletor de características entre os canais.

Em seguida, na segunda etapa as características extraídas até o gargalo são descomprimidas e expandidas por interpolação de pixels e "convoluções" (utilizando interpolação bi-cúbica na saída das camadas de "convolução") (setas verdes para cima no diagrama 2.17)). Em cada camada de decodificação, os mapas de características extraídos em cada camada da primeira etapa (indicadas pelas setas cinzas no diagrama 2.17 - *skip connection*) são recortados e concatenadas com as características que estão sendo interpoladas. Por fim, na última camada da rede, é realizada a convolução em 1 dimensão, que tem a finalidade de ajustar a dimensão dos canais para quantidade de classes existentes. Ao todo se somam 23 camadas convolucionais. Devido a semelhança gráfica do diagrama do modelo e o formato da letra U (Figura 2.17), este sistema é denominado U-*Net*.

[13] filtros ou *kernels* são matrizes nxn (em que n representa a dimensão dos filtros) que abrigam os pesos do modelo em cada camada convolutiva, estes filtros são capazes de capturar as principais características das imagens, como bordas, aspectos circulares, lineares, etc.

[14] São camadas de mapeamento por máximo, que são capazes de evidenciar os pixels com maior valor dentre a região sobreposta pelos filtros

Figura 2.17 – Diagrama da rede neural profunda U-*Net*.



Fonte: (RONNEBERGER; FISCHER; BROX, 2015).

Um detalhe importante informado no texto descrito por (RONNEBERGER; FISCHER; BROX, 2015), e que não aparece no diagrama, é a utilização da técnica de *DropOut* que consiste em atribuir uma chance de anular temporariamente alguns neurônios durante as iterações. Esta técnica foi utilizada nesta arquitetura como forma de realizar "*Data Augmentation*" de forma implícita, segundo os autores.

A arquitetura também pode ser utilizada em outras aplicações além da segmentação como compressão de imagens, redução de dimensionalidade, extração de características, geração de imagens, entre outros.

2.6.3.2 Outras Arquiteturas

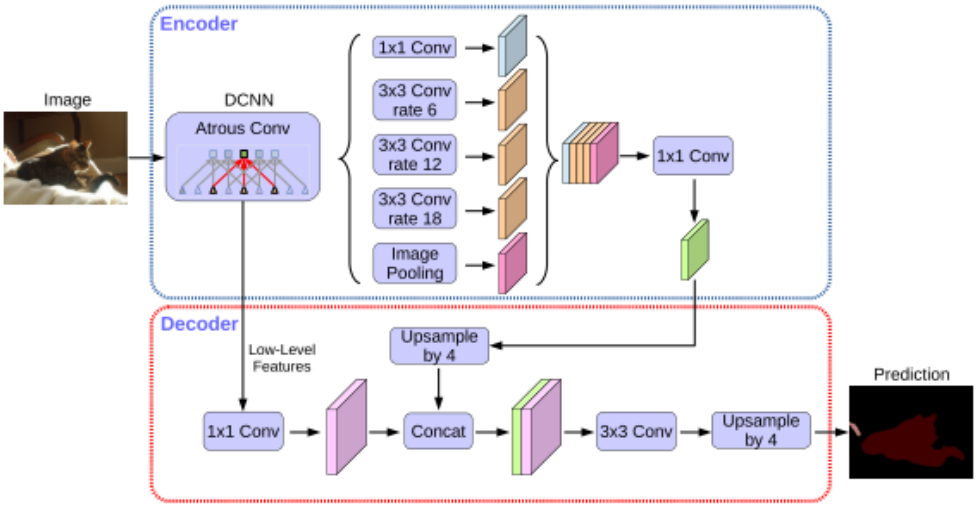
A partir da U-*Net*, diversos outros modelos surgiram como forma de aprimorar a segmentação de imagens, principalmente imagens utilizadas em estudos medicinais. Algumas das arquiteturas propostas posteriormente à U-*Net*, propuseram pequenas variações, como é o caso da arquitetura V-*Net*, que é uma arquitetura modificada para segmentação de volumes Milletari, Navab e Ahmadi (2016). Enquanto a U-*Net* convencional trabalha com imagens bidimensionais, a V-*Net* utiliza filtros em 3 dimensões, também gerando máscaras em 3 dimensões. Também é importante destacar a arquitetura U-Net++ Zhou *et al.* (2018),

que consiste no aninhamento de duas redes U-Net++, cujos resultados de segmentação ficam melhores e mais precisos com esta arquitetura.

2.6.3.3 *DeepLabV3+*

Dentre as diversas arquiteturas utilizadas para segmentação semântica, a *DeepLab* Chen *et al.* (2017) possui notável quantidade de elementos voltados especificamente para este propósito. Uma das principais características desta arquitetura é a convolução dilatada (do inglês, *Atrous Convolution*) (CHEN *et al.*, 2017) (definida no Apêndice A.2.2.1).

Figura 2.18 – Diagrama da rede neural profunda *DeepLabV3+*.



Fonte: (CHEN *et al.*, 2018).

Como pode ser observado na Figura 2.18, o *encoder* é composto por uma estrutura similar a ASPP (seção A.2.2.2), com a distinção de que esta arquitetura utiliza a operação de convolução separável, ficando de forma similar ao mostrado na seção (A.2.2.3). Os autores nomearam o formato como *atrous separable convolution.* É importante ressaltar que os mapas de características obtidos na etapa de codificação são redimensionados por interpolação bilinear e posteriormente concatenados nas respectivas camadas do decodificador de mesma resolução.

Existem outras redes mais elaboradas como é o caso da *TransUnet* Chen *et al.* (2021) que utiliza mecanismos de atenção da arquitetura *Vision-Transformer*, juntamente com um *backbone*[15] de uma CNN (ResNet).

[15] Esqueleto ou arquitetura base de um determinado modelo.

2.6.4 Medidas de Desempenho

Para a tarefa de avaliar se um algoritmo consegue classificar cada pixel de uma imagem, ou seja, se ele consegue definir uma região específica onde está um objeto, existem diversas medidas de pontuação, conforme mostra (JADON, 2020). Uma delas é a entropia cruzada binária, mostrada na seção anterior. Como ela avalia uma classificação binária, e, um modelo de segmentação semântica realiza a classificação binária pixel a pixel, ela pode ser utilizada como função de custo. Além da entropia cruzada, também pode ser usado o conceito da interseção sobre união.

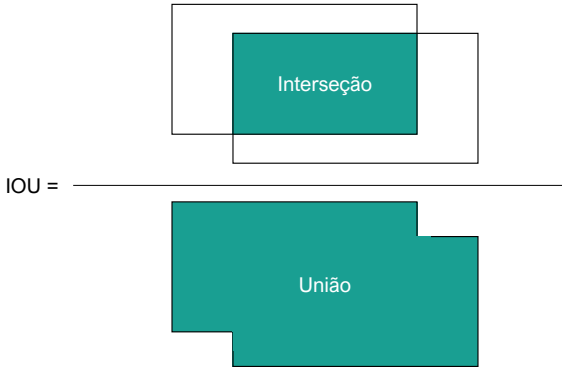
2.6.4.1 Interseção Sobre União

Uma forma bem conhecida de avaliar a saída de um modelo de segmentação, é utilizar algum índice de similaridade entre esta saída e a respectiva máscara anotada na base de dados. Dados dois conjuntos de pixels A e B, o índice de similaridade de *Jaccard* é definido conforme a equação (2.39) (ZHANG *et al.*, 2020a).

$$J_{Jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{2.39}$$

Na prática, se tratando de segmentação, para determinar a pontuação de *Jaccard* Jaccard (1912), o primeiro passo é calcular a interseção, multiplicando a máscara de saída (do modelo) pela máscara anotada (da base de dados). Para determinar a união, basta fazer o somatório das duas máscaras e subtrair a interseção que foi calculada no passo anterior. Finalmente a pontuação é calculada pela divisão de ambas, quanto maior for a pontuação mais similar a máscara de saída estará da máscara real.

Figura 2.19 – Interseção sobre União.



Fonte: Adaptado de (ZHANG *et al.*, 2020a).

Note que quando se trata de funções de custo, esta não é uma medida adequada para minimizar e então retro-propagar o erro de um modelo, uma vez que a tendência é sempre aumentar. Logo, para mensurar o erro, basta inverter a equação, ficando conforme a equação (2.40).

$$J_{Jaccard}(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B| + \epsilon}. \tag{2.40}$$

É uma medida comumente utilizada para aplicações de segmentação, também conhecida como distância *Jaccard*. É importante ressaltar ainda que, é necessário utilizar um termo ϵ no denominador para evitar a divisão por zero (KOSUB, 2019; PASZKE *et al.*, 2019).

2.6.4.2 *Dice*

Outra medida também muito utilizada para aplicações que envolvem segmentação é o coeficiente *Dice*. Também utilizando os conceitos de interseção e união, o coeficiente *Dice* pode ser dado pela equação (2.41).

$$J_{Dice}(A,B) = \frac{2 \cdot |A \cap B|}{|A \cup B|} = \frac{2 \cdot |A \cap B|}{|A| + |B| - |A \cap B|}. \tag{2.41}$$

De forma semelhante ao *Jaccard* a interseção é calculada pela multiplicação das máscaras e dobrada, em seguida soma-se as duas máscaras e subtrai-se a interseção, e por fim, é realizada a divisão para determinar o coeficiente *Dice*. Além disso a função de custo deve ser invertida para minimização, conforme a equação (2.42).

$$J_{Dice}(A,B) = 1 - \frac{2\cdot|A\cap B|}{|A\cup B|} = 1 - \frac{2\cdot|A\cap B|}{|A|+|B|-|A\cap B|+\epsilon}. \tag{2.42}$$

Note que assim como na equação de *jaccard*, existe um termo somando no denominador, simplesmente para que a equação não fique indefinida em 0, logo $0 < \epsilon << 1$. Alguns preferem contornar este problema somando 1 tanto no numerador quanto no denominador (JADON, 2020).

2.6.4.3 *Combo Loss*

A entropia cruzada binária (que será melhor explicada na sub-seção 2.8.10) avalia o desempenho do modelo de forma bem minuciosa no pixel da imagem. Ela pode ser diretamente utilizada como função de custo uma vez que ela compara duas distribuições. A proposta da *Combo Loss* por Taghanaki *et al.* (2019), consiste na combinação da BCE (do inglês, *Binary Cross Entropy*) com o coeficiente *Dice*, a fim de reduzir o desbalanceamento para casos em que se utiliza mais de duas classes. O que pode ocorrer nestes casos é de uma determinada classe possuir maior quantidade de pixels (ou área) da imagem, do que as outras. Neste contexto, a adequação das duas métricas fica conhecida como *Combo Loss*:

$$L = \alpha\left(-\frac{1}{N}\sum_{i=1}^{N}\beta(t_i - \ln p_i) + (1-\beta)[(1-t_i)\ln(1-p_i)]\right) - (1-\alpha)\sum_{i=1}^{K}\left(\frac{2\sum_{i=1}^{N}p_i t_i + S}{\sum_{i=1}^{N}p_i + \sum_{i=1}^{N}t_i + S}\right), \tag{2.43}$$

em que p_i corresponde a ativação ou predição do i-ésima conjunto de pixels (ou *voxel*), enquanto t_i corresponde ao valor real na base de dados; β é uma constante definida para penalizar falsos positivos[16](FP) e falsos negativos[17] (FN). Quando β for configurado menor que $0,5$, FP são penalizados mais que FN e vice versa; S também é uma constante para que a parte "Dice" da equação fique definida em 0; α simplesmente pondera qual das duas medidas terá maior influência sobre o custo que será retro-propagado.

[16] FP: Pixels erroneamente demarcados pelo modelo como sendo de uma classe.
[17] FN: Pixels de uma classe não demarcados pelo modelo.

2.7 Aumento Sintético de Amostras

Em muitas ocasiões a quantidade de amostras disponíveis em um banco de dados não supre minimamente a dimensionalidade dos dados, o que pode se tornar um problema grave, uma vez que o mapeamento estatístico que será realizado pelo modelo sobre os dados, se limita a um recorte relativamente pequeno da distribuição total (que não é factivelmente conhecida). Logo, o modelo pode não ser satisfatório quando testado em amostras fora do recorte.

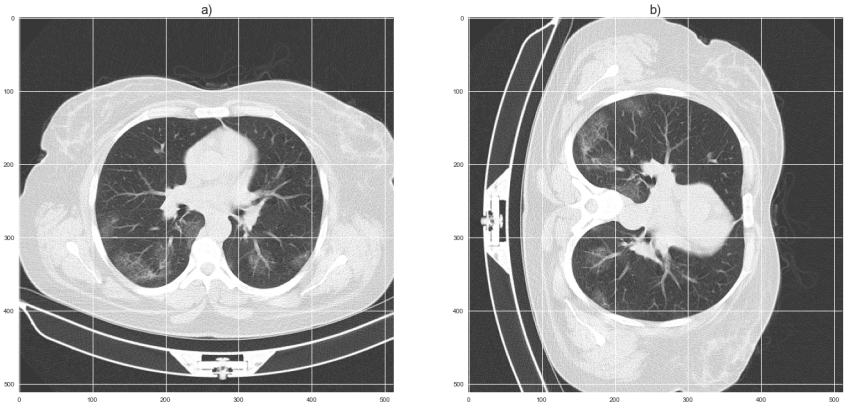
Para tentar aumentar a variabilidade interna do modelo em relação ao banco de dados, e buscar um maior nível de perspectivas para o modelo, podem ser aplicadas algumas transformações, denominadas *Data Augmentation*.

2.7.1 *Data Augmentation*

Algumas técnicas simples que demandam menor consumo computacional, podem ser observadas em (PLANCHE; ANDRES, 2019a) e (AYYADEVARA; REDDY, 2020). Algumas destas operações estão exemplificadas a seguir:

i) Rotação: A rotação consiste em girar uma imagem de forma suave em torno de seu eixo, ela também pode aumentar a variabilidade das amostras na base de dados, para uma imagem de tomografia, a transformação fica conforme a Figura 2.20.

Figura 2.20 – Exemplo da aplicação da rotação sobre uma imagem. Imagem Original a); Imagem Rotacionada 90º b).

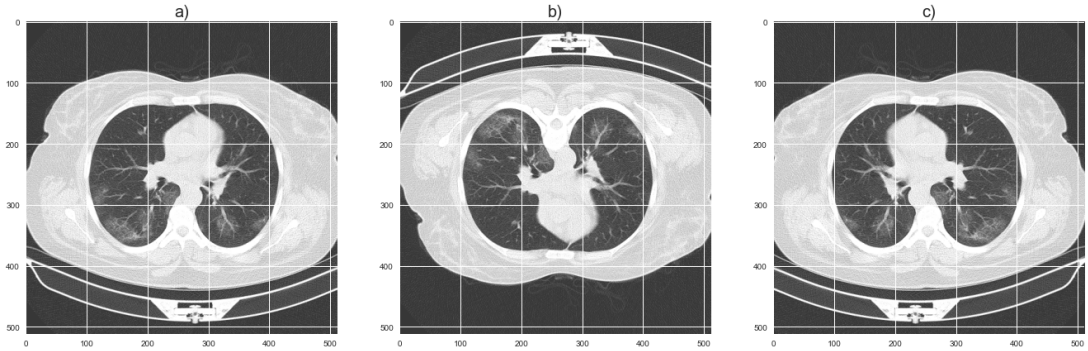


Fonte: Arquivo pessoal do Autor.

O exemplo acima faz uma rotação de 90° na imagem, porém na prática podem ocorrer rotações de variados ângulos, por isso para classificação, é possível ajustar o ângulo para uma determinada faixa;

ii) Espelhamento: Inverter uma imagem em relação ao eixo horizontal ou vertical também pode ser considerado uma técnica de aumento da variabilidade das amostras, basicamente os pixels do lado direito da imagem são trocados com os pixels do lado esquerdo, como pode ser observado na Figura 2.21.

Figura 2.21 – Exemplo da aplicação do espelhamento sobre uma imagem. a) Imagem original; b) Imagem Espelhada na vertical e c) Imagem Espelhada na horizontal.

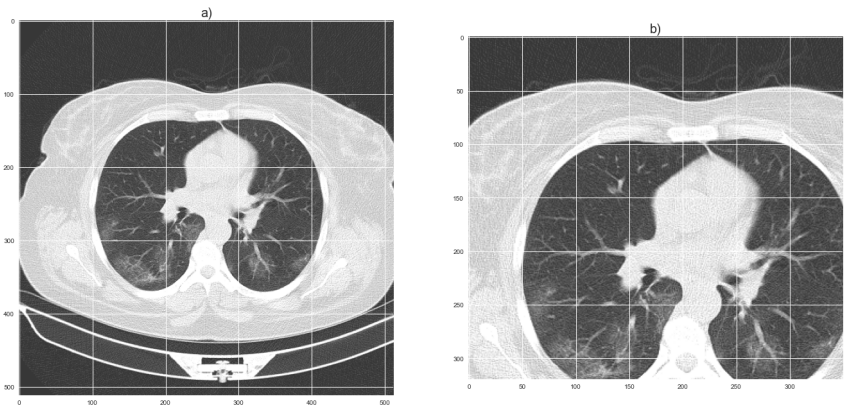


Fonte: Arquivo pessoal do Autor.

A imagem ao centro 2.21.b) se trata de um espelhamento na vertical, e a direita 2.21.c) se trata de um espelhamento na horizontal;

iii) Recortes: Selecionar aleatoriamente regiões de uma imagem e redimensioná-las ao tamanho original da base de dados pode ser útil também em certas aplicações. Observe o exemplo da Figura 2.22.

Figura 2.22 – Exemplo da aplicação do recorte sobre uma imagem. a) Imagem Original; b) Imagem Recortada.

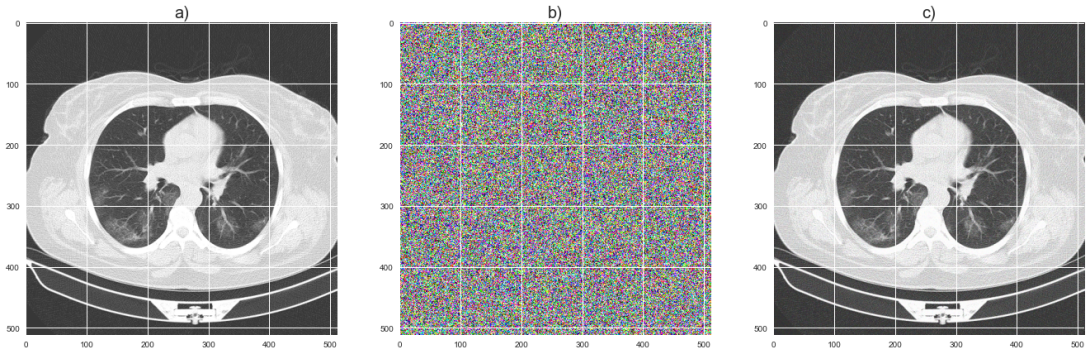


Fonte: Arquivo pessoal do Autor.

Este método pode ser muito útil para algumas aplicações, porém deve se ter muita cautela em sua utilização, pois a região removida pode comprometer o resultado.

iv) Adição de ruído: A adição de ruído pode parecer um método inusitado, pois ao adicionar ruído em um sistema, geralmente o faz reagir a um estímulo para o qual o sistema não foi projetado para trabalhar, porém quando se adiciona ruído em amostras de uma base de dados que será utilizada para treinamento de um modelo, o ruído confere um maior grau de variabilidade da base de dados, forçando o treinamento do modelo para um nível em que os efeitos do ruído na resposta, passam a ser, de certa forma "desconsiderados", como exemplo tem-se a Figura 2.23.

Figura 2.23 – Exemplo da aplicação do ruído Gaussiano sobre uma imagem. a) Imagem Original; b) Ruído; c) Imagem com ruído.

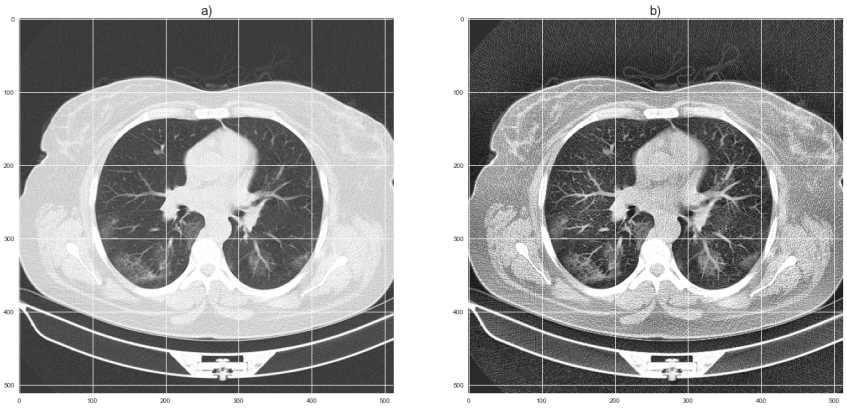


Fonte: Arquivo pessoal do Autor.

Para um especialista, uma imagem de baixa resolução ainda pode ser identificada como de um pulmão de um paciente infectado com SARS-COV-2, porém um modelo de aprendizado profundo pode perder sua capacidade generalização em tais condições. Por isso em alguns casos é interessante adicionar o ruído Gaussiano;

v) Aprimoramento de Contraste: Algumas imagens podem apresentar pouca ou baixa distinção entre suas regiões, uma forma de contornar é utilizando o algoritmo de equalização de histograma adaptável limitada ao contraste (do inglês, *Contrast Limited Adaptive Histogram Equalization*(CLAHE). Consiste em uma técnica capaz de alterar o contraste da imagem de entrada, ela pode ajudar no aumento da variabilidade do modelo em relação a diferentes tipos de intensidade das lesões. Como pode ser observado no exemplo da Figura 2.24.

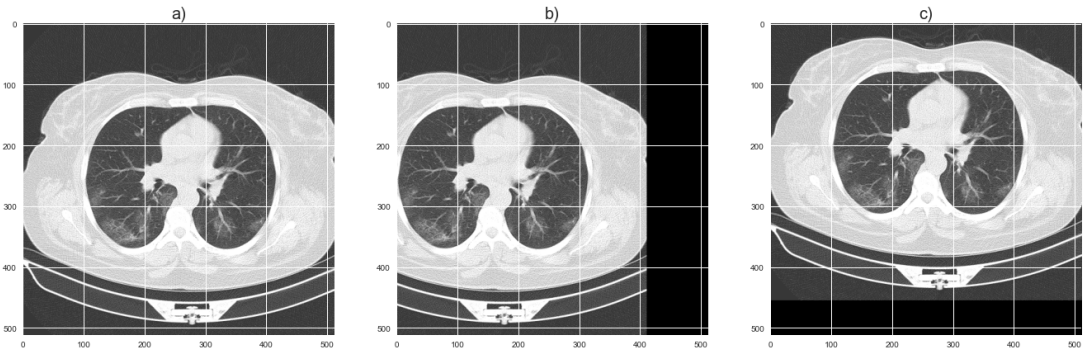
Figura 2.24 – Exemplo da aplicação do algoritmo CLAHE sobre uma imagem de tomografia. a) Imagem de paciente infectado; b) Imagem aprimorada pelo método CLAHE.



Fonte: Arquivo pessoal do Autor.

vi) Translação: A translação consiste em mudar a posição de pixels em um dos eixos ou em ambos, removendo os pixels próximos à borda, e adicionando zeros onde ficar vazio, como pode ser observado na Figura (2.25).

Figura 2.25 – Exemplo da aplicação da translação. a) Imagem original; b) Imagem transladada para esquerda; c) Imagem transladada para cima.

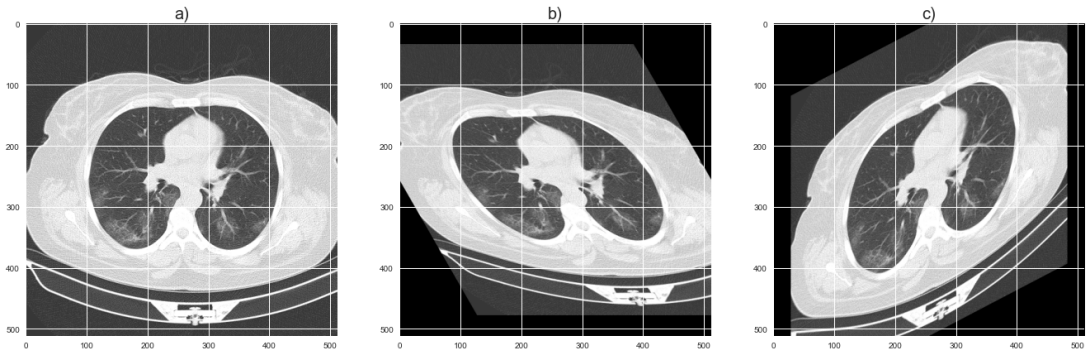


Fonte: Arquivo pessoal do Autor.

Para aplicações em que existem poucas amostras, esta transformação pode conferir uma ótima alternativa para aumentar a variabilidade do modelo;

vii) Cisalhamento: O cisalhamento consiste em alterar a posição dos pixels de acordo com um dos eixos (ou ambos). Visualmente o cisalhamento se comporta como uma rotação em torno de um dos eixos. Por exemplo, um cisalhamento horizontal deslocará os pixels da parte superior para a esquerda e os da parte inferior para a direita, veja o exemplo da Figura 2.26.

Figura 2.26 – Exemplo da aplicação do cisalhamento sobre uma imagem. a) Imagem Original; b) Cisalhamento horizontal; c) Cisalhamento vertical.



Fonte: Arquivo pessoal do Autor.

Como pode ser observado a imagem b) é um cisalhamento na horizontal, enquanto a imagem c) remete a um cisalhamento vertical (JUNG, 2020).

É importante destacar que todas estas técnicas se encaixam em um princípio básico dos modelos de visão computacional, que consiste em gerar um sistema com invariância na translação e no princípio da localidade, ou seja, modelos que conseguem distinguir objetos independentemente do local onde se encontram e da posição que estão neste local, bem como o contexto que eles se encontram (ZHANG *et al.*, 2020a).

Estas são apenas algumas das diversas técnicas utilizadas para aumento de dados, embora sejam eficazes em alguns casos. Quando o objetivo está relacionado identificar a orientação específica de um objeto, como placas de trânsito por exemplo (HALOI, 2015), deve-se atentar ao espelhar a imagem, pois uma placa de vire à direita pode se tornar vire à esquerda, logo seria necessário corrigir o alvo desta imagem.

2.8 Funções de Ativação ou Custo

No contexto de inteligência computacional, as funções de ativação introduzem uma não linearidade ao problema, e, em certos casos auxilia no treinamento e na predição do modelo. Elas são operadores diferenciáveis fundamentais para aplicações de aprendizado profundo (ZHANG *et al.*, 2020a). Nesta seção serão discutidas algumas das funções comumente utilizadas em aplicações de aprendizado de máquina e aprendizado profundo, comentando suas vantagens e desvantagens.

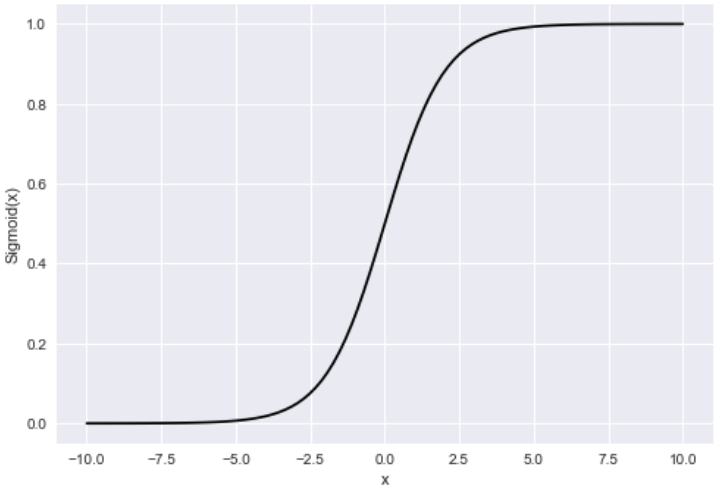
2.8.1 Sigmoid

A função *Sigmoid* é muito comum em aplicações de classificação para predizer a probabilidade da ocorrência de um evento, todas as entradas do neurônio serão convertidas a uma faixa entre 0 e 1, por isso alguns autores determinam sua saída como uma probabilidade, a equação pode ser escrita conforme 2.44 (BISHOP, 2016; MOOCARME; ABDOLAHNEJAD; BHAGWAT, 2020).

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.44}$$

em que x corresponde ao valor de entrada. O gráfico da função *Sigmoid* está representado pela Figura 2.27. E a retro-propagação fica a cargo de sua derivada, dada pela equação 2.45,

Figura 2.27 – Função *Sigmoid*.

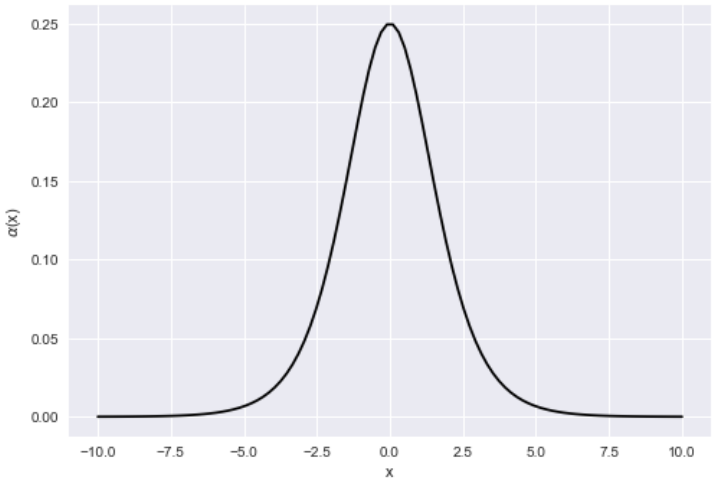


Fonte: Arquivo pessoal do Autor.

$$\frac{\partial\sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x)), \tag{2.45}$$

cujo gráfico fica conforme mostrado na Figura 2.28. O problema em utilizar a função de

Figura 2.28 – Derivada da função *Sigmoid*.



Fonte: Arquivo pessoal do Autor.

ativação *sigmoid* é que, pelo fato de achatar os valores entre 0 e 1, quando retro-propagado os erros, o gradiente poderá se esvaecer e os pesos irão ficar com valores ínfimos (“a rede aprende de forma mais lenta”), além disso, o treinamento é computacionalmente mais pesado em relação a outras funções de ativação como a ReLU por exemplo.

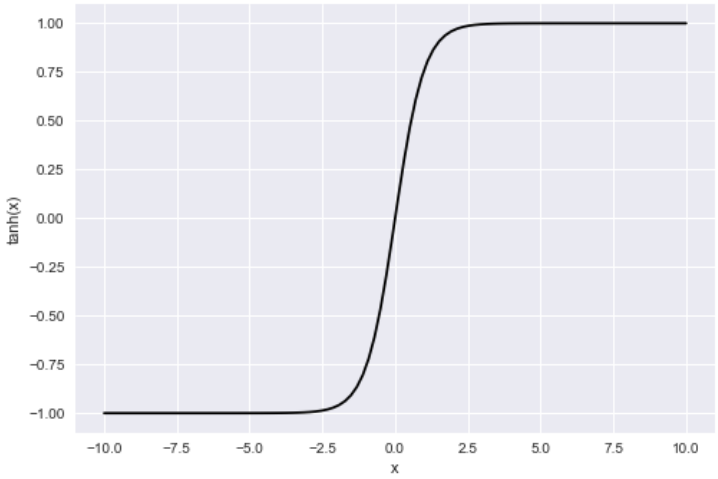
2.8.2 Tangente Hiperbólica

Assim como a função *sigmoid*, a função de ativação por tangente hiperbólica também “espreme” as entradas em uma faixa de valores, porém entre -1 e 1, ela pode ser escrita conforme a equação 2.46 (ZHANG *et al.*, 2020a).

$$tanh(x) = \frac{1 - exp(-2x)}{1 + exp(-2x)}, \tag{2.46}$$

que pode ser observada na Figura 2.29.

Figura 2.29 – Função Tangente Hiperbólica.



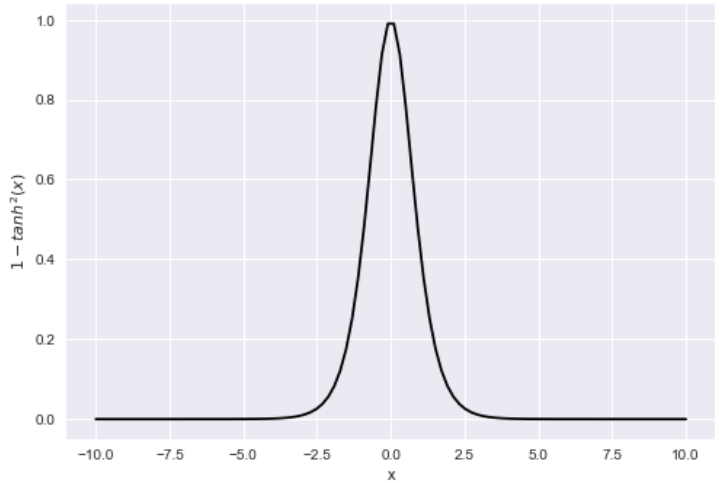
Fonte: Arquivo pessoal do Autor.

A derivada da tangente hiperbólica é dada pela equação 2.47.

$$\frac{\partial}{\partial x} tanh(x) = 1 - tanh^2(x). \tag{2.47}$$

Que pode ser observada na Figura 2.30.

Figura 2.30 – Derivada da função Tangente Hiperbólica.



Fonte: Arquivo pessoal do Autor.

Assim como a *sigmoid*, a $tanh$ sofre do problema do desaparecimento do gradiente.

2.8.3 ReLU

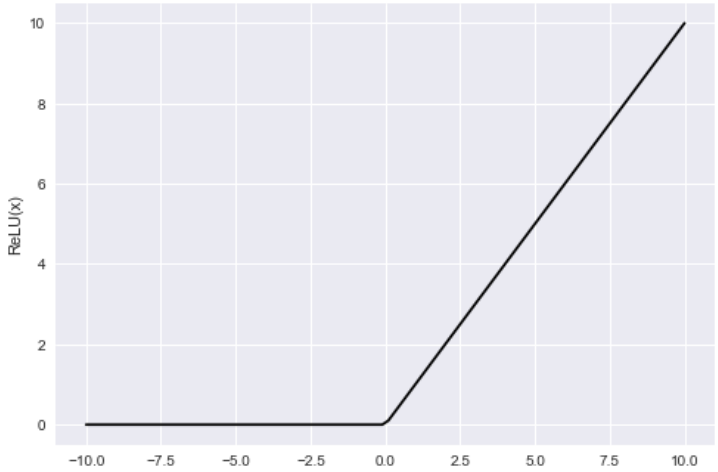
A função de ativação unidade linear retificada (do inglês, *Rectified Linear Unit* - ReLU), possui a capacidade de não deixar o gradiente reduzir o valor. Para valores negativos

na entrada de cada nó, a saída é anulada, e para valores acima de zero a saída é igual a entrada, conforme a equação 2.48 (ZHANG *et al.*, 2020a).

$$f(x) = \begin{cases} x, \text{ caso } x > 0 \\ 0, \text{ caso contrário} \end{cases} \tag{2.48}$$

Que é representada graficamente pelo gráfico da Figura 2.31.

Figura 2.31 – Função ReLU.



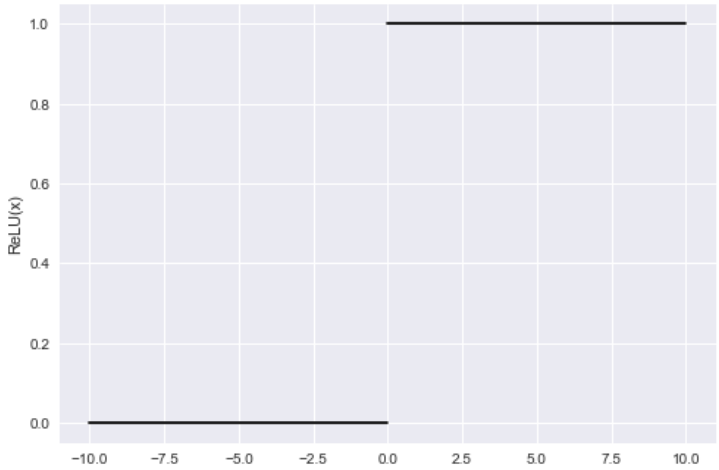
Fonte: Arquivo pessoal do Autor.

Ao contrário das funções anteriores, a ReLU não possui derivada definida em $x = 0$, pois os limites laterais existem mas não são iguais. Sua derivada fica conforme a equação 2.49.

$$f(x) = \begin{cases} 1, \text{ caso } x > 0 \\ 0, \text{ caso contrário} \end{cases} \tag{2.49}$$

Na prática, quando $x = 0$ a saída é truncada em um dos valores 0 ou 1, ficando conforme mostra a Figura 2.32.

Figura 2.32 – Derivada da função ReLU.



Fonte: Arquivo pessoal do Autor.

Alguns autores preferem utilizar o algoritmo com saída 0 pelo fato da entrada ser 0, e não interferir muito nos resultados. A ReLU não permite que o gradiente seja anulado ou se esvaeça durante o treinamento (para valores positivos), porém se algum parâmetro for inicializado com valor negativo, ou o erro retro-propagado de alguma derivada fizer um parâmetro se tornar negativo, a função de ativação terá valor nulo na etapa de propagação, e sua derivada também será nula na próxima iteração, podendo ocorrer a anulação de algum parâmetro (LU *et al.*, 2019).

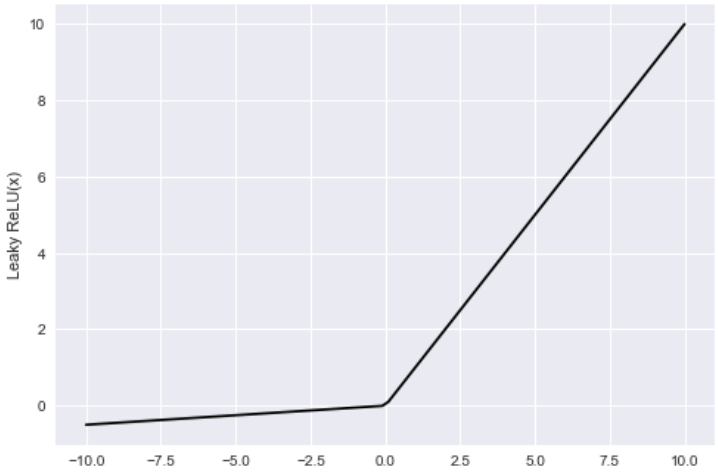
2.8.4 *Leaky ReLU*

Para solucionar o problema "morte de neurônios" presente na função de ativação ReLU, têm-se a função de ativação *Leaky ReLU*, que é similar à função anterior, porém com a parte negativa alterada, ao invés de assumir valor nulo para entradas negativas, o parâmetro "a" (muito próximo de zero) é multiplicado pelo valor negativo, ficando conforme a equação 2.50 (ZHANG *et al.*, 2020a).

$$f(x) = \begin{cases} x, \text{ caso } x > 0 \\ ax, \text{ caso contrário} \end{cases} \tag{2.50}$$

A função pode ser observada na Figura 2.33.

Figura 2.33 – Função *Leaky* ReLU.



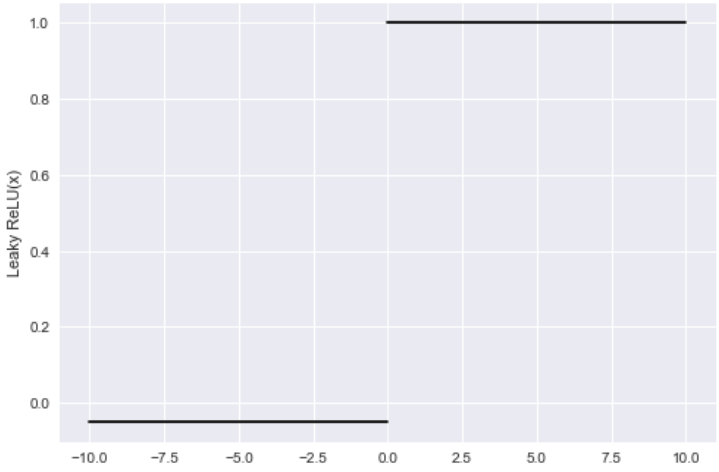
Fonte: Arquivo pessoal do Autor.

Sua derivada pode ser definida conforme a equação 2.51.

$$f(x) = \begin{cases} 1, \text{ caso } x > 0 \\ a, \text{ caso contrário} \end{cases} \tag{2.51}$$

O gráfico pode ser observado na Figura 2.34.

Figura 2.34 – Derivada da função *Leaky* ReLU.



Fonte: Arquivo pessoal do Autor.

Desta maneira as saídas dos nós não são anuladas, causando o problema conhecido como "morte de neurônios", ocorre principalmente quando a rede "aprende"parâmetros de bias com valores negativos grandes o suficiente para alterar a polaridade de toda entrada, e todos os valores acabam se tornando negativos, se anulando ao passar pela ReLU. Ainda na mesma tentativa de solucionar os problemas da função ReLU, têm-se a PReLU (*Para-*

metric Rectified Linear Unity, em que "a" é um termo que pode ser atualizado ao longo do aprendizado do modelo (HE *et al.*, 2015).

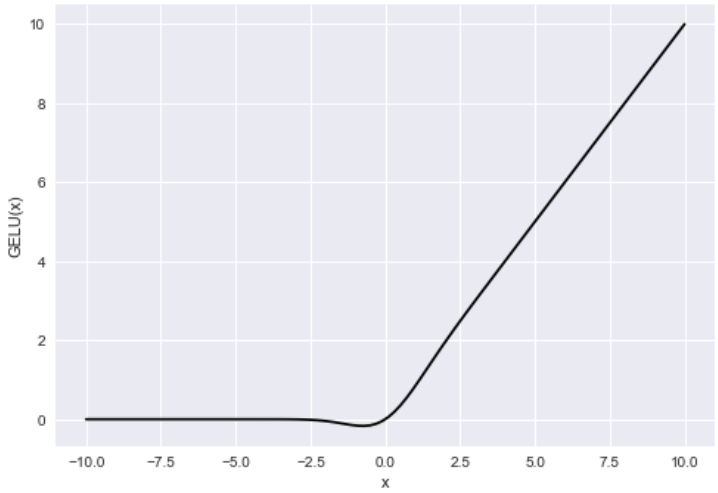
2.8.5 GELU

Esta função foi definida proposta por Hendrycks e Gimpel (2016) com o objetivo de substituir a função RELU em aplicações em que os dados ou pesos podem apresentar valores negativos, como a RELU pode levar estes valores para zero e anular os parâmetros, a GELU surge como uma ótima alternativa, ela é definida pela equação 2.52.

$$gelu(x) = 0,5x\left(1+\tanh\left[\sqrt{2/\pi}(x+0,044715x^3)\right]\right). \tag{2.52}$$

O gráfico pode ser observado na Figura 2.35.

Figura 2.35 – Função GELU.



Fonte: Arquivo pessoal do Autor.

Os autores realizaram um estudo comparativo, em que a GELU apresentou melhores resultados nos bancos de dados utilizados.

2.8.6 SELU

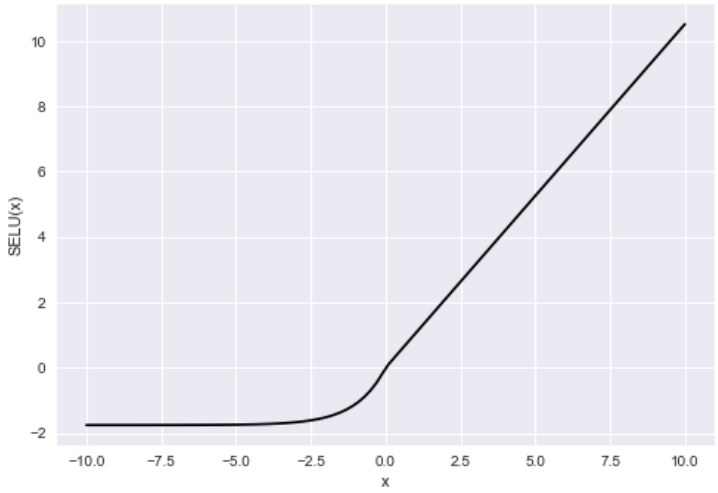
A Unidade Linear Exponencial Escalonada (do inglês, *Scaled Exponential Linear Unit*), foi proposta inicialmente por Klambauer *et al.* (2017), para atender a uma rede neural auto-normalizante. Isto significa que a rede é capaz de manter os parâmetros dentro de uma faixa fixa. O que é bom tanto para o problema do esvaecimento do gradiente quanto

para o impedimento da explosão para infinito do gradiente. A equação desta função é dada pela equação 2.53.

$$selu(x) = \lambda \begin{cases} x, \text{ caso } x > 0 \\ \alpha e^x - \alpha, \text{ caso } x \leq 0 \end{cases} \tag{2.53}$$

Cujo gráfico pode ser observado na Figura 2.36.

Figura 2.36 – Função SELU.



Fonte: Arquivo pessoal do Autor.

Em que λ e α são constantes obtidas pelos autores, e equivalem a 1.0507 e $1,6733$ respectivamente. É importante ressaltar que para esta função funcionar, ou seja, para que a normalização ocorra internamente no treinamento, é necessário inicializar os pesos com distribuição normal centralizada em 0 com desvio padrão $\sigma = \sqrt{1/\text{fan\_in}}$ em que fan_in[18] é o número de unidades de entrada no tensor de pesos. Além disso se for desejável utilizar a técnica *DropOut* é necessário utilizar a técnica *Alpha-DropOut*, a fim de garantir que a normalização ocorra.

2.8.7 *Softmax*

Assim como a função *sigmoid*, a *softmax* pode ser interpretada como uma probabilidade, forçando as saídas da rede neural a ficarem entre os valores zero e um. A equação pode ser definida de acordo com a equação 2.54 (BISHOP, 2016).

$$y_k(x,w) = \frac{e^{a_k(x,w)}}{\sum_j e^{a_j(x,w)}} \tag{2.54}$$
$$\text{Que satisfaz: } 0 \leq y_k \leq 1 \text{ e } \sum_k y_k = 1$$

[18] Esta normalização também é conhecida como normalização LeCun (LECUN *et al.*, 1998).

Em que a ($a_k(x) = w_k^T x + w_{k0}$) representa as funções de ativação que mapeiam as entradas (x) i.i.d[19] pelos pesos w de um modelo. Apesar de ser muito utilizada para problemas de classificação multi-classes, a função *softmax* também é suscetível ao efeito do decaimento do gradiente (BISHOP, 2016; ZHANG *et al.*, 2020a).

2.8.8 *Log-Softmax*

Procurando amenizar o efeito do decaimento, matemáticos surgiram com a versão logarítmica da função *softmax*, que pode ser escrita conforme a equação 2.55.

$$y_k(x,w) = \log\left(\frac{e^{a_k(x,w)}}{\sum_j e^{a_j(x,w)}}\right) \tag{2.55}$$

Apesar de evitar o decaimento do gradiente, a função logarítmica admite valores negativos, que não podem ser interpretados como probabilidade sem reajuste da escala de valores. Na comparação realizada em Brébisson e Vincent (2015) esta função conseguiu resultados razoáveis para bancos de dados de maior dimensionalidade. Bem como outras variações como a *Taylor Softmax* e *Log Spherical Softmax*.

2.8.9 *Cross Entropy*

A entropia cruzada é uma medida de desempenho de um modelo, ela consiste na comparação entre duas distribuições de probabilidade. Considerando o somente uma amostra cujo rótulo y está associado a uma classe k, e a saída do modelo informa que a amostra está associada a classe $\hat{y}$, logo a equação da função de custo para entropia cruzada pode ser dada conforme a equação 2.56 (PONTI; COSTA, 2018).

$$l_{CE}(y,\hat{y}) = -\sum_j y_j \cdot \log(\hat{y}), \tag{2.56}$$

De forma mais geral, para duas distribuições de probabilidade (ao invés de uma amostra), a função de custo fica conforme a equação 2.57.

$$L_{CE} = -\sum_{i=1}^{N}\sum_{k=1}^{K} P(y_i = k) \cdot \log(Q(y_i = k)), \tag{2.57}$$

[19] Independentes e identicamente distribuídas.

em que $P(y_i = k)$ é a distribuição de probabilidade da i-ésima amostra sobre a classe real, e a distribuição de probabilidade da i-ésima observação ser da classe k é $Q(y_i = k)$.

2.8.10 *Binary Cross Entropy*

Também conhecida como probabilidade logarítmica negativa (do inglês, *negative log likelihood*), esta função de custo consiste em um caso a parte da entropia cruzada, que é para casos em que só se tenha duas classes. Ela pode ser definida conforme 2.58 (JADON, 2020).

$$L_{BCE}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - y)), \tag{2.58}$$

assim como a entropia cruzada, y é o rótulo e $\hat{y}$ é o valor estimado pelo modelo.

2.8.11 *Softmax Cross Entropy*

Esta função combina as funções entropia cruzada com a *softmax*, pode ser definida conforme a equação 2.59 (WANG *et al.*, 2022).

$$L_{SCE}(w, b) = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{w_{y_i}^T x_i + b_{y_i}}}{\sum_k e^{w_k^T x_i + b_k}}, \tag{2.59}$$

em que w representa os pesos; b representa os *biases* e o T sobrescrito indica que o tensor deve ser transposto. É uma função muito utilizada por desenvolvedores que utilizam as bibliotecas *Pytorch* e *Tensorflow* (ABADI *et al.*, 2015; PASZKE *et al.*, 2019).

2.9 Métodos de Regularização

Na maioria dos casos em que se busca tratar problemas reais com algoritmos de inteligência computacional, o analista possui uma amostragem do problema, ou seja, um subconjunto do problema que foi amostrado, e que não necessariamente representa toda a distribuição estatística do problema. Neste contexto, algumas técnicas foram propostas com o objetivo de regular o modelo, a fim de torná-lo o mais "plástico" possível perante aos dados reais, ou seja, que ao realizar a inferência sobre dados reais (não conhecidos pelo modelo), ele tenha uma capacidade de generalização. Para este propósito existem alguns algoritmos de regularização por penalização, dois dos quais serão contemplados no apêndice A.4.

No apêndice também é abordada uma outra técnica baseada na normalização dos pesos e na desativação de neurônios com uma certa probabilidade. De forma geral estas técnicas buscam ajustar o modelo durante o treinamento para aumentar a capacidade de generalização e reduzir o sobre-ajuste do mesmo (GREENSHTEIN; RITOV, 2004; HASTIE, 2020; MOOCARME; ABDOLAHNEJAD; BHAGWAT, 2020; BISHOP, 2016; ELGENDY, 2020).

2.10 Avaliação de Modelos

Avaliar o desempenho de um modelo durante e após o treinamento é um procedimento essencial, pois possibilita parametrizar a taxa erro/acerto do modelo sobre o conjunto de dado, além de permitir o monitoramento do comportamento do modelo durante o treinamento. Estas métricas e critérios são diferentes entre tarefas de regressão e classificação, serão discutidos conforme proposto em Johnston e Mathur (2019).

2.10.1 Regressão

As métricas de regressão se preocupam em mensurar o erro entre o valor real que uma determinada amostra possui e o valor aproximado. Isto ocorre pois em alguns casos não é possível categorizar os valores de saída, como por exemplo, em series temporais, nas quais é necessário realizar a predição do valor que possivelmente será encontrado, dadas as características/atributos das amostras.

2.10.2 Classificação

Em sistemas de classificação, a saída indicará um valor simbólico, que associará a amostra com sua respectiva classe, para casos de aprendizado supervisionado. Para casos de aprendizado não supervisionado, os algoritmos simplesmente agrupam as amostras a partir das distâncias entre as amostras no hiper espaço de características, e cada amostra possui uma pertinência para com as classes.

2.10.2.1 Acurácia

O método mais simples e básico de avaliar o desempenho de um modelo durante e depois de um processo de treinamento, é pela acurácia, que pode ser determinada pela equação 2.60.

$$\text{acurácia} = \frac{\text{quantidade de acertos}}{\text{total de previsões}} \tag{2.60}$$

Apesar deste método ser muito utilizado ele não é capaz de capturar devidamente o desempenho de um modelo em um problema de base de dados desbalanceada. Por exemplo, se uma base de dados possuir 1000 amostras, sendo que 990 são da classe A e somente 10 são da classe B, então se o modelo “aprender” a classificar todas as amostras como classe A,

ele terá uma acurácia de 99%. Por isso outras métricas são utilizadas para avaliar melhor o desempenho do modelo em relação à cada classe e suas respectivas taxas de acerto.

2.10.2.2 Matriz de confusão

A matriz de confusão tem por objetivo mostrar especificamente a quantidade de erros e acertos que foram obtidos pela saída de um modelo. Caso o modelo acerte corretamente a ocorrência (VP) e não ocorrência (VN) dos eventos (SEBASTIANI, 2002). A matriz de confusão para o caso binário pode ser observado na tabela 2.1.

Tabela 2.1 – Matriz de Confusão.

		Valor Real	
		$Y = 0$	$Y = 1$
Valor de Saída	$\hat{Y} = 0$	VN (Verdadeiro Negativo)	FN (Falso Negativo)
	$\hat{Y} = 1$	FP (Falso Positivo)	VP (Verdadeiro Positivo)

Fonte: Arquivo Pessoal do Autor.

Na qual os termos VN e VP, correspondem a quantidade de amostras que tiveram suas classes acertadas, e, os termos FP e FN, correspondem a quantidade de amostras que foram classificadas de forma incorreta. Esta é uma tabela de confusão binária (comparação entre duas classes), porém poderiam ser comparadas diversas classes, por isso o nome generalizado matriz de confusão.

A partir da tabela, pode-se definir as seguintes medidas (IZBICKI; SANTOS, 2020):

- Sejam duas Classes A (doentes) e B (não doentes), com os alvos 1 (doentes) e 0 (não doentes), respectivamente. Estatisticamente, a classe positiva faz menção à ocorrência do evento, e, quando o modelo prediz esta ocorrência de forma correta, esta predição é denominada verdadeiro positivo (VP). Assim, podemos definir as seguintes medidas:

1) **Sensibilidade/*Recall***: A sensibilidade indica a taxa de acerto do modelo sobre a classe 1 (doentes) entre os doentes somente, ou seja, quantos dos pacientes doentes (rótulo real), foram corretamente identificados (intra-classe), pode ser definida pela equação 2.61.

$$S = \frac{VP}{VP + FN} \tag{2.61}$$

2) **Especificidade**: A especificidade indica a taxa de acerto do modelo sobre a classe 0 (amostras de não doentes) entre os não doentes somente, ou seja, quantos dos

pacientes não doentes (rótulo real) foram corretamente identificados, ou seja, indica a quantidade de amostras com alvo 0 que o modelo conseguiu acertar (intra-classe), ela pode ser definida conforme a equação 2.62

$$E = \frac{VN}{VN + FP} \tag{2.62}$$

3) **Valor Preditivo Positivo/*Precision***: O valor preditivo positivo indica quantos acertos o modelo obteve na classe 1 (doentes), entre os dados com ou sem a doença foram corretamente classificados como doentes, ou seja, ao contrário da sensibilidade, ele indica quantos acertos inter-classe houveram, pode ser definida conforme a equação 2.63.

$$VPP = \frac{VP}{VP + FP} \tag{2.63}$$

4) **Valor Preditivo Negativo**: O valor preditivo negativo indica quantos acertos o modelo obteve na classe 0 (não doentes) entre os dados com ou sem a doença foram corretamente classificados como não doentes, ou seja, ao contrário da especificidade, ele indica quantos acertos inter-classe houveram, conforme a equação 2.64

$$VPN = \frac{VN}{VN + FN} \tag{2.64}$$

5) **Pontuação F1**: A pontuação F1 (do inglês, *F1-Score* é uma medida de precisão baseada na sensibilidade e no valor preditivo positivo, que consegue avaliar o desempenho do modelo em predizer o valor real da da amostra de acordo com a classe de origem, conforme mostra a equação 2.65.

$$F1 = \frac{2}{1/S + 1/VPP} = \frac{2 * S * VPP}{S + VPP} \tag{2.65}$$

F_1 é o único dos parâmetros que está ligado de forma indireta à matriz de confusão, como forma de associar o valor preditivo positivo VPP, com o valor de sensibilidade S. Assim, o F_1 é um invólucro (do inglês, *Wrappers*) sobre dois parâmetros, também conhecido como média harmônica, e seu valor máximo é 1, quando não há erros no modelo. O problema da pontuação F_1 está no fato de que a métrica dá mesma

importância para S e VPP, por este motivo existem suas variações com F_β e F_α (HAND; CHRISTEN, 2018).

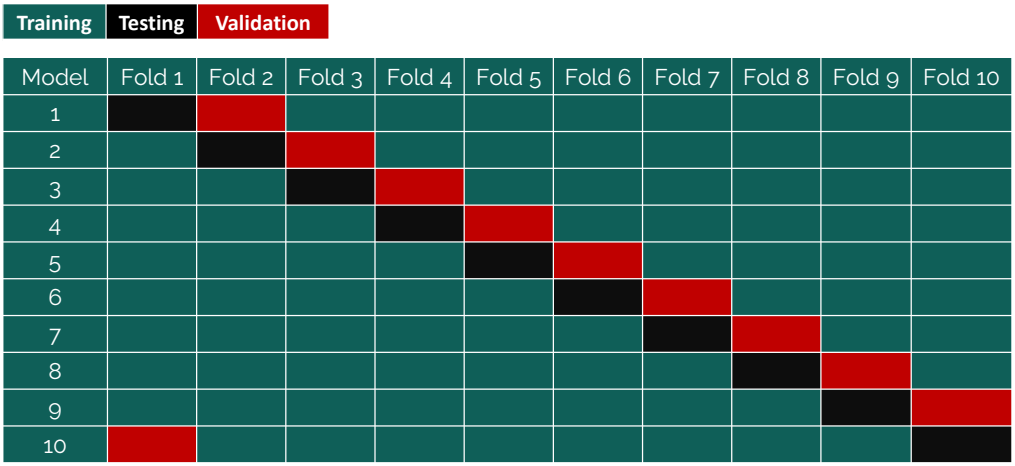
2.11 Validação de Modelos

A ideia geral por trás da validação é avaliar se o modelo que foi treinado possui minimamente a capacidade de generalização necessária para entrar em operação. Para tal, é necessário separar um subconjunto da base de dados para avaliar o modelo repetidas vezes e estimar o seu desempenho por alguma métrica comparável ao desempenho em treinamento. Assim é possível estimar como o modelo se comportará fora da amostragem que em que foi treinado.

2.11.1 Validação Cruzada

A validação cruzada consiste em um método de avaliação de modelos, através dele é possível ter uma ideia de como será o desempenho do modelo na prática. O método consiste no treinamento de K modelos em K-2 subconjuntos da base de dados, um subconjunto restante é utilizado para teste e outro para validação, ou, caso não se utilize um conjunto para validação, faz-se o treinamento em K-1 subconjuntos, e 1 subconjunto é utilizado para teste (RASCHKA, 2018). A Figura 2.37 exemplifica esta divisão. É importante ressaltar no diagrama, que a cada modelo que é treinado, as pastas são comutadas, ou seja, o primeiro modelo terá pastas de teste e validação diferentes do segundo modelo, podendo repetir pastas somente de treino de um modelo para o outro. Esta característica permite que todas as amostras sejam utilizadas tanto para treino quanto teste e validação.

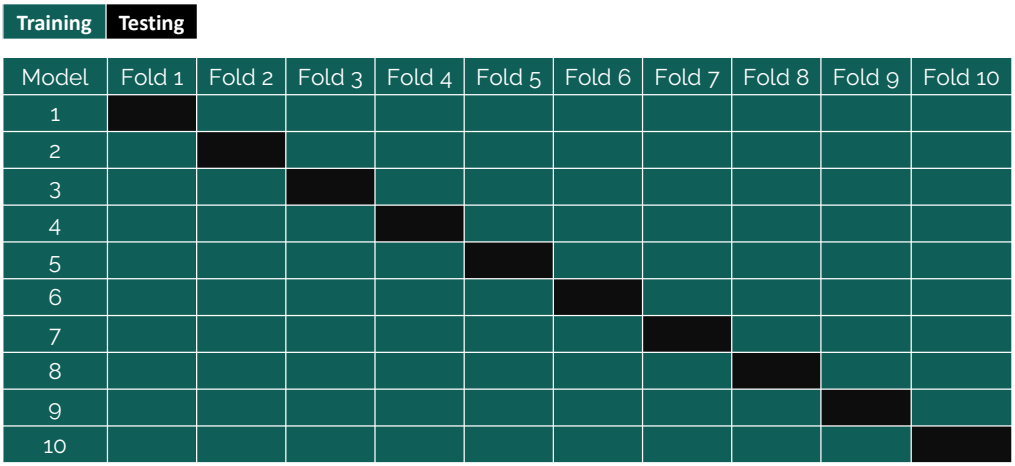
Figura 2.37 – Esquema de divisão por K-*Fold* (Treino/Teste/Validação).



Fonte: Arquivo pessoal do Autor.

Também existe um segundo cenário em que o conjunto de validação não aparece. Em alguns casos, quando a quantidade de dados é menor e/ou não exista a necessidade de monitorar o desempenho do modelo durante a etapa de treinamento, não é utilizado a separação da amostragem em um conjunto de validação para treinamento. Neste caso a configuração é mostrada na Figura 2.38, Assim como no primeiro caso, o conjunto de teste é utilizado somente uma única vez em cada modelo, e só poderá ser utilizado para treino nos modelos subsequentes.

Figura 2.38 – Esquema de divisão por K-*Fold* (Treino/Teste).



Fonte: Arquivo pessoal do Autor.

Na validação por *K* pastas (do inglês, *K-fold*), existem alguns casos particulares, que recebem nomenclaturas especificas, alguns dos quais são:

- Caso a base de dados seja pequena o suficiente, pode-se escolher o valor de K igual à quantidade de amostras na base de dados. O método é denominado validação cruzada deixando um de fora (do inglês, *Leave-One-Out Cross-Validation*–LOOCV).
- Caso a base seja desbalanceada, a divisão pode fazer com que não se tenha em todas as K pastas, amostras referentes a todas as classes possíveis na base de dados, portanto pode se fazer uma divisão estratificada, na qual todas as pastas recebem amostras referentes as respectivas classes na mesma proporção que as outras.

2.12 Trabalhos Correlatos

Neste capítulo são discutidos os principais trabalhos que deram base para elaboração desta proposta.

Em muitas aplicações de visão computacional, quando se trata de problemas complexos e/ou delicados como por exemplo a identificação de doenças em exames médicos, a etapa de pré-processamento pode ser uma etapa fundamental, e qualquer detalhe que colabore para um resultado mais eficiente deve ser empregado. No caso da COVID-19, o trabalho de Rahimzadeh, Attar e Sakhaei (2020) propõe um algoritmo para identificar a região aproximada em que estão os pulmões do paciente dentro da imagem, e dentro desta região são contabilizados os pixels que estão dentro da faixa de valores que o pulmão possui dentro do conjunto de imagens disponível. Todo este processo contribui para remover do conjunto, imagens que não apresentam uma quantidade mínima de pixels na região, ou seja, que o pulmão do paciente não aparece significativamente. Logo, estas imagens são removidas da análise, reduzindo a quantidade de amostras no banco de dados, e otimizando o tempo gasto para treinamento.

Pensando em regiões de interesse, a utilização da segmentação poderia contribuir significativamente para o pré-processamento, uma vez que ela possibilita a eliminação de regiões da imagem não pertinentes ao problema, como foi feito em Zhang *et al.* (2020b), proposta na qual foram extraídas as regiões de interesse, e, juntamente com outras informações sobre os pacientes, foram treinados modelos para diagnosticar pacientes com COVID-19.

Uma grande iniciativa proposta em 2020, foi a COVID NET, proposta por Gunraj, Wang e Wong (2020). Ela consiste em uma ampla gama de ferramentas e bancos de dados disponibilizados publicamente para auxiliar no desenvolvimento de novas ferramentas e pesquisas acerca da doença COVID-19. Dentre os materiais disponibilizados, foram reunidas aproximadamente 200 mil imagens de pacientes de diferentes países com três diferentes diagnósticos (COVID-19, Normal e Pneumonia). Entre as ferramentas propostas, estão os modelos especificamente modificados para classificar imagens de tomografia. Estes modelos treinados consistem de modelos baseados em redes "convolucionais", e utilizando a classificação com as 3 classes disponíveis, foram obtidos bons desempenhos para classificação da doença. Além disso foi utilizado o método de explicabilidade[20] GRAD-

[20] Estes métodos tendem a demonstrar quais as regiões da imagem foram capturadas pelo modelo e que mais contribuíram para o resultado final.

CAM (do inglês, *Gradient-Weighted Class Activation Mapping*) Selvaraju *et al.* (2016) para identificar as regiões capturadas pelo modelo para classificar as imagens, e foram obtidas regiões consistentes com a opinião de especialistas na doença.

No âmbito de aplicações médicas e tratamento de pacientes por análises assistidas por modelos de aprendizado profundo, busca-se ao máximo demonstrar resultados consistentes. Em Mondal *et al.* (2021) por exemplo, foram utilizados modelos baseados em mecanismos de atenção, juntamente com a proposta de explicabilidade proposta por Chefer, Gur e Wolf (2021), mostrando as regiões na imagem mais relevantes para o modelo com respeito a classificação gerada.

O procedimento descrito no capítulo seguinte, engloba todos estes trabalhos descritos, levando em conta a segmentação no pré-processamento, modelo *Vision Transformer* bem como um método para explicabilidade, finalizando com a segmentação de lesões. A presente proposta exige uma base de dados extensa e descritiva de exames de tomografia bem como seus respectivos rótulos de diferentes tipos de lesões, sendo necessário um nível de detalhes e anotações que somente um profissional especialista conseguiria realizar.

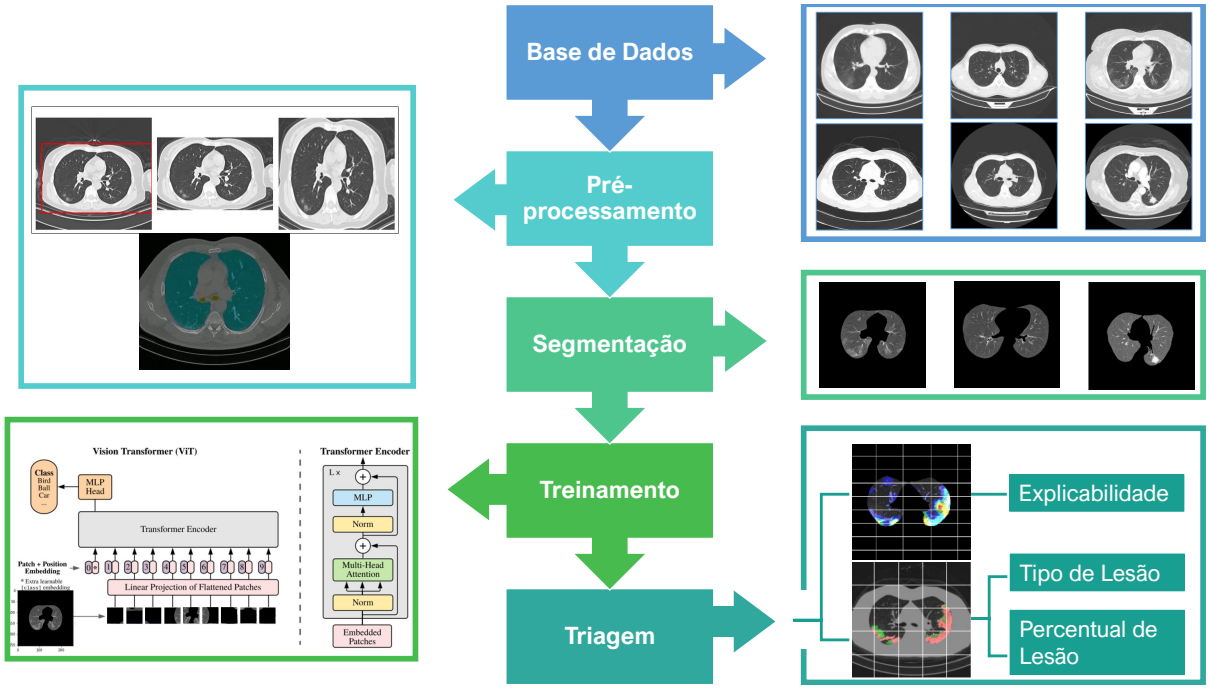
3 MÉTODO UTILIZADO

Neste capítulo serão detalhadas as técnicas e algoritmos utilizados no desenvolver deste trabalho.

3.1 Etapas

Primeiramente, têm-se o fluxo de processamento das imagens retiradas de exames de tomografia, proposta para o desenvolvimento deste trabalho. Elas percorrem desde o pré-processamento até a tomada de decisão do diagnóstico, como mostrado no diagrama da Figura 3.1.

Figura 3.1 – Diagrama Generalizado do método proposto.



Fonte: Arquivo pessoal do Autor.

As etapas apresentadas na Figura 3.1 possuem algumas peculiaridades que serão detalhadas nas seções subsequentes.

3.1.1 Primeira Etapa: Base de dados

O primeiro bloco da Figura 3.1 corresponde à aquisição e manipulação inicial da base de dados. O conjunto de imagens utilizadas neste projeto foi obtida pelo trabalho de Gunraj, Wang e Wong (2020) (maiores informações são discutidas na seção 3.1.1.1). Note que o bloco contém 6 imagens, porém apenas uma imagem é analisada por vez, e somente

após a análise de várias imagens é possível dar um parecer sobre o paciente. Nesta etapa, o único metadado utilizado foi a condição de avaliação do especialista (item 7 da seção 3.1.1.1), a fim de realizar a seleção das imagens dos pacientes que efetivamente seriam utilizados para a construção do sistema, apenas os pacientes dos quais as imagens foram selecionadas por especialistas passaram para próxima etapa.

3.1.1.1 COVIDx-CT

O principal conjunto de imagens de exames de tomografia utilizado foi reunido pela iniciativa COVIDx-CT (GUNRAJ *et al.*, 2021), disponibilizada publicamente e hospedada na plataforma *KAGGLE*[1]. A iniciativa consiste em uma coleção de bases públicas organizadas em uma somente, para que pesquisadores do mundo todo possam contribuir para o avanço na pesquisa, e na construção de sistemas capazes de dar suporte aos profissionais da linha de frente na luta contra a doença.

A primeira versão foi publicada em março de 2020 (com cerca de 150 mil imagens), e até então, a base de dados COVIDx-CT possui uma grande quantidade de imagens reunidas, contando com aproximadamente 194 mil imagens. Para cada paciente, além das imagens, existem três tipos de diagnósticos, que são COVID-19, Pneumonia e Normal. São mais de 3 mil pacientes de pelo menos 15 países diferentes, conferindo uma variedade de imagens e proporções altamente diversificada.

Além das informações de origem, forma de identificação da doença, também é informado uma anotação da região em que o corpo do paciente se encontra, consiste em uma seção retangular da imagem onde o corpo do paciente se encontra, aproximadamente, o que pode ser muito útil para etapas de pré-processamento. Além das imagens de cada paciente, a base de dados reúne as seguintes informações sobre cada paciente:

1) Fonte: Se refere à organização original que publicou os dados;

2) País: Se trata do país de origem do paciente;

3) Gênero: Se trata do sexo do paciente;

[1] Versão 2 da base de dados disponível no seguinte link: <https://www.kaggle.com/datasets/hgunraj/covidxct>, que pode ser encontrada no seguinte endereço eletrônico: <https://www.kaggle.com/datasets/c395fb339f210700ba392d81bf200f766418238c2734e5237b5dd0b6fc724fcb/versions/4>. Acesso em: 26/06/2022

4) Idade: Faz referência à idade do paciente, e possui uma gama de idades que vai desde crianças até idosos;

5) Achado: Informa qual foi o diagnóstico: COVID-19/Pneumonia/Normal;

6) Achado Verificado: Faz referência ao item anterior, e informa se o achado encontrado no paciente foi confirmado por um especialista ou não;

7) Fatias Selecionadas: Esta informação diz respeito às imagens do paciente, e informa quem selecionou as imagens do paciente, se foi um especialista ou automático;

8) Panorama: Se trata da perspectiva do exame, na grande maioria dos exames, as imagens estão na perspectiva axial;

9) Identificador: Cada paciente possui uma chave identificadora, que une suas informações com as respectivas imagens no banco de dados.

Estas informações foram tabeladas em um único arquivo (no formato csv), em que cada paciente possui estas informações supracitadas. Também está disponibilizado um arquivo de texto (no formato txt) com as informações de rótulo, em que 0 corresponde à classe normal, 1 corresponde à classe pneumonia e 2 corresponde à classe COVID-19. Neste arquivo de texto também se encontram as informações de caixa delimitadora (do inglês, *Bounding Box*)[2], esta informação consiste em quatro coordenadas cartesianas, que correspondem aos pixels que compõem uma região de formato retangular que delimitam, no interior da imagem, a localização aproximada do corpo do paciente (observado na primeira imagem à esquerda na Figura 3.5).

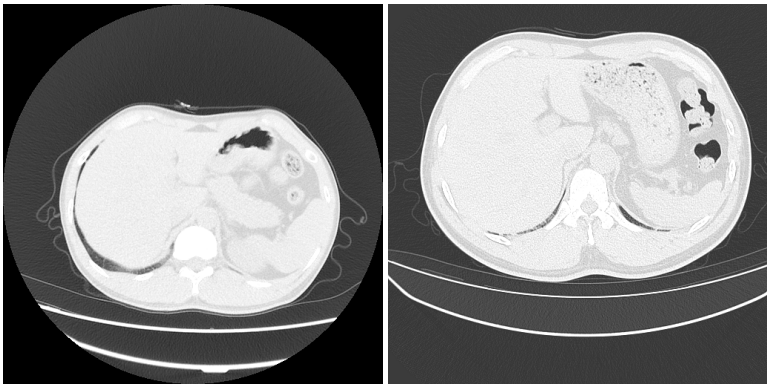
Além das informações associadas aos exames (também chamadas de metadados), é importante destacar algumas características visuais de alguns exames. Alguns dos exames foram diretamente agregados ao banco de imagens, sem uma seleção devida, e em algumas imagens o parênquima pulmonar não está visível, como mostra a Figura 3.2.

Apesar de pertencerem a pacientes diagnosticados com COVID-19, as Figuras 3.2.(a) e 3.2.(b) podem interferir no desempenho do algoritmo, uma vez que os modelos são treinados a partir destes dados que não apresentam minimamente o pulmão, tão pouco as características associadas à doença. Além disso, existem imagens com diferentes proporções

[2] Esta informação pode ser obtida de forma manual (realizando delimitações por meio de um *software*), ou por um modelo de segmentação por instância, como o YOLO, MobileNet, Mask R-CNN, entre outros.

Figura 3.2 – Exemplos em que o parênquima não está visível.

(a) (b)

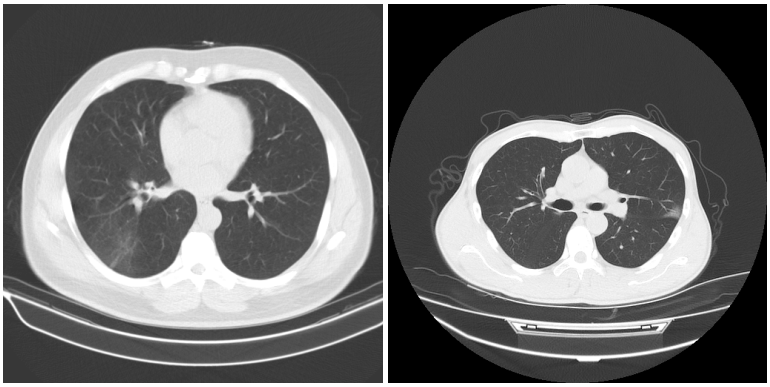


Fonte: (GUNRAJ *et al.*, 2021).

entre a região do pulmão e a imagem como um todo, como pode ser observado nos exemplos da Figura 3.3.

Figura 3.3 – Exemplos em que os parênquimas estão em diferentes proporções.

(a) (b)



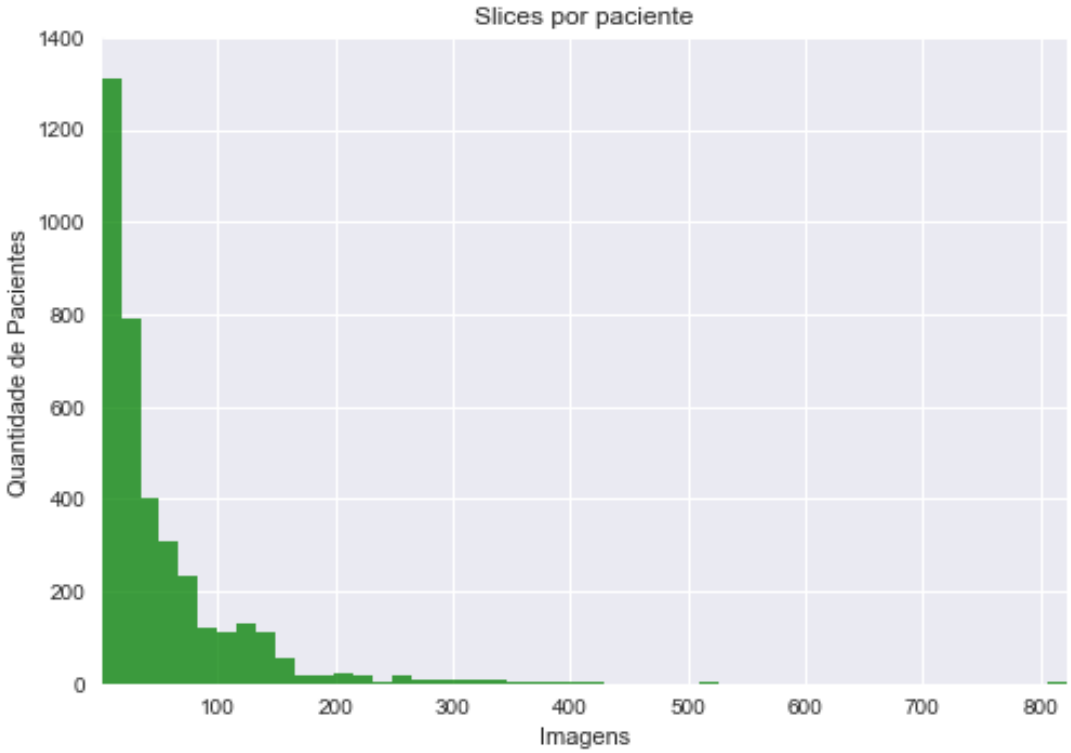
Fonte: (GUNRAJ *et al.*, 2021)

Este segundo problema se refere ao tamanho do pulmão em relação à imagem, note que a Figura 3.3.(a) apresenta pulmões maiores em relação à Figura 3.3.(b). Isto ocorre devido as diferentes origens dos exames, obtidos por equipamentos diferentes, e também pelo tamanho dos pacientes (adultos e crianças principalmente).

Além de amostras com diferentes proporções e com imagens sem uma visibilidade adequada do parênquima pulmonar, a quantidade de imagens por paciente também não é uniforme, como pode ser observado no histograma da Figura 3.4.

Para tentar contornar os problemas presentes nesta base de dados, alguns pré-processamentos foram necessários, eles serão detalhados na seção a seguir.

Figura 3.4 – Histograma de imagens por paciente.



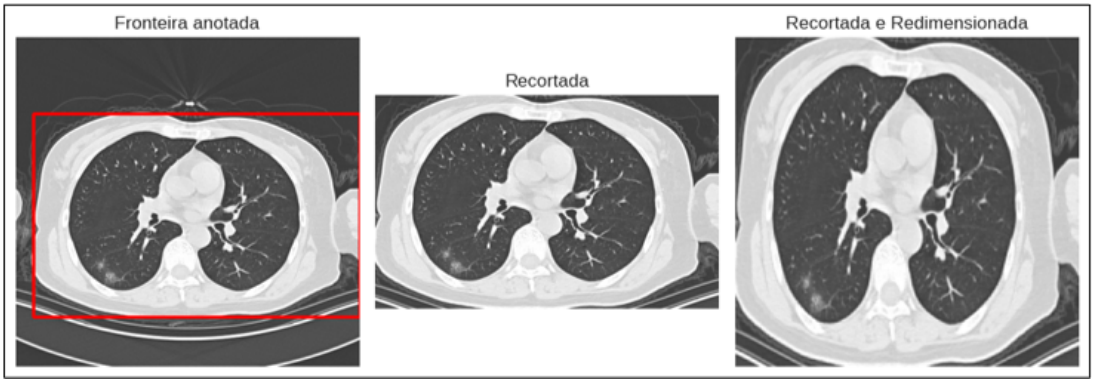
Fonte: Arquivo pessoal do Autor.

3.1.2 Segunda Etapa: Pré-Processamento

Na segunda etapa (referente ao segundo bloco da Figura 3.1), um algoritmo foi desenvolvido com a finalidade de selecionar imagens a serem utilizadas no treinamento. Inspirado pelo algoritmo proposto em Rahimzadeh, Attar e Sakhaei (2020), em que é utilizada uma região retangular para contabilizar a quantidade de pixels na região de interesse (próximo aos parênquimas pulmonares), foi desenvolvido um algoritmo similar para estimar a razão de pixels sobre a imagem.

Com base em uma informação contida na base de dados caixa delimitadora, foram realizados os procedimentos de recorte, redimensionamento e segmentação da imagem, como pode ser observado a seguir na Figura 3.5.

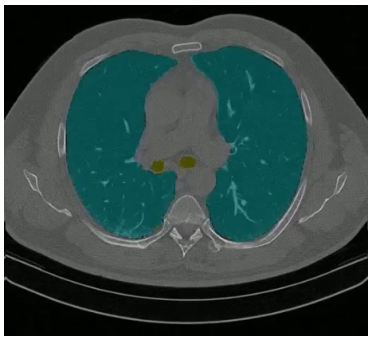
Figura 3.5 – Pré-processamento: Recorte e Redimensionamento.



Fonte: Arquivo pessoal do Autor.

Com base na informação de delimitação, a imagem é recortada e redimensionada ao formato original. Em seguida é utilizada como entrada em um modelo de segmentação do tipo UNET++[3], que retorna uma máscara binária na saída, ilustrada pela região sombreada em verde na Figura 3.6.

Figura 3.6 – Area Efetiva.



Fonte: Arquivo pessoal do Autor.

A partir deste resultado, são contabilizados os valores unitários na máscara binária gerada nesta segmentação, ou seja, é realizada a contagem de pixels na imagem referente ao pulmão do paciente. Esta contagem é então dividida pela dimensão total da imagem, gerando assim uma razão de proporcionalidade da área pulmonar em relação à área da imagem total.

Como é possível notar na Figura 3.3, ambas as imagens (a) e (b) possuem os parênquimas pulmonares visíveis, porém em proporções diferentes em relação a imagem, por isso existiu a necessidade do redimensionamento das imagens, para que o cálculo da

[3] Consiste na arquitetura UNET aninhada, ou seja, uma configuração em que a saída de um modelo UNET é utilizada na entrada de um segundo modelo. Os algoritmos utilizados para segmentação foram implementados em Python (utilizando a biblioteca Pytorch) por Yakubovskiy (2020).

proporção fosse corrigida pelo recorte e redimensionamento da imagem, realizando uma estimativa um pouco mais justa para diferentes pacientes.

A proporção foi então tomada como parâmetro para selecionar as imagens com uma área efetiva de pulmão. As imagens com uma taxa superior a 25%[4], foram direcionadas para o novo banco de dados, otimizando a quantidade de imagens que são utilizadas na etapa de treino, e tornando o treinamento computacionalmente mais eficiente, dadas limitações de processamento e quantidade de dados.

É importante destacar que a operação de recorte foi utilizada exclusivamente nesta etapa, para determinar a proporção aproximada dos pulmões sobre a região aproximada do corpo do paciente. Já o redimensionamento também ocorre na etapa de classificação, mas para adequar a imagem ao modelo, como será descrito.

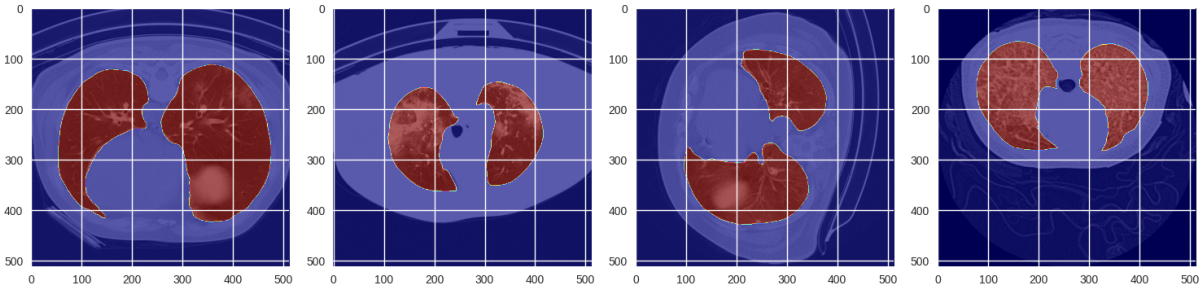
3.1.3 Terceira Etapa: Segmentação

A terceira etapa (quarto bloco da Figura 3.1) corresponde à segmentação dos parênquimas pulmonares, consiste na extração da região de interesse por meio da utilização da arquitetura Unet++, que tem seus pilares baseados em técnicas de segmentação semântica, especificamente da U-Net (seção 2.6.3.1), e que particularmente foi proposta no contexto de segmentação de imagens médicas (RONNEBERGER; FISCHER; BROX, 2015).

Para o treinamento do modelo de segmentação, foram aplicadas as técnicas de "*data augmentation*" de rotação, adição de ruído gaussiano e espelhamento vertical e horizontal; todos com uma taxa de 50%, fazendo a variabilidade aumentar consideravelmente. Estas transformações foram realizadas utilizando a biblioteca *Albunmentations* juntamente com o treinamento pela biblioteca *Pytorch* (BUSLAEV *et al.*, 2018; PASZKE *et al.*, 2019). As transformações ocorrem à medida que as imagens são carregadas, como apresentado na Figura 3.7.

[4] Esta taxa foi selecionada pela averiguação visual de imagens que continham minimamente os parênquimas pulmonares e também apresentavam lesões.

Figura 3.7 – Transformações utilizadas na segmentação. Da esquerda para direita são realizados: Espelhamento: 1ª, 2ª e 4ª imagens; Rotação: 3ª



Fonte: Arquivo pessoal do Autor.

As imagens e suas respectivas máscaras são carregadas ao mesmo tempo e passam pelo mesmo processo durante o treinamento. O modelo treina com o objetivo de gerar máscaras semelhantes às que foram anotadas. Além disso, as transformações não são utilizadas durante a fase de validação, e tão pouco na fase de testes.

Para segmentação do parênquima pulmonar, foram realizadas anotações pelo software *SLICER*-3D® com uma quantidade total de 800 imagens e máscaras, com auxílio das ferramentas semiautomáticas de *Grow From Seeds* e *Watershed*. Após as anotações devidamente organizadas, o treinamento foi realizado no formato de validação cruzada com $K = 10$ pastas, ou seja, os dados foram separados em 10 pastas, sendo oito pastas utilizadas para treinamento, uma para validação e uma para teste em cada iteração. Logo, foram treinados dez modelos de cada arquitetura. Para ambos os modelos foi utilizado o algoritmo otimizador ADAM, com taxa de aprendizado inicial igual a 1×10^{-4}, podendo ser reduzida conforme o passar das épocas (seção 2.3.3.2), caso a variação do erro de validação não se altere por 3 épocas, a taxa de aprendizado é reduzida. Como não foi utilizada a transferência de aprendizado, os pesos foram inicializados pelo método *GLOROT* Glorot e Bengio (2010), que utiliza uma distribuição uniforme.

Antes de prosseguir para próxima etapa, é importante ressaltar que todos os modelos foram treinados na configuração *mini-batch* com *batch size* igual a 8 para segmentação e 4 para classificação. Isto significa que o banco de dados é subdividido em 4 ou 8 partes iguais, e em cada época o modelo é treinado com um subconjunto.

3.1.4 Quarta Etapa: Treinamento

A quarta etapa corresponde ao treinamento do modelo, que consiste na arquitetura *Vision Transformer*, baseada em mecanismos de atenção (seção 2.5) . Esta arquitetura foi

selecionada por possuir um ótimo desempenho em problemas de classificação de imagens e a possibilidade da interpretabilidade dos resultados, ou seja, além da classificação, existem métodos ((CHEFER; GUR; WOLF, 2020); (CHEFER; GUR; WOLF, 2021)[5]) baseados na decomposição profunda de *Taylor*, que são utilizadas para determinar a relevância dos pesos, e, portanto, consegue determinar aproximadamente, quais pixels da imagem mais interferem no resultado final do modelo. Na implementação disponibilizada pelos autores, várias configurações de modelos foram disponibilizadas, entre elas foi escolhida a configuração "*vit_base_patch16_224*", em que "*base*" corresponde ao modelo básico, com menor quantidade de parâmetros ($85.648.130$); o valor 224 corresponde à dimensão de entrada das imagens $(224, 224, 3)$, e *patch16* informa que a imagem é recortada em fatias de dimensão $(16, 16)$, como a imagem possui dimensão $(224, 224, 3)$, no total são 14 fatias de $(16, 16, 3)$. O modelo também consiste de 24 camadas ou blocos de atenção, seguidos por uma camada de saída.

Como o modelo utilizado foi originalmente configurado para operar com imagens no formato $(224, 224, 3)$, todas as imagens foram redimensionadas para este formato no momento em que são carregadas no ambiente de treinamento virtual do GOOGLE COLAB®. Para tal, um método de interpolação foi utilizado, o método de interpolação por área (Veja o apêndice A.5).

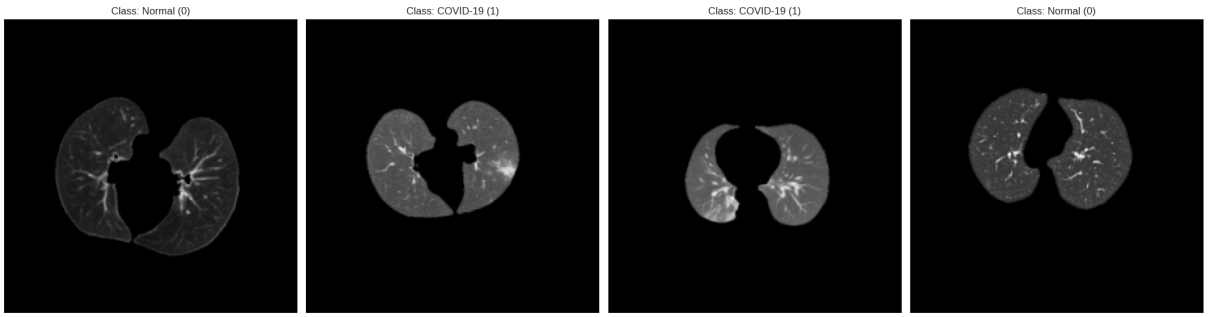
Em muitos casos quando a dimensão de atributos por amostra (quantidade de características) supera a quantidade de amostras com grande diferença, então surge uma dificuldade maior em solucionar problemas e identificar padrões, tal problema é chamado de maldição da dimensionalidade (BISHOP, 2016). Por este motivo, foram escolhidas algumas técnicas de *Data Augmentation* para aumentar a variabilidade das amostras e forçar o modelo a aprender múltiplas representações do mesmo problema.

Como descrito na seção 2.7.1, os algoritmos de "*data augmentation*" são transformações relativamente simples, quando comparados ao aumento por redes adversárias generativas (do inglês, *Generative Adversarial Networks* - GANs) - (MIKOŁAJCZYK; GROCHOWSKI, 2019). Dentre as técnicas mencionadas na seção 2.7.1, foram utilizadas as transformações de rotação (entre 0^o e 15^o) e o espelhamento, tanto para os modelos de segmentação do parênquima pulmonar quanto para classificação. Apesar de se tratar de modificações simples, elas são plausíveis de ocorrerem na prática, seja pela posição do

[5] No repositório disponibilizado, são contempladas diferentes implementações e variações do modelo *vision tranformer*, com implementação baseada no Pytorch.

paciente ou pela configuração do software utilizado, as imagens podem ser alteradas neste aspecto. Abaixo são mostrados alguns exemplos na Figura 3.8:

Figura 3.8 – Data Augmentation. Da esquerda para direita: 1ª e 2ª: Espelhamento vertical e rotação; 3ª Imagem: Espelhamento Horizontal e 4ª Imagem: Rotação.



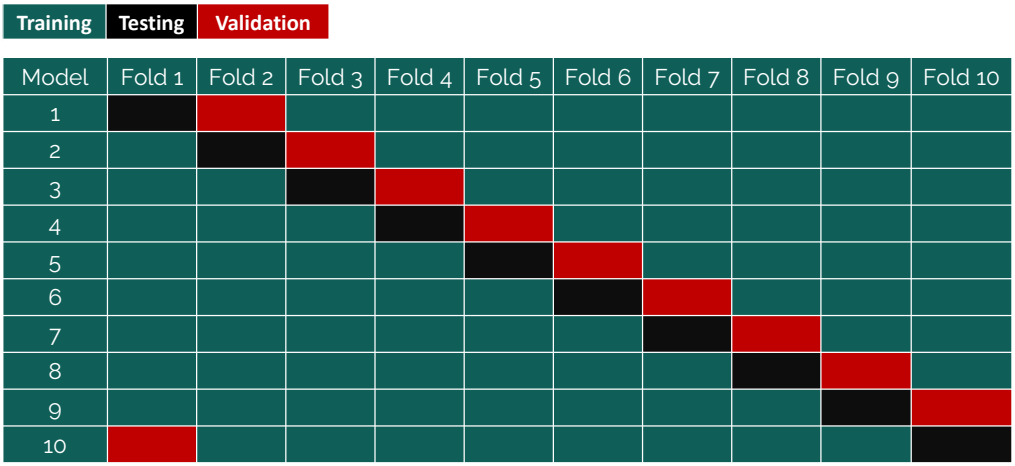
Fonte: Arquivo pessoal do Autor.

Note que as imagens apresentam somente a região de interesse, ou seja, as imagens são previamente segmentadas. Estas modificações de "*data augmentation*" foram realizadas por meio da biblioteca *Opencv-Transforms* Bohnslav (2019).

Além das transformações, o algoritmo otimizador foi alterado para o AdamW proposto em Loshchilov e Hutter (2017), este algoritmo otimizador consiste em uma variação do algoritmo ADAM, que possibilita a regularização de forma desacoplada com respeito à função de custo. A taxa de aprendizado também foi variável, inicializada com $1x10^{-4}$ e variando no caso da não redução do erro de validação a cada 10 épocas.

Por fim, o modelo é validado por uma técnica comumente utilizada, a validação cruzada com $K = 10$, conforme a Figura 3.9. Como descrito na seção 2.11.1, a cada modelo treinado, as pastas são comutadas entre si, fazendo com que os modelos sempre tenham conjuntos de treino teste e validação distintos.

Figura 3.9 – Esquema de divisão por K-*Fold* (Treino/Teste/Validação).



Fonte: Arquivo pessoal do Autor.

Ao final do treinamento, têm-se uma margem de erro que o modelo possui, ou intervalo de confiança. Após a validação do modelo, a próxima etapa consiste em fornecer ao médico, o resultado final do modelo.

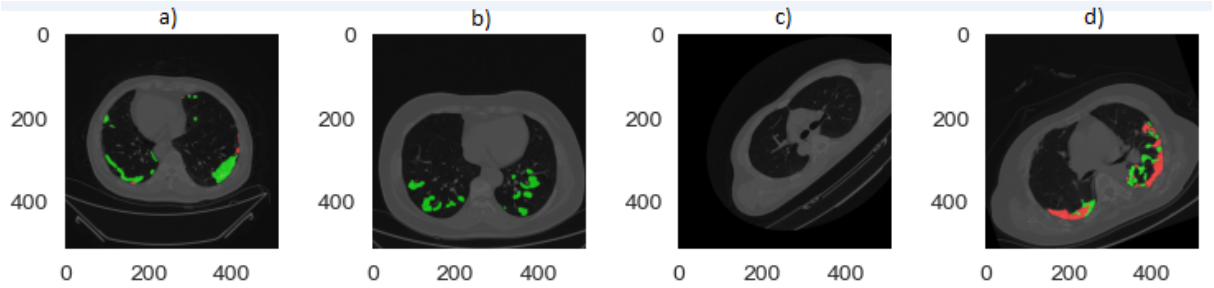
3.1.5 Quinta Etapa: Triagem

A quinta e última etapa consiste na proposta final deste projeto, que reúne todas as informações extraídas das imagens do paciente e informa ao médico responsável. Como pode ser observado na Figura 3.1, além da informação de classificação fornecida pelo modelo, também é informada qual a região da imagem o modelo (aproximadamente) utiliza para realizar tal inferência. Para tal, foi aplicado o algoritmo de interpretabilidade proposto em Chefer, Gur e Wolf (2021). Esta informação fornece ao radiologista a região da imagem com maior grau de relevância de acordo com a respectiva classificação.

Para complementar ainda mais esta informação de interpretabilidade, também foi realizada uma segmentação de lesões, especificamente das lesões opacidade em vidro fosco e consolidação, mencionadas na seção (2.2.1). Esta segmentação de lesões também passa por um processo de treinamento semelhante à segmentação do parênquima pulmonar, com uma leve distinção em relação às técnicas de "*data augmentation*". Devido a baixa quantidade de amostras e à dificuldade em realizar anotações, foram utilizadas técnicas de

aumento/aprimoramento um pouco mais elaboradas, como a técnica CLAHE utilizando a biblioteca *OpenCV* (BRADSKI, 2000). Também foram aplicadas as técnicas de translação, cisalhamento, rotação e espelhamento, com auxílio da biblioteca *Imgaug* Jung (2020) e da implementação de Verma (2020). É importante ressaltar que todas estas técnicas foram aplicadas somente durante a etapa de treinamento, com uma frequência pré-determinada. Para esta segmentação também foi realizado uma validação cruzada com $K = 5$, justamente pelo fato da quantidade escassa de amostras. Como exemplo, podem ser observadas as modificações realizadas nas imagens, na Figura 3.10.

Figura 3.10 – Data Augmentation. Utilizando a biblioteca imgaug. Imagens sem alterações (a e b) e com operações como rotação, translação e cisalhamento (imagens c e d).



Fonte: Arquivo pessoal do Autor.

Assim como na segmentação dos parênquimas pulmonares, na etapa de lesões foram comparados os modelos, UNet++ e DeepLabV3+. A principal diferença está na quantidade de imagens anotadas, que foram anotadas com auxílio das ferramentas base nos trabalhos mencionados na seção 2.2.1, totalizando 1810 imagens de 24 pacientes distintos com suas respectivas máscaras. As imagens foram selecionadas com base na informação presente na tabela descritiva de cada paciente (Fatias Selecionadas)[6], em caso afirmativo o paciente foi analisado com base na quantidade de fatias com lesões, os pacientes com lesões visíveis foram selecionados. Os exames presentes nesta base possuem diversos tipos de lesões, porém, devido à limitação de conhecimentos e experiência acerca de todos os tipos de lesões, para esta análise as anotações foram limitadas em consolidação e opacidade em vidro fosco, a primeira para regiões com maior predominância de pixels brancos agregados em uma região, enquanto para opacidade foram consideradas as regiões acinzentadas (mas ainda com uma visibilidade do pulmão).

Obtendo este resultado da segmentação de lesão no pulmão, é possível que o profissional compare o resultado de explicabilidade de forma visual com o resultado de seg-

[6] Se os *slices* foram selecionados por especialistas, seção 3.1.1.1.

mentação de lesões. E também possibilita conferir se o modelo realmente tem seu foco na região infectada.

Além disso, é possível contabilizar quantos pixels no parênquima pulmonar correspondem a cada tipo de lesão, e gerar uma taxa de infeção de cada imagem do paciente. Como foi mostrado na etapa 2 (seção 3.1.2), cada paciente possui uma quantidade de imagens diferentes, e com o pulmão em diferentes proporções em relação à imagem como um todo; logo, esta proporção pode ser comprometida e não informar o real estado do paciente. Para tentar contornar estas adversidades e tomar uma proporção adequada, a taxa de lesão é tomada em relação à quantidade total de pixels do parênquima pulmonar (fornecidos pelo modelo de segmentação do parênquima). Fornecendo uma taxa mais justa, caso seja necessário comparar o grau de severidade de dois pacientes diferentes.

Os modelos de segmentação do parênquima pulmonar foram validados com $K = 10$, bem como os modelos de classificação. Somente os modelos de segmentação de lesões foram treinados com $K = 5$, ficando com 20% de amostras em cada uma das pastas.

4 RESULTADOS

Neste capítulo, serão discutidos os principais resultados obtidos durante o desenvolvimento deste trabalho. Em todos os modelos, desde a segmentação do parênquima pulmonar até a classificação do modelo *Vision Transformer*, foi utilizada a validação cruzada, com intuito de fornecer um resultado estimado do desempenho do modelo em dados desconhecidos.

4.1 Segmentação

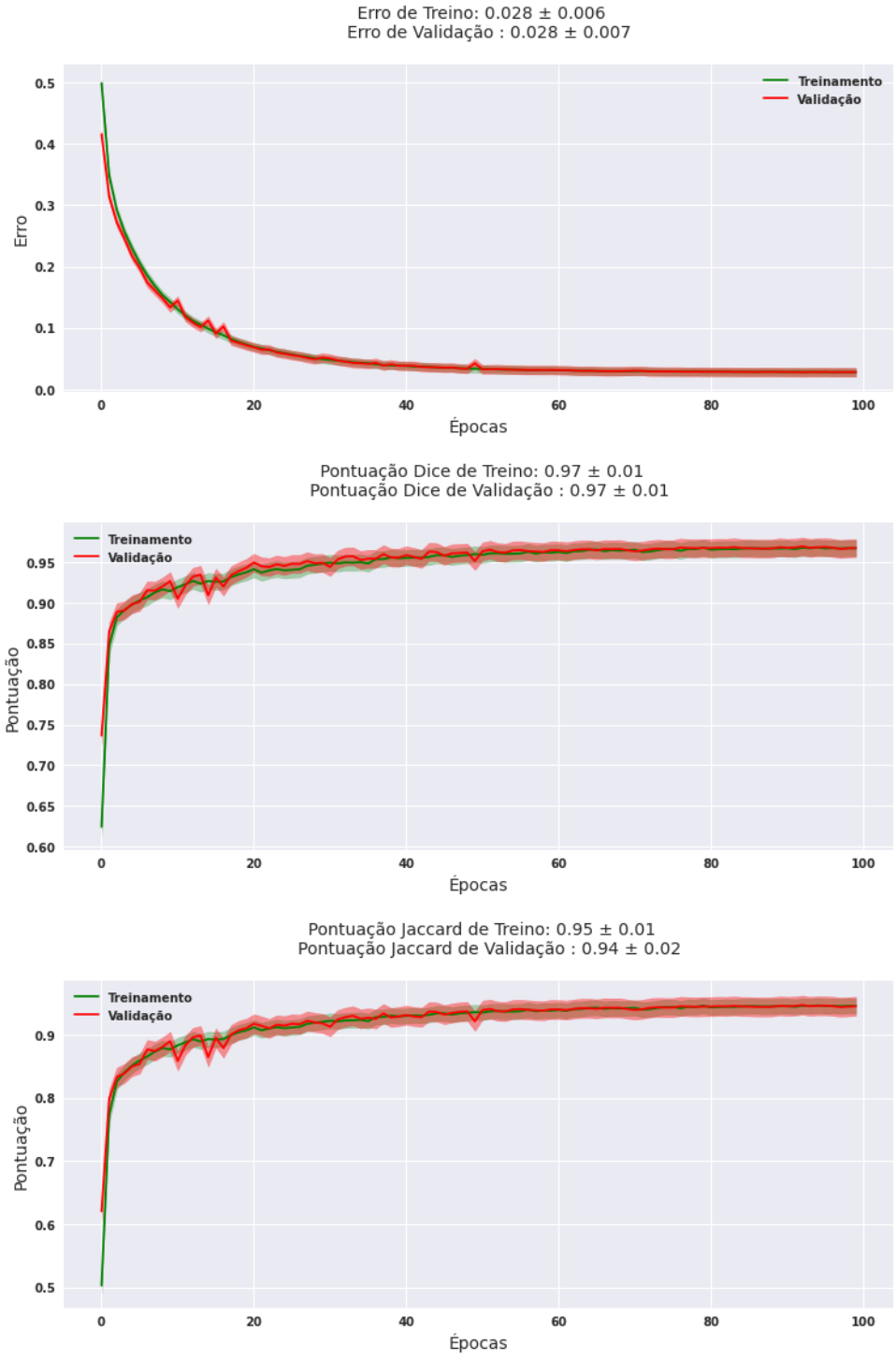
O modelo de segmentação foi utilizado em três etapas deste projeto, no pré-processamento, auxiliando na seleção de imagens com maior proporção do parênquima pulmonar; na etapa de segmentação efetiva, na qual não é realizado nenhum ajuste prévio na imagem com base na informação *bounding box*; e também na etapa de triagem, confrontando o resultado de explicabilidade do modelo, com a segmentação de lesões.

4.1.1 *Unet++ vs DeepLabV3+*

O resultado de treinamento para a arquitetura UNet++ ficou conforme mostra a Figura 4.1. Como pode ser observado o modelo conseguiu progredir com razoável estabilidade, havendo apenas algumas flutuações nas épocas iniciais. Porém, como foi utilizada uma taxa de aprendizado variável através das épocas, a flutuação foi menor na etapa final.

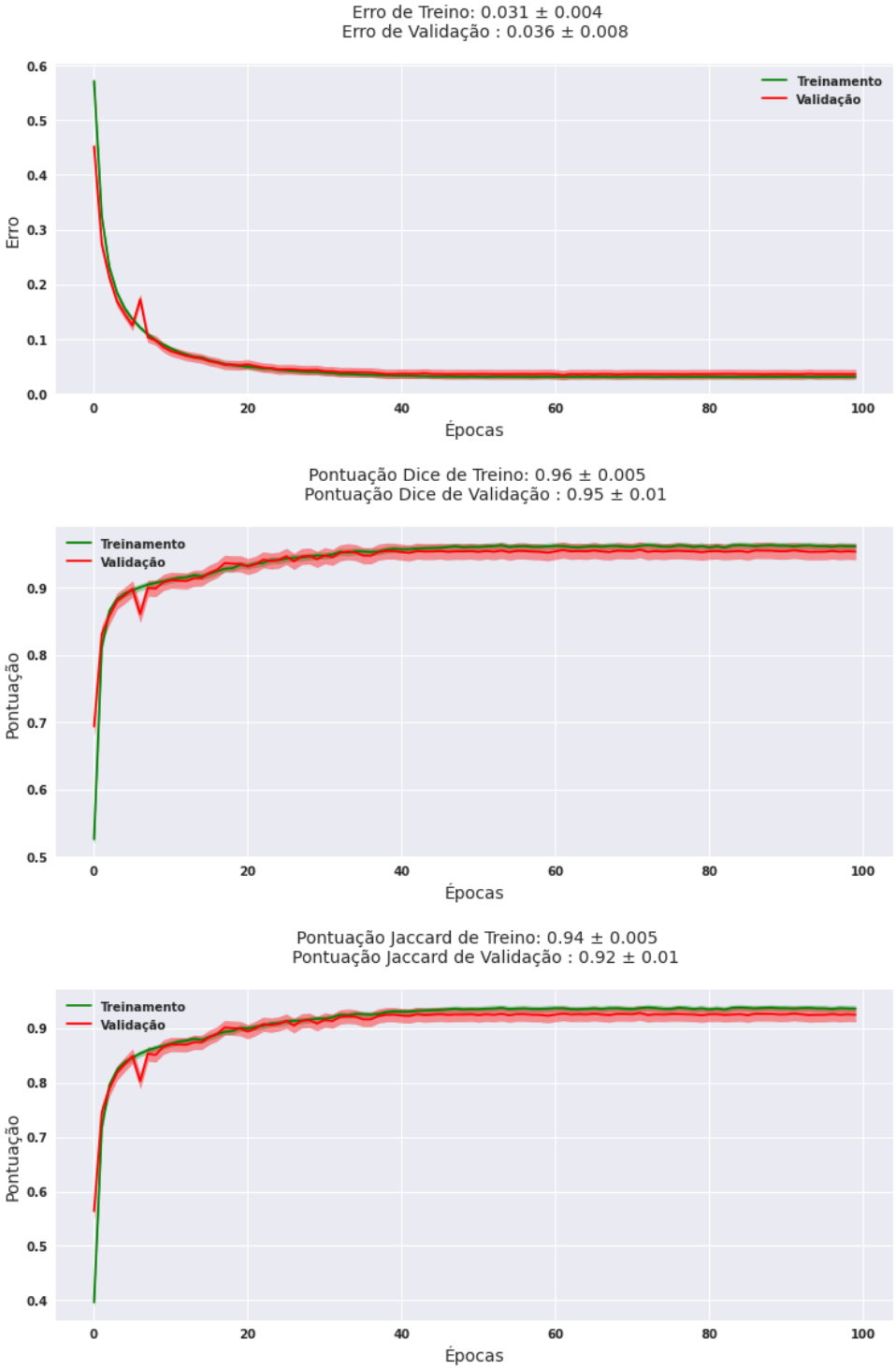
Para o modelo *DeepLab V3+*, foram utilizadas as mesmas configurações de treino e conjunto amostral. Os resultados de treinamento para esta arquitetura podem ser visualizados na Figura 4.2. Assim como a Unet++, houve somente algumas flutuações no início do treinamento.

Figura 4.1 – Resultado das etapas de treinamento e validação para arquitetura Unet++.



Fonte: Arquivo pessoal do Autor.

Figura 4.2 – Resultado das etapas de treinamento e validação para arquitetura *DeepLab V3 Plus*.

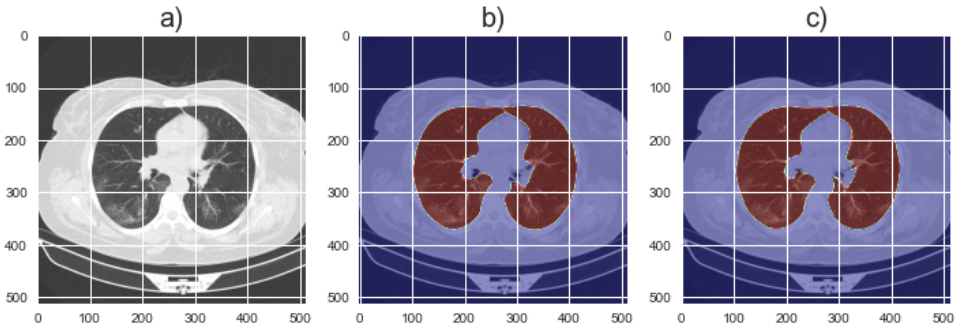


Fonte: Arquivo pessoal do Autor.

A título de comparação a quantidade de parâmetros no UNet++ é $26.078.609$ enquanto que o *DeepLab V3+* possui uma quantidade total de $22.437.457$ parâmetros. Vale destacar que os valores medidos e mostrados nas Figuras (4.1) e (4.2) são os valores médios das 10 pastas, bem como os valores nos gráficos, as margyens foram calculadas pelo desvio padrão da média, e o desvio padrão final foi adicionado como incerteza.

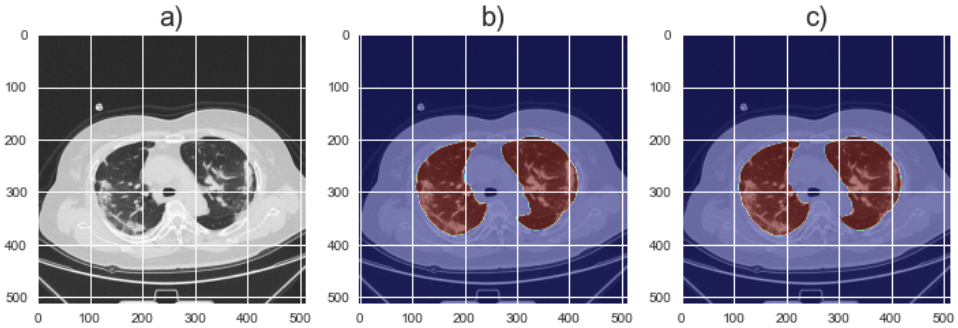
Como comparação visual, ambos os modelos apresentaram bons resultados, como pode ser visto nos exemplos a seguir das Figuras (4.3) e (4.4):

Figura 4.3 – Exemplo 1 (Paciente com lesões não severas) de Comparação entre os resultados visuais obtidos pelas arquiteturas testadas, a) imagem original do paciente, b) resultado do modelo UNet++ , e c) resultado do modelo *DeepLabV3+*.



Fonte: Arquivo pessoal do Autor.

Figura 4.4 – Exemplo 2 (Paciente com lesões severas) de Comparação entre os resultados visuais obtidos pelas arquiteturas testadas, a) imagem original do paciente, b) resultado do modelo UNet++ , e c) resultado do modelo *DeepLabV3+*.



Fonte: Arquivo pessoal do Autor.

Na Figura 4.4, a imagem a) corresponde a imagem original do paciente, a imagem b) corresponde à imagem sobreposta pela região (máscara) que o modelo UNet++ classifica como pulmão (em vermelho), e a imagem c) se refere a região que o modelo *DeepLabV3+* classifica como pulmão. Tanto na Figura 4.3 quanto a Figura 4.4 são de pacientes diagnosticados com COVID-19, porém o paciente da Figura 4.4, possui os parênquimas pulmonares

significativamente mais lesionados pelo vírus que o primeiro. Além disso, vale destacar que nas primeiras etapas de treinamento, antecedentes a efetivação do projeto e seus modelos, algumas imagens com regiões acometidas por lesão do tipo consolidação (região na tonalidade branca na Figura 4.4), fizeram com que os modelos de segmentação apresentassem regiões de falso negativo nestas regiões. Isto ocorreu, pois, a primeira base de máscaras para segmentação utilizada foi proposta para outra finalidade, particularmente para exames de tomografia provenientes de pacientes com fibrose pulmonar (SÁNDOR, 2020). Para contornar este problema, esta nova base de dados com 800 imagens foi proposta com as máscaras de pacientes com COVID-19. As anotações nestes casos se tornam uma tarefa complexa, pois quando a infecção se encontra no limiar entre a região do pulmão e o restante do corpo, as regiões parecem ser homogêneas em alguns casos, principalmente na ocorrência do derrame pleural.

Após a etapa de treinamento, foi realizada uma etapa de teste, onde é efetivada a validação cruzada, nesta etapa foram realizadas as inferências de todas as amostras no conjunto de teste pelos seus respectivos modelos, do resultado foi estimada a média e desvio padrão, organizados na Tabela (4.1).

Tabela 4.1 – Resultado comparativo de desempenho dos modelos UNet++ e *DeepLabV3+* aplicados aos conjuntos de teste.

Unet++ vs Deep Lab V3+		
Modelo	Pontuação Dice	Pontuação Jaccard
UNET++	0,97 ± 0,02	0,95 ± 0,05
DeepLabV3+	0,97 ± 0,03	0,94 ± 0,05

Fonte: Arquivo Pessoal do Autor.

Da mesma forma que o treinamento, os resultados de teste foram muito aproximados entre os dois modelos, mostrando que ambos conseguem identificar o pulmão corretamente dentro da amostragem que foi realizada. A tabela 4.2 reúne as informações de treino, teste e validação de ambos os modelos.

Tabela 4.2 – Resultado comparativo de desempenho dos modelos UNet++ e *DeepLabV3+* aplicados aos conjuntos de treino, teste e validação.

	Unet++ vs Deep Lab V3+					
	Treino		Validação		Teste	
Modelo	Pontuação Dice	Pontuação Jaccard	Pontuação Dice	Pontuação Jaccard	Pontuação Dice	Pontuação Jaccard
Unet++	0,97 ± 0,01	0,95 ± 0,01	0,97 ± 0,01	0,94 ± 0,02	0,97 ± 0,02	0,95 ± 0,05
Deep Lab V3+	0,96 ± 0,01	0,94 ± 0,01	0,95 ± 0,01	0,92 ± 0,01	0,97 ± 0,03	0,94 ± 0,05

Fonte: Arquivo Pessoal do Autor.

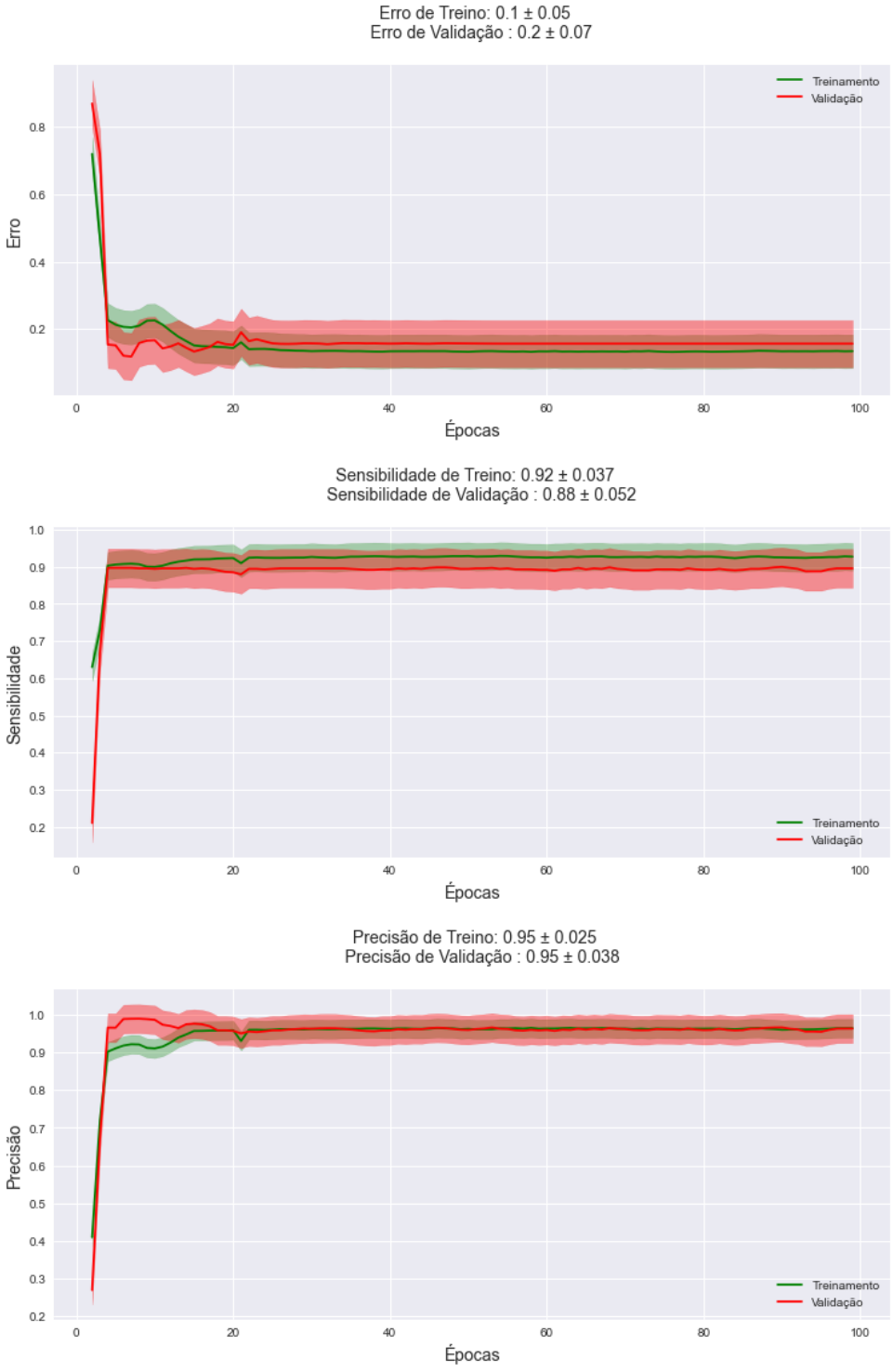
4.2 Classificação COVID-19 vs Normal

Como mencionado anteriormente, o modelo de classificação escolhido foi o *Vision Transformer*, e os algoritmos para treinar os modelos, bem como para gerar os mapas de interpretabilidade, foram adaptados de (CHEFER; GUR; WOLF, 2021). Este modelo foi escolhido devido ao seu bom desempenho e a possibilidade de se gerar mapas de explicabilidade, pois no contexto delicado que é tratar da saúde de outras pessoas, a classificação simples pode não ser o suficiente, ou seja, somente a rotulação não informa o estado real do paciente.

Todos os 10 modelos foram treinados por um pouco mais de 100 épocas, e em seguida os resultados foram truncados em 100 para exibir o comportamento das medidas monitoradas no treinamento. Os resultados de treinamento podem ser observados na Figura 4.5. Pelo gráfico da curva de aprendizado, nota-se que os modelos tiveram uma rápida aprendizagem na classificação, e também apresentaram um desvio um pouco mais acentuado que os modelos de segmentação do parênquima pulmonar. Além disso, alguns modelos apresentaram uma flutuação um pouco mais acentuada no treinamento, sugerindo uma maior variabilidade entre as amostras dos conjuntos de treino, validação e de teste.

No conjunto de teste os resultados foram similares, como é possível notar na tabela (4.3). Além disso, ao final do treinamento, o modelo *Vision Transformer* apresentou uma maior especificidade do que sensibilidade, ou seja, o modelo consegue identificar melhor as imagens de pacientes normais do que os pacientes com COVID-19. Algumas hipóteses para esta diferença podem ser levantadas, como por exemplo a inexistência de lesões em algumas imagens de alguns pacientes diagnosticados com COVID-19. Além disso, a distribuição de imagens não é uniforme, como observado no histograma da Figura 3.4, possivelmente influenciando nestes resultados. Na tabela 4.4 estão resumidos os resultados das medidas monitoradas no treinamento, validação e na fase de teste. Como a especificidade e valor

Figura 4.5 – Monitoramento da sensibilidade e precisão para arquitetura *Vision Transformer*, durante as etapas de treinamento e validação.



Fonte: Arquivo pessoal do Autor.

Tabela 4.3 – Resultado final do modelo *Vision Transformer* aplicado ao conjunto de teste.

	COVID-19 vs Normal			
Modelo	Sensibilidade	Especificidade	Precisão	Valor Preditivo Negativo
Vision Transformer	0,915 ± 0,094	0,985 ± 0,034	0,985 ± 0,035	0,929 ± 0,075

Fonte: Arquivo Pessoal do Autor.

preditivo negativo não foram monitoradas no treinamento dos 10 modelos, não foram tabulados.

Tabela 4.4 – Resultado final do modelo *Vision Transformer* aplicado ao conjunto de treino, validação e teste.

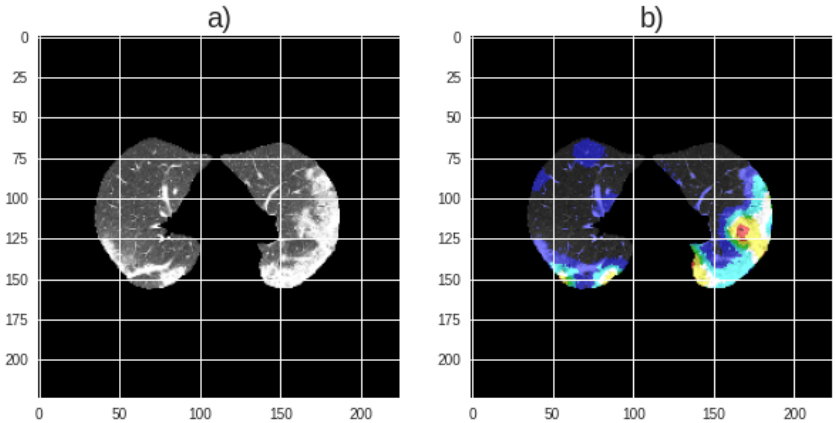
	COVID-19 vs Normal (*Vision Transformer*)			
Etapa	Sensibilidade	Especificidade	Precisão	Valor Preditivo Negativo
Treino	0,920 ± 0,037	-	0,950 ± 0,025	-
Validação	0,880 ± 0,052	-	0,950 ± 0,038	-
Teste	0,915 ± 0,094	0,985 ± 0,034	0,985 ± 0,035	0,929 ± 0,075

Fonte: Arquivo Pessoal do Autor.

4.2.1 Interpretabilidade

Utilizando o método proposto em (CHEFER; GUR; WOLF, 2021), foram realizadas algumas inferências com imagens de pacientes com COVID-19, e o modelo consegue captar com eficiência as lesões, salvo algumas exceções. Veja o exemplo da Figura 4.6.

Figura 4.6 – Exemplo 1 (Interpretabilidade aplicada sobre imagem de paciente), a) imagem Original, b) Resultado de Interpretabilidade do modelo *Vision Transformer*.

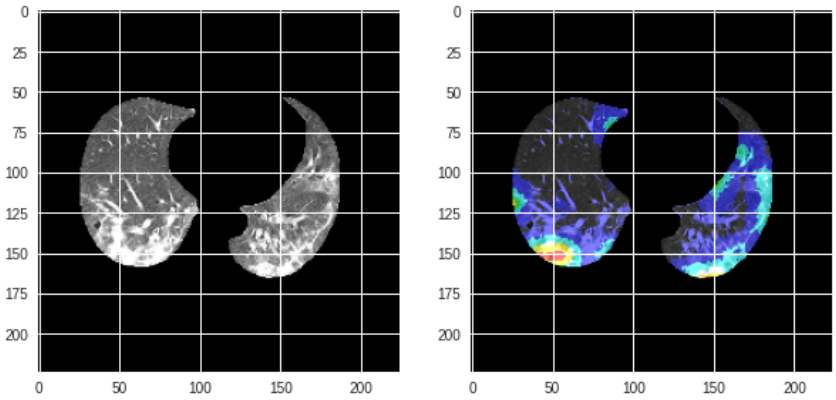


Fonte: Arquivo pessoal do Autor.

A região demarcada corresponde ao mapa de explicabilidade gerado pelo método baseado na decomposição de *Taylor*. A escala de cores corresponde ao nível de explicabilidade,

a cor vermelha representa a maior importância, a cor amarela representa uma importância intermediária e a verde é a menor importância dentro do mapa de explicabilidade. Nota-se que a maior parte foi mapeada pelo método nesta amostra. Mas em outros pacientes, algumas regiões podem ter sido ignoradas pelo modelo, como observado na Figura 4.7.

Figura 4.7 – Exemplo 2 (Interpretabilidade aplicada sobre imagem de paciente), a) imagem Original, b) Resultado de Interpretabilidade do modelo *Vision Transformer*.



Fonte: Arquivo pessoal do Autor.

Neste caso, algumas regiões não foram identificadas relevantes pelo modelo, apesar de conterem lesões, o que na prática seria imediatamente observado por um especialista.

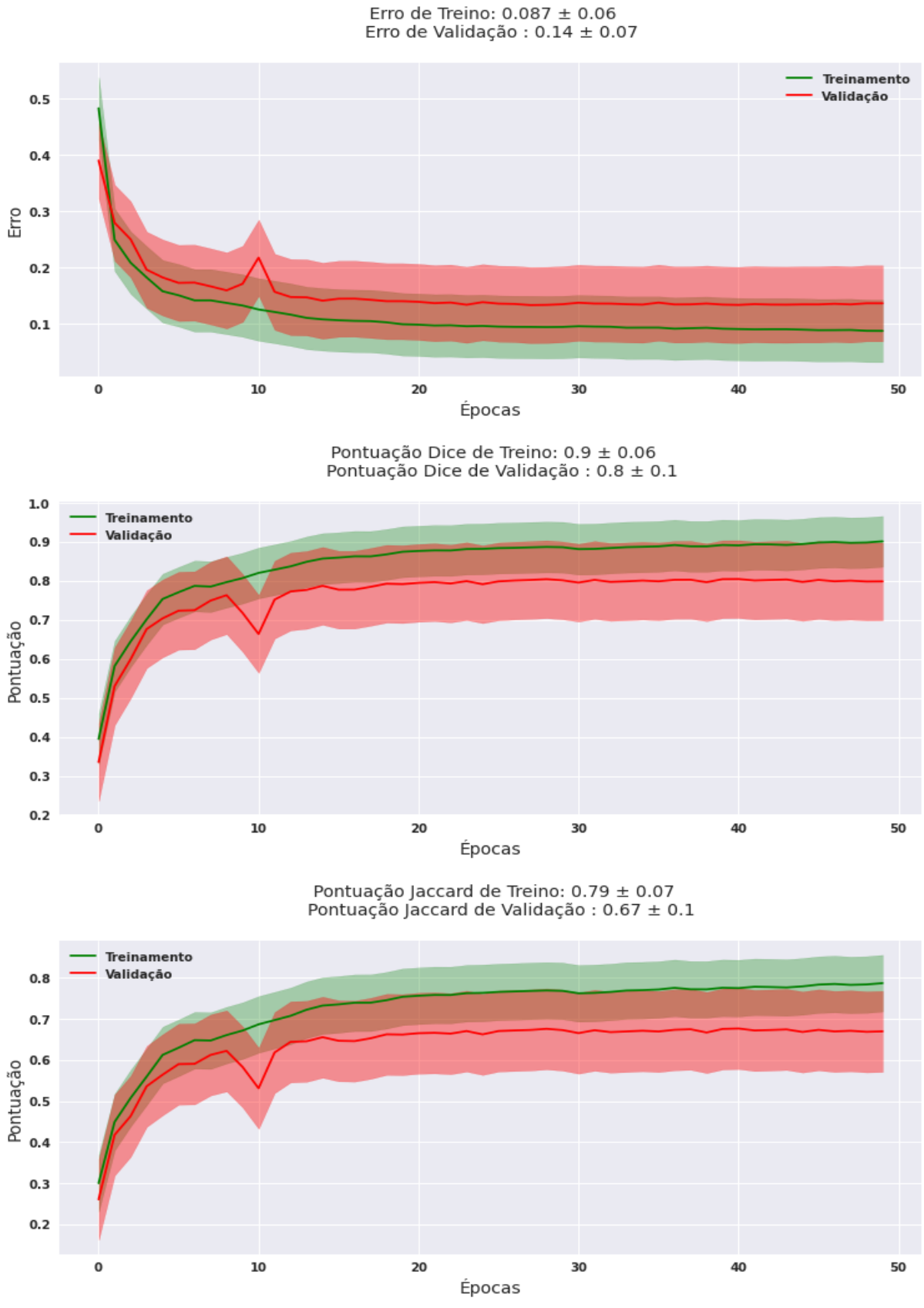
Esta informação de explicabilidade, pode ajudar tanto o profissional a conferir se o resultado condiz com a real condição do paciente, como também para o sistema como um todo, uma vez que é possível avaliar se o modelo realmente detecta regiões internas que são pertinentes ao problema.

4.3 Classificação dos tipos de lesões: Opacidade em Vidro Fosco e Consolidação

Após o treinamento dos modelos de segmentação de lesões, foi realizada uma validação cruzada com $K = 5$ para ambos os modelos (Unet++ e *DeepLab*V3+). Os resultados podem ser observados nas Figuras 4.8 e 4.9.

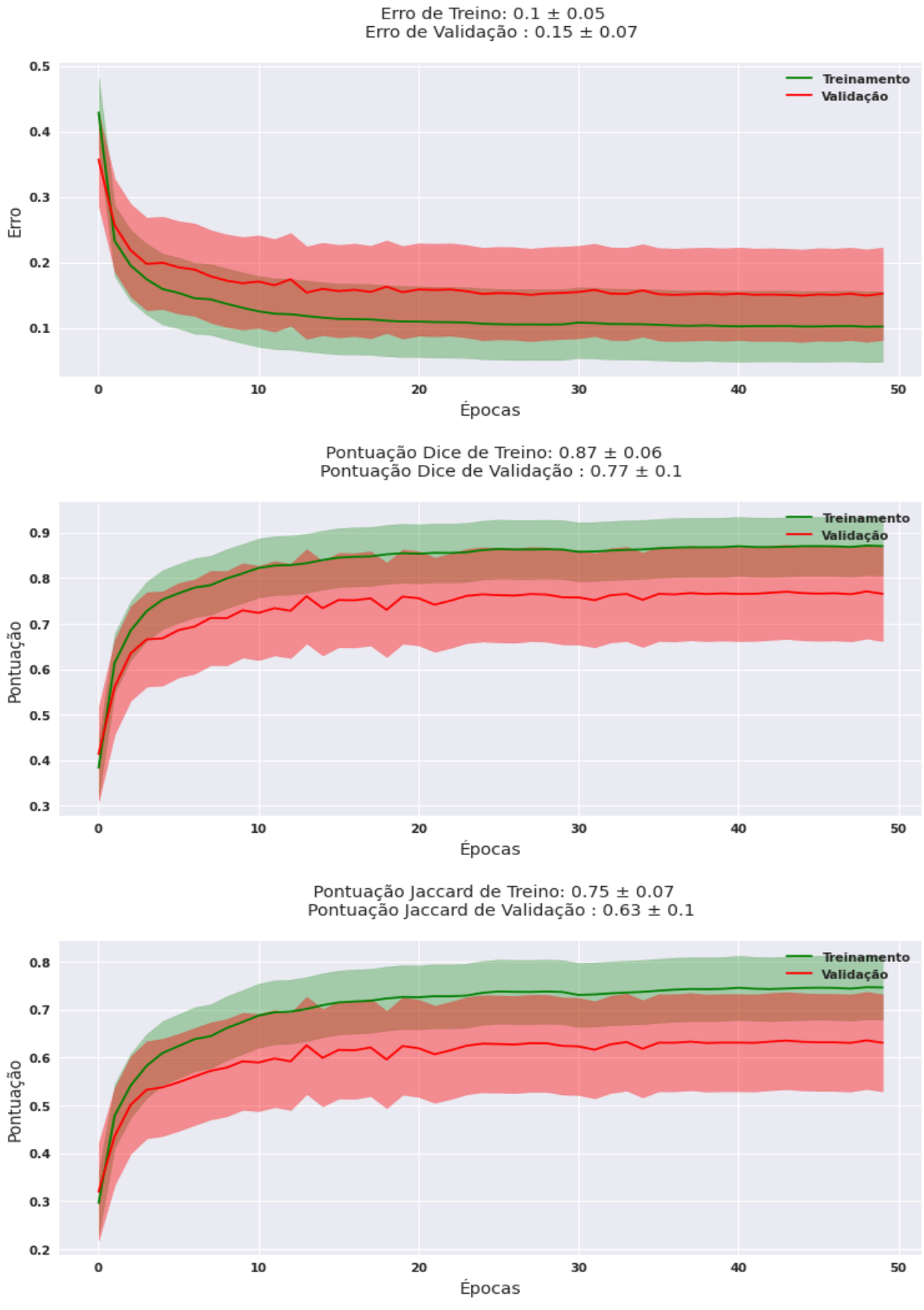
Como pode ser observado, nesta etapa os modelos apresentaram uma flutuação bem maior que nas etapas anteriores. Apesar de parecer um problema simples, diferenciar pixels na região pulmonar com a tonalidade cinza com pixels na tonalidade branca, alguns pacientes possuem regiões em que os dois tipos de lesões se mesclam, aumentando a dificuldade em realizar as anotações. As duas lesões são bem diferenciáveis quando o paciente apresenta apenas uma delas, porém quando o paciente apresenta diferentes tipos

Figura 4.8 – Resultado das etapas de treinamento e validação para arquitetura *Unet++*, considerando o Erro e as pontuações Dice e Jaccard.



Fonte: Arquivo pessoal do Autor.

Figura 4.9 – Resultado das etapas de treinamento e validação para arquitetura *DeepLab V3 Plus*, considerando o Erro e as pontuações Dice e Jaccard.

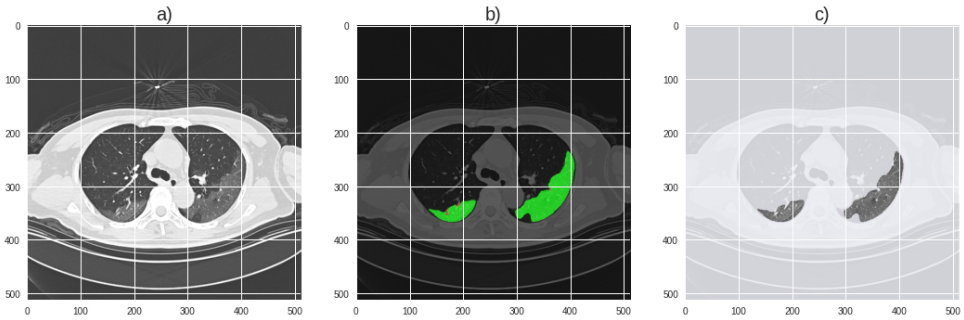


Fonte: Arquivo pessoal do Autor.

de lesão (similares à opacidade e consolidação), é necessário o conhecimento e experiência de um especialista sobre o tipo de lesão de que se trata, pois existem outras nomenclaturas para tais lesões, como a pavimentação em mosaico, sinal do Halo, sinal do Halo invertido, entre outros.

Ao aplicar o modelo treinado UNet++ sobre uma imagem de paciente com COVID-19, obtém-se o resultado da Figura 4.10. Neste exame é possível observar a predominância da lesão do tipo opacidade em vidro fosco, e o modelo consegue distinguir com precisão a região afetada.

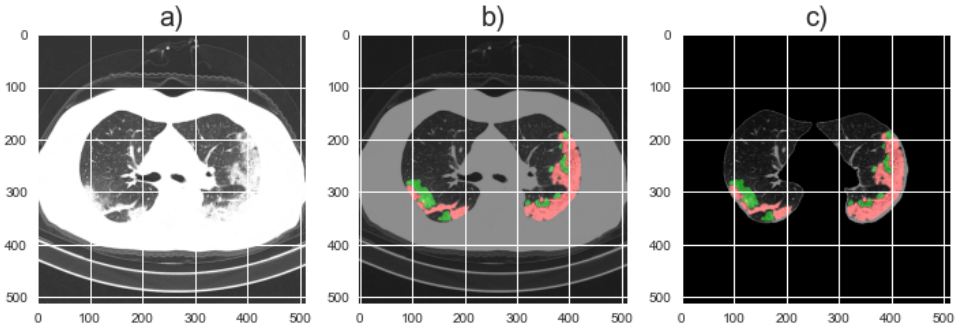
Figura 4.10 – Resultado de um exame somente com opacidade analisado pelo modelo *Unet++*. a) Imagem Original; b) Resultado informado pelo modelo sobre uma imagem; c) Região lesionada sobreposta à imagem.



Fonte: Arquivo pessoal do Autor.

Quando o paciente está em um estágio mais avançado e desenvolve lesões nas extremidades do parênquima pulmonar, a segmentação se torna mais complexa, pois a região de fronteira entre o órgão e o corpo pode ficar imperceptível. Observe na Figura 4.11.a) a região com tonalidade branca dentro da região pulmonar, se trata da lesão consolidação, que se confunde com a região externa, do corpo do paciente. Com uma quantidade suficientemente extensa de anotações, é possível fazer com que o modelo estime essa região de forma precisa, como é possível observar na Figura 4.11.b).

Figura 4.11 – Resultado de um exame (do conjunto de teste) somente com consolidação analisado pelo modelo *Unet++*. a) Imagem original, b) Imagem sobreposta pelos resultados de segmentação, e imagem



Fonte: Arquivo pessoal do Autor.

A fim de comparar a explicabilidade com a segmentação das lesões, foram calculadas as quantidades de pixels de ambos os resultados em relação à região dos parênquimas pulmonares. Para o exemplo da Figura 4.6.b), a região determinada pelo algoritmo de explicabilidade foi de $20,6\%$. Já a região determinada pelo resultado de segmentação da Figura 4.11.c) foi de $24,4\%$.

Após a divisão do banco de dados em 10 pastas e o treinamento dos modelos correspondentes, foram computadas as pontuações *dice* e *jaccard* nos conjuntos de testes. Os resultados podem ser observados conforme a Tabela 4.5.

Tabela 4.5 – Resultado comparativo dos modelos de identificação de lesão aplicados aos conjuntos de teste.

	Unet++ vs Deep Lab V3+			
	Consolidação		Opacidade em Vidro Fosco	
Modelo	Dice	Jaccard	Dice	Jaccard
UNET++	0,85 ± 0,13	0,75 ± 0,14	0,77 ± 0,11	0,65 ± 0,1
DeepLabV3+	0,83 ± 0,14	0,72 ± 0,14	0,77 ± 0,13	0,64 ± 0,11

Fonte: Arquivo Pessoal do Autor.

Pelo resultado obtido nos conjuntos de teste, as medidas de desempenho utilizadas apontam o quão bom o modelo consegue estimar as regiões anotadas. Logo, pode-se considerar que a qualidade das máscaras anotadas na base de lesões não possui tanta precisão quanto aquelas anotadas na base de parênquima pulmonar somente. Entretanto, de forma visual, os resultados conseguiram identificar corretamente as regiões, com maior assertividade que o modelo de explicabilidade.

Para resumir as etapas de treino, validação e teste nos dois tipos de lesões, os resultados foram dispostos nas tabelas 4.6 e 4.7.

Tabela 4.6 – Resultado comparativo dos modelos de identificação da lesão consolidação aplicados aos conjuntos de treino, validação e teste.

	Consolidação					
	Treino		Validação		Teste	
Modelo	Dice	Jaccard	Dice	Jaccard	Dice	Jaccard
Unet++	0,87 ± 0,06	0,75 ± 0,07	0,77 ± 0,1	0,63 ± 0,1	0,85 ± 0,13	0,75 ± 0,14
DeepLabV3+	0,90 ± 0,06	0,79 ± 0,07	0,80 ± 0,1	0,67 ± 0,1	0,83 ± 0,14	0,72 ± 0,14

Fonte: Arquivo Pessoal do Autor.

Tabela 4.7 – Resultado comparativo dos modelos de identificação da lesão opacidade em vidro fosco aplicados aos conjuntos de treino, validação e teste.

	Opacidade em Vidro Fosco					
	Treino		Validação		Teste	
Modelo	Dice	Jaccard	Dice	Jaccard	Dice	Jaccard
Unet++	0,87 ± 0,06	0,75 ± 0,07	0,77 ± 0,1	0,63 ± 0,1	0,77 ± 0,11	0,65 ± 0,10
DeepLabV3+	0,90 ± 0,06	0,79 ± 0,07	0,80 ± 0,1	0,67 ± 0,1	0,77 ± 0,13	0,64 ± 0,11

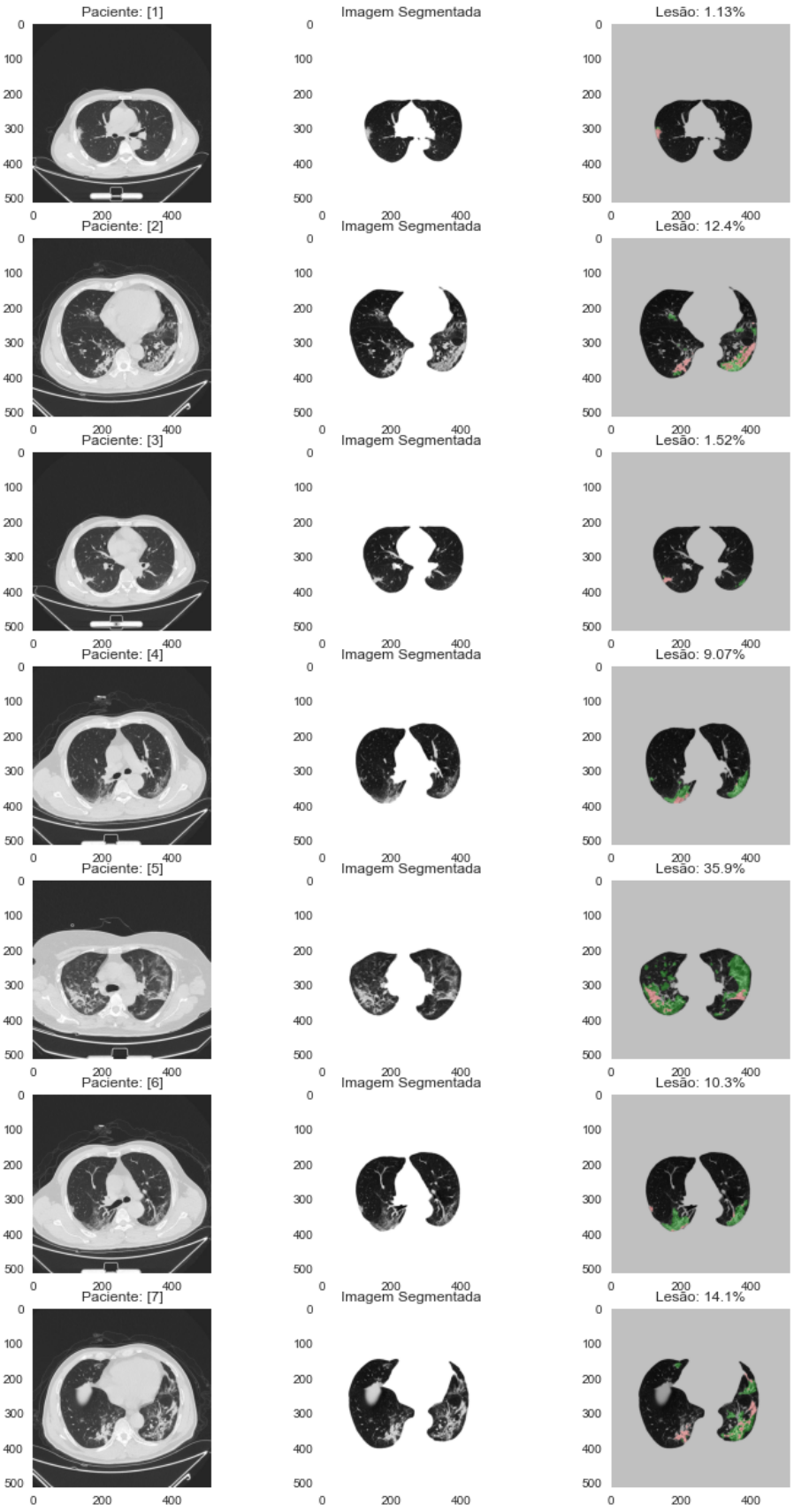
Fonte: Arquivo Pessoal do Autor.

Como pode ser observado, as medidas *Dice* e *Jaccard* para o treino foram as mesmas em ambas as tabelas, isto ocorre porque durante o treinamento é monitorado o valor médio das pontuações. Apesar de apresentar pontuações inferiores em relação aos modelos de segmentação dos parênquimas pulmonares, os resultados visuais para os modelos de identificação de lesões conseguem encontrar e destingir com uma qualidade razoável, segundo especialistas. Em posse das máscaras dos parênquimas pulmonares e das máscaras das respectivas lesões presentes, é possível estimar o percentual de lesão no paciente.

4.3.1 Severidade de lesão para Triagem

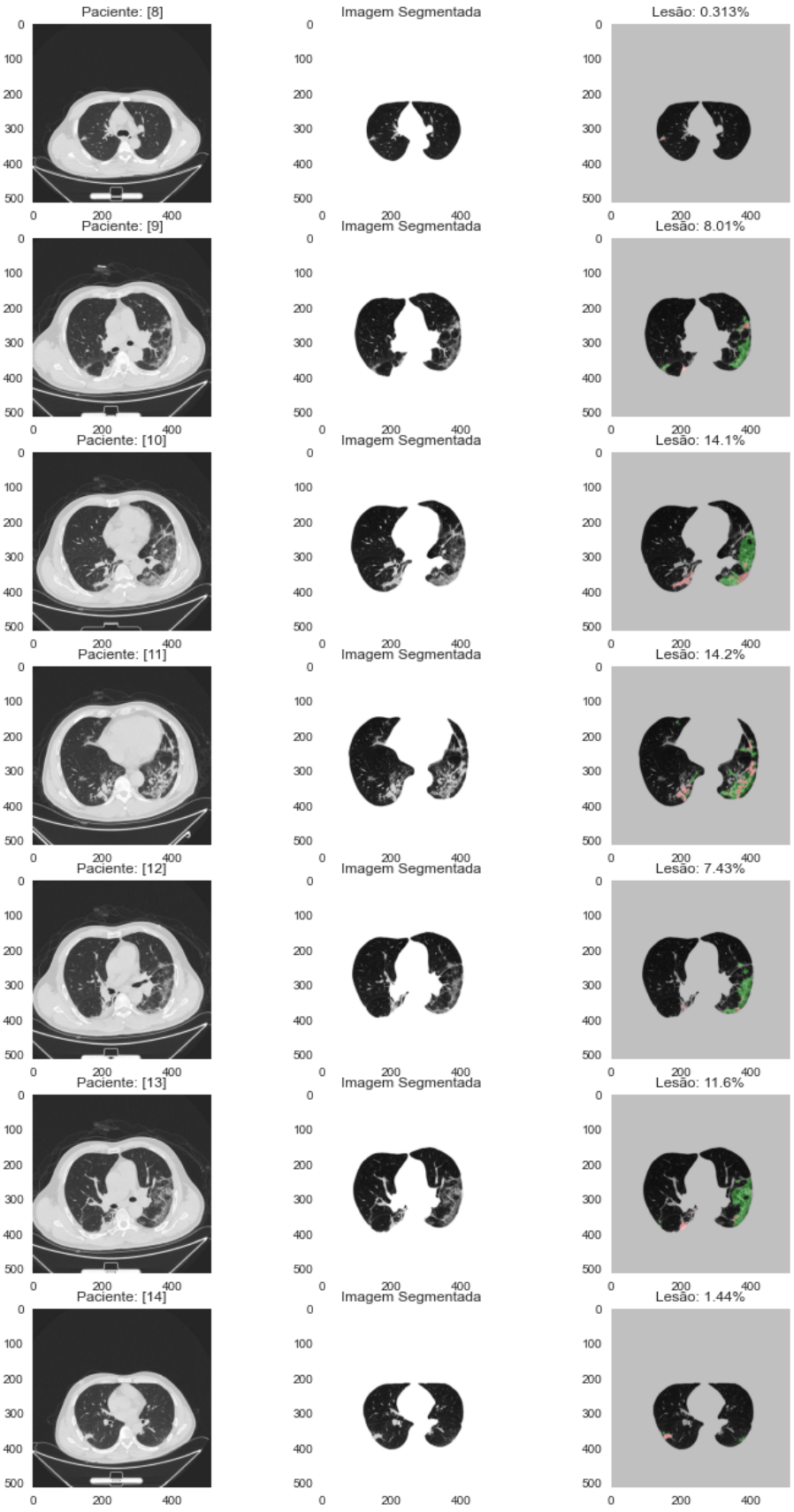
Como etapa adicional ao suporte na tomada de decisão e triagem, foi realizado um estudo da taxa de lesão por *slice* de diversos pacientes do conjunto de teste, apresentando a taxa por cada tipo de lesão e a taxa superficial total afetada. Esta taxa corresponde à razão entre a quantidade de pixels demarcados pelo modelo de segmentação pulmonar, pela quantidade de pixels demarcados pelo modelo de segmentação de lesão, ou seja, à razão da "área" lesionada pela "área" pulmonar, observe as Figuras 4.12 e 4.13. Cada conjunto de imagens corresponde a análise da fatia de um paciente, na seguinte ordem, imagem original, imagem sobreposta pela lesão e imagem com pulmão segmentado sobreposto pela lesão. A terceira imagem proporciona a severidade de lesão do paciente, calculada pela razão entre a quantidade de pixels de cada lesão, sobre a quantidade de pixels do pulmão, determinando uma área relativa que esta acometida por lesão no paciente.

Figura 4.12 – Análise de infecção por paciente, cada 3 imagens correspondem a um *slice* de um paciente.



Fonte: Arquivo pessoal do Autor.

Figura 4.13 – Continuação: Análise de infecção por paciente, cada 3 imagens correspondem a um *slice* de um paciente.



Fonte: Arquivo pessoal do Autor.

A razão entre os pixels das lesões em cada *slice* de cada paciente, informa ao profissional de saúde o nível aproximado da severidade do paciente. Podendo expor algum detalhe que passe desapercebido pelo radiologista. Além de calcular rapidamente o nível de lesão que o paciente possui, sem a necessidade de uma análise minuciosa. Na tabela 4.8 são informados os valores detalhados de lesão para cada uma das fatias na Figura 4.12.

Tabela 4.8 – Resultado percentual de área lesionada do primeiro modelo UNET++ aplicado a 14 pacientes do conjunto de teste 1.

Paciente	Opacidade em Vidro Fosco	Consolidação	Total
1	0.18	0.95	1.13
2	6.26	6.09	12.35
3	0.74	0.78	1.52
4	7.79	1.28	9.07
5	30.67	5.21	35.88
6	9.42	0.85	10.27
7	9.36	4.71	14.07
8	0.05	0.26	0.31
9	6.91	1.10	8.01
10	10.66	3.42	14.08
11	8.95	5.27	14.22
12	6.73	0.69	7.42
13	10.32	1.27	11.59
14	0.62	0.83	1.45

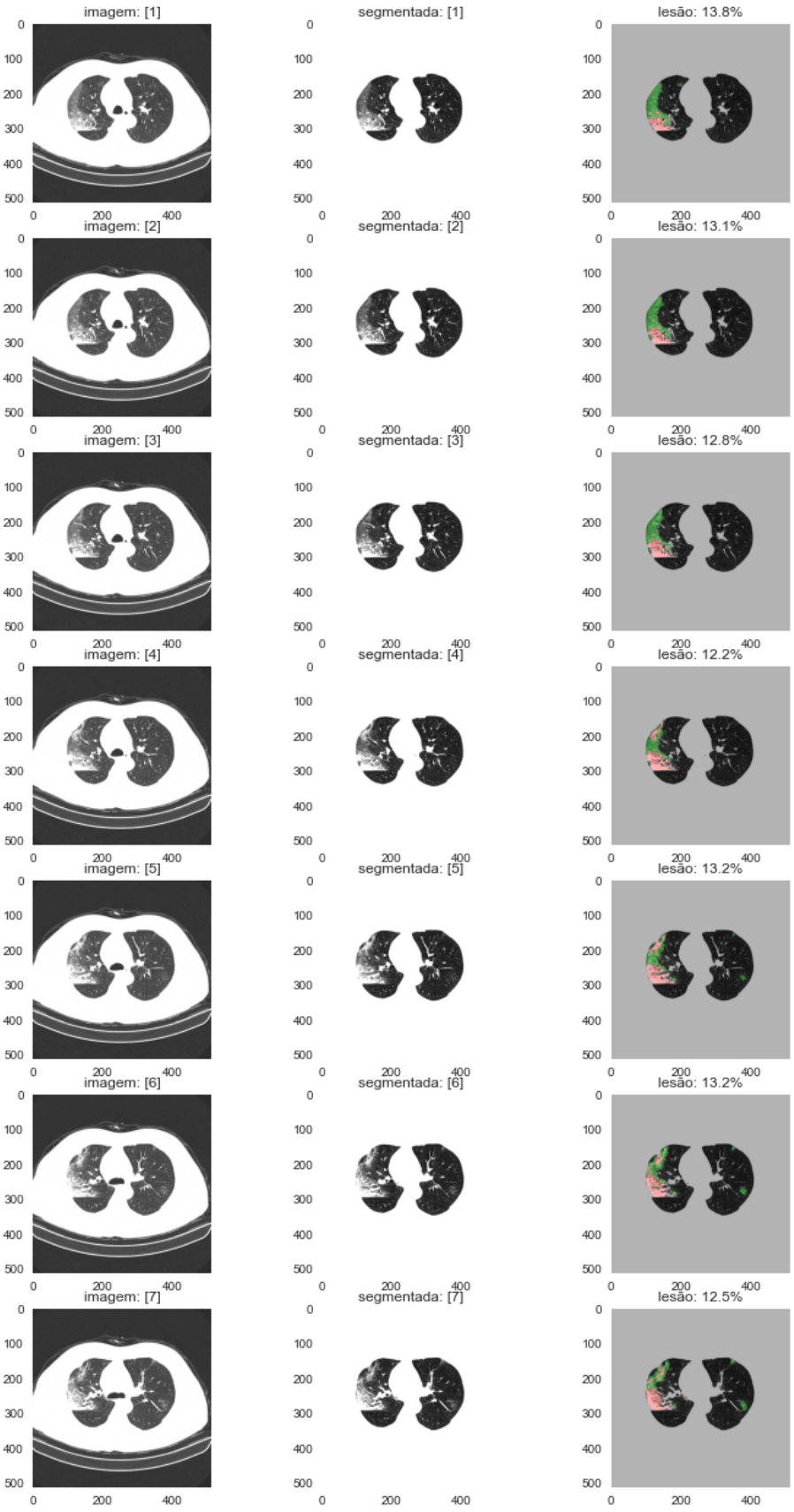
Fonte: Arquivo Pessoal do Autor.

Porém, a análise superficial (por imagem) pode não ser o suficiente para auxiliar na tomada de decisão sobre o paciente para que a triagem seja efetuada, uma vez que não informa o estado geral do paciente. Para realizar uma análise geral sobre o paciente, é necessário que todas as fatias (ou pelo menos a maior parte delas) estejam disponíveis. A partir da segmentação individual de cada fatia, pode-se somar todos os pixels referentes a lesões e dividir pelo número de pixels correspondentes ao pulmão em todas as fatias. Todas as imagens da Figura 4.14 são de um paciente externo ao conjunto de treino, teste e validação utilizados. Este paciente possui um total de 72 fatias[1] selecionadas por especialista, ou seja, o exame não está completo, mas possui grande parte das informações.

Neste caso, cada imagem corresponde a uma fatia do exame de um paciente, externo ao banco de dados utilizado no treinamento. Primeiramente tem-se a análise superficial por cada fatia do exame, resultado disposto na tabela 4.9. Nela foram computados somente a razão entre a quantidade de pixels na lesão pela quantidade de pixels nos pulmões. A

[1] O restante das Figuras podem ser observadas no apêndice C.

Figura 4.14 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



Fonte: Arquivo pessoal do Autor.

estimativa volumétrica pode ser observada na tabela 4.10, somente pelas imagens pode ser dispendioso analisar vários pacientes, mas com este método, pode-se estimar rapidamente o nível de lesão geral do paciente, além de permitir a comparação entre pacientes, uma vez que a taxa de pixels é calculada em relação à porção de lesão dentro da região pulmonar.

Tabela 4.9 – Resultado percentual de área lesionada do modelo UNET++ aplicado a um paciente externo aos conjuntos de treino e teste.

Imagem	Opacidade em Vidro Fosco	Consolidação	Total
1	9.14	4.67	13.81
2	8.44	4.64	13.09
3	7.71	5.10	12.81
4	5.18	7.02	12.21
5	6.01	7.16	13.17
6	6.41	6.84	13.24
7	5.71	6.79	12.49
8	5.52	6.64	12.16
9	6.64	6.98	13.61
10	7.92	5.73	13.65
11	7.41	4.57	11.97
12	7.06	3.00	10.06
13	5.30	1.94	7.24
14	5.32	1.33	6.65
15	6.23	1.25	7.48
16	7.06	1.30	8.36
17	7.12	0.88	8.00
18	5.71	0.61	6.32
19	4.23	0.53	4.76
20	3.04	0.28	3.33
21	2.58	0.11	2.69
22	2.69	0.05	2.75
23	0.56	0.00	0.56
24	0.08	0.00	0.08

25	0.40	0.02	0.42
26	1.08	0.06	1.14
27	1.35	0.08	1.42
28	3.15	0.03	3.18
29	3.41	0.18	3.59
30	5.99	0.09	6.09
31	7.94	0.13	8.07
32	8.64	0.01	8.65
33	9.63	0.09	9.72
34	10.12	0.31	10.43
35	8.56	0.96	9.52
36	9.02	1.09	10.11
37	7.83	1.20	9.03
38	7.62	0.91	8.53
39	8.21	0.71	8.92
40	6.43	0.75	7.17
41	3.56	0.79	4.36
42	1.84	1.02	2.86
43	1.87	1.55	3.42
44	1.41	1.87	3.28
45	0.77	1.71	2.48
46	1.69	1.67	3.36
47	3.00	1.76	4.77
48	2.74	0.53	3.26
49	4.55	0.17	4.72
50	6.56	0.31	6.86
51	6.14	0.27	6.41
52	9.25	4.74	13.99
53	7.37	6.29	13.67
54	7.94	6.71	14.65
55	8.07	6.95	15.03

56	9.86	4.63	14.50
57	6.38	1.62	8.00
58	9.18	0.97	10.15
59	6.89	0.59	7.48
60	3.84	0.15	3.99
61	0.89	0.00	0.89
62	1.05	0.07	1.12
63	4.47	0.03	4.50
64	9.37	0.01	9.38
65	10.81	0.07	10.88
66	9.71	1.25	10.96
67	9.38	0.88	10.26
68	7.69	1.00	8.70
69	2.46	2.02	4.48
70	4.44	1.97	6.42
71	5.90	0.40	6.30
72	9.53	0.25	9.77

Fonte: Arquivo Pessoal do Autor.

Tabela 4.10 – Taxa de Lesão Volumétrica do paciente NCP 215.

Taxa Volumétrica de Consolidação (%)	Taxa Volumétrica de Opacidade (%)	Taxa Volumétrica Lesão (%)
1.764478249598692	5.650904314186865	7.415382563785557

Fonte: Arquivo Pessoal do Autor.

5 CONCLUSÃO E PROPOSTAS DE CONTINUIDADE

A pandemia gerada pela disseminação do SARS-COV-2 provocou um colapso nos sistemas de saúde de diversos países ao redor do mundo, aumentando a carga de pacientes hospitalizados com necessidade de respiradores artificiais e profissionais preparados para lidar rapidamente com a situação. Toda essa crise provocada pelo vírus, gerou uma demanda por forma eficazes e/ou automáticas de lidar com procedimentos médicos, a fim de evitar o contato entre os pacientes e profissionais de saúde e contaminar mais pessoas, sobrecarregando ainda mais o sistema de saúde. Neste contexto, foi desenvolvido a proposta deste trabalho, baseada nos principais estudos e pesquisas encontradas desde o princípio da pandemia. Partindo desde o pré-processamento, passando pela classificação, explicabilidade, até a identificação das lesões comumente encontradas nos pacientes diagnosticados com COVID-19. Entende-se que aplicações médicas exigem uma entrega de resultados com consistência e que sejam explicáveis. As bases de dados criadas podem auxiliar na geração de novos estudos, sendo uma proposta preliminar que necessita de novos avanços em busca de aprimoramentos, tanto no quesito de diferenciabilidade de lesões quanto na quantidade de lesões identificáveis. O sistema como um todo, consegue classificar com precisão pacientes infectados, além de fornecer informações auxiliares na triagem.

5.0.1 Propostas de Continuidade

Após a conclusão dos modelos e análise dos resultados obtidos, algumas possíveis melhorias foram observadas, como por exemplo:

5.0.2 Pré-processamento Por Lesões

Neste trabalho, foi realizado um pré-processamento baseado na segmentação dos parênquimas pulmonares, selecionando as imagens com maior taxa de pulmão sobre a imagem como um todo, para que o modelo não fosse submetido a imagens sem o parênquima pulmonar. Porém, somente este requisito não garante que imagens sem lesões sejam selecionadas. Logo, o modelo de identificação de lesões poderia ser utilizado para selecionar imagens com lesões, e assim garantir que o modelo seja treinado somente com imagens de pacientes que desenvolveram lesões.

5.0.3 Adicionar Novas Classes

Durante as etapas iniciais do desenvolvimento deste projeto, foram treinados modelos *Vision Transformers* para classificação binária entre imagens de pacientes com COVID-19 e imagens de pacientes com Pneumonia de outras origens, tais resultados não foram satisfatórios, possivelmente devido às lesões geradas pela COVID-19 serem não patognomônicas (como mencionado na seção 2.2.1.1). Para solucionar ou mitigar esta dificuldade em diferenciar COVID-19 com Pneumonia de outras origens, podemos listar duas melhorias. Primeiramente, poderiam ser treinados modelos para identificar a região na qual as lesões se encontram, pois a COVID-19 possui regiões com maior incidência do que outras doenças. Além disso adicionar mais tipos de lesões ao pré-processamento por lesões ajudaria a identificar não somente lesões geradas pelo SARS-COV-2, como também as lesões provenientes de outras doenças.

5.0.4 Localização de Lesões

Além da identificação das lesões, e da contabilização da área total infectada, também é possível identificar a posição que se encontra a lesão, até mesmo de forma volumétrica, como mostrado nos trabalhos Carvalho *et al.* (2021), Afshar *et al.* (2021). A posição auxilia o médico a identificar o estágio em que o paciente se encontra. Realizar tal procedimento de forma automática poderia facilitar a rapidez nas tomadas de decisões.

Em novas propostas, espera-se que seja possível a realização de interfaces amigáveis e de fácil utilização para que os médicos possam analisar e conferir estes resultados.

5.1 Produção Científica

Durante o desenvolvimento deste projeto foram desenvolvidos e publicados os seguintes trabalhos:

i) Título: Unsupervised Class-Expert Learning For Supporting Covid-19 Triage Based On Computed Tomography Data.
Journal Learning and Nonlinear Models (LNLM).
Autores: Taís Aparecida Alvarenga, Luís Otávio Santos, Demóstenes Zegarra Rodríguez, Danton Diego Ferreira, Bruno Henrique Groenner Barbosa, José Manoel de Seixas;

ii) Título: Algoritmos Não-Supervisionados para Suporte à Tomada de Decisão e Diagnóstico de Covid-19 usando Dados de Tomografia Computadorizada.
Congresso Brasileiro de Inteligência Computacional – 2021.
Autores: Taís Aparecida Alvarenga, Luís Otávio Santos, Demóstenes Zegarra Rodríguez, Danton Diego Ferreira, Bruno Henrique Groenner Barbosa, José Manoel de Seixas;
DOI: 10.21528/CBIC2021-121;

iii) Título: Implementação em hardware de um Sistema de Reamostragem Dinâmica Coerente.
Sociedade Brasileira de Automática – 2020.
Autores: Luís Otávio Santos, Marcelo Antônio Alves Lima, Leandro Rodrigues Manso Silva, Henrique Luis Moreira Monteiro, Carlos Augusto Duque;
DOI: https://doi.org/10.48011/asba.v2i1.1533.

REFERÊNCIAS

ABADI, M. *et al.* **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Acesso em: 10 jan. 2022.

AFSHAR, P. *et al.* Covid-ct-md, covid-19 computed tomography scan dataset applicable in machine learning and deep learning. **Scientific Data**, v. 8, n. 1, p. 121, Apr 2021. ISSN 2052-4463. Disponível em: <https://doi.org/10.1038/s41597-021-00900-3>.

AI, T. *et al.* Correlation of chest ct and rt-pcr testing for coronavirus disease 2019 (covid-19) in china: A report of 1014 cases. **Radiology**, v. 296, n. 2, p. E32–E40, 2020. PMID: 32101510. Disponível em: <https://doi.org/10.1148/radiol.2020200642>.

ALLAM, H. H. *et al.* Pericardial fluid in a COVID-19 patient: Is it exudate or transudate? **Eur. J. Case Rep. Intern. Med.**, SMC Media, v. 7, n. 6, p. 001703, maio 2020.

ANDERSEN, K. G. *et al.* The proximal origin of SARS-CoV-2. **Nature Medicine**, v. 26, n. 4, p. 450–452, 2020. ISSN 1546170X.

Anil Ananthaswamy. **Artificial Neural Nets Finally Yield Clues to How Brains Learn | Quanta Magazine**. 2021. Disponível em: <https://www.quantamagazine.org/artificial-neural-nets-finally-yield-clues-to-how-brains-learn-20210218/>. Acesso em: 27 mar. 2022.

ARBEL, Y.; MURPHY, A.; DONCHIN, E. On the Utility of Positive and Negative Feedback in a Paired-associate Learning Task. **Journal of Cognitive Neuroscience**, v. 26, n. 7, p. 1445–1453, 07 2014. ISSN 0898-929X. Disponível em: <https://doi.org/10.1162/jocn_a_00617>.

ASIM, M.; AHMED, A.; HAND, P. Invertible generative models for inverse problems: mitigating representation error and dataset bias. **CoRR**, abs/1905.11672, 2019. Disponível em: <http://arxiv.org/abs/1905.11672>.

AYYADEVARA, V.; REDDY, Y. **Modern Computer Vision with PyTorch: Explore deep learning concepts and implement over 50 real-world image applications**. Packt Publishing, 2020. ISBN 9781839216534. Disponível em: <https://books.google.com.br/books?id=GfILEAAAQBAJ>.

BAHDANAU, D.; CHO, K. H.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, International Conference on Learning Representations, ICLR, sep 2014. Disponível em: <https://arxiv.org/abs/1409.0473v7>.

BAI, L.; ZHAO, Y.; HUANG, X. A CNN Accelerator on FPGA Using Depthwise Separable Convolution. **IEEE Transactions on Circuits and Systems II: Express Briefs**, IEEE, v. 65, n. 10, p. 1415–1419, 2018. ISSN 15497747.

BANKMAN, I. **Handbook of Medical Image Processing and Analysis**. Elsevier Science, 2008. 73–85 p. (Academic Press series in biomedical engineering). ISBN 9780080559148. Disponível em: <https://books.google.com.br/books?id=AnRPBKb7qHUC>.

BENTLEY, E. G. *et al.* SARS-CoV-2 Omicron-B.1.1.529 Variant leads to less severe disease than Pango B and Delta variants strains in a mouse model of severe COVID-19. **bioRxiv**, Cold Spring Harbor Laboratory, p. 2021.12.26.474085, dec 2021.

Disponível em: <https://www.biorxiv.org/content/10.1101/2021.12.26.474085v1https://www.biorxiv.org/content/10.1101/2021.12.26.474085v1.abstract>.

BHALLA, A. S. *et al.* Role of Chest Radiographs during COVID-19 Pandemic. **Annals of the National Academy of Medical Sciences (India)**, Georg Thieme Verlag KG, v. 56, n. 03, p. 138–144, jul 2020. ISSN 0379-038X. Disponível em: <http://www.thieme-connect.de/DOI/DOI?10.1055/s-0040-1714158>.

BISHOP, C. **Pattern Recognition and Machine Learning**. Springer New York, 2016. (Information Science and Statistics). ISBN 9781493938438. Disponível em: <https://books.google.com.br/books?id=kOXDtAEACAAJ>.

BOHNSLAV, J. **opencv-transforms · PyPI**. 2019. Disponível em: <https://pypi.org/project/opencv-transforms/>. Acesso em: 10 ago. 2021.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BRÉBISSON, A. de; VINCENT, P. An Exploration of Softmax Alternatives Belonging to the Spherical Loss Family. **4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings**, International Conference on Learning Representations, ICLR, nov 2015. Disponível em: <http://arxiv.org/abs/1511.05042>.

BRIDGE, J. *et al.* Introducing the GEV Activation Function for Highly Unbalanced Data to Develop COVID-19 Diagnostic Models. **IEEE Journal of Biomedical and Health Informatics**, v. 24, n. 10, p. 2776–2786, 2020. ISSN 21682208.

BUSLAEV, A. *et al.* Fully convolutional network for automatic road extraction from satellite imagery. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**. [S.l.: s.n.], 2018.

CAROTTI, M. *et al.* Chest CT features of coronavirus disease 2019 (COVID-19) pneumonia: key points for radiologists. **La Radiologia Medica**, Nature Publishing Group, v. 125, n. 7, p. 1, jul 2020. Disponível em: </pmc/articles/PMC7270744//pmc/articles/PMC7270744/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC7270744/>.

CARVALHO, A. R. S. *et al.* Estimating covid-19 pneumonia extent and severity from chest computed tomography. **Frontiers in Physiology**, v. 12, 2021. ISSN 1664-042X. Disponível em: <https://www.frontiersin.org/article/10.3389/fphys.2021.617657>.

CHAMOLA, V. *et al.* A Comprehensive Review of the COVID-19 Pandemic and the Role of IoT, Drones, AI, Blockchain, and 5G in Managing its Impact. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 90225–90265, 2020. ISSN 21693536.

CHAPLE, G. N.; DARUWALA, R.; GOFANE, M. S. Comparisions of robert, prewitt, sobel operator based edge detection methods for real time uses on fpga. In: IEEE. **2015 International Conference on Technologies for Sustainable Development (ICTSD)**. [S.l.], 2015. p. 1–4.

CHEFER, H.; GUR, S.; WOLF, L. Transformer Interpretability Beyond Attention Visualization. dec 2020. Disponível em: <https://arxiv.org/abs/2012.09838v2>.

CHEFER, H.; GUR, S.; WOLF, L. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers. mar 2021. Disponível em: <https://arxiv.org/abs/2103.15679v1>.

CHEN, J. **COVID19 CT Segmentation 3D Slicer**. 2020. Disponível em: <https://github.com/junyuchen245/COVID19_CT_Segmentation_3DSlicer>. Acesso em: 08 fev. 2022.

CHEN, J. *et al.* TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. p. 1–13, 2021. Disponível em: <http://arxiv.org/abs/2102.04306>.

CHEN, L.-C. *et al.* Rethinking Atrous Convolution for Semantic Image Segmentation. jun 2017. Disponível em: <https://arxiv.org/abs/1706.05587v3>.

CHEN, L. C. *et al.* Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, Springer Verlag, v. 11211 LNCS, p. 833–851, feb 2018. ISSN 16113349. Disponível em: <https://arxiv.org/abs/1802.02611v3>.

CONNECT, V. **Vital Patch Device Description Indications for Use**. 2019. 13 p. Disponível em: <https://vitalconnect.com/docs/ifu010/rev09-production/IFU-10_Rev09_VitalPatch_2.0_Instructions_for_Use.pdf>. Acesso em: 28 jul. 2021.

DAVID, C.; De Abajo, F. J. G. **Nonlocal effects in the optical response of metal nanoparticles**. [S.l.: s.n.], 2010. v. 1291. 43–45 p. ISSN 0094243X. ISBN 9780735408463.

DOSOVITSKIY, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. **CoRR**, abs/2010.11929, 2020. Disponível em: <https://arxiv.org/abs/2010.11929>.

DUTTA, A.; GUPTA, A.; ZISSERMANN, A. **VGG Image Annotator (VIA)**. 2016. Version: 2.0.1. Disponível em: <https://www.robots.ox.ac.uk/~vgg/software/via/via.html>. Acesso em: 12 fev. 2022.

DUTTA, A.; ZISSERMAN, A. The VIA annotation software for images, audio and video. In: **Proceedings of the 27th ACM International Conference on Multimedia**. New York, NY, USA: ACM, 2019. (MM '19). ISBN 978-1-4503-6889-6/19/10. Disponível em: <https://doi.org/10.1145/3343031.3350535>.

El Hassani, A.; SKOURT, B. A.; MAJDA, A. Efficient Lung CT Image Segmentation using Mathematical Morphology and the Region Growing algorithm. **Proceedings - 2019 International Conference on Intelligent Systems and Advanced Computing Sciences, ISACS 2019**, IEEE, n. 1, p. 0–5, 2019.

ELGENDY, M. **Deep Learning for Vision Systems**. Manning Publications, 2020. ISBN 9781617296192. Disponível em: <https://books.google.com.br/books?id=97YCEAAAQBAJ>.

FAN, D. P. *et al.* Inf-Net: Automatic COVID-19 Lung Infection Segmentation from CT Images. **IEEE Transactions on Medical Imaging**, Institute of Electrical and Electronics Engineers Inc., v. 39, n. 8, p. 2626–2637, aug 2020. ISSN 1558254X.

FARHAT, H.; SAKR, G. E.; KILANY, R. Deep learning applications in pulmonary medical imaging: recent updates and insights on COVID-19. **Machine Vision and Applications**, Springer, v. 31, n. 6, sep 2020. ISSN 14321769. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/32834523/>.

FEDOROV, A. *et al.* 3d slicer as an image computing platform for the quantitative imaging network. **Magnetic Resonance Imaging**, v. 30, n. 9, p. 1323–1341, 2012. ISSN 0730-725X. Quantitative Imaging in Cancer. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0730725X12001816>.

FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: AMARI, S.-i.; ARBIB, M. A. (Ed.). **Competition and Cooperation in Neural Nets**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982. p. 267–285. ISBN 978-3-642-46466-9.

GARCIA, J. V. R. *et al.* COVID-19 on resonance magnetic: an incidental but important finding in times of pandemic. **Einstein (São Paulo)**, Instituto Israelita de Ensino e Pesquisa Albert Einstein, v. 18, p. eAI5891, nov 2020. ISSN 1679-4508. Disponível em: <http://www.scielo.br/j/eins/a/vDFRxtnbmZkQrqtRZQCGdyD/?lang=en>.

GARG, T. *et al.* Convolutional Neural Networks with Transfer Learning for Recognition of COVID-19: A Comparative Study of Different Approaches. **AI**, Multidisciplinary Digital Publishing Institute, v. 1, n. 4, p. 586–606, dec 2020. ISSN 2673-2688. Disponível em: <https://www.mdpi.com/2673-2688/1/4/34>.

GIRI, B. *et al.* Review of analytical performance of COVID-19 detection methods. **Analytical and Bioanalytical Chemistry**, Springer Science and Business Media Deutschland GmbH, v. 413, n. 1, p. 35–48, jan 2021. ISSN 16182650. Disponível em: <https://link.springer.com/article/10.1007/s00216-020-02889-x>.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256. Disponível em: <https://proceedings.mlr.press/v9/glorot10a.html>.

GOH, G. *et al.* Multimodal neurons in artificial neural networks. **Distill**, 2021. <https://distill.pub/2021/multimodal-neurons>.

GOMES, J. C. *et al.* Optimizing the molecular diagnosis of Covid-19 by combining RT-PCR and a pseudo-convolutional machine learning approach to characterize virus DNA sequences. **bioRxiv**, p. 2020.06.02.129775, 2020. Disponível em: <https://doi.org/10.1101/2020.06.02.129775>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GREENSHTEIN, E.; RITOV, Y. Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. **Bernoulli**, v. 10, n. 6, p. 971–988, 2004. ISSN 13507265.

GUNRAJ, H. *et al.* Covid-net ct-2: Enhanced deep neural networks for detection of covid-19 from chest ct images through bigger, more diverse learning. 2021. Disponível em: <https://www.kaggle.com/datasets/hgunraj/covidxct>.

GUNRAJ, H.; WANG, L.; WONG, A. Covidnet-ct: A tailored deep convolutional neural network design for detection of covid-19 cases from chest ct images. **Frontiers in Medicine**, v. 7, p. 1025, 2020. ISSN 2296-858X. Disponível em: <https://www.frontiersin.org/article/10.3389/fmed.2020.608525>.

HALOI, M. A novel pLSA based Traffic Signs Classification System. mar 2015. Disponível em: <http://arxiv.org/abs/1503.06643>.

HAN, K. *et al.* Transformer in transformer. In: RANZATO, M. *et al.* (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2021. v. 34, p. 15908–15919. Disponível em: <https://proceedings.neurips.cc/paper/2021/file/854d9fca60b4bd07f9bb215d59ef5561-Paper.pdf>.

HAND, D.; CHRISTEN, P. A note on using the F-measure for evaluating record linkage algorithms. **Statistics and Computing**, Springer New York LLC, v. 28, n. 3, p. 539–547, may 2018. ISSN 0960-3174. Disponível em: <http://link.springer.com/10.1007/s11222-017-9746-6>.

HANSELL, D. M. *et al.* Fleischner Society: Glossary of terms for thoracic imaging. **Radiology**, v. 246, n. 3, p. 697–722, 2008. ISSN 00338419.

HASTIE, T. Ridge Regularizaton: an Essential Concept in Data Science. **Technometrics**, American Statistical Association, v. 62, n. 4, p. 426–433, may 2020. Disponível em: <http://arxiv.org/abs/2006.00371>.

HE, K. *et al.* Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv, 2015. Disponível em: <https://arxiv.org/abs/1502.01852>. Acesso em: 06 mar. 2022.

HEMANTH, D. **Artificial Intelligence Techniques for Satellite Image Analysis**. Springer International Publishing, 2019. (Remote Sensing and Digital Image Processing). ISBN 9783030241780. Disponível em: <https://books.google.com.br/books?id=piy-DwAAQBAJ>.

HENDRYCKS, D.; GIMPEL, K. Gaussian Error Linear Units (GELUs). jun 2016. Disponível em: <http://arxiv.org/abs/1606.08415>.

HENRIQUES, J. F. *et al.* Small steps and giant leaps: Minimal newton solvers for deep learning. **CoRR**, abs/1805.08095, 2018. Disponível em: <http://arxiv.org/abs/1805.08095>.

HUANG, C. *et al.* Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. **The Lancet**, Lancet Publishing Group, v. 395, n. 10223, p. 497–506, feb 2020. ISSN 1474547X.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **The Journal of Physiology**, Wiley-Blackwell, v. 160, n. 1, p. 106, jan 1962. ISSN 14697793. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/>.

IMRAN, A. *et al.* AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app. **Informatics in Medicine Unlocked**, Elsevier Ltd, v. 20, p. 100378, jan 2020. ISSN 23529148.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. **32nd International Conference on Machine Learning, ICML 2015**, International Machine Learning Society (IMLS), v. 1, p. 448–456, feb 2015. Disponível em: <https://arxiv.org/abs/1502.03167v3>.

IZBICKI, R.; SANTOS, T. M. dos. **Aprendizado de máquina: uma abordagem estatística**. [S.l.: s.n.], 2020. ISBN 978-65-00-02410-4.

JACCARD, P. The distribution of the flora in the alpine zone.1. **New Phytologist**, v. 11, n. 2, p. 37–50, 1912. Disponível em: <https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x>.

JADON, S. A survey of loss functions for semantic segmentation. **2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020**, Institute of Electrical and Electronics Engineers Inc., oct 2020.

JIAO, C. *et al.* Burn image segmentation based on Mask Regions with Convolutional Neural Network deep learning framework: More accurate and more convenient. **Burns and Trauma**, Oxford University Press, v. 7, 2019. ISSN 23213876.

JOHNSTON, B.; MATHUR, I. **Applied Supervised Learning with Python: Use scikit-learn to build predictive models from real-world datasets and prepare yourself for the future of machine learning**. Packt Publishing, 2019. ISBN 9781789955835. Disponível em: <https://books.google.com.br/books?id=I_eVDwAAQBAJ>.

JUNG, A. **Imgaug ·**. 2020. Disponível em: <https://imgaug.readthedocs.io/en/latest/>. Acesso em: 10 fev. 2022.

KAUR, S. *et al.* Understanding COVID-19 transmission, health impacts and mitigation: timely social distancing is the key. **Environment, Development and Sustainability**, Springer, p. 1–17, jul 2020. ISSN 15732975. Disponível em: <https://doi.org/10.1007/s10668-020-00884-x>.

KHALILI, N.; HASELI, S.; IRANPOUR, P. Lung Ultrasound in COVID-19 Pneumonia: Prospects and Limitations. Elsevier USA, v. 27, n. 7, p. 1044–1045, jul 2020. ISSN 18784046. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/32444253/>.

KHUNDAQJI, H. *et al.* Smart shirts for monitoring physiological parameters: Scoping review. **JMIR mHealth and uHealth**, v. 8, n. 5, 2020. ISSN 22915222.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–15, 2015.

KLAMBAUER, G. *et al.* Self-normalizing neural networks. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1706.02515>.

KOSUB, S. A note on the triangle inequality for the Jaccard distance. **Pattern Recognition Letters**, Elsevier B.V., v. 120, p. 36–38, 2019. ISSN 01678655. Disponível em: <https://doi.org/10.1016/j.patrec.2018.12.007>.

KROON, D.-J. **Region growing**. MathWorks, 2008. Disponível em: <https://www.mathworks.com/matlabcentral/fileexchange/19084-region-growing>. Acesso em: 12 fev. 2022.

LANDRO, N.; GALLO, I.; GRASSA, R. L. Mixing ADAM and SGD: a Combined Optimization Method. 2020. Disponível em: <https://gitlab.com/nicolalandro/multi>.

LECUN, Y. *et al.* Efficient BackProp. Neural Networks: Tricks of the Trade. **Springer**, 1998.

LEONILDO COSTA E SILVA. Aprendizado De Máquina Com Treinamento Continuado Aplicado À Previsão De Demanda De Curto Prazo: O Caso Do Restaurante Universitário Da Universidade Federal De Uberlândia. **Mestardo EE-Universidade Federal de Uberlândia**, p. 132, 2019.

LI, K. *et al.* The Clinical and Chest CT Features Associated With Severe and Critical COVID-19 Pneumonia. **Investigative Radiology**, Wolters Kluwer Health, v. 55, n. 6, p. 327–331, jun 2020. Disponível em: </pmc/articles/PMC7147273//pmc/articles/PMC7147273/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC7147273/>.

LINNAINMAA, S. Taylor expansion of the accumulated rounding error. **Bit**, v. 16, n. 2, p. 146–160, 1976. ISSN 15729125.

LIU, Z. *et al.* Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. mar 2021. Disponível em: <http://arxiv.org/abs/2103.14030>.

LOSHCHILOV, I.; HUTTER, F. Decoupled Weight Decay Regularization. **7th International Conference on Learning Representations, ICLR 2019**, International Conference on Learning Representations, ICLR, nov 2017. Disponível em: <https://arxiv.org/abs/1711.05101v3>.

LOSHCHILOV, I.; HUTTER, F. Fixing weight decay regularization in adam. 2018. Disponível em: <https://openreview.net/forum?id=rk6qdGgCZ>. Acesso em: 17 jun. 2021.

LU, L. *et al.* Dying ReLU and Initialization: Theory and Numerical Examples. **Communications in Computational Physics**, Global Science Press, v. 28, n. 5, p. 1671–1706, mar 2019. Disponível em: <http://arxiv.org/abs/1903.06733http://dx.doi.org/10.4208/cicp.OA-2020-0165>.

LUO, L. *et al.* Adaptive Gradient Methods with Dynamic Bound of Learning Rate. **arXiv**, arXiv, feb 2019. Disponível em: <http://arxiv.org/abs/1902.09843>.

MAHMUD, T. *et al.* CovTANet : A Hybrid Tri-level Attention Based Network for Lesion Segmentation , Diagnosis , and Severity Prediction of COVID-19 Chest CT Scans. v. 3203, n. c, p. 1–10, 2020.

MAJEED, T. *et al.* Covid-19 detection using CNN transfer learning from X-ray Images. 2020. Disponível em: <https://doi.org/10.1101/2020.05.12.20098954>.

MATOS, M. J. R. de *et al.* Differential diagnoses of acute ground-glass opacity in chest computed tomography: pictorial essay. **Einstein (São Paulo)**, Instituto Israelita de Ensino e Pesquisa Albert Einstein, v. 19, p. eRW5772, mar 2021. ISSN 1679-4508. Disponível em: <http://www.scielo.br/j/eins/a/63XjZvQsqzBTZDfmmnYVQ4q/>.

MEYER, F. Color image segmentation. In: IET. **1992 international conference on image processing and its applications**. [S.l.], 1992. p. 303–306.

MIKOŁAJCZYK, A.; GROCHOWSKI, M. Data augmentation for improving deep learning in image classification problem. **2019 International Interdisciplinary PhD Workshop, IIPhDW 2019**, IEEE, p. 117–122, 2019.

MILLETARI, F.; NAVAB, N.; AHMADI, S. A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. **Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016**, Institute of Electrical and Electronics Engineers Inc., p. 565–571, jun 2016. Disponível em: <https://arxiv.org/abs/1606.04797v1>.

MINAEE, S.; ABDOLRASHIDI, A. Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network. **arXiv**, arXiv, feb 2019. Disponível em: <http://arxiv.org/abs/1902.01019>.

MIRA, J. M. Symbols versus connections: 50 years of artificial intelligence. **Neurocomputing**, v. 71, n. 4-6, p. 671–680, 2008. ISSN 09252312.

MIT. **Marvin Minsky's Home Page**. 1951. Disponível em: <https://web.media.mit.edu/$\sim$minsky/>. Acesso em: 28 mar. 2022.

MONDAL, A. K. *et al.* xViTCOS : Explainable Vision Transformer Based COVID-19 Screening Using Radiography. p. 0–13, 2021.

MOOCARME, M.; ABDOLAHNEJAD, M.; BHAGWAT, R. **The Deep Learning with Keras Workshop: Learn how to define and train neural network models with just a few lines of code**. Packt Publishing, 2020. ISBN 9781800564756. Disponível em: <https://books.google.com.br/books?id=XxL0DwAAQBAJ>.

NAIDICH, D. *et al.* **Computed Tomography and Magnetic Resonance of the Thorax**. Wolters Kluwer/Lippincott Williams & Wilkins, 2007. ISBN 9780781757652. Disponível em: <https://books.google.com.br/books?id=zOiDtdtyOQEC>.

NAJMAN, L.; SCHMITT, M. Watershed of a Continuous Function. **Signal Processing**, Elsevier, v. 38, n. 1, p. 99–112, 1994. Special issue on Mathematical Morphology. Disponível em: <https://hal-upec-upem.archives-ouvertes.fr/hal-00622129>.

NASAJPOUR, M. *et al.* Internet of Things for Current COVID-19 and Future Pandemics: an Exploratory Study. **Journal of Healthcare Informatics Research**, Springer Science and Business Media Deutschland GmbH, v. 4, n. 4, p. 325–364, dec 2020. ISSN 2509498X.

NIU, Z.; ZHONG, G.; YU, H. A review on the attention mechanism of deep learning. **Neurocomputing**, Elsevier B.V., v. 452, p. 48–62, 2021. ISSN 18728286. Disponível em: <https://doi.org/10.1016/j.neucom.2021.03.091>.

ORMROD, J. **How We Think and Learn: Theoretical Perspectives and Practical Implications**. Cambridge University Press, 2017. ISBN 9781107165113. Disponível em: <https://books.google.com.br/books?id=DITuDQAAQBAJ>.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 9, n. 1, p. 62–66, 1979.

OTTAVIANI, S. *et al.* Lung ultrasonography in patients with COVID-19: comparison with CT. **Clinical Radiology**, W.B. Saunders Ltd, v. 75, n. 11, p. 877.e1–877.e6, nov 2020. ISSN 1365229X. Disponível em: <https://doi.org/10.1016/j.crad.2020.07.024>.

PAREKH, M. *et al.* Review of the Chest CT Differential Diagnosis of Ground-Glass Opacities in the COVID Era. **https://doi.org/10.1148/radiol.2020202504**, Radiological Society of North America, v. 297, n. 3, p. E289–E302, jul 2020. Disponível em: <https://pubs.rsna.org/doi/abs/10.1148/radiol.2020202504>.

PASZKE, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In: WALLACH, H. *et al.* (Ed.). **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

PETERS, J. **Foundations of Computer Vision: Computational Geometry, Visual Image Structures and Object Shape Detection**. Springer International Publishing, 2018. (Intelligent Systems Reference Library). ISBN 9783319849126. Disponível em: <https://books.google.com.br/books?id=oHaCuQEACAAJ>.

PINKER, J. *et al.* **Connections and Symbols**. Elsevier, 1988. (A Bradford book). ISBN 9780262660648. Disponível em: <https://books.google.com.br/books?id=vZ1t2AnZy3UC>.

PLANCHE, B.; ANDRES, E. **Hands-On Computer Vision with TensorFlow 2**. [S.l.: s.n.], 2019. 1–306 p. ISBN 9781788830645.

PLANCHE, B.; ANDRES, E. **Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras**. Packt Publishing, 2019. ISBN 9781788839266. Disponível em: <https://books.google.com.br/books?id=r_eaDwAAQBAJ>.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. arXiv, 2018. Disponível em: <https://arxiv.org/abs/1806.07908>.

RABI, F. A. *et al.* SARS-CoV-2 and Coronavirus Disease 2019: What We Know So Far. **Pathogens**, MDPI AG, v. 9, n. 3, p. 231, mar 2020. ISSN 2076-0817. Disponível em: <https://www.mdpi.com/2076-0817/9/3/231>.

Rachel Draelos. **Convolution vs. Cross-Correlation – Glass Box**. 2019. Disponível em: <https://glassboxmedicine.com/2019/07/26/convolution-vs-cross-correlation/>. Acesso em: 23 mar. 2022.

RAHIMZADEH, M.; ATTAR, A.; SAKHAEI, S. M. A fully automated deep learning-based network for detecting covid-19 from a new and large lung ct scan dataset. **medRxiv**, Cold Spring Harbor Laboratory Press, 2020. Disponível em: <https://www.medrxiv.org/content/early/2020/06/12/2020.06.08.20121541>.

RASCHKA, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. **arXiv**, arXiv, nov 2018. Disponível em: <http://arxiv.org/abs/1811.12808>.

ROBERTS, L. Machine perception of three-dimensional solids. 01 1963.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. [S.l.], 2015. Disponível em: <http://lmb.informatik.uni-freiburg.de/>.

ROSA, R. L. *et al.* Event Detection System Based on User Behavior Changes in Online Social Networks: Case of the COVID-19 Pandemic. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 158806–158825, aug 2020. ISSN 2169-3536.

ROSENBLATT, F. Principles of Neurodynamics. **Zhurnal Prikladnoy Mekhaniki i Tecnicheeskoy**, p. 626, 1962. Disponível em: <https://safari.ethz.ch/digitaltechnik/spring2019/lib/exe/fetch.php?media=neurodynamics1962rosenblatt.pdf>.

RUDER, S. An overview of gradient descent optimization algorithms. **CoRR**, abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>.

SÁNDOR, K. **CT Lung & Heart & Trachea segmentation | Kaggle**. 2020. Disponível em: <https://www.kaggle.com/sandorkonya/ct-lung-heart-trachea-segmentation>. Acesso em: 10 jun. 2022.

SANTURKAR, S. *et al.* How Does Batch Normalization Help Optimization? **Advances in Neural Information Processing Systems**, Neural information processing systems foundation, v. 2018-December, p. 2483–2493, may 2018. ISSN 10495258. Disponível em: <https://arxiv.org/abs/1805.11604v5>.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM Computing Surveys (CSUR)**, ACM PUB27 New York, NY, USA, v. 34, n. 1, p. 1–47, mar 2002. ISSN 03600300. Disponível em: <https://dl.acm.org/doi/abs/10.1145/505282.505283>.

SEGRELLES-CALVO, G.; De Saraújo, G. R.; FRASES, S. Systemic mycoses: A potential alert for complications in COVID-19 patients. **Future Microbiology**, v. 15, n. 14, p. 1405–1413, 2020. ISSN 17460921.

SELVARAJU, R. R. *et al.* **Grad-CAM: Why did you say that?** arXiv, 2016. Disponível em: <https://arxiv.org/abs/1611.07450>. Acesso em: 21 abr. 2022.

SHANG, Y. *et al.* Clinical characteristics and changes of chest CT features in 307 patients with common COVID-19 pneumonia infected SARS-CoV-2: A multicenter study in Jiangsu, China. **International Journal of Infectious Diseases**, Elsevier, v. 96, p. 157–162, jul 2020. ISSN 1201-9712. Disponível em: <http://www.ijidonline.com/article/S120197122030312X/fultext>.

SHEREEN, M. A. *et al.* COVID-19 infection: Origin, transmission, and characteristics of human coronaviruses. Elsevier B.V., v. 24, p. 91–98, jul 2020. ISSN 20901232.

SHI, F. *et al.* Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation and Diagnosis for COVID-19. **IEEE Reviews in Biomedical Engineering**, Institute of Electrical and Electronics Engineers, 2020. ISSN 19411189.

SHIH, F. Y. **Image processing and mathematical morphology: Fundamentals and applications**. [S.l.]: CRC press, 2009. 1–417 p. ISBN 9781420089448.

SHIH, M. L. *et al.* 3D Photography using Context-aware Layered Depth Inpainting. **arXiv**, 2020.

SNYDER, J.; FEDOROVSKAYA, E. **Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools, Second Edition, by Scott E. Umbaugh**. [S.l.: s.n.], 2011. v. 20. 039901 p. ISSN 10179909. ISBN 9781439802069.

SOBEL, I.; FELDMAN, G. An Isotropic 3x3 Image Gradient Operator. **Stanford Artificial Intelligence Project (SAIL)**, n. June, p. 271–272, 2015.

Soham Pal. **LBFGS vs Adam**. 2021. Disponível em: <https://soham.dev/posts/linear-regression-pytorch/>. Acesso em: 31 mar. 2022.

SRIVASTAVA, N. *et al.* **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. [S.l.], 2014. v. 15, 1929–1958 p.

SUZUKI, S. *et al.* Topological structural analysis of digitized binary images by border following. **Computer vision, graphics, and image processing**, Elsevier, v. 30, n. 1, p. 32–46, 1985.

SZE, V. *et al.* Efficient Processing of Deep Neural Networks: A Tutorial and Survey. **Proceedings of the IEEE**, Institute of Electrical and Electronics Engineers Inc., v. 105, n. 12, p. 2295–2329, mar 2017. ISSN 15582256. Disponível em: <https://arxiv.org/abs/1703.09039v2>.

SZEGEDY, C. *et al.* Rethinking the Inception Architecture for Computer Vision. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, IEEE Computer Society, v. 2016-December, p. 2818–2826, dec 2015. ISSN 10636919. Disponível em: <https://arxiv.org/abs/1512.00567v3>.

TAGHANAKI, S. A. *et al.* Combo loss: Handling input and output imbalance in multi-organ segmentation. **Computerized Medical Imaging and Graphics**, Elsevier Ltd, v. 75, p. 24–33, 2019. ISSN 18790771. Disponível em: <https://doi.org/10.1016/j.compmedimag.2019.04.005>.

TAN, H. H.; LIM, K. H. Review of second-order optimization techniques in artificial neural networks backpropagation. **IOP Conference Series: Materials Science and Engineering**, 2019.

TEODORO, A. A. *et al.* A skin cancer classification approach using gan and roi-based attention mechanism. **Journal of Signal Processing Systems**, Springer, p. 1–14, 2022.

The GIMP Development Team. **GIMP**. 2019. Disponível em: <https://www.gimp.org>. Acesso em: 06 abr. 2022.

TIBSHIRANI, R. Regression Shrinkage and Selection Via the Lasso. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley, v. 58, n. 1, p. 267–288, jan 1996. ISSN 0035-9246. Disponível em: <https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.2517-6161.1996.tb02080.x>.

VERMA, R. **One-Stop-for-COVID-19-Infection-and-Lung-Segmentation-plus-Classification**. 2020. Disponível em: <https://github.com/deadskull7/One-Stop-for-COVID-19-Infection-and-Lung-Segmentation-plus-Classification>. Acesso em: 11 dez. 2021.

VIEILLARD-BARON, A.; GOFFI, A.; MAYO, P. Lung ultrasonography as an alternative to chest computed tomography in COVID-19 pneumonia? Springer Science and Business Media Deutschland GmbH, v. 46, n. 10, p. 1908–1910, oct 2020. ISSN 14321238. Disponível em: <https://doi.org/10.1007/s00134-020-06221-0>.

VUOLA, A. O.; AKRAM, S. U.; KANNALA, J. Mask-RCNN and u-net ensembled for nuclei segmentation. In: **Proceedings - International Symposium on Biomedical Imaging**. IEEE Computer Society, 2019. v. 2019-April, p. 208–212. ISBN 9781538636411. ISSN 19458452. Disponível em: <https://arxiv.org/abs/1901.10170v1>.

WACHARAPLUESADEE, S. *et al.* Evidence for SARS-CoV-2 related coronaviruses circulating in bats and pangolins in Southeast Asia. **Nature Communications**, Springer Science and Business Media LLC, v. 12, n. 1, p. 1–9, dec 2021. ISSN 2041-1723. Disponível em: <https://doi.org/10.1038/s41467-021-21240-1>.

WADA, K. **labelme: Image Polygonal Annotation with Python**. 2016. <https://github.com/wkentaro/labelme>. Acesso em: 03 fev. 2021.

WALLER, J. V. *et al.* Diagnostic Tools for Coronavirus Disease (COVID-19): Comparing CT and RT-PCR Viral Nucleic Acid Testing. **American Journal of Roentgenology**, v. 215, n. 4, p. 834–838, 2020. ISSN 15463141.

WANG, D. *et al.* Clinical Characteristics of 138 Hospitalized Patients with 2019 Novel Coronavirus-Infected Pneumonia in Wuhan, China. **JAMA - Journal of the American Medical Association**, American Medical Association, v. 323, n. 11, p. 1061–1069, mar 2020.

WANG, L.; LIN, Z. Q.; WONG, A. COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images. **Scientific Reports**, Nature Publishing Group UK, v. 10, n. 1, p. 1–12, 2020. ISSN 20452322. Disponível em: <https://doi.org/10.1038/s41598-020-76550-z>.

WANG, Q. *et al.* A Comprehensive Survey of Loss Functions in Machine Learning. **Annals of Data Science**, Springer Berlin Heidelberg, v. 9, n. 2, p. 187–212, 2022. ISSN 21985812. Disponível em: <https://doi.org/10.1007/s40745-020-00253-5>.

W.H.O. **World Health Organization, Coronavirus (COVID-19) Dashboard | Coronavirus (COVID-19) Dashboard With Vaccination Data**. 2021. Disponível em: <https://covid19.who.int/>. Acesso em: 22 jun. 2022.

WILKERSON, R. G. *et al.* Silent hypoxia: A harbinger of clinical deterioration in patients with COVID-19. **American Journal of Emergency Medicine**, Elsevier Inc, v. 38, n. 10, p. 2243.e5–2243.e6, 2020. ISSN 15328171. Disponível em: <https://doi.org/10.1016/j.ajem.2020.05.044>.

WU, F. *et al.* A new coronavirus associated with human respiratory disease in China. **Nature**, v. 579, n. 7798, p. 265–269, 2020. ISSN 14764687.

XIE, N. *et al.* Explainable Deep Learning: A Field Guide for the Uninitiated. **Journal of Artificial Intelligence Research**, v. 73, p. 329–397, 2020. ISSN 10769757.

YAKUBOVSKIY, P. **Segmentation Models Pytorch**. [S.l.]: GitHub, 2020. <https://github.com/qubvel/segmentation_models.pytorch>. Acesso em: 11 abr. 2022.

ZHANG, A. *et al.* **Dive Into Deep Learning: Release 0.16.1**. The authors, 2020. Disponível em: <https://books.google.com.br/books?id=TNMTzgEACAAJ>.

ZHANG, K. *et al.* Clinically applicable ai system for accurate diagnosis, quantitative measurements, and prognosis of covid-19 pneumonia using computed tomography. **Cell**, v. 182, n. 5, p. 1360, September 2020. ISSN 0092-8674. Disponível em: <https://europepmc.org/articles/PMC7470724>.

ZHOU, Z. *et al.* UNet++: A Nested U-Net Architecture for Medical Image Segmentation. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, Springer Verlag, v. 11045 LNCS, p. 3–11, jul 2018. ISSN 16113349. Disponível em: <https://arxiv.org/abs/1807.10165v1>.

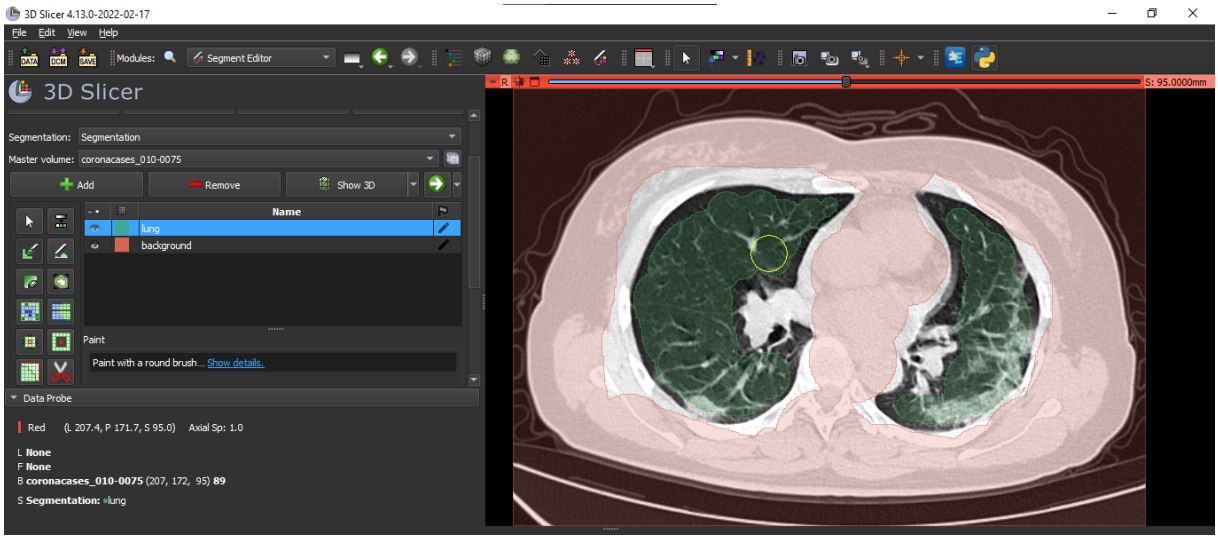
ZHUANG, J. *et al.* AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. **Advances in Neural Information Processing Systems**, Neural information processing systems foundation, v. 2020-December, oct 2020. ISSN 10495258. Disponível em: <https://arxiv.org/abs/2010.07468v5>.

ZIELESKIEWICZ, L. *et al.* Comparative study of lung ultrasound and chest computed tomography scan in the assessment of severity of confirmed COVID-19 pneumonia. **Intensive Care Medicine**, Springer, v. 46, n. 9, p. 1707–1713, sep 2020. ISSN 0342-4642. Disponível em: <http://link.springer.com/10.1007/s00134-020-06186-0>.

APÊNDICE A – Anotações Manuais com *SLICER* 3D

Para realizar as anotações foram utilizados os *softwares SLICER* 3D e *LabelMe*, (FEDOROV *et al.*, 2012; WADA, 2016). Ambas possuem ferramentas de fácil manipulação, com destaque para o *SLICER* 3D, que além das ferramentas de anotação mais simples, também possui ferramentas que utilizam algoritmos semiautomáticos para auxiliar no processo de anotação. Como é o caso da ferramenta *Grow From Seeds*, que utiliza internamente o algoritmo crescimento de regiões, definido na seção 2.6.2.5. Dentre todos os tipos de pacientes, a maior dificuldade em realizar anotações está naqueles em que as lesões aparecem nas extremidades do pulmão, junto ao corpo, nestas imagens o limiar entre os parênquimas e o corpo fica impreciso, dificultando as marcações. Para exemplificar o processo de anotações, observe a Figura A.1.

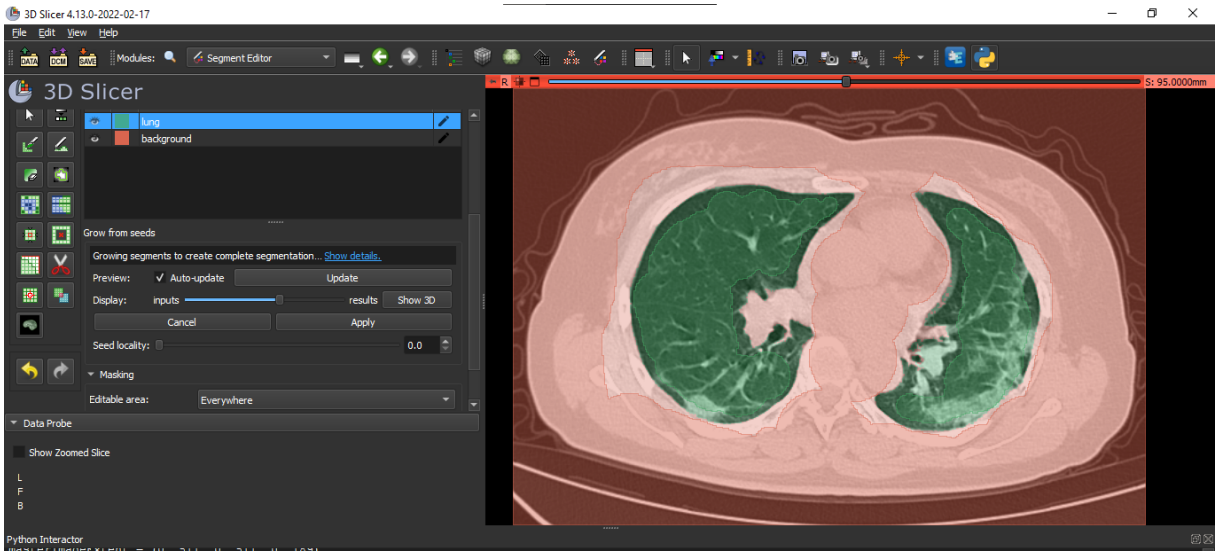
Figura A.1 – Exemplo de marcações via *SLICER* 3D.



Fonte: Arquivo Pessoal do Autor.

Após carregar o exame do paciente no *software*, utiliza-se o editor de segmentação para gerar duas marcações sobre uma imagem. De uma cor são marcadas as regiões de interesse (parênquimas pulmonares), e de outra cor são demarcadas o restante da imagem que não é de interesse. Em seguida utiliza-se a ferramenta automatizada que fará o preenchimento da imagem atual e de várias imagens adjacentes com base na marcação atual. Veja o resultado na Figura (A.2).

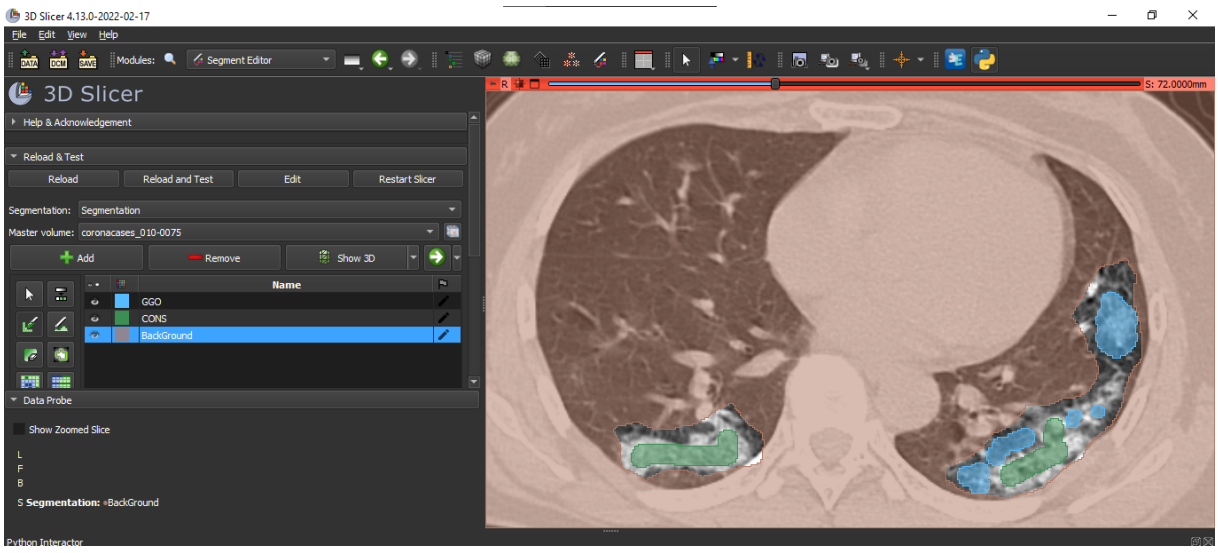
Figura A.2 – Exemplo de marcações via *SLICER* 3D - *Grow From Seeds* Aplicado aos parênquimas pulmonares.



Fonte: Arquivo Pessoal do Autor.

Utilizando este procedimento em diversos exames, foi possível gerar um banco de dados de máscaras para segmentação automática dos pulmões, bem como um banco de dados para segmentação de lesões. As mesmas ferramentas foram utilizadas para segmentar lesões, com a diferença de que são utilizadas 3 marcações, uma para opacidade em vidro fosco, uma para consolidação e outra para o restante da imagem. Como pode ser observado na Figura (A.3). Após a anotação, cada marcação funciona como semente para o algoritmo

Figura A.3 – Exemplo de marcações via *SLICER* 3D - *Grow From Seeds* Aplicado em Lesões.

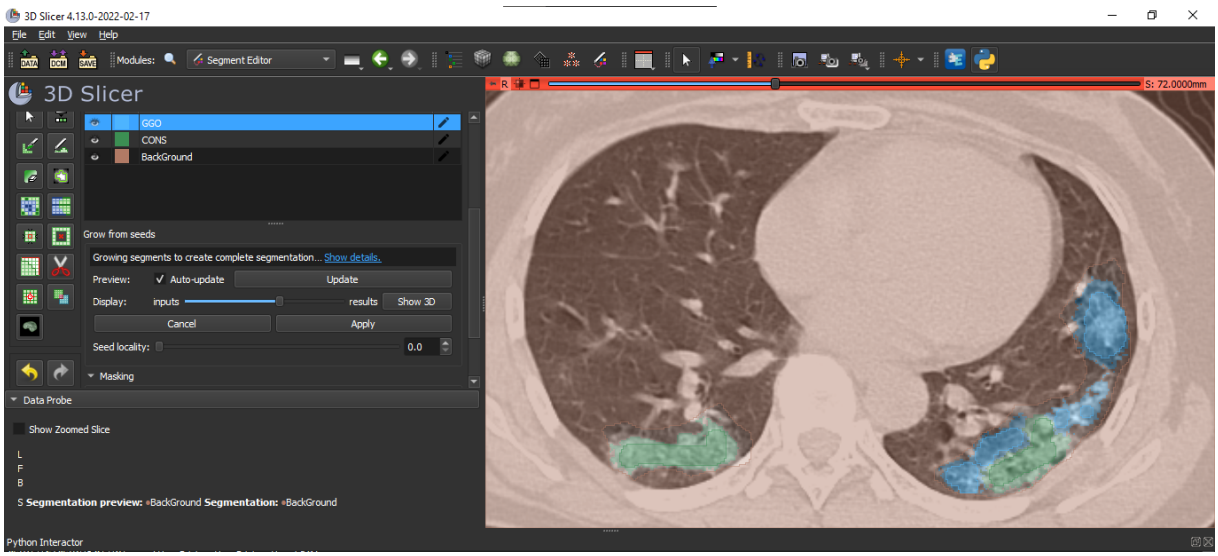


Fonte: Arquivo Pessoal do Autor.

agregar a região que mais se assemelha àquela demarcada. Por fim, caso o resultado não seja

satisfatório, é possível utilizar novamente a ferramenta com diferentes configurações para melhorar o resultado. Observe na Figura (A.4). Caso o resultado obtido com a ferramenta

Figura A.4 – Exemplo de marcações via *SLICER* 3D - *Grow From Seeds*.



Fonte: Arquivo Pessoal do Autor.

Grow From Seeds não seja satisfatório, mesmo com correções, também é possível utilizar outras ferramentas e extensões, como o *watershed* (seção 2.6.2.4). Para aprimorar ainda mais a velocidade de anotações, também podem ser utilizadas extensões que operam com modelos de aprendizado profundo, (CHEN, 2020).

APÊNDICE B – Referencial Teórico Complementar

A.1 Contexto histórico: Inteligência Computacional

Apesar da explosão de algoritmos, modelos e técnicas que tem surgido atualmente, as redes neurais começaram a ser desenvolvidas há muito mais tempo. Em 1943 o neurofisiologista *Warren Sturgis McCulloch* e o matemático *Walter Pitts*, publicaram um estudo sobre o cálculo lógico e sua relação com o sistema nervoso, o que deu origem ao paradigma conexionista. Este estudo consistia na primeira tentativa de se modelar o neurônio matematicamente. Poucos anos depois, outros estudos sobre o assunto despontaram, como o trabalho de *A. Turing* em 1950 complementando o paradigma simbólico, e também lançando o seu famoso teste para avaliar a inteligência de um algoritmo.

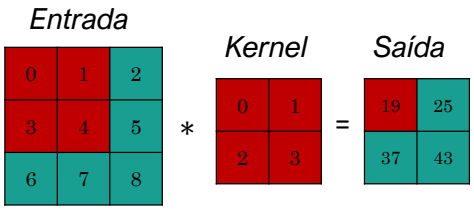
Em seguida, por volta de (1951), *Marvin Lee Minsky* desenvolve o primeiro simulador de redes neurais, o SNARC. Em 1956 se origina os dois paradigmas, o simbólico que busca mais interpretabilidade de modelos e interpretabilidade de seus resultados, e o conexionista, que é mais voltado aos resultados e à matemática dos modelos. Em 1958 *Frank Rosenblatt* introduz em seu livro Princípio da Neurodinâmica (do inglês, *Principles of Neurodynamics*), entre outros modelos (como o modelo topológico do sistema nervoso) o modelo perceptron. Nas décadas seguintes também houveram diversos avanços, como o *Multilayer Perceptron*, já que o *Perceptron* não resolvia problemas não lineares. Algoritmos para otimização dos modelos como o ADALINE (1960), o algoritmo retro-propagador (do inglês, *Backpropagation*) e diversos outros. Até o contexto atual, em que diversas áreas da ciência vêm utilizando e desenvolvendo modelos para solucionar os mais diversos problemas (MIT, 1951; PINKER *et al.*, 1988; MIRA, 2008; ROSENBLATT, 1962; LINNAINMAA, 1976).

A.2 Operações Elementares de CNN

A.2.1 Correlação Cruzada

A operação de correlação cruzada é comumente utilizada em aplicações práticas, tornando o nome redes convolucionais um erro em muitos casos práticos. Consiste de uma operação entre a entrada e o *kernel* (definida matematicamente na seção 2.4), pode ser observada na Figura A.5. Os quadros destacados em vermelho indicam como a operação é realizada: como o filtro possui dimensão 2×2, opera-se o produto de *Hadamard* entre a

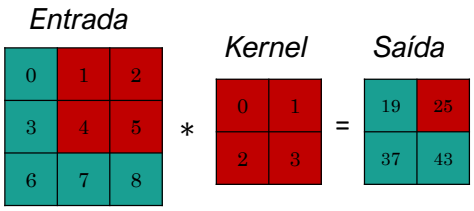
Figura A.5 – Exemplo da operação correlação cruzada.



Fonte: Adaptado de (ZHANG *et al.*, 2020a).

entrada e o filtro, e assim por diante, neste caso foi realizado um passo de 1 posição, o que significa que a próxima operação deve ser realizada de acordo com a Figura A.6. Quando é

Figura A.6 – Exemplo da operação correlação cruzada.



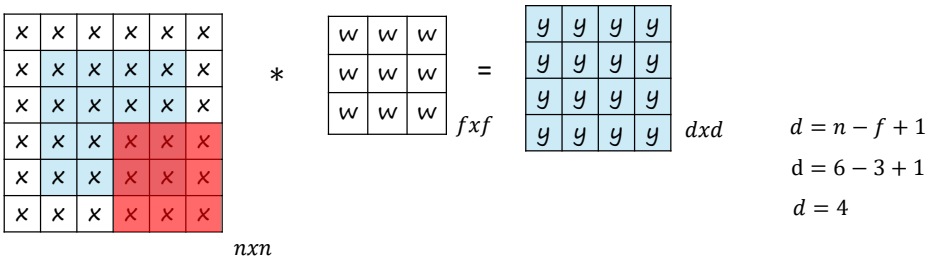
Fonte: Adaptado de (ZHANG *et al.*, 2020a).

utilizada a convolução, os pesos do filtro são alternados posicionalmente.

A.2.1.1 *Padding*

Quando se aplica a operação de correlação cruzada ou convolução espacial de um *kernel* nxn sobre uma imagem ou espaço de características, a saída tende a ser reduzida, caso não seja realizado o procedimento de *Padding*. A Figura A.7 exemplifica o processo de convolução e como ele reduz a dimensão da entrada. Como é possível notar, a saída

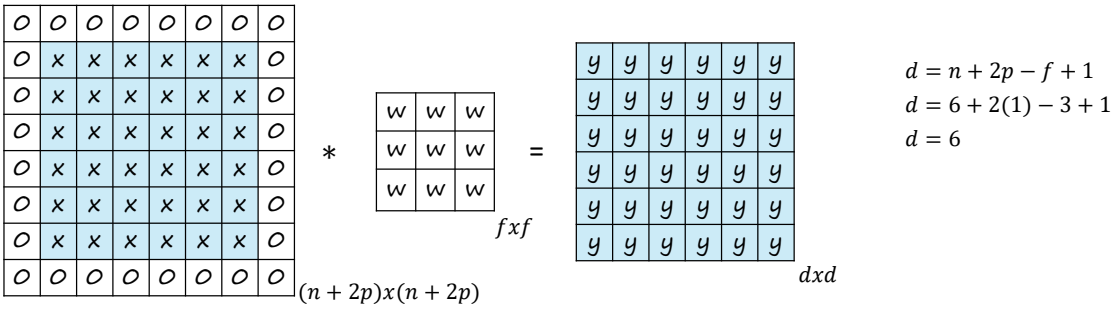
Figura A.7 – Exemplo de convolução $3x3$, em que não é realizado o *padding*.



Fonte: Arquivo Pessoal do Autor.

é reduzida em 2 dimensões em relação à entrada. A operação de *padding* consiste em aplicar-se uma camada de zeros em torno da entrada, esse preenchimento é útil em casos que se deseja manter a mesma dimensão de entrada na saída. Nestes casos, aumenta-se a dimensão da entrada e preenche-se com zeros, como pode ser visto na Figura A.8. Após realizar o *padding*, realiza-se a operação de correlação/convolução normalmente, e como esperado a dimensão de entrada e de saída são as mesmas (SZE *et al.*, 2017; PASZKE *et al.*, 2019; ABADI *et al.*, 2015).

Figura A.8 – Exemplo de preenchimento $1x1$.



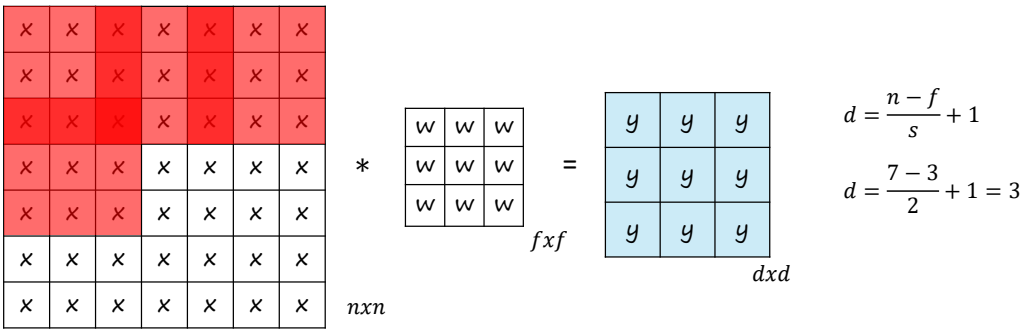
Fonte: Arquivo Pessoal do Autor.

A.2.1.2 *Stride*

A operação de *stride* realiza um passo a cada iteração e irá reduzir significativamente a dimensão de entrada, e consequentemente torna o algoritmo mais rápido. Entretanto, dependendo do valor do passo, muita informação de entrada pode ser perdida, colocando em risco o mapeamento estatístico dos dados. Um exemplo genérico da saída ao aplicar a operação *stride* é dado pela Figura A.9.

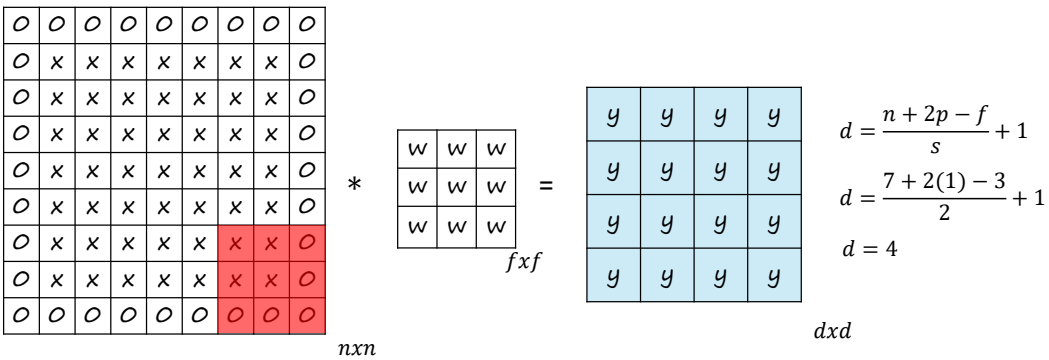
Como pode ser observado, a ação do passo $2x2$ reduz bem mais a dimensionalidade da entrada. Caso seja utilizado em conjunto com o *padding*, o efeito é atenuado, como mostra a Figura A.10.

Figura A.9 – Exemplo de passo $2x2$.



Fonte: Arquivo Pessoal do Autor.

Figura A.10 – Exemplo de passo $(2x2)$ e preenchimento $(1x1)$.



Fonte: Arquivo Pessoal do Autor.

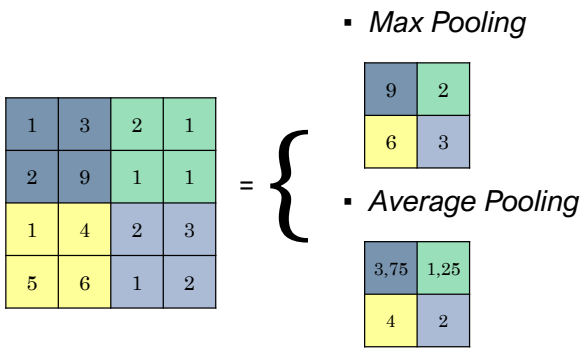
Vale destacar que a matriz na cor vermelha que sobrepõe as matrizes simplesmente indicam que o filtro percorre a entrada da esquerda para direita até o canto inferior direito. Além disso, quando uma arquitetura é planejada as equações que foram utilizadas nos

exemplos para calcular as saídas são utilizadas de forma inversa, ou seja, o termo do filtro é colocado em evidencia para se obter a dimensão dos mesmos (SZE *et al.*, 2017; PASZKE *et al.*, 2019; ABADI *et al.*, 2015).

A.2.1.3 *Pooling*

A operação de *pooling* possui diversas variações, até mesmo na composta com algoritmos de atenção (*attention pooling* (ZHANG *et al.*, 2020a)). Os casos mais simples correspondem ao agrupamento por média e agrupamento por máximo. Um filtro nxn percorre a entrada da mesma forma como ocorre na convolução, porém ao invés de realizar qualquer tipo de operação entre ele e a entrada, o maior valor (ou valor médio) encontrado dentro da submatriz sobreposta será a saída (ABADI *et al.*, 2015; PASZKE *et al.*, 2019). O *max-pooling* e *average-pooling* são amplamente utilizados para redução de complexidade e compressão da entrada, na Figura A.11 podem ser observados exemplos de como elas funcionam.

Figura A.11 – Exemplo de *pooling*.



Fonte: Arquivo Pessoal do Autor.

As operações de *max* e *average pooling* são utilizadas para reduzir a quantidade de parâmetros da rede ao longo das camadas, uma vez que estas operações não utilizam os *kernels* (ZHANG *et al.*, 2020a).

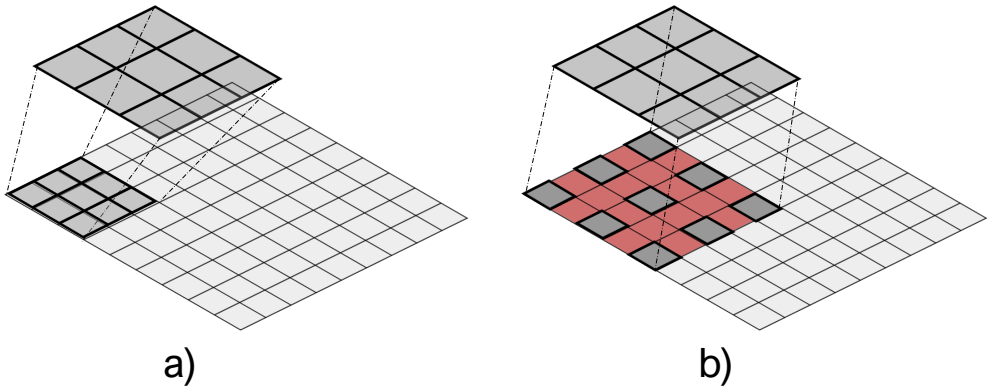
A.2.2 Outras Operações

Além das operações básicas descritas na seção anterior, ao longo do desenvolvimento de aplicações em DL e visão computacional, diversos mecanismos foram se desenvolvendo para aprimorar o mapeamento de características e a eficiência dos algoritmos.

A.2.2.1 *Dilated Convolution*

Na convolução por dilatação também conhecida por *Atrous Convolution*, ao contrário da convolução regular como foi mostrado anteriormente, a ação do filtro sobre a entrada é alterada de forma a aumentar o campo de visão/ação do filtro. Conforme a Figura A.12:

Figura A.12 – Exemplo de *Atrous Convolution*.



Fonte: Arquivo Pessoal do Autor.

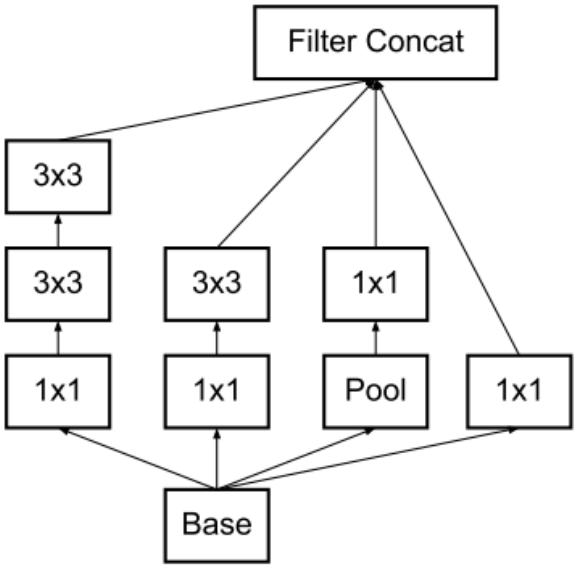
Em visão computacional, o campo receptivo[1] (do inglês, *receptive field*) é o mapa de características de entrada que cada camada recebe. O campo de visão (do inglês, *field of view*), consiste na dimensão que a camada atual consegue processar por vez, ou seja, para um filtro nxn a dimensão do campo de visão é definida por n. A ideia da convolução dilatada é expandir o campo de visão sem aumentar a quantidade de parâmetros necessários, isto é feito pela incorporação de valores nulos ao filtro (região em vermelho). Logo, a dimensão do filtro utilizado é a mesma, mas a região mapeada é diferente (JIAO *et al.*, 2019).

[1] Biologicamente se trata das parte anatômica (células) do córtex visual que é estimulada (HUBEL; WIESEL, 1962) ao ser submetida a um ambiente visual.

A.2.2.2 *Inception* and ASPP

Uma estratégia frequentemente aplicada em modelos de DL, é a de obter múltiplos mapas de hiper-parâmetros, que foi desenvolvida até chegar na proposta *Inception*, na qual são utilizados múltiplos filtros com diferentes dimensões para gerar diferentes níveis de representatividade (campos visuais), esta técnica faz com que o modelo consiga absorver diferentes relações entre os dados tabulares, como mostra a Figura A.13:

Figura A.13 – Exemplo *InceptionV3*.

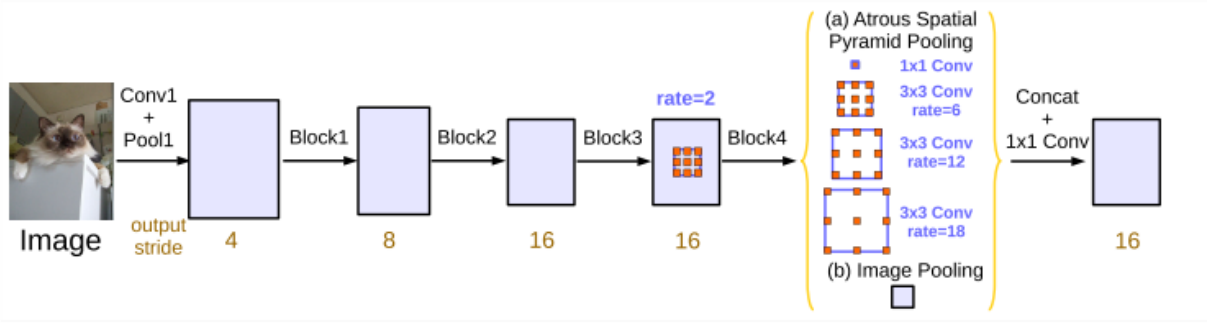


Fonte: (SZEGEDY *et al.*, 2015).

Utilizando as convoluções $1x1$ é possível reduzir significativamente a dimensão de entrada. Por exemplo na Figura A.13, suponha que em uma certa camada seja necessário realizar a redução da dimensionalidade (quantidade de canais) do espaço de características que foram acumulados através das camadas da rede, a convolução $1x1$ não só pode cumprir este papel como também selecionar as melhores características dentre os vários canais. Para o bloco *Pool* (*Max*/*Avarage Pooling*), não existem pesos como na convolução $1x1$ e, portanto, elimina a necessidade de retropopagar o erro, porém a seletividade fica somente a cargo da operação associada (SZEGEDY *et al.*, 2015).

Uma outra proposta para aumentar a variabilidade de mapas de hiper-parâmetros é o método ASPP (*Atrous Spatial Pyramid Pooling*). Assim como o método *Inception*, este método busca aumentar a quantidade de representações de características, fazendo com que o modelo aprenda de diferentes maneiras, porém utilizando a estratégia *Atrous* (descrita na seção A.2.2.1). Observe na Figura A.14:

Figura A.14 – Exemplo ASPP.



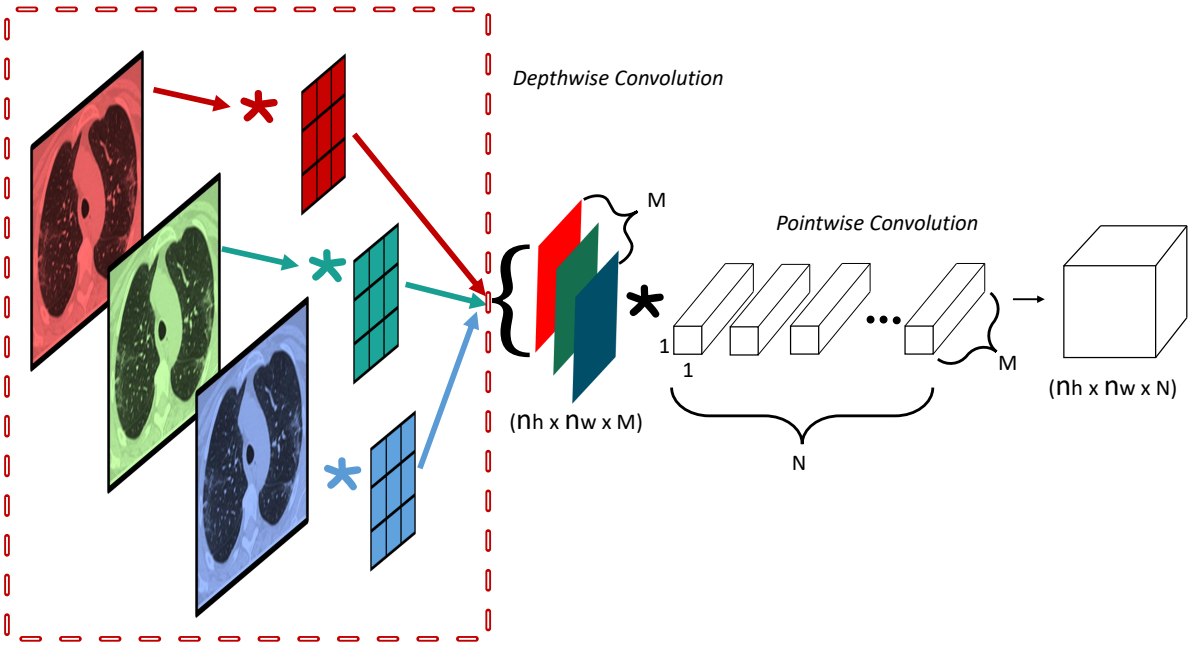
Fonte: (CHEN *et al.*, 2017).

Conforme (CHEN *et al.*, 2017) no qual foi proposto o modelo de segmentação *DeepLabV3*, as múltiplas operações de convolução dilatada possibilitam uma extração mapas de características capazes de capturar um contexto de longo alcance.

A.2.2.3 *DepthWise Separable Convolution*

Na operação de "convolução" convencional, todos os filtros de uma camada da arquitetura operam (convolução ou correlação cruzada) sobre a imagem/mapa de entrada. Já a operação de convolução separável busca subdividir a convolução em duas etapas: A primeira orientada à profundidade (*Depthwise Convolution*), significa que para cada canal de entrada somente um filtro opera sobre um canal. A segunda faz a convolução elemento a elemento (*Pointwise Convolution*). Quando estas duas operações são combinadas formam a operação *Depthwise Separable Convolution*. A operação *Pointwise Convolution* consiste na convolução $1x1$ (na qual cada filtro possui a mesma quantidade de canais que a entrada). É importante destacar que nesta etapa é realizada a operação entre cada filtro (1 x 1 x M) com o mapa de saída na etapa anterior (n_H x n_W x M), e cada filtro irá compor um canal da camada de saída (n_H x n_W x N). Ao final da operação o resultado é semelhante ao obtido com a convolução convencional, porém com uma quantidade significativamente menor de operações (BAI; ZHAO; HUANG, 2018).

Figura A.15 – Exemplo *Depthwise Separable Convolution*.



Fonte: Arquivo Pessoal do Autor.

Na Figura A.15 temos a ilustração das duas operações supracitadas, *Depthwise Convolution*, que é realizada entre as imagens (ou mapas de características) e cada filtro separadamente (representados por matrizes $3x3$ dentro da região tracejada), e o resultado desta operação é concatenado (região central da imagem), e então realiza-se sobre ele a operação *Pointwise Convolution*, também conhecida como convolução $1x1$. Como mencionado anteriormente esta convolução faz o papel de aumentar ou reduzir a quantidade de canais entre dois estágios sem alterar a dimensão do formato (altura e largura), e selecionando as melhores características entre os canais.

Muitos trabalhos optam por utilizar estas operações de aumento da representatividade, primeiramente para redução na quantidade de operações necessárias, e em segundo por reduzir a "redundância" de filtros, pois muitas arquiteturas geram filtros semelhantes em uma mesma camada quando utilizam a convolução convencional (BAI; ZHAO; HUANG, 2018; CHEN *et al.*, 2017; CHEN *et al.*, 2021).

A.3 Morfologia Matemática

A Morfologia matemática é uma subárea de estudos matemáticos que se concentra em análises sobre formatos geométricos e/ou estruturas. Além disso, ela pode colaborar para fins práticos em estudos científicos ou aplicações industriais que são relacionados à imagens, como realizar algum aprimoramento na visualização ou aspecto da imagem, reconhecimento de objetos, extração de formatos, entre outros (SHIH, 2009). Operações morfológicas basicamente envolvem duas imagens, a primeira é a que está sendo processada, denominada imagem ativa, e a segunda é o *kernel*, referido como elemento estruturante. As operações mais básicas da morfologia matemática aplicada ao processamento de imagens são as operações de dilatação e erosão, que podem ser combinadas para produzir outras operações, como a abertura e o fechamento. Conforme Shih (2009), as operações morfológicas podem ser definidas com imagens binárias ou em escala de cinza. Tanto a definição quanto os exemplos serão realizados sobre imagens binárias e a definição é equivalente para imagens em escala de cinza, que podem ser representadas como imagens binárias em três dimensões. Como as imagens no banco de dados, que são foco do presente estudo, estão em grande maioria na escala de cinza ou RGB, foi necessário converter para o formato binário utilizando o método *Otsu*[2] (OTSU, 1979).

A.3.1 Dilatação

Sejam A e B dois conjuntos em E^N, com elementos $a \in A$ e $b \in B$, em que $a = (a_1, a_2, \ldots, a_N)$ e $b = (b_1, b_2, \ldots, b_N)$. A dilatação de A (imagem ativa) por B (elemento estruturante), é o conjunto de todas as somas vetoriais possíveis de pares de elementos, um proveniente de A e outro de B.

$$A \oplus_b B = \{c \in E^N | c = a + b \text{ para alguma} a \in A e b \in B\}. \tag{A.1}$$

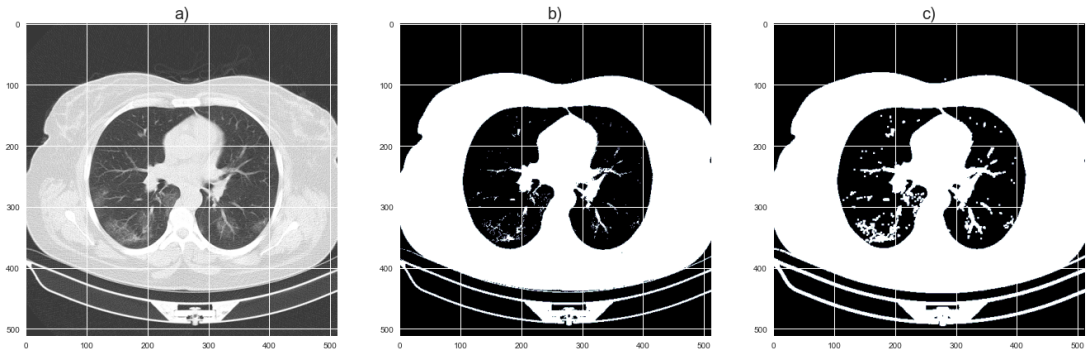
Como exemplo, iremos utilizar um elemento estruturante do tipo:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{A.2}$$

[2] O método Otsu é utilizado para encontrar o limiar automático de uma imagem, considerando duas classes distintas o algoritmo busca um limite de separação entre os pixels claros e escuros.

Após a conversão da Figura A.16.a) original; para forma binária na Figura A.16.b), aplica-se o operador de dilatação, o resultado fica de acordo com a Figura A.16.c).

Figura A.16 – Exemplo de Dilatação. a) Imagem Original, b) Imagem no formato binário com filtro *Otsu*, e c) Imagem com Dilatação.



Fonte: Arquivo Pessoal do Autor.

É importante ressaltar que a dilatação realiza uma expansão nas regiões onde se concentra os pixels brancos, como pode ser observado na região no interior dos parênquimas, em que as regiões em branco na Figura A.16.b), são expandidas na Figura A.16.c).

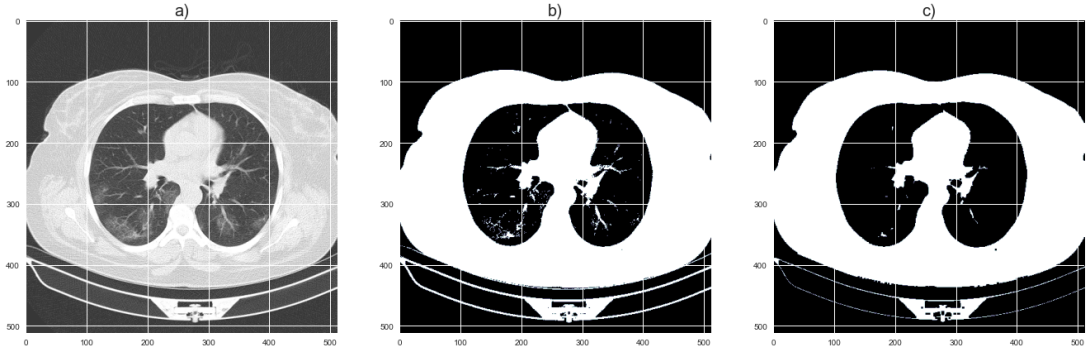
A.3.2 Erosão

Ao contrário da dilatação, a erosão fará os pixels brancos da imagem se encolherem ou "achatarem". Utilizando os mesmos conjuntos A e B descritos na dilatação, a erosão pode ser definida pela equação A.3.

$$A \ominus_b B = \{x \in E^N | x + b \in A \text{ para todo} b \in B\}. \tag{A.3}$$

Como exemplo (utilizando o mesmo elemento estruturante de A.2), pode ser observado na Figura A.16.a) (original); a sua forma binária na Figura A.16.b), e o resultado após a erosão na Figura A.16.c).

Figura A.17 – Exemplo de Erosão. a) Imagem Original, b) Imagem no formato binário com filtro *Otsu*, e c) Imagem com Erosão.



Fonte: Arquivo Pessoal do Autor.

Observe que na operação de erosão, a região branca interna da Figura A.17.b) é parcialmente suprimida pela erosão, como pode ser observado na Figura A.17.c).

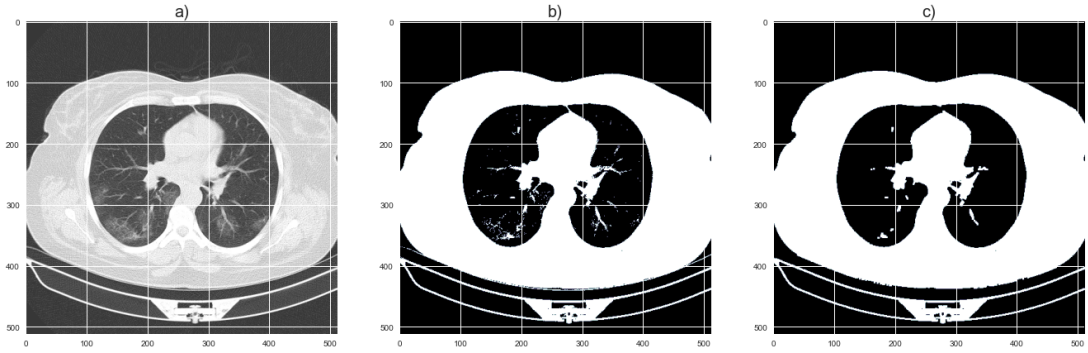
A.3.3 Abertura

A abertura, é definida por uma erosão seguida por uma dilatação, ficando definida pela equação A.4.

$$A \circ_b B = (A \ominus_b B) \oplus_b B \tag{A.4}$$

Em que $\circ_b$ é a notação para operação de abertura binária. Como exemplo temos a Figura original A.18.a), a forma binária A.18.b) e a o resultado da abertura (A.18.c).

Figura A.18 – Exemplo de Abertura. a) Imagem Original, b) Imagem no formato binário com filtro *Otsu*, e c) Imagem com Abertura.



Fonte: Arquivo Pessoal do Autor.

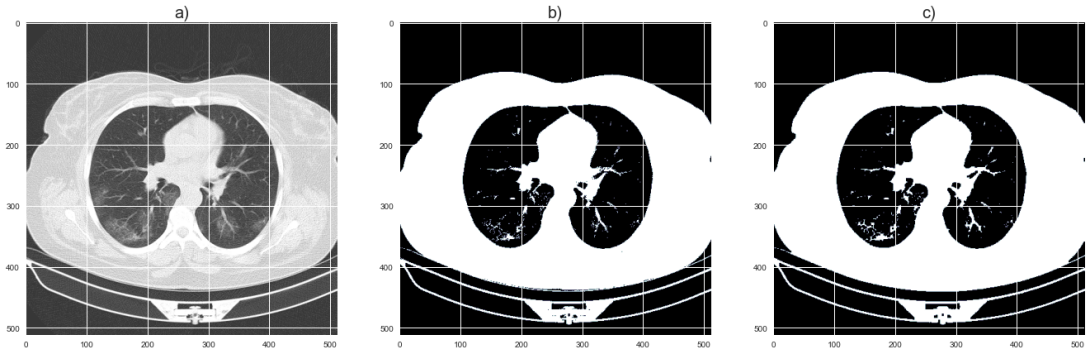
A.3.4 Fechamento

De forma semelhante, o fechamento é definido por uma dilatação seguida por uma erosão, ficando definida pela equação A.5.

$$A \bullet_b B = (A \oplus_b B) \ominus_b B \tag{A.5}$$

Em que $\bullet_b$ é a notação para operação de fechamento binário. Como exemplo temos a Figura A.19.a) (original); a sua forma binária na Figura A.19.b) e o resultado do fechamento na Figura A.19.c).

Figura A.19 – Exemplo de Fechamento. a) Imagem Original, b) Imagem no formato binário com filtro *Otsu*, e c) Imagem com Fechamento.



Fonte: Arquivo Pessoal do Autor.

Estas operações são as mais básicas em morfologia matemática, e serão discutidas em alguns dos algoritmos a seguir. Principalmente nos métodos semi-automáticos.

A.4 Regularização

A.4.1 Penalização

Os métodos de regularização clássicos consistem em penalizar a função de erro de acordo com os valores dos pesos durante o treinamento buscando adaptar o modelo para as amostras fora do conjunto amostrado, ou seja, para os dados desconhecidos. Caso o peso seja alto para um determinado atributo, a penalização será proporcionalmente alta, e os pesos serão reajustados por um erro ainda maior na próxima iteração, fazendo com que eles sejam reduzidos, adicionando um termo regularizador na equação A.6.

$$\text{Função de erro}_{nova} = \text{Função de erro} + \lambda \cdot \text{Regularização} \tag{A.6}$$

Na qual λ é a "força" da "punição" pelo alto valor dos pesos. O termo regularizador pode ser determinado pelo método denominado Operador de seleção e redução mínima absoluta (do inglês, *Least Absolute Shrinkage and Selection Operator* – LASSO), também conhecido como regularizador L_1, ou pelo método de *Ridge*, também conhecido como regularizador L_2.

A.4.1.1 Regularização de *Lasso*

O erro penalizado pelo método L_1 de cada i-ésima amostra entre um conjunto de dados total de N, pode ser determinado de forma genérica pela equação A.7.

$$\text{Erro}_{Lasso}(w,b) = \sum_{i=1}^{N} (y_i - (w \cdot x_i + b))^2 + \lambda \sum_{j=1}^{P} |w_j| \tag{A.7}$$

Com a regularização por *Lasso*, os pesos são reduzidos consideravelmente, alguns podendo ser até mesmo anulados ($w = 0$), realizando a "poda" de alguns neurônios do modelo e deixando a arquitetura um pouco mais esparsa (TIBSHIRANI, 1996).

A.4.1.2 Regularização de *Ridge*

Também conhecida como regularização L_2, a regularização de *Ridge* Hastie (2020), também tem por objetivo fazer com que o modelo fique menos complexo e aumente sua capacidade de generalização, porém, ao contrário da regularização pelo método *Lasso*, a regularização L_2 não anula os valores dos parâmetros, apenas os "encolhe". O erro pode ser

determinado de acordo com a equação A.8.

$$\text{Erro}_{Ridge}(w,b) = \sum_{i=1}^{N}(y_i - (w \cdot x_i + b))^2 + \lambda \sum_{j=1}^{P} |w_j|^2 \tag{A.8}$$

É interessante destacar que a única diferença entre as equações está no expoente do somatório de penalização. Enquanto em L_1 se utiliza o valor absoluto do somatório dos pesos, em L_2, utiliza-se a soma do valor ao quadrado para penalização, em ambos os casos os parâmetros decaem proporcionalmente ao valor de penalização, porém somente o método L_1 possui o decaimento linear, tendo, portanto, a possibilidade de anulação de alguns parâmetros durante o treinamento.

A.4.2 *Dropout*

De acordo com Srivastava *et al.* (2014) o *Dropout* pode ser visto como uma forma de fazer uma média igualmente ponderada de diversos modelos com hiper parâmetros compartilhados. Por outro lado, de forma empírica, os autores conseguiram mostrar como o método contribui para a ponderação dos pesos após o treinamento.

A forma original desta técnica consiste em anular uma ativação com uma probabilidade p na fase de treinamento, e na fase de testes os pesos devem ser multiplicados por p para escalonar e manter a média. Para eliminar a necessidade deste procedimento, pode ser utilizado o "*DropOut* Inverso", no qual a probabilidade de anular se torna $1-q$ para $0 < q \leq 1$. Para preservar as médias, as ativações são escalonadas em $1/q$ durante o treinamento, e na etapa de teste não é necessário realizar alterações nos pesos. Se as entradas x tem média $E(x) = \mu$ (E : Valor Esperado) e variância Var$(x) = v$, então a variável de *DropOut* d segue uma distribuição binomial $B(1,q)$, e a média $E(1/qdx) = \mu$ é mantida. A eficácia da técnica foi comprovada apenas de forma empírica (SRIVASTAVA *et al.*, 2014; KLAMBAUER *et al.*, 2017).

A.4.2.1 *Alpha DropOut*

Um caso particular de *DropOut* foi proposto em Klambauer *et al.* (2017), em que a técnica é modificada para manter a média sob a influência de uma função de ativação específica a SELU (que foi discutida na seção 2.8.6), que basicamente leva o valor das ativações para um valor próximo de 0 $(0,05\,0,1)$.

A.4.3 Regressão

A.4.3.1 Erro Médio Absoluto

Para sistemas cuja saída possui uma estimação numérica contínua, em um cenário ideal o valor calculado na saída do modelo possui o mesmo valor alvo que amostra possui. Porém na prática o que ocorre são valores aproximados, que podem flutuar para valores acima ou abaixo do valor alvo, logo, é comum utilizar o valor absoluto do erro, para que os termos não se cancelem. Além disso, para análise em grandes quantidades, a tendência do erro pode ser melhor analisada quando se observa a média de diversos valores ao invés de analisar simplesmente o somatório dos erros individuais, tendo assim uma percepção de tendência do erro. Portanto, é interessante que o erro seja computado sobre uma faixa de amostras ao longo do aprendizado de um modelo de inteligência computacional, tendo assim o erro médio absoluto (do inglês, *Mean absolute Error* - MAE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y^i - \hat{y}^i| \tag{A.9}$$

O erro médio absoluto analisa de forma linear os erros, o que significa que os erros são penalizados na mesma proporção para valores altos ou baixos.

A.4.3.2 Erro Médio Quadrático

O erro médio quadrático (do inglês, *Mean Squared Error* - MSE) por sua vez, penaliza altos valores de erro com mais “severidade”.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2 \tag{A.10}$$

Um problema comumente encontrado em aplicações que utilizam o MSE, é a penalização acentuada em amostras com valores abruptamente fora do padrão, chamadas de “pontos fora da curva” (do inglês, *outiliers*).

A.4.3.3 Raiz do Erro Médio Quadrático

A raiz erro médio quadrático (do inglês, *Root Mean Squared Error* - RMSE), busca ponderar o erro médio quadrático, tanto para evitar que os valores e correções sejam muito

abruptos, quanto para conferir maior interpretabilidade ao resultado quando comparado ao MSE, a métrica pode ser determinada pela equação A.11.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2} \tag{A.11}$$

A.4.3.4 Coeficiente de Determinação

Também chamado de R-quadrado (do inglês, *R-Squared*), esta métrica compara os resultados de saída do modelo com um modelo imaginário que sempre indica o valor médio dos alvos do conjunto de dados em questão em sua saída.

$$R^2 = 1 - \frac{MSE(modelo)}{MSE(modelo_{base})} \tag{A.12}$$

No qual o MSE (modelo) é dado pela equação A.10 e o erro MSE (modelo$_{base}$) é dado por:

$$MSE(modelo_{base} = \frac{1}{n}\sum_{i=1}^{n}(\overline{y}^i - \hat{y}^i)^2 \tag{A.13}$$

A.5 Redimensionamento por Interpolação por Área

De acordo com a documentação da biblioteca OpenCV Bradski (2000), a interpolação por área é a mais indicada para casos em que se deseja reduzir a dimensão da imagem, uma vez que o algoritmo se baseia na informação de área para realizar tal procedimento. O algoritmo primeiramente verifica se a dimensão original da imagem é múltipla escalar da dimensão de saída (informada no momento de chamada da função). Ou seja, verifica-se se a divisão da altura e largura de entrada pela saída, é um valor inteiro.

Caso seja um valor inteiro n, e a imagem de entrada é maior que a de saída, então o algoritmo fará uma média ponderada por área em uma região n por n na imagem original, ou seja, a soma dos pixels contidos em uma região dividida pela área.

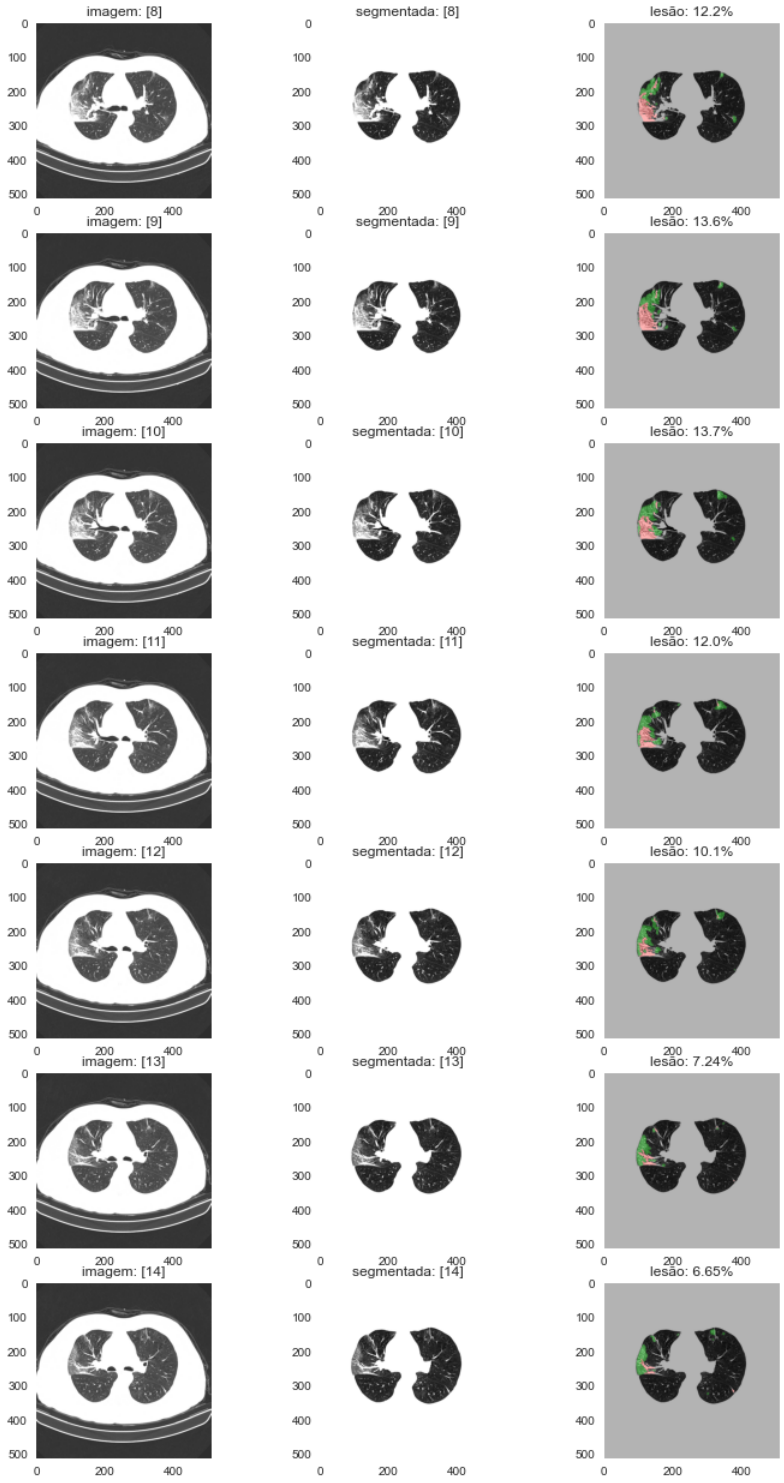
Caso a relação entre a entrada e a saída não seja um valor inteiro, e a imagem de entrada é maior que a de saída, então o algoritmo fará um procedimento semelhante, porém ponderando a soma dos pixels pela proporção que eles ocupam o quadro n por n na imagem original.

Para casos em que a imagem de entrada é menor que a de saída, é realizada uma interpolação baseada no conceito “vizinho mais próximo” (do inglês, *Nearest Neighbor*. Esta interpolação utiliza o pixel mais próximo para interpolação (BRADSKI, 2000).

APÊNDICE C – Resultados Adicionais

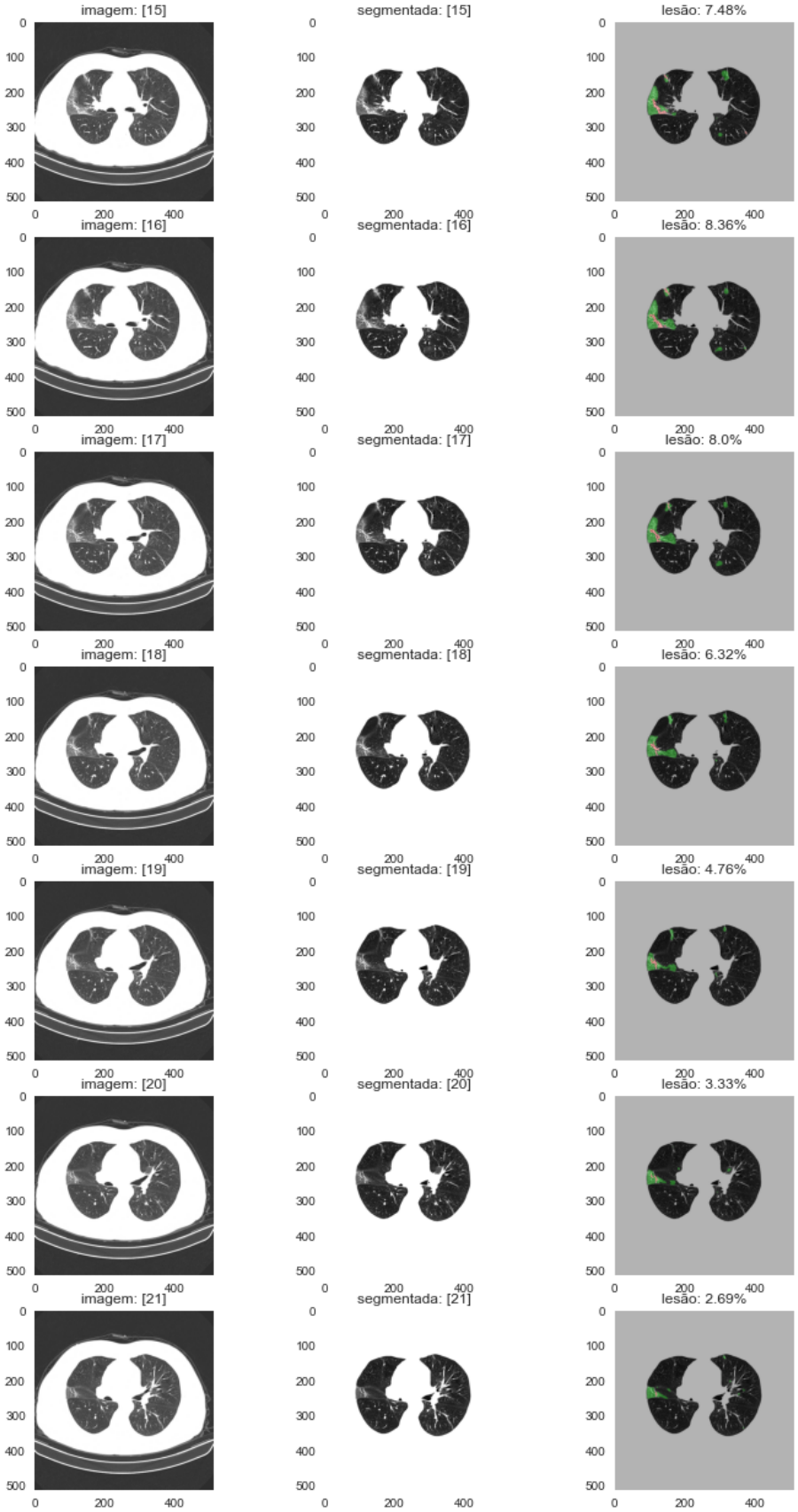
Paciente contido na base de dados, *patient id* NCP 215. Restante das fatias:

Figura A.20 – Análise de severidade de lesão pulmonar de um paciente.



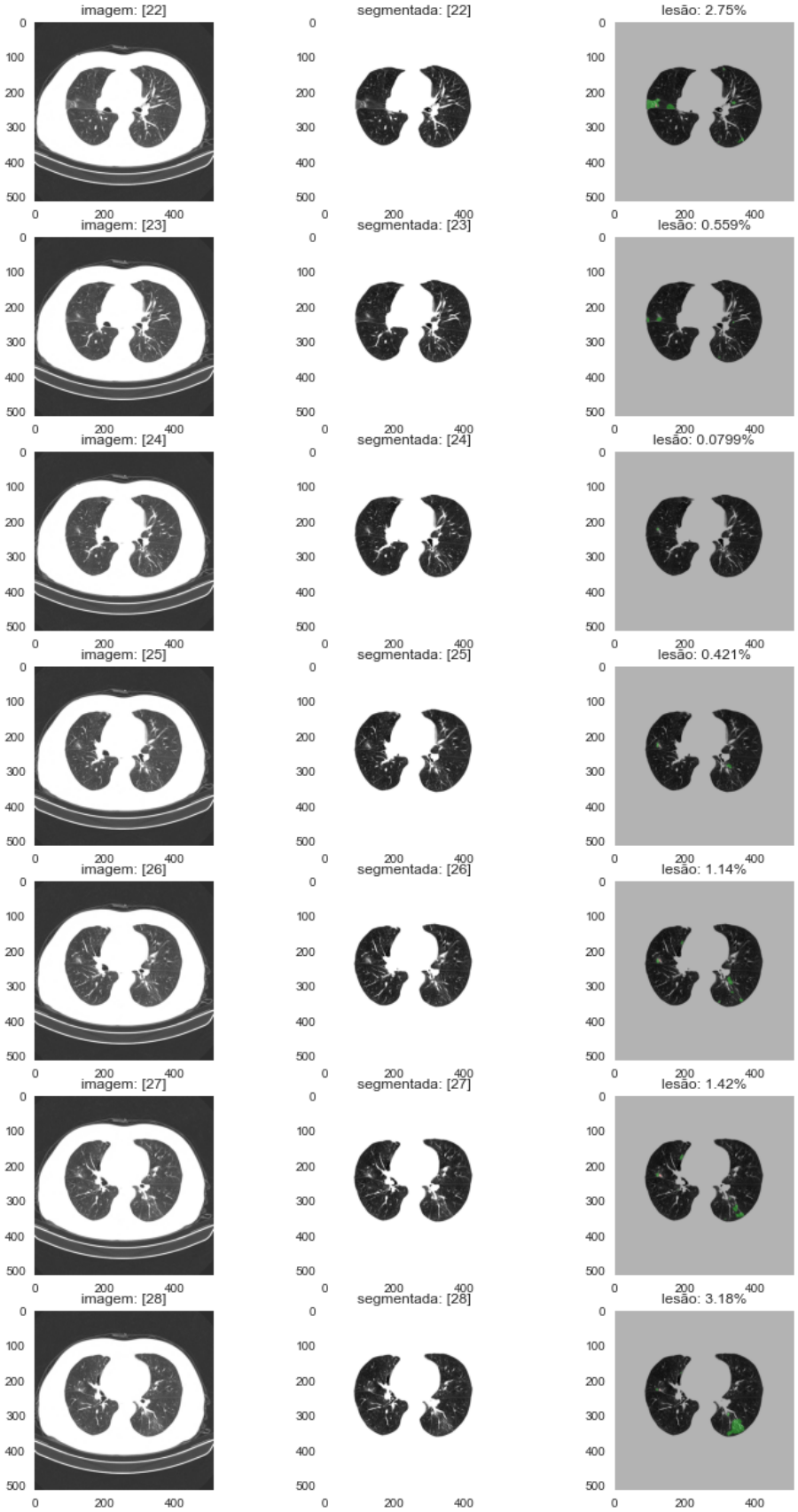
Fonte: Arquivo pessoal do Autor.

Figura A.21 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



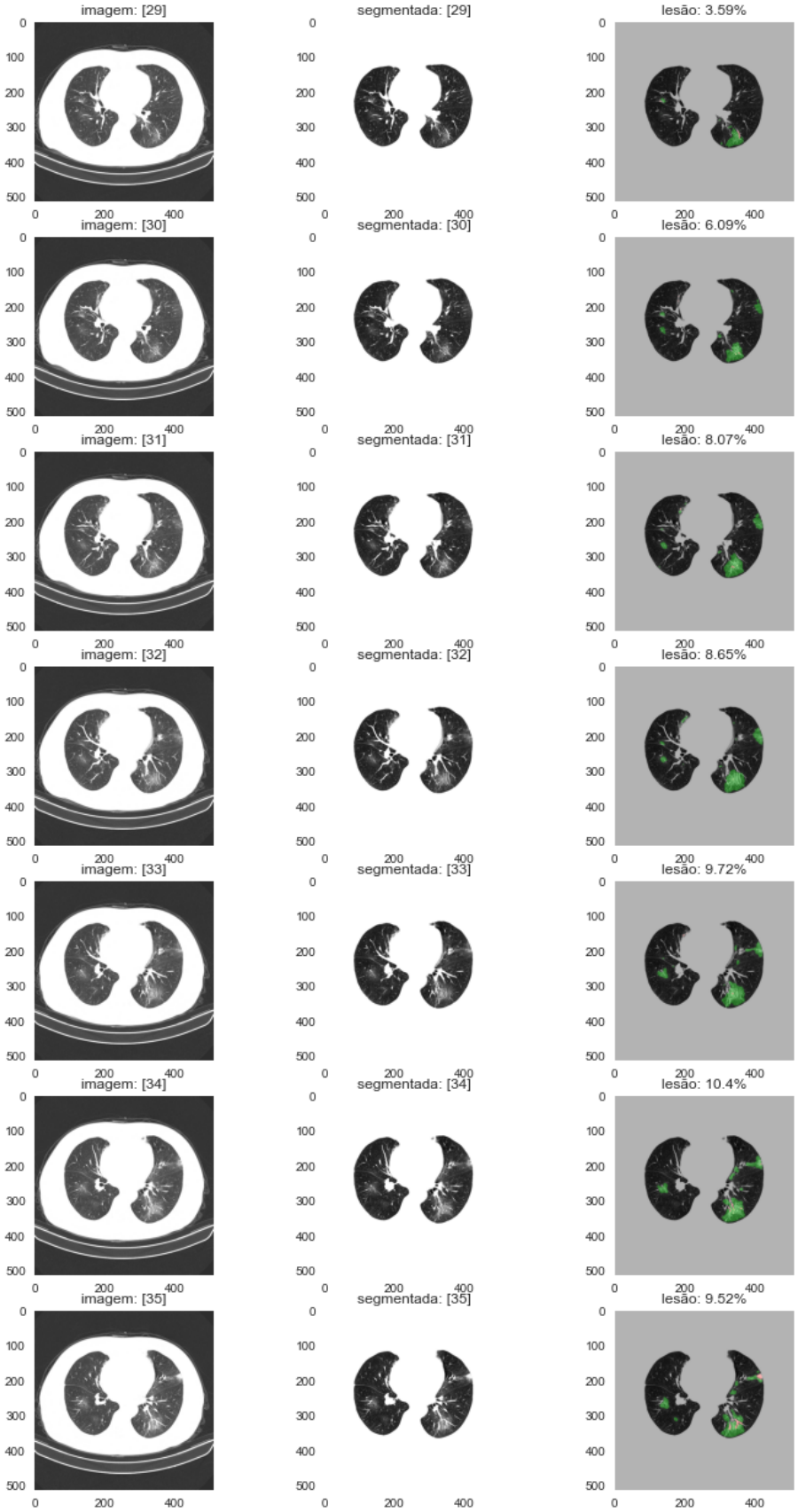
Fonte: Arquivo pessoal do Autor.

Figura A.22 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



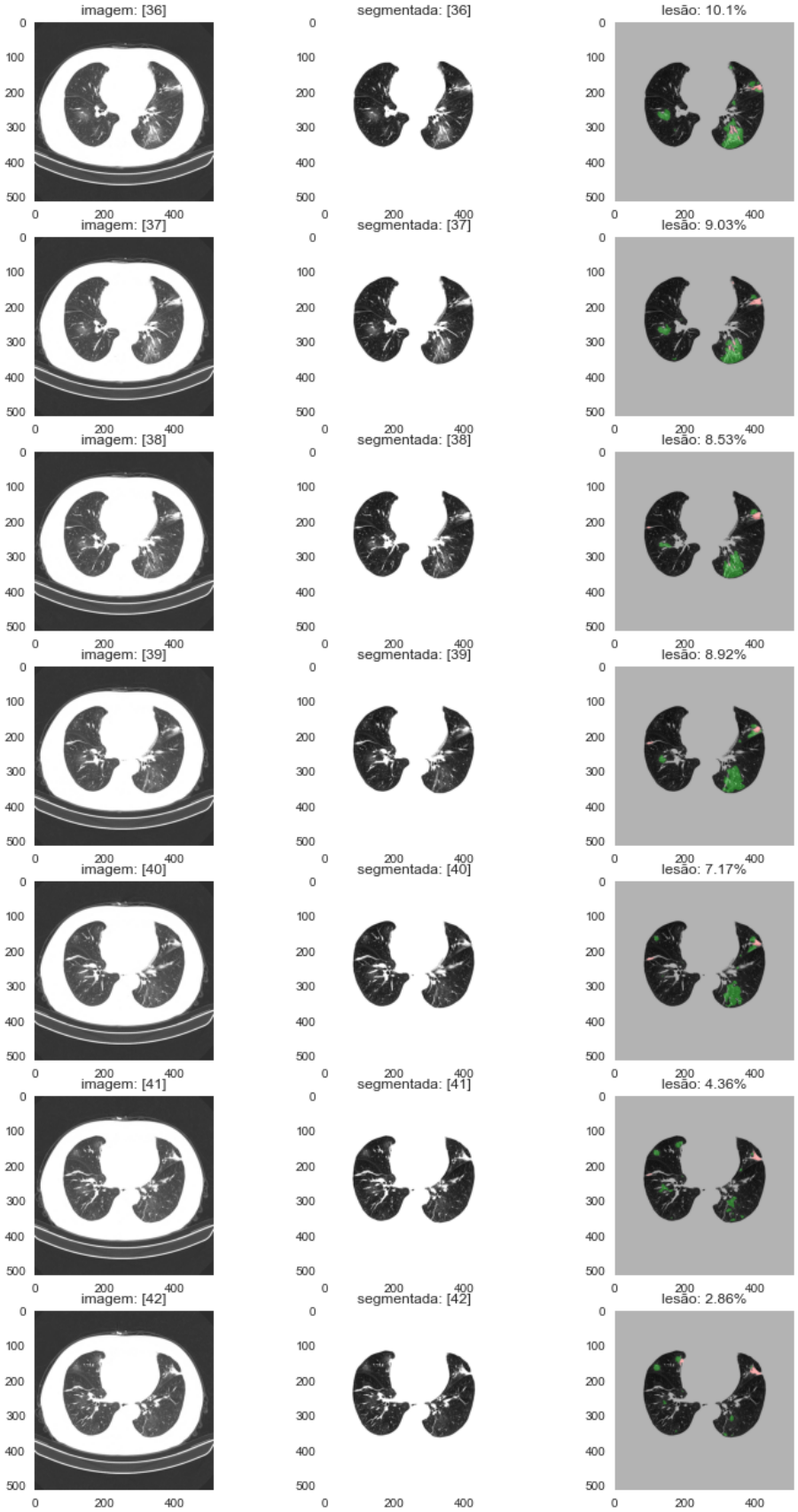
Fonte: Arquivo pessoal do Autor.

Figura A.23 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



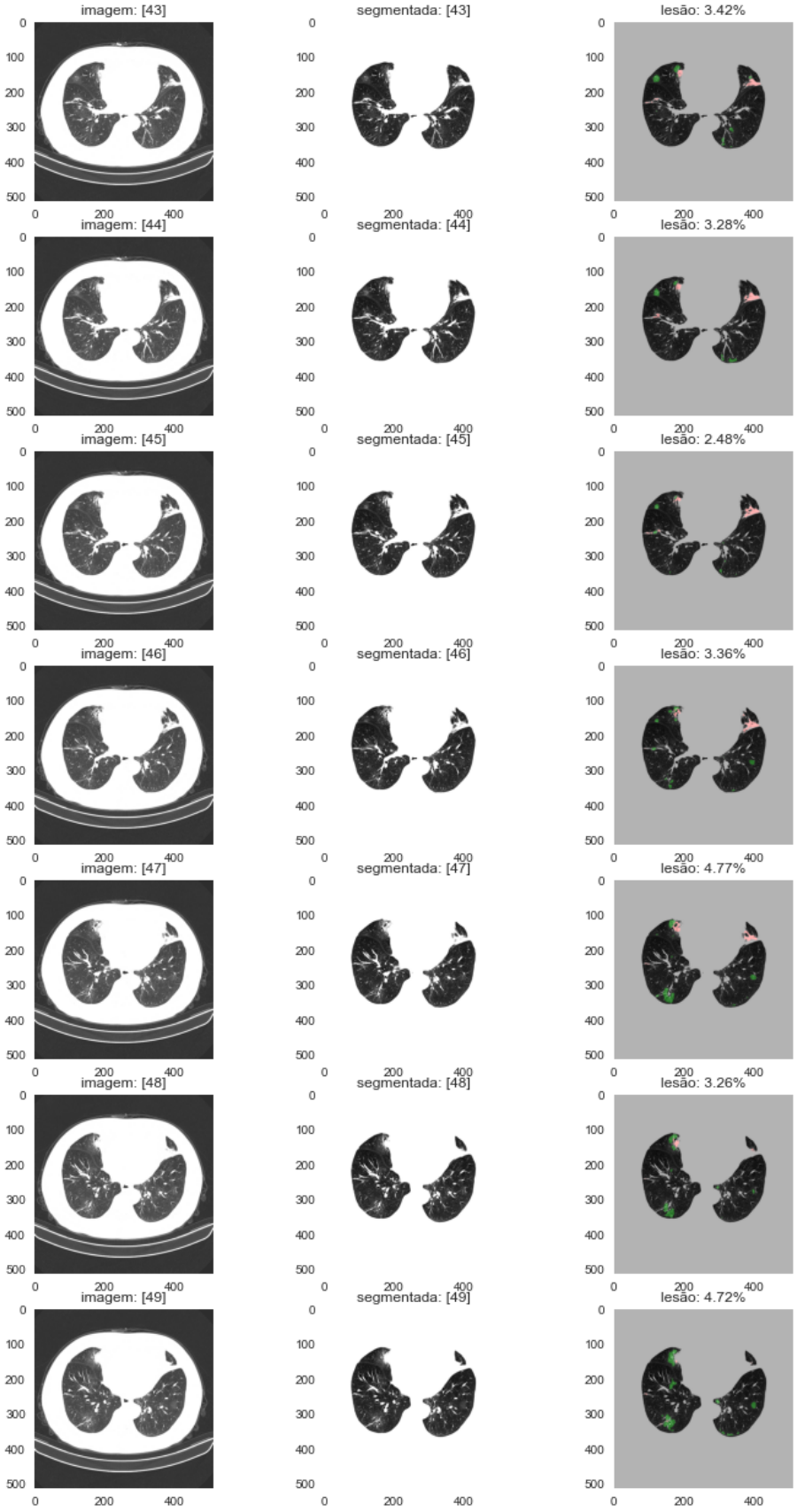
Fonte: Arquivo pessoal do Autor.

Figura A.24 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



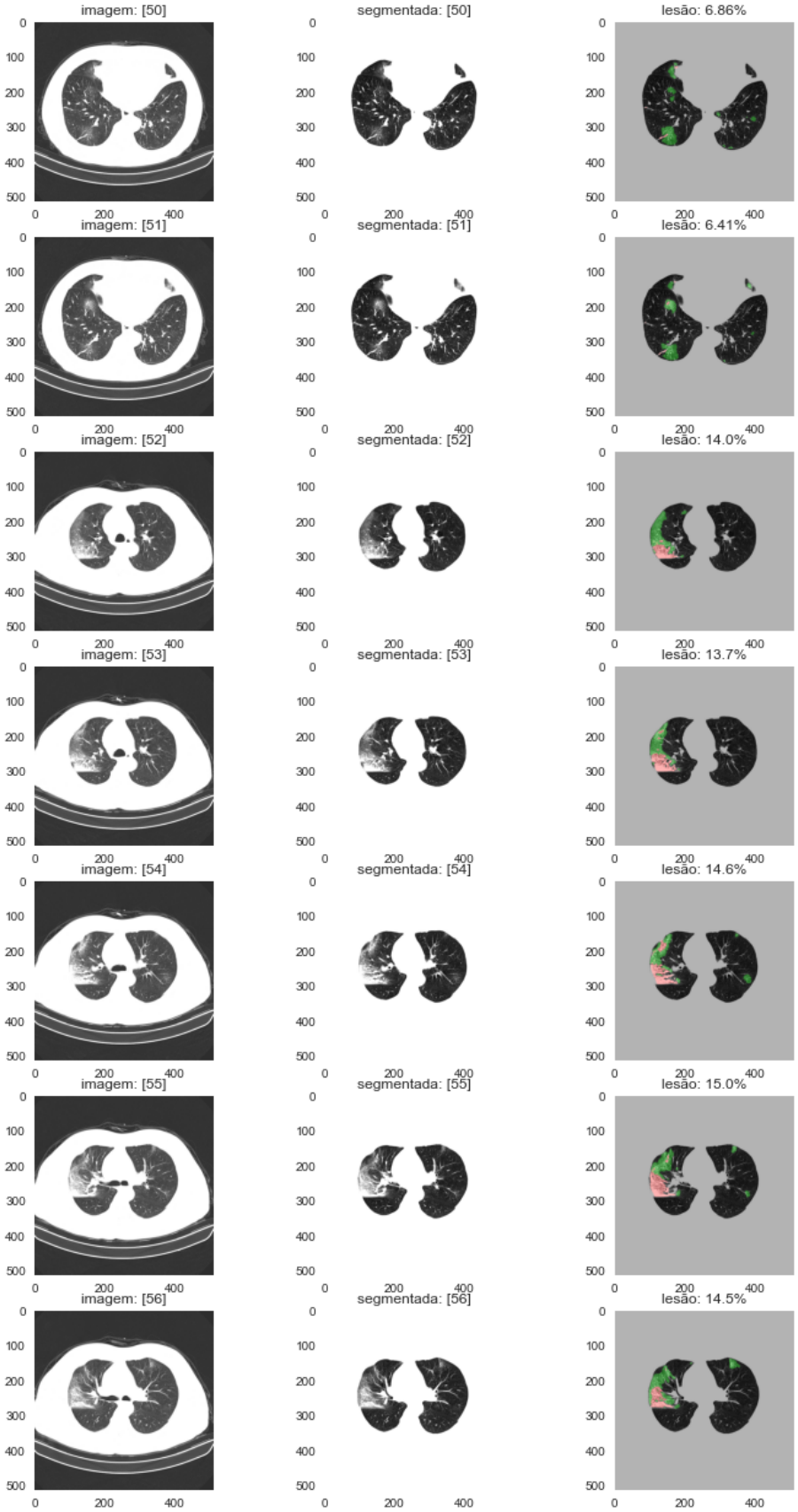
Fonte: Arquivo pessoal do Autor.

Figura A.25 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



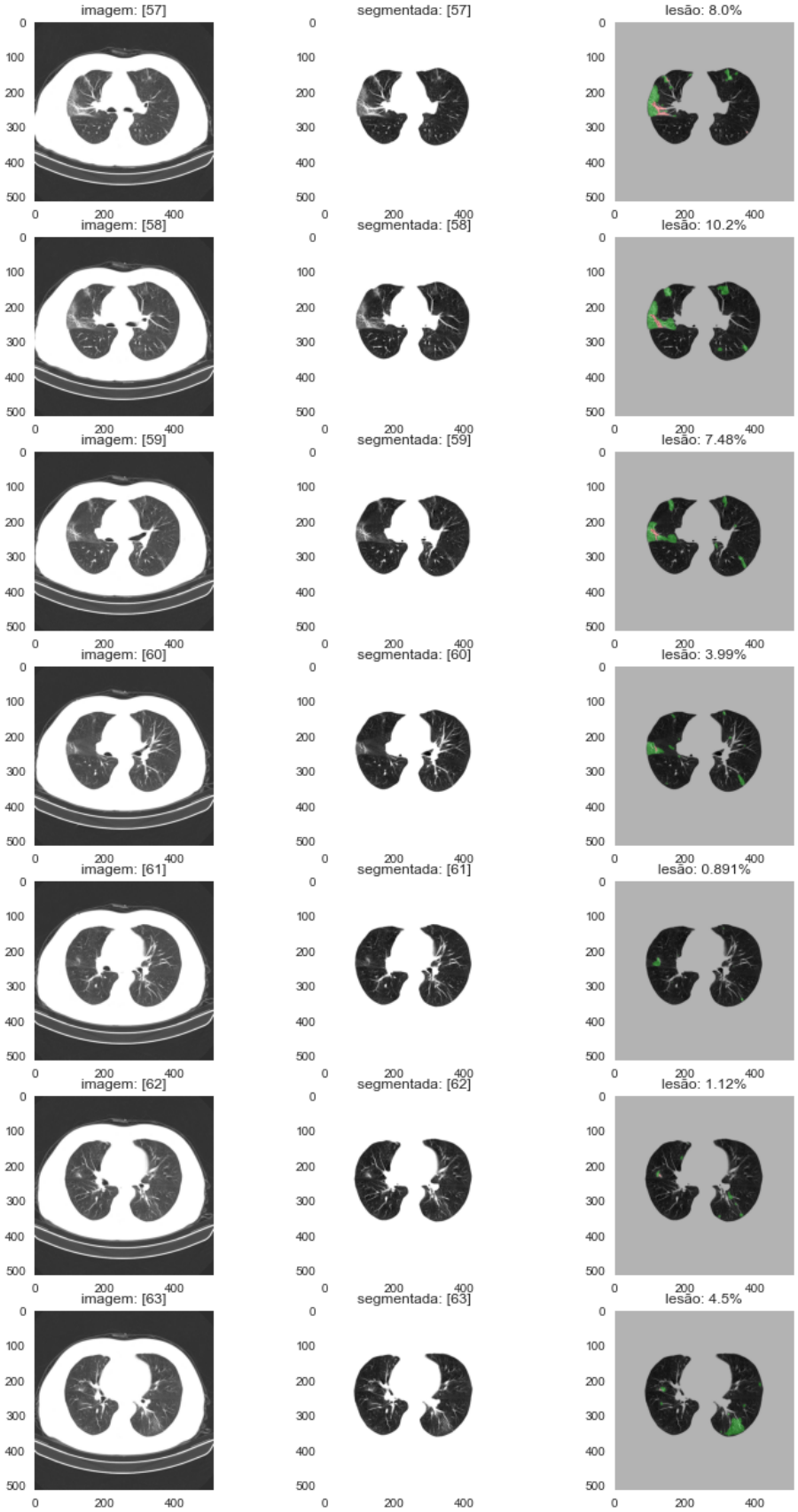
Fonte: Arquivo pessoal do Autor.

Figura A.26 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



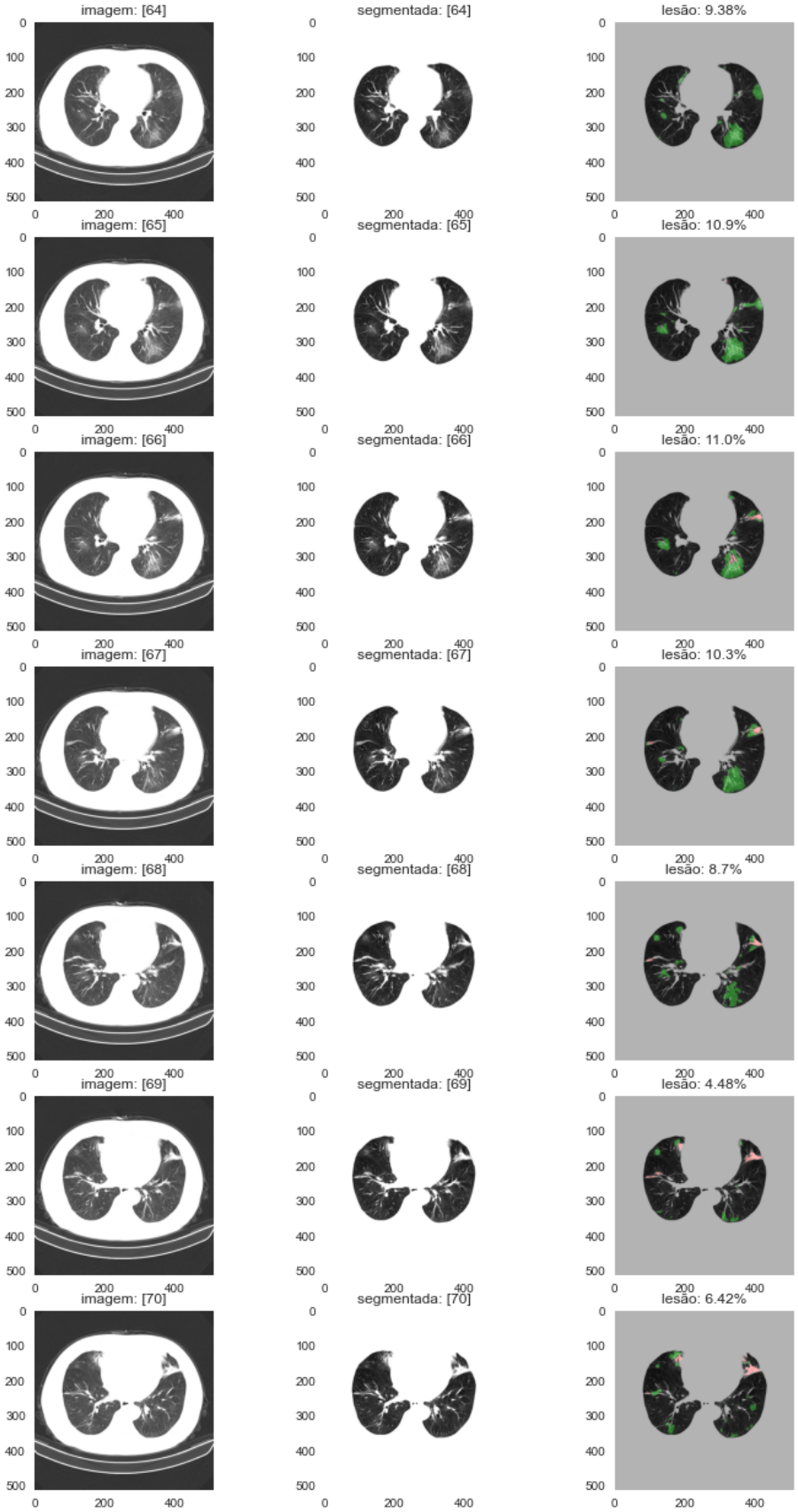
Fonte: Arquivo pessoal do Autor.

Figura A.27 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



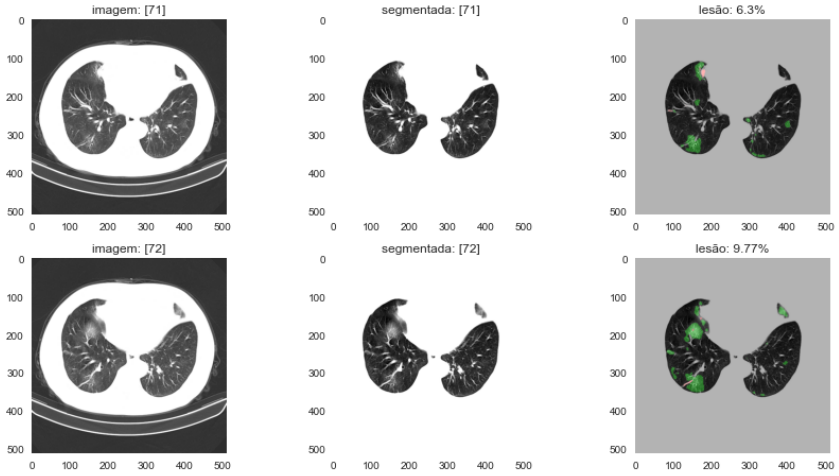
Fonte: Arquivo pessoal do Autor.

Figura A.28 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



Fonte: Arquivo pessoal do Autor.

Figura A.29 – Análise de severidade de lesão pulmonar de um paciente, cada linha (imagem, mascara e lesão) correspondem a um *slice*.



Fonte: Arquivo pessoal do Autor.