CENTRO UNIVERSITÁRIO IESB
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA
CIÊNCIA DA COMPUTAÇÃO

RODRIGO TOURIÑO MACHADO DE OLIVEIRA LIMA

ARQUITETURA BASEADA EM REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO AUTOMÁTICA DA COVID-19 USANDO IMAGENS DE RAIO X

BRASÍLIA – DF
2021

RODRIGO TOURIÑO MACHADO DE OLIVEIRA LIMA

ARQUITETURA BASEADA EM REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO AUTOMÁTICA DA COVID-19 USANDO IMAGENS DE RAIO X

Trabalho de Conclusão de Curso apresentado como pré-requisito para obtenção da graduação em Ciência da Computação pelo Centro Universitário Instituto de Educação Superior de Brasília.

Orientadora: Profa. Dra. Leticia Toledo Maia Zoby.

BRASÍLIA – DF

2021

RODRIGO TOURIÑO MACHADO DE OLIVEIRA LIMA

ARQUITETURA BASEADA EM REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO AUTOMÁTICA DA COVID-19 USANDO IMAGENS DE RAIO X

Trabalho de Conclusão de Curso apresentado como pré-requisito para obtenção da graduação em Ciência da Computação pelo Centro Universitário Instituto de Educação Superior de Brasília.

Orientadora: Profa. Dra. Leticia Toledo Maia Zoby.

Brasília, 20/05/2021

Banca Examinadora:

_____

Profa. Dra. Leticia Toledo Maia Zoby

_____

Prof. Dr. Thiago Raposo Milhomem de Carvalho

_____

Prof. Me. Jorge Osvaldo Alves de Lima Torres

AGRADECIMENTOS

Primeiramente, gostaria de agradecer à minha família por todo o apoio e carinho ao longo desta jornada.

Gostaria de agradecer à minha namorada Marina, que sempre me incentivou e me ajudou a ser uma pessoa melhor.

Também gostaria de agradecer à minha professora e orientadora Leticia Toledo Maia Zoby por todos os ensinamentos ao longo destes quatro anos de curso.

À coordenadora do curso de Ciência da Computação, Patrícia Moscariello, que nunca deixou de me apoiar desde o meu primeiro dia de IESB.

Aos professores e às professoras do curso de Ciência da Computação por toda a dedicação e a sabedoria que vocês tiveram para nos ensinar.

Aos meus colegas que tive o prazer de compartilhar momentos tão especiais nesta instituição de ensino.

E finalmente, à comunidade acadêmica por todas as contribuições que, indubitavelmente, foram essenciais para o desenvolvimento deste trabalho.

Wir müssen wissen. Wir werden wissen.

David Hilbert

RESUMO

O surto da doença causada pelo novo coronavírus (COVID-19) representa uma emergência de saúde em escala global. Para coibir o contágio e tratar infectados, é de suma importância que haja métodos eficazes na detecção do vírus. O teste RT-PCR tem sido adotado por autoridades de saúde, mas apresenta possíveis falhas quanto à sua sensibilidade e estabilidade. Além disso, com o uso de exames de imagem, médicos radiologistas têm observado características distintas na anatomia pulmonar de pacientes com COVID-19. Considerando isto, pesquisadores estão avaliando o desempenho diagnóstico de radiografias para a detecção da COVID-19. Incorporando ferramentas de inteligência artificial que automatizam o processo de classificação e identificação de características da doença poderia reduzir a quantidade de diagnósticos errôneos e aumentar a confiabilidade dos resultados. Com isso, o trabalho elaborou uma aplicação *web* que recebe como parâmetros de entrada imagens de raio X de pacientes com suspeita de COVID-19 e, utilizando técnicas de aprendizagem profunda, classifica como COVID positivo ou COVID negativo. Com o intuito de maximizar o desempenho do modelo de aprendizagem de máquina, foram comparadas três arquiteturas pré-treinadas (DenseNet121, ResNet50V2 e VGG16) com dois tipos de procedimentos de pré-processamento (sem segmentação das regiões de interesse e com segmentação). Além disso, cada classificador testado teve seus hiperparâmetros de taxa de aprendizagem e número de épocas otimizados ao longo do treinamento usando o conjunto de amostras de validação. Verificou-se que a configuração ResNet121 com pré-processamento sem segmentação das regiões de interesse foi a que obteve a melhor acurácia na tarefa de identificação da doença, com 94,5%, sendo então a escolhida para uso no sistema *web*. Com isso, esta aplicação se propõe a ser mais uma ferramenta no combate contra a COVID-19, podendo auxiliar profissionais da saúde na avaliação de exames radiológicos para o diagnóstico da doença.

Palavras-chave: COVID-19. Redes Neurais Convolucionais. Raio X.

ABSTRACT

The outbreak of the disease caused by the new coronavirus (COVID-19) represents a health crisis of global proportions. In order to curb contagion and treat infected people, effective methods for detecting the virus are of vital importance. Health authorities worldwide have adopted the RT-PCR test, but it has possible flaws regarding its sensitivity and stability. In addition, with the use of imaging tests, radiologists have observed distinct characteristics in the pulmonary anatomy of patients with COVID-19. Considering this, researchers are evaluating the diagnostic performance of using X-ray images for the detection of COVID-19. Incorporating artificial intelligence tools that automate the process of classifying and identifying disease characteristics could reduce the amount of misdiagnoses and increase the reliability of the results. As such, the present work developed a web application that receives as input parameters X-ray images of patients with suspected COVID-19 and, using deep learning techniques, classifies them as either COVID positive or COVID negative. In order to maximize the performance of the machine learning model, three pre-trained architectures (DenseNet121, ResNet50V2 and VGG16) were compared with two types of pre-processing procedures (without segmentation of the regions of interest and with segmentation). In addition, each classifier tested had its learning rate and number of epochs optimized throughout training using the validation set. The ResNet121 architecture with pre-processing without segmentation of the regions of interest obtained the best accuracy in the disease identification task, with 94.5%, being the chosen combination for the web application. Therefore, this tool could help health professionals in the evaluation of radiological exams for the diagnosis of the disease.

Keywords: COVID-19. Convolutional Neural Networks. X-Rays.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	Antero-Posterior
API	Interface de Programação de Aplicação
COVID-19	Doença causada pelo SARS-CoV-2
CPU	Unidade Central de Processamento
CUDA	Compute Unified Device Architecture
DL	Aprendizado Profundo
EAM	Erro Absoluto Médio
ELU	Unidade Linear Exponencial
EQM	Erro Quadrático Médio
GPU	Unidade de Processamento Gráfico
IA	Inteligência Artificial
ML	Aprendizagem de Máquina
MLP	Perceptron Multicamadas
OMS	Organização Mundial da Saúde
OpenCV	Open Source Computer Vision Library
PA	Póstero-Anterior
ReLU	Unidade Linear Retificada
RL	Aprendizagem por Reforço
RNA	Rede Neural Artificial
RNC	Rede Neural Convolucional
RT-PCR	Transcrição Reversa Seguida de Reação em Cadeia da Polimerase
SARS-CoV-2	Coronavírus da Síndrome Respiratória Aguda Grave 2
SL	Aprendizagem Supervisionada
SUS	Sistema Único de Saúde
Tanh	Tangente Hiperbólica
TL	Transferência de Aprendizado
UL	Aprendizagem Não Supervisionada
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

O surto da doença causada pelo novo coronavírus (COVID-19) representa uma emergência de saúde em escala global. Em 11 de março de 2020, a Organização Mundial da Saúde (OMS) caracterizou a COVID-19 como uma pandemia, tendo em vista sua severidade e o ritmo acelerado de disseminação (GHEBREYESUS, 2020). Em maio de 2021, dados extraídos do painel de monitoramento em tempo real da COVID-19 do Johns Hopkins Coronavirus Resource Center (2021) apontaram que havia mais de 162 milhões de casos, com o vírus estando presente em 192 países. Desses, 3.361.016 casos foram a óbito. Neste mesmo levantamento, o Brasil foi o país com o terceiro maior número de casos (15,5 milhões ou ~9,6%) e com o segundo maior número de mortes (432.628 ou ~12,9%).

Para coibir o contágio e tratar infectados, é de suma importância que haja métodos eficazes na detecção do vírus. Corman *et al*. (2020) propõem que o diagnóstico da COVID-19 seja realizado por meio do teste transcrição reversa seguida de reação em cadeia da polimerase (RT-PCR), que verifica a presença do ácido ribonucleico do vírus na secreção nasofaríngea. Esta metodologia tem sido adotada por autoridades de saúde, mas apresenta possíveis falhas quanto à sua sensibilidade.

No estudo realizado por Li *et al*. (2020) avaliando o RT-PCR com 610 pacientes, 384 (63%) testaram negativo em um primeiro momento. Destes, 48 (12,5%) receberam resultados positivos no segundo teste. Além disso, os autores constataram uma preocupação com relação à estabilidade do RT-PCR. 17 pacientes que inicialmente testaram positivo, após alguns dias de tratamento obtiveram resultados negativos. Porém, em uma terceira coleta, mesmo com a melhora dos sintomas, estes 17 voltaram a testar positivo. Xiao, Tong e Zhang (2020) também observaram um alto índice de falso negativo nos testes com RT-PCR, onde 15 (21.4%) de 70 pacientes testaram positivo após dois resultados negativos consecutivos.

Com o uso de exames de imagem, médicos radiologistas têm observado características distintas na anatomia pulmonar de pacientes com COVID-19. De acordo com Xie *et al*. (2020), as principais mudanças constatadas são opacidades em vidro fosco e consolidações (acúmulo de fluído e/ou tecido). Exames de imagem do tórax também são ferramentas indicadas para a detecção de diferentes doenças como câncer, pneumonia e fibrose cística (WIELPÜTZ *et al*., 2014).

Em um estudo envolvendo 51 pacientes com COVID-19, Fang *et al*. (2020) compararam a sensibilidade da tomografia computadorizada e do RT-PCR. Na primeira leva

de testes, a tomografia identificou anormalidades em 50 dos 51 (98%) e o RT-PCR detectou 36 dos 51 (71%). Porém, a tomografia introduz desafios que inibem sua aplicação em larga escala no Brasil para o diagnóstico da doença. Em média, tomografias expõem um paciente à 350 vezes mais radiação que radiografias, o que está associado a uma maior incidência de cânceres (CHODICK *et al*., 2009). Além disso, um levantamento em 2015 analisando a média de tomógrafos do Sistema Único de Saúde (SUS) por 100 mil habitantes demonstrou uma baixa disponibilidade de aparelhos nas regiões Norte e Nordeste do país, com 0,63 e 0,66 respectivamente (SILVA, 2017).

Schiaffino *et al*. (2020) avaliaram o desempenho diagnóstico do uso de imagens de raio X para a detecção da COVID-19 em um estudo com 535 pacientes. A metodologia produziu resultados próximos ao desempenho da tomografia computadorizada, com 89% de sensitividade e especificidade de 60,6%. Além disso, os autores constataram que a capacidade preditiva poderia ser melhorada incorporando ferramentas de inteligência artificial (IA) que automatizam o processo de classificação e identificação de características da doença.

A descoberta automática de lesões e regiões de interesse pode ser realizada empregando técnicas de aprendizado profundo (DL). Devido aos avanços no poder de processamento de sistemas computacionais modernos, no desenvolvimento de algoritmos e na facilidade de acesso à grandes massas de dados, o uso de redes neurais convolucionais (RNCs) na medicina tem sido uma área de pesquisa bastante explorada, com estudos produzindo resultados tão bons quanto ou até melhores que especialistas humanos (MAZUROWSKI *et al*., 2018).

Com isso, o trabalho busca responder: Como desenvolver uma arquitetura baseada em RNC para a detecção automática da COVID-19 usando imagens de raio X?

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral é elaborar uma aplicação *web* que receba como parâmetros de entrada imagens de raio X de pacientes com suspeita de COVID-19 e, utilizando técnicas de DL, classificar como COVID positivo ou COVID negativo.

1.1.2 Objetivos Específicos

Define-se como os objetivos específicos a escolha da arquitetura de RNC com melhor desempenho para a tarefa de detecção da doença, assim como os hiperparâmetros ideais para otimizar seu poder preditivo, a definição dos procedimentos de pré-processamento de imagem que maximizam a capacidade diagnóstica do modelo e a construção de uma aplicação *web* para implementar esta ferramenta.

1.2 ORGANIZAÇÃO

O restante do trabalho está dividido em mais cinco capítulos. O segundo capítulo contém uma revisão da literatura relevante, abordando os tópicos COVID-19, radiografias e seu uso na detecção da doença, aprendizagem de máquina (ML) e seus subtópicos, processamento de imagens digitais, trabalhos correlatos e as ferramentas empregadas na construção da arquitetura proposta. Já o terceiro capítulo detalha a metodologia abordada na construção do trabalho. Na seção seguinte, são mostrados detalhes de implementação. O capítulo cinco apresenta e discute os resultados obtidos. Finalmente, o capítulo seis sintetiza o trabalho e aponta as limitações de pesquisa e ideias para trabalhos futuros.

2 REFERENCIAL TEÓRICO

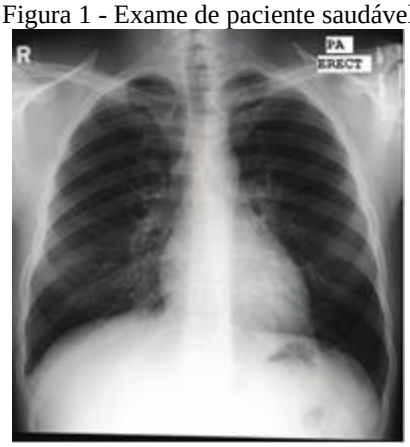
2.1 COVID-19

A COVID-19 é uma doença transmissível causada pelo coronavírus da síndrome respiratória aguda grave 2 (SARS-CoV-2) que teve o seu primeiro diagnóstico em seres humanos na cidade de Wuhan, na China, em dezembro de 2019 (WU; LEUNG; LEUNG, 2020). Desde então, o vírus tem se apresentado altamente contagioso, com estimativas de seu R0, o número básico de reprodução, sendo de 5,7 (SANCHE *et al*., 2020).

De acordo com a OMS (ORGANIZAÇÃO MUNDIAL DA SAÚDE, 2020), em cerca de 80% dos casos, pacientes apresentam sintomas leves a moderados, como febre, tosse seca e fadiga, e não necessitam de cuidados hospitalares. Os restantes 20% manifestam sintomas graves, como falta de ar e dores no peito, o que pode evoluir para um estado crítico que exige o emprego de suporte ventilatório para amenizar a insuficiência respiratória provocada pela doença.

Pacientes em estados mais graves têm desenvolvido complicações relacionadas ao sistema respiratório, como pneumonia em ambos os pulmões, o acúmulo de muco e fluido nos sacos aéreos e síndrome respiratória aguda grave (ARCHER; SHARP; WIER, 2020). Lesões mais agressivas podem acarretar em um comprometimento pulmonar irreversível. Uma autópsia conduzida em um estudo por Xu *et al*. (2020) de um paciente de COVID-19, que faleceu após uma piora no quadro clínico devido a uma infecção grave, revelou dano bilateral difuso nos alvéolos pulmonares, que são pequenas estruturas responsáveis pelo fornecimento de oxigênio ao sangue.

Figura 1 - Exame de paciente saudável



Fonte: Jaegar (2015)

Figura 2 - Exame de paciente com COVID-19

Fonte: Foust *et al.* (2020)

A Figura 1 é uma radiografia de um paciente com pulmões saudáveis e sem sinais de infecção. Já a Figura 2 demonstra os pulmões de um paciente com a COVID-19. Foust *et al.* (2020) observam que é possível verificar no exame do paciente doente opacidades em vidro fosco com distribuição periférica (asterisco) e central (seta).

2.2 RADIOGRAFIAS

A radiografia é um tipo de exame de imagem baseado no uso de radiação eletromagnética para visualizar uma representação bidimensional da estrutura interna de um objeto. De acordo com Brant e Helms (2007), comparados à luz visível, raios X possuem um comprimento de onda menor, possibilitando que os raios atravessem a maioria dos tecidos no corpo humano. Logo, partes com maior densidade radiográfica, como ossos, absorvem mais

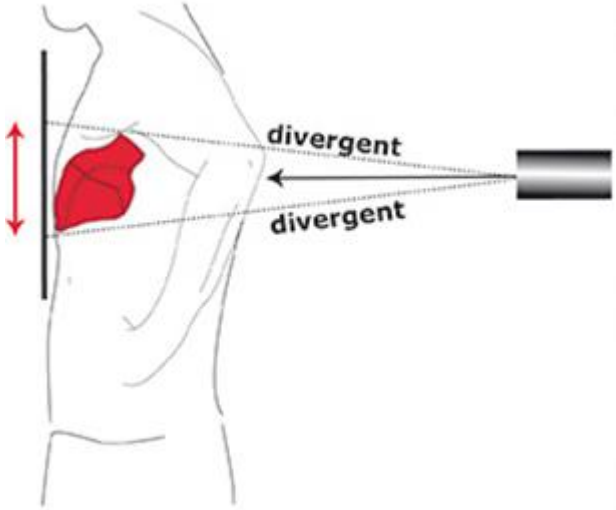
fótons e estruturas com menos densidade, como músculo e gordura, permitem que os fótons atravessem.

Para realizar um exame, o que se deseja observar é posicionado entre uma fonte e um detector. Os raios são gerados e absorvidos em diferentes quantias. O filme produzido é um negativo. Logo, quanto mais radiação atravessa os tecidos observados e chega ao detector em um determinado ponto, mais escuro aquele ponto é na imagem resultante. Assim, cria-se um contraste entre as partes de um corpo, tornado a radiografia uma técnica capaz de mapear sua organização estrutural (BRANT; HELMS, 2007).

Radiografias do tórax produzem imagens que permitem avaliar, dentre outros elementos, os pulmões. Segundo Wada, Rodrigues e Santos (2019), comparada a outros tipos de exames de imagem, a radiografia é a forma mais accessível e menos custosa de analisar o interior do tórax de um paciente. Além disso, em situações onde um paciente não pode ser facilmente transportado dentro de uma unidade de saúde, é possível realizar o exame utilizando uma versão móvel do aparelho.

Sobre as metodologias empregadas na execução desta técnica de imagiologia médica, Wada, Rodrigues e Santos (2019) avaliam que a projeção póstero-anterior (PA) e em perfil, onde o paciente fica de pé e com as costas viradas para a fonte de raio X, é a forma protocolar adotada. Isto porque é a posição que rende as imagens mais claras e interpretáveis, onde os principais elementos observados estão representados com maior nitidez e fidelidade. Como pode ser visto na Figura 3, nesta incidência o coração está mais distante do centro de projeção, mitigando a ampliação do órgão na imagem resultante. Além disso, as escápulas do paciente interferem menos na visualização dos pulmões.

Figura 3 - Raio X de tórax na projeção PA



Fonte: Plas (2016)

Porém, em situações específicas como confinamento ao leito, pacientes com mobilidade reduzida e suspeita de certas doenças, a radiografia pode ser executada usando diferentes posições. Projeções alternativas incluem a antero-posterior (AP), a decúbito lateral com raios horizontais, a apicolordótica, entre outros (WADA; RODRIGUES; SANTOS, 2019).

2.3 APRENDIZAGEM DE MÁQUINA

ML é um subcampo da IA que contempla algoritmos que aprendem a realizar uma determinada tarefa sem que haja uma programação explícita. Isto é feito extraindo conhecimento de uma massa de dados para modelar o comportamento subjacente. Diferentemente, na programação clássica o programador é responsável por fornecer todos os passos a serem desempenhados. Em domínios onde uma modelagem explícita é demasiadamente complexa, como nos campos de visão computacional e reconhecimento de fala, torna-se útil uma abordagem baseada em algoritmos que possuem a capacidade de aprender automaticamente por meio da experiência (CHOLLET, 2018).

Existem três categorias principais de algoritmos de ML: aprendizagem supervisionada (SL), onde o modelo recebe as entradas com suas respectivas respostas e procura aprender a regra que realiza este mapeamento para novos dados, aprendizagem não supervisionada (UL), no qual há apenas as entradas e busca-se encontrar uma estrutura nos dados, e aprendizagem por reforço (RL), em que o programa recebe informações sobre seu ambiente e visa maximizar o *feedback* positivo enquanto minimiza o negativo (MÜLLER; GUIDO, 2018). Na medicina, o

diagnóstico automatizado, por se tratar de um processo que envolve identificar a presença ou não de alguma doença, está mais ligado à SL.

2.3.1 Aprendizagem Supervisionada

De acordo com Chollet (2018), a SL depende de três elementos fundamentais, sendo eles os dados de entrada, as saídas esperadas e uma forma de avaliar o desempenho do algoritmo. Assim, ao processar os exemplos e ter sua execução mensurada, o modelo realiza ajustes que visam diminuir a incidência de erros. Diz-se, então, que o estimador "aprendeu" ao otimizar sua capacidade de prever as respostas corretas.

Müller e Guido (2018) apontam que os problemas de SL podem ser separados entre problemas de classificação e problemas de regressão. Na classificação, para cada entrada o programa designa um rótulo pertencente a um conjunto discreto de classes possíveis. Já na regressão, as saídas são valores dentro de um contradomínio contínuo.

O tipo de problema abordado também influi nas métricas de avaliação aplicadas. Quando se trata de valores contínuos, há duas métricas comumente empregadas: erro quadrático médio (EQM), descrito na Equação 1, e erro absoluto médio (EAM), como visto na Equação 2. Em ambas, os termos Y_i e $\hat{Y}_i$ representam a resposta correta e o valor estimado, respectivamente. As métricas descritas nas Equações 3, 4 e 5 são usadas em cenários discretos (HANDELMAN *et al*., 2020)

$$EQM = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{1}$$

$$EAM = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{2}$$

$$Acurácia = \frac{Verdadeiros\ Positivos + Verdadeiros\ Negativos}{Total} \tag{3}$$

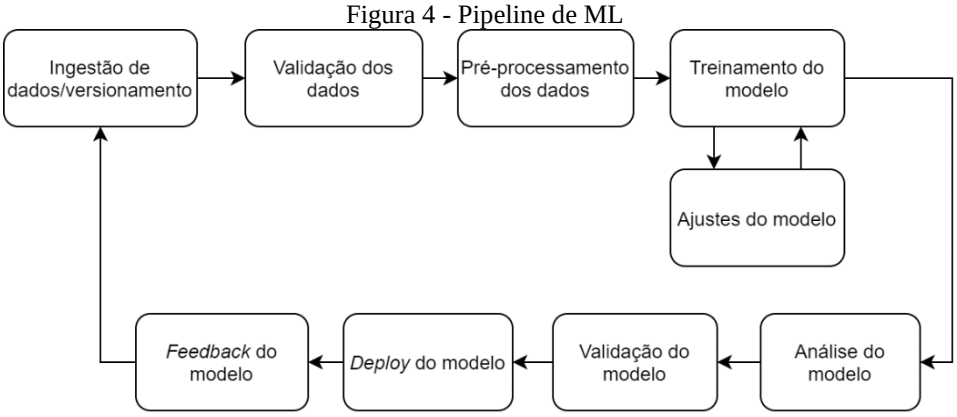
$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos} \tag{4}$$

$$Revocação = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos} \quad (5)$$

2.3.2 O Ciclo de Aprendizagem de Máquina

A correta implementação de um procedimento de ML depende da adesão a um roteiro padronizado. Hapke e Nelson (2020) descrevem os passos a serem seguidos na forma de um ciclo composto por 8 etapas. O *pipeline* proposto, como visto na Figura 4, tem como objetivo principal orquestrar todos os procedimentos, desde a coleta dos dados até a publicação do modelo para seus usuários.

Figura 4 - Pipeline de ML



Fonte: Adaptada de Hapke e Nelson (2020)

O ponto de partida se dá com a aquisição dos dados e seu armazenamento em estruturas de dados adequadas. O segundo passo consiste em validar os dados, garantindo que não há anormalidades presentes que possam deturpar os resultados, como determinadas classes estando super-representadas. Depois, realiza-se o pré-processamento dos dados onde ocorre operações como padronização, conversão de tipo e adequação para o modelo escolhido. Na quarta etapa, o modelo de ML recebe os dados pré-processados para realizar seu treinamento, visando minimizar a taxa de erro. Após este procedimento, analisa-se o desempenho do estimador por meio de diferentes métricas de avaliação. O próximo passo envolve validar o modelo e gerenciar o seu versionamento, visando prepará-lo para implantação. A sétima etapa é onde se disponibiliza para uso o que foi construído ao longo de todo o *pipeline*. Finalmente, a etapa de número oito fecha o ciclo ao mensurar a efetividade do modelo em produção, reincorporando este *feedback* nas versões subsequentes para que se tenha uma melhora contínua (HAPKE; NELSON, 2020).

2.3.3 *Overfitting*

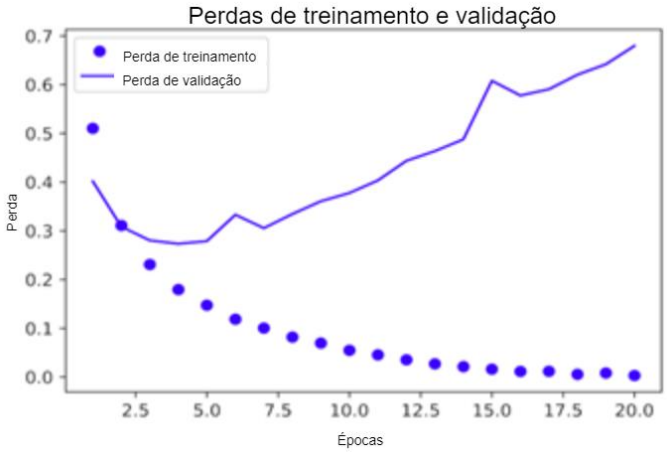
Um dos problemas centrais em ML é o *overfitting*. Isto ocorre quando um modelo aprende detalhes dos dados de treinamento que não estão presentes na população como um todo. Assim, o modelo perde sua capacidade de generalizar para dados nunca antes vistos (CHOLLET, 2018).

Provost e Fawcett (2013) avaliam que quanto mais flexível for um modelo, mais rapidamente ele aprende os padrões presentes nos dados. Porém, modelos mais complexos tendem a sofrer de *overfitting* com maior facilidade. Logo, o ideal é que se busque um equilíbrio entre otimização e generalização.

Para verificar a presença deste problema, reserva-se parte do conjunto de dados de treinamento para validação. Quando há uma grande quantia de dados ou quando o tempo de treinamento é um fator prioritário, isto pode ser feito aplicando o método *holdout*, onde se realiza uma simples divisão do conjunto de dados de treinamento em dois subconjuntos menores. Se há uma limitação na quantidade de dados e não há maiores preocupações com o tempo gasto na etapa de treinamento, emprega-se o método *k-fold*. Nele, o conjunto de dados de treinamento é dividido em *k* partições. Usando uma estrutura de repetição, *k* cópias do modelo são treinadas, cada uma usando uma partição diferente para validação e as restantes *k-1* para aprendizado. Calcula-se então a média dos erros, que é mais representativa do que uma única observação (CHOLLET, 2018).

Estas amostras de validação são usadas para mensurar o erro ao longo do treinamento em dados que não influenciam no cálculo dos parâmetros. Como se pode observar na Figura 5, os erros de validação e treinamento diminuem no começo, mas o erro de validação logo estabiliza e depois passa a aumentar. Quando isto ocorre, é possível avaliar que o modelo apresentou *overfitting*.

Figura 5 - Perdas de treinamento e validação



Fonte: Adaptada de Chollet (2018)

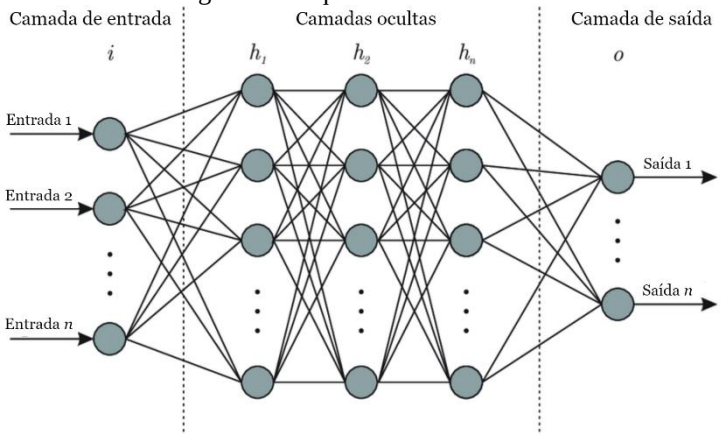
Para diminuir os efeitos de *overfitting*, adotam-se algumas medidas nas fases de projeto e aprendizagem. Primeiramente, expor um modelo a mais dados de treinamento fará com que o mesmo tenha uma capacidade aumentada de generalização. Além disso, restringir sua complexidade ao diminuir o número de parâmetros controla a velocidade com que se chega em um estado de *overfitting*. Também é possível regular a influência que os parâmetros exercem aplicando penalidades quando seus valores absolutos crescem demais (PROVOST; FAWCETT, 2013).

2.3.4 Redes Neurais Artificiais

Uma rede neural artificial (RNA) é um sistema computacional inspirado no funcionamento do cérebro humano. Este modelo é estruturado em camadas interligadas de neurônios, a unidade básica de uma RNA capaz de armazenar algum valor. Para que um neurônio "acenda", é preciso que uma combinação de sinais provenientes de outros neurônios gere um valor que ultrapasse um determinado limiar (HAYKIN, 2009).

Uma rede é organizada conforme a Figura 6, com uma camada de entrada, uma camada de saída e zero ou mais camadas ocultas. Haykin (2009) define que a camada de entrada é responsável por receber os valores iniciais a serem processados. Já a camada de saída apresenta o resultado final gerado pelo sistema. As camadas ocultas são usadas para realizar transformações nos dados de forma que podem ser utilizados pela camada de saída para efetuar alguma predição.

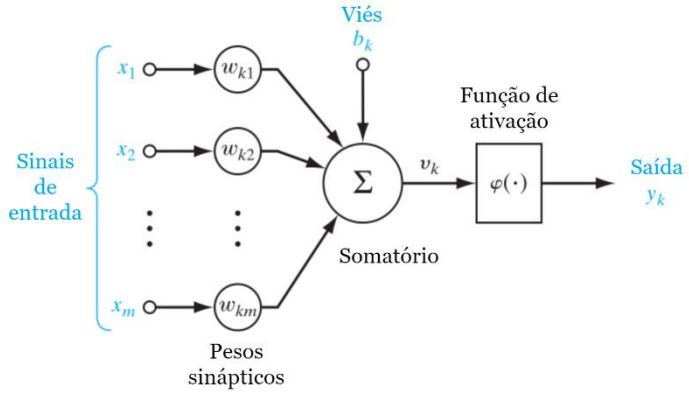
Figura 6 - Arquitetura de uma RNA



Fonte: Adaptada de Bre, Gimenez e Fachinotti (2018)

Uma ligação conecta dois neurônios, um transmissor que envia um sinal de saída e um receptor que recebe aquele sinal como entrada. Este valor é transmitido e ajustado pelo peso da conexão, que pode ser tanto positivo quanto negativo. Quanto maior for o peso, em termos absolutos, mais influente se torna aquela conexão no valor final do neurônio receptor. Logo, o valor de um determinado neurônio depende de todas as ligações feitas a ele (JAIN; MAO; MOHIUDDIN, 1996).

Figura 7 - Cálculo do valor de um neurônio



Fonte: Adaptada de Haykin (2009)

A Figura 7 demonstra como o valor que um neurônio armazena é calculado. Primeiro, realiza-se o produto escalar entre os pesos *w* e as entradas *x*. Com isso, soma-se (opcionalmente) uma constante *b*, que representa um viés. Finalmente, uma função de ativação é aplicada para definir a saída daquele neurônio. Isto pode ser expressado conforme a Equação 6.

$$y = f\left(b + \sum_{i=1}^{m} w_i\, x_i\right) \tag{6}$$

Segundo Sharma, Sharma e Athaiya (2020), as funções de ativação são usadas para restringir o conjunto de valores possíveis dos neurônios e também para efetuar transformações não-lineares nos dados. Algumas delas incluem: função de limiar, função sigmoide, tangente hiperbólica (Tanh), unidade linear retificada (ReLU) e unidade linear exponencial (ELU). Conforme descrito por Krizhevsky, Sutskever e Hinton (2017), a ReLU é preferível quando o modelo é complexo e a base de dados contém muitos registros. Isto porque esta função de ativação proporciona um alto ganho de desempenho no processo de treinamento de uma rede, exigindo menos iterações para atingir os resultados esperados. A Equação 7 descreve a ReLU:

$$f(x) = \max(0, x) \tag{7}$$

Goodfellow, Bengio e Courville (2017) expõem que o aprendizado em uma RNA é um processo iterativo feito em épocas. Inicialmente, atribuem-se de forma aleatória os pesos de todas conexões da rede. A partir disto começa o procedimento, onde a cada época a rede recebe um conjunto de amostras e efetua um ciclo de processamento. Os dados são transformados, camada por camada, até que se chegue em uma representação final. Com isso, os valores calculados são comparados às saídas esperadas, gerando uma taxa de erro para aquela época. A partir desta taxa de erro, os pesos da rede são reajustados usando o algoritmo de retropropagação de erro.

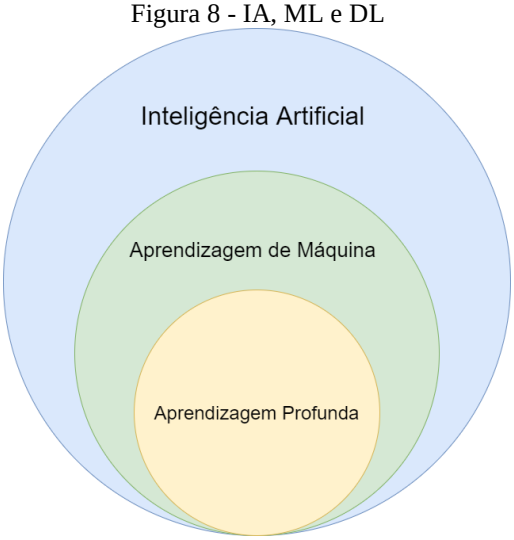
O procedimento pode ser expresso conforme a Equação 8. De acordo com Werbos (1990), o algoritmo de retropropagação de erro calcula o gradiente da função de erro *E* com respeito aos pesos W_k, apontando a direção cuja função mais crescerá em um determinado ponto. Como o objetivo é minimizar a taxa de erro, toma-se a direção oposta do gradiente, o que também é conhecido como a descida do gradiente. O grau de reajuste é influenciado pela taxa de aprendizagem α, que multiplica o gradiente, e o momento μ para considerar o gradiente calculado na época anterior.

$$W_{k+1} = W_k - \alpha \frac{\partial E}{\partial W_k} + \mu \Delta W_{k-1} \tag{8}$$

2.3.5 Aprendizado Profundo

Segundo Goodfellow, Bengio e Courville (2017), DL é um subconjunto da ML (Figura 8) com foco em algoritmos baseados em RNA para aprender representações dos dados. Esta área estuda técnicas que utilizam da experiência para resolver problemas que são difíceis de formalizar, como a identificação de objetos em uma cena ou o reconhecimento de fala. Os autores notam que o termo ‘profundo’ se refere ao fato de que a aprendizagem é realizada em múltiplas camadas estruturadas de forma hierárquica, onde conceitos mais complexos são aprendidos a partir de abstrações de conceitos mais elementares.

Figura 8 - IA, ML e DL



Fonte: Elaborada pelo autor

Goodfellow, Bengio e Courville (2017) avaliam que uma das principais vantagens da DL é a extração automática de características. Tipicamente em um problema de ML, é necessária a colaboração de especialistas humanos para determinar as variáveis fundamentais que devem ser observadas para realizar uma inferência sobre determinado fenômeno. Esta etapa é onerosa e complexa, pois necessita que se saiba exatamente quais são as características relevantes a um dado problema. As técnicas de DL conseguem automatizar este processo por meio de uma cadeia de operações que efetuam transformações simples em cima das representações de camadas anteriores.

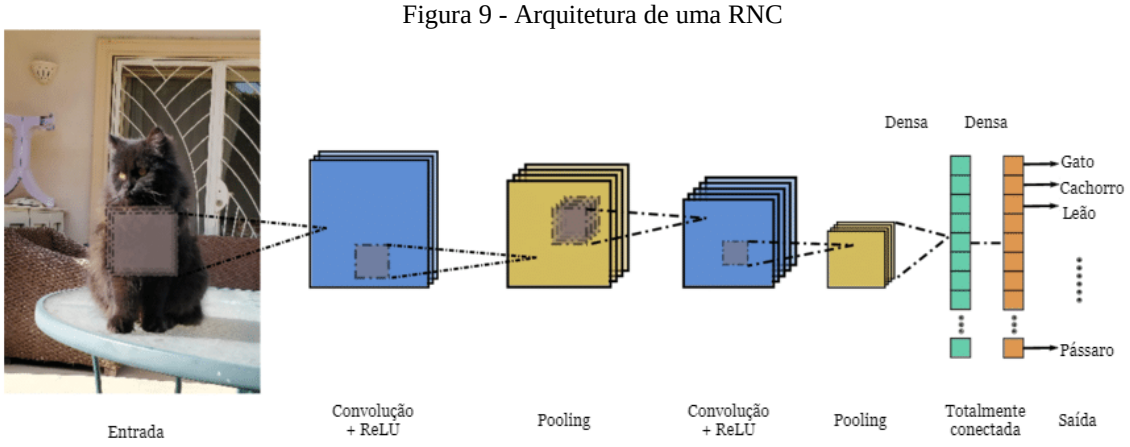
2.3.6 Redes Neurais Convolucionais

Uma RNC é um tipo de RNA que emprega a operação de convolução para encontrar padrões invariantes a espaço nas entradas processadas. Estas entradas geralmente possuem uma estrutura em grade, como uma imagem ou uma série temporal. O aprendizado gira em torno dos filtros convolucionais que descobrem as características mais relevantes de uma dada entrada. Com isso, essa família de redes possui aplicações no campo de visão computacional, onde modelos conseguem aprender a reconhecer detalhes como bordas, cantos, linhas, entre outros (GOODFELLOW; BENGIO; COURVILLE, 2017).

Diferentemente de um perceptron multicamadas (MLP), os padrões que uma RNC aprende não precisam estar nas mesmas localizações para serem detectados. Por exemplo, se um filtro aprender a detectar as bordas no quadrante superior direito de uma imagem, caso exista este padrão em outras imagens, mas em localizações diferentes, o filtro também as identificará. Isto, segundo Chollet (2018), proporciona uma maior flexibilidade para encontrar as características de diferentes imagens e permite que a rede seja computacionalmente eficiente.

De acordo com Chollet (2018), outro fator que favorece o uso de RNC no domínio de reconhecimento de imagens é a forma hierárquica como ocorre a aprendizagem. Por exemplo, dado um conjunto de imagens de gatos, as camadas convolucionais iniciais poderiam aprender a detectar elementos simples como bordas. Já os filtros subsequentes seriam mais sofisticados, combinando bordas para reconhecer formas geométricas. As camadas convolucionais finais combinariam todo conhecimento prévio para distinguir componentes mais complexos, como olhos ou orelhas.

Figura 9 - Arquitetura de uma RNC



Fonte: Adaptada de Radwan (2019)

Por se tratar de uma classe de RNAs profundas, as RNCs são estruturadas com uma camada de entrada, uma de saída e várias camadas ocultas. Um exemplo de arquitetura pode ser visto na Figura 9. A seção oculta inclui, em sua maior parte, camadas convolucionais e camadas de *pooling*.

Em uma camada convolucional, realizam-se convoluções entre a entrada e os filtros, produzindo mapas de características. Estes mapas são transmitidos para uma camada de *pooling*, onde a rede combina os resultados de vários neurônios, efetivamente reduzindo a dimensionalidade dos dados. Após passar pelas camadas de convolução e *pooling*, os dados são propagados para uma camada totalmente conectada onde a classificação é feita (GOODFELLOW; BENGIO; COURVILLE, 2017).

2.3.7 Transferência de Aprendizado

Modelos de DL requerem uma quantia substancial de dados para aprender seus padrões. Porém, em domínios como a imagiologia médica, pode ser inviável obter as amostras necessárias. Buscando contornar esta dificuldade, esforços têm sido desenvolvidos com foco no reaproveitamento de estimadores já treinados. A técnica transferência de aprendizado (TL) visa transmitir o conhecimento gerado na resolução de um problema para outros problemas similares (TAN *et al*., 2018).

Sob o contexto das RNCs e da visão computacional, esta reutilização envolve treinar a rede em uma base de dados maior com diversas imagens de diferentes categorias. Para este propósito, a base mais comum é a ImageNet, composta por 14 milhões de imagens e 22 mil categorias (YANG *et al*., 2018). Assim, o bloco convolucional de uma rede é treinado, extraindo as características necessárias para a identificação das imagens.

Segundo Romero *et al*. (2020), para beneficiar deste conhecimento, novos estimadores são construídos em cima da base convolucional de outra rede pré-treinada. Esta base é congelada, impedindo que os pesos já otimizados sejam alterados pelo processo de treinamento do novo modelo. Apenas as camadas introduzidas pelo novo modelo têm seus pesos modificados com o algoritmo de retropropagação dos erros. Assim, é possível construir um estimador de alto desempenho, mesmo com uma quantia relativamente pequena de dados.

O desempenho de um modelo que faz uso de TL pode ser potencializado por meio de pequenos ajustes na base convolucional. Para tanto, as últimas camadas convolucionais da arquitetura pré-treinada são descongeladas gradualmente. O treinamento é então realizado

usando uma taxa de aprendizagem baixa para evitar que grandes alterações ocorram. Assim, a extração de características é otimizada para o problema específico (ROMERO *et al*., 2020).

2.4 PROCESSAMENTO DE IMAGENS DIGITAIS

Como visto na subseção 2.3.2, a etapa de pré-processamento dos dados de entrada é fundamental para o desenvolvimento adequado de um modelo de ML. Este passo busca normalizar os dados e eliminar quaisquer características irrelevantes que possam interferir na capacidade de aprendizagem do estimador. Stirenko *et al.* (2018) aponta que no âmbito da radiologia, utiliza-se do processamento de imagens digitais para remover artefatos indesejados das imagens, como textos ou instrumentos médicos, e para reduzir discrepâncias de qualidade decorrentes de diferentes técnicas de exame ou dos equipamentos usados.

Estas variações de qualidade também podem acarretar em níveis de contraste inadequados, o que acaba por dificultar uma análise das imagens. Com isso, aplica-se a técnica de equalização de histograma que balanceia a distribuição dos níveis de cinza em uma imagem. Assim, gera-se um contraste maior entre os objetos de uma cena, realçando determinados elementos antes difíceis de discernir. Baseado na frequência de cada nível de cinza, os valores são mapeados a um novo conjunto e os *pixels* da imagem são transformados (DOROTHY *et al*., 2015).

Para efeitos de padronização, imagens também são redimensionadas a um tamanho em comum, o que envolve interpolação. Existem diversos algoritmos para este propósito, como interpolação por vizinho mais próximo, interpolação bilinear e interpolação bicúbica. A interpolação bilinear define o valor de um *pixel* com base nos quatro *pixels* adjacentes e representa um meio termo entre qualidade e desempenho (GETREUER, 2011). O redimensionamento faz com que as imagens de entrada entram em conformidade com a camada inicial de uma RNA.

A segmentação de regiões de interesse é uma técnica que pode ser empregada na fase de pré-processamento para isolar as porções mais relevantes das amostras. Este procedimento é usado na área da medicina para facilitar a análise de exames. Isto é benéfico para modelos de ML, pois detalhes desprezíveis de uma imagem não são contemplados. Assim, a extração de características sem real poder de predição é largamente reduzida. Essa técnica pode ser implementada com uma U-Net, uma RNC totalmente convolucional criada especificamente

para a tarefa de segmentação de imagens biomédicas (RONNEBERGER; FISCHER; PHILIPP, 2015).

2.5 TRABALHOS CORRELATOS

Bressem *et al*. (2020) compararam 16 modelos baseados em seis arquiteturas diferentes para verificar como a estrutura de uma RNC influencia na capacidade diagnóstica da COVID-19. Foram usadas as arquiteturas ResNet (HE *et al.*, 2015), DenseNet (HUANG; LIU; WEINBERGER, 2016), VGG (SIMONYAN; ZISSERMAN, 2014), SqueezeNet (IANDOLA *et al*., 2016), Inception-v4 (SZEGEDY; IOFFE; VANHOUCKE, 2016) e AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017). As redes DenseNet-121, ResNet-50 e VGG-16 apresentaram resultados estado da arte.

Uma limitação de trabalhos que envolvem DL é a falta de dados para treinar modelos. Para minimizar os efeitos na capacidade de generalização do estimador, Minaee *et al*. (2020) combinaram dois *datasets*, um com exames de raio X de pessoas saudáveis e outro com exames de pessoas com COVID-19. Os autores notaram que essa quantia maior de dados acarretou em um aumento na capacidade preditiva e na diminuição da quantidade de classificações erradas.

Buscando otimizar o processo de extração de características, Ahmed *et al.* (2020) propõem que o processo de TL utilize de um conjunto de dados do mesmo domínio. Desta forma, realizaram o treinamento da base convolucional da arquitetura usando o *dataset* CheXpert (IRVIN *et al*., 2019) e não o ImageNet. Com isso, o modelo aprendeu a identificar a presença de doenças similares como pneumonia durante a etapa de TL.

Maguolo e Nanni (2020) conduziram um experimento para verificar a validade dos protocolos de testes usados na avaliação de modelos de ML destinados à detecção da COVID-19. Os autores propositalmente modificaram as imagens de raio X para remover os pulmões. Mesmo assim, conseguiram produzir um modelo de alto desempenho. Isto indica que o modelo aprendeu características que não são correlacionadas à COVID-19. Para contornar este problema, sugeriram que se use de procedimentos de pré-processamento para atenuar quaisquer fontes de viés das bases de dados.

Tartaglione *et al*. (2020) apontam dois procedimentos para mitigar o viés presente nas bases de dados de radiografias de tórax para COVID-19. Primeiramente, as imagens são transformadas por um procedimento de equalização de histograma, criando assim um nível de qualidade mais constante entre as amostras. Depois, para que os modelos extraem das imagens

apenas as características relevantes à doença, cada uma tem a região dos pulmões segmentadas. O produto final é um classificador menos propício ao *overfitting*.

2.6 FERRAMENTAS

2.6.1 Python

Python é uma linguagem de programação de alto nível que pode ser empregada para resolver problemas de diversos domínios, como ciência de dados, ML e desenvolvimento *web*. Suas características incluem código interpretado e portável, tipagem dinâmica e suporte aos paradigmas funcional, orientado a objetos e imperativo (VAN ROSSUM; DRAKE, 2010). A oferta dos pacotes TensorFlow e Keras aumenta a capacidade da linguagem para construir soluções de ML de forma performática, possibilitando que funções sejam executadas tanto na Unidade Central de Processamento (CPU) quanto na Unidade de Processamento Gráfico (GPU). Além disso, sua sintaxe clara e concisa proporciona uma maior agilidade e eficiência no desenvolvimento de projetos.

2.6.1.1 Numpy e Pandas

Numpy é uma biblioteca que permite criar e manipular *arrays* multidimensionais, o que é útil em problemas de ML (HARRIS *et al.*, 2020). Estes *arrays* podem ser usados diretamente com o TensorFlow em operações entre matrizes. Já o Pandas é uma biblioteca usada para analisar e manipular dados, sendo seu elemento principal o *dataframe*. Esta estrutura de dados é derivada dos Numpy *arrays* e é usada para armazenar em um formato tabular as amostras sendo estudadas (MCKINNEY, 2010).

2.6.1.2 TensorFlow e Keras

TensorFlow é uma plataforma de ML desenvolvida pela equipe do Google Brain. Internamente, suas funcionalidades são desenvolvidas em C++ e Compute Unified Device Architecture (CUDA), o que resulta em uma base altamente performática, paralelizável e escalável (ABADI *et al*., 2016). Os recursos de DL do TensorFlow, como os modelos de RNA,

podem ser acessados por meio do Keras, uma API escrita em Python que tem como foco facilitar e simplificar o uso da plataforma (CHOLLET, 2018).

2.6.1.3 OpenCV

O Open Source Computer Vision Library (OpenCV) é uma biblioteca de funções voltada a aplicações de visão computacional originada na Intel em 1998 (PULLI *et al*., 2012). Esta ferramenta possui um conjunto de rotinas que aproveitam do poder de processamento de uma GPU com CUDA. A biblioteca foi escrita em C++, mas possui interface em outras linguagens, como Python. Dentre suas funcionalidades, o projeto usufruirá de seus procedimentos para manipular e processar imagens.

2.6.1.4 Matplotlib e Seaborn

Matplotlib (HUNTER, 2007) e Seaborn (WASKOM *et al.*, 2020) são bibliotecas de Python com foco na criação de visualizações de dados. Elas possuem funcionalidades para criar diferentes tipos de gráficos que permitem inspecionar os dados e os resultados produzidos. Assim, é possível realizar análises e avaliar fatores como a estrutura, a distribuição, o desempenho, entre outros. Estas ferramentas dão suporte a visualizações estáticas, dinâmicas e interativas.

2.6.1.5 Django

Django é um *framework full-stack* de Python para desenvolvimento de aplicações *web*. No pacote base, existem várias funcionalidades integradas, como mapeamento objeto-relacional, resolução de Uniform Resource Locator (URL), autenticação de usuários e administração de conteúdo. Além disso, oferece o Django Template, que provê maior dinamismo para o conteúdo exibido na camada de interface com o usuário. Isto torna o Django um *framework* altamente padronizado, organizado e completo. Com esta ferramenta, é possível criar sistemas que oferecem escalabilidade, segurança e consistência (DJANGO SOFTWARE FOUNDATION, 2020).

2.6.1.6 Gunicorn

Gunicorn é um servidor HTTP WSGI desenvolvido para Python que permite o desenvolvimento de aplicações que aceitam requisições HTTP (CHESNEAU, 2021). Com isso um cliente usando um navegador, por exemplo, consegue solicitar ou alterar algum recurso *web*. A grande vantagem do Gunicorn está na facilidade de implementação e integração com um *framework* como Django.

2.6.2 Tecnologias *Web*

2.6.2.1 HTML e CSS

HTML é uma linguagem de marcação para arquivos que são visualizados em navegadores *web*. Estes arquivos definem o conteúdo das páginas *web*, como textos e imagens. Para que se altere a aparência e a disposição deste conteúdo, utiliza-se do CSS. Com esta ferramenta, é possível definir classes que modificam os aspectos visuais como as cores, o tamanho e a posição dos elementos (DUCKETT, 2011).

2.6.2.2 JavaScript

JavaScript é uma linguagem de programação de alto nível compilada em tempo de execução que oferece suporte para múltiplos paradigmas, como imperativo, funcional, orientado a objetos e orientado a eventos. Esta ferramenta é um dos principais componentes por trás do sistema da World Wide Web, permitindo que usuários interajam com as páginas (DUCKETT; RUPPERT; MOORE, 2014). Devido a sua ubiquidade nas aplicações *web*, existem inúmeras bibliotecas e *frameworks* que podem ser usufruídos em um projeto, o que facilita na implementação de componentes reutilizáveis. Esta ferramenta será aproveitada na construção dos aspectos mais dinâmicos da aplicação, como respostas a eventos gerados pelo usuário.

2.6.2.3 Nginx

Nginx é um servidor *web* que possui diferentes finalidades como balanceamento de carga, *proxy* reverso ou cache. Assim, o servidor pode otimizar o tráfego de requisições ao redirecionar aquelas destinadas aos recursos computacionais sobrecarregados para os menos onerados. Além disso, conteúdo estático pode ser armazenado em memória volátil de rápido acesso, agilizando sua entrega às maquinas clientes (F5, 2021).

2.6.3 Git

Git é uma ferramenta usada para controle de versionamento que permite armazenar e gerenciar o histórico dos arquivos de um projeto. Assim, é possível reverter mudanças e restaurar versões antigas. Além disto, o Git oferece uma maneira fácil de acompanhar a evolução de um *software*, catalogando todas as alterações realizadas, e permite que o código seja acessado e desenvolvido em diferentes máquinas (CHACON; STRAUBE, 2014).

2.6.4 Docker

Docker é uma ferramenta usada para rodar aplicações em ambientes virtualizados. Estes ambientes, conhecidos como contêineres, possuem tudo que é necessário para uma aplicação rodar. Isto inclui as bibliotecas, arquivos de configuração, binários, além da própria aplicação. Os contêineres conseguem operar de forma isolada do restante do sistema, aumentando a segurança e facilitando a portabilidade para outra máquina. Também, com o Docker Compose diferentes contêineres podem cooperar numa mesma tarefa, permitindo que um sistema seja quebrado em microsserviços distintos, cada um rodando em seu próprio contêiner (MORABITO; BEIJAR, 2016).

2.6.5 PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional gratuito e de código aberto. Com esta ferramenta, é possível armazenar e recuperar dados em tabelas relacionais onde as colunas representam os atributos e as linhas são os registros. Este tipo de abordagem é ideal para aplicações que fazem uso de dados com estruturas bem definidas. O

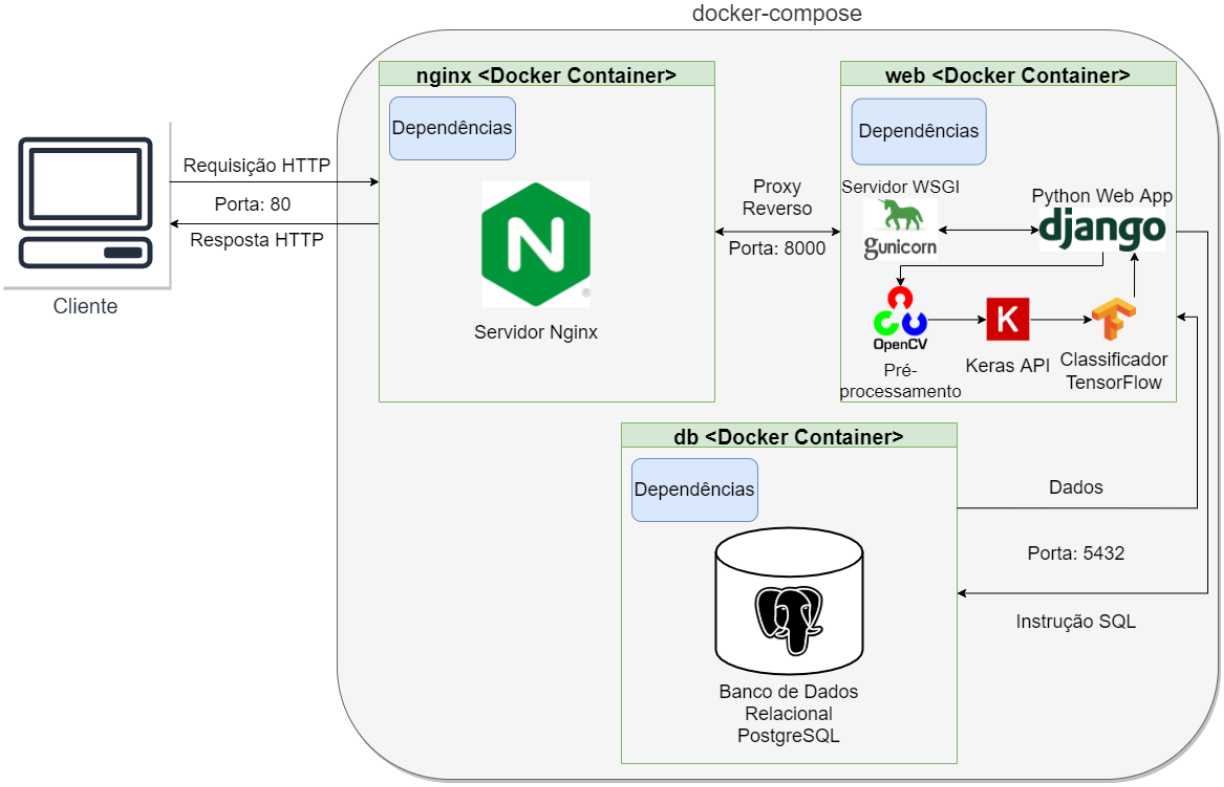
mapeamento objeto-relacional do Django será responsável por converter as classes definidas em Python nas tabelas criadas no banco de dados gerenciado pelo PostgreSQL (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2021).

3 METODOLOGIA

O presente trabalho é classificado como uma pesquisa experimental. De acordo com Gil (2010), em uma pesquisa experimental definem-se um objeto de estudo, as variáveis a serem manipuladas, uma forma de controle para efeitos comparativos e como os resultados produzidos serão mensurados e avaliados. Sua principal finalidade é de testar alguma hipótese visando entender melhor o fenômeno estudado.

A arquitetura geral da aplicação desenvolvida pode ser vista na Figura 10.

Figura 10 - Arquitetura da aplicação para a detecção da COVID-19



Fonte: Elaborada pelo autor

O sistema é dividido em três contêineres Docker, um para a aplicação *web* Django, um para a base de dados com PostgreSQL e um para o servidor Nginx. Esta separação permite que cada unidade tenha suas próprias configurações sem que uma afete as demais. Os contêineres comunicam entre si por meio de portas específicas, possibilitando uma orquestração dos diferentes serviços.

O ponto de entrada se dá com a interação do usuário com o *front-end* da aplicação *web*. Esta interface, construída utilizando as tecnologias HTML, CSS, JavaScript e Django Template,

permite que o usuário faça o envio de exames radiológicos e visualize os resultados computados pelo modelo de ML.

Esta camada é apoiada pelo *back-end* da aplicação, construído com a linguagem de programação Python e o *web framework* Django. Aqui, reside toda a lógica responsável por responder às requisições geradas pelos usuários. A comunicação entre as partes é feita seguindo os padrões REST (W3C, 2004). Ao receber as imagens, cabe ao *back-end* acionar o *pipeline* de ML e fornecer as informações requisitadas.

Para que o modelo de ML consiga adequadamente processar os dados recebidos, estes passam por uma etapa de pré-processamento. Visando mensurar os efeitos dos procedimentos de pré-processamento na capacidade diagnóstica do sistema, foram testadas duas formas de pré-processamento. Na primeira forma, uma imagem tem seu histograma equalizado, depois é redimensionada para se conformar à camada de entrada da RNC e finalmente tem os valores de seus *pixels* normalizados do intervalo [0, 255] para o intervalo [0, 1]. A segunda forma de pré-processamento envolve todos os passos da primeira forma e acrescenta ao final a segmentação dos pulmões usando uma U-Net para tentar remover detalhes menos relevantes presentes nas radiografias originais.

O modelo recebe os dados pré-processados e fornece uma classificação. Antes que isso seja possível, o classificador precisa ser treinado em uma massa de dados. Para maximizar a quantidade de dados, diferentes fontes de imagens são empregadas. Além disso, o processo de treinamento faz uso apenas dos exames radiológicos nas incidências PA e AP. Logo, quaisquer outras projeções foram descartadas.

O COVIDx Dataset é uma agregação de diferentes conjuntos de dados organizado por Wang, Lin e Wong (2020). Ele possui um total de 16.352 imagens, sendo 2.358 amostras de COVID positivo e 13.994 de pacientes COVID negativo. Assim, há um desequilíbrio na representatividade das classes, com COVID negativo sendo aproximadamente seis vezes mais comum. O conjunto de dados é dividido em três subconjuntos de teste, treino e validação. Os conjuntos de teste e validação possuem 200 amostras de cada classe e o conjunto de treino contém as 15.552 amostras restantes (1.958 COVID positivo e 13.594 COVID negativo).

O conjunto de treino é utilizado para melhor aproximar os pesos ideais de cada uma das conexões da rede, minimizando a função de perda. Já as amostras contidas no conjunto de validação são avaliadas ao final de cada época de treinamento e o desempenho do modelo nesta avaliação é usado para otimizar os hiperparâmetros, como o número de épocas e a taxa de aprendizagem. Após concluir o processo de treinamento, o modelo é testado uma única vez no

conjunto de testes para estabelecer o seu desempenho final. Este teste considera as métricas de acurácia, precisão, revocação e F1 *score*.

Visando aumentar o número de amostras, são utilizadas técnicas de *data augmentation*, onde novas imagens são geradas artificialmente ao efetuar aleatoriamente transformações como rotação, reflexão, cisalhamento e ampliação em imagens do conjunto de treino. Também, para equilibrar a representatividade das classes durante o processo de treinamento, às mesmas são atribuídos pesos distintos. Assim, uma amostra da classe COVID positivo tem maior influência no gradiente da função de perda do que uma amostra da classe COVID negativo.

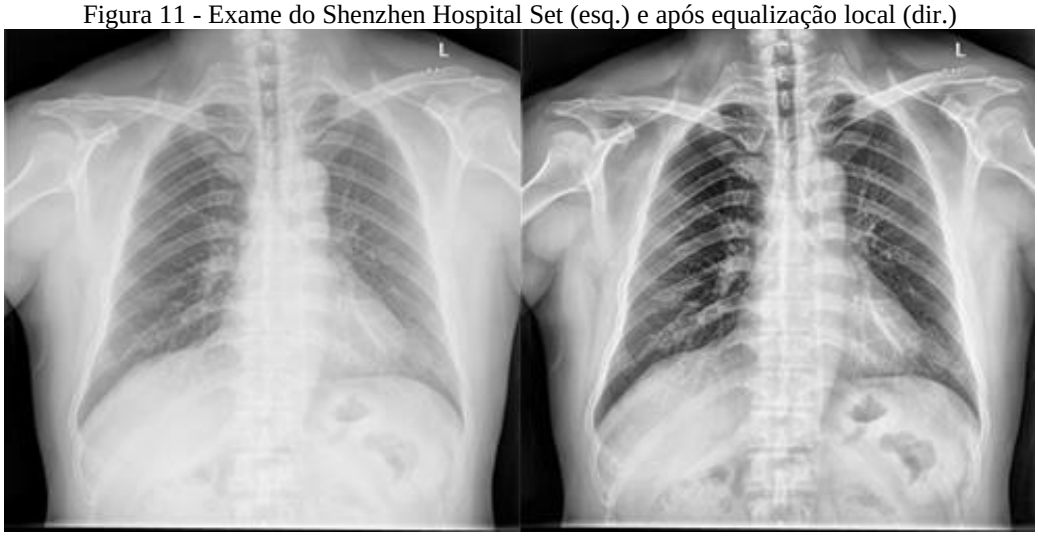
O modelo de ML é uma RNC composta por uma base convolucional de uma rede pré-treinada e camadas totalmente conectadas para produzir uma saída. A base convolucional possui os pesos treinados na base de dados ImageNet e é responsável pela extração das características das imagens. O treinamento é feito em duas etapas. Na primeira, apenas as camadas totalmente conectadas são treinadas, com a base convolucional tendo seus pesos congelados. Na segunda etapa, as camadas superiores da base convolucional são descongeladas e o treinamento é feito com uma taxa de aprendizagem menor para que se tenha um ajuste final dos pesos. As arquiteturas ResNet50V2, DenseNet121 e VGG16 foram comparadas para determinar a que atinge o melhor desempenho para a tarefa de detecção da doença. Como foram aplicadas duas formas de pré-processamento, um total de seis modelos foram treinados e avaliados.

4 IMPLEMENTAÇÃO

4.1 PRÉ-PROCESSAMENTO

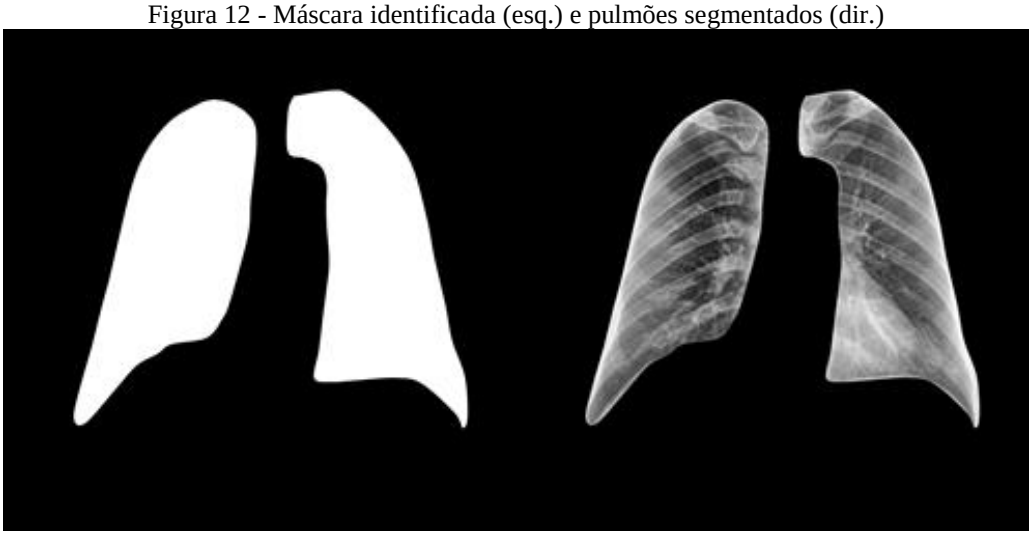
Conforme definido no capítulo 3, cada imagem é submetida a procedimentos de pré-processamento que visam melhorar a qualidade das amostras e tentam remover quaisquer detalhes irrelevantes que possam influenciar negativamente no treinamento do classificador. Estes procedimentos foram implementados com a biblioteca OpenCV. As Figuras 11 e 12 mostram a imagem original e os resultados obtidos da aplicação dos métodos.

Figura 11 - Exame do Shenzhen Hospital Set (esq.) e após equalização local (dir.)



Fonte: Elaborada pelo autor

Figura 12 - Máscara identificada (esq.) e pulmões segmentados (dir.)



Fonte: Elaborada pelo autor

O procedimento de equalização de histograma busca aprimorar a distribuição de intensidades dos *pixels*. Assim, discrepâncias advindas de fatores externos, como as configurações das máquinas usadas nos exames, são atenuadas. Logo, o processo de treinamento do classificador é menos enviesado. Para a implementação, aplicou-se o método Contrast Limited Adaptive Histogram Equalization do OpenCV, que calcula e modifica o histograma local usando janelas 8X8. Optou-se por um procedimento local ao invés de global, pois radiografias são totalmente escuras nas seções que não contém alguma parte do corpo, o que poderia influenciar negativamente na verificação da distribuição dos níveis de cinza nas partes relevantes da imagem.

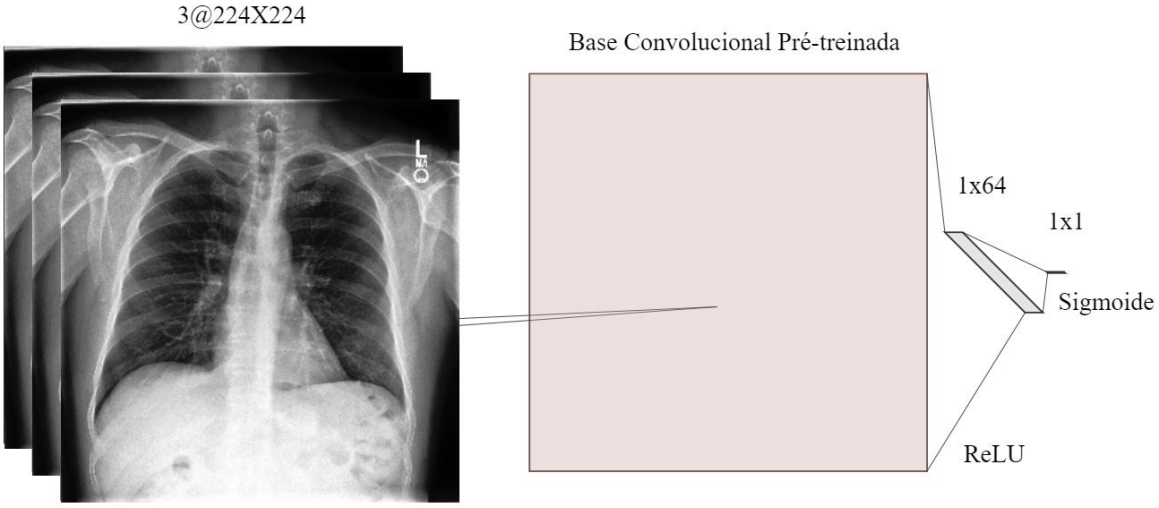
Feita a equalização, a imagem é redimensionada para uma altura e largura de 224 *pixels*. Isto porque a camada de entrada da RNC possui essas dimensões. Depois, a imagem é normalizada, dividindo o valor de cada *pixel* por 255. Isto faz com que os valores da imagem original que são inteiros entre 0 e 255 sejam transformados para o intervalo de reais [0, 1].

A segunda forma de pré-processamento também efetua a segmentação, onde as regiões de interesse são extraídas e o restante da imagem é descartado. Para tanto, foi treinada uma U-Net usando as bases de dados Montgomery County X-ray Set e Shenzhen Hospital X-ray Set (JAEGER *et al.*, 2014). Este modelo aprende a localizar e demarcar os pulmões em exames de raio X do tórax. O resultado é um conjunto de máscaras que, quando aplicadas aos seus respectivos exames, produzem as imagens com os pulmões segmentados.

4.2 CONSTRUÇÃO E TREINAMENTO DO MODELO

A principal ferramenta na construção da RNC é a biblioteca Keras. A Figura 13 demonstra a arquitetura geral da rede. A camada de entrada recebe imagens de cor com dimensões 224X224. Como as radiografias são imagens em escala de cinza com um único canal, a matriz de *pixels* é replicada para os canais R, G e B. Esta camada de entrada é conectada à base convolucional que é instanciada com os pesos treinados na ImageNet. A base é então congelada e o restante do modelo é montado em cima dela. Para tanto, inserem-se duas camadas totalmente conectadas denominadas Dense Layers. A primeira é uma Dense Layer com 64 nós e função de ativação ReLU. A segunda é a camada de saída da rede e é uma Dense Layer com um único nó e função de ativação sigmoide. Assim, caso este último nó apresente um valor menor ou igual à 0,5, a classificação é de COVID positivo e se o valor for maior que 0,5 então é considerado como COVID negativo.

Figura 13 - Arquitetura da Rede Neural Convolucional



Fonte: Elaborada pelo autor

A primeira fase de treinamento é inicializada com 10 épocas e uma taxa de aprendizagem igual à 10^{-3}. A cada época o modelo é testado no conjunto de validação verificando o erro e a acurácia. Caso o erro no conjunto de validação não apresente melhorias após cinco épocas seguidas, reduz-se a taxa de aprendizagem multiplicando-a por 0,2. Ao final desta fase de treinamento, os pesos do modelo que melhor conseguiram reduzir o erro no conjunto de validação são salvos. Estes procedimentos envolvendo o conjunto de validação são para otimizar os hiperparâmetros, encontrando a época e a taxa de aprendizagem que produziram os melhores resultados.

O modelo com os pesos encontrados na fase anterior passa então para a segunda parte do treinamento, onde as camadas superiores da base convolucional são descongeladas. Esta fase também é processada por 10 épocas, mas com uma taxa de aprendizagem menor de 10^{-5}. De novo, os pesos do modelo salvos são aqueles que minimizam a função de erro no conjunto de validação.

Com o treinamento concluído, resta apenas verificar o desempenho do modelo no conjunto de testes. Aplicam-se as métricas apontadas no capítulo 3, além de verificar o erro produzido pela função de perda. Assim, é possível decidir qual combinação de base convolucional e tipo de pré-processamento que melhor consegue diagnosticar a doença. O modelo que atingir o melhor resultado é exportado como um arquivo HDF5 para ser importado e utilizado pela aplicação *web*.

O modelo criado usando a rede DenseNet121 como base convolucional possui 7.103.169 parâmetros distribuídos em 124 camadas. As camadas ocultas estão agrupadas em cinco blocos. Na primeira fase do treinamento, 65.665 parâmetros foram treinados. Já na segunda etapa, foram treinados 2.225.793 parâmetros.

Já o modelo construído em cima da rede ResNet50V2 possui 49.699.585 parâmetros com uma profundidade de 53 camadas. Nesta arquitetura, as camadas ocultas também ficam distribuídas em cinco blocos. A primeira fase do treinamento envolveu a aprendizagem de 2.049 parâmetros. Descongelar o quinto e último bloco da base convolucional fez com que o treinamento envolvesse um total de 14.972.929 parâmetros.

A RNC construída tendo como sua base convolucional a rede VGG16 tem um total de 14.747.585 parâmetros e 19 camadas. Esta rede é estruturada em cinco blocos convolucionais. A primeira parte do treino envolveu 32.897 parâmetros. Descongelar o último bloco levou ao treinamento de 7.112.321 parâmetros.

4.3 CONTÊINERES DOCKER

As características de cada contêiner são expressas em um arquivo de texto denominado Dockerfile. Este arquivo contém as instruções que são executadas para construir aquele contêiner e é estruturado em camadas. Os três contêineres Docker da aplicação *web* (web), do banco de dados (db) e do servidor Nginx (nginx) são orquestrados usando o Compose. Quando o Compose é acionado, o mesmo processa o Dockerfile de cada contêiner e os recursos são gerados.

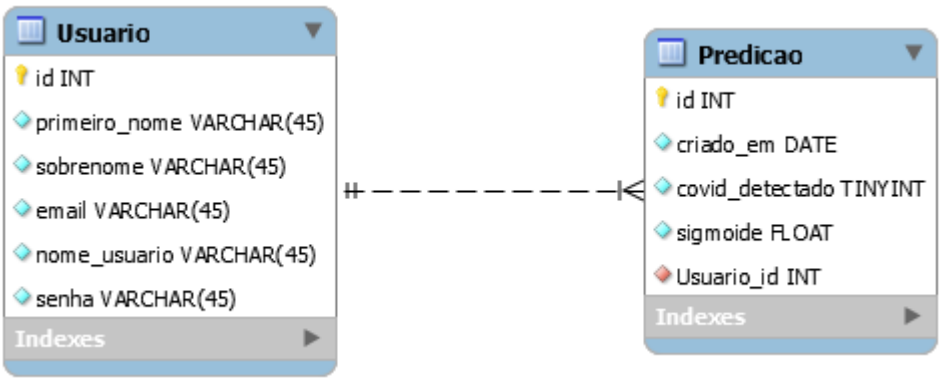
O contêiner web se comunica com o contêiner db usando a porta 5432 que é o padrão do PostgreSQL. O contêiner nginx fica escutando na porta 80 para requisições dos usuários. Ao recebê-las, o nginx as redireciona para o contêiner web através da porta 8000. Além de se comunicarem pela rede, os contêineres web e nginx também compartilham espaço em disco usando dois volumes: um para arquivos estáticos (CSS e JS) e um para arquivos de mídia enviados por usuários (imagens).

4.4 *BACK-END* DA APLICAÇÃO *WEB*

A arquitetura da aplicação *web* segue o padrão Django e é dividida em três partes: *model*, *view* e *template*. A *model* contém as estruturas de dados que são armazenadas no banco

de dados. A Figura 14 mostra as entidades que são definidas como classes em Python e mapeadas usando o Django para tabelas no PostgreSQL.

Figura 14 - Diagrama Entidade Relacionamento



Fonte: Elaborada pelo autor

Quando um usuário realiza o cadastro no sistema, uma nova tupla é inserida na tabela de usuários no banco de dados. Após um exame ser submetido e analisado, uma instância da classe Predição é gerada e associada ao usuário. Um usuário pode ter vários exames enquanto que um exame só pode estar relacionado a um único usuário.

Figura 15 - Principais *views* da aplicação *web*

Fonte: Elaborada pelo autor

A *view* é a parte responsável pela lógica da aplicação. As URLs do sistema são mapeadas às *views* que recebem as requisições dos clientes e produzem respostas HTTP. Como pode ser visto na Figura 15, as três principais *views* do sistema são: *signup*, que carrega a página inicial e processa o formulário de registro dos usuários, *upload*, que carrega e processa o

formulário para envio do exame raio X, e *results*, que chama a função responsável pelo pré-processamento da imagem, depois chama a função responsável por carregar o modelo de ML e realizar a predição da classe, e finalmente retorna o resultado para o usuário. O sistema também conta com *views* pré-configuradas pelo Django para realizar ações de autenticação como entrar, sair e redefinir senha.

O *template* faz parte do *front-end* da aplicação. As *views* carregam os *templates* que são essencialmente arquivos HTML. Ao carregar esses *templates*, as *views* conseguem passar variáveis, permitindo a disponibilização de conteúdo dinâmico. Este recurso é aplicado na página de resultados, com a *view* enviando as predições associadas ao usuário.

4.5 *FRONT-END* DA APLICAÇÃO *WEB*

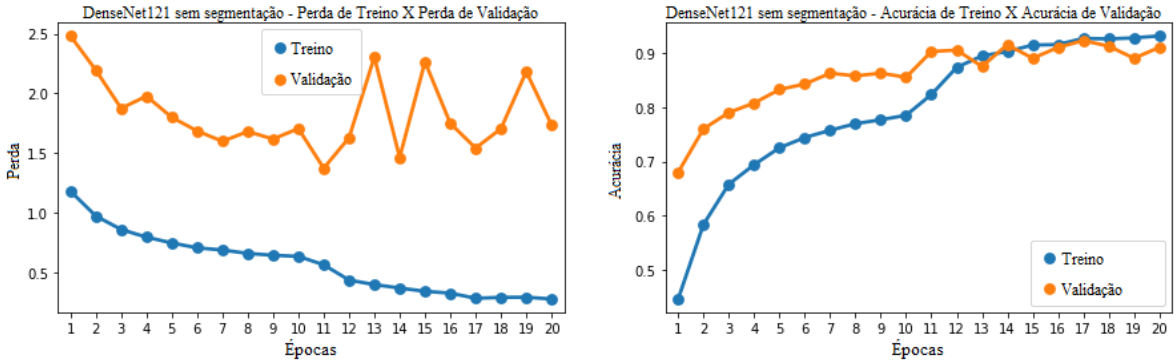
O *front-end* da aplicação *web* é a camada responsável por toda a interface gráfica que os usuários visualizam e interagem. Esta parte contém três componentes: os *templates*, que são os arquivos HTML com o conteúdo das páginas, os *style sheets*, que são os arquivos CSS que modificam o aspecto visual deste conteúdo, e os *scripts*, que são arquivos escritos em JavaScript com funções que geram animações em determinadas ações do usuário.

Os *style sheets* e os *scripts* são considerados arquivos estáticos, pois não são modificados dinamicamente. Quando a aplicação inicializa, estes arquivos são todos copiados automaticamente pelo Django para um diretório de arquivos estáticos que é monitorado pelo contêiner nginx. O uso do nginx para servir estes arquivos aos usuários que acessam o sistema agiliza o carregamento da página *web*.

5 RESULTADOS E DISCUSSÃO

5.1 DENSENET121 SEM SEGMENTAÇÃO

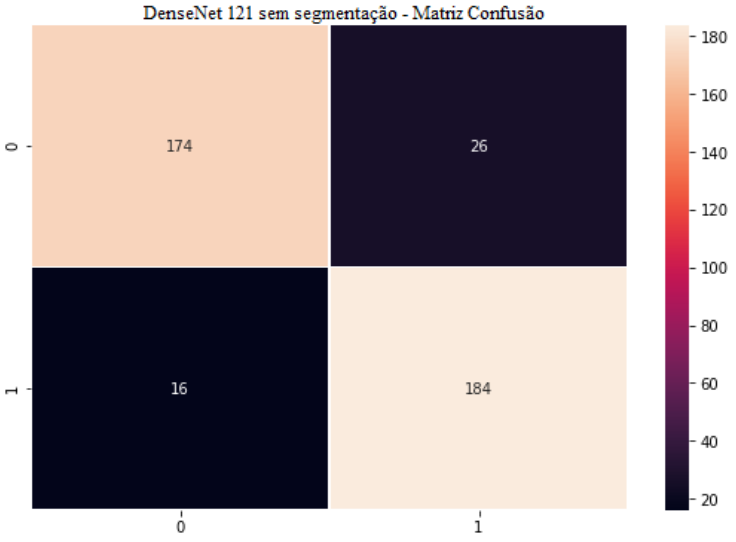
Figura 16 - DenseNet121 sem segmentação: perda e acurácia



Fonte: Elaborada pelo autor

A Figura 16 mostra a progressão das métricas de perda (gráfico à esquerda) e acurácia (à direita) para o modelo DenseNet121 sem segmentação durante as 20 épocas de treinamento efetuadas. As curvas em laranja representam tais métricas avaliadas no conjunto de validação e as curvas em azul são para o conjunto de treino. É possível verificar que o desempenho do modelo obteve melhorias significantes nas primeiras épocas e oscilou ao aproximar 0,9 de acurácia. Isto indica *overfitting* nas épocas finais e sugere que uma parada antecipada do processo de treinamento pode ser benéfica para o classificador.

Figura 17 - DenseNet121 sem segmentação: matriz confusão

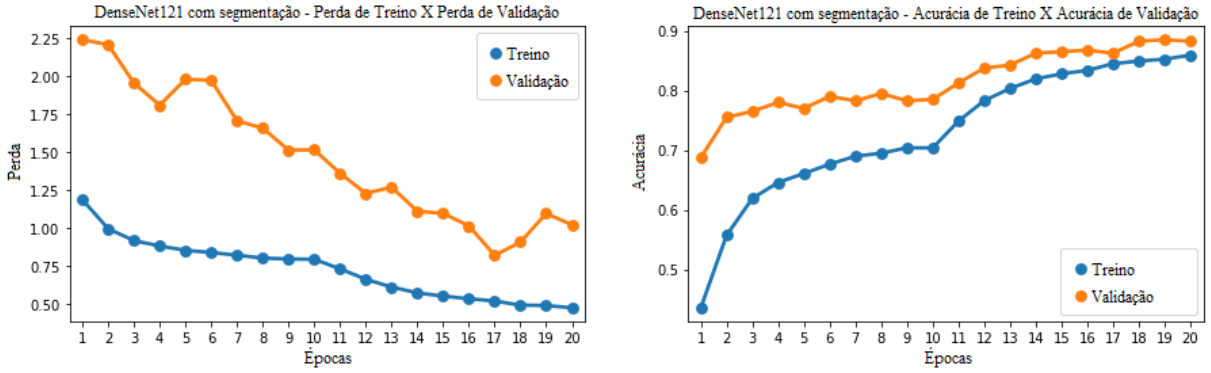


Fonte: Elaborada pelo autor

Os resultados da avaliação do modelo no conjunto de testes estão representados na matriz confusão da Figura 17. Os valores na diagonal principal são os acertos do classificador e demonstram que para 358 amostras a classificação foi correta. Assim, o modelo obteve uma acurácia de 89,5%. É possível verificar também que o número de falsos negativos (26) foi maior que o número de falsos positivos (16). Dada a natureza da aplicação, é mais desejável que se minimize a quantidade de falsos negativos, pois este tipo de diagnóstico pode resultar no não tratamento de um paciente doente.

5.2 DENSENET121 COM SEGMENTAÇÃO

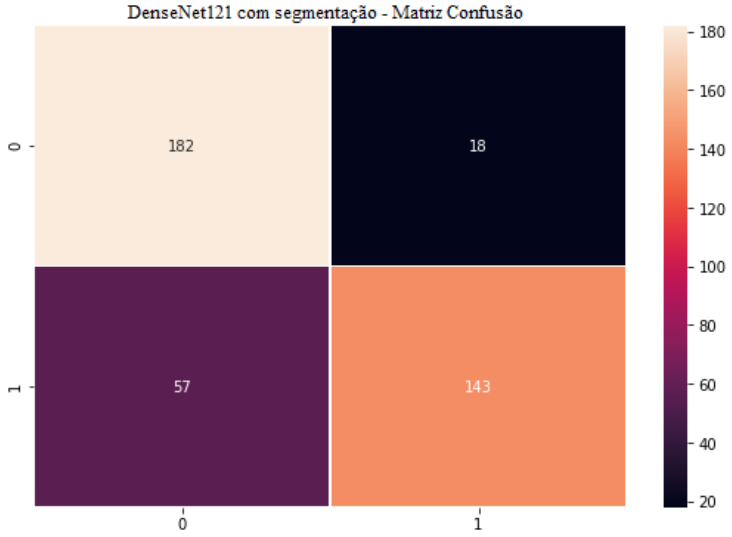
Figura 18 - DenseNet121 com segmentação: perda e acurácia



Fonte: Elaborada pelo autor

Olhando a Figura 18, é possível observar que o treinamento do modelo utilizando a segmentação dos pulmões resultou em um desempenho com menos oscilações ao longo das épocas. Conforme a Figura 19, a capacidade preditiva no conjunto de testes foi prejudicada em relação aos falsos positivos (57) mas obteve uma quantia menor de falsos negativos (18). Para esta configuração, a acurácia constatada foi de 81,25%.

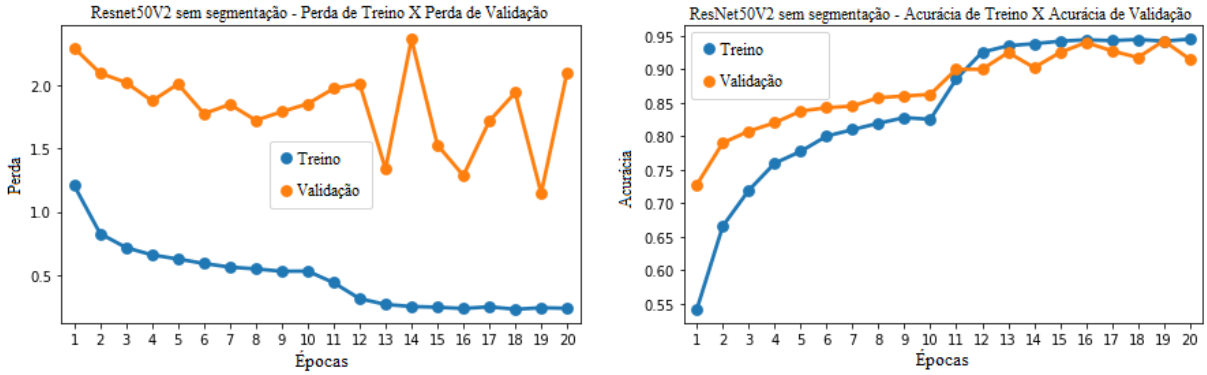
Figura 19 - DenseNet121 com segmentação: matriz confusão



Fonte: Elaborada pelo autor

5.3 RESNET50V2 SEM SEGMENTAÇÃO

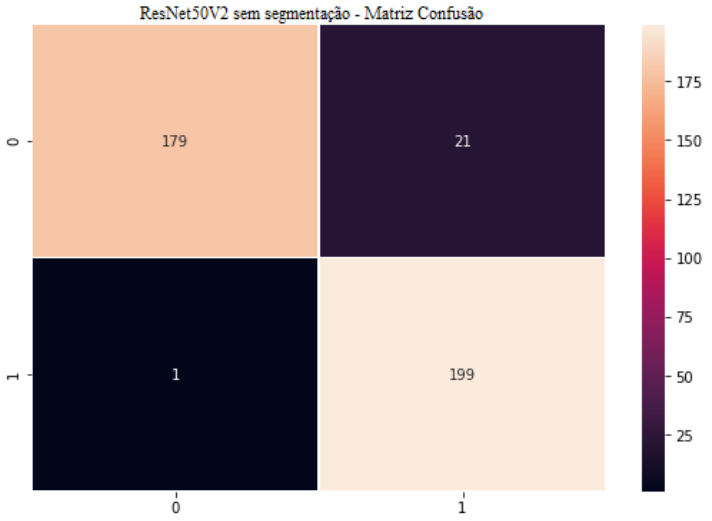
Figura 20 - ResNet50V2 sem segmentação: perda e acurácia



Fonte: Elaborada pelo autor

De maneira similar à arquitetura DenseNet121, a aprendizagem sem a segmentação da ResNet50V2 retratada na Figura 20 apresentou uma variância aumentada na segunda fase do treinamento. A acurácia final foi de 94,5%. Como mostra a Figura 21, esta rede conseguiu reduzir significativamente a incidência de falsos positivos (1), mas não foram constadas melhorias quanto à quantidade de falsos negativos (21). Esta ocorrência pode estar ligada ao desequilíbrio no *dataset* usado para o treinamento das redes. Como há uma quantia significativamente maior de amostras COVID negativo, é de se esperar que o classificador tenda a priorizar esta classificação ao avaliar as amostras de teste.

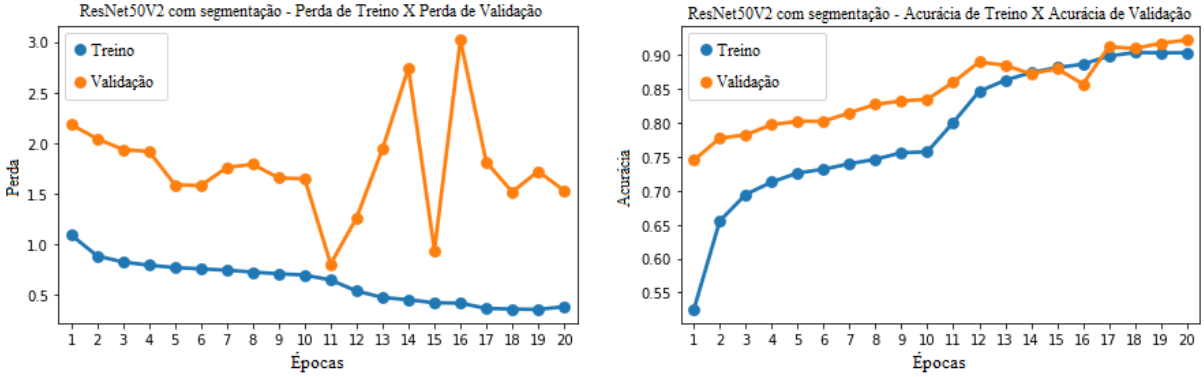
Figura 21 - ResNet50V2 sem segmentação: matriz confusão



Fonte: Elaborada pelo autor

5.4 RESNET50V2 COM SEGMENTAÇÃO

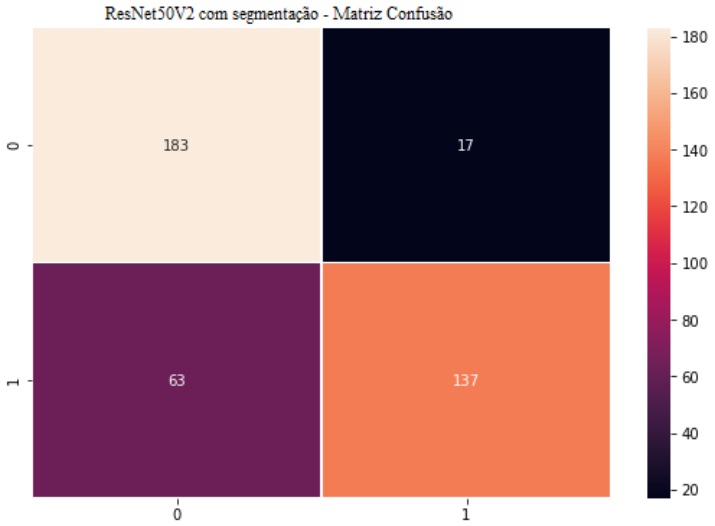
Figura 22 - ResNet50V2 com segmentação: perda e acurácia



Fonte: Elaborada pelo autor

A combinação de pré-processamento com segmentação dos pulmões e o modelo baseado na ResNet50V2 continuou demonstrando elevadas oscilações na perda do conjunto de validação (Figura 22). Além disso, a Figura 23 revela que houve um aumento considerável de casos falso positivo (63). Este aumento pode ser o resultado de uma perda de informação durante o processo de extração das regiões de interesse. A segmentação, por se tratar de um procedimento automatizado e sem garantia de exatidão, pode acarretar na remoção de partes do pulmão na imagem final, o que implicaria em maiores erros de classificação. Desta maneira, a acurácia final observada foi de 80%.

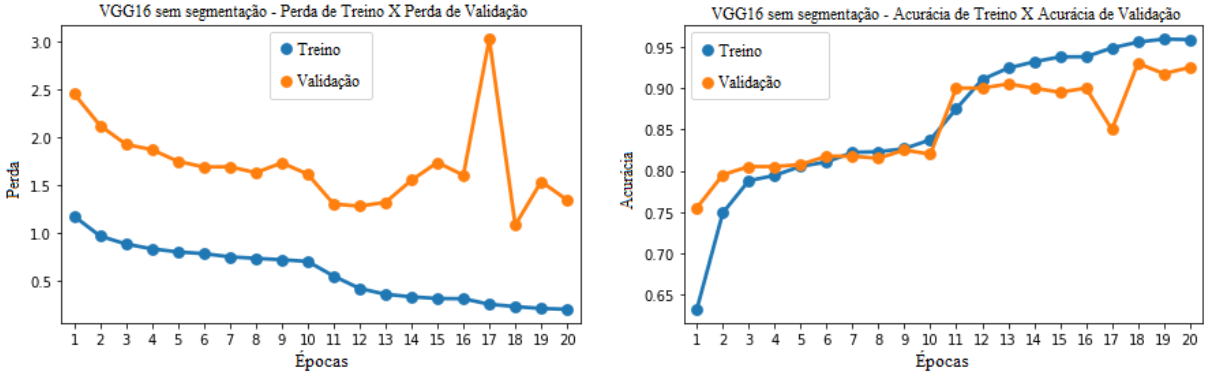
Figura 23 - ResNet50V2 com segmentação: matriz confusão



Fonte: Elaborada pelo autor

5.5 VGG16 SEM SEGMENTAÇÃO

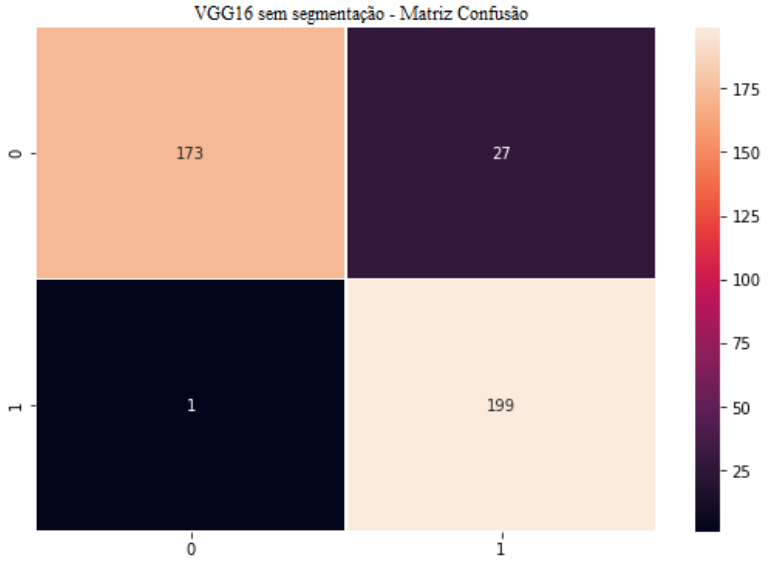
Figura 24 - VGG16 sem segmentação: perda e acurácia



Fonte: Elaborada pelo autor

Ao analisar a Figura 24, é possível verificar que o erro de validação teve um aumento acentuado na época 17. Isto pode ter sido causado pelo algoritmo da descida do gradiente que, ao tentar atingir um mínimo local, acabou excedendo o alvo. Mesmo assim, a rede conseguiu recuperar seu desempenho nas épocas seguintes. A Figura 25 demonstra um certo padrão nos modelos: quando a etapa de pré-processamento não envolve a segmentação, a quantidade de falsos positivos (1) é baixa. A acurácia deste modelo ficou em 93%.

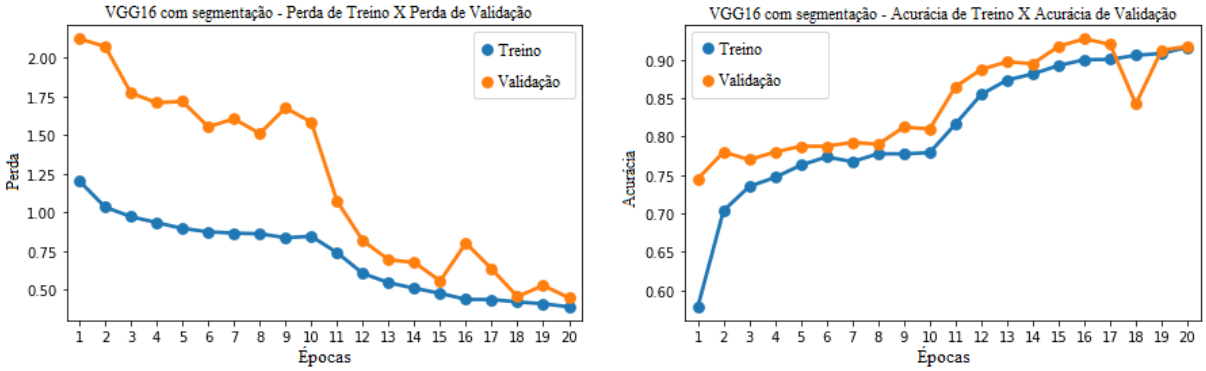
Figura 25 - VGG16 sem segmentação: matriz confusão



Fonte: Elaborada pelo autor

5.6 VGG16 COM SEGMENTAÇÃO

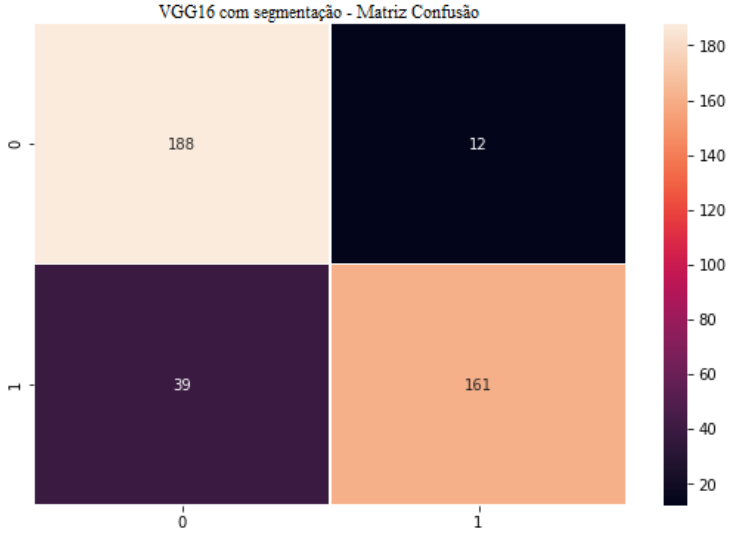
Figura 26 - VGG16 com segmentação: perda e acurácia



Fonte: Elaborada pelo autor

Como pode ser visto na Figura 26, a rede com VGG16 e uso da segmentação demonstrou um dos menores erros de validação (aproximadamente 0,5). Além disso, a Figura 27 revela a menor parcela de falsos negativos dentre as arquiteturas testadas com apenas 12 casos. Assim, esta combinação se mostrou menos propícia a classificar um paciente COVID positivo como saudável. Avaliando o conjunto de teste, o classificador obteve uma acurácia geral de 87,25%.

Figura 27 - VGG16 com segmentação: matriz confusão



Fonte: Elaborada pelo autor

5.7 COMPARANDO AS DIFERENTES ARQUITETURAS

Tabela 1 - Métricas de avaliação para as diferentes arquiteturas

	Acurácia	**Perda**	**Precisão COVID+**	**Precisão COVID-**	**Revocação COVID+**	**Revocação COVID-**	**F1 *score* COVID+**	**F1 *score* COVID-**
DenseNet121 sem segmentação	0,895	0,263	0,92	0,88	0,87	0,92	0,89	0,90
DenseNet121 com segmentação	0,812	0,362	0,76	0,89	0,91	0,71	0,83	0,79
ResNet50V2 sem segmentação	0,945	0,199	0,99	0,90	0,90	0,99	0,94	0,95
ResNet50V2 com segmentação	0,80	0,365	0,74	0,89	0,92	0,69	0,82	0,77
VGG16 sem segmentação	0,93	0,188	0,99	0,88	0,86	0,99	0,93	0,93
VGG16 com segmentação	0,873	0,301	0,83	0,93	0,94	0,81	0,88	0,86

Fonte: Elaborada pelo autor

A Tabela 1 demonstra o desempenho de cada arquitetura no conjunto de testes considerando as métricas de acurácia, precisão, revocação e F1 *score*. Verifica-se que o pré-processamento com a segmentação das regiões de interesse resultou em um menor número de acertos de classificação. É possível que as aplicações das máscaras dos pulmões geradas pela U-Net tenham provocado uma perda de características de interesse para a detecção da doença. Como este procedimento de segmentação é feito de maneira automatizada, não se pode garantir que as máscaras produzidas extraiam a região pulmonar de forma íntegra em todas as amostras.

Já a combinação VGG16 com segmentação produziu os maiores níveis de revocação para a classe COVID positivo. Como explicado anteriormente, por se tratar de uma aplicação médica, um baixo grau de falsos negativos é de suma importância porque caso um paciente com a doença seja classificado erroneamente como saudável, poderia suceder naquela pessoa não recebendo os tratamentos adequados. Contudo, o modelo escolhido para a aplicação foi a combinação ResNet50V2 sem segmentação por ter, em média, o melhor desempenho.

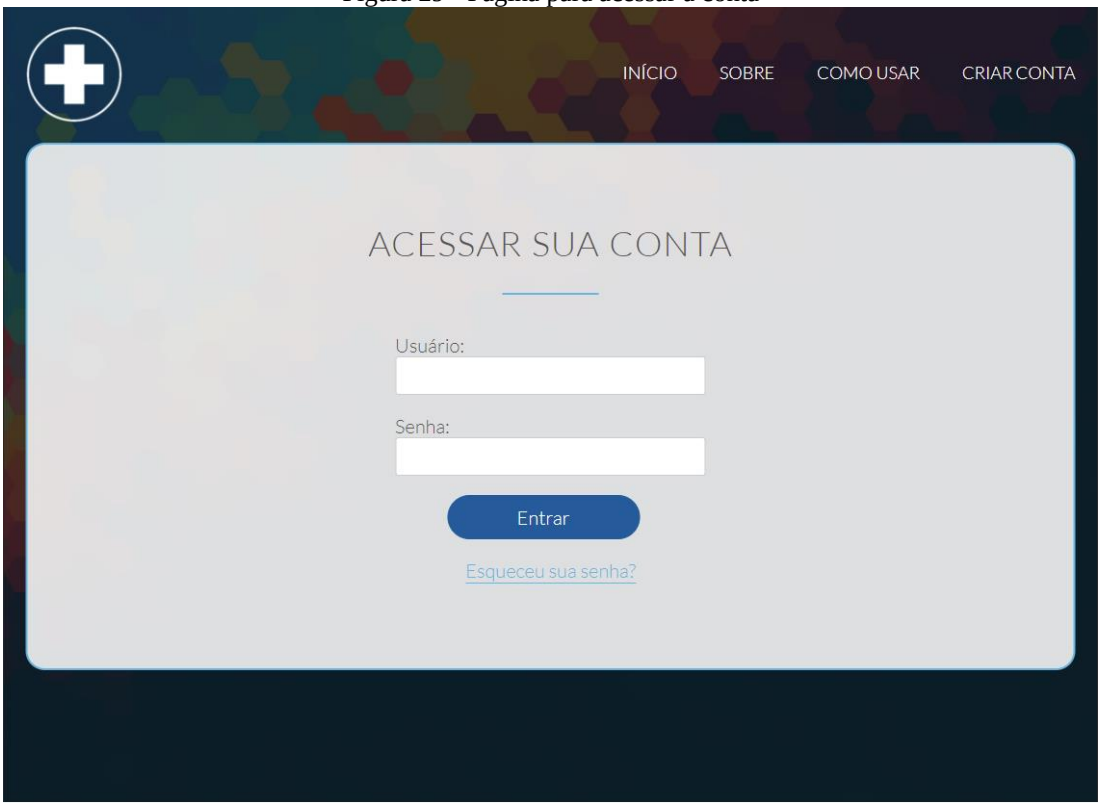
5.8 APLICAÇÃO *WEB*

Figura 28 - Página inicial da aplicação *web*

Fonte: Elaborada pelo autor

A ferramenta permite que usuários façam o envio de seus exames para que os mesmos possam ser avaliados pelo classificador. Este sistema foi construído tendo como metas uma fácil navegação, usabilidade simplificada e disponibilização apenas das informações mais importantes. A Figura 28 mostra a página inicial da aplicação.

Figura 29 - Página para acessar a conta



Fonte: Elaborada pelo autor

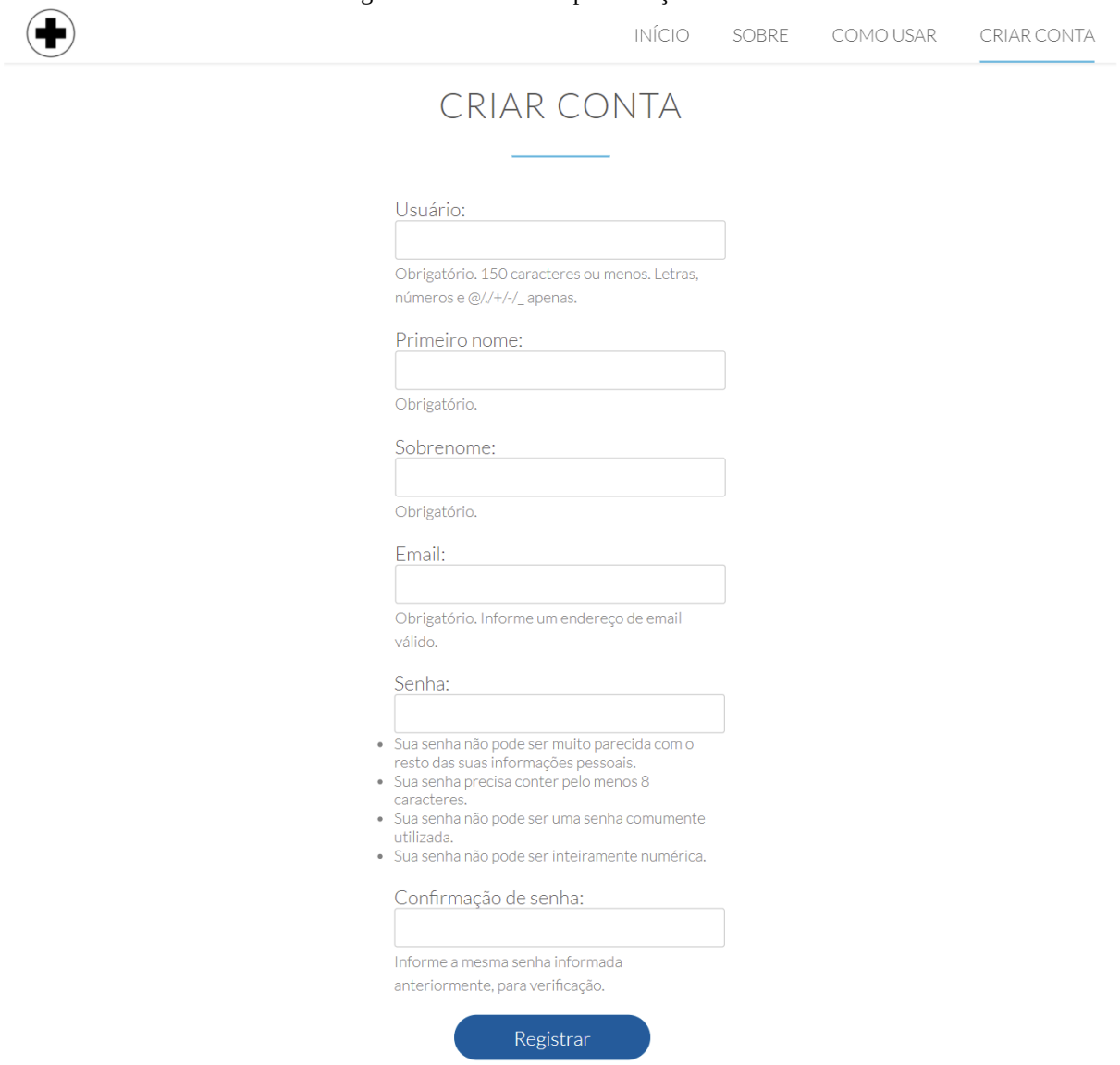
O usuário consegue navegar pelas diferentes seções da plataforma clicando nos *links* localizados na parte superior da página. O botão para acessar a conta redireciona o usuário para a página de *login* (Figura 29). Aqui, também é possível solicitar a redefinição de senha.

Figura 30 - Página com instruções de uso

Fonte: Elaborada pelo autor

Conforme está exposto na Figura 30, todo o procedimento pode ser realizado em quatro passos. Antes que os exames possam ser submetidos para análise, o usuário deve acessar sua conta. Caso ainda não possua uma, é possível criá-la preenchendo o cadastro (Figura 31).

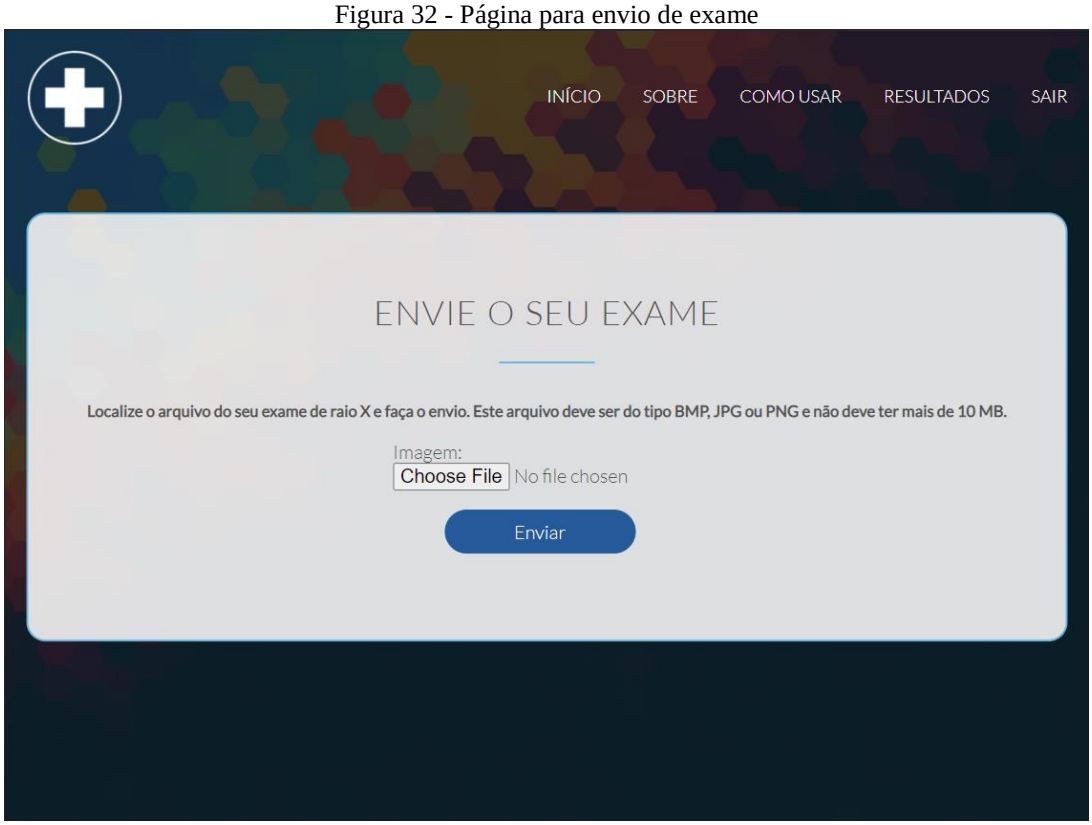
Figura 31 - Formulário para criação de conta



Fonte: Elaborada pelo autor

Após se autenticar no sistema, fica disponível o botão para o envio de exames. Ao clica-lo, o usuário é redirecionado para a página de envio. Aqui, o arquivo em formato de imagem (BMP, JPEG ou PNG) deve ser localizado, anexado e submetido (Figura 32).

Figura 32 - Página para envio de exame



Fonte: Elaborada pelo autor

Com isso, o modelo de ML é acionado e, após classificar o exame, gera o resultado. Na página principal, uma nova seção fica disponível para apresentar uma relação contendo todos os resultados já calculados para os exames submetidos pelo usuário (Figura 33).

Figura 33 - Resultados calculados pelo classificador

Fonte: Elaborada pelo autor

6 CONCLUSÃO

Neste trabalho, foi desenvolvida uma aplicação *web* cuja finalidade é de classificar radiografias de tórax para predizer se há ou não a presença da COVID-19. O sistema oferece uma interface para que usuários cadastrados consigam submeter seus exames e receber os resultados calculados pelo modelo de ML. Assim, foi elaborada uma plataforma de fácil acesso que fornece um retorno imediato ao usuário.

Para realizar a classificação das imagens, foi utilizado um estimador baseado em RNCs. Diferentes combinações de redes e de procedimentos de pré-processamento foram comparadas com o objetivo de definir a arquitetura ideal. Além disso, cada RNC teve seus hiperparâmetros de taxa de aprendizagem e quantidade de épocas de treinamento otimizados usando o conjunto de validação. Durante o treinamento, quando o desempenho do classificador no conjunto de validação não apresentava melhorias, a taxa de aprendizagem era reduzida por um fator de 0,2. Também, todo o histórico de treinamento foi verificado para definir a época que produziu os melhores resultados.

Ao realizar os testes, duas combinações se sobressaíram. Por um lado, a combinação VGG-16 com segmentação dos pulmões produziu o menor índice de falsos negativos, com apenas 12 casos, e resultou em uma acurácia de 0,873. Já a combinação ResNet50V2 sem segmentação das regiões de interesse apresentou mais erros de falsos negativos, com 21 amostras, mas obteve o melhor desempenho em geral, com 0,945 de acurácia. Levando isto em consideração, a combinação ResNet50V2 sem segmentação foi adotada como a arquitetura da aplicação final.

Um fator limitante foi o desequilíbrio presente entre as classes no conjunto de dados usado para treinar o classificador. Com 2.358 amostras de COVID positivo e 13.994 de pacientes COVID negativo, a segunda classe teve uma representatividade aproximadamente seis vezes maior. Mesmo aplicando pesos diferentes em cada classe com o intuito de valorizar mais as amostras de COVID positivo, há uma variedade maior nas amostras de COVID negativo, permitindo que o classificador aprendesse mais de suas características. Como forma de mitigar este problema, é crucial que radiografias de pacientes com a COVID-19 confirmada continuam sejam disponibilizadas em plataformas de livre acesso.

Outra limitação de pesquisa envolveu o processo de extração das regiões de interesse. Por utilizar de procedimentos automatizados, não foi possível segmentar com exatidão as áreas

dos pulmões. Assim sendo, estas imperfeições acabaram por comprometer a capacidade de aprendizagem e de generalização do estimador.

Para trabalhos futuros, sugere-se a implementação de um sistema de votação na classificação de amostras, onde a aplicação final emprega um conjunto de modelos de ML. Assim, o sistema produziria uma resposta fruto da combinação das predições geradas por cada estimador. Com isso, espera-se que haja um aumento na acurácia e na confiança dos resultados.

Outra sugestão seria em testar diferentes métodos para a segmentação das regiões de interesse. Aplicando uma forma mais precisa de extrair os pulmões poderia resultar em um classificador menos propício ao *overfitting* e com maior capacidade de generalização. Tal medida serviria também para reduzir o viés encontrado em uma base de dados e produziria imagens com um nível de qualidade mais padronizado.

Procedimentos com eficácia na detecção da enfermidade são fundamentais para tratar pacientes doentes e para diminuir o número total de infecções. A automação dos métodos de identificação da doença usando técnicas de DL agiliza e padroniza o processo de descoberta. Assim, esta aplicação se propõe a ser mais uma ferramenta no combate contra a COVID-19, podendo auxiliar profissionais da saúde na avaliação de exames radiológicos para o diagnóstico da doença.

REFERÊNCIAS

ABADI, M. *et al*. **TensorFlow:** A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, p. 265-283, 2016. Disponível em: <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>. Acesso em: 29 out. 2020.

AHMED, S. *et al*. **ReCoNet**: Multi-level Preprocessing X-rays for of Chest COVID-19 Detection Using Convolutional Neural Networks. 2020. Disponível em: <https://doi.org/10.1101/2020.07.11.20149112>. Acesso em: 04 nov. 2020

ARCHER, S. L.; SHARP, W. W.; WEIR, E. K. **Differentiating COVID-19 Pneumonia From Acute Respiratory Distress Syndrome and High Altitude Pulmonary Edema**. *Circulation*, v. 142, n. 2, p. 101-104, 2020. Disponível em: <https://www.ahajournals.org/doi/pdf/10.1161/CIRCULATIONAHA.120.047915>. Acesso em: 5 out. 2020.

BRANT, W. E.; HELMS, C. A. **Fundamentals of diagnostic radiology**. 3. ed. Philadelphia: Lippincott, Williams & Wilkins, 2007.

BRE, F.; GIMENEZ, J. M.; FACHINOTTI, V. D. **Prediction of wind pressure coefficients on building surfaces using artificial neural networks**. *Energy and Buildings*, v. 158, p. 1429-1441, 2018. Disponível em: <https://doi.org/10.1016/j.enbuild.2017.11.045>. Acesso em: 12 set. 2020.

BRESSEM, K. K. *et al*. **Comparing different deep learning architectures for classification of chest radiographs**. *Scientific Reports*, v. 10, n. 1, 2020. Disponível em: <http://dx.doi.org/10.1038/s41598-020-70479-z> Acesso em: 15 set. 2020.

CHACON, S.; STRAUB, B.. **Pro Git**. 2. ed. Berkeley, CA: Apress, 2014.

CHESNEAU, B. **Gunicorn - WSGI server — Gunicorn documentation**. *Gunicorn*, 2021. Disponível em: <https://docs.gunicorn.org/en/stable/>. Acesso em: 14 mar. 2021.

CHODICK, G. *et al*. **Radiation Risks from Pediatric Computed Tomography Scanning**. *Pediatric endocrinology reviews*, v. 7, n. 2, p. 29-36, 2009. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6298033/>. Acesso em: 19 set. 2020.

CHOLLET, F. **Deep Learning with Python**. Shelter Islands: Manning, 2018.

COHEN, J. P. *et al*. **COVID-19 Image Data Collection: Prospective Predictions are the Future**. *arXiv 2006*.11988, 2020. Disponível em: <https://arxiv.org/abs/2006.11988>. Acesso em: 15 out. 2020.

CORMAN, V. M. *et al*. **Detection of 2019 novel coronavirus (2019-nCoV) by real-time RT-PCR**. *Eurosurveillance*, v. 25, n. 3, 2020. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6988269/>. Acesso em: 14 set. 2020.

DJANGO SOFTWARE FOUNDATION. **Django**. [S.I.] 2020. Disponível em: <https://djangoproject.com>. Acesso em: 5 set. 2020.

DOROTHY, R. *et al*. **Image enhancement by Histogram equalization**. *International Journal of Nano Corrosion Science and Engineering*, v. 2, p. 21-30, 2015. Disponível em: <https://www.researchgate.net/publication/283727396_Image_enhancement_by_Histogram_equalization>. Acesso em: 9 out. 2020.
DUCKETT, J. **HTML & CSS**: design and build websites. Indianapolis: Wiley, 2011.

DUCKETT, J.; RUPPERT, G.; MOORE, J. **JavaScript & jQuery**: interactive front-end web development. Indianapolis: Wiley, 2014.

F5. **NGINX Docs**. *NGINX*, 2021. Disponível em: <https://docs.nginx.com/>. Acesso em: 5 abr. 2021.

FANG, Y. *et al*. **Sensitivity of Chest CT for COVID-19:** Comparison to RT-PCR. *Radiology*, v. 296, n. 2, p. E115-E117, 2020. Disponível em: <https://pubs.rsna.org/doi/full/10.1148/radiol.2020200432>. Acesso em: 19 set. 2020.

FOUST, A. *et al*. **International Expert Consensus Statement on Chest Imaging in Pediatric COVID-19 Patient Management**: Imaging Findings, Imaging Study Reporting and Imaging Study Recommendations. *Radiology: Cardiothoracic Imaging*, v. 2, n. 2, p. e200214, 2020. Disponível em: <https://doi.org/10.1148/ryct.2020200214>. Acesso em: 23 set. 2020.

GETREUER, P. **Linear Methods for Image Interpolation.** *Image Processing On Line*, v. 1, 2011. Disponível em: <https://doi.org/10.5201/ipol.2011.g_lmii>. Acesso em: 4 nov. 2020.

GHEBREYESUS, Tedros Adhanom. **WHO Director-General's opening remarks at the media briefing on COVID-19 - 11 March 2020**. [S.I.] 2020. Disponível em: <https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020>. Acesso em: 13 set. 2020.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, Mass: The MIT Press, 2017.

HANDELMAN, G. S. *et al*. **Peering Into the Black Box of Artificial Intelligence:** Evaluation Metrics of Machine Learning Methods. *American Journal of Roentgenology*, v. 212, n. 1, p. 38–43, 2019. Disponível em: <http://dx.doi.org/10.2214/AJR.18.20224>. Acesso em: 16 out. 2020.

HAPKE, H.; NELSON, C. **Building Machine Learning Pipelines**. Sebastopol: O'Reilly Media, 2020.

HARRIS, C. R. *et al*. **Array programming with NumPy**. *Nature*, v. 585, n. 7825, p. 357-362, 2020. Disponível em: <http://doi.org/10.1038/s41586-020-2649-2>. Acesso em: 2 out. 2020.

HAYKIN, S. S. **Neural networks and learning machines**. 3. ed. Upper Saddle River: Prentice Hall, 2009.

HE, K. *et al*. **Deep Residual Learning for Image Recognition**. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Acesso em: 25 set. 2020.

HUANG, G.; LIU, Z.; WEINBERGER, K. Q. **Densely Connected Convolutional Networks**. *CoRR*, abs/1608.06993, 2016. Disponível em: <http://arxiv.org/abs/1608.06993>. Acesso em: 28 set. 2020.

HUNTER, J. D. **Matplotlib**: A 2D Graphics Environment. Computing in Science & Engineering, v. 9, n. 3, p. 90-95, 2007. Disponível em: <https://doi.org/10.1109/MCSE.2007.55>. Acesso em: 7 out. 2020.

IANDOLA, F. N. *et al*. **SqueezeNet**: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *CoRR*, abs/1602.07360, 2016. Disponível em: <http://arxiv.org/abs/1602.07360>. Acesso em: 22 set. 2020.

IRVIN, J. *et al*. **CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison**. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, n. 1, p. 590-597, 2019. Disponível em: <http://www.doi.org/10.1609/aaai.v33i01.3301590>. Acesso em: 17 out. 2020.

JAEGER, S. **Detecting Disease in Radiographs with Intuitive Confidence**. *The Scientific World Journal*, v. 2015, p. 1-9, 2015. Disponível em: <https://doi.org/10.1155/2015/946793>. Acesso em: 23 set. 2020.

JAEGER, S. *et al*. **Two public chest X-ray datasets for computer-aided screening of pulmonary diseases**. *Quantitative Imaging in Medicine and Surgery*, v. 4, n. 6, p. 475–477, 2014. Disponível em: <https://doi.org/10.3978/j.issn.2223-4292.2014.11.20>. Acesso em: 21 nov. 2020.

JAIN, A. K.; MAO, J.; MOHIUDDIN, K. M. **Artificial neural networks:** a tutorial. *Computer*, v. 29, n. 3, p. 31-44, 1996. Disponível em: <http://doi.org/10.1109/2.485891>. Acesso em: 1 nov. 2020.

JOHNS HOPKINS CORONAVIRUS RESOURCE CENTER. **COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)**. [S.I.] 2021. Disponível em: <https://coronavirus.jhu.edu/map.html>. Acesso em: 15 maio 2020.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **ImageNet Classification with Deep Convolutional Neural Networks**. *Communications of the ACM*, v. 60, n. 6, p. 84-90, 2017. Disponível em: <https://doi.org/10.1145/3065386>. Acesso em: 7 out. 2020.

LI, Y. *et al*. **Stability issues of RT-PCR testing of SARS-CoV-2 for hospitalized patients clinically diagnosed with COVID-19**. *Journal of Medical Virology*, v. 92, n. 7, p. 903-908, 2020. Disponível em: <https://onlinelibrary.wiley.com/doi/full/10.1002/jmv.25786>. Acesso em: 14 set. 2020.

MAGUOLO, G.; NANNI, L. **A Critic Evaluation of Methods for COVID-19 Automatic Detection from X-Ray Images**. abs/2004.12823, 2020. Disponível em: <https://arxiv.org/abs/2004.12823>. Acesso em: 16 nov. 2020.

MAZUROWSKI, M. A. *et al*. **Deep learning in radiology:** An overview of the concepts and a survey of the state of the art with focus on MRI. *Journal of Magnetic Resonance Imaging*, v. 49, n. 4, p. 939-954, 2018. Disponível em: <https://arxiv.org/abs/1802.08717>. Acesso em: 21 set. 2020.

MCKINNEY, W. **Data Structures for Statistical Computing in Python**. *Proceedings of the 9th Python in Science Conference*, p. 51-56, 2010. Disponível em: <https://doi.org/10.25080/Majora-92bf1922-00a>. Acesso em: 3 out. 2020.

MINAEE, S. *et al*. **Predicting COVID-19 from chest X-ray images using deep transfer learning**. *Medical image analysis*, v. 65, 2020. Disponível em: <https://doi.org/10.1016/j.media.2020.101794>. Acesso em: 11 out. 2020.

MORABITO, R; BEIJAR, N. **Enabling Data Processing at the Network Edge through Lightweight Virtualization Technologies**. *2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, 2016. Disponível em: <http://doi.org/10.1109/SECONW.2016.7746807>. Acesso em: 3 mar. 2021.

MÜLLER, A. C.; GUIDO, S. **Introduction to Machine Learning with Python**. Sebastopol: O'Reilly Media, 2018.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. **Q&A on coronaviruses (COVID-19)**. [S.I.] 2020. Disponível em: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/q-a-coronaviruses>. Acesso em: 4 out. 2020.

PLAS, Annelies van der. **x-thorax - Startradiology**. Startradiology. [S.I.] 2016. Disponível em: <https://www.startradiology.com/internships/internal-medicine/thorax/x-thorax/>. Acesso em: 16 out. 2020.

PROVOST, F.; FAWCETT, T. **Data Science for Business**. Sebastopol: O'Reilly Media, 2013.

PULLI, K. *et al*. **Real-time computer vision with OpenCV**. *Communications of the ACM*, v. 55, n. 6, p. 61-69, 2012. Disponível em: <https://doi.org/10.1145/2184319.2184337>. Acesso em: 22 nov. 2020.

RADWAN, N. **Leveraging Sparse and Dense Features for Reliable State Estimation in Urban Environments**. Tese (Doutorado em Ciências) - Universidade de Friburgo, Friburgo, Alemanha, 2019. Disponível em: <https://doi.org/10.6094/UNIFR/149856>. Acesso em: 9 out. 2020.

ROMERO, M. *et al*. **Targeted transfer learning to improve performance in small medical physics datasets**. *Medical Physics*, 2020. Disponível em: <https://doi.org/10.1002/mp.14507>. Acesso em: 9 out. 2020.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net:** Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597, 2015. Disponível em: <http://arxiv.org/abs/1505.04597>. Acesso em: 3 nov. 2020.

SANCHE, S. *et al*. **High Contagiousness and Rapid Spread of Severe Acute Respiratory Syndrome Coronavirus 2**. *Emerging Infectious Diseases*, v. 26, n. 7, p. 1470-1477, 2020. Disponível em: <https://dx.doi.org/10.3201/eid2607.200282>. Acesso em: 28 set. 2020.

SCHIAFFINO, S. *et al*. **Diagnostic Performance of Chest X-Ray for COVID-19 Pneumonia During the SARS-CoV-2 Pandemic in Lombardy, Italy**. *Journal of Thoracic Imaging*, v. 35, n. 4, p. W105-W106, 2020. Disponível em: <https://journals.lww.com/thoracicimaging/FullText/2020/07000/Diagnostic_Performance_of_Chest_X_Ray_for_COVID_19.15.aspx>. Acesso em: 19 set. 2020.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. **Activation Functions in Neural Networks**. *International Journal of Engineering Applied Sciences and Technology*, v. 4, n. 12, p. 310-316, 2020. Disponível em: <https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf>. Acesso em: 1 nov. 2020.

SILVA, G. L. **Análise da distribuição de tomógrafos no Brasil e avaliação do grau de utilização e usabilidade dos tomógrafos de um estabelecimento assistencial de saúde público**. p. 55-56, 2017.101 f. Dissertação (Mestrado em Ciências) - Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: <http://doi.org/10.14393/ufu.di.2017.480>. Acesso em: 19 set. 2020.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. *arXiv 1409.1556*, 2014. Disponível em: <https://arxiv.org/abs/1409.1556>. Acesso em: 28 set. 2020.

STIRENKO, S. *et al*., **Chest X-Ray Analysis of Tuberculosis by Deep Learning with Segmentation and Augmentation**. *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, p. 422-428, 2018. Disponível em: <https://doi.org/10.1109/ELNANO.2018.8477564>. Acesso em: 19 set. 2020.

SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V. **Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning**. *CoRR*, abs/1602.07261, 2016. Disponível em: <http://arxiv.org/abs/1602.07261>. Acesso em: 19 out. 2020.

TAN, C. *et al*. **A Survey on Deep Transfer Learning**. *CoRR*, abs/1808.01974, 2018. Disponível em: <http://arxiv.org/abs/1808.01974>. Acesso em: 26 out. 2020.

TARTAGLIONE, E. *et al*. **Unveiling COVID-19 from Chest X-ray with deep learning**: a hurdles race with small data. *International Journal of Environmental Research and Public Health*, v. 17, 2020. Disponível em: <https://doi.org/10.3390/ijerph17186933>. Acesso em: 10 out. 2020.

THE DEEPRADIOLOGY TEAM. **Pneumonia Detection in Chest Radiographs**. *CoRR*, abs/1811.08939, 2018. Disponível em: <http://arxiv.org/abs/1811.08939>. Acesso em: 21 nov. 2020.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL 12.7 Documentation**. *PostgreSQL*, 2021. Disponível em: <https://www.postgresql.org/files/documentation/pdf/12/postgresql-12-US.pdf>. Acesso em: 11 fev. 2021.

VAN ROSSUM, G.; DRAKE, F. L. **The Python language reference**. Hampton, NH: Python Software Foundation, 2010.

W3C. **Web Services Architecture**. [S.I.] 2004. Disponível em: <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>. Acesso em: 14 nov. 2020.

WADA, D. T.; RODRIGUES, J. A. H.; SANTOS, M. K. **Aspectos técnicos e roteiro de análise da radiografia de tórax**. *Medicina (Ribeirao Preto Online)*, v. 52, n. supl1., p. 5-15, 2019. Disponível em: <https://doi.org/10.11606/issn.2176-7262.v52isupl1.p5-15>. Acesso em: 6 out. 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*, v. 10, n. 1, 2020. Disponível em: <https://doi.org/10.1038/s41598-020-76550-z>. Acesso em: 27 fev. 2021.

WASKOM, M. *et al*. **mwaskom/seaborn**. [S.I.] 2020. Disponível em: <https://doi.org/10.5281/zenodo.592845>. Acesso em: 11 out. 2020.

WERBOS, P. J. **Backpropagation through time:** what it does and how to do it. *Proceedings of the IEEE*, v. 78, n. 10, p. 1550-1560, 1990. Disponível em: <http://doi.org/10.1109/5.58337>. Acesso em: 30 out. 2020.

WIELPÜTZ, M. O. *et al*. **Radiological Diagnosis in Lung Disease**. *Deutsches Arzteblatt international*, v. 111, n. 11, p. 181-187, 2014. Disponível em: <https://doi.org/10.3238/arztebl.2014.0181>. Acesso em: 19 set. 2020.

WU, J. T.; LEUNG, K.; LEUNG, G. M. **Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China:** a modelling study. *The Lancet*, v. 395, n. 10225, p. 689-697, 2020. Disponível em: <https://doi.org/10.1016/S0140-6736(20)30260-9>. Acesso em: 4 out. 2020.

XIAO, A. T.; TONG, Y. X.; ZHANG, S. **False negative of RT-PCR and prolonged nucleic acid conversion in COVID-19:** Rather than recurrence. *Journal of Medical Virology*, v. 92, p. 1755-1756, 2020. Disponível em: <https://onlinelibrary.wiley.com/doi/full/10.1002/jmv.25855>. Acesso em: 15 set. 2020.

XIE, X. *et al*. **Chest CT for Typical Coronavirus Disease 2019 (COVID-19) Pneumonia:** Relationship to Negative RT-PCR Testing. *Radiology*, v. 296, n. 2, p. E41-E45, 2020. Disponível em: <https://pubs.rsna.org/doi/10.1148/radiol.2020200343>. Acesso em: 19 set. 2020.

XU, Z. *et al*. **Pathological findings of COVID-19 associated with acute respiratory distress syndrome**. *The Lancet Respiratory Medicine*, v. 8, n. 4, p. 420-422, 2020. Disponível em: <https://doi.org/10.1016/S2213-2600(20)30076-X>. Acesso em: 5 out. 2020.

YANG, K. *et al*. **Towards Fairer Datasets**: Filtering and Balancing the Distribution of the People Subtree in the ImageNet Hierarchy. *ImageNet*. Disponível em: <http://image-net.org/update-sep-17-2019>. Acesso em: 26 out. 2020.