INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS

CAMPUS MANAUS DISTRITO INDUSTRIAL

CURSO ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VINICIUS LOUREIRO CAVALCANTE

USO DE UMA REDE NEURAL CONVOLUCIONAL PARA DETECÇÃO DE COVID-19 AUTOMÁTICA ATRAVÉS DE IMAGENS DE RAIO-X

MANAUS - AM

2023

VINICIUS LOUREIRO CAVALCANTE

USO DE UMA REDE NEURAL CONVOLUCIONAL PARA DETECÇÃO DE COVID-19 AUTOMÁTICA ATRAVÉS DE IMAGENS DE RAIO-X

Trabalho de Conclusão de Curso apresentado ao Instituo Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito-Industrial, Curso de Bacharelado em Engenharia de Controle e Automação para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2023

Dados Internacionais de Catalogação na Publicação (CIP)

C377u

Cavalcante, Vinicius Loureiro.
Uso de uma rede neural convolucional para detecção de COVID-19 automática através de imagens de Raio-X / Vinicius Loureiro Cavalcante. — Manaus, 2024.
71f.: il. color.

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Engenharia de Controle e Automação, 2024.
Orientador: Prof.º Alyson de Jesus dos Santos, Dr.

1. Rede neural. 2. Radiografia de tórax. 3. COVID-19. I. Santos, Alyson de Jesus dos. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.895

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734)

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 22 dias do mês de Dezembro, de 2023, às 09:30h, o(a) discente Vinicius Loureiro Cavalcante apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). Msc. Lucélia Cunha da Rocha Santos (Membro 1) e Prof(a). Michaella Socorro Bruce Fialho (Membro 2). A sessão publica de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título Uso De Uma Rede Neural Convolucional Para Detecção De COVID-19 Automática Através de Imagens de Raio-X,. Na sequencia, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 9,4 (Nove, Quatro) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 30 / 01 / 2023, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 10 h 05 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Avaliador 1: Lucélia Cunha da Rocha Santos

Prof.(a) Avaliador 2: Michaella Socorro Bruce Fialho

Discente: Vinicius Loureiro Cavalcante

AGRADECIMENTOS

Não foi fácil percorrer essa maratona que é o curso de engenharia de Controle e Automação e eu jamais teria conseguido sem o apoio da minha família, amigos e todos que me ajudaram ou contribuíram de alguma forma para essa caminhada.

Agradeço aos meus pais, Héllen e Rogério, que sempre se esforçaram para me propor condições de estudar e poder alcançar meus objetivos, mas também me apoiaram quando não encontrava motivação para continuar, me aconselharam quando necessário e, principalmente, me ouviram quando eu só precisava desabafar.

Agradeço especialmente aos meus avós, Claudete e Nerival, principais responsáveis pela minha criação na infância e adolescência, aos quais sou infinitamente grato por todo esforço que fizeram para que eu pudesse ter chego até aqui e conquistar tudo que já conquistei e ainda vou conquistar. Vocês me ensinaram muita coisa, mas, principalmente, a ser quem eu sou.

Por fim, agradeço a mim, pois não foi fácil ser eu nesses anos de graduação.

"Entre os mais fortes existem os que nasceram com um dom, e aqueles que trabalham duro... E eu sou um dos que trabalhou duro!" - ROCK, Lee

RESUMO:

Este trabalho tem como objetivo avaliar a eficácia do uso de redes neurais na detecção de COVID-19 por meio de radiografia de tórax. Com base em uma pesquisa bibliográfica, será definida a metodologia para a construção da rede neural, que será treinada com dados coletados de fontes confiáveis e analisados para avaliar a acurácia da detecção. A utilização de redes neurais pode ser uma alternativa promissora e não-invasiva para o diagnóstico de COVID-19, especialmente em regiões onde os testes de PCR são escassos ou demorados. Além disso, o uso de redes neurais pode oferecer vantagens em relação a outras formas de diagnóstico, como a tomografia computadorizada (TC), pois as radiografias de tórax são mais amplamente disponíveis e menos onerosas. No entanto, é importante considerar as limitações e desafios encontrados no uso de redes neurais para esse fim, como a falta de especificidade em casos leves ou assintomáticos e a necessidade de equipamentos de qualidade e profissionais treinados para interpretar as imagens. Com este estudo, espera-se contribuir para o avanço do diagnóstico de COVID-19 por meio de métodos não-invasivos e eficazes, além de identificar possíveis limitações e desafios no uso de redes neurais para esse fim.

Palavras-chave: COVID-19, rede neural, radiografia de tórax, diagnóstico, acurácia.

ABSTRACT:

This study aims to evaluate the effectiveness of using neural networks in the detection of COVID-19 through chest X-rays. Based on a literature review, the methodology for building the neural network will be defined, and it will be trained with data collected from reliable sources and analyzed to evaluate the accuracy of detection. The use of neural networks can be a promising and non-invasive alternative for the diagnosis of COVID-19, especially in regions where PCR tests are scarce or time-consuming. Additionally, the use of neural networks may offer advantages over other forms of diagnosis, such as computed tomography (CT), as chest radiographs are more widely available and less costly. However, it is important to consider the limitations and challenges encountered in using neural networks for this purpose, such as the lack of specificity in mild or asymptomatic cases and the need for quality equipment and trained professionals to interpret the images. This study aims to contribute to the advancement of COVID-19 diagnosis through non-invasive and effective methods, as well as to identify possible limitations and challenges in using neural networks for this purpose.

Keywords: COVID-19, neural network, chest radiography, diagnosis, accuracy.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. **INTRODUÇÃO**

A pandemia de COVID-19 tem sido um dos maiores desafios da saúde pública mundial nas últimas décadas. Desde o surgimento do primeiro caso em Wuhan, China, em dezembro de 2019, o vírus se espalhou rapidamente pelo mundo, causando milhões de mortes e mudando drasticamente a vida de bilhões de pessoas em todo o planeta. Uma das maiores dificuldades no combate à doença é o diagnóstico precoce e preciso, a fim de permitir a rápida identificação de casos e a tomada de medidas para evitar a disseminação do vírus.

A radiografia de tórax é uma das técnicas de imagem mais amplamente utilizadas no diagnóstico de COVID-19, devido à sua acessibilidade e ao fato de que muitos pacientes com COVID-19 apresentam anormalidades na imagem torácica. No entanto, a interpretação das imagens de raios-X é muitas vezes difícil e pode ser afetada pela experiência do radiologista.

Nesse contexto, a utilização de redes neurais para auxiliar no diagnóstico de COVID-19 por meio de radiografia de tórax tem se mostrado uma alternativa promissora e não invasiva. Segundo Kheyroddin et al. (2021), "a rede neural artificial tem a capacidade de imitar o processo de aprendizado humano e, assim, pode ser usada para diagnosticar doenças de uma forma mais precisa e rápida" (p. 1). Vários estudos têm explorado a utilização de redes neurais para a detecção de COVID-19 por meio de radiografia de tórax, com resultados encorajadores (Narin et al., 2021; Shu et al., 2020).

Dito isso, o objetivo deste trabalho é avaliar a eficácia do uso de redes neurais na detecção de COVID-19 por meio de radiografia de tórax. Para isso, será realizada uma pesquisa bibliográfica para embasar o estudo, seguida da definição da metodologia para a construção da rede neural. A partir daí, serão coletados os dados necessários para o treinamento da rede, que serão analisados para avaliar a acurácia da detecção. Serão também identificadas as principais limitações e desafios encontrados no uso de redes neurais para esse fim. Espera-se, com este trabalho, contribuir para o avanço do diagnóstico de COVID-19 por meio de métodos não-invasivos e eficazes.

1.1. JUSTIFICATIVA

A pandemia de COVID-19 tem causado um grande impacto na sociedade, na economia e na saúde pública em todo o mundo. O diagnóstico

precoce e preciso da doença é fundamental para o controle da disseminação do vírus e para a adoção de medidas terapêuticas e preventivas adequadas. No entanto, o diagnóstico clínico baseado em sintomas e testes laboratoriais pode levar a resultados falsos negativos ou positivos, o que pode levar a uma subnotificação ou superestimação do número de casos.

A radiografia de tórax é uma das técnicas de imagem mais amplamente utilizadas no diagnóstico de COVID-19, devido à sua acessibilidade e baixo custo. No entanto, a interpretação das imagens de raios-X é muitas vezes difícil e pode ser afetada pela experiência do radiologista. Nesse contexto, a utilização de redes neurais para auxiliar no diagnóstico de COVID-19 por meio de radiografia de tórax tem se mostrado uma alternativa promissora e não invasiva. Segundo Narin et al. (2021), "a utilização de técnicas de aprendizado de máquina tem o potencial de melhorar a precisão do diagnóstico e permitir um rastreamento mais rápido e eficiente de pacientes suspeitos de COVID-19" (p. 1207).

Além disso, a pandemia de COVID-19 tem gerado uma grande demanda por soluções tecnológicas para o diagnóstico da doença. Nesse sentido, a utilização de redes neurais para a detecção de COVID-19 por meio de radiografia de tórax pode representar uma oportunidade para a aplicação de técnicas avançadas de inteligência artificial no campo da saúde. Segundo Shu et al. (2020), "o desenvolvimento de sistemas de detecção automatizados para COVID-19 pode fornecer uma ferramenta útil para ajudar os profissionais de saúde a lidar com a pandemia em andamento" (p. 1).

Contudo, o presente trabalho se justifica pela importância de se avaliar a eficácia do uso de redes neurais na detecção de COVID-19 por meio de radiografia de tórax, a fim de contribuir para o avanço do diagnóstico da doença por meio de métodos não invasivos e eficazes.

1.2. OBJETIVO

O objetivo geral deste trabalho é contribuir para o avanço do diagnóstico de COVID-19 por meio de métodos não invasivos e eficazes, utilizando a tecnologia de redes neurais na análise de radiografias de tórax. Considerando a alta propagação do vírus e a escassez de testes em diversos locais, o diagnóstico por imagem pode ser uma alternativa viável e acessível para a identificação da doença. Além disso, as redes neurais são uma ferramenta de aprendizado de máquina capaz de realizar análises mais precisas e rápidas, o que pode ser fundamental para a detecção precoce e tratamento adequado da COVID-19.

Nesse sentido, este trabalho busca avaliar a eficácia do uso dessas redes neurais na detecção da doença por meio de radiografia de tórax e contribuir para o avanço das pesquisas nessa área.

1.3. OBJETIVOS ESPECÍFICOS

Com intuito de realizar um trabalho satisfatório listamos os seguintes objetivos específicos:

1. Realizar uma revisão bibliográfica sobre o diagnóstico de COVID-19 por meio de radiografia de tórax e o uso de redes neurais na detecção da doença;
2. Definir a metodologia para a construção da rede neural;
3. Coletar os dados necessários para o treinamento da rede neural;
4. Treinar a rede neural e avaliar sua acurácia na detecção de COVID-19;
5. Identificar as principais limitações e desafios encontrados no uso de redes neurais para a detecção de COVID-19 por meio de radiografia de tórax;
6. Contribuir para o avanço do diagnóstico de COVID-19 por meio de métodos não invasivos e eficazes.

1.4. METODOLOGIA

A metodologia deste trabalho será baseada em uma pesquisa geral aplicada, combinando abordagens qualitativas e quantitativas. A pesquisa bibliográfica será realizada para embasar o estudo, com levantamento de artigos, teses, dissertações e outros trabalhos relacionados ao diagnóstico de COVID-19 por meio de radiografia de tórax e uso de redes neurais na detecção da doença. Essa revisão bibliográfica será fundamental para o embasamento teórico e definição dos procedimentos metodológicos a serem seguidos. A abordagem qualitativa será utilizada para a análise e discussão dos resultados, buscando identificar as principais limitações e desafios encontrados no uso de redes neurais para a detecção de COVID-19 por meio de radiografia de tórax. Por outro lado, a abordagem quantitativa será adotada para a coleta e análise dos dados necessários para o treinamento da rede neural, visando avaliar a acurácia da detecção. A partir daí, será possível definir a metodologia para a construção da rede neural e coletar os dados necessários para o seu treinamento.

De acordo com Yin (2015), a combinação de abordagens qualitativas e quantitativas pode ser fundamental para a compreensão de um fenômeno complexo, permitindo uma análise mais completa e detalhada dos resultados obtidos. Além disso, a pesquisa geral aplicada busca a solução de problemas

práticos e aplicados, por meio da construção e aplicação de teorias. Nesse sentido, a pesquisa adotada neste trabalho busca contribuir para a melhoria do diagnóstico de COVID-19, aplicando a teoria de redes neurais na detecção da doença por meio de radiografia de tórax.

1.5. ESTRUTURA DO TCC

O restante do trabalho será apresentado da seguinte forma: no Capítulo 2, será apresentada a fundamentação teórica necessária para o desenvolvimento dele, contendo explicações sobre inteligência artificial, visão computacional e técnicas para classificação de imagens, além das ferramentas de software utilizadas, bem como a linguagem de programação e as bibliotecas. O capítulo 3 aborda alguns trabalhos semelhantes já realizados e busca discutir os métodos utilizados e seus resultados. No capítulo 4, discute-se os métodos utilizados neste presente trabalho, demonstra-se de que forma foi realizada a análise e pré-processamento das imagens, construção e treinamento da rede neural para o objetivo proposto. O capítulo 5 trará resultados e discussões com o intuito de avaliar a eficiência do trabalho aqui proposto e, por fim, o capítulo 6 apresentará a conclusão do trabalho.

2. **REFERENCIAL TEÓRICO**

O intuito deste capítulo é abordar conceitos teóricos necessários para as discussões práticas do TCC.

2.1. EPIDEMIOLOGIA E CARACTERÍSTICAS CLÍNICAS DA COVID-19

A COVID-19 é uma doença infecciosa causada pelo coronavírus SARS-CoV-2, que teve início em Wuhan, China, em dezembro de 2019, e se espalhou rapidamente pelo mundo, tornando-se uma pandemia global em março de 2020 (HUANG et al., 2020). A transmissão ocorre principalmente através de gotículas respiratórias, com o contágio sendo favorecido em ambientes fechados e aglomerações (CDC, 2021).

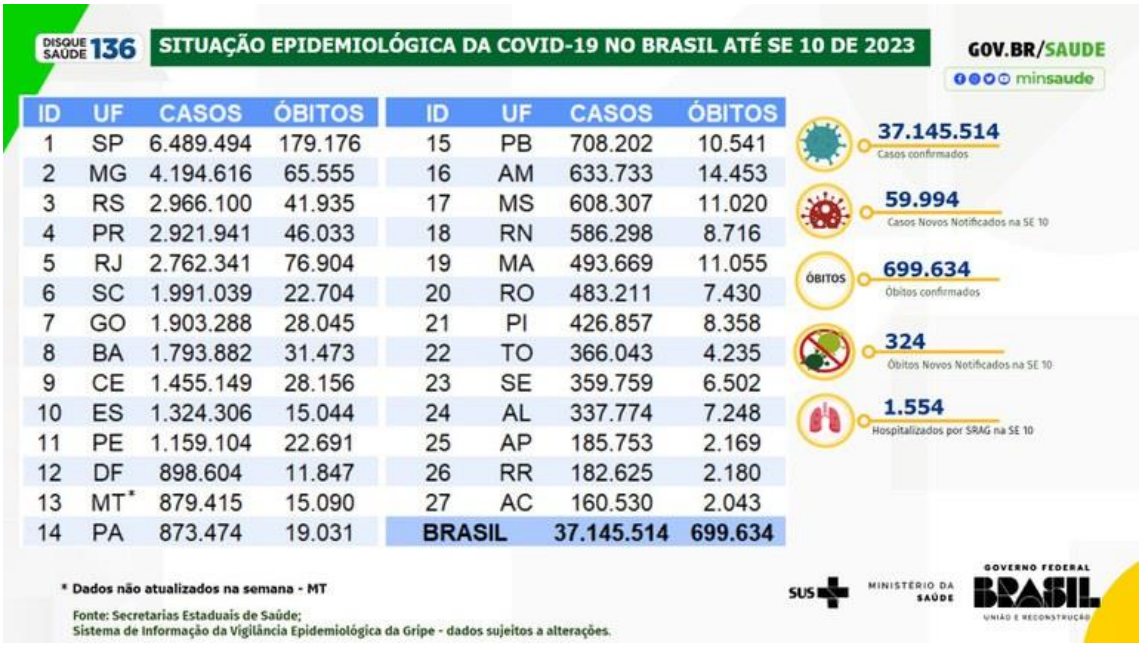
A maioria dos pacientes infectados pela COVID-19 apresenta sintomas leves ou moderados, como febre, tosse seca e fadiga, enquanto outros podem não apresentar sintomas (CARFÌ et al., 2020). No entanto, cerca de 15% dos casos evoluem para doenças graves, como pneumonia, síndrome do desconforto respiratório agudo (SDRA), falência de múltiplos órgãos e morte (GUAN et al., 2020).

A taxa de mortalidade da COVID-19 varia entre os países e grupos populacionais, com a idade e as comorbidades sendo fatores de risco para evolução grave da doença (GRASSELLI et al., 2020). A taxa de mortalidade na população geral varia de 1% a 2%, enquanto em idosos e pacientes com doenças crônicas, como diabetes e doenças cardiovasculares, a taxa é significativamente maior (GUAN et al., 2020).

A COVID-19 também pode afetar o sistema nervoso central e periférico, causando sintomas neurológicos como dor de cabeça, tontura, alterações do olfato e paladar, além de manifestações neurológicas graves, como convulsões, encefalopatia e derrame cerebral (ELLUL et al., 2020).

A compreensão dos aspectos epidemiológicos e clínicos da COVID-19 é fundamental para o desenvolvimento de estratégias eficazes de prevenção, diagnóstico e tratamento da doença. A identificação precoce dos casos e a implementação de medidas de controle são essenciais para mitigar a disseminação do vírus e reduzir o impacto da pandemia na saúde pública (WHO, 2020).

Figura 1 - Estatísticas sobre o COVID-19 no Brasil em 2023.



ID	UF	CASOS	ÓBITOS	ID	UF	CASOS	ÓBITOS
1	SP	6.489.494	179.176	15	PB	708.202	10.541
2	MG	4.194.616	65.555	16	AM	633.733	14.453
3	RS	2.966.100	41.935	17	MS	608.307	11.020
4	PR	2.921.941	46.033	18	RN	586.298	8.716
5	RJ	2.762.341	76.904	19	MA	493.669	11.055
6	SC	1.991.039	22.704	20	RO	483.211	7.430
7	GO	1.903.288	28.045	21	PI	426.857	8.358
8	BA	1.793.882	31.473	22	TO	366.043	4.235
9	CE	1.455.149	28.156	23	SE	359.759	6.502
10	ES	1.324.306	15.044	24	AL	337.774	7.248
11	PE	1.159.104	22.691	25	AP	185.753	2.169
12	DF	898.604	11.847	26	RR	182.625	2.180
13	MT*	879.415	15.090	27	AC	160.530	2.043
14	PA	873.474	19.031	BRASIL		37.145.514	699.634

Fonte - Ministério da Saúde (https://www.gov.br/saude/pt-br/coronavirus/informes-semanais-covid-19/covid-19-situacao-epidemiologica-do-brasil-ate-a-se-10-de-2023, acesso em julho de 2023).

2.2. FUNDAMENTOS DA RADIOLOGIA E IMAGENS DE RAIO X DE TÓRAX

A radiologia é uma especialidade médica que utiliza diversas modalidades de imagem para diagnóstico, incluindo a radiografia, ou raio X. Segundo a Sociedade Brasileira de Radiologia (2017), a radiografia é uma técnica de imagem que utiliza radiação ionizante para produzir uma imagem bidimensional de estruturas internas do corpo humano. A imagem é produzida pela absorção diferencial de raios X pelo tecido corporal, sendo que estruturas densas, como osso, aparecem brancas na imagem, enquanto tecidos moles aparecem em tons de cinza.

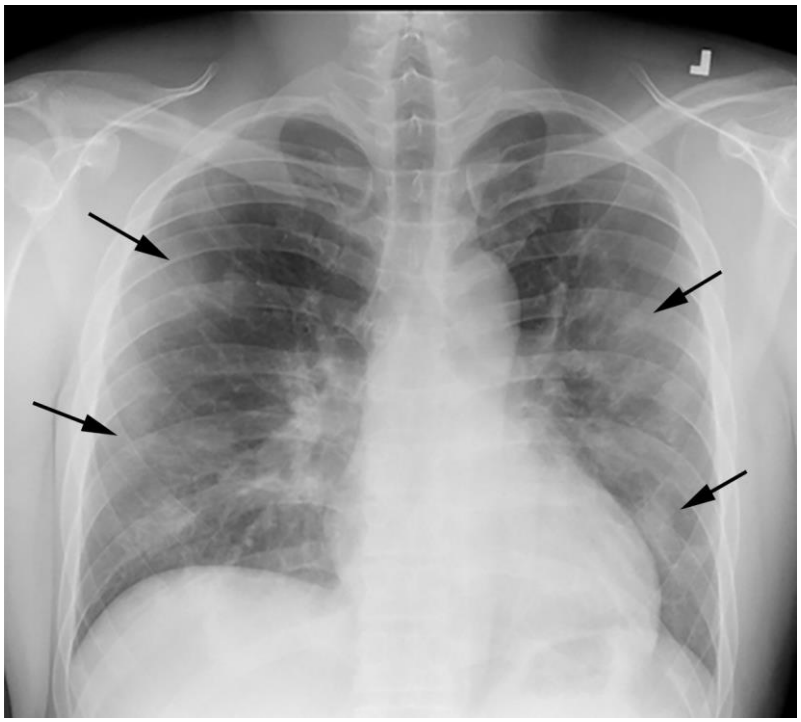
A radiografia de tórax é uma das mais comuns em radiologia e tem sido amplamente utilizada para auxiliar no diagnóstico de diversas patologias pulmonares, incluindo a COVID-19. Através da imagem de raio X, é possível avaliar a presença de infiltrados e opacidades pulmonares, que são comuns em casos de pneumonia, incluindo os casos de COVID-19. De acordo com Wong et al. (2020), as imagens de raio x de tórax são uma ferramenta útil para

triagem de pacientes com suspeita de COVID-19, podendo auxiliar na identificação de padrões radiológicos sugestivos da doença.

Para a interpretação correta das imagens de raio X, é fundamental o conhecimento da anatomia do tórax e das estruturas que podem ser visualizadas na imagem. Além disso, é necessário compreender os diferentes padrões radiológicos que podem ser observados em casos de doenças pulmonares. Segundo o Conselho Federal de Medicina (2018), os principais padrões radiológicos observados em imagens de raio X de tórax são: infiltrado, que se refere a uma área opaca que substitui o ar nos pulmões; consolidação, que é a opacidade completa do pulmão afetado; atelectasia, que se refere a uma área de pulmão colapsada; e derrame pleural, que é a presença de líquido entre as camadas da pleura.

Para uma análise mais precisa das imagens de raio X, é possível utilizar ferramentas de processamento de imagens, incluindo técnicas de segmentação, que permitem a identificação e separação das diferentes estruturas presentes na imagem. Segundo Rajpurkar et al. (2017), o uso de algoritmos de segmentação pode auxiliar na detecção de patologias pulmonares em imagens de raio X de tórax, contribuindo para um diagnóstico mais preciso e eficiente.

Figura 2 – Raio X de tórax de um paciente COVID positivo.



Fonte - Interação Diagnóstica (http://www.interacaodiagnostica.com.br/noticias/Galeria-de-fotos-mostra-como-o-coronavirus-aparece-em-imagens-medicas-1891, acesso em julho de 2023).

2.3. PYTHON

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos, que vem se destacando no desenvolvimento de aplicações de Inteligência Artificial (IA) e Aprendizado de Máquina (AM). De acordo com Brownlee (2019), Python é amplamente utilizado no campo de IA e AM devido à sua sintaxe simples e clara, bem como pela sua vasta coleção de bibliotecas específicas para essas áreas.

Uma das principais razões para o uso de Python em IA e AM é a grande variedade de bibliotecas disponíveis, que facilitam o trabalho do desenvolvedor. Segundo Géron (2019), a biblioteca mais conhecida e amplamente utilizada em Python para AM é o *scikit-learn*, que contém uma vasta coleção de algoritmos de AM e ferramentas de pré-processamento de dados. Além disso, há outras bibliotecas importantes como *TensorFlow* e *Keras*, que são amplamente utilizadas para construir redes neurais.

Outra vantagem do Python em IA e AM é a sua grande comunidade de desenvolvedores. De acordo com Chollet (2018), Python é uma linguagem de programação muito popular no mundo da IA e do AM, e possui uma comunidade de usuários muito ativa e colaborativa. Isso significa que, além de muitas bibliotecas disponíveis, há também muitos tutoriais, fóruns e projetos *open-source* que podem ser usados como referência e ajudam a resolver problemas comuns.

Além disso, Python é uma linguagem de programação versátil e permite uma fácil integração com outras tecnologias e linguagens. De acordo com Oliveira (2018), Python pode ser utilizado em conjunto com outras ferramentas, como *Hadoop* e *Spark*, que são muito utilizadas no processamento de grandes conjuntos de dados. Isso torna Python uma escolha popular para o desenvolvimento de aplicações de *Big Data*.

Por fim, outra vantagem do Python em IA e AM é a sua facilidade de uso. De acordo com Raschka e Mirjalili (2017), Python é uma linguagem de programação fácil de aprender e utilizar, mesmo para iniciantes na área. Isso significa que a curva de aprendizado é mais rápida, o que é especialmente importante em áreas como a IA e o AM, que exigem constante atualização de conhecimentos.

Figura 3 - Logotipo da linguagem de programação Python.

Fonte - Wikipedia (https://pt.m.wikipedia.org/wiki/Ficheiro:Python_logo_and_wordmark.svg, acesso em julho de 2023).

2.4. VISÃO COMPUTACIONAL

Visão Computacional é uma área de estudo que se dedica a desenvolver algoritmos e técnicas capazes de interpretar e compreender imagens e vídeos digitais. De acordo com Szeliski (2010), a visão computacional tem como objetivo principal extrair informações úteis a partir de dados visuais, como a detecção e identificação de objetos, reconhecimento de faces, rastreamento de movimentos, entre outros.

A Visão Computacional tem diversas aplicações práticas em áreas como medicina, indústria, segurança, transporte, entre outras. Por exemplo, a Visão Computacional pode ser utilizada para monitorar a qualidade de produtos em uma linha de produção, detectar anomalias em imagens médicas e reconhecer placas de veículos em uma rodovia (Szeliski, 2010).

Uma das principais técnicas utilizadas em Visão Computacional é o Processamento de Imagens, que consiste em aplicar uma série de transformações matemáticas em imagens digitais para melhorar a qualidade da imagem ou extrair informações úteis dela. Além disso, a aprendizagem de máquina tem sido amplamente utilizada em Visão Computacional para treinar modelos capazes de reconhecer padrões em imagens e vídeos (Szeliski, 2010).

No entanto, a Visão Computacional ainda enfrenta diversos desafios, como a interpretação de imagens com baixa qualidade, a identificação de objetos em imagens complexas e a detecção de objetos em movimento. Para superar esses desafios, pesquisadores estão desenvolvendo novas técnicas de processamento de imagens, aprendizagem de máquina e inteligência artificial, além de utilizar grandes conjuntos de dados para treinar modelos mais precisos (Szeliski, 2010).

Em resumo, a Visão Computacional é uma área de estudo que tem como objetivo desenvolver algoritmos e técnicas capazes de interpretar e compreender imagens e vídeos digitais. A Visão Computacional tem diversas aplicações práticas em áreas como medicina, indústria, segurança e transporte. Apesar dos desafios, a Visão Computacional continua evoluindo com o desenvolvimento de novas técnicas de processamento de imagens e aprendizagem de máquina.

2.5. INTELIGÊNCIA ARTIFICIAL

Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de algoritmos e sistemas que podem realizar tarefas que normalmente exigem inteligência humana, como raciocínio, aprendizagem, percepção e tomada de decisões. A IA é uma área de pesquisa multidisciplinar que combina conhecimentos de várias áreas, como matemática, estatística, ciência da computação, psicologia e filosofia.

De acordo com Russel e Norvig (2010), a IA é o “estudo de agentes inteligentes”, onde um agente inteligente é definido como “um sistema que percebe seu ambiente e toma ações que maximizam suas chances de sucesso”. Essa definição enfatiza a importância da percepção e da tomada de decisões em um sistema de inteligência artificial.

Existem várias abordagens para criar sistemas de inteligência artificial, incluindo a lógica simbólica, as redes neurais e a aprendizagem de máquina.

A IA tem muitas aplicações práticas em diversas áreas, incluindo medicina, finanças, transporte, segurança, entre outras. A IA tem atraído a atenção de governos e organizações em todo o mundo, que estão investindo em pesquisa e desenvolvimento para aproveitar seu potencial.

Figura 4 - Áreas da Inteligência Artificial.

Fonte - SERPRO (https://www.serpro.gov.br/menu/noticias/noticias-2019/democratizando-a-inteligencia-artificial, acesso em julho de 2023).

2.6. APRENDIZADO DE MÁQUINA

Aprendizado de máquina (*machine learning*) é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos e modelos que podem aprender a partir de dados e realizar tarefas específicas. Segundo Alpaydin (2010), o aprendizado de máquina pode ser definido como "um método para ensinar computadores a aprender a partir de exemplos, sem terem sido explicitamente programados."

O aprendizado de máquina é baseado em uma abordagem indutiva, onde um modelo é treinado com um conjunto de dados de treinamento para aprender a relação entre as variáveis de entrada e saída. Segundo Goodfellow et al. (2016), o objetivo do aprendizado de máquina é "encontrar padrões em dados complexos e usá-los para fazer previsões ou tomar decisões".

Existem vários tipos de aprendizado de máquina, incluindo aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Segundo Mitchell (1997), no aprendizado supervisionado, o modelo é treinado com exemplos rotulados, ou seja, exemplos em que a saída desejada é conhecida. O objetivo do modelo é aprender a relação entre as variáveis de entrada e saída, para que possa prever a saída para novos exemplos não rotulados.

No aprendizado não supervisionado, o modelo é treinado com exemplos não rotulados, ou seja, exemplos em que a saída desejada não é conhecida.

Segundo Bishop (2006), o objetivo do aprendizado não supervisionado é “encontrar padrões ocultos nos dados e agrupá-los em categorias ou *clusters*”.

Já no aprendizado por reforço, o modelo é treinado com base em recompensas e punições, onde o objetivo do modelo é aprender a tomar ações que maximizam uma recompensa ao longo do tempo. Segundo Sutton e Barto (2018), o aprendizado por reforço pode ser definido como “aprender a tomar decisões em um ambiente para maximizar uma recompensa numérica”.

O aprendizado de máquina tem muitas aplicações práticas, incluindo reconhecimento de fala, visão computacional, processamento de linguagem natural, deteção de fraudes e previsão de demanda. Segundo Jordan e Mitchell (2015), o aprendizado de máquina é “uma das áreas mais excitantes da ciência da computação” e tem atraído um grande interesse da indústria e da academia.

Figura 5 - Aprendizado de máquina: subáreas e aplicações.

Fonte - AQUARELA (https://www.aquare.la/aprendizado-de-maquina-subareas-e-aplicacoes/, acesso em julho de 2023).

2.7. REDES NEURAIS ARTIFICIAIS

As Redes Neurais artificiais são compostas por unidades interconectadas chamadas neurônios, que recebem entradas e produzem uma saída. Essas conexões entre os neurônios são ponderadas por valores

numéricos chamados pesos, que são ajustados durante o treinamento para produzir a saída desejada (Lecun et al., 2015). De acordo com Esteve et al. (2017), as Redes Neurais têm sido amplamente utilizadas em diversas áreas, como reconhecimento de padrões, processamento de linguagem natural, visão computacional e previsão de séries temporais. Na medicina, as Redes Neurais têm sido aplicadas em tarefas como diagnóstico de doenças, previsão de riscos e tratamentos personalizados.

As Redes Neurais têm um grande potencial para auxiliar os profissionais de saúde no diagnóstico, tratamento e prevenção de doenças, além de ajudar na gestão de recursos e na redução de custos (Esteve et al., 2017). Um exemplo de aplicação de Redes Neurais na medicina é o diagnóstico de câncer de mama. As Redes Neurais podem ser treinadas para identificar características específicas em imagens de mamografia que indicam a presença de tumores. Essas Redes Neurais podem ser utilizadas como uma ferramenta de apoio ao diagnóstico, auxiliando os radiologistas na detecção precoce da doença (Lecun et al., 2015).

No entanto, a aplicação de Redes Neurais na medicina enfrenta desafios, como a necessidade de dados precisos e confiáveis para treinamento e validação do modelo. Além disso, a interpretabilidade dos modelos de Redes Neurais é um desafio, já que muitas vezes é difícil entender como o modelo chegou a uma determinada decisão (Gulrajani et al., 2016). Para superar esses desafios, pesquisadores estão desenvolvendo novas técnicas de interpretabilidade de modelos de Redes Neurais, como a visualização dos pesos das conexões entre os neurônios. Além disso, a utilização de Redes Neurais híbridas, que combinam a capacidade de aprendizado das Redes Neurais com a interpretabilidade de modelos lineares, pode ser uma alternativa para a aplicação de Redes Neurais na medicina (Esteve et al., 2017).

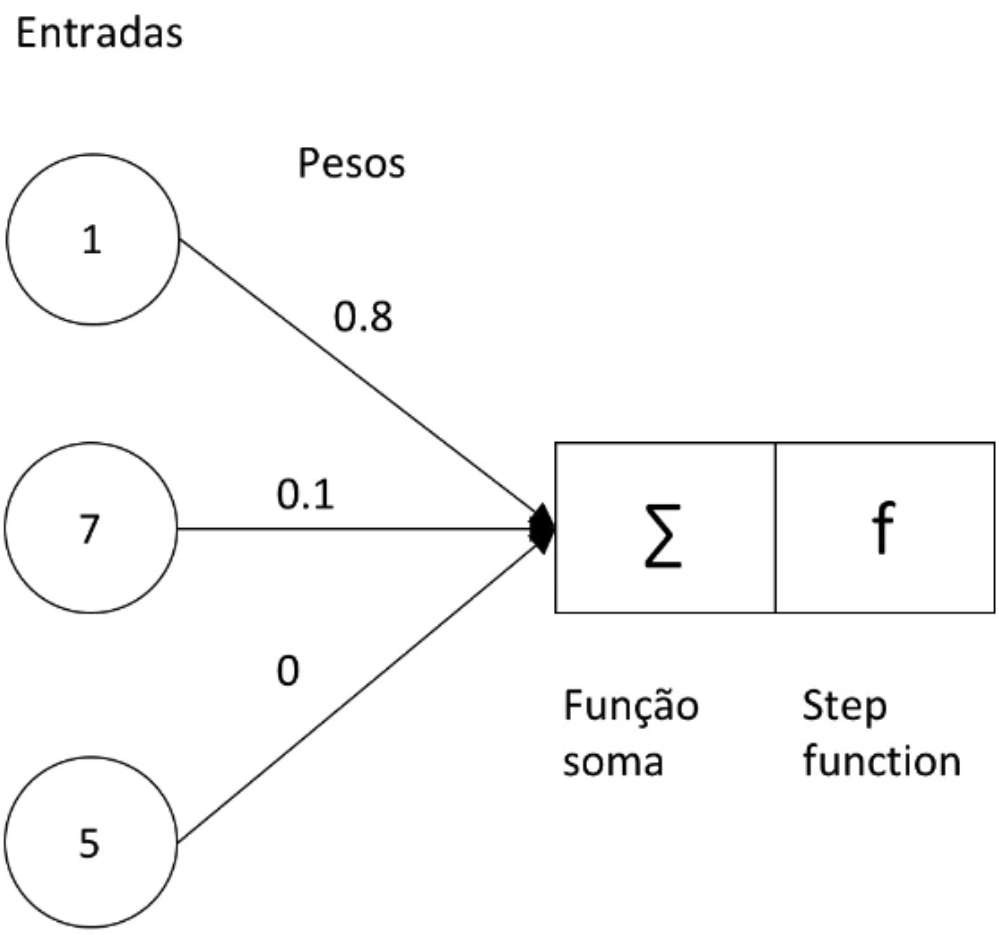
As Redes Neurais são um modelo computacional inspirado no cérebro humano, que tem sido amplamente utilizado em diversas áreas, incluindo a medicina. Apesar dos desafios, as Redes Neurais têm um grande potencial para auxiliar os profissionais de saúde na prevenção, diagnóstico e tratamento de doenças (Lecun et al., 2015).

2.7.1. PERCEPTRON DE UMA CAMADA

No contexto das redes neurais artificiais os Perceptrons De Uma Camada, ou *Single Layer Perceptrons* (SLP), é o modelo de rede mais simples,

consistindo numa única camada de neurônios artificiais. Segundo Rosenblatt (1958), “Um Perceptron é um dispositivo que faz decisões lineares sobre inputs representados por sinais binários”.

Figura 6 - Perceptron de uma camada.



Fonte - IA Expert Academy (https://iaexpert.academy/topic/perceptron-de-uma-camada-19/, acesso em julho de 2023).

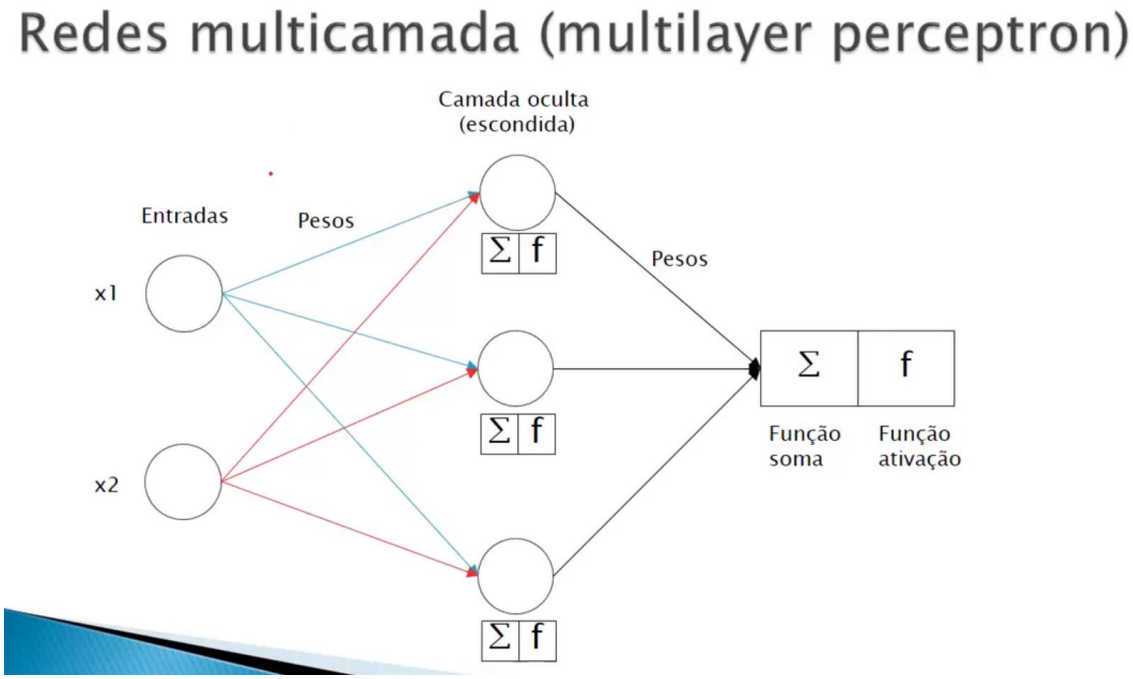
Cada neurônio do Perceptron recebe entradas ponderadas e aplica uma função de ativação, comumente a função degrau (*step function*), para gerar uma saída. Durante seu treinamento, os pesos associados às entradas são atualizados com base nos erros gerados. (Rosenblatt, 1958)

Porém, o Perceptron é um modelo linear que só é aplicável para situações linearmente separáveis, como afirma Rosenblatt (1958), “O Perceptron de uma camada só pode aprender a realizar tarefas que possam ser expressas como um problema de classificação linear”.

2.7.2. REDES MULTICAMADAS

As redes multicamadas, ou *multilayer Perceptrons*, são arquiteturas de redes neurais que incluem uma, ou mais, camadas ocultas entre a camada de entrada e a camada de saída, conforme representado na imagem abaixo.

Figura 7 - Ilustração de uma Rede Multicamada.



Fonte - IA Expert Academy (https://iaexpert.academy/topic/introducao-a-redes-multicamada/, acesso em julho de 2023).

Para Goodfellow et al. (2016), “As redes neurais profundas, também chamadas de redes multicamadas, são redes com múltiplas camadas de unidades, o que lhes permite aprender representações hierárquicas dos dados”. Por conta dessa hierarquia, a rede neural consegue capturar características mais complexas dos dados á medida que as camadas ocultas são ativadas.

É possível com que a rede aprenda representações não lineares e capture padrões complexos nos dados de entrada por conta da utilização do cálculo de ativações, onde cada neurônio recebe as entradas ponderadas da camada anterior e aplica uma função de ativação na etapa de propagação direta.

Ainda, a utilização do algoritmo de retropropagação, que calcula o gradiente da função de perda, permite uma melhor atualização dos pesos

durante o treinamento. Segundo Lecun et al. (1998), “O algoritmo de retropropagação é a base do treinamento de redes multicamadas”.

2.7.3. FUNÇÕES DE ATIVAÇÃO

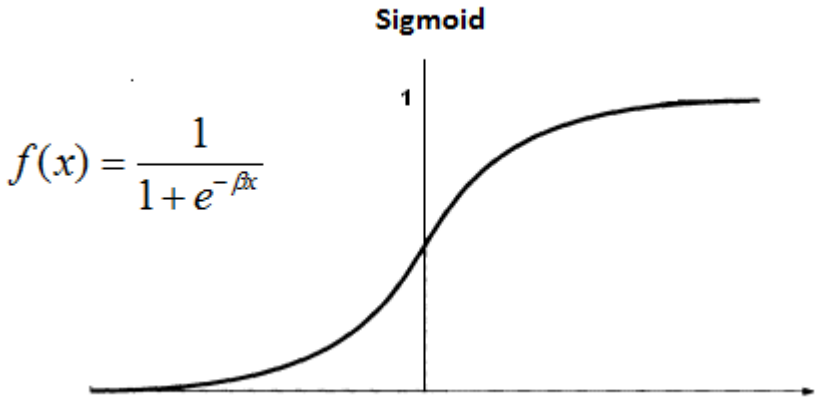
As funções de ativação são uma parte essencial das redes neurais convolucionais e têm um papel importante na introdução de não-linearidade nas camadas da rede. De acordo com Goodfellow et al. (2016), “a função de ativação de um neurônio define a saída desse neurônio dado um determinado conjunto de entradas”.

Algumas funções de ativação mais comuns incluem a função sigmoide, a função tangente hiperbólica (tanh), a função ReLU (*Rectified Linear Unit*) e a função *Softmax*.

A função sigmoide é uma função de ativação utilizada em redes neurais, que tem como objetivo transformar um valor real em um valor no intervalo entre 0 e 1 (Goodfellow et al., 2016). Essa função é frequentemente utilizada em problemas de classificação binária, onde a saída deve indicar a probabilidade de pertencimento a uma das duas classes.

Devido às suas limitações, a função sigmoide vem sendo substituída em algumas aplicações por outras funções de ativação, como a ReLU, que apresenta uma melhor convergência e reduz o problema de saturação dos gradientes. (Krizhevsky et al., 2012).

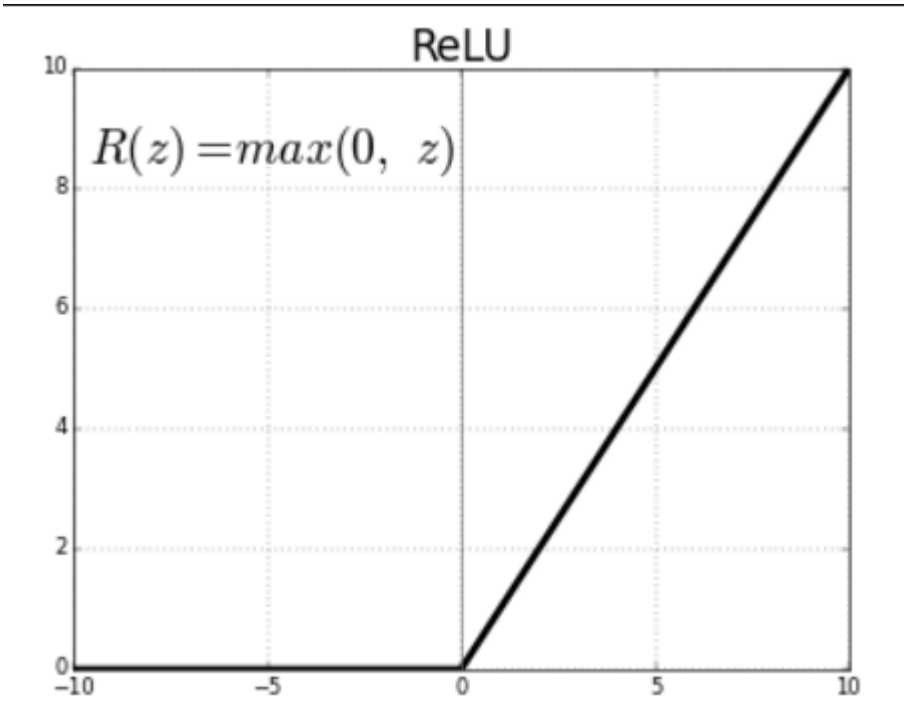
Figura 8 – Função de ativação Sigmoide.



Fonte - SABEDORIA RAREFEITA (https://sabedoriararefeita.wordpress.com/2016/02/24/redes-neurais-parte-2-preguica/, acesso em julho de 2023).

A função ReLU é uma das mais populares e amplamente utilizadas nas redes neurais convolucionais, uma vez que é simples e eficiente computacionalmente. Segundo Glorot et al. (2011), “as funções ReLU têm a vantagem de acelerar a convergência do treinamento em comparação com funções de ativação não lineares mais suaves, como a tangente hiperbólica.

Figura 9 - Função de ativação ReLU.



Fonte - Research Gate (https://www.researchgate.net/figure/Grafico-da-funcao-de-ativacao-ReLU-Conforme-os-valores-se-aproximam-de-zero-menos_fig4_342369912, acesso em julho de 2023).

Em geral, a escolha da função de ativação depende do problema em questão e da arquitetura da rede. Segundo Goodfellow et al. (2016), “não existe uma função de ativação que funcione melhor em todos os casos”. Por isso é importante testar diferentes funções de ativação e avaliar seu desempenho em relação à acurácia e ao tempo de treinamento da rede.

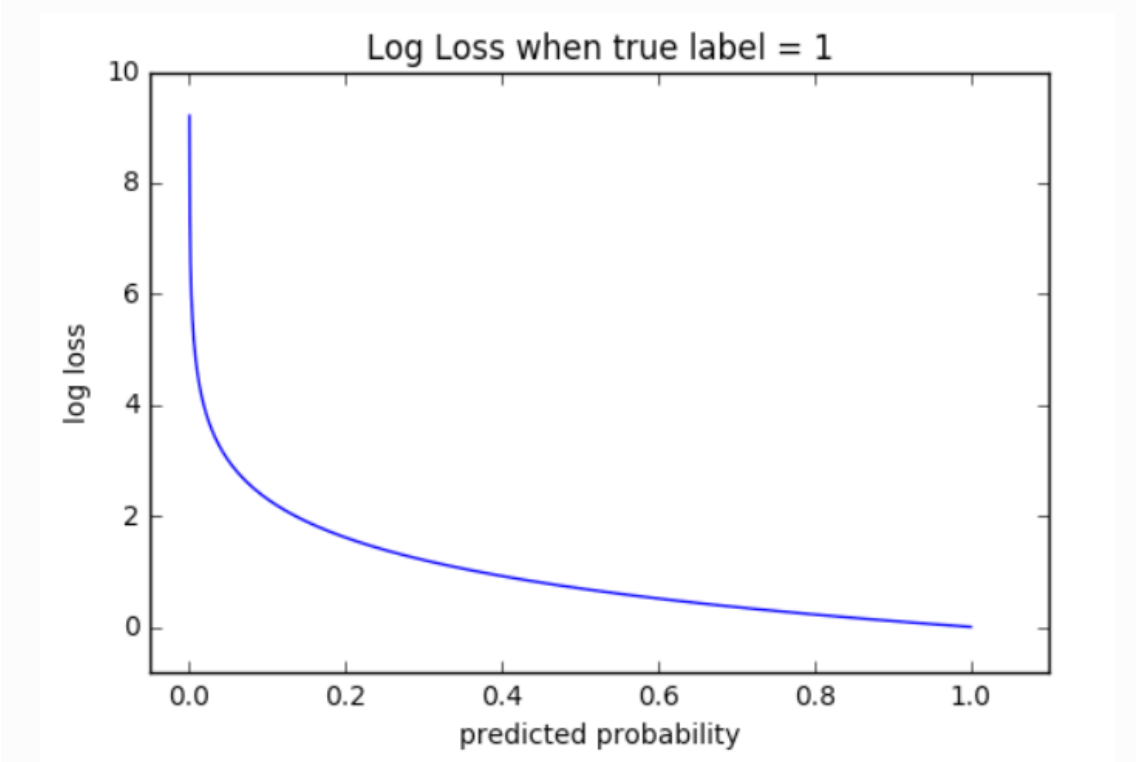
As funções de ativação são aplicadas em cada neurônio da rede, após a etapa de convolução ou *pooling*, e antes da camada totalmente conectada. Elas são responsáveis por introduzir não-linearidades na rede, permitindo que a rede aprenda relações mais complexas entre os dados de entrada e a saída desejada.

2.7.4. CÁLCULO DO ERRO

Para que as redes neurais artificiais consigam aprender, é necessário realizar o cálculo do erro, para tal é preciso comparar as saídas preditas pela rede e os valores alvo desejado. A diferença entre esses valores é o erro que é quantificado através de uma função de perda (*loss function*). Durante o treinamento da rede, o objetivo é minimizar a perda, sendo necessário ajustar os parâmetros de rede afim de alcançar esse objetivo.

Para problemas de classificação, a função de perda mais comumente usada é a Entropia Cruzada (*Cross Entropy*). Goodfellow et al. (2016) afirmam que “A entropia cruzada é uma função de perda comum para problemas de classificação”. Essa função mede a divergência entre a distribuição de probabilidade predita pela rede e a distribuição real dos rótulos.

Figura 10 - Representação da função de perda Cross Entropy ou Log Loss.



Fonte - ML Glossary Documentation (https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html, acesso em julho de 2023).

2.7.5. DESCIDA DO GRADIENTE

Para que a função de perda seja minimizada, a rede utiliza o algoritmo de descida do gradiente. Segundo Goodfellow et al. (2016), “A descida do gradiente é um algoritmo comum de otimização que é utilizada para treinar redes neurais”.

O processo de descida do gradiente envolve os seguintes passos:

- Inicialização dos parâmetros;
- Cálculo do gradiente;
- Atualização dos parâmetros.

Sequencialmente, os pesos da rede são inicializados de forma aleatória ou segundo algum valor pré-estabelecido, a função de perda é diferenciada em relação a cada parâmetro da rede, resultando no gradiente que indica a direção

em que função de perda está se elevando mais rapidamente, ajudando a rede a entender a direção que deve ser seguida para atualizar os parâmetros. A atualização acontece através da multiplicação do gradiente pela taxa de aprendizagem (passo), subtraindo esse valor dos parâmetros existentes, assim, os parâmetros se movem na direção oposta ao gradiente, minimizando gradualmente a função de perda.

Vale destacar que a taxa de aprendizagem é um hiperparâmetros crucial na descida do gradiente, pois controla o tamanho do passo em cada iteração. Já que uma taxa de aprendizagem baixa, pode tornar o aprendizado lento, porém uma taxa muito alta, pode causar divergências e oscilações no resultado da rede.

Sendo assim, a definição da taxa de aprendizagem é importante, mas não é tão simples, conforme Ruder (2016), “Determinar a taxa de aprendizagem apropriada é um desafio, e os praticantes geralmente fazem tentativa e erro”.

2.7.6. BACKPROPAGATION

Segundo Rumelhart et al. (1986), “A técnica de retropropagação permite o ajuste dos pesos da rede de acordo com o gradiente da função de erro com respeito a esses pesos”.

A técnica de *backpropagation* é um algoritmo importantíssimo para o treinamento de redes neurais, utilizado para calcular o gradiente da função de perda em relação aos parâmetros da rede. Consiste em propagar o erro da camada de saída até as camadas anteriores, atualizando os pesos da rede de acordo com o gradiente calculado.

O processo de *backpropagation* funciona da seguinte forma:

1. *Feedforward*: durante essa etapa, os sinais de entrada são propagados pelas camadas da rede neural. Cada camada recebe os dados da camada anterior e aplica uma transformação não linear aos sinais de entrada. Para Rumelhart et al. (1986), “As entradas são propagadas para a frente através da rede, camada por camada, para produzir as saídas”.
2. Cálculo do erro: após o *feedforward*, o erro é calculado. De acordo com Rumelhart et al. (1986), “O erro é calculado comparando-se a saída da rede com a saída desejada”.
3. Retropropagação do erro: o erro calculado é retropropagado através da rede, atualizando-se os pesos. Segundo Rumelhart et al. (1986), “O erro é então propagado de volta pela rede,

permitindo que a contribuição de cada unidade de processamento seja calculada".

4. Atualização dos pesos: Conforme Rumelhart et al. (1986) mencionaram, "Os pesos são ajustados proporcionalmente ao gradiente da função de erro com relação aos pesos".

2.8. APRENDIZADO PROFUNDO

Aprendizado Profundo, também conhecido como *deep learning*, é um subcampo do aprendizado de máquina que usa algoritmos de redes neurais profundas para modelar e solucionar problemas complexos. Segundo Goodfellow et al. (2016), o aprendizado profundo pode ser definido como "a capacidade de aprender representações de dados de alto nível em camadas sucessivas, onde cada camada aprende uma representação mais abstrata da entrada do que a camada anterior".

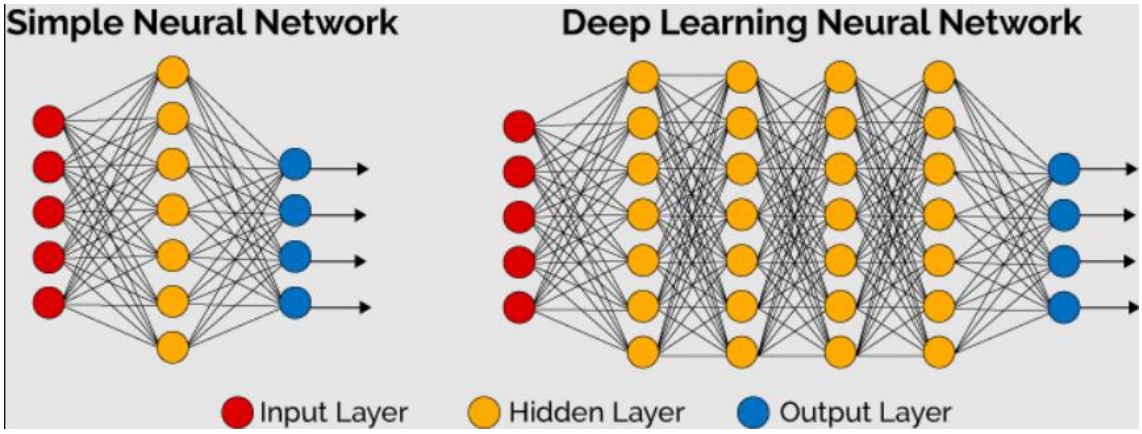
As redes neurais profundas são modelos matemáticas compostos por várias camadas de neurônios interconectados, que são capazes de aprender a partir de dados para realizar tarefas como reconhecimento de imagens, processamento de linguagem natural e reconhecimento de fala. Segundo Bengio et al. (2013), o aprendizado profundo é "uma área importante da inteligência artificial e do aprendizado de máquina, que tem potencial para transformar muitas áreas da ciência e da indústria".

Uma das principais vantagens do aprendizado profundo é sua capacidade de aprender recursos ou características relevantes automaticamente a partir dos dados, eliminando a necessidade de recursos humanos para selecionar manualmente as características relevantes. Segundo LeCun et al. (2015), "em vez de depender de recursos cuidadosamente projetados, a rede neural profunda aprende características que são relevantes para a tarefa em questão, a partir dos próprios dados".

As redes neurais profundas são compostas por várias camadas, cada uma com vários neurônios interconectados. Cada camada processa os dados de entrada de maneira progressivamente mais abstrata, permitindo que a rede neural profunda aprenda representações complexas de dados. Segundo Goodfellow et al. (2016), "as redes neurais profundas podem aprender a reconhecer padrões em dados de maneira hierárquica, capturando conceitos simples em camadas mais baixas e conceitos mais complexos em camadas mais altas".

De acordo com Schmidhuber (2015), o aprendizado profundo é “o principal catalisador do atual renascimento da inteligência artificial”.

Figura 11 - Comparação entre uma rede neural simples e uma rede neural com aprendizado profundo.



Fonte - Deep Learning Book (https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/, acesso em julho de 2023).

2.9. REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (*Convolutional Neural Networks* ou *CNNs*) são um tipo de rede neural profunda que têm sido amplamente utilizadas em aplicações de visão computacional, como reconhecimento de objetos e classificação de imagens. Elas são baseadas no conceito de convolução, que envolve a aplicação de filtros a uma imagem para extrair características úteis. De acordo com Goodfellow et al. (2016), “as redes neurais convolucionais são particularmente eficazes em problemas de reconhecimento de imagem porque elas podem aprender automaticamente características importantes a partir dos dados brutos”.

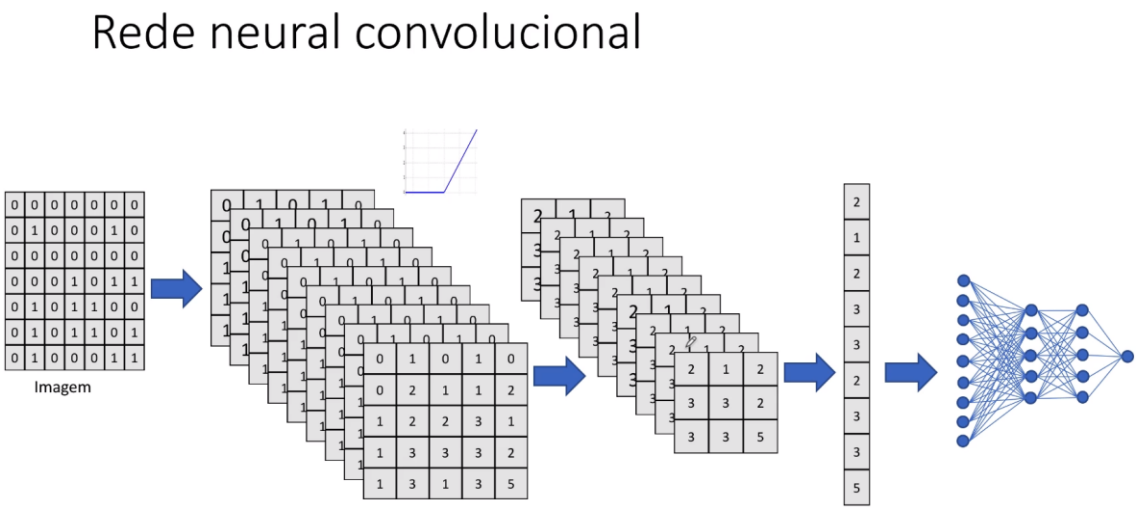
As CNNs são compostas por camadas de convolução, camadas de pooling e camadas totalmente conectadas. A camada de convolução é responsável por extrair características da imagem de entrada, enquanto a camada de *pooling* reduz a dimensão da imagem para melhor a eficiência computacional e evitar o *overfitting*. A camada totalmente conecta é responsável por gerar a saída final da rede, com base nas características extraídas nas camadas anteriores.

De acordo com Zhang e Zhao (2021), as CNNs têm sido utilizadas em várias aplicações de visão computacional, como reconhecimento facial, classificação de imagens médicas, detecção de objetos e segmentação de imagens. Além disso, eles afirmam que as CNNs são muito robustas em

relação a ruído e variações de iluminação, tornando-as uma escolha popular em muitas aplicações de visão computacional.

O treinamento de uma CNN envolve a aplicação do algoritmo de retropropagação, que ajusta os pesos da rede com base no erro de previsão. Isso é feito através do cálculo do gradiente da função de perda em relação aos pesos da rede. De acordo com LeCun et al. (2015), “o sucesso das CNNs em várias tarefas de visão computacional pode ser atribuído em grande parte à sua habilidade em aprender representações hierárquicas das imagens de entrada através do treinamento em grandes conjuntos de dados”.

Figura 10 - Ilustração de uma Rede Neural Convolucional.



Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

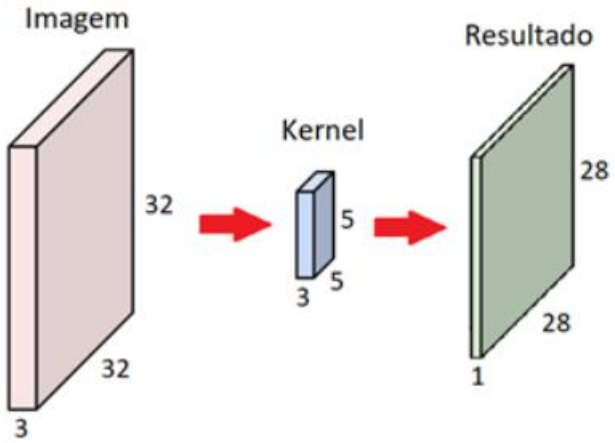
2.9.1. CONVOLUÇÕES

As convoluções são um conceito fundamental nas redes neurais convolucionais (CNNs) e são usadas para extrair características úteis de imagens. De acordo com Goodfellow et al. (2016), “a convolução é uma operação matemática entre duas funções que produz uma terceira função que expressa como uma das funções é afetada pela outra”. Em outras palavras, a convolução pode ser vista como um processo de sobreposição de uma matriz de pesos, ou filtro, na imagem de entrada, onde cada elemento do filtro é multiplicado pelos elementos correspondentes da imagem e o resultado é somado para produzir um único valor na saída.

Tomando como exemplo uma imagem com dimensões 32x32x3, onde a altura e a largura têm 32 pixels, com 3 canais de cores, um filtro com dimensões 5x5x3, onde a altura e a largura possuem 5 pixels, com

obrigatoriamente a mesma quantidade de canais de cores. O resultado é uma matriz convoluída de dimensões 28x28x1.

Figura 11 - Ilustração do processo de Convolução.



Fonte - Viceri (https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/, acesso em julho de 2023).

Levando em consideração uma imagem NxN, um filtro FxF e passo S, as dimensões da matriz resultante, GxG, podem ser calculadas pela fórmula seguinte:

Figura 12 - Cálculo das dimensões de uma matriz após a operação de Convolução.

$$G = \frac{N - F}{S} + 1$$

Fonte - Viceri (https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/, acesso em julho de 2023).

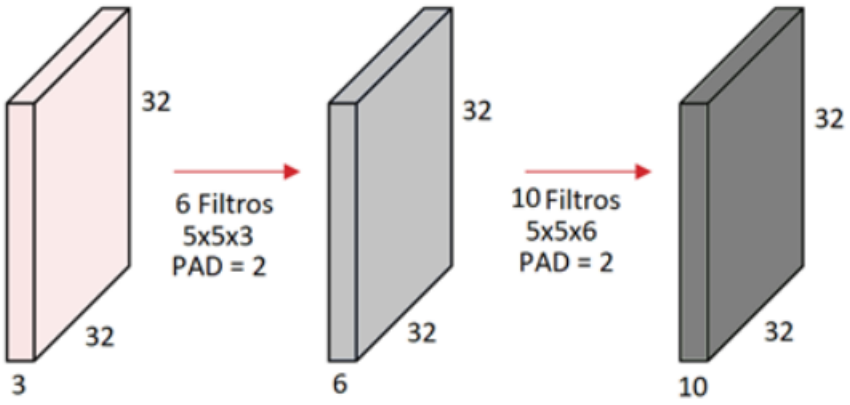
2.9.2. CAMADA DE CONVOLUÇÃO

Para Zhang e Zhao (2021), “a camada de convolução é a principal camada para a extração de características” (p.173). Sendo a base para a arquitetura de redes neurais, a camada de convolução desempenha um papel importante na aprendizagem de recursos e na classificação de imagens. (Zhang e Zhao, 2021).

Para ajustar a resolução espacial da saída da convolução, os hiperparâmetros, como o tamanho do passo (*stride*) e o preenchimento (*padding*) são utilizados.

De forma geral, a camada de convolução consiste na aplicação de uma série de filtros. Por exemplo, supondo uma imagem de dimensões 32x32x3, adicionada de um *padding* de valor 2, isto é, adicionando 2 camadas de zero acima, abaixo e nas laterais da imagem, ao aplicar uma primeira convolução com seis filtros 5x5x3, obtém-se uma imagem de 32x32x6. Após uma segunda convolução, com dez filtros 5x5x6, obtém-se uma imagem de 32x32x10, como ilustrado na figura abaixo.

Figura 13 - Resultado da operação de convolução numa imagem 32x32x3.



Fonte - Viceri (https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/, acesso em julho de 2023).

Nota-se, portanto, que as dimensões de altura e largura da entrada não se alteram devido a escolha adequada dos hiperparâmetros. Conforme explica Géron (2019), o tamanho da saída da convolução pode ser menor que a imagem de entrada, dependendo do tamanho do filtro e do passo, enquanto o preenchimento pode ser usado para manter as dimensões espaciais da imagem de saída.

Por fim, a camada de convolução é geralmente seguida por outras camadas em uma CNN, como a camada de *pooling* e a camada totalmente conectada, que processam a saída da convolução e ajudam a realizar a classificação ou a previsão da tarefa em questão.

2.9.3. FILTRO

O filtro, também conhecido como kernel, é um elemento fundamental em redes neurais convolucionais para a extração de características de entrada. O filtro trata-se de uma matriz numérica que se desloca pela imagem de entrada, produzindo uma saída que representa as características da imagem que o filtro está procurando.

Para Zhang e Zhao (2021), “o filtro é a parte mais importante da convolução” (p.178), e diferentes tamanhos e valores de elementos do filtro podem resultar em diferentes características sendo extraídas da imagem de entrada.

Outros autores, como LeCun et al. (1998), descrevem o uso de diferentes tipos de filtros em CNNs, incluindo filtros de borda, filtros de linha e filtros Gaussianos. Esses filtros podem ser usados em diferentes camadas da rede para detectar diferentes tipos de características da imagem.

Além do filtro, outros hiperparâmetros importantes em CNNs incluem o tamanho do passo (*stride*) e o preenchimento (*padding*) da imagem de entrada. Esses hiperparâmetros podem afetar a resolução espacial da saída da convolução e, portanto, influenciar a capacidade da rede em detectar características específicas da imagem. De acordo com Deng e Yu (2014), “o tamanho do filtro, o tamanho do passo e o preenchimento são hiperparâmetros importantes que precisam ser ajustados para cada aplicação específica” (p.675).

2.9.4. STRIDE

O *Stride* que, por sua vez, determina o deslocamento do filtro de convolução na imagem de entrada. Para Zhang e Zhao (2021), “*Stride* é a

distância entre duas posições sucessivas em que o filtro é aplicado na imagem de entrada".

2.9.5. PADDING

O *Padding* que, para Goodfellow et al. (2016), "o *Padding* é a adição de zeros adicionais em torno da borda da imagem de entrada, de modo que a imagem de saída tenha a mesma dimensão espacial que a imagem de entrada".

Figura 14 - Fórmula para determinar o padding.

$$P = \frac{F - 1}{2}$$

Fonte - Viceri (https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/, acesso em julho de 2023).

Com isso, o tamanho da imagem de saída da camada convolucional é:

Figura 15 - Fórmula para determinar o tamanho da imagem de saída.

$$G = \frac{N + 2*P - F}{S} + 1$$

Fonte1 - Viceri (https://viceri.com.br/insights/entendendo-de-vez-a-convolucao-base-para-processamento-de-imagens/, acesso em julho de 2023).

2.9.6. CAMADA DE POOLING

A camada de *pooling* é comumente utilizada em arquiteturas de Redes Neurais Convolucionais após a camada de convolução. Ela tem como objetivo reduzir a dimensão espacial da representação da imagem, diminuindo o número de parâmetros e computações necessárias para processar as informações nas camadas seguintes da rede.

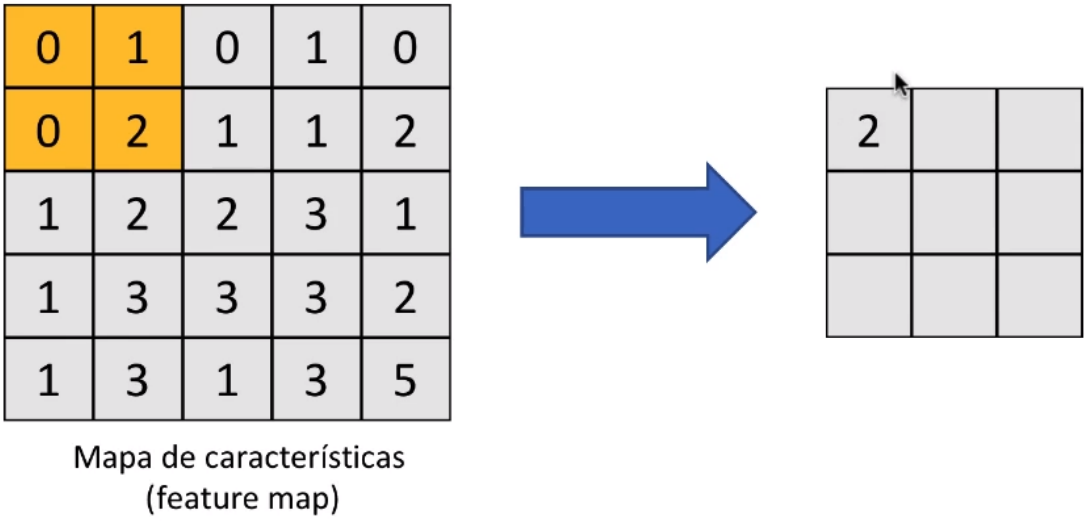
Conforme explica Géron (2019), a camada de *pooling* divide a imagem de entrada em regiões não sobrepostas e realiza uma operação estatística, como o máximo ou a média, sobre cada região, gerando uma única saída que representa aquela região. Essa saída é então utilizada como entrada para a

próxima camada da rede. Com isso, a camada de *pooling* pode reduzir a dimensão espacial da imagem, preservando as informações mais importantes para a classificação da imagem.

Há diferentes tipos de *pooling*, como o máximo (*max-pooling*), que retorna o valor máximo da região, e a média (*average-pooling*), que retorna a média dos valores da região. Outros tipos de pooling, como o *pooling* L2, também são utilizados em algumas aplicações.

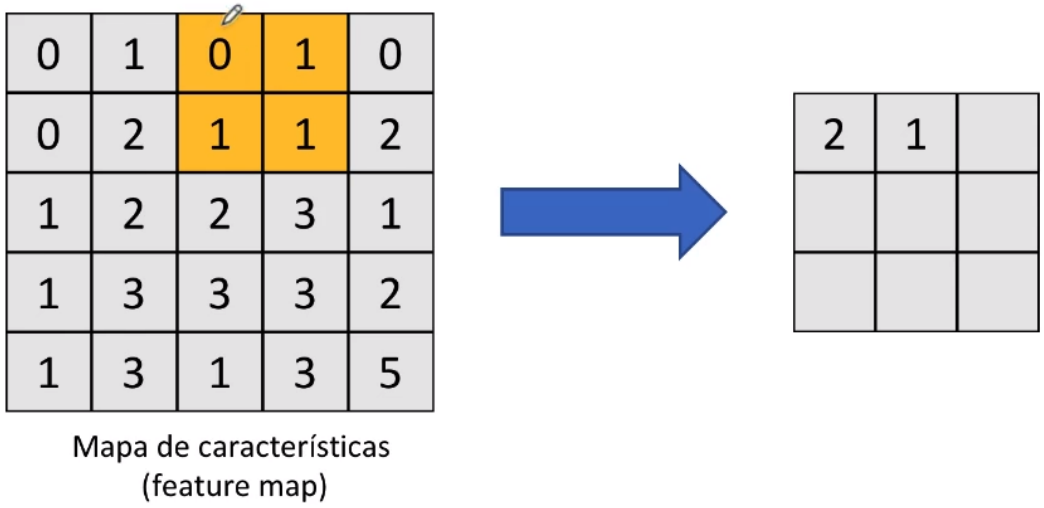
Abaixo segue a ilustração da aplicação do max-pooling a um mapa de características:

Figura 16 - Primeira etapa da aplicação de max-pooling.



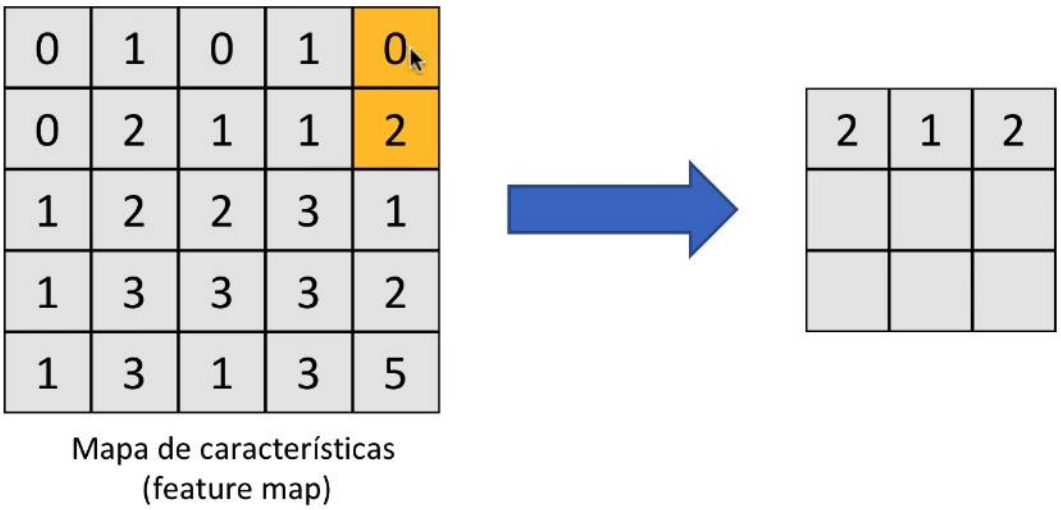
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 12 - Segunda etapa da aplicação de max-pooling.



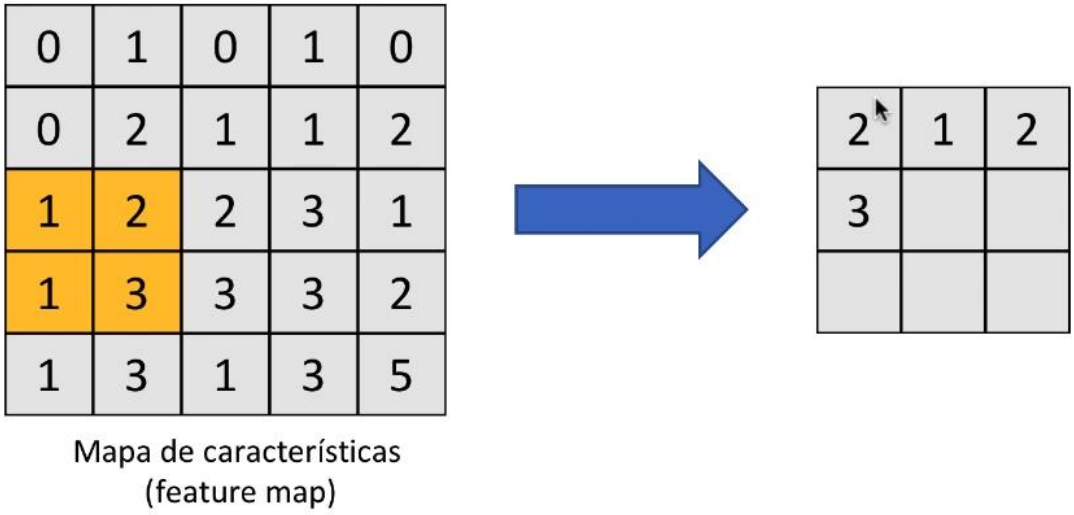
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 13 - Terceira etapa da aplicação de max-pooling.



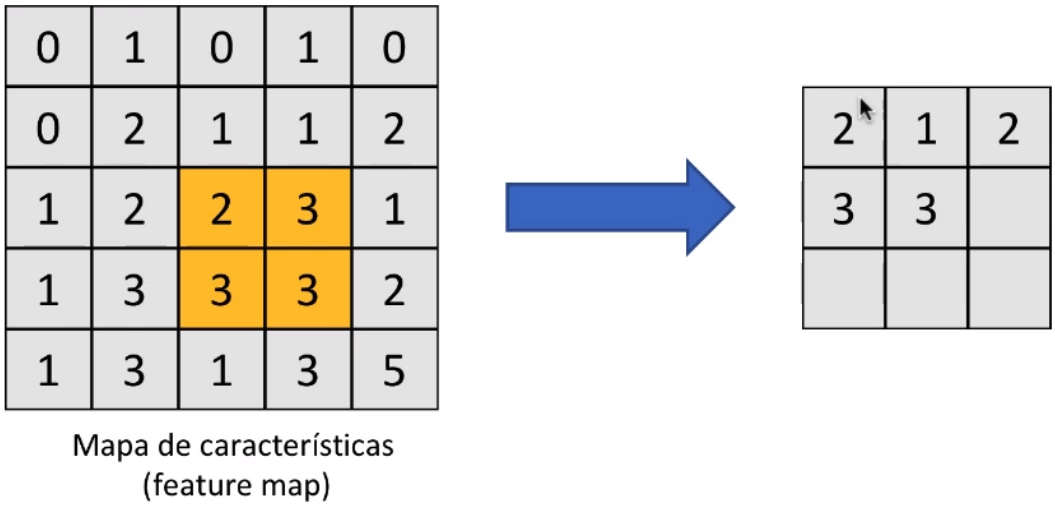
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 14 - Quarta etapa da aplicação de max-pooling.



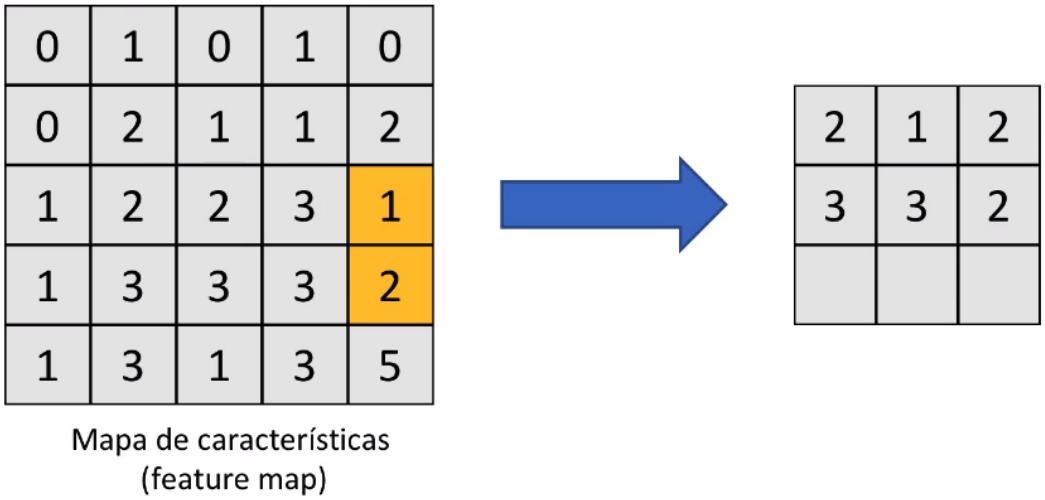
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 15 - Quinta etapa da aplicação de max-pooling.



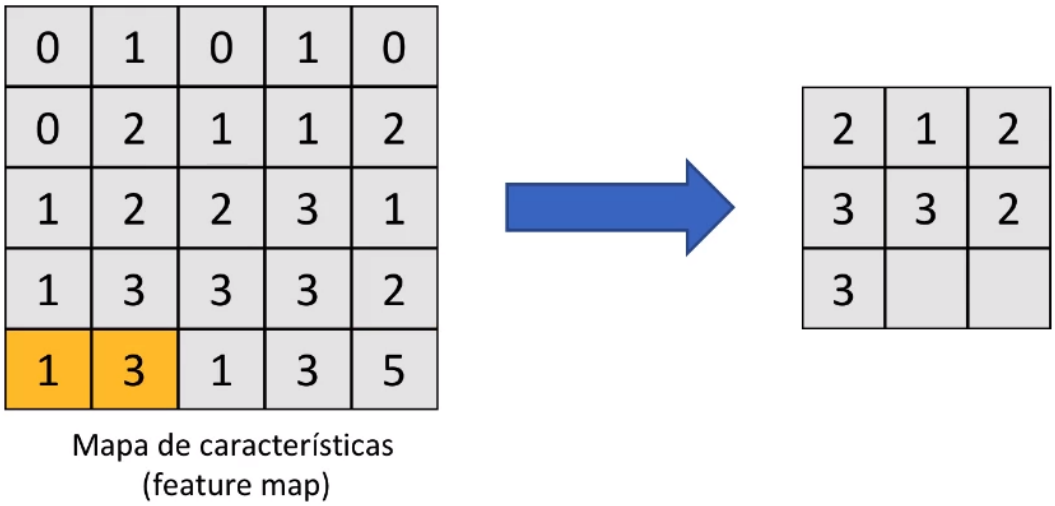
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 16 - Sexta etapa da aplicação de max-pooling.



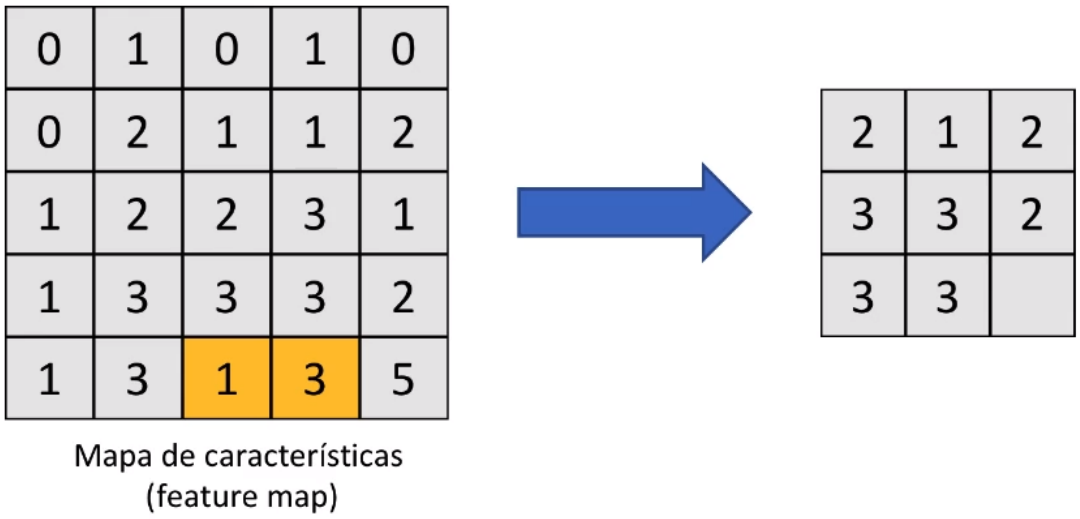
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 17 - Sétima etapa da aplicação de max-pooling.



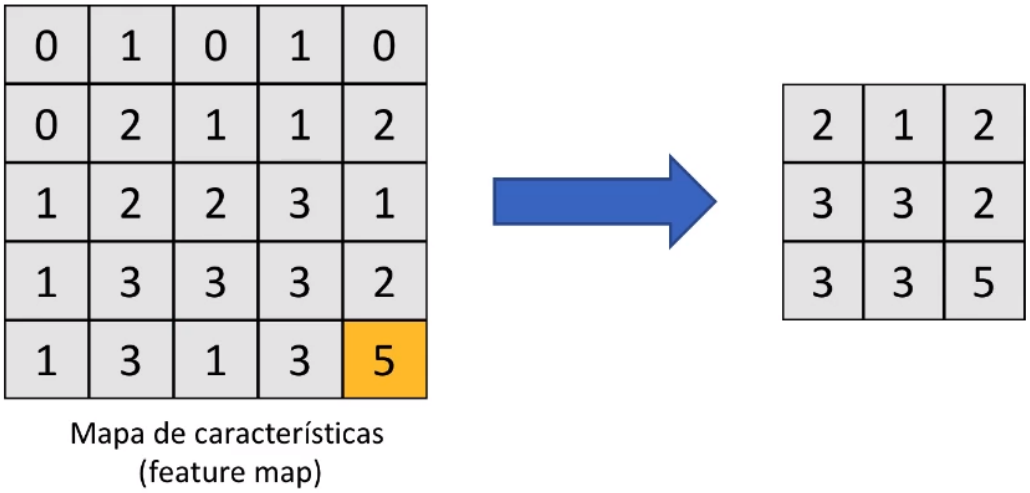
Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 18 - Oitava etapa da aplicação de max-pooling.



Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Figura 19 - Última etapa da aplicação de max-pooling.



Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

Observa-se que a aplicação do *max-pooling* reduziu a dimensionalidade do mapa de características, inicialmente 5x5 e posteriormente 3x3.

Por mais que a utilização da camada de pooling seja opcional, vale destacar que ela pode evitar o *overfitting* nas CNNs, ao reduzir o número de parâmetros da rede e criar uma representação mais geral da imagem.

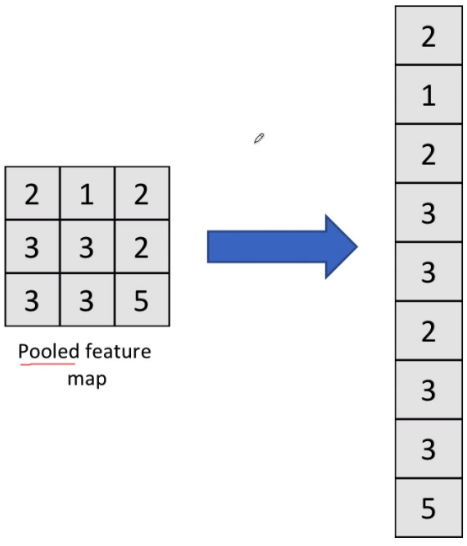
Conforme explica Goodfellow et al. (2016), o *overfitting* ocorre quando a rede se ajusta muito bem aos dados de treinamento, mas tem uma baixa capacidade de generalização para novos dados de teste. Isso pode acontecer em redes muito complexas, com muitos parâmetros e poucos dados de treinamento. Logo, a camada de *pooling* pode ajudar a evitar esse problema, pois, conforme Géron (2019), a operação de *pooling* escolhe apenas o máximo ou a média da região, ignorando os detalhes precisos de localização, tornando a rede menos sensível a pequenas mudanças na posição dos objetos na imagem, o que pode ajudar a evitar o *overfitting*.

2.9.7. FLATTENING

A técnica de *flattening* (ou achatamento) é amplamente utilizada em redes neurais convolucionais (CNNs) como parte do processo de transição entre as camadas convolucionais e as camadas totalmente conectadas. Essa técnica tem o objetivo de converter o tensor tridimensional de características gerado pelas camadas convolucionais em um vetor unidimensional, que pode ser alimentado em uma camada totalmente conectada.

LeCun et al. (1998) ressalta a importância do achatamento nas CNNs: "O achatamento é uma etapa crítica em CNNs, pois permite que as informações espaciais sejam transformadas em uma representação vetorial apropriada para a classificação". Essa transformação facilita o processamento posterior da informação e ajuda a reduzir a dimensionalidade dos dados.

Figura 20 - Flattening aplicado após o pooling, transforma uma matriz num vetor.



Fonte - IA Expert Academy (https://iaexpert.academy/, acessado em julho de 2023).

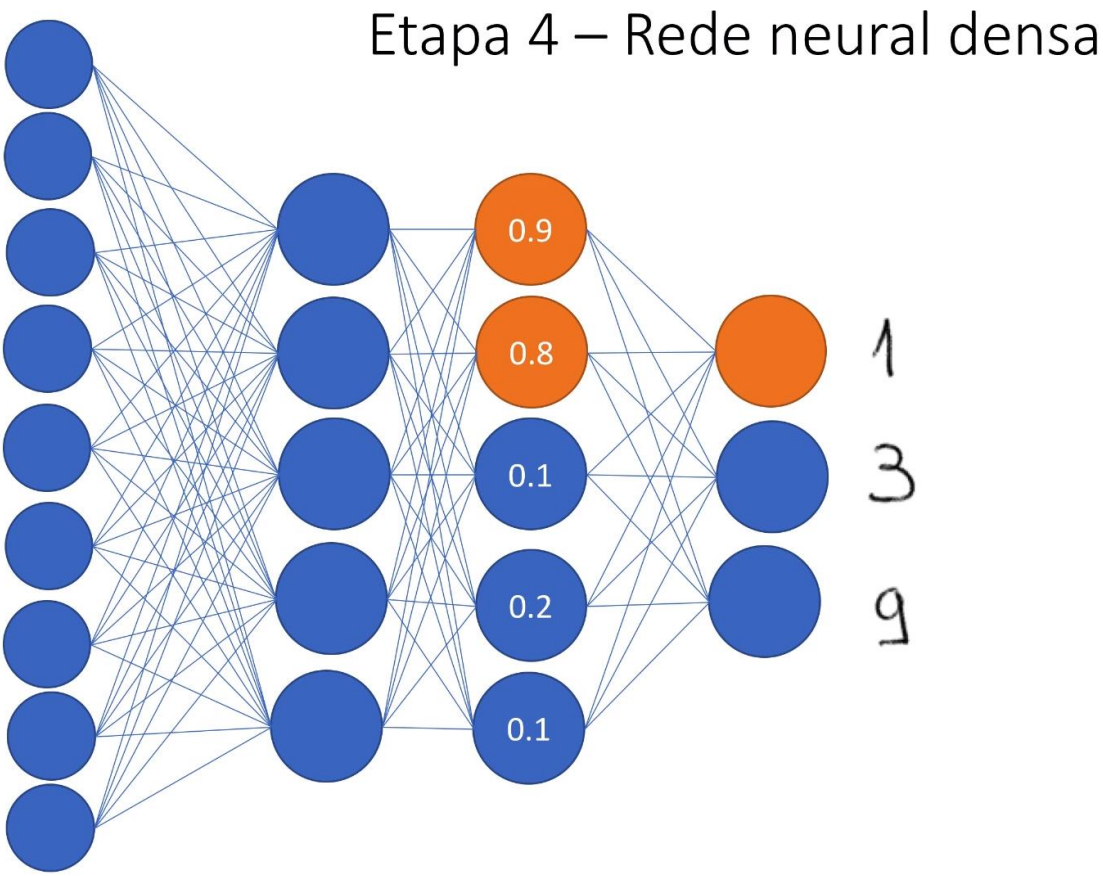
2.9.8. CAMADA TOTALMENTE CONECTADA

Em uma CNN, a camada totalmente conectada é geralmente colocada no final da rede, recebendo as características achatadas das camadas de convolução e *pooling*. Cada neurônio na camada totalmente conectada é conectado a todos os neurônios da camada anterior, com pesos que são ajustados durante o treinamento da rede.

No entanto, a inclusão de uma camada totalmente conectada pode levar a *overfitting*, especialmente se a rede for muito grande. A técnica de *dropout*, introduzida por Srivastava et al. (2014), é frequentemente usada para reduzir o *overfitting* em redes neurais convolucionais, incluindo a camada totalmente conectada.

A ideia por trás do *dropout* é simplesmente a de “desligar” aleatoriamente algumas unidades da rede durante cada etapa de treinamento. Segundo Srivastava et al. (2014), o *dropout* “remove aleatoriamente um subconjunto de unidades de entrada ou oculta a rede durante o treinamento. O *dropout* ajuda a evitar o *overfitting*, impedindo que as unidades da rede coadaptem muito”. Em outras palavras, o *dropout* força a rede a aprender características mais robustas e independentes dos dados de treinamento.

Figura 21 - Ilustração da camada totalmente conectada.



Fonte - IA Expert Academy (https://iaexpert.academy/topic/rede-neural-densa/, acesso em julho de 2023).

3. TRABALHOS ANTERIORES RELACIONADOS À DETECÇÃO DE COVID-19 POR MEIO DE RAIO-X DE TÓRAX USANDO REDES NEURAIS

O objetivo desta seção é apresentar trabalhos e/ou publicações científicas similares ao presente trabalho, com intuito de sustentar a importância do tópico selecionado e bem como discutir os métodos utilizados por outros autores a fim de comparação com o método aqui elaborado e resultados obtidos.

3.1. Detecção de COVID-19 em Imagens de Raio-X de Tórax através de Seleção Automática de Pré-processamento e de Rede Neural Convolucional

No artigo publicado por João et al. (2023), foi proposta uma técnica de seleção automática de melhoramento de imagens e da rede neural convolucional mais adequada para o problema. O objetivo é classificar imagens de CXR (*chest X-rays*) em normal ou COVID-19.

O método proposto pelo artigo é realizado em 4 passos. O primeiro é a obtenção de um *dataset* confiável, para tal foi utilizada uma base oficial do *Kaggle*: *COVID-19 Radiography Database*. O segundo passo consiste na seleção do melhor filtro de pré-processamento automático. Adiante, o terceiro passo visa selecionar o melhor modelo de CNN para classificar as imagens pré-processadas. Por fim, as etapas anteriores são concatenadas, o *dataset* é pré-processado pelo melhor filtro e utilizada para treinar o melhor modelo, extraindo assim as métricas de validação.

Os filtros utilizados foram os seguintes:

- Filtro da média: filtro linear que calcula a média de um conjunto de pixels vizinhos para cada pixel na imagem.
- Filtro da mediana: filtro não linear que substitui cada pixel na imagem pelo vetor da mediana de um conjunto de pixels vizinhos.
- Filtro gaussiano: filtro linear que suaviza as imagens, mas preserva mais detalhes do que o filtro da média.
- Equalização do Histograma: é uma técnica de processamento de imagens que visa melhorar o contraste da imagem, tornando-a mais nítida e fácil de visualizar.
- CLAHE (*Contrast Limited Adaptive Histogram Equalization*): extensão da técnica de equalização do histograma que leva em conta as diferenças locais na distribuição de intensidade de pixel.

- Filtro *Bottom Hat* (*Bottom Hat Filter*): é realizada aplicando-se uma operação de fechamento seguida de uma operação de abertura sobre a imagem original.

“Para o pré-processamento, foi escolhida a ResNet, cuja principal inovação é o uso de blocos residuais, que permitem a construção de redes muito mais profundas do que as tradicionais sem sofrer degradamento no desempenho”, afirmam os autores.

Para a escolha do modelo final, foram considerados os seguintes modelos:

- ResNet: é uma arquitetura projetada para melhorar o desempenho em conjuntos de dados muito grandes, utilizando blocos residuais para a prevenção do degradamento de redes muito profundas.
- DenseNet: arquitetura que utiliza ligações densas entre as camadas da rede neural. As camadas recebem as entradas de todas as camadas anteriores e envia suas saídas para todas as camadas subsequentes.
- VGG: é uma arquitetura mais simples e popular para a classificação de imagens. Utiliza convoluções em tamanho fixo de 3x3 e camadas de pooling para extrair as principais características das imagens.
- Inception: projetada para extrair características de diferentes escalas em uma imagem. Utiliza convoluções de diferentes tamanhos em paralelo e as combina em uma única camada.
- Xception: utiliza separações de canais e profundidades para extrair as características das imagens. A arquitetura usa convoluções separáveis em profundidade, que executam convoluções separadamente em cada canal da imagem e em seguida, combina os resultados.

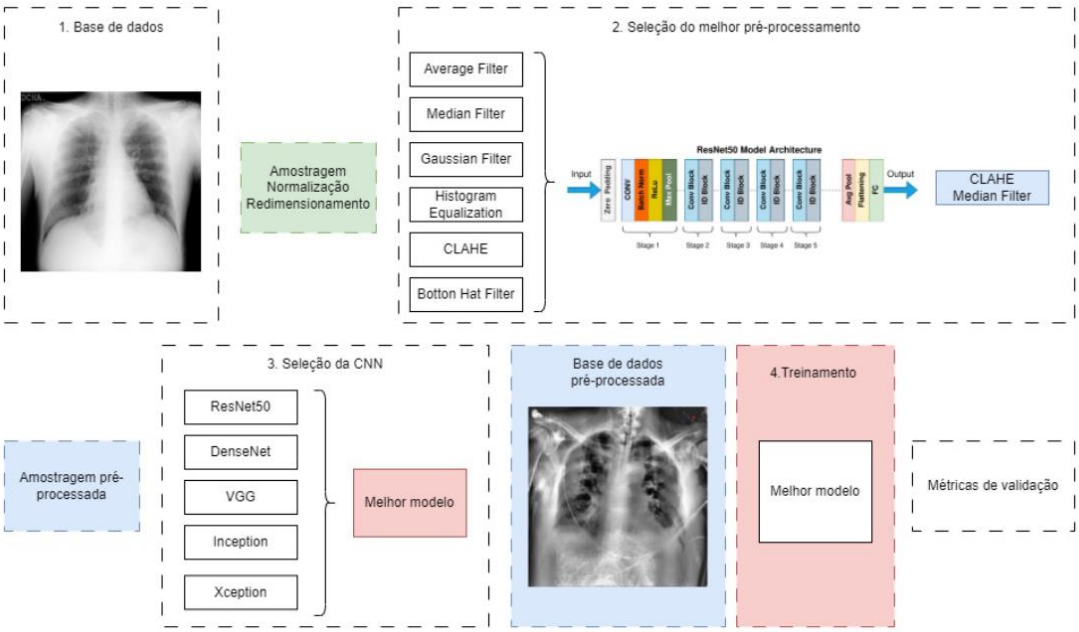
As métricas de avaliação escolhidas pelos autores foram: acurácia, precisão, sensibilidade e F1-score.

De acordo com os autores, foi observado que o método escolhido apresenta vantagens e limitações.

Dentre as vantagens vale destacar a melhoria na métrica de validação, devido a seleção de filtros que melhor realça as estruturas da imagem, ainda, ao iterar sobre variadas arquiteturas de CNN, foi possível selecionar aquela que melhor se adequou ao trabalho proposto.

As principais desvantagens pontuadas pelos autores foram: a utilização de um kernel, hiperparâmetros e arquiteturas padrões, além da ausência de um teste com outra base de dados para validar as etapas da seleção automática.

Figura 22 - Imagem representativa do método adotado pelos autores.



Fonte - Autores (https://sol.sbc.org.br/index.php/sbcas/article/view/25286/25107, acesso em novembro de 2023).

3.2. Identificação de Imagens de Raio-X com Redes Neurais Convolucionais para Detecção de Pneumonia Causada por COVID-19

No trabalho proposto por David et al. (2023), dez modelos diferentes de redes neurais convolucionais (RNC) foram utilizados para a obtenção dos melhores resultados entre três métricas utilizadas: acurácia, sensibilidade e especificidade.

As principais diferenças dos modelos utilizados são o número de camadas convolucionais e camadas densas, o número de neurônios nas camadas densas, o número de filtros nas camadas convolucionais, a função de *backpropagation* e a dimensão dos kernels em cada camada.

Os autores destacam duas funções de erro utilizadas. A primeira é a *Focal Loss* (FL), cujo objetivo é atuar em cima de bases de dados desbalanceadas e/ou classes com diferentes dificuldades de classificação, adicionando parâmetros para a função *Cross Entropy* (CE) tradicional. Como a função CE não atribui pesos diferentes para diferentes classes, em bases de dados desbalanceadas torna-se mais relevante utilizar a função FL.

A fim de classificar o desempenho perante as métricas de avaliação selecionadas, foram construídos dez modelos distintos, variando no número de camadas de convolução, número de camadas densas e função de erro utilizada.

Figura 23 - Modelos propostos pelos autores e suas respectivas arquiteturas.

Modelo	Nº de Camadas	Nº de Cama. Conv.	Filtros por Cama. Conv.*	Dimen. dos *Kernels*	Nº de Cama. Densas	Neurônios por Camada Densa*	Função de Perda
1	11	3	64, 64, 32	(3,3)	2	256, 1	*Binary Crossentropy*
2	11	4	64, 64, 32, 32	(3,3)	2	256, 1	*Binary Crossentropy*
3	12	5	64, 64, 32, 32, 16	(2,2)	2	256, 1	*Binary Crossentropy*
4	13	5	64, 64, 32, 32, 16	(2,2)	3	128, 128, 1	*Binary Crossentropy*
5	13	5	128, 64, 32, 32, 16	(2,2)	3	256, 128, 1	*Binary Crossentropy*
6	11	3	64, 64, 32	(3,3)	2	256, 1	*Focal Loss*
7	11	4	64, 64, 32, 32	(3,3)	2	256, 1	*Focal Loss*
8	12	5	64, 64, 32, 32, 16	(2,2)	2	256, 1	*Focal Loss*
9	13	5	64, 64, 32, 32, 16	(2,2)	3	128, 128, 1	*Focal Loss*
10	13	5	128, 64, 32, 32, 16	(2,2)	3	256, 128, 1	*Focal Loss*

Fonte - Autores (https://regrasp.spo.ifsp.edu.br/index.php/regrasp/article/view/1152/870, acesso em novembro de 2023).

Dentre os modelos, destacaram-se os modelos 1, 2, 6 e 8, devido aos resultados de acurácia, sensibilidade e especificidade obtidos, que foram maiores do que os demais. Os modelos 6 e 8, por utilizarem a função de erro FL, possibilitam ainda a variação do parâmetro α. Os autores variaram esse parâmetro entre 0.25 e 10, escolhendo o valor final de acordo com os melhores resultados das métricas pré-estabelecidas.

Por fim, verificou-se que os modelos 1 e 6 foram os mais promissores, obtendo os melhores valores para a curva ROC, os resultados seguem abaixo:

Figura 24 - Resultado dos modelos.

Modelo	Valor da AUC (curva ROC)
1	**0,94759**
2	0,93927
6	**0,94306**
8	0,93795

Fonte - Autores (https://regrasp.spo.ifsp.edu.br/index.php/regrasp/article/view/1152/870, acesso em novembro de 2023).

3.3. Uma Arquitetura de Rede Neural Convolucional Simplificada para Reconhecimento da COVID-19 em Imagens de Raios-X

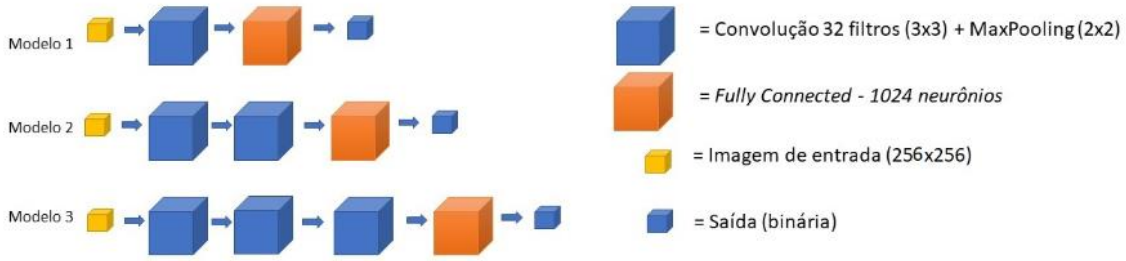
Neste artigo publicado por Fabio J et al (2021), são propostas três arquiteturas distintas de redes neurais convolucionais, cujo objetivo é realizar a classificação binária entre pacientes saudáveis e pacientes COVID positivos.

O *dataset* utilizado consiste em 6.432 imagens de radiografias de tórax, sendo 1.583 imagens de pacientes saudáveis, 576 imagens de pacientes COVID positivos e o restante de imagens são de pacientes com pneumonia viral. As imagens de pneumonia foram descartadas. Os autores comentam ainda que a única técnica de pré-processamento aplicada foi converter as imagens para a escala de cinza.

A primeira arquitetura consiste numa rede com quatro camadas, sendo elas a camada de entrada, uma única camada de convolução, uma camada densa e a camada de saída. A camada de entrada recebe a imagem que contém 256x256 pixels, a camada de convolução é responsável por extrair 32 filtros de tamanho 3x3 e é ativada através da função de ativação ReLU (*Rectifier Linear Unit*), logo após a operação de convolução é realizada a operação de *MaxPooling*, com parâmetros 2x2, adiante é aplicada a operação de *Flatten* (achatamento), para alterar a entrada da camada densa para o formato de um vetor unidimensional. A camada densa apresenta 1.024 neurônios, ativados através da função ReLU. Por fim, a camada de saída é composta por um único neurônio, que realiza a classificação binária, utilizando a função de ativação *Sigmoid*.

As demais arquiteturas propostas seguem os mesmos parâmetros, a diferença principal é que, para a segunda arquitetura, a rede apresenta dois blocos sequenciais de convolução e *MaxPooling*, já a terceira, apresenta três blocos sequenciais de convolução e *MaxPooling*.

Figura 25 - Modelos construídos a partir das arquiteturas propostas.



Fonte - Autores (https://sol.sbc.org.br/index.php/ercas/article/view/17428/17264, acesso em novembro de 2023).

A fim de validar os modelos construídos, foi utilizado o algoritmo K-Fold Cross-Validation com k = 5. O algoritmo distribui a base de dados em k partes e realiza a mesma quantidade de validações em cada uma delas, porém, em cada iteração ele varia a parte usada para treinamento e a outra para validação. Assim, são calculadas as métricas de avaliação (acurácia, sensibilidade, especificidade, área abaixo da curva ROC e a taxa de erro) e é possível comparar os resultados a fim de determinar quais modelos se saíram melhor.

Os resultados obtidos pelos autores seguem abaixo.

Figura 26 - Resultados dos testes dos modelos propostos.

Modelo	Acurácia(%)	Sensibilidade(%)	Especificidade(%)	AUC(%)	Loss(%)
Modelo 1	97,69	97,94	97,44	99,63	6,93
Modelo 2	98,06	98,44	97,67	99,83	4,84
Modelo 3	99,36	99,33	99,39	99,98	1,59

Fonte - Autores (https://sol.sbc.org.br/index.php/ercas/article/view/17428/17264, acesso em novembro de 2023).

4. **MATERIAS E MÉTODOS**

O presente capítulo relata os materiais, softwares, sites e ferramentas utilizadas, para o desenvolvimento do trabalho. Posteriormente, o método abordado é discutido amplamente, bem como é feito um resumo e apresentação das arquiteturas utilizadas para a construção de cada modelo individualmente.

A motivação para a adição, ou remoção, de camadas dos modelos é discutida brevemente nesta seção, todavia é possível encontrar a motivação mais elaborada no capítulo 5 durante a discussão dos resultados obtidos.

4.1. MATERIAIS

Para a realização do presente trabalho, os principais materiais utilizados foram: o ambiente de desenvolvimento *Google Colab* e o *dataset* utilizado para treinar e validar os modelos utilizados.

Figura 27 - Símbolo do Google Colab.

Fonte - TensorFlow Blog (https://blog.tensorflow.org/2018/05/colab-easy-way-to-learn-and-use-tensorflow.html, acesso em novembro de 2023).

O *Colab* é um serviço gratuito, disponibilizado pelo *Google*. Atua como um *Jupyter Notebook* hospedado na nuvem, possibilitando aos usuários utilizarem os Notebooks sem nenhuma necessidade de configuração. Ainda, não há custo computacional para o usuário, já que são os servidores do Google os responsáveis por processar todo o código desenvolvido ali.

O *Colab* foi escolhido pela facilidade de uso, já que basta ter acesso à internet para poder utilizá-lo e, também, por utilizar servidores externos com maior capacidade de processamento e memória RAM, agilizando bastante o treinamento dos modelos.

Figura 28 - Símbolo do Kaggle.

Fonte - The .ISO zone (https://theisozone.com/what-is-kaggle/, acesso em novembro de 2023).

Kaggle é a maior comunidade de ciência de dados do mundo, contendo uma enorme quantidade de *datasets* dos mais variáveis tópicos possíveis, além de ter acesso aos dados, por lá também é possível participar de diversas competições mensais para análise e ciência de dados.

O *dataset* utilizado nesse trabalho, para treinamento, é o *COVID-19 Radiography Database* disponibilizado pelos usuários Tawsifur Rahman, Dr. Muhammad Chowdhury e Amith Khandakar. Esse *dataset* foi escolhido pela comunidade do Kaggle como o ganhador do prémio COVID-19 Dataset Award. As imagens foram adquiridas através da parceria entre as universidades de Catar, Doha, Dhaka e Bangladesh, contando ainda com colaboradores do Paquistão e Malásia. Logo, por possuir o selo Kaggle Dataset Award, esse dataset foi escolhido por conta da confiabilidade presente nos dados.

O *dataset* contém cerca de 3.616 imagens de COVID-19 positivo e 10.192 imagens e COVID-19 negativo. No entanto, antes do treinamento dos modelos ser iniciado, foi necessário fazer alguns ajustes já que o *dataset* também continha imagens de pneumonia viral, além de máscaras e outros metadados, como planilhas de Excel. Todo esse excedente de informação foi excluído, restando apenas as imagens de raio-X de tórax para a tarefa de classificação binária dos modelos.

Por sua vez, o *dataset* utilizado para validação foi o *Curated COVID-19 Chest X-Ray Dataset*. Esse *dataset* foi escolhido por apresentar um conjunto de imagens já pré-processado, eliminando a necessidade de realizar o pré-processamento das imagens antes de realizar de fato o teste do modelo.

Por fim, vale destacar o uso da linguagem de programação python, na versão 3.10, bem como o uso de diversas bibliotecas dessa linguagem, como:

- Matplotlib: biblioteca utilizada para plotar os gráficos de métricas dos resultados de treinamento e teste;
- Numpy: bliblioteca popular na comunidade de Python, tem como objetivo realizar o processamento de grandes arrays e matrizes multidimensions;
- PIL: biblioteca para visualização de imagens;
- Scikit-learn: biblioteca que apresenta diversas funções de estatística, utilizada para calcular as métricas de teste dos modelos;
- TensorFlow: framework para aprendizado de máquina, amplamente utilizado neste trabalho para elaborar, treinar e validar todos os modelos desenvolvidos;

4.2. MÉTODO

O presente trabalho propõe o uso de Redes Neurais Convolucionais para a detecção de COVID-19 em imagens de raio-X de tórax. A escolha das redes neurais ocorreu por alguns motivos, dentre eles: por tratar-se de análise de imagens, as Redes Neurais Convolucionais apresentam alta performance, quando comparada a outras técnicas, além de não necessitar um grande esforço para o pré-processamento dos dados e contar com alta capacidade de aprendizagem na regulação de seus filtros/camadas (*Feature Maps*).

Sendo assim, alguns modelos foram propostos a fim de encontrar o melhor modelo possível, classificado através de algumas métricas de avaliação, como acurácia, sensibilidade e a especificidade. A sensibilidade é a capacidade do modelo em classificar corretamente os pacientes que estão positivos para a COVID-19, já especificidade, por sua vez, é a habilidade do modelo em classificar corretamente os pacientes que não possuem a doença.

Logo, a sensibilidade é a capacidade do modelo em detectar corretamente os verdadeiros positivos e, a especificidade, a capacidade do modelo em classificar corretamente os verdadeiros negativos.

Ao todo foram elaborados 6 modelos, diferindo entre si, principalmente, na quantidade de camadas convolucionais e número total de camadas.

Tabela 1 - Modelos elaborados para validar o uso de uma CNN na detecção de COVID-19.

Modelo	Nº total de camadas	Nº de camadas Convolucionais
Modelo 1	8	1
Modelo 2	10	2
Modelo 3	11	3
Modelo 4	14	4
Modelo 5	16	5
Modelo 6	18	6

Fonte - Autor.

A arquitetura base dos modelos consiste em camadas de convolução, *MaxPooling* e camadas densas. Com a intenção de evitar ao máximo o overfitting, são aplicadas camadas de Dropout entre as camadas densas e uma camada de *BatchNormalization* para introduzir normalização na rede. A ativação das camadas de entrada é a ReLU, já a de saída é a Sigmoid, que permite realizar classificação binária interpretando os valores de 0 e 1 gerados pela rede.

Ademais, cada modelo foi ajustado individualmente a fim de se obter o máximo daquela arquitetura, o que justifica alguns modelos diferirem entre si entre valores de parâmetros.

Figura 29 - Arquitetura Modelo 1.

```
base_model_1 = Sequential([
    rescale,
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.BatchNormalization(),
    layers.Flatten(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

O modelo 1 é o modelo mais simples. Conta com apenas um conjunto de camada de convolução, camada de *Max Pooling* e *Dropout* antes da saída ser redimensionada num vetor unidimensional na camada de *Flatten* e a classificação ser realizada após a saída da camada Densa.

Figura 30 - Arquitetura Modelo 2.

```
base_model_2 = Sequential([
    rescale,
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.BatchNormalization(),
    layers.Flatten(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.6),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

O modelo 2 apresenta dois conjuntos de convolução, Max Pooling e Dropout, que é utilizada a fim de normalizar a saída de camada, o que é benéfico no combate ao Overfitting, devido à baixa quantidade de imagens necessárias para treinar o modelo a fim de reconhecer todas as nuances necessárias das imagens de raios-X.

Figura 31 - Arquitetura do Modelo 3.

```
base_model_3 = Sequential([
    rescale,
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.6),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

O modelo 3, com três camadas de convolução, é onde já se começa a observar os resultados melhores e quase que sem presença de overfitting, como será mostrado nos resultados.

Figura 32 - Arquitetura do Modelo 4.

```
base_model_4 = Sequential([
    rescale,
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.BatchNormalization(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.6),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

Os modelos 4, 5 e 6, respectivamente contam com quatro, cinco e seis conjuntos de camadas de convolução, operação de Max Pooling, e Dropout. Os modelos 5 e 6 contam com 3 camadas densas.

Figura 33 - Arquitetura Modelo 5.

```
base_model_5 = Sequential([
    rescale,
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(16, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.BatchNormalization(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.6),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

Figura 34 - Arquitetura Modelo 6.

```
base_model_6 = Sequential([
    rescale,
    layers.Conv2D(32, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(256, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(512, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(1024, (3, 3), activation="relu"),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.BatchNormalization(),
    layers.Dense(64, activation="relu", kernel_regularizer='l2'),
    layers.Dropout(0.6),
    layers.Dense(1, activation="sigmoid")
])
```

Fonte - Autor.

Todos os modelos foram treinados contendo os seguintes parâmetros globais:

Figura 35 - Parâmetros globais.

```
[21] epochs = 75
     rescale = tf.keras.Sequential([
       layers.Resizing(img_height, img_width),
       layers.Rescaling(1./255)
     ])
```

Fonte – Autor.

O número de épocas foi escolhido conforme tentativa e erro, buscou-se uma quantidade que permitisse a todos os modelos desenvolverem completamente sua curva de aprendizado e se levou em consideração o tempo necessário para treinamento de cada modelo.

A camada *rescale* tem por objetivo redimensionar as imagens de raios-X para o tamanho de 256x256 pixels, e aplicar o *rescaling* para normalizar o resultado das entradas.

Figura 36 - Variáveis de dimensão e tamanho de batch.

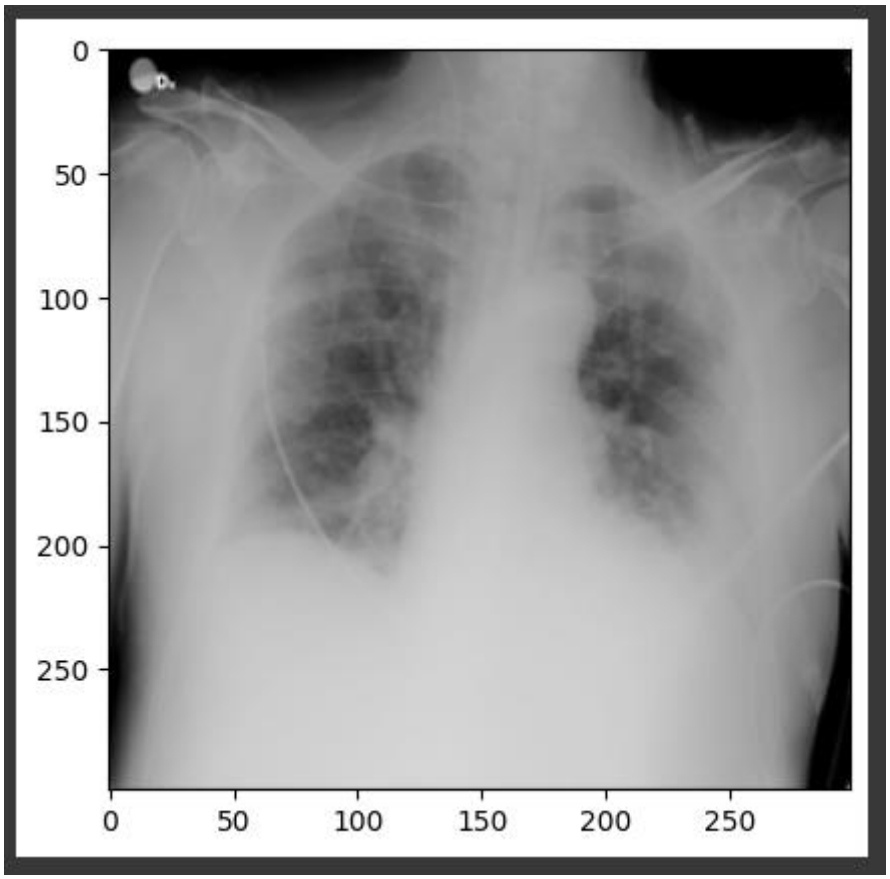
```
[12] batch_size = 64
     img_height = 256
     img_width = 256
     img_size = (img_height, img_width)
```

Fonte - Autor.

Na figura acima é possível ver a definição dos valores das variáveis que representam as dimensões da imagem e do tamanho do batch aplicado.

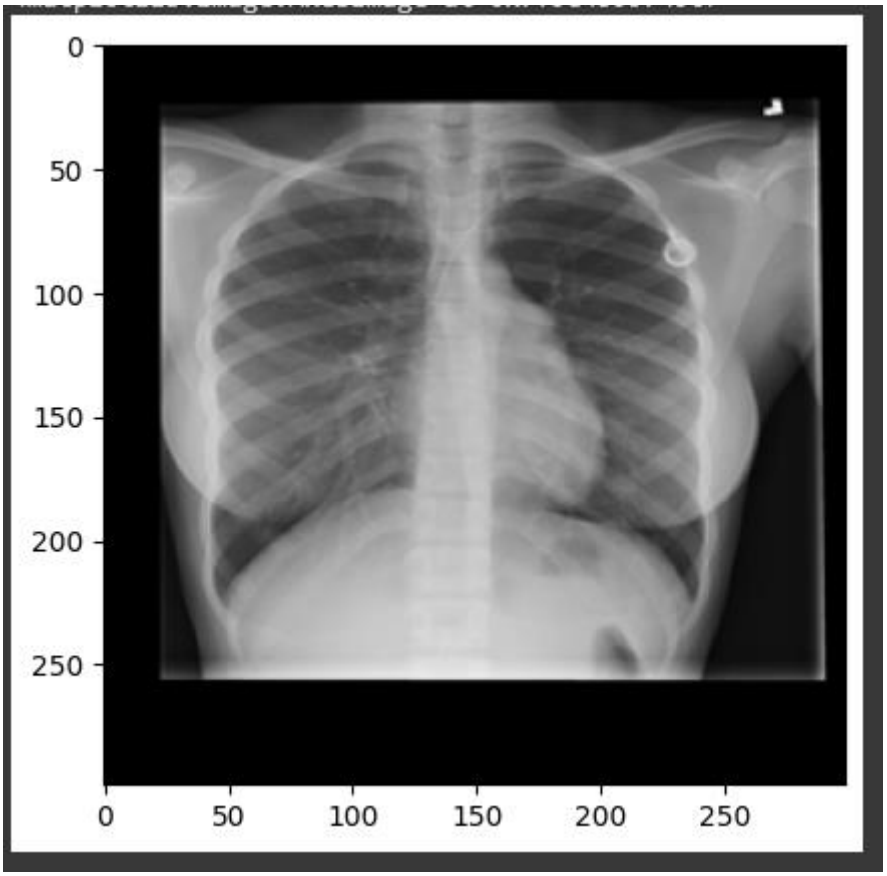
Abaixo, são exibidas algumas imagens do *dataset*, sendo uma para COVID negativo e outra para COVID positivo, representando as classes a serem identificadas pelo modelo.

Figura 37 - COVID Positivo.



Fonte - COVID-19 Radiography Database (https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database, acesso em dezembro de 2023).

Figura 38 - COVID Negativo.



Fonte - COVID-19 Radiography Database (https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database, acesso em dezembro de 2023).

Os *datasets* foram divididos em três categorias: *dataset* de treinamento, *dataset* de validação e *dataset* de teste. O *dataset* de treinamento tem por objetivo treinar o modelo para reconhecer os padrões presentes em ambas as classes, já o *dataset* de validação serve para validar os resultados obtidos através do treinamento. Por fim, o *dataset* de teste, que não é utilizado durante o treinamento do modelo, serve para avaliar e classificar o modelo quanto as métricas determinadas.

Figura 39 - Definição do objeto tf.data.Dataset para treinamento.

```
train_ds = tf.keras.utils.image_dataset_from_directory(
    train_data_dir,
    image_size = img_size,
    validation_split = validation_split,
    subset = "training",
    seed = seed_train_validation,
    color_mode = "grayscale",
    batch_size = batch_size,
    shuffle = shuffle_value
)
```

Fonte - Autor.

Figura 40 - Definição do objeto tf.data.Dataset para validação.

```
val_ds = tf.keras.utils.image_dataset_from_directory(
    train_data_dir,
    image_size = img_size,
    validation_split = validation_split,
    subset = "validation",
    seed = seed_train_validation,
    color_mode = "grayscale",
    batch_size = batch_size,
    shuffle = shuffle_value
)
```

Fonte - Autor.

Figura 41 - Definição do objeto tf.data.Dataset para teste.

```
test_ds = tf.keras.utils.image_dataset_from_directory(
    test_data_dir,
    image_size = img_size,
    seed = seed_train_validation,
    color_mode = "grayscale",
)
```

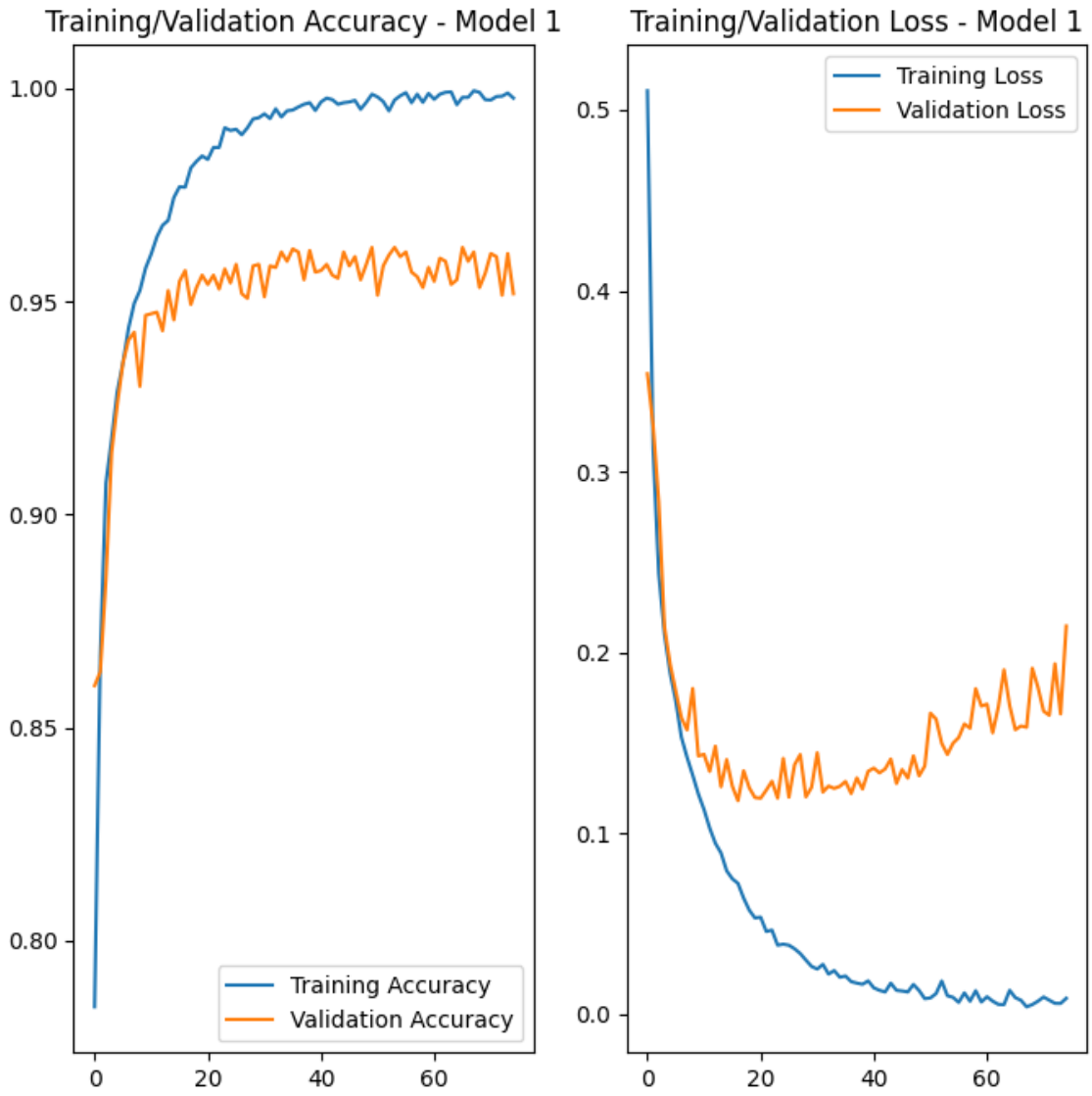
Fonte - Autor.

5. RESULTADOS E DISCUSSÃO

Para a obtenção dos resultados, os modelos são treinados num conjunto de treinamento e depois são validados através de um conjunto de validação. A divisão do *dataset* entre treinamento e validação foi: 80% das imagens destinadas para teste e 20% destinadas para validação. Ou seja, das 13.808 imagens presentes no *dataset*, 11.047 foram destinadas para o treinamento e 2.761 imagens para a validação.

Após o treinamento de cada modelo, alguns gráficos foram traçados, com intuito de avaliar o desempenho geral do modelo no treinamento.

Figura 42 - Resultados do treinamento do Modelo 1.



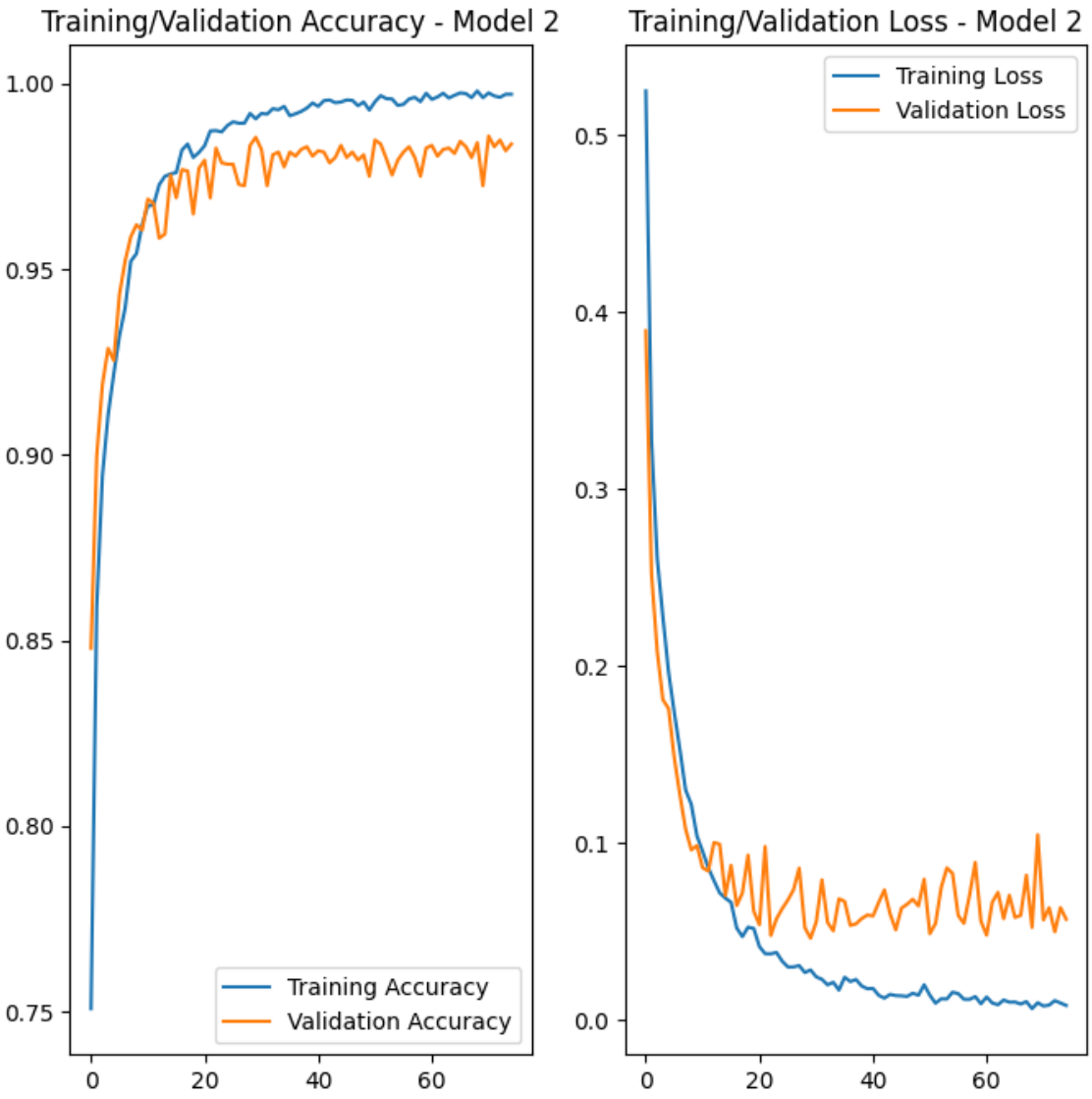
Fonte - Autor.

Após ser treinado por um pouco mais de 10 épocas, já é possível observar o *overfitting* no modelo 1.

O overfitting pode ser causado por vários fatores, dentre eles os principais são a alta complexidade do modelo (alto números de parâmetros treináveis) e a baixa presença de imagens, o que faz com que o modelo não consiga aprender os padrões necessários e passa a “decorar” a resposta para a classificação de cada imagem. É possível perceber que a acurácia de treino chega a patamares próximos de 100%, porém a acurácia de validação fica variando entre 95% e 96%. Por sua vez, o erro no treinamento chega a patamares próximos de 0%, porém o erro na validação atinge patamares de mais ou menos 15% quando começa a aumentar conforme as épocas vão passando.

Levando em consideração os resultados do Modelo 1, foi proposto para o Modelo 2 o aumento no número de camadas de convolução a fim de diminuir a quantidade de parâmetros treináveis, com intuito de diminuir a complexidade geral do modelo e diminuir os efeitos de overfitting.

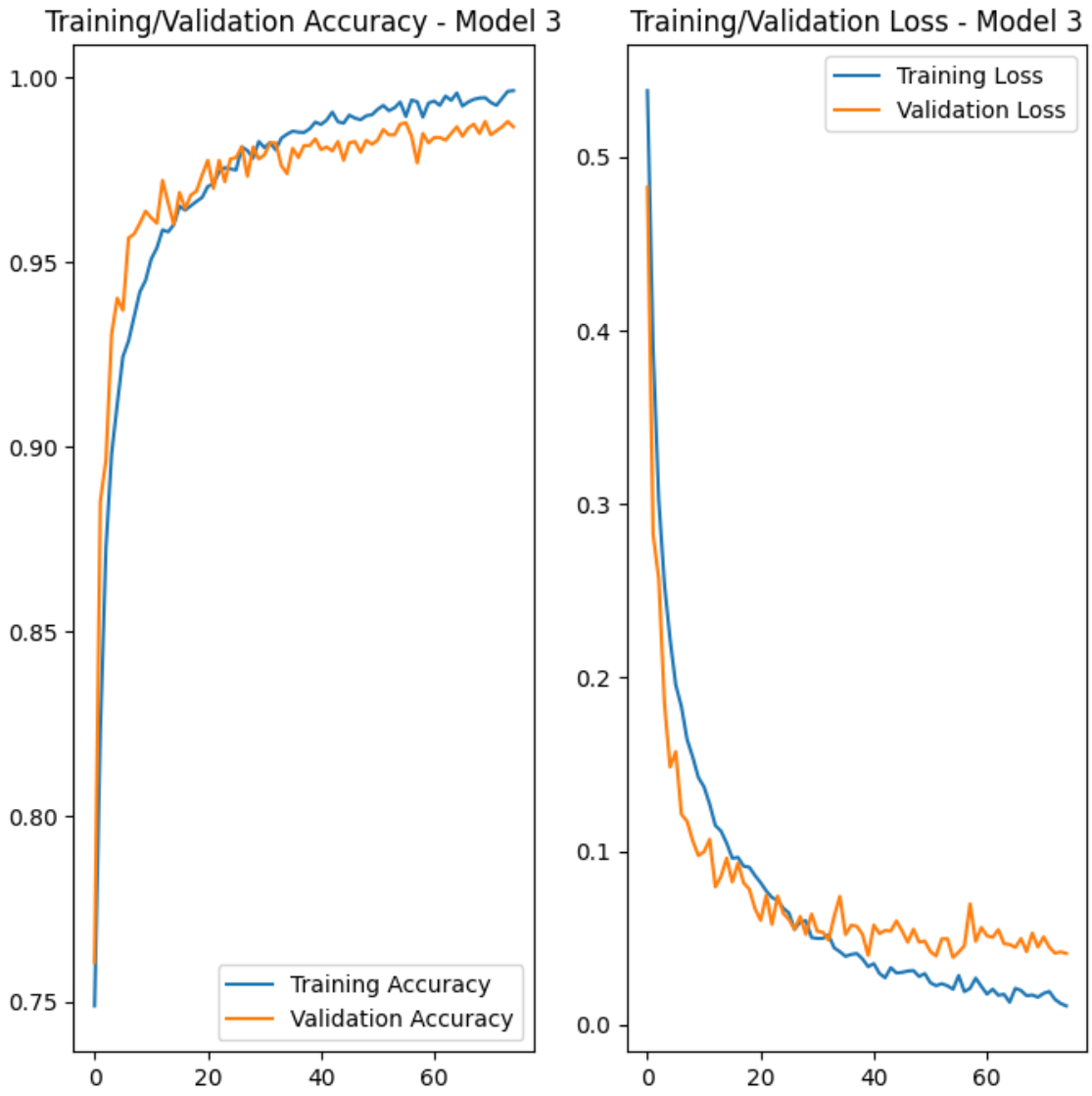
Figura 43 - Resultados do treinamento do Modelo 2.



Fonte - Autor.

Analisando os resultados do treinamento do Modelo 2 é possível notar uma ligeira melhoria em relação ao Modelo 1. Ao acrescentar uma nova camada de convolução, a complexidade do modelo diminui, o que ajuda a combater o overfitting. Os valores de acurácia e erro de treinamento estão mais próximos dos valores de validação, no entanto, ainda é possível notar o overfitting por volta da época 20.

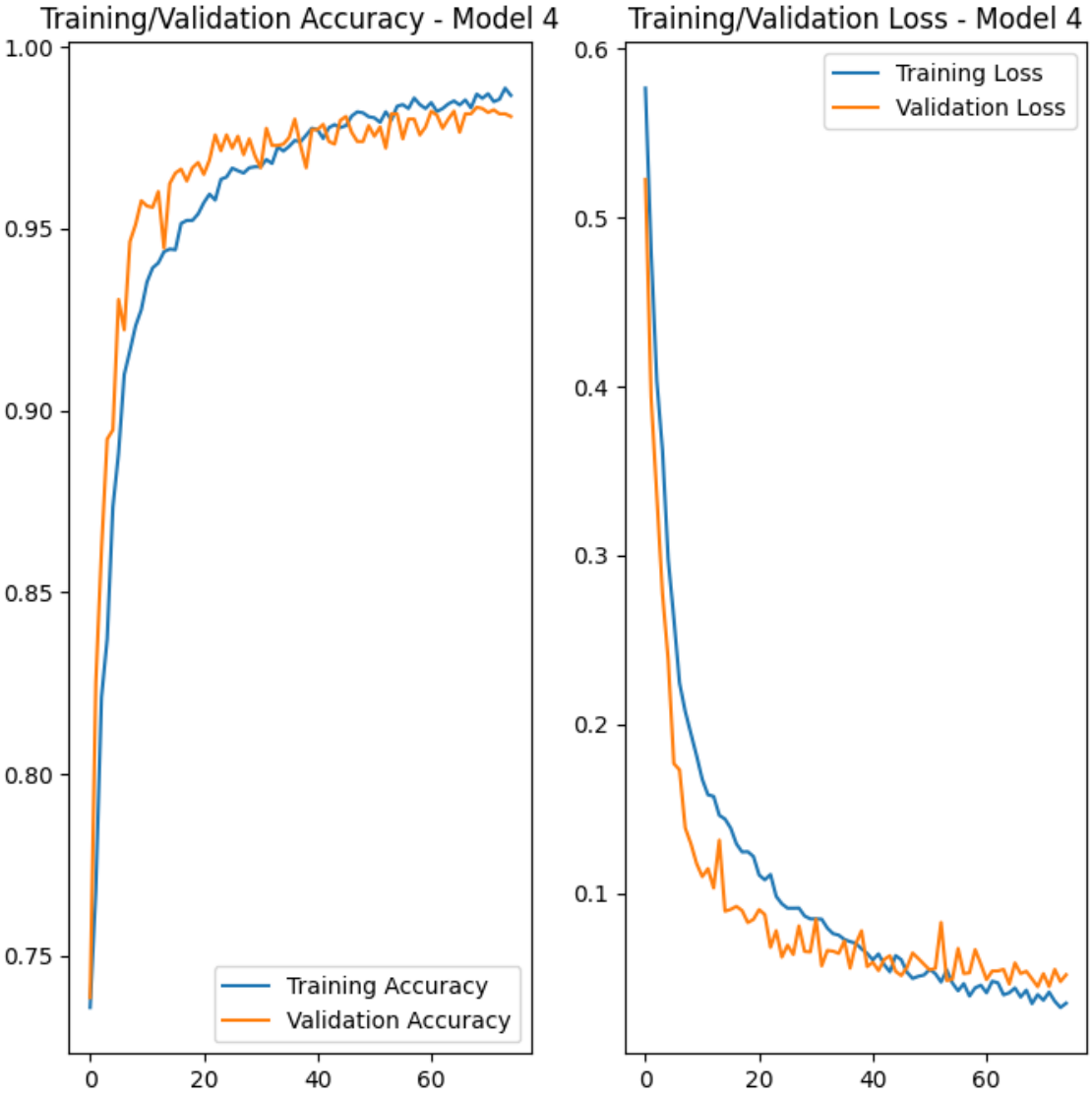
Figura 44 - Resultados do treinamento do Modelo 3.



Fonte - Autor.

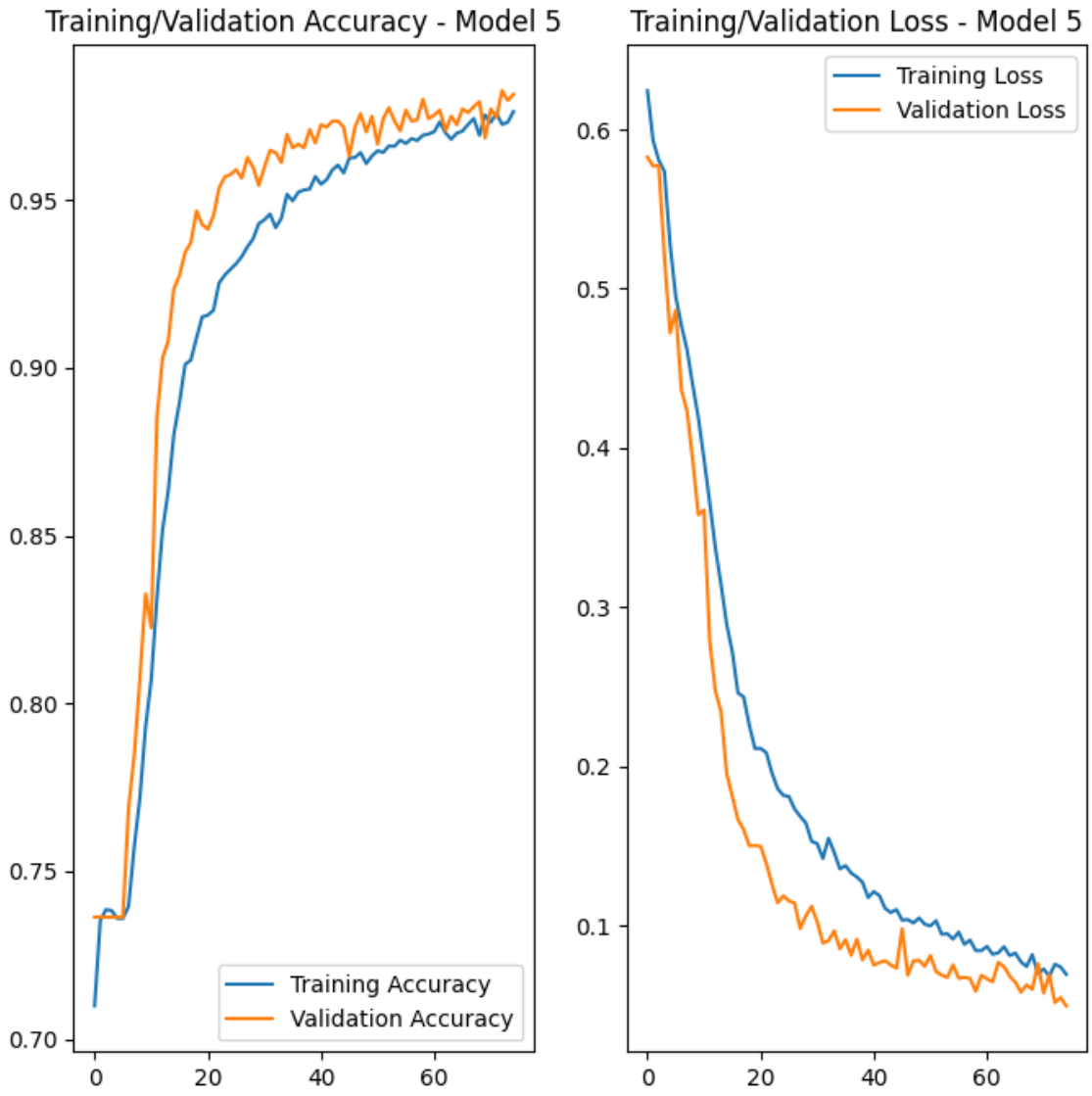
O Modelo 3 já mostra grandes melhorias em relação ao Modelo 2. O overfitting acontece entre as épocas 20 e 40, porém também é possível notar que os resultados de treinamento e validação são praticamente os mesmos quando atingem um patamar de erro inferior a 10% e acurácia superior a 97%.

Figura 45 - Resultado de treinamento do Modelo 4.



Fonte - Autor.

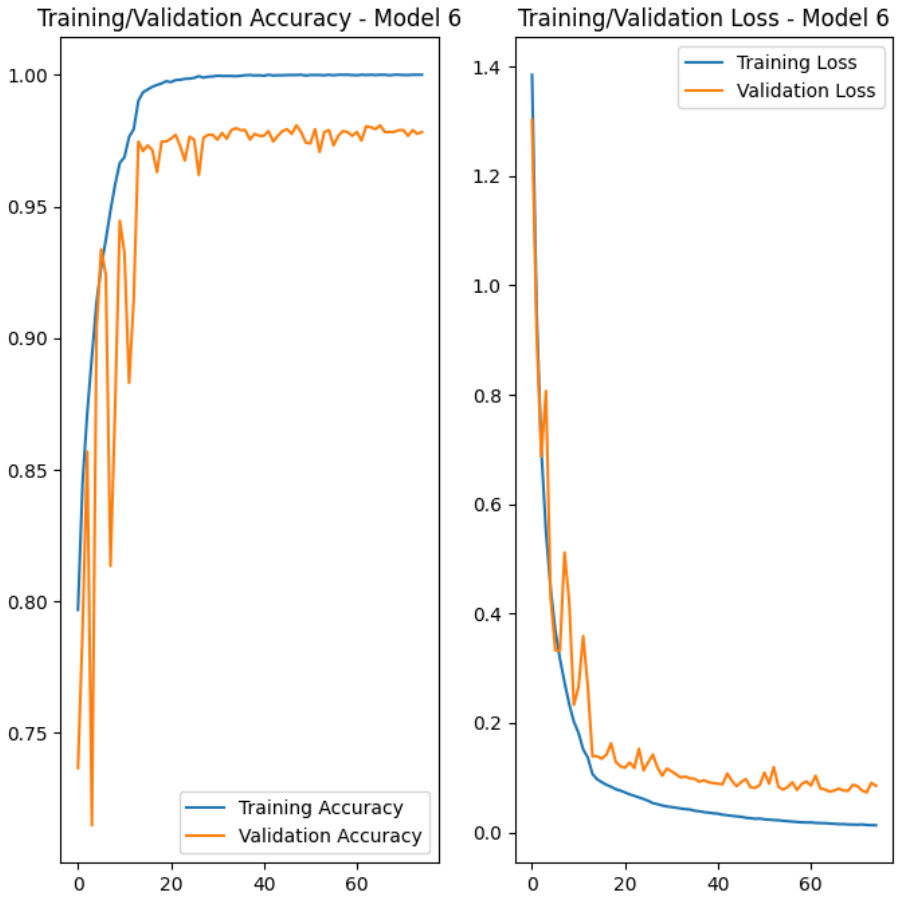
Figura 46 - Resultado de treinamento do Modelo 5.



Fonte - Autor.

Os modelos 4 e 5, principalmente o modelo 5, começam a apresentar um comportamento diferente do esperado. Em ambos é possível notar que as métricas de validação, aumentam, no caso da acurácia, ou diminuem, no caso do erro, mais rapidamente que os seus respectivos valores de treino. Esse fenômeno pode ser causado devido ao excesso de normalização nos modelos. A normalização é inserida através das camadas de Dropout, após as operações de convolução e após a saída de cada camada densa até a última. Essa técnica é utilizada também para combater o overfitting do modelo, que era visivelmente perceptível nos modelos anteriores, porém, ao adicionar camadas demais ou valores altos demais para o Dropout, pode-se esperar que aconteça tal fenômeno.

Figura 47 - Resultado de treinamento do Modelo 6



Fonte - Autor.

O modelo 6, por conter 6 camadas de convolução, torna-se o valor com o menor número de parâmetros treináveis dentre os modelos aqui testados. Sendo assim, conforme é possível ver na imagem acima, o número de parâmetros treináveis disponíveis no modelo 6 é insuficiente para realizar a classificação binária. As grandes divergências entre os valores de acurácia entre épocas devem-se ao fato do modelo começar a palpitar de forma aleatória a classe dos modelos e/ou decorar a alternativa correta no dataset de validação.

A tabela 2 demonstra os resultados obtidos em cada modelo.

Tabela 2 - Resultados Obtidos.

Modelo	Acurácia	Sensibilidade	Especificidade
Modelo 1	95.47%	98.50%	87.74%
Modelo 2	**96.61%**	**98.89%**	**90.78%**
Modelo 3	**96.76%**	**99.51%**	**89.77%**

Modelo 4	**96.83%**	**99.23%**	**90.71%**
Modelo 5	95.49%	98.92%	86.72%
Modelo 6	95.03%	98.96%	85.01%

Fonte - Autor.

De forma geral, os modelos 2, 3 e 4 obtiveram resultados muito similares. Os modelos 1, 5 e 6 obtiveram resultados abaixo dos demais, especialmente o modelo 6, com 6 camadas de convolução, que contou o resultado mais baixo de acurácia. No entanto, todos os modelos acabaram obtendo resultados satisfatórios, tendo em vista a simplicidade de suas arquiteturas.

6. **CONCLUSÃO**

A epidemia de COVID-19 causou grande impactos na sociedade. No auge da pandemia, era difícil precisar com exatidão a quantidade de infectados por conta de os testes serem de difícil acesso a população em geral, o que dificultava traçar estratégias para melhor combater o avanço da doença. Sendo assim, este trabalho propõe o desenvolvimento de uma Rede Neural Convolucional que permita classificar, com confiança, a contaminação por COVID-19 através de um método mais acessível, os raios-X de tórax comumente solicitados por médicos na rede pública de saúde.

Ao propor arquiteturas simples e obter resultados que atingem praticamente 97% de acurácia e 99% de sensibilidade, fica evidente a capacidade do uso desta tecnologia para detectar com precisão COVID-19 em raios-X de tórax de diversos pacientes. Porém, ressalta-se que tal tecnologia serve apenas como ferramenta auxiliar de profissionais de saúde capacitados cuja responsabilidade de diagnosticar cabe somente a eles.

Para melhorias futuras, pretende-se utilizar técnicas de Transferência de Aprendizagem e Ajuste Fino afim de obter resultados ainda melhores e poupar esforço computacional na busca por pesos ideais para aprendizagem da rede.

REFERÊNCIAS BIBLIOGRÁFICAS

Kheyroddin, R., Mohammadi, R., Nikbakht, S., & Asadian, M. (2021). Diagnosis of COVID-19 using chest X-ray images and a deep learning convolutional neural network. Medical & Biological Engineering & Computing, 59(1), 1-9.

Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. Pattern Analysis and Applications, 24(3), 1207-1220.

Shu, Y., Qiu, Y., Li, W., Zhan, Z., He, W., & Ren, B. (2020). A convolutional neural network-based framework for COVID-19 pneumonia detection from radiography images. Engineering Applications of Artificial Intelligence, 94, 103817.

Yin, R. K. (2015). Estudo de caso: planejamento e métodos. Bookman Editora.

GUAN, W.-J. et al. Clinical Characteristics of Coronavirus Disease 2019 in China. New England Journal of Medicine, v. 382, n. 18, p. 1708-1720, 30 abr. 2020. DOI: 10.1056/NEJMoa2002032.

CARFÌ, A. et al. Persistent Symptoms in Patients After Acute COVID-19. JAMA, v. 324, n. 6, p. 603-605, 2020.

CDC. Coronavirus Disease 2019 (COVID-19). Centers for Disease Control and Prevention, 2021. Disponível em: https://www.cdc.gov/coronavirus/2019-ncov/index.html. Acesso em: 1º de abril de 2023.

ELLUL, M. A. et al. Neurological associations of COVID-19. Lancet Neurology, v. 19, n. 9, p. 767-783, 2020.

GRASSSELI, G. et al. Baseline Characteristics and Outcomes of 1591 Patients Infected With SARS-CoV-2 Admitted to ICUs of the Lombardy Region, Italy. JAMA, v. 323, n. 16, p. 1574-1581, 2020.

Conselho Federal de Medicina. Resolução CFM nº 2.107/2014. Dispõe sobre o uso da radiologia no Brasil e dá outras providências. Brasília, DF: Conselho Federal de Medicina, 2018. Disponível em: http://www.portalmedico.org.br/resolucoes/CFM/2014/2107_2014.pdf. Acesso em: 01 abr. 2023.

Wong, H. Y. F. et al. Frequency and Distribution of Chest Radiographic Findings in COVID-19 Positive Patients. Radiology, v. 296, n. 2, p. E72-E78, 2020.

Rajpurkar, P. et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. PLOS Medicine, v. 15, n. 11, p. 1-18, 2018.

BROWNLEE, Jason. Why Python is the Best Programming Language for Machine Learning. Machine Learning Mastery, 2019. Disponível em: https://machinelearningmastery.com/why-python-is-the-best-platform-for-machine-learning/. Acesso em: 02 abr. 2023.

CHOLLET, François. Why Keras? Deep Learning for Humans. Disponível em: https://keras.io/why_keras/. Acesso em: 02 abr. 2023.

GÉRON, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., 2019.

OLIVEIRA, André. Big Data: A Revolução dos Dados. Novatec Editora, 2018.

RASCHKA, Sebastian; MIRJALILI, Vahid. Python Machine Learning. Packt Publishing, 2017.

Bisong, A. (2021). The ultimate guide to Google Colab for machine learning. Medium. Disponível em: https://towardsdatascience.com/the-ultimate-guide-to-google-colab-for-machine-learning-e5f7163907c3. Acesso em: 03 de abril de 2023.

Brownlee, J. (2016). Why Python is the language of choice for deep learning. Machine Learning Mastery. Disponível em: https://machinelearningmastery.com/why-python-is-the-language-of-choice-for-deep-learning/. Acesso em: 03 de abril de 2023.

Kluyver, T. et al. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. Positioning and Power in Academic Publishing: Players, Agents and Agendas: Proceedings of the 20th

DOMINGOS, Pedro. O algoritmo mestre. São Paulo: Novatec, 2018.

OZTURK, Tulin et al. Automated detection of COVID-19 cases using deep neural networks with X-ray images. Computers in Biology and Medicine, v. 121, p. 103792, 2020.

WANG, Lei et al. A COVID-19 diagnostic model based on CT images and machine learning. Neural Computing and Applications, v. 32, n. 10, p. 5849-5858, 2020.

APOSTOLOPOULOS, Ioannis D.; BESSIANA, Kristalia Mezei. Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine, v. 43, n. 2, p. 635-640, 2020.

GOODFELLOW, Ian et al. Deep learning. Nature, v. 521, n. 7553, p. 436-444, 2016.

LECUN, Yann et al. Deep learning. Nature, v. 521, n. 7553, p. 436-444, 2015.

LITJENS, Geert et al. A survey on deep learning in medical image analysis. Medical image analysis, v. 42, p. 60-88, 2017.

ZEILER, Matthew D. et al. Visual

ESTEVA, Andre et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature, v. 542, n. 7639, p. 115-118, 2017.

GULRAJANI, Ishaan et al. Understanding neural networks through deep visualization. arXiv preprint arXiv:1611.05431, 2016.

SZELISKI, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

Hemdan, E. E. D., Shouman, M. A., & Karar, M. E. (2020). COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images. arXiv preprint

GONZALEZ, Rafael C.; WOODS, Richard E. Processamento de imagens digitais. Editora Edgard Blucher, 2018.

Horry, M., Salem, M. Z. M., Ali, S., & Abdelfattah, M. (2021). A review of image preprocessing techniques for COVID-19 detection using chest X-ray images. IEEE Reviews in Biomedical Engineering, 14, 288-299.

Khan, A. I., Shah, J. L., Bhat, M. M., & Coroama, V. C. (2021). A review on techniques for detection of COVID-19 using chest X-ray images. IOP Conference Series: Materials Science and Engineering, 1063, 012039.

Sarwar, S., & Arshad, F. (2020). An overview of deep learning based approaches for classification and detection of COVID-19 from chest X-ray images. Computers in Biology and Medicine, 121, 103805.

Ko, H., Park, C., & Shin, S. (2021). COVID-19 Detection from Chest X-ray Images using Deep Learning and Convolutional Neural Networks. Journal of Healthcare Engineering, 2021.

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach. Prentice Hall Press.

Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books.

Alpaydin, E. (2010). Introduction to machine learning (2nd ed.). MIT Press.

Bishop, C. M. (2006). Pattern recognition and machine learning (1st ed.). Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (1st ed.). MIT Press.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

Mitchell, T. (1997). Machine learning (1st ed.). McGraw Hill.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (1st ed.). MIT Press.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Zhang, X., & Zhao, L. (2021). Convolutional Neural Networks. In Deep Learning: A Comprehensive Guide to Artificial Intelligence - With Examples in Python (pp. 173-204). Springer.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

KRIZHEVSKY, Alex et al. ImageNet Classification with Deep Convolutional Neural Networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012. p. 1097-1105.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (pp. 315-323).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.'

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408.

João, et al (2023). Detecção de COVID-19 em Imagens de Raio-X de Tórax através de Seleção Automática de Pré-processamento e de Rede Neural Convolucional.

SAIT, UNAIS; k v, Gokul Lal; Prajapati, Sunny; Bhaumik, Rahul; Kumar, Tarun; S, Sanjana; Bhalla, Kriti (2020), "Curated Dataset for COVID-19 Posterior-Anterior Chest Radiography Images (X-Rays).", Mendeley Data, V1, doi: 10.17632/9xkhgts2s6.1

M.E.H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I. Reaz, M. T. Islam, "Can AI help in screening Viral and COVID-19 pneumonia?" IEEE Access, Vol. 8, 2020, pp. 132665 - 132676.

Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and

Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray