
Universidade Estadual do Maranhão
Centro de Ciências Tecnológicas - CCT
Programa de Pós-Graduação em Engenharia da Computação e Sistemas

# Uma aplicação móvel para classificação de imagens utilizando Deep Learning: Um Estudo de Caso sobre Doenças Pulmonares

**Elilson Santos**

**São Luís-MA, 2021**

Elilson Santos

# Uma aplicação móvel para classificação de imagens utilizando Deep Learning: Um Estudo de Caso sobre Doenças Pulmonares

Dissertação apresentada ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para obtenção do título de Mestre em Engenharia de Computação e Sistemas.

Universidade Estadual do Maranhão

Centro de Ciências Tecnológicas - CCT

Programa de Pós-Graduação em Engenharia da Computação e Sistemas

Orientador: Prof. Dr. Omar Andrés Carmona Corte

Coorientador: Prof. Dr. Bruno Feres de Souza

São Luís-MA

2021

Santos Elilson
Uma aplicação móvel para classificação de imagens utilizando Deep Learning: Um Estudo de Caso sobre Doenças Pulmonares/ Elilson Santos. – São Luís-MA, 2021.
87

Dissertação (Mestrado Profissional) – Curso de engenharia da computação e Sistemas, Universidade Estadual do Maranhão,2021.

Orientador: Prof. Dr. Omar Andrés Carmona Corte

1. Deep Learning. 2. Classificação. 3. Aplicação Móvel. 4. Doença Pulmonar. 5. COVID-19. I. Uma aplicação móvel para classificação de imagens utilizando Deep Learning: Um Estudo de Caso sobre Doenças Pulmonares

CDU 004.932.2:616.24

Elilson Santos

# Uma aplicação móvel para classificação de imagens utilizando Deep Learning: Um Estudo de Caso sobre Doenças Pulmonares

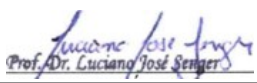
Dissertação apresentada ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para obtenção do título de Mestre em Engenharia de Computação e Sistemas.

Trabalho apresentado em 08 de Abril de 2021
**Banca Examinadora**


**Prof. Dr. Omar Andrés Carmona Cortes (IFMA)**
Orientador

**Prof. Dr. Bruno Feres de Souza (UFMA)**
Co-Orientador

**Prof. Dr. Antonio Fernando Lavareda Jacob Jr. (UEMA)**
Convidado




**Prof. Dr. Luciano José Senger (UEPG)**
Convidado

São Luís-MA
2021

*Aos meus pais,*
*por sempre estarem comigo em todos os momentos.*

# Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador Prof. Dr. Omar Andrés Carmona Corte pelos mais de 10 anos de aulas do ensino técnico ao mestrado, obrigado pelos ensinamentos, confiança e inspiração foram de extrema importância durante toda essa jornada. Obrigado pela contribuição para o meu crescimento pessoal e profissional.

Ao Prof. Dr. André Santos e Prof. Dra. Karla Fook pelo incentivo em cursar programa de Pós graduação.

A Chirlene Botelho, por nunca desistir e sempre apoiar minhas decisões. Obrigado pelo companheirismo, dedicação, carinho, respeito e por todas as conversas de incentivo.

Aos meus amigos da Coordenação de Otimização e Sequenciamento da Vale, por toda parceria, pelo ambiente harmonioso, por terem me recebido com todo carinho e principalmente pelas discussões que sempre agregaram e contribuíram para o meu crescimento pessoal e profissional.

A minha família (Sebastiana Santos, Enedon Izidoro e Marilda Oliveira), pelos cuidados, amor, carinho e dedicação, torcendo sempre para que eu chegasse nessa etapa da minha vida.

A todos os professores e colegas do Programa de Pós-Graduação, aos amigos do CEFET/IFMA. Aos mestres e doutores do IFMA pelos ensinamentos e experiências compartilhadas.

A minha família, aos amigos, meus sinceros agradecimentos...

*"Ele deu sua vida por nós,ele caiu perante a cruz*
*Para morrer por todos aqueles que nunca lamentaram sua perda*
*Não foi destinado a nós, sentir a dor novamente*
*Me diga o por que,me diga o porque"*
*(For The Greater Good of God - Iron Maiden)*

# Resumo

Assim como a pneumonia viral, a COVID-19 é mais uma doença infecciosa e pode causar uma síndrome respiratória aguda e o portador deve ser diagnosticado precocemente para se tomar medidas que contenham propagação do vírus. A doença espalhou-se rapidamente pelo mundo e pode levar à morte em apenas alguns dias. Assim, a investigação de formas rápidas de detecção que ajudem os profissionais de saúde no processo de tomada de decisões é essencial para ajudar na tarefa de salvar vidas. Nesse contexto, propõe-se uma aplicação móvel flexível que possa ajudar na classificação de imagens usando deep learning. O desenvolvimento se deu com flutter e dart gerando executáveis para IOS e Android bem como construção de API backend em python e os módulos baseado nos seus casos de uso e embarcando modelo mais eficiente para realização das predições com suas respectivas probabilidades de categorização. Como estudo de caso utilizou-se a COVID embarcando o rede neural convolucional chamada de VGG16 que apresentou os melhores resultados atingindo uma acurácia média de 96.5%, uma precisão de 96.8%, um recall de 97.1% e um F1-Score de aproximadamente 96.9%. Para os testes utilizou-se uma base de dados constituída por 7,178 radiografias divididas em três classes: Normal, COVID-19 e Pneumonia Viral.

**Palavras-chaves**: Deep Learning; Classificação; Aplicação Móvel; Doença Pulmonar; COVID-19.

# Abstract

Like viral pneumonia, COVID-19 is another infectious disease. It can cause the severe respiratory syndrome, and the carrier must be diagnosed early to take measures to contain the spread of the virus. The disease has spread quickly worldwide and can lead to death in just a few days. Thus, research into fast ways of detection that help health professionals in the decision-making process is essential to help save lives. A flexible mobile application that can help image classification using deep learning is proposed in this context. The development was done using flutter and dart generating executables for IOS and Android, building the backend API in python and the modules based on their use cases, and embedding the most efficient model for making predictions with their respective categorization probabilities. As a case study, COVID was used with the convolutional neural network called VGG16, which presented the best results, reaching an average accuracy of 96.5%, a precision of 96.8%, a recall of 97.1%, and an F1-Score of approximately 96.9%. A database consisting of 7,178 radiographs divided into three classes: Normal, COVID-19, and Viral Pneumonia was used for testing.

**Keywords**: Deep Learning; Classification; Mobile Application; Lung Disease; COVID-19.

# Lista de Figuras

# Lista de Tabelas

# Lista de Códigos

# Lista de abreviaturas e siglas

| | |
|---|---|
| AI4COVID-19 | Aplicativo de diagnóstico COVID-19 |
| AM | Aprendizado de máquina |
| AMA | American Medical Association |
| ANN | Artificial Neural Network |
| API | Application Programming Interface |
| AOT | Ahead-Of-Time |
| APP | Aplicativo Móvel |
| AUC | Area Under The Curve |
| C++ | Linguagem de programação |
| CDC | Centers for Disease Control and Prevention |
| CNNs | Redes Neurais Convolucionais |
| COVID-19 | Coronavírus |
| CRISP-DM | Cross Industry Standard Process for Data Mining |
| CRM | Conselho Regional de Medicina |
| DPOC | Doença Pulmonar Obstrutiva Crônica |
| DENSENE | Dense Network |
| EUA | Estados Unidos da América |
| FN | False Negative |
| FP | False Positive |
| GB | Gigabyte |
| GHz | GigaHertz |
| GPU | Unidade de Processamento Gráfico |
| HD | Hard Disk |

| | |
|---|---|
| IA | Inteligência Artificial |
| IRMA | Image Retrieval in Medical Applications |
| IOS | Sistema operacional móvel da Apple |
| JAVA | Ambiente computacional/plataforma |
| JIT | Just-In-time |
| JPEG | Joint Photographic Experts Group |
| OMS | Organização Mundial da Saúde |
| MATLAB | MATrix LABoratory - Linguagem de programação |
| MLP | Perceptrons de Múltiplas Camadas |
| MVP | Produto Viável Mínimo |
| NF | Não Funcional |
| PHP | Hypertext Preprocessor linguagem de script |
| PNG | Portable Network Graphics |
| RAM | Memória de Acesso Randômico |
| RELU | Rectified Linear Unit |
| RESNET | Residual Network |
| RF | Requisito Funcional |
| RGB | Red, Green, Blue |
| RNA | Rede Neural Artificial |
| ROC | Receiver Operator Characteristic |
| RTX | Radiografia de Tórax |
| SARS-CoV-2 | Coronavírus da síndrome respiratória aguda grave 2 |
| SDRA | Síndrome do Desconforto Respiratório Agudo |
| SDKs | Software Development Kits |
| TEP | Tomografia por Emissão de Prótons |
| TNS | Taylor Nelson Sofres |

| | |
|---|---|
| TN | True Negative |
| TP | True Positive |
| UI | User Interface |
| UTI | Unidades de Terapia Intensiva |
| VGG16 | Visual Geometry Group 16 Layers |
| VGG19 | Visual Geometry Group 19 Layers |
| VM | Virtual Machine |
| WEB | Rede Mundial de Computadores |
| ZIP | Zone Information Postal |

# Lista de símbolos

| | |
|---|---|
| $\in$ | Pertence |
| $\%$ | Percentual |
| $D_0$ | Profundidade inicial da imagem |
| $D$ | Profundidade final da imagem |
| $W_0$ | Largura inicial da imagem |
| $W$ | Largura final da imagem |
| $H_0$ | Altura inicial da imagem |
| $H$ | Altura final da imagem |
| $F1$ | Dimensão de filtros |
| $F2$ | Dimensão de filtros |
| $\forall$ | Para todo |
| $\mathbb{N}$ | Conjunto dos números naturais |
| $\mathbb{R}$ | Conjunto dos números reais |
| $\sum$ | Somatório |
| $\sigma(z)_i$ | Softmax |

# Sumário

# 1 Introdução

O aprendizado de máquina (AM) é uma área que se destaca por ser extremamente abrangente, sua utilização oferece grandes oportunidades e gera conhecimentos que culminam no desenvolvimento de benefícios ou recursos que podem ser essenciais à sociedade. Desta forma, pode apresentar diversas aplicações práticas.

Neste contexto, a inteligência artificial é um ramo da ciência que estuda e reproduz inteligência para beneficiar as máquinas de algum tipo de habilidade e que simule a inteligência natural do homem (FERNANDES, 2003). Sua área mais acentuada são as de planejamento autônomo, jogos, controle autônomo, diagnóstico, planejamento logístico, reconhecimento de linguagem e resolução de problemas (PEREIRA, 2005).

Entre as diversas técnicas é possível destacar as redes neurais, modelos matemáticos que se assemelham a estruturas neurais biológicas (FERNEDA, 2006), sendo considerado ferramentas essenciais para diagnóstico de doenças. O que aproxima novamente o AM ao contexto atual.

O presente trabalho se enquadra em um novo campo denominado Medicina 4.0 (WOLF; SCHOLZE, 2017), em que uma das aplicações foco inclui uma combinação de tecnologias inovadoras de inteligência artificial, incluindo algoritmos de deep learning, para assim desenvolver Sistemas de Apoio à Decisão Clínica.

O sistema de apoio à decisão está relacionado a procedimentos que melhoram as decisões clínicas, fornecendo como base evidências e informações médicas no momento do contato médico-paciente ou na decisão do tratamento (SCHNURR et al., 2018).

O termo Digital Health foi designado para a indústria 4.0, que acabou atingindo o setor da saúde. Tal termo é definido como uma mudança de paradigma interdisciplinar que capacita pacientes, médicos, desenvolvedores, engenheiros e cientistas a serem colaborativamente responsáveis pela aquisição e interpretação de dados de natureza genética, fisiológica e patológica (YODA, 2016). Com esses dados coletados através de médicos e pacientes por meio de exames, operações, smartphones, aplicativos e sensores, o segmento da medicina 4.0 se abastece de informações para desenvolvimento de técnicas que aceleram, inovam e prescrevem terapias e tratamentos médicos mais direcionados.

A medicina 4.0 traz transformação digital na saúde em toda sua extensão. Mas isso exige criatividade, conhecimento científico, ética, assim como criação de funções para construção e gerenciamento de novas tecnologias e desafios do ramo. Essa transformação digital desenvolvida em conformidade permite a criação de tecnologias mais efetivas para apoio aos profissionais de saúde.

Devido ao rápido contágio do COVID-19 e a morte em um curto intervalo de tempo, é imprescindível a busca e investigação de maneiras que possam melhorar e agilizar a tarefa do diagnóstico (WILSON et al., 2020). Neste contexto, atendendo às novas tendências da Medicina 4.0 este trabalho baseia-se na investigação do desempenho de quatorze redes neurais convolucionais (CNNs). As redes neurais profundas, especialmente as CNNs são concebidas por muitas camadas e conexões, tornando-os computacionalmente intensivas para o treinamento (MARQUES et al., 2018). Assim, uma solução chamada aprendizagem por transferência pode ser aplicada para superar tal problema (MENEGOLA et al., 2017). Diante do exposto, este trabalho investiga o uso de redes neurais profundas para ajudar no diagnóstico de doenças pulmonares inicialmente (COVID-19 e pneumonia viral) analisando imagens de radiografias torácicas.

As CNNs estão incluídas na área de visão computacional por serem utilizadas para reconhecimento de padrões e classificação de imagens. Na análise de uma imagem as CNNs trabalham com campos receptivos, que tem como base a reprodução do funcionamento do córtex visual biológico. O objetivo é poder analisar cada imagem em pequenas regiões processando pixels em neurônios diferentes e manter a proximidade espacial dos pixels para captura de traços e detalhes importantes das imagens.

Avaliando quatorze principais arquiteturas de redes neurais disponibilizadas pela API do keras para download (KERAS, 2020)(DenseNet169, DenseNet201, Resnet50, Mobilenet, VGG16, Mobilenet-V2, DenseNet121, Nasnet Large, VGG19, Xception, Inception, Resnet-V2, Inception-V3, Nasnet Mobile e Resnet50-V2) para classificar as doenças pulmonares citadas, é realizado treinamento em uma base de imagens composta por 7,187 imagens de raios-X levantadas e disponibilizadas por profissionais de saúde e disponibilizadas na plataforma Kaggle (KAGGLE, 2021c), as características comuns observadas nas imagens de raios-X de pacientes com COVID-19 (infiltrados irregulares ou com opacidades) que apresentam semelhanças com outras características de pneumonia viral sendo considerada a forma mais comum de detecção de doenças pulmonares, sendo, portanto, viável para detectar a infecção por COVID-19.

Profissionais da saúde atualmente estão em busca de uma solução para gerenciar o histórico de seus pacientes e os registros de tratamento. Isso é altamente essencial para que prestem seus serviços de forma eficiente e pontual, juntamente com o atendimento direcionado aos pacientes. Uma solução que auxilie os profissionais de saúde é de suma importância para um diagnóstico preciso e efetivo no tratamento das doenças pulmonares citadas, com isso é proposto um aplicativo embarcando um modelo treinado que atenda aos devidos fins de apoio clínico.

## 1.1 Motivação

O tratamento de doenças pulmonares voltou a ser um tema discutido e estudado devido à corrida contra a cura do novo coronavírus, e para primeiras medidas, inclusive isolamento social do portador, é preciso primeiramente diferenciá-la de outras doenças pulmonares (OLIVEIRA et al., 2020). Entre os métodos de diagnósticos para detecção de doenças pulmonares estão a radiografia torácica, tomografia computadorizada, angio tomografia, tomografia por emissão de prótons (TEP), ressonância magnética, ultrassonografia e cintilografia nuclear do pulmão (DEZUBE, 2019), a mais popular e acessível tanto pelo custo quanto disponibilidade continua sendo a radiografia torácica (BARROS et al., 2006). Nesse contexto observa-se uma necessidade de diferenciação dos tipos de doenças pulmonares, assim como o diagnóstico preciso e direcionado para preconizar medidas e aumentar as chances de cura do paciente.

De acordo com o CDC (Centers for Disease Control and Prevention), o novo coronavírus de 2019 é um vírus identificado como a causa de um surto de doença respiratória detectado pela primeira vez em Wuhan, na China (ZHU et al., 2020).

O vírus já se espalhou para praticamente todos os países do mundo, causando muitas mortes e sérios problemas na economia. Durante a pandemia gerada pelo novo coronavírus, a humanidade se deparou com um novo contexto de adaptação e sobrevivência (MíGUEZ, 2020), diante desse problema, além das doenças pulmonares existentes inclui-se mais uma para tratamento e/ou mitigação.

O advento de aplicativos móveis de saúde inteligentes oferece soluções para os desafios enfrentados por médicos e pacientes, especialmente para tratamentos de longa duração. De acordo com a American Medical Association (AMA, 2016), foi realizada pesquisa conduzida pela Kantar TNS com 1300 médicos onde 85% concordaram que os aplicativos e as ferramentas digitais os ajudaram a melhorar o atendimento ao paciente. A pesquisa revelou que há um sentimento de entusiasmo entre os médicos pela saúde digital e que tem potencial para melhorar a eficiência da prática, a segurança do paciente e a capacidade de diagnóstico. No trabalho de Nerminathan et al. (2017) foi realizada uma pesquisa respondida por 109 médicos, mostrou que 91% possuíam um smartphone e 88% usavam seus dispositivos móveis com frequência no ambiente clínico.

Atrelado ao contexto, as técnicas de AM, deep learning, redes neurais convolucionais e transferência de aprendizagem ligadas a avaliação das imagens geradas das radiografias torácicas podem ser de grande auxílio aos profissionais de saúde para alguns tipos de diagnósticos, o trabalho se complementa quando todas essas técnicas podem ser consolidadas e gerada uma ferramenta funcional para submissão de imagens e diagnóstico com um bom grau de precisão das doenças pulmonares.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

Desenvolver uma aplicação móvel para classificação de doenças pulmonares usando imagens torácicas de raio-X.

### 1.2.2 Objetivos Específicos

- Testar diferentes arquiteturas de transfer learning;
- Melhorar o treinamento com técnica de Data Augmentation;
- Embarcar o modelo treinado em uma aplicação móvel;
- Desenvolver aplicação móvel para auxílio no diagnóstico de doenças pulmonares com base no modelo treinado e embarcado;

## 1.3 Trabalhos Relacionados

Redes neurais regulares, como redes MLP, geralmente eram treinadas do zero, contando apenas com dados de treinamento de orientação. Então, em 1991, Pratt et al. (1991) sugeriram que o treinamento de uma rede neural poderia ser "reciclado" a fim de acelerar o processo de aprendizagem, chamando essa abordagem de aprendizagem por transferência. A técnica melhorou ao longo dos anos com mais recursos computacionais.

Em 1998, Lecun et al. (1998) propôs uma extensão de Redes Neurais, denominada CNN, que é concebida por várias camadas visando a classificação de imagens; portanto, é uma evolução natural que essa nova arquitetura também use a técnica de transferência de aprendizagem. Particularmente, esses tipos de redes neurais têm a capacidade inata de detectar padrões em imagens, tornando-as atraentes para o processamento de imagens biomédicas.

Desde então, muitas aplicações que ajudam a diagnosticar muitas doenças têm sido propostas. Por exemplo, Gulshan et al. (2016) tenta detectar a retinopatia diabética em fotografias de retina, apresentam excelentes resultados, porém o método é investigado em somente uma CNN (Inception-V3). O trabalho de Bejnordi et al. (2017) ajuda a detectar metástases nos gânglios linfáticos, esse trabalho avalia 25 redes neurais convolucionais, porém a base de dados é pequena para generalização (399 imagens). Ismail e Sovuthy (2019) utilizam duas CNNs, chamadas VGG16 e ResNet50, para identificar o cancro da mama no conjunto de dados de raios X da IRMA. Šarić et al. (2019) investiga a classificação do cancro do pulmão em imagens histopatológicas, utilizando também VGG16 e ResNet50.

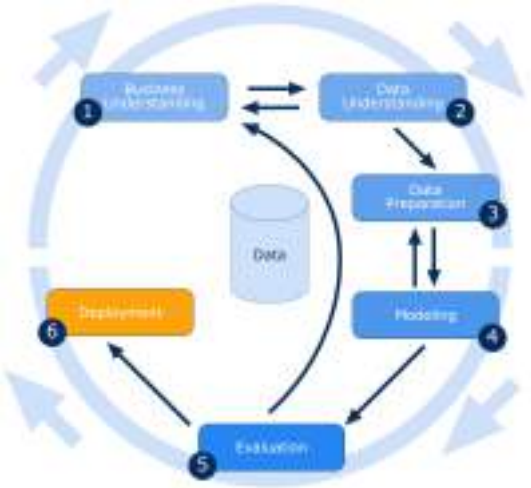
Os dois últimos trabalhos apresentam excelentes resultados porém sem um produto final para utilização (software).

Recentemente, devido à pandemia de COVID de 2020, esforços foram direcionados para ajudar no diagnóstico de COVID-19 porque é uma doença rápida; portanto, quanto mais rápido o diagnóstico, melhores são as chances do paciente. Wang et al. (2020) investiga o uso de três arquiteturas CNN denominadas ResNet50, ResNet101 e ResNet152 em um banco de dados de imagens de raios-X, trabalho com bons resultados e mesclagem de redes Resnet porém a base de imagem categorizada como COVID-19 bem reduzida (136) imagens no total. Waheed et al. (2020) propõe um modelo para aumentar um conjunto de dados de raios-X para melhorar a classificação do VGG16 CNN, conjunto de dados pequenos porém com a técnica de aumento de imagem consegue sair de 85% a 95% de acurácia. Horry et al. (2020) estuda sete modelos CNN para classificar COVID em três conjuntos de dados diferentes. No presente trabalho, investiga-se o desempenho de 14 CNN chamados: DenseNet169, DenseNet201, Resnet50, Mobilenet, VGG16, Mobilenet-V2, DenseNet121, Nasnet Large, VGG19, Xception, Inception Resnet-V2, Inception-V3, Nasnet Mobile e Resnet50-V2. Conjunto de imagens de raio-X e ultrassonografia pequenos com resultados dos testes para raio-X menores que 75% de acurácia.

Gurbeta et al. (2018) apresenta o desenvolvimento e teste em tempo real de um sistema automatizado de telessaúde para diagnóstico especializado de 2 doenças respiratórias, asma e Doença Pulmonar Obstrutiva Crônica (DPOC). O sistema utiliza tecnologias Android, Java, MATLAB e PHP e consiste em um espirômetro, um aplicativo móvel e um sistema de diagnóstico especializado. O sistema é baseado em uma conexão via Bluetooth ao APP que possui um modelo treinado para identificação das duas possíveis doenças respiratórias. Apesar da ótima precisão do método, a aplicação necessita de dispositivo extra com valor elevado e as tecnologias de desenvolvimento de software são limitadas.

Imran et al. (2020) desenvolveu um teste por meio de um aplicativo móvel denominado AI4COVID-19. O aplicativo AI4COVID-19 requer registros de tosse de 2 segundos do indivíduo. Ao analisar as amostras de tosse por meio de um mecanismo de IA em execução na nuvem, o aplicativo retorna um diagnóstico preliminar em um minuto. Infelizmente, a tosse é um sintoma comum em mais de duas dúzias de condições médicas não relacionadas ao COVID-19. Isso torna o diagnóstico COVID-19 de tosse isolada um problema extremamente desafiador. Resolveram esse problema desenvolvendo uma nova arquitetura de IA avessa a riscos centrada em mediador multifacetado que minimiza diagnósticos incorretos. O mecanismo de IA pode distinguir entre tosse de paciente COVID-19 e vários tipos de tosse não COVID-19 com mais de 90% de precisão. O sistema é bem promissor porém de processamento intensivo (2 minutos para retorno de predição), o sistema só funciona com conexão de internet ativa (processamento na nuvem).

Figura 1 – Ciclo da Metodologia CRISP-DM




Fonte: (CRISP-DM, 2000)

## 1.4 Metodologia

Adotou-se o Cross Industry Standard Process for Data Mining ou Processo padrão para a mineração de dados em toda a indústria (CRISP-DM) para construção do trabalho, trata-se de uma metodologia de mineração de dados e análise exploratória de dados cuja principal vantagem é a adaptação a qualquer tipo de negócio (CRISP-DM, 2000), e não depende de ferramentas para ser executada. Em termos simples, CRISP-DM é uma metodologia abrangente de mineração de dados e modelo de processo que fornece a qualquer pessoa, desde novatos a peritos em mineração de dados, um plano completo para a realização de um projeto de mineração de dados. O CRISP-DM decompõe o ciclo de vida de um projeto de mineração de dados em seis fases: compreensão do negócio, compreensão de dados, preparação de dados, modelagem, avaliação e implementação (SHEARER, 2000), a figura 1 representa o ciclo da metodologia.

Pensando no auxílio aos profissionais de saúde, inicialmente em verificar a diferença entre os casos de COVID-19 e Pneumonia viral, foi avaliado a necessidade de criação de uma ferramenta que auxiliasse no diagnóstico rápido e assertivo para tomadas de decisões por meio de app embarcado com modelo treinado para diagnóstico de doenças pulmonares. Diante disso utilizou-se a metodologia CRISP-DM.

No processo de ***entendimento do negócio*** foi considerado a necessidade de criação de uma ferramenta capaz de auxiliar os profissionais de saúde no diagnóstico de doenças pulmonares e assim direcionar tratamento efetivo diante do tipo de enfermidade, uma vez que na fase inicial da doença é de extrema importância para a adoção de medidas terapêuticas adequadas (ATHANAZIO, 2012). Para critério de sucesso estima-se o desenvolvimento de um modelo que possua bons parâmetros de avaliação capaz de ser exportado/consumido por uma aplicação móvel que será o produto final.

Para a ***compreensão de dados*** foi realizada a coleta de imagens de raio-X da região torácica de indivíduos. Essa fase de captura das imagens foi a junção de duas bases de radiografias disponibilizadas na plataforma Kaggle disponibilizadas para competição no diagnóstico de COVID-19.

Na fase de ***preparação dos dados*** foi realizada a segregação das bases. A base final contabilizou 7,187 imagens nos formatos (PNG e JPEG) segregadas em 3 categorias (NORMAL, COVID-19 e PNEUMONIA VIRAL). Para a *seleção dos dados* foi realizada a divisão das bases de treino e teste para cada categoria com propósito organizacional das classes de treino e teste dos modelos. No processo *limpeza dos dados* foi realizada a padronização das dimensões das imagens obtidas para atender requisito das camadas de entrada das arquiteturas utilizadas no desenvolvimento do modelo target_size(224x224). Entrando na *construção de dados*, foi realizado conversão das imagens para array (formato que se adequa a entrada de dados e construção dos inputs de treinamento e teste das arquiteturas selecionadas.

A fase de ***modelagem*** foi realizada a configuração e implementação de 14 CNNs para avaliação de desempenho de treino e teste, além disso foi utilizado algumas técnicas de melhoria de desempenho das redes como congelamento de partes das redes com o aproveitamento da transferência de aprendizagem, data augmentation para evitar overfitting bem como aplicação da validação cruzada. Na fase de ***avaliação*** foi verificado algumas métricas de desempenho no desenvolvimento dos modelos (acurácia, precisão, recall e f1 score) com base na matriz de confusão e curva ROC AUC. Para a fase de ***implantação*** é realizada a exportação do modelo treinado de melhor desempenho, criado app para consumir modelos e realizar predição de imagem submetida.

## 1.5 Organização do Trabalho

Este trabalho é dividido nos seguintes capítulos:

- No Capítulo 2 é apresentado uma explanação sobre doenças pulmonares seus aspectos e fundamentos, é realizado um apanhado das técnicas e ferramentas de uso na elaboração do projeto;
- No Capítulo 3 é apresentado a configuração do projeto bem como as implementações das arquiteturas e desenvolvimento da aplicação;
- No Capítulo 4 é apresentado os resultados, métricas e aplicação;
- O Capítulo 5 são apresentado a conclusão deste trabalho e trabalhos futuros;
- O Apêndice A apresenta o documento de requisitos do app criado;

- O Apêndice B são apresentados o link da base de imagens consolidada, codificação do modelo e link de projeto no github;

# 2 Referencial Teórico

Neste capítulo serão abordados os conceitos que foram essenciais para elaboração deste trabalho, os tópicos a seguir refletem a abordagem inicial para entendimento do problema bem como algumas técnicas e ferramentas para desenvolvimento da solução. Os tópicos de Doenças Pulmonares e COVID-19 abordam as causas, relevância e importância de diagnóstico e direcionamento. Os tópicos de Redes Neurais Convolucionais, Keras, TensorFlow, Transfer Learning, Firebase, Flutter e Dart abordam a parte técnica e servem de embasamento para criação de modelo e desenvolvimento de aplicações.

## 2.1 Doenças Pulmonares

Com o avanço tecnológico do início deste século, criou-se a falsa expectativa de que a cura das doenças ou tratamentos eficientes e definitivos seriam uma realidade. Sob esta perspectiva, avaliar e medir qualidade de vida seria uma tarefa supérflua e sem utilidade prática (RAMOS-CERQUEIRA; CREPALDI, 2000). Porém, apesar dos progressos da tecnologia, vem-se tornando claro que a maioria das doenças não é passível de cura e, mesmo que tratamentos eficientes estejam disponíveis é necessário recursos que podem auxiliar no diagnóstico.

Nesse contexto, as doenças pulmonares são consideradas uma das maiores preocupações da saúde a nível mundial, isso ocorre devido a grande taxa de mortalidade no mundo. Uma variedade de fatores, como tabagismo (diretamente ou por meio do fumo passivo), poeira, poluição do ar, vapores químicos, entre outros, estão associados ao início e ao desenvolvimento de doenças pulmonares (KAJEKAR, 2007).

Dentre as infecções respiratórias agudas mais preocupantes causando internações graves, destaca-se a pneumonia (VASCONCELLOS et al., 2020), existem grandes variedades de pneumonia, sendo que os sintomas podem variar entre si, tendo um único órgão afetado por essa condição, o pulmão. Os principais sintomas incluem febre alta, tosse, dor no tórax, alterações da pressão arterial, confusão mental, mal-estar generalizado, falta de ar entre outros.

Em relação à abordagem diagnóstica, a radiografia de tórax (RTX) é considerada o padrão ouro para definição de pneumonia em estudos epidemiológicos (CHERIAN et al., 2005). A RTX é um dos exames radiográficos mais comum e um dos mais difíceis de interpretar, sendo uma das primeiras modalidades na avaliação de várias doenças, entre elas a pneumonia. Ele produz uma grande quantidade de informações anatômicas e fisiológicas, mas é difícil e às vezes até impossível extrair todas as informações que contém.

Consequentemente, a RTX pode ter apenas valor limitado na avaliação de algumas doenças, enquanto em outras pode servir como um dos medidores mais sensíveis e confiáveis do curso da doença (POURALIAKBAR, 2005). A Figura 2 apresenta o exame radiográfico do tórax (normal).

Figura 2 – Radiografia do Tórax (Classificação Normal)

Fonte: (KAGGLE, 2021c)

A RTX tem como objetivo servir de registro para a investigação de possíveis alterações da saúde de pacientes sintomáticos ou assintomáticos. Deve ser produzida uma imagem de boa definição e com a menor dose de radiação possível para o paciente, compatível com um diagnóstico adequado. A qualidade da imagem e a dose de radiação em um exame radiográfico estão intimamente relacionadas com as características técnicas e as condições operacionais do aparelho de raios-X, da revelação dos filmes, da combinação tela-filme, da técnica radiográfica, do operador e das condições físicas do paciente, como presença de drenos, acamado, debilitado, etc (EISENBERG et al., 1980); (KOTSUBO; AZEVEDO, 2003). Para a precisão da detecção de alguma anormalidade seja ela discreta, a imagem necessita do contraste, caso o aspecto seja insuficiente pode restringir o detalhamento. A falta de definição é um fator que geralmente limita a detecção de detalhes lineares e circulares de pequenas dimensões (KOTSUBO; AZEVEDO, 2003).

Ressalta-se que além da pneumonia, a nova variação do coronavírus (COVID-19) gerou um número crescente de pacientes com problemas respiratórios, estima-se que em 4% dos indivíduos infectados pela doença apresentam sintomas inespecíficos, como febre, astenia ou diarreia, e alguns indivíduos não apresentam qualquer tipo de sintomas. (SHI et al., 2020). Dessa forma, o diagnóstico dessas doenças de maneira rápida e eficiente é de real importância, para assim buscar a melhor forma de tratamento ou medidas de contingenciamento.

A nova síndrome respiratória aguda grave coronavírus 2 (SARS-CoV-2) associada à doença coronavírus 2019 (COVID-19), surgiu na China em dezembro de 2019 e se espalhou globalmente, criando uma pandemia mundial e uma crise de saúde pública de dimensão histórica (ZHOU et al., 2020). O espectro clínico de COVID-19 varia amplamente, de doença assintomática a pneumonia e complicações com risco de vida, incluindo síndrome do desconforto respiratório agudo (SDRA), falência de órgãos multissistêmicos e, em última análise, morte (CHEN et al., 2020); (GUAN et al., 2020); (GRASSELLI et al., 2020); (BERENGUER et al., 2020).

O quadro clínico inicial da doença é caracterizado como uma síndrome gripal. As pessoas diagnosticadas com COVID-19 geralmente desenvolvem sinais e sintomas, incluindo problemas respiratórios leves e febre persistente, em média de 5 a 6 dias após a infecção (período médio de incubação de 5 a 6 dias, intervalo de 1 a 14 dias). A febre nem sempre está presente em todos os casos, como, por exemplo, em pacientes jovens, idosos, imunossuprimidos ou em algumas situações que possam ter utilizado o medicamento antitérmico (LIMA, 2020).

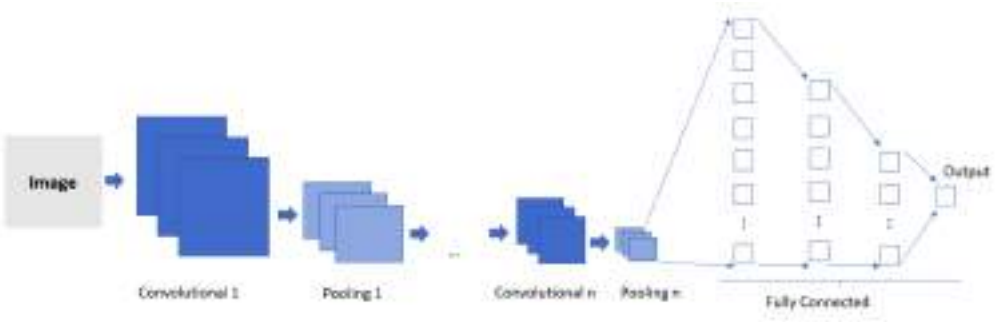
Há uma necessidade urgente de um diagnóstico e tratamento precoce da doença para evitar uma maior propagação e controlar o número de mortos, COVID-19 é caracterizada por uma infecção pulmonar (SHI et al., 2021). Nesse sentido, é comprovado que o raio-X pode ser usado de forma eficaz para o diagnóstico de COVID-19 (JACOBI et al., 2020), por ser um método rápido e acessível.

A leitura manual de raios-X de um grande número de pacientes pode ser demorada. Dessa forma, um método de diagnóstico auxiliado por uma aplicação móvel poderia ajudar a prever COVID-19 a partir de imagens de raios-X (SHI et al., 2021). Nessa perspectiva, a rede neural convolucional (CNN) tem contribuído e mostrado resultados promissores na área de detecção e diagnóstico de várias doenças.

## 2.2 Redes Neurais Convolucionais – CNN

Uma Rede Neural Convolucional é uma variação das redes de Perceptrons de Múltiplas Camadas (MLP), em vez de usar camadas ocultas totalmente conectadas, como a MLP, a arquitetura de uma CNN se baseia na alternação de camadas de convolução – camada que dá o nome a rede; e camadas de pooling. Cada camada possuirá um conjunto de filtros, também conhecido como kernel, que serão responsáveis por extrair características locais de uma entrada. Com isso, diversos mapas de convolução e pooling podem ser criados, contendo diversas características específicas como: bordas, intensidade de cor, contornos e formas. Cada mapa de características possuirá um conjunto de pesos compartilhados, que diminui a complexidade computacional da rede (VARGAS et al., 2016). A figura 3 apresenta uma ilustração de uma CNN.

Figura 3 – Representação de uma CNN




Fonte: (SILVA; CORTES, 2020)

Diferentemente das redes neurais convencionais, as camadas convolucionais das CNNs, possuem a responsabilidade de aplicar diversos filtros no dado de entrada, de forma a constituir uma espécie de hierarquia de características que resultam em descritores da imagem precisos (LIMA et al., 2017).

As redes neurais convolucionais fazem parte de uma classe de RNAs denominada Deep Learning, ferramenta poderosa da neurocomputação (BENGIO et al., 2012). Tem sido utilizada em uma gama de aplicações, onde pode-se destacar o emprego na tecnologia de carros autônomos (BECHTEL et al., 2018). Na área de visão computacional são destinadas a classificação de imagens. São redes que em sua maioria possuem treinamento supervisionado, ou seja, as amostras devem estar rotuladas com as classes pertencentes (SOUTO et al., 2003). Diante disso, o treinamento é dividido em duas etapas, uma composta por processamento das imagens e outra composta por ajuste de pesos sinápticos.

A etapa de processamento de imagem consiste na aplicação de filtros cujo papel baseia-se na extração das características mais relevantes da imagem e na redução da dimensionalidade dos dados. Além disso possui a etapa de ajuste de pesos pois configura-se com redes do tipo feedforward que podem ser usados algoritmos clássicos como backpropagation ou aperfeiçoamentos.

Nesse sentido, a estrutura de uma CNN é formada basicamente por:

### 2.2.1 Camada de Convolução

A camada convolução em uma CNN é responsável por extrair as chamadas características de entrada. O processo de extração dessas características se dá por meio de filtros convolucionais de tamanhos reduzidos, onde os filtros percorrem os dados de entrada em largura, altura e profundidade (chamada de dimensão) realizando a operação de convolução sobre os dados. Com a execução de entrada na rede, os filtros vão aprendendo estruturas cada vez mais complexas, ou seja, quanto mais filtros convolucionais, mais características extraímos da entrada, porém isso tem um custo de memória e processamento, o que precisa

ser balanceado na hora de definir a arquitetura (RODRIGUES, 2018).

Na camada de convolução é executado o operador linear onde a partir de duas funções obtém-se uma terceira resultando a soma do produto dessas funções nas regiões subentendidas pela superposição delas em função do deslocamento existente entre elas.

Esse tipo de operação está presente em diversas áreas, tais como óptica de Fourier, teoria das vibrações, sistemas lineares invariantes no tempo, média móvel, funções de correlação e Autocorrelação, Processamento de Sinais e Processamento de imagens.

A expressão geral para operação de convolução é dada por:

$$g(n,m) = f(n,m) * h(n,m) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(u,v) \cdot h(n-u, m-v) du dv \tag{2.1}$$

Para a qual a versão discreta é dada por:

$$g(n,m) = f(n,m) * h(n,m) = \sum_{u=-F1}^{F1} \sum_{v=F2}^{F2} f(u,v) \cdot h(n-u, m-v) \tag{2.2}$$

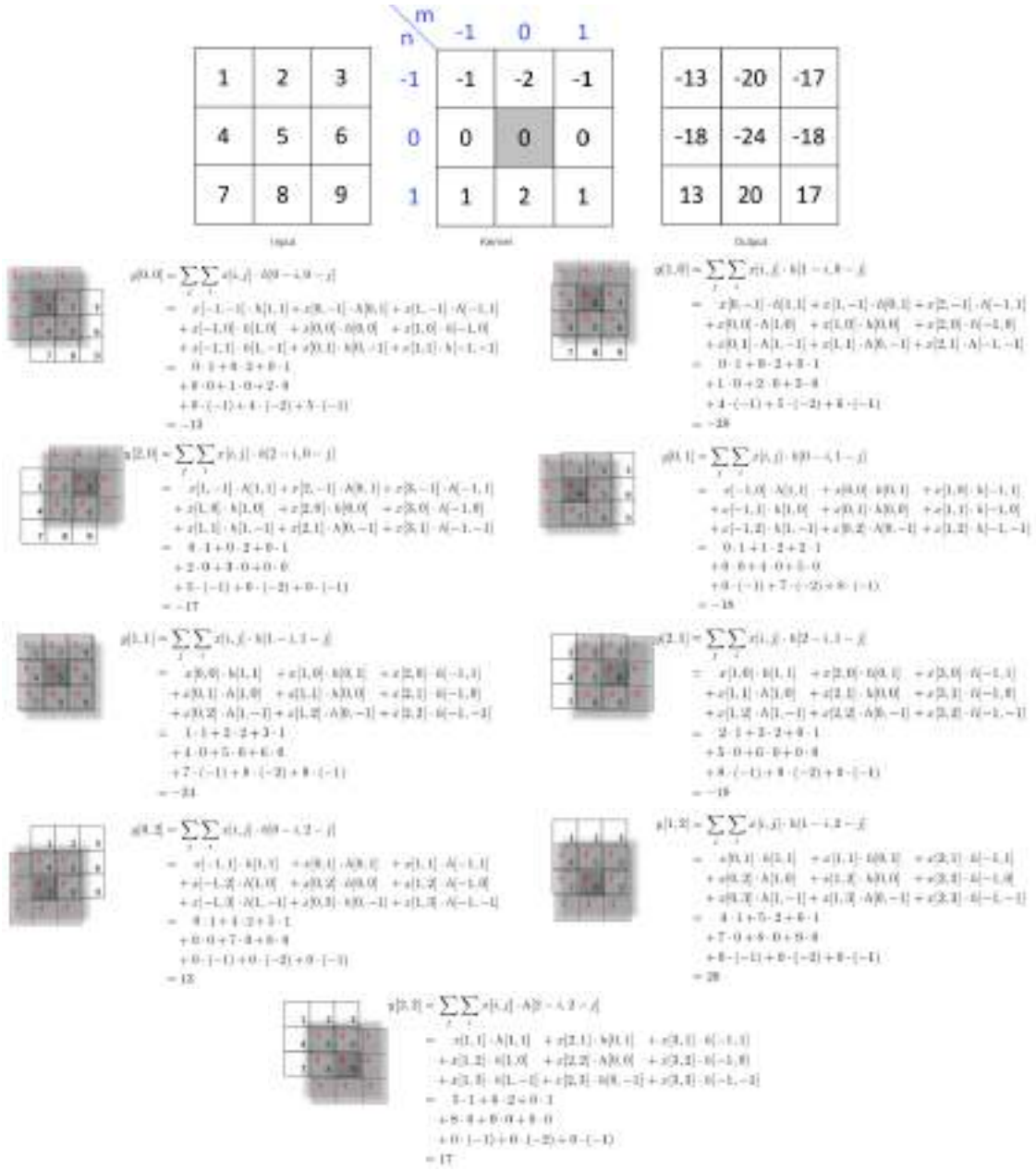
Onde $\forall\ F1, F2\ \in \mathbb{N}$, e representam as dimensões do filtro espacial, e $g(n,m)$ é a imagem resultante da aplicação do filtro de convolução, $f(n,m)$ é a imagem a ser submetida ao filtro de convolução e $h(n,m)$ é o filtro de convolução também conhecido como núcleo ou *(kernel)*. Na prática, a operação na camada de convolução consiste em uma multiplicação não usual de matrizes conforme Equação 2.2. O exemplo de um processo de convolução está disponível na figura 4 que consiste em uma aplicação de um filtro 3x3 aplicado em um segmento de imagem com dimensão 3x3 qualquer e valores aleatórios como demonstração, o kernel bloco sombreado $n \times m$ percorre a matriz de entrada, cada célula da matriz é multiplicada pela célula do kernel totalizando 9 somas para o resultado da célula de saída, é importante ressaltar que o kernel é invertido e as matrizes são no formato [coluna, linha] para que todos as células sejam inclusas no processo de filtragem das características.

O operador de convolução em redes CNN, está diretamente relacionado ao tratamento de imagens. Atua sobre regiões de pixels na perspectiva de filtros que permitem mapear características com ações como por exemplo o realce ou suavização de elementos da imagem.

Opcionalmente, pode-se utilizar funções de ativação na operação de convolução, com o propósito por exemplo de eliminação ou destaque de partes mais escuras ou claras das imagens. Um exemplo de função de ativação usada em redes CNN é a função ReLU.

A função ReLU é utilizada para projetar redes neurais e por ser não linear significa que podemos copiar erros de passos anteriores e ter várias camadas de neurônios ativados

Figura 4 – Exemplo de convolução com filtro 3x3 sobre um segmento de imagem 3x3, valor do pixel é calculado através da equação 2.2

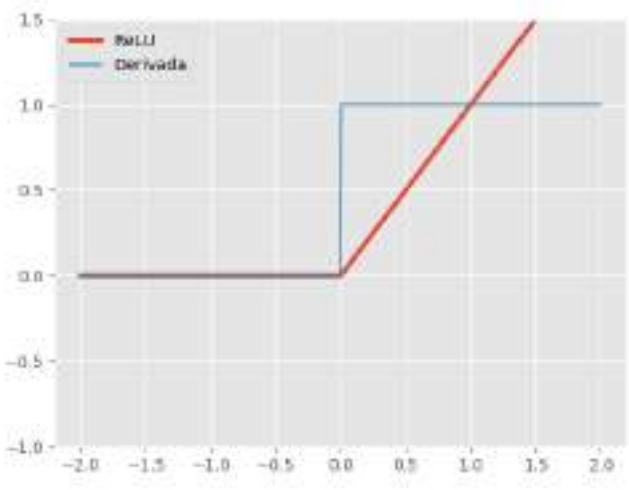



Fonte: (SONGHO, 2021)

com a função. A principal vantagem da função é tornar a rede esparsa e eficiente uma vez que na existência de entradas negativas, as mesmas serão convertidas em zero e o neurônio não será ativado. O que implica que nem todos os neurônios serão ativados (DEEPLEARNINGBOOK, 2019).

$$ReLU(x) = max\left\{0, x\right\}, ReLU' = \begin{cases} 1, \text{ se } x \geqslant 0 \\ 0, \text{ c.c} \end{cases} \tag{2.3}$$

Figura 5 – Função ReLU




Fonte: (DEEPLEARNINGBOOK, 2019)

### 2.2.2 Camada de Pooling

Também conhecida como sub amostragem, essa camada reduz a dimensionalidade de um mapa de característica fornecido como entrada e produz outro mapa de característica, uma espécie de resumo do primeiro (HIJAZI. et al., 2018). Uma técnica bastante utilizada para executar o pooling é chamada de max-pooling. Nessa técnica, reduzimos subpartes dos dados originais pelo maior valor encontrado nessas sub-regiões, reduzindo com isso o tamanho da imagem por um fator de filtro $m\,x\,n$ (RODRIGUES, 2018). A Figura 6 apresenta um mapa de características, resultante da operação de max-pooling.

O operador de pooling é aplicado após a camada de convolução para reduzir ainda mais a dimensionalidade da imagem. Ele visa extrair as características mais relevantes de uma região de pixels, podendo usar medidas matemáticas como o valor máximo, mínimo ou a média dos valores dos pixels. Esta camada opera de forma autônoma em cada canal de entrada e redimensiona o espaço, exceto a profundidade. A camada de pooling especificamente recebe um volume de tamanho $W_0$ x $H_0$ x $D_0$ Dimensões (Largura, altura e profundidade das imagens), hiperparâmetros requerem tamanho dos filtros $F$ idêntico às camadas convolutivas e stride $S$ também iguais às da camada convolutiva, obtém-se um volume de saída com dimensões $W$ x $H$ x $D$ dadas por:

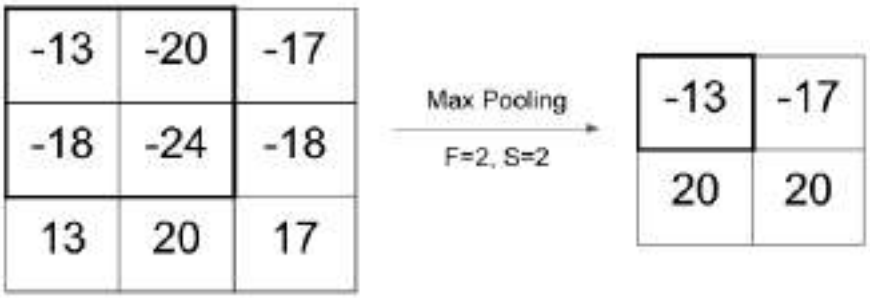
$$W = \frac{W_0 - F}{S + 1}, \tag{2.4}$$

$$H = \frac{H_0 - F}{S + 1}, \tag{2.5}$$

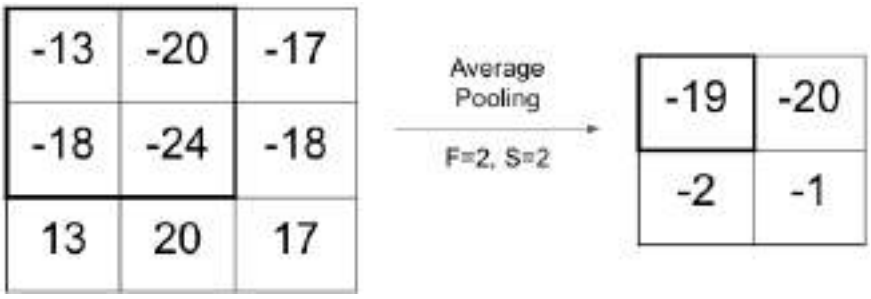
$$D = D_0 \tag{2.6}$$

Figura 6 – (a) MaxPooling mantém maior valor da máscara, (b) Average Pooling mantém a média dos valores da máscara. (c) Downsampling de imagem RGB após Maxpooling da imagem. Mantido canais (D = 3).

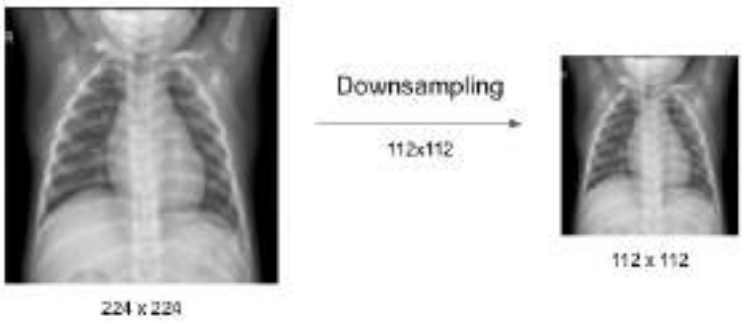
(a) MaxPooling




(b) Average Pooling




(c) DownSampling




Fonte: Autor

### 2.2.3 Camada Totalmente Conectada

As camadas totalmente conectadas normalmente se situam ao final da rede. Nessas camadas os features extraídos nas camadas de convolução anteriores são utilizados para se ter a saída de classificação da rede (RODRIGUES, 2018).

Ao realizar a conexão da camada de pooling para a camada totalmente conectada é necessário na maioria dos casos a existência do operador flattening que consiste em converter os dados obtidos após a aplicação do pooling em formato unidimensional para submissão a rede densa ou MLP.

Uma vez produzido o flattening para conexão na camada densa, objetiva-se trabalhar com essa rede MLP, os neurônios atuam sobre todos os dados fornecidos pelas camadas anteriores e cada neurônio da camada de saída representa uma classe do problema.

É esperado na camada totalmente conectada o aprendizado de como classificar os atributos em um conjunto de dados. Para as saídas que representam as classificações é utilizado a função softmax que é uma função utilizada quando a rede é voltada para tarefas de classificação, permite interpretar os valores de saída como uma distribuição de probabilidades conforme Equação 2.7.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{k_z}} \; para \; i = 1, ..., k \; e \; z = (z_1, ..., z_k) \in \mathbb{R}^k \tag{2.7}$$

A função softmax é apropriada para a saída de um classificador multiclasse onde espera-se a recepção de probabilidades das classes de entrada. Consiste em transformar as saídas para cada classe em valores entre 0 e 1 e divide-se pela soma das saídas, representado a probabilidade da entrada estar em uma classe.

## 2.3 Arquiteturas de Redes Neurais Convolucionais

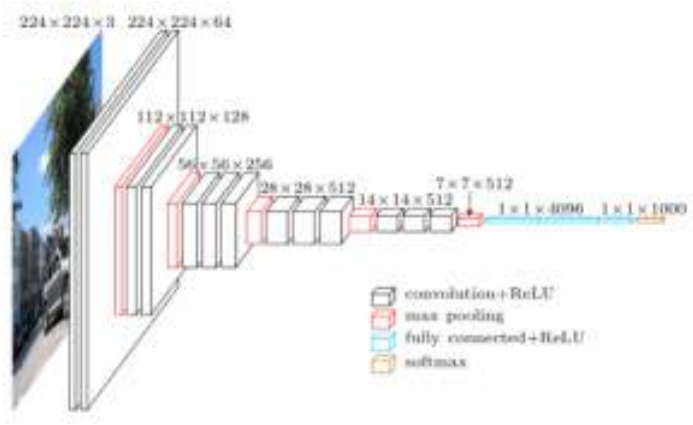
Como projetos iniciais de arquiteturas de redes neurais CNNs podemos citar a rede LeNet que foi proposto por Lecun et al. (1998) que já continham camadas de convolução com filtros 5x5 e agrupamento de filtros 2x2, primogênita entre as redes convolucionais em seguida podemos destacar a AlexNet proposta por Krizhevsky et al. (2012) composta por 5 camadas de convolução e a primeira utilização da função ReLU nas CNNs, Essa arquitetura ganhou a competição ImageNet Large Scale Visual Recognition Challenge (ILSVRC) com erro de 10% a menos que o segundo colocado. A partir de então as redes CNNs se tornaram populares para processamento de imagens, reconhecimento de fala e processamento de linguagem natural. A seguir apresenta-se uma explanação de 3 redes que foram essenciais no desenvolvimento deste trabalho e que possuem os melhores resultados atualmente na comunidade e o número total de notebooks implementados na plataforma kaggle (ResNet - 2,939, VGGNet - 1,541, DenseNet - 761) (KAGGLE, 2021c) .

### 2.3.1 VGGNet

VGGnet é uma rede neural convolucional proposta por Simonyan e Zisserman (2015) da universidade de Oxford. Consiste em uma entrada de imagem 224x224 RGB

com pixels no intervalo de 0 a 255 e realiza subtração de valores médios calculados sobre o conjunto de treinamento do ImageNet. As imagens de entrada são passadas por camadas de pesos, e as imagens de treinamento são submetidas para uma pilha de camadas de convolução. Como a VGGNet também é explorada em grandes conjuntos de dados, a aprendizagem por transferência é frequentemente usada com ela para evitar problemas de overfitting quando o conjunto de dados de treinamento é pequeno (NASIRI et al., 2020). A figura 8, apresenta a arquitetura da Rede Convolucional (VGGNet).

Figura 8 – Arquitetura VGGNet



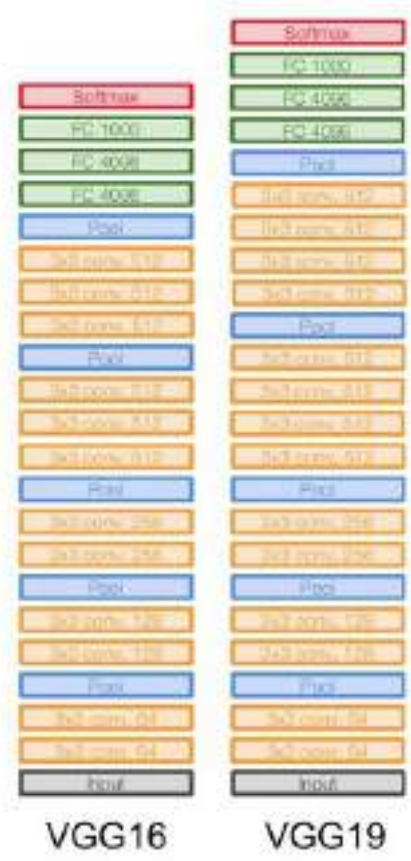

Fonte: (BEZDAN; BACANIN, 2019)

O VGGNet inclui o VGG-16 e a VGG-19 Figura 9, o que aumenta a profundidade da rede (DA SILVA COTRIM et al., 2020), sendo um total de 13 camadas convolucionais e 3 camadas totalmente conectadas na arquitetura VGG-16 tendo filtros menores 3 x 3 com mais profundidade em vez de filtros grandes. Outra variação do VGGNet tem 19 camadas de peso consistindo em 16 camadas convolucionais com 3 camadas totalmente conectadas e as mesmas 5 camadas de agrupamento. Nesse sentido, a VGG19 usa uma estrutura alternada de várias camadas convolucionais e camadas de ativação não linear, que é mais eficaz do que uma única camada convolucional e facilita uma melhor extração de recursos de imagem (YU et al., 2021).

### 2.3.2 ResNet

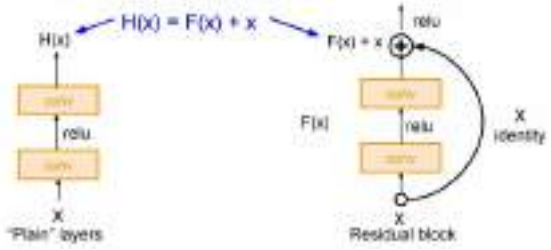
A principal característica das arquiteturas Resnet é o bloco residual, estes blocos tem uma entrada x que passa por uma série de operações de convolução-relu-convolução, o resultado da operação $F(x)$ é adicionado à entrada original $x$. Quanto mais a rede se aprofunda com um grande número de camadas, a computação se torna mais complexa. Essas camadas são empilhadas e todas as camadas tentam aprender algum mapeamento subjacente da função desejada e assim realizado o mapeamento residual. A Figura 10 mostra a arquitetura do bloco.

Figura 9 – Arquiteturas: VGG16 e VGG19




Fonte: (SIMONYAN; ZISSERMAN, 2015)

Figura 10 – Blocos ResNet




Fonte: (HE et al., 2016)

Essas camadas adicionadas são o mapeamento de identidade e as outras camadas são copiadas do modelo mais raso aprendido. Esta solução indica que um modelo mais profundo não deve produzir um erro de treinamento maior do que sua versão da rede mais rasa (HE et al., 2016); (DA SILVA, 2018). A equação 2.8 mostra o processamento que o bloco residual faz:

$$H(x) = F(x) + x \tag{2.8}$$

A função $F(x)$ é apenas uma regularização. Nesse sentido, o espaço de saída é somente uma alteração do espaço de entrada. Logo, ResNet mostra que os mapas residuais são mais fáceis de serem otimizados por meio dessa alteração no espaço de entrada (HE et al., 2016).

### 2.3.3 DenseNet

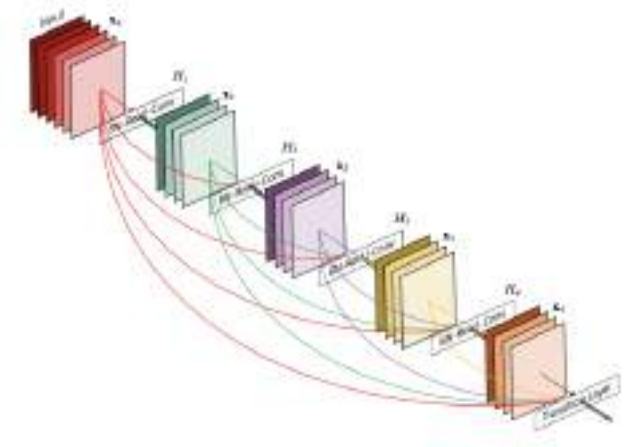
O DenseNet foi desenvolvido em 2017, consiste de camadas convolutivas densamente conectadas. As saídas de cada camada são conectadas com todas as camadas sucessivas, formando um bloco denso. Esse modelo de rede é eficiente para o reuso de mapas de características diversas, o que reduz bastante o número de parâmetros da rede (DA ROSA, 2019). Portanto, a rede DenseNet possui vários blocos densos e blocos de transição, estes últimos são colocados entre dois blocos densos adjacentes. Este modelo apresenta precisão de estado da arte com um número razoável de parâmetros para tarefas de reconhecimento de objetos (HUANG et al., 2017); (DA ROSA, 2019).

As principais vantagens já percebidas através da utilização das DenseNets aplicadas nas tarefas de reconhecimento de padrões incluem: redução do problema do gradiente de desaparecimento, aumento na propagação/reutilização de características e redução significativa da quantidade de parâmetros. Logo nessas CNNs não há necessidade de reaprender mapas de características redundantes, exigindo menos computação e obtendo assim um alto desempenho (HUANG et al., 2017); (REZENDE, 2019). Uma representação da densent e algumas camadas é mostrado na figura 11

### 2.3.4 Mobilinet

Em 2017, (HOWARD et al., 2017) propuseram uma nova classe de CNN chamada MobileNet. Designada para ser uma rede pequena, permite uma integração mais rápida e fácil em aplicações para dispositivos móveis. Além disso, (TOGAçAR et al., 2020), apresenta a MobileNet como um modelo de aprendizagem a ser utilizado em dispositivos de baixo custo de hardware. Devido à sua profundidade separável estratégia de convolução, a MobileNet reduziu a sua complexidade computacional. A figura mostra a arquitectura

Figura 11 – Representação de 5 camadas Densenet



Fonte: (HUANG et al., 2017)

desta CNN. A arquitectura da MobileNet disponível na documentação da biblioteca Keras apresenta a rede pré-formada na base de dados ImageNet

Figura 12 – Arquitetura Mobilinet

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32$ dw | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64$ dw | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| $5\times$ Conv dw / s1 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| $5\times$ Conv / s1 | $1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024$ dw | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

Fonte: (HOWARD et al., 2017)

## 2.4 Keras

Keras é uma API escrita em Python com uma rede neural de alto nível de código aberto (CHOLLET, 2017). Foi desenvolvida por François Chollet, que atualmente é

engenheiro de Software da Google. O Keras pode ser utilizado/executado em diferentes bibliotecas de aprendizado de máquina, na qual podemos citar: TensorFlow, CNTK e Theano, desenvolvido pelo Google, pela Microsoft e pela University of Montreal, respectivamente (NHU et al., 2020). O Keras visa fornecer aos usuários uma biblioteca com grande facilidade de uso, modularidade e extensibilidade. E além de modelos sequenciais, o Keras também oferece suporte às arquiteturas de modelo mais complexas por meio de sua API funcional. Isso permite que os modelos tenham várias entradas e saídas, ramificações e mesclagens internas, além de reutilizar camadas específicas várias vezes no mesmo modelo e oferecer diversos algoritmos de otimização padrão (CONLIN et al., 2021). Então devido a sua notoriedade na comunidade não somente científica, a biblioteca Keras é empregada neste estudo para construir modelos juntamente com o TensorFlow.

## 2.5 TensorFlow

Atualmente o TensorFlow é considerado uma estrutura de aprendizado profundo mais utilizada, por ser projetada para simplificar a construção de redes neurais profundas e acelerar o processo de aprendizagem com um ambiente computacional distribuído heterogêneo, dessa forma, se tornou uma biblioteca mais genérica para computação numérica, facilitando problemas de otimização numérica em grande escala, ou seja, problemas e soluções inversas de equações diferenciais parciais (LIU; GRANA, 2020).

O TensorFlow adota o gráfico de fluxo de dados, para representar toda a computação e estado em um algoritmo de aprendizado de máquina, incluindo as operações matemáticas individuais, os parâmetros e suas regras de atualização e o pré-processamento de entrada (ABADI et al., 2016). Este modelo é utilizado para expressar e organizar o fluxo de trabalho de computação e, em seguida, mapear as operações matemáticas no gráfico para diferentes dispositivos computacionais (CPUs, GPUs e TPUs) (LIU; GRANA, 2020).

O TensorFlow é considerado altamente modular podendo ser utilizado com interface cliente servidor, neste trabalho a sua utilização foi vinculado ao keras com o objetivo de melhorar o desempenho/processamento na criação de deep learning, facilitando a utilização.

## 2.6 Transfer Learning

Normalmente, uma arquitetura CNN é concebida por três componentes: camada convolucional, camada de pooling e uma totalmente conectada. As camadas convolucionais usam filtros que examinam uma parte da imagem e extraem recursos dela. Esses recursos são geralmente cores, formas e bordas que definem uma imagem específica (BEYSOLOW, 2017). CNNs podem ter quantas camadas convolucionais forem necessárias. Quanto mais

camadas convolucionais, mais recursos são extraídos.

Após as camadas convolucionais, adicionamos camadas de agrupamento que são responsáveis por agrupar os mapas de recursos em uma imagem (BEYSOLOW, 2017), proporcionando uma redução de dimensionalidade, que é produzida pela aplicação de um único máximo ou média dos valores dentro de uma caixa produzida pela camada convolucional. Assim, as camadas totalmente conectadas, que é uma rede MLP normal, recebem uma imagem menor do que aquela apresentada na entrada da CNN.

Como várias camadas planejam uma CNN, podemos rapidamente encontrar outras pré-treinadas para "reciclar" o conhecimento anterior. Normalmente, essas CNNs eram treinadas por um conjunto de dados de imagens acessível denominado ImageNet (IMAGE-NET, 2020); (DENG et al., 2009), que é composto por 14.197.122 imagens; no entanto, outros conjuntos de dados também podem ter sido usados, dependendo do aplicativo. Assim, a ideia do aprendizado de transferência é atualizar os parâmetros da CNN usando um novo conjunto de imagens. Nesse contexto, (TORREY; SHAVLIK, 2009) definem aprendizagem por transferência como a melhoria da aprendizagem em uma nova tarefa por meio da transferência de conhecimento de uma tarefa relacionada que já foi aprendida.

Ao todo, a aprendizagem por transferência pode apresentar três benefícios principais (SILVA; CORTES, 2020):

- Podemos usar modelos cuidadosamente projetados por especialistas;
- Como os especialistas criaram esses modelos, não precisamos nos preocupar com qual arquitetura ou camadas usar ou incluir;
- Devido ao design cuidadoso, eles tendem a ter um bom desempenho na detecção de imagens;

O ImageNet acelerou significativamente o desenvolvimento da área de AM, e competições foram criadas em cima da estrutura da mesma, para julgar a qualidade de algoritmos de classificação de imagem. E seus benefícios à área não param neste problema de classificação de imagens: um benefício percebido por pesquisadores, fora da competição, foi que seus algoritmos apresentaram resultados melhores se fossem treinados em cima do banco de dados da ImageNet.

## 2.7 Flutter e Dart

Flutter (2020) é um framework de interface para aplicações móveis do Google onde permite-se criar interfaces nativas de alta qualidade no iOS e Android em tempo recorde. O Flutter possui integração com a linguagem Dart, é usado por desenvolvedores

e organizações em todo o mundo e é gratuito e de código aberto. Quando se trata de aplicativos móveis, ter desempenho de processamento e fluidez é de suma importância para atender as expectativas da experiência do usuário.

Uma das maiores vantagens é a construção de uma única codificação e execução em múltiplas plataformas de forma nativa e com um desempenho superior aos demais frameworks. Por exemplo, o ionic é uma plataforma que trabalha com webview, o react trabalha em uma linha parecida com o flutter com componentes nativos porém converte o código em javaScript fazendo chamadas aos componentes nativos citados. A arquitetura do flutter é dividida em camadas com mostra a figura 13, sendo cada uma delas as responsáveis por cada funcionalidade em uma aplicação.

Figura 13 – Arquitetura framework Flutter


Fonte: (FLUTTER, 2020)

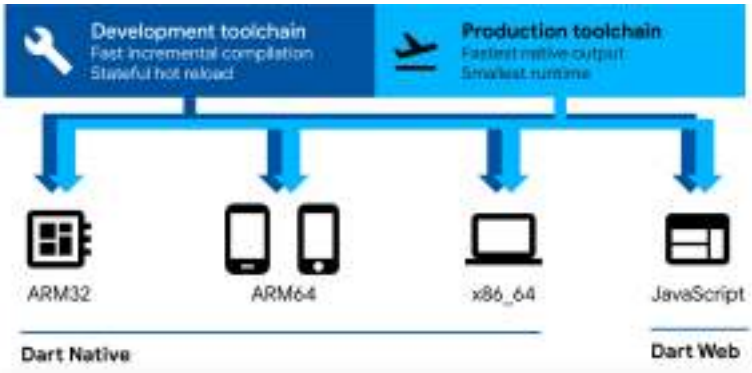
Conforme a arquitetura do framework observa-se o Dart como linguagem de programação baseada em C++.

Para programas que visam dispositivos (móveis, desktop, servidor e outros), o Dart Native inclui uma Dart VM com compilação JIT (just-in-time) e um compilador AOT (antecipado) para produzir código de máquina (DART, 2020). O Dart é usado para escrever scripts simples ou aplicativos completos. Útil para criar aplicativos móveis, aplicativos para web, script de linha de comando ou aplicativos para servidores. A tecnologia de compilador flexível permite a execução do código Dart de maneiras diferentes, dependendo de sua plataforma de destino e objetivos conforme figura 14.

## 2.8 Firebase

A Plataforma Firebase foi desenvolvida pela Google, tendo o objetivo auxiliar na criação de aplicativos. O Firebase disponibiliza diversas ferramentas para desenvolver aplicativos móveis de alta qualidade e ampliar a base de usuários, desde ferramentas de autenticação de usuário, até ferramentas que monitoram a atividade do aplicativo

Figura 14 – Plataformas de compilação Dart




Fonte: Dart (2020)

(GAJARDO, 2018). A Firebase tem o propósito de resolver três problemas principais: a criação de um aplicativo de forma rápida, a liberação e o monitoramento de um aplicativo com confiabilidade e a capacidade de envolver usuários.

Os recursos do Firebase são ideais para desenvolvimento de aplicações e sistemas, possui uma vasta gama de ferramentas para auxílio no desenvolvimento conforme necessidade que atendem desde criação de banco de dados artefatos de ML (FIREBASE, 2021)

Para criação do aplicativo, teremos os seguintes recursos no desenvolvimento do trabalho:

- Cloud Firestore: Banco de dados de nuvem NoSQL flexível e escalável a fim de armazenar e sincronizar dados para o desenvolvimento do cliente e do servidor.
- Firebase Authentication : Fornece serviços de back-end, SDKs fáceis de usar e bibliotecas de IU prontas para autenticar usuários no seu aplicativo. Ele oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook e Twitter, entre outros.
- Cloud Storage para Firebase: É um serviço de armazenamento de objetos avançado, simples e econômico criado para a escala do Google. Com os SDKs do Firebase para Cloud Storage, é possível usar a segurança do Google para fazer upload e download de arquivos nos apps do Firebase, independentemente da qualidade da rede.

## 2.9 Considerações Finais do Capítulo

O capítulo apresentou os principais conceitos que integraram a parte de desenvolvimento dos modelos bem como recursos técnicos para construção da aplicação. É realizado um apanhado sobre as doenças pulmonares em destaque o COVID-19, também é mostrado

a técnica de redes neurais convolucionais utilizadas para o desenvolvimento dos modelos bem como sua execução e funcionamento, em seguida é realizado uma explanação sobre as redes neurais convolucionais mais utilizadas atualmente e com melhores performances para classificação de imagens (VGGNet, ResNet e DenseNet). Estas redes utilizam a transferência de aprendizado como abordado para melhorar a técnica e obtenção de resultados de processamento e predição. Por fim é apresentado os frameworks que serão trabalhados na construção das CNNs o Keras e TensorFlow que serão essenciais para o desenvolvimentos dos modelos, é apresentado as linguagens utilizadas para construção da aplicação flutter e dart que permitem construção em múltiplas plataformas e fazem a interface com os modelos resultantes do processo da modelagem das redes, foi mostrado ainda o Firebase, plataforma do Google para auxílio na criação de base de dados e autenticação. Com base nesses conceitos, fundamentou-se a construção da aplicação que será apresentada no próximo capítulo.

# 3 Materiais e Métodos

Pensando no auxílio aos profissionais de saúde, inicialmente em verificar a diferença entre os casos de COVID-19 e Pneumonia viral, foi avaliado criação de uma ferramenta que auxiliasse no diagnóstico rápido e assertivo para tomadas de decisões por meio de aplicacão embarcada com modelo treinado para diagnóstico de doenças pulmonares.

## 3.1 Ambiente e Configuração

Implementou-se os modelos em Python 3.6 (PYTHON, 2020) utilizando as Redes Neurais do Keras (KERAS, 2020). O código e a etapa de treinamento foram feitos no Google Colaboratory (GOOGLE, 2020), também conhecido como Google Colab. Esta plataforma foi essencial para o desenvolvimento uma vez que podemos usar computação GPU para treinar a ANN. A máquina virtual é possui CPUs Intel®Xeon de 2,30 GHz, 14 GB de RAM e 37,11 GB de HD. Apesar de termos usado GPUs na etapa de treinamento, não pudemos escolher que tipo de GPU poderíamos conectar. Normalmente, Colab inclui NvidiaK80s, T4s e P4s. Usando GPUs, a etapa de treinamento de cada pasta leva cerca de 40 minutos. Cada CNN foi treinado usando 10 épocas e k-vezes com $k = 5$; portanto, a proporção de treinamento e teste foi de 80/20 com o objetivo de generalização dos modelos a partir do conjunto de dados. A otimização dos parâmetros foi feita usando o algoritmo de Adam (KINGMA; BA, 2017), o Adam é um método de descida do gradiente estocástico baseado na estimativa adaptativa de momentos de primeira e segunda ordem possui como hiperparâmetros padrão: taxa de aprendizagem = 0,001, $\beta_1 = 0{,}9$ e $\beta_2 = 0{,}999$ que são controladores do decaimento exponencial das médias móveis do gradiente e do quadrado do gradiente.

Para desenvolvimento da aplicação utilizou-se a linguagem Dart (DART, 2020) e Flutter (FLUTTER, 2020) é um UI toolkit do Google para criar aplicativos agradáveis esteticamente e nativamente compilados para dispositivos móveis , web e desktop a partir de um único código-base. Para construção da API e ambiente de backend onde é hospedado o modelo treinado foi utilizado o framework do Python FastAPI. O FastAPI é uma framework de alta performance para construção de APIs com Python 3.6 ou superior e utiliza os type hints padrões da linguagem. O framework é intuitivo, de fácil utilização, enxuto na codificação mas por ter características é bem robusto e aderente aos principais padrões abertos de APIs: OpenAPI e JSON Schema (FASTAPI, 2021).

Ambientes utilizados para desenvolvimento:

- Implementação das arquiteturas e criação de modelo - Google Colab

- Implementação do APP - Android Studio (4.1.1)
- Desenvolvimento de API - Spyder (4.1.5)
- Emulador APP Iphone - Xcode Version 12.4 (12D4e) Iphone XR
- Emulador APP Android - Open Android Emulator Pixel 2 API 26

## 3.2 Aquisição de Imagens

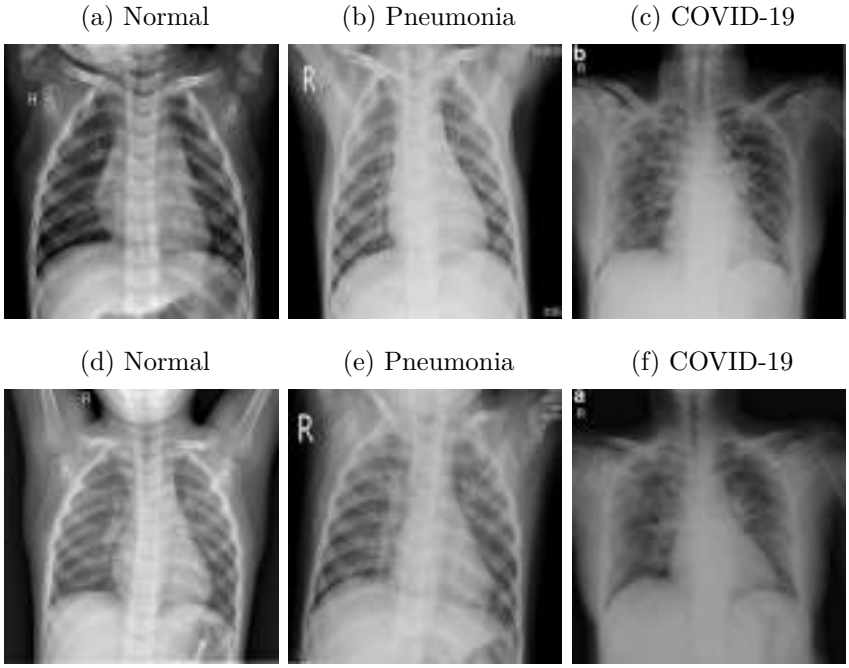
No processo de obtenção de imagens para treinamento e teste das arquiteturas foi realizada a consolidação de duas bases disponibilizadas na plataforma Kaggle de forma gratuita. A primeira base foi consolidada por uma equipe de pesquisadores da Qatar University, Doha, Qatar e da University of Dhaka, Bangladesh, juntamente com seus colaboradores do Paquistão e da Malásia, em colaboração com médicos, criaram um banco de dados de imagens de raios-X de tórax para casos COVID-19 positivos, juntamente com magens de pneumonia viral e normal (KAGGLE, 2021b). A segunda base Chest X-Ray Images (Pneumonia), foi analizada e disponibilizada por dois médicos especialistas para treinamento de sistemas baseados em IA e verificado por um terceiro especialista consiste em imagens de raios-X de tórax (ântero-posterior) que foram selecionadas de coortes retrospectivas de pacientes pediátricos do Guangzhou Women and Children's Medical Center, Guangzhou. Todas as radiografias de tórax foram realizadas como parte dos cuidados clínicos de rotina dos pacientes (KAGGLE, 2021a). Após juntar as bases, as imagens foram agrupadas conforme figura 15 em suas respectivas classes.

## 3.3 Pré-processamento

### 3.3.1 Preparação dos Dados

Nesse processo foi verificado extensão de todas as imagens, as imagens estão no formato de arquivo Portable Network Graphics (PNG) e Joint Photographic Experts Group (JPEG) e a resolução é de 1024 por 1024 pixels e 256 por 256 pixels, que podem ser facilmente convertidos para 224 por 224 ou 227 por 227 pixels normalmente exigido pelas Redes Neurais Convolucionais (Padrão de dimensionamento das imagem de entrada para o processamento dos filtros convolucionais das CNNs TensorFlow e keras). Nesse processo foi realizado o agrupamento das imagens em suas respectivas classes (COVID-19, Pneumonia Viral, Normal) para preparação do banco de imagens e realização de upload da base para o google Drive.

Figura 15 – Representação de 6 imagens de Raio-X e suas classificações




Fonte: (KAGGLE, 2021b) (KAGGLE, 2021a)

### 3.3.2 Seleção de Dados

Na seleção dos dados foi realizada a divisão das bases de treino e teste para cada categoria, essa seleção visa a organização das imagens para criação do data augmentation e por não terem indexação padronizada optou-se por segregação em pastas, durante a etapa de treinamento o data augmentation é realizado dinamicamente na base fazendo a validação cruzada e mantendo aleatoriedade para evitar overfitting.

Tabela 1 – Segmentação da base de imagens.

| | COVID-19 | Normal | Pneumonia | Total |
|---|---|---|---|---|
| Treino | 1.134 | 2.338 | 2.269 | 5741 |
| Teste | 284 | 585 | 568 | 1437 |
| Total | 1418 | 2923 | 2837 | **7178** |

Fonte: Autor

Após seleção e classificação foi realizado compressão e upload da base no google Drive, onde poderá ser importada facilmente pela plataforma do colab e consumida nas arquiteturas desenvolvidas.

## 3.4 Modelagem

Na inicialização da construção dos modelos primeiramente foi necessário importação da base de dados de imagens e extração das pastas compactadas. Após a extração é criada uma árvore de pastas com treinamento e teste conforme figura 17. O processo de extração do arquivo base_v2.zip está descrito no código 3.1, que realiza a descompactação do arquivo conforme localização no drive.

```
from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/My Drive/base_v2.zip"
zip_object = zipfile.ZipFile(file=path, mode="r")
zip_object.extractall("./")
zip_object.close()
```

Código 3.1 – Importação da base de imagens

Figura 17 – Árvore de Pastas Base da dados


Fonte: Autor

Uma vez montada a base está disponível para tratamento e modelagem, diante disso é necessário distribuição das bases de treino e teste fazendo o redimensionamento e atribuindo parâmetros para submissão do algoritmo de criação dos modelos. Através da técnica de Data Augmentation é possível a criação de novas imagens para aumentar a generalização diante o modelo.

O Data Augmentation consiste na geração de novas amostras do conjunto de dados com fins de aumentar o conjunto e/ou transformar amostras existentes, essa técnica se dedica a melhorar a precisão e estabilidade das classificações e tornar menos tendenciosas e invariantes na generalização dos modelos para novos conjuntos de dados (PEREZ; WANG, 2017) (WANG et al., 2018) (ZHANG et al., 2017). Realizou-se neste trabalho a atribuição de parâmetros para a transformação das dimensionalidades das imagens que simulassem possíveis entradas no modelo modelo de predição, é possível através do método conseguir a generalização e identificação de características em qualquer posicionamento da imagem.

No trabalho foi considerado a transformação da imagem em ângulo de 50 graus de rotação para direita e esquerda através do $rotation\_range$, mudança na largura e altura da imagem em 0.2 ou seja 20% da quantidade pixels da imagem na vertical e horizontal $width\_shift\_range$ e $height\_shift\_range$, é realizado também a captura de zoom em 0.1 ou 10% de foco na imagem, entre os parâmetros também é aceito o rotacionamento da imagem vertical e horizontal $horizontal\_flip$, $vertical\_flip$, parâmetros inseridos na funcão $train\_datagen$ que é a tranformação no formato do tensorflow.

Na função de geração de imagens de treino utiliza-se o $target\_size = 224x224$ no redimensionamento da imagem de entrada, $batch\_size = 32$ para determinar a frequência de registros para cálculo de erro e atualização dos pesos no treinamento da rede, utiliza-se o $class\_mode = categorical$ uma vez que estamos trabalhando com uma predição de 3 classes, na função é utilizado o $shuffle$ para embaralhamento dos registros durante geração das imagens para evitar overfitting.

Assim, com base na transferência de aprendizagem é criado a função base do modelo para treinamento, diante disso a função promove o download do imagenet e captura dos pesos das redes do keras, os parâmetros da função são: $weights = imagenet$ que é a referência da base do imagenet e o $include\_top = False$ que caracteriza o congelamento da CNN apenas para as convoluções deixando assim a camada totalmente conectada para implementação e posteriormente criação da camada de saída da rede com os resultados. Foram testadas 14 arquiteturas tabela 2 e implementadas as camadas densamente conectadas para tratar da classificação do problema em específico.

Tabela 2 – Arquiteturas CNNs Keras.

| Arquitetura | $n^{\circ}$ Layers |
|---|---|
| DenseNet121 | 426 |
| DenseNet169 | 594 |
| DenseNet201 | 706 |
| InceptionResNetV2 | 779 |
| InceptionV3 | 310 |
| MobileNetV2 | 153 |
| MobileNet | 85 |
| NASNetLarge | 1038 |
| NASNetMobile | 768 |
| ResNet50V2 | 189 |
| ResNet50 | 174 |
| VGG16 | 18 |
| VGG19 | 21 |
| Xception | 131 |

Fonte: Keras (2020)

No processo de criação da camada densamente conectada iniciou-se com o rece-

bimento do output da CNN baixada e realizado acoplando a uma camada de Average Pooling e duas camadas ocultas cada camada com 1024 neurônios e função de ativação ReLU e uma camada com 512 neurônios também com função de ativação ReLU. Por fim é criado a camada de saída com 3 neurônios e função de ativação $softmax$ para gerar a classificação, o esquema de estruturação na linguagem python está descrito no Código 3.2

```
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
x = tf.keras.layers.Dense(1024, activation='relu')(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
preds = tf.keras.layers.Dense(3, activation='softmax')(x)
model = tf.keras.Model(inputs = base_model.input, outputs = preds)
```

Código 3.2 – Camada Densamente Conectada

Após o acoplamento da camada totalmente conectada a rede neural foi necessário aplicar a técnica de "frozen", que consiste em congelar a CNN importada e disponibilizar somente a camada densamente conectada para recebimento do treinamento uma vez que a camada não possui os pesos gerados do imagenet, essa técnica é utilizada para melhorar o desempenho da arquitetura e prover saídas conforme necessidade de classificação. Exemplificando o procedimento na linguagem python o congelamento das 426 camadas iniciais da rede através do parâmetro $layer.trainable = False$ e disponibilização do restante da rede para treinamento $layer.trainable = True$ conforme descrito no Código 3.3

```
for layer in model.layers[:427]: #congelamento ate layer 426
  layer.trainable = False

 for layer in model.layers[427:]: #liberacao a partir da layer 427
  layer.trainable = True
```

Código 3.3 – Aplicação de Frozen na CNN DenseNet121

Definida a arquitetura da rede, é realizado a compilação do modelo, é necessário nessa fase de treinamento a realização da validação cruzada, opta-se por 5 folds e cada fold é processado em 10 épocas. A função de compilação do modelo é composto pelos hiperparâmetros $optimizer = Adam$ que é o otimizador da rede e o $loss = categorical\_crossentropy$ que calcula a perda de entropia cruzada entre os rótulos e as previsões e sua função é verificar o progresso do processo de aprendizagem, e junto ao compilador o hiperparâmetro $metrics$ que recebe um vetor das métricas de avaliação da compilação (Acurácia, precisão, recall e F1-Score).

Após compilação é realizado o $fit\_generator$ do modelo, é composto pelos hiperparâmetros $generator$ que recebe a base de treino, $epochs$ consiste na parametrização de 10

épocas de treinamento, o *steps_per_epoch* é a divisão da quantidade de registros da base de treino pela tamanho do *batch_size*, variável criada no geração da base de treino para compor a quantidade de blocos de treino, no *fit_generator* é realizado a validação dos dados através do *validation_data* que recebe a base de teste e o *step_size_test* composto pelo total de registros da base de teste. No passo seguinte é realizada a avaliação do treinamento com o método 'evaluate' que recebe a base de treino gerada para verificação das métricas da rede treinada. O código completo da implementação do modelo está disponível no apêndice B.

Em seguida foi testado modelo para verificação de eficácia, é realizado input de imagem externa às bases de teste para diagnóstico, no recebimento dessa imagem é feito a normalização da mesma e submetida ao processamento do modelo, assim a imagem é diagnosticada, o processo de entrada processamento e saída de resultados está descrito em python conforme Código 3.4

```
# Busca imagem no drive
image = tf.keras.preprocessing.image.load_img(r'path', target_size =
    (224,224))
# Preprocessamento da imagem e conversao.
image = tf.keras.preprocessing.image.img_to_array(image)
# Insercao de quarta parametro para se adequar a entrada na CNN
image = np.expand_dims(image, axis = 0)
# Aplicacao de preprocessamento para o tipo de rede
image = tf.keras.applications.densenet.preprocess_input(image)
# Realiza predicao da imagem no modelo
predictions = model.predict(image)
# Retorna array com valores de previsao para cada classe
predictions[0]
```

Código 3.4 – Verificação de Modelo

## 3.5 Implantação

Seguindo o processo de criação da aplicação é necessário a exportação do modelo criado para a aplicação móvel, com o modelo treinado e testado é possível a disponibilização para realizar predições dentro de qualquer aplicação com entradas de imagens do usuário para prever a probabilidade das classes no diagnóstico. Seguindo a ideia de consumo da aplicação, foi realizado duas exportações para atender o objetivo, a primeira, o modelo atende aos requisitos da linguagem dart, necessita de um arquivo .tflite para compatibilidade das entradas de dados e processamento da imagem, esse arquivo é requerido nos assets do projeto, ou seja, permite operação dentro da aplicação sem auxílio de internet, a segunda exportação foi realizada com o intuito de execução em um backend web, para esse processo optou-se por exportação do modelo no formato .hdf5 que trabalha em uma API web

para prover as probabilidades de classificação das imagens submetidas pelo usuário. A exportação dos modelos treinados está descrita na linguagem python conforme Código 3.5.

```
# salvar modelo
import pickle

# salvar o modelo no arquivo my_model.hdf5
tf.keras.models.save_model(model,'my_model.hdf5')

# salvar o modelo no arquivo my_model.tflite
model_save_path = "model_save_path/"
tf.saved_model.save(model, model_save_path)
converter = tf.lite.TFLiteConverter.from_saved_model(model_save_path)
tflite_model = converter.convert()
open("converted_model.tflite", "wb").write(tflite_model)
```

Código 3.5 – Conversão/Exportação do modelo treinado

## 3.6 Métricas de Avaliação

Em primeiro lugar, é necessário definir o significado dos verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. Os verdadeiros positivos e verdadeiros negativos são classificações corretas, ou seja, normal, pneumonia e COVID-19 classificados corretamente. Em contraste, falso positivo e falso negativo são classificações erradas. Dito isso, podemos definir as métricas.

O número total de testes das classes correspondem a linha da tabela verdadeiro positivo (TP) + falsos negativos (FN), o total de falsos negativos é a soma da linha da classe correspondente excluindo o valor de TP, o total de falsos positivos (FN) para uma classe é a soma da coluna correspondente excluindo o (TP), o total de verdadeiros negativos (TN) é a soma de todas as linhas e colunas da tabela excluindo a linha e coluna da classe correspondente. Podemos verificar o formato de classificação na tabela 3, que apresenta a estrutura das classes e seus respectivos valores.

Tabela 3 – Matriz de confusão Multiclasse - Acertos (TP) e Erros (E)

| Classe | A | B | C |
|---|---|---|---|
| A | $TP_A$ | $E_{AB}$ | $E_{AC}$ |
| B | $E_{BA}$ | $TP_B$ | $E_{BC}$ |
| C | $E_{CA}$ | $E_{CB}$ | $TP_C$ |

Fonte: Autor

A Equação 3.1 apresenta a primeira métrica chamada precisão. Essa métrica é a proporção entre verdadeiros positivos e verdadeiros positivos mais falsos positivos. Assim, uma baixa precisão indica que o número de classificações corretas é muito baixo ou o

número de falsos positivos também é alto. Nas equações 3.2, 3.3, 3.4 apresentam-se os valores de precisão para cada classe.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Precision_A = \frac{TP_A}{TP_A + E_{BA} + E_{CA}} \tag{3.2}$$

$$Precision_B = \frac{TP_B}{TP_B + E_{AB} + E_{CB}} \tag{3.3}$$

$$Precision_C = \frac{TP_C}{TP_C + E_{AC} + E_{BC}} \tag{3.4}$$

A próxima métrica é a acurácia, conforme apresentada na Equação 3.5, que é a porcentagem de classificações corretas. Uma baixa precisão pode indicar que o número de classificações incorretas (falsos positivos e falsos negativos) é alto. Na equação 3.6 apresenta-se o valor da acurácia com os valores de cada classe.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.5}$$

$$Accuracy = \frac{TP_A + TP_B + TP_C}{TP_A + E_{AB} + E_{AC} + TP_B + TP_B + E_{BA} + E_{BC} + TP_C + E_{CA} + E_{CB}} \tag{3.6}$$

O recall apresentado na Equação 3.7 é a razão entre os verdadeiros positivos e verdadeiros positivos mais os falsos positivos. Essa métrica indica que o algoritmo está tendo um bom desempenho na classificação de verdadeiros positivos. No entanto, se essa métrica for baixa, pode significar que um grande número de erros de classificação está acontecendo. Assim, essa métrica é essencial para minimizar o número de falsos negativos, que podem produzir o pior cenário para o paciente. Nas equações 3.8, 3.9, 3.10 apresentam-se os valores de reall para cada classe.

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

$$Recall_A = \frac{TP_A}{TP_A + E_{AB} + E_{AC}} \tag{3.8}$$

$$Recall_B = \frac{TP_B}{TP_B + E_{BA} + E_{BC}} \tag{3.9}$$

$$Recall_C = \frac{TP_C}{TP_C + E_{CA} + E_{CB}} \tag{3.10}$$

Finalmente, o F1-Score apresentado na Equação 3.11 é a média harmônica entre a precisão e o recall. Nesse contexto, F1 acaba sendo uma grande figura do desempenho porque a precisão leva em consideração falsos positivos e o recall leva em consideração falsos negativos. Assim, F1 Score dá uma ideia se o algoritmo de classificação está fornecendo muitas classificações incorretas. Nas equações 3.12, 3.13, 3.14 apresentam-se os valores de F1-Score para cada classe.

$$F1\_score = \frac{2 * precision * recall}{precision + recall} \tag{3.11}$$

$$F1\_score_A = \frac{2 * precision_A * recall_A}{precision_A + recall_A} \tag{3.12}$$

$$F1\_score_B = \frac{2 * precision_B * recall_B}{precision_B + recall_B} \tag{3.13}$$

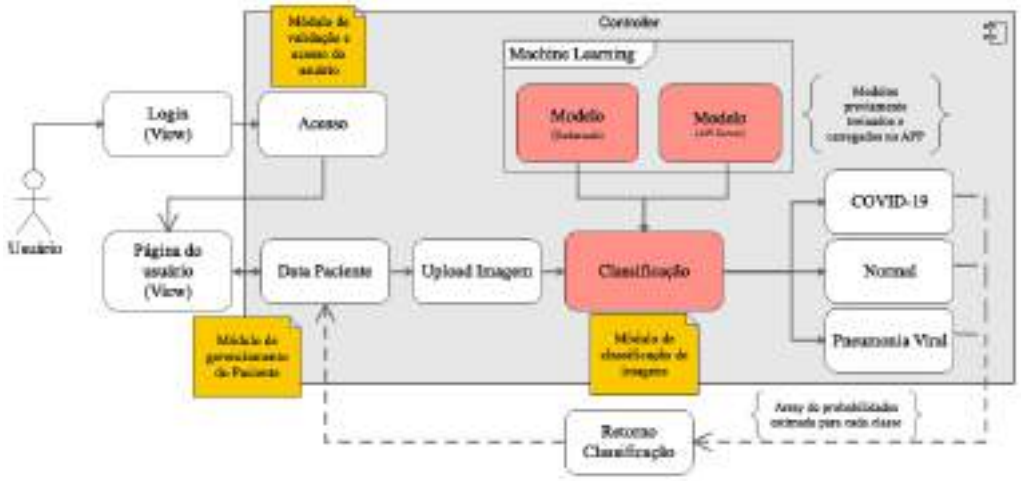
$$F1\_score_C = \frac{2 * precision_C * recall_C}{precision_C + recall_C} \tag{3.14}$$

## 3.7 Construção da Aplicação

No desenvolvimento da aplicação foram levantados requisitos e desenvolvido fluxos para entendimento da plataforma e seus recursos de usabilidade, os requisitos estão disponíveis no Apêndice A. O intuito do APP é fazer o registro de um usuário para ter acesso a base de pacientes que serão cadastrados e possam ter as imagens de suas respectivas radiografias analisadas pelo modelo, realizar acompanhamento e manter histórico.

Para estruturação do projeto foi elaborado arquitetura que atendesse e suportasse os módulos desenvolvidos, a aplicação possui três módulos principais: acesso, gerenciamento e classificação de imagens, através da aplicação é estruturado uma pagina incial para login, o módulo de acesso requer registro do usuário e a validação se dá por meio de registro ativo do CRM disponível em <https://www.consultacrm.com.br>, o sistema faz o rastreio para efetivar registro. O módulo de gerenciamento do paciente é composto por informações dos pacientes cadastrados bem como informações pessoais e imagens das radiografias submetidas para diagnóstico. No módulo de classificação a imagem é submetida para um dos módulos disponíveis na aplicação para prover as predições das imagens. Arquitetura da aplicação é apresentada na figura 18.

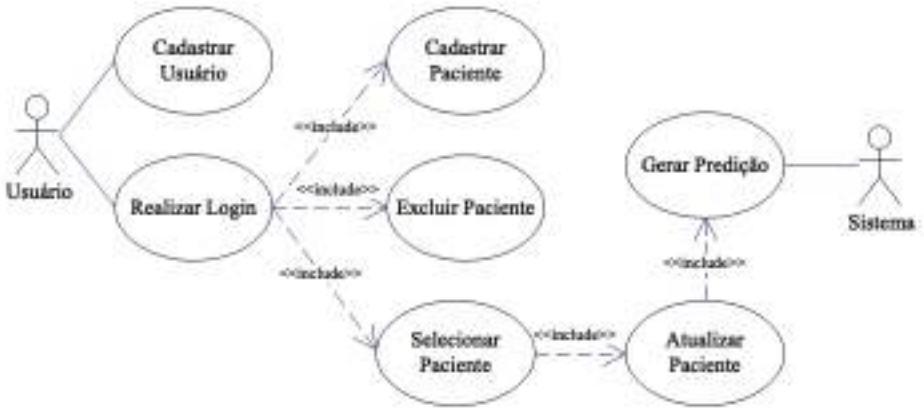
Figura 18 – Arquitetura da aplicação




Fonte: Autor

Assim como a arquitetura da aplicação também foi elaborado diagrama de caso de uso, consiste em representar uma unidade funcional disponível no sistema, mensagens que se relacionam com um ou mais atores do sistema, cada caso de uso é representado por uma elipse (VAZQUEZ; SIMõES, 2016). No sistema foram descritos sete casos de uso e dois atores (usuário e sistema), é importante ressaltar a relação de inclusão em alguns casos de uso, ou seja só podem ser executados caso haja ação correspondente. Entre os casos somente o 'gerar predição' é ação do sistema por se tratar do processamento interno. A figura 19 representa todos as ações dos atores.

Figura 19 – Diagrama de Caso de Uso

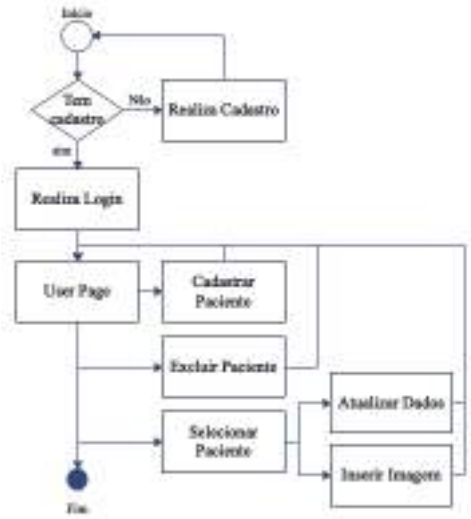



Fonte: Autor

O fluxograma de execução da aplicação é composto por entrada processamento e saída, o mesmo reflete o fluxo do processo das ações do usuário, inicialmente o usuário acessa a aplicação caso não haja registro o realiza, a partir de então tem todas as funcionalidades disponíveis para uso, Na tela inicial pode realizar o cadastro, exclusão, ou seleção do

paciente para realizar atualização e/ou fazer predição de imagem, fluxo da aplicação disponível na figura 20.

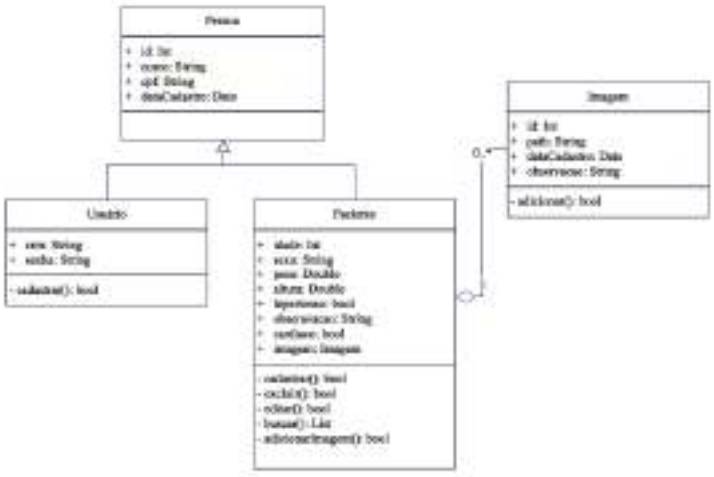
Figura 20 – Fluxograma do APP




Fonte: Autor

Na estruturação das classes pertencentes a aplicação foi criado o diagrama de classes que abstraem o conceito de orientação a objetos proposta na linguagem dart para construção das instâncias necessárias na aplicação. No diagrama foram contempladas quatro classes, a classe pessoa é uma generalização dos atributos comuns entre as classes usuário e paciente, em seguida é construída a classe imagem que é agregação da classe paciente. Cada classe possui seus métodos que são implementados na construção da aplicação. O diagrama está descrito na figura 21.

Figura 21 – Diagrama de Classes do APP




Fonte: Autor

Através do flutter e linguagem dart foi realizado o desenvolvimento da aplicação móvel que possui conforme estrutura apresentada na figura 18 dois ambientes de execução dos modelos (online e offline), os modelos treinados foram exportados e embarcados no APP (offline) e API web (online). Como base de dados foi utilizado o firebase, base de dados orientada a documentos.

O ambiente de desenvolvimento e os métodos no flutter são baseados em widgets, o que pode ser considerado um fator relevante para quem preza pela organização do código e facilita na construção de aplicações estáveis e de interface agradável. É importante ressaltar a utilização dos pacotes de dependências para construção da aplicação, essas dependências possuem métodos que tornam os recursos acessíveis e completos, os pacotes importados conforme figura 22 são necessários para a implementação, destacando as dependências do *http* que possuem métodos de conexão com APIs, o *Dio* tem métodos que facilitam o envio de imagem para a API desenvolvida no uso do modelo online, e o $tflite$ que permite acesso ao modelo embarcado (offline).

Figura 22 – Dependências Flutter para construção da aplicação

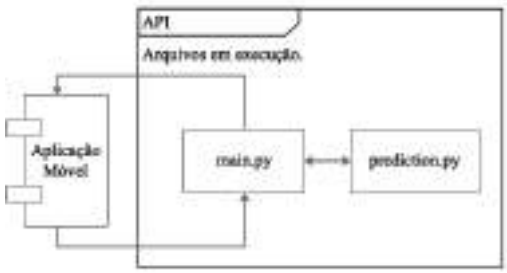
```
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.0
  flutter_staggered_grid_view: ^0.3.0
  cloud_firestore: ^0.12.9
  carousel_pro: ^1.0.0
  transparent_image: ^1.0.0
  scoped_model: ^1.0.1
  firebase_auth:
  firebase_core:
  url_launcher: ^5.1.1
  intl:
  tflite: ^1.1.1
  image_picker: ^0.6.7
  percent_indicator: ^2.1.1+1
  share: ^0.6.1+1
  http:
  dio: ^3.0.10
```

Fonte: Autor

Na construção da API web foram desenvolvidos dois arquivos (main.py e prediction.py) para fazer a interface com o APP na predição das imagens submetidas. Nessa construção foi utilizado o framework FastAPI que possui recursos para construção de backend para execução do modelo. No arquivo main.py são criados algumas rotas de teste e a rota principal POST, nesse arquivo é realizado o recebimento da imagem vinda da aplicação móvel (upload da imagem), nessa etapa o modelo é carregado no backend e enviado com a imagem através da função *predict_api*, ela faz interface com o arquivo prediction.py, o arquivo faz a normalização e submete a imagem ao modelo fazendo com que o mesmo gere a predição e devolva resultado para a função main, em seguida a função main

constrói o objeto com valores para aplicação móvel. O fluxo de entrada, processamento e saída é mostrado na figura 23, a codificação dos dois arquivos estão disponíveis no apêndice B

Figura 23 – Processamento de imagem FastAPI




Fonte: Autor

## 3.8 Considerações Finais do Capítulo

O capítulo mostrou como foi desenvolvida a construção dos modelos, apresentou as configurações e ambientes em que o trabalho foi criado, o processo de aquisição da base de imagens bem como sua etapas de pré processamento e seleção de dados, também foi mostrado o processo de criação da modelagem das CNNs até o processo de exportação do modelo treinado, através das métricas de avaliação foi possível mostrar como será validado o desempenho dos modelos. Também foi mostrado a estruturação da aplicação móvel através de arquitetura criada, fluxograma de execução da aplicação e diagramas de caso de uso e classe, por fim foram mostrados os complementos do flutter para desenvolvimento e o processo de criação da API web para processamento das imagens no modelo backend. A codificação completa do projeto está descrita no apêndice B.

# 4 Resultados e Discussão

Neste capítulo serão apresentados os resultados das principais etapas que constituíram a produção do trabalho, conforme apresentado no capítulo anterior, serão mostrados os produtos da criação dos modelos CNN, API para execução do modelo em um servidor e a aplicação móvel como produto final integrando todas as etapas.

## 4.1 Arquiteturas e Modelo

A Tabela 4 mostra os melhores resultados das quatorze CNNs de acordo com as métricas apresentadas anteriormente. Na mesma tabela, apresenta-se também a média da precisão e a média da perda durante o treinamento e validação das CNNs. Como podemos ver, o VGG16 atinge a melhor os melhores resultados em todas as métricas de avaliação acurácia = 96.52%, precisão = 96.86%, recall = 97.12% e F1Score = 96.99%, ainda possui o menor loss médio entre os 5 folds de todas as arquiteturas na etapa de validação 13.03%, todas as arquiteturas treinadas e validadas com os métodos adotados obtiveram resultados satisfatórios acima de 90% em todas as métricas, com exceção da Resnet50-V2 que obteve resultados medianos não ultrapassando os 80% em acurácia, precisão, recall e F1-Score.

Tabela 4 – Média dos resultados (folds) nas quatorze CNNs: Acurácia, precisão, recall, F1-Score, e Loss.

| Modelo | Acurácia | Precisão | Recall | F1-Score | Loss |
|---|---|---|---|---|---|
| VGG16 | **0.9652** | **0.9686** | **0.9712** | **0.9699** | **0.1303** |
| DenseNet201 | 0.9617 | 0.9669 | 0.9646 | 0.9657 | 0.1465 |
| Resnet50 | 0.9603 | 0.9645 | 0.9660 | 0.9652 | 0.1413 |
| Mobilenet-V2 | 0.9575 | 0.9620 | 0.9654 | 0.9632 | 0.1463 |
| Mobilenet | 0.9561 | 0.9615 | 0.9634 | 0.9617 | 0.1367 |
| DenseNet169 | 0.9540 | 0.9612 | 0.9576 | 0.9593 | 0.1437 |
| VGG19 | 0.9526 | 0.9568 | 0.9619 | 0.9589 | 0.1668 |
| DenseNet121 | 0.9498 | 0.9559 | 0.9566 | 0.9562 | 0.1638 |
| Nasnet-Mobile | 0.9380 | 0.9431 | 0.9449 | 0.9440 | 0.1904 |
| Xception | 0.9373 | 0.9439 | 0.9415 | 0.9413 | 0.1920 |
| Inception-Resnet-V2 | 0.9304 | 0.9351 | 0.9444 | 0.9382 | 0.2084 |
| Inception-V3 | 0.9290 | 0.9286 | 0.9393 | 0.9335 | 0.2336 |
| Nasnet-Large | 0.9255 | 0.9311 | 0.9313 | 0.9302 | 0.2060 |
| Resnet50-V2 | 0.7710 | 0.7902 | 0.8010 | 0.7834 | 0.5799 |

Fonte: Autor

Considerando que a VGG16 apresentou os melhores resultados, a Tabela 5 mostra sua matriz de confusão, na qual podemos observar que a VGG16 apresentou registros

incorretos em 50 casos, dentre os casos 49 foram classificados incorretamente em relação ao diagnóstico de pneumonia viral, 49 casos podem afetar os resultados do Recall porque podem ser considerados falsos negativos para pneumonia viral. Por outro lado, o fato notável é que apenas um caso de COVID-19 foi erroneamente classificado como NORMAL. Por outro lado, três casos de COVID-19 foram classificados como pneumonia viral.

Tabela 5 – Matriz de confusão - VGG16.

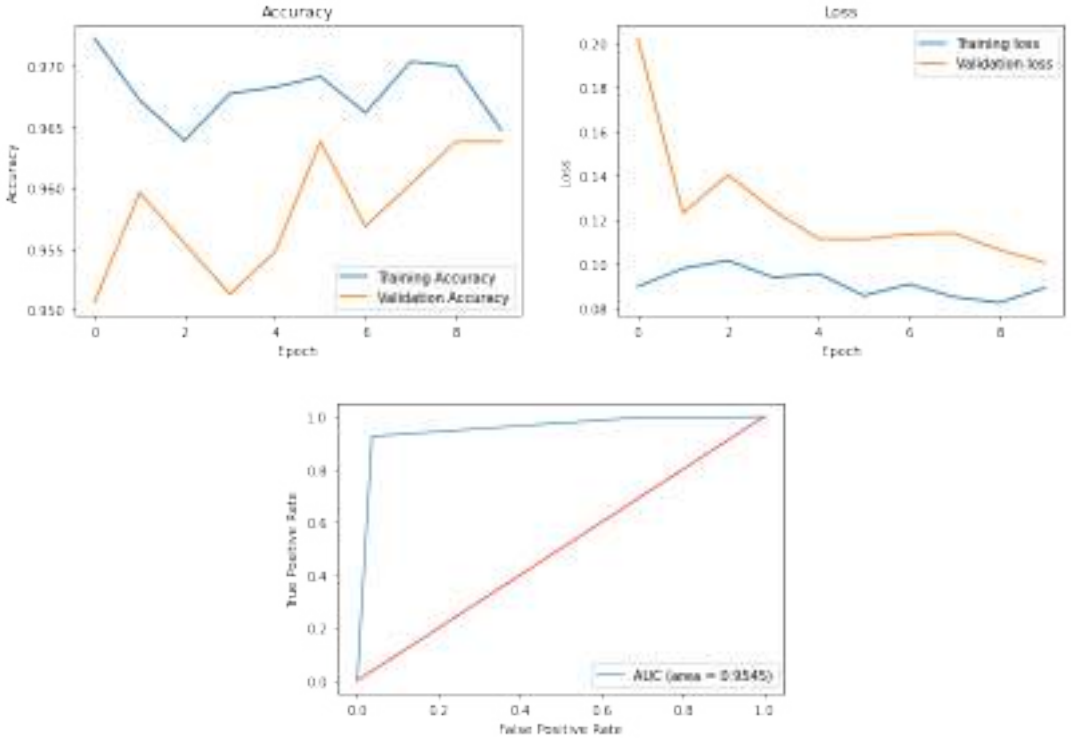
| | COVID-19 | Normal | Pneumonia |
|---|---|---|---|
| COVID-19 | 280 | 0 | 0 |
| Normal | 1 | 565 | 26 |
| Pneumonia | 3 | 20 | 542 |

Fonte: Autor

Além disso, as curvas de acurácia e perda apresentaram o comportamento esperado indo em direção a um na precisão e em direção a zero no loss. Em outras palavras, a precisão tendeu para 100% assim como o loss tendeu para 0%. De modo geral a frequência do classificador se mostrou bem assertivo como mostra a acurácia obtida, e das amostras supervisionadas o modelo obteve uma boa adequação e assertividade às classes efetivas correlacionadas na validação, para o recall métrica importante para o modelo, foi verificado que para cada classe a frequência de classificação correlata obteve ótimo percentual. Os valores de precisão e recall obtiveram bom equilíbrio uma vez que é natural a dificuldade de balancear a rigidez de acerto (melhorar a precisão) com menos propensão ao erro (aumentar o recall), valores esses que refletem no F1-Score, métrica capaz de qualificar se o modelo trabalha bem com conjuntos de dados com classes desproporcionais. Os valores representados na curva ROC mostra o quão assertivo o modelo foi capaz de distinguir as classes e representa a taxa de verdadeiros positivos pela taxa de falsos positivos, ou seja, a medida em que se aumenta a sensibilidade, reduz-se a especificidade. O AUC fornece o valor da área que tem proximidade a 1, ou seja, apresenta uma boa medida de separabilidade das classes em geral. Resultados expressos nos gráficos da figura 24

Assim como foi apresentado o melhor resultado das redes (VGG16) é mostrado também o resultado menos promissor Resnet50-V2 a matriz de confusão é apresenada na tabela 6. De modo geral a frequência do classificador se mostrou bem abaixo da na assertivo com relação aos demais modelos como mostram os resultados das métricas, quanto a acurácia obteve 77,10%, precisão = 79,02%, Recall = 80,01%, F1_Score = 78,34% e loss médio de 57,9%. Dos 329 casos classificados erroneamente, 226 foram falsos positivos da classe “normal“, 21 casos (falso positivo) apontados como COVID-19 e 205 casos (falso positivo) como Pneumonia Viral, os resultados foram bastante divergentes da versão do Resnet50 que obteve a terceira melhor classificação entre as redes uma das principais mudanças entre as redes é a forma de uso de pré ativação de camadas de peso em vez

Figura 24 – Métricas VGG16




Fonte: Autor

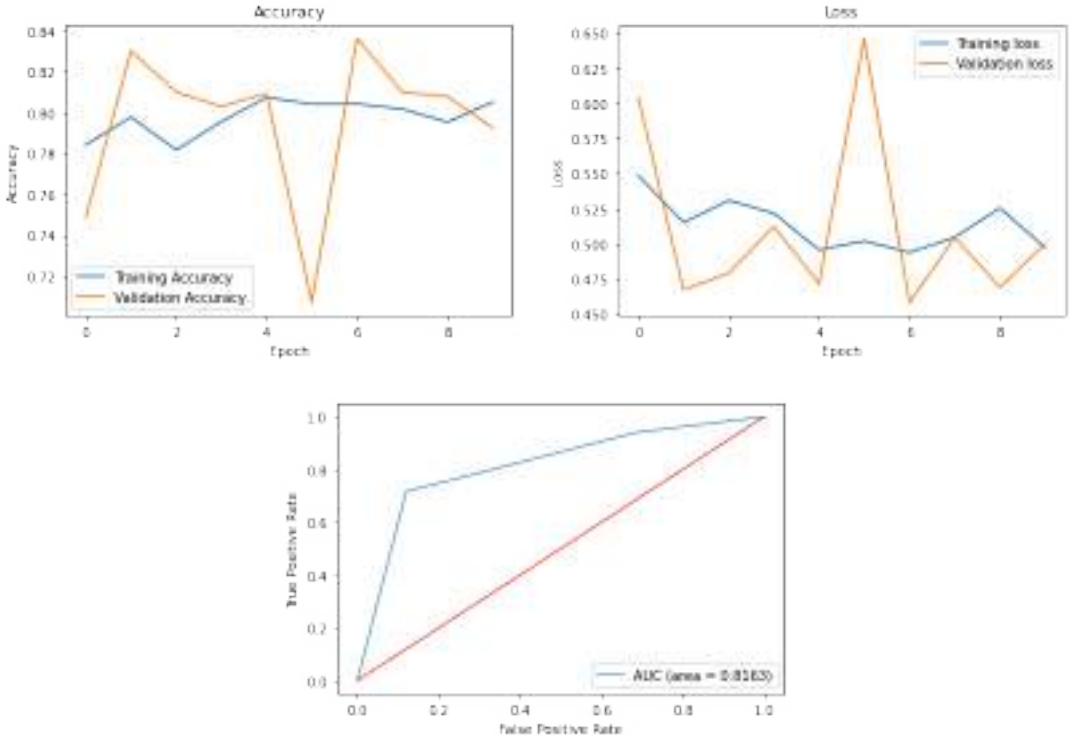
de pós ativação. Os valores da curva ROC AUC assumiram menores na assertividade do modelo conforme figura 26.

Tabela 6 – Matriz de confusão - Resnet50-V2.

| | COVID-19 | Normal | Pneumonia |
|---|---|---|---|
| COVID-19 | 249 | 21 | 26 |
| Normal | 7 | 359 | 42 |
| Pneumonia | 28 | 205 | 500 |

Fonte: Autor

Figura 26 – Métricas Resnet50-V2




Fonte: Autor

## 4.2 API

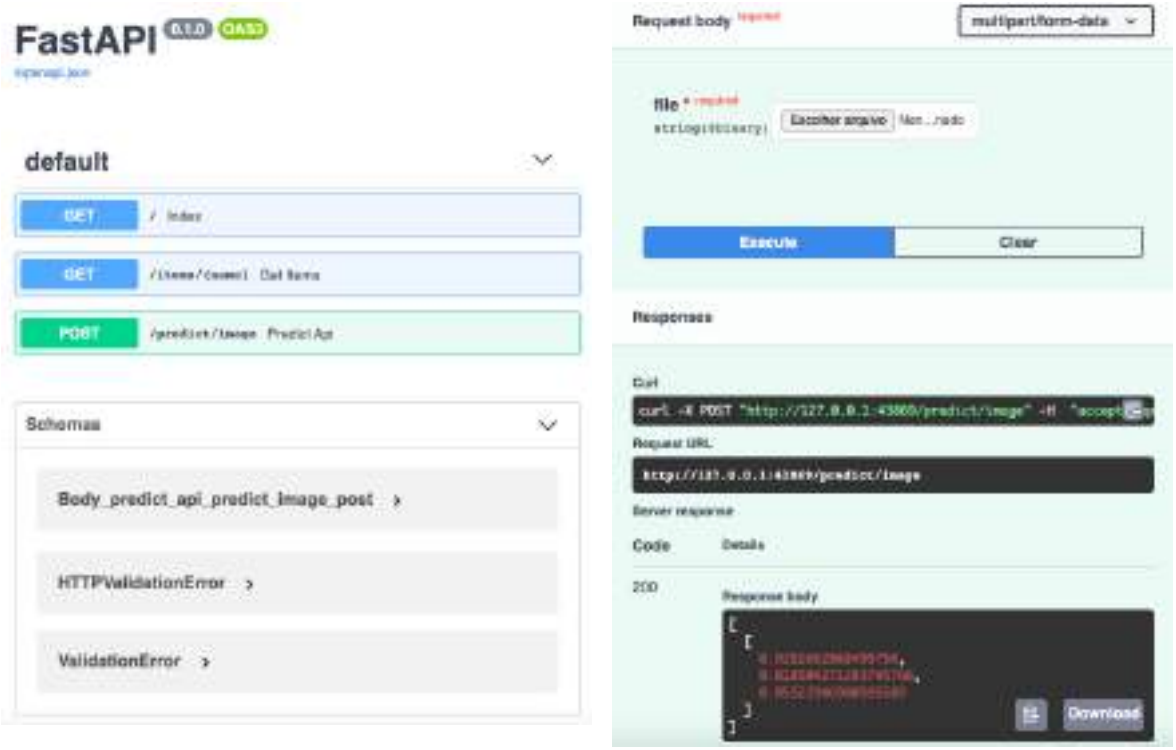
Uma das abordagens do trabalho foi de exportação do modelo treinado de melhor resultado para a execução no aplicativo por meio de uma API web, com o objetivo de classificar caso haja conexão de internet. O framework FastAPI foi implementado na solução, ela possui alto desempenho, fácil implementação e codificação e facilita o deploy para produção. O framework é utilizado em grandes empresas, como Microsoft, Netflix e Uber (FASTAPI, 2021). A necessidade de criação da interface obedece a realidade dos apps atuais, grandes volumes de dados e ansiedade por informações e processamentos rápidos requerem apps leves e de execução "clean" para atender as expectativas de usabilidade dos portadores.

A API desenvolvida possui um método POST para recebimento da imagem por meio do APP, a imagem é processada e submetida para avaliação no modelo, em seguida é dada uma resposta para o app com o retorno das previsões da cada classe e suas respectivas probabilidades de classificação.

É de extrema relevância a criação da interface pois se adequa a necessidade de uso para cada plataforma, inclusive é possível ser consumida por sistemas web, ambientes

tipo Desktop e aplicações móveis como é o caso do trabalho apresentado. Uma vez disponibilizada ficará apta a receber a imagem, fazer o processamento e responder com as probabilidades de classificação para os três tipos de classe: (COVID-19, Normal, Pneumonia Viral). A interface inicialmente conta somente com o recebimento da imagem, podendo assim ser implementado novas funcionalidades para a mesma futuramente, a estrutura web é apresentada na figura 28, também é possível a consulta ao código completo da criação e processamento da imagem para resultados da API no apêndice B

Figura 28 – FastAPI desenvolvida




Fonte: Autor

## 4.3 APP

Como produto final do trabalho conforme especificação da secão 3.7, foi desenvolvido APP para gerenciamento de pacientes e suas respectivas imagens radiográficas pulmonar, mantendo o histórico para acompanhamento e avaliação. O APP desenvolvido com auxílio do framework Flutter e linguagem dart foi possível execução tanto em ambientes Android e IOS. Conforme estrutura apresentada no capítulo citado, foi realizado a construção do APP com as seguintes telas:

- **Início:** Tela de início da aplicação, o usuário verá tela inicial que acompanha menu lateral conforme figura 30.

Figura 30 – Tela de início

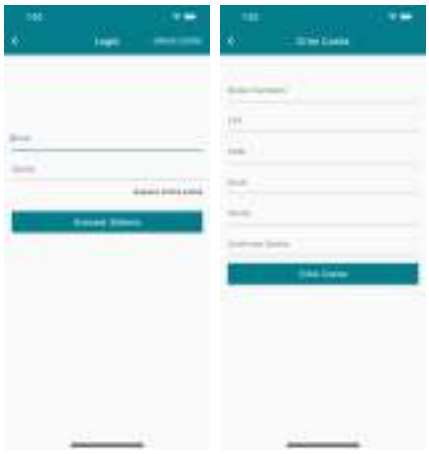
- **Barra Lateral:** Drawer com opções de identificação do usuário, cadastro, acesso e consulta de pacientes cadastrados, nesse menu é possível o usuário acessar tela de login, cadastro de novo usuário ou fazer logof da aplicação. Menu conforme figura 31;

Figura 31 – Barra lateral


- **Cadastro de Usuário:** Tela de login e cadastro do usuário, esse esquema apresenta as telas de login caso o usuário já tenha cadastro ou poderá acessar o botão ‘criar

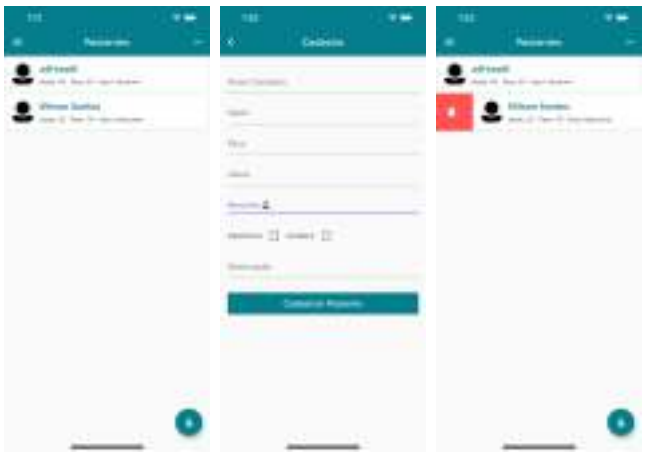
conta‘. Caso o usuário tenha esquecido a senha é possível recuperação através do email cadastrado, telas conforme figura 32;

Figura 32 – Acesso a aplicação



- **Gerenciamento de Paciente:** Tela de listagem de pacientes cadastrados, cadastro de novo paciente e exclusão de paciente, assim que o usuário realiza o login o menu lateral habilita a opção de paciente para consulta, nesse botão, o usuário acessa a lista de pacientes cadastrados, podendo assim criar novos registros ou excluir registro existente conforme figura 34;

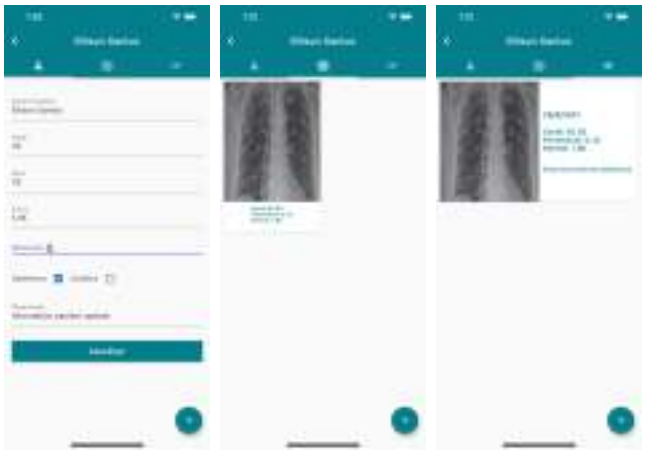
Figura 34 – Gerenciamento de Paciente



- **Acompanhamento Paciente:** Tela de registro de informações do paciente e modos de visualizacão de imagem, no ato da seleção do paciente na lista é carregado tela

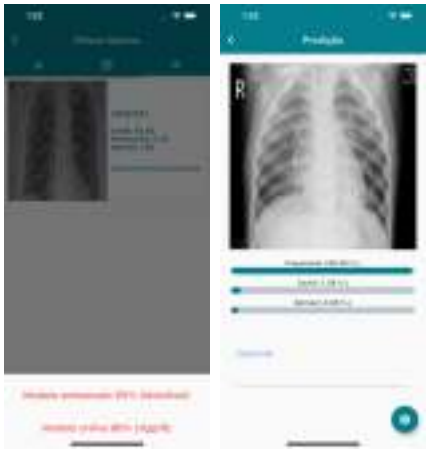
de atualização das informações do paciente bem como as imagens já cadastradas, a visualização pode ser em lista ou grade, a visualização dos dados do paciente estão nas telas conforme figura 36

Figura 36 – Visualização informações do Paciente



- **Predição de Nova Imagem:** Tela de escolha de tipo de predição e tela de informações da predição, nesse módulo é possível escolher o modo de predição online ou embarcado e assim gerar tela com resultados das probabilidades de classificação conforme figura 38.

Figura 38 – Visualização de Predição



- **Validação da imagem:** Para a correta verificação da imagem de raio X, foi criado um modelo auxiliar para determinar a sua validade, ou seja, o modelo reconhece antes de processar a previsão da imagem carregada é, de facto, uma imagem de

raio X. Foi criado um modelo em CNN com uma arquitetura baseada na rede Mobilinet, essa rede foi treinada com 340 imagens divididas em (raio-x e não raio-x), utilizando 50 épocas com batch size de 32 e learning rate de 0,001 com o objetivo de diferenciar imagens radiográficas de outras qualquer, esse modelo após treinamento é realizado o deploy e embarcado na aplicação e é executado ao carregar a imagem para diagnóstico, esse método verifica se de fato é uma imagem radiográfica. O modelo obteve 95,4% de precisão para distinguir entre duas classes (raio-x e não raio-x). Para validação optou-se por uma acurácia acima de 80% para classificar a imagem como raio-X, a imagem sendo confirmada como raio-X é enviada para os modelos de previsão de doenças pulmonares, caso contrário é informado ao usuário da aplicação que a imagem submetida não é identificada como uma radiografia torácica. conforme figura 40.

Figura 40 – Validação da imagem de raio X

## 4.4 Considerações Finais do Capítulo

Conforme abordagem do capítulo foram apresentados os desempenhos dos modelos treinados e destacado as redes com melhor desempenho e a rede com desempenho menos promissor, bem como suas matrizes de confusão e métricas, também é mostrado a construção e produção de uma das abordagem de processamento do modelo em uma API web para recepção de imagens e realização de predição, por fim é mostrado o produto final (aplicação móvel) representado nas telas e suas funcionalidades. Em seguida veremos o capítulo com a conclusão e trabalhos futuros.

# 5 Conclusão e Trabalhos Futuros

Este trabalho apresenta a implementação e teste de quatorze CNNs utilizando aprendizado por transferência na classificação de infecções pulmonares com base em raios-x, particularmente COVID-19 e pneumonia viral. Também é proposto uma aplicação móvel embarcada com modelo treinado em dois esquemas, online através de API web e offline no próprio APP, com o objetivo de auxiliar profissionais de saúde no diagnóstico de doenças pulmonares.

Na primeira parte do trabalho os resultados mostraram que a VGG16 melhores métricas de avaliação. A notável atuação da referida CNN foi a de classificar corretamente mais casos do que as demais CNNs na fase de teste dos folds. As técnicas de Data Augmentation e transfer learning aplicadas às arquiteturas de CNN obtiveram excelentes resultados de reconhecimento de padrões nas imagens de raio-X. Com o auxílio do transfer learning modificando apenas as estruturas de camadas finais para aplicar as características extraídas do problema é bem mais viável em termos de desempenho que tentar criar uma rede neural robusta totalmente original, e enfrentar todas as dificuldades de tempo e requisitos de capacidade computacional para treinar uma rede sobre um banco de dados do tamanho do ImageNet.

Entre as redes que mais se destacaram nas métricas de avaliação do aprendizado e teste estão o VGG, DenseNet e Mobilinet dentro de suas versões, a diferença entre as métricas são bem sutis, uma vez que o VGG16 se destaca por ser uma rede de arquitetura leve, mas que consegue se adaptar para processamento de imagens em larga escala, ela utiliza apenas kernels 3x3 em cada camada de convolução para realizar as operações.

Os resultados demonstram um caminho interessante para a obtenção de modelos capazes de inferir diagnósticos para COVID-19 e pneumonia viral a partir de imagens de radiografias torácicas. Outra constatação é que as CNNs de uma certa forma se tornam especialistas em diagnóstico dependendo da arquitetura, tipo e quantidade de imagens de treino.

A criação da API que executa o modelo em backend traz praticidade e dinamismo à aplicação, com ela é verificado níveis de segurança de acesso, agilidade e compatibilidade na integração entre sistemas e aplicações, permite a possibilidade de inovação e tratamento dos dados assim como armazenamento de informações. A importância das APIs nos sistemas e aplicações da atualidade são essenciais para o desempenho e processamento dos dados.

O melhor modelo (VGG16) foi exportado e executado no servidor web formato .htf5 e tamanho do arquivo de 61.4 MB, no APP foi embarcado o modelo MobileNet-V2 convertido para .tflite ,extensão de dependências do flutter e volume de 2,1 MB , que

deteve boas métricas de avaliação do modelo (95,57%, 96,20%, 96,54%, 96,32% e 14,13%) para acurácia, precisão, recall, F1-Score e Loss respectivamente. Os modelos obtiveram um desempenho excelente em diagnóstico e tempo de resposta na aplicação sem "Janks" travamentos e erros de execução, o que afeta diretamente a experiência do usuário no uso da aplicação.

O sucesso do desempenho da aplicação está diretamente ligado ao framework flutter, a aplicação mobile trabalha offline com modelo embarcado. O flutter possui grandes vantagens: velocidade e dinamismo, conversão em múltiplas plataformas, as aplicações são executados de forma rápida e suave e é perfeito para criação de apps básicos, complexos e MVPs.

## 5.1 Trabalhos Futuros

Para trabalhos futuros é necessário o teste com usuários reais para verificação de desempenho e receber o feedback para melhorias na aplicação, a aplicação poderá ser disponibilizada nas plataformas App Store para usuários IOS e na Google Play Store para usuários Android.

Futuramente poderá ser criado atualização de melhorias da resolução das imagens submetidas dentro da aplicação (modificação de filtros nas imagens carregadas, exemplo filtro do Instagram) uma vez que o fator luminosidade, contraste, brilho e outros atributos da imagem podem ajudar na predição mais assertiva dos dados. Com base nessa proposta poderá ser apresentado ao usuário as imagens processadas destacando os principais pontos identificados pelo algoritmo para determinar classificação.

Como a aplicação já faz o armazenamento das imagens e seus respectivos diagnósticos é possível a criação de mais uma rota na API para disponibilização das imagens para estudo e utilização pela comunidade.

Também é válida a ideia de gerar uma aplicação com base na que foi criada para qualquer tipo de predição feita por imagem, som, ou processamento de linguagem, fazendo o upload de qualquer modelo na própria aplicação e a configurando para gerar as devidas classificações.

# Referências

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; KUDLUR, M.; LEVENBERG, J.; MONGA, R.; MOORE, S.; MURRAY, D. G.; STEINER, B.; TUCKER, P. A.; VASUDEVAN, V.; WARDEN, P.; WICKE, M.; YU, Y.; ZHANG, X. Tensorflow: A system for large-scale machine learning. ***CoRR***, abs/1605.08695, 2016. Citado na página 38.

AMA. *American Medical Association Page.* Visit on 06-02-2021., 2016. Disponível em: <https://www.ama-assn.org/press-center/press-releases/survey-finds-physicians-enthusiastic-about-digital-health-innovation>. Citado na página 19.

ATHANAZIO, R. Airway disease: similarities and differences between asthma, copd and bronchiectasis. ***Clinics***, v. 67, p. 1335–1343, 2012. Citado na página 22.

BARROS, J. A. A.; VALLADARES, G.; FARIA, A. R.; FUGITA, E. M.; RUIZ, A. P.; VIANNA, A. A. G. D.; TREVISAN, G. L. A.-s.; OLIVEIRA, F. A.-c. A. M. d. Diagnóstico precoce do câncer de pulmão: o grande desafio. variáveis epidemiológicas e clínicas, estadiamento e tratamento. ***Jornal Brasileiro de Pneumologia***, v. 32, p. 221 – 227, 2006. Citado na página 19.

BECHTEL, M.; MCELLHINEY, E.; KIM, M.; YUN, H. Deeppicar: A low-cost deep neural network-based autonomous car. In: *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. [S.l.: s.n.], 2018. p. 11–21. Citado na página 28.

BEJNORDI, B. E.; VETA, M.; DIEST, P. J. V.; GINNEKEN, B. V.; KARSSEMEIJER, N.; LITJENS, G.; LAAK, J. A. W. M. V. D.; CONSORTIUM, T. C. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. ***JAMA***, v. 318, n. 22, p. 2199–2210, 2017. Citado na página 20.

BENGIO, Y.; COURVILLE, A. C.; VINCENT, P. Unsupervised feature learning and deep learning: A review and new perspectives. ***CoRR***, abs/1206.5538, 2012. Citado na página 28.

BERENGUER, J.; RYAN, P.; RODRíGUEZ-BAñO, J.; JARRíN, I.; CARRATALà, J.; PACHóN, J.; YLLESCAS, M.; ARRIBA, J. Baseline characteristics and outcomes of 1591 patients infected with sars-cov-2 admitted to icus of the lombardy region, italy. ***Clinical Microbiology and Infection***, v. 26, p. 1525–1536, 2020. Citado na página 27.

BEYSOLOW, T. I. **Introduction to Deep Learning Using R: a step-by-step guide to learning and implementing Deep Learning Models Using R**. In: . [S.l.]: Apress, 2017. p. 333. ASIN : B07447TH85. Citado 2 vezes nas páginas 38 e 39.

BEZDAN, T.; BACANIN, N. Convolutional neural network layers and architectures. In: . [S.l.: s.n.], 2019. p. 445–451. Citado na página 34.

CHEN, N.; ZHOU, M.; DONG, X.; QU, J.; GONG, F.; HAN, Y.; QIU, Y.; WANG, J.; LIU, Y.; WEI, Y.; XIA, J.; YU, T.; ZHANG, X.; ZHANG, L. Epidemiological and clinical characteristics of 99 cases of 2019 novel coronavirus pneumonia in wuhan, china: a descriptive study. ***Lancet***, v. 395, p. 507–513, 2020. Citado na página 27.

CHERIAN, T.; MULHOLLAND, E.; CARLIN, J.; OSTENSEN, H.; AMIN, R.; CAMPO, M. de; GREENBERG, D.; LAGOS, R.; LUCERO, M.; MADHI, S.; O'BRIEN, K.; OBARO, S.; STEINHOFF, M. Standardized interpretation of chest x-rays pediatric patients for the diagnosis of pneumonia in studies epidemiological. ***Bulletin of the World Health Organization***, v. 85, p. 353–359, 2005. Citado na página 25.

CHOLLET, F. In: _____. *EDeep Learning with Python.* Chollet: black white, 2017. p. 384. Citado na página 37.

CONLIN, R.; ERICKSON, K.; ABBATE, J.; KOLEMEN, E. Keras2c: A library for converting keras neural networks to real-time compatible c. ***Engineering Applications of Artificial Intelligence***, v. 100, p. 104182, 2021. Citado na página 38.

CRISP-DM. Cross industry standard process for data mining. Visit on 13-02-2021., 2000. Disponível em: <http://www.crisp-dm.org/download.htm>. Citado na página 22.

DA ROSA, M. Redes neurais convolutivas aplicadas à detecção de ervas daninhas. ***Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Rio Grande do Sul, Porto Alegre***, 2019. Citado na página 36.

DA SILVA COTRIM, W.; MINIM, V.; FELIX, L.; MINIM, L. Short convolutional neural networks applied to the recognition of the browning stages of bread crust. ***Journal of Food Engineering***, v. 277, p. 109916, 2020. Citado na página 34.

DA SILVA, R. E. V. Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens. ***Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Quixadá*** , 2018. Citado na página 36.

DART. *Page Language Dart.* Visit on 09-12-2020., 2020. Disponível em: <https://dart.dev/>. Citado 3 vezes nas páginas 40, 41 e 43.

DEEPLEARNINGBOOK. *Data Science Academy. Deep Learning Book - Capítulo 8 – Função de Ativação.* Visit on 25-03-2021., 2019. Disponível em: <www.deeplearningbook.com.br/funcao-de-ativacao/>. Citado 2 vezes nas páginas 30 e 31.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. ***In IEEE Computer Vision and Pattern Recognition***, p. 248–255, 2009. Citado na página 39.

DEZUBE, R. Exames de imagem do tórax. ***MANUAL MSD - Versão para Profissionais de Saúde***, Visit on 14-01-2021., 2019. Disponível em: <https://www.msdmanuals.com/>. Citado na página 19.

EISENBERG, R.; HEDGCOCK, M.; WILLIAMS, E.; LYDEN, B.; AKIN, J.; GOODING, G.; OVENFORS, C. Chest radiography in cardiovascular disease. ***AJR Am J Roentgenol***, v. 135, p. 1071–1074, 1980. Citado na página 26.

FASTAPI. *FastAPI Page.* Visit on 02-02-2021., 2021. Disponível em: <https://fastapi.tiangolo.com/>. Citado 2 vezes nas páginas 43 e 60.

FERNANDES, A. M. d. R. **Inteligência artificial: noções gerais**. In: *nteligência Articial Mito e Ciência*. [S.l.]: Florianópolis: Visual Books, 2003. p. 159. ISBN 8575021141. Citado na página 17.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. ***Ci. Inf., Brasília, Scielo***, v. 35, p. 25–30, 2006. Citado na página 17.

FIREBASE. *Firebase tools*. Acesso em on 28-03-2021., 2021. Disponível em: <https://firebase.google.com/docs>. Citado na página 41.

FLUTTER. *FLutter Page*. Visit on 10-06-2020., 2020. Disponível em: <https://flutter.dev>. Citado 3 vezes nas páginas 39, 40 e 43.

GAJARDO, A. Software de gestão para aplicativo educacional no ensino de programação para crianças. ***Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação***, Universidade de Caxias do Sul, Caxias do Sul, 2018. Citado na página 41.

GOOGLE. *Google Colaboratory*. Visit on 06-15-2020., 2020. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Citado na página 43.

GRASSELLI, G.; ZANGRILLO, A.; ZANELLA, A.; ANTONELLI, M.; CABRINI, L.; CASTELLI, A.; CEREDA, D.; COLUCCELLO, A.; FOTI, G.; FUMAGALLI, R.; IOTTI, G.; LATRONICO, N.; LORINI, L.; MERLER, S.; NATALINI, G.; PIATTI, A.; RANIERI, M. V.; SCANDROGLIO, A. M.; STORTI, E.; CECCONI, M.; PESENTI, A. Baseline characteristics and outcomes of 1591 patients infected with sars-cov-2 admitted to icus of the lombardy region, italy. ***JAMA***, v. 323, p. 1574–1581, 2020. Citado na página 27.

GUAN, W.-J.; NI, Z.-Y.; HU, Y.; LIANG, W.-H.; OU, C.-Q.; HE, J.-X.; LIU, L.; SHAN, H.; LEI, C.-L.; HUI, D. S. C.; DU, B.; LI, L.-J.; ZENG, G.; YUEN, K.-Y.; CHEN, R.-C.; TANG, C.-L.; WANG, T.; CHEN, P.-Y.; XIANG, J.; LI, S.-Y.; WANG, J.-L.; LIANG, Z.-J.; PENG, Y.-X.; WEI, L.; LIU, Y.; HU, Y.-H.; PENG, P.; WANG, J.-M.; LIU, J.-Y.; CHEN, Z.; LI, G.; ZHENG, Z.-J.; QIU, S.-Q.; LUO, J.; YE, C.-J.; ZHU, S.-Y.; ZHONG, N.-S. Clinical characteristics of coronavirus disease 2019 in china. ***Lancet***, v. 382, p. 1708–1720, 2020. Citado na página 27.

GULSHAN, V.; PENG, L.; CORAM, M.; STUMPE, M. C.; WU, D.; NARAYANASWAMY, A.; VENUGOPALAN, S.; WIDNER, K.; MADAMS, T.; CUADROS, J.; KIM, R.; RAMAN, R.; NELSON, P. C.; MEGA, J. L.; WEBSTER, D. R. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. ***JAMA***, v. 316, p. 2402–2410, 2016. Citado na página 20.

GURBETA, L.; BADNJEVIC, A.; MAKSIMOVIC, M.; OMANOVIC-MIKLICANIN, E.; E, S. A telehealth system for automated diagnosis of asthma and chronical obstructive pulmonary disease. ***Journal of the American Medical Informatics Association***, v. 25, p. 1213–1217, 2018. Citado na página 21.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. ***2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, p. 7110515, 2016. Citado 2 vezes nas páginas 35 e 36.

HIJAZI., S.; KUMAR., R.; ROWEN, C. Using convolutional neural networks for image recognition. ***Cadence***, p. 1–12, 2018. Citado na página 31.

HORRY, M. J.; CHAKRABORTY, S.; PAUL, M.; ULHAQ, A.; PRADHAN, B.; SAHA, M.; SHUKLA, N. Covid-19 detection through transfer learning using multimodal imaging data. ***IEEE Access***, v. 8, p. 149808–149824, 2020. Citado na página 21.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017. Citado 2 vezes nas páginas 36 e 37.

HUANG, G.; Z., L.; MAATEN, L.; WEINBERGER, Q. K. Densely connected convolutional networks. ***IEEE conference on computer vision and pattern recognition (CVPR)***, p. 2261–2269, 2017. Citado 2 vezes nas páginas 36 e 37.

IMAGENET. *ImageNet.* Visit on 05-31-2020., 2020. Disponível em: <http://www.image-net.org/>. Citado na página 39.

IMRAN, A.; POSOKHOVA, I.; QURESHI, H. N.; MASOOD, U.; RIAZ, M. S.; ALI, K.; JOHN, C. N.; HUSSAIN, M. I.; NABEEL, M. Ai4covid-19: Ai enabled preliminary diagnosis for covid-19 from cough samples via an app. ***Informatics in Medicine Unlocked***, v. 20, p. 100378, 2020. Citado na página 21.

ISMAIL, N. S.; SOVUTHY, C. **Breast Cancer Detection Based on Deep Learning Technique**. In: *2019 International UNIMAS STEM 12th Engineering Conference (EnCon)*. [S.l.: s.n.], 2019. p. 89–92. Citado na página 20.

JACOBI, A.; CHUNG, M.; BERNHEIM, A.; EBER, C. Portable chest x-ray in coronavirus disease-19 (covid-19): a pictorial review. ***Clin Imaging***, v. 64, p. 35–42, 2020. Citado na página 27.

KAGGLE. *Chest X-Ray Images (Pneumonia).* Visit on 26-03-2021., 2021. Disponível em: <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>. Citado 2 vezes nas páginas 44 e 45.

KAGGLE. *COVID-19 Radiography Database.* Visit on 26-03-2021., 2021. Disponível em: <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>. Citado 2 vezes nas páginas 44 e 45.

KAGGLE. *Kaggle Page.* Visit on 02-10-2021., 2021. Disponível em: <https://www.kaggle.com/>. Citado 3 vezes nas páginas 18, 26 e 33.

KAJEKAR, R. Enviromental factors and developmental outcomes in the lung. ***Pharmacol***, v. 114, p. 129–145, 2007. Citado na página 25.

KERAS. *Keras Page.* Visit on 06-15-2020., 2020. Disponível em: <https://keras.io/>. Citado 3 vezes nas páginas 18, 43 e 47.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. ***3rd International Conference for Learning Representations***, v. 9, p. 15, 2017. Citado na página 43.

KOTSUBO, K.; AZEVEDO, M. E. D. Estudo dosimétrico de radiografias de tórax com o emprego de técnicas de alta quilovoltagem. ***Radiologia Brasileira***, v. 36, 2003. Citado na página 26.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. ***Advances in neural information processing systems***, p. 1097–1105, 2012. Citado na página 33.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. ***Proceedings of the IEEE***, v. 86, p. 2278–2324, 1998. Citado 2 vezes nas páginas 20 e 33.

LIMA, A.; SALAME, M.; FILHO, F.; ATSROCH, A. Redes neurais convolucionais aplicadas ao processo de classificação de cultivares de guaranazeiros. ***XIV Encontro Nacional de Inteligência Artificial e Computacional***, 2017. Citado na página 28.

LIMA, C. M. A. d. O. Informações sobre o novo coronavírus (covid-19). ***Radiologia Brasileira***, v. 53, p. 5–6, 2020. Citado na página 27.

LIU, M.; GRANA, D. Accelerating geostatistical seismic inversion using tensorflow: A heterogeneous distributed deep learning framework. ***Computers Geosciences***, v. 124, p. 37–45, 2020. Citado na página 38.

MARQUES, J.; FALCAO, G.; ALEXANDRE, L. A. Distributed learning of cnns on heterogeneous cpu/gpu architectures. ***Applied Artificial Intelligence***, v. 32, p. 822–844, 2018. Citado na página 18.

MENEGOLA, A.; FORNACIALI, M.; PIRES, R.; BITTENCOURT, F. V.; AVILA, S. E. F. de; VALLE, E. Knowledge transfer for melanoma screening with deep learning. ***CoRR***, abs/1703.07479, 2017. Citado na página 18.

MíGUEZ, I. R. La pandemia y el sistema-mundo. ***la jornada***, p. 12, 2020. Citado na página 19.

NASIRI, A.; OMID, M.; TAHERI-GARAVAND, A. An automatic sorting system for unwashed eggs using deep learning. ***Journal of Food Engineering***, v. 283, p. 110036, 2020. Citado na página 34.

NERMINATHAN, A.; HARRISON, A.; PHELPS, M.; ALEXANDER, S.; SCOTT, K. Doctors' use of mobile devices in the clinical setting: a mixed methods study. ***National Center for Biotechnology Information***, v. 47, p. 291–298, 2017. Citado na página 19.

NHU, V.-H.; HOANG, N.-D.; NGUYEN, H.; NGO, P. T. T.; Thanh Bui, T.; HOA, P. V.; SAMUI, P.; Tien Bui, D. Effectiveness assessment of keras based deep learning with different robust optimization algorithms for shallow landslide susceptibility mapping at tropical area. ***CATENA***, v. 188, p. 104458, 2020. Citado na página 38.

OLIVEIRA, A. C. d.; LUCAS, T. C.; IQUIAPAZA, R. A. What has the covid-19 pandemic taught us about adopting preventive measures? ***Texto Contexto - Enfermagem***, v. 29, 2020. Citado na página 19.

PEREIRA, L. M. Inteligência articial mito e ciência. *Researchgate*, v. 1, p. 13, 2005. Citado na página 17.

PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. ***arXiv***, p. 1712.04621, 2017. Citado na página 46.

POURALIAKBAR, H. Chest radiography in cardiovascular disease. ***Practical Cardiology***, p. 113–130, 2005. Citado na página 26.

PRATT, L. Y.; MOSTOW, J.; KAMM, C. A. Direct transfer of learned information among neural networks. ***In Ninth National Conference on Artificial Intelligence***, p. 584–589, 1991. Citado na página 20.

PYTHON. *Python Page.* Visit on 06-15-2020., 2020. Disponível em: <https://www.python.org/>. Citado na página 43.

RAMOS-CERQUEIRA, A. T. de A.; CREPALDI, A. L. Qualidade de vida em doenças pulmonares crônicas: aspectos conceituais e metodológicos. *Jornal Brasileiro de Pneumologia*, v. 26, p. 207–213, 2000. Citado na página 25.

REZENDE, V. C. Metodologia para a classificação automática de doenças em plantas utilizando redes neurais convolucionais. ***Dissertação (Mestrado em Engenharia Elétrica) -Instituto de Tecnologia, Universidade Federal do Pará, Belém***, 2019. Citado na página 36.

RODRIGUES, D. A. Deep learning e redes neurais convolucionais: Reconhecimento automático de caracteres em plavas de licenciamento automotivo. ***Monografia***, 2018. Citado 3 vezes nas páginas 29, 31 e 32.

SCHNURR, H. P.; ARONSKY, D.; WENKE, D. Medicine 4.0: Interplay of intelligent systems and medical experts. ***Knowledge Management in Digital Change, Springer International Publishing***, p. 51–63, 2018. Citado na página 17.

SHEARER, C. The crisp-dm model: The new blueprint for data mining. ***Jorunal of Data Warehousing***, v. 5, p. 13–22, 2000. Citado na página 22.

SHI, F.; WANG, J.; SHI, J.; WU, Z.; WANG, Q.; TANG, Z.; HE, K.; SHI, Y.; SHEN, D. Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for covid-19. ***IEEE Reviews in Biomedical Engineering***, v. 14, p. 4–15, 2021. Citado na página 27.

SHI, H.; HAN, X.; JIANG, N.; CAO, Y.; ALWALID, O.; GU, J.; FAN, Y.; ZHENG, C. Radiological findings from 81 patients with covid-19 pneumonia in wuhan, china: a descriptive study. ***The Lancet Infectious Diseases***, v. 20, 2020. Citado na página 26.

SILVA, D.; CORTES, O. On convolutional neural networks and transfer learning for classifying breast cancer on histopathological images using gpu. ***In Congresso Brasileiro de Engenharia Biomédica***, p. 1–6, 2020. Citado 2 vezes nas páginas 28 e 39.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 33 e 35.

SONGHO. $convolution2d_example.Visiton24-03-2021., 2021.Disponívelem : <>$. Citado na página 30.

SOUTO, M. D.; LORENA, A.; DELBEM, A.; CARVALHO, A. de. Técnicas de aprendizado de máquina para problemas de biologia molecular. ***Sociedade Brasileira de Computaçao***, v. 1, 2003. Citado na página 28.

TOGAçAR, M.; ERGEN, B.; CöMERT, Z. Covid-19 detection using deep learning models to exploit social mimic optimization and structured chest x-ray images using fuzzy color and stacking approaches. ***Computers in Biology and Medicine***, v. 121, p. 103805, 2020. ISSN 0010-4825. Citado na página 36.

TORREY, L.; SHAVLIK, J. Transfer learning. ***In. Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques In E.S. Olivas, J.D.M. Guerrero, M.M. Sober, J.R.M. Benedito, and A.J.S. Lopez (eds.)***, p. 242–264, 2009. Citado na página 39.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. ***Proceedings of the XXIX Conference on Graphics, Patterns and Images***, p. 1–4, 2016. Citado na página 27.

VASCONCELLOS, G.; CLARêNCIO, J.; ANDRADE, D.; ARAúJO-NETO, C. A.; BARRAL, A.; NASCIMENTO-CARVALHO, C. M. Systemic cytokines / chemokines associated with radiographic abnormalities in pneumonia in children. ***Cytokine***, v. 135, p. 155191, 2020. Citado na página 25.

VAZQUEZ, C.; SIMõES, G. In: _____. *Engenharia de Requisitos: Software Orientado ao Negócio*. Vazquez: Editora Brasport Livros Técnicos e Digitais, 2016. p. 328. Citado na página 53.

WAHEED, A.; GOYAL, M.; GUPTA, D.; KHANNA, A.; AL-TURJMAN, F.; PINHEIRO, P. R. Covidgan: Data augmentation using auxiliary classifier gan for improved covid-19 detection. ***IEEE Access***, v. 8, p. 91916–91923, 2020. Citado na página 21.

WANG, F.; ZHONG, S.-h.; PENG, J.; JIANG, J.; LIU YAN", e. K.; CHALIDABHONGSE, T. H.; NGO, C. W.; ARAMVITH, S.; O'CONNOR, N. E.; HO, Y.-S.; GABBOUJ, M.; ELGAMMAL, A. Data augmentation for eeg-based emotion recognition with deep convolutional neural networks. In: *Data Augmentation for EEG-Based Emotion Recognition with Deep Convolutional Neural Networks*. [S.l.]: Springer International Publishing, 2018. p. 82–93. ISBN 978-3-319-73600-6. Citado na página 46.

WANG, N.; LIU, H.; XU, C. Deep learning for the detection of covid-19 using transfer learning and model integration. ***IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)***, p. 281–284, 2020. Citado na página 21.

WILSON, N.; KVALSVIG, A.; BARNARD, L.; BAKER, M. G. Case-fatality risk estimates for covid-19 calculated by using a lag time for fatality. emerging infectious diseases. ***Emerging Infectious Diseases***, v. 20(6), p. 1339–1441, 2020. Citado na página 18.

WOLF, B.; SCHOLZE, C. Medicine 4.0: The role of electronics, information technology and microsystems in modern medicine. ***Current Directions in Biomedical Engineering***, v. 3, p. 183–186, 2017. Citado na página 17.

YODA, T. The effect of collaborative relationship between medical doctors and engineers on the productivity of developing medical devices. ***R&D Management***, v. 46, p. 193–206, 2016. Citado na página 17.

YU, F.; LU, T.; HAN, B.; XUE, C. A quantitative study of aggregation behaviour and integrity of spray-dried microcapsules using three deep convolutional neural networks with transfer learning. ***Journal of Food Engineering***, v. 300, p. 770–778, 2021. Citado na página 34.

ZHANG, C.; BENGIO, S.; HARDT, M.; RECHT, B.; VINYALS, O. Understanding deep learning requires rethinking generalization. ***arXiv***, p. 1611.03530, 2017. Citado na página 46.

ZHOU, P.; YANG, X.; WANG, X.; HU, B.; ZHANG, L.; ZHANG, W.; SI, H.; ZHU, Y.; LI, B.; HUANG, C.; CHEN, H.; CHEN, J.; LUO, Y.; GUO, H.; JIANG, R.; LIU, M.; CHEN, Y.; SHEN, X.; WANG, X.; ZHENG, X.; ZHAO, K.; CHEN, Q.; DENG, F.; LIU, L.; YAN, B.; ZHAN, F.; WANG, Y.; XIAO, G.; SHI, Z. A pneumonia outbreak associated with a new coronavirus of probable bat origin. ***Nature***, v. 81, p. 18–19, 2020. Citado na página 27.

ZHU, N.; ZHANG, D.; WANG, W.; LI, X.; YANG, B.; SONG, J.; ZHAO, X.; HUANG, B.; SHI, W.; LU, R.; NIU, P.; ZHAN, F.; MA, X.; WANG, D.; XU, W.; WU, G.; GAO, G. F.; TAN, W. A novel coronavirus from patients with pneumonia in china, 2019. ***New England Journal of Medicine***, v. 382, p. 727–733, 2020. Citado na página 19.

ŠARIć, M.; RUSSO, M.; STELLA, M.; SIKORA, M. **CNN-based Method for Lung Cancer Detection in Whole Slide Histopathology Images**. In: *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*. [S.l.: s.n.], 2019. p. 1–4. Citado na página 20.

# Apêndices

# APÊNDICE A – Documento de Requisitos

## Introdução

Este documento especifica os requisitos do sistema do Preditor de Doença Pulmonar versão 1.0, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do sistema.

### Visão Geral do Documento

As seções seguintes estão organizadas como descrito abaixo:

- **Descrição geral do sistema:** apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e descrevendo funções do usuário.;
- **Requisitos funcionais (casos de uso):** especifica todos os casos de uso do sistema, descrevendo os fluxos de eventos, prioridades, atores, entradas e saídas de cada caso de uso a ser implementado.;
- **Requisitos não-funcionais:** especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.;
- **Referências:** apresenta referências para outros documentos utilizados para a confecção deste documento.

### Identificação dos Requisitos

Por convenção, a referência a requisitos é feita através do nome da subseção onde eles estão descritos, seguidos do identificador do requisito, de acordo com a especificação a seguir:

**[nome da subseção. identificador do requisito]**

Por exemplo, o requisito funcional [Recuperação de dados.RF016] deve estar descrito em uma subseção chamada “Recuperação de dados”, em um bloco identificado pelo número [RF016]. Já o requisito não-funcional [Confiabilidade.NF008] deve estar descrito na seção de requisitos não-funcionais de Confiabilidade, em um bloco identificado por [NF008].

Os requisitos devem ser identificados com um identificador único. A numeração inicia com o identificador [RF001] ou [NF001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos.

### Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos, nas seções 4 e 5, foram adotadas as denominações **"essencial", "importante" e "desejável"**.

- **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

## Descrição Geral do Sistema

### Abrangência e Sistemas Relacionados

A aplicação de previsão de doença pulmonar consistem em uma ferramenta com intuito de suporte para tomada de decisões e auxílio no diagnósticos de doenças pulmonares, através dessa aplicação o usuário submete a imagem de uma radiografia pulmonar para o app que realiza a classificação da imagem em três categorias a princípio: (COVID-19, pneumonia viral e normal).

## Requisitos Funcionais (Casos de Uso)

### [RF001] Cadastrar Usuário

**Descrição do caso de uso:** Permite no ato de entrada do app realizar cadastro de usuário para utilização do APP.

**Prioridade:** Essencial.

**Entradas e pré-condições:** Possuir registro de CRM Ativo.

**Saídas e pós-condição:** Usuário cadastrado na base de dados e com acesso à aplicação, permissão para fazer login.

## [RF002] Realiza Login

**Descrição do caso de uso:** Permite que o usuário acesse a aplicação.

**Prioridade:** Essencial.

**Entradas e pré-condições:** Cadastrar usuário [RF001].

**Saídas e pós-condição:** Verificação de usuário e entrada na aplicação.

## [RF003] Cadastrar Paciente

**Descrição do caso de uso:** Permite que o usuário realize o cadastro das informações do paciente para gerenciamento e acompanhamento.

**Prioridade:** Essencial.

**Entradas e pré-condições:** Realiza Login [RF002].

**Saídas e pós-condição:** Paciente cadastrado na base do usuário.

## [RF004] Excluir Paciente

**Descrição do caso de uso:** Permite que o usuário realize a exclusão de pacientes cadastrados.

**Prioridade:** Essencial.

**Entradas e pré-condições:** Realiza Login [RF002], Cadastrar Paciente [RF003].

**Saídas e pós-condição:** Paciente removido da base de dados.

## [RF005] Selecionar Paciente

**Descrição do caso de uso:** Permite seleção de paciente para realizar atualização de informações e inserir imagens.

**Prioridade:** Essencial.

**Entradas e pré-condições:** [RF002] Realiza Login, [RF003] Cadastrar Paciente.

**Saídas e pós-condição:** Paciente selecionado e informações disponíveis para visualização.

## [RF006] Atualizar Paciente

**Descrição do caso de uso:** Permite atualização de informações do paciente.

**Prioridade:** Essencial.

**Entradas e pré-condições:** Realiza Login [RF002], Cadastrar Paciente [RF003], Selecionar Paciente [RF005].

**Saídas e pós-condição:** Paciente com informações atualizadas.

### [RF007] Gerar Predição

**Descrição do caso de uso:** Permite submissão da imagem de raio-x do paciente para realizar predição pela aplicação.

**Prioridade:** Essencial.

**Entradas e pré-condições:** [Realiza Login [RF002], Cadastrar Paciente [RF003], Selecionar Paciente [RF005]

**Saídas e pós-condição:** Paciente selecionado e informações disponíveis para visualização.

## Requisitos não-funcionais

### [NF001] Usabilidade

A interface com o usuário é de vital importância para a experiência do usuário com o App. A aplicação deverá ser intuitiva possuir recursos simples e de fácil localização.

**Prioridade:** Essencial.

### [NF002] Desempenho

Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um fator de qualidade de software. A aplicação deverá conter modelo embarcado isento de conexão com internet para facilitar o desempenho e atender a necessidade de locais remotos que possuam apenas o uso de uma máquina de radiografia. Deverá ser desenvolvida em plataforma moderna que atenda principalmente os requisitos de experiência do usuário.

**Prioridade:** Importante.

### [NF003] Hardware e Software

Visando criar um produto com maior extensibilidade, reusabilidade e flexibilidade, deve ser adotar como linguagem principal de desenvolvimento Dart no framework Flutter modelagem dos widgets. Opta-se pelo embarque do modelo de predição no APP assim como construção de API web para processamento no backend. O produto final (sofware) estará nas versões para IOS e Android.

**Prioridade:** Importante.

## Referências

Furlan, J. D. Modelagem de Objetos através da UML. São Paulo, Makron Books, 1998.

Kruchten, P. The Rational Unified Process – An introduction. Addison-Wesley, 1998.

# APÊNDICE B – Codificação do Projeto

## Banco de Imagens

<https://drive.google.com/file/d/1u8FK0XtOnf-Ma49jHr-zxwfs4xB03sYr/view?usp=sharing>

## Modelo de Predição VGG16

```
!pip install keras-metrics
%tensorflow_version 2.x
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import  StratifiedKFold
from tensorflow.keras.metrics import *
import matplotlib.pyplot as plt
import seaborn as sns
import zipfile
import numpy as np
import keras
import keras_metrics as km
tf.__version__

from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/My Drive/base_v2.zip"
zip_object = zipfile.ZipFile(file=path, mode="r")
zip_object.extractall("./")
zip_object.close()

train_datagen = ImageDataGenerator(preprocessing_function=tf.keras.
    applications.vgg16.preprocess_input, rotation_range = 50,
    width_shift_range = 0.2, height_shift_range = 0.2, zoom_range = 0.1,
    horizontal_flip = True, vertical_flip = True)

train_generator = train_datagen.flow_from_directory('/content/base_v2/
    train', target_size = (224,224), batch_size = 32, class_mode = '
    categorical', shuffle = True, seed = 42)

step_size_train = train_generator.n // train_generator.batch_size

train_generator
```

```
30 for X, Y in train_generator:
31    break
32
33 test_datagen = ImageDataGenerator(preprocessing_function=tf.keras.
   applications.vgg16.preprocess_input)
34
35 test_generator = train_datagen.flow_from_directory('/content/base_v2/
   test', target_size = (224,224), batch_size = 1, class_mode = '
   categorical', shuffle = False)
36
37 step_size_test = test_generator.n // test_generator.batch_size
38
39 base_model = tf.keras.applications.VGG16(weights='imagenet', include_top
   =False)
40
41 x = base_model.output
42 x = tf.keras.layers.GlobalAveragePooling2D()(x)
43 x = tf.keras.layers.Dense(1024, activation='relu')(x)
44 x = tf.keras.layers.Dense(1024, activation='relu')(x)
45 x = tf.keras.layers.Dense(512, activation='relu')(x)
46 preds = tf.keras.layers.Dense(3, activation='softmax')(x)
47
48 model = tf.keras.Model(inputs = base_model.input, outputs = preds)
49
50 for i, layer in enumerate(model.layers):
51   print(i, layer.name)
52
53 for layer in model.layers[:19]:
54   layer.trainable = False
55
56 for layer in model.layers[19:]:
57   layer.trainable = True
58
59 # define 10-fold cross validation test harness
60 seed = 42
61 kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
62
63 for train, test in enumerate(kfold.split(X, Y.argmax(1))):
64     print("Fold", train+1)
65
66     # Compile model
67     model.compile(optimizer='Adam', loss='categorical_crossentropy',
   metrics=['accuracy',km.categorical_precision(), km.categorical_recall
   (), km.categorical_f1_score()])
68     # Fit generator the model
69     history = model.fit_generator(generator=train_generator,epochs = 10,
   steps_per_epoch = step_size_train,validation_data = test_generator,
```

```
    validation_steps = step_size_test)
70     # evaluate the model
71     scores = model.evaluate(train_generator, verbose=0)
72 
73     print((np.mean(history.history['val_loss'])))
74     print((np.mean(history.history['val_accuracy'])))
75     print((np.mean(history.history['val_precision'])))
76     print((np.mean(history.history['val_recall'])))
77     print((np.mean(history.history['val_f1_score'])))
78 
79 #testar predicao
80 predictions = model.predict_generator(test_generator, steps = len(
    filenames))
81 
82 predictions2 = []
83 for i in range(len(predictions)):
84   predictions2.append(np.argmax(predictions[i]))
85 
86 image = tf.keras.preprocessing.image.load_img(r'/content/drive MyDrive/
    imagens de teste/NORMAL (4).png', target_size = (224,224))
87 plt.imshow(image);
88 
89 image = tf.keras.preprocessing.image.img_to_array(image)
90 image = np.expand_dims(image, axis = 0)
91 image = tf.keras.applications.vgg16.preprocess_input(image)
92 
93 predictions = model.predict(image)
94 print(predictions)
```

Código B.1 – Modelo VGG16

# API da Aplicação FastAPI

## Funcão Main.py

```
1
2 #!/usr/bin/env python3
3 # -*- coding: utf-8 -*-
4 """
5 Created on Mon Feb  8 01:21:25 2021
6 @author: elilsonsantos
7 """
8 import uvicorn
9 from fastapi import FastAPI, File, UploadFile
10 from prediction import predict, read_imagefile
11
12 app = FastAPI()
13
14 # Routes
15 @app.get('/')
16 async def index():
17     return{"text": "Hello API Elilson"}
18
19 @app.get('/items/{name}')
20 async def get_items(name):
21     return{"name": name}
22
23 @app.post('/predict/image')
24 async def predict_api(file: UploadFile = File(...)):
25     extension = file.filename.split(".")[-1] in ("jpg", "jpeg", "png")
26     if not extension:
27         return "Image must be jpg or png format!"
28     image = read_imagefile(await file.read())
29     prediction = predict(image)
30
31     val = prediction.tolist()
32     return val
33
34 if __name__ == '__main__':
35     uvicorn.run(app,host="127.0.0.1", port =43869)
```

Código B.2 – Arquivo main.py da API web

## Funcão Prediction.py

```
1
2 #!/usr/bin/env python3
3 # -*- coding: utf-8 -*-
4 """
5 Created on Mon Feb  8 01:24:31 2021
6 @author: elilsonsantos
7 """
8 from io import BytesIO
9 import numpy as np
10 import tensorflow as tf
11 import keras
12 from PIL import Image
13 from tensorflow.keras.applications.imagenet_utils import
    decode_predictions
14
15 model = None
16
17 input_shape = (224,224)
18
19 def load_model():
20     model = tf.keras.models.load_model('model/my_model.hdf5')
21     print("Model loaded")
22     return model
23
24 def predict(image: Image.Image):
25     global model
26     if model is None:
27         model = load_model()
28
29     image = np.asarray(image.resize((224, 224)))
30     image = tf.keras.preprocessing.image.img_to_array(image)
31     image = np.expand_dims(image, axis = 0)
32     image = image / 127.5 - 1.0
33
34     result = model.predict(image)
35     print(result[0])
36
37     return result
38
39 def read_imagefile(file) -> Image.Image:
40     print(type(file))
41     image = Image.open(BytesIO(file)).convert('RGB')
42     return image
```

Código B.3 – Arquivo predict.py API web

## Aplicação Flutter

<https://github.com/eliltion/preditor_doenca_pulmonar.git>