UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Ciência e Tecnologia
Campus de Sorocaba

FERNANDA TREVISAN WATANABE

Sistema para Diagnóstico de Patologias do Sistema Respiratório Utilizando Redes Neurais

Sorocaba - SP
2022

FERNANDA TREVISAN WATANABE

SISTEMA PARA DIAGNÓSTICO DE PATOLOGIAS DO SISTEMA RESPIRATÓRIO UTILIZANDO REDES NEURAIS

Trabalho de graduação apresentado ao Instituto de Ciência e Tecnologia, da Universidade Estadual Paulista "Júlio de Mesquita Filho" Campus Sorocaba, como parte dos requisitos para obtenção do grau de bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Ivando Severino Diniz.

Coorientador: Prof. Dr. Wesley Angelino de Souza.

Sorocaba - SP

2022

W324s

Watanabe, Fernanda Trevisan
Sistema para Diagnóstico de Patologias do Sistema Respiratório Utilizando Redes Neurais / Fernanda Trevisan Watanabe. -- Sorocaba, 2022
52 p. : il., tabs., fotos + 1 CD-ROM

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Ivando Severino Diniz
Coorientador: Wesley Angelino de Souza

1. Pulmões Doenças. 2. Redes neurais (Computação). 3. Processamento de imagens. 4. Tórax Radiografia. 5. Aplicativos móveis. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Ciência e Tecnologia
Campus de Sorocaba

SISTEMA PARA DIAGNÓSTICO DE PATOLOGIAS DO SISTEMA RESPIRATÓRIO UTILIZANDO REDES NEURAIS

FERNANDA TREVISAN WATANABE

BANCA EXAMINADORA:

Prof. Dr. IVANDO SEVERINO DINIZ

UNESP–ICTS - Engenharia de Controle e Automação

Prof. Dr. EVERSON MARTINS

UNESP–ICTS - Engenharia de Controle e Automação

Prof. MSc. LUCAS NUNES MONTEIRO

Centro Universitário FACENS – Engenharia Mecatrônica

Junho de 2022

AGRADECIMENTOS

Durante toda a vida existem pessoas que vão andar conosco, mudar nossas perspectivas, nos apoiar e nos fazer crescer como ser humano. E eu não poderia deixar de agradecer as pessoas que entraram na minha vida e me transformaram em alguém melhor.

Agradeço primeiramente a minha família, meus pais Sandra e Luis, minhas irmãs Ana Lúcia e Renata, meu cunhado Júlio e minha sobrinha Carolina, por todo o suporte e carinho que tive ao longo da minha vida, e que foram fundamentais para o ingresso e realização do curso de Engenharia de Controle e Automação em uma das melhores universidades do país.

Ao professor Ivando, pela paciência e orientação ao longo de toda a execução deste trabalho, além de outros projetos da faculdade, sempre muito solícito e atencioso.

Ao professor Wesley pela incrível ajuda fornecida com as aulas de Ciências de Dados, imprescindíveis para o desenvolvimento deste projeto.

Ao meu amigo e companheiro Caique Mello por todo o apoio durante a graduação e realização deste trabalho.

Às minhas amigas do coração: Tainá Preto, Ana Beatriz Oliveira, Gabriela Mello e Letícia Leal, por todo o apoio e acreditarem no meu potencial.

E finalmente à UNESP, por proporcionar alguns dos melhores momentos da minha vida junto a melhor turma já reunida no curso de engenharia, TURMA XIII.

A todos aqui, serei eternamente grata.

RESUMO

Atualmente o diagnóstico de um exame radiológico por imagem depende inteiramente da interpretação visual do profissional de saúde, sendo este unicamente responsável pelos resultados e tratamentos derivados desta interpretação. Ainda na formação dos profissionais da saúde, a interpretação destas imagens pode ser de difícil definição, uma vez que com o desenvolvimento de tecnologias que se agregam à Radiologia, pode tornar o estudo desta mais complexo. Diante deste cenário, o presente trabalho de graduação teve por objetivo criar um protótipo de uma ferramenta que auxilie estudantes de medicina e enfermagem a examinar com maior precisão imagens de radiografias torácicas para obter um melhor diagnóstico. O processo permite que uma nova imagem radiográfica seja inserida em um aplicativo para dispositivos móveis, onde será analisada e comparada a outras similares com o auxílio de uma rede neural convolucional treinada, retornando ao usuário o diagnóstico mais preciso da patologia presente nesta imagem. Esta interface foi desenvolvida utilizando os *frameworks* Jupyter Notebook, Ionic e Angular, e implementada em um ambiente simulado em uma máquina de desenvolvimento. Após todas as etapas de construção da ferramenta, o sistema foi devidamente testado, inserindo imagens com patologias mapeadas pela rede neural, patologias não mapeadas e imagens sem patologias detectáveis, a fim de se garantir o resultado entregue ao usuário. Os resultados mostram que, para a atual modelagem, a rede neural consegue prever uma imagem com aproximadamente 70% de acurácia geral, considerando uma base de dados de radiografias torácicas sem patologias (normais), com presença de pneumonia bacteriana e com pneumonia viral.

Palavras-Chave: Patologias pulmonares; Redes neurais convolucionais; Treinamento; Estudo; Processamento de imagens; Reconhecimento de radiografias torácicas.

ABSTRACT

Currently, the diagnosis of a radiological imaging exam depends entirely on the visual interpretation of the health professional, who is solely responsible for the results and treatments derived from this interpretation. During the academic education of health professionals, the interpretation of these images can be difficult to define once the development of new technologies is being combined to the study of radiology, making its study more complex. Given this scenario, the present graduation paper aimed to create a prototype of a tool that helps medical and nursing students to examine more accurately chest radiographic images in order to obtain a better diagnosis. The process allows a new radiographic image to be inserted into an application for mobile devices, where it will be analyzed and compared to similar ones, with the help of a trained neural network, returning to the user the most accurate diagnosis of the pathology present in this image. This interface was developed using Jupyter Notebook, Ionic and Angular frameworks, and implemented in a simulated environment on a development machine. After all the steps to build the tool, the system was properly tested, inserting images with pathologies mapped by the neural network, unmapped pathologies and images without detectable pathologies, in order to ensure the result delivered to the user. The results show that for the current model, the neural network is able to predict an image with approximately 70% overall accuracy, considering a database of chest x-rays without pathologies (normal), with the presence of bacterial pneumonia, and with viral pneumonia.

Keywords: Pulmonary pathologies; Convolutional neural networks; Training; Study; Image processing; Recognition of chest x-rays.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CNNs	*Convolutional Neural Networks*
ConvLayer	*Convolutional Layer*
ReLU	*Rectified Linear Unit*
APK	*Android Package Kit*
SPA	*Single Page Application*
API	*Application Programming Interface*
RNA	Rede Neural Artificial

SUMÁRIO

1. INTRODUÇÃO

As infecções respiratórias baixas representaram em 2016 a quarta principal causa de morte no Brasil, seguida pela Doença Pulmonar Obstrutiva Crônica (DPOC), e em décimo lugar, o câncer de traqueia, brônquio e pulmão. No cenário mundial estes números não diferem muito, apenas tendo leve alterações nas posições do ranking (MINISTÉRIO DA SAÚDE, 2019). Pelos últimos dois anos vivenciamos um período de pandemia devido a uma doença infecciosa de elevada transmissibilidade, que afeta justamente o sistema respiratório, a Covid-19.

As doenças respiratórias são enfermidades que atingem os órgãos e as estruturas do sistema respiratório (vias nasais, faringe, laringe, brônquios, traqueia, diafragma, pulmões e alvéolos pulmonares). Esses distúrbios causam inflamações e irritação na região respiratória, além de provocarem a obstrução das vias aéreas, de forma a dificultar a passagem do ar e impedir a respiração completa (OMRON HEALTHCARE, 2020).

São inúmeras as possíveis causas das doenças respiratórias, sendo algumas delas: infecções Virais e Bacterianas; uso recorrente de cigarro (que contém diversas substâncias nocivas ao corpo humano); a poluição do ar, sendo um fator que tem contribuído cada vez mais para as doenças respiratórias devido à industrialização das cidades; efeitos colaterais de medicamentos; fungos. As estações mais frias do ano também contribuem para o desenvolvimento de algumas doenças respiratórias, por conta do ar seco, temperaturas baixas e aglomerações em espaços fechados, contribuindo para a disseminação de doenças contagiosas.

Em suspeitas de doenças respiratórias, a radiografia torácica é um dos exames mais requeridos pelos médicos a fim de se avaliar os órgãos, evidenciando os pulmões. Este foi um exame imprescindível durante a fase de pico da pandemia da Covid-19, averiguando as condições dos pacientes durante e após a enfermidade, além de auxiliar especialistas no estudo dos efeitos desta nova doença sob o corpo humano.

1.1. Estrutura Do Trabalho

O presente trabalho está organizado em 5 capítulos da seguinte forma: o Capítulo 1 (atual) contém uma breve introdução, estrutura do trabalho, justificativa e objetivos; no Capítulo 2 são explicados os conceitos básicos necessários para a compreensão deste trabalho; o Capítulo 3 demonstra os sistemas utilizados para o desenvolvimento do

projeto e os modelos utilizados, e também descreve as metodologias utilizadas para realização deste; no Capítulo 4 são abordados os experimentos realizados e os resultados obtidos do sistema final; e, por fim, o Capítulo 5 expõe as conclusões geradas com o estudo e desenvolvimento da aplicação, bem como as sugestões para trabalhos futuros.

1.2. Justificativa

O erro médico é um assunto pouco debatido durante a formação de um profissional da saúde, e que pode acarretar em diversos problemas se estes erros/imprevistos não forem detectados e tratados com antecedência. Segundo o II Anuário da Segurança Assistencial Hospitalar No Brasil:

> Somados, os hospitais públicos e privados do Brasil registraram, em 2017, seis mortes, a cada hora, decorrentes dos chamados "eventos adversos graves", ocasionados por erros, falhas assistenciais ou processuais ou infecções, entre outros fatores. Desses, mais de quatro óbitos seriam evitáveis (COUTO, 2018).

Buscando auxiliar nesta área da medicina, a implementação de uma ferramenta que possa fazer a análise de uma radiografia torácica e retornar a maior probabilidade de detecção de uma determinada doença mitigaria as possíveis falhas humanas na conferência de exames de radiografia. Além da aplicação prática em um consultório, a ferramenta ainda pode auxiliar alunos de graduação no estudo de padrões de determinadas patologias detectadas nas imagens dos exames.

1.3. Objetivo

O presente trabalho tem por objetivo auxiliar estudantes e profissionais da medicina a diagnosticar corretamente patologias pulmonares a partir de uma imagem de radiografia torácica, contribuindo com o aprendizado e acurácia da análise. Esta avaliação será feita por meio de um aplicativo para dispositivos móveis Android, que se utilizará de: uma rede neural convolucional como base para a análise e diferenciação das imagens radiológicas e um banco de imagens de patologias pulmonares, que servirá tanto para o treinamento desta rede neural como para a comparação de novas imagens que o usuário do aplicativo possa inserir.

2. CONCEITOS FUNDAMENTAIS

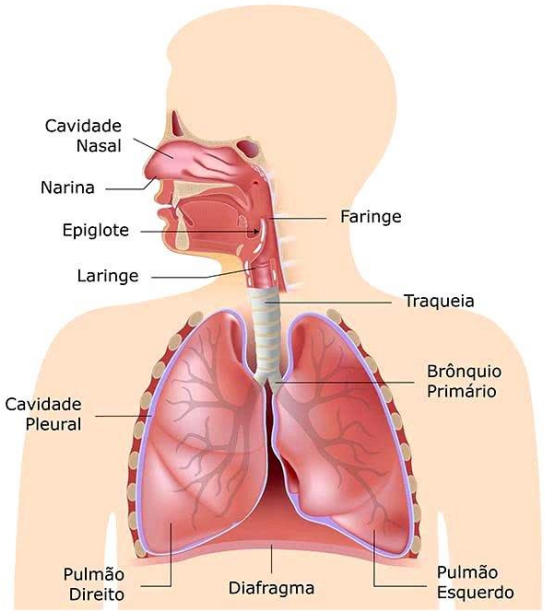
Esta seção tem o objetivo de introduzir o leitor nos diversos conceitos práticos e teóricos a serem utilizados ao longo do desenvolvimento do trabalho.

2.1. Fisiologia Pulmonar

O sistema respiratório é composto pelos pulmões e as vias de condução, que por sua vez, compreende as fossas nasais, nasofaringe, laringe, traqueia, brônquios e bronquíolos, como observado na Figura 1. É na porção condutora que o ar inspirado é limpo, aquecido e umedecido, de forma a proteger o revestimento dos alvéolos pulmonares.

Os pulmões têm como função principal a respiração, processo em que ocorre a troca gasosa entre o organismo e o meio ambiente. Em uma pessoa viva e saudável, os pulmões apresentam-se leves, macios e esponjosos; em contrapartida, um pulmão que apresenta alguma enfermidade pode sofrer alterações na coloração, tamanho, peso e textura (DIANA, 2019).

Figura 1 - Órgãos que compõem o Sistema Respiratório.



Fonte: (DIANA, 2019).

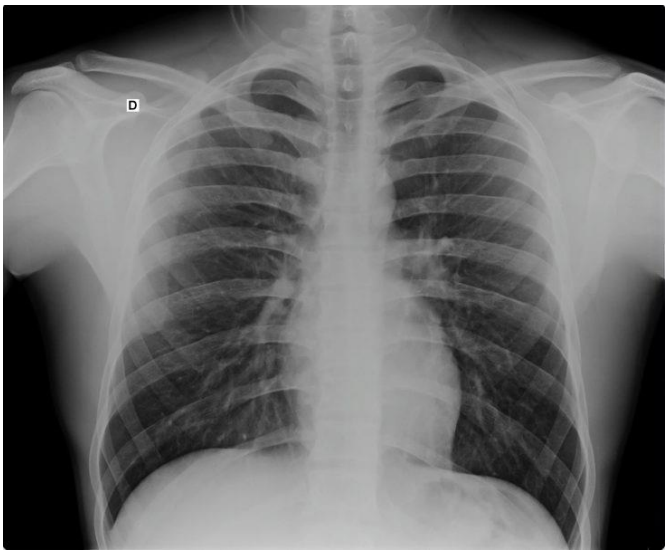
2.2. Patologias Pulmonares Comuns No Brasil

Uma das maiores preocupações em saúde no mundo são as doenças pulmonares. Elas respondem por grande parcela das mortes em todo o mundo, podem implicar em um potencial incapacitante, visto que em muitas destas doenças existe um aspecto crônico que pode se estende por anos. As doenças pulmonares também implicam em alto custo relacionado aos cuidados primários, bem como relacionado aos tratamentos hospitalares. No Brasil, as principais doenças pulmonares são:

- Apneia obstrutiva do sono: é um transtorno do sono, onde a via aérea torna-se repetidamente bloqueada pelo relaxamento dos tecidos da faringe e da base da língua, limitando a quantidade de ar que atinge os pulmões;
- Asma: é uma inflamação crônica nos pulmões onde a pessoa afetada apresenta dificuldades para respirar, falta de ar e sensação de pressão no peito;
- DPOC (Doença Pulmonar Obstrutiva Crônica): é a obstrução da passagem do ar pelos pulmões provocada geralmente pela fumaça do cigarro ou de outros compostos nocivos. A doença se instala depois que há um quadro persistente de bronquite ou enfisema pulmonar. Pode ocorrer também em não-fumantes, mas em poucos casos;
- Pneumonia: é uma inflamação que atinge principalmente os bronquíolos (pequenos tubos que transportam o ar dos brônquios para os alvéolos, onde ocorre a troca gasosa) e o interstício (tecido mais interno do órgão);
- Embolia pulmonar: é uma condição grave, também conhecida como trombose pulmonar, que surge quando um coágulo entope um dos vasos que leva sangue para o pulmão, fazendo com que o oxigênio não consiga chegar nos tecidos da parte afetada do pulmão, resultando em falta de ar repentina, tosse intensa e dor no peito;
- Tuberculose: é uma doença infecciosa e transmissível que afeta prioritariamente os pulmões, embora possa atingir outros órgãos e/ou sistemas, e é causada por uma bactéria;
- Câncer de pulmão: é o crescimento/multiplicação anormal de células no pulmão, tendo como principais causas o tabagismo, fumo passivo, exposição a determinadas toxinas e histórico familiar;
- Doenças intersticiais pulmonares: são um grupo de doenças respiratórias caracterizadas pela progressiva cicatrização do interstício pulmonar resultando em

insuficiência respiratória. Pode ocorrer quando uma lesão nos pulmões desencadeia uma resposta de cura excessiva (EQUIPE VERITÀ, 2021).

Figura 2 - Exemplo de radiografia torácica.



Fonte: (EQUIPE VERITÀ, 2021)

Todas as patologias acima podem ser detectadas via exames de imagem e algumas serviram como base de dados para o presente projeto.

2.3. Radiologia

> A Radiologia é uma especialidade em constante avanço e atualização, o que torna difícil, mesmo para o especialista, manter-se sempre atualizado em relação a todos os métodos e doenças dos diversos sistemas. Os avanços são tantos que nos últimos anos a especialidade tem se aproximado cada vez mais de outras áreas médicas como a genética e biologia molecular, além de outros campos da ciência como a informática biomédica e bioengenharia, interagindo com ferramentas de inteligência artificial e criando novas tecnologias, como a Radiômica e Radiogenômica. Para o aluno de graduação em medicina, que precisa aprender tudo o que faz parte das grandes áreas clínicas para sua formação como médico generalista, conhecer melhor a Radiologia torna-se um desafio maior ainda (SANTOS, 2019).

Objeto de estudo da Radiologia, o exame de radiografia - focado para este trabalho na avaliação do tórax - é um dos método de imagem mais antigo, de baixo custo e acessível à população, representando na maioria das situações, o primeiro exame

radiológico a ser solicitado para avaliação das doenças torácicas. Apesar de todos os avanços tecnológicos de demais métodos de imagem, a radiografia ainda representa o método de imagem radiológica com a maior resolução espacial (definida como a habilidade em se distinguir pequenos objetos em alto contraste, limitada pelo tamanho mínimo do pixel) (WADA, RODRIGUES e SANTOS, 2019).

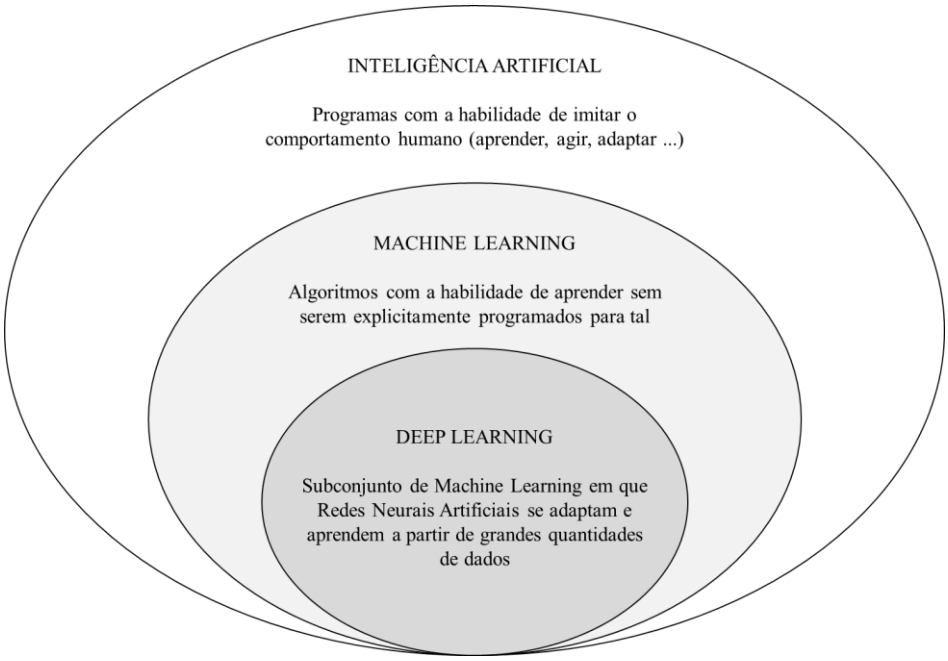
2.4. Redes Neurais

Partindo primeiramente do conceito de Inteligência Artificial, esta refere-se a sistemas que buscam imitar a inteligência humana para o desempenho de tarefas, e que podem se aprimorar com base nas informações coletadas (ORACLE). Ela possui dois subconjuntos principais, retratados na Figura 3: o Machine Learning (ou Aprendizado de Máquina) e o Deep Learning (Aprendizado Profundo), sendo este último onde estão inseridas as Redes Neurais.

As redes neurais artificiais (ou RNA) consistem em um método para solucionar problemas de inteligência artificial, construindo um sistema que tenha circuitos que simulem o nosso cérebro, inclusive em seu comportamento, com erros e acertos, e ainda aquisição de conhecimento a partir de experiências. Sua estrutura é baseada no modelo de neurônios humanos e da forma como trabalham. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento, enquanto que o cérebro de um mamífero pode ter muitos bilhões de neurônios.

Uma RNA consiste em um grande número de elementos de processamento simples chamados de neurônios, ou nós. Cada neurônio está conectado a outros neurônios por meio de ligações direcionadas, cada um com um peso associado. Os pesos representam informações que estão sendo usadas pela rede para resolver um problema. As RNAs podem ser aplicadas a uma ampla variedade de problemas, como armazenamento e recuperação de dados, classificar padrões, realizar mapeamentos em geral a partir de padrões de entrada, agrupando padrões similares, ou encontrar soluções para problemas de otimização (OLIVEIRA, 2011).

Figura 3 - Diagramada ilustrando os conceitos básicos de Inteligência Artificial, Machine Learning e Deep Learning.



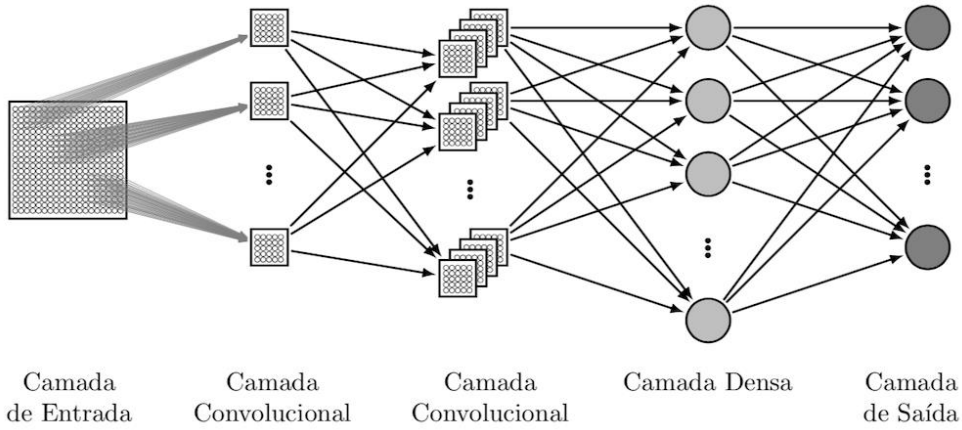
Fonte: Autor.

2.5. Redes Neurais Convolucionais

As Redes Neurais Convolucionais (ou *CNNs*), são redes neurais utilizadas em especial para trabalhos com imagens, como classificação, agrupar por similaridade e reconhecimento de objetos em cenas. Por serem tão eficientes no processo de reconhecimento de imagens, as redes neurais convolucionais evidenciaram a eficácia da aprendizagem profunda (*Deep Learning*), de forma a também impulsionar avanços no campo da Visão Computacional, com aplicações no ramo da robótica, segurança, diagnósticos médicos e muitos outros.

Uma Rede Neural Convolucional é um algoritmo de *Deep Learning* que pode captar uma imagem de entrada, atribuir valor (pesos e vieses que são adquiridos durante a fase de treinamento) a diversos aspectos e/ou objetos da imagem a serem analisados e ser capaz de diferenciar um de outro. O pré-processamento necessário em uma *CNN* é muito menor em comparação com outros algoritmos de classificação. Enquanto nos métodos primitivos os pesos são projetados manualmente, com treinamento suficiente, as *CNNs* têm a capacidade de aprender independentemente estes pesos/características (DEEP LEARNING BOOK).

Figura 4 - Exemplo de arquitetura de uma rede neural convolucional.



Fonte: (RAFAEL SAKURAI, 2017)

2.6. Convolução

Para fins matemáticos, uma convolução é a medida integral de quanto duas funções se sobrepõem, quando estas se cruzam. Ela pode ser dita como uma maneira de misturar duas funções por meio da multiplicação destas, gerando uma terceira. Este processo pode ser determinado como um filtro (ou *kernel*) que transforma uma imagem de entrada (ou tensor).

Uma *CNN* é capaz de capturar com sucesso as dependências espaciais e temporais em uma imagem através da aplicação de filtros relevantes. A arquitetura executa um melhor ajuste ao conjunto de dados da imagem devido à redução no número de parâmetros envolvidos e na reutilização de pesos. Em outras palavras, a rede pode ser treinada para entender melhor a sofisticação da imagem.

O papel da *CNN* é reduzir as imagens para um formato mais fácil de ser processado, sem que perca os recursos essenciais para obter uma boa previsão. Isso é importante quando se deve projetar uma arquitetura que não seja apenas boa em aprender recursos, mas também seja escalável para conjuntos de dados maciços.

O objetivo da operação de convolução é extrair os recursos de alto nível, como por exemplo as bordas da imagem de entrada. As redes neurais convolucionais não precisam ser limitados apenas a uma camada convolucional. Convencionalmente, a primeira camada convolucional (ou primeira *ConvLayer*), é responsável por capturar os

recursos de baixo nível como bordas, cores, orientação de gradiente, etc. de imagens no conjunto de dados, semelhante a como seria feito à mão.

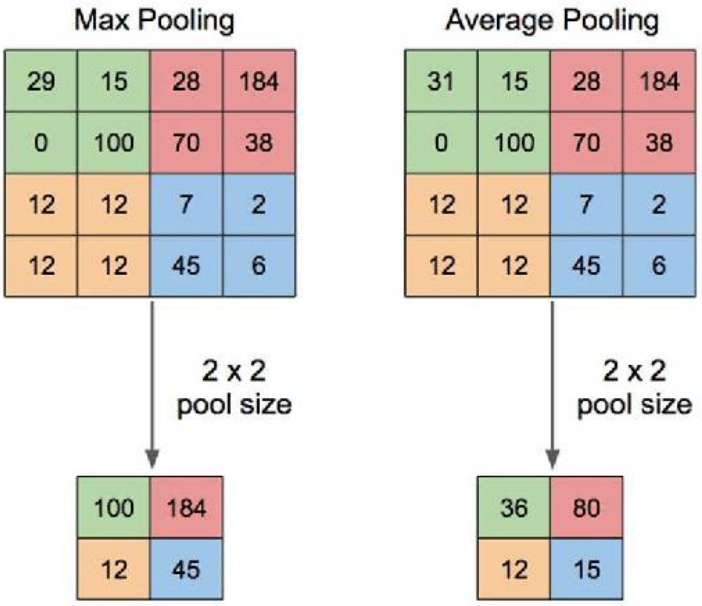
Existem dois tipos de resultados para a operação - um no qual o recurso envolvido é reduzido em dimensionalidade em comparação com a entrada e o outro no qual a dimensionalidade é aumentada ou permanece a mesma. No caso de uma *CNN*, a convolução é realizada nos dados de entrada com o uso de um *kernel* para produzir um Mapa de Recursos (ou *Feature Map*), que é um tensor de saída. Este mapa será então a soma das multiplicações de matrizes para cada região da imagem em que for aplicado o filtro.

2.7. Pooling e Downsampling

Semelhante à camada convolucional, a camada *Pooling* (em tradução livre, camada de agrupamento), é responsável por reduzir o tamanho espacial do recurso envolvido, ou seja, é um processo de *Downsampling* (em tradução livre, redução de amostras). Esta redução se faz necessária para diminuir o processamento computacional necessário para analisar os dados. Além disso, é útil para extrair características dominantes que são invariantes rotacionais e posicionais, mantendo assim o processo de treinamento eficaz do modelo.

Existem dois tipos principais de *pool*: *Pool* Máximo (*Max Pooling*) e *Pool* Médio (*Average Pooling*). Ambas as estratégias seguem os mesmos princípios, diferenciando apenas na forma de calcular o valor final. O *Max Pooling* retorna o valor máximo da parte da imagem coberta pelo *kernel*. Por outro lado, o *Average Pooling* retorna a média de todos os valores da parte da imagem coberta pelo *kernel*.

Figura 5 - Exemplo de operações de *Max Pooling* e *Average Pooling*.



Fonte: (MUHAMAD Y., 2019)

A matemática por trás do *pool* é no nível básico, consiste em passar uma pequena janela pela imagem e obter o valor máximo da janela em cada etapa. Na prática, uma janela de 2 ou 3 pixels em um lado e etapas de 2 pixels são ideais.

Após o agrupamento, uma imagem possui cerca de um quarto dos pixels iniciados. Como mantém o valor máximo de cada janela, preserva os melhores ajustes de cada recurso dentro da janela. Isso significa que ele não se importa exatamente onde o recurso se encaixa, desde que se encaixe em algum lugar dentro da janela. O resultado disso é que as *CNNs* podem descobrir se um recurso está em uma imagem sem se preocupar com a localização. Isso ajuda a resolver o problema de computadores serem hiper literais.

O *Max Pooling* também funciona como um supressor de ruído, descartando completamente as ativações ruidosas e também realiza a remoção do ruído junto com a redução da dimensionalidade. Por outro lado, o *Average Pooling* simplesmente realiza a redução da dimensionalidade como um mecanismo de supressão de ruído. Portanto, pode-se dizer que o *Max Pooling* tem um desempenho superior ao *Average Pooling*.

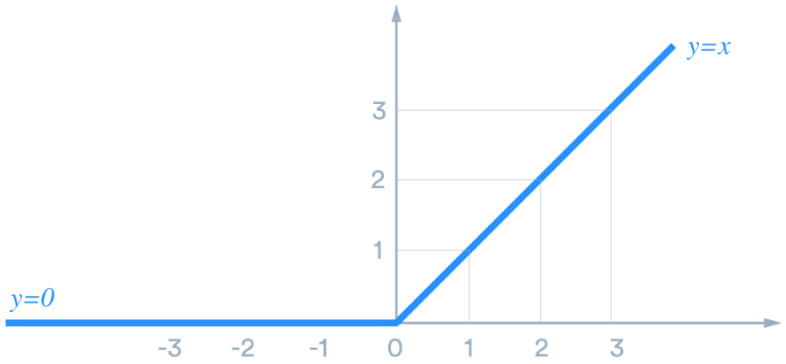
A camada convolucional e a camada de *pooling* formam juntas a i-ésima camada de uma rede neural convolucional. Dependendo da complexidade das imagens, o número de camadas pode ser aumentado para capturar detalhes de níveis ainda mais baixos, mas ao custo de maior poder computacional.

2.8. Função de Ativação

A Função de Ativação é a função responsável por adicionar a não-linearidade à *CNN*, sendo a Unidade Linear Retificada (ou *ReLU*) a mais usada em Deep Learning atualmente. Ela é um nó que é colocado no final ou entre as camadas, onde a saída transformada é então enviada para a próxima camada de neurônios como entrada.

A matemática também é muito simples, a ReLU é definida como $y = \max(0, x)$, representado na Figura 6.

Figura 6 - Função de Ativação *ReLU*.



Fonte: (CLAPPIS, 2019)

Outra função de ativação utilizada como a camada final da rede é a função Softmax. Geralmente utilizada para quando se tem mais de duas possíveis classificações, ela converte a saída de uma rede neural em probabilidades de os dados serem de uma das classes definidas, e a soma de todas as probabilidades é igual a 1. A equação que representa esta função de ativação é descrita abaixo:

$$\Phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Onde i representa o índice do neurônio de saída sendo calculado e j os índices de todos os neurônios de um nível. A variável z representa o vetor de neurônios de saída. A função de ativação Softmax se diferencia das demais no quesito da dependência do valor de saída, das demais possibilidades de saída (REIS, 2016).

2.9. Camada Totalmente Conectada

Nesta etapa, os neurônios têm uma conexão completa com todas as ativações das camadas anteriores. Suas ativações podem, portanto, ser calculadas com uma multiplicação de matrizes seguida por um deslocamento de polarização. Esta é a última fase para uma rede neural convolucional.

A *CNN* é na verdade composta de camadas ocultas e camadas totalmente conectadas. Adicionar uma camada totalmente conectada é uma maneira (geralmente) barata de aprender combinações dos recursos de alto nível, conforme representado pela saída da camada convolucional.

Após a conversão da imagem de entrada em um formato adequado de vários níveis, busca-se achatar a imagem em um vetor de coluna, processo denominado de *Flatten* (de tradução livre, achatar). A saída achatada é alimentada a uma rede neural do tipo f*eed-forward* e o *backpropagation* é aplicado a todas as iterações do treinamento. Ao longo de uma série de épocas, o modelo é capaz de distinguir entre certos recursos de baixo nível e recursos dominantes nas imagens e classificá-los usando a técnica de Classificação *Softmax*. Na prática, essa etapa final é um modelo de Aprendizado de Máquina (no inglês, *Machine Learning*), para Classificação, que recebe como entrada os recursos aprendidos nas camadas convolução.

2.10. Treinamento E Backpropagation

A maneira como o computador é capaz de ajustar seus valores (ou pesos) de filtro é através de um processo de treinamento chamado retropropagação ou *Backpropagation*. Antecedendo o conceito de *Backpropagation*, é necessário definir o que uma rede neural necessita para funcionar corretamente. Antes do início do treinamento da *CNN*, os pesos ou valores do filtro são escolhidos de forma aleatória, pois ainda não se sabe quais são os melhores valores. Isso é o que a rede vai aprender.

Os filtros não sabem procurar arestas e curvas. No entanto, a ideia de receber uma imagem e um rótulo é o processo de treinamento pelo qual as *CNNs* passam. Assim, o treinamento da rede neural pode ser separado em 4 seções distintas:

- Passagem para frente (*feed-forward*)
- Função de perda (*loss function*)
- Passagem para trás (*backpropagation*)

- Atualização de peso

O processo é iniciado com uma imagem de treinamento, passando-a por toda a *CNN*. Para o primeiro exemplo, com todos os pesos ou valores de filtro inicializados aleatoriamente, a saída provavelmente será algo que não dá preferência a nenhum número em particular. A rede, com seus pesos atuais, não é capaz de procurar esses recursos de baixo nível ou, portanto, não é capaz de chegar a uma conclusão razoável sobre qual poderia ser a classificação.

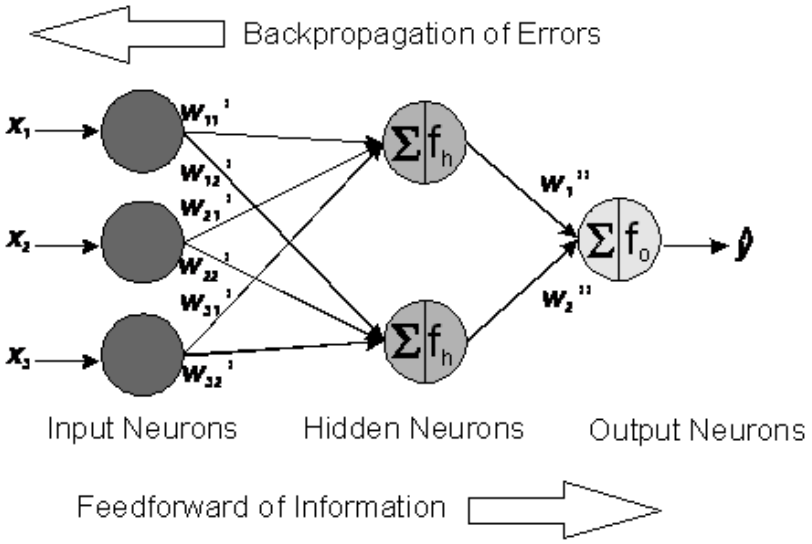
Isso vai para a parte da função de perda da retropropagação. Uma função de perda pode ser definida de várias maneiras diferentes, uma comum é a Mean-Square Error (ou Erro Quadrático Médio), que é a média da diferença entre o valor do estimador e do parâmetro ao quadrado.

A perda será extremamente alta para as duas primeiras imagens de treinamento. O ponto a ser alcançado equivale ao que o rótulo previsto (saída da *CNN*) seja o mesmo que o rótulo de treinamento (isso significa que a rede acertou sua previsão). Para chegar lá, é necessário minimizar a quantidade de perdas (erros). Visualizando isso como apenas um problema de otimização no cálculo, precisa-se descobrir quais entradas (pesos) contribuíram mais diretamente para a perda (ou erro) da rede, e então atualizar estes pesos para a próximo treinamento. Esse processo é repetido dezenas, centenas ou milhares de vezes até que o aprendizado aconteça.

A taxa de aprendizado (ou *learning rate*) é um parâmetro escolhido pelo Cientista de Dados. Uma alta taxa de aprendizado significa que são tomadas medidas maiores nas atualizações de peso e, portanto, pode levar menos tempo para o modelo convergir para um conjunto ideal de pesos. No entanto, uma taxa de aprendizado muito alta pode resultar em saltos muito grandes e pouco precisos para atingir o ponto ideal.

O processo de passagem para frente, função de perda, passagem para trás e atualização de pesos é uma iteração de treinamento. O programa repetirá esse processo para um número fixo de iterações para cada conjunto de imagens de treinamento (geralmente chamado de lote). Depois de concluir a atualização do parâmetro no último exemplo de treinamento, espera-se que a rede seja treinada o suficiente para que os pesos das camadas sejam ajustados corretamente.

Figura 7 - Sinalização dos processos de *Backpropagation* e *Feed-Forward*.



Fonte: (DIETERLE, F., 2019)

2.11. Overfitting

Overfitting é o termo que se refere a um modelo que decorou o conjunto de dados de treinamento, incluindo os ruídos estatísticos ou flutuações aleatórias presentes na base de dados. Isso se torna um problema porque quanto mais especializado o modelo se torna para os dados de treinamento, menos ele é capaz de generalizar para novos dados.

2.12. MobileNetV2

MobileNetV2 é uma rede neural convolucional que tem foco em desempenho em dispositivos móveis via o reconhecimento de imagens, incluindo classificação, detecção de objetos e segmentação semântica (SANDLER, 2018). Ela tem por base o conceito de *Depthwise Separable Convolution*, que é uma forma de convolução fatorada, e também introduz uma nova estrutura residual invertida, onde as ligações residuais se encontram entre as camadas de estrangulamento (ou *bottleneck*).

A arquitetura do MobileNetV2 é composta por uma camada convolucional inicial com 32 filtros, seguida por 19 camadas *bottleneck*. Para a convolução espacial, são utilizados *kernels* de tamanho 3 × 3.

Figura 8 - Arquitetura geral MobileNetV2, onde t: fator de expansão, c: número de canais.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fonte: (SANDLER, 2018)

2.13. Parâmetros de Compilação e Treinamento

No momento de compilação e posteriormente no treinamento do modelo, é necessário estipular alguns parâmetros:

- Otimizador: é um algoritmo que busca auxiliar a rede durante o treinamento para obter seu melhor desempenho;
- Função de Perda: é um método que avalia quão bem o modelo lida com a base de dados, indicando os erros encontrados durante uma época de treinamento. Quanto melhor o modelo, menor o número de erros encontrado, retornando então um valor mais próximo de zero;
- Tamanho do batch: é a quantidade de amostras que são utilizadas para cada iteração de treino (tamanho padrão: 32 amostras);
- Taxa de aprendizado: valor escalar pelo qual são multiplicados os gradientes antes de aplicá-los aos pesos;
- Número de iterações de treino: por quantas iterações foi treinada a rede;
- Número de épocas de treino: medida aproximada de quantas vezes a rede "viu" o conjunto de treino inteiro durante o treino. Pode ser obtido pelo número de imagens visto durante o treino (tamanho do batch multiplicado

pelo número de iterações de treino) dividido pelo número de imagens de treino no banco de dados (UTSCH, 2018).

Estes parâmetros podem ser combinados de diversas formas para se obter o melhor desempenho da Rede Neural.

2.14. Matriz de Confusão, Precisão e Recall

A partir do treinamento do modelo, podemos inferir informações de seu desempenho pela Matriz de Confusão, como a precisão e recall. A precisão é a métrica quem revela quantas das ocorrências são relevantes, e o recall quantas ocorrências relevantes foram encontradas. A Matriz de Confusão mostra as predições da rede divididas em falsos positivos, verdadeiros positivos, falsos negativos e verdadeiros negativos. Com isso podemos inferir a métrica de predição, que corresponde à proporção de verdadeiros positivos em relação ao total de positivos previstos, enquanto o recall é a proporção de verdadeiros positivos em relação ao total real de positivos.

- Verdadeiros Positivos (N_{VP}), é o número de instâncias positivas classificados como positivas.
- Falsos Positivos (N_{FP}), é o número de instâncias negativas classificados como positivas.
- Verdadeiros Negativos (N_{VN}), é o número de instâncias negativas classificados como negativas.
- Falsos Negativos (N_{FN}), é o número de instâncias positivas classificados como negativas (UTSCH, 2018).

$$predição = \frac{N_{VP}}{N_{VP} + N_{FP}}$$

$$recall = \frac{N_{VP}}{N_{VP} + N_{FN}}$$

3. MATERIAIS E MÉTODOS

3.1. Javascript, HTML e CSS

O Javascript, HTML e CSS são três linguagens de programação, consideradas os pilares da programação Frontend. Cada uma possui uma função específica dentro de uma aplicação:

3.1.1. HTML

O HTML, abreviado do inglês "*HyperText Markup Language*", ou Linguagem de Marcação de Hipertexto, é responsável pela estruturação de uma página web. É através deste que os elementos são posicionados na tela, porém sem definir nenhum tipo de estilo (PEREIRA, 2018).

3.1.2. CSS

CSS, do inglês "*Cascading Style Sheets*", é a linguagem responsável por adicionar estilo à aplicação, com cores, fontes, espaçamento, entre outras propriedades. Em vez de colocar a formatação dentro do documento, o CSS cria um link para uma página que contém os estilos. Porém, a aplicação ainda não possui ações (PEREIRA, 2018).

3.1.3. Javascript

Javascript é uma linguagem de programação criada para ser parte dos navegadores web, para que scripts possam ser executados do lado do cliente e interajam com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido (PEREIRA, 2018).

3.2. Python

Python é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada que possibilita maior controle e estabilidade de códigos para projetos de grandes proporções.

É uma linguagem de sintaxe relativamente simples e de fácil compreensão, onde um de seus maiores atrativos é possuir um grande número de bibliotecas, nativas e de terceiros, tornando-a muito difundida e útil em uma grande variedade de setores dentro de desenvolvimento web, e também em áreas como análise de dados, Machine Learning e IA. (KRIGER, 2022)

No presente trabalho, estas foram as linguagens de programação escolhidas para o desenvolvimento da interface gráfica e do modelo de rede neural do aplicativo.

3.3. Angular

Para o presente trabalho, foi escolhido o framework Angular para desenvolvimento de toda a interface do usuário, sendo uma página web e um aplicativo Android. Este tem como base a linguagem Javascript, e possui uma arquitetura de código aberto, mantido pela Google para a construção de um *SPA* (ou Aplicações de Página Única).

Uma *SPA*, resumidamente, é uma aplicação web construída em uma única página, de forma que a navegação entre as sessões de uma página ocorre de maneira na qual não é necessário recarregar a página em cada uma dessas mudanças.

O Angular tem como finalidade dar ferramentas necessárias para criação de aplicações *SPA*, e busca deixar o desenvolvimento deste tipo de aplicação mais simples e otimizado. Seu uso é voltado para desenvolver aplicações web que se adequam tanto para resoluções desktop quanto para resoluções mobile, tornando-as dinâmicas, modernas e escaláveis. (GUEDES, 2021)

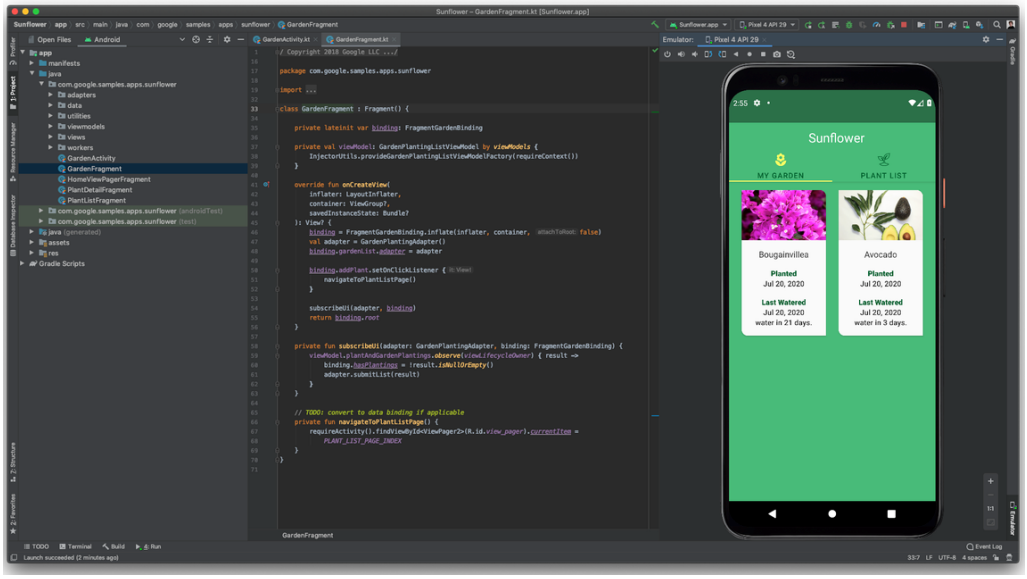
3.4. Capacitor

Capacitor é uma plataforma de execução nativa que facilita a construção de aplicações web modernas que funcionam nativamente em sistemas operacionais iOS, Android e Web. Ele fornece um conjunto centrado em *APIs* que permitem que uma aplicação se mantenha o mais próximo possível dos padrões da web, mas que ao mesmo tempo que atende a funcionalidades de dispositivos nativos. É possível adicionar funcionalidades nativas facilmente com uma *API* de Plugin simples para a linguagem Swift em sistemas iOS, Java para Android, e Javascript na Web (CAPACITORJS).

3.5. Android Studio

O Android Studio é uma plataforma especializada no desenvolvimento para a plataforma Android. Essa ferramenta será utilizada para o gerar o aplicativo Android a partir da *web view* criada utilizando o IONIC. Na Figura 9 é demonstrada um exemplo da interface gráfica do ambiente.

Figura 9 - Interface gráfica do ambiente de desenvolvimento Android Studio.



Fonte: (DEVELOPERS, 2021).

3.6. Visual Studio Code

O desenvolvimento do aplicativo do presente trabalho foi realizado na plataforma Visual Studio Code, uma interface de desenvolvimento (IDE) para implementar e editar o código necessário para a execução do aplicativo, bem como a conexão com o banco de dados. Seguindo o que é utilizado pelo Ionic, o código será implementado usando Javascript, HTML e CSS, por meio do uso dos frameworks Angular e Ionic.

3.7. Ionic

O Ionic é um Framework Open Source gratuito sobre a licença MIT para desenvolvimento de aplicações mobile híbridas a partir da geração de uma *web view*.

Aplicações híbridas são aplicativos móveis construídos de maneira alternativa a aplicações nativa. São construídos, geralmente, utilizando HTML, CSS e Javascript, desta maneira se tornaram extremamente populares, pois permite o desenvolvimento multiplataforma, utilizando o mesmo HTML para diferentes sistemas operacionais.
Projetado para funcionar e ser exibido em diferentes plataformas, possui um design limpo, simples e funcional, com componentes padrões, tipografia, paradigmas interativos e diversos modelos.

Além disso, possui um cliente de linha de comando (ou CLI) para gerenciar todo projeto criado com o Ionic. O CLI é uma ferramenta que cria aplicativos Ionic de forma rápida e fornece vários comandos úteis para facilitar o desenvolvimento utilizando o *framework* (ANDRADE, 2020).

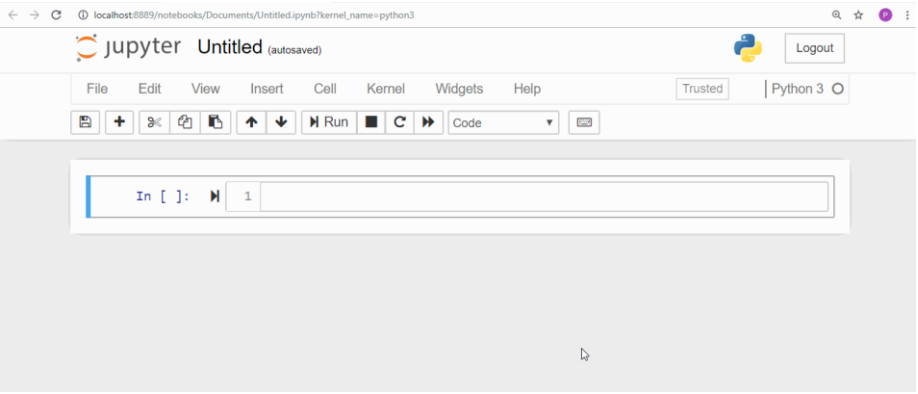
3.8. Jupyter Notebook

O Jupyter Notebook é uma interfarce gráfica que permite a edição de *notebooks* em um navegador web. O nome Jupyter foi composto a partir de um acrônimo, criado a partir das linguagens de programação que inicialmente foram aceitas pelo Projeto Jupyter: Julia, Python e R.

Um documento do tipo *notebook* é um documento virtual que permite a execução de códigos de uma linguagem de programação juntamente com ferramentas para edição de textos comuns; ou seja, além das rotinas usuais de programação, o usuário pode documentar todo o processo de produção do código. Dessa forma, o notebook permite uma maneira interativa de programar.

Os *notebooks* também oferecem uma programação mais dinâmica, oferecendo ao usuário a saída imediata do código, de forma que não há a necessidade de compilar ou executar todo o documento (LET'S CODE, 2019).

Figura 10 - Interface gráfica do ambiente de desenvolvimento Jupyter Notebook.



Fonte: (LET'S CODE, 2019)

3.9. Descrição da Base de Dados

O conjunto de dados de imagem utilizado para o desenvolvimento do presente trabalho foi extraído da plataforma de aprendizado de Data Science, o Kaggle, onde está disponível para a comunidade de pesquisa. Este conjunto contém no total 5863 imagens radiográficas (KERMANY, 2018).

Figura 11 - Exemplos ilustrativos de raio-X em pacientes com Pneumonia.

Fonte: (KERMANY, 2018)

A base de dados foi organizada da seguinte maneira: foram criados três diretórios de imagens: teste, validação e treinamento, onde cada um contém subdiretórios com as classificações desejadas: imagens de radiografias torácicas sem patologias detectáveis (nomeado como "Normal"), pulmões com tuberculose bacteriana (nomeado como "Bactéria") e pulmões com tuberculose viral (nomeado como "Vírus"). Sua divisão é vista na Tabela 1.

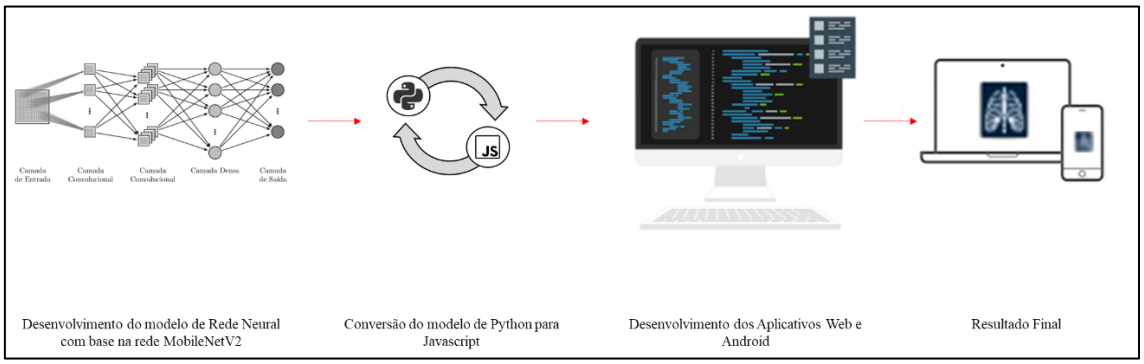
Tabela 1 - Distribuição das imagens da base de dados nas categorias de classificação.

Diretórios	Subdivisão	Quantidade de Imagens
Teste	Bactéria	115
	Vírus	58
	Normal	95
Validação	Bactéria	488
	Vírus	148
	Normal	234
Treinamento	Bactéria	2176
	Vírus	1287
	Normal	1253

3.10. Metodologia

Dando início ao desenvolvimento, a Figura 12 representa as etapas do projeto, compreendendo a criação do modelo de Rede Neural; a conversão do modelo da linguagem Python para Javascript, para então ser inserido no framework Ionic; o desenvolvimento do aplicativo Web e a exportação do aplicativo para dispositivos Android.

Figura 12 - Diagrama das etapas de desenvolvimento do projeto.



Fonte: Autor.

3.10.1. Desenvolvimento do Modelo de Rede Neural

Utilizando o Jupyter Notebook para a primeira etapa, foi necessário determinar como seria realizada a construção do modelo da Rede Neural. As redes neurais, como descritas na seção 2, podem sem compostas por diferentes camadas, onde cada composição pode alterar significativamente o resultado final de predição.

A implementação do modelo de arquitetura descrito na seção 2.12 foi realizada com o Keras, que é uma *API* de redes neurais de alto nível, escrita na linguagem Python e executada sobre a biblioteca de *Machine Learning*, o *TensorFlow*. Além da arquitetura e especialização descritas anteriormente sobre a rede MobileNetV2, ela foi escolhida em particular por exigir um menor poder de processamento, além de ocupar um menor espaço em armazenamento e que fosse de fácil conversão e importação para o nosso aplicativo Web/Android. No total esta rede possui 14MB de conteúdo, sendo uma vantagem para uso em aplicativos móveis (que prezam pela redução em espaço ocupado nos dispositivos). Em contrapartida, esta redução de tamanho, que acarreta no número reduzido de parâmetros dentro da rede, acaba comprometendo em partes a acurácia das análises.

Para iniciar a construção da rede neural foi necessário realizar a organização e o tratamento das imagens da base de dados, uma vez que a rede MobileNetV2 aceita um formato específico de entrada. Após a devida separação, as imagens são então submetidas ao pré-processamento para entrada na rede.

Utilizando o Keras, para cada diretório de entrada foram gerados lotes (ou *batches*) com uma quantidade de amostras a serem utilizadas na iteração durante a etapa de treinamento; e estes *batches* são transformados em tensores normalizados de ordem 3. Um tensor corresponde a uma generalização de vetores e matrizes para espaços de dimensões mais altas. As imagens são também redimensionadas neste processo para o tamanho esperado pela rede de 224 x 224 pixels. Especificamente, este pré-processamento dimensiona os valores de cada pixel da imagem entre -1 e 1, retornando os dados da imagem em um vetor multidimensional de elementos do mesmo tipo (*NumPy Array*).

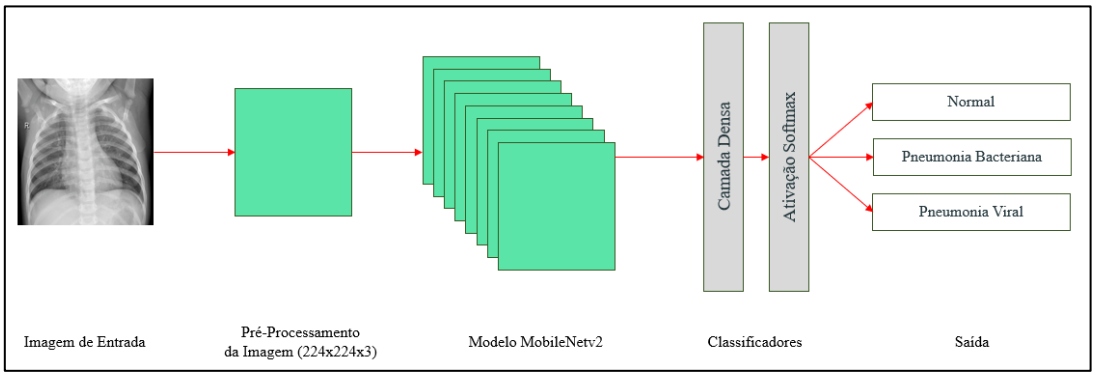
Em seguida, realizou-se a montagem do modelo de rede neural customizado. Foram removidas as 5 últimas camadas da rede MobileNetV2 por ser demonstrado em (DEEPLIZARD, 2020) que para este tipo de análise de imagens, a remoção das últimas

5 camadas se mostrou positivamente no refinamento da acurácia da rede. Com esta configuração, o modelo manterá 83 das 88 camadas originais.

Como saída, criou-se uma camada totalmente conectada com 3 possíveis alternativas, classificadas com auxílio da função de ativação Softmax. Dessa forma, é possível realizar a construção do novo Modelo de rede neural.

Seguindo com o desenvolvimento, como a arquitetura MobileNetV2 já foi previamente treinada pela base de dados *ImageNet*, ela já possui pesos pré-estipulados para suas camadas. Ainda assim, precisou-se treinar este novo modelo customizado com o banco de dados desejado do presente trabalho. Para que não fosse necessário retreinar todo o modelo, congelou-se os pesos das primeiras 23 camadas. Isso reduz o tempo necessário para treinamento de cada iteração (época). Este foi o número que melhor desempenhou no presente modelo, de forma a obter a melhor acurácia das imagens analisadas.

Figura 13 - Diagrama do processo de classificação de imagens radiográficas no modelo customizado.

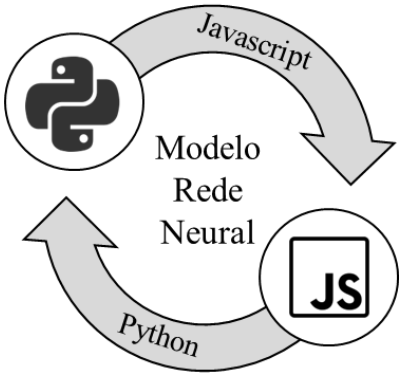


Fonte: Autor.

3.10.2. Integração via Tensorflowjs

Como mencionado na seção acima, para o desenvolvimento do modelo foi utilizada a biblioteca *Tensorflow*. Esta biblioteca possui uma variante com integração ao Javascript, a *Tensorflow.js*, e dessa forma foi possível realizar facilmente a conversão do modelo em Python (*backend*) para ser integrado à aplicação em Javascript (*frontend*). A conversão foi feita diretamente no Jupyter Notebook após o treinamento, e o modelo convertido e seus respectivos arquivos dos pesos de treinamento gerados foram então carregados na aplicação Ionic pela biblioteca *Tensorflow.js*.

Figura 14 - Conversão do modelo em Python para Javascript.



Fonte: Autor.

3.10.3. Desenvolvimento da Aplicação Web e Android

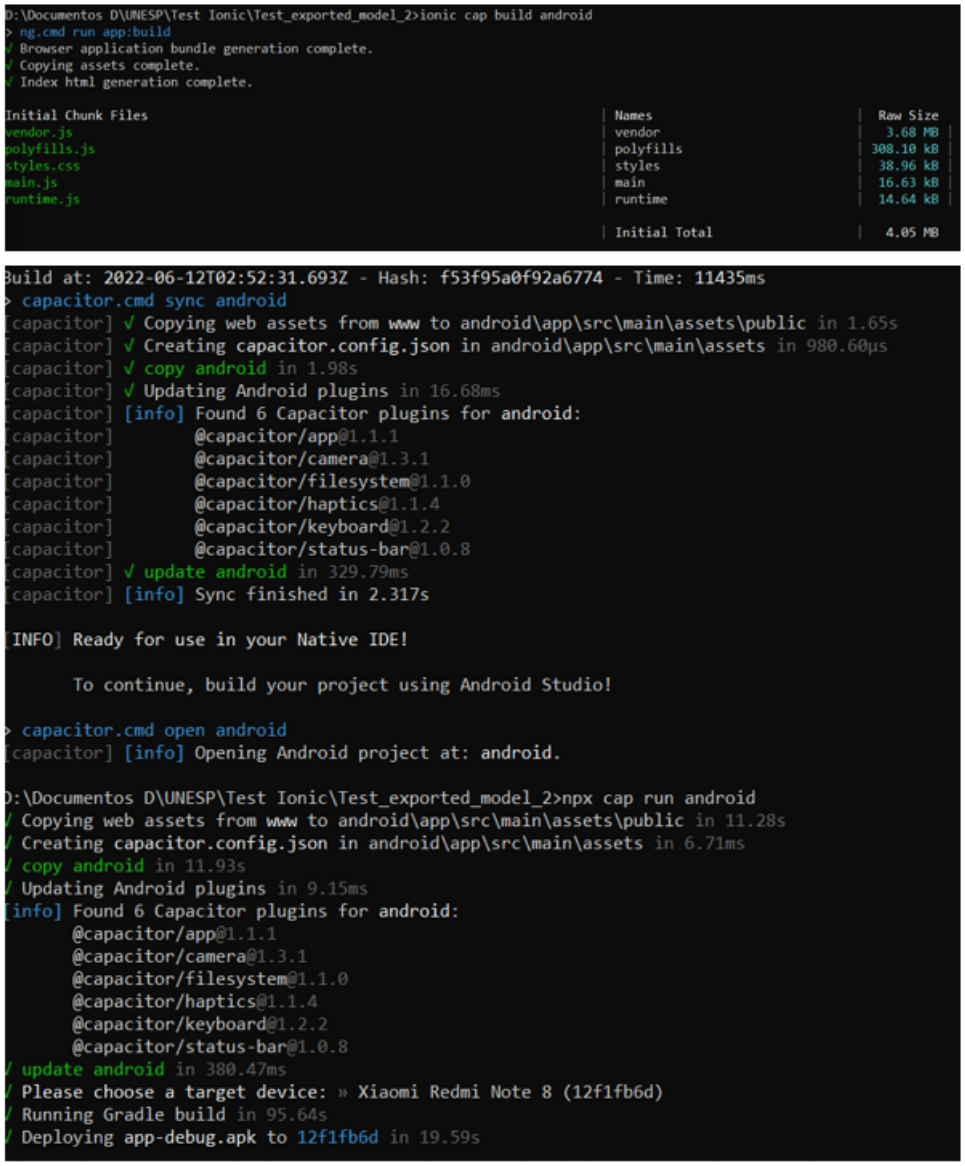
A aplicação foi feita com base em um layout pré-estabelecido pelo *framework* Ionic, e arquitetura de componentes baseados no framework Angular. A aplicação conta com uma tela interativa, para escolha da imagem, análise, e deleção da mesma, conforme demonstrado na seção 4.2 dos Resultados. O Capacitor foi utilizado nesta implementação como acesso à Câmera e Galeria do dispositivo, bem como armazenamento da imagem em um diretório próprio, atuando como um banco de dados local. Também foi utilizado para integrar o sistema Android à aplicação, gerando os arquivos necessários para a posterior criação da *APK*.

3.10.4. Criando APK para instalação em dispositivos móveis Android

Por meio da integração do Ionic com o Capacitor, foi possível a conversão da aplicação web gerada para um aplicativo para dispositivos móveis do tipo Android. O presente trabalho teve como foco modelos de dispositivos Android por ser a ferramenta disponível para teste local, porém o Capacitor também permite a conversão da aplicação para dispositivos iOS. A geração da APK pode ser realizada de duas formas: diretamente

no terminal, via linha de comando, ou pelo Android Studio. Ambas as alternativas foram testadas e resultaram em um mesmo aplicativo funcional.

Figura 15 - Gerando APK via prompt de comando Windows.



Fonte: Autor

4. RESULTADOS E DISCUSSÕES

4.1. Resultados do modelo de Rede Neural Customizado

Durante a modelagem da Rede Neural, foram testados diversos parâmetros, a fim de se encontrar o conjunto que proporcionasse o resultado mais eficaz. Os parâmetros finais escolhidos foram: o uso do otimizador *Adam* (equação de gradiente aplicado ao peso das camadas treináveis), com uma taxa de aprendizado de 0.00005, treinado por 50 épocas, com 156 interações por época. Parte do treinamento é visto na Figura 16, onde *accuracy* é a porcentagem de acertos do modelo e *loss* a somatória de todos os erros encontrados, ambos para o diretório de treinamento; *val_accuracy* e *val_loss* a porcentagem de acerto e os erros encontrados para o diretório de validação, respectivamente. Quanto mais próximo o valor dos erros de zero, mais preciso está o modelo.

Figura 16 - Treinamentos do modelo e suas saídas a cada época.

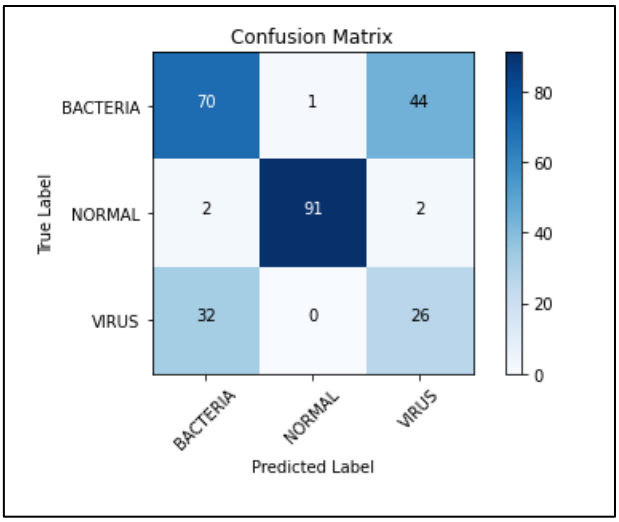
```
Epoch 1/50
156/156 - 127s - loss: 0.5688 - accuracy: 0.7660 - val_loss: 1.5656 - val_accuracy: 0.5513 - 127s/epoch - 815ms/step
Epoch 2/50
156/156 - 54s - loss: 0.3746 - accuracy: 0.8535 - val_loss: 1.8117 - val_accuracy: 0.5529 - 54s/epoch - 349ms/step
Epoch 3/50
156/156 - 123s - loss: 0.3008 - accuracy: 0.8839 - val_loss: 1.8556 - val_accuracy: 0.5513 - 123s/epoch - 787ms/step
Epoch 4/50
156/156 - 54s - loss: 0.2403 - accuracy: 0.9115 - val_loss: 1.6793 - val_accuracy: 0.5913 - 54s/epoch - 348ms/step
Epoch 5/50
156/156 - 53s - loss: 0.1773 - accuracy: 0.9452 - val_loss: 1.5776 - val_accuracy: 0.6026 - 53s/epoch - 341ms/step
Epoch 6/50
156/156 - 53s - loss: 0.1277 - accuracy: 0.9700 - val_loss: 1.5134 - val_accuracy: 0.6202 - 53s/epoch - 342ms/step
Epoch 7/50
156/156 - 53s - loss: 0.0940 - accuracy: 0.9805 - val_loss: 1.3195 - val_accuracy: 0.6410 - 53s/epoch - 339ms/step
Epoch 8/50
156/156 - 53s - loss: 0.0639 - accuracy: 0.9907 - val_loss: 1.3577 - val_accuracy: 0.6522 - 53s/epoch - 339ms/step
Epoch 9/50
156/156 - 53s - loss: 0.0452 - accuracy: 0.9946 - val_loss: 1.4326 - val_accuracy: 0.6458 - 53s/epoch - 342ms/step
Epoch 10/50
156/156 - 53s - loss: 0.0323 - accuracy: 0.9976 - val_loss: 1.2520 - val_accuracy: 0.6843 - 53s/epoch - 340ms/step

  ...

Epoch 40/50
156/156 - 53s - loss: 0.0098 - accuracy: 0.9986 - val_loss: 1.7342 - val_accuracy: 0.7035 - 53s/epoch - 340ms/step
Epoch 41/50
156/156 - 53s - loss: 0.0035 - accuracy: 0.9998 - val_loss: 1.6739 - val_accuracy: 0.7131 - 53s/epoch - 338ms/step
Epoch 42/50
156/156 - 53s - loss: 0.0056 - accuracy: 0.9992 - val_loss: 1.9181 - val_accuracy: 0.6939 - 53s/epoch - 339ms/step
Epoch 43/50
156/156 - 53s - loss: 0.0044 - accuracy: 0.9988 - val_loss: 2.6195 - val_accuracy: 0.6603 - 53s/epoch - 340ms/step
Epoch 44/50
156/156 - 53s - loss: 0.0047 - accuracy: 0.9990 - val_loss: 2.6899 - val_accuracy: 0.6314 - 53s/epoch - 340ms/step
Epoch 45/50
156/156 - 53s - loss: 0.0058 - accuracy: 0.9984 - val_loss: 2.0407 - val_accuracy: 0.6827 - 53s/epoch - 342ms/step
Epoch 46/50
156/156 - 53s - loss: 0.0067 - accuracy: 0.9990 - val_loss: 1.1445 - val_accuracy: 0.7548 - 53s/epoch - 338ms/step
Epoch 47/50
156/156 - 53s - loss: 0.0028 - accuracy: 0.9998 - val_loss: 1.2030 - val_accuracy: 0.7468 - 53s/epoch - 339ms/step
Epoch 48/50
156/156 - 53s - loss: 0.0039 - accuracy: 0.9992 - val_loss: 1.2924 - val_accuracy: 0.7340 - 53s/epoch - 338ms/step
Epoch 49/50
156/156 - 53s - loss: 0.0089 - accuracy: 0.9972 - val_loss: 1.7654 - val_accuracy: 0.6891 - 53s/epoch - 338ms/step
Epoch 50/50
156/156 - 53s - loss: 0.0073 - accuracy: 0.9980 - val_loss: 1.9539 - val_accuracy: 0.6522 - 53s/epoch - 339ms/step
```

Fonte: Autor

Pode-se inferir então que o modelo teve uma acurácia de 99, 98% para o diretório de treinamento, e uma acurácia de 75,48% para o diretório de validação. Considerando a evolução de *loss* e *val_loss*, pode-se observar que os valores decaem, e posteriormente começam a subir, indicando o *overfitting* do modelo. Mesmo com a presença do overfitting, o modelo gerado teve o melhor desempenho dentre as demais tentativas.

Na Figura 17 temos a Matriz de Confusão gerada para o nosso modelo. Pode-se observar que o modelo teve bom desempenho para a predição de imagens sem patologias e um desempenho mediano para a análise de imagens radiográficas com bactéria ou vírus por conta do *overfitting*. A diagonal do canto superior esquerdo até o canto inferior direito são as previsões corretas da rede, e os demais pontos correspondem a falsos positivos, previstos incorretamente.

Figura 17 - Matriz de Confusão do modelo gerado.

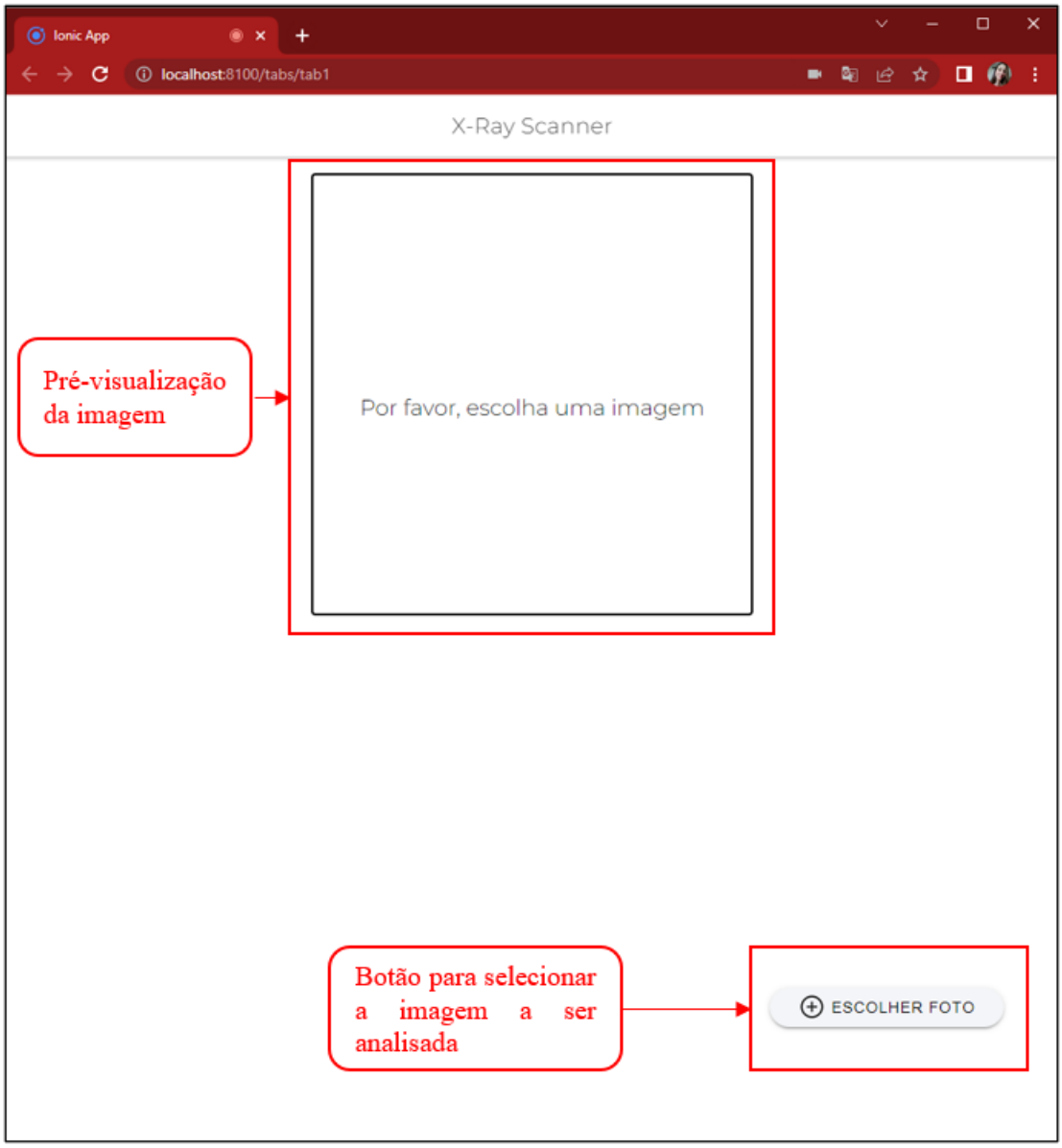


Fonte: Autor

4.2. Resultados da Aplicação Ionic Web e *APK*

A Figura 18 mostra a tela inicial da aplicação, com um espaço de pré-visualização da imagem bem como o botão para adicionar uma foto.

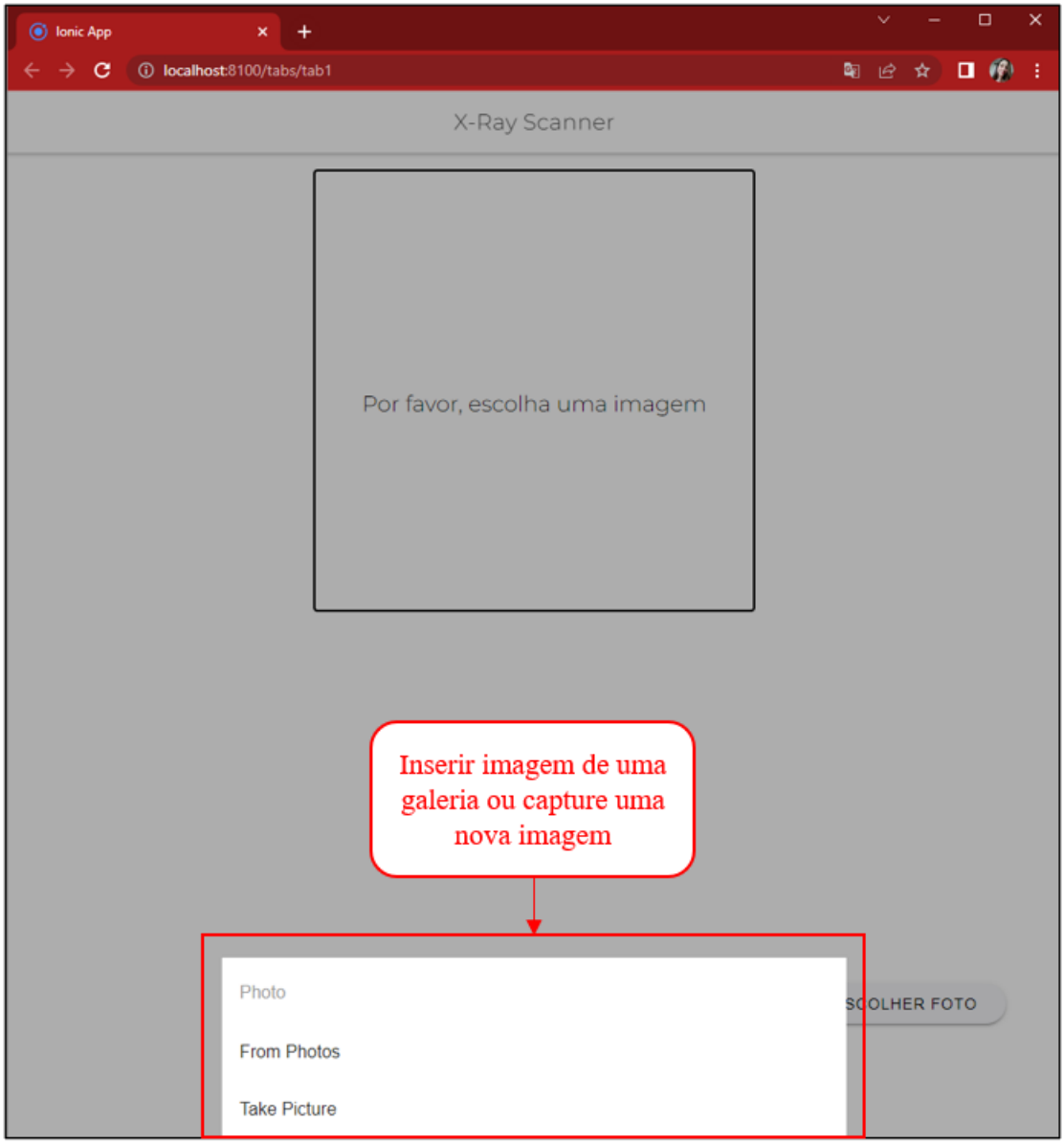
Figura 18 - Página inicial da aplicação Web.



Fonte: Autor

Ao clicar no botão para selecionar uma imagem, é exibido ao usuário duas opções de entrada: escolher uma imagem pré-existente da galeria do dispositivo/ pasta de documentos, ou capturar uma nova imagem com a câmera do dispositivo, como demonstrado na Figura 19.

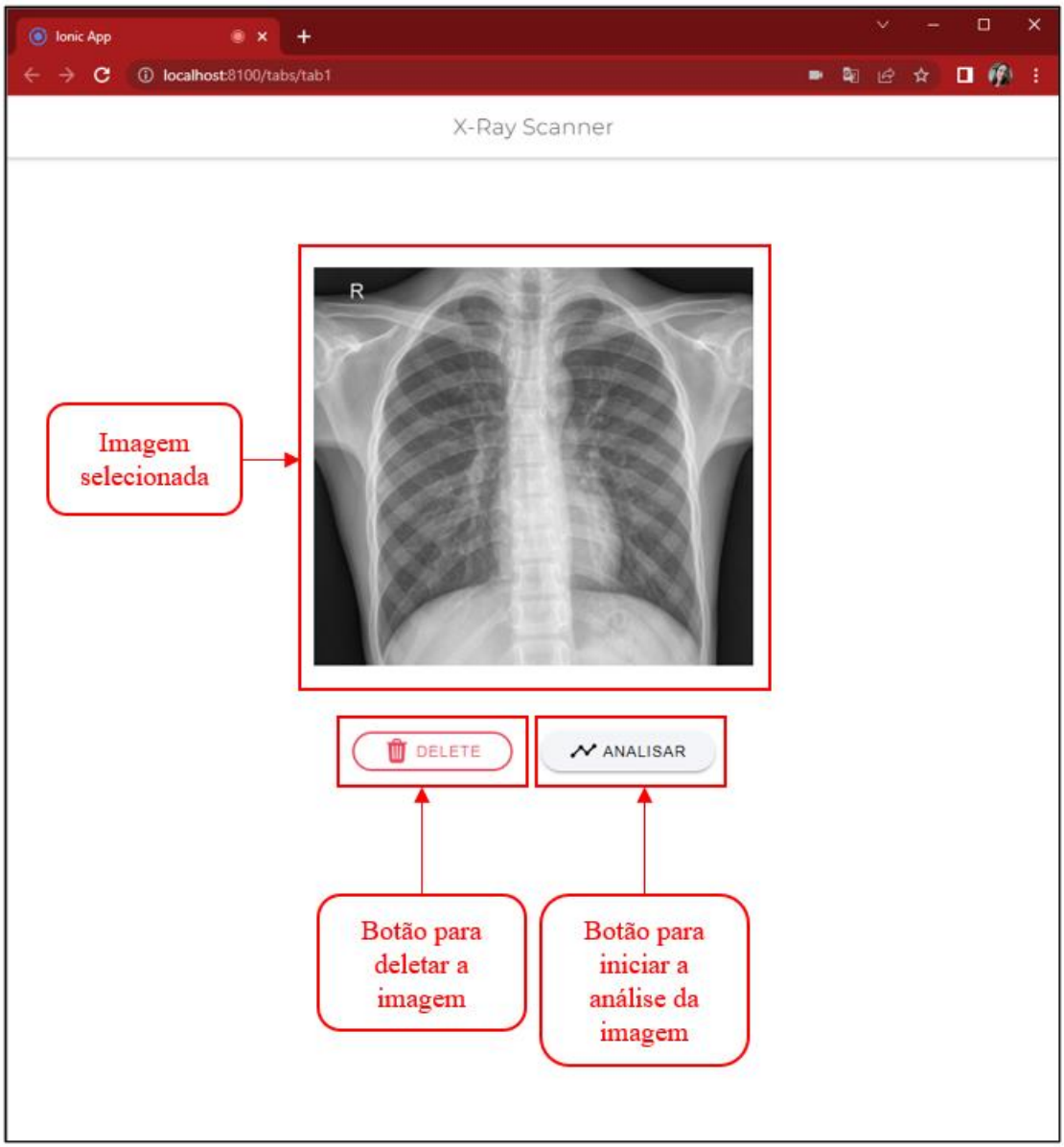
Figura 19 - Ação ao clicar no botão “Escolher Foto”.



Fonte: Autor

Selecionada a imagem, dois novos botões são apresentados ao usuário: “Deletar” e “Analisar”, apresentados na Figura 20. Intuitivamente, o botão “Deletar” remove a imagem, retornando o usuário à primeira tela, para seleção de uma nova foto. O botão “Analisar” dá início à execução do pré-processamento da imagem de entrada e sua predição pelo modelo importado.

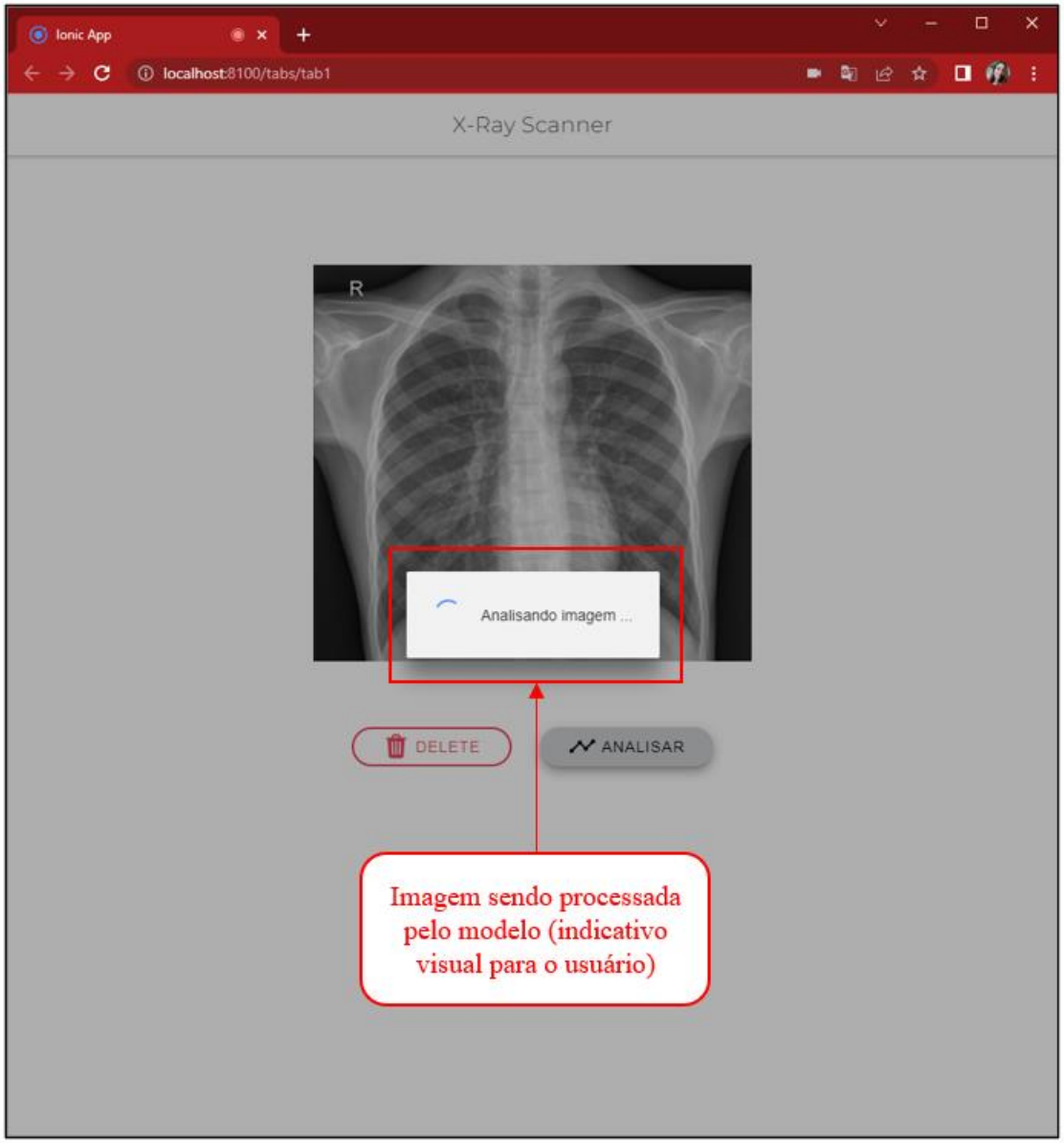
Figura 20 - Imagem escolhida pronta para análise.



Fonte: Autor

Semelhante ao pré-processamento realizado na seção 3.10.1, também se fez necessário uma modelagem da entrada. Os dados de entrada da imagem são convertidos novamente a um tensor de ordem 3, e redimensionada para 224 x 224 pixels. Após a transformação, a imagem foi inserida no modelo pré-treinado, que foi executado para obter a predição final. Esta análise pode tomar um tempo até ser totalmente concluída, então foi adicionado um modal de carregamento na interface para indicar esta etapa ao usuário, como visto na Figura 21.

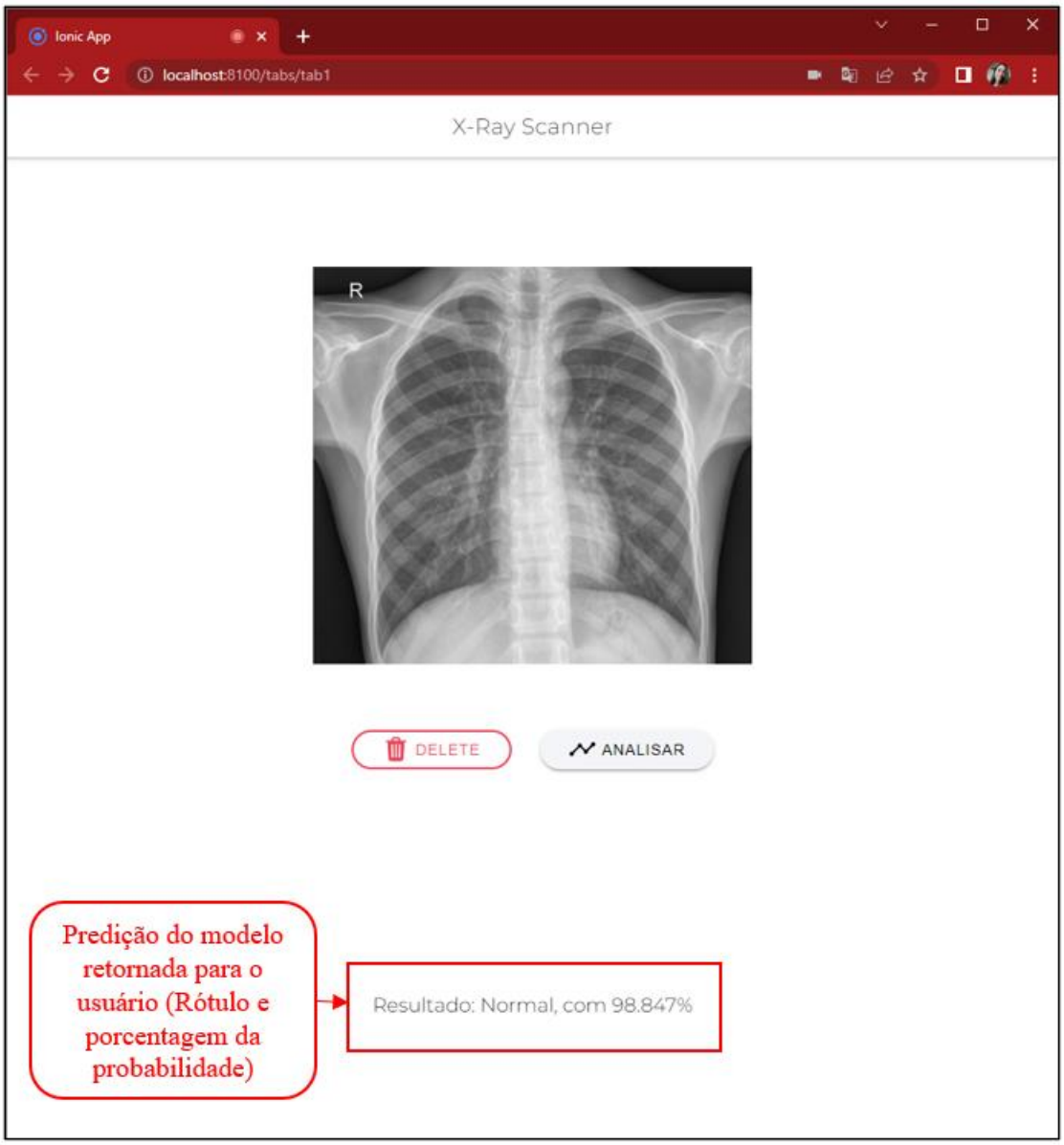
Figura 21 - Análise da Imagem.



Fonte: Autor

Como o modelo Softmax retorna a probabilidade de cada categoria em formato de um vetor, foi feito uma função laço simples para obter o maior resultado dentre as demais obtidas. O índice deste resultado foi comparado a um objeto contendo os três possíveis rótulos: Normal, Bactéria e Vírus. É retornado para o usuário então, o rótulo e a porcentagem da maior probabilidade prevista pelo modelo, indicado na Figura 22.

Figura 22 - Resultado da análise com a porcentagem da predição.



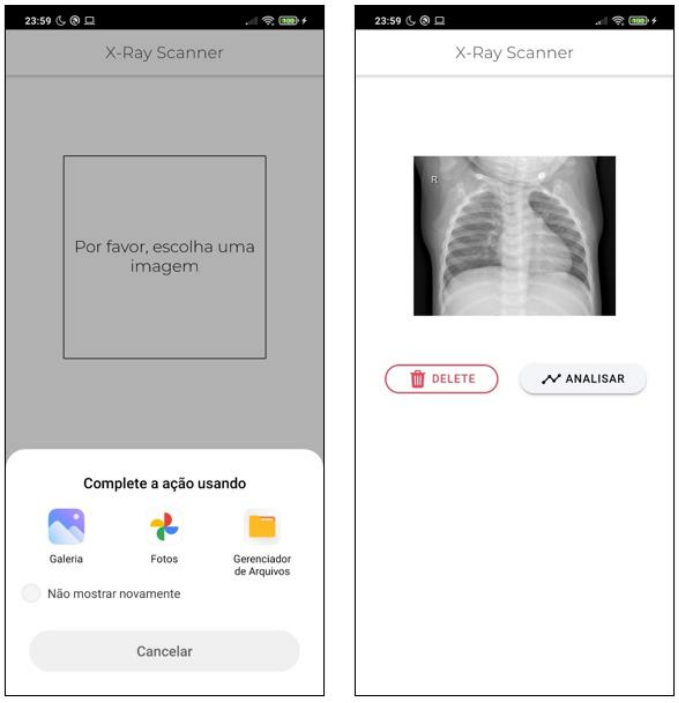
Fonte: Autor

As figuras 23 à 25 mostram a *APK* executando em um dispositivo Android, seguindo o mesmo layout da aplicação Web.

Figura 23 - Interface do aplicativo Android: Tela inicial; Tela após pressionar o botão "Escolher Foto".

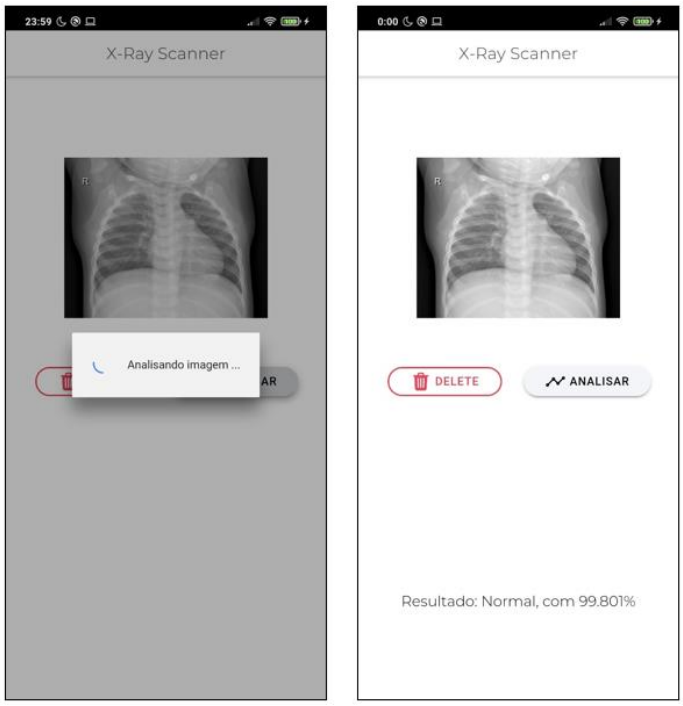
Fonte: Autor

Figura 24 - Interface do aplicativo Android: Opção de selecionar uma imagem da galeria ou capturar uma nova imagem; Tela após a seleção da imagem.



Fonte: Autor

Figura 25 - Interface do aplicativo Android: processando imagem e resultado da análise com rótulo e porcentagem.



Fonte: Autor

4.3. Discussões

Todos os testes foram realizados dentro do ambiente de desenvolvimento, utilizando uma máquina de desenvolvedor. Dentro do ambiente simulado, o aplicativo Android e a página Web foram gerados, testados e refinados.

Para os testes da aplicação *mobile*, um celular de teste foi utilizado. Nele foi instalado o aplicativo gerado via *APK*, que permite acesso à câmera ou armazenamento do dispositivo. Nesta etapa o aplicativo já está totalmente independente da aplicação Web e do ambiente de teste.

Foram testadas em torno de 100 imagens, 20 já validadas e o restante não validadas pela rede neural, a fim de inferir a porcentagem de acerto para novas imagens. Como descrito na seção 4.1, a acurácia da rede manteve-se perto de 70% para novas imagens.

5. CONCLUSÕES E TRABALHOS FUTUROS

Por meio do desenvolvimento deste trabalho, conclui-se que as redes neurais podem ser incorporadas e empregadas em diversos segmentos e atividades do mercado, não apenas focada na visão computacional. A incorporação de uma tecnologia capaz de realizar a análise e classificação de imagens de radiografias torácicas demonstrou ser uma ferramenta facilitadora para o aprendizado da avaliação de casos de patologias pulmonares, e até mesmo para um rápido diagnóstico em tempos de alta de casos, como foi o recente período da pandemia do coronavírus, onde um dos exames de acompanhamento mais realizado foi a radiografia torácica.

Conclui-se que com a utilização do sistema implementado, pode-se evitar o processo manual de avaliação das radiografias e buscar com isso diminuir a possibilidade de erro humano ao diagnosticar a presença ou não de uma patologia pulmonar. Além disso, pode-se também auxiliar profissionais em formação a aprimorar a leitura destes exames.

Para trabalhos futuros, podemos destacar alguns pontos que podem ser estudados, melhorados e incorporados ao sistema.

Para a interface de usuário:

- Adicionar uma seção de imagens previamente analisadas e sua predição.

Para a rede neural:

- Alterar a forma em que os pesos das camadas são tratados, sendo atualizados ou congelados, a fim de se obter uma saída com maior precisão;
- Alterar o número de camadas da rede, a fim de se obter uma saída com maior precisão;
- Alterar o tratamento das imagens de entrada, de forma a enriquecer os detalhes/ formas a serem analisados no processo de Downsampling;
- Treinar a rede com uma base de dados maior;
- Adicionar novas categorias de análise.

REFERÊNCIAS

ANDRADE, A. P. D. O que é Ionic? **TreinaWeb**, 2020. Disponivel em: <https://www.treinaweb.com.br/blog/o-que-e-ionic>.

CAPACITORJS. Capacitor: Cross-platform Native Runtime for Web Apps. **CapacitorJs**. Disponivel em: <https://capacitorjs.com/docs>. Acesso em: 2022.

CLAPPIS, A. M. Uma introdução as redes neurais convolucionais utilizando o Keras. **Medium**, 2019. Disponivel em: <https://medium.com/data-hackers/uma-introdu%C3%A7%C3%A3o-as-redes-neurais-convolucionais-utilizando-o-keras-41ee8dcc033e>. Acesso em: 2022.

COUTO, R. C. E. A. II Anuário Da Segurança Assistencial Hospitalar No Brasil. **IESS - Instituto de Estudos de Saúde Suplementar**, Belo Horizonte, 2018.

DEEP LEARNING BOOK. As 10 Principais Arquiteturas de Redes Neurais. **Deep Learning Book**. Disponivel em: <https://www.deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/>. Acesso em: 2022.

DEEPLIZARD. Fine-Tuning MobileNet On Custom Data Set With TensorFlow's Keras API. **Deeplizard**, 2020. Disponivel em: <https://deeplizard.com/learn/video/Zrt76AIbeh4>. Acesso em: 2022.

DEVELOPERS, G. Developers. **Developers Android**, 2021. Disponivel em: <https://developer.android.com/studio/preview/features#run-emulator-studio>. Acesso em: 14 out. 2021.

DEVMEDIA. Vue.js Tutorial. **DevMedia**, 2017. Disponivel em: <https://www.devmedia.com.br/vue-js-tutorial/38042>.

DIANA, J. Sistema Respiratório. **Toda Matéria**, 2019. Disponivel em: <https://www.todamateria.com.br/sistema-respiratorio/>. Acesso em: 2022.

DIETERLE, F. Principles of Neural Networks. **Frank Dieterle**, 2019. Disponivel em: <http://www.frank-dieterle.de/phd/2_7_1.html>. Acesso em: 2022.

EQUIPE VERITÀ. Exames de imagem para diagnóstico de doenças pulmonares. **Equipe Verità**, 2021. Disponivel em: <https://www.veritadiagnosticos.com.br/exames-de-imagem-para-diagnostico-de-doencas-pulmonares/>. Acesso em: 2022.

GUEDES, M. O que é o Angular e para que serve? **Treinaweb**, 2021. Disponivel em: <https://www.treinaweb.com.br/blog/o-que-e-o-angular-e-para-que-serve>. Acesso em: 2022.

JAEGER S, C. S. A. S. W. Y. L. P. T. G. Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. **Quant Imaging Med Surg**, 2014. Disponivel em: <https://data.lhncbc.nlm.nih.gov/public/Tuberculosis-Chest-X-ray-Datasets/Montgomery-County-CXR-Set/MontgomerySet/index.html>. Acesso em: 2022.

KERMANY, D. S. E. A. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. **CellPress**, v. 172, n. 5, 2018. ISSN P1122-1131.E9. Disponivel em: <https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5>.

KRIGER, D. O que é Python, para que serve e por que aprender? **Kenzie**, 2022. Disponivel em: <https://kenzie.com.br/blog/o-que-e-python/>. Acesso em: 2022.

LET'S CODE. Introdução ao Jupyter Notebook. **Let's Code**, 2019. Disponivel em: <https://letscode.com.br/blog/introducao-ao-jupyter-notebook>. Acesso em: 2022.

MINISTÉRIO DA SAÚDE. **Saúde Brasil 2018: Uma análise da situação de saúde e das doenças e agravos crônicos: desafios e perspectivas**. Ministério da Saúde. Brasília, p. 80. 2019. (ISBN 978-85-334-2701-3).

MUHAMAD Y., E. A. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. **Journal of Physics: Conference Series**, n. 1201 012052, 2019. Acesso em: 2022.

OLIVEIRA, J. Redes Neurais Artificiais. **Jornal PET News - Grupo PET Computação**, 2011. Disponivel em: <http://www.dsc.ufcg.edu.br/~pet/jornal/setembro2011/materias/informatica.html>. Acesso em: 2022.

OMRON HEALTHCARE. Entenda tudo sobre as principais doenças respiratórias e como se livrar das crises. **OMRON Brasil**, 2020. Disponivel em: <https://conteudo.omronbrasil.com/doencas-respiratorias/>. Acesso em: 2022.

ORACLE. O que é inteligência artificial (IA)? Saiba mais Sobre Inteligência Artificial. **Oracle**. Disponivel em: <https://www.oracle.com/br/artificial-intelligence/what-is-ai/>. Acesso em: 2022.

PEREIRA, F. HTML, CSS e Javascript - Entendendo melhor a base da programação Front-End. **Apex Ensico**, Julho 2018. Disponivel em: <https://apexensino.com.br/base-da-programacao-front-end/#:~:text=O%20HTML%20%C3%A9%20a%20base,fosse%20o%20esp%C3%ADrito%20do%20corpo)>.

RAFAEL SAKURAI. Implementando a estrutura de uma Rede Neural Convolucional utilizando o MapReduce do Spark. **Rafael Sakurai**, 2017. Disponivel em: <https://www.sakurai.dev.br/cnn-mapreduce/>. Acesso em: 2022.

REIS, B. Redes Neurais - Funções de Ativação. **UFOP - Universidade Federal de Ouro Preto**, 2016. Disponivel em: <http://www2.decom.ufop.br/imobilis/redes-neurais-funcoes-de-ativacao/#:~:text=A%20fun%C3%A7%C3%A3o%20de%20ativa%C3%A7%C3%A3o%20softmax,maior%20indica%20a%20classe%20vencedora.>. Acesso em: 2022.

REMESSA ONLINE. Firebase: descubra para que serve, como funciona e como usar. **Remessa Online**, Novembro 2021. Disponivel em: <https://www.remessaonline.com.br/blog/firebase-descubra-para-que-serve-como-funciona-e-como-usar/>.

SANDLER, M. . H. A. MobileNetV2: The Next Generation of On-Device Computer Vision Networks. **Google AI Blog**, 2018. Disponivel em: <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>. Acesso em: 2022.

SANTOS, M. K. A radiografia simples de tórax. **Medicina (Ribeirão Preto)**, v. 52, p. 1-3, 2019. Disponivel em: <https://www.revistas.usp.br/rmrp/article/view/154871>. Acesso em: Março 2022.

UTSCH, K. G. **Uso de Redes Neurais Convolucionais para classificação de imagens digitais de lçesões de pele**. Universidade Federal do Espírito Santo. Vitória, p. 46 - 52. 2018.

WADA, D. T.; RODRIGUES, J. A. H.; SANTOS, M. K. Aspectos técnicos e roteiro de análise da radiografia de tórax. **Medicina (Ribeirão Preto)**, p. 5-15, 2019. Disponivel em: <https://www.revistas.usp.br/rmrp/article/view/154763>. Acesso em: Março 2022.