

UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Felipe de Moraes Soares

**UMA APLICAÇÃO MOBILE PARA CLASSIFICAÇÃO DE TIPOS DE  
LESÃO DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**

Santa Maria, RS  
2022

**Felipe de Moraes Soares**

**UMA APLICAÇÃO MOBILE PARA CLASSIFICAÇÃO DE TIPOS DE LESÃO DE PELE  
UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Engenharia de Computação**.

ORIENTADOR: Prof. Daniel Welfer

Santa Maria, RS  
2022

**Felipe de Moraes Soares**

**UMA APLICAÇÃO MOBILE PARA CLASSIFICAÇÃO DE TIPOS DE LESÃO DE PELE  
UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Engenharia de Computação**.

**Aprovado em 3 de setembro de 2021:**

---

**Daniel Welfer, Dr. (UFSM)**  
(Presidente/Orientador)

Santa Maria, RS  
2022

## **AGRADECIMENTOS**

*Aos meus pais, Marizete Lemes de Moraes Soares e Paulo Antunes Soares, por todo carinho, apoio e por serem meu maior exemplo. Obrigado terem passado os mais valiosos ensinamentos que pude ter.*

## **RESUMO**

### **UMA APLICAÇÃO MOBILE PARA CLASSIFICAÇÃO DE TIPOS DE LESÃO DE PELE UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**

**AUTOR:** Felipe de Moraes Soares

**ORIENTADOR:** Daniel Welfer

O câncer de pele é o tipo de câncer de maior incidência no Brasil, correspondendo cerca de 30% dos tumores malignos registrados no país, no entanto, apresenta um alto índice de cura quando detectado precocemente. Com o avanço da tecnologia, novas ferramentas são apresentadas na literatura para auxiliar na detecção precoce da doença. O presente trabalho apresenta uma aplicação mobile para classificação de tipos de câncer de pele utilizando redes neurais convolucionais (CNN). O objetivo é implementar de forma funcional uma aplicação, onde a parte mobile capta imagens e envia para um algoritmo de classificação, que através de redes neurais convolucionais indica o tipo de lesão de pele e retorna essa informação para usuário em seu smartphone. As arquiteturas de redes neurais convolucionais estudadas ao longo do projeto foram VGG11, ResNet50 e DenseNet121, realizando o treinamento das mesmas e comparando em sua fase de validação qual modelo obteve melhor desempenho. A rede DenseNet121 alcançou melhores resultados sobre as demais, com uma precisão de 91,24% na fase de validação. Uma vez estabelecida qual rede neural será aplicada no sistema, foi desenvolvida a aplicação mobile, para a captação de imagens, utilizando a ferramenta Kivy na linguagem Python e API Rest para realizar a troca de informações entre as partes frontend e backend. O aplicativo se mostrou funcional, após testes realizados correspondendo a performance da rede já treinada e se mostrando útil para o fomento da discussão sobre o tema proposto.

**Palavras-chave:** Câncer de pele. CNN. Aplicativo Mobile. Redes Neurais

## **ABSTRACT**

### **MOBILE APPLICATION FOR CLASSIFICATION OF TYPES OF SKIN LESION USING CONVOLUTIONAL NEURAL NETWORKS**

**AUTHOR:** Felipe de Moraes Soares

**ADVISOR:** Daniel Welfer

Skin cancer is the type of cancer that has the highest incidence in Brazil, corresponding to about 30% of malignant tumors registered in the country, but with high chances of cure rate when detected early. With technology evolution, new tools are presented in the literature to assist in the early detection of the disease. This project presents a mobile application for classification of skin cancer types using convolutional neural networks (CNN). The objective is to functionally implement an application, where the mobile part captures images and sends them to a classification algorithm, which through convolutional neural networks indicates the type of skin lesion and returns this information to the user on their smartphone. The architectures of convolutional neural networks studied throughout the project were VGG11, ResNet50 and DenseNet121, performing their training and comparing in their validation phase which model obtained better results. The DenseNet121 network achieved better results than the others, with an accuracy of 91.24% in the validation phase. Once established which neural network will be applied in the system, the mobile application was developed to capture images, using the Kivy tool in the Python language and Rest API to exchange information between the parts, which we can call frontend and backend. The application proved to be functional, after tests carried out corresponding to the performance of the already trained network and proving to be useful for promoting the discussion on the proposed topic.

**Keywords:** Skin Cancer. CNN. Software Mobile. Deep Learning.

## LISTA DE FIGURAS

Figura 2.1 – Fases de um algoritmo de ML .....	14
Figura 2.2 – Representação Inteligência Artificial .....	16
Figura 2.3 – Comparação neurônio biológico x artificial .....	16
Figura 2.4 – Rede Neural Artificial .....	17
Figura 2.5 – Uso na Inteligência Artificial na medicina .....	18
Figura 2.6 – Arquitetura de uma Rede Neural Convolucional .....	19
Figura 2.7 – Disposição espacial de entrada de Rede Neural Convolucional .....	20
Figura 2.8 – Mapa de ativação na primeira camada convolucional .....	20
Figura 2.9 – Caminho percorrido pelo <i>kernel</i> .....	21
Figura 2.10 – Exemplo de <i>Max pooling</i> e <i>Average pooling</i> .....	21
Figura 2.11 – Exemplo de um sistema com API REST .....	30
Figura 2.12 – Exemplo de Melanoma .....	31
Figura 2.13 – Exemplo de Carcinoma .....	32
Figura 2.14 – Exemplo de Nevo Melanócito .....	32
Figura 2.15 – Exemplo de Queratose Seborreica .....	33
Figura 2.16 – Exemplo de Queratose Actínicas .....	34
Figura 2.17 – Exemplo de Dermatofibroma .....	34
Figura 2.18 – Exemplo de Lesões Vasculares .....	35
Figura 4.1 – Células de Código Google Colaboratory .....	49
Figura 4.2 – Vincular Google Drive ao Google Colaboratory .....	50
Figura 4.3 – Balanceamento de classes do conjunto de treino .....	53
Figura 4.4 – Código para salvar rede neural treinada .....	55
Figura 4.5 – Código para carregar rede neural treinada .....	57
Figura 4.6 – Código de classificação de imagem fornecida .....	57
Figura 4.7 – Código para criação de um botão .....	58
Figura 4.8 – Layout do botão na interface gráfica .....	59
Figura 4.9 – Código para criação de uma rota para API REST .....	60
Figura 4.10 – Exemplo de um POST Request .....	60
Figura 4.11 – Emprego da API Rest .....	61
Figura 5.1 – Treinamento VGG .....	63
Figura 5.2 – Teste VGG .....	63
Figura 5.3 – Treinamento ResNet .....	64
Figura 5.4 – Teste ResNet .....	65
Figura 5.5 – Treinamento DenseNet .....	66
Figura 5.6 – Teste DenseNet .....	66
Figura 5.7 – Interfaces gráficas iniciais do aplicativo .....	68
Figura 5.8 – Interface gráfica de login do aplicativo .....	68

## LISTA DE TABELAS

Tabela 2.1 – Camadas de uma rede VGG.....	24
Tabela 2.2 – Definição do bloco Bottleneck .....	25
Tabela 2.3 – Camadas de uma rede Resnet50.....	26
Tabela 2.4 – Definição do bloco DenseLayer.....	27
Tabela 2.5 – Definição do bloco Transição .....	27
Tabela 2.6 – Camadas de uma rede Densenet.....	28
Tabela 3.1 – Pesquisa bibliográfica na base de dados Scopus .....	37
Tabela 3.2 – Pesquisa bibliográfica na base de dados Wiley.....	37
Tabela 3.3 – Pesquisa bibliográfica na base Emerald.....	38
Tabela 3.4 – Pesquisa bibliográfica na base de dados Proquest.....	39
Tabela 3.5 – Pesquisa bibliográfica na base de dados IEEE.....	39
Tabela 3.6 – Pesquisa bibliográfica de dados Scielo.....	40
Tabela 3.7 – Pesquisa bibliográfica na base de dados PubMed.....	40
Tabela 3.8 – Pesquisa bibliográfica na base de dados ACM.....	41
Tabela 3.9 – Pesquisa bibliográfica na base de dados SPIE.....	41
Tabela 4.1 – Tipos de lesões <i>dataset</i> HAM10000 .....	51
Tabela 4.2 – Exemplo de informações presentes no dataframe para o treinamento ...	52
Tabela 4.3 – Conjunto de Treino Balanceado.....	53
Tabela 4.4 – Valores dos Hiperparâmetros das CNN.....	55
Tabela 5.1 – Comparativo entre as CNN Utilizadas no Projeto.....	67



## LISTA DE ABREVIATURAS E SIGLAS

*API* Application Programming Interface

*CNN* Convolutional Neural Network

*CPU* Central Processing Units

*DenseNet* Densely Connected Convolutional Network

*FC* Fully Connected Layer

*FNN* Feedforward Neural Network

*GAN* Generative Adversarial Network

*GPU* Graphic Processing Units

*HTTP* Hyper Text Transfer Protocol

*IA* Inteligência Artificial

*ML* Machine Learning

*REST* Representational State Transfer

*RNA* Rede Neural Artificial

*RNN* Recurrent Neural Network

*SGD* Stochastic Gradient Descent

*SO* Sistema Operacional

*VGG* Visual Geometry Group

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>11</b>
1.1	OBJETIVO .....	12
1.2	ORGANIZAÇÃO DO TRABALHO .....	12
<b>2</b>	<b>REFERENCIAL TEÓRICO.....</b>	<b>13</b>
2.1	INTELIGÊNCIA ARTIFICIAL.....	13
2.2	MACHINE LEARNING .....	13
<b>2.2.1</b>	<b>Pré-processamento .....</b>	<b>14</b>
<b>2.2.2</b>	<b>Treinamento .....</b>	<b>15</b>
<b>2.2.3</b>	<b>Teste .....</b>	<b>15</b>
2.3	DEEP LEARNING .....	15
2.4	INTELIGÊNCIA ARTIFICIAL NA MEDICINA .....	17
2.5	REDES NEURAIS CONVOLUCIONAIS .....	18
<b>2.5.1</b>	<b>Camadas de uma Rede Neural Convolutacional .....</b>	<b>18</b>
2.5.1.1	<i>Camadas de Convolução .....</i>	<i>19</i>
2.5.1.2	<i>Pooling .....</i>	<i>20</i>
2.5.1.3	<i>Camada Totalmente Conectada .....</i>	<i>21</i>
2.6	TÉCNICA DE TRANSFER LEARNING .....	22
2.7	TREINAMENTO DE UMA REDE NEURAL CONVOLUCIONAL .....	22
<b>2.7.1</b>	<b>Backpropagation .....</b>	<b>23</b>
<b>2.7.2</b>	<b>Stochastic Gradient Descent .....</b>	<b>23</b>
2.8	ARQUITETURAS UTILIZADAS NO PROJETO .....	23
<b>2.8.1</b>	<b>VGG .....</b>	<b>23</b>
<b>2.8.2</b>	<b>ResNet .....</b>	<b>25</b>
<b>2.8.3</b>	<b>DenseNet .....</b>	<b>26</b>
2.9	SISTEMA OPERACIONAL ANDROID .....	28
2.10	API REST .....	29
2.11	TIPO DE LESÕES DE PELE ABORDADAS NO PROJETO .....	30
<b>2.11.1</b>	<b>Melanoma .....</b>	<b>30</b>
<b>2.11.2</b>	<b>Carcinoma .....</b>	<b>31</b>
<b>2.11.3</b>	<b>Nevos Melanócitos .....</b>	<b>32</b>
<b>2.11.4</b>	<b>Queratose Seborreica .....</b>	<b>33</b>
<b>2.11.5</b>	<b>Queratose Actínicas .....</b>	<b>33</b>
<b>2.11.6</b>	<b>Dermatofibroma .....</b>	<b>34</b>
<b>2.11.7</b>	<b>Lesões Vasculares .....</b>	<b>35</b>
<b>3</b>	<b>REVISÃO BIBLIOGRÁFICA .....</b>	<b>36</b>
3.1	PROTOCOLO DE PESQUISA .....	36
<b>3.1.1</b>	<b>Pesquisa na Base de Dados Scopus .....</b>	<b>36</b>
<b>3.1.2</b>	<b>Pesquisa na Base de Dados Wiley .....</b>	<b>37</b>
<b>3.1.3</b>	<b>Pesquisa na Base de Dados Emerald .....</b>	<b>38</b>
<b>3.1.4</b>	<b>Pesquisa na Base de Dados Proquest .....</b>	<b>38</b>
<b>3.1.5</b>	<b>Pesquisa na Base de Dados IEEE .....</b>	<b>39</b>
<b>3.1.6</b>	<b>Pesquisa na Base de Dados Scielo .....</b>	<b>39</b>
<b>3.1.7</b>	<b>Pesquisa na Base de Dados PubMed .....</b>	<b>40</b>
<b>3.1.8</b>	<b>Pesquisa na Base de Dados ACM .....</b>	<b>41</b>
<b>3.1.9</b>	<b>Pesquisa na Base de Dados SPIE .....</b>	<b>41</b>

3.2	TRABALHOS RELACIONADOS .....	42
3.2.1	<b>Redes Neurais Convolucionais na Detecção de Lesões de Pele .....</b>	<b>42</b>
3.2.1.1	<i>Publicações com Utilização do Dataset HAM10000 .....</i>	<i>42</i>
3.2.1.2	<i>Publicações com Utilização de Diferentes Datasets .....</i>	<i>44</i>
3.2.2	<b>Aplicações para Detecção Automática de Lesões de Pele .....</b>	<b>45</b>
4	<b>DESENVOLVIMENTO .....</b>	<b>47</b>
4.1	DEEP LEARNING .....	47
4.1.1	<b>Ambiente de Desenvolvimento .....</b>	<b>47</b>
4.1.2	<b>Utilização do Google Colaboratory .....</b>	<b>48</b>
4.1.3	<b>Banco de dados .....</b>	<b>50</b>
4.1.3.1	<i>Preparação do Banco de Dados .....</i>	<i>51</i>
4.1.4	<b>Treinamento e Teste .....</b>	<b>53</b>
4.2	DEPLOY DO SOFTWARE .....	56
4.2.1	<b>Ambiente de Desenvolvimento .....</b>	<b>56</b>
4.2.2	<b>Carregamento do Modelo de CNN Treinado .....</b>	<b>57</b>
4.2.3	<b>Desenvolvimento da Aplicação Mobile .....</b>	<b>58</b>
4.2.3.1	<i>Desenvolvimento Interface gráfica com Kivy .....</i>	<i>58</i>
4.2.4	<b>API REST .....</b>	<b>59</b>
4.2.4.1	<i>Desenvolvimento do API REST com Flask .....</i>	<i>59</i>
5	<b>RESULTADOS .....</b>	<b>62</b>
5.1	TESTE DAS REDES NEURAI CONVOLUCIONAIS.....	62
5.1.1	<b>Resultados da Arquitetura VGG .....</b>	<b>62</b>
5.1.2	<b>Resultados da Arquitetura ResNet .....</b>	<b>64</b>
5.1.3	<b>Resultados da Arquitetura DenseNet.....</b>	<b>65</b>
5.2	COMPARAÇÃO DAS REDES NEURAI CONVOLUCIONAIS .....	67
5.3	SOFTWARE MOBILE .....	67
6	<b>CONCLUSÃO .....</b>	<b>70</b>
	<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>72</b>

## 1 INTRODUÇÃO

O avanço da tecnologia ligada a área médica, tem possibilitado aquisição de imagens para o auxílio de diagnósticos de doenças, criando vastos conjuntos de dados referente a diferentes tipos de doenças. Devido a facilidade de aplicações de aprendizado de máquinas para reconhecer padrões e manipular grandes bases de dados, o emprego destas técnicas tem trazido resultados relevantes para a colaboração do diagnóstico de patologias. Novas técnicas com a utilização de inteligência artificial surgem na literatura com grande frequência, dentre as doenças estudadas estão patologias referentes a lesões de pele, como o câncer de pele que pode ser diagnosticado através de padrões em manchas presentes no corpo.

Segundo o Instituto Nacional de Câncer (2021), o câncer de pele é o tipo de câncer mais frequente no Brasil, correspondendo cerca de 30% dos tumores malignos registrados no país. Este sendo mais comum em pessoas brancas, acima de 40 anos de idade, geralmente causado em virtude de constantes exposições a raios solares ao longo da vida. Quando detectado em seus estágios iniciais, as chances de cura são altas, pois é possível evitar que o tumor se desenvolva criando metástases difíceis de eliminar.

Existem diferentes tipos de câncer de pele, podendo ser dividido em melanoma e não melanoma. O melanoma tem um índice de mortalidade maior, tendo uma estimativa de 8450 novos casos e 1978 óbitos no ano de 2020 (INSTITUTO NACIONAL DE CÂNCER, 2022a). O câncer de pele apresenta diferentes tipos de tumores, esses diferentes tipos apresentam características distintas, que são utilizadas em sua classificação.

O aprendizado de máquina tem se mostrado útil nas inovações recentes no ramo na medicina, devido a facilidade dessas técnicas conseguirem manusear grandes bases de dados (PAIXÃO et al., 2022). O *Deep Learning* sendo um ramo dos métodos de aprendizado de máquina, tem se destacado abordando Redes Neurais Convolucionais no aprendizado de padrões de imagens. Redes Neurais Convolucionais, conseguem aprender padrões a partir de determinados conjuntos de imagens para a resolução de problemas de segmentação ou classificação.

Neste contexto, o presente projeto realiza um estudo de Redes Neurais Convolucionais para classificação de imagens de lesões de pele, afim de desenvolver uma aplicação mobile para auxiliar a detecção de sinais positivos de patologias. Mostrando que, com o avanço da tecnologia e a disponibilidade de inúmeras ferramentas e técnicas novas de computação, podemos construir aplicações com finalidades benevolentes, ajudando assim a ter uma sociedade mais consciente e com mais chances de acesso à informação. Também trazer benefícios, não somente na área da saúde, mas nas mais diversas áreas que abrangem o cotidiano da população.

## 1.1 OBJETIVO

O objetivo deste trabalho é estudar diferentes abordagens de Redes Neurais Convolucionais afim de desenvolver um software *mobile* capaz de captar imagens de lesões de pele, e através de meios computacionais, realizar o *deploy* da rede neural que apresentou o melhor desempenho no estudo para gerar a classificação do tipo de lesão de pele.

Os objetivos específicos, que são detalhamentos do objetivo geral, são:

1. Estudo da Literatura de Redes Neurais Convolucionais;
2. Treinar Modelos de Redes Convolucionais diferentes;
3. Comparar os resultados de precisão dos modelos identificando a arquitetura com o melhor resultado para o conjunto de dados proposto;
4. Desenvolvimento de um software *mobile* para obtenção da imagem da lesão de pele;
5. Integração entre a aplicação e o algoritmo de classificação;
6. Analise dos resultados obtidos;

## 1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em 6 capítulos, de forma que o Capítulo 1 apresenta uma introdução a respeito do tema do trabalho, incluindo a motivação, a definição do problema, a justificativa e os objetivos do mesmo.

No Capítulo 2 é apresentada a base teórica utilizada para o desenvolvimento do trabalho. Nesse são incluídos conteúdos como a definição de aprendizado de máquina, redes neurais e conceitos referente aplicações *mobile*.

A revisão bibliográfica de artigos e trabalhos que abordam temas ligados ao do presente trabalho é apresentada e discutida no Capítulo 3.

No Capítulo 4 é definida a metodologia utilizada para o desenvolvimento deste trabalho.

O Capítulo 5 apresenta os resultados obtidos. Ele contém os resultados das simulações realizadas para cada estudo de caso.

Por fim, o Capítulo 6 contém as conclusões alcançadas com a finalização deste trabalho.

## 2 REFERENCIAL TEÓRICO

Este capítulo tem por finalidade definir os conceitos teóricos necessário para o entendimento do trabalho, tendo como objetivo apresentar os referenciais utilizados ao longo do processo a partir assuntos que foram estudados e pesquisados em diversas fontes, bem como os conhecimentos adquiridos durante o curso de Engenharia de Computação, possibilitando a realização deste projeto. Será discorrido sobre alguns conceitos essenciais, tais como câncer de pele, redes neurais, aprendizado profundo e redes neurais convolucionais.

### 2.1 INTELIGÊNCIA ARTIFICIAL

Tendo início após a Segunda Guerra Mundial, a Inteligência Artificial (IA) é um ramo da computação que utiliza algoritmos que são capazes de identificar um problema ou uma tarefa a ser realizada e, através de análise de dados, tomar decisões simulando a capacidade humana. IA abrange uma grande variedade de áreas e subcampos de aplicação, desde tomadas de decisões em um jogo de xadrez, até criação de poemas e auxílio a diagnóstico de doenças.

McCarrthy (2017) define IA como uma ciência e uma engenharia “É a ciência e engenharia de fazer máquinas inteligentes, especialmente programas de computador inteligentes. Está relacionado à tarefa semelhante de usar computadores para entender a inteligência humana, mas a Inteligência Artificial não precisa se limitar a métodos biologicamente observáveis”.

### 2.2 MACHINE LEARNING

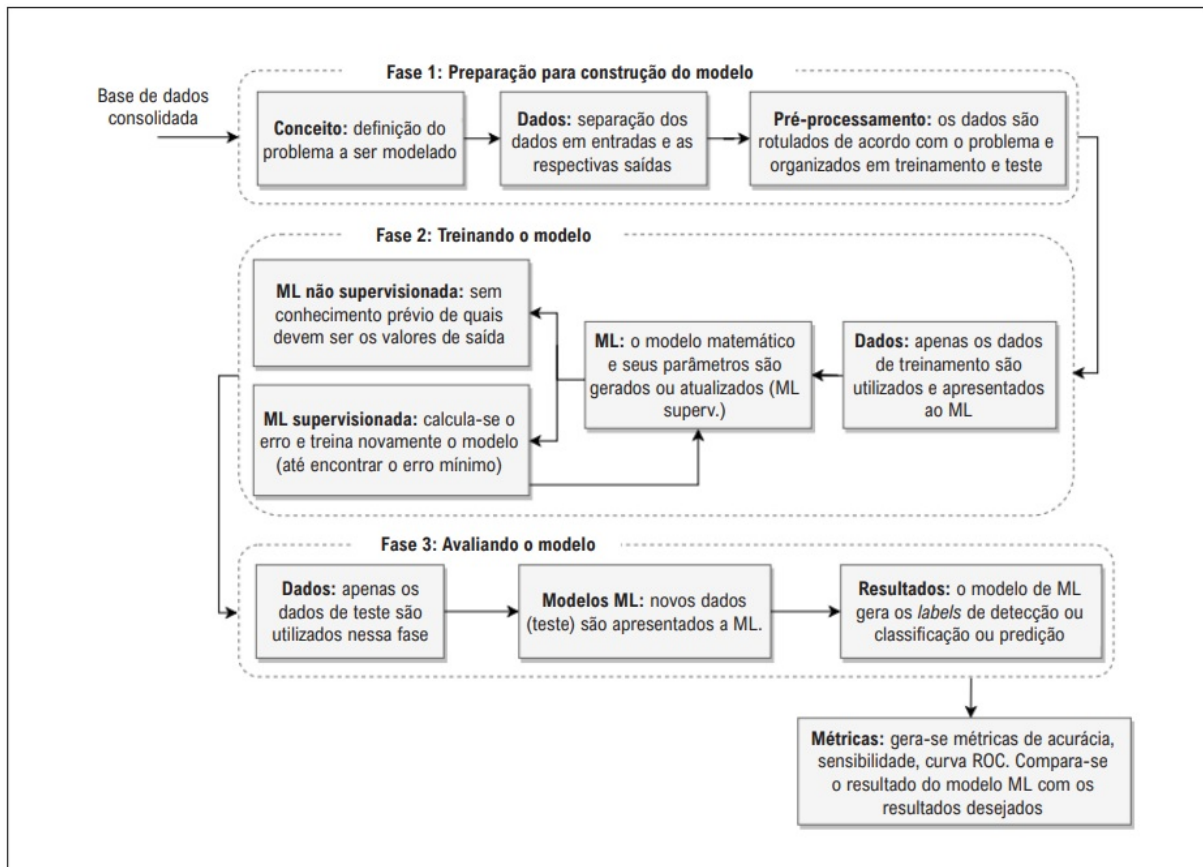
*Machine Learning* ou aprendizado de máquina (ML), é um ramo da IA que explora o estudo e a criação de algoritmos computacionais através do aprendizado por uma base de dados (GIRIRAJAN; CAMPBELL; EICHLER, 2011). Tendo como objetivo, gerar um sistema capaz de extrair característica de dados fornecidos e gerar um modelo que detecte, classifique ou gere uma previsão sobre algum problema apresentado.

Utilizando algoritmos com os conceitos de IA, o aprendizado de máquina busca uma interseção de técnicas matemáticas e estatísticas com passos computacionais. Visando encontrar padrões em um conjunto de variáveis, com o objetivo de prever um resultado que é do interesse aplicável (PAIXÃO et al., 2022).

Sendo assim, modelos de ML podem realizar iterações diversas vezes, refinando

os resultados gerados e assim estabelecendo um padrão preciso, onde novos dados, que ainda não foram vistos pelo algoritmo, possam ser classificados com alta confiabilidade e precisão. O desenvolvimento de métodos de ML pode ser dividido em três fases: pré-processamento, treinamento e teste, suas especificações podem ser demonstradas conforme a Figura 2.1.

Figura 2.1 – Fases de um algoritmo de ML



Fonte: (PAIXÃO et al., 2022).

### 2.2.1 Pré-processamento

A primeira fase do algoritmo de ML, consiste em definir o problema a ser atacado, coletando uma base de dados que contenha as informações correspondentes para ensinar o modelo sobre o problema proposto. A base de dados, que também pode ser chamada de *dataset*, deve ser dividida em um montante de dados para treinamento e um montante de dados para teste. Geralmente se usa um percentual maior dos dados para o treinamento do modelo de ML, e a fatia menor do mesmo usado para teste deste modelo.

### 2.2.2 Treinamento

Na fase de treinamento, o aprendizado do modelo de ML pode ocorrer de forma supervisionada ou não, aprendizado supervisionado é feito de modo que as amostras de dados já contêm sua classificação atribuída enquanto o não supervisionado é capaz de organizar as informações e aprender sem o dado amostrado ter uma classificação estabelecida.

### 2.2.3 Teste

Na fase de teste, que também pode ser chamada de fase de teste ou avaliação, o modelo treinado é comparado com a base de teste que foi separada no pré-processamento, gerando os resultados e a acurácia do algoritmo como um todo.

## 2.3 DEEP LEARNING

Técnicas convencionais de aprendizado de máquina são limitadas na sua capacidade de processar dados naturais em sua forma bruta (LECUN; BENGIO; HINTON, 2015), pois exigem uma engenharia cuidadosa para extrair recursos que transformaria dados brutos em uma representação de vetor de recursos que o modelo de ML tradicional possa extrair características e detectar padrões.

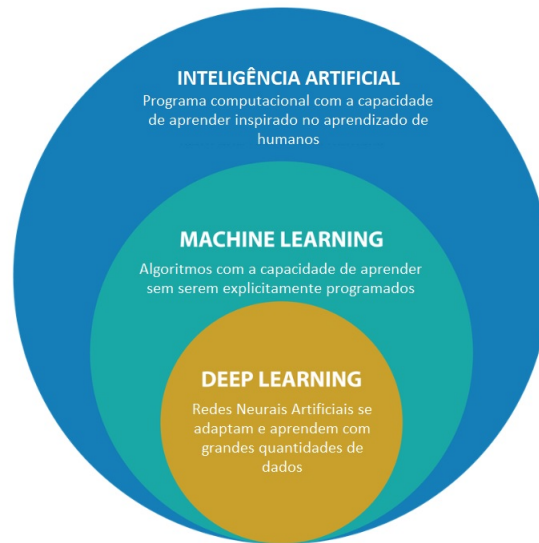
O termo *Deep Learning* ou *Deep Neural Network*, refere-se a Redes Neurais Artificiais (RNA) com múltiplas Camadas (ALBAWI; MOHAMMED; ALZAWI, 2017). Como mostra Figura 2.2, é um ramo dos métodos de ML, podendo ser definida como uma arquitetura de aprendizado com múltiplos métodos de representação, onde pode haver entrada de dados brutos sendo transformados em uma reprodução de nível mais alto, proporcionando o aprendizado de funções muito mais complexas.

Nas últimas décadas o *Deep Learning* tem sido considerada uma das mais poderosas ferramentas para lidar com uma larga escala de dados, pois consegue gerar modelos de alto nível de dados utilizando várias camadas de processamento através de transformações lineares ou não.

Seguindo a definição de Lecun, Bengio e Hinton (2015) *Deep Learning* permite que modelos computacionais compostos por várias camadas de processamento aprendam representações de dados com vários níveis de abstração.



Figura 2.2 – Representação Inteligência Artificial

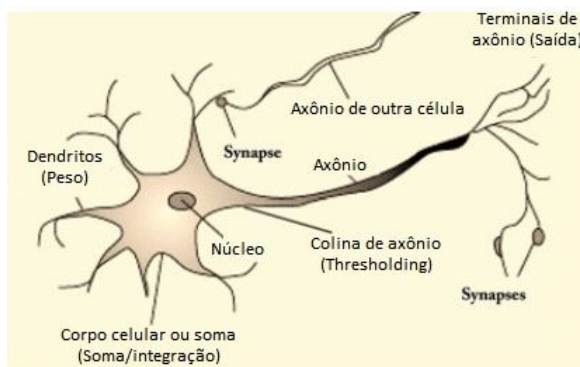


Fonte: Adaptação de Universidade Federal de Santa Maria (2021).

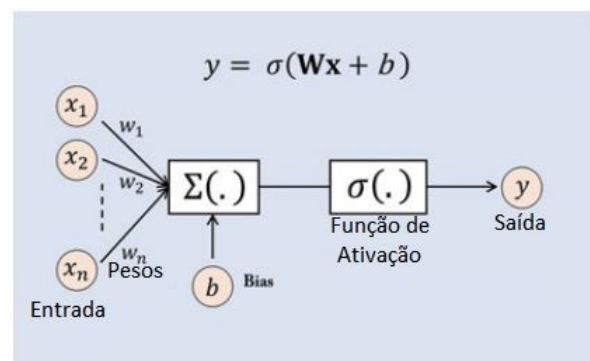
Conforme mostrado na Figura 2.3, uma rede neural artificial apresenta uma topologia inspirada na estrutura neurológica de organismos inteligentes. Essa estrutura computacional apresenta modelos matemáticos onde um neurônio é conectado com outro através de sinapses que são pesos definidos.

Figura 2.3 – Comparação neurônio biológico x artificial

(a) Neurônio biológico



(b) Neurônio artificial



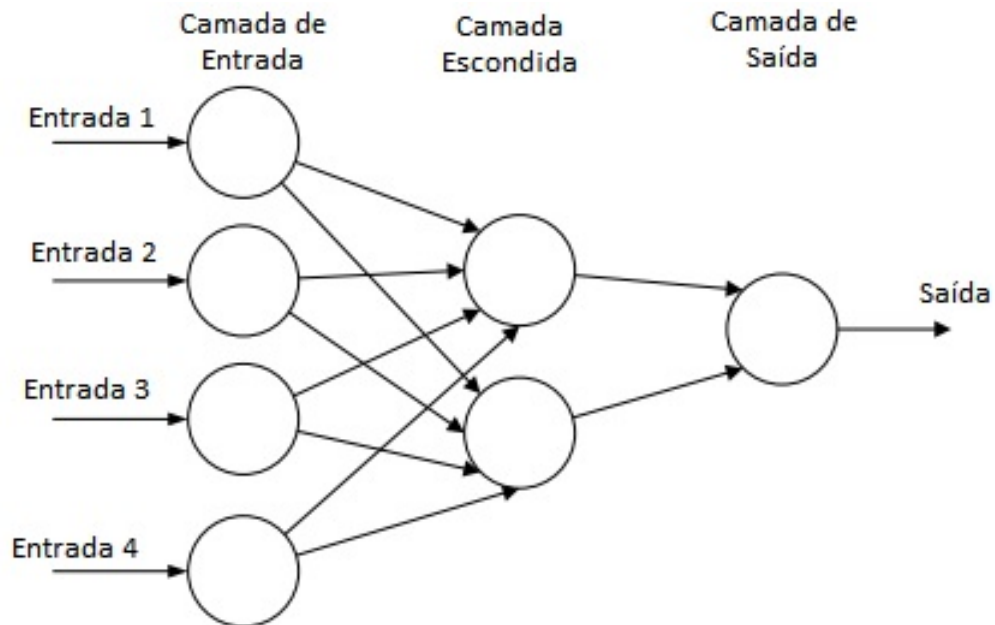
Fonte: Adaptação de Khan et al. (2018).

Existem inúmeras arquiteturas de uma RNA, diferentes tipos para diferentes propósitos como Redes Neurais Recorrentes (RNN), Redes Neurais de *Feedforward* (FNN), Redes Adversárias Generativas (GAN) e Redes Neurais Convolucionais (CNN), (IA EXPERT ACADEMY, 2020). Quantidades de arquiteturas diferentes que vem aumentando devido à alta exploração do tema *Deep Learning*.

Uma estrutura básica de uma RNA pode ser definida conforme a Figura 2.4, onde as entradas são carregadas, normalmente em forma de um vetor multidimensional, em

uma camada de entrada, que distribuirá para as camadas ocultas. Essas camadas ocultas são responsáveis por tomadas de decisão melhorando ou prejudicando o resultado final, todo esse processo descrito é referido como o processo de aprendizagem, e uma rede dispor de várias camadas ocultas é o que classifica a técnica como *Deep Learning*.

Figura 2.4 – Rede Neural Artificial



Fonte: Adaptação de O'Shea e Nash (2015)

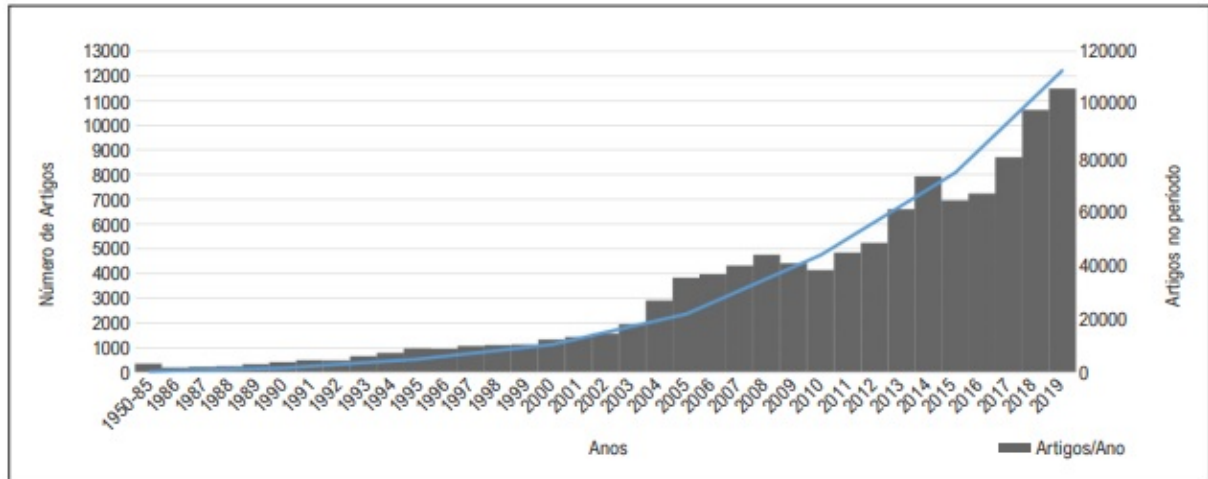
## 2.4 INTELIGÊNCIA ARTIFICIAL NA MEDICINA

Aplicações de problemas complexos onde *datasets* consolidados possam ser explorados, tem aderido ao uso de abordagens de ML. Devido às limitações de técnicas convencionais de computação, as aplicações de ML se sobressaem pela facilidade de manuseio de grandes bases de dados. Dentre os campos que utilizam essas técnicas estão sistemas bancários para detecção de fraudes, segurança de dados, logísticas de empresas, sistemas de segurança por vídeo e robótica.

Na medicina não é diferente, devido ao desenvolvimento da tecnologia, hoje em dia é possível salvar dados como prontuários de exames, imagens laboratoriais em banco de dados e disponibilizar o fácil acesso dos mesmos. Assim, o uso de IA é explorado nos campos de diagnóstico e prognóstico visando uma identificação precoce e a prevenção no tratamento de doenças. Pesquisas na base de dados PUBMED (NCBI) e Medline tem apontado um crescimento exponencial nos trabalhos de viés medicinal envolvendo os termos "*Machine Learning*", "*Artificial Intelligence*", "*Unsupervised Learning*", "*Supervised*

*Learning*” e “*Neural networks*” (PAIXÃO et al., 2022), conforme mostra a Figura 2.5 durante o período de 1951 a 2019.

Figura 2.5 – Uso na Inteligência Artificial na medicina



Fonte: (PAIXÃO et al., 2022)

## 2.5 REDES NEURAIAS CONVOLUCIONAIS

Rede Neural Convolutacional ou *Convolutional Neural Network* (CNN) é um dos tipos de rede neural artificial mais popular e utilizados nos últimos tempos, a CNN recebe esse nome por conta das operações matemáticas lineares entre matrizes chamada convolução (ALBAWI; MOHAMMED; ALZAWI, 2017). Esse tipo de rede neural tem obtido excelentes performances nos resultados de ML na última década, especialmente quando se trata de problemas de reconhecimento de padrões, reconhecimento de voz e principalmente no processamento de imagem onde trazem, por exemplo, problemas de classificação.

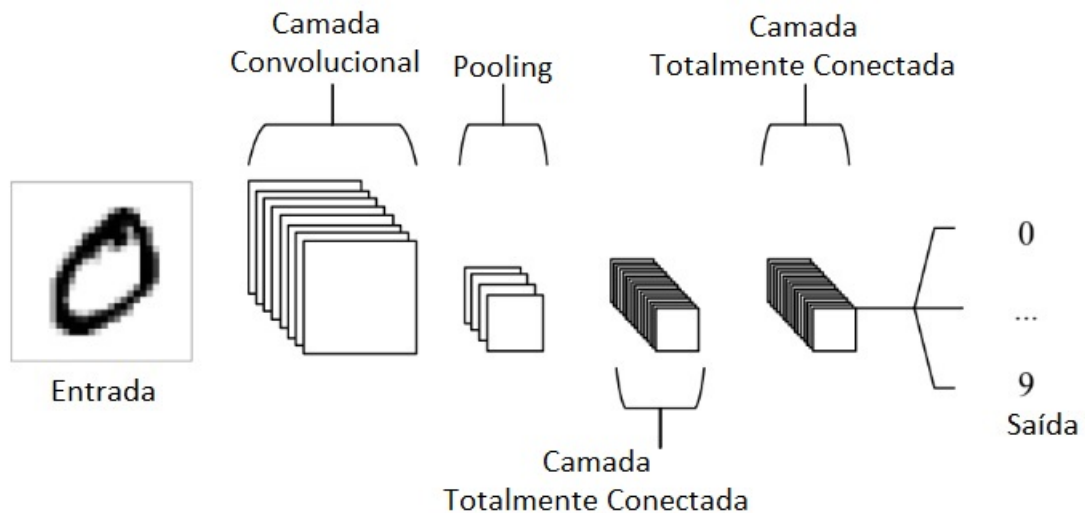
Uma das vantagens das CNN, é reduzir o número de parâmetros utilizados em relação a uma RNA clássica, o que possibilita os modelos de redes neurais serem maiores, resolvendo tarefas mais complexas devido a essa facilidade. Na literatura há um grande número de diferentes arquiteturas de uma CNN, na seção a seguir são abordadas as principais camadas que compõem uma CNN.

### 2.5.1 Camadas de uma Rede Neural Convolutacional

Uma CNN é composta por três tipos de camadas (O'SHEA; NASH, 2015), a camada convolutacional, a camada de *pooling* e a camada totalmente conectada. Só é dita uma

arquitetura convolucional, quando estas três camadas se encontram conectadas, conforme mostra a Figura 2.6.

Figura 2.6 – Arquitetura de uma Rede Neural Convolucional



Fonte: Adaptação de O'Shea e Nash (2015)

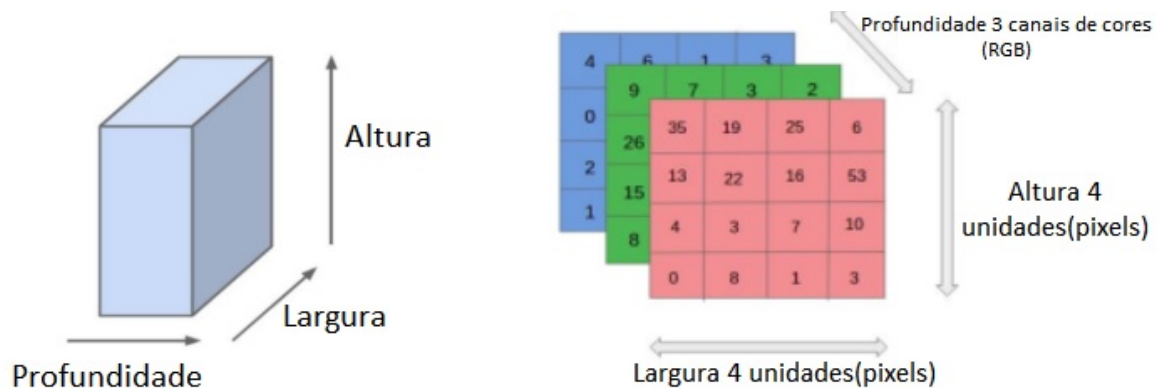
Em seguida, serão detalhadas as informações referentes a cada uma das partes da arquitetura de uma CNN conforme apresentado na Figura 2.6

#### 2.5.1.1 Camadas de Convolução

O principal objetivo de uma camada de convolução é aprender características das entradas apresentadas (O'SHEA; NASH, 2015). A partir dessa entrada de dado, que normalmente em CNN são imagens, é fornecido o tamanho de pixel dessa imagem, por exemplo, uma imagem com  $224 \times 224$  irá gerar uma matriz com dimensionalidade de 224 de altura por 224 de largura e adicionando um canal a mais quando se trata de uma imagem colorida (RGB), um canal de profundidade, definido as dimensões que serão trabalhadas conforme a Figura 2.7.

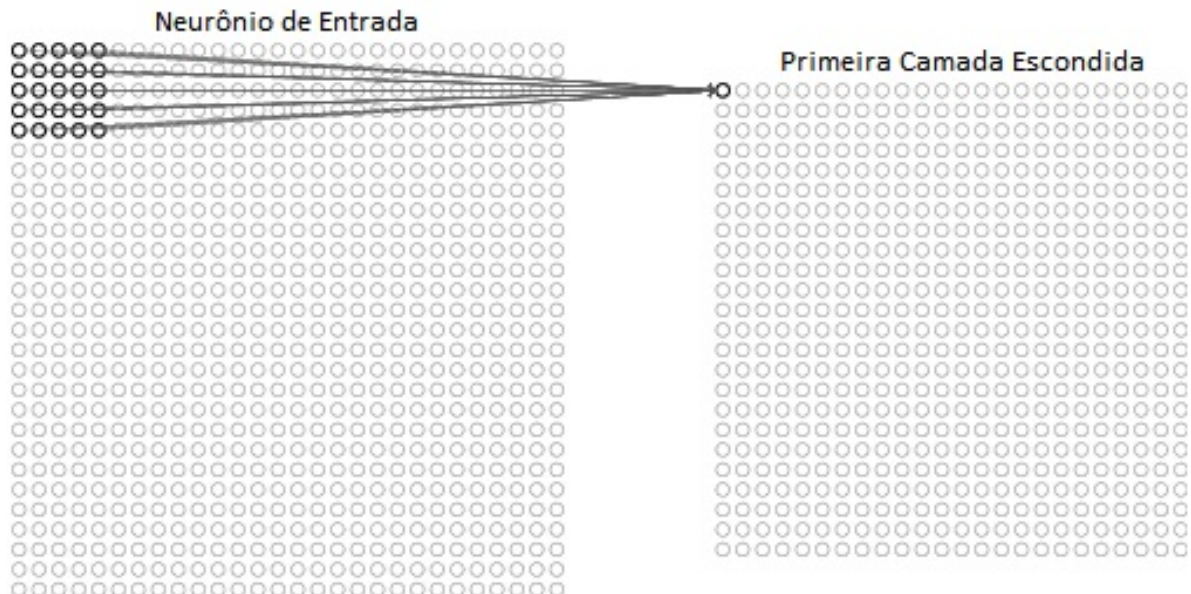
Para esse aprendizado, a convolução é composta por *kernels*, que são filtros que calculam os mapas de características da matriz como exemplifica a Figura 2.8. Esse mapa de características é extraído de acordo com o *kernel*, esse filtro define um caminho dos *pixels* que seguirá por toda a imagem. Pode-se tomar como exemplo a Figura 2.8, onde uma imagem  $32 \times 32 \times 3$  e um filtro que cobre uma área de  $5 \times 5$  da imagem com movimento de 2 saltos (chamado de *stride*), o filtro passará pela imagem inteira, por cada um dos canais, formando no final um *feature map* ou mapa de ativação de  $28 \times 28 \times 1$  (NEURONIO BR, 2018).

Figura 2.7 – Disposição espacial de entrada de Rede Neural Convolutucional



Fonte: Autor.

Figura 2.8 – Mapa de ativação na primeira camada convolutucional

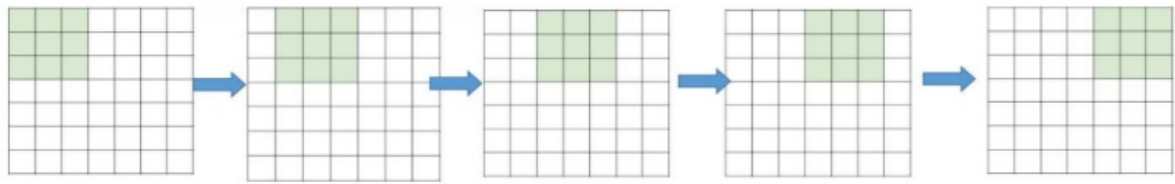


Fonte: Adaptação de Neuronio BR (2018)

### 2.5.1.2 Pooling

A ideia principal do *pooling* é o *downsampling*, que é uma redução da amostragem para diminuir a complexidade para outras camadas (ALBAWI; MOHAMMED; ALZAWI, 2017), reduzindo assim, o número de parâmetros dentro da ativação e possibilitando uma invariância espacial (SANTOS et al., 2020). Essa operação é realizada através de um *stride* que é o passo que o *kernel* vai percorrer conforme o demonstrado na Figura 2.9, esse *stride* diminuirá a dimensão da entrada e fornecerá para a outra camada uma entrada com os parâmetros reduzidos.

Figura 2.9 – Caminho percorrido pelo *kernel*

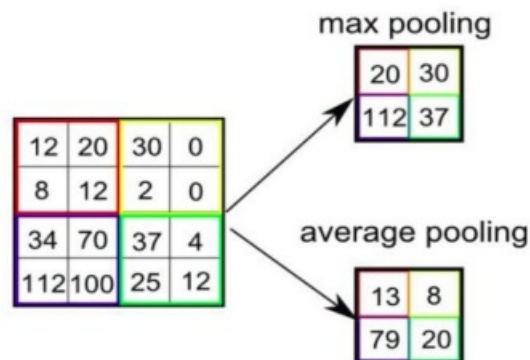


Fonte: (O'SHEA; NASH, 2015)

Os tipos de *pooling* comumente usado são definidos abaixo e a diferença entre eles é ilustrada na Figura 2.10 em um exemplo de aplicação:

- *Average pooling* - Retorna a média dos valores do mapa de ativação para o próximo passo.
- *Max pooling* - Retorna o maior valor de ativação dentro do agrupamento de uma vizinhança.

Figura 2.10 – Exemplo de *Max pooling* e *Average pooling*



Fonte: Autor.

### 2.5.1.3 Camada Totalmente Conectada

A camada Totalmente Conectada ou *Fully-Connected* (FC), como o próprio nome descreve, é a camada que conecta todos os neurônios da camada anterior e faz conexão com os neurônios da camada atual. Essa camada tem como objetivo, produzir pontuações de classes das ativações para ser usado na classificação. São sugeridas funções de ativações que podem melhorar o desempenho da rede, tal como a função ReLU (O'SHEA; NASH, 2015). A principal desvantagem de uma camada totalmente conectada é que ela inclui muitos parâmetros que precisam de computação complexa em exemplos de treinamento (ALBAWI; MOHAMMED; ALZAWI, 2017).

Após a camada totalmente conectada vem a camada de saída da CNN, onde, em problemas de classificação, o número de neurônios na saída tende a ser o número de classes do problema. A função de ativação *softmax* pode ser utilizada para transformar as saídas dos neurônios em probabilidades gerando uma classificação, retornando como resultado o neurônio de maior ou menor probabilidade como resposta.

## 2.6 TÉCNICA DE TRANSFER LEARNING

Aprendizado transferido ou *Transfer Learning* é uma abordagem de ML para a resolução de tarefas utilizando um conhecimento aprendido previamente, reutilizando essa informação com o intuito de economizar recursos nas etapas de treinamento.

Treinar algumas arquiteturas de redes muito profundas com grandes *datasets*, para trabalhos complexos, pode demandar um tempo demasiado, como semanas para chegar a um ponto de convergência, mesmo utilizando processadores especializados para esse tipo de tarefas. O emprego da técnica de *Transfer Learning* reduz, de forma acentuada, o tempo despendido na etapa de treinamento de uma RNN ajudando a rede a convergir em um tempo menor e obter um resultado mais satisfatório.

## 2.7 TREINAMENTO DE UMA REDE NEURAL CONVOLUCIONAL

O treinamento de uma rede neural, é definido por ajustar os pesos entre dois neurônios (fator multiplicativo) para cada iteração, com o intuito de reduzir o erro observado comparando a saída da rede com uma previsão esperada. Para treinamento de uma CNN, tanto os pesos nas camadas FC quanto o tamanho dos filtros de convolução são pré-definidos no treinamento, podendo ser otimizados conforme as iterações.

É comum realizar treinamentos para frações dos *datasets* de treino, essas frações são chamadas de *batch*. O seu tamanho definido, o *batch size*, serve para permitir que todos os dados de treino caibam na memória, e evitar mínimos locais das funções de perda devido a uma seleção aleatória das entradas de treinamento.

Em suma as CNN utilizam os algoritmos de aprendizado para ajustar seus parâmetros (pesos e bias) a fim de obter os valores de saída esperados, para isso, alguns algoritmos são comumente utilizados *Backpropagation* e *Stochastic Gradient Descent* (SGD).



### 2.7.1 Backpropagation

O algoritmo de *Backpropagation* calcula o gradiente de uma função de perda, também chamada de função de custo ou *loss*. Este tem o objetivo de determinar os meios de ajustar os parâmetros da rede, minimizando assim os erros que afetam o desempenho dos resultados finais.

### 2.7.2 Stochastic Gradient Descent

O algoritmo de *Stochastic Gradient Descent* (SGD) é uma aproximação iterativa do método da descida do gradiente, com o objetivo de otimizar os parâmetros de uma CNN. No método é realizado o gradiente descendente apenas sobre um *batch* do conjunto de treino em cada iteração.

## 2.8 ARQUITETURAS UTILIZADAS NO PROJETO

Neste projeto, foram utilizadas as arquiteturas de redes convolucionais VGG11, ResNet50 e Densenet121. Estas arquiteturas foram estabelecidas através da biblioteca Pytorch (PYTORCH, 2022c) na linguagem de programação Python e adaptadas de acordo com as necessidades apresentadas.

### 2.8.1 VGG

A arquitetura VGG foi desenvolvida pelo Visual Geometry Group, grupo do Departamento de Engenharia e Ciência da Universidade de Oxford, criado para participar do concurso ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, onde garantiu o segundo lugar no campo de classificação de imagem (SIMONYAN; ZISSERMAN, 2015).

O modelo VGG mostra o efeito da profundidade de uma rede convolucional no que diz respeito a sua precisão no reconhecimento de imagem em grande escala. Utilizando filtros convolucionais muito pequenos (3 x 3) e definindo dezesseis camadas convolucionais e três camadas totalmente conectadas, provando que a precisão de classificação pode ter um desempenho efetivo utilizando uma arquitetura convolucional convencional, porém aumentando substancialmente a profundidade da rede (SIMONYAN; ZISSERMAN, 2015). A Tabela 2.1 apresenta as camadas da arquitetura utilizada para o projeto, que foi a versão VGG11, disponibilizada pelo modulo de modelos da biblioteca Pytorch.



Tabela 2.1 – Camadas de uma rede VGG

<b>Camada</b>	<b>Entrada</b>	<b>Kernel</b>	<b>Stride</b>	<b>Saída</b>
Conv2d	3	(3, 3)	(1, 1)	64
BatchNorm2d	-	-	-	64
ReLU	-	-	-	-
MaxPool2d	-	(2, 2)	(2, 2)	-
Conv2d	64	(3, 3)	(1, 1)	128
BatchNorm2d	-	-	-	128
ReLU	-	-	-	-
MaxPool2d	-	(2, 2)	(2, 2)	-
Conv2d	128	(3, 3)	(1, 1)	256
BatchNorm2d	-	-	-	256
ReLU	-	-	-	-
Conv2d	256	(3, 3)	(1, 1)	256
BatchNorm2d	-	-	-	256
ReLU	-	-	-	-
MaxPool2d	-	(2, 2)	(2, 2)	-
Conv2d	256	(3, 3)	(1, 1)	512
BatchNorm2d	-	-	-	512
ReLU	-	-	-	-
Conv2d	512	(3, 3)	(1, 1)	512
BatchNorm2d	-	-	-	512
ReLU	-	-	-	-
MaxPool2d	-	(2, 2)	(2, 2)	-
Conv2d	512	(3, 3)	(1, 1)	512
BatchNorm2d	-	-	-	512
ReLU	-	-	-	-
Conv2d	512	(3, 3)	(1, 1)	512
BatchNorm2d	-	-	-	512
ReLU	-	-	-	-
MaxPool2d	-	(2, 2)	(2, 2)	-
AdaptativeAvgPool2d	-	-	-	-
Linear	25088	-	-	4096
ReLU	-	-	-	-
Linear	4096	-	-	4096
ReLU	-	-	-	-
Linear	4096	-	-	7

Fonte: Autor.

### 2.8.2 ResNet

A ResNet, vencedora do concurso ILSVRC 2015, é uma arquitetura caracterizada por ser muito profunda, possuindo 152 camadas, esse modelo também utiliza filtros de convolução simples ( $3 \times 3$ ) e faz uso de conexões residuais (HE et al., 2016). Em conexões residuais a entrada de uma camada é passada sem modificações para a próxima camada, e então apenas uma diferença sobre este valor é aprendido pelas camadas convolucionais passando sua soma para a saída.

Para melhor entendimento, será definido um bloco com uma sequência de camadas chamado *Bottleneck*, pois, esse mesmo conjunto de camadas é repetido diversas vezes na arquitetura geral da ResNet, mostrado na Tabela 2.3. O detalhamento do bloco *Bottleneck* é demonstrado na Tabela 2.2.

Tabela 2.2 – Definição do bloco Bottleneck

Camada	Kernel	Stride
Conv2d	(1, 1)	(1, 1)
BatchNorm2d	-	-
Conv2d	(3, 3)	(2, 2)
BatchNorm2d	-	-
Conv2d	(1, 1)	(1, 1)
BatchNorm2d	-	-
ReLU	-	-

Fonte: Autor.

A configuração de todas as camadas da CNN Resnet está representada na Tabela 2.3, nesta podemos observar que se caracteriza por uma RNA muito profunda, com diversas camadas de convolução. Neste, o mesmo número de camadas é tratado através dos blocos de normalização *Batch Normalization*, isto é, o número de neurônio na entrada do bloco é o mesmo número de neurônios na saída, fazendo a rede aprender somente a diferença de valores nessa camada (HE et al., 2016).

Tabela 2.3 – Camadas de uma rede Resnet50

Camada	Quantidade	Entrada	Saída
Conv2d	1	3	64
BatchNorm2d	1	-	64
ReLU	1	-	-
MaxPool2d	1	-	-
Bottleneck	1	64	256
Conv2d	1	256	256
BatchNorm2d	1	-	256
Bottleneck	4	256	512
Conv2d	1	512	1024
BatchNorm2d	1	-	256
Bottleneck	6	1024	2048
Conv2d	1	1024	2048
BatchNorm2d	1	-	2048
Bottleneck	2	1024	2048
AvgPool2d	1	-	-
Linear	1	2048	7

Fonte: Autor.

### 2.8.3 DenseNet

Densely Connected Convolutional Network (DenseNet) parte do conceito que CNN podem ser mais precisas e eficientes no treinamento, quando são potencialmente mais profundas (HUANG et al., 2017). Sendo assim, a DenseNet utiliza os mapas de características da camada atual como entrada para as próximas camadas do bloco *DenseLayer*.

A DenseNet apresenta algumas vantagens em relação as outras RNA, tais como, o alívio do problema *vanishing-gradient*. Assim fortalece a propagação de recursos da rede, reutiliza os recursos e reduz substancialmente o número de parâmetros e também exige menos gasto computacional para alcançar resultados satisfatórios (HUANG et al., 2017).

Para melhor visualização, a Tabela 2.4, mostra um bloco convolucional que é denominado *DenseLayer* ou *Dense Block*, onde a quantidade de neurônios das camadas de entrada pode ser variável. Este bloco, é replicado diversas vezes na arquitetura principal da rede conforme mostra a Tabela 2.6.

Tabela 2.4 – Definição do bloco DenseLayer

Camada	Entrada	Kernel	Stride	Saída
BatchNorm2d	-	-	-	Variável
ReLU	-	-	-	-
Conv2d	Variável	(1, 1)	(1, 1)	128
BatchNorm2d	-	-	-	128
ReLU	-	-	-	-
Conv2d	128	(3, 3)	(1, 1)	32

Fonte: Autor.

Também na arquitetura de Rede Neural DenseNet pode-se definir um bloco, podendo ser denominado como bloco de transição, onde é realizado uma normalização e extração de características, esse mapa de característica extraído é a entrada do próximo bloco de convolução que denominamos *DenseLayer*, a resolução disso é o principal resultado da rede DenseNet. O bloco de Transição pode ser visualizado na Tabela 2.5.

Tabela 2.5 – Definição do bloco Transição

Camada	Kernel	Stride
BatchNorm2d	-	-
ReLU	-	-
Conv2d	(1, 1)	(1, 1)
AvgPool2d	(2, 2)	(2, 2)

Fonte: Autor.

Tabela 2.6 – Camadas de uma rede Densenet

Camada	Quantidade	Entrada	Saída
Conv2d	1	3	64
BatchNorm2d	1	-	64
ReLU	1	-	-
MaxPool2d	1	-	-
DenseLayer	6	64	32
Transição	1	-	128
DenseLayer	12	128	32
Transição	1	-	256
DenseLayer	24	256	32
Transição	1	-	512
DenseLayer	16	512	32
BatchNorm2d	1	-	1024
Linear	1	1024	7

Fonte: Autor.

## 2.9 SISTEMA OPERACIONAL ANDROID

Android é um Sistema Operacional (SO), baseado no núcleo Linux, presente em milhões de aparelhos de diferentes categorias ao redor do mundo, sendo o SO mais utilizado atualmente (ANDROID, 2016). Esta é uma ferramenta *Open Source* tendo como principal colaborador desenvolvedor a empresa Google, utilizado principalmente em dispositivos móveis com *display* sensível ao toque, como tablets e smartphones.

O Android chegou ao mercado no dia 23 de setembro de 2008 (ANDROID, 2016), com uma versão bastante simples, porém já disponibilizando recursos como interface gráfica de bloqueio, *player* de música nativo e aplicações Google embarcados, essa primeira versão tinha o nome de Petit Four. Desde então, o SO vem sendo aperfeiçoado pela Google trazendo novas tecnologias e possibilidades de personalização, também sendo utilizado em diversas plataformas como, *smartwatches*, televisões, *videogames* e carros. A versão mais recente é Android 12 denominado *Snow Cone* (ANDROID, 2021).

Devido ao grande número de usuários e por ser um SO *Open Source*, o desenvolvimento para a plataforma Android é muito popular, com diversas possibilidades e empregos de diferentes linguagens. A linguagem de programação mais comum no desenvolvimento de aplicativos Android é o JAVA, também se destacam as linguagens Kotlin e Flutter. Neste projeto, a parte mobile, que podemos chamar de *frontend*, foi desenvolvida através do *fra-*

*mework* Kivy, que é uma biblioteca *Open Source* escrita em Python, escolhida devido ao baixo poder de processamento computacional disponível para desenvolvimento e podendo ser compilada para diferentes Sistemas Operacionais além do Android, como Linux, OS X, iOS e Windows (KIVY, 2022).

## 2.10 API REST

O termo API provém de *Application Programming Interface* e pode ser caracterizado por um conjunto de rotinas e padrões implementados por software, com o principal objetivo usar os serviços de uma funcionalidade presente em um diferente sistema. Podem ser descritas como um contrato entre um provedor e um usuário de informações, estabelecendo o conteúdo exigido pelo consumidor (a chamada) e o conteúdo exigido pelo produtor (a resposta) (RED HAT, 2020).

REST ou RESTful, *Representational State Transfer*, é um conjunto de restrições de arquitetura utilizadas para serviços web, que possibilita sistemas solicitantes acessarem e manipularem recursos utilizando conjunto predefinido de operações (RED HAT, 2020). Podemos criar o seguinte contexto, quando um cliente faz uma solicitação usando uma API REST, essa API transfere uma representação do que o solicitante necessita no momento através de métodos HTTP que pode ser *GET*, *HEAD*, *POST*, *PUT*, *PATCH*, *DELETE*, *CONNECT*, *OPTIONS* e *TRACE* (MOZILLA, 2021). O retorno da informação é entregue via HTTP utilizando um dos vários formatos possíveis: Javascript Object Notation (JSON), HTML, XML, Python, PHP ou texto sem formatação.

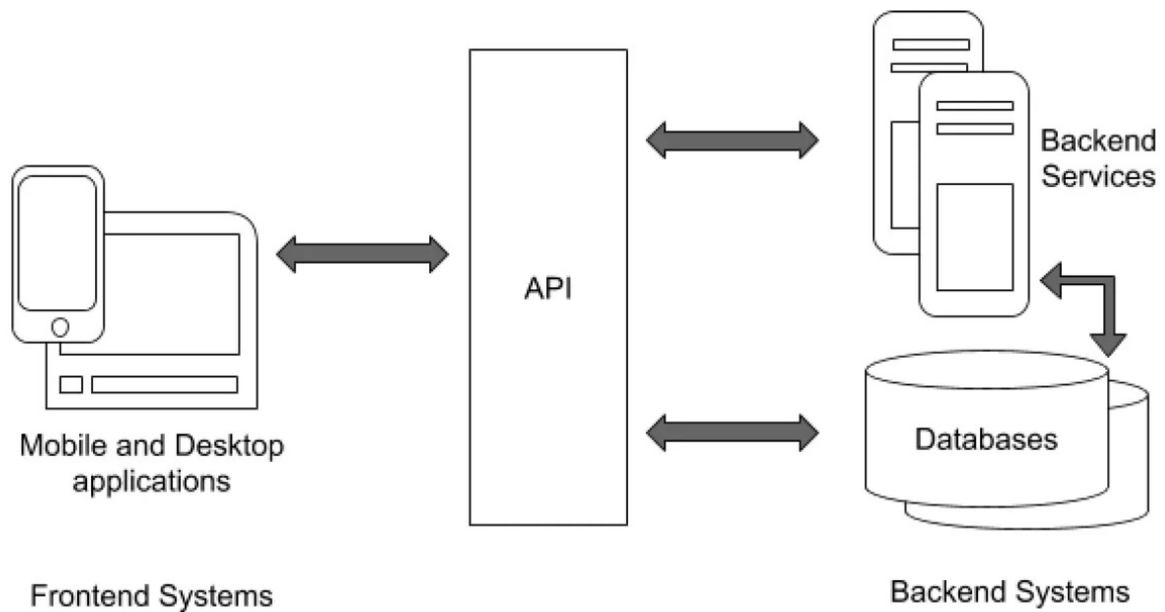
Para melhor exemplificar, são apresentados os conceitos de *frontend* e *backend*, termos utilizados para diferenciar partes específicas de um sistema, e podem ser caracterizados como:

- *Frontend* - Pode ser classificado como a parte visual da aplicação, a interface gráfica, onde o usuário final terá acesso e enviará e receberá os dados de uma aplicação. Neste projeto é a parte mobile onde o usuário fornece as imagens e recebe o resultado da classificação.
- *Backend* - Como o próprio nome sugere, é o que fica por trás da aplicação, a parte onde acontece o processamento e salvamento dos dados. É um ambiente restrito onde o usuário final não consegue acessar ou manipular. No caso deste projeto é a rede neural que classifica as imagens.

De modo geral, pode-se dizer que uma API REST realiza a comunicação entre o *frontend* e o *backend*, realizando a integração do sistema como um todo, a Figura 2.11, exemplifica este sistema. Um exemplo prático neste projeto é a classificação da lesão da

pele, onde a aplicação mobile (*frontend*), faz uma requisição através da API REST passando uma imagem, utilizando o método POST para a rede neural (*backend*) que realiza os cálculos e retorna a classe que o modelo definiu para aquela imagem.

Figura 2.11 – Exemplo de um sistema com API REST



Fonte: (INFOQ, 2020)

## 2.11 TIPO DE LESÕES DE PELE ABORDADAS NO PROJETO

O principal intuito do projeto é tratar imagens relacionadas a câncer de pele, porém, para abordar esse problema utilizando a CNN, é necessário fornecer a rede, diferentes tipos de classes para que ela consiga distinguir e extrair as diferentes características apresentadas em cada classe. O *dataset* utilizado neste projeto traz 7 diferentes classes que serão abordadas e detalhadas nas seções seguintes.

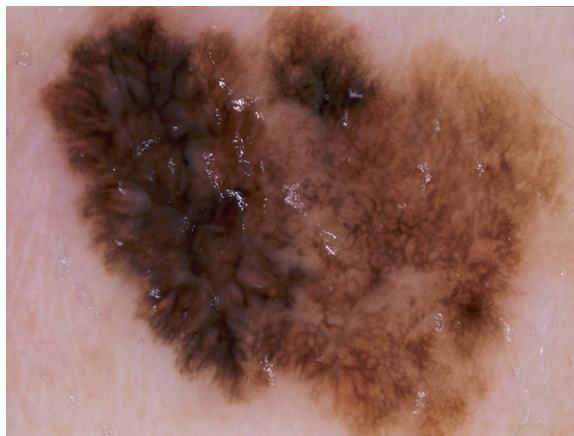
### 2.11.1 Melanoma

Segunda o INCA (INSTITUTO NACIONAL DE CÂNCER, 2022a), embora o câncer de pele seja o mais frequente no Brasil e corresponda a cerca de 30% de todos os tumores malignos registrados no país, o melanoma representa apenas 3% das neoplasias malignas do órgão. O melanoma é tumor maligno originário dos melanócitos (células que produzem

pigmento), com incidência inferior aos outros tipos de câncer de pele. Ocorre em partes como olhos, orelhas, trato gastrointestinal, membranas mucosas e genitais.

É o tipo mais grave de câncer de pele, pois o melanoma tem a capacidade de invadir qualquer órgão, e tem altas possibilidades de criar metástases, inclusive no cérebro e coração. Portanto, é um câncer com grande letalidade. Aparece como uma pinta escura Figura 2.12 que vai se deformando ao longo do tempo.

Figura 2.12 – Exemplo de Melanoma



Fonte: (TSCHANDL, 2018)

### 2.11.2 Carcinoma

O câncer de pele do tipo não melanoma é o mais frequente no Brasil, correspondendo a cerca de 30% de todos os tumores malignos registrados no país, sendo mais frequentes dentro deste tipo o carcinoma basocelular e o carcinoma espinocelular (INSTITUTO NACIONAL DE CÂNCER, 2022b).

Dentre os tipos de câncer de pele o carcinoma basocelular é o mais comum, é caracterizado como um tipo não melanoma, e tem incidência de 70% dos casos. É constituído nas células basais da pele, ocorrendo uma multiplicação de forma desordenada, gerando o tumor. Apresenta crescimento muito lento e dificilmente se espalha para outros tecidos. É encontrado em forma de mancha rosa na pele com mais frequência nas partes que ficam mais expostas ao sol, como o rosto.

O carcinoma espinocelular é o segundo tipo mais comum de câncer de pele, caracterizado como um tipo não melanoma. É mais presente em pacientes da sexta e sétima geração de vida, crescendo em áreas mais expostas ao sol como o couro cabeludo e as orelhas. O carcinoma espinocelular se forma a partir das células epiteliais (ou células escamosas) e do tegumento (todas as camadas da pele e mucosa). Tem a forma de um



nó que cresce rápido e forma uma espécie de casca, como cicatrizes ou feridas que não cicatrizam, como mostra a Figura 2.13.

Figura 2.13 – Exemplo de Carcinoma



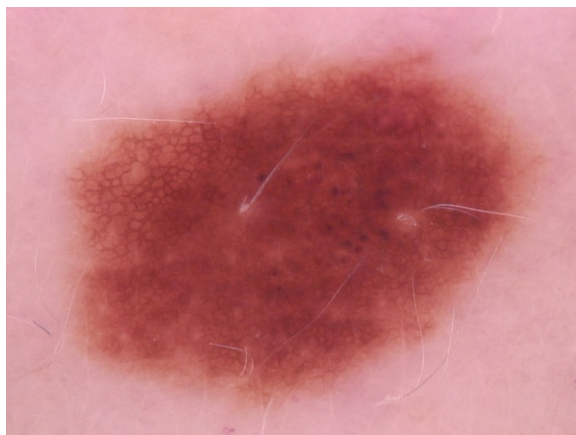
Fonte: (TSCHANDL, 2018)

### 2.11.3 Nevos Melanócitos

Nevos Melanócitos ou também nevo pigmentado, pode ser descrito como uma pinta na pele que costuma ser grande causada por um distúrbio que afeta nos melanócitos, células que produzem pigmento como mostrado na Figura 2.14.

Pode ser caracterizado como um tumor benigno, e é conhecido popularmente como pinta, ou sinal, surgindo como pequenas manchas pretas ou marrons que podem permanecer planas ou aumentarem sua espessura com o tempo (HOSPITAL DA PLÁSTICA, 2020).

Figura 2.14 – Exemplo de Nevo Melanócito



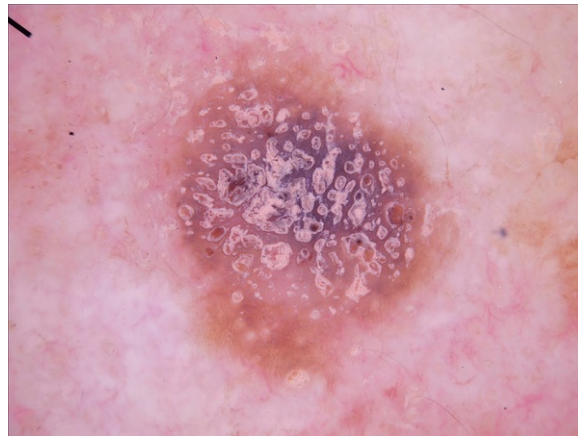
Fonte: (TSCHANDL, 2018)

#### 2.11.4 Queratose Seborreica

Queratoses seborreicas são tumores epiteliais benignos de diagnóstico usualmente fácil pelo exame clínico e dermatoscópico. Em algumas situações podem simular lesões malignas, em especial o melanoma (YORADJIAN; CYMBALISTA; PASCHOAL, 2011).

De acordo com (Wollina U, 2019), este tipo de lesão é um dos tumores de pele mais comuns e, por se tratar de um tumor benigno, o tratamento não é obrigatório. A queratose seborreica é mais comum em adultos e tende a aumentar a incidência com a idade, tendo ocorrência principalmente no tronco e testa. A Figura 2.15 apresenta um exemplo de queratose seborreica.

Figura 2.15 – Exemplo de Queratose Seborreica



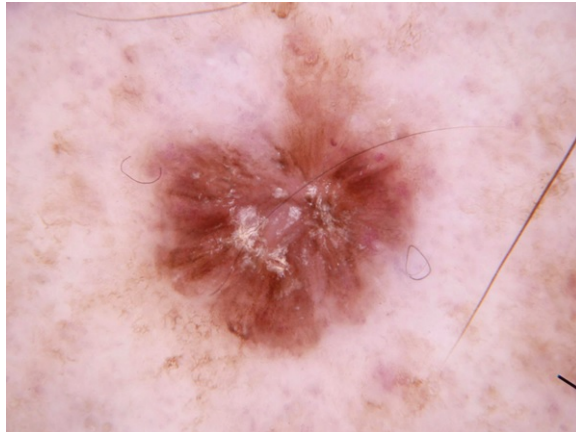
Fonte: (TSCHANDL, 2018)

#### 2.11.5 Queratose Actínicas

Segundo o INCA (SAÚDE, 2021), a queratose actínica é uma neoplasia benigna com potencial de se transformar em um tipo de câncer de pele, o carcinoma. Este tipo de lesão ocorre em áreas com exposição à luz solar, pela radiação ultravioleta. A Figura 2.16 apresenta um exemplo de queratose actínica.

A sua ocorrência se concentra em pessoas adultas e idosas, de fototipo claro, representando o quarto diagnóstico dermatológico mais comum no Brasil (SCHMITT; MIOT, 2012). A localização da queratose actínica é principalmente na cabeça, face, parte de trás dos braços e na parte de trás das mãos de pessoas de idade avançada.

Figura 2.16 – Exemplo de Queratose Actínicas



Fonte: (TSCHANDL, 2018)

#### 2.11.6 Dermatofibroma

Segundo (MYERS; FILLMAN, 2022), o Dermatofibroma é um tumor benigno que ocorre em pessoas de todas as idades, sendo mais comum entre 20 e 40 anos, e tem maior ocorrência em mulheres do que em homens. Este é um tumor comum, o qual é muitas vezes assintomático e geralmente relativamente pequeno, com diâmetro menor ou igual a 1 centímetro (ALVES et al., 2014) como mostra a Figura 2.17.

Figura 2.17 – Exemplo de Dermatofibroma



Fonte: (TSCHANDL, 2018)

### 2.11.7 Lesões Vasculares

De acordo com (NATIONAL CANCER INSTITUTE, 2018), lesões vasculares é um tipo de câncer que ocorre nas células do sangue. Este tumor vascular pode ser tanto benigno quanto maligno e pode ocorrer na pele e em outros tecidos dos órgãos. No total tem-se 9 tipos de tumores vasculares, sendo o mais comum o hemangioma congênito, tumor benigno que tem maior ocorrência em crianças. Neste trabalho é considerado o grupo de lesões vasculares contendo todos os tipos. A Figura 2.18 apresenta um exemplo de lesão vascular de pele.

Figura 2.18 – Exemplo de Lesões Vasculares



Fonte: (TSCHANDL, 2018)

### 3 REVISÃO BIBLIOGRÁFICA

Nesta seção será apresentada a revisão da literatura referente ao tema proposto, detalhando o protocolo de pesquisa utilizado. A metodologia científica abordada neste capítulo, foi baseada no processo de revisão sistemática apresentado por Ferenhof e Fernandes (2016), adequando seu método de utilização para a ferramenta Mendeley.

#### 3.1 PROTOCOLO DE PESQUISA

O protocolo de pesquisa para a revisão da literatura do tópico abordado neste projeto, considerou, definir as palavras-chaves sobre o tema mostrado e refinar as pesquisas de acordo com a quantidade de artigos retornados, assim como o período de publicação dos trabalhos, priorizando os periódicos com temáticas mais recentes. Foram buscadas publicações referentes apenas aos últimos cinco anos, sendo assim definida a busca por artigos de 2017 até 2022.

A consulta dos artigos foi feita através de bases de dados especializadas, PubMed (NCBI)<sup>1</sup>, Scopus<sup>2</sup>, Wiley<sup>3</sup>, Emerald<sup>4</sup>, PROQUEST<sup>5</sup> IEEE<sup>6</sup>, SCIELO<sup>7</sup>, ACM<sup>8</sup> e SPIE<sup>9</sup>, armazenando os artigos resultantes na ferramenta Mendeley, que foi utilizada para o gerenciamento desses artigos proporcionando uma filtragem criteriosa da relevância dos mesmos para o projeto. Nas seções seguintes, é apresentada uma tabela para cada base de dados, que relaciona as palavras-chaves com o número de artigos encontrados.

##### 3.1.1 Pesquisa na Base de Dados Scopus

Na Tabela 3.1 é possível observar que ao utilizar uma palavra-chave abrangente como “*image classification*” ou “melanoma”, houve o retorno de um número muito alto de arquivos, o que torna inviável avaliar todos em tempo hábil. Sendo assim, não se fez necessário salvar os artigos retornados na ferramenta Mendeley para essas palavras-chave.

---

<sup>1</sup> <<https://pubmed.ncbi.nlm.nih.gov/>>

<sup>2</sup> <<https://www.scopus.com/home.uri>>

<sup>3</sup> <<https://onlinelibrary.wiley.com/>>

<sup>4</sup> <<https://www.emerald.com/insight/>>

<sup>5</sup> <<https://www.proquest.com/>>

<sup>6</sup> <<https://ieeexplore.ieee.org/Xplore/home.jsp>>

<sup>7</sup> <<https://scielo.org/>>

<sup>8</sup> <<https://dl.acm.org/>>

<sup>9</sup> <<https://www.spiedigitallibrary.org/>>

Com a necessidade de refinar as buscas para retornar artigos mais específicos e relevantes para o projeto, foram utilizadas buscas com palavras-chave em conjunto e definindo a busca somente para a classe *articles*. A base Scopus acabou resultando em um número alto de publicações, mesmo quando as buscas feitas foram mais específicas, o que ocasiona uma grande dificuldade de gerenciamento e validação dos artigos.

Tabela 3.1 – Pesquisa bibliográfica na base de dados Scopus

Palavras-chave	Resultados
"image classification"	13257
"skin cancer"	5879
"skin cancer classification"	9
melanoma	31706
"skin cancer" AND "deep learning"	182
"melanoma" AND "deep learning"	300
"carcinoma" AND "deep learning"	826
"skin cancer" AND "CNN"	22

Fonte: Autor.

### 3.1.2 Pesquisa na Base de Dados Wiley

A Tabela 3.2 mostra o retorno de uma quantidade alta de artigos quando utilizada uma palavra-chave abrangente como "*image classification*" ou "melanoma" na base de dados Wiley. Refinando a busca, os resultados foram mais específicos. Todas as buscas realizadas na base Wiley foram apenas pela classe de publicações do tipo *journal*.

Tabela 3.2 – Pesquisa bibliográfica na base de dados Wiley

Palavras-chave	Resultados
"image classification"	704
"skin cancer"	346
"skin cancer classification"	18
melanoma	1841
"skin cancer" AND "deep learning"	9
"melanoma" AND "deep learning"	10
"carcinoma" AND "deep learning"	23
"skin cancer" AND "CNN"	2

Fonte: Autor.

### 3.1.3 Pesquisa na Base de Dados Emerald

A base de dados Emerald resultou em um alto número de artigos para serem gerenciados no Mendeley, conforme apresentado na Tabela 3.3. Pode-se observar que mesmo refinando as buscas, houve retorno de um grande número de resultados. Todas as buscas realizadas na base Emerald foram apenas pela classe de publicações do tipo *journal articles*.

Tabela 3.3 – Pesquisa bibliográfica na base Emerald

Palavras-chave	Resultados
"image classification"	7049
"skin cancer"	447
"skin cancer classification"	96
melanoma	44
"skin cancer" AND "deep learning"	103
"melanoma" AND "deep learning"	11
"carcinoma" AND "deep learning"	15
"tools for skin cancer identification"	66
"skin cancer" AND "CNN"	20

Fonte: Autor.

### 3.1.4 Pesquisa na Base de Dados Proquest

A base de dados Proquest oferece uma busca mais abrangente, mesmo selecionando somente a classe periódicos acadêmicos, retornou milhares de dados conforme mostrado na Tabela 3.4, isto tornou inviável o uso dessa base de dados para importação na ferramenta Mendeley.

Tabela 3.4 – Pesquisa bibliográfica na base de dados Proquest

<b>Palavras-chave</b>	<b>Resultados</b>
“image classification”	27322
“skin cancer”	23404
“skin cancer classification”	96
melanoma	93867
“skin cancer” AND “deep learning”	1874
“melanoma” AND “deep learning”	2286
“carcinoma” AND “deep learning”	4816
“tools for skin cancer identification”	0
“skin cancer” AND “CNN”	1184

Fonte: Autor.

### 3.1.5 Pesquisa na Base de Dados IEEE

A Tabela 3.5 mostra que a base de dados IEEE retornou resultados satisfatórios quando a busca é feita de forma mais específica. Realizando a classificação das buscas pelo tipo *journals*, foi possível refinar os artigos conforme o tema proposto.

Tabela 3.5 – Pesquisa bibliográfica na base de dados IEEE

<b>Palavras-chave</b>	<b>Resultados</b>
“image classification”	1854
“skin cancer”	29
“skin cancer classification”	5
melanoma	2
“skin cancer” AND “deep learning”	7
“melanoma” AND “deep learning”	15
“carcinoma” AND “deep learning”	4
“skin cancer” AND “CNN”	2

Fonte: Autor.

### 3.1.6 Pesquisa na Base de Dados Scielo

Conforme apresentado na Tabela 3.6, a base de dados Scielo acabou por não retornar artigos quando feito o refinamento nas buscas em algumas palavras-chave. Todas



as buscas realizadas na base foram apenas pela classe de publicações do tipo artigo.

Tabela 3.6 – Pesquisa bibliográfica de dados Scielo

<b>Palavras-chave</b>	<b>Resultados</b>
“image classification”	167
“skin cancer”	188
“skin cancer classification”	7
melanoma	153
“skin cancer” AND “deep learning”	0
“melanoma” AND “deep learning”	0
“carcinoma” AND “deep learning”	1
“skin cancer” AND “CNN”	0

Fonte: Autor.

### 3.1.7 Pesquisa na Base de Dados PubMed

Devido a base de dados PubMed ter uma finalidade voltada para o ramo da biomedicina, os resultados retornaram um grande número de artigos quando utilizadas palavras-chave referentes a medicina, como “*skin cancer*” ou “melanoma” como mostra a Tabela 3.7. As buscas foram refinadas colocando um conjunto de palavras-chave, mas mesmo assim, a quantidade de resultados obtidos foi alta, dificultando encontrar artigos específicos pertinentes para o trabalho.

Tabela 3.7 – Pesquisa bibliográfica na base de dados PubMed

<b>Palavras-chave</b>	<b>Resultados</b>
“image classification”	1964
“skin cancer”	9577
“skin cancer classification”	28
melanoma	39511
“skin cancer” AND “deep learning”	176
“melanoma” AND “deep learning”	263
“carcinoma” AND “deep learning”	766
“tools for skin cancer identification”	77
“skin cancer” AND “CNN”	77

Fonte: Autor.

### 3.1.8 Pesquisa na Base de Dados ACM

Pesquisando somente o tipo *journals* na base de dados ACM, refinando as buscas de acordo com a necessidade, foi obtido um resultado satisfatório dos artigos para gerenciamento e revisão, conforme mostra a Tabela 3.8.

Tabela 3.8 – Pesquisa bibliográfica na base de dados ACM

Palavras-chave	Resultados
"image classification"	815
"skin cancer"	20
"skin cancer classification"	1
melanoma	59
"skin cancer" AND "deep learning"	7
"melanoma" AND "deep learning"	19
"carcinoma" AND "deep learning"	41
"skin cancer" AND "CNN"	8

Fonte: Autor.

### 3.1.9 Pesquisa na Base de Dados SPIE

Na base de dados SPIE foram pesquisados somente a classe *journals*. A Tabela 3.9 mostrar que resultado do refinamento das buscas não obteve um decréscimo significativo dos artigos obtidos, o que dificultou a gerenciamento dos mesmos na ferramenta Mendeley.

Tabela 3.9 – Pesquisa bibliográfica na base de dados SPIE

Palavras-chave	Resultados
"image classification"	901
"skin cancer"	112
"skin cancer classification"	2
melanoma	98
"skin cancer" AND "deep learning"	48
"melanoma" AND "deep learning"	18
"carcinoma" AND "deep learning"	58
"skin cancer" AND "CNN"	13

Fonte: Autor.

## 3.2 TRABALHOS RELACIONADOS

Nesta seção são descritos os principais artigos relacionados a este projeto presentes na literatura. As informações apresentadas foram coletadas através do protocolo de pesquisa e filtrados de acordo com os critérios descritos na seção anterior.

Na literatura, há uma grande quantidade de trabalhos relacionados a detecção automática de lesões de pele, como estudos que tratam do pré-processamento das imagens, trabalhos onde são utilizadas características de forma e cor de manchas na pele, classificação e segmentação destas imagens e também trabalho comparativos abordando a qualidade de diferentes sistemas existentes. Para melhor compreensão, os artigos foram separados em duas categorias.

A primeira categoria contém os artigos relacionados a trabalhos voltados ao uso de redes neurais convolucionais no auxílio a detecção de lesões de pele. Onde são mostrados diferentes modelos e arquiteturas utilizadas na literatura, bem como diferentes abordagens com finalidade de melhorar a classificação de lesões de pele.

A segunda categoria trata de publicações relacionadas a aplicações de detecções automáticas de lesões de pele. Onde são apresentadas as tecnologias mais comuns utilizadas e os diferentes métodos de implementação destas aplicações, sejam elas *mobile* ou não.

### 3.2.1 Redes Neurais Convolucionais na Detecção de Lesões de Pele

Visando uma abordagem mais clara do tema, a subseção será dividida em estudos de CNN utilizando o conjunto de dados HAM10000, o mesmo que foi utilizado neste projeto e publicações referente a outros *datasets* que apresentam relevância para a discussão vigente neste trabalho.

#### 3.2.1.1 Publicações com Utilização do Dataset HAM10000

No que tange o pré-processamento de imagens de lesões de pele, Kaur, GholamHosseini e Sinha (2022) apresentam uma abordagem para a remoção de pelos das imagens e aprimoramento de imagens de baixo contraste, que podem ser caracterizadas como ruído, utilizando CNN. Duas abordagens são tomadas neste projeto, a primeira é a remoção dos pelos, baseado em ajuste de intensidade (IA-HR) utilizando operadores morfológicos no ajuste dos níveis de intensidade e usando critérios de vizinho mais próximo para detectar e eliminar os *pixels* onde se encontram os pelos. A segunda abordagem utiliza uma Rede Neural Convolucional de Agregação de Contexto Multiescala (MCACNN),

afim de reduzir a perda de conteúdo de alta frequência e aumentar a resolução das imagens.

Em Bardou et al. (2022), o pré-processamento da imagem referente a remoção de pelos aborda o uso de variação de *autoencoder* sem a necessidade de amostras pareadas. O *encoder* assume a entrada de imagem e constrói uma distribuição latente, considerando os pelos como um ruído, ignorando os mesmos na reconstrução da imagem feita pelo *encoder*. A avaliação das imagens reconstruídas é realizada através do mapeamento de recursos distribuídos *stochastic neighbor embedding* (SNE). Segundo o autor, pré-processamento através destes métodos apresentaram resultados muito eficientes.

Se tratando de hiperparâmetros, Xiang et al. (2021) apresenta uma estratégia para atenuação dos valores dos pesos na utilizando RNA. Foram utilizados cinco tipos de RNA's para a abordagem neste projeto, SqueezeNet, MnasNet, MobileNetV3, Xception e InceptionV3, todas arquiteturas caracterizadas por serem de baixa densidade. Este tipo de técnica não altera o tamanho dos modelos utilizados, realiza a melhora do desempenho através do *finetuning*. Esta estratégia de treino teve 83,50% no seu melhor resultado utilizando o modelo MnasNet.

Abdelhalim, Mohamed e Mahdy (2021) apresenta uma abordagem de *data augmentation* baseado no modelo de RNA denominado Redes Adversárias Generativas (GANs). O trabalho destaca a necessidade de modelos de *deep learning* exigirem quantidades consideráveis de dados, enquanto a disponibilidade de imagens anotadas geralmente é limitada, propondo a aplicação de *Self-attention Progressive Growing of GANs* (SPGGANs) para gerar imagens de lesão de pele 256 x 256 *pixels*. Os resultados apresentam uma melhoria de sensibilidade de 5,6% e 2,5% no caso geral de todas as classes de imagem em relação as contrapartes não aumentadas, conseguindo também a marca de melhora de 13,8% e 8,6% para a classe específica de melanoma.

Em Basak, Kundu e Sarkar (2022) a técnica utilizada é a segmentação de imagens, afim de localizar a posição da doença na imagem, monitorando alterações morfológicas e extraindo características discriminativas para diagnósticos adicionais. O trabalho demonstra uma estrutura denominada MFS-NET (Multi-Focus Segmentation Network) que utiliza diferentes escalas de *features maps* para calcular a máscara de segmentação final da imagem.

Abordagens com ênfase em Transfer Learning também se destacam na referência de classificação de imagens de lesões de pele. Rahi et al. (2021) examina as arquiteturas MobileNetv1, InceptionV3, VGG16, VGG19 e U-Net propondo uma solução usando o Transfer Learning o qual possui camadas de treino altamente densas. Em Younis, Bhatti e Azeem (2019) é proposto, através de Transfer Learning, um ajuste nos pesos da rede MobileNet obtendo resultados de precisão de 97,07%.

A solução para classificação de imagens de lesões de pele apresentada em Srinivasu et al. (2021), aborda os modelos de *deep learning* MobileNetV2 e Long Short Term

Memory (LSTM). A performance destes modelos é comparada a outras presentes na literatura e ambas apresentam como resultado performances acima de 85% de precisão em seus modos de teste.

Em Bedeir, Mahmoud e Zayed (2022) é proposto um sistema automático de classificação de câncer de pele concatenando dois modelos de CNN diferentes, os modelos utilizados são o ResNet50 e o VGG16. A união desses dois modelos resultou em um percentual de acerto em sua fase de teste de 94,14%.

Devido a grande quantidade de trabalhos relacionados a classificação de imagens de lesão de pele, Pranav et al. (2021) apresentam uma análise da eficácia de diferentes tipos de redes neurais profundas no reconhecimento destas lesões malignas de pele. Dentre as redes testadas estão as populares ResNetv2, MobileNet e Inception. Após feito o levantamento a combinação das redes InceptionResNetV2 + ResNeXt101 é o modelo que apresentou uma acurácia mais alta com 92,83 % em seu teste.

### 3.2.1.2 *Publicações com Utilização de Diferentes Datasets*

Perez et al. (2018) realiza uma investigação de 13 cenários na utilização de *data augmentation* em três diferentes tipos de CNN's. Os cenários incluem cores tradicionais, transformações geométricas e aumentos mais incomuns, como transformações elásticas, apagamento aleatório e um novo aumento que mistura diferentes lesões. Os resultados do melhor corresponderam a uma precisão de 88,2%.

Os conjuntos de imagens podem conter vieses, muitas vezes não intencionais, devido a forma que foram adquiridos e salvos. Esses vieses podem distorcer o desempenho do aprendizado de máquina, desconstruindo correlações que esses modelos poderiam aprender. Em Bissoto et al. (2019), é realizado um estudo sobre os pontos positivos e negativos desses vieses e os resultados mostram que o modelo apresentado pode classificar corretamente imagens de lesões de pele sem informações clinicamente significativa. Porém em Bissoto, Valle e Avila (2020) é ressaltado que métodos de última geração para a remoção desses vieses não estão prontos para diferentes conjuntos de dados nos modelos de lesões de pele.

O trabalho de Razzak e Naz (2022) realiza uma abordagem de Redes Neurais Residuais, para a classificação de câncer de pele. Este projeto apresenta uma rede residual densa e de multiestágios onde são adicionados blocos de supervisões com a finalidade de encurtar as conexões. Os resultados chegaram a 98,05% de precisão em sua fase de teste, apresentando melhoras em comparação com outros modelos da literatura.

O trabalho de Haenssle et al. (2018) chama a atenção por realizar uma comparação entre os modelos de redes neurais para detecção de câncer de pele e um grande grupo internacional de 58 dermatologistas, incluindo 30 especialistas. Os resultados dos

experimentos foi que a maioria dos dermatologistas foi superada pela CNN, o que prova que, independentemente da experiência de qualquer médico, eles podem se beneficiar da assistência da classificação de imagens da CNN.

A revisão bibliográfica de trabalhos relacionados a métodos de detecção de lesões de pele demonstra, mesmo com a vasta quantidade de publicações referente ao tema, ainda é possível realizar contribuições, sejam elas relacionadas a hiperparâmetros, novas abordagens de modelos e até novos conjuntos de dados, visando apresentar melhoras nos resultados em trabalhos já publicados ou fomentar a discussão do tema.

### 3.2.2 Aplicações para Detecção Automática de Lesões de Pele

Em Huo (2021), é apresentado um projeto *fullstack* para diagnósticos de câncer de pele utilizando o modelo de CNN. Neste trabalho a parte *frontend* é desenvolvida em Swift UI e o *backend* utilizando ExpressJS. O treinamento da CNN é realizado utilizando a biblioteca Tensor Flow e o mesmo dataset utilizado neste presente trabalho, o HAM10000. A arquitetura de CNN também integra métodos de linguagem natural em sua função, o que traz uma melhora significativa dos resultados da rede neural. Este modelo apresentou uma acurácia de 75% em seu teste. Neste projeto também foi desenvolvido um serviço de chat, onde o usuário final consegue tirar dúvidas relacionadas a câncer de pele. A comunicação entre a aplicação mobile e o software que classifica a imagem fornecida é feita através de API e a parte de serviço de chat através de *Web Socket Connection*. Esta aplicação foi feita somente para o SO iOS onde se encontra funcional.

O trabalho apresentado em Shihadeh, Ansari e Ozunfunmi (2019), descreve uma aplicação utilizando os modelos de CNN AlexNet e GoogleNet para uma classificação de imagem para diagnósticos médicos de forma remota. Chama a atenção neste projeto, todos os treinamentos serem executados na GPU NVIDIA Jetson TX2, que é um kit de desenvolvimento embarcado projetado pela empresa NVIDIA, onde permite o uso de periféricos como a câmera para a captação de imagens de lesões de pele. O intuito de Shihadeh, Ansari e Ozunfunmi (2019) é futuramente desenvolver um kit multissensor que poderá fornecer um sistema completo de diagnóstico médico de forma remota.

Castro et al. (2020) apresenta uma abordagem para lidar com dados desbalanceados em algoritmos evolucionários em um aplicativo para detecção de melanoma. A aplicação utiliza o modelo de CNN alimentado por um *dataset* com imagens coletadas através de *smartphones* e informações clínicas da lesão. Como a aplicação trata somente de melanoma, um conjunto muito menor que as outras lesões de pele, a maioria dos *datasets* apresentam conjuntos desequilibrados de dados. Assim, após feita a abordagem do balanceamento do conjunto de imagens, a aplicação obteve resultados em 92% para o *dataset* utilizado. O aplicativo (*frontend*) foi desenvolvido utilizando React Native, que

permite ser multiplataforma *mobile*, as comunicações com o servidor de banco de dados utilizam Java Web Server (Tomcat) e MySQL e a comunicação entre o algoritmo de detecção, que foi desenvolvido em Python através da biblioteca Pytorch, são feitas através de API Rest. O treinamento da CNN é realizado utilizando camadas do modelo ResNet50, um dos modelos utilizados no presente trabalho, porém o *dataset* é diferente, denominado PAD-UFES.

Hartanto e Wibowo (2020) mostram uma comparação entre duas arquiteturas de CNN, o modelo MobileNet v2 e o modelo Faster R-CNN, na validação das mesmas através de um computador pessoal usando Jupyter Notebook e de um aplicativo Android. O Aplicativo Android utilizado neste projeto foi desenvolvido pela mesma equipes em trabalhos anteriores para a utilização da arquitetura MobileNet v2 (WIBOWO; HARTANTO; WIRAWAN, 2020). O trabalho traz como resultado a arquitetura MobileNet v2 obtendo uma performance mais alta de precisão quando aplicada em um *smartphone*, chegando a uma acurácia de 86,3%, já a Faster R-CNN tendo um resultado melhor utilizando o Jupyter Notebook em *desktop*, com a precisão de 87,2%.

Após a realização desta revisão bibliográfica de aplicações auxiliando na detecção de lesões de pele, é verificado que, apesar da vasta quantidade de publicações na literatura referente ao tema, ainda é possível contribuir para a ampliação da discussão acerca destes tópicos. As contribuições podem ser relevantes realizando implementações com arquiteturas de RNA diferentes em diferentes conjuntos de dados aumentando a precisão em relação aos resultados. Também novos métodos de implementações de softwares capazes de facilitar o acesso à informação e detecção precoce de lesões de pele, auxiliando no tratamento e aumentando as chances de cura quando o diagnóstico for positivo.

## 4 DESENVOLVIMENTO

Este capítulo tem como objetivo apresentar os métodos empregados ao longo do desenvolvimento do projeto, bem como os materiais utilizados para a construção do software. O desenvolvimento pode ser dividido em duas etapas, a primeira onde é tratada toda a parte referente a *Deep Learning*, no que tange o algoritmo de classificação de imagens, onde os modelos são treinados para reconhecer padrões através das imagens fornecidas pela base de dados, utilizando Redes Neurais Convolucionais. A segunda parte do desenvolvimento pode ser caracterizada pela passagem desse algoritmo de classificação para uma aplicação funcional, neste caso sendo um software *mobile* essa operação pode ser chamada de *deploy*. A produção da segunda etapa inicia uma vez que a primeira fase esteja apresentando resultados satisfatórios.

### 4.1 DEEP LEARNING

Nesta primeira etapa, como descrito anteriormente, são apresentados os métodos utilizados para o treinamento de uma Rede Neural Convolucional, e também para o teste dos resultados obtidos.

#### 4.1.1 Ambiente de Desenvolvimento

Devido ao alto gasto computacional que uma RNA profunda necessita para realizar o treinamento, o uso de computadores pessoais se tornou uma complicação para o desenvolvimento do projeto. Devido a isso, tornou-se necessário encontrar alternativas para a elaboração e teste das redes neurais. A solução encontrada foi a ferramenta Google Colaboratory, que disponibiliza ambientes de desenvolvimento com alto recurso computacional que pode ser acessado através do navegador.

A especificações do ambiente de desenvolvimento, bem como a linguagem de programação utilizada e suas bibliotecas são descritas a seguir:

- Ambiente de desenvolvimento Google Colaboratory
  - Sistema Operacional Linux Ubuntu versão 18.04
  - Dois Processadores Intel(R) Xeon(R) CPU @ 2.20GHz
  - Placa de Vídeo NVIDIA Tesla T4
  - 13GB de Memória RAM



- Linguagem de Programação Python versão 3.7.13

Biblioteca Numpy versão 1.21.6 utilizada para cálculos matemáticos

Biblioteca Pandas versão 1.3.5 utilizada para manipulação e visualização de listas e tabelas

Biblioteca MatplotlibLib versão 3.2.2 utilizada para plotar imagens e gráficos

Biblioteca Glob utilizada para o acesso e manipulação de pastas e arquivos no sistema operacional

Biblioteca Pillow versão 7.1.2 utilizada para manipulação vetorial das imagens

Biblioteca Pytorch versão 1.11.0 e todas as suas dependências, utilizada para o trabalho com *Deep Learning*

Biblioteca Scikit-learn versão 1.1 utilizado para a separação dos dados em treino e teste

É válido salientar que a linguagem de desenvolvimento Python foi escolhida pela facilidade de trabalhar com redes neurais, devido a vasta opções de bibliotecas disponibilizadas, como a biblioteca Pytorch. Também foi fato fundamental, na escolha destas linguagens e bibliotecas, o conhecimento prévio na utilização das mesmas e a comunidade ativa de desenvolvedores que estas ferramentas possuem.

Também é necessário pontuar que a escolha de processamento no desenvolvimento das redes neurais se deu pela Unidade de Processamento Gráfico (GPU) ao invés do processador padrão (CPU), já que este é disponibilizado pela ferramenta Google Colaboratory. Com o desenvolvimento do algoritmo realizado utilizando GPU, o tempo de execução dos treinamentos e teste da rede se torna consideravelmente menor, otimizando a produção do projeto.

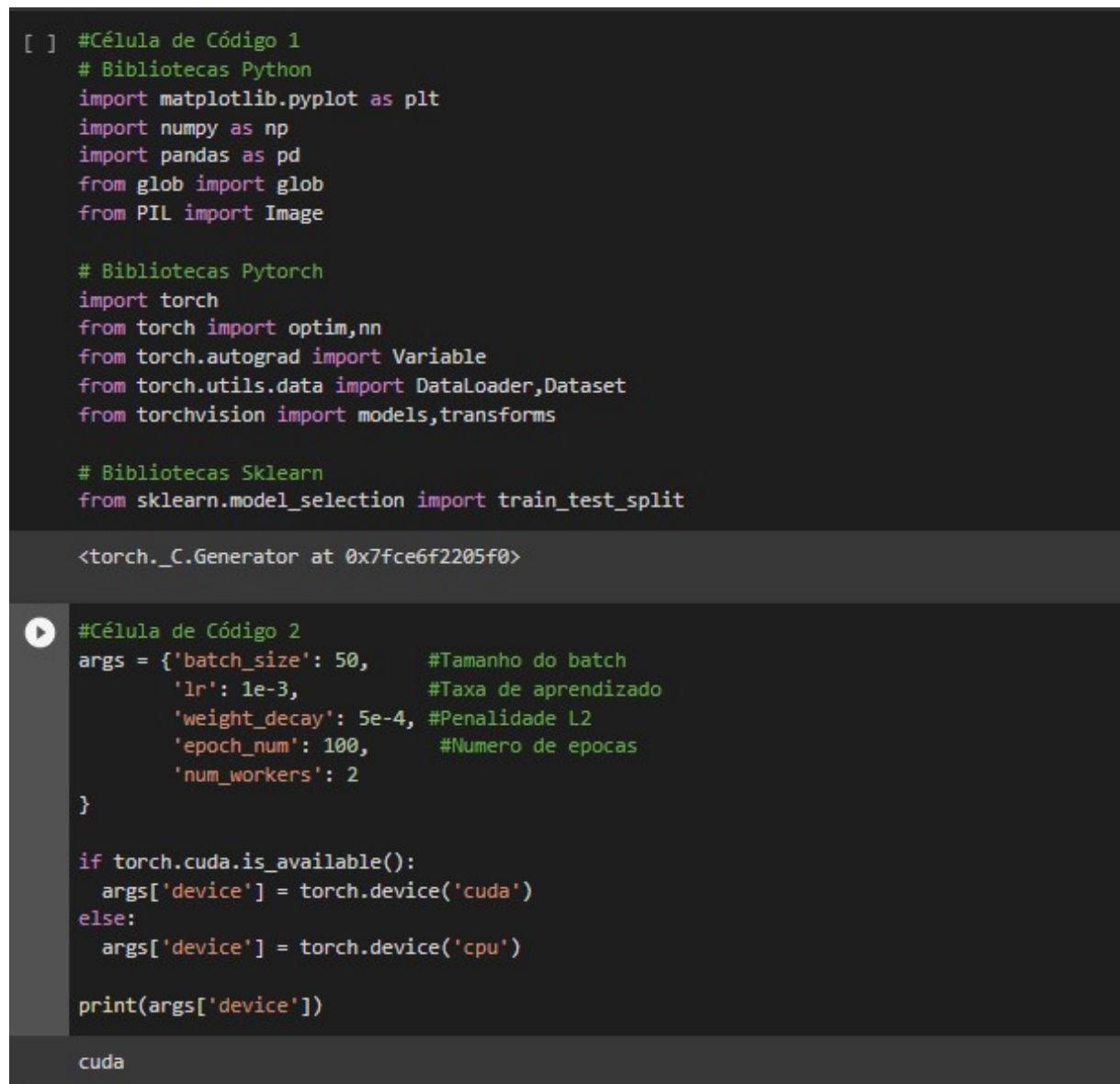
#### 4.1.2 Utilização do Google Colaboratory

A utilização do Google Colaboratory <sup>1</sup> que é popularmente chamado Google Colab, permite a execução dos algoritmos de programação por células como mostra a Figura 4.1. A célula 1 está fazendo a importação das bibliotecas e a célula 2 está criando um índice com as variáveis que serão utilizadas posteriormente no treinamento, bem como verificando se há disponibilidade para o uso de GPU (Cuda) no ambiente de desenvolvimento.

---

<sup>1</sup> <<https://colab.research.google.com/>>

Figura 4.1 – Células de Código Google Colaboratory



```
[ ] #Célula de Código 1
# Bibliotecas Python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from glob import glob
from PIL import Image

# Bibliotecas Pytorch
import torch
from torch import optim,nn
from torch.autograd import Variable
from torch.utils.data import DataLoader,Dataset
from torchvision import models,transforms

# Bibliotecas Sklearn
from sklearn.model_selection import train_test_split

<torch._C.Generator at 0x7fce6f2205f0>
```

```
#Célula de Código 2
args = {'batch_size': 50,      #Tamanho do batch
        'lr': 1e-3,          #Taxa de aprendizado
        'weight_decay': 5e-4, #Penalidade L2
        'epoch_num': 100,    #Número de epocas
        'num_workers': 2
    }

if torch.cuda.is_available():
    args['device'] = torch.device('cuda')
else:
    args['device'] = torch.device('cpu')

print(args['device'])

cuda
```

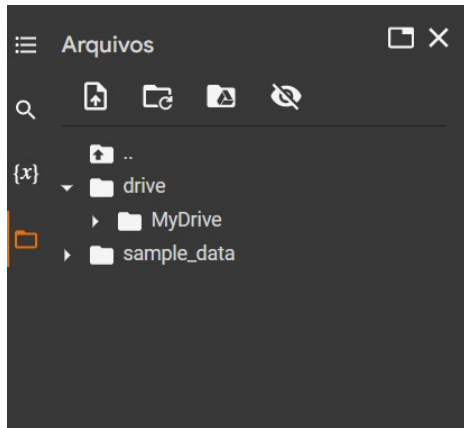
Fonte: Autor.

Cada Célula de código pode ser executada individualmente, não tendo obrigatoriedade em executar de forma sequencial a criação das mesmas. Desde que haja a lógica computacional certa, uma célula pode ser dependente ou não da outra. O resultado da execução de cada célula é mostrado logo a baixo das mesmas individualmente.

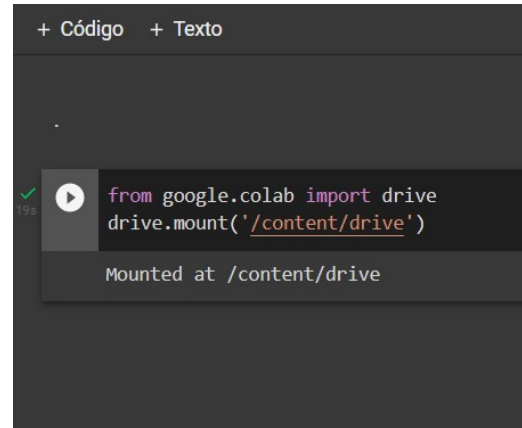
O Google Colaboratory permite o acesso aos arquivos do Google Drive da conta vinculada, neste projeto, o dataset utilizado, foi carregado para a conta do Google Drive e utilizado como forma de acesso a arquivos. A Figura 4.2b mostra a célula de código que foi executada para conectar o Colab ao Google Drive e a Figura 4.2a como podemos acessar qualquer diretório ou arquivo através da interface gráfica.

Figura 4.2 – Vincular Google Drive ao Google Colaboratory

(a) Interface gráfica para acesso as pastas



(b) Célula de código para montagem do Drive



Fonte: Autor.

É relevante destacar que o Google Colaboratory não fornece livremente o acesso computacional ao usuário, seja limite de espaço no Google Drive ou acesso por tempo ilimitado a GPU do ambiente de desenvolvimento. Para usufruir de tais ferramentas o usuário deve pagar uma taxa de serviço. Este projeto não utilizou de nenhum recurso financeiro, utilizando somente as ferramentas em sua versão padrão, o que de certa forma dificultou o treinamento da rede, pois nem sempre a GPU se encontrava disponível para uso.

#### 4.1.3 Banco de dados

O conjunto de dados ou *dataset* utilizado para o desenvolvimento do projeto é o Human Against Machine with 10000 training images (HAM10000), coletado e disponibilizado para acesso público pela International Skin Imaging Collaboration (ISIC). Este *dataset* consiste em 10015 imagens de lesões pigmentadas de pele catalogadas através de metadados também disponibilizados para acesso (TSCHANDL, 2018).

Conforme citado na seção 2.11, são 7 os tipos de lesões abordados pelo HAM10000, Melanoma, Carcinoma, Nevos Melanócitos, Queratose Seborreica, Queratose Actínicas, Dermatofibroma e Lesões Vasculares. A Tabela 4.1 mostra a quantidade referente a cada lesão que o *dataset* HAM10000 disponibiliza.

Tabela 4.1 – Tipos de lesões *dataset* HAM10000

<b>Tipo</b>	<b>Quantidade</b>
Melanoma	1113
Carcinoma	514
Nevos Melanócitos	6705
Queratose Seborreica	1099
Queratose Actínicas	327
Dermatofibroma	115
Lesões Vasculares	142

Fonte: (TSCHANDL, 2018)

Os metadados disponibilizados pelo HAM10000 apresentam o ID da lesão referente ao próprio banco de dados, o ID da imagem referente a ISIC, a classificação da imagem, para a realização de um treinamento supervisionado, o local onde a lesão se encontra no paciente, bem como a idade e o sexo do paciente. Também é disponibilizado o tipo de validação técnica que foi realizado por um dermatologista especialista, essa validação técnica é uma correção manual no histograma das imagens, e podem ser divididos em quatro tipos diferentes de validação: Histopatologia, Confocal, Acompanhamento e Consenso (TSCHANDL, 2018).

É importante afirmar que um dos problemas é a característica do tom de pele das imagens contidas pela maioria dos conjuntos de dados, essas imagens tem como característica predominante eurocêntrica, isto é, tom de pele mais claro, predominantemente branco, o que não condiz com a realidade da diversidade de tons de pele presentes na população, uma vez que tons de pele escuros, embora com menos frequência, também são suscetíveis a câncer de pele. De maneira alguma a aplicação pode ser excludente, dito isso, um dos principais objetivos deste projeto, é fomentar a ideia de criação de uma base de dados mais diversificada que seja condizente com toda a população mundial.

#### 4.1.3.1 *Preparação do Banco de Dados*

Antes de começar o treinamento, é necessário preparar o *dataset*, de maneira que a rede consiga aprender e testar as informações quando solicitada. Para isso, como este é um treinamento supervisionado, foi vinculada a informação presente nos metadados com a imagem fornecida no banco de dados, isso é, para cada imagem colocar informações de qual tipo de lesão de pele ela pertence. Também se faz necessário checar se existe algum tipo de duplicidade no banco de dados, e se houver, utilizar somente uma das imagens correspondentes.

O HAM10000 fornece os metadados do *dataset* em forma de arquivo CSV, este

arquivo é trabalhado e modificado utilizando a biblioteca Pandas através da criação de um *dataframe*. A fim de facilitar o treinamento e teste dos dados, é necessário remover algumas informações dos metadados fornecido e adicionar outras que se fazem relevantes.

Informações como a localização da ferida, idade e sexo da pessoa que forneceu a imagem são irrelevantes para o tipo de treinamento utilizado, para isso essas colunas foram removidas do *dataframe*. Para o acesso de cada imagem, foi adicionado o caminho onde ela se encontra no Google Drive, para captar o caminho da imagem foi utilizado a biblioteca Glob.

Também foram adicionados os tipos de lesão por índice numérico, o que facilita a comparação com a saída da rede neural na sua fase de teste. A Tabela 4.2 demonstra o resultado final das colunas do *dataframe* que foram utilizadas para o treinamento e teste das imagens de lesões de pele.

Tabela 4.2 – Exemplo de informações presentes no dataframe para o treinamento

ID Imagem	Tipo de Lesão	ID Lesão	Caminho da Imagem
ISIC-27419	Queratose Seborreica	2	/MyDrive/HAM10000/ISIC-27419.jpg
ISIC-05276	Dermatofibroma	3	/MyDrive/HAM10000/ISIC-05276.jpg
ISIC-30171	Melanoma	6	/MyDrive/HAM10000/ISIC-30171.jpg
ISIC-31197	Lesão Vascular	5	/MyDrive/HAM10000/ISIC-31197.jpg
ISIC-25955	Melanoma	6	/MyDrive/HAM10000/ISIC-25955.jpg
ISIC-34093	Carcinoma	1	/MyDrive/HAM10000/ISIC-34093.jpg

Fonte: Autor.

Assim que este pré-processamento é feito, o *dataset* é separado entre conjuntos de treino e teste. É comumente utilizado uma maior escala do banco de dados para o conjunto de treino e uma fatia menor para teste, para este projeto foi utilizado 80% do *dataset* para o treinamento da rede e 20% para a teste. A separação do conjunto de dados foi feita através da biblioteca Scikit-learn.

Como mostrado na Tabela 4.1, existe um desequilíbrio na quantidade de imagens de cada lesão, o que pode piorar o resultado final do treinamento. A fim de melhorar a resposta da rede, foi realizado um balanceamento para as classes que contém menos dados.

A realização do balanceamento foi feita utilizando a técnica de *data augmentation*, técnica essa que replica os dados com pequenas modificações através de transformações na imagem. A Tabela 4.3 mostra os valores do conjunto de treinamento após o balanceamento dos valores ser feito.

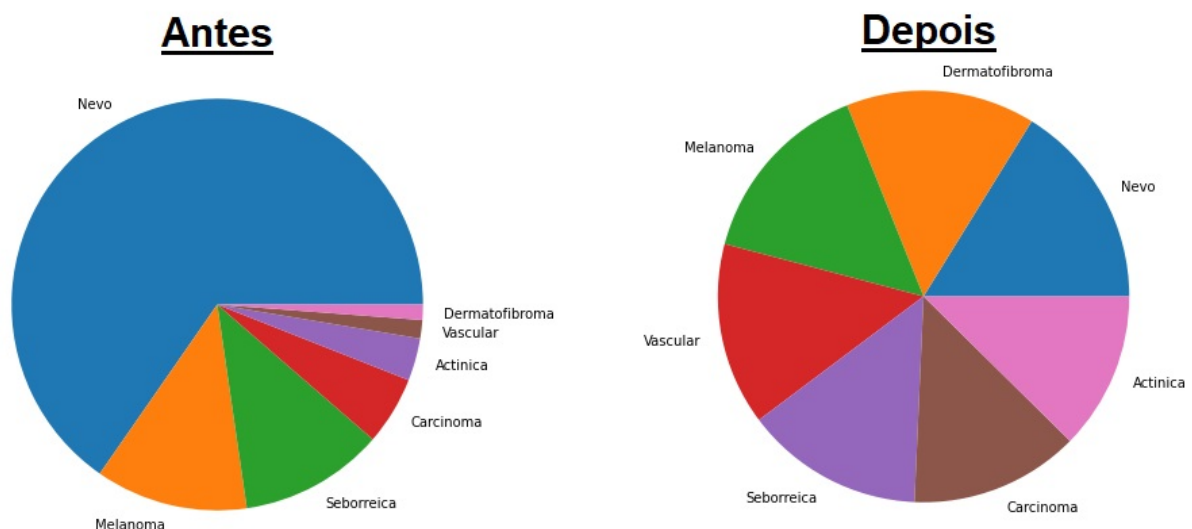
Tabela 4.3 – Conjunto de Treino Balanceado

Tipo	Quantidade
Melanoma	5335
Carcinoma	4790
Nevos Melanócitos	5822
Queratose Seborreica	5055
Queratose Actínicas	4455
Dermatofibroma	5350
Lesões Vasculares	5160

Fonte: Autor.

Para melhor ilustrar, a Figura 4.3 mostra graficamente a distribuição entre as classes do conjunto de imagens HAM10000 antes e depois do balanceamento feito para o conjunto de treino.

Figura 4.3 – Balanceamento de classes do conjunto de treino



Fonte: Autor.

#### 4.1.4 Treinamento e Teste

Para o início do treinamento, foi calculada a média e o desvio padrão de todas as imagens do *dataset*, que posteriormente serão utilizados para a normalização da transformação dos conjuntos de treino e teste. Esse cálculo é realizado para cada canal da imagem, que neste caso por ser uma imagem RGB, tem o total de 3 canais.

Para o entendimento dos métodos de treinamento e teste de uma CNN, é válido definir os conceitos do que chamamos de hiperparâmetros de aprendizado de uma RNA.

- Tamanho do *Batch* - A quantidade de amostras (imagens) que são utilizadas em cada iteração de treino.
- Taxa de Aprendizado - Valor escalar pelo qual são multiplicados os gradientes antes de aplicá-los aos pesos.
- Número de iterações de treino - Quantas iterações a rede é treinada.
- Número de épocas de treino - Quantas vezes a rede passa pelo conjunto inteiro no treinamento. O conjunto pode ser caracterizado pelo número de *batch* multiplicado pelo número de interações de treino, isso relacionado somente a porção de treinamento do *dataset*.
- Penalidade (Regularização) - Decaimento de peso ou *Weight decay*, utilizado para adicionar uma penalidade ou regularização no otimizador para cálculo de perda.
- Workers ou Num-Workers - Informa quantos subprocessos simultâneos serão feitos para o carregamento dos dados na unidade de processamento.

No que tange a ML, a entrada de uma CNN é vista como um tensor de ordem 3. Conforme o descrito, quando utilizado várias instâncias de amostras, o *batch*, é esperado uma entrada de um tensor de ordem 4. A ordem dos valores referente ao tensor de entrada esperado é (N, A, L, C), onde:

- N Tamanho do *batch*
- A Altura da imagem
- L Largura da Imagem
- C Numero de canais

Neste projeto, são utilizadas imagens coloridas, assim, C tem três números de canais (RGB), se as imagens fossem em escala de cinza, este número de canais seria apenas um. Para a altura da Imagem A, e a largura da imagem L, o valor é descrito em *pixels*, uma imagem 32x32 tem 32 *pixels* de altura e 32 *pixels* de largura.

As topologias de CNN esperam diferentes tamanhos de imagens como entrada da rede neural. Neste projeto, as arquiteturas de CNN utilizadas, vide seção 2.8, esperam imagens com tamanho 224x224 *pixels*.

Ocorreram treinamentos distintos para os três tipos de CNN utilizadas neste projeto. Foram carregados modelos com pesos pré-treinados, disponibilizados pela biblioteca

Pytorch, as camadas das redes foram apresentadas na seção 2.8, e os hiperparâmetros seguiram os valores de acordo com a Tabela 4.4.

Tabela 4.4 – Valores dos Hiperparâmetros das CNN

Hiperparâmetros	Valor
Tamanho do batch	32
Taxa de Aprendizado	1e-3
Épocas	20
Penalidade (Regularização)	0
Num-Workers	2

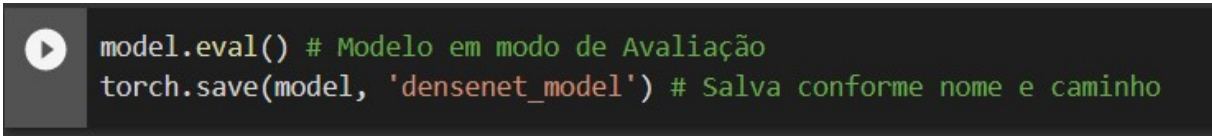
Fonte: (TSCHANDL, 2018)

O valor relacionado para o hiperparâmetro *num-worker* foi sugerido pelo ambiente de desenvolvimento do Google Colaboratory. Devido a seu *hardware*, a sugestão foi colocar o valor igual a 2 para um melhor desempenho no processamento da CNN.

Para o cálculo das funções de perda do treinamento da CNN foi utilizado o otimizador ADAM (PYTORCH, 2022a) e o critério de *loss* Cross Entropy (PYTORCH, 2022b). Ambos, disponibilizados pela biblioteca Pytorch, transformados em tensores e executados diretamente na GPU, métodos esse carregados com os hiperparâmetros descritos na Tabela 4.4.

O teste dos dados bem como o treinamento foram feitos para os três tipos de arquiteturas de CNN, onde cada época do teste é comparada para definir o melhor percentual de aproveitamento da rede. Uma vez que a rede convergir, conseguindo atingir resultados satisfatórios, o modelo é colocado em modo *eval* e realizado o salvamento para o posterior carregamento no algoritmo que classificará a imagem fornecida pelo usuário final.

Figura 4.4 – Código para salvar rede neural treinada



```
model.eval() # Modelo em modo de Avaliação
torch.save(model, 'densenet_model') # Salva conforme nome e caminho
```

Fonte: Autor.

A Figura 4.4 demonstra por linha de código o salvamento da rede neural treinada. É possível observar como a biblioteca Pytorch facilita o salvamento da rede, somente sendo necessário colocar a rede em modo de avaliação e indicar o caminho e o nome do arquivo onde o mesmo será salvo. Arquivo este que será carregado posteriormente na fase de *deploy* do software.



## 4.2 DEPLOY DO SOFTWARE

Nesta subsecção serão detalhados o desenvolvimento e os métodos utilizados para o *deploy* do software, como a criação da aplicação *mobile* e a comunicação dessa aplicação com a parte de classificação de imagens, através de métodos de API REST.

### 4.2.1 Ambiente de Desenvolvimento

As especificações do ambiente de desenvolvimento para a aplicação *mobile* e criação da API REST, bem como a linguagem de programação utilizada e suas bibliotecas são descritas a seguir:

- Ferramenta utilizada para desenvolvimento notebook Dell G7 7588
  - Sistema Operacional Windows 11
  - Processador Intel i7-8750H
  - Placa de Vídeo NVIDIA GTX 1050 TI
  - 8GB de Memória RAM DDR4
- Linguagem de Programação Python versão 3.7.13
  - Biblioteca JSON versão 1.6.1 formato utilizado para a troca de dados através de API REST
  - Biblioteca Requests versão 2.9.1 utilizada para realização de requisições entre servidor e cliente
  - Biblioteca Pillow versão 7.1.2 utilizada para manipulação vetorial das imagens
  - Biblioteca Pytorch versão 1.11.0 utilizada para o carregamento do modelo de CNN treinado para classificação
  - Biblioteca Flask versão 2.1.2 utilizada para a criação da API REST
  - Biblioteca Kivy 2.1.0 utilizada para a criação do aplicativo Android
  - Biblioteca Kivymd 0.104.2 utilizada para a modelagem gráfica do aplicativo Android
- Ambiente de Desenvolvimento Pycharm Community Edition 2021.3.3

### 4.2.2 Carregamento do Modelo de CNN Treinado

O primeiro passo para dar início ao *deploy* do software é o carregamento do modelo de CNN treinado no sistema de *backend*. Para realizar esta tarefa, com a biblioteca PyTorch é possível carregar o arquivo que foi gerado no treinamento mostrado na Seção 4.1.4.

A Figura 4.5 mostra como com poucas linhas de código é possível realizar o carregamento do modelo de CNN treinado para o código *backend*, apenas referenciando o local onde o arquivo se encontra. Também se faz necessário carregar o modelo na GPU, de acordo com o ambiente em que a arquitetura foi treinada, e colocar este modelo em fase de avaliação.

Figura 4.5 – Código para carregar rede neural treinada

```
model_ft = torch.load('densenet_model') # Carregamento da arquitetura (Pasta raiz)
model = model_ft.to(device) # Coloca o Modelo na GPU
model.eval() # Coloca o modelo em modo avaliação
```

Fonte: Autor.

Com o modelo carregado, já se torna possível realizar predições das imagens fornecidas ao algoritmo. Para isso é necessário realizar as transformações das imagens carregadas, do mesmo modo que usado para o treinamento da rede, deixando as mesmas com o tamanho e formato necessário que o modelo de CNN espera e carregando-as na GPU. A Figura 4.6 evidencia através de linhas de código o funcionamento de uma imagem quanto a sua classificação a partir do modelo carregado.

Figura 4.6 – Código de classificação de imagem fornecida

```
def get_prediction(image_bytes):
    tensor = transform_image(image_bytes=image_bytes).to(device) # Capta a imagem e coloca na GPU
    outputs = model.forward(tensor) # Passa a imagem pela CNN Rede treinada
    _, y_hat = outputs.max(1) # Retorna a classe de saída da rede como índice
    predicted_idx = str(y_hat.item()) # Identifica o índice na lista e retorna o nome da lesão
    return predicted_idx
```

Fonte: Autor.

A função *get\_prediction*, mostrada na Figura, 4.6 retorna o resultado da classificação da imagem de lesão de pele de acordo com uma imagem fornecida. Uma vez implementado este código no algoritmo de *backend*, o próximo passo é a realização de meios computacionais para o usuário fornecer a imagem a ser classificada por este algoritmo.

### 4.2.3 Desenvolvimento da Aplicação Mobile

Existem diversas ferramentas e linguagens de programação que possibilitam a criação de aplicativos para *smartphones* em diferentes sistemas operacionais, dentre elas se destacam o Java para Android e o Swift para IOS. Para a elaboração da aplicação *mobile* deste projeto foi definido que o SO de destino seria o Android, devido sua facilidade para desenvolvimento, seu fácil acesso e vasta utilização. Para a produção do aplicativo foi escolhida a linguagem de programação Python juntamente com a biblioteca Kivy, alguns fatores foram fundamentais nesta escolha, ser a linguagem de programação utilizada nos outros blocos do trabalho e também pelo fato de ser necessário processamento computacional maior na utilização de outros métodos de desenvolvimento, o que dificultaria as implementações e testes.

#### 4.2.3.1 Desenvolvimento Interface gráfica com Kivy

O desenvolvimento do *layout* do aplicativo foi feito através da biblioteca KivyMD <sup>2</sup>, uma extensão da biblioteca Kivy que possui uma coleção de *widgets* para uma estrutura de interface gráfica específica para aplicativos *mobile*. O *design* final da aplicação é mostrado no capítulo 5 onde são apresentados os resultados do projeto.

A criação da interface gráfica ocorre através de blocos, com carregamento de *widgets* disponibilizadas pela biblioteca, como botões, campos de textos e diversas outras ferramentas que podem ser encontrados em sua documentação. A Figura 4.7 demonstra, através de código, a criação de um botão na interface gráfica do aplicativo. É possível observar as definições de tamanho de letra e a posição do mesmo no perante ao tamanho da interface criada.

Figura 4.7 – Código para criação de um botão

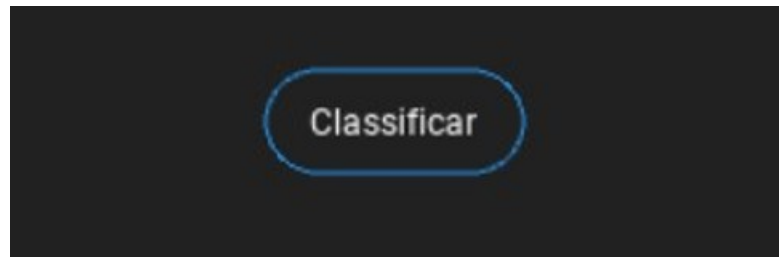
```
MDRoundFlatButton:
    text: "Classificar" # Texto do botão
    font_size: 12 # Tamanho da letra
    pos_hint: {"center_x": .5, "center_y": .25} # Posição na página
    on_press: app.file_chooser() # Função quando apertar botão
```

Fonte: Autor.

A biblioteca KivyMD possui modelos diferentes de *widgets*, com diferentes *designs* para estes componentes. Figura 4.8 apresenta o resultado da criação do botão com o *design* de MDRoundFlatButton de acordo com o código da Figura 4.7.

<sup>2</sup><https://kivymd.readthedocs.io/en/latest/>

Figura 4.8 – Layout do botão na interface gráfica



Fonte: Autor.

Terminado a criação das interfaces gráficas e finalizado todo o *layout* do aplicativo, foram implementadas as funcionalidades dos *widgets*, como o funcionamento de botões, o acesso a câmera fotográfica e também o funcionamento em conjunto com a API REST. Considerado funcional e testado é possível gerar um *Android Application Pack* (APK), que é um arquivo de instalação para a aplicação nos *smartphones* (NEXT PIT, 2020).

#### 4.2.4 API REST

Estabelecidos o modo como a imagem é captada através do aplicativo mobile e a classificação da imagem perante a rede neural, é necessário desenvolver o meio de comunicação entre as partes. Para realizar a troca de informação entre o *frontend* e o *backend* foi desenvolvida uma API REST em linguagem Python através da biblioteca Flask <sup>3</sup>.

Para o desenvolvimento da API REST, foi definida a comunicação entre cliente e servidor, essa comunicação, por escolha de desenvolvimento e facilidade para testes, foi feita utilizando um servidor local (*localhost*), ou seja, a comunicação entre o aplicativo e o classificador de imagens é feita na mesma máquina. Destaca-se que isto não é impeditivo para colocar o software *backend* em um servidor externo e distribuir a aplicação *mobile frontend* para algum usuário final em um momento futuro.

##### 4.2.4.1 Desenvolvimento do API REST com Flask

Foi definido a utilização do *framework web* Flask para a criação do API REST (FLASK, 2022), ferramenta esta que possui bibliotecas de desenvolvimento através da linguagem Python. Assim foram implementadas rotas para a solicitação entre cliente e servidor através de *request* e métodos HTTP.

---

<sup>3</sup><<https://flask.palletsprojects.com/en/2.1.x/tutorial/factory/>>

A Figura 4.9 mostra a criação de uma rota pela biblioteca Flask. A rota demonstra que o caminho `http://localhost:5000/predict` utilizando a porta 5000, através do método POST receberá um arquivo, nesse caso uma imagem, e encaminhará essa imagem fornecida pelo app para a função de classificação `get_predict` retornando o tipo da lesão de acordo com a CNN.

Figura 4.9 – Código para criação de uma rota para API REST

```
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST': # Se o método HTTP for POST realiza a sequência
        file = request.files['file'] # Biblioteca Request carrega o arquivo
        img_bytes = file.read() # Le o arquivo
        tipo = get_prediction(image_bytes=img_bytes) # Carrega a imagem na função de classificação
        return jsonify({'class_id': tipo, 'class_name': classe[int(tipo)]}) # Retorna o id e nome da lesão
```

Fonte: Autor.

O método HTTP utilizado foi o POST, o que possibilita envio de parâmetros juntamente com a requisição. Neste caso o POST envia a imagem fornecida pelo usuário final, o servidor recebe essa imagem e encaminha para o algoritmo de classificação (*backend*) que, após categorizar o tipo de lesão de pele, retorna através do modelo JSON o resultado para ser apresentado na aplicação *mobile* (*frontend*). A Figura 4.10 descreve uma requisição POST, através da rota criada, passando uma imagem para o funcionamento da API REST.

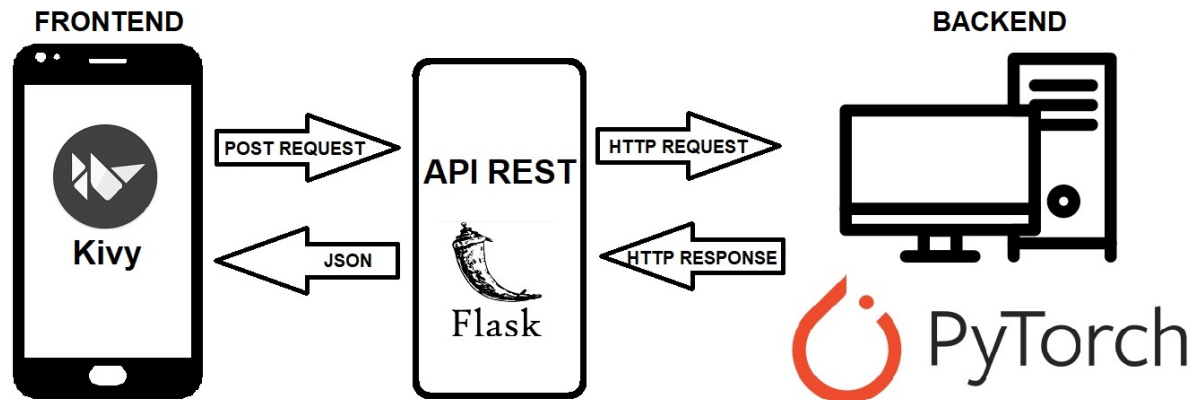
Figura 4.10 – Exemplo de um POST Request

```
resp = requests.post("http://localhost:5000/predict",
                    files={"file": open('D:/Workspace/HAM10000/reorganized/mel/ISIC_0024315.jpg', 'rb')})
```

Fonte: Autor.

De modo a exemplificar o fluxo de trabalho, a Figura 4.11 ilustra o funcionamento da API REST na comunicação entre o aplicativo e o software que realiza a classificação da imagem. Também clarifica as ferramentas utilizadas em cada módulo. A arquitetura de CNN empregada para a classificação da imagem foi o modelo que obteve o melhor resultado no teste da rede neural, esta é apresentada na seção 5 onde são discutidos os resultados do projeto.

Figura 4.11 – Emprego da API Rest



Fonte: Autor.

A orquestração de ferramentas permitiu o *pipeline* do software desde o treinamento até a existência de um sistema de informação, concretizando a possibilidade de uma imagem ser fornecida por um usuário e realizar a verificação desta como uma possível lesão de pele ou não. Todos os códigos desenvolvidos para a realização deste projeto estão disponibilizados em GitHub (2022).

## 5 RESULTADOS

Esta seção exibe os resultados obtidos no presente trabalho referente a validação da convergência das redes neurais apresentadas para um resultado que possa caracterizar-se como satisfatório, a comparação entre as CNNs, identificando qual teve o melhor desempenho por consequência implementando-a para o uso em conjunto com a aplicação mobile e, por fim, são mostradas as interfaces gráficas da versão final do software de classificação de lesões de pele.

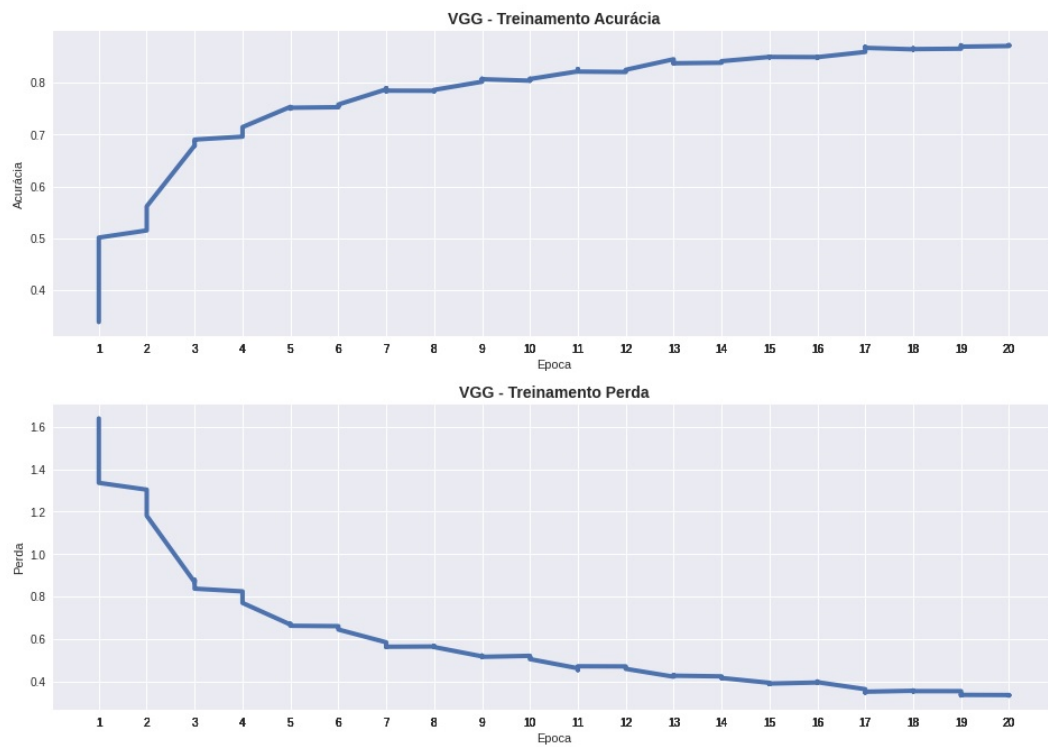
### 5.1 TESTE DAS REDES NEURAIAS CONVOLUCIONAIS

A análise do desempenho das CNN é feita através da existência da convergência ou não ao longo das épocas treinadas e do percentual de acerto que o modelo teve em fase de teste. As arquiteturas utilizadas apresentaram grau de convergência semelhante, foram realizados treinamentos de 50 épocas para cada modelo. Foi identificado que a partir da vigésima época, o treinamento não apresentava melhoras, assim, constatando em 20 o número de épocas ideais para o aprendizado.

#### 5.1.1 Resultados da Arquitetura VGG

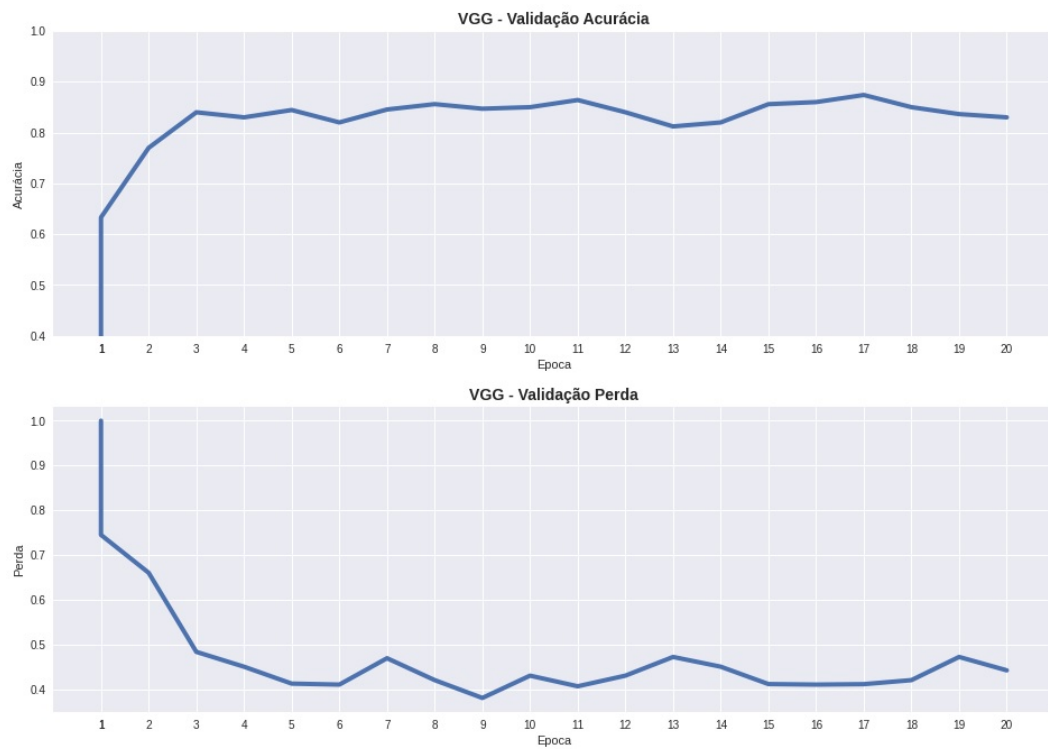
A arquitetura de CNN VGG 11 é, dentre os modelos utilizados no projeto, a rede que contém menos camadas, como é mostrado na seção 2.8, e iniciou o treinamento com um percentual de perda bastante elevado demonstrado na Figura 5.1. Porém, conforme o passar das épocas a perda foi atenuando até convergir conforme o esperado. O percentual de acurácia do treino teve um resultado satisfatório, conseguindo chegar a 87,31% na época 17 e se mantendo com valores aproximados a essa referência até o final do treinamento. Uma análise semelhante pode ser feita para o teste do modelo VGG 11, no entanto, o percentual de precisão cresceu de forma tênue como mostra a Figura 5.2, atingindo um valor máximo de 87,40%. Isto indica que o aprendizado da rede se deu de forma mais lenta e não obteve um avanço tão significativo.

Figura 5.1 – Treinamento VGG



Fonte: Autor.

Figura 5.2 – Teste VGG



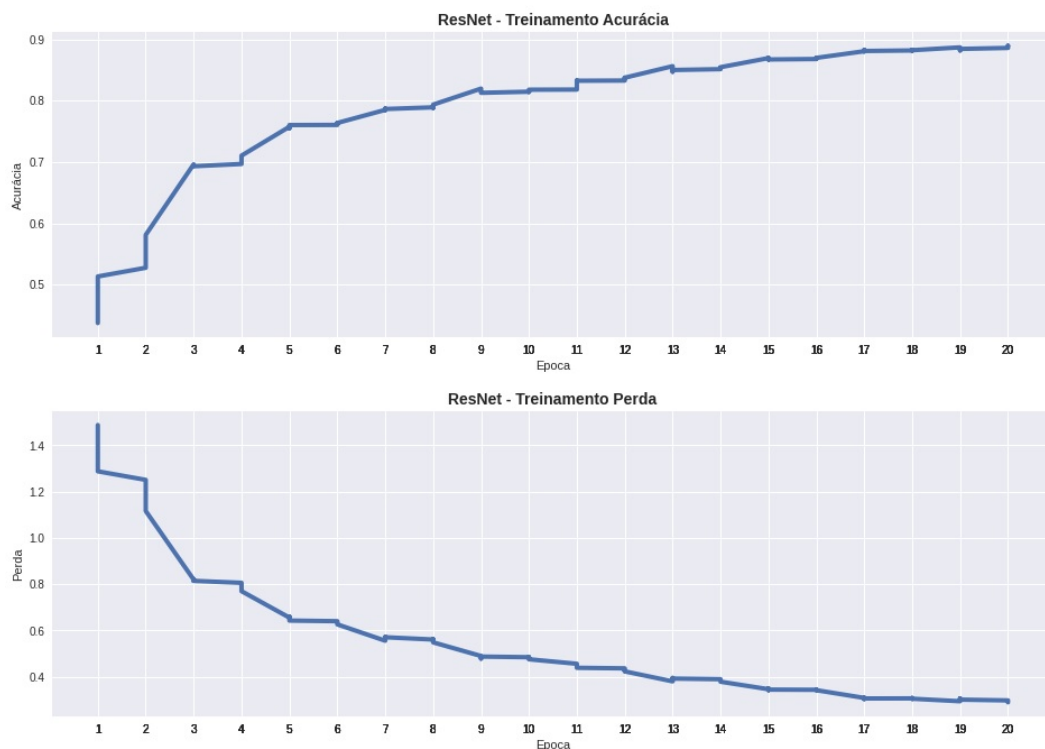
Fonte: Autor.



### 5.1.2 Resultados da Arquitetura ResNet

O comportamento mostrado pela CNN ResNet50 apresentou convergência tanto na fase de treinamento quanto na fase de teste, ambos para 20 épocas. A Figura 5.3 demonstra a atenuação da perda ao longo de todo o treinamento, bem como a acurácia deste treino, mantendo o aprendizado de forma constante. O melhor percentual de acertos no treinamento foi de 88,95%, mostrando um bom indicativo para o uso dessa rede.

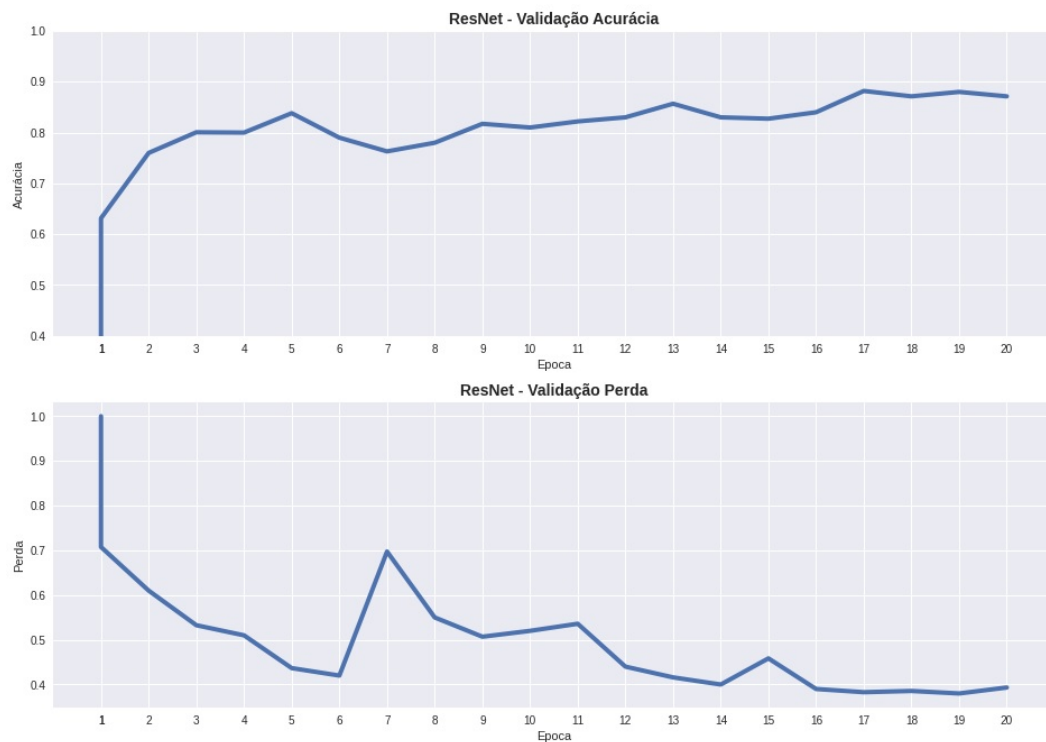
Figura 5.3 – Treinamento ResNet



Fonte: Autor.

A fase de teste do modelo ResNet50 demonstrou um grau de acurácia crescente, obtendo 88,19% de acerto em seu melhor desempenho na época 17, e posteriormente mantendo valores aproximados, indicando convergência e nenhum aprimoramento na rede após 20 épocas. O valor de perda da ResNet50 demonstrou uma certa oscilação, sobretudo na época 7, onde o valor aumenta de forma brusca, para em seguida voltar a atenuar e realizar a convergência como indica a Figura 5.4.

Figura 5.4 – Teste ResNet



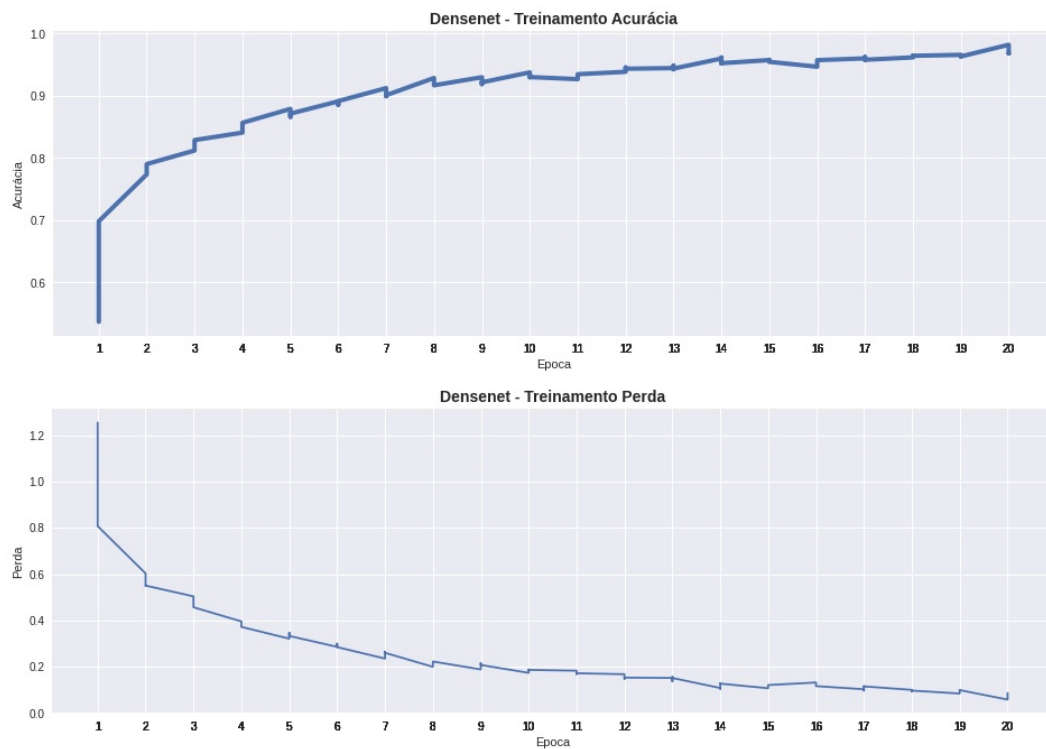
Fonte: Autor.

### 5.1.3 Resultados da Arquitetura DenseNet

Confirmando sua convergência tanto no treino quanto no teste. O treinamento da DenseNet apresentou atenuação da perda de forma satisfatória, chegando a valores próximos a zero, como mostrado na Figura 5.5, e o percentual de acerto obtendo a melhor marca de 98,22% em sua fase de treino.

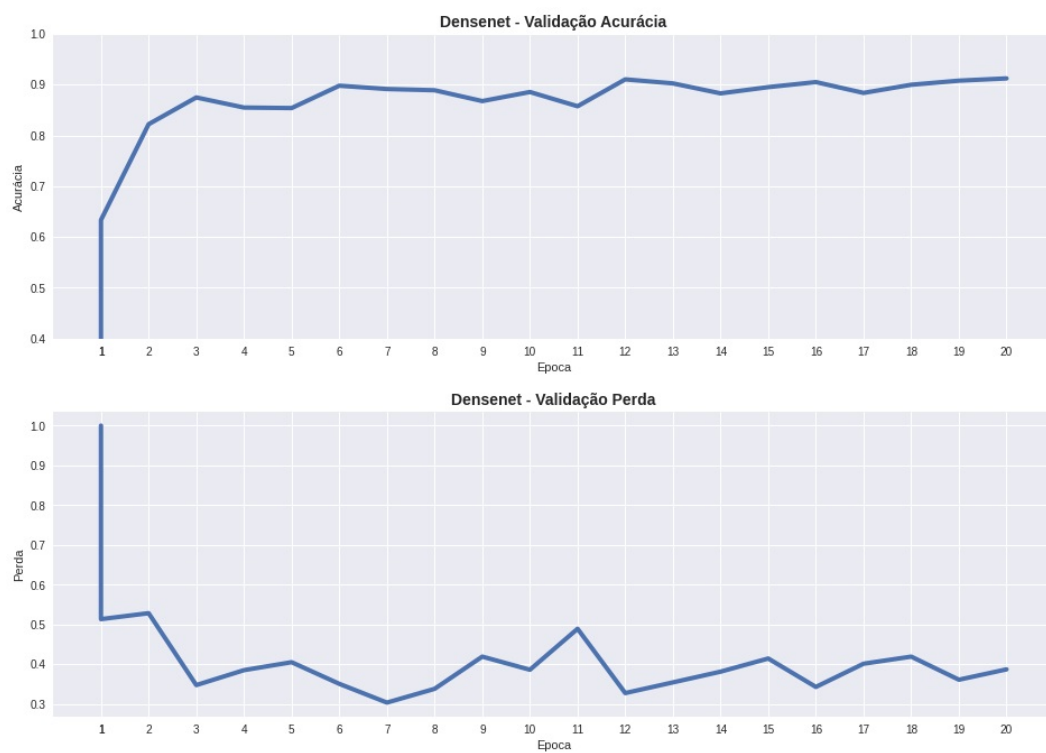
A Figura 5.6 mostra que a acurácia do teste do modelo DenseNet121 tomou um grau maior de aprimoramento no início do teste, posteriormente permaneceu aumentando de forma tênue até caracterizar sua convergência, conseguindo um percentual de acerto de 91,24% em sua melhor época. A perda operou de maneira semelhante, decaindo de forma mais acentuada no início do teste e depois tendo uma certa oscilação no meio do teste, voltando a diminuir em seguida.

Figura 5.5 – Treinamento DenseNet



Fonte: Autor.

Figura 5.6 – Teste DenseNet



Fonte: Autor.

## 5.2 COMPARAÇÃO DAS REDES NEURAIS CONVOLUCIONAIS

Após realizada a análise de desempenho e teste das arquiteturas de CNN utilizadas, se faz necessário a comparação dos resultados obtidos de cada modelo, com o intuito de definir qual rede teve o melhor desempenho para o problema de classificações de lesão de pele utilizando o conjunto de dados HAM10000. A Tabela 5.1 mostra este comparativo entre as três abordagens realizadas no projeto.

Tabela 5.1 – Comparativo entre as CNN Utilizadas no Projeto

<b>Modelo</b>	<b>Característica</b>	<b>Acurácia Treino</b>	<b>Acurácia Teste</b>
VGG11	Menos Camadas	87,31%	87,40%
ResNet50	Conexões Residuais	88,95%	88,19%
DenseNet121	Mais Camadas	98,22%	91,24%

Fonte: Autor.

Como exibido na Tabela 5.1, a arquitetura DenseNet121 apresentou os melhores resultados, tanto em relação ao treino quando ao teste, sendo assim a rede implementada juntamente com o software *mobile*, afim de contribuir para a classificação de imagens de lesão de pele. Os resultados corroboraram para afirmação de Huang et al. (2017), onde expressa que redes mais profundas, isto é, com mais camadas, são mais precisas e eficientes em seu treinamento.

## 5.3 SOFTWARE MOBILE

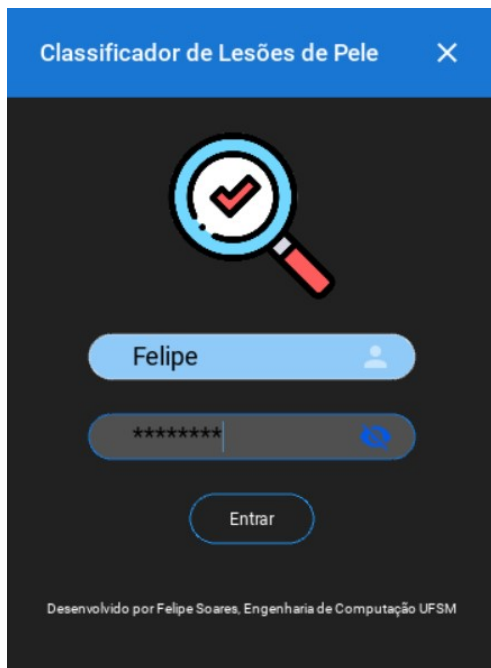
As ferramentas e métodos empregados para o desenvolvimento da aplicação foram detalhados na seção de desenvolvimento 4.2.3. Esta seção exhibe o layout das janelas do aplicativo bem como seu fluxo de trabalho e a determinação para o uso do software.

A Figura 5.7a mostra a interface gráfica inicial do aplicativo, onde o usuário deve inserir suas credenciais de acesso para obter as funcionalidades do sistema. Uma vez que o acesso é permitido o menu principal, Figura 5.7b, é mostrado ao utilizador, onde o mesmo tem a opção de tirar uma foto de alguma lesão de pele ou carregar fotos que se encontram sistema de arquivos do Android. Com a Imagem Carregada, basta clicar no botão de Classificar para aplicação enviar essa imagem pelo método HTTP POST através da API Rest como já mostrado na Figura 4.11.

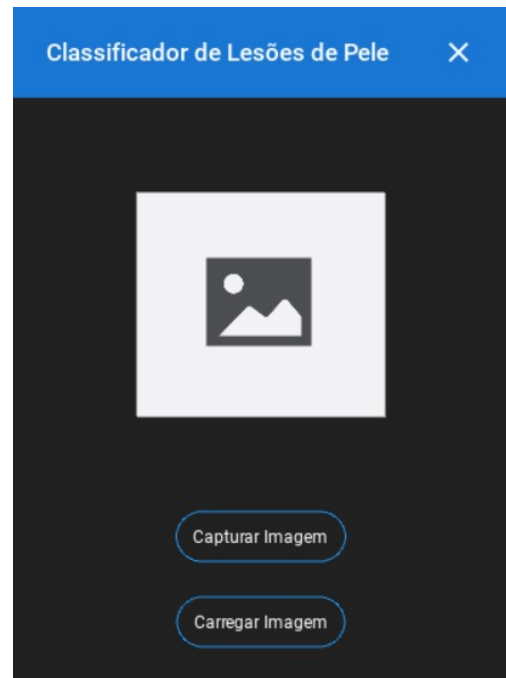
A Figura 5.8 indica o retorno do resultado da classificação do tipo de lesão de pele para a imagem fornecida pelo usuário, apresentando na interface gráfica qual classe a rede entendeu que a lesão informada pertence. Também é mostrado ao usuário uma mensagem de que a classificação da lesão de pele pelo aplicativo, de maneira nenhuma

Figura 5.7 – Interfaces gráficas iniciais do aplicativo

(a) Interface gráfica de Login



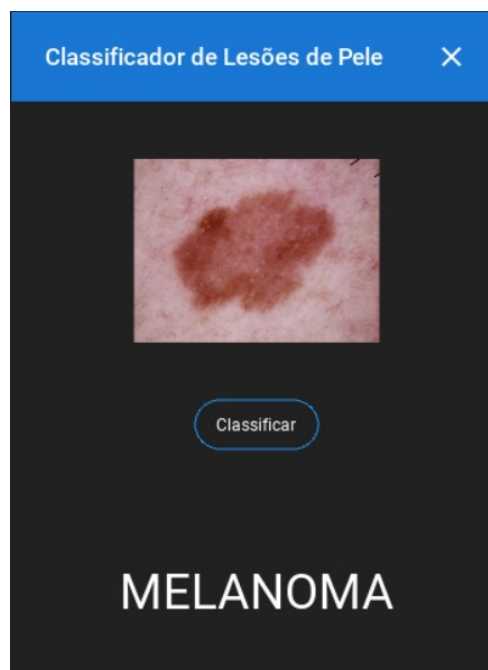
(b) Interface gráfica de carregamento de imagem



Fonte: Autor.

torna a consulta médica excludente, sendo necessário sempre procurar um especialista no assunto em caso de dúvidas.

Figura 5.8 – Interface gráfica de login do aplicativo



Fonte: Autor.

Assim, de acordo com os resultados apresentados, é possível classificar com sucesso as imagens fornecidas pelo usuário. Também constatar a eficácia das CNN utilizadas, verificando a densenet como a arquitetura de melhor desempenho entre as treinadas neste projeto. De mesma forma confirmar o funcionamento de um sistema de informação através das ferramentas utilizadas, realizando corretamente a comunicação entre os módulos de *backend* e *frontend* validando o *deploy* do sistema.

## 6 CONCLUSÃO

Este capítulo apresenta as considerações finais do trabalho, bem como os problemas enfrentados ao longo do desenvolvimento e também, trabalhos futuros que podem ser realizados partindo do ponto inicial apresentado.

Devido ao grande número de incidências de casos de câncer de pele, diversas pesquisas resultam diferentes técnicas para aprimorar e acelerar a detecção da doença e confirmar seu diagnóstico, visto que, quando detectado de forma precoce, as chances de cura das doenças relacionadas a pele aumentam consideravelmente. Com o objetivo de fomentar a discussão referente ao tópico e auxiliar na descoberta precoce de lesões de pele, o presente trabalho propôs uma aplicação *mobile* para classificações de tipos de câncer de pele, utilizando redes neurais convolucionais.

Primeiramente, foi realizado um estudo comparativo sobre três diferentes modelos de redes neurais convolucionais afim de constatar qual apresenta melhor performance e posteriormente, ser implementado junto com um software *mobile* onde a comunicação é realizada através de API REST. O estudo comparativo das redes neurais convolucionais, apresentou a DenseNet121 como o modelo de melhor desempenho para o conjunto de dados empregado, tendo em seguida a ResNet50 e a VGG11. Este resultado corrobora com o estudo onde arquiteturas que contém camadas mais profundas desempenham melhor na classificação de imagens, uma vez que DenseNet121 é a arquitetura que possui mais camadas dentre as estudadas e a VGG11 a menos densa entre elas, tendo o pior resultado experimental.

Os principais problemas encontrados no desenvolvimento foram problemas relacionados a processamento computacional. O computador pessoal utilizado para desenvolvimento carece de poder de processamento, sendo assim, se tornou necessário encontrar outros meios para realizar o treinamento da rede neural, como o Google Colaboratory, e diferentes abordagens para a criação do aplicativo, como a utilização da ferramenta Kivy com a linguagem Python ao invés de abordagens costumeiras como Java e Android Studio.

O funcionamento do software como um todo se mostrou satisfatório. A parte *mobile* realizando a integração com o sistema de classificação através de API REST, retomou conceitos estudados ao longo do curso de Engenharia de Computação e espera-se que esta aplicação possa ser aprimorada para novas funcionalidades em projetos futuros.

Também como trabalhos futuros pretende-se desenvolver um banco de dados com imagens de lesões de pele com características da população brasileira. Uma vez que as características, principalmente do tom de pele, dos *datasets* encontrados são de pessoas com pele predominantemente branca, o que não condiz com a maioria do povo brasileiro, onde também são diagnosticados tipos de câncer dermatológicos em tons de pele mais escuros.

É de grande importância salientar, que o presente trabalho não tem o intuito e não é autorizado a realizar diagnósticos de doenças, isto cabe a profissionais da área da saúde. A ferramenta tem a intenção de se apresentar como um auxílio e indica fortemente, uma vez que ocorram dúvidas em relação a qualquer enfermidade, a procura em um especialista responsável no assunto.

Por fim, espera-se que esse trabalho possa contribuir de alguma forma para uma maior discussão a respeito do tema proposto, possibilitando o surgimento de novas ideias e sugestões de melhorias.



## REFERÊNCIAS BIBLIOGRÁFICAS

ABDELHALIM, I. S. A.; MOHAMED, M. F.; MAHDY, Y. B. Data augmentation for skin lesion using self-attention based progressive generative adversarial network. **Expert Systems with Applications**, v. 165, p. 113922, 2021. ISSN 0957-4174. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0957417420307132>>.

ALBAWI, S.; MOHAMMED, T. A. M.; ALZAWI, S. Layers of a Convolutional Neural Network. **IEEE**, p. 16, 2017.

ALVES, J. V. P. et al. Variants of dermatofibroma - A histopathological study. **Anais Brasileiros de Dermatologia**, v. 89, n. 3, p. 472–477, 2014. ISSN 18064841.

ANDROID. **O Android e para todos**. [S.l.], 2016. Acesso em 01 jun. 2022. Disponível em: <[https://www.android.com/intl/pt-BR\\_br/everyone/](https://www.android.com/intl/pt-BR_br/everyone/)>.

\_\_\_\_\_. **Android 12**. [S.l.], 2021. Acesso em 01 jun. 2022. Disponível em: <[https://www.android.com/intl/pt-BR\\_br/android-12/](https://www.android.com/intl/pt-BR_br/android-12/)>.

BARDOU, D. et al. Hair removal in dermoscopy images using variational autoencoders. **Skin Research and Technology**, John Wiley and Sons Inc, v. 28, p. 445–454, 5 2022. ISSN 16000846.

BASAK, H.; KUNDU, R.; SARKAR, R. MFSNet: A multi focus segmentation network for skin lesion segmentation. **Pattern Recognition**, Elsevier Ltd, v. 128, aug 2022. ISSN 00313203.

BEDEIR, R. H.; MAHMOUD, R. O.; ZAYED, H. H. Automated multi-class skin cancer classification through concatenated deep learning models. **IAES International Journal of Artificial Intelligence**, v. 11, n. 2, p. 764 – 772, 2022. Cited by: 0; All Open Access, Gold Open Access.

BISSOTO, A. et al. (De) constructing bias on skin lesion datasets. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops**, v. 2019-June, n. 1c, p. 2766–2774, 2019. ISSN 21607516.

BISSOTO, A.; VALLE, E.; AVILA, S. Debiasing skin lesion datasets and models? not so fast. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops**, v. 2020-June, n. 1c, p. 3192–3201, 2020. ISSN 21607516.

CASTRO, P. B. et al. An app to detect melanoma using deep learning: An approach to handle imbalanced data based on evolutionary algorithms. **Proceedings of the International Joint Conference on Neural Networks**, p. 2–7, 2020.

FERENHOF, H. A.; FERNANDES, R. F. Passo-a-passo para construção da Revisão Sistemática e Bibliometria Utilizando a ferramenta Endnote. **Tutorial**, n. April 2014, p. 1–46, 2016.

FLASK. **Flask**. [S.l.], 2022. Acesso em 07 jun. 2022. Disponível em: <<https://flask.palletsprojects.com/en/2.1.x/>>.

GIRIRAJAN, S.; CAMPBELL, C.; EICHLER, E. Machine learning in medicine. **Physiology behavior**, v. 176, n. 5, p. 139–148, 2011.

GITHUB. **Aplicativo para Classificação de Lesões e Pele**. [S.l.], 2022. Acesso em 10 jul. 2022. Disponível em: <<https://github.com/felipemoraisssoares/SkinCancerClassification.git>>.

HAENSSLE, H. A. et al. Man against machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. **Annals of Oncology**, Oxford University Press, v. 29, p. 1836–1842, 8 2018. ISSN 15698041.

HARTANTO, C. A.; WIBOWO, A. Development of mobile skin cancer detection using faster r-cnn and mobilenet v2 model. **7th International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2020 - Proceedings**, p. 58–63, 2020.

HE, K. et al. Deep residual learning for image recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 2016-December, p. 770–778, 2016. ISSN 10636919.

HOSPITAL DA PLÁSTICA. **Nevo Melanocítico**. [S.l.], 2020. Acesso em 02 mai. 2022. Disponível em: <<http://www.hplas.com.br/especialidades-2/dermatologia/dermatologia-clinica/nevo-melanocitico/>>.

HUANG, G. et al. Densely connected convolutional networks. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, v. 2017-January, p. 2261–2269, 2017.

HUO, Y. Full-stack application of skin cancer diagnosis based on cnn model. In: . [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2021. p. 754–758. ISBN 9780738146492.

IA EXPERT ACADEMY. **Os tipos de redes neurais**. [S.l.], 2020. Acesso em 05 mai. 2022. Disponível em: <<https://iaexpert.academy/2020/06/08/os-tipos-de-redes-neurais/>>.

XIANG, K. et al. A novel weight pruning strategy for light weight neural networks with application to the diagnosis of skin disease. **Applied Soft Computing**, v. 111, 2021.

INFOQ. **API First Integration**. [S.l.], 2020. Acesso em 10 jun. 2022. Disponível em: <<https://www.infoq.com/articles/api-first-integration/>>.

INSTITUTO NACIONAL DE CÂNCER. Câncer de pele. 2021. Acesso em 02 mai. 2022. Disponível em: <<https://www.inca.gov.br/assuntos/cancer-de-pele>>.

\_\_\_\_\_. Câncer de pele melanoma. 2022. Acesso em 02 mai. 2022. Disponível em: <<https://www.inca.gov.br/tipos-de-cancer/cancer-de-pele-melanoma>>.

\_\_\_\_\_. Câncer de pele não melanoma. 2022. Acesso em 02 mai. 2022. Disponível em: <<https://www.inca.gov.br/tipos-de-cancer/cancer-de-pele-nao-melanoma>>.

KAUR, R.; GHOLAMHOSSEINI, H.; SINHA, R. Hairlines removal and low contrast enhancement of melanoma skin images using convolutional neural network with aggregation of contextual information. **Biomedical Signal Processing and Control**, v. 76, p. 103653, 2022. ISSN 1746-8094. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1746809422001756>>.

KHAN, S. et al. A Guide to Convolutional Neural Networks for Computer Vision. **Synthesis Lectures on Computer Vision**, v. 8, n. 1, p. 1–207, 2018. ISSN 2153-1056.

KIVY. **Kivy**. [S.l.], 2022. Acesso em 01 jun. 2022. Disponível em: <<https://kivy.org/#home>>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. ISSN 14764687.

MCCARRTHY, J. **What is artificial intelligence?** [S.l.]: Stanford University, 2017.

MOZILLA. **Métodos de requisição HTTP**. [S.l.], 2021. Acesso em 10 jun. 2022. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>>.

MYERS, D. J.; FILLMAN, E. P. **Dermatofibroma**. [S.l.]: StatPearls Publishing, 2022.

NATIONAL CANCER INSTITUTE. **vascular tumor**. [S.l.], 2018. Acesso em 05 mai. 2022. Disponível em: <<https://www.cancer.gov/publications/dictionaries/cancer-terms/def/vascular-tumor>>.

NEURONIO BR. **Entendendo Redes Convolucionais (CNNs)**. [S.l.], 2018. Acesso em 05 mai. 2022. Disponível em: <<https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>>.

NEXT PIT. **O que é um arquivo apk**. [S.l.], 2020. Acesso em 07 jun. 2022. Disponível em: <<https://www.nextpit.com.br/arquivo-apk-o-que-e>>.

O'SHEA, K.; NASH, R. An Introduction to Convolutional Neural Networks. p. 1–11, 2015. Disponível em: <<http://arxiv.org/abs/1511.08458>>.

PAIXÃO, G. M. d. M. et al. Machine Learning in Medicine: Review and Applicability. **Arquivos Brasileiros de Cardiologia**, v. 118, n. 1, p. 95–102, 2022. ISSN 16784170.

PEREZ, F. et al. Data augmentation for skin lesion analysis. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 11041 LNCS, n. i, p. 303–311, 2018. ISSN 16113349.

PRANAV, M. V. et al. Analyzing the Diagnostic Efficacy of Deep Vision Networks for Malignant Skin Lesion Recognition. In: **Proceedings of IEEE International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications, CENTCON 2021**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2021. p. 194–199. ISBN 9781665400176.

PYTORCH. **Adam Optimizer**. [S.l.], 2022. Acesso em 01 jun. 2022. Disponível em: <<https://pytorch.org/docs/stable/generated/torch.optim.Adam.html>>.

\_\_\_\_\_. **Cross Entropy Loss**. [S.l.], 2022. Acesso em 01 jun. 2022. Disponível em: <<https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html?highlight=crossentropyloss>>.

\_\_\_\_\_. **Pytorch Documentantion**. [S.l.], 2022. Acesso em 05 mai. 2022. Disponível em: <<https://pytorch.org/docs/stable/index.html>>.

RAHI, M. M. I. et al. Transfer learning approach and analysis for skin cancer detection. In: **2021 International Conference on Science Contemporary Technologies (ICSCT)**. [S.l.: s.n.], 2021. p. 1–6.

RAZZAK, I.; NAZ, S. Unit-Vise: Deep Shallow Unit-Vise Residual Neural Networks With Transition Layer For Expert Level Skin Cancer Classification. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, v. 19, n. 2, p. 1225–1234, 2022. ISSN 15579964.

RED HAT. **API REST**. [S.l.], 2020. Acesso em 10 jun. 2022. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>>.

SANTOS, A. et al. Uma Abordagem de Classificação de Imagens Dermatoscópicas Utilizando Aprendizado Profundo com Redes Neurais Convolucionais. p. 2010–2019, 2020.

SAÚDE, M. da. **Deteccção Precoce do Câncer | INCA - Instituto Nacional de Câncer**. [S.l.: s.n.], 2021. 84–85 p. ISBN 9786588517222.

SCHMITT, J. V.; MIOT, H. A. Queratoses actínicas: Revisão clínica e epidemiológica. **Anais Brasileiros de Dermatologia**, v. 87, n. 3, p. 425–434, 2012. ISSN 03650596.

SHIHADDEH, J.; ANSARI, A.; OZUNFUNMI, T. Deep learning based image classification for remote medical diagnosis. **GHTC 2018 - IEEE Global Humanitarian Technology Conference, Proceedings**, 2019.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–14, 2015.

SRINIVASU, P. N. et al. Classification of skin disease using deep learning neural networks with mobilenet v2 and lstm. **Sensors**, v. 21, n. 8, 2021. ISSN 1424-8220. Disponível em: <<https://www.mdpi.com/1424-8220/21/8/2852>>.

TSCHANDL, P. **The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions**. Harvard Dataverse, 2018. Disponível em: <<https://doi.org/10.7910/DVN/DBW86T>>.

UNIVERSIDADE FEDERAL DE SANTA MARIA. **Introdução a Machine Learning - PET Sistemas de Informação**. [S.l.], 2021. Acesso em 02 abr. 2022. Disponível em: <<https://www.ufsm.br/pet/sistemas-de-informacao/2021/05/11/introducao-a-machine-learning>>.

WIBOWO, A.; HARTANTO, C. A.; WIRAWAN, P. W. Android skin cancer detection and classification based on mobilenet v2 model. **International Journal of Advances in Intelligent Informatics**, Universitas Ahmad Dahlan, v. 6, p. 135–148, 7 2020. ISSN 25483161.

Wollina U. Recent advances in managing and understanding seborrheic keratosis. **F1000Research**, v. 8, p. 1–10, 2019. ISSN 2046-1402. Disponível em: <<https://pubmed.ncbi.nlm.nih.gov/31508199/>>.

YORADJIAN, A.; CYMBALISTA, N. C.; PASCHOAL, F. M. Queratose seborreica simuladora de Seborrheic keratosis that resemble melanoma aplicada. v. 3, n. 2, p. 169–171, 2011.

YOUNIS, H.; BHATTI, M. H.; AZEEM, M. Classification of skin cancer dermoscopy images using transfer learning. In: **2019 15th International Conference on Emerging Technologies (ICET)**. [S.l.: s.n.], 2019. p. 1–4.