ELIEL REGIANI
GUSTAVO VALENTIM DIAS

IDENTIFICADOR DE DOENÇAS EM PLANTAS POR REDE NEURAL CONVOLUCIONAL

São Paulo
2021

ELIEL REGIANI
GUSTAVO VALENTIM DIAS

IDENTIFICADOR DE DOENÇAS EM PLANTAS POR REDE NEURAL CONVOLUCIONAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com Ênfase em Computação.

São Paulo
2021

ELIEL REGIANI
GUSTAVO VALENTIM DIAS

IDENTIFICADOR DE DOENÇAS EM PLANTAS POR REDE NEURAL CONVOLUCIONAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com Ênfase em Computação.

Orientador:
Prof. Dr. Carlos Eduardo Cugnasca

São Paulo
2021

Prof. Dr. Carlos Eduardo Cugnasca

AGRADECIMENTOS

Inicialmente, prestamos agradecimentos aos nossos familiares, que nos ajudaram tanto com apoio emocional quanto com conhecimentos sobre o caminho trilhado, suas dificuldades, desafios e impacto no futuro.

Agradecemos ao professor doutor Carlos Cugnasca, que nos orientou, facilitou nosso planejamento e apoiou durante nossa jornada de criação do projeto.

Agradecemos imensamente a Escola Politécnica da USP, por nos agregar imenso valor, gerando um imenso conhecimento para o trabalho e oportunidades únicas para o futuro.

RESUMO

Com os avanços do 5G e seus possiveis impactos na agricultura global, novas aplicações estão sendo desenvolvidas, visando automatizar processos com o auxílio da conectividade. Tendo esse cenário em vista, foi desenvolvida uma aplicação para dispositivos móveis que permita ao usuário realizar consultas sobre qual fitopatologia atinge uma determinada cultura, por meio de uma simples foto. O projeto foi desenvolvido em três partes. Inicialmente, foi realizado um estudo das tecnologias envolvidas na criação de uma aplicação móvel híbrida, com tomada de decisão automática, por meio de redes neurais. Assim, foram levantados requisitos funcionais, que indicaram a escolha da plataforma, da arquitetura da aplicação e dos módulos a serem utilizados na implementação. Na segunda parte, foi apresentada a modelagem escolhida pelo grupo, para um aplicativo mobile com o intuito de proporcionar a identificação de fitopatologias. Na terceira e última, foram abordados os tópicos relativos à implementação da aplicação proposta, com foco no design da aplicação, modo de operação da rede neural e funcionalidades dos módulo de aplicação e servidor. Com esta estrutura, foi possível obter um aplicativo identificador de fitopatologias com alta precisão, chegando a mais de noventa porcento para algumas doenças, como a ferrugem, com elevada flexibilidade, funcionando tanto em *smartphones* com sistema operacional Android quanto iOS.

Palavras-Chave – fitopatologias. identificador. aplicativo. 5G. agricultura. inteligência artificial. redes neurais.

ABSTRACT

With the current 5G propagation, evolution and transformative impact in agriculture, new applications are being developed, focusing in decision making with the help of connectivity. In this scenario, it was developed an application for mobile devices that can discover which plant disease is striking certain herbs, using only one picture. The project was developed in three parts. In the first part, it was made a study about which technology was the ideal for a hybrid application, with automatic decision making, using neural networks. Furthermore, it was created functional requirements, responsible for indicating which platform, architecture and modules were going to be used in the implementation. In the second part, it was presented the modeling part for the mobile application, responsible for identifying the plant pathology. In the last part, it was presented the implementation for the application, how the neural network was programmed and works, the design established and special features of the application. Using that structure, the application was completed and had a precision above ninety percent for some diseases, like rust, and high flexibility, working on both Android and iOS smartphones.

Keywords – plant pathology's, identifier, application, 5G, agriculture, artificial intelligence, neural networks.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

MVP	Mínimo Produto Viável
ReLU	*Rectified Linear Unit*
MTV	*Model Template View*
API	*Application Program Interface*
HTTP	*Hypertext Transfer Protocol*
TCP	*Transmission Control Protocol*
JSON	*JavaScript Object Notation*
RNC	Rede Neural Convolucional
FFN	*Feed Forward Network*
IA	Inteligência Artificial

SUMÁRIO

PARTE I

INTRODUÇÃO

1 INTRODUÇÃO

O desenvolvimento e rendimento das plantas dependem do correto manuseio, fatores ambientais e cuidados corretos. Por consequência, qualquer fator que afete o seu desenvolvimento, como as fitopatologias, pode ocasionar perdas e reduzir sua utilidade.

1.1 Histórico

As doenças de plantas estão presentes de forma intensa e são conhecidas há muito tempo, desde os primórdios da agricultura. As referências mais antigas sobre fitopatologias podem ser encontradas na Bíblia e, sempre eram atribuídas a fatores espirituais e místicos.

Os gregos e hebreus, durante a antiguidade, tiveram tantos problemas com as fitopatologias que elas eram motivos de estudo de filósofos. O filósofo grego Teofrasto (372-287 A.C.) foi o primeiro estudioso a escrever sobre doenças de árvores, cereais e legumes (BERGAMIN FILHO, 1995).

Durante a idade média, as referências sobre doenças de plantas eram esparsas. Contudo, os árabes, no século X, publicaram um catálogo das doenças de plantas, principalmente dedicado às árvores frutíferas e à videira (BERGAMIN FILHO, 1995).

Com o desenvolvimento da Micologia e Botânica nos períodos seguintes, encontram-se relatos e evidencias sobre sintomas e até mesmo condições do ambiente que favoreceriam o desenvolvimento de fitopatologias. Segundo R. MOURA (2000), considera-se que, como ciência, seu início foi apenas no século XIX, em 1861 quando De Bary demonstrou que a causa da doença requeima da batata era um fungo, *Phytophthora infestans*.

Atualmente, grande parte dos produtores de plantas, sendo tanto agricultores quanto produtores domésticos, enfrentam problemas como doenças ou pragas em suas plantações, sendo causadas por insetos, fungos, vírus ou deficiência de nutrientes, o que prejudica grandemente o andamento de suas culturas e dificulta na economia ou sustento destes. Segundo Bergamin Filho, algumas das doenças mais comuns encontradas em plantas são

a ferrugem (Figura 1), o míldio e a fumagina, sendo essas objetos de estudo do presente projeto.

Figura 1: Planta contaminada com ferrugem, uma das fitopatologias mais comuns.

Fonte: extraído de (BERGAMIN FILHO, 1995).

1.2 Fitopatologia no Brasil

Segundo Costa (1975), a história inicial da fitopatologia no Brasil está ligada a estudiosos estrangeiros, que vieram ao país e estudaram problemas especiais de doenças relacionados com as plantas de importância para a agricultura do século XVIII como a cana-de-açúcar, cafeeiro, videira, batata e coqueiro.

Atualmente existem grupos de pesquisa e ensino em fitopatologia em praticamente todos os estados do Brasil, conforme apresentado por Bergamin Filho e Kitajima (2011).

1.3 Doenças comuns

Visando uma maior abrangência e um público alvo maior, as principais fitopatologias foram mapeadas. Com isso, informações foram coletadas para a inserção delas no aplicativo.

1.3.1 Ferrugem

A ferrugem (fungos da ordem Pucciniales) é uma das mais devastadoras enfermidades em plantas. É assim denominada devido à lesão em forma de massa pulverulenta de coloração amarela, vermelha ou, em alguns casos, branca.

1.3.2 Míldio

O míldio ocorre mais frequentemente em regiões com elevada umidade e temperaturas amenas (condições favoráveis). É causado, principal,emte, por um grupo de patógenos chamados de oomicetos e ataca preferencialmente as folhas, podendo ocorrer também em frutos e ramos.

1.3.3 Mela

A mela é uma fitopatologia que possui como principal sintoma a formação de cancros na região do colo da planta, os quais promovem a exudação de goma. É causada pelo fungo *Phytophthora parasitica* e afeta todos os órgãos da planta. No Brasil, ocorre principalmente em abacaxi, alho, berinjela e cebola (FILGUEIRA, 2000),

1.3.4 Podridão negra

A podridão negra, causada pela bactéria fitopatogênica Xanthomonas campestris, é a doença mais importante para o cultivo de brássicas no mundo (SCHUMANN, 2018). A doença pode ocorrer em qualquer estágio de desenvolvimento da planta.

Quando a infecção pelas bactérias ocorre nas plântulas, os cotilédones apresentam os bordos escurecidos, ocorrendo queda prematura. No campo o sintoma mais comum é o aparecimento de lesões e manchas amareladas, em forma de V, com o vértice voltado para o centro da folha.

1.3.5 Mancha Foliar

Caracteriza por lesões circulares sobre as folhas, a Mancha Foliar tem como principal prejuízo na cultura a redução da área fotossintética. A ocorrência do patógeno é generalizada em todos os locais de cultivo do morangueiro, sendo este seu principal alvo. Os

sintomas também podem ocorrer em pecíolos, estolhos, cálices, frutos, porém somente em casos de alta incidência.

1.4 Motivações

As doenças de plantas constituem um problema antigo da sociedade, estando presentes desde os primórdios da agricultura e, consequentemente, da civilização. Atualmente, segundo a EMBRAPA, elas causam prejuízos não somente para produtores rurais mas também para os consumidores em si, uma vez que diminuem a quantidade e a qualidade dos produtos agrícolas, afetando a economia e a biodiversidade da região.

Como problemas econômicos, é possível citar as perdas de produção de uma lavoura ou cultivos agrícolas, ou ainda o aumento de custos para controle de doenças por meio de agrotóxicos, inseticidas ou insumos agrícolas sintetizados.

Outro ponto a ser citado é também a desvalorização dos produtos no mercado, devido a lesões nos frutos, tamanho, ou mesmo qualidade, causadas pelas fitopatologias. Ademais, há o risco de redução das variedades de uma determinada planta suscetível a doenças, com a possível perda futura dessas variedades.

Isso é agravado pelo fato de grande parte dos produtores terem problemas na identificação e enfrentamento das doenças ou pragas em suas plantações e cultivos. Tal fato ocorre devido a, principalmente, falta de assistência técnica e informações centralizadas para isso, o que os obriga a sempre utilizar-se de formas de tratamento prejudicias para as plantas em si e para toda a fauna e flora do ambiente ao redor.

Tendo isso em vista, uma aplicação foi pensada para facilitar o processo de identificação das doenças em plantas. O intuito do projeto é propor uma forma dos produtores identificarem as doenças em suas culturas, de forma rápida e prática, obtendo informações necessárias para melhor tratar suas plantações, melhorando a qualidade de seus produtos agrícolas e a biodiversidade do meio ambiente.

1.5 Objetivo

Este projeto de formatura tem como objetivo principal propor uma forma de facilitar e automatizar processos de identificação de fitopatologias, auxiliando agricultores, produtores, jardineiros e usuários domésticos em seus cultivos. Para tanto, foi proposto um

sistema, que consiste em um aplicativo para celular, que por meio de fotos retiradas pelo usuário realiza a identificação da respectiva fitopatologia e fornece dados sobre prevenção, tratamento e outras informações gerais sobre a doença.

Com o uso do sistema, o complexo procedimento necessário para a identificação é simplificado, sem a necessidade de consultas com especialistas em botânica. Além disso, os produtores podem identificar as doenças em suas culturas, de forma rápida e prática, obtendo informações necessárias para melhor tratar suas plantações.

O sistema fornece também infomações de clima da localização do usuário, tais como temperatura, umidade e condições do tempo, informações úteis para o dia-a-dia do usuário.

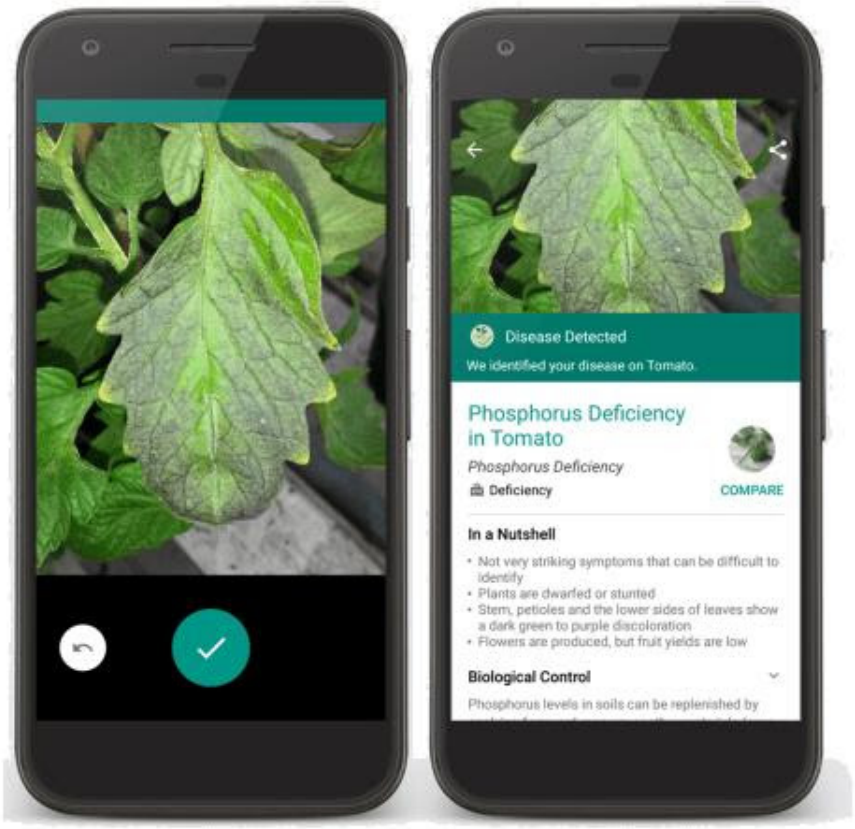
Com os avanços do 5G, aplicações que facilitem tarefas por meio da conectividade à internet serão cada vez mais requisitadas. Assim, o sistema, trará eficiência e redução de gastos para a agricultura, ganhando destaque no cenário do agronegócio.

1.6 Estado da arte

A fitopatologia é um tema comum em diversos projetos, com os mais diversos intuitos: identificação da doença, recomendações sobre cultivo e tratamento, identificação do tipo da planta e consulta com especialistas.

Como principal trabalho correlato, podemos citar o aplicativo *Plantix* (Figura 2), um aplicativo gratuito, desenvolvido por pesquisadores alemães, que identifica 30 das principais culturas, oferece recomendações e detecta mais de 400 danos às plantas através de fotos da planta doente.

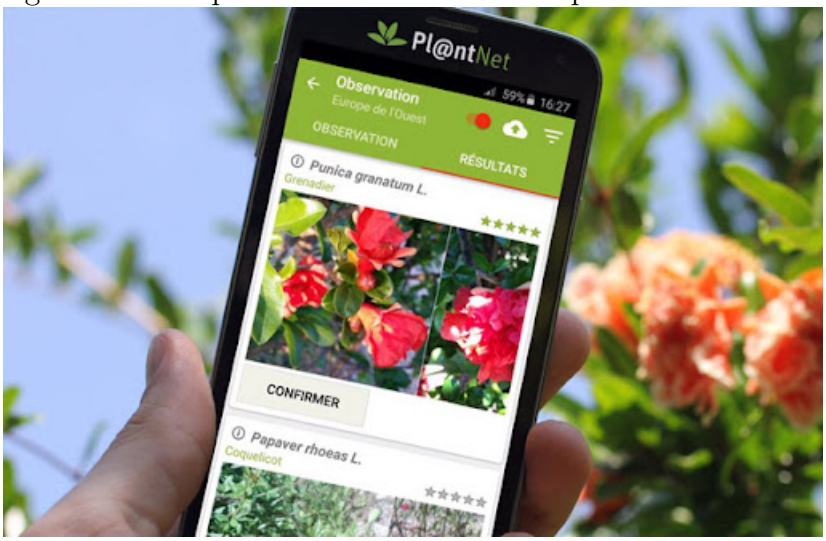
Figura 2: Aplicativo PLANTIX.



Fonte: extraído de Plantix, 2015.

Outro projeto de destaque é o *PlantNet* (Figura 3), um aplicativo de identificação de espécies de plantas por meio de fotos nítidas. O reconhecimento da espécie é realizado a partir de um banco de dados com hortaliças e plantas ornamentais cadastradas.

Figura 3: Exemplo de interface e uso do aplicativo PlantNet.



Fonte: extraído de PLANTNET, 2016.

1.7 Metodologia

A metodologia empregada é iniciada com a definição do tema e objetivos do trabalho e finalizada com o desenvolvimento de um produto mínimo viável (MVP) do projeto.

1.7.1 Definição do Tema

A definição do tema foi feita após pesquisa dos integrantes do grupo sobre a situação atual da agricultura no Brasil. Com isso, foi observado o problema em relação as identificação de doenças em plantas e foi levantado o escopo do projeto, levando-se em conta o cronograma estipulado pelo Departamento de Engenharia de Computação e Sistemas Digitais da USP. Assim, foi delimitado o escopo pelo que seria possível de se realizar a partir desses limites temporais.

1.7.2 Definição do modelo de Rede Neural

Para a etapa de definição do modelo da rede neural do projeto, foi feita uma intensa pesquisa bibliográfica e estudo de conceitos de aprendizado de máquina e redes neurais, reforçando-se os conhecimentos teóricos aprendidos durante a graduação.

Após isso, foi feito um estudo mais objetivo sobre as técnicas de reconhecimento de padrões e computação visual necessárias para o projeto. Com isso, foi possível desenvolver alguns modelos para a identificação das doenças em plantas do projeto. Esses modelos foram treinados a partir de uma *dataset* encontrado pelo grupo de doenças em plantas rotulado e, a partir dos resultados obtidos, foi possível definir o modelo certo para o projeto, com a acurácia requerida pelo escopo definido.

1.7.3 Criação do Protótipo

Para esta etapa, foi feito o desenvolvimento do aplicativo e a construção de um MVP, utilizado para validação da proposta, simulações, obtenção de novos requisitos e testes.

Inicialmente, foram realizados testes no *Figma*, para a geração de debates acerca do projeto, realização de brainstorm sobre o desenvolvimento do produto e a obtenção do *design* final desejado para a aplicação.

Para a detecção de padrões em imagens, com a ampla utilização de inteligência ar-

tificial em aplicações modernas, foram selecionadas, para a tomada de decisões sobre as imagens, as redes neurais convolucionais.

Com as informações sobre o produto final desejado, foi desenvolvido o MVP, separado em dois módulos principais: aplicação e servidor. O módulo de aplicação é responsável pela interação com o usuário, enquanto o servidor realiza o processamento da imagem e abriga a rede neural. Assim, utilizando as tecnologias React Native, para a construção do módulo de aplicação, e Django, para o módulo do servidor, a execução do MVP foi iniciada.

1.7.4 Elaboração da monografia

A redação da monografia foi feita com a linguagem Latex por tratar da formatação do texto de forma simples e automática. Para isso, utilizou-se a plataforma Overleaf (OVERLEAF, 2021), que fornece controle de versionamento de documentos e permite edição online e concorrente.

1.8 Organização do trabalho

O presente trabalho foi documentado da seguinte forma: A parte 1 apresenta a introdução do projeto, consistindo de 8 seções que vão desde o histórico do tema metodologia adotada pelo grupo para a concepção e validação do projeto. A seção 6 informa sobre projetos correlatos, suas principais conquistas e características.

Ademais, a parte 2 apresenta o desenvolvimento do projeto, consistindo dos aspectos conceituais utilizados pelo grupo na execução deste, das tecnologias envolvidas na criação do sistema do projeto, das especificações levantadas pelo grupo para o projeto e, também, a implementação em si do sistema com base nos principais requisitos funcionais e não funcionais levantados pela seção de especificações.

A parte 3 compreende o planejamento dos testes para a validação da solução e resultados obtidos, além também de abordar os próximos passos para a finalização do projeto.

PARTE II

DESENVOLVIMENTO

2 ASPECTOS CONCEITUAIS

Essa seção se destina a apresentar os conceitos teóricos utilizados no desenvolvimento do sistema proposto no projeto.

2.1 Aprendizado de Máquina

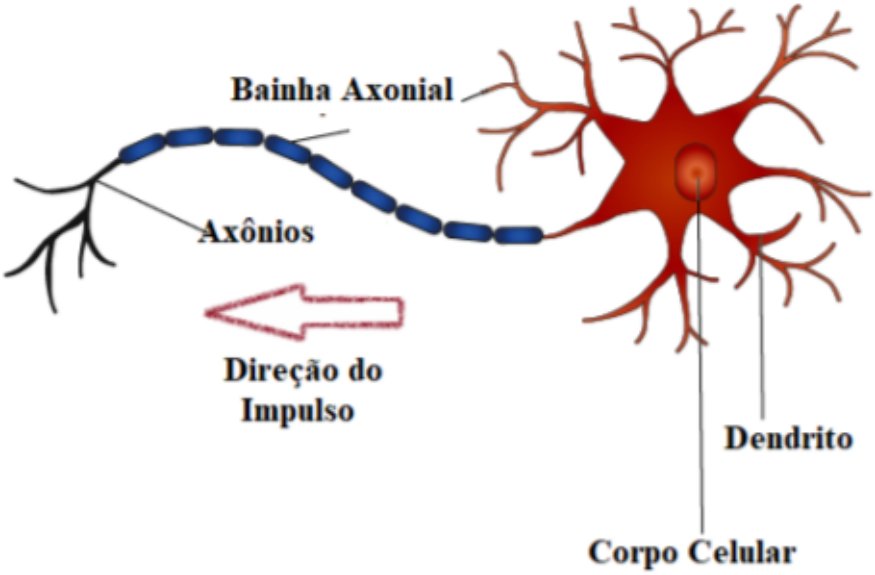
O aprendizado de máquina (do inglês *machine learning*), consiste na criação de algoritmos com o fim de permitir a um computador aprender sem ser explicitamente programado para tal. Um sistema baseado em algoritmos de aprendizagem é capaz de, pouco a pouco, melhorar os seus resultados à medida em que vai sendo exposto a dados e consegue acumular experiência a partir deles, algo similar ao que acontece no aprendizado humano (NATALI, 2021).

2.2 Redes Neurais Artificiais

Redes neurais é uma área de aprendizado de máquina importante na atualidade e constituem técnicas computacionais que apresentam um modelo matemático inspirado no sistema nervoso de seres vivos. O sistema nervoso é formado por um conjunto complexo de células, chamados neurônios. Eles têm um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. De acordo com Arbib (2002), o neurônio é constituído por três partes principais: o corpo celular, que possui algumas ramificações denominadas de dendritos, e por fim outra ramificação descendente do corpo celular mais extensa chamada axônio. Nas extremidades dos axônios estão os nervos terminais dos quais é realizada a transmissão das informações para outros neurônios, esta transmissão é conhecida como sinapse (Figura 4).

A informação transmitida pelos neurônios na realidade são impulsos elétricos que, segundo Reis (2008), consitituem sinais que correspondem a uma mensagem transmitida

Figura 4: Esquema simplificado de um neurônio.



Fonte: (ARBIB, 2002).

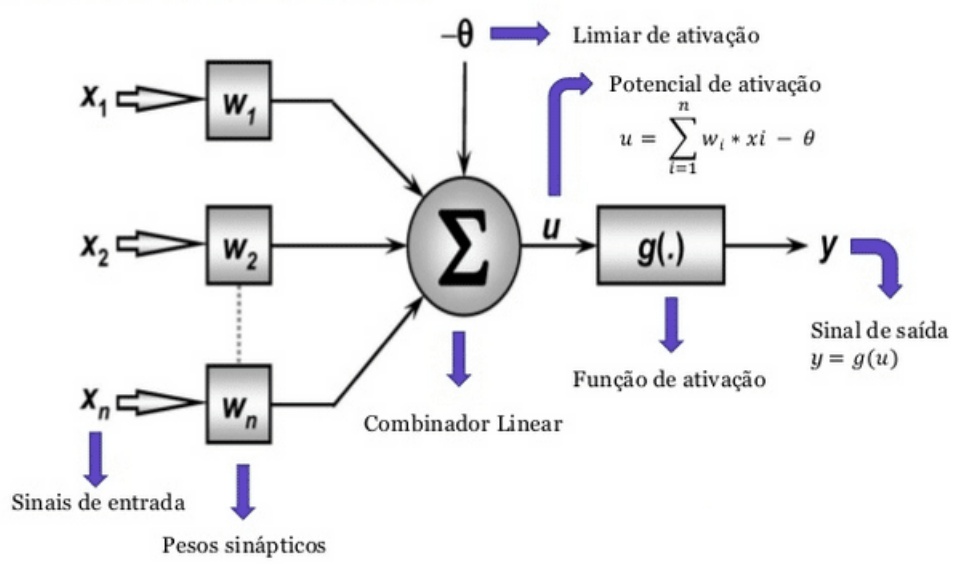
entre neurônios.

A partir da estrutura e funcionamento do sistema nervoso biológico, mais especificamente do neurônio, surgiu um modelo matemático que procura aproximar-se do funcionamento do neurônio biológico. Neste modelo, os impulsos elétricos provenientes de outros neurônios são representados pelos chamados sinais de entrada, x_n. Dentre os vários estímulos recebidos, alguns excitarão mais e outros menos o neurônio receptor, essa medida de quão excitatório é o estímulo é representada no modelo de McCulloch e Pitts (Figura 5) através dos pesos sinápticos.

Uma rede neural artifical típica é constituída por um conjunto de neurônios interligados que influenciam uns aos outros e formam um sistema maior. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento, ou neurônios. Tais modelos tem a capacidade de adquirirem conhecimento através da experiência, armazenando conhecimento adquirido por meio de exemplos apresentados e assim podendo realizar inferências sobre novos exemplos desconhecidos (ARBIB, 2002).

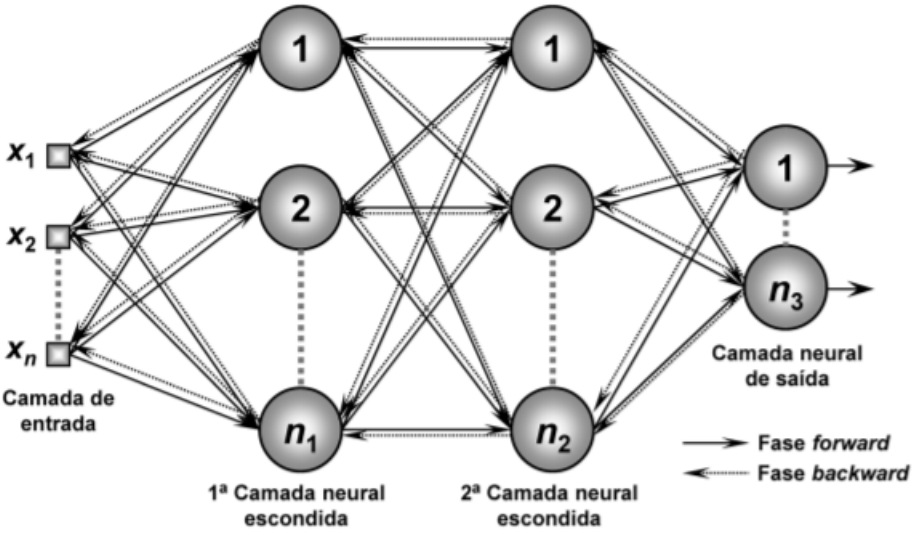
As redes neurais são treinadas, com o intuito de realizar uma transformaçaõ não linear na entrada e chegar-se a resultados em que o erro é aceitável. Para minimizar o erro, o treinamento é feito modificando os pesos até se obterem resultados satisfatórios.

Figura 5: Modelo de neurônio artificial.



Fonte: (ARBIB, 2002).

Figura 6: Modelo de rede neural artificial.



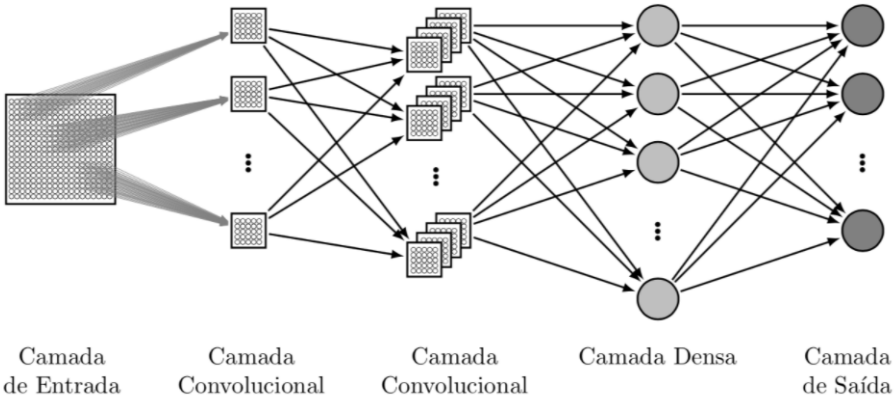
Fonte: (ARBIB, 2002).

2.3 Redes Neurais Convolucionais

As redes neurais convolucionais são estruturas de aprendizado de máquina que trabalham com entradas matriciais. Tais redes possuem camadas convolucionais, que realizam a operação matemática da convolução. A camada convolucional é responsável por extrair as chamadas *features* da entrada. O processo se dá por meio de filtros que percorrem os dados de entrada em suas três dimensões de entrada, realizando a operação de convolução sobre os dados. Com o percurso da entrada pela rede, os filtros vão aprendendo estruturas cada vez mais complexas, porém isso tem um custo de memória e processamento que

precisa ser balanceado na hora de definir a arquitetura.

Figura 7: Exemplo de topologia da Rede Neural Convolucional.

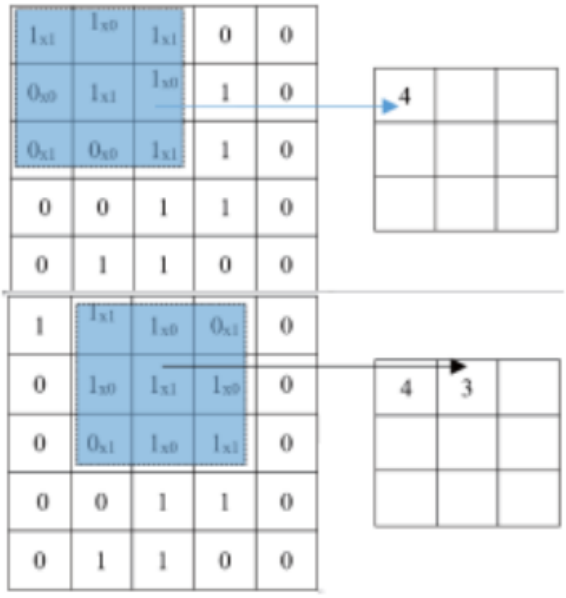


Fonte: (ARBIB, 2002)

2.3.1 Camada convolucional

Camadas convolucionais têm a função de registrar a saída de camadas anteriores, que consiste em pesos e tendências que são aprendidas. Nessa camada, uma sequência de processos matemáticos são feitos para se extrair as chamadas *features* da imagem de entrada. A Figura 8 exibe a operação de convolução de camada para uma imagem de entrada de tamanho 5x5, cujo resultado corresponde a um filtro 3x3. A figura também mostra o deslocamento do filtro começando pelo canto superior esquerdo da imagem de entrada. Os valores de cada passo são então multiplicados por valores do filtro e a soma ods valores correspondem ao resultado. Uma nova matriz com tamanho reduzido é formada da imagem de entrada (SAMMY, 2019).

Figura 8: Exemplo de processo de convolução.

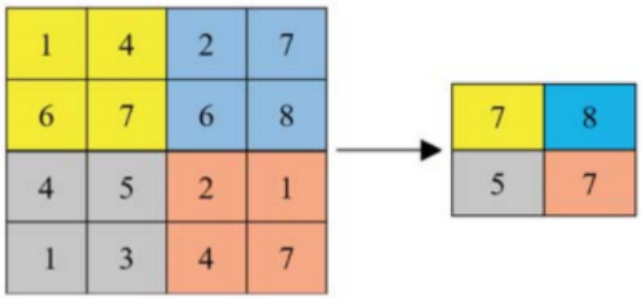


Fonte: (SAMMY, 2019).

2.3.2 Camada *Pooling*

A camada de Pooling tem a função de reduzir *overfitting* e de reduzir o número de neurônios da rede. A Figura 9 ilustra um exemplo de operação de *pooling*. Essa camada reduz o número de parâmetros, tempo de treino, taxa de computação e controla o *overfitting*. *Overfitting* é definido quando o modelo atinge acurácia de 100% ou próximo desse valor no *dataset* de treino mas uma taxa bem menor (cerca de 60%) no *dataset* de validação (SAMMY, 2019).

Figura 9: Exemplo de processo de *pooling*.

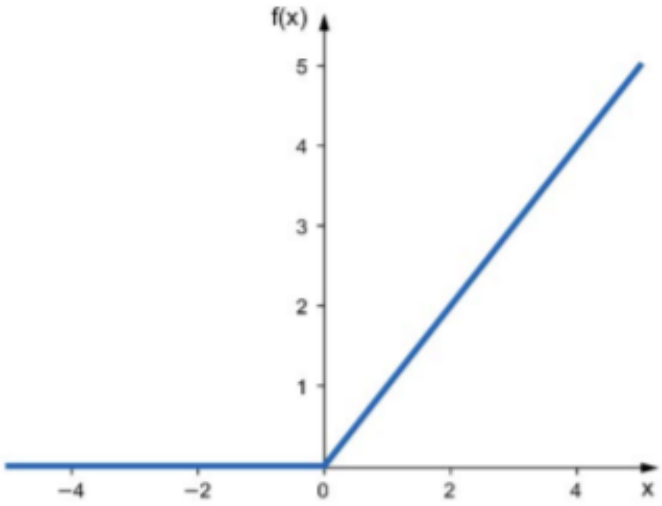


Fonte: (SAMMY, 2019).

2.3.3 Camada de ativação

Usualmente utiliza-se a função *Rectified Linear Unit* (ReLU) como função de ativação em todas as camadas convolucionais. A única exceção seria na camada de saída da rede, em que utiliza-se a função *softmax*. A função de ativação ReLU possui a função de criar um limite para cada entrada após a camada de convolução. Dessa forma, esta pode ser definida pela seguinte função $max\{0, x\}$ (ARBIB, 2002). A Figura 10 representa a função ReLU.

Figura 10: Representação da função ReLU.



Fonte: (ARBIB, 2002).

2.3.4 Arquitetura *MobileNet*

Uma arquitetura de rede neural bastante importante para o escopo do projeto é arquitetura *MobileNet*. Criada pelo *Google* em 2017, consiste em uma arquitetura destinada a ser usada em aplicações mobile para computação visual. Pequena e com baixa latência, foi desenhada para maximizar a acurácia evitando exigir grande processamento de *smartphones* e aplicações embarcadas (BENGIO, 2016).

2.3.5 *Transfer Learning*

Transfer Learning é uma técnica de *machine learning* onde um modelo treinado desenhado para uma tarefa é utilizado como ponto de partida para uma segunda tarefa (BENGIO, 2016). É uma abordagem popular em *deep learning* onde modelos pré-treinados são usados como ponto de partida em tarefas de computação visual e processamento de

linguagem natural, dados os vastos recursos de processamento e tempo exigidos para desenvolver uma rede neural para resolver tarefas em tais problemas, e também pelo grande aumento de desempenho fornecidos. Segundo (SORIA, 2009), *Transfer learning* é uma otimização que permite progresso rápido ou performance aperfeiçoada quando modelando uma segunda tarefa.

3 TECNOLOGIAS UTILIZADAS

Nesta seção são descritas as tecnologias específicas escolhidas pelo grupo para a execução deste projeto.

3.1 *Python*

O *Python* é uma linguagem de programação de alto nível interpretada. É conhecida por ser extremamente versátil, devido a sua simplicidade e tipagem forte e dinâmica (PYTHON, 2021).

A escolha de tal linguagem para a implementação da lógica da solução baseia-se fundamentalmente nessas características, as quais também possibilitam a ampla utilização do *Python* nas principais soluções e *frameworks* existentes no campo de aprendizado de máquina, como, por exemplo, *Tensorflow* e *scikit-learn*.

3.1.1 *NumPy*

NumPy é um pacote de código aberto para *Python* que viabiliza o uso de estruturas de dados poderosas, principalmente arrays multidimensionais, e a realização de cálculos numéricos e álgebra linear (NUMPY, 2021). Foi utilizado tanto para trabalhar com a estrutura das imagens quanto para realizar operações sobre as mesmas.

3.1.2 *Django*

Criado em 2005 por Adrian Holovaty e Simon Willison, é um *framework* web de código aberto baseado em *Python* e que utiliza o padrão *model-template-view* (MTV) (DJANGO, 2021). No projeto, servirá como *Back-end* da aplicação a ser desenvolvida.

3.1.3 PIL

PIL (*Python Imaging Library*) corresponde a uma biblioteca de gerenciamento de imagens do python (PILLOW, 2021). No projeto, foi utilizado o Pillow no desenvolvimento do *back-end* em *django*, que corresponde a uma ramificação do PIL para versões mais recentes de *Python*.

3.1.4 OpenCV

OpenCV (*Open Source Computer Vision Library*) corresponde a uma biblioteca de código aberto desenvolvida em C++, mas possuindo suporte para *Java*, *Python* e *Visual Basic*, popular nos campos de *machine learning* e visão computacional (OPENCV, 2021). No projeto, a biblioteca foi utilizada para fazer o redimensionamento das imagens das plantas.

3.2 *React Native*

React Native é uma biblioteca para desenvolvimento de interfaces gráficas com interfaces gráficas de usuário que possam executar como aplicações nativas em sistemas operacionais móveis. Ela é baseada na biblioteca *React* (ou *ReactJS*) para o desenvolvimento de páginas *Web* e em seu conceito de tornar o código de interfaces mais declarativo. Isso é feito ao utilizar funções para retornar novos elementos de interface em lugar de usá-las somente para modificar elementos declarados estaticamente, como ocorre em páginas HTML (OCCHINO, 2015). A tecnologia foi lançada em 2015 pela Facebook e tem compatibilidade com os sistemas móveis *Android* e *iOS*.

3.3 *TensorFlow*

É um conjunto de bibliotecas, ferramentas, *datasets* e recursos que auxiliam no desenvolvimento de soluções de *machine learning* e treinamento de modelos. É uma ferramenta *open-source* mantida pela *Google* e desenvolvida em *Python* e C++ (TENSORFLOW, 2021).

3.4 *Keras*

É uma API de alto nível, em *Python*, construída em cima do *TensorFlow* e de outras bibliotecas, que abstrai e simplifica o desenvolvimento de protótipos de *machine learning* (KERAS, 2021). Com o *Keras*, desenvolveu-se um código mais compreensível e com maior facilidade. Além disso, essa API é amplamente referenciada na literatura e utilizada em várias outras bibliotecas.

4 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Nesta seção será descrito o projeto do sistema, apresentando seus principais componentes e discutindo possibilidades de implementação, levando em consideração as definições e conceitos do tópico anterior.

4.1 Requisitos do sistema

A fim de resolver as necessidades dos usuários do sistema, é necessário atender aos requisitos, funcionais e não funcionais, fundamentais na experiência de usuário, e para o efetivo funcionamento do identificador de plantas. Seguem abaixo os principais requisitos para o protótipo.

4.1.1 Requisitos funcionais

Para a aceitação do sistema e melhor experiência do usuário, foram elencados os requisitos funcionais fundamentais para a aplicação, indicadas abaixo:

- Capacidade de receber e enviar dados de imagens obtidas através de fotos tiradas pelo celular, como também armazenadas em memória, a um servidor para processamento;
- Tela principal simples, sem a exigência de login e autenticação, com apenas botões para o envio da foto e visualização do histórico de identificações;
- Tela do histórico de identificação do usuário, com as informações obtidas em outras requisições feitas ao servidor;
- Informar dados relativos à doença identificada na planta e formas de tratamento;

- Informar dados sobre geolocalização das fotos tiradas;
- Informar dados de clima da região onde se encontra o produtor;
- Informar dados de culturas de interesse do produtor;
- Fornecer identificação precisa de, no mínimo, uma fitopatologia.

4.1.2 Requisitos não funcionais

Com parte do sistema elencado à internet, se faz necessário a definição de requisitos não-funcionais, para o funcionamento adequado do servidor:

- Funcionamento nos smartphones modernos;
- O sistema deve se comunicar com o servidor Web, por meio do protocolo HTTP;
- O sistema deve se comunicar com um banco de dados relacional;
- A interface deve ser agradável e de fácil utilização;
- O resultado do processamento da foto deve ser enviado de forma rápida ao aplicativo.

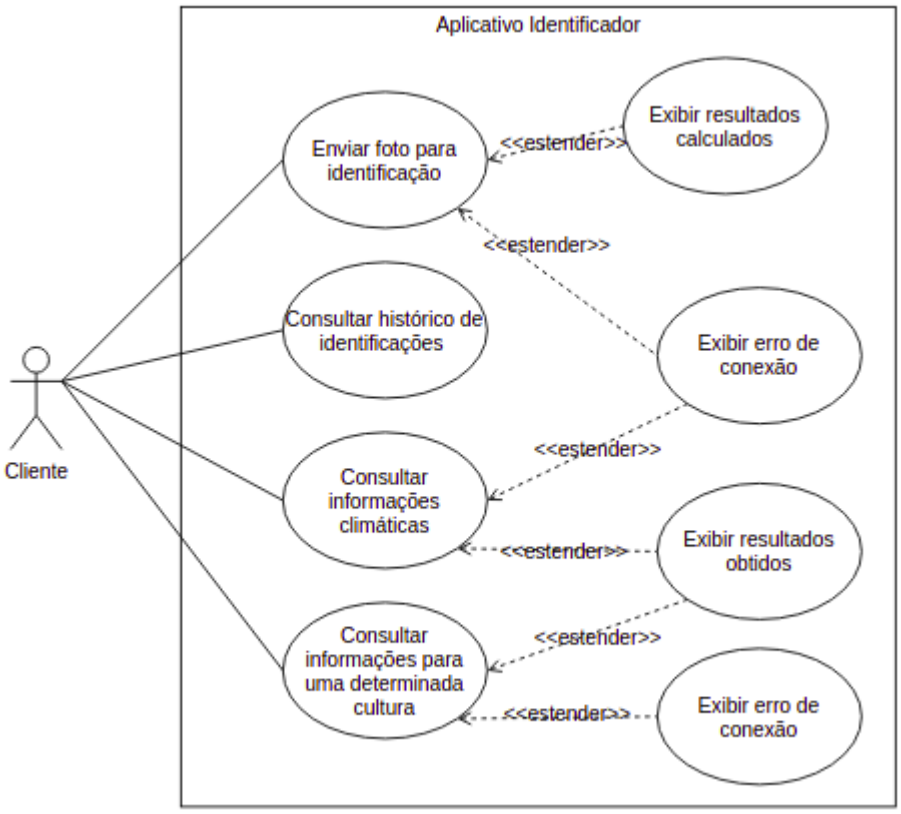
4.2 Planejamento

Para a melhor definição da solução, o planejamento do *software*, por meio de diagramas, esquemas e documentação, é essencial. Tendo isso em vista, foram construídos os diagramas de casos de uso e de classes para o sistema proposto. Estas especificações têm como objetivos principais servir de insumo para um projetista ou programador, para o projeto/codificação da funcionalidade, servir de documento para a validação da especificação, dimensionar o esforço necessário para a conclusão do projeto e descrever os cenários dos casos de teste.

4.2.1 Diagrama de casos de uso

A principal função de um diagrama de casos de uso é documentar o que o sistema faz do ponto de vista do usuário, identificar novos requisitos e esclarecer o sistema para um cliente. Assim, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os seus usuários.

Figura 11: Diagrama de casos de uso do sistema.

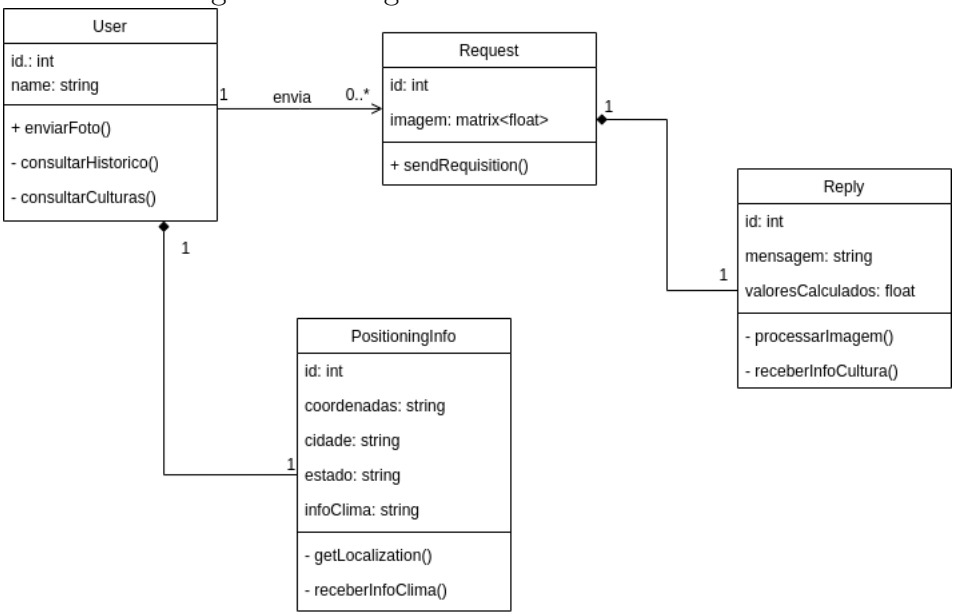


Fonte: Os autores.

4.2.2 Diagrama de classes

Diagramas de classes possuem como objetivo principal mapear de forma clara a estrutura de um determinado sistema, por meio da modelagem de suas classes, seus atributos, operações e relações entre objetos. É uma ferramente fundamental para a identificação de lacunas e melhorar a especificação de um sistema para um cliente ou time.

Figura 12: Diagrama de classes do sistema.



Fonte: Os autores.

5 PROJETO E IMPLEMENTAÇÃO

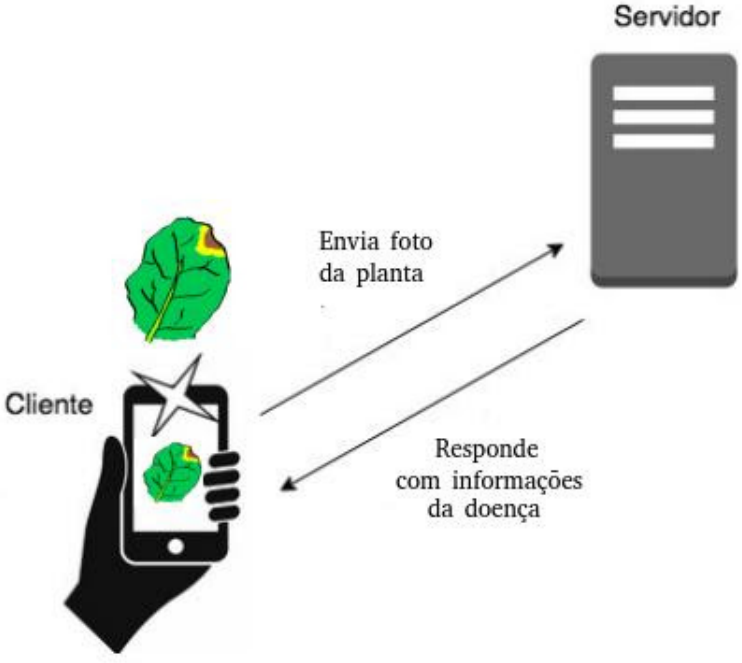
A partir da especificação e escopo do projeto levantados em seções anteriores, pode-se propor uma arquitetura e solução para o sistema proposto.

5.1 Concepção do projeto

Foi definido que o sistema pensado para o projeto deve consistir de dois módulos principais: uma aplicação *mobile* híbrida e um servidor.

Por meio da aplicação, o usuário consegue enviar uma fotografia da planta a ser analisada pelo servidor que, ao término do processamento da imagem pela rede neural, identifica a doença na planta e gera informações úteis ao usuário que serão exibidas como resultado final no aplicativo.

Figura 13: Esquema sobre a comunicação e funcionamento do sistema.



Fonte: Os autores.

5.2 Módulo da Aplicação

O módulo da aplicação consiste de aplicação *mobile* híbrida compatível com os sistemas operacionais *Android* e *iOS*. Dentre as telas previstas para a aplicação, estas podem ser divididas basicamente em 4 telas:

- Tela inicial, que possui uma breve apresentação do aplicativo e um botão para iniciar o fluxo;
- Tela histórico, que apresenta o histórico com as informações sobre as últimas requisições do usuário.
- Tela de clima, que trará algumas informações ao usuário quanto ao clima de sua região, como possibilidade de chuva, umidade do ar e temperatura.
- Tela de diagnóstico, que será acessada após a foto ser processada pelo servidor e trará as seguintes informações:
  - Doença identificada (com maior probabilidade);
  - Agente causador (fungo, inseto, deficiência de nutrientes);
  - Sintomas da doença;
  - Formas de tratamento;
  - Formas de prevenção;
  - Outras observações relevantes sobre a doença.

Figura 14: Design Tela de Diagnóstico.

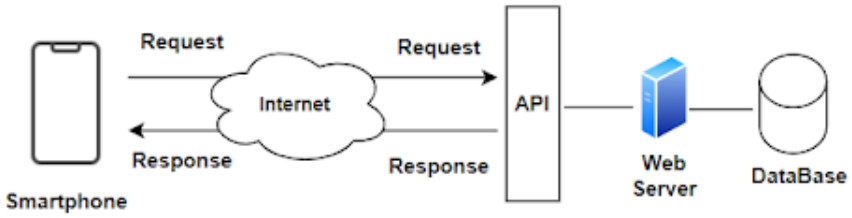
Fonte: Os autores.

Foi utilizado o modelo cliente servidor no projeto, com o protocolo TCP como meio de comunicação entre o aplicativo e o módulo de serviço. Assim, para a construção do módulo da aplicação foram utilizadas as ferramentas *React Native* e *Axios* para a conexão com a API.

5.3 Módulo do Servidor

O módulo servidor consiste de uma aplicação *Web* que utiliza uma rede neural artificial convolucional para classificar as imagens provenientes do aplicativo. As requisições para o servidor são feitas através de uma API *Restful*, que envia a imagem ao servidor, que processa a imagem e envia a resposta através de um arquivo *JSON* que conterá o resultado da classificação da doença e as outras informações relativas a esta (Figura 14).

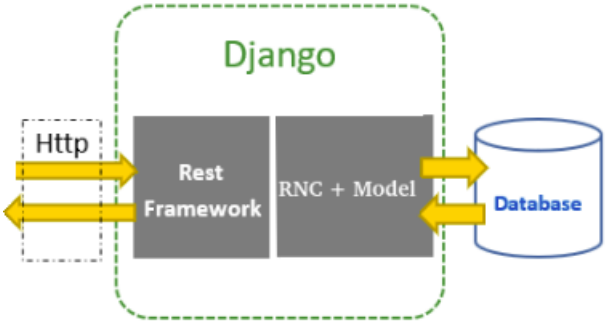
Figura 15: Esquema simplificado sobre a conectividade do sistema.



Fonte: Os autores.

Na construção do módulo servidor, que abriga a rede neural, foi utilizada a tecnologia *Django* associada as bibliotecas *TensorFlow* e *Keras*, usadas na rede neural convolucional. Estes, por sua vez, se comunicam com um banco de dados, o *SQLite*, proveniente do próprio *framework Django* (Figura 15).

Figura 16: Diagrama simplificado sobre o funcionamento do servidor.



Fonte: Os autores.

5.4 Processo de reconhecimento da Imagem

Para o atendimento dos requisitos do projeto (reconhecimento de padrões em imagens), um modelo de *Deep Learning*, que consiste em um algoritmo de aprendizagem de máquina utilizando redes neurais, foi usado para a classificação da imagem. Dentre os modelos de redes neurais existentes, o mais adequado e utilizado para a classificação de uma imagem é baseado em redes neurais convolucionais. Assim, para a criação de uma aplicação precisa, serão necessários muitos dados sobre as doenças em plantas.

5.4.1 Definição dos dados

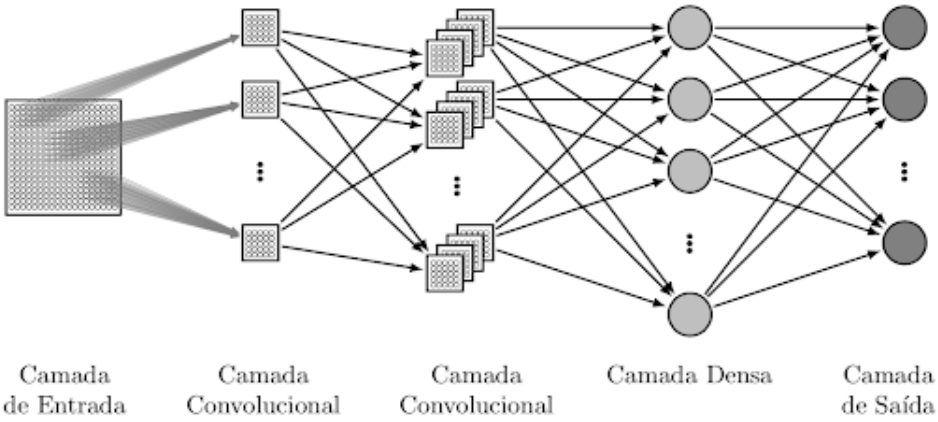
Tendo em vista a necessidade de precisão no reconhecimento de uma doença, foram necessários três conjuntos de dados, em grande quantidade: dados de treino, dados de validação e dados de teste. Os dados de treino são usados para treinar o algoritmo e então criar o modelo preditivo, os dados de validação para avaliar o modelo durante o treinamento e os dados de teste para validar a performance do modelo já treinado, ou seja, apresenta-se ao modelo dados que ele não viu durante o treinamento, a fim de garantir que ele é capaz de fazer previsões.

Para a obtenção dos dados, foi utilizado o banco virtual, disponibilizado pelo *TensorFlow*, denominado *plantVillage*, que possui 54 mil dados sobre plantas saudáveis e doentes, com a respectiva doença indicada nas informações da imagem. Assim, por ser de fácil acesso e com uma boa quantidade de amostras, este banco se torna o ideal para se realizar as primeiras análises.

5.4.2 Construção da rede neural convolucional

Como já mencionado em seções anteriores, uma rede neural convolucional é um algoritmo de *Deep Learning* que pode captar uma imagem de entrada, atribuir importância (pesos e vieses que podem ser aprendidos) a vários aspectos da imagem e ser capaz de diferenciar um do outro. O pré-processamento exigido em uma RNC é muito menor em comparação com outros algoritmos de classificação.

Figura 17: Esquema simplificado de uma RNC.

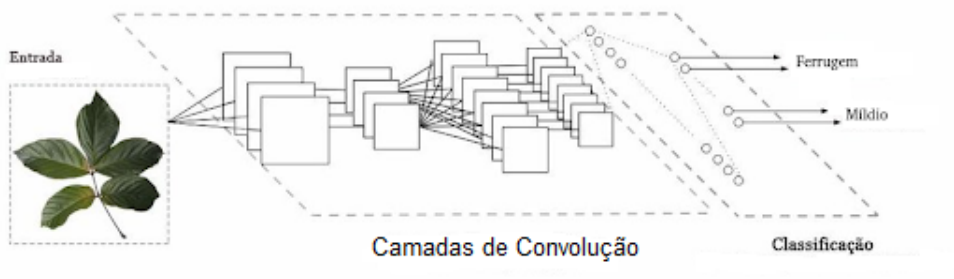


Fonte: extraído de Sakurai, 2017.

Nas camadas de convolução, a informação passa por vários filtros (que na prática são matrizes numéricas) com a função de acentuar padrões regulares locais, ao mesmo tempo

em que vão reduzindo a dimensão dos dados originais. Os resultados de vários filtros são sumarizados por operações de *pooling*. Na parte mais profunda das convoluções, espera-se que os dados num espaço dimensional reduzido contenham informação suficiente sobre esses padrões locais para atribuir um valor semântico ao dado original. Esses dados passam então por uma estrutura de FFN clássica para a tarefa de classificação.

Figura 18: Modelo de RNC e suas fases de processamento.



Fonte: extraído de Lima, 2017.

Por essas características, a aplicação mais comum das RNCs é na classificação de imagens; os filtros acentuam atributos dos objetos necessários à sua correta classificação.

5.5 Estudo do modelo

Para realizar a etapa de identificação das doenças nas plantas foram desenvolvidos e testados diversos modelos de redes neurais. Dois modelos se destacaram no percurso e apresentaram acurácia satisfatória para o escopo do projeto.

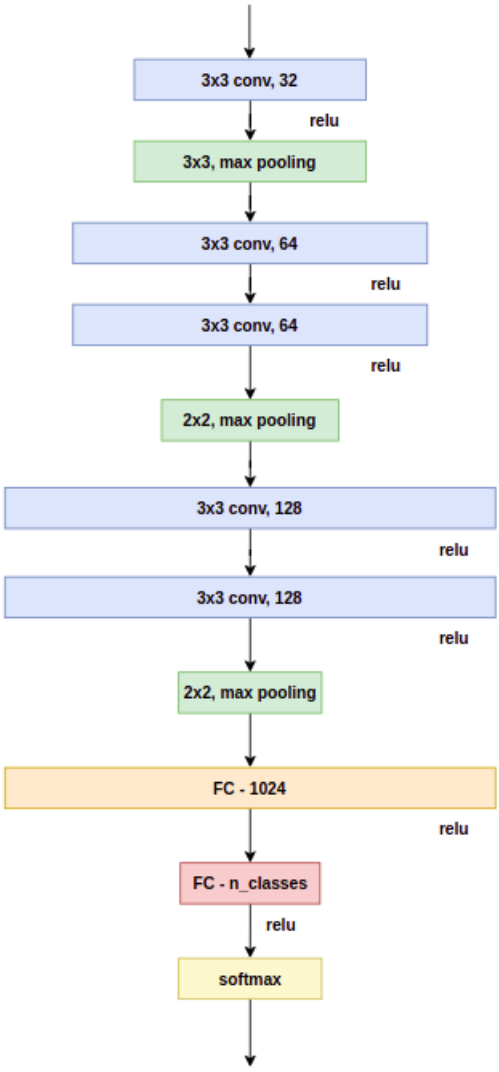
5.5.1 Modelo inicial de Rede Neural Convolucional

O primeiro modelo de Rede Neural desenvolvido foi um modelo composto por 8 camadas de convolução e 2 camadas *fully-connected* na saída, sendo a última camada com tamanho dado pelo número de classes a serem identificadas. Foram utilizadas também camadas de *Dropout* em algumas camadas do modelo.

A técnica de *Dropout* consiste em, durante o processo de treinamento da rede, atribuir uma probabilidade de alguns neurônios, ou determinadas áreas da rede neural, serem desativadas durante o processo de treinamento. Esta tem como principal objetivo evitar que determinadas partes da rede neural possam ficar muito sensíveis a pequenas alterações e também serve como forma de evitar *overfitting* do modelo (ILYA, 2014).

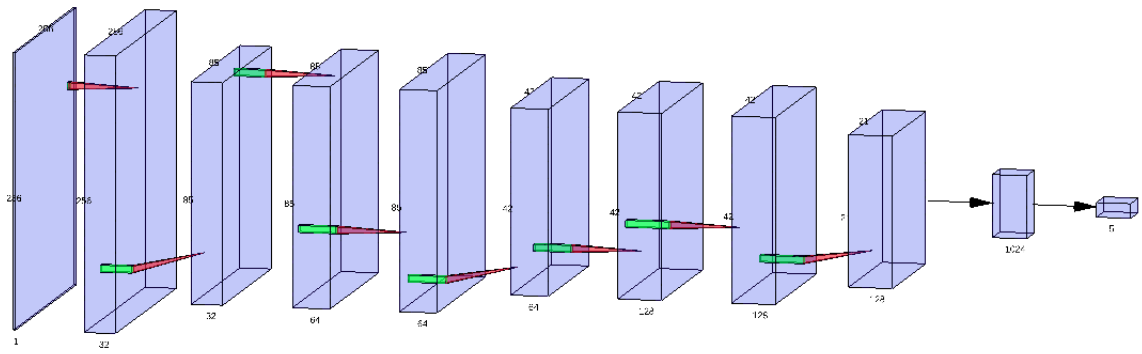
As funções de ativação escolhidas para a rede foram funções do tipo ReLU, com a exceção sendo a última camada que utiliza a função de ativação softmax (Função exponencial normalizada). Os diagramas das Figuras 18 e 19 representam o modelo desenvolvido nessa etapa.

Figura 19: Esquema simplificado da Rede Neural.



Fonte: Os autores.

Figura 20: Diagrama de blocos da Rede Neural desenvolvida.



Fonte: Os autores.

5.5.2 Rede neural com MobileNet

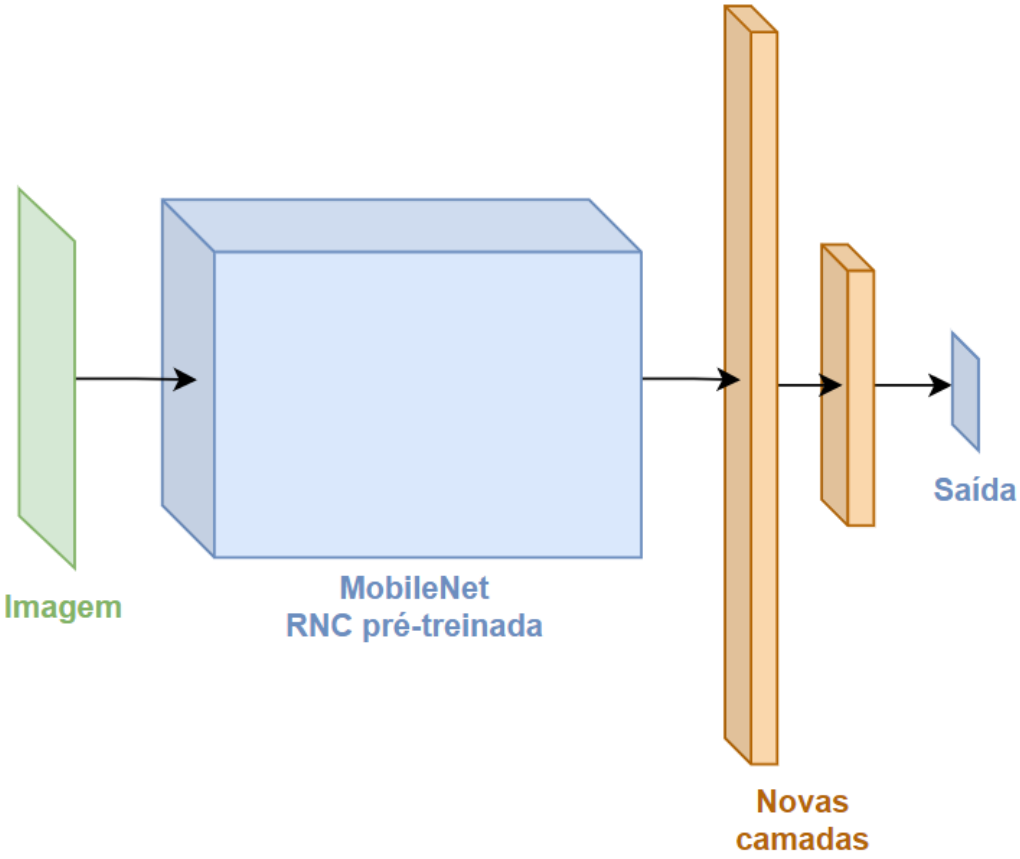
Para o segundo modelo, foi impregada a técnica de *Transfer Learning*. Para tanto, utilizou-se da arquitetura de rede neural *MobileNet* que, por suas características, é uma arquitetura mais leve e, em contrapartida, bastante eficiente, o que a torna perfeita para ser usada na aplicação deste projeto.

A arquitetura resultante do estudo consiste da rede *MobileNet* (com todos os seus pesos já treinados), que recebe a imagem da planta como entrada e, em sua saída, foram colocadas 2 camadas totalmente conectadas de 1024 e 256 neurônios, respectivamente. Por fim, após essas camada tem-se camada de saída, que é constituída por 6 neurônios correspondentes as classes propostas para serem identificadas (Figura 15).

A arquitetura de rede *MobileNet* utilizada no projeto foi treinada com o *dataset "ImageNet"*, um grande banco de dados visual projetado para uso em pesquisa de *software* de reconhecimento de objetos visuais e que, segundo Huh (2014), é um *dataset* que vem sendo bastante utilizado como pré-treino de redes neurais devido aos excelentes resultados obtidos.

Para este caso também foi utilizado o *dataset PlantVIllage* para obter as imagens e, assim, realizar o treino do modelo. As funções de ativação escolhidas para a rede também foram funções do tipo ReLU, com a exceção sendo a última camada que utiliza a função de ativação softmax (Função exponencial normalizada).

Figura 21: Diagrama esquemático da Rede neural com *MobileNet*.



Fonte: Os autores.

5.6 Treinamento do modelo

Com a arquitetura de IA definida, foi possível então treinar a rede, o que consiste em apresentar os dados ao algoritmo para que o aprendizado ocorra. Para tanto, como já mencionado em seções anteriores, utilizou-se um *dataset* proveniente do banco virtual denominado *plantVillage*. As 54 mil imagens presentes neste dataset estão divididas 14 espécies de plantas e em 38 classes de doenças de plantas. Para o treino da rede neural desenvolvida, foram utilizadas 3600 imagens de folhas de 9 classes de espécies de plantas, dentre elas maças, milho, batata e tomate. As doenças escolhidas no treino da rede foram ferrugem, mildio, podridao negra, mancha foliar e mela.

Por fim, o modelo treinado pode ser utilizado para fazer as classificações. Novas imagens serão entregues ao modelo e ele terá que classificar se a imagem representa uma das cinco doenças implementadas (ferrugem, mildio, podridao negra, mancha foliar e mela), com o mais alto nível de acurácia possível.

PARTE III

TESTES E RESULTADOS

6 RESULTADOS OBTIDOS

Nesta seção, serão mostrados todos os resultados encontrados para a aplicação *front-end* em *React Native*, o servidor em *Django*, a integração com o protocolo HTTP para o envio de imagens pelo *Axios* e a rede neural com *TensorFlow* e *Keras*.

6.1 Aplicação

Para a elaboração do planejamento e design para o aplicativo *mobile*, foram prototipadas as telas finais do MVP Figma. Assim, utilizando o *framework React Native* e a plataforma Expo, as telas foram concluídas, sendo acessadas de acordo com a seleção do usuário no menu.

A tela de clima obtém as informações sobre clima por meio da API aberta *Open Weather Map* que, com base nas coordenadas do usuário no mapa, envia uma resposta para a aplicação.

A tela de diagnóstico é mostrada após a recepção, pelo aplicativo, da confirmação do servidor (resposta HTTP), que indica que o processamento da foto tirada foi finalizado.

Figura 22: Telas completas da aplicação *mobile*.

Fonte: Os autores.

6.2 Testes de Módulo

Os módulos da aplicação foram testados através de ferramentas convenientes para cada uma. A descrição do procedimento usado para os testes está abaixo.

6.2.1 *Front-End*

Foram utilizadas duas abordagens para os testes no *front-end*: testes de *UX/UI* e testes automatizados de interface (fim-a-fim), utilizando a tecnologia *Appium*.

Para os testes de *UX/UI*, foram selecionados alguns possíveis usuários, que utilizaram o aplicativo por um determinado período, reportando bugs e sugerindo melhorias ao fim do período. Com isso, foram obtidos requisitos funcionais importantes para o aplicativo, como a remoção da tela de menu e implementação de uma barra inferior para a seleção da tela.

Com a finalidade de melhorar a robustes da aplicação, os testes de interface, utilizando *Appium*, foram elaborados. Realizaram-se simulações dos fluxos possíveis para as telas, com a seleção, pelo código no *Appium*, de cada elemento da interface, até a retirada da foto.

6.2.2 *Back-End*

O *Back-end* foi testado utilizando algumas ferramentas como o *Postman* e *Google Colab*. O *Postman* é um programa que permite fazer requisições HTTP, utilizadas nos testes para simular o comportamento do *Front-end* e testar a integração da rede neural treinada ao sistema.

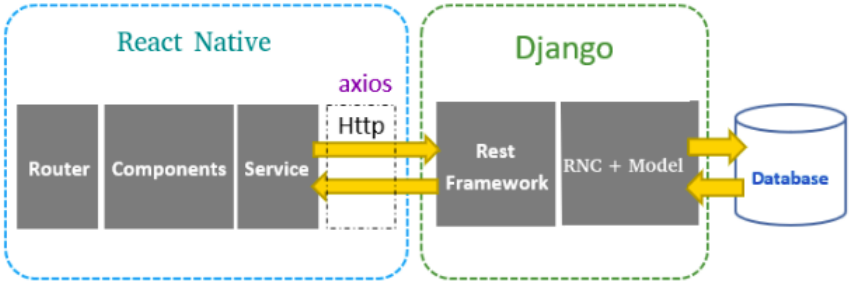
No *Google Colab* foi possível executar código *Python* de forma modular e, ao longo da execução, as variáveis permaneciam em memória. Assim, antes que o código fosse efetivamente implementado no *Back-end*, ele era testado na plataforma.

6.3 Servidor e integração

Para a recepção de imagens pelo servidor em *Django*, foram estabelecidos headers específicos nos *requests* HTTP enviados pelo *front-end*, em conjunto com códigos que realizam o *upload* da imagem, a ser processada pela rede neural convolucional, no servidor.

Assim, as imagens foram recebidas e processadas com sucesso, tornando a integração bem sucedida entre os módulos.

Figura 23: Diagrama sobre a integração do *front-end* com o *back-end*.



Fonte: Os autores.

6.4 Treino do modelo da RNC

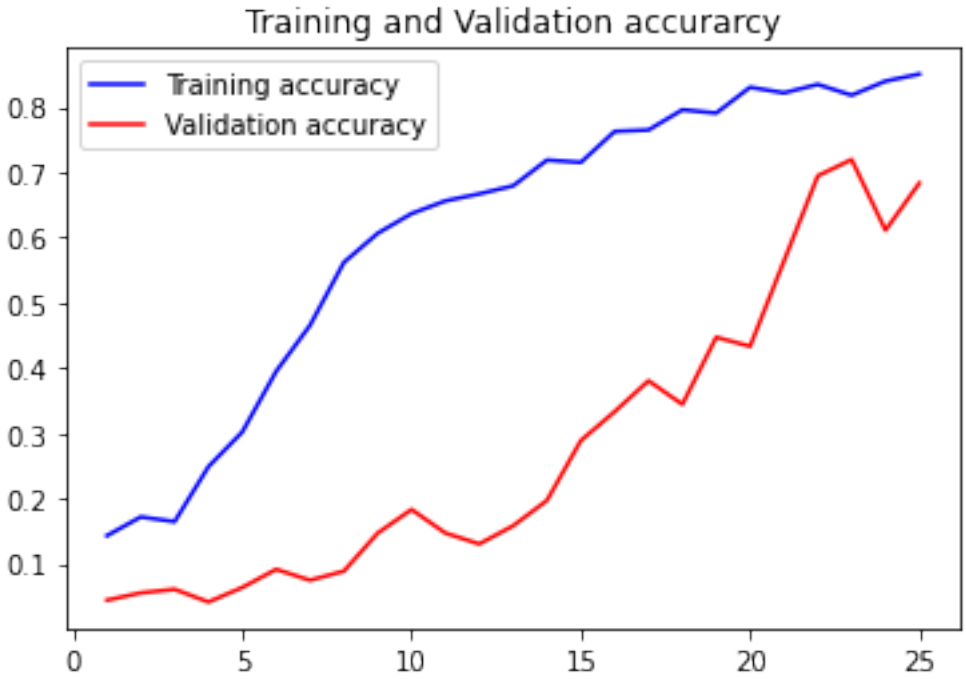
Para o treino das redes foi utilizado o *dataset plantVillage*. As doenças escolhidas no treino da rede foram ferrugem, mildio, mancha foliar, prodridão negra e mela, além também de identificar se as folhas da planta estão saudáveis.

6.4.1 Modelo inicial de Rede Neural Convolucional

Como resultado do treinamento da rede neural, temos a Figura 23 que mostra o gráfico da acurácia obtida com os dados de validação, que avaliam o modelo durante o treinamento, e obtida também com os dados de teste, que servem para validar a performance do modelo já treinado. Na Figura 24, temos duas curvas que representam as funções de custo utilizada no treinamento da rede neural. Analisando-se os dois gráficos, tem-se que conforme a acurácia do treino aumenta, a acurácia da validação também aumenta. O mesmo pode ser observado na função custo, conforme o custo durante o treinamento decresce, o custo da validação também decresce. É possível notar que houve grande *overfitting* entre as acurácias durante o treino.

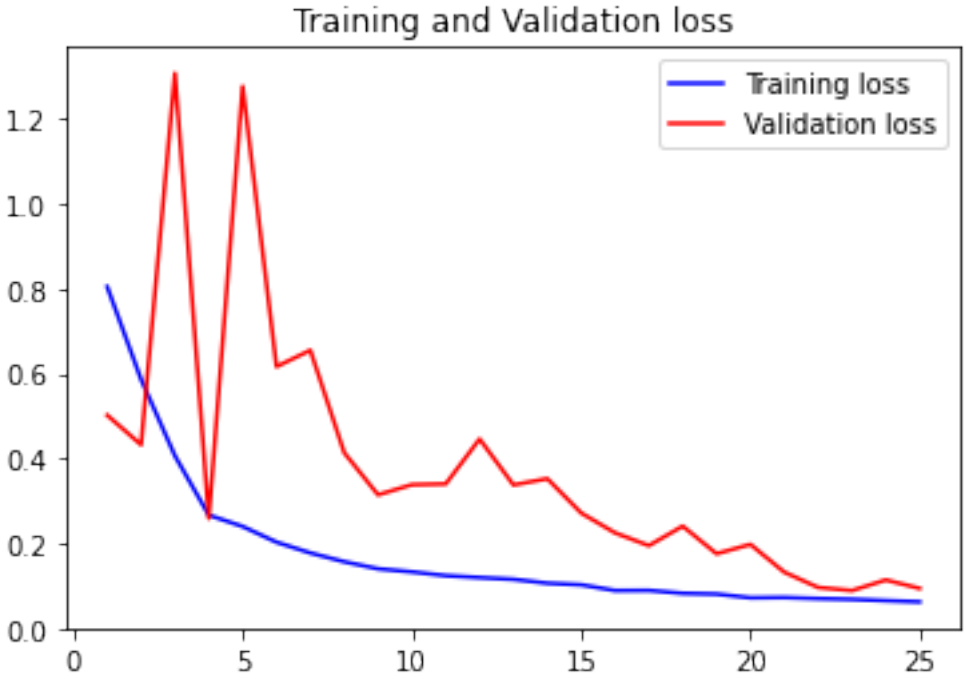
Como resultado final, temos que a acurácia obtida, ao final do treinamento, para a tarefa de identificar as 5 doenças de plantas, já citadas em seção anterior, é de 68,3% .

Figura 24: Acurácia obtida para o treino e validação do modelo da rede neural



Fonte: Os autores.

Figura 25: Resultado para a função de custo do modelo.



Fonte: Os autores.

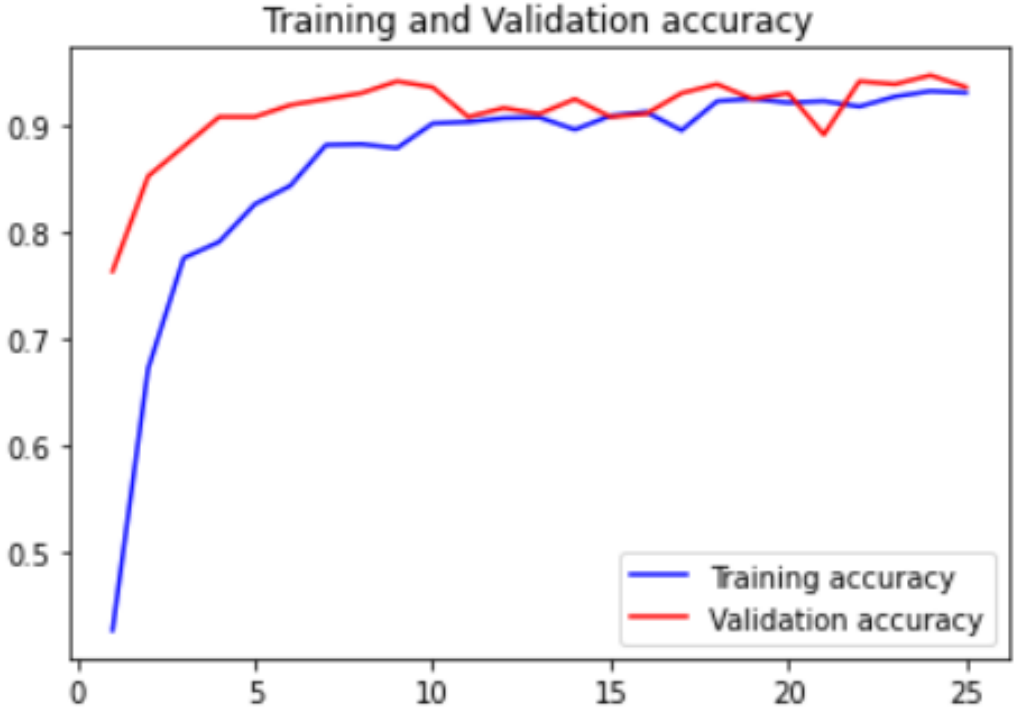
6.4.2 Rede neural com *MobileNet*

Para o treino da rede neural com *MobileNet*, temos como resultado do treinamento da rede neural, temos a Figura 25 que mostra o gráfico da acurácia obtida com os dados de

validação e obtida também com os dados de teste, que servem para validar a performance do modelo já treinado. Na Figura 26, temos as duas curvas que representam as funções de custo obtidas no treinamento. Analisando-se os dois gráficos, tem-se que conforme a acurácia do treino aumenta, a acurácia da validação também aumenta. O mesmo pode ser observado na função custo, conforme o custo durante o treinamento decresce, o custo da validação também decresce. É possível notar que houve redução drástica de *overfitting* durante o treino do modelo.

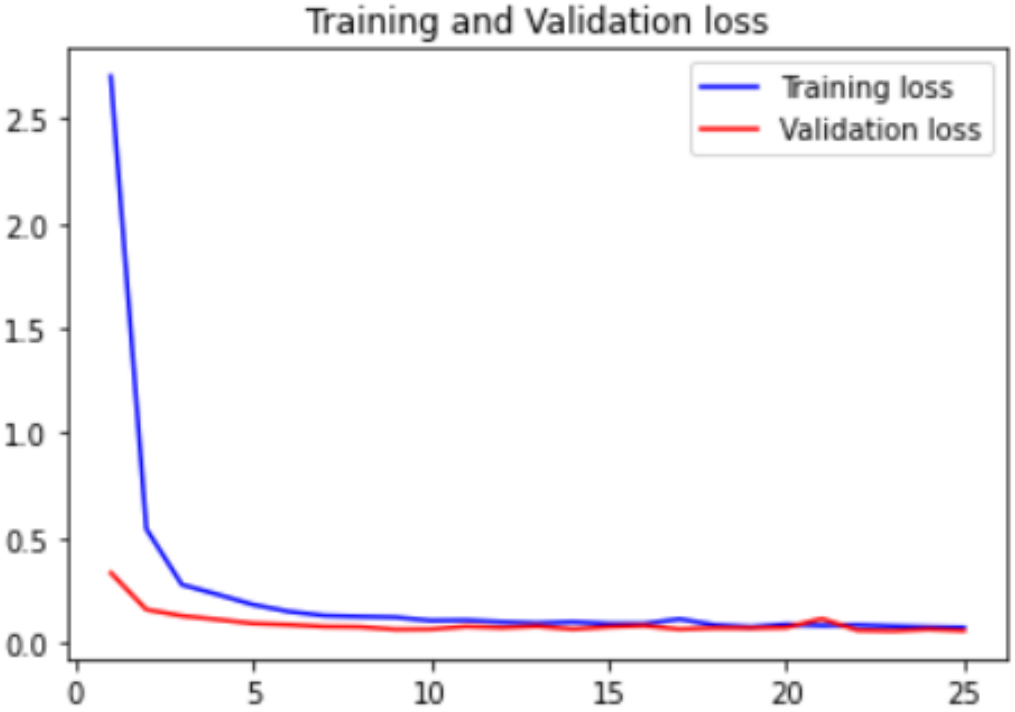
Como resultado final, temos que a acurácia obtida, ao final do treinamento deste modelo é de 93,4% .

Figura 26: Acurácia obtida para o treino e validação do modelo da rede neural



Fonte: Os autores.

Figura 27: Resultado para a função de custo do modelo.



Fonte: Os autores.

6.5 Testes de *software*

Testes de *software* possuem como objetivos principais avaliar a qualidade da aplicação e reduzir o risco de falha em operação do sistema. São essenciais para garantir um MVP adequado e de qualidade.

Para o presente projeto, serão utilizados testes automatizados, tanto para verificar a qualidade do *back-end* (testes unitários) quanto do *front-end* (testes *end-to-end*). As possíveis ferramentas são o Doctests, para testes unitários, e o *Cypress*, para os testes *end-to-end* do sistema.

7 CONCLUSÕES

Como objetivo do projeto, foi estabelecido a criação de uma aplicação *mobile* que resolvesse o seguinte problema: disponibilizar para um usuário, portador de um *smartphone*, um identificador de fitopatologias de fácil utilização, prático e amigável, responsável por fornecer informações sobre a respectiva doença, prevenção e tratamentos, por meio de uma foto.

Ao longo da realização do trabalho, a partir do estudos realizados, da prototipação no *Figma* e das ferramentas e tecnologias disponíveis para o uso, definiu-se um escopo para a aplicação do modelo estabelecido: um módulo de aplicação, responsável pela interface do cliente, e um módulo de servidor, que abriga a rede neural para a identificação da doença. Com o planejamento feito, foi possível a testagem do modelo de aplicação proposto e sua consequente avaliação, perante a sua capacidade de resolver o problema encontrado, definido como objetivo principal do trabalho.

As escolhas referentes a quais tecnologias deveria se utilizar foram vitais no desenvolvimento deste trabalho, com destaque para a definição da utilização de plataformas móveis hibridas, do uso das tecnologias *React Native*, *Expo* e *Django*, da utilização de redes neurais convolucionais para a identificação e do aproveitamento dos serviços de fornecimento de dados climáticos da *OpenWeatherAPI*. O resultado da implementação do modelo foi a aplicação *PlantID*.

7.1 Consideraçõs finais

A partir da análise do cumprimento dos requisitos e da conclusão dos testes, foi possível observar que o desempenho e precisão do sistema foram satisfatórios e acima do esperado. A aplicação está muito próxima de poder ser distribuída para todos os usuários das plataformas *Android* e *iOS*, acrescentando um leque de novas possibilidades para os criadores de plantas domésticas, agricultores e botânicos.

O objetivo do trabalho foi alcançado, sendo possível criar um modelo de aplicação que resolve o problema proposto, a identificação de fitopatologias de forma simples, por meio de uma foto, e que por sua generalização e flexibilidade não está preso a nenhuma plataforma *mobile* específica, nem a algum tipo de usuário ou localidade, permitindo sua implementação de outras formas além da que foi utilizada para o presente projeto.

7.2 Perspectivas de continuidade

Devido a versatilidade da aplicação, sua extensão e aprimoramento são de fácil implementação. A perspectiva da inserção de novas doenças, melhorias na velocidade de processamento da rede e adição de novas features torna a ideia de transformar o MVP em um produto comercial em algo real. Contudo, em um primeiro momento, o projeto está sendo continuado paulatinamente.

REFERÊNCIAS

Deep Learning Book, 2021. **Reconhecimento de imagens com redes neurais convolucionais em python**. Disponível em: https://www.deeplearningbook.com.br. Acesso em: 2 de junho de 2021.

RAVA, C. A.; SARTORATO, A. **Conceitos básicos sobre doenças de plantas**, Embrapa. Disponível em: https://www.embrapa.br. Acesso em: 2 de junho de 2021.

BERGAMIN FILHO, A.; KIMATI, H. História da Fitopatologia. In BERGAMIN FILHO, A.; KIMATI, H.; AMORIM, L. (eds.) **Manual de Fitopatologia: princípios e conceitos**. São Paulo: Editora Agronômica Ceres. 3. ed., v.1, cap.1, p. 2-12, 1995.

MOREIRA, Rosa. **Doenças nas plantas: fique a conhecer as principais, Actual Agricultura e Mar**. Disponível em: https://agriculturaemar.com. Acesso em: 1 de junho de 2021.

MOURA, R.M. **Fundamentos Históricos e Evolutivos da Nematologia de Interesse Agrícola: Uma Visão no Ano 2000**. Nematologia Brasileira 24:1-21. 2000.

COSTA, A. S. **História da Fitopatologia no Brasil**. Summa Phytopathologica, Piracicaba, v.1, p.155-163, 1975.

BERGAMIN FILHO, A.; KITAJIMA, E. W. História da Fitopatologia. In AMORIM, L.; REZENDE, J. A. M.; BERGAMIN FILHO, A. (eds.) Manual de Fitopatologia: princípios e conceitos. São Paulo: Editora Agronômica Ceres. 4. ed., v.1, cap.1, p. 3-17, 2011.

OCCHINO, T. (2015). **React native: Bringing modern web techniques to mobile**. Disponível em: https://engineering.fb.com. Acesso em 26 de julho de 2021.

EMBRAPA. **Doenças de plantas e sustentabilidade são discutidas em Congresso em Manaus**. Disponível em: https://www.embrapa.br. Acesso em 1 de dezembro de 2021.

OVERLEAF (2021). **Overleaf**. https://www.overleaf.com/. Acesso em: 11 dez. 2021.

HUH, MiINYOUNG; AGRAWAL, PULKIT; EFROS, A.ALEXEI. **What makes ImageNet good for transfer learning?** Disponível em: https://www.researchgate.net/, 2014. Acesso em 11 de dezembro de 2021.

SRIVASTAVA, NITISH; HINTON, GEOFFREY; KRIZHEVSKY, ALEX; SUTSKEVER, ILYA; SALAKHUTDINOV, RUSLAN. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. Disponível em: https://www.researchgate.net/, 2014. Acesso em 11 de dezembro de 2021.

ARBIB, M. A. **The Handbook Of Brain Theory and Neural Networks**. 2. ed. London, England: The MIT Press, 2002.

REIS, M. C. **A unidade básica do Sistema Nervoso: Neurônio**. Disponível em: https://www.unifal-mg.edu.br/. Acesso em 11 de dezembro de 2021.

MCCULLOCH, W. S.; PITTS, W. H. **A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.

BENGIO Yoshua; Goodfellow Ian; Courville Aaron. **Deep Learning (Adaptive Computation and Machine Learning series)**. Página 526, 2016.

SORIA Olivas, Emilio; David Martin Guerrero, Jose; Sober, Marcelino Martinez. **Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques**. Capítulo 11, 2009.

FILGUEIRA, F. A. R. **Novo manual de olericultura: agrotecnologia moderna na produção e comercialização de hortaliças**. Viçosa, MG: UFV. 2000. 402 p.

SCHUMANN, IS. **Controle da podridão negra do repolho com o uso de fertilizante foliar à base de cobre**. Dissertação de Mestrado. Instituto Federal Goiano. 2018. Morrinhos, Goiás, Brasil.

PYTHON, 2021. Disponível em https://www.python.org/. Acesso em 17 de dezembro de 2021.

NUMPY, 2021. Disponível em https://numpy.org/. Acesso em 17 de dezembro de 2021.

DJANGO, 2021. Disponível em https://www.djangoproject.com/. Acesso em 17 de dezembro de 2021.

PILLOW, 2021. Disponível em https://pypi.org/project/Pillow/. Acesso em 17 de dezembro de 2021.

OPENCV, 2021. Disponível em https://opencv.org/. Acesso em 18 de dezembro de 2021.

TENSORFLOW, 2021. Disponível em https://www.tensorflow.org/?hl=pt-br. Acesso em 11 de dezembro de 2021.

KERAS, 2021. Disponível em https://keras.io/. Acesso em 11 de dezembro de 2021.

NATALI, Lucas. **Tipos de aprendizado de máquina e algumas aplicações**. TerraLAB, 2021. Disponível em: http://www2.decom.ufop.br/. Acesso em 17 de dezembro de 2021.

SAMMY, V. Militante; BOBBY, D. Gerardo; NANETTE, V. Dionisio. **Plant Leaf Detection and Disease Recognition using Deep Learning**. IEEE, 2019. Disponível em: https://www.ieee.org/. Acesso em 17 de dezembro de 2021.