Trabalho de Conclusão de Curso

Um estudo sobre o TinyML a partir de uma aplicação mobile para detecção de doenças em folhas de manga

Itallo Patrick Castro Alves da Silva
ipcas@ic.ufal.br

Orientador:

Prof. Dr. Erick de Andrade Barboza

Maceió, Setembro de 2023

Itallo Patrick Castro Alves da Silva

Um estudo sobre o TinyML a partir de uma aplicação mobile para detecção de doenças em folhas de manga

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação pelo Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Erick de Andrade Barboza

Maceió, Setembro de 2023

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Girlaine da Silva Santos – CRB-4 – 1127

S586u Silva, Itallo Patrick Castro Alves da.
Um estudo sobre o TinyML a partir de uma aplicação mobile para detecção de doenças em folhas de manga / Itallo Patrick Castro Alves da Silva. – 2023.
36 f. : il. color.

Orientador: Erick de Andrade Barboza.
Monografia (Trabalho de Conclusão de Curso em Engenharia da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2023.

Bibliografia: f. 33- 36.

1. Inteligência artificial. 2. Aprendizagem de máquina. 3.Manga - doença. 4. Engenharia de software. I. Título.

CDU: 004.8 : 634.441

Agradecimentos

Primeiramente, gostaria de agradecer a Deus, que me capacitou, abriu mares para que eu pudesse atravessar e em nenhum momento deixou de me amar.

Agradeço também a minha mãe Lena Clese, que me deu todo o suporte durante toda a minha vida e nunca permitiu que eu deixasse os estudos de lado, por ser minha inspiração e a minha motivação diária para que eu possa ser um ser humano melhor, pelos incontáveis sacrifícios que ela fez para que eu e meus irmãos pudéssemos ter uma educação digna, sem ela, com toda certeza, eu não teria chegado até aqui. Aos meus irmãos, Matheus e Bruno, em quem sempre me espelhei, por me incentivarem desde cedo aos estudos e por sempre acreditarem em mim. A minha Tia Lucimar, que sempre foi uma segunda mãe para mim e para os meus irmãos, que sempre nos estendeu a mão sem pedir nada em troca. Ao restante da minha família, que sempre acreditou na minha capacidade de dar o meu melhor a cada dia.

Agradeço a minha namorada Bruna, que sempre esteve ao meu lado durante toda essa jornada e sempre me apoiou em todas as decisões que tomei durante este tempo de graduação.

Meus agradecimentos também aos meus amigos e colegas da graduação, que tornaram essa etapa da minha vida muito mais leve e agradável. Em especial ao: Lucas Buarque, Jeferson Fernando, João Victor e Sandoval Júnior, que foram verdadeiros companheiros em todas as dificuldades encontradas pelo caminho e também por serem um espelho para que eu pudesse seguir. Agradeço aos meus amigos que não fazem parte da graduação, como o grupo *Friends*, que sempre confiaram na minha capacidade e por todo apoio durante essa jornada.

Ao professor Erick, por ser uma grande inspiração, sendo um exemplo de profissional, no qual eu me espelho. Por ter aceitado de prontidão ser meu orientador, por ser sempre solícito e atencioso em qualquer demanda que o aluno possa ter, por isso meus sinceros agradecimentos. A minha banca, por ter aceitado o convite de constituí-la, muito obrigado.

Por último, gostaria de agradecer ao Instituto de Computação, na figura de seu corpo administrativo e docente, por ter me feito chegar até aqui. Além disso, a toda Universidade Federal de Alagoas, deixo o meu agradecimento.

"Foi por você
Que na cruz, meu sangue foi derramado
Por isso sinta-se querido e amado
Pois é o meu amor que cura a sua dor
Que cura a sua dor"

– Anjos de Resgate

Resumo

No contexto mundial, a manga é uma das frutas mais importantes e, no Brasil, ela também se estabelece com um papel de grande importância. Contudo, as doenças que acometem a mangueira podem prejudicar de forma bastante significativa a colheita de seus frutos. Nesse sentido, a aprendizagem de máquina pode se fazer presente para ajudar o agricultor a descobrir se a sua mangueira sofre de alguma enfermidade. A partir disso, tem-se o *TinyML* que é uma tecnologia em plena ascensão que permite trazer o poder da aprendizagem de máquina para hardwares com recursos limitados e traz diversas vantagens com a sua utilização, como o baixo custo, latência reduzida, eficiência energética e segurança. Com isso, este trabalho se propôs a trazer uma aplicação mobile para detecção de doenças em folhas de manga e, além disso, trazer o impacto do *TinyML* em comparação à utilização de modelos de aprendizagem de máquina tradicionais, normalmente, hospedados em algum servidor na nuvem. Notou-se, portanto, que os resultados obtidos neste trabalho corroboram com as vantagens da utilização dessa tecnologia e que em situações onde o acesso à internet é limitado essa aplicação pode ser de grande valia, porque não se faz necessária a utilização de internet.

Palavras-chave: Aprendizagem de Máquina; Inteligência Artificial; *TinyML*; Aprendizado Profundo; Agricultura Inteligente; Doenças em folhas de manga; Aplicativo Mobile.

Abstract

In the global context, mango is one of the most important fruits in the world and, in Brazil, it also plays a very important role. However, diseases that affect mango trees can significantly harm the harvest of its fruits. In this sense, machine learning can be done to help the farmer find out if his mango tree suffers from any disease. From this, we have TinyML, which is a technology on the rise that allows the power of machine learning to be brought to hardware with limited resources and brings several advantages with its use, such as low cost, reduced latency , energy efficiency and safety. With this, this work proposed to bring a mobile application for detecting diseases in manga leaves and, in addition, to bring the impact of TinyML in comparison to the use of traditional machine learning models, normally hosted on some server in the cloud. It was noted, therefore, that the results obtained in this work corroborate the advantages of using this technology and that in situations where access to the internet is limited, this application can be of great value, because the use of the internet is not necessary.

Keywords: Machine Learning; Artificial Intelligence; TinyML; Deep Learning; Smart Agriculture; Mango Leaf Diseases; Mobile Application.

Lista de Figuras

Lista de Tabelas

Conteúdo

1

Introdução

A manga (*Mangifera indica L.*) é a quinta fruta mais importante do mundo [10] e apresenta grande importância em regiões de clima tropical e subtropical [22]. O Brasil é um grande produtor e consumidor de mangas, fruta que está presente no país desde a colonização, resultante da introdução e de cruzamentos naturais a partir de material genético trazido da Ásia pelos portugueses [29]. Nesse contexto, o Brasil se posiciona como o 7° maior produtor de manga do mundo, com aproximadamente 1.8 milhão de toneladas produzidas [10]. Além disso, a produção de manga desempenha um papel muito importante no país, visto que tem uma participação média de 10% nas importações globais, além de ser um grande consumidor [29].

Como afirmado em [23], a tecnologia moderna aplicada na agricultura pode aumentar substancialmente a produtividade e a sustentabilidade. Nesse sentido, aplicações em *Machine Learning* (Aprendizagem de máquina) e visão computacional introduziram uma nova tendência em monitoramento e previsões [17], que contribuem para o aumento dos características citadas por [23].

A partir disso, nota-se a vantagem de incorporar a aprendizagem de máquina no contexto da agricultura. Desse modo, há um claro estudo sobre a utilização de modelos de aprendizagem de máquina para a detecção de doenças em folhas de plantações [16]. Segundo [5], estima-se que 40% das culturas alimentares são afetadas por pragas e doenças, com isso, uma detecção precoce destas pode aumentar consideravelmente a qualidade das culturas.

Entretanto, no Brasil, em áreas rurais o acesso à internet a partir de serviços de telefonia móvel ainda é limitado a cerca de 69.5% dos domicílios rurais [13]. Com isso, a utilização de modelos de aprendizagem de máquina da maneira tradicional pode não ser a mais indicada, visto que o acesso à internet ainda é limitado.

TinyML é um conceito emergente que visa adaptar modelos complexos de aprendizado de máquina para serem executados em dispositivos com restrição de desempenho e consumo de energia. A partir disso, o dispositivo pode extrair resultados inteligentes em tempo real sem a necessidade de interagir com entidades externas, como em um servidor, por exemplo.

Diante disso, o objetivo geral deste trabalho é realizar um estudo sobre o *TinyML* no contexto da agricultura brasileira a partir da construção de um aplicativo mobile para detecção de doenças em folhas de manga.

Ademais, os objetivos específicos deste trabalho são:

- Construir um aplicativo mobile para detecção de doenças em folhas de manga.
- Avaliar o modelo após a conversão para ser executado no dispositivo móvel em termos de qualidade da classificação e utilização de recursos.

A partir disso, espera-se que este trabalho possa contribuir com a agricultura brasileira no sentido de trazer alternativas para a escassez do acesso à internet, bem como apresentar uma aplicação prática de um problema que pode ser corriqueiro tanto no contexto das folhas de manga, como também em diversos outros problemas que possam envolver detecção de doenças, classificação de imagens e dispositivos móveis.

Por fim, os capítulos seguintes estão organizados da seguinte forma:

- No Capítulo 2, será apresentada a fundamentação teórica necessária para a construção deste trabalho.
- No Capítulo 3, serão apresentadas as metodologias utilizadas.
- No Capítulo 4, serão apresentados os resultados obtidos e as discussões referentes a este.
- No Capítulo 5, será apresentada a conclusão deste trabalho junto com uma lista de sugestões para trabalhos futuros.

2

Fundamentação Teórica

2.1 Doenças em folhas de manga

A mangueira (*Mangifera indica L.*) é uma das principais espécies frutíferas tropicais cultivadas em todo o mundo [3]. Contudo, as doenças podem afetar todos os órgãos desta importante árvore. Essas doenças podem: danificar as mudas e as plantas enxertadas no viveiro, diminuir a fixação e a retenção dos frutos e se tornarem importantes problemas pré e pós-colheita dos frutos. Em muitas regiões, são o problema mais crítico à produção dos frutos. Além disso, a maioria dessas doenças é causada por fungos, contudo algas e bactéria também podem ser patógenos importantes [21].

Neste trabalho, algumas dessas doenças serão exploradas para a construção do nosso modelo, são elas: Anthracnose, Bacterial Canker, Cutting Weevil, Die Back, Gall Midge, Powdery Mildew e Sooty Mould. Essas doenças apresentam características peculiares e serão brevemente descritas nesta seção. Ademais, pode-se observar nas figuras 2.1(a) - 2.1(h), retiradas da base de dados, exemplos de cada uma das doenças e de uma folha saudável. Segundo, Ahmed et al., tem-se:

1. **Anthracnose**: Apresenta manchas necróticas em ambos os lados da folha da mangueira. Na maioria das vezes, essas manchas se formam ao longo das margens das folhas, onde as lesões se fundem; Veja a Figura 2.1(a). Ademais, as folhas severamente afetadas começam a enrolar. Normalmente, a infecção ocorre em folhas jovens. Contudo, isso não impede que essa doença afete as folhas em diferentes idades.

2. **Bacterial Canker**: Causado pela bactéria *Bacterium pseudomonas mangifera*, faz com que frutos, folhas, caules e galhos da manga fiquem com manchas encharcadas de água que se transformam em cancros; Veja a Figura 2.1(b).

3. **Cutting Weevil**: Essa doença corta a folha da mangueira, de modo que aparenta ter sido cortada por uma tesoura; Veja a Figura 2.1(c).

4. **Dieback**: Essa doença faz com que os galhos da mangueira sequem e quebrem de cima para baixo. Além disso, as folhas ficam na coloração marrom e seca; Veja a Figura 2.1(d).

5. **Gall Midge**: Essa doença faz com que as folhas tenham o que parecem ser espinhos. Com isso, surtos dessa doença resultam em redução da produção de frutos; Veja a Figura 2.1(e).

6. **Powdery Mildew**: O surgimento de fungos brancos na superfície das folhas, hastes de flores, flores e frutos jovens são as características dessa doença; Veja a Figura 2.1(f).

7. **Sooty Mould**: Essa doença é caracterizada por um mofo que se espalha lentamente sobre a superfície da parte afetada da planta, tornando-a preta; Veja a Figura 2.1(g).

(a) Anthracnose (b) Bacterial Canker (c) Cutting Weevil

(d) Die Back (e) Gall Midge (f) Powdery Mildew

(g) Sooty Mould (h) Healthy

Figura 2.1: Exemplos das doenças que serão estudadas e uma folha em estado saudável. (Fonte: Ahmed et al.)

2.2 Aprendizagem de máquina

Segundo Mohri et al., *Machine Learning* (ou Aprendizagem de Máquina) pode ser definido como métodos computacionais que usam a experiência para melhorar o desempenho ou fazer previsões de maneira precisa. Neste caso, a experiência se refere às informações disponíveis para a máquina, na forma de dados que foram coletados e disponibilizados previamente.

Além disso, Goodfellow et al. acreditam que um algoritmo de Machine Learning é um algoritmo capaz de aprender a partir de dados.

Mohri et al. apresenta alguma tarefas que foram extensivamente estudadas e que podem ser resolvidas com o uso de aprendizagem de máquina, como:

- Classificação: Este é o problema de atribuir uma categoria a cada item. Por exemplo, classificar qual o tipo de doença de uma planta presente em uma imagem.
- Regressão: Este é o problema que acaba por prever o valor real para cada item. Por exemplo, prever qual o valor de uma fruta com base em diversos fatores, como safra e demanda.
- Ranqueamento: Este é o problema de aprender a ordenar itens de acordo com algum critério. Por exemplo, ordenar páginas da web relevantes a partir de uma consulta.
- Agrupamento: Este é o problema de particionar um conjunto de itens em subconjuntos homogêneos. Por exemplo, no contexto de redes sociais, encontrar comunidades de usuários com interesse comum.
- Redução de dimensionalidade: Este é o problema de transformar uma representação inicial de itens em uma representação de menor dimensão, preservando algumas propriedades da representação inicial. Por exemplo, o pré-processamento de imagens digitais em visão computacional.

Além disso, as abordagens em Aprendizado de Máquina podem ser classificadas, normalmente, em três tipos: Aprendizado Supervisionado, Aprendizado Não-Supervisionado e Aprendizado por Reforço, como pode ser visto na Figura 2.2. As definições, segundo [18] são:

- Aprendizado supervisionado: O modelo recebe um conjunto de exemplos rotulados como dados de treinamento e faz previsões para todos os pontos não vistos. Exemplos: Árvores de decisão, Regressão Linear, SVM e Redes Neurais.
- Aprendizado não-supervisionado: O modelo recebe um conjunto de exemplos não rotulados como dados de treinamento e faz previsões para todos os pontos não vistos. *Clustering* e redução de dimensionalidade são exemplos de problemas de aprendizagem não-supervisionada.
- Aprendizado por reforço: As fases de treinamento e de teste são misturadas. Para coletar informação, o modelo interage de forma ativa com o ambiente e em alguns casos afeta o ambiente, e recebe uma recompensa imediata a cada ação. O objetivo do modelo é maximizar a recompensa durante um curso de ações e interações com o ambiente. Exemplos: Q-Learning e o Monte Carlo.

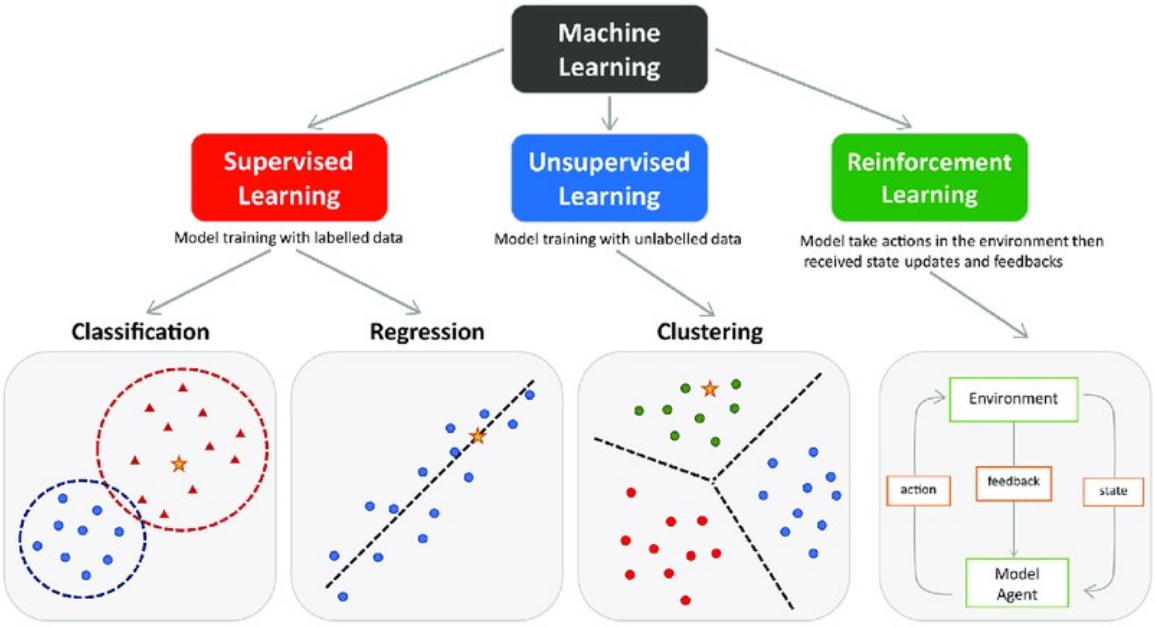

Figura 2.2: Tipos de aprendizagem. (Fonte: Peng et al.)

2.2.1 *Deep Learning*(Aprendizado Profundo)

Segundo Chollet, *Deep learning* é um subcampo específico da Aprendizagem de máquina, como pode-se ver na Figura 2.3, surge como uma nova abordagem no aprendizado de representações que enfatiza o aprendizado de camadas sucessivas de representações cada vez mais significativas. Ainda segundo o autor, essas representações são "quase sempre" aprendidas por meio de modelos chamados redes neurais.

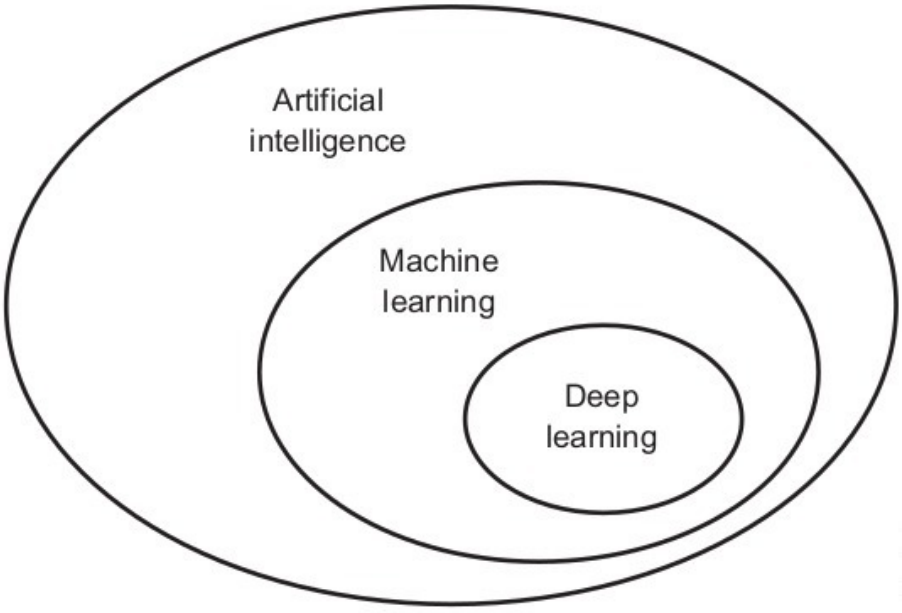

Figura 2.3: Inteligência artificial, *Machine Learning* e *Deep Learning*. (Fonte: Chollet)

Ainda convém mencionar que o autor apresenta um exemplo de como são essas representações aprendidas por um algoritmo de aprendizado profundo. A ideia é analisar como um algoritmo de aprendizado profundo transforma a imagem de um dígito com o intuito de reconhecer qual o dígito que se trata. Pela Figura 2.4, pode-se observar a arquitetura utilizada para a classificação do dígito. Ademais, a Figura 2.5 mostra que a rede transforma a imagem em representações cada vez mais diferentes da imagem original, pode-se pensar como uma operação de destilação de informação em vários estágios, onde a informação passa por filtros sucessivos e a saída é cada vez mais purificada com o intuito de ser útil para a resolução de alguma tarefa. Portanto, o aprendizado profundo é uma maneira de aprender representações de dados em vários estágios.

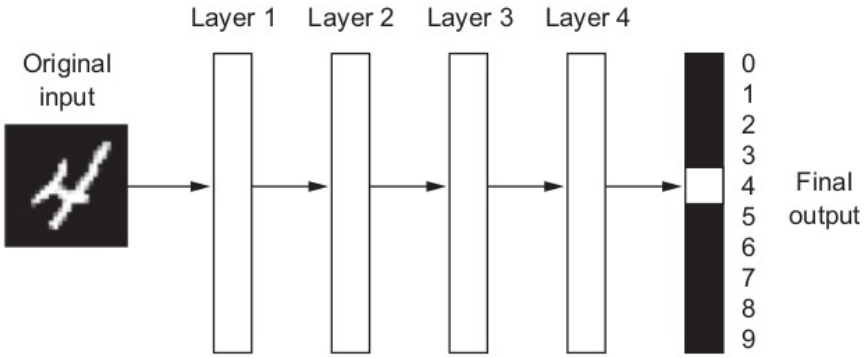

Figura 2.4: Rede neural profunda para classificar dígito. (Fonte: Chollet)

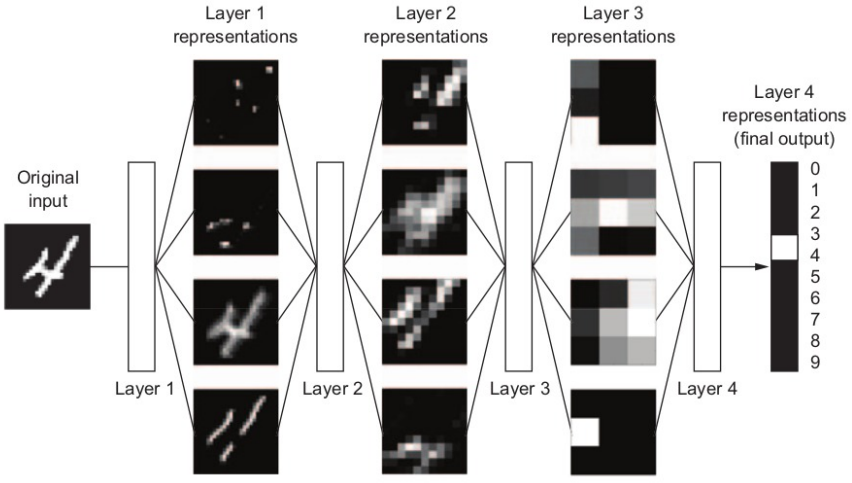

Figura 2.5: Representação aprendida pela rede neural profunda. (Fonte: Chollet)

2.2.2 Prevenção do *overfitting*

Segundo Shorten and Khoshgoftaar, modelos que apresentam baixa generalização acabam por ter superajustado (*overfitted*) os dados de treinamento e isso vai de encontro com o

intuito geral de um bom modelo, que seria o de apresentar uma capacidade de generalização alta. Com isso, para construir modelos úteis de *Deep Learning*, o erro de validação deve continuar a diminuir junto com o erro dos dados de treinamento.

A partir disso, surge a ideia de soluções funcionais para a redução de *overfitting* em modelos de *Deep Learning* que possuem conjuntos de dados menores. Alguma dessas soluções são: *dropout*, *batch normalization*, *data augmentation* e o aprendizado por transferência.

Dropout

Segundo Shorten and Khoshgoftaar, *Dropout* é uma técnica de regularização que zera os valores de ativação de neurônios escolhidos aleatoriamente durante o treinamento. Essa restrição força a rede a aprender recursos mais robustos, ao invés de depender da capacidade preditiva de um pequeno subconjunto de neurônios da rede.

Data augmentation

Conforme descrito por Shorten and Khoshgoftaar, o Data augmentation abrange um conjunto de técnicas que aprimoram o tamanho e a qualidade dos conjuntos de dados de treinamento, de modo que modelos de Deep Learning melhores possam ser construídos usando essa técnica.

Batch normalization

O *Batch normalization* normaliza o conjunto de ativações em uma camada. A normalização funciona subtraindo a média do lote de cada ativação e dividindo pelo desvio padrão do lote.

Aprendizado por transferência

Para definição do tema, Pan and Yang apresenta um exemplo bem lúdico. Imagine duas pessoas que desejam aprender a tocar piano. Uma pessoa não tem experiência anterior em tocar música e a outra pessoa tem amplo conhecimento por meio do violão. A pessoa com o conhecimento prévio na música poderá aprender a tocar piano de forma mais eficiente, ao transferir os conhecimentos previamente adquiridos. A partir disso, podemos notar semelhanças com o problema em questão, visto que um modelo previamente treinado com milhares de outras imagens pode trazer o seu conhecimento prévio para a resolução do problema de detecção de doenças em folha de manga. Uma ideia mais formal, segundo Weiss et al., seria que o aprendizado por transferência é utilizado para melhorar um modelo de destino transferindo informações a partir de um modelo cujo domínio esteja relacionado ao domínio do modelo de destino.

2.3 *TinyML*

O *TinyML*, ou o *Tiny Machine Learning*, é uma tecnologia emergente que está ganhando grande impulso em vários campos devido à sua capacidade de permitir inteligência não-intrusiva. O *TinyML* não é novo, visto que produtos eletrônicos de consumo como alto-falantes inteligentes e *smartwatches* já fazem uso de sua tecnologia. Contudo, os recentes avanços em hardware e software tornaram-no mais acessível e prático do que nunca [14]. Essa tecnologia é capaz de reduzir o tamanho de modelos de *Machine Learning* e de *Deep Learning* e trazer o poder da aprendizagem de máquina para hardwares com recursos limitados. Um sistema que implementa o *TinyML* apresenta diversas vantagens, como: baixo custo, latência reduzida, eficiência energética e privacidade e segurança [27].

- Baixo custo: o barateamento do hardware permite que o *TinyML* seja utilizado em diversas situações e dispositivos.
- Eficiência energética: como a execução dos modelos de aprendizado de máquina ocorre diretamente no dispositivo, então se elimina a necessidade de transferir dados para a nuvem. Com isso, economiza-se energia e prolonga a vida útil da bateria dos dispositivos.
- Latência reduzida: ao executar o modelo de maneira local, não há a latência relacionada com a comunicação com servidores remotos. Com isso, reduz-se a latência.
- Privacidade e segurança: como não há necessidade de comunicação com servidores remotos, então não se precisa compartilhar dados, muitas vezes, confidenciais com esses serviços na nuvem, o que, por sua vez, aumenta tanto a privacidade quanto a segurança desses dados.

Para implementação do *TinyML* faz-se necessária a utilização de técnicas para a diminuição do tamanho dos modelos originais, para que caibam no dispositivo de destino. Alguma dessas técnicas são: destilação de conhecimento (*knowledge distillation*), poda (*prunning*) e quantização [9].

- Destilação de conhecimento: fornece uma estrutura unificada para transferir o conhecimento adquirido por um grande modelo pré-treinado (professor) para outro modelo de tamanho muito pequeno (aluno).
- Poda: processo iterativo de remoção sistemática de parâmetros de uma rede neural.
- Quantização: refere-se à redução da precisão numérica de uma rede. Normalmente, os pesos e ativações são representados em pontos flutuantes (32/64 bits). Ao representar esses valores em ponto flutuante como inteiros, há um modelo quantizado. Contudo, essa técnica pode diminuir bastante a precisão de um modelo.

3

Metodologia

Neste capítulo serão apresentados, de forma detalhada, os recursos e estudos realizados para a construção da aplicação para a detecção de doenças em folhas de manga.

3.1 Estrutura geral do projeto

A construção da aplicação depende de um conjunto de etapas, como apresentado na Figura 3.1 de forma simplificada. Ao longo deste capítulo serão apresentados mais detalhes sobre cada uma dessas etapas.

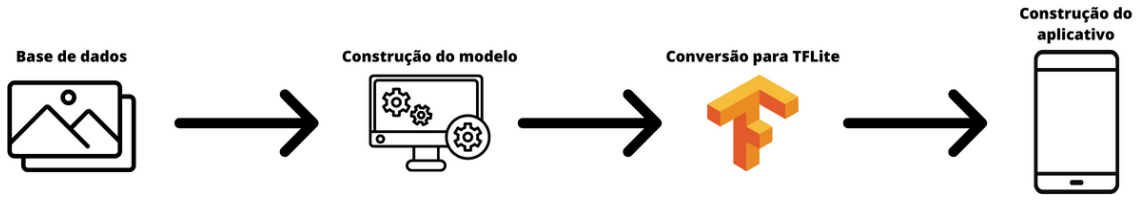

Figura 3.1: Conjunto de etapas para a construção do aplicativo (Fonte: Elaborada pelo autor).

Na primeira etapa, Base de dados, foi realizada a obtenção da base de dados de folhas de manga. Após isso, o modelo, baseado em redes neurais, foi construído utilizando diversas técnicas, como o aprendizado por transferência. Na etapa seguinte, o modelo gerado foi convertido para o formato TFLite, que permitirá o *deploy* deste no aplicativo mobile. Por fim, o aplicativo foi construído para que o usuário possa classificar qual doença uma determinada folha de mangueira possui a partir de uma foto tirada na hora ou até mesmo uma foto em sua galeria do seu celular. Além disso, para este trabalho foi inserido uma funcionalidade de monitoramento, para avaliarmos as vantagens de um modelo no conceito do *TinyML* em comparação a modelos tradicionais armazenados na nuvem.

3.2 Base de dados

A base de dados utilizada neste trabalho foi publicada em Ahmed et al.. Esta base foi construída em quatro pomares situados em Bangladesh, um dos maiores produtores de manga do mundo. Apesar da base não ter sido construída no Brasil, nota-se a existência dessas doenças em nosso território nacional [3]. Ainda convém mencionar que a base, inicialmente, contava com 1800 imagens de folhas de manga em diversos estados que serão citados a seguir. Contudo, Ahmed et al. aplicou técnicas de data augmentation, como: zoom e rotação. Ademais, as imagens foram pré-processadas utilizando: limpeza de ruído e rotulagem das fotos por humanos. Com isso, a base final contém 4000 imagens, que apresentam oito estados, sete deles são doenças foliares da mangueira e um que indica o estado saudável da folha, cada estado apresenta 500 imagens. Os estados são: Anthracnose, Bacterial Canker, Cutting Weevil, Die Back, Gall Midge, Healthy, Powdery Mildew, Sooty Mould. Por fim, o conjunto de dados está liberado para uso público e está disponível para download [1].

3.3 Construção do modelo

Nesta seção, serão descritos todos os passos para a construção do modelo, desde a definição de sua arquitetura até o resultado final com as métricas obtidas. Vale salientar que todo o código utilizado para geração do modelo e seus resultados está disponível no repositório remoto deste trabalho [2].

3.3.1 Tecnologias utilizadas

Foi utilizada a linguagem de programação Python versão 3.10[3], porque é uma linguagem de script de fácil manuseio e baixa curva de aprendizagem. Além disso, essa linguagem é altamente utilizada na área de machine learning e ciência de dados. Ademais, possui diversas bibliotecas que facilitam na resolução dos problemas das áreas citadas acima. Dentre as principais bibliotecas e frameworks utilizados neste trabalho, estão: Tensorflow[4], Keras[5], NumPy[6], Pandas[7], Matplotlib[8] e Sklearn[9].

[1] https://data.mendeley.com/datasets/hxsnvwty3r/1
[2] https://github.com/itallocastro/Model-and-Results-TCC.git
[3] https://www.python.org/downloads/release/python-31013/
[4] https://github.com/tensorflow/tensorflow/releases/tag/v2.13.0
[5] https://github.com/keras-team/keras/releases/tag/v2.13.1
[6] https://github.com/numpy/numpy/releases/tag/v1.24.3
[7] https://github.com/pandas-dev/pandas/releases/tag/v2.0.3
[8] https://github.com/matplotlib/matplotlib/releases/tag/v3.7.2
[9] https://github.com/scikit-learn/scikit-learn/releases/tag/1.3.0

3.3.2 Arquitetura do modelo

Nesta seção, será descrito como a arquitetura do modelo foi definida, cujo resultado final encontra-se na Figura 3.2.

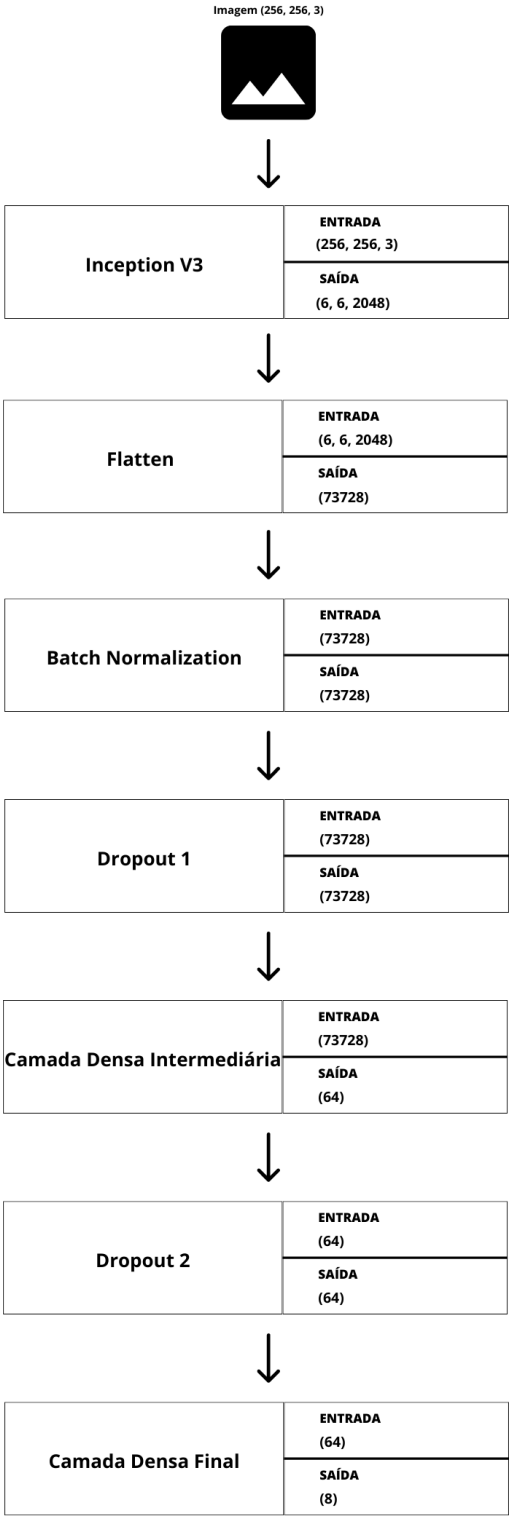

Figura 3.2: Arquitetura final do modelo (Fonte: Elaborada pelo autor (2023)).

Aprendizado por transferência

Para a construção do modelo de redes neurais para detecção de doenças em folhas de manga, foi utilizada uma técnica conhecida como aprendizado por transferência. Com isso, foi utilizado um modelo de rede neural convolucional pré-treinado, cuja arquitetura é o Inception-V3 (Szegedy et al.). Para o problema em questão, utilizamos essa arquitetura pré-treinada a partir da base de dados do ImageNet (Deng et al.). Podemos observar a arquitetura do Inception-V3 na Figura 3.3. Ainda convém mencionar que o Inception-V3 apresentou 78,1%, segundo Cloud, de precisão no conjunto de dados do ImageNet.

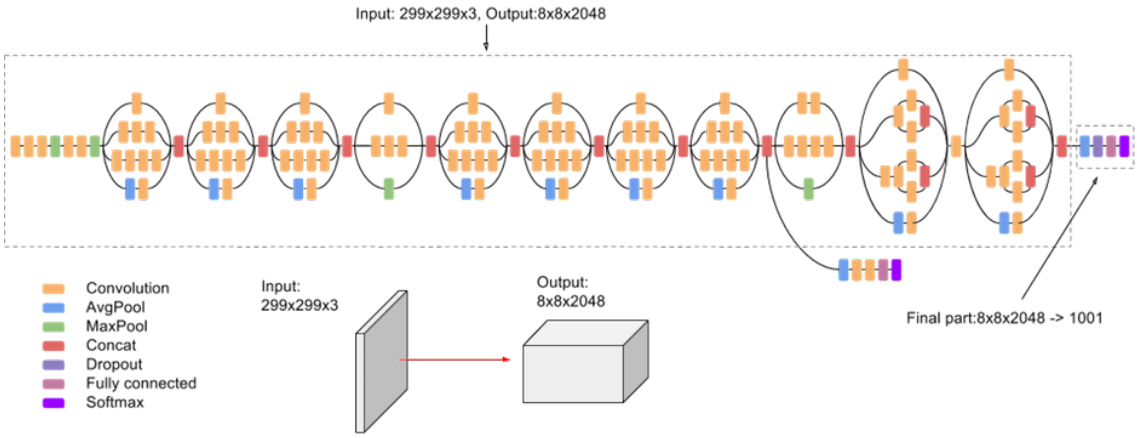

Figura 3.3: Arquitetura do modelo Inception-V3 (Fonte: Cloud).

Restante do modelo

Conforme visto acima, a primeira parte do nosso modelo advém do aprendizado por transferência. Depois disso, foi inserido um *Flatten* que permite a transformação da saída da etapa de Inception V3, como mostrado na Figura 3.2 essa saída é no formato (6, 6, 2048), em um vetor único de tamanho 73728. Além disso, foi adicionada uma camada de Batch Normalization, com isso, evita-se o retardo no treinamento de um modelo, causado pela covariável interna, conforme descrito por Ioffe and Szegedy.

Ademais, foram adicionadas camadas de Dropout intercaladas para se tentar evitar um overfitting em nosso modelo Srivastava. Por fim, existem duas camadas densas em nossa arquitetura, a última apresenta como saída um vetor de tamanho 8, onde cada posição do vetor está relacionada com um estado possível da nossa folha de mangueira.

3.3.3 Treinamento do modelo

Para o treinamento do modelo, os dados contendo as 4000 imagens de folhas de manga, foram divididos em dados para treinamento, validação e teste. Com isso, foram utilizadas 3200

imagens para treinamento (80%), 400 imagens para validação (10%) e 400 imagens para teste (10%). Os dados foram distribuídos conforme a Figura 3.4.

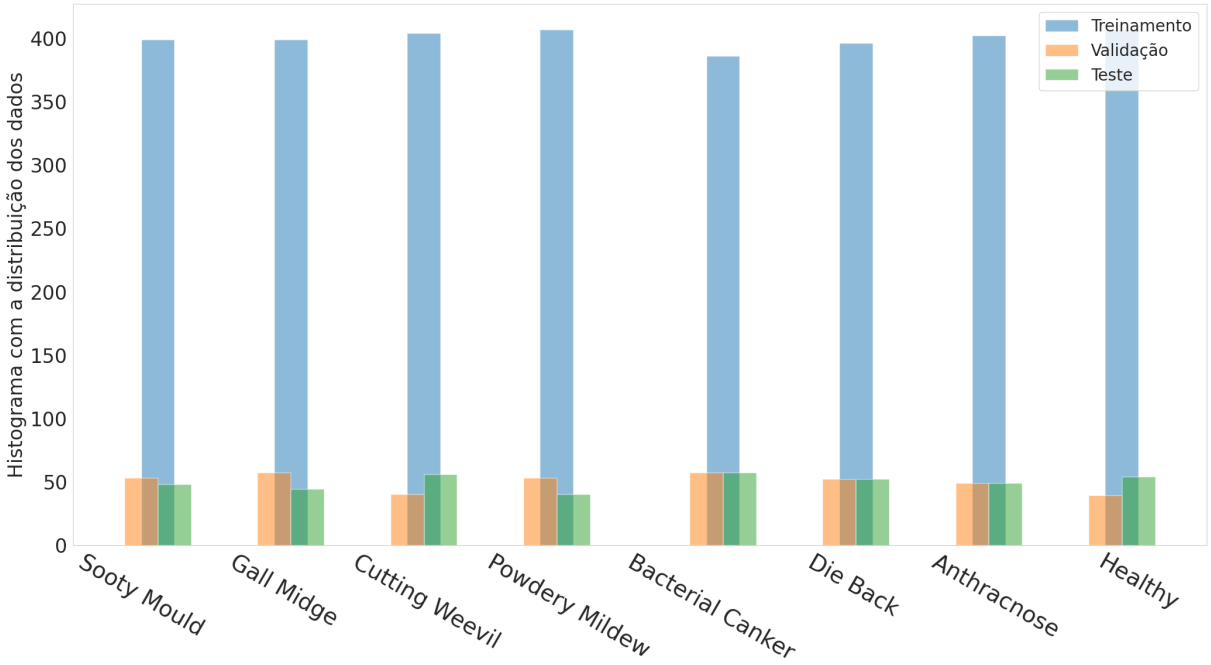

Figura 3.4: Histograma com a distribuição dos dados de treinamento (azul), validação (laranja) e teste (verde) por estado da folha da mangueira (Fonte: Elaborada pelo autor (2023)).

Ainda convém mencionar que foi utilizada a técnica de Data Augmentation durante o treinamento do modelo a partir do *Image Generator* do Tensorflow. Com isso, tentou-se evitar mais uma vez que o modelo tivesse o problema do overfitting. Nesse contexto, foram utilizados como Data Augmentation as seguintes transformações: zoom, flip, shear(cisalhamento), shift(deslocamento).

3.3.4 Hiperparâmetros utilizados

Nesta seção, serão apresentados os hiperparâmetros que foram definidos para a construção do modelo.

A Tabela 3.1 traz os hiperparâmetros utilizados no ImageDataGenerator. Com isso, os hiperparâmetros definidos foram utilizados no intuito de se evitar um *overfitting* no modelo. Além disso, temos a Tabela 3.2 que mostra os hiperparâmetros mais gerais que foram utilizados pelo modelo para o seu treinamento. Por fim, na Tabela 3.3, tem-se os hiperparâmetros utilizados para as camadas do modelo.

Tabela 3.1: Hiperparâmetros definidos para o ImageDataGenerator

Hiperparâmetros	**Valor**
Rescale	1/255
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.3
zoom_range	0.2
horizontal_flip	True
fill_mode	Nearest

Tabela 3.2: Hiperparâmetros definidos para o modelo

Hiperparâmetros	**Valor**
Tamanho de lote	64
Tamanho da imagem	(256, 256, 3)
Otimizador	Adam
Função de perda	*Categorical Cross-Entropy*
Número de épocas	25

Tabela 3.3: Hiperparâmetros definidos para as camadas do modelo

Hiperparâmetros	**Valor**
Batch Normalization	Padrão
Dropout 1	0.2
Camada Densa Intermediária	**Unidades**: 64; **Ativação**: relu
Dropout 2	0.2
Camada Densa Final	**Unidades**: 8; **Ativação**: *Softmax*

3.4 Conversão do modelo

Após a construção do modelo e sua avaliação, que será melhor descrita na seção Resultados, houve a conversão do modelo utilizando o TensorFlow Lite[10]. O TensorFlow Lite é uma biblioteca móvel para implantação de modelos em dispositivos mobile, microcontroladores e outros dispositivos de borda [1]. Além disso, apresenta como uma de suas características

[10] https://www.tensorflow.org/lite

principais as seguintes restrições: latência, privacidade, conectividade, tamanho e consumo de energia [1].

Para a conversão do modelo, foi utilizada a otimização padrão fornecida pelo TensorFlow Lite, esta utiliza a técnica de quantização para otimizar o modelo. A quantização funciona reduzindo a precisão dos números usados para representar parâmetros de um modelo que por padrão são números de ponto flutuante de 32 bits [1]. Com isso, há uma redução significativa do tamanho do modelo e uma computação mais rápida [1]. Em contrapartida, existe uma tendência de queda na precisão do modelo a partir desta otimização. O modelo foi convertido para um arquivo *tflite* para poder ser utilizado no aplicativo para detecção de doenças em folhas de manga.

3.5 Desenvolvimento do aplicativo mobile

Nesta seção, será descrito como foi feita a construção do aplicativo para detecção de doenças em folhas de manga[11]. Além disso, quais as funcionalidades que foram implementadas. Entretanto, pode-se deixar evidente que o aplicativo foi construído para funcionar exclusivamente em dispositivos móveis Android.

3.5.1 Linguagem de programação

Para a construção do aplicativo, foi utilizada a linguagem de programação Kotlin[12]. Sua escolha foi determinada, principalmente, pela facilidade de trabalhar com o Tensorflow Lite. Ademais, essa linguagem é estaticamente tipada e considerada moderna entre os seus usuários, além ser utilizada por mais de 60% dos desenvolvedores Android profissionais [8].

3.5.2 SDK utilizado

Como o intuito do aplicativo é que ele seja simples, a escolha do SDK se deu com o objetivo de tornar o aplicativo disponível nos mais diversos dispositivos mobile Android. A partir disso, a versão mínima utilizada foi a 24 (Android 7.0 Nougat) e a versão alvo foi a 33 (Android 13.0 Tiramisu). Conforme a Figura 3.5 abaixo, temos que a nossa versão mínima 24 permite que o nosso aplicativo funcione em cerca de 95,4% dos dispositivos móveis Android.

3.5.3 Funcionalidades do aplicativo

Na construção do aplicativo mobile, foi adicionada uma funcionalidade básica, que será melhor descrita abaixo, que permite ao usuário de forma simples e eficaz tirar uma foto ou até

[11] https://github.com/itallocastro/TCCApp
[12] https://github.com/JetBrains/kotlin/releases/tag/v1.8.0

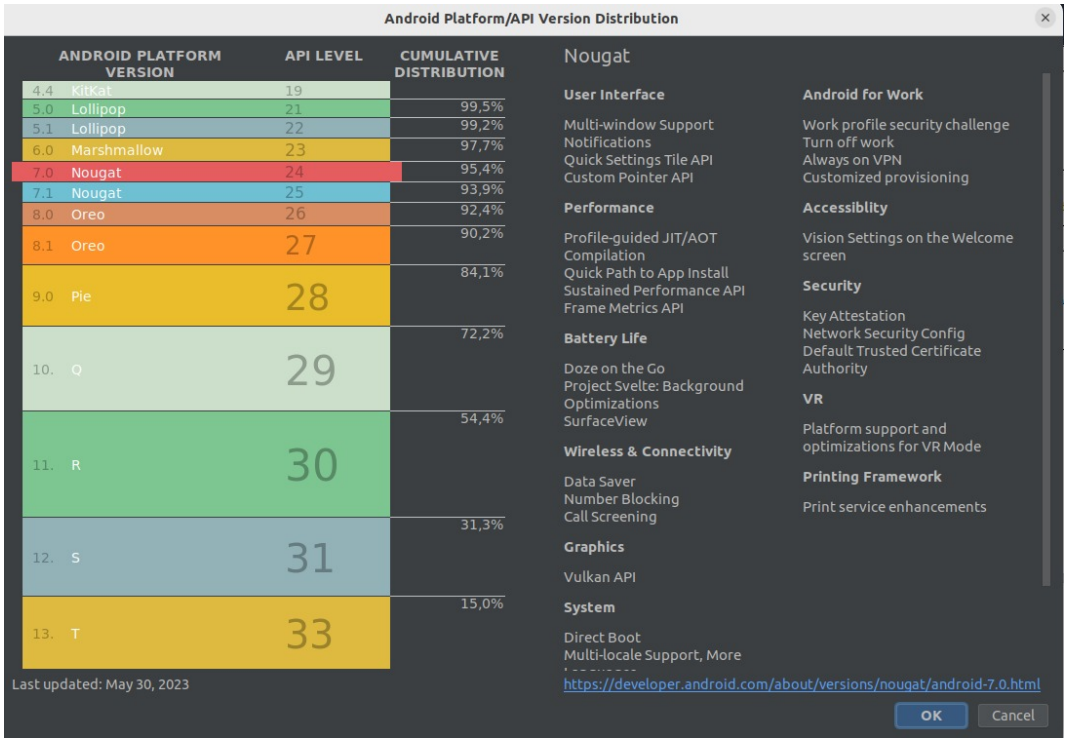

Figura 3.5: *Android Platform/API Version Distribution* (Fonte: [12]).

mesmo importar uma imagem de sua galeria para que o modelo convertido no aplicativo possa realizar uma predição sobre uma folha de manga.

Entretanto, ainda convém mencionar que foram adicionados duas outras funcionalidades com o intuito de realizar comparações entre um modelo convencional em um servidor e o modelo gerado para rodar dentro do aplicativo. Com isso, a ideia dessas outras funcionalidades é apenas servir de base para a geração de parte dos resultados deste trabalho.

Página inicial (Funcionalidade principal)

A página inicial, que pode ser vista na Figura 3.6, apresenta a aplicação real do modelo gerado e convertido para *TinyML*. A partir disso, nota-se que sua utilização se dá de maneira bastante simples. O usuário pode tirar uma foto de uma folha de manga e o aplicativo com o modelo embutido fará uma classificação dessa foto, cujas classes de doenças e a classe saudável já foram citadas neste trabalho. De maneira semelhante, ocorre se o usuário preferir carregar uma imagem de sua galeria. Ademais, pode-se observar o resultado da classificação feita pelo modelo, na Figura 3.7.

Monitoramento

A funcionalidade de monitoramento, que pode ser vista na Figura 3.8, foi construída para a comparação de desempenho entre o modelo convertido e o modelo original sem conversão. Como essa comparação foi realizada, será abordado em seções posteriores.

Figura 3.6: Página inicial do aplicativo (Fonte: Elaborada pelo autor (2023)).

Entretanto, podemos observar que a funcionalidade traz qual o modelo a ser usado para efeito de comparação, cujas opções são: Local e Servidor. Além disso, há o tipo de conexão que será avaliado, o modelo local apresenta apenas a opção de conexão “Modo Avião”, ou seja, o monitoramento será feito com o *smartphone* em modo avião. Ademais, no modo “Servidor”, têm-se as opções: 4G, 3G e *WiFi*.

Ainda convém mencionar que há um campo “Quantidade de vezes” que permite a realização de uma multiplicação entre esse valor do campo e as fotos selecionadas, a partir do botão “Selecionar”. Por exemplo, se “Quantidade de vezes” for igual a 10 e forem selecionadas 400 fotos, então o aplicativo irá capturar a latência e o uso de bateria 4000 vezes ao apertar o botão “Testar”.

Ao selecionar o modelo no modo “Servidor”, o aplicativo irá fazer requisições para o

Figura 3.7: Página inicial do aplicativo após a seleção de uma imagem da galeria (Fonte: Elaborada pelo autor (2023)).

servidor, que será melhor descrito na seção posterior. Todavia, se o modelo selecionado for o "Local", a classificação feita pelo modelo irá ocorrer dentro do próprio aplicativo.

A partir disso, temos os resultados que serão obtidos pelo teste efetuado anteriormente, os campos que serão observados serão "Latência Média" em milissegundos e "Uso de bateria" em miliampere-hora. A latência é medida a partir do tempo em que cada imagem leva para ser processada pelo modelo, então para cada foto será capturada sua latência e, após isso, será tirada uma média para exibição no aplicativo. O uso de bateria é obtido a partir da subtração entre a bateria inicial quando as imagens ainda não foram processadas e a bateria final após todas as imagens serem processadas.

Figura 3.8: Aba de monitoramento (Fonte: Elaborada pelo autor (2023)).

Configurações

A funcionalidade de configurações, que pode ser vista na Figura 3.9, foi desenvolvida exclusivamente para a seleção do modelo a ser usado e o tipo de conexão que será realizado na aba de monitoramento. Com isso, pode-se selecionar o modelo “Local” ou “Servidor”, já o “Tipo de comunicação” depende do modelo escolhido, ou seja, o modelo “Local” permite apenas o monitoramento em “Modo Avião”, já o modelo “Servidor” permite o monitoramento com *WiFi*, 4G e 3G.

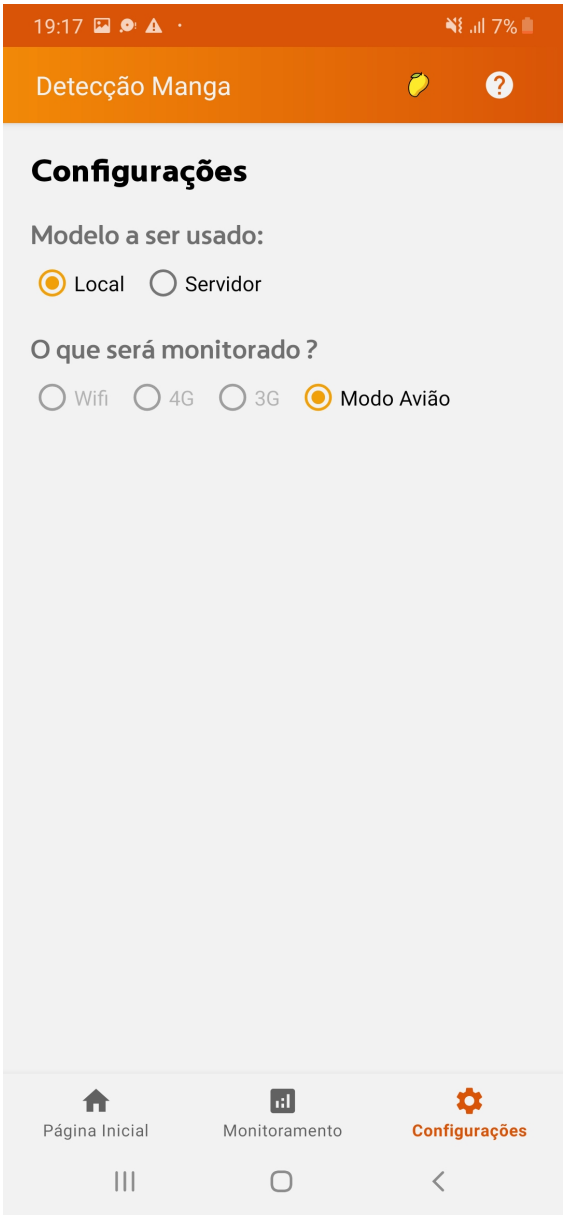

Figura 3.9: Aba de configurações (Fonte: Elaborada pelo autor (2023)).

3.6 Configuração dos ambientes

Nesta seção, serão apresentadas as configurações para os ambientes “Local” e “Servidor” utilizados para o monitoramento e para funcionalidade principal do aplicativo, classificar imagens com o objetivo de detectar doenças em folhas de manga.

3.6.1 Ambiente Servidor

Para o ambiente servidor, foi utilizado o framework Flask[13], com a linguagem de programação Python, para a construção de uma API para servir o nosso modelo original sem conversão

[13] https://flask.palletsprojects.com/en/2.3.x/

para *TinyML*. Além disso, fizemos o deploy dessa API na nuvem para realizarmos o monitoramento pelo aplicativo, para isso se fez necessária a utilização da AWS (*Amazon Web Services*), mais precisamente do serviço EC2[14] que permite a execução de servidores virtuais, ou seja, é um serviço que disponibiliza capacidade computacional segura e redimensionável na nuvem. As configurações da instância foram:

- Tipo de instância: *t3.large* (8GB de memória ram; 2 vCPUS)
- Volume da instância: SSD de uso geral (gp2) (30GB de memória)

3.6.2 Ambiente Local

Para o ambiente local, foi utilizado o próprio TensorFlow Lite para Android[15]. Com isso, o modelo convertido *tflite* foi implantado no próprio aplicativo.

3.6.3 Smartphone para o estudo

Para o estudo sobre o desempenho do modelo local e o modelo no servidor, foi utilizado o *smartphone* Samsung Galaxy M30, com as características elencadas abaixo:

- Android: Versão 9.
- Número do modelo: SM-M305M
- Processador: 1.8GHz Octacore
- Bateria: 5000mAh
- Memória RAM: 4GB
- Armazenamento: 64GB
- Resolução das Câmeras Traseiras (Câmera tripla): 13MP + 5MP + 5MP

[14] https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/concepts.html
[15] https://www.tensorflow.org/lite/android?hl=pt-br

4

Resultados e Discussões

Neste capítulo serão apresentados os resultados obtidos, de forma detalhada, na construção do aplicativo de detecção de doenças em folhas de manga.

4.1 Resultados dos modelos

Nesta seção, serão apresentados e discutidos os resultados obtidos pelo modelo original e o modelo após a conversão para ser executado no dispositivo móvel.

4.1.1 Modelo sem conversão

A partir dos hiperparâmetros e arquitetura citados no capítulo anterior, foi construído o modelo. Com isso, vale salientar que o modelo final apresentou **152MB** de tamanho depois de salvo.

Após o treinamento do modelo, foi realizada a avaliação de sua acurácia para os dados de teste, os quais o modelo não teve acesso durante o treinamento, os resultados desta avaliação, junto com as avaliações de treinamento e validação podem ser observadas na Tabela 4.1. Com isso, nota-se que o modelo apresentou quase 97% de acurácia com as imagens de teste, resultado muito parecido com o obtido por Mahbub et al., que apresentou cerca de 98% de acurácia com o seu melhor modelo.

A matriz de confusão, na Figura 4.1, permite salientar que o modelo gerado dificilmente se confundiu em suas classificações, o que corrobora com a taxa alta de acurácia. Além disso, foram geradas outras métricas, como: *precision*, *recall*, *f1-score* e *support*; essas métricas podem ser observadas na Tabela 4.2. Com isso, pode-se observar que o modelo apresentou ótimos resultados em todas as classes propostas e, ademais, percebe-se que em um contexto real e prático os falsos negativos devem ser mitigados, ou seja, o modelo deve evitar classificar como

saudável uma folha que na verdade está doente. Nesse caso, o modelo obteve excelentes resultados, porque errou apenas uma vez ao dizer que a folha era saudável, mas na verdade estava com a doença *Gall Midge*, como se pode observar na Figura 4.1.

Tabela 4.1: Avaliação do modelo original.

Fase	**Acurácia**	**Perda**
Treinamento	0.989	0.0757
Validação	0.971	0.4615
Teste	0.967	0.137

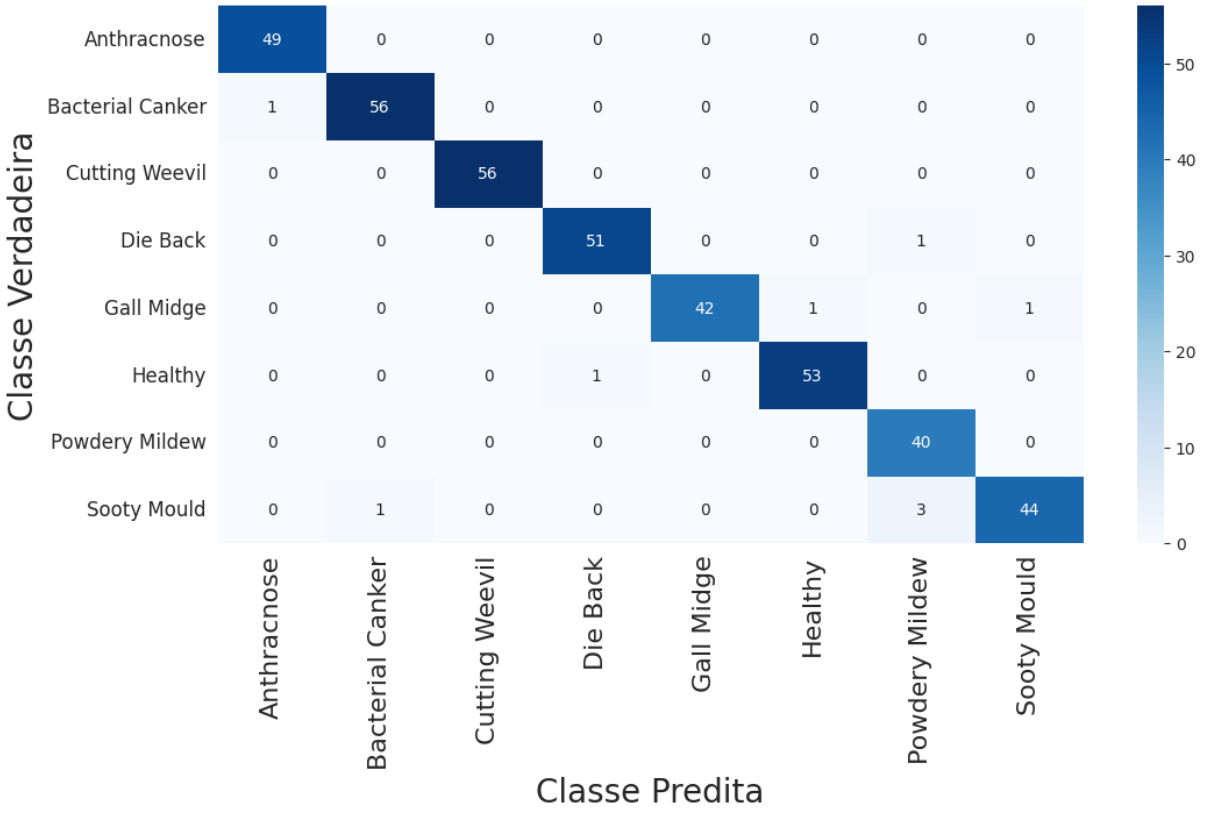

Figura 4.1: Matriz de confusão para o modelo original (Fonte: Elaborado pelo autor (2023)).

Tabela 4.2: Métricas retornadas pelo modelo original.

	Precision	***Recall***	***F1-Score***	***Support***
Anthracnose	1.0	1.0	1.0	49
Bacterial Canker	1.0	1.0	1.0	57
Cutting Weevil	1.0	1.0	1.0	56
Die Back	0.98	1.0	0.99	52
Gall Midge	0.98	0.98	0.98	44
Healthy	0.98	0.96	0.97	54
Powdery Mildew	0.95	0.97	0.96	40
Sooty Mould	0.98	0.96	0.97	48

4.1.2 Modelo convertido

Após a construção e avaliação do modelo, que foi descrita na seção anterior, este foi convertido para a extensão *tflite* aplicando a quantização como otimização. Com isso, nosso modelo convertido passou a ocupar **26,8 MB** de memória, uma redução de aproximadamente 82% quando comparado com o original sem conversão. Portanto, nota-se uma redução bastante significativa no espaço de armazenamento necessário para a implantação. Além disso, o modelo convertido apresentou uma taxa similar de acurácia se comparado ao sem conversão, chegando a 98%.

A matriz de confusão pode ser observada na Figura 4.2. Além disso, as métricas avaliadas no modelo sem conversão podem ser também observadas na Tabela 4.3. A partir disso, nota-se pequenas variações nas métricas para cada classe possível, algumas melhoraram, outras pioraram. Contudo, ainda se pode observar que o modelo, mesmo convertido, apresenta um bom resultado em relação aos falsos negativos.

Tabela 4.3: Métricas avaliadas no modelo convertido.

	Precision	***Recall***	***F1-Score***	***Support***
Anthracnose	0.96	0.98	0.97	49
Bacterial Canker	1.0	0.96	0.98	57
Cutting Weevil	1.0	1.0	1.0	56
Die Back	1.0	0.98	0.99	52
Gall Midge	0.96	1.0	0.98	44
Healthy	0.98	0.98	0.98	54
Powdery Mildew	0.98	1.0	0.99	40
Sooty Mould	0.98	0.96	0.97	48

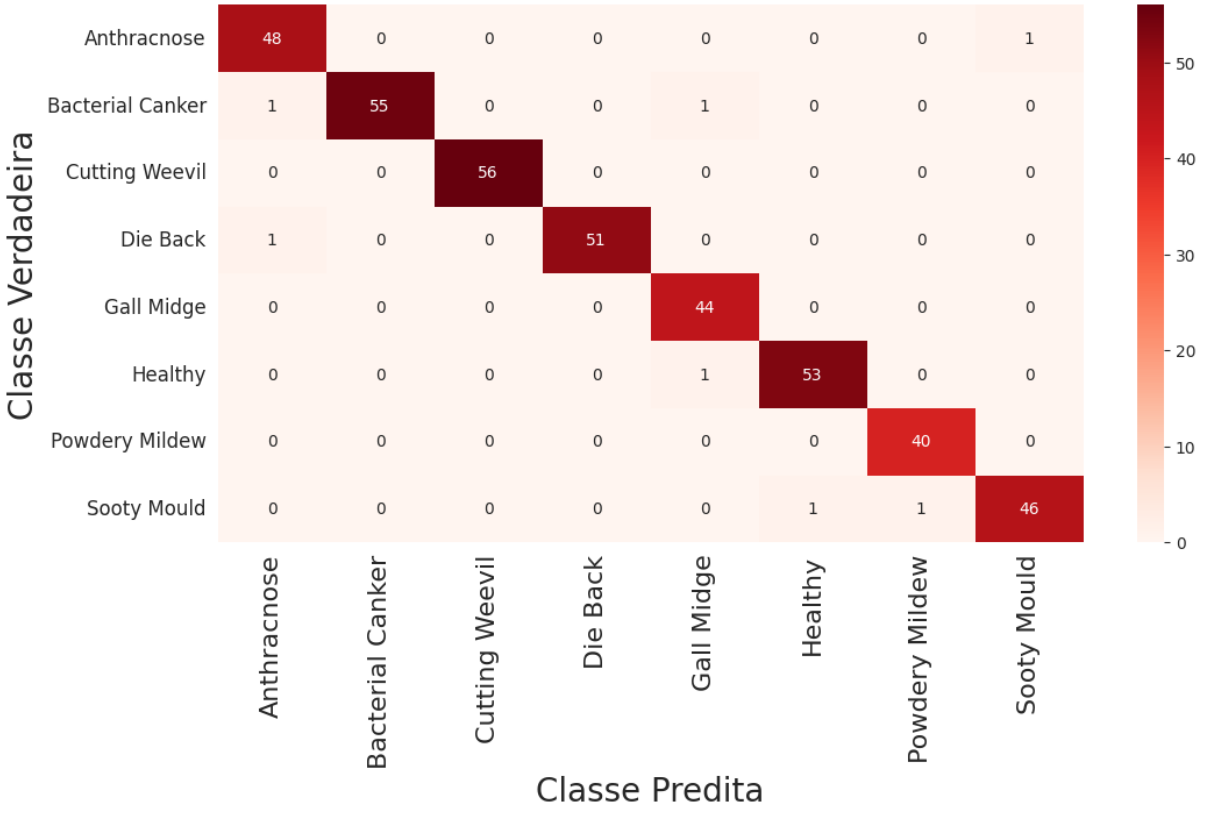

Figura 4.2: Matriz de confusão para o modelo convertido (Fonte: Elaborado pelo autor (2023)).

4.2 Resultados obtidos no monitoramento

Nesta seção, serão apresentados os resultados obtidos no monitoramento de bateria e de latência na classificação feita pelo modelo a partir do aplicativo construído. Além disso, serão feitos comparativos entre o modelo "Local" convertido e o modelo "Servidor" sem conversão, em relação às conexões: 3G, 4G, *WiFi* e "Modo avião", este consideramos a conexão no sentido de comunicação direta com o modelo implantado no aplicativo. Portanto, as conexões 3G, 4G e *WiFi* fizeram comunicação com o modelo no servidor, já o modo avião fez uso do modelo convertido e implantado no aplicativo.

Ainda convém mencionar que todas as possibilidades de modelo e conexão partiram de uma mesma configuração, 50% de bateria do smartphone e 10000 iterações(400 imagens de teste x 25).

4.2.1 Latência

A latência permite mensurar o tempo decorrido entre o envio da imagem até a resposta da classificação chegue ao aplicativo. Para o monitoramento, foram capturados 10000 mil dados de latência que correspondem às 10000 imagens utilizadas, como afirmado acima. Após isso, o próprio aplicativo calcula um valor médio para essas 10000 latências, como pode ser observado na Figura 3.8. Além disso, foi utilizado um arquivo *.xml* para salvar esses resultados.

A partir disso, fez-se uso de métricas estatísticas como média, mediana e desvio padrão para se observar o comportamento da latência quando variamos o modelo a ser usado (local ou servidor) e o tipo de conexão efetuada (3G, 4G ou *WiFi*, modo avião). Os resultados obtidos podem ser observados na Tabela 4.4 e na Figura 4.3. Observa-se, portanto, que quando foi utilizado o modelo local no modo avião, houve uma menor latência e um menor desvio padrão quando comparado com as outras possibilidades. Isso se dá, principalmente, pelo fato de que não há necessidade de conexão com internet, porque o modelo está rodando de maneira local no próprio aplicativo, o que evita esse aumento da latência, que é encontrada nas outras conexões, e também uma maior estabilidade, porque não lida com eventuais instabilidades de conexão. Por fim, na Figura 4.4, plotada com *outliers*, observa-se que as conexões 4G, 3G e *WiFi* apresentam diversos *outliers* bastante distantes de seus valores médios, o que corrobora com a ideia de instabilidade causada pela necessidade de conexão com a internet.

Tabela 4.4: Métricas para latência.

Conexão	**Modelo**	**Média (ms)**	**Mediana (ms)**	**Desvio padrão**
Modo Avião	Local	579.8	585.0	24.5
4G	Servidor	753.0	749.5	157.0
WiFi	Servidor	843.3	749.0	300.9
3G	Servidor	929.7	903.0	251.9

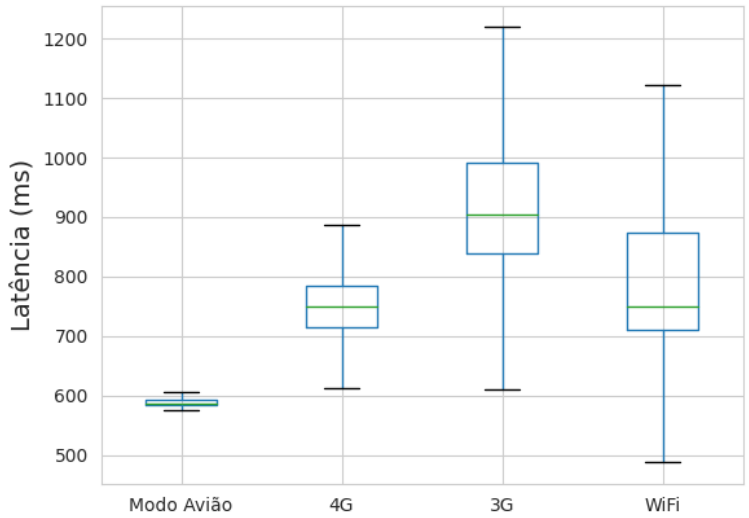

Figura 4.3: Boxplot das latências por conexão sem *outliers* (Fonte: Elaborado pelo autor (2023)).

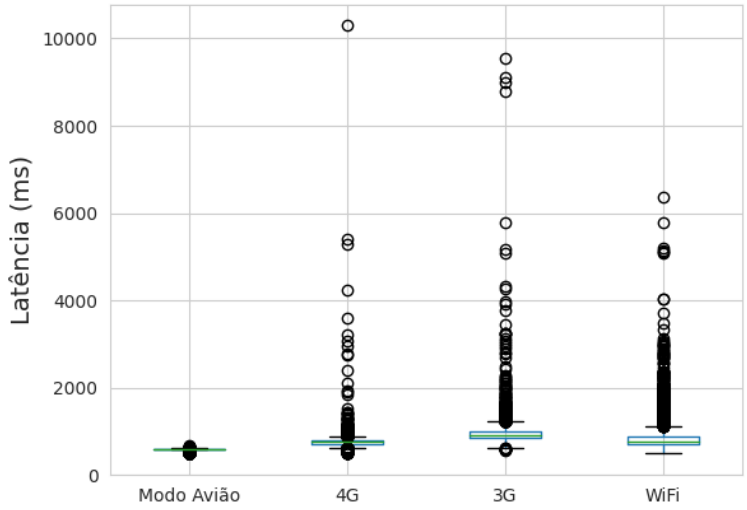

Figura 4.4: Boxplot das latências por conexão com *outliers* (Fonte: Elaborado pelo autor (2023)).

4.2.2 Consumo de bateria

Outro grande ponto estudado no problema em questão foi o consumo de bateria do *smartphone*. Com isso, foi capturada a bateria no momento anterior ao começo das classificações e depois foi capturada a bateria após todas as 10000 iterações, com o intuito de monitorar o consumo de bateria após esse tempo.

Os resultados obtidos, tanto em miliampere-hora (mAh) quanto em porcentagem (%), podem ser observados nas figuras 4.5 e 4.6. A partir deles, pode-se observar que o consumo de bateria com o modelo rodando no aplicativo é menor quando comparado com os que se conectam com o servidor, o que indica a vantagem de se utilizar o modelo de maneira local. Ao se olhar para o tipo de comunicação que mais utilizou a bateria, tem-se o 3G que ao ser comparado com o modo avião houve uma redução de aproximadamente 34% quando se utiliza o modelo de forma local, uma redução bastante significativa.

4.3 Discussões sobre os resultados

A partir de tudo que foi mencionado nas seções anteriores, pode-se observar a clara vantagem entre o modelo convertido e implantado no aplicativo e o modelo no servidor. O modelo local apresentou menor consumo de bateria, menor latência e ainda apresentou precisão de classificação semelhante quando comparado ao modelo original no servidor. Dessa forma, corrobora-se com os conceitos atribuídos ao *TinyML* [14], como: eficiência energética (rodar o modelo de maneira local no aplicativo, permitiu um menor consumo de bateria), latência reduzida (novamente, o modelo local apresentou menor latência), privacidade e segurança (com o aplicativo em funcionamento, não seria necessário o compartilhamento das imagens com a nuvem, o que aumenta a segurança) e baixo custo (nosso modelo convertido apresentava apenas

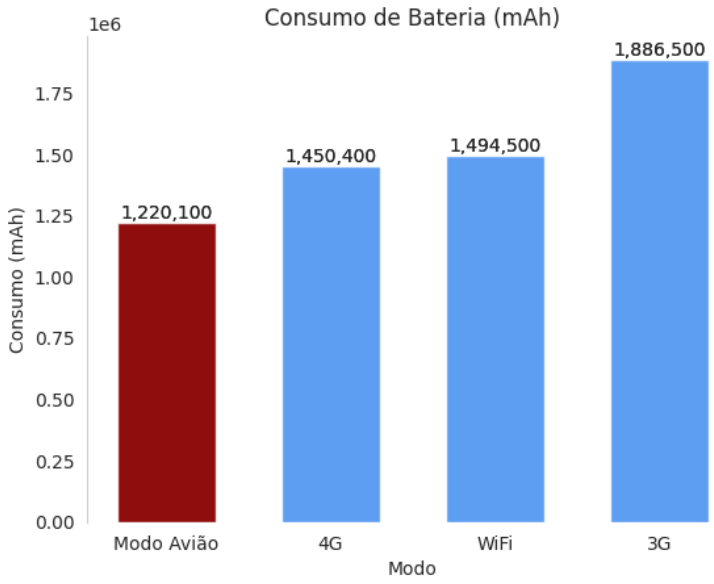

Figura 4.5: Consumo de bateria em mAh (Fonte: Elaborado pelo autor (2023)).

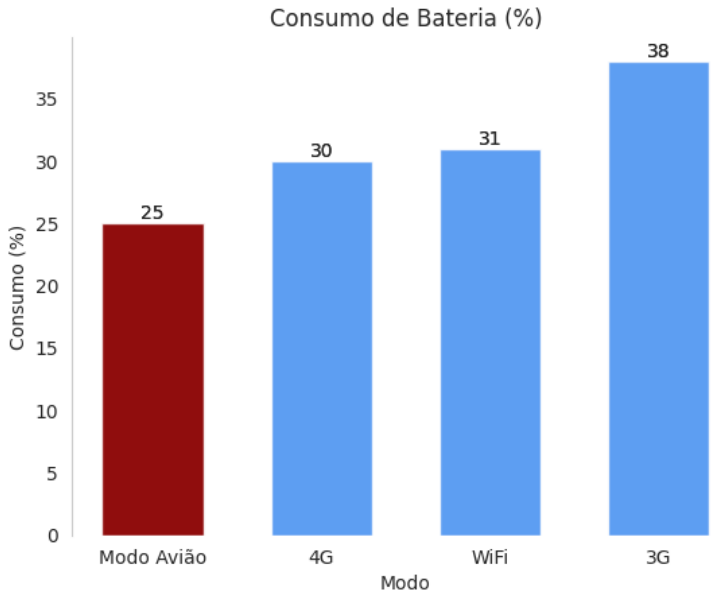

Figura 4.6: Consumo de bateria em % (Fonte: Elaborado pelo autor (2023)).

26,8 MB de consumo de armazenamento, o que indica que não precisa de muitos recursos para funcionar).

Entretanto, algumas limitações puderam ser notadas neste trabalho. A mais evidente foi a falta de folhas de manga devidamente rotuladas em posse do usuário do aplicativo para que a aplicação pudesse classificá-las, com intuito de simular uma situação real.

5

Conclusão

Este trabalho apresentou um estudo sobre os benefícios do *TinyML* na agricultura brasileira a partir de uma aplicação mobile para detecção de doenças em folhas de manga. Notadamente, pode-se perceber a importância da manga no contexto mundial e especificamente no contexto brasileiro. Dessa forma, há a necessidade de se detectar as doenças que afetam essa planta de forma precoce com o intuito de otimizar sua produção. Além disso, a construção deste trabalho permitiu um estudo mais geral sobre os benefícios do *TinyML* no contexto da agricultura. Com isso, foi possível observar toda a construção da aplicação final, desde a análise do banco de dados, passando pela construção do modelo, conversão do modelo, construção do aplicativo mobile e a implementação final do modelo para realizar as classificações, até a análise dos resultados obtidos em termos dos recursos utilizados.

Os resultados dos modelos mostraram que a conversão para *TinyML* proporcionou uma redução drástica no armazenamento necessário para a implantação do modelo no aplicativo e, ainda assim, não teve impacto significativo em sua acurácia. Os resultados obtidos a partir da análise dos recursos utilizados mostraram que o modelo convertido e implantado no dispositivo foi superior em termos de latência e bateria quando comparado com as conexões feitas, via 3G, 4G e *WiFi*, para o modelo original em um servidor hospedado na nuvem.

Portanto, quando são trazidos esses resultados para o contexto brasileiro, onde 90% da população que vive em áreas rurais possuem telefone móvel [13], tem-se uma grande área de utilização para o aplicativo. Ainda convém mencionar que, segundo IBGE, apenas 74.7% da população rural tem internet em seus domicílios, mas apenas em 69.5% dos domicílios rurais funciona serviços de telefonia móvel celular, o que indica que o acesso a internet ainda é limitado nessas regiões. Dessa maneira, pode-se imaginar que apesar de existir acesso à internet esse acesso acaba, muitas vezes, se limitando ao domicílio do usuário, ou seja, em suas plantações pode não haver conexão com a internet, o que dificultaria uma classificação se o modelo gerado rodasse em um servidor e não no aplicativo. Por fim, haveria uma redução inclusive no custo de internet, visto que o aplicativo não necessita de conexão alguma. Então, o aplicativo

construído neste trabalho pode colaborar para a detecção das doenças nas folhas de manga no contexto da agricultura brasileira, porque não se faz necessária a utilização de qualquer tipo de comunicação (3G, 4G e *WiFi*) para a sua utilização. Em um sentido mais amplo as aplicações em *TinyML* podem trazer os seus benefícios como o baixo custo, a eficiência energética, a latência reduzida e a segurança para diversos outros problemas e, assim, ajudar na resolução destes de forma eficiente e com redução de custos.

5.1 Trabalhos futuros

A partir do que foi identificado na seção de discussões dos resultados como uma limitação deste trabalho, serão listados alguns trabalhos a serem executados posteriormente:

- Trabalhos em colaboração com engenheiros agrônomos para que folhas em posse dos pesquisadores possam ser rotuladas para a aplicação possa classificá-las.
- Construção de novas bases de dados com doenças em folhas de outras frutas típicas da agricultura brasileira.
- Construção de um modelo que possa detectar doenças em folhas de diversas frutas diferentes.
- Construção de um modelo que possa detectar doenças em diversas folhas em tempo real.
- Construção de um modelo ainda mais otimizado em termos de armazenamento para que se possa rodá-lo em dispositivos com capacidade de armazenamento muito baixa.
- Trabalhos que analisem o impacto da conversão do modelo de diferentes maneiras em termos de acurácia.
- Trabalhos que façam o monitoramento da latência e da bateria com o aplicativo no campo.

Referências bibliográficas

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/lite/guide. Software available from tensorflow.org.

[2] Sarder Iftekhar Ahmed, Muhammad Ibrahim, Md Nadim, Md Mizanur Rahman, Maria Mehjabin Shejunti, Taskeed Jabid, and Md Sawkat Ali. Mangoleafbd: A comprehensive image dataset to classify diseased and healthy mango leaves. *Data in Brief*, 47:108941, 2023.

[3] Diógenes Batista, Pedro Júnior, Maria Barbosa, Juliana Andrade, and Daniel Terao. Doenças da mangueira. 37:82–91, 01 2016.

[4] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017. ISBN 1617294438.

[5] I Citaristi. Specialized agencies and related organizations within the un system: Food and agriculture organization of the united nations-fao, in: The europa directory of international organizations 2022. routledge. pages 307–315, 2022.

[6] Google Cloud. Guia avançado do inception v3. URL https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=pt-br. publisher: Google Cloud.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. **DOI** 10.1109/CVPR.2009.5206848.

[8] Google For Developers. Desenvolver apps android com o kotlin. URL https://developer.android.com/kotlin?hl=pt-br. publisher: Google For Developers.

[9] Dr. Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021. ISSN 2542-6605. **DOI** https://doi.org/10.1016/j.iot.2021.100461. URL https://www.sciencedirect.com/science/article/pii/S2542660521001025.

[10] FOOD and AGRICULTURE ORGANIZATION OF THE UNITED NATIONS (FAO). Faostat, 2020. URL http://www.fao.org/faostat/en/#data/QC. publisher: FAO.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Google. Painel de distribuição. URL https://developer.android.com/about/dashboards?hl=pt-br. publisher: Google for Developers.

[13] IBGE. Pesquisa nacional por amostra de domicílios contínua - acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal 2021, 2021. URL https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101963. publisher: IBGE.

[14] Gian Marco Iodice and Ronan Naughton. *TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter*. Packt Publishing Ltd, 2022. ISBN 9781801814973. URL https://www.packtpub.com/product/tinyml-cookbook/9781801814973.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[16] Nosin Ibna Mahbub, Feroza Naznin, Md Imran Hasan, Syed Mahfuzur Rahman Shifat, Md. Alamgir Hossain, and Md Zahidul Islam. Detect bangladeshi mango leaf diseases using lightweight convolutional neural network. In *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6. IEEE, 2023. **DOI** 10.1109/ECCE57851.2023.10101648.

[17] Azeem Mirani, Engr Dr Muhammad Suleman Memon, Rozina Chohan, Asif Wagan, and Mumtaz Qabulio. Machine learning in agriculture: A review. *LUME*, 10:229–234, 05 2021.

[18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2 edition, 2018. ISBN 978-0-262-03940-6.

[19] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. **DOI** 10.1109/TKDE.2009.191.

[20] Junjie Peng, Elizabeth Jury, Pierre Dönnes, and Coziana Ciurtin. Machine learning techniques for personalised medicine approaches in immune-mediated chronic inflammatory diseases: Applications and challenges. *Frontiers in Pharmacology*, 12, 09 2021. **DOI** 10.3389/fphar.2021.720694.

[21] R. C. Ploetz. Diseases of mango. *CABI Books*, page 327–363, 2003. URL https://doi.org/10.1079/9780851993904.0327.

[22] Adrielle Rodrigues Prates, Patrícia Graosque Ulguim Züge, Sarita Leonel, Jackson Mirellys Azevêdo Souza, and Jorgiani de Ávila. Flowering induction in mango tree: updates, perspectives and options for organic agriculture. *Pesquisa Agropecuária Tropical*, 51:e68175, 2021. ISSN 1983-4063. **DOI** 10.1590/1983-40632021v5168175. URL https://doi.org/10.1590/1983-40632021v5168175.

[23] Abdul Rehman, Luan Jingdong, Rafia Khatoon, and Imran Hussain. Modern agricultural technology adoption its importance, role and usage for the improvement of agriculture. *American-Eurasian Journal of Agricultural Environmental Sciences*, 16:284–288, 02 2016. **DOI** 10.5829/idosi.aejaes.2016.16.2.12840.

[24] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 07 2019. **DOI** 10.1186/s40537-019-0197-0.

[25] Nitish Srivastava. Improving neural networks with dropout, June 2013. Available at http://www.cs.toronto.edu/~nitish/msc_thesis.pdf.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 06 2016. **DOI** 10.1109/CVPR.2016.308.

[27] Vasileios Tsoukas, Anargyros Gkogkidis, and Athanasios Kakarountas. A tinyml-based system for smart agriculture. In *Proceedings of the 26th Pan-Hellenic Conference on Informatics*, PCI '22, page 207–212, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450398541. **DOI** 10.1145/3575879.3575994. URL https://doi.org/10.1145/3575879.3575994.

[28] Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), 05 2016. **DOI** 10.1186/s40537-016-0043-6.

[29] Lucas Moura Xavier and Thales Augusto Medeiros Penha. O desempenho das exportaÇÕes da manga no brasil: Uma anÁlise de constant market share. *Revista Análise Econômica e Políticas Públicas - RAEPP*, (01), jul. 2021. URL https://periodicos.apps.uern.br/index.php/RAEPP/article/view/3286.