UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Classificação e detecção de doenças no plantio de batata por meio de redes neurais convolucionais

Thiago Francisco Martins

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Thiago Francisco Martins

Classificação e detecção de doenças no plantio de batata por meio de redes neurais convolucionais

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Da. Moacir Antonelli Ponti

Versão original

São Carlos

2022

Thiago Francisco Martins

Potato disease recognition using computer vision

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Moacir Antonelli Ponti

Original version

São Carlos

2022

Dedico este trabalho a todos
os interessados por melhorar a agricultura
por meio da tecnologia.

AGRADECIMENTOS

Agradeço aos meus pais pelo empenho e dedicação para que eu pudesse ter uma educação de qualidade. Agradeço a minha esposa pelo companheirismo e suporte até nos momentos mais díficeis.

"Se você não está melhorando,
você está se piorando."

RESUMO

Martins, T.F. **Reconhecimento de doenças no plantio de batata por meio de visão computacional**. 2022. 61p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

A agricultura auxiliada por tecnologia, em particular a visão computacional e aprendizado de máquina, pode auxiliar provendo informações sobre os cultivos. Em particular nesse trabalho abordamos o plantio da batata.Este documento trata sobre a visão computacional no reconhecimento de doenças na plantação de batata, um dos vegetais mais consumidos em alguns países. O reconhecimento de doenças na plantação é essencial para que medidas possam ser tomadas com efetividade. Esta técnica preventiva é promissora pois pode identificar pragas e doenças em estágios que até mesmo humanos teriam dificuldade em identificar, principalmente em larga escala, e com o crescimento do poder de processamento de dispositivos portáteis como smartphones, torna-se uma solução de interesse prático. Por meio de modelos de aprendizado de máquina, foi desenvolvido uma solução capaz de classificar e detectar regiões de doença no plantio de batata. Os resultados mostram uma alta taxa de acerto em ambas tarefas com um tempo de processamento em dezenas de mili segundos. Por fim, é discutido como a solução é capaz de reconhecer com confiança doenças, bem como os motivos de ser uma ótima solução para o produtor agrícola do ponto de vista prático.

Palavras-chave: plantio de batata, redes neurais, visão computacional. Dissertação. Trabalho de conclusão de curso (TCC).

ABSTRACT

Martins, T.F. **Potato disease recognition using computer vision**. 2022. 61p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Technology-aided agriculture, in particular computer vision and machine learning, can help by providing information about crops. In particular, in this work we approach the planting of potatoes. This document deals with computer vision in the recognition of diseases in the potato plantation, one of the most consumed vegetables in some countries. Recognition of crop diseases is essential so that measures can be taken effectively. This preventive technique is promising as it can identify pests and diseases at stages that even humans would have difficulty identifying, especially on a large scale, and with the growth in processing power of portable devices such as smartphones, it becomes a solution of practical interest. Using machine learning models, a solution capable of classifying and detecting disease regions in potato crops was developed. The results show a high success rate in both tasks with a processing time around dozen milliseconds. Finally, it is discussed how the solution is able to confidently recognize diseases, as well as the reasons for being a great solution for the agricultural producer from a practical point of view.

Keywords: Computer vision. Neural networks. Mobilenet. Potato Cultive. Thesis. Dissertation. Conclusion course paper.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Batata é um dos alimentos mais consumidos globalmente, sendo produzidas 370 milhões de toneladas anualmente no mundo (FOOD; NATIONS, 2021). O consumo médio anual *per capita* deste alimento no mundo chega a 32,3 kg (FOOD; NATIONS, 2021), é o vegetal mais consumido em países como os Estados Unidos (SERVICE, 2021). Batata também é um alimento sensível a doenças, sendo acometida por cerca de 70 diferentes tipos (SALAS; TÖFOLI, 2017) oriundas de fungos, bactérias, vírus e nematódeos.

De acordo com Salas and Töfoli (2017), um produtor deve estar atento às possíveis doenças durante seu plantio e um monitoramento constante deve ser feito tendo em vista a detecção de pragas em seus estágios iniciais. Desta forma, pode-se efetuar medidas de combate como a aplicação agrotóxicos, tratamento do solo e fertilizantes ou até mesmo a remoção da região da plantação onde há indícios de doença, para que ela não se espalhe para outras áreas da lavoura. Os mesmos autores apontam que a produção pode ser severamente degradada caso problemas não sejam detectados a tempo, pois geralmente algumas doenças são altamente contagiosas.

Com base no fato de que pragas e doenças são um dos principais riscos na produção agrícola conforme mencionado anteriormente, técnicas de aprendizado de máquina e visão computacional foram exploradas com objetivo de detectar esses eventos em escala, consequentemente reduzindo potenciais riscos. Desta forma pode-se prevenir perdas excessivas na lavoura e até mesmo reduzir o uso excessivo de agrotóxicos.

Ainda conforme Salas and Töfoli (2017), há um grande número de doenças específicas do cultivo da batata, e o objetivo deste projeto é também ajudar o produtor a identificar o nome de tais doenças (dentro do leque de rótulos de treinamento usado durante o treinamento) que estão se desenvolvendo em sua plantação. Espera-se que o produtor, com o auxílio de dispositivos de captura de imagens em sua lavoura (exemplo: *smartphones* e dispositivos com câmera embarcada como *drones*), possa detectar com antecedência doenças e pragas em seu plantio.

1.1 Justificativa e Motivação

Identificar pragas em produção agrícola é uma tarefa difícil até mesmo para humanos, geralmente o produtor deve ter conhecimento de todas as possíveis pragas e doenças que podem impactar sua plantação e também inspecionar visualmente toda a área de produção com uma frequência determinada (JONG; JONG; SIECZKA, 2011). Esta tarefa se dificulta conforme o aumento da área do plantio. Técnicas de visão computacional e aprendizado de máquina podem ajudar nesta tarefa pois podem ser efetuadas de forma

automática por dispositivos eletrônicos.

No contexto de plantação de batata, outros fatores são essenciais para o bom cultivo da batata, como a correta preparação do solo, irrigação, método de plantio, entre outros (JONG; JONG; SIECZKA, 2011), a detecção antecipada de pragas é apenas uma das formas de atingir o sucesso da colheita. De forma geral, estas pragas podem ser divididas entre dois segmentos: as que atacam a parte subterrânea das plantas e as que atacam a parte aérea (a partir do coleto da planta). Este projeto de pesquisa propõe meios de ajudar o produtor nas doenças do segundo segmento, ou seja, pragas que podem assolar a plantação na sua parte aérea, principalmente doenças ligadas às folhas das plantas.

Ganhos relacionados à detecção de pragas nos estágios iniciais envolvem a possibilidade do combate antecipado a estas pragas, evitando assim o contágio para outras áreas da plantação, a redução do uso de agrotóxicos, reduzir gastos com recursos humanos para este tipo de tarefa, e ganhos na produção. A proposta deste projeto envolve estudar a eficácia de diferentes métodos propostos pela literatura (MOHANTY; HUGHES; SALATHÉ, 2016), especificamente técnicas de aprendizado supervisionado em imagens de plantas tanto saudáveis quanto doentes disponíveis na literatura. Há diversos estudos relacionados à detecção de doenças em diversos cultivos na agricultura (JIANG *et al.*, 2019; KIRATIRATANAPRUK *et al.*, 2020; MOHANTY; HUGHES; SALATHÉ, 2016) utilizando aprendizado supervisionado e não supervisionado, entretanto o objetivo deste projeto terá como foco o plantio da batata.

Existem vários fatores que podem impactar o desenvolvimento de um algoritmo confiável e robusto para esta tarefa, entre elas estão variações nas condições de iluminação do ambiente, variação no ângulo de captura de imagem, a falta de amostras com doença para o aprendizado e a falta de poder de processamento nos dispositivos que serão usados em campo (como *smartphones* ou câmeras embarcadas). Este projeto irá também investigar formas de mitigar estes problemas usando técnicas de aumento de dados ou *data augmentation* (PEREZ; WANG, 2017), técnicas de aprendizado chamadas de few-shot learning (SUNG *et al.*, 2018) e técnicas para inferência em dispositivos com baixo processamento.

1.2 Questões de Pesquisa

Neste trabalho, espera-se que com a rotulação de imagens em diferentes classes de doenças no plantio da batata, o algoritmo possa reconhecer as classes de interesse em ambientes não vistos anteriormente. Diante dos desafios e problemas atualmente enfrentados em sistemas de detecção e classificação de doenças através de imagens, foi elaborada a seguinte questão de pesquisa que norteará este projeto:

Q1 *"Algoritmos de aprendizado de máquina possuem a capacidade de aprender a*

partir de exemplos visuais e generalizar para imagens não vistas para detecção de pragas em plantações de batata de forma eficiente?"

Diante desta questão de pesquisa, são definidos os seguintes objetivos para o desenvolvimento deste trabalho:

- Mapear algoritmos de aprendizado de máquina disponíveis na literatura, analisando suas lacunas e desempenhos de classificação na detecção e classificação de doenças e pragas no plantio da batata, considerando fatores como a escassez de exemplos de doenças, robustez às condições de iluminação e baixo poder de processamento. Aplicar algoritmos que se destacam na literatura no desenvolvimento de uma aplicação simples para detecção de imagens de folhas doentes;

- Comparar os desempenhos obtidos a algoritmos de classificação do estado da arte utilizando a medida F1. Este objetivo está relacionado à questão de pesquisa Q1;

- Avaliar se o sistema proposto é rápido o suficiente para dispositivos de menor poder computacional;

- Discutir os resultados em nível de classificação de imagens e detecção de regiões afetadas pela doença.

A partir do modelo proposto, espera-se que o desempenho do algoritmo proposto tenha baixa revocação, baixo tempo de inferência em *smartphones* e precisão comparável ao estado da arte neste ramo.

1.3 Organização do Trabalho

Este trabalho está distribuído em 5 capítulos, incluindo esta introdução, dispostos conforme a descrição que segue:

Capítulo 1: Apresenta uma introdução sobre problemas encontrados no plantio da batata, bem como soluções tecnológicas para solucioná-las.

Capítulo 2: Nesta parte, o sistema concebido é abordado de forma teórica apresentando os dados utilizados para esta tarefa, métodos de visão computacional, redes neurais e técnicas de pré-processamento, e métricas utilizadas.

Capítulo 3: Descreve os materiais utilizados, bem como todos os métodos utilizados para desenvolver a solução. Todas as etapas de desenvolvimento do projeto são detalhadas.

Capítulo 4: Os resultados do sistema desenvolvidos são apresentados. Tópicos como requerimentos, consumo de recursos, taxa de quadros, taxa de acerto dos algoritmos de visão computacional considerando as métricas discutidas.

Capítulo 5: São apresentadas as conclusões do trabalho e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os fundamentos teóricos e práticos necessários ao entendimento do desenvolvimento do projeto. São abordados os algoritmos de visão computacional para detecção e reconhecimento de doenças, bem como descrição dos dados utilizados.

2.1 Visão computacional

Uma imagem digital pode ser entendida como uma imagem contínua amostrada em um arranjo matricial $M \times N$, sendo o valor de cada elemento da matriz o nível de cinza do pixel correspondente no plano de imagem (GONZALEZ; WOODS; MASTERS, 2009).

Conforme observa-se na Figura 1, pixels de intensidade mais clara possuem valores superiores aos de pixels de intensidade escura. A missão da visão computacional é processar estes valores para que características da imagem digital possam ser extraídas e analisadas. Um dos campos da visão computacional é a área de redes neurais, métodos que se baseiam na estrutura do sistema nervoso central humano e pode ser utilizada em várias tarefas, entre elas o reconhecimento de padrões.

2.1.1 NMS

Non-Maximum Suppression (NMS) (BODLA *et al.*,), é um algoritmo de visão computacional que elimina caixas delimitadoras que estejam sobrepostas uma a outra ou que tenham baixa confiança.

Figura 1 – Imagem digital. (a)Exemplo de uma imagem digital. (b) Representação matricial correspondente.

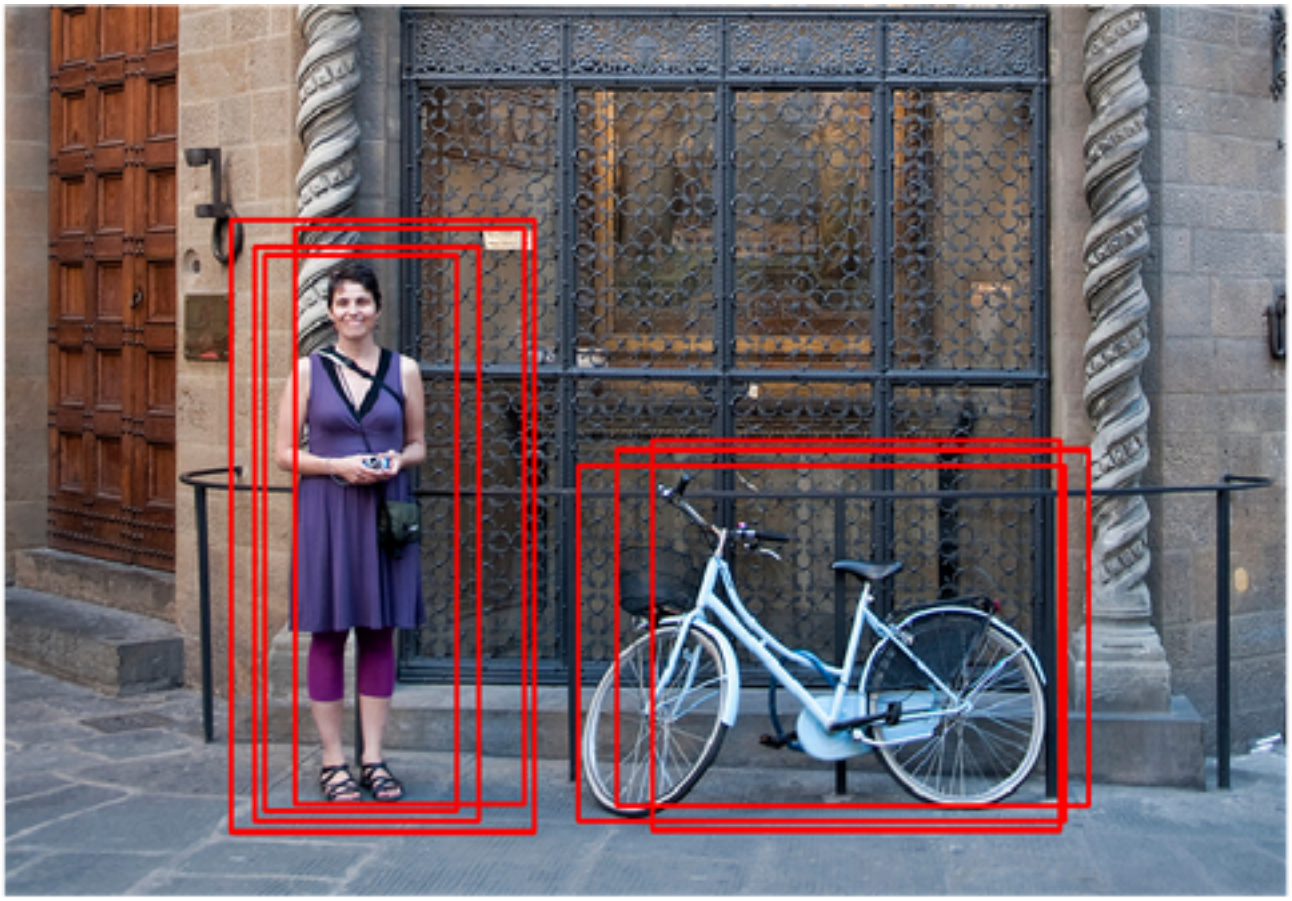

Figura 2 – Algoritmo NMS em ação.

O algoritmo consiste em primeiro encontrar a caixa delimitadora com maior confiança gerada pelo modelo. Posteriormente, eliminamos todas as caixas delimitadoras que tenham alta sobreposição (exemplo: IoU maior que 50%). Após isto, repete-se o processo com as próximas caixas delimitadoras até que não haja nenhuma caixa para eliminar.A Figura 2 ilustra um caso onde o algoritmo entra em ação

2.2 Fotografia

Dado que a aplicação em questão envolve captura de imagens em condições desafiadoras, foi necessário estudar algumas características de fotografia que devemos levar em consideração antes de desenvolver um projeto com o intuito de evitar cenários não previstos pelo sistema. Um caso a ser evitado por exemplo são as condições diferentes de captura realizadas em campo pelo usuário e as condições de captura realizadas pelos criadores da base de dados, que conforme será visto na sessão 2.10 pode ser obtida com iluminação fluorescente, em ambientes fechados, com câmeras estabelecidas em um tripé, e com pouco movimento dentro da cena. Estas condições de fotografia serão bem diferentes em ambientes externos de iluminação proveniente do sol e com movimento advindos de

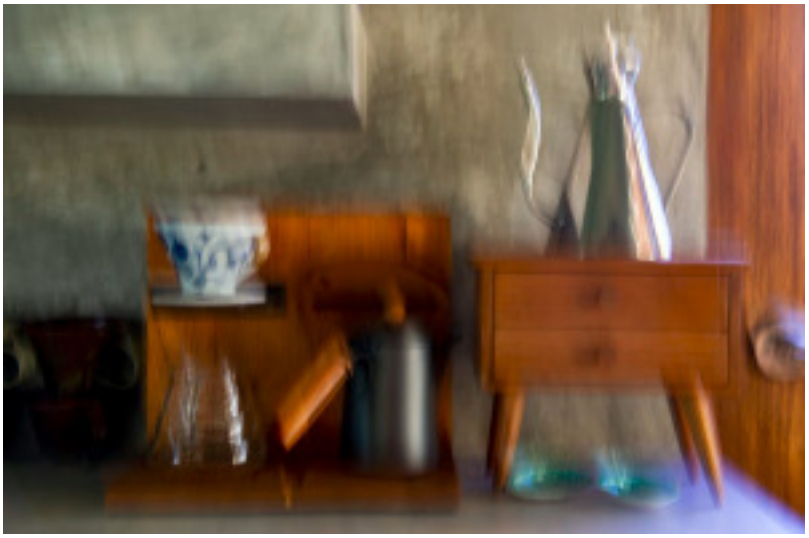

Figura 3 – Exemplo de imagem com borrão causado por movimento.

ventos e até mesmo do próprio usuário. Por isso algumas considerações de fotografia serão consideradas para o desenvolvimento do projeto.

2.2.1 Exposição

Existem três formas de se controlar a quantidade de luz que entra nos sensores de câmeras. A principal forma dela é variando a abertura. Controlando o tamanho da abertura, uma pessoa consegue deixar com que a luz proveniente de ambientes externos entre em maior ou menor quantidade nos sensores, alterando assim o resultado do brilho da imagem (TAYLOR *et al.*, 2015). Uma segunda forma de controlar a intensidade de iluminação é pela velocidade do obturador, a velocidade do obturador nada mais é do que o tempo que o sensor da câmera fica ativo capturando luz proveniente de ambientes externos. Aumentando o tempo de captura pode-se conseguir uma imagem com maior brilho, porém o fotógrafo deve estar atento à possibilidade de movimento da câmera ou da cena, já que se houver movimento nestas condições a foto pode sair "borrada", com traços de movimento conforme Figura 3.

Uma última forma de controlar a iluminação é através do ISO, que seria a sensibilidade do sensor. Podemos aumentar a sensibilidade do sensor à luz proveniente de ambientes externos para aumentar a iluminação, entretanto esta ação pode geras aumento indesejado de ruído na foto, conforme visto em Figura 4.

2.2.2 Cor e balanceamento de branco

Um dos aspectos mais importantes na fotografia é o balanço de branco. Cada tipo de iluminação emite um comprimento de onda diferente, e por isso geram como resultado cores diferentes quando atingem os sensores. A maioria das câmeras possuem o balanço de branco automático, que corrige a tendência de cores de determinada câmera. Por exemplo,

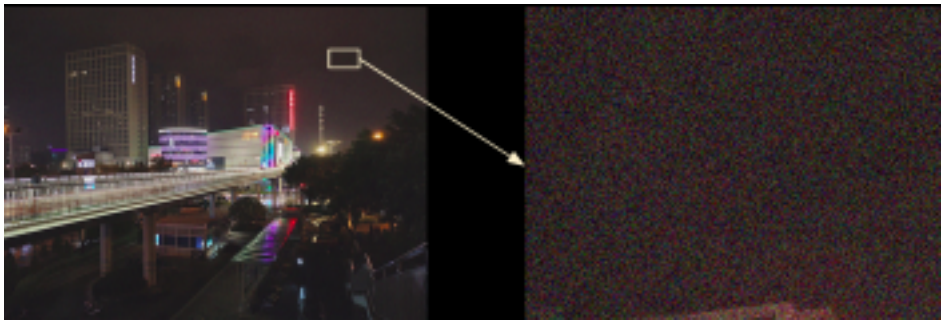

Figura 4 – Exemplo de imagem com ruído causado por alto ISO.

uma superfície branca em uma cena não terá cores neutras se iluminadas por uma lâmpada incandescente, da mesma forma uma superfície branca não terá tons neutros se capturada usando uma iluminação fluorescente sem correção de balanço de branco (TAYLOR *et al.*, 2015). Kelvin é uma medida que visa quantificar o tipo de iluminação em uma cena. Cenas como pôr do sol, nascer do sol possuem uma iluminação de 2000K enquanto uma iluminação fluorescente pode chegar a 4000K. O algoritmo de correção de balanço de branco não é uniforme entre diferentes fabricantes, já que cada um pode optar por realçar determinadas cores para um publico específico, sem contar que características físicas da câmera como a qualidade da lente, qualidade do sensor, tipo de sensor, podem impactar bastante na qualidade das cores obtidas, por isso a importância de um algoritmo que seja robusto à variação de cor e exposição.

2.3 Redes Neurais

Muitas das tecnologias inventadas pela humanidade foram inspiradas na natureza. Aviões inspirados em pássaros, vacinas foram inspiradas no sistema imunológico. Não foi diferente com as redes neurais artificiais, que foram criadas baseando-se no cérebro.

Redes neurais foram introduzidas inicialmente em 1943 pelo autor Walter Pitts McCulloch and Pitts (1943), nele, o autor apresenta um modelo computacional simplificado baseado em cérebros de animais para realizar cálculos computacionais complexos. Entretanto esta abordagem ganhou tração somente nas últimas décadas, com o advento do crescimento do poder computacional e da quantidade de dados disponíveis.

Antes de entrar a fundo na arquitetura das redes neurais, é interessante entender o funcionamento do neurônio (Figura 5) .

O neurônio é composto pelo corpo celular (região onde se encontra o núcleo), dendritos (ramificações provenientes do corpo celular que recebem os impulsos nervosos) e axônio (prolongamento onde os sinais nervosos são levados a outros neurônios). Neurônios recebem sinais nervosos de outros neurônios chamados sinapses, e quando um neurônio recebe um número suficiente de sinais provenientes de outros neurônios, seu próprio sinal

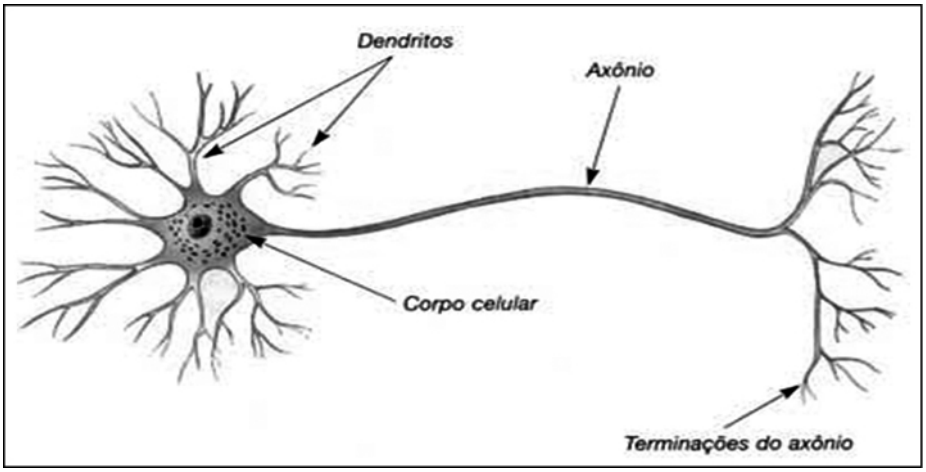

Figura 5 – Neurônio.

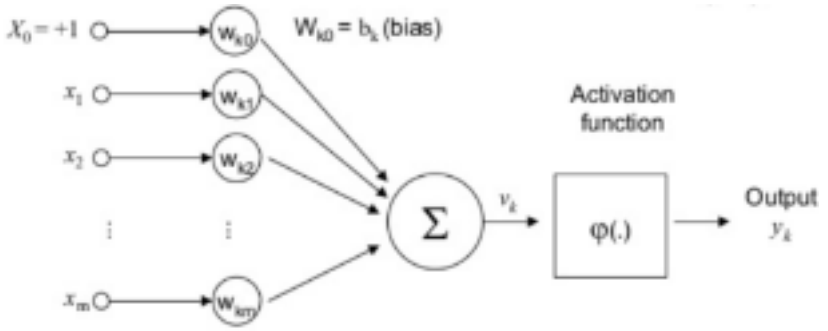

Figura 6 – Unidade Lógica de Ativação (ULA).

é ativado e enviado a novos neurônios (LAWRENCE, 1993). Os neurônios se comportam de forma demasiadamente simples, mas seu conjunto de bilhões de neurônios em conjunto formam um sistema complexo e funcional.

O processamento lógico por trás de uma rede neural artificial segue o mesmo padrão, sendo o seu componente mais básico chamado perceptron. O perceptron foi inicialmente discutido por Frank Rosenblatt (ROSENBLATT, 1958) e é basicamente composto de várias unidades lógicas de ativação (ULA), que pode ser descrita pela Figura 6. Podemos ainda ter um perceptron com um único neurônio, neste caso sua arquitetura seria muito similar a de uma unidade lógica aritmética, entretanto geralmente perceptrons possuem várias ULAs conectadas entre si, para que se possa ter um poder de processamento maior. Geralmente, todas as entradas de cada ULA são as mesmas de seu respectivo perceptron, onde cada entrada está associada com um peso. A unidade computa a soma ponderada de suas entradas e depois aplica um função de ativação. Uma das possíveis funções de ativação é descrita em 2.1.

$$f(x) = \begin{cases} 1 & : \ x \geq 0 \\ 0 & : \ x < 0 \end{cases} \tag{2.1}$$

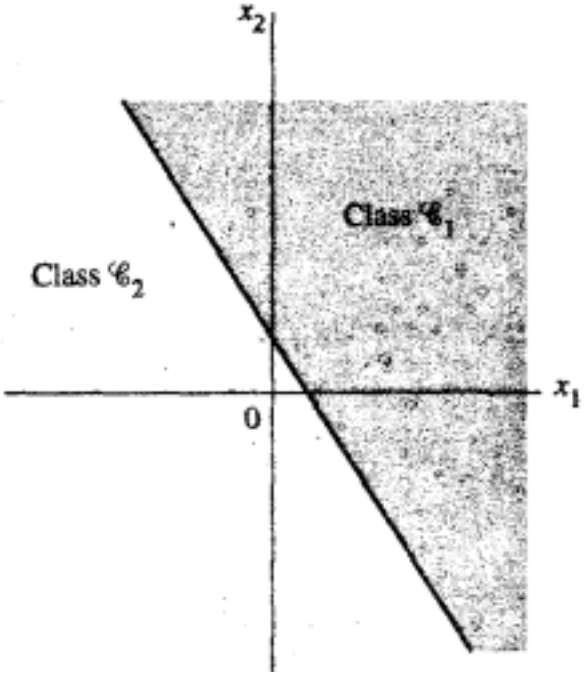

Figura 7 – Hiperplano representando a saída de um perceptron de duas classes.

ULAs podem ser usadas isoladamente para resolução de problemas simples, no entanto o perceptron possui um poder de processamento maior, por ser composto por várias ULAs onde o conjunto de entrada de cada unidade é o mesmo conjunto de entrada do perceptron de forma geral.

O Perceptron é construído em volta de um neurônio não linear, conforme o modelo de McCulloch-Pitts 2.2:

$$\sum_{1}^{m} w_i x_i + b \tag{2.2}$$

O objetivo do perceptron é classificar o conjunto de estímulos $x_1, \ldots, x_n$ aplicados externamente, à classe C_1 se a saída for y = +1 ou à classe C_2 se a saída for 1. Na forma mais simples do perceptron, há duas regiões de decisão separadas por um hiperplano definido por Figura 2.3

$$\sum_{1}^{m} w_i x_i - \theta = 0 \tag{2.3}$$

Em 1969, Marvin Minsky e Seymour Papert publicaram um estudo documentando alguns pontos fracos dos Perceptrons, como a inabilidade de resolver problemas simples como a porta lógica XOR (OU Exclusivo), o que desapontou muitos pesquisadores da época ao ponto de abandonar a frente de perceptron e dar foco a outras estratégias pra resolução de problemas de lógica (GÉRON, 2019) . Entretanto, estes problemas podem ser

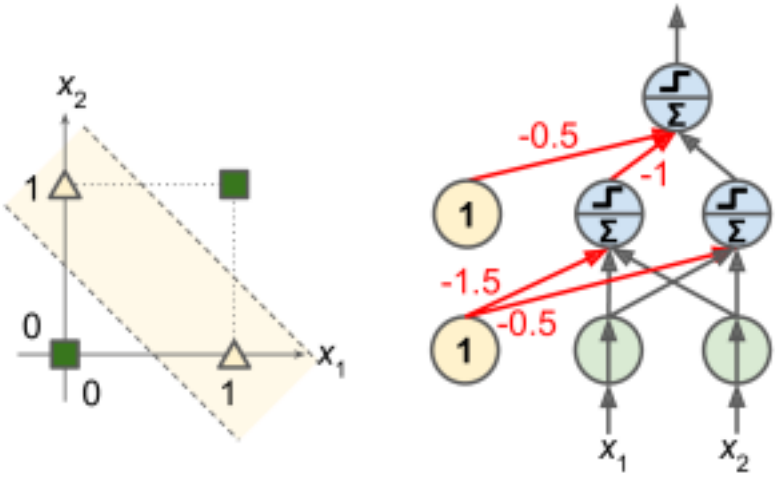

Figura 8 – Perceptron de múltiplas camadas.

resolvidos por meio da junção de vários Perceptrons em paralelo e sequência, resultando no que chamamos de *Multi Layer Perceptron* (Figura 8), sendo o percursor das redes neurais artificiais profundas. Essas redes aprendem hiperplanos separadores em múltiplas dimensões por meio do ajuste dos parâmetros de camadas Perceptron (MELLO; PONTI, 2018).

As MLP no entanto sofriam com a dificuldade de treinamento o que foi resolvido com o algoritmo Backpropagation Rumelhart, Hinton and Williams (1985). Basicamente, o algoritmo passa por todas as instâncias de treinamento e computa a saída de cada camada dado os pesos e entradas de cada perceptron, as saídas de cada camada são alimentadas para as camadas mais profundas continuamente até atingir a saída. Após isto, o algoritmo calcula o erro geral baseado em uma função de perda previamente estabelecida). Por fim, o algoritmo calcula o quanto cada neurônio e cada camada contribui para erro indo da camada de saída até a entrada (Este passo é chamado *backpropagation*). Este algoritmo é usado até hoje e é uma das razões pelo grande sucesso das redes neurais artificiais no atualmente.

2.4 Transferência de aprendizado

Caso uma rede neural esteja disponível e que cuja tarefa é similar à desejada, pode-se utilizar um técnica chamada transferência de aprendizado (*transfer learning*). Esta técnica consiste em utilizar esta rede previamente treinada com similar objetivo, mas desta vez utilizando a base dados específica para tarefa de interesse (SANTOS; THUMÉ; PONTI, 2021). Alguns autores sugerem ainda congelar os pesos da rede nas camadas mais inferiores (próximas à entrada) nas primeiras épocas, desta forma a rede não irá precisar aprender características de baixo nível que ocorrem em muitas imagens, mas apenas as estruturas de alto nível (formato da orelha em gatos por exemplo).

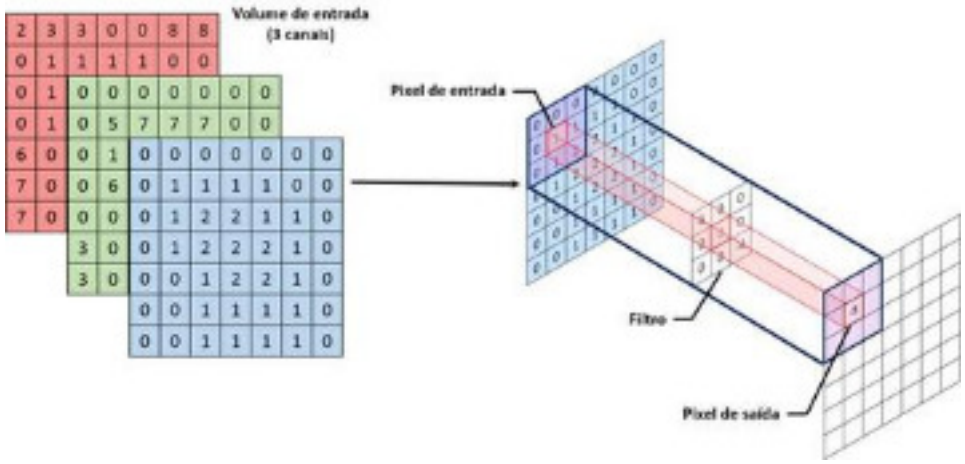

Figura 9 – Exemplo de um passo no processo de convolução.

2.5 Redes Convolucionais

A principal diferença entre as redes convolucionais e as *Multi Layer Perceptrons* é que neurônios da primeira camada convolucional não estão conectados a cada pixel da imagem de entrada. Na verdade, cada neurônio está conectado a um número limitado de neurônios, dentro do limite do filtro aplicado. Figura 9 mostra o passo considerando apenas um pixel em uma imagem aleatória.

Um neurônio localizado na linha i, coluna j de uma dada camada, estará conectado as saídas da camada anterior localizadas nas linhas i a $i + h$, e colunas j até $j + w$, onde h e w são a altura e largura do filtro do campo de visão utilizado.

Filtros utilizados na convolução também possuem pesos que serão utilizados pela rede para computar os resultados da operação. O resultado das operações em cada camada resulta em uma imagem secundária, chamada de mapa de características (*feature map*). Mapas de características podem possuir vários níveis de complexidade e uma imagem ilustrando estes mapas pode ser vista na Figura 10. Cada um destes mapas irá se especializar em reconhecer uma determinada característica na imagem, por exemplo, um filtro composto por valores nulos e uma linha vertical de valores unitários irá produzir um mapa onde a imagem terá características verticais em destaque, por isso quando uma imagem com estas características de entrada for fornecida, a rede irá ativar seus neurônios com maior facilidade. Imagens mais complexas irão requerer mapas mais complexos.

Quando há a combinação de múltiplos destes mapas, a rede neural terá um leque maior de características a procurar na imagem de entrada e consequentemente irá reconhecer com maior facilidade se determinado tipo de imagem fornecido é similar às instâncias de treinamento.

Uma última parte importante em algumas redes convolucionais é a camada de *pooling*. Esta operação pode ser entendida melhor como um método capaz de reduzir o

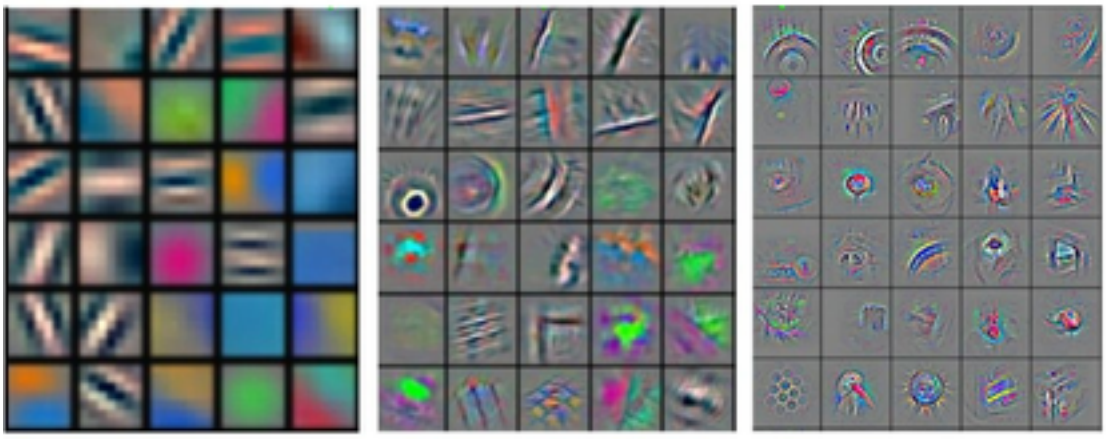

Figura 10 – Exemplo de vários mapas de características (alto nível e baixo nível).

custo computacional de todo o processamento, pois ela reduz a dimensão dos mapas de características aplicando técnicas matemáticas como média aritmética ou máximo valor dos pesos.

2.6 Arquiteturas de Redes Convolucionais

A seguir são listadas algumas das arquiteturas mais populares considerando o campo de visão computacional, redes neurais, e o fato delas terem desempenho razoável até em dispositivos de baixo poder computacional como *smartphones*. Enquanto em meados da década de 2010 era comum escolher a arquitetura mais recente, a busca por arquiteturas mais adequadas para determinadas tarefas e tipos de dados tem sido uma constante nos últimos anos (PONTI *et al.*, 2021).

2.6.1 VGGNet

VGG-Net (SIMONYAN; ZISSERMAN, 2014) foi a rede neural que ganhou segundo lugar no desafio ILSVRC 2014, cujo objetivo é classificar corretamente um conjunto extensivo de imagens. Ela é composta por uma arquitetura relativamente simples, composta por 3 camadas convolucionais seguida por uma camada pooling, que se repetem por 16 vezes até uma camada final do tipo densa.

2.6.2 Resnet

Em 2015, foi desenvolvida uma arquitetura mais complexa, chamada Resnet (HE *et al.*, 2016), que até hoje é usada como base para novas arquiteturas. Ela é composta por 152 camadas e para tanto utiliza de um recurso onde além da conexão de uma determinada camada para sua próxima camada, há também uma conexão entre esta mesma camada e a saída de camadas mais profundas na rede chamado *skip connection* (Figura 11). Uma das grandes vantagens desta rede é a possibilidade da rede aprender padrões mais gerais,

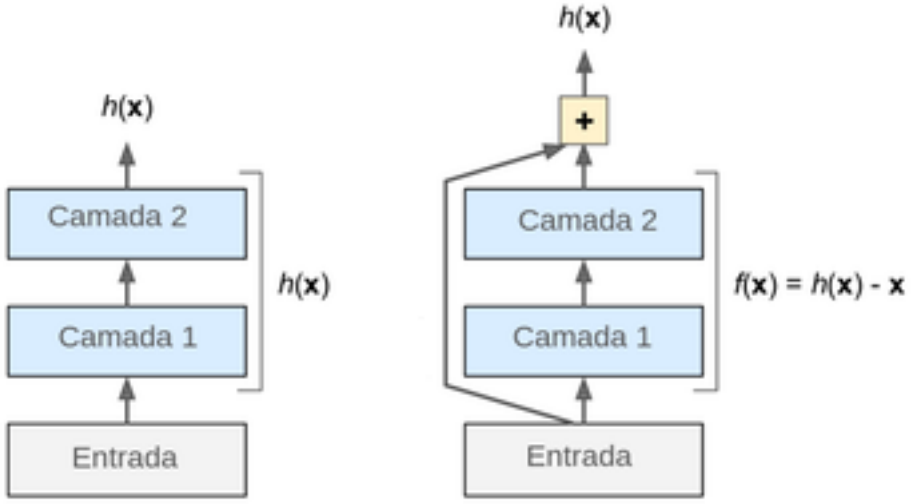

Figura 11 – *skip connection*.

já que a entrada da rede também pode ser fornecida para as camadas mais profundas. Ela também possui o tempo de treinamento reduzido devido a estas conexões, já que a rede as entradas da rede são alimentadas diretamente nas camadas mais profundas, fazendo com que o tempo de treinamento nas primeiras épocas seja mais rápido.

2.6.3 Mobilenet

A Mobilenet é um tipo de rede neural convolucional projetada especificamente para uso em aparelhos com menor poder de processamento como celulares e sistemas embarcados. Sua arquitetura é capaz de produzir um modelo com menos parâmetros (e consequentemente menor tamanho em disco) e que também possua menos operações. Comparando a arquiteturas citadas neste texto como a Resnet e VGGNet, a MobileNet e suas derivadas possuem melhor performance em dispositivos com baixo poder computacional.

A arquitetura proposta em (HOWARD *et al.*, 2017) consiste em uma rede contendo uma convolução separável em profundidade (*separable depthwise convolution*), que é na verdade uma forma fatorada de convolução, o que resulta em duas principais camadas: uma convolução tradicional fatorada em uma convolução em profundidade e uma convolução pontual. Uma imagem ilustrando o processo pode ser vista em Figura 12

O artigo mostra que a redução computacional comparando a operação proposta com uma convolução tradicional chega a ser 8 a 9 vezes menor para um filtro 3x3. A fórmula proposta pelos autores que mostra esta redução 2.4.

$$\frac{1}{N} + \frac{1}{Dk^2} \tag{2.4}$$

Onde N é o número de canais da imagem e Dk é o o tamanho do filtro utilizado (*kernel size*).

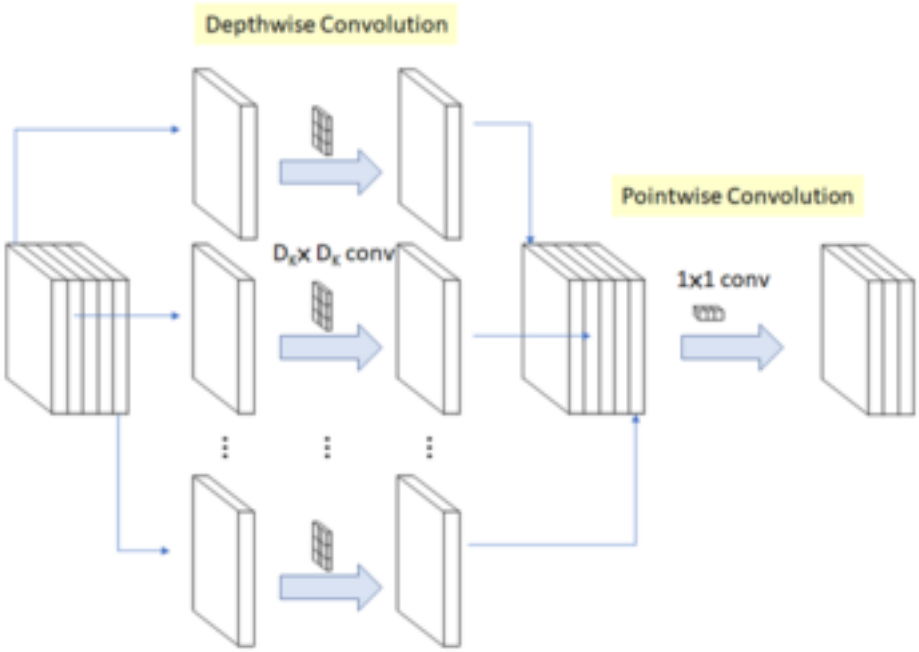

Figura 12 – Convolução utilizada na Mobilenet.
Fonte: (HOWARD *et al.*, 2017)

É importante lembrar que segundo o artigo, toda esta redução computacional pode ser alcançada com perdas baixas em precisão.

2.6.4 MobilenetV2 e MobilenetV3

Seguindo a proposta da Mobilenet, onde há a ênfase em aumentar a eficiência da rede em sistemas de baixo poder computacional, foi obtido a MobilenetV2 (SANDLER *et al.*, 2018), onde estruturas chamadas gargalo linear (*linear bottleneck*) e estrutura residual invertida (*residual inverted structure*) são introduzidas para redução ainda maior do número de parâmetros e latência. A rede Mobilenet subsequente é uma das principais redes utilizadas no âmbito de processamento em dispositivos com processamento limitado como *smartphones*. A MobilenetV3 (HOWARD *et al.*, 2019), diferente das estruturas anteriores, onde foram desenhadas fielmente à concepção de seus pesquisadores, foi obtida por meio de AutoML, uma técnica que pesquisa diferentes arquiteturas amigáveis à tarefas de reconhecimento de imagens em poder de processamento limitado. Isto foi possível utilizando duas técnicas MnasNet (TAN *et al.*, 2019), que usa aprendizado por reforço procurando uma arquitetura variando alguns parâmetros dentro de seu grau de liberdade, seguida pela NetAdapt (YANG *et al.*, 2018), que consiste em uma técnica complementar com o objetivo de reduzir camadas e neurônios sub utilizados na rede. Há ainda pequenas mudanças em relação à arquitetura inicial como a utilização de diferentes funções de ativação (porém com comportamento similar) e modificações nas camadas finais das arquiteturas anteriores.

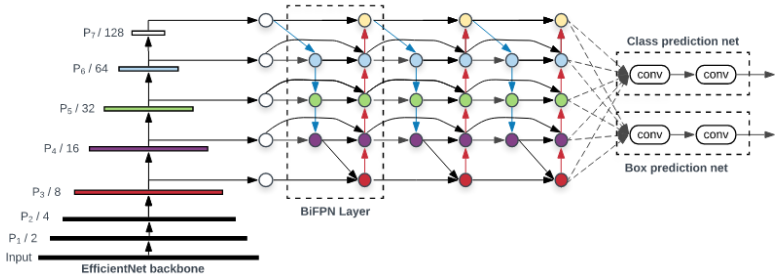

Figura 13 – Arquitetura EfficentDet.
Fonte: (TAN; PANG; LE, 2020)

O artigo demonstra que a MobilenetV3 pode ser até 27% mais rápida que a MobilenetV2 e ainda obter resultados similares em relação a precisão em tarefas de classificação, segmentação ou detecção de objetos.

Mobilenet se tornou uma das redes mais populares no desenvolvimento de tarefas de visão computacional usando redes neurais em dispositivos com poder de processamento limitado. MobilenetV3.

2.6.5 EfficientDet

EfficientDet é um tipo de modelo de detecção de objetos que utiliza vários ajustes de otimização como o uso de um BiFPN *Weighted Bi-directional Feature Pyramid Network*, e um método de dimensionamento composto que dimensiona uniformemente a resolução, profundidade e largura para todos os *backbones*, redes de recursos e previsão de caixa/classe redes ao mesmo tempo.

A figura 13 ilustra esta arquitetura, que faz uso de EfficientNet como base, BiFPN como rede auxiliar de características.

2.7 Métricas de detecção de objetos

2.7.1 IOU

Uma das medidas mais comuns entre tarefas de detecção de objetos é a IOU (*Intersection Over Union*) (REZATOFIGHI *et al.*, 2019). Dada a região onde a rede obteve um predição de onde o objeto está localizado e também a região onde o objeto de fato está na imagem (Geralmente obtido por meio de interação humana), é possível calcular a área de intersecção, que seria nada mais do que as áreas em comum entre as duas caixas delimitadoras, como também a área de união entre estas caixas. Idealmente, deseja-se que ambas medidas sejam próximas, o que resulta em um IOU de valor igual a um.

2.7.2 mAP

mAP (*mean Average Precision*) é uma métrica que visa obter uma visão abrangente do comportamento do modelo em diferentes condições de aceitação e pode ser complementada dependendo das particularidades de cada projeto.

Existe o caso em que a região do objeto é detectada corretamente, mas a classe do objeto não. Nestes casos uma forma de atacar este problema seria determinar que o modelo só acertou aquele objeto caso a classificação de sua classe seja correta e a medida IOU atinga pelo menos 70%. Neste caso a medida poderia ser denotada como AP_{70}. Alguns pesquisadores ainda optam por calcular esta medida com diferentes níveis de aceitação para a medida IOU, por exemplo de 5% a 95% e após isto calcular a média entre todas estas medidas, resultando na métrica mAP.

2.8 Métricas de classificação

Uma métrica simples de classificação é simplesmente o número de instâncias em que o modelo classificou corretamente sobre o número total de instâncias. Apesar de ser útil muitas vezes ela não é suficiente para uma completa avaliação do comportamento do modelo. Dentre uma das métricas mais populares estão a precisão (precision) e a revocação (recall).

A precisão consiste em na porcentagem de exemplos positivos classificados como positivos sobre o total de instâncias classificadas como positivos 2.8.

$$precisão = \frac{VP}{VP + FN}$$

VP: Verdadeiro Positivo, FN: Falso Positivo (2.5)

A revocação pode ser entendida como a porcentagem de exemplos classificados como positivos que são realmente positivos 2.8. A Figura 14 ilustra visualmente a diferença entre precisão e revocação.

$$revocação = \frac{VP}{VP + FP}$$

VP: Verdadeiro Positivo, FN: Falso Positivo (2.6)

A medida F consiste na combinação das métricas anteriores por meio de uma média harmônica ponderada 2.7

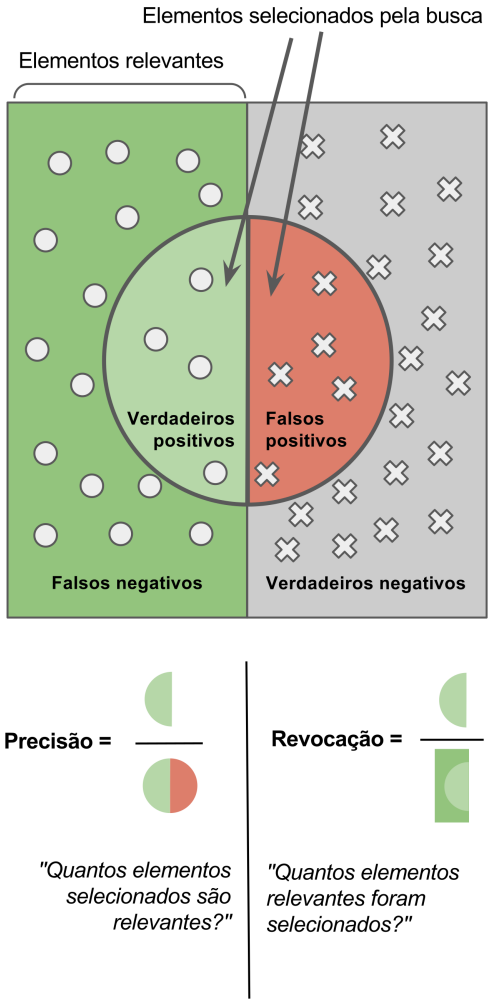

Figura 14 – Precisão versus Revocação.
Fonte: (PáDUA, 2021)

$$F = \frac{(1+\alpha) \times precisão \times revocação}{\alpha \times (precisão + revocação)} \tag{2.7}$$

Quando precisão e revocação possuem o mesmo peso, obtém-se a métrica F1 2.8:

$$F_1 = \frac{2 \times precisão \times revocação}{precisão + revocação} \tag{2.8}$$

A utilização das métricas precisão e revocação são recomendadas para análise da ocorrência de falsos positivos e falsos negativos, respectivamente, já o F1 consiste na combinação das duas métricas descritas de forma a mensurar a performance de um classificador em uma única métrica.

A métrica F1 favorece classificadores que possuem métricas similares entre precisão e revocação, dependendo da aplicação pode não ser a métrica mais recomendada. Por exemplo, se um classificador é criado com o intuito de classificar positivamente imagens

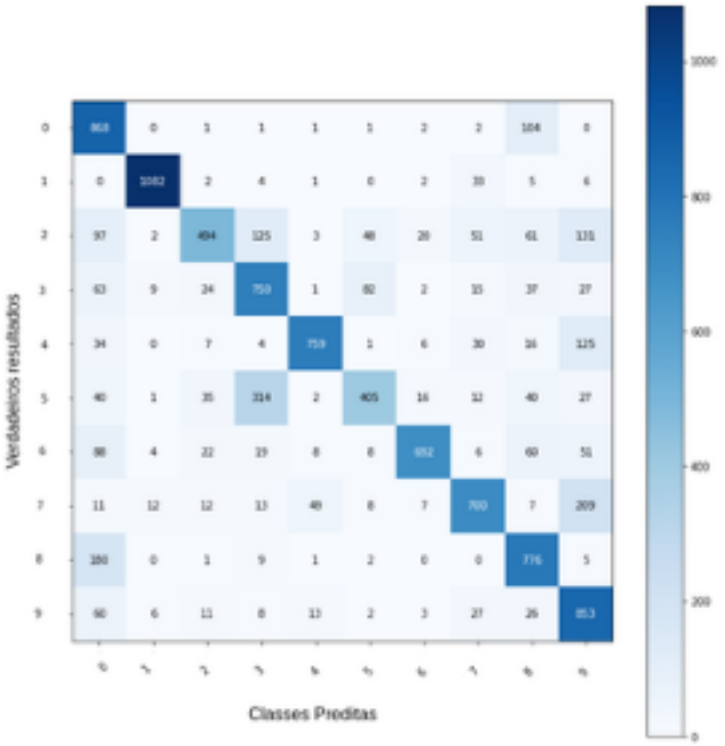

Figura 15 – Matriz de confusão para um projeto de reconhecimento de dígitos alfanuméricos.

de peças sem defeitos, é interessante para a empresa um classificador que rejeita algumas amostras boas (baixa revocação) e mantém apenas as amostras seguras. Por outro lado se o seu classificador é criado para detectar imagens de possíveis doenças em seres humanos, o responsável pode decidir em priorizar a revocação, o que no caso faria com o que o classificador emita um número elevado de falsos positivos, mas que detecta mais facilmente os casos verdadeiramente positivos.

Ao decorrer do desenvolvimento de uma solução, um pesquisador pode querer estudar melhor casos em que somente uma das classes apresenta baixa acurácia. Uma forma simples e visualmente agradável para estes casos é a matriz de confusão. As linhas da matriz consiste nos resultados verdadeiros para cada instância enquanto as colunas representam as predições do modelo.

A Figura 15 ilustra um exemplo de uma rede treinada para identificar dígitos de 0 a 9 escritos à mão, podemos ver na linha do dígito 5, que há um número grande de valores corretamente classificados como 5, mas também há um número grande de instâncias desse dígito classificadas como 3. Este tipo de informação é importante e facilmente identificada nesta métrica, fato que não seria possível somente com as métricas F1 ou precisão.

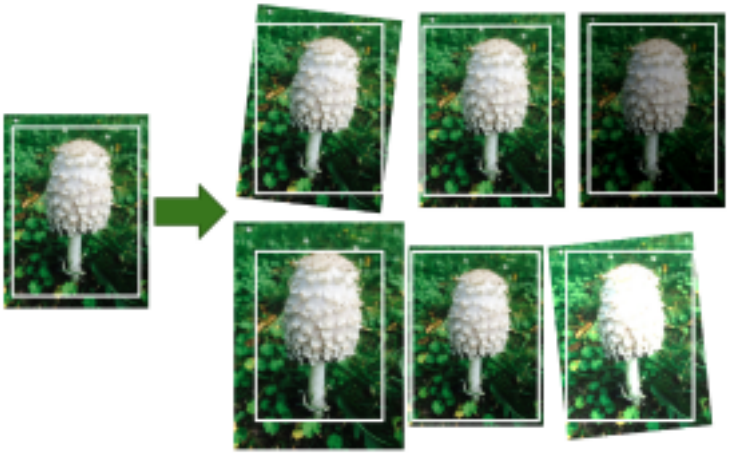

Figura 16 – Aumento de dados em ação.

2.9 Aumento de dados

Dentro do escopo de imagens, o aumento de dados (*data argumentation*) é uma técnica capaz de aumentar o tamanho dos dados de treinamento por meio da geração de novas imagens que são variantes das imagens encontradas em treinamento (GÉRON, 2019). Esta técnica é capaz de reduzir o *overfitting* e pode ser usada como regularização.

Pode-se por exemplo modificar ligeiramente uma imagem de treinamento alterando alguns aspectos da imagem como a cor, brilho, contraste rotação entre outros aspectos relacionados à imagem. Isto força o modelo a ser tolerante a variações destes aspectos como posição orientação, condições de iluminação entre outros. Para aplicações em campo como a tratada neste projeto, torna-se uma técnica importante para trazer maior robustez a condições desafiadoras. Pode-se ainda gerar novas instâncias utilizando combinações de aumentação de diferentes aspectos, o que pode aumentar a quantidade de instâncias de treinamento. A Figura 16 ilustra um caso onde uma base de dados foi aumentada com a posse de apenas uma imagem utilizando técnicas de redução e aumento de brilho, como também rotação e realização de recorte na região central da imagem.

2.10 Base de dados

Apesar de haver dados específicos à imagens de doenças no plantio de batata, geralmente estas bases são pequenas. Algumas estratégias serão utilizadas no desenvolvimento deste projeto, entre elas utilizar também dados de imagens de doenças não relacionadas ao plantio da batata como forma de transferência de aprendizado, a seguir serão abordados amostras de dados encontradas na literatura, que são pertinentes ao objetivo deste projeto.

2.10.1 Tiago de Miranda Leite

Este conjunto de dados (LEITE, 2021) consiste em 1048 arquivos de imagens de doenças em folhas de cidreira (árvore produtora de maçã). Consiste ainda de 11,328 caixas delimitadoras. A base de dados total contém 1820 imagens, entre imagens de folhas saudáveis, e doenças ferrugem e sarna, cuja característica visual é similar à algumas doenças encontradas nas folhas de batata.

2.10.2 *Plant Village*

Esta é uma iniciativa onde os autores co-criaram uma base de dados (HUGHES; SALATHÉ *et al.*, 2015) contendo imagens de doenças em folhas de vários tipos de plantação. Os registros de dados contêm 54.309 imagens, entre elas imagens de 14 espécies de culturas: maçã, mirtilo, cereja, milho, uva, laranja, pêssego, pimentão, batata, framboesa, soja, abóbora, morango, tomate. Contém imagens de 17 doenças fúngicas, 4 doenças bacterianas, 2 provenientes do mofo, 2 doenças virais e 1 doença causada por um ácaro. 12 destas espécies de culturas também têm imagens de folhas saudáveis que não são visivelmente afetadas por uma doença.

Esta base de dados torna-se interessante pelo número elevado de amostras de imagens considerando o problema em doenças em diferentes cultivos agrícola. Torna-se mais interessante por possuir imagens de duas doenças no plantio de batata, uma proveniente do fungo Alternaria-solani, que produz a doença conhecida como **pinta preta**, e outra proveniente de mofo produzida por Phytophthora-Infestans, que gera a doença conhecida como **requeima da batata**. Estão disponíveis 1000 amostras para cada uma das doenças e 152 imagens de folhas saudáveis.

Esta base de dados não possui dados para localização dos pontos da doença, ou seja, não possui informações de caixas delimitadoras.

2.10.3 Base de dados do Kaggle para plantação de mandioca

A plataforma Kaggle disponibilizou uma competição para correta classificação de diferentes imagens de folhas de mandioca [1] , cultivo popular na África sub-saariana que pode sobreviver até mesmo em condições inóspitas. É composto por mais de 20.000 imagens, dentre elas 2576 de folhas saudáveis e o restante dividido entre 4 diferentes tipos de doenças.

[1] https://www.kaggle.com/c/cassava-leaf-disease-classification

3 MATERIAIS E MÉTODOS

Neste tópico são detalhados os métodos utilizados para o desenvolvimento do sistema. Dentre os principais aspectos do projeto descrito neste tópico estão a forma de treinamento do classificador, a forma de treinamento do detector e como o aplicativo processa as informações para gerar os resultados ao usuário de forma efetiva.

3.1 Materiais

A seguir são apresentados os materiais utilizados para o desenvolvimento do projeto. Para o treinamento dos modelos apresentados neste capítulo, foi utilizado a plataforma Google Colab [1], plataforma que permite a utilização de recursos de GPU e memória. Para a implementação deste projeto, foi utilizado GPU de 25GB e espaço de memória de 150GB. O Colab também disponibiliza ambientes com as principais bibliotecas de aprendizado de máquina instaladas, como o Tensorflow[2], que foi utilizado neste projeto.

Para o desenvolvimento do aplicativo, foi utilizado o software Android Studio [3] e para testes foi utilizado um *smartphone* modelo *Motorola Edge 20 Pro*.

3.2 Classificador

Dentro do contexto de imagens de plantação de batata, o objetivo do classificador é dado uma imagem de entrada, classificar qual tipo de doença existe naquela figura. Um banco de dados que possui uma extensa quantidade de imagens deste legume é o *Plant Village*.

3.2.1 Tratamento dos dados

Tabela 1 – Banco de dados inicial.

Classe	Número de instâncias
Saudável	152
Pinta Preta	1000
Requeima	1000

Tabela 1 retrata a quantidade de amostras para cada classe relevante para o nosso caso, chamaremos ele de banco inicial, de acordo com ela, o número de classes que mostram classes de folhas saudáveis é quase 10 vezes menor que o número de classes de doenças.

[1] https://research.google.com/colaboratory/faq.html
[2] https://www.tensorflow.org/
[3] https://developer.android.com/studio

Figura 17 – Comparação entre folhas saudáveis de batata e soja.

Para contornar este problema, foram adicionados 5090 imagens de folhas saudáveis de soja para tornar a quantidade de amostras saudáveis e com doença mais balanceada.

A Figura 17 retrata a semelhança visual entre folhas saudáveis de soja e batata. Podemos usar isto para nossa vantagem para construir um banco de dados aumentado e mais equilibrado.

Outro problema que pode ocorrer é o caso de doenças desconhecidas. Para deixar o classificador robusto a doenças que não estejam no banco de dados utilizado foi adicionado algumas imagens de doenças de outras plantações. Entretanto, para isto é recomendado que sejam adicionadas um número de amostras semelhantes a quantidade de amostras das classes relacionadas à batata, portanto foram adicionadas 1000 imagens de tais doenças desconhecidas. Foi utilizado o banco de dados de plantio de mandioca [4] para este caso. Nesta etapa todas as classes do banco de dados de plantio de mandioca foram separadas entre saudáveis e não saudáveis. Foram então selecionadas aleatoriamente 1000 amostras de imagens com doenças no plantio de mandioca e então estas foram concatenadas ao banco de dados utilizado neste projeto.

Por fim, obtivemos um banco de dados de 4 classes: Folhas saudáveis, Pinta Preta, Requeima e Doenças desconhecidas. resultando em um banco com estrutura representada na Tabela 2:

Os números obtidos na tabela 2 foram obtidos conforme descrito a seguir. A classe saudável possui 152 folhas de batata saudáveis e 5090 folhas de soja saudáveis, resultando em 5242 imagens no total. A classe Pinta Preta possui 1000 imagens de folhas de batata com esta doença. A classe Requeima possui 1000 imagens de folhas de batata com esta doença. a classe Desconhecida possui 1000 imagens de folhas de mandioca com doenças.

[4] https://www.kaggle.com/c/cassava-leaf-disease-classification

Tabela 2 – Banco de dados final.

Classe	Número de instâncias
Saudável	5242
Pinta Preta	1000
Requeima	1000
Desconhecida	1000

Os índices das categorias foram modificados para estarem entre os valores 0 e 3. Por fim, o banco de dados final foi separado entre treinamento (80%), validação (10%) e teste (10%).

3.2.2 Aumento de dados

Diversas técnicas foram utilizados no pré-processamento das imagens para tornar o modelo robusto à variações de condições ambientais, o que é muito susceptível a ocorrer em no local do plantio.

Seguem abaixo a lista de operações feitas nas instâncias de treinamento (todas as operações foram feitas randomicamente):

1. alteração do brilho da imagem (até 20%).
2. alteração da saturação da imagem (até 10%).
3. Espelhamento da imagem na horizontal.
4. Espelhamento da imagem na vertical.
5. alteração da qualidade da imagem (entre 75 e 95).
6. Rotação da imagem em multiplosde 90º.
7. Redimensionamento da imagem para o tamanho do modelo.

Para as instâncias de validação e testes somente foi realizado o redimensionamento da imagem.

3.2.3 Modelo

Foram feitos experimentos com 3 modelos na técnica de transferência de aprendizado: MobilenetV2 , MobilenetV3Small e EfficientNetV2. Em todos os casos, além das modificações para adequar a imagem ao tamanho esperado pelo modelo e normalização, foi adicionado uma camada de *Pooling*(*GlobalAveragePooling*) e uma camada densa com ativação *Softmax* de 4 neurônios (devido a 4 classes que queremos classificar). Em todos os casos, foram obtidos modelos pré-treinados com o banco de imagens da *imagenet*, para

posterior transferência de aprendizagem. Todos os pesos dos modelos pré-treinados foram congelados, fazendo com que os parâmetros treináveis estejam apenas nas conexões da última e penúltima camadas.

Os hiper parâmetros do modelo foram escolhidos de acordo com recomendações em casos de modelos de classificação abordados por (GÉRON, 2019). A seguir seguem os hiper parâmetros utilizados para treinamento dos modelos:

1. Função de Perda: *Categorical Cross Entropy*. Esta é a perda padrão em casos de classificação.
2. Número de épocas: 100.
3. Optimizador: Nadam, este *optmizador* costuma ajudar o modelo a convergir mais rapidamente
4. Estratégia de interrupção precoce.
5. Paciência: 10. Caso as métricas de avaliação não estejam melhorando por 10 épocas, o treinamento é interrompido.

3.3 Detector

Outra importante tarefa no auxílio ao usuário, é a detecção de possíveis pontos de atenção nas folhas da plantação para averiguação. Neste contexto treinamos um detector de doenças que irá informar ao usuário a localização destes pontos.

3.3.1 Tratamento dos dados

Infelizmente, não foi encontrado nenhum banco de dados com caixas delimitadoras. No entanto, (LEITE, 2021) propõe um banco de dados que consiste 1820 imagens de folhas de plantação de cidreira, com imagens de folhas saudáveis e doenças ferrugem e sarna, cuja característica visual é similar à algumas doenças encontradas nas folhas de batata. Embora a utilização deste banco de dados dificulte a obtenção de uma boa taxa de acerto devido a diferenças das plantações, ainda assim é possível obter um detector aceitável utilizando estes dados.

Para esta tarefa, não foi feito nenhum tipo de aumento de dados, sendo que o único processamento de imagem feito foi o redimensionamento das imagens para adequar a entrada do modelo estabelecida. Além disto, o banco de dados foi separado em imagens de treino (80%), validação (10%) e testes (10%).

Foi necessário o redimensionamento das imagens para adequação dos modelo utilizado, o tamanho reduzido dos modelos focados em dispositivos de baixo poder computacional ajuda a reduzir o tempo de inferência e ao mesmo tempo evitar processamentos

de imagens adicionais na aplicação. Como algumas imagens do banco de dados tinham proporção diferente do tamanho de entrada escolhido, as caixas delimitadoras foram redimensionadas para se adequar às novas condições.

3.3.2 Modelo

O modelo utilizado faz uso de uma arquitetura EficientD0, previamente treinados para detecção de objetos no banco de dados COCO (LIN *et al.*, 2014), do ano de 2017. Os hiper parâmetros do modelo foram escolhidos de forma que haja convergência para os novos dados fornecidos.

1. Função de Perda: *Focal Loss*. Função de perda utilizada pelos autores da publicação.
2. Número de épocas: 500.
3. Taxa de aprendizado: 0.01.
4. Optimizador: Nadam, este *optmizador* costuma ajudar o modelo a convergir mais rapidamente
5. Estratégia de interrupção precoce.
6. Paciência: 20. Caso as métricas de avaliação não estejam melhorando por 20 épocas, o treinamento é interrompido.

A entrada do modelo de detecção é simplesmente a imagem de entrada convertida para o formato esperado pele Tensorflow. A saída do modelo consiste em 4 valores:

1. Caixas delimitadoras: Uma lista representando cada caixa detectada e suas respectivas coordenadas em porcentagem da image(4 valores)
2. Uma lista com as classes detectadas.
3. Uma lista com a confiança das detecções.
4. Um valor único, representando o número de detecções.

Muitas das caixas delimitadoras geradas pela saída do modelo podem estar sobrepostas uma a outra ou terem baixa confiança nas detecções.

Para eliminarmos caixas delimitadoras sobrepostas e ao mesmo tempo eliminar resultados com baixa confiança, utilizamos o algoritmo de visão computacional *Non-Maximum Suppression* (NMS). Consideramos válidas apenas caixas delimitadoras com confiança acima de 0,5. Após isto teremos um número limitado de caixas delimitadoras para mostrar ao usuário.

3.4 Aplicativo

Como resultado adicional, utilizamos a biblioteca do tensorflow para converter os modelos treinados nas seções anteriores para um formato que possa ser utilizado em aplicativos para celulares ou até mesmo em sistemas embarcados como *Drones*. O objetivo principal da implementação desse aplicativo é demonstrar a capacidade de utilização em dispositivos móveis ou sistemas embarcados. No contexto de modelos leves como os que são estudados nesta seção, o desempenho em sistemas embarcados pode ser ainda melhor que alguns celulares.

Conforme será possível ver nos resultados, o tempo de execução do classificador é bem mais rápido se comparado a detector de objetos, sendo possível fazer inferências a cada quadro disponibilizado pela câmera sem perda na taxa de quadros observada pelo usuário. Dado esta condição, elabora-se uma lógica onde os quadros obtidos pela câmera são enviados ao classificador, e caso sejam classificadas como de alguma doença, estas são adicionadas à lista a ser processada pelo detector posteriormente. As imagens de doenças são armazenadas em intervalos de 5 segundos, onde somente a imagem com classificação de maior confiança pelo classificador é armazenada.

Outro ponto importante, é que devido à natureza do projeto, podem haver balanços no dispositivo, alterações de brilho, foco e cor que podem fazer com que as imagens disponibilizadas tenham certa variância entre um quadro e outro, fazendo com que a classificação seja instável. Para evitar isso, faz-se uso de uma lista circular que armazena as últimas 20 imagens e então verifica-se qual a classe predominante nestes 20 quadros. Isto adiciona maior estabilidade ao sistema e maior precisão na classificação.

Posteriormente, o detector irá ser aplicado nas imagens armazenadas em plano de fundo. Ao final do processamento, obtém-se uma lista de imagens classificadas como doentes, bem como as caixas delimitadoras impressas com a ajuda do detector.

O detector irá detectar possíveis locais de atenção nas imagens e os resultados das predições serão desenhados na tela em forma de caixas sinalizadoras. Devido ao tempo de execução do detector, estas caixas podem estar alguns mili segundos defasadas da imagem atual, mas tempo não alto o suficiente para impactar a experiência do usuário.

Além disto, para que o usuário possa visualizar as imagens que requerem atenção com mais calma após a coleta de dados, o próprio aplicativo salva algumas destas imagens, onde é possível visualizar as imagens de atenção já com informações de classificação e detecção em formato de galeria de imagens, sendo possível até a aplicação de *zoom* para visualização com maior detalhe.

4 RESULTADOS E DISCUSSÕES

Este tópico apresenta as características técnicas e resultados obtidos com o desenvolvimento do sistema de reconhecimento e detecção de doenças. Foram verificados aspectos como tempo de inferência, consumo de memória, e avaliação da qualidade da classificação/detecção. Por fim, algumas ilustrações do aplicativo e suas funcionalidades são descritas.

4.1 Classificador

O objetivo desta seção é comparar as métricas de qualidade de classificação das classes disponíveis no banco de dados (Saudável, Pinta Preta, Requeima, Desconhecida), tempo de inferência e consumo de memória para diferentes modelos testados. Foram feitos experimentos com 3 modelos na técnica de transferência de aprendizado: MobilenetV2 , MobilenetV3Small e EfficientNetV2. A tabela 3 mostra os resultados obtidos

Tabela 3 – Métricas obtidas para o classificador em diferentes arquiteturas.

Modelo	F1 (Macro)	Inicialização	Tempo de inferência	Consumo de memória
EfficientNetV2	0.962	15,77 ms	212,9 ms	13MB
MobilenetV2	0.955	1,35s	4,92 ms	51,73MB
MobilenetV3Small	0.974	1,59s	5,34 ms	46,80MB

É notável que a EfficientNetV2 possui um tempo de inferência maior e um menor consumo de memória. Isto acontece pois algumas operações utilizadas por esse modelo não são compatíveis com a delegação de GPU fornecida pelo Tensorflow Lite, não sendo possível utilizar este modelo com o benefício de processamento em GPU. Para a MobilenetV2 e MobilenetV2Small, foi possível utilizar o benefício de processamento em GPU.

Também nota-se que a MobilenetV3Small possui maior taxa de acerto melhor e um tempo de inferência similar ao obtido na MobilenetV2, além de um menor consumo de memória que a MobilenetV2. Portanto os resultados mais detalhados a seguir são referentes a este modelo.

4.1.1 Matriz de confusão

A matriz de confusão visa visualizar de forma mais agradável e incisiva os acertos e erros para cada uma das classes utilizadas. A matriz de confusão referente aos resultados deste projeto é ilustrada na Figura 18.

Observa-se que quando o classificador recebe imagens referentes ás classes de doenças de batata aqui estudadas (Pinta preta e requeima), o classificador obteve uma

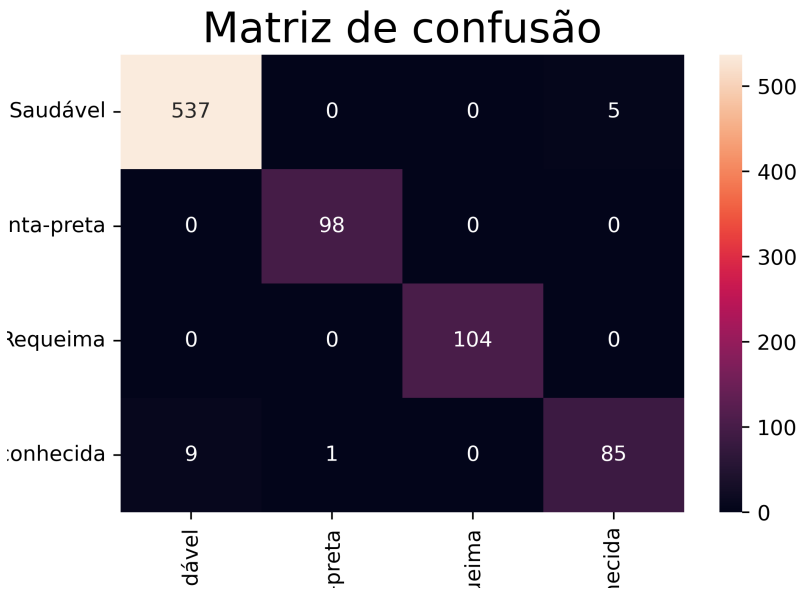

Figura 18 – Matriz de confusão do classificador utilizado.

precisão de 100%, isto significa que em todas as imagens de pinta preta e requeima, o classificador acertou a classe corretamente. A classe saudável também foi bem classificada, classificando erroneamente 5 imagens de folhas saudáveis como doença desconhecida. A classe de doenças desconhecida obteve a menor precisão, mas ainda assim pode-se considerar uma alta precisão, já que 89% das imagens desta classe foram classificadas corretamente. Um ponto a se ficar atento é justamente as 9 imagens de doença desconhecida que o classificador classificou com folhas saudáveis. Idealmente, não deseja-se que nenhuma imagem de folha doente seja classificada como saudável, entretanto este resultado significa uma revocação de 0.983, que ainda pode ser considerada alta.

4.1.2 Ilustrações utilizando o aplicativo

A cada quadro ou imagem disponibilizada pela câmera, pode-se aplicar o processo de classificação, que conforme visto na tabela 3 é rápida o suficiente para não impactar a experiência do usuário. A Figura 19 mostra como o aplicativo informa ao usuário que uma classificação de doença ocorreu.

4.2 Detecção

Nesta seção são apresentados os resultados para o detector de doenças referentes à métricas de acerto, tempo de execução no dispositivo, consumo de memória e considerações a respeito deste projeto.

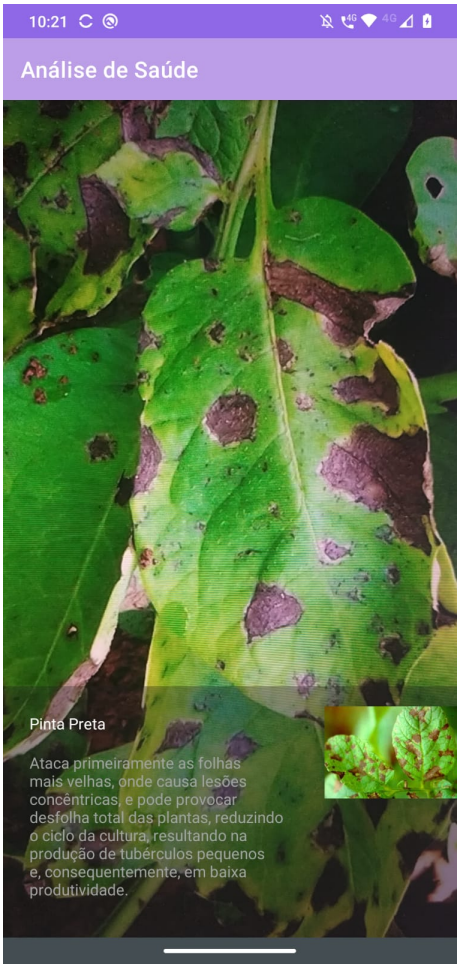

Figura 19 – Classificação de doenças utilizando o aplicativo

4.2.1 Métricas de acerto

Inicialmente, foram feitos testes com duas arquiteturas, a Resnet e a EffcientDet. A Tabela 4 mostra os resultados para as métricas de acerto. É importante notar que a detecção é um problema mais difícil do que a classificação global da imagem, e isso se reflete em métricas mais baixas na avaliação da detecção como IoU e mAP.

Tabela 4 – Métricas obtidas pelo detector de objetos.

Modelo	Precisão média (IoU=0.5)	mAP	Tempo de Inferência	Consumo de memória
Resnet	0.43	0.45	1,05s	448MB
EfficientDet-D0	0.66	0.69	97ms	57,54MB

É notável que o modelo EfficientDet-D0 foi melhor em praticamente todos os requisitos. Os resultados mais detalhados que seguem adiante são referentes a este modelo.

Além disso, verificou-se como os valores de mAP variam quando diferentes valores de limiar são definidos para IoU. Estabelecendo-se 10 valores para IoU no intervalo fechado

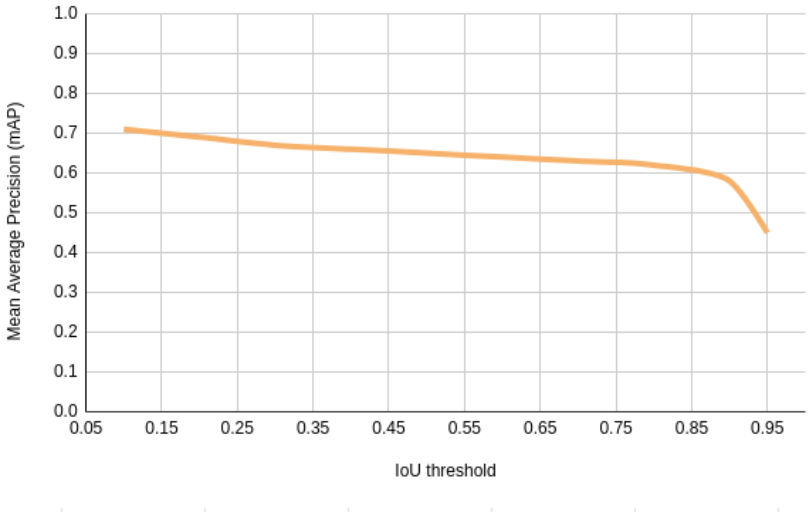

Figura 20 – Valores de mAP para diferentes limiares de IoU, variando de 0.5 a 0.95.

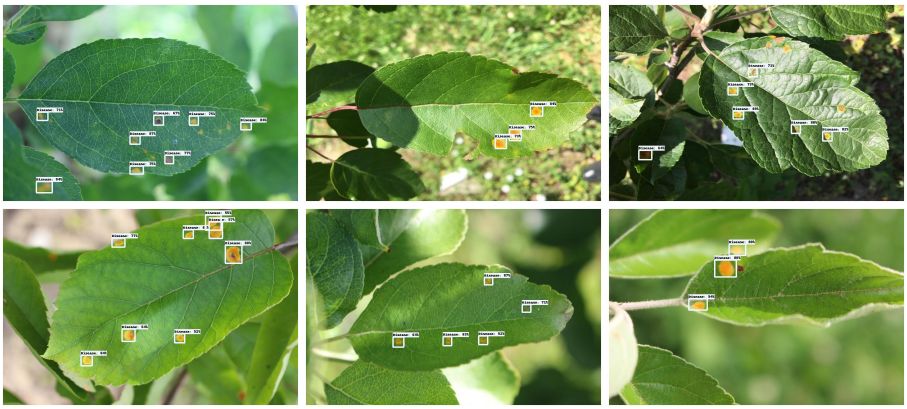

Figura 21 – Resultado da detecção no banco de dados original.

de 0.5 a 0.95, com incrementos de 0.05, os correspondentes valores de mAP são exibidos na Figura 20 mostra a precisão para diferentes *thresholds*.

É importante salientar que a medida 0.69 mAP pode ser considerada boa em um detector de objetos. Modelos estado da arte quando avaliados no banco de dados COCO (LIN *et al.*, 2014), banco de dados mais complexo com 328.000 imagens e 80 categorias, possuem semelhante métrica como 0.64 Yang *et al.* (2021) e 0.621 Dai *et al.* (2021).

4.2.2 Detecção no banco de dados original

Inicialmente, a verificação visual do detector de doenças foi feita em imagens do banco de dados de teste que foram separados para teste. A Figura 21 ilustra alguns destes casos. Pode-se notar que as detecções estão situadas perto das regiões de doenças. Nota-se também que em alguns casos o modelo não é capaz de inferir todas as caixas delimitadoras presentes ou as vezes pode ocorrer imprecisões na localização das caixas.

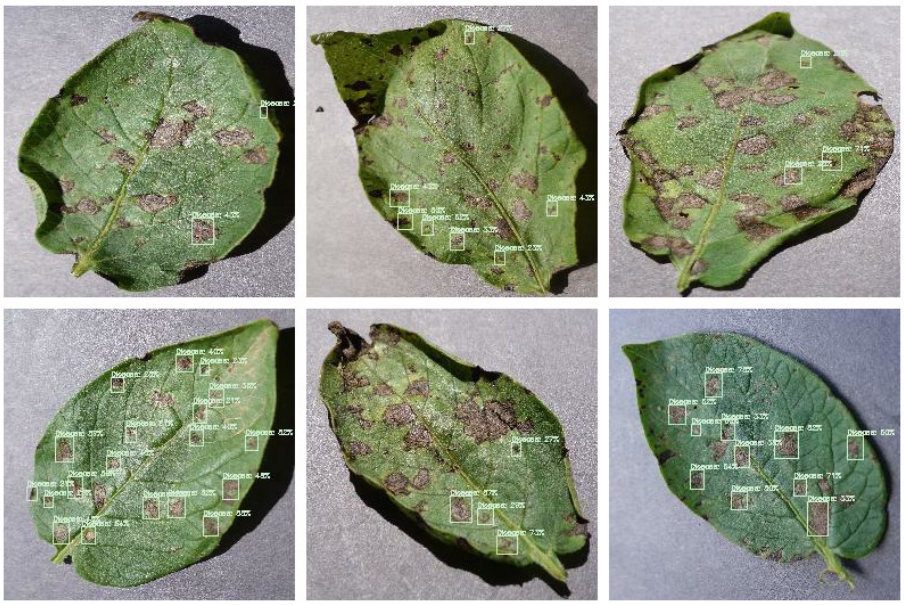

Figura 22 – Resultado da detecção em imagens de batata (Pinta preta).

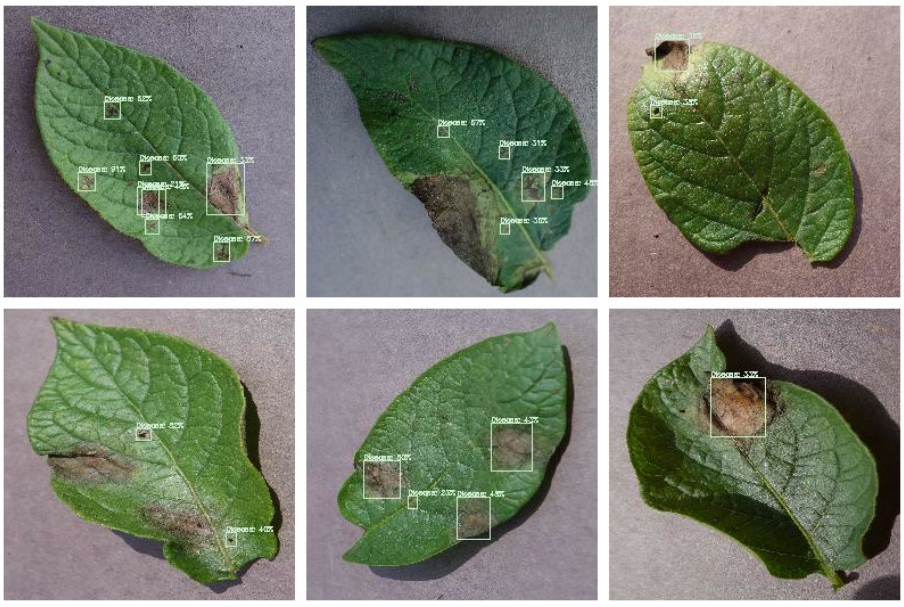

Figura 23 – Resultado da detecção em imagens de batata (Requeima).

4.2.3 Detecção em imagens de batata

Ao final, é verificado visualmente a taxa de acerto em imagens de batata. Como não há banco de dados relacionado à tarefa de detecção de objetos no plantio de batata especificamente, foi feita uma verificação visual dos resultados. As Figuras 22 e 23 mostram alguns destes resultados.

Nota-se que o modelo tem certa dificuldade em detecções da doença pinta preta. Isto ocorre pois as classes de doenças utilizadas no banco de dados disponível não tinham alta similaridade visual no quesito cor (Na pinta preta a cor predominantemente é

preta enquanto no banco de dados utilizado a cor predominante das doenças é amarela). Entretanto, o detector ainda funciona adequadamente na maioria dos casos.

Outro ponto que pode-se notar é que o detector tem maior dificuldade quando há uma grande quantidade de manchas nas folhas, e também se essas manchas são demasiadamente grandes. Novamente, como não utilizou-se um banco de dados específico para a detecção de doenças no plantio de batata é esperado que nem todas as manchas sejam detectadas. No entanto o modelo parece ser confiável quando o estágio da doença não está avançado.

4.2.4 Ilustrações utilizando o aplicativo

O detector irá detectar possíveis locais de atenção nas imagens e os resultados das predições serão desenhados na tela em forma de caixas sinalizadoras. Entretanto, estas imagens não serão disponibilizadas em tempo real já que o tempo de execução do detector adicionaria um gargalo de processamento em todo o sistema. Ao invés disto, o usuário pode abrir uma nova aba no próprio aplicativo, onde este poderá visualizar as imagens de atenção já com informações de classificação e detecção em formato de galeria de imagens, sendo possível até a aplicação de *zoom* para visualização com maior detalhe.

A Figuras 24 mostra a lista que é apresentada no final da execução e a Figura 25 mostra os resultados da detecção.

Alguns sistemas propostos na literatura possuem ótima precisão mas não são focados em problemas que podem acontecer em zonas rurais. (LEITE, 2021) propôs um sistema onde o usuário deve tirar uma foto com a câmera, fazer um requisição via internet e obter o resultado 10 segundos depois.

O sistema proposto neste projeto não necessita de auxílio de *internet*, o que torna a aplicação viável para produtores rurais afastados da zona urbana e que tenham limitações quanto à internet.

Outro benefício para o produtor é a velocidade de todo o sistema, o tempo de processamento total gira na casa de dezenas de mili segundos para maioria dos aparelhos mais recentes, sendo possível detectar possíveis problemas na própria interface de câmera, não sendo necessário tirar fotos para que a análise seja feita posteriormente.

4.3 Taxa de quadros

Com o intuito de avaliar a velocidade da aplicação, ou seja, a taxa de quadros por segundo (ou FPS *Frames Per Second*) visualizada pelo usuário nas telas onde há a utilização da câmera, mensurou-se a taxa de quadros na aplicação quando o classificador está sendo utilizado. Foram feitas 30 medidas e feita a média aritmética entre elas. Os resultados são apresentados na Tabela 5. Conforme apresentado na seção anterior Detecção,

Figura 24 – Tela de lista de imagens provida pelo aplicativo.

o tempo de inferência para o detector é demasiadamente alto (1,07s) para utilização em quadros da câmera, por isto ele não foi adicionado neste estudo.

Tabela 5 – Taxa de quadros do aplicativo.

Configuração	Média (FPS)
Classificador	24,62
Classificador+Detector	17,2
Classificador+Detector a cada 10 quadros	21,2

Pode-se notar uma queda significativa na taxa de quadros quando aplicamos o detector nos quadros coletados, além de problemas de consumo de memória alto neste caso. Por estas razões, decidiu-se que o detector só seria aplicado periodicamente na captura dos quadros (ele será acionado a cada 10 quadros analisados pelo classificador). É importante estarmos atentos a taxa de quadros pois o dispositivo utilizado por este projeto tem um poder computacional relativamente alto, portanto esta também é uma forma de tornar a experincia de uso interessante até mesmo em dispositivos com poder computacional menor. Utilizando esta técnica, pode-se obter uma taxa de quadros perto de 20 *fps*, que é

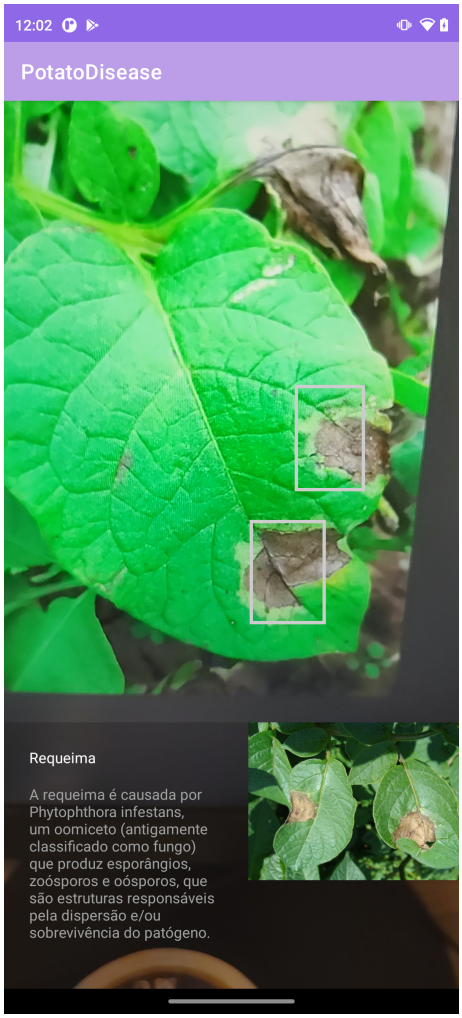

Figura 25 – Resultado da detecção de objetos no dispositivo.

considerada aceitável para o propósito deste projeto, possibilitando a visualização rápida de cada novo quadro.

5 CONCLUSÃO

A existência de um meio eficiente de se fazer o controle de doenças em plantações com o apoio de visão computacional e aprendizado de máquina é interessante para vários produtores rurais. O aplicativo proposto neste trabalho visa sanar problemas relacionados a esta área, utilizando técnicas que visam aprimorar a classificação e detecção de doenças no plantio de batata.

Foram utilizadas técnicas de aprendizado de máquina para tornar o sistema robusto a variações de ambiente e com alta taxa de acerto. Um sistema auxiliar de detecção e localização de possíveis pontos de atenção na imagem foi adicionado ao sistema.

Testes mostraram que o sistema é capaz de reconhecer com alta confiança as doenças pinta preta e requeima no plantio de batata, além de conseguir distinguir entre folhas saudáveis e doentes. Para todas as classes abordadas, o classificador apresentou uma medida F1 de 0,97 e para o detector mAP 0,69. Não obstante, métodos foram utilizados para aliviar possíveis problemas de processamento em aparelhos com menor poder computacional.

O sistema não necessita de auxílio de *internet*, o que torna a aplicação viável para produtores rurais afastados da zona urbana e que tenham limitações quanto à internet. Outro benefício para o produtor é a velocidade de todo o sistema, o tempo de processamento total gira na casa de dezenas de mili segundos para maioria dos aparelhos mais recentes, sendo possível detectar possíveis problemas na própria interface de câmera, não sendo necessário tirar fotos para que a análise seja feita posteriormente.

Com base nas questões de pesquisa propostas, este estudo selecionou alguns algoritmos de aprendizado de máquina disponíveis na literatura, analisando suas lacunas e desempenhos de classificação na detecção e classificação de doenças e pragas no plantio da batata. Para contornar a escassez de exemplos de doenças, foi utilizado imagens de outros plantios e para amenizar os efeitos de iluminação e balanceamento de branco, foram utilizadas técnicas de aumento de dados.

O modelo *MobilenetV3* apresentou um medida F1 de 0,97, resultado considerado alto para um classificador. Este modelo consegue classificar uma imagem em torno de 10 mili segundos, mesmo para celulares que não estejam entre os melhores em termos de performance no mercado.

Embora a detecção de doenças nas imagens exige um maior tempo de processamento, ela ainda pode ser utilizada em plano de fundo para auxiliar nos resultados. O detector apresentou uma medida 0,69 mAP, que pode ser considerado um bom detector de objetos. Este detector tende a detectar com maior facilidade doenças parecidas com as vistas no

banco de dados e tende a detectar com maior precisão regiões pequenas nas folhas.

O subitem seguinte apresenta sugestões de trabalhos futuros identificados ao longo deste projeto que visam aprimorar o aplicativo. Os testes realizados mostram, porém, que a ferramenta já pode ser utilizada de forma experimental.

5.1 Trabalhos futuros

Para evoluir a robustez, precisão e usabilidade deste sistema, recomenda-se alguns dos itens a seguir:

1. Evoluir o tamanho do banco de dados com mais exemplos de folhas de batata para que a taxa de acerto do classificador seja ainda maior;
2. Como a maior parte das imagens coletadas foram coletadas em ambiente controlado de laboratório, seria interessante coletar imagens de campo para que o sistema seja mais robusto a casos inesperados como imagens de outras plantações em conjunto com a batata, imagens de terra, folhas molhadas, entre outros;
3. Gerar um banco de dados com caixas delimitadoras em folhas de plantação de batata, já que o detector apresentado foi treinado com imagens de outra plantação;
4. Estudo de usabilidade e experiência de usuário com o aplicativo;
5. Explorar o funcionamento do sistema em um *drone* ao invés de um aplicativo.

REFERÊNCIAS

BODLA, N. *et al.* Improving object detection with one line of code. 2017. **Source:< https://arxiv. org/pdf/1704.04503. pdf**.

DAI, X. *et al.* Dynamic head: Unifying object detection heads with attentions. *In*: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2021. p. 7373–7382.

FOOD; NATIONS agriculture organization of the united. **Potato production globally**. 2021. http://faostat.fao.org/site/339/default.aspx. Acessado em 23 Setembro 2021.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. [*S.l.: s.n.*]: O'Reilly Media, 2019.

GONZALEZ, R. C.; WOODS, R. E.; MASTERS, B. R. **Digital image processing**. [*S.l.: s.n.*]: Society of Photo-Optical Instrumentation Engineers, 2009.

HE, K. *et al.* Deep residual learning for image recognition. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2016. p. 770–778.

HOWARD, A. *et al.* Searching for mobilenetv3. *In*: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [*S.l.: s.n.*], 2019. p. 1314–1324.

HOWARD, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HUGHES, D.; SALATHÉ, M. *et al.* An open access repository of images on plant health to enable the development of mobile disease diagnostics. **arXiv preprint arXiv:1511.08060**, 2015.

JIANG, P. *et al.* Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. **IEEE Access**, IEEE, v. 7, p. 59069–59080, 2019.

JONG, H. D.; JONG, W. D.; SIECZKA, J. B. The complete book of potatoes: What every grower and gardener needs to know. *In*: _____. [*S.l.: s.n.*]: Timber Press, 2011.

KIRATIRATANAPRUK, K. *et al.* Using deep learning techniques to detect rice diseases from images of rice fields. *In*: SPRINGER. **International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems**. [*S.l.: s.n.*], 2020. p. 225–237.

LAWRENCE, J. **Introduction to neural networks**. [*S.l.: s.n.*]: California Scientific Software, 1993.

LEITE, T. d. M. Deep learning em dois estágios para detecção e classificação de doenças em folhas de plantas com aplicação em dispositivos móveis. 2021.

LIN, T.-Y. *et al.* Microsoft coco: Common objects in context. *In*: SPRINGER. **European conference on computer vision**. [*S.l.: s.n.*], 2014. p. 740–755.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MELLO, R. Fernandes de; PONTI, M. A. Machine learning: a practical approach on the statistical learning theory. **Cham: Springer International Publishing**, 2018.

MOHANTY, S. P.; HUGHES, D. P.; SALATHÉ, M. Using deep learning for image-based plant disease detection. **Frontiers in plant science**, frontiers, v. 7, p. 1419, 2016.

PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. **arXiv preprint arXiv:1712.04621**, 2017.

PONTI, M. A. *et al.* Training deep networks from zero to hero: avoiding pitfalls and going beyond. *In*: IEEE. **2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [*S.l.: s.n.*], 2021. p. 9–16.

PáDUA, M. **Métricas de Avaliação**. 2021. https://medium.com/@mateuspdua. Acessado em 31 Janeiro 2022.

REZATOFIGHI, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. *In*: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S.l.: s.n.*], 2019. p. 658–666.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [*S.l.*], 1985.

SALAS, F. J. S.; TÖFOLI, J. G. Cultura da batata: pragas e doenças. *In*: _____. [*S.l.: s.n.*]: Instituto Biológico, 2017.

SANDLER, M. *et al.* Mobilenetv2: Inverted residuals and linear bottlenecks. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2018. p. 4510–4520.

SANTOS, F. P. dos; THUMÉ, G. S.; PONTI, M. A. Data augmentation guidelines for cross-dataset transfer learning and pseudo labeling. *In*: IEEE. **2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [*S.l.: s.n.*], 2021. p. 207–214.

SERVICE, E. R. **Potatoes and tomatoes are the most commonly consumed vegetables in United States**. 2021. https://www.ers.usda.gov/data-products/chart-gallery/gallery/chart-detail/?chartId=58340. Acessado em 23 Setembro 2021.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SUNG, F. *et al.* Learning to compare: Relation network for few-shot learning. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2018. p. 1199–1208.

TAN, M. *et al.* Mnasnet: Platform-aware neural architecture search for mobile. *In*: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [*S.l.: s.n.*], 2019. p. 2820–2828.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. *In*: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2020. p. 10781–10790.

TAYLOR, D. *et al.* **Digital photography complete course**. [*S.l.: s.n.*]: DK Publishing, 2015.

YANG, J. *et al.* Focal self-attention for local-global interactions in vision transformers. **arXiv preprint arXiv:2107.00641**, 2021.

YANG, T.-J. *et al.* Netadapt: Platform-aware neural network adaptation for mobile applications. *In*: **Proceedings of the European Conference on Computer Vision (ECCV)**. [*S.l.: s.n.*], 2018. p. 285–300.