Classificação de doenças nas plantas utilizando Transfer Learning numa aplicação móvel

Pedro Asseiro Rodrigues - a32653

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Prof. Doutor Pedro João Soares Rodrigues

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2019-2020

Classificação de doenças nas plantas utilizando Transfer Learning numa aplicação móvel

Pedro Asseiro Rodrigues - a32653

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Prof. Doutor Pedro João Soares Rodrigues

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2019-2020

Resumo

Atualmente, com o crescimento da população, os pequenos agricultores das zonas rurais transmontanas têm ficado cada vez mais desprotegidos, não possuindo meios para aumentar a produtividade das suas explorações. Consequentemente, a fraca produtividade, ligada à presença de carências nutritivas e doenças associadas, num ambiente cada vez mais competitivo, poderá levar ao abandono dos terrenos podendo acentuar a desertificação vivida em Portugal. Para tentar mitigar este problema esta dissertação pretende desenvolver uma solução de baixo custo, automática e fiável, para combater a presença de carências severas de nutrientes e doenças nas plantas, particularmente na oliveira, árvore relevante em Portugal. Neste sentido, foi desenvolvida uma aplicação móvel que utiliza o Transfer Learning para a classificação de doenças, visíveis na folha, obtidas com recurso à fotografia. Deste modo, construiu-se um dataset de imagens contendo quatro classes, oliveira saudável, carência de boro, carência de potássio e olho de pavão, que serviu de base de treino ao modelo usado. Este foi treinado recorrendo à rede neuronal MobileNetV2, possuindo uma precisão de 99%, com perdas de 1%. Os resultados em ambiente real mostraram-se bastante bons pois a taxa de acerto é de 85%, sendo uma solução bastante fiável.

Palavras-chave: Doenças em Plantas, Transfer Learning, App para Classificação de Imagens

Abstract

Currently, with the growth of the population, small farmers in rural areas in Trás-os-Montes have become increasingly unprotected, not having the means to increase the productivity of their farms. Consequently, the low productivity, linked to the presence of nutritional deficiencies and associated diseases, in an increasingly competitive environment, may lead to the abandonment of land and may accentuate the desertification experienced in Portugal. To try to mitigate this problem, this dissertation intends to develop a low-cost, automatic and reliable solution to combat the presence of severe nutrient deficiencies and diseases in plants, particularly in the olive tree, a relevant tree in Portugal. In this sense, a mobile application was developed that uses Transfer Learning for the classification of diseases, visible on the leaf, obtained using photography. In this way, an image dataset was built containing four classes, healthy olive, boron deficiency, potassium deficiency, and peacock spot, which served as the training base for the model used. This was trained using the MobileNetV2 neuronal network, with an accuracy of 99%, with losses of 1%. The results in real environment proved to be quite good since the hit rate is 85%, being a very reliable solution.

Keywords: Plant Diseases, Transfer Learning, App for Image Classification

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

CLAHE Contrast Limited Adaptative Histogram Equalization. 22, 34

CNN Convolutional Neural Network. 17, 18, 25, 27, 28, 43

DL Deep Learning. 17, 30

DNN Deep Neural Network. 17

FAO Food and Agriculture Organization. 5

IA Inteligência Artificial. 9, 15, 30

ML Machine Learning. 15, 17

MLP Multilayer Perceptron. 12, 13

PIL Python Imaging Library. 33

RNA Redes Neuronais Artificiais. 9–12

TL Transfer Learning. 26

VAB Valor Acrescentado Bruto. 5

Capítulo 1

Introdução

O capítulo 1 descreve o enquadramento e contexto da realização desta dissertação, a sua motivação, objetivos e estrutura do documento.

1.1 Enquadramento

O desenvolvimento atual tem potenciado o crescimento da população em zonas urbanas em detrimento de locais rurais, facto totalmente visível em Portugal, onde apenas cerca de um terço das pessoas vive num meio rural, estando esta ligada, em muitas das vezes, à agricultura [1]. Esta desigualdade demográfica é mais acentuada no interior de Portugal, nomeadamente em Trás-Os-Montes, onde apesar dos avanços e novas investigações no setor agrícola, esta zona do país continua a ser povoada maioritariamente por pequenos agricultores, muitas das vezes sem grande capacidade económica [2].

Neste sentido, os pequenos agricultores têm, em muitas das vezes, dificuldade em encontrar os melhores meios para potenciar a sua atividade, quer seja pela ausência monetária, pelo seu isolamento, pela fragmentação e dispersão das parcelas agrícolas, ..., entre outros fatores [2]. Deste modo, é possível observar, em determinadas explorações agrícolas, nomeadamente em olivais, a falta de acompanhamento especializado dos mesmos, denotando-se, entre outras, certas patologias como a carências nutritivas e doenças. Estas são a maior causa do insucesso de produtividade de uma exploração agrícola.

As carências nutritivas e doenças são visíveis, entre outros aspetos, através da folha da planta. Assim, é neste sentido que a análise foliar e visual se torna importante para a identificação correta de uma anomalia verificada numa planta [3].

Em consequência, as recentes investigações da metodologia do Transfer Learning ligadas ao conceito de Deep Learning e Inteligência Artificial, tem potenciado o seu uso de forma a identificar determinados padrões nas imagens em diversas áreas. Esta prática aplicada ao setor agrícola, nomeadamente à análise e identificação visual das folhas das plantas com ajuda de um smartphone, pode ser um meio simples e rápido de detetar determinados comportamentos numa planta.

1.2 Motivação

Tal como referido anteriormente, o setor agrícola transmontano é maioritariamente composto por pequenos agricultores que, em muitas das vezes, têm as suas culturas enfraquecidas, devido aos baixos rendimentos da exploração. As carências nutritivas e doenças são um dos exemplos. Neste sentido, o uso indevido de adubos e produtos fitofármacos ou a sua ausência pode causar danos nas plantas tornando a exploração agrícola inviável.

Por conseguinte, esta dissertação tem como finalidade implementar um sistema de identificação de determinadas doenças e carências nutritivas nas oliveiras através da análise visual foliar. Este sistema funcionará numa aplicação móvel recorrendo às técnicas de Transfer Lerning, sendo uma solução rápida e de baixo custo para o utilizador.

1.3 Objetivos

O objetivo desta dissertação é implementar e testar modelos de Transfer Learning para a deteção de doenças e carências nutritivas em plantas da região transmontana, nomeadamente a oliveira, classificando a sua folha. Pretende-se, também, testar a viabilidade desta solução em ambiente real utilizando uma aplicação móvel funcionando num smartphone. Para a implementação do objetivo geral, os seguintes objetivos específicos são definidos:

- Realizar um estudo teórico sobre as metodologias de Inteligência Artificial, tais como: Redes Neuronais, Deep Learning e Transfer Lerning.
- Obter um conjunto de imagens de folhas de oliveira, saudáveis e doentes, que constituirão o dataset.
- Estudar diferentes modelos de Transfer Learning e verificar a sua eficácia.
- Desenvolver uma aplicação para a deteção de determinadas doenças e carências nutritivas, verificáveis através da análise foliar visual, recorrendo a um smartphone.
- Testar a solução obtida em ambiente real e verificar os seus resultados.

1.4 Estrutura do Documento

A dissertação está estruturada da seguinte forma:

O **Capítulo 2** apresenta a fundamentação teórica sobre a Sanidade Vegetal das Culturas Agrícolas, metodologias de Inteligência Artificial, Machine Learning, Redes Neuronais Artificiais, Deep Learning, e Transfer Learning. Neste capítulo são, também, descritos Trabalhos Relacionados.

No **Capítulo 3** expõe e detalha os processos de desenvolvimento da solução proposta, nomeadamente a construção do dataset, desenvolvimento do modelo de classificação das imagens e a aplicação móvel.

O **Capítulo 4** mostra e analisa os resultados da solução desenvolvida através dos testes realizados em ambiente real.

Por fim, o **Capítulo 5** apresenta as conclusões da dissertação e possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O capítulo 2 descreve e exemplifica os conceitos necessários ao entendimento desta dissertação. Assim, em primeiro lugar, irá ser feita referência à importância da Sanidade Vegetal, seguida dos conceitos ligados à informática, nomeadamente, Inteligência Artificial, Machine Learning, Redes Neuronais Artificiais, Deep Learning e Transfer Learning. A finalizar, estão apresentados e analisados Trabalhos Relacionados à dissertação.

2.1 Sanidade Vegetal das Culturas

A sanidade vegetal das culturas agrícolas é importante para a sustentabilidade económica e ambiental, sendo um motor de impulsionamento socioeconómico. Neste sentido, em Trás-os-Montes, o setor primário, nomeadamente o agrícola, tem um peso três vezes superior à média nacional, relativamente ao Valor Acrescentado Bruto (VAB) [2]. É neste sentido que as Nações Unidas celebram em 2020 o Ano Internacional da Sanidade Vegetal, com o intuito de consciencializar a população mundial para a importância da sanidade vegetal das culturas agrícolas [4]. Segundo a Food and Agriculture Organization (FAO) cerca de 40% das culturas anuais são perdidas devido a pragas e doenças nas plantas [4].

O diagnóstico visual nas folhas das plantas tem grande importância prática, pois pode pode ser um elemento essencial para a tomada de decisões rápidas na correção de deficiências [5]. Desta forma, esta metodologia de análise constitui uma ferramenta importante

para a avaliação do estado nutricional das plantas, nomeadamente nas fruteiras, pois é possível observar a interação de todos os fatores que influenciam a nutrição da mesma [6].

A observação direta da folha das árvores poderá indicar a presença de sintomas de deficiência ou toxidade provocada por carência ou excesso de um determinado nutriente [3]. Esta técnica requer experiência por parte do observador, pois em determinadas situações pode ser facilmente confundível por doenças ou sintomas anómalos parecidos [3]. Consequentemente, quando uma folha se mostra com sinais visíveis de carência ou excesso de um nutriente, a planta já se encontra em estado avançado de stresse nutritivo profundo [3]. Na figura 2.1 é possível observar a carência severa de boro na oliveira, visível na árvore, folha e fruto. É neste sentido que é importante detalhar e identificar corretamente a má nutrição ou a presença de doenças, que atacam de forma severa as culturas, no estado inicial, de modo a reduzir as perdas de produtividade.

Figura 2.1: Carência severa de boro na oliveira, adpatado de [7]

2.1.1 Importância do setor olivícola

Tal como referido anteriormente, é extremamente importante garantir a correta nutrição e proteção fitossanitária das culturas, prevenindo o avanço severo das doenças causando danos nas plantas. Desta forma, representando o olival em Portugal cerca de 48% da área de produção de culturas permanentes, estando presente em 43% das explorações agrícolas e sendo que 9% são especializadas em olivicultura é de grande importância garantir o bom

estado de produção e rentabilidade da mesma [8].

Neste sentido, a produção do azeite, proveniente da transformação da azeitona, contribuiu em 2019 para mais de 100 milhões de euros na economia portuguesa, sendo que Trás os Montes representa uma parte na produção total [9]. Consequente, é na região transmontana onde os azeites são reconhecidos a nível mundial pela sua qualidade única representando cerca de 30 milhões de euros na economia [10].

Por outro lado, é importante ressaltar que este setor só é viável se a correta sanidade e nutrição das plantas estiver dentro dos parâmetros sustentáveis, de forma a obter uma produção regular.

2.1.2 Principais patologias no olival trasmontano

Em Trás os Montes o olival está maioritariamente instalado em solos de meia encosta onde a espessura é reduzida, significando que têm uma baixa capacidade de reter nutrientes, sendo os teores de potássio baixos a médios e muito deficientes em boro [7]. Por outro lado, no que se refere ao estado fitossanitário provocados por fungos, o olho de pavão, torna-se problemático nesta zona do país, assumindo estragos podendo reduzir a produção do olival [11].

É pois, importante, ressaltar a relevância na identificação das carências nutritivas apresentadas acima, carência de boro e potássio, bem como o reconhecimento da doença olho de pavão.

Características principais, visíveis na folha, da carência de potássio, boro e olho de pavão na oliveira:

- Carência de potássio - Amarelecimento seguido de necroses na zona do ápice e nas margens das folhas, figura 2.2 [11].
- Carência de boro - Folhas cloróticas, por vezes com uma separação evidente entre a parte apical amarelecida e a parte basal ainda verde, figura 2.3 [7].
- Olho de Pavão - Manchas arredondadas concêntricas com aspeto esbranquiçado ou

prateado (Outono/Inverno), mudando para cores amareladas à medida que a doença evolui, figura 2.4 [11].

Figura 2.2: Folhas de oliveira com carência de potássio [7]

Figura 2.3: Evolução da carência de boro na folha da oliveira [12]

Figura 2.4: Evolução do estado fenológico do olho de pavão na oliveira [13]

Neste sentido, a deteção no estado inicial da doença/carência nutritiva é um fator chave para a redução dos estragos e prejuízos. Com efeito, esta dissertação tem como objetivo encontrar e desenvolver uma solução fiável e rápida que pretende prevenir estes problemas na sanidade das culturas, através da análise visual das folhas das oliveiras em ambiente real.

2.2 Inteligência Artificial

A Inteligência Artificial (IA) é uma área da informática que pretende simular a inteligência humana usando apenas computadores. O termo IA surgiu pela primeira vez em 1955 por John McCarthy definindo como:

> "The science and engineering of making intelligent machines." [14]

Autores mais recentes como Andreas Kaplan e Michael Haenlein definem a IA como a capacidade para interpretar dados externos, aprender com esses dados e utilizar a aprendizagem adquirida para atingir objetivos e tarefas específicas [15]. A IA consiste na criação de sistemas que tenham um comportamento inteligente capacitados para realizar tarefas com um nível de competência igual ou superior ao do ser humano [16]. Consequentemente, este campo da computação tem a particularidade de fornecer modelos de apoio à tomada de decisão e ao controlo com base em factos reais, mesmo apoiado em dados incompletos [17].

2.3 Redes Neuronais Artificiais

As Redes Neuronais Artificiais (RNA) ou em inglês Neural Networks (NN) são metodologias computacionais inspiradas no funcionamento das redes neuronais do cérebro humano, criadas pela primeira vez em 1943 por McCulloch e Pitts [19]. Assim, as RNA são sistemas complexos formados por uma enorme interligações entre as unidades de processamento, que pretendem simular as funções e características básicas das redes cerebrais humanas

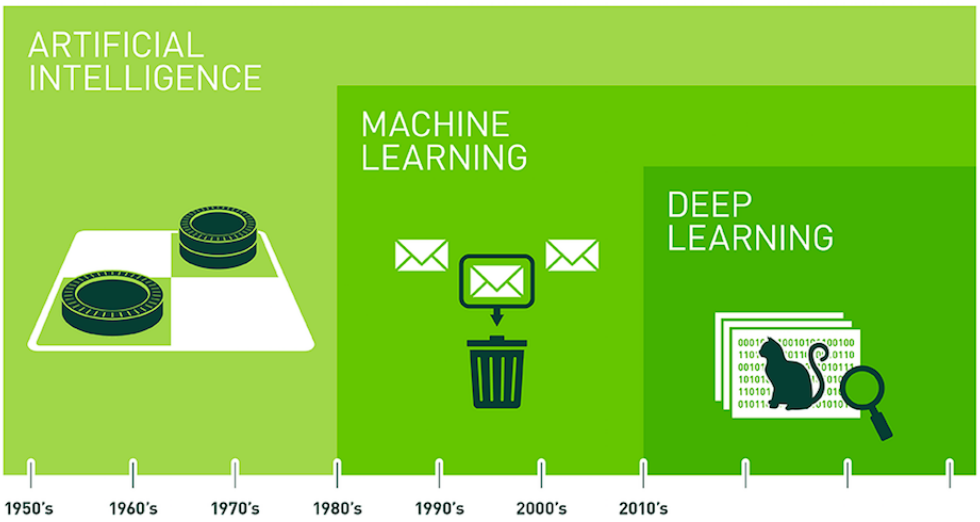

Figura 2.5: Relação entre Inteligência Artificial, Machine Learning e Deep Learning [18]

[20]. Estas, sendo conjunto de ligações entre unidades simples (neurónios artificiais ou nós), possuem capacidade de efetuar cálculos paralelos para o processamento de dados e conhecimento [21]. Por outro lado, algumas das vantagens da utilização das RNA são a capacidade de resolver problemas não lineares, adaptar os seus pesos sinápticos com base nas condições do ambiente e habilidade de mostrar confiança na decisão e tolerância a falhas [19]. Para compreender melhor o funcionamento das RNA é importante compreender alguns conceitos que irão ser mostrados e exemplificadas de seguida.

2.3.1 Analogia entre Redes Neuronais Artificiais e Biológicas

O sistema nervoso biológico humano possui enormes quantidades de neurónios interconetados que formam múltiplas ligações entre eles desempenhando variadíssimas funções [21].

Na figura 2.6 é possível observar o neurónio constituído por três componentes principais: dendritos, corpo celular e axónio. O corpo celular carrega informações sobre as suas características, sendo também o local de recepção de estímulos, através de contactos sinápticos. Os dendritos são ramificações dos neurónios que recebem e passam sinais eléctricos e químicos para a célula. Os axónios são um extenso tecido celular, que recebem

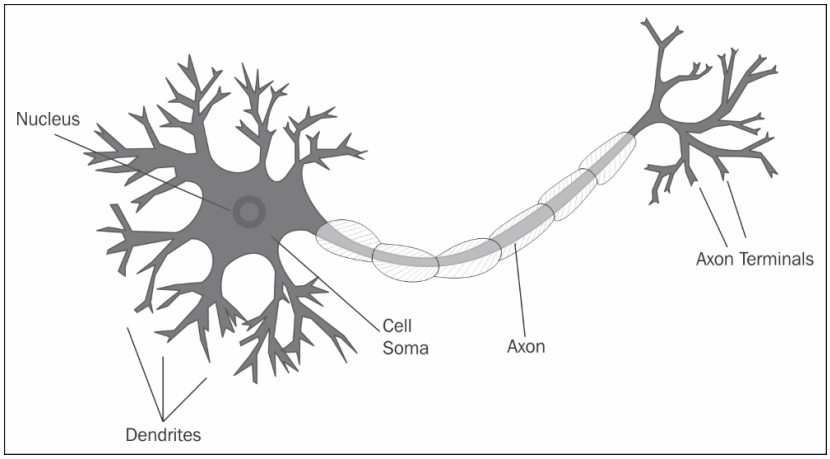

Figura 2.6: Neurónio biológico [22]

sinais do corpo da célula e transmitem-nos para os dendritos, através da sinapse [21].

Desta forma, nas RNA, tal como nas redes biológicas, as conexões entre os nós representam os axónios e dendritos e os pesos das conexões representam as sinapses. A figura 2.7 ilustra o neurónio biológico, com vários sinais de intensidade X e a força sináptica W e sistema RNA equivalente, representado pelo perceptron (o tipo de RNA mais básico) [21].

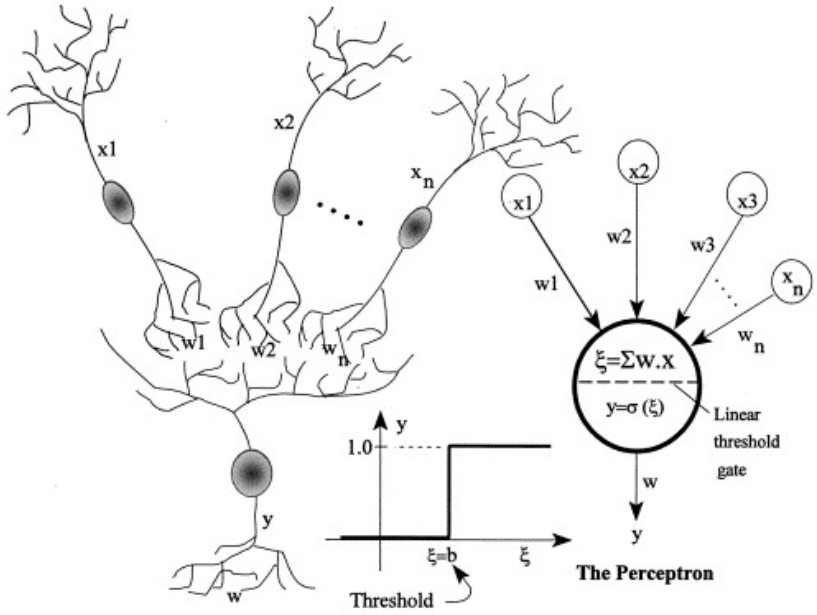

Figura 2.7: Analogia entre o neurónio biológico e o perceptron [21]

2.3.2 Perceptron e Multilayer Perceptron

O Perceptron é o tipo de RNA mais básica, foi introduzida por Frank Rosenblatt, inspirada nos trabalhos anteriores de Warren McCulloch e Walter Pitts, tendo a capacidade de classificar padrões linearmente separáveis, conjunto de dados separáveis através de um plano [23]. Este tipo de rede é constituída apenas por uma camada, possuindo várias entradas com pesos ajustáveis e apenas uma saída binária, figura 2.8. A saída do perceptron, equação 2.1, 0 ou 1, é determinada se a soma ponderada dos valores de entrada X, multiplicados pelos pesos de cada entrada W, é menor ou maior que algum valor limite (threshold) [23].

$$output = \begin{cases} 0, & if \quad \sum_j w_j x_j \quad \leq threshold \\ 1, & if \quad \sum_j w_j x_j \quad > threshold \end{cases} \tag{2.1}$$

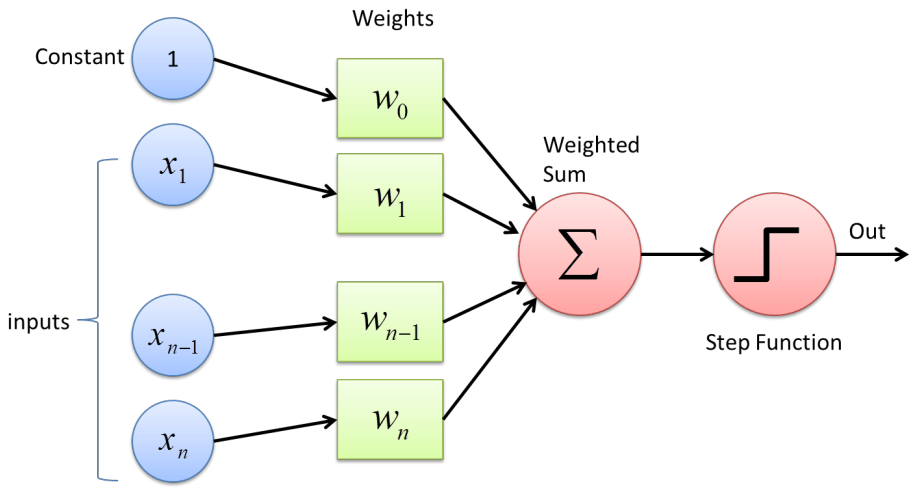

Figura 2.8: Exemplo de um perceptron [24]

O Multilayer Perceptron (MLP) surgiu devido à necessidade de resolver problemas mais complexos não linearmente separáveis, conjunto de dados que não é separável por um plano ou hiperplano, adicionando camadas ocultas ao modelo do perceptron. Assim, uma MLP, representada na figura 2.9, é constituída por possuir uma camada de input, uma camada de output e pelo menos uma camada intermédia, que liga cada nó a todos os nós da camada seguinte. Este modelo funciona como uma rede feedforward, onde a

saída do neurónio é conectada ao neurónio da camada seguinte [19]. Neste sentido, este tipo de redes com múltiplas camadas pode ser entendido como um grafo onde os nós são neurónios e as ligações fazem as funções das sinapses [25]. As MLP utilizam técnicas de aprendizagem supervisionada recorrendo ao algoritmo backpropagation para o seu treino, onde a complexidade é dada pela quantidade de camadas ocultas e número de nós nessas camadas [26] [19].

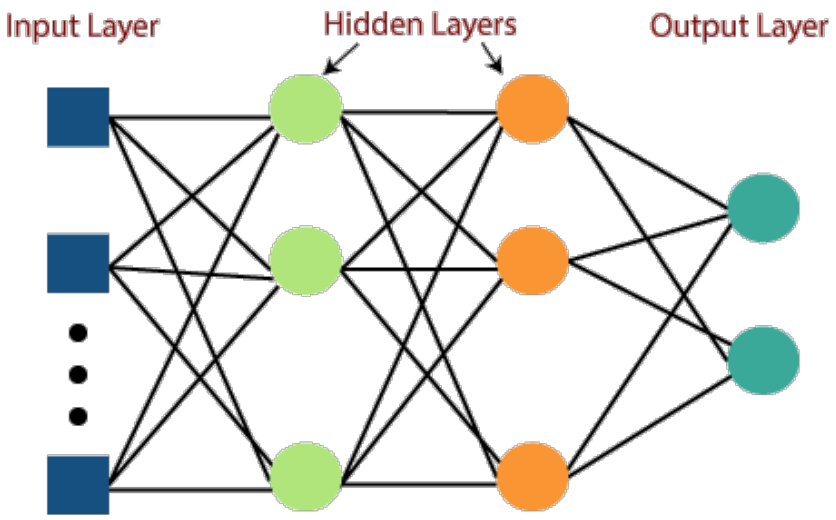

Figura 2.9: Exemplo de uma rede Multilayer Perceptron com duas camadas ocultas [27]

2.3.3 Função de Ativação

Os neurónios artificias, tal como os biológicos, apresentam diferentes estados, ativos ou não ativos [28]. Neste sentido, as funções de ativação têm a capacidade de decidirem se um neurónio ou nó vai ser ativo ou não. Geralmente, nas redes feedforward são usadas funções que combinam vetores lineares com escalares nos nós ocultos, que recebem valores dos nós da camada anterior. No entanto, existem diversas funções que podem influenciar o desempenho e a complexidade da rede [29]. Na figura 2.10 estão presentes algumas funções de ativação, onde é possível observar a sua representação matemática e gráfica.

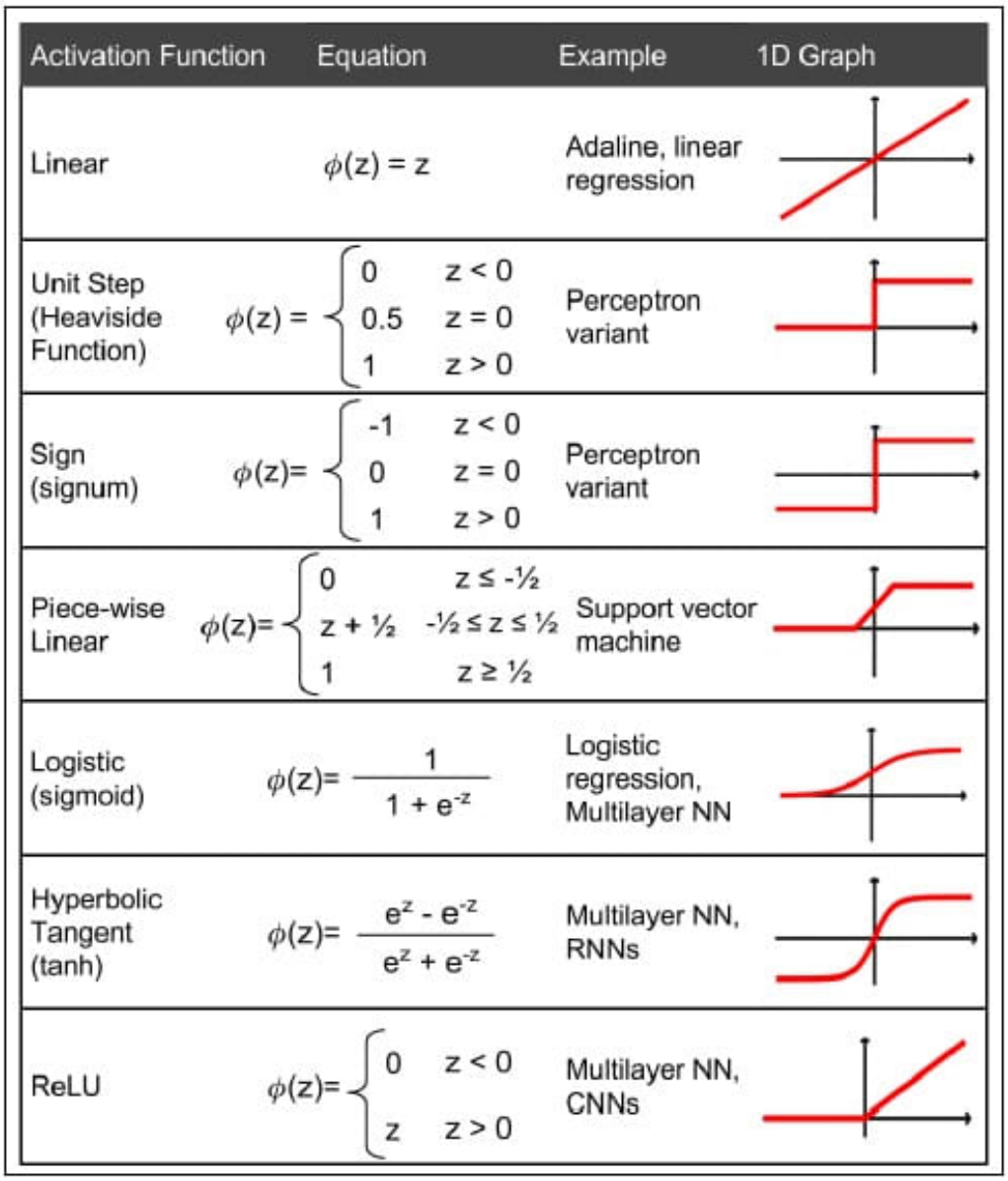

Figura 2.10: Diferentes funções de ativação com representação matemática e gráfica [30]

2.3.4 Algoritmo Backprogation

O algoritmo de Backpropagation é o processo onde ocorre a aprendizagem da rede, comparando o valor obtido com o valor desejado [19]. Este processo acontece, pois pode ocorrer que no conjunto dos pesos atribuídos a cada camada não sejam os mais corretos. Neste sentido, este algoritmo é propagado pelas camadas internas até se ajustar aos pesos da

rede. Este baseia-se na descida de gradiente, onde os pesos são atualizados de forma a encontrar um mínimo local [31].

A descida de gradiente é um algoritmo de otimização interativa de primeira ordem que tenta encontrar um mínimo local numa função [32].

Assim, o backpropagation consiste em duas etapas, a primeira onde os valores de entrada são propagados pelas camadas para gerar um conjunto de saída, permanecendo os pesos inalterados. Na segunda etapa, os pesos são ajustados com base na diferença entre a saída obtida e esperada [19].

2.4 Machine Learning

O Machine Learning (ML) ou em português Aprendizagem Máquina é uma área da informática directamente ligada à Inteligência Artificial que consiste na execução de algoritmos que extraem conhecimento a partir de bases de dados [33]. O termo ML foi definido por Arthur Samuel, pioneiro no campo da IA, como o estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados [34]. Segundo a mesma linha de pensamento os algoritmos de ML são aprendizes, não precisando da intervenção humana e quanto maior for a quantidade e diversidade de dados mais eficazes se tornam [35]. De acordo com Langley, a principal diferença entre a IA e o ML é o facto de esta última usar modelos estatísticos e teorias da probabilidade em detrimento de abordagens de natureza prática e simbólica [36].

No ML existem diferentes tipos de aprendizagem consoante os dados estejam ou não identificados. Assim, o ML pode ser dividido em três principais tipos conforme representado na figura 2.11, sendo eles aprendizagem supervisionada, não supervisionada e por reforço, existindo também a semi-supervisionada.

- Aprendizagem supervisionada: este tipo usa algoritmos que constroem um modelo matemático, onde o conjunto de exemplos é rotulado (conhecido), cujo objetivo é encontrar uma hipótese capaz de classificar novos exemplos entre as classes atuais [38]. Em sistemas deste tipo são classificados problemas de regressão, tentando

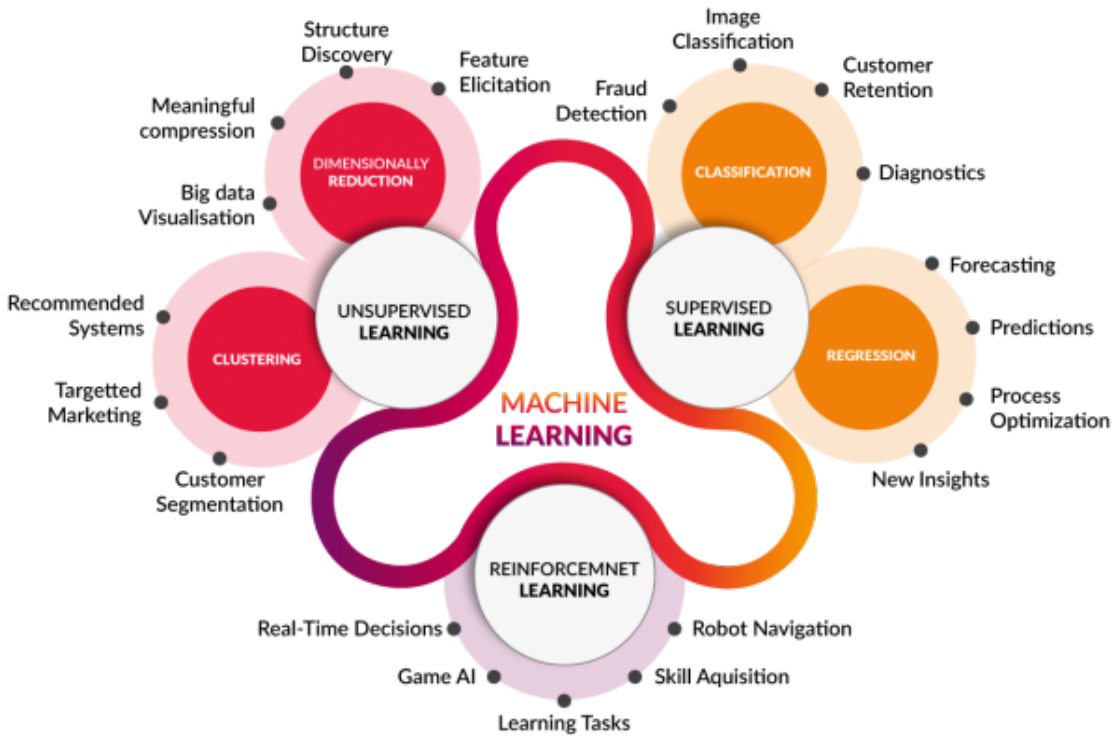

Figura 2.11: Tipos de aprendizagem de Machine Learning [37]

prever resultados de uma saída contínua, ou problemas de classificação, onde o objetivo é antever resultados de uma saída discreta [38].

- Aprendizagem não supervisionada: este tipo usa um conjunto de dados não rotulados onde o propósito é tentar prever e estabelecer a igualdades ou a existência de grupos. Este conceito torna a sua representação mais fácil de entender podendo reutilizá-la para outros fins [38].

- Aprendizagem semi-supervisionada: este tipo utiliza dados, onde alguns são rotulados e outros não rotulados. Normalmente, a quantidade de dados rotulados é menor divido ao seu custo, servindo de impulso ao outro tipo de dados [39] [40].

- Aprendizagem por reforço: é diferente dos outros tipos de aprendizagem, existindo um agente que interage com o ambiente desconhecido. Este tenta aprender, via

tentativa erro com uma ação direta com o ambiente, baseando-se em reforços (punições ou recompensas), sendo o seu objetivo estabelecer uma política que maximize a quantidade de recompensas [38] [41].

2.5 Deep Learning

O Deep Learning (DL) ou em português Aprendizagem Profunda, é um sub-campo do Machine Learning que utiliza o processamento de informações em níveis não lineares, onde são adicionadas diversas camadas escondidas com conexões parciais [42]. Estas camadas ocultas representam conjuntos de estruturas de dados muitas das vezes complexas para o entendimento humano [43].

Assim, o DL apresenta uma estrutura baseada em grafos com camadas de processamento de aprendizagem complexas, descrevendo diferentes conjuntos de padrões com múltiplos níveis [44]. Contrariamente ao ML, onde os dados nas camadas mais superficiais tendem a convergir, no DL são utilizados algoritmos que permitem a escala dos dados, ou seja, quanto mais dados existirem mais eficiente será a rede, melhorando à medida que esta aumenta [45].

Neste sentido, um dos grandes objetivos na utilização de este tipo de redes foca-se no facto destas arquitecturas tentarem identificar abstrações nos dados, dos níveis mais baixos para os mais altos, obtendo novas representações [46]. Por outro lado, uma das grandes diferenças para o ML foca-se na extracção de características, sendo que no DL são feitas de forma automática em detrimento do modo manual [47]. O DL apesar de ser difícil de treinar mostra-se particularmente eficiente com enormes quantidades de dados, aumentando significativamente a sua performance, comparando com o ML, na qual um grande aumento de dados não traduz grande relevância na redução de erro [42].

2.5.1 Redes Neuronais Convolucionais

As Convolutional Neural Network (CNN) ou em português Redes Neuronais Convolucionais são uma categoria de redes Deep Neural Network (DNN) que são eficazes e assertivas

no reconhecimento e classificação de imagens, sendo amplamente mencionadas no estado de arte em várias áreas [48]. Este tipo de redes possuem duas características principais que as distinguem das redes tradicionais: [49]

1. As CNN utilizam uma operação de convolução, que consiste na aplicação de um kernel à imagem resultando num mapa de características.

2. O uso das CNN deve utilizar de forma explicita uma entrada do tipo matriz, tal como uma imagem.

Assim, o processo básico do funcionamento das CNN é receber uma imagem, processar e classificar, sendo compostas por 3 etapas: Convolução, Pooling, e Fully Connected. Na figura 2.12 está representada um exemplo de uma CNN.

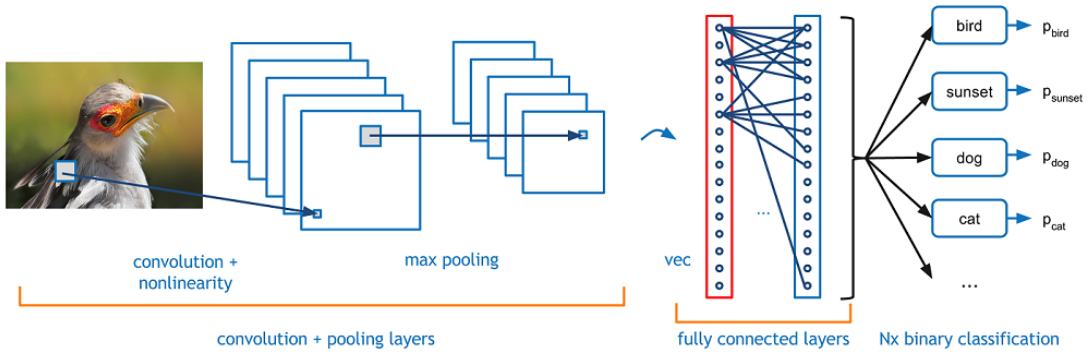

Figura 2.12: Exemplo de uma Rede Neuronal Convolucional [50]

Convolução

A convolução é a primeira etapa, onde são extraídas características de uma imagem aplicando um kernel [51]. Este kernel, também denominado de feature detector (detetor de características) desempenha uma função importante na rede, pois é composto por operações matemáticas como a convolução [52]. Após a aplicação da operação de convulção através do kernel é possível obter o feature map (mapa de características) da imagem, onde está a informação relevante da mesma [51]. Deste modo, o ponto importante neste

processo é detetar as características principais de uma imagem, podendo ser aplicados vários kernels de forma a obter um número arbitrário de mapa de características, tal como exemplificado na figura 2.13 [52].

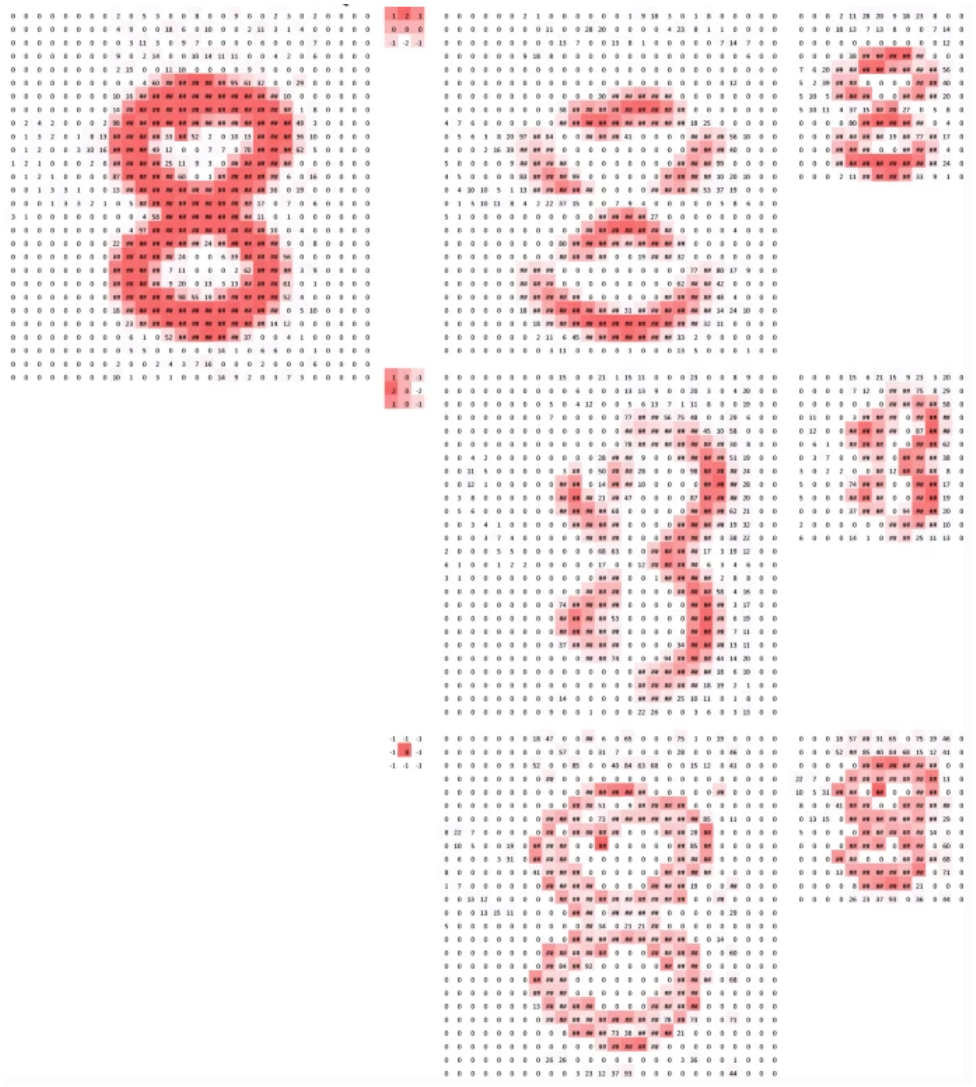

Figura 2.13: Exemplos de diferentes kernels aplicados numa imagem [52]

Pooling

A camada de Pooling tem como função reduzir a dimensão do feature map, obtido anteriormente, reduzir overfitting, ruídos desnecessários e salientar de forma mais evidente as características obtidas na etapa anterior. O método de redução ou sumarização das

características pode ser de diferentes tipos como: Max Pooling (escolhe o valor maior) ou Average Pooling (escolhe o valor médio) [51].

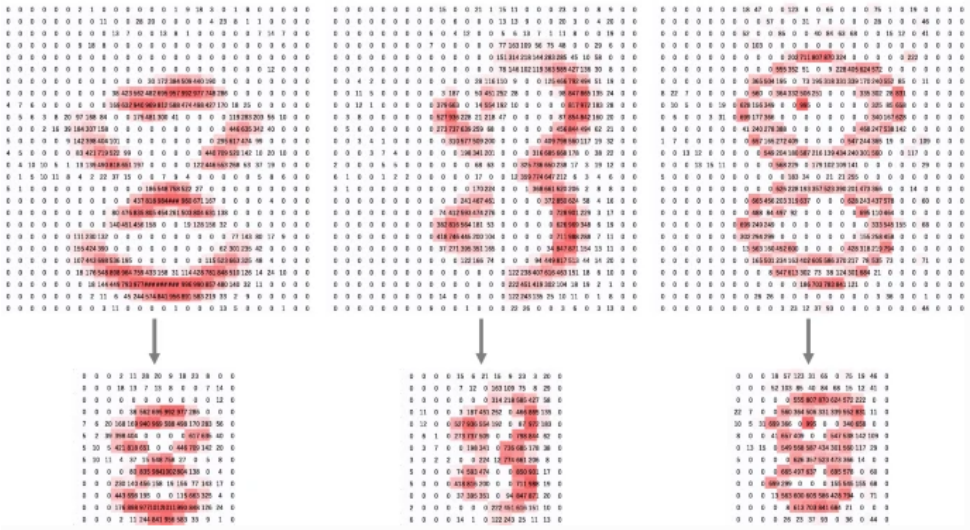

Figura 2.14: Exemplo da operação max pooling [52]

Fully Connected

O objetivo desta camada é, a partir da camada de convolução e pooling, classificar a imagem de entrada nas classes pertencentes ao conjunto de treino [53]. Assim, após a extração das características pelas camadas anteriores, estas são transformadas numa matriz unidimensional (Flattening) e conetadas a uma ou mais camadas totalmente conetadas, em que cada entrada é conetada a cada saída por um peso aprendível. A camada final possui o mesmo número de nós de saída que o número de classes [52].

2.5.2 Overfitting e underfitting

O overfitting é um dos principais problemas do mau funcionamento de um modelo treinado numa rede neuronal. Este acontece quando o modelo se ajusta demais aos dados de treino. Neste caso a precisão no conjunto de treino pode ser elevada e as perdas relativamente baixas, mas no conjunto de validação as perdas são claramente mais elevadas, notando-se um sobreajuste do mesmo. Por outro lado, o underfitting é o oposto do overfitting, ou

seja, quando o modelo treinado não é suficientemente eficaz, tendo uma precisão baixa, havendo espaço para melhorias [54].

Assim sendo, o overfitting, poderá acontecer quando o modelo é treinado vezes a mais. Por outro lado, o underfitting poderá representar a ausência de um modelo poderoso, número de treinos insuficientes, ou se estiver super regularizado [54]. Na figura 2.15 é possível observar graficamente o overfitting, underfitting e o modelo ideal de treino.

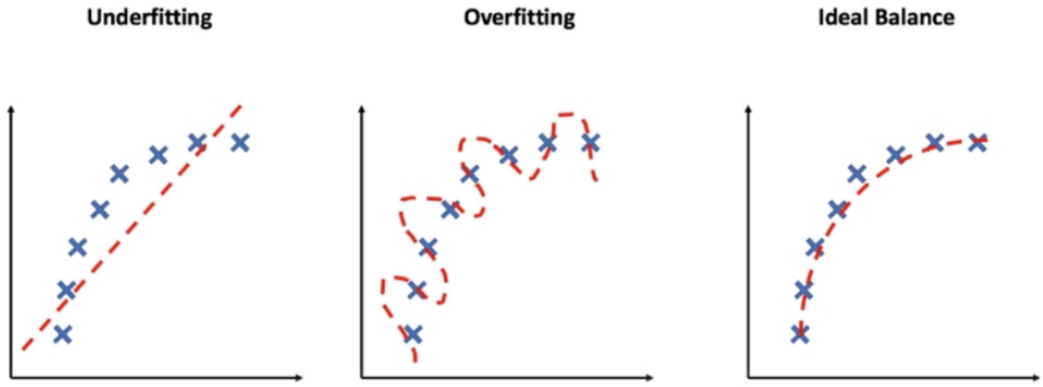

Figura 2.15: Exemplos de modelos com underfitting, overfitting e o modelo ideal [55]

Para evitar problemas de overfitting devem-se adotar as seguintes estratégias: [42]

- Reduzir a complexidade do modelo de treino.
- Reduzir o número de camadas, neurónio ou épocas.
- Usar a função de dropout.
- Parar antecipadamente o treino das características.
- Regularizar o kernel no treino do modelo.

Outra solução para diminuir o overfitting é recorrer ao data augmentation. Esta técnica consiste em aumentar o conjunto de dados fornecendo mudanças em determinados aspetos como a modificação em termos de cores ou a transformação geométrica, tal como mostrado na figura 2.16 [56].

A utilização de filtros Contrast Limited Adaptative Histogram Equalization (CLAHE), técnica de melhoramento do contraste, também pode ser utilizada no data augmentation. Este método divide a imagem em blocos de pequenas dimensões, aplicando-lhe equalização do histograma (redistribuir os valores dos níveis de cinza, de forma a uniformizar o histograma), garantindo contraste mais detalhado aos objetos presentes em cada bloco na imagem [57].

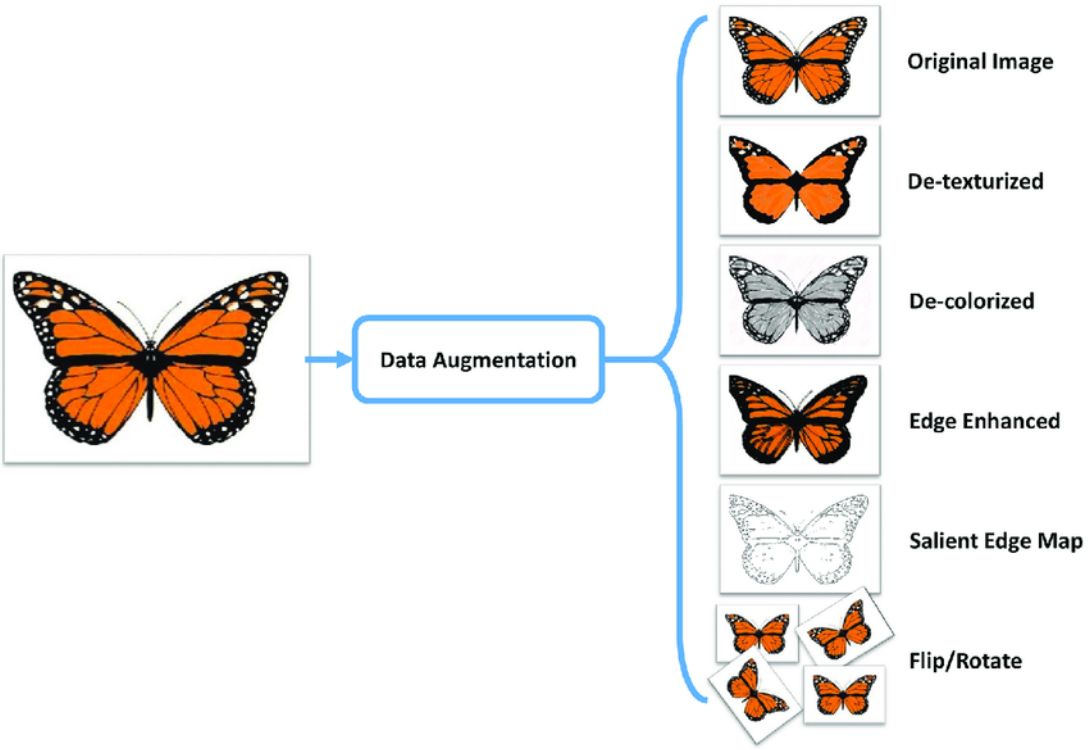

Figura 2.16: Exemplo de data augmentation numa imagem [58]

2.5.3 Métricas de Avaliação

O modelo treinado depende na sua maioria da veracidade dos resultados obtidos. Neste sentido, a matriz de confusão, exemplificada na figura 2.17, é um método importante para validar os resultados obtidos no treino, sendo necessário conhecer os valores reais. Esta matriz é composta por quatro medidas: Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP), Falsos Negativos (FN) [59] [60].

- Verdadeiros Positivos (VP) - Número de exemplos da classe positiva classificados corretamente.

- Verdadeiros Negativos (VN) - Número de exemplos da classe negativa classificados corretamente.

- Falso Positivo (FP) - Número de exemplos da classe negativa classificados incorretamente, ou seja, reconhecidos na classe positiva.

- Falso Negativo (FN) - Número de exemplos da classe positiva classificados incorretamente, ou seja, reconhecidos na classe negativa.

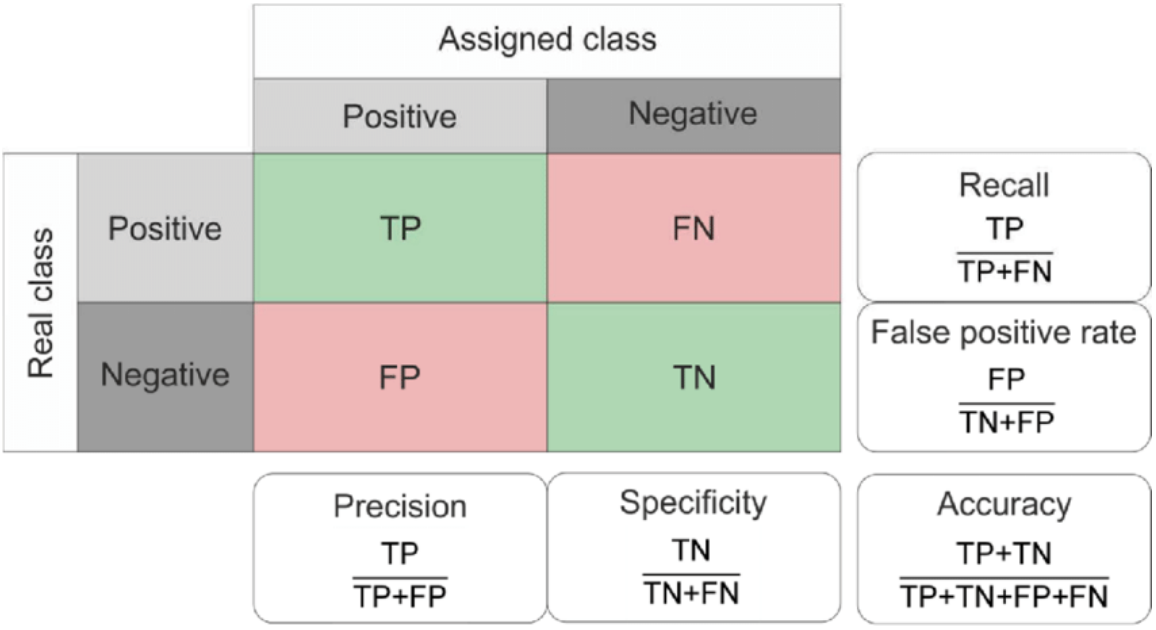

Figura 2.17: Exemplo de uma matriz de confusão com a relação entre precisão, especificidade, sensibilidade e acurácia [61]

A partir da matriz de confusão é possível obter a taxa de acerto, precisão, sensibilidade, especificidade, detalhadas a seguir [59].

- Taxa de acerto (Accuracy) - Mede a taxa de acerto do modelo. É calculada com a soma dos valores VP e VN (valores da diagonal da matriz) dividindo pela soma

de todos os elementos da matriz. Na equação 2.2 está a expressão matemática que calcula a acurácia.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.2}$$

- Precisão (Precision) - Esta medida descreve quantos dos elementos classificados como positivos foram previstos de positivos. É calculada através dos valores VP com a divisão da soma dos valores VP e FP. (Equação 2.3)

$$Precision = \frac{VP}{VP + FP} \tag{2.3}$$

- Sensibilidade (Recall) - Esta medida descreve de todas as amostras positivas quantas foram classificadas como positivas. É calculada através dos valores VP com divisão da soma dos valores de VP e FN. (Equação 2.4)

$$Recall = \frac{VP}{VP + FN} \tag{2.4}$$

- Especificidade (Specificity) - Esta medida descreve quantos dos exemplos classificados como negativos foram previstos de negativos, oposto da precisão. É calculada através dos valores VN com divisão da soma dos valores de VN e FP. (Equação 2.5)

$$Specificity = \frac{VN}{VN + FP} \tag{2.5}$$

- Medida F (F1-Score) - Esta medida mede a fiabilidade do modelo. Quanto mais próximo de um mais fiável será. É calculada através da média harmónica entre a relação da Precisão (Precision) e Sensibilidade (Recall). Esta medida é muito utilizada, pois apresenta uma avaliação mais confiável, principalmente nos modelos desbalanceados. (Equação 2.6)

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.6}$$

2.5.4 Arquiteturas CNN

Atualmente existem diferentes arquiteturas de redes CNN, sendo importante detalhar algumas, a fim de se obter alguns exemplos das mesmas.

MobileNet

A MobileNet é um tipo de rede neuronal desenvolvida pela Google que tem como o objetivo melhorar o desempenho em tempo real do Deep Learning sob condições de hardware limitadas, como os smartphones, podendo reduzir o número de parâmetros sem sacrificar a precisão [62]. A MobileNetV1 é baseada no uso de convoluções separáveis em profundidade [63]. A nova versão MobileNetV2 baseia-se na anterior, usando convolução separável em blocos de profundidade, apresentando novos recursos [64]:

1. Bottlenecks lineares entre camadas.

2. Conexões de atalho entre bottlenecks.

A estrutura da rede MobileNetV2 está representada na figura 2.18, onde é possível observar os conceitos descritos anteriormente.

GoogLeNet

A GoogLeNet, também conhecida como InceptionV1, primeira versão deste tipo de redes, foi introduzida em 2014 por Christian Szegedy com 22 camadas [65]. Este tipo de rede introduziu um novo conceito, denomidado de "inception module", que reduz significativamente o número de parâmetros em 12 vezes comparando com a AlexNet [66].

A InceptionV2 introduz a fatorização, em convoluções menores e algumas mudanças à versão InceptionV1, por exemplo utiliza 3 convoluções de 3×3 em detrimento de 7×7.

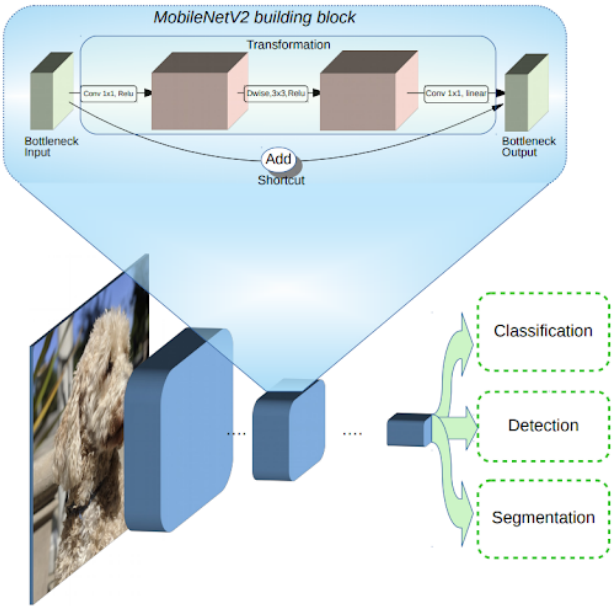

Figura 2.18: Arquitetura da rede MobileNetV2 [64]

Já a InceptionV3 é uma variante do InceptionV2 que adiciona a BN-auxiliar (camada totalmente conetada, Fully Connected, é normalizada) [66].

2.6 Transfer Learning

O Transfer Learning (TL), em português Aprendizagem por Transferência, está relacionado com o Machine Learning e com o Deep Learning, pretendendo resolver problemas relacionados com conhecimento adquirido em situações anteriores [67]. Um exemplo deste tipo de utilização poderá ser o conhecimento adquirido ao reconhecer automóveis e aplicá-lo no reconhecimento de camiões. Usando esta técnica é possível transferir os pesos das camadas de extracção de recursos de um modelo treinado sobre um conjunto de dados para outro modelo que irá ser treinado com um novo conjunto de dados [68].

O TL é geralmenente utilizado em modelos onde o conjunto de dados é pequeno para treinar um modelo em escala real do zero. A utilização mais comum do TL é feita segundo os seguintes passos [70]:

1. Obter as camadas de um modelo pré-treinado.

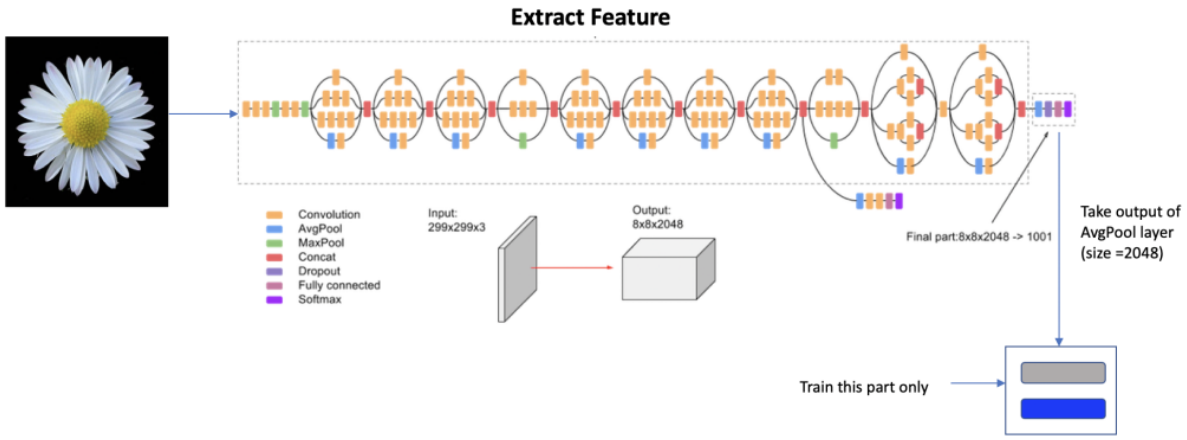

Figura 2.19: Transfer Learning usando a rede InceptinV3 [69]

2. Congelar estas camadas, evitando a destruição da informação durante futuros treinos.

3. Adicionar novas camadas treináveis em cima das camadas congeladas, transformando recursos antigos em previsões no novo conjunto de dados.

4. Treinar novas camadas para o conjunto de dados.

2.7 Trabalhos relacionados

Nesta secção serão descritas as análises aos trabalhos relacionados à classificação de doenças nas plantas utilizando Deep Learning e Transfer Learning. Neste sentido, foram encontrados dois trabalhos de relevância, sendo que na tabela 2.1 se encontram de forma resumida.

No trabalho de Daniel Bento [71] foram feitas diversas experiências com o dataset PlantVillage, com o objetivo de encontrar a melhor taxa de acerto, bem como a diminuição do tempo de treino. Sendo assim, a melhor performance foi obtida com a CNN Mobilenet, usando 10 épocas, um bacth size de 100 e duas camadas congeladas, com uma accuracy de 97,31%.

Por outro lado, o traballho realizado por Prasanna et al. [72] utiliza também o dataset

Título	**Autor**	**Ano**	**Dataset**	**Performance**
Deteção e identificação de doenças em plantas utilizando Deep Learning [71]	Bento, Daniel Carlos Pires Garcia Costa	2019	PlantVillage	Accuracy: 97,31% Ambiente real: -
Using Deep Learning for Image-Based Plant Disease Detection [72]	Prasanna Mohanty, Sharada; Hughes, David; Salathe, Marcel	2016	PlantVillage	Accuracy: 99,35% Ambiente real: 31%

Tabela 2.1: Título, autor, ano, dataset e performance dos trabalhos relacionados

PlantVillage, sendo os melhores resultados obtidos com a CNN GoogleNet, usando 80% de imagens para treino e 20% para teste, onde o treino é feito com 30 épocas e com um taxa de acerto de 99,35%. O teste em ambiente real é de apenas 31%, contudo na seleção aleatória, onde não é especificada a espécie da planta, reduz para apenas 2,6%. No entanto, é referido que um conjunto de dados mais diversificado é suficiente para aumentar a precisão.

O layout da aplicação móvel foi baseado no trabalho feito por Yannick Serge Obam [73], na qual é possível fazer a classificação das imagens das folhas recorrendo à fotografia da camera e da galeria, apresentando, também, a taxa de acerto de cada doença.

Capítulo 3

Desenvolvimento e Implementação

Neste capítulo, inicialmente são mostradas e justificadas as ferramentas e linguagens utilizadas, de seguida é descrito o processo de desenvolvimento do trabalho, constituído por três fases: construção do dataset de imagens, treino do modelo de classificação de imagens e desenvolvimento da aplicação móvel.

3.1 Ferramentas e Linguagens de programação

Nesta secção irão ser exemplificadas e justificadas as escolhas das linguagens de programação e respetivas bibliotecas usadas na dissertação.

3.1.1 GIMP

O software de edição de imagem GIMP é open-source e possibilita o uso de diferentes ferramentas e scripts desenvolvidos por terceiros. Esta ferramenta foi particularmente importante no recorte e padronização das imagens, tal como a criação de uma sombra para cada folha da planta, essencial no data-augmentation do dataset [74].

3.1.2 Python

A linguagem de programação python é de alto nível, interativa e que permite utilizar múltiplas e diferentes bibliotecas e scripts. Python é de código aberto, estável e muito abrangente o que permite encontrar documentação e exemplos rapidamente. É também, muito associada a áreas de IA [75]. Python é a principal linguagem suportada pela framework TensorFlow, que foi utilizada na fase de implementação de treino do modelo de classificação das imagens.

3.1.3 TensorFlow

O TensorFlow é uma plataforma de código aberto que permite o uso de técnicas de Machine Learning e Deep Learning, sendo abrangente e flexível, possibilitando o uso de diferentes bibliotecas e recursos de uma vasta comunidade de programadores [76].

O TensorFlow permite o uso em dispositivos móveis de forma fácil e interativa, havendo soluções para problemas recorrentes, tal como a classificação de imagens, que é o tema desta dissertação [76].

Esta biblioteca permite a execução através do meio de desenvolvimento do Google Colabs, permitindo que o treino do algoritmo seja feito online, não necessitando de um computador potente. O Google Colabs permite que a utilização de código de diferentes utilizadores e em várias plataformas viabilizando a opção de partilha de código.

3.1.4 Keras

Keras é uma API de alto nível funcionando no TensorFlow usada para DL, permitindo criar e treinar modelos. As principais vantagens de utilizar esta API é o facto de ser de fácil utilização, pois tem uma interface simples e consistente, fornecendo ao utilizador feedback claro e prático. A facilidade de expansão é outra das vantagens, pois permite criar camadas, métricas e funções de perda personalizados [77] [78].

3.1.5 Android Studio

O Android Studio é o software IDE oficial da Google, detentora da Android, que permite o desenvolvimento e programação de aplicações móveis. É um software bastante intuitivo e simples de usar havendo muitos recursos e exemplos que facilitam o seu uso. Possibilita o uso de diferentes linguagens como o Java e Kotlin, diferentes bibliotecas, tem um sistema auto-complete de código e um sistema de criação de conteúdos gráficos que facilita bastante antes da execução da aplicação [79].

3.1.6 Kotlin

Kotlin é uma linguagem de programação criada pela JetBrains em 2011, sendo estaticamente tipada, usada por mais de 60% dos programadores Android, aumentando a produtividade, satisfação e segurança do código. Kotlin é expressa e concisa, permitindo diminuir drasticamente a quantidade de código, segura, pois evita classes inteiras de erro como NullPointerExceptions, interoperável, aproveitando bibliotecas Java existentes [80].

3.2 Construção do dataset de imagens

Para o dataset de imagens, que serviu de base de dados de treino ao algoritmo de classificação de imagens, foram feitas duas tentativas de construção adquirindo imagens de diferentes modos. Assim, a primeira forma de adquirir as imagens foi com recurso a um scanner e a segunda através de fotografias tiradas por uma camera do telemóvel.

Neste sentido optou-se em primeiro lugar, pela digitalização das folhas da oliveira, pela sua rapidez de execução. Posteriormente, verificou-se que após alguns testes, em ambiente real, a classificação de folhas não tinha um resultado muito satisfatório. Assim sendo, foi feita a recolha de imagens das folhas com recurso à camera do telemóvel, com a palma da mão debaixo da folha, obtendo um dataset mais realista e com resultados melhores.

O dataset final possui imagens obtidas com recurso à fotografia e posterior data augmentation do mesmo. A seguir irão ser descritas a forma de construção do dataset dos diferentes modos explicados acima.

3.2.1 Dataset com recurso a scanner

O dataset foi construído segundo os seguintes passos:

1. Identificação da doença ou carência nutritiva na planta;
2. Recolha de folhas previamente seleccionadas e identificativas de cada patologia;
3. Digitalização das folhas para posterior obtenção de imagem;
4. Padronização das imagens, recorte com 256 pixeis em formato quadrado e fundo constante;
5. Data augmentation de cada imagem, pretendendo variações de luminosidade, contraste, cor, nitidez e sombras;

Recolha das folhas para o dataset

O dataset de imagens é constituído por um conjunto de fotografias de folhas de plantas doentes e saudáveis de várias variedades e formatos de forma a ter uma base relativamente abrangente.

Neste sentido, foi necessário identificar corretamente a patologia da planta e posteriormente foram recolhidas folha exemplificativas da mesma. O passo seguinte foi a digitalização visual das folhas e o seu recorte individualmente.

Padronização das imagens

A padronização das imagens consistiu no corte na proporção quadrada de 256×256 pixeis e a eliminação do seu fundo. O fundo foi eliminado de forma a se poder fazer o data augmentation mais realista, podendo assim, variar somente valores da folha e não

do seu fundo. Esta etapa foi executada com a ajuda da ferramenta removebg presente no site `https://www.remove.bg/`.

Data Augmentation

O data augmentation foi executado em relação ao dataset de imagens original para prevenir problemas de sobreajuste de dados aquando do reconhecimento das mesmas. Este processo seguiu-se ao recorte de 256 × 256 pixeis da folha em cada imagem e posterior remoção de fundo.

As variações de luminosidade, contraste, cor e nitidez foram obtidas através da biblioteca Python Imaging Library (PIL) da linguagem python. Os valores de variação foram testados previamente e dentro da mesma gama gerados de forma aleatória, de maneira a obter imagens bastante diferentes. Na tabela 3.1 está apresentado o processo de data augmentation e a respetiva variação da gama de valores.

Classe	**Processo**	**Valores**
ImageEnhance.Brightness(image)	Luminosidade	[0,5;2]
ImageEnhance.Contrast(image)	Contraste	[0,5;1,5]
ImageEnhance.Sharpness(image)	Nitidez	[-2;2]
ImageEnhance.Color(image)	Cor	[0,5;2,5]

Tabela 3.1: Processo de data augmentation

A geração de sombras na imagem foi feita com a ferramenta de GIMP e com o plugin BIMP que permitiu a edição de um conjunto de imagens simultaneamente e de forma automática.

Depois de cada variação de parâmetro inseriu-se uma fotografia da palma da mão como fundo em cada imagem. Na figura 3.1 é possível ver a imagem após a digitalização e depois com o data augmentation, neste caso variando a sombra, e acrescentando o fundo.

Com a execução desta etapa permitiu-se que cada classe com 200 imagens fosse expandida para 1000, ficando muito mais abrangente ganhando uma maior diversidade, facto

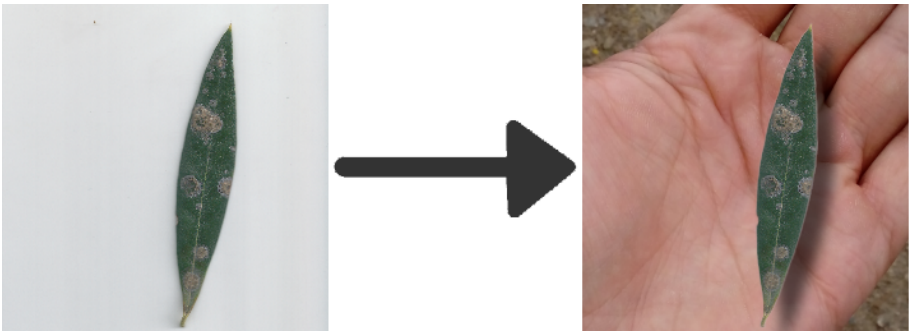

Figura 3.1: Imagem de folha de oliveira doente com olho de pavão, digitalizada à esquerda e à direita depois do data augmentation

importante para a execução da aplicação em ambiente real.

3.2.2 Dataset com recurso a fotografia

O dataset com resurso a fotografia foi construído segundo os seguintes passos:

1. Identificação da doença ou carência nutritiva na planta;
2. Recolha de folhas previamente seleccionadas e identificativas de cada patologia;
3. Fotografia da folha com a palma da mão por baixo, de forma a obter um fundo constante;
4. Padronização das imagens, recorte com 256 pixeis em formato quadrado;
5. Data augmentation de cada imagem, pretendendo variações de luminosidade, contraste, cor, nitidez e filtros CLAHE;

Estas etapas foram feitas de forma análoga à tentativa anterior, dataset com recurso a scanner. Para cada classe de imagens foram tiradas 350 fotografias. Após a aplicação do data augmentation, valores apresentados na tabela 3.2, foi possível obter 6 variações de cada imagem, sendo elas a original, luminosidade, contraste, cor, nitidez e aplicação de filtros CLAHE, estando no total 2100 imagens em cada classe.

Classe	Processo	Valores
ImageEnhance.Brightness(image)	Luminosidade	[0,9;1,5]
ImageEnhance.Contrast(image)	Contraste	[0,8;1,7]
ImageEnhance.Sharpness(image)	Nitidez	[-2;2]
ImageEnhance.Color(image)	Cor	[0,5;2]
cv2.createCLAHE	CLAHE	clipLimit=1.2, tileGridSize=(8,8)

Tabela 3.2: Processo de data augmentation da fotografia

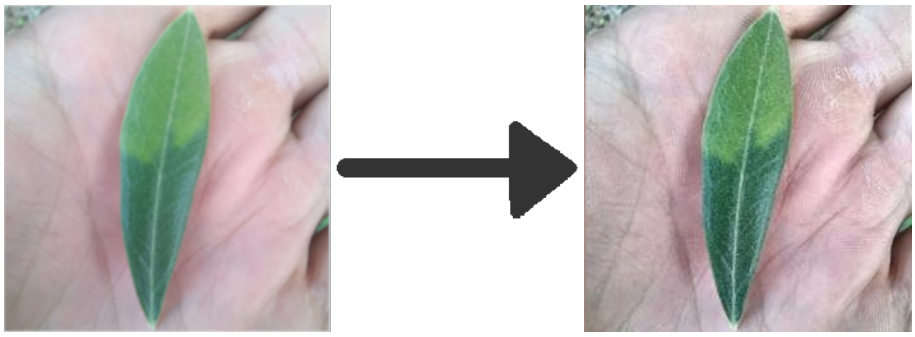

Figura 3.2: Fotografia original de folha com carência de boro à direita e posterior aplicação de filtro CLAHE

3.3 Treino do modelo de classificação de imagens

A fase de identificação e treino do modelo de classificação das imagens constitui uma das fases mais importantes no desenvolvimento de todo o projeto. O modelo utilizado nesta dissertação assenta na metodologia de Transfer Learning, que permite reaproveitar modelos desenvolvidos. Neste caso foi utilizado o modelo MobileNetV2 disponibilizado pelo TensorFlow. Este modelo pré treinado utiliza o conjunto de imagens ImageNet e foi executado de forma online através da ferramenta do Google Colab.

O modelo utiliza o dataset, que foi previamente construído como mostrado na secção anterior, e os repetivos conjuntos de treino e validação. A percentagem de imagens para o conjunto de treino é cerca de 70%, sendo as restantes 30% utilizadas no conjunto de

validação. Inicialmente, as imagens são convertidas para cores RGB (com 3 canais de cores) e redimensionadas para 224×224 pixeis, valor padrão do algoritmo utilizado, criadas de forma automática. Através do método `image.ImageDataGenerator` e utilizando um tamanho de lote (batch size) de 64 do TensorFlow, as classes treino e validação são criadas com os respetivos rótulos de identificação.

Posteriormente, é criado o modelo pré treinado, que utiliza o conjundo de dados ImagetNet, que servirá de base para a extração de recursos para o modelo de classificação de imagens. Na tabela 3.3 é possível ver a instanciação do modelo com os pesos treinados no ImageNet carregando a rede que não inclui a camada de output, pois é ideal para a extração de recursos, e são definidos, também, a resolução das imagens a ser utilizadas.

Parâmetros de entrada MobileNet V2	**Valores**
Input Shape	(224, 224, 3)
Include Top Layers	False
Weights	imagenet

Tabela 3.3: Parâmetros de entrada da MobileNet V2

A extração de características é feita de seguida, onde é construída a classificação, através da adição da camada de convolução 2D, exemplificadas na tabela 3.4. A camada de convulsão `tf.keras.layers.Conv2D` utiliza 18 filtros, sendo a dimensão do kernel 3, e a função relu como função de ativação. É adicionada de seguida uma camada de dropout com o valor 0.1, permitindo diminuir o possível overfitting do modelo. A camada de pooling utilizada é obtida através do método `tf.keras.layers.GlobalAveragePooling2D()`, pertendendo reduzir a dimensão do feature map obtido com a camada de convulsão 2D. Finalmente, a última camada densamente conetada possui 4 saídas, correspondentes às 4 classes, sendo a função de ativação `softmax`, adicionando, também, a regularização L2 com valor de 0,001.

Camadas	**Parâmetros**	**Valores**
Conv2D	filters	18
	kernel size	3
	activation	relu
Dropout	rate	0,1
GlobalAveragePooling2D	-	-
Dense	units	4
	activation	softmax
	kernel_regularizer	regularizers.l2(0.001)

Tabela 3.4: Parâmetros da rede neuronal proposta

Na figura 3.3 está representado o modelo compilado, após a optimização Adam, que utiliza uma taxa de aprendizagem de 0,001, sendo possível observar o número de parâmetros totais, treináveis e não treináveis.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 5, 5, 18)          207378
_________________________________________________________________
dropout_16 (Dropout)         (None, 5, 5, 18)          0
_________________________________________________________________
global_average_pooling2d_16  (None, 18)                0
_________________________________________________________________
dense_16 (Dense)             (None, 4)                 76
=================================================================
Total params: 2,465,438
Trainable params: 207,454
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figura 3.3: Parâmetros e características treináveis e não treináveis no algoritmo

O treino do modelo é feito de forma imediata, sendo possível observar a taxa de acerto (accuracy) à medida que decorre cada execução de cada época. Neste caso, foram executadas 8 épocas, valor que foi sendo testado até obter um resultado satisfatório, com

92 nós em cada, gerados de forma automática. Na figura 3.4 é possível observar a taxa de acerto do modelo, sendo cerca de 99%, tanto no conjunto de treino como no conjunto de validação.

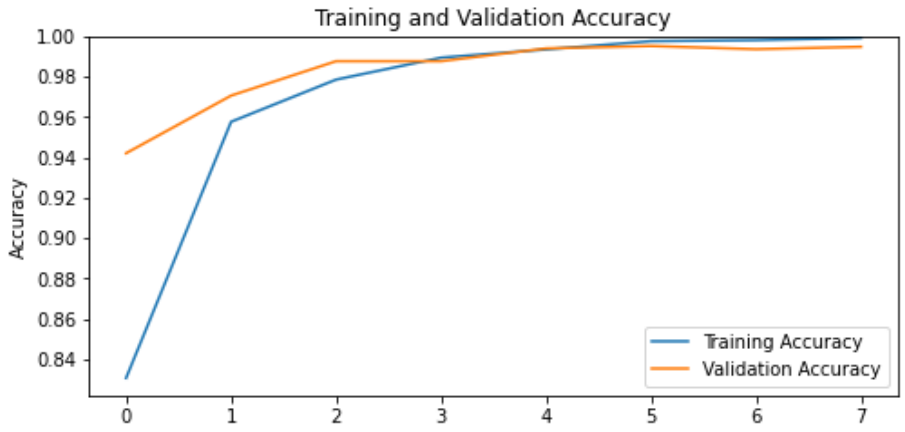

Figura 3.4: Taxa de acerto do modelo

Na figura 3.5 é possível observar perdas do modelo treinado, cerca de 1% no conjunto de treino, sendo que no conjunto de validação são aproximadamente 3%.

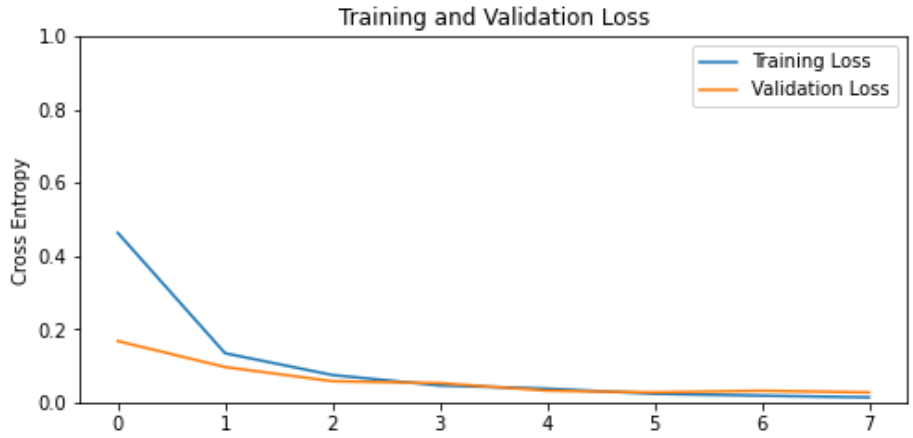

Figura 3.5: Percentagem de perdas do modelo à medida do treino

Conjunto	**Perdas**	**Accuracy**
Treino	1,35%	99,93%
Validação	2.71%	99.48%

Tabela 3.5: Perdas e taxa de acerto do modelo exatas

Finalmente, o modelo treinado e exportado no formato tflite, tal como os rótulos das classes de imagens utilizadas no treino. Estes ficheiros farão parte da aplicação que irá classificar as imagens.

3.4 Desenvolvimento da aplicação móvel

A aplicação móvel, tal como referido no capítulo anterior, foi baseada no trabalho desenvolvido por Yannick Serge Obam [73]. É importante ressaltar que a mesma se encontra programada em Kotlin, funcionando apenas para o sistema Android. A aplicação possui algumas alterações comparada com a original, ao nível de design, bem como informações adicionais relativas à classificação das folhas, estando disponível através do seguinte link: `https://bit.ly/2T7Gsvz`

A aplicação possui uma classe principal (MainActivity), que contém as funções relativas ao funcionamento dos botões da mesma, e uma classe secundária (Classifier), possuindo funções relativas à classificação das imagens. Na figura 3.6 está representado o protótipo da aplicação, onde é possível observar os três botões, sendo eles o "CAMERA", "PHOTO", "DETECT".

- O botão "CAMERA" é responsável por abrir a camera do telemóvel para tirar uma fotografia.
- O botão "PHOTO" é responsável por abrir a galeria do telemóvel.
- O botão "DETECT" é responsável por classificar a imagem obtida através da câmera ou galeria.

Assim, na classe Classifier as imagens obtidas através da camera ou galeria são redimensionadas para 224×224 pixeis. As imagens obtidas com a camera são convertidas para um buffer, não sendo guardadas no telemóvel. Estas são posteriormente classificadas através da interpretação do ficheiro tflite, que contém os dados do modelo treinado, com a imagem obtida. Posteriormente, é mostrado no ecrã a percentagem de acerto da

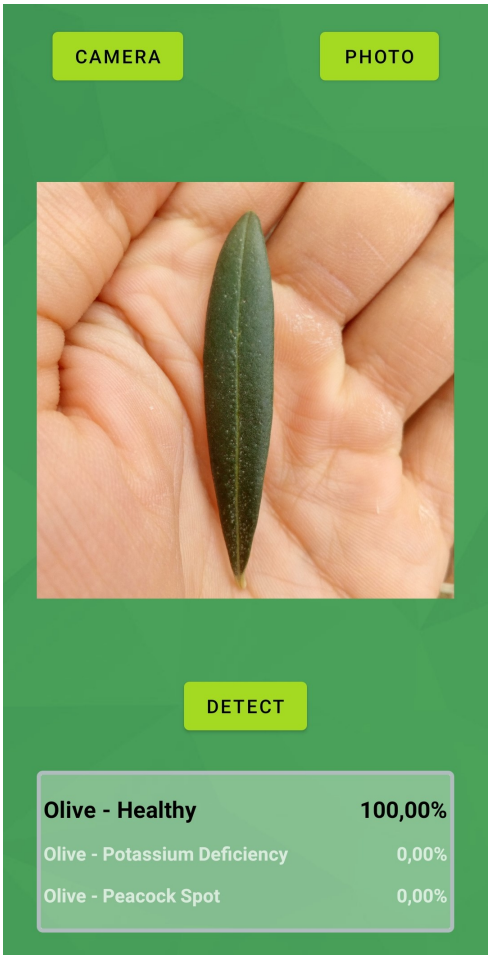

Figura 3.6: Protótipo da aplicação

classe mais provável, estando organizado de forma decrescente da taxa de probabilidade de acerto do modelo, apresentando as três mais prováveis.

O adequado manuseamento e utilização da aplicação móvel potencia a correta precisão na classificação das classes de imagens, neste caso a classificação de doenças ou a sua ausência na oliveira. Sendo assim, a fotografia ou imagem a classificar deverá conter a folha da oliveira, virada para cima, na palma da mão. A imagem correta deverá apresentar exclusivamente uma só folha, servindo a palma da mão como fundo, não devendo estar visíveis outros planos, de uma forma muito abrangente.

A uso da palma da mão como fundo deve-se ao facto de após alguns testes ser notório que o uso de um fundo constante na folha potenciaria a precisão do modelo treinado.

Consequentemente, sendo uma aplicação móvel que pretende ser usada no exterior junto da oliveira, à qual se deseja classificar as possíveis doenças, o uso da palma da mão como fundo irá otimizar não só a precisão do modelo, mas também o uso da aplicação, pois o utilizador poderá agarrar a folha com uma mão e fotografá-la com a outra.

A tentativa de utilização da aplicação que não esteja nos padrões acima apresentados, poderá apresentar resultados que não reproduzam a realidade, pois o modelo apenas foi treinado para suportar a classificação de imagens nos moldes descritos. Outro ponto importante, é o facto da solução desenvolvida apenas suportar a classificação de quatro classes (oliveira saudável, carência de boro, carência de potássio, olho de pavão), sendo que numa tentativa de classificação de uma doença que não seja uma destas é mostrado a taxa de acerto da classe mais provável.

Por outro lado, tal como referido em capítulos anteriores, é de extrema importância a identificação correta de uma doença ou carência nutritiva no seu estado inicial, de forma a minimizar os estragos provocados. Desta forma, a solução construída classifica de forma correta patologias nos estados iniciais, referidas nas figuras 2.2, 2.3, 2.4, não garantindo a classificação de uma doença num estado mais evoluído.

Na imagem 3.7 está apresentada uma folha de oliveira saudável fotografada corretamente, estando bastante visível e nítida, denotando-se, neste caso, a ausência de anomalias na mesma.

Figura 3.7: Imagem de uma folha de oliveira fotografada corretamente

A imagem 3.8 representa uma folha de oliveira com a presença de olho de pavão fotografada incorretamente, sendo possível verificar a presença de um terceiro plano, por baixo da palma da mão, em grande percentagem. Neste caso, sendo que a folha está um bocado longe da camera não é possível observar de forma clara a presença das manchas típicas da doença ollho de pavão na folha da oliveira.

Figura 3.8: Imagem de uma folha de oliveira fotografada de forma incorreta

Na imagem 3.9 é possível verificar a presença de olho de pavão na folha de oliveira em estado avançado. Neste caso, a aplicação não garante a correta identificação da doença, pois não foi treinada para tal.

Figura 3.9: Imagem de uma folha de oliveira com olho de pavão no estado avançado

Capítulo 4

Análise de Resultados

O capítulo 4 apresenta e analisa os resultados relativos à classificação de doenças ou a sua ausência nas plantas, nomeadamente a oliveira, em ambiente real, testando a fiabilidade da solução desenvolvida.

4.1 Teste da aplicação móvel em ambiente real

Os testes realizados em ambiente real consistiram na utilização de um smartphone, com a aplicação desenvolvida, fotografando folhas de oliveira, previamente identificadas relativamente à presença das doenças que esta dissertação se propõe a classificar.

O dataset utilizado no treino do modelo, para a classificação de imagens, que serviu de base à aplicação, foi o construído com recurso à camera.

A arquitetura CNN utilizada para treino do modelo foi a MobileNetV2, usando um conjunto de dados ImageNet, visto se tratar de um modelo que serviu de base sendo bastante fiável e rápido, com taxas de precisão elevadas.

Neste sentido foi efetuado um teste em ambiente real, que serviu de base à análise de resultados que aqui irá ser feita. Na imagem 4.1 é possível observar a matriz de confusão que contém os resultados do teste realizado em ambiente real.

Neste teste foram utilizadas 30 imagens de cada uma das classes, sendo elas representadas por:

Classe Real		Classe Prevista				
		Bor	Hea	Pea	Pot	
	Bor	21 **17,50%**	4 3,33%	2 1,67%	3 2,50%	30 70,00% 30,00%
	Hea	1 0,83%	26 **21,67%**	3 2,50%	0 0,00%	30 86,67% 13,33%
	Pea	0 0,00%	1 0,83%	29 **24,17%**	0 0,00%	30 96,67% 3,33%
	Pot	0 0,00%	1 0,83%	2 1,67%	27 **22,50%**	30 90,00% 10,00%
		22 95,45% 4,55%	32 81,25% 18,75%	36 80,56% 19,44%	30 90,00% 10,00%	120 **85,83%** 14,17%

Figura 4.1: Matriz de confusão relativa aos testes em ambiente real

- Bor - Oliveira com carência de boro
- Hea - Oliveira saudável
- Pea - Oliveira com olho de pavão
- Pot - Oliveira com carência de potássio

Através da matriz de confusão é possível retirar o valor da taxa de acerto do modelo, sendo ele de 85,83%, que é um resultado bastante bom. Por outro lado, é possível observar na última linha os valores da precisão em cada classe, sendo que todos são superiores a 80%. Os valores apresentados na última coluna dizem respeito à sensibilidade de cada classe.

Com estes valores é possível concluir que quando a aplicação classifica uma folha como sendo da classe Pea (classe com a precisão mais baixa), pode-se deduzir que cerca de 80% dos resultados apresentados estão corretos. Fazendo a mesma analogia à sensibilidade da classe com os resultados mais baixos, classe Bor, pode-se concluir que quando uma folha com carência de boro é apresentada à aplicação, esta tende a classificá-la corretamente 70% das vezes.

Com os resultados da precisão e sensiblidade de cada classe e recorrendo à equação 2.6, descrita anteriormente, é possível obter a medida F1. Esta medida que mede a fiabilidade do modelo e que quanto mais próxima de um mais fiável será, obteve-se os seguintes resultados em cada classe:

- F1 Bor - 0,81
- F1 Hea - 0,84
- F1 Pea - 0,88
- F1 Pot - 0,90

Neste sentido, é possível observar que os valores se encontram relativamente próximos de 1, sendo portanto uma solução bastante fiável em todas as classes.

4.2 Análise detalhada de cada classe

A análise detalhada consiste em observar os resultados obtidos na classificação de cada classe. Assim, quando a classe real é a Bor a aplicação classifica-a em 70% das vezes corretamente, sendo que em 30 testes realizados, 21 previsões foram acertadas. Olhando atentamente à figura A.1 é possível observar que destes 21 acertos, a média de previsão situa-se nos 96,39%, um valor muito elevado. Por outro lado, quando é feita uma previsão incorreta, os valores de precisão apresentados são mais dispersos, onde das 9 classificações 3 apresentam-se abaixo dos 85% (acurácia do modelo). Com estes dados, se houver um

threshold (limite) de 85%, a precisão da classe Bor sobe para 76,92%, pois as imagens 5,9,15,25, apresentam-se com valores inconclusivos.

A figura A.2 mostra a taxa de acerto na classificação da classe Hea. Através desta figura é possível concluir que dos 30 testes realizados, 26 foram previstos de forma correta, sendo a precisão de 86,67%. Do número de acertos resulta uma média de 94,30%, sendo que apenas 4 destas imagens classificadas possuem um valor abaixo da média. Para um threshold de 85%, a precisão nesta classe aumentaria para 91,67%, pois 4 imagens classificadas corretamente e 2 classificadas incorretamente (oli-hea___0011,12,15,16,19,20,30) possuem um valor abaixo do limite.

Na figura A.3 é possível observar a taxa de precisão da classe Pea, sendo de 96,67%, dos 30 testes realizados apenas um não foi classificado corretamente. Nesta figura, tal como nas anteriores o valor médio da precisão das classes corretas é de 97,91%. Por outro lado, com um threshold de 85% a precisão diminui ligeiramente, pois existem duas imagens (oli-pea___0005,0008) que não possuem um valor de confiança superior para este limite.

A figura A.4 apresenta os resultados da previsão da classe Pot, onde dos 30 testes realizados a precisão foi de 90%. Do número de imagens classificadas corretamente, 27, a média de acerto é de 99,46% um valor muito elevado, sendo todos os valores acima 90% com um desvio padrão de 1,61%. Com um threshold de 85% a precisão é de 96,43%, pois duas das imagens classificadas de forma incorreta têm um valor de confiança abaixo do limite.

Com a análise feita acima é possível concluir que se for utilizado um threshold de 85%, valor considerado muito bom para testes em ambiente real, a accuracy do modelo e a precisão em cada classe aumenta. Na tabela 4.1 é possível comparar os valores obtidos nos testes realizados e o valor obtido com limite de confiança.

Threshold	**Accuracy**
Sem threshlod	85,83%
85%	91,43%

Tabela 4.1: Comparação entre o valor de precisão do modelo testado e a taxa de precisão obtida com limite de confiança de 85%

Capítulo 5

Conclusões e Trabalhos Futuros

Neste capítulo 5 serão apresentados as conclusões desta dissertação e sugeridas sugestões para possíveis futuros trabalhos.

5.1 Conclusões

Esta dissertação apresentou uma solução para a classificação de doenças nas plantas, nomeadamente a oliveira, utilizando Transfer Learning numa aplicação móvel.

Assim, após a construção do dataset com recurso à fotografia foi possível obter resultados bastante satisfatórios em ambiente real, sendo a taxa de acerto 85,83%. Com os valores obtidos é possível verificar que nas quatro classes a serem classificadas os valores de sensibilidade são superiores a 70% e na precisão superiores a 80% em todas as classes.

Com a análise feita ao detalhe da experiência realizada em ambiente real foi possível verificar que quando a aplicação classificou de forma incorreta, 17 em 120 fotografias analisadas, 7 destas apresentam um valor abaixo de um possível limite de confiança a 85%. Com este threshold a 85% a accuracy da solução apresentada aumentaria para 91,43%, valor razoavelmente bom, quando comparado com a precisão do modelo treinado, 99%.

De um modo geral, concluiu-se que o uso de metodologias de Transfer Learning aplicadas a problemas reais, como a classificação de doenças nas plantas, é bastante fiável, pois é uma solução de baixo custo, simples e rápida, que poderá auxiliar os agricultores

a terem melhores rentabilidades nas suas explorações.

5.2 Trabalhos Futuros

Esta dissertação teve como resultado uma solução que poderá ser alvo de futuros desenvolvimentos, como a inclusão de imagens de outras partes das plantas que sejam relevantes numa análise visual de uma possível doença ou carência nutritiva das classes apresentadas, no dataset treinado. Esta sugestão de utilizar um dataset mais abrangente aumentaria a precisão em ambiente real, pois o utilizador poderia classificar várias partes de uma planta, que se mostrassem doentes, tal como o fruto ou ramos, com objetivo de aumentar a certeza do modelo treinado, visto que estariam a ser analisados vários fatores. Outra sugestão seria alargar o conjunto de classes de plantas e patologias.

Relativamente à aplicação móvel, uma mais valia seria disponibilizar a análise em tempo real, ou seja, usando a câmara, mas como vídeo, podendo o utilizador obter diferentes perspectivas da classe a ser analisada, com os resultados a serem mostrados automaticamente. A melhoria da interface gráfica e o acrescento de menus com informações pertinentes seria relevante para o utilizador.

Bibliografia

[1] Worldometer. (mai. de 2020). Portugal Population (LIVE), URL: https://www.worldometers.info/world-population/portugal-population/.

[2] c. CIM-TTM, *Plano Estratégico de desenvolvimento intermunicipal das Terras de Trás Montes para o período 2014-2020*, dez. de 2014. URL: https://www.cim-ttm.pt/cimttm/uploads/document/file/96/plano_estrategico_de_desenvolvimento_intermunicipal_das_terras_de_tras_os_montes__2014_2020.pdf.

[3] M. Â. Rodrigues e C. M. Correia, "Manual da safra e contra a safra do olival", *Bragança: Instituto*, 2009.

[4] GPP. (2020). Ministério da Agricultura assinala Ano Internacional da Sanidade Vegetal, URL: https://www.gpp.pt/index.php/noticias/ministerio-da-agricultura-assinala-ano-internacional-da-sanidade-vegetal.

[5] E. Molina, "Fertilización y nutrición de pejibaye para palmito", Research report, Centro de Investigaciones Agronómicas. Universidade de ..., rel. téc., 1997.

[6] R. Fernandez-Escobar, R. Moreno e M. Garcıa-Creus, "Seasonal changes of mineral nutrients in olive leaves during the alternate-bearing cycle", *Scientia Horticulturae*, vol. 82, n.º 1-2, pp. 25–45, 1999.

[7] Rodrigues e M. Arrobas, "Principais problemas nutricionais dos olivais transmontanos e que quantidades de fertilizantes aplicar", 2013.

[8] P. Jordão, "Boas práticas no olival e no lagar", *P. Jordão, Boas Práticas no Olival e no Lagar*, pp. 56–65, 2014.

[9] INE, *Contas Económicas da Agricultura 2019*, dez. de 2019. URL: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=354592855&DESTAQUESmodo=2&xlang=pt.

[10] DN/Lusa. (2018). Peso de Trás-os-Montes na produção de azeite cai para quase metade numa década, URL: https://www.dn.pt/dinheiro/peso-de-tras-os-montes-na-producao-de-azeite-cai-para-quase-metade-numa-decada-9042142.html.

[11] A. G. e Elder Lima Leite, *Olival - Nutrição e Sanidade das Culturas*, first. Agrobook, mai. de 2018.

[12] F. C. Pedro Jordão Maria da Encarnação Marcelo, "A importância de boro no olival", em *olivicultura*, set. de 2019, p. 3.

[13] IRIAF, *BOLETIN FITOSANITARIO DE AVISOS E INFORMACIONES*, 2ª ed., ESTACIÓN REGIONAL DE AVISOS AGRÍCOLAS / SANIDAD VEGETAL – C.I.A.G. EL CHAPARRILLO (IRIAF), Ctra. de Porzuna s/n -13071 CIUDAD REAL, fev. de 2020.

[14] J. McCarthy, "What is artificial intelligence?", 1998.

[15] A. Kaplan e M. Haenlein, "Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence", *Business Horizons*, vol. 62, n.º 1, pp. 15–25, 2019.

[16] C. Nikolopoulos, *Expert systems: introduction to first and second generation and hybrid knowledge based systems*. Marcel Dekker, Inc., 1997.

[17] M. A. Sellitto, "Inteligência artificial: uma aplicação em uma indústria de processo contínuo", *Gestão & Produção*, vol. 9, n.º 3, pp. 363–376, 2002.

[18] D. S. Brigade. (ago. de 2020). A Diferença Entre Inteligência Artificial, Machine Learning e Deep Learning, URL: https://medium.com/data-science-brigade/a-diferen%5C%C3%5C%A7a-entre-intelig%5C%C3%5C%AAncia-artificial-machine-learning-e-deep-learning-930b5cc2aa42.

[19] S. Haykin, *Redes neurais: princípios e prática*. Bookman Editora, 2007.

[20] Z. Liu, C. Peng, W. Xiang, D. Tian, X. Deng e M. Zhao, "Application of artificial neural networks in global climate change and ecological research: An overview", *Chinese science bulletin*, vol. 55, n.º 34, pp. 3853–3863, 2010.

[21] I. A. Basheer e M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", *Journal of microbiological methods*, vol. 43, n.º 1, pp. 3–31, 2000.

[22] J. D. Miller e R. M. Forte, *Mastering Predictive Analytics with R*. Packt Publishing Ltd, 2017.

[23] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, 2015, vol. 2018.

[24] S. SHARMA. (set. de 2019). What the Hell is Perceptron? | The Fundamentals of Neural Networks, URL: `https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53`.

[25] E. Ferneda, "Redes neurais e sua aplicação em sistemas de recuperação de informação", *Ciência da Informação*, vol. 35, n.º 1, pp. 25–30, 2006.

[26] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms", Cornell Aeronautical Lab Inc Buffalo NY, rel. téc., 1961.

[27] *Multi-layer Perceptron in TensorFlow - Javatpoint*, `https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow`, (Accessed on 2020-08-26).

[28] D. J. Matich, "Redes Neuronales: Conceptos básicos y aplicaciones", *Universidad Tecnológica Nacional, México*, 2001.

[29] S. Lingireddy e G. M. Brion, *Artificial neural networks in water supply engineering*. ASCE Publications, 2005.

[30] *What is Perceptron*, `https://www.simplilearn.com/what-is-perceptron-tutorial`, (Accessed on 2020-08-26).

[31] H. B. Curry, "The method of steepest descent for non-linear minimization problems", *Quarterly of Applied Mathematics*, vol. 2, n.º 3, pp. 258–261, 1944.

[32] C. Lemaréchal, "Cauchy and the gradient method", *Doc Math Extra*, vol. 251, p. 254, 2012.

[33] L. F. B. da Silva, L. N. Utimura, K. A. P. da Costa, M. A. Z. Meira, S. d. G. D. Prado et al., "Estudo sobre Técnicas de Machine Learning para Detecção de Botnets", *IEEE Latin America Transactions*, vol. 18, n.º 5, pp. 881–888, 2020.

[34] A. L. Samuel, "Some studies in machine learning using the game of checkers. II—recent progress", em *Computer Games I*, Springer, 1988, pp. 366–400.

[35] P. Domingos, *O algoritmo mestre: como a busca pelo algoritmo de machine learning definitivo recriará nosso mundo*. Novatec Editora, 2017.

[36] P. Langley, *The changing science of machine learning*, 2011.

[37] *O que é machine learning? | Ironhack Blog*, `https://www.ironhack.com/br/data-analytics/o-que-e-machine-learning`, (Accessed on 2020-08-24).

[38] S. J. Russell e P. Norvig, *Artificial intelligence: a modern approach. Malaysia*, 2016.

[39] Y. Bengio, A. Courville e P. Vincent, "Representation learning: A review and new perspectives", *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, n.º 8, pp. 1798–1828, 2013.

[40] M. Paluszek e S. Thomas, "Machine Learning Examples in MATLAB", em *MATLAB Machine Learning*, Springer, 2017, pp. 85–88.

[41] C. H. C. Ribeiro, "A tutorial on reinforcement learning techniques", em *Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks*, 1999.

[42] I. Goodfellow, Y. Bengio, A. Courville e Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[43] L. Deng e D. Yu, *Deep learning: methods and applications. Foundations and Trends (R) in Signal Processing, 7 (3-4), 197-387*, 2014.

[44] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[45] *What Is Deep Learning?*, `https://www.mathworks.com/discovery/deep-learning.html`, (Accessed on 2020-08-31).

[46] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[47] H.-D. Wehle, "Machine learning, deep learning, and ai: What's the difference?", em *International Conference on Data scientist innovation day, Bruxelles, Belgium*, 2017.

[48] A. C. G. Vargas, A. Paes e C. N. Vasconcelos, "Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres", em *Proceedings of the xxix conference on graphics, patterns and images*, vol. 1, 2016.

[49] S. Khan, H. Rahmani, S. A. A. Shah e M. Bennamoun, "A guide to convolutional neural networks for computer vision", *Synthesis Lectures on Computer Vision*, vol. 8, n.º 1, pp. 1–207, 2018.

[50] A. Deshpande. (jul. de 2016). A Beginner's Guide To Understanding Convolutional Neural Networks, URL: `https://adeshpande3.github.io/A-Beginner%5C%27s-Guide-To-Understanding-Convolutional-Neural-Networks/`.

[51] H. H. Aghdam e E. J. Heravi, "Guide to convolutional neural networks", *New York, NY: Springer*, vol. 10, pp. 978–973, 2017.

[52] R. Yamashita, M. Nishio, R. K. G. Do e K. Togashi, "Convolutional neural networks: an overview and application in radiology", *Insights into imaging*, vol. 9, n.º 4, pp. 611–629, 2018.

[53] U. Karn, "An intuitive explanation of convolutional neural networks", *The data science blog*, 2016.

[54] *Overfit e underfit | TensorFlow Core*, `https://www.tensorflow.org/tutorials/keras/overfit_and_underfit`, (Accessed on 2020-10-03).

[55] *Overfitting and Underfitting (BUG IN ML MODELS) – mc.ai*, https://mc.ai/overfitting-and-underfitting-bug-in-ml-models/, (Accessed on 2020-10-03).

[56] A. K. Jain, J. Mao e K. M. Mohiuddin, "Artificial neural networks: A tutorial", *Computer*, vol. 29, n.º 3, pp. 31–44, 1996.

[57] S. Kim, J. Jeon e I. Eom, "Image contrast enhancement using entropy scaling in wavelet domain", *Signal Processing*, vol. 127, pp. 1–11, out. de 2016.

[58] S. Kumar, *Data Augmentation Increases Accuracy of your model*, https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722, (Accessed on 2020-10-03), 2019 de jun.

[59] M. Hossin e M. Sulaiman, "A review on evaluation metrics for data classification evaluations", *International Journal of Data Mining & Knowledge Management Process*, vol. 5, n.º 2, p. 1, 2015.

[60] A. J. Casson, E. Luna e E. Rodriguez-Villegas, "Performance metrics for the accurate characterisation of interictal spike detection algorithms", *Journal of neuroscience methods*, vol. 177, n.º 2, pp. 479–487, 2009.

[61] S. Rothe, S. Wirtz e D. Söffker, "About the reliability of diagnostic statements: fundamentals about detection rates, false alarms, and technical requirements", nov. de 2016.

[62] Y. Li, H. Huang, Q. Xie, L. Yao e Q. Chen, "Research on a surface defect detection algorithm based on MobileNet-SSD", *Applied Sciences*, vol. 8, n.º 9, p. 1678, 2018.

[63] L. Sifre e S. Mallat, "Rigid-motion scattering for texture classification", *arXiv preprint arXiv:1403.1687*, 2014.

[64] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov e L. Chen, "Mobilenetv2: The next generation of on-device computer vision networks", *URL https://ai. googleblog. com/2018/04/mobilenetv2-next-generation-of-on. html*, 2018.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke e A. Rabinovich, "Going deeper with convolutions", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens e Z. Wojna, "Rethinking the inception architecture for computer vision", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[67] J. West, D. Ventura e S. Warnick, "Spring research presentation: A theoretical foundation for inductive transfer", *Brigham Young University, College of Physical and Mathematical Sciences*, vol. 1, n.º 08, 2007.

[68] M. Oquab, L. Bottou, I. Laptev e J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks", em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[69] *Transfer Learning – Using Inception V3 for developing Image Classifier (Part 2) – Alquarizm*, `https://alquarizm.wordpress.com/2019/03/11/transfer-learning-using-inception-v3-for-developing-image-classifier-part-2/`, (Accessed on 2020-10-08).

[70] F. Chollet. (abr. de 2020). Transfer learning & fine-tuning, URL: `https://keras.io/guides/transfer_learning/#setup`.

[71] D. C. P. G. C. Bento, "Deteção e identificação de doenças em plantas utilizando Deep Learning", tese de doutoramento, 2019.

[72] S. P. Mohanty, D. P. Hughes e M. Salathé, "Using deep learning for image-based plant disease detection", *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[73] Y. S. Obam. (jul. de 2020). Plant Disease Classification with TensorFlow Lite on Android Part 2, URL: `https://medium.com/@yannicksergeobam/plant-disease-classification-with-tensorflow-lite-on-android-part-2-c2d47371cea3`.

[74] *GIMP - GNU Image Manipulation Program*, `https://www.gimp.org/`, (Accessed on 2020-10-02).

[75] *O Tutorial Python — documentação Python 3.8.6*, https://docs.python.org/pt-br/3/tutorial/, (Accessed on 2020-10-02).

[76] *TensorFlow*, https://www.tensorflow.org/, (Accessed on 2020-10-02).

[77] *Keras*, https://www.tensorflow.org/guide/keras/, (Accessed on 2020-09-30).

[78] *Keras: the Python deep learning API*, https://keras.io/, (Accessed on 2020-09-30).

[79] *Android Studio*, https://developer.android.com/studio, (Accessed on 2020-09-30).

[80] *Develop Android apps with Kotlin*, https://developer.android.com/kotlin, (Accessed on 2020-10-14).

Apêndice A

Detalhes dos testes em ambiente real

A.1 Análise detalhada aos testes da classe Bor, Hea, Pea e Pot

A primeira coluna representa o nome da fotografia, e as restantes a taxa de acerto em cada classe.

Análise da classe real Bor

	Classe prevista			
Imagem	**Bor**	**Hea**	**Pea**	**Pot**
oli-bor___0001	99,56%	0,00%	0,00%	0,44%
oli-bor___0002	2,87%	0,03%	0,00%	97,10%
oli-bor___0003	5,35%	0,08%	0,00%	94,57%
oli-bor___0004	2,40%	0,01%	0,00%	97,60%
oli-bor___0005	19,35%	66,74%	8,74%	0,00%
oli-bor___0006	85,23%	14,57%	0,00%	0,17%
oli-bor___0007	88,72%	11,16%	0,00%	0,12%
oli-bor___0008	7,22%	92,78%	0,00%	0,00%
oli-bor___0009	54,82%	43,29%	0,00%	1,89%
oli-bor___0010	0,00%	14,67%	85,09%	0,16%
oli-bor___0011	99,99%	0,01%	0,00%	0,00%
oli-bor___0012	100,00%	0,00%	0,00%	0,00%
oli-bor___0013	100,00%	0,00%	0,00%	0,00%
oli-bor___0014	99,20%	0,02%	0,00%	0,78%
oli-bor___0015	21,40%	4,27%	74,01%	0,00%
oli-bor___0016	99,99%	0,01%	0,00%	0,00%
oli-bor___0017	100,00%	0,00%	0,00%	0,00%
oli-bor___0018	98,49%	1,21%	0,30%	0,00%
oli-bor___0019	100,00%	0,00%	0,00%	0,00%
oli-bor___0020	100,00%	0,00%	0,00%	0,00%
oli-bor___0021	99,44%	0,56%	0,00%	0,00%
oli-bor___0022	99,50%	0,00%	0,00%	0,50%
oli-bor___0023	0,00%	96,52%	0,20%	3,24%
oli-bor___0024	99,44%	0,56%	0,00%	0,00%
oli-bor___0025	4,47%	69,25%	0,00%	24,77%
oli-bor___0026	99,71%	0,11%	0,00%	0,17%
oli-bor___0027	100,00%	0,00%	0,00%	0,00%
oli-bor___0028	100,00%	0,00%	0,00%	0,00%
oli-bor___0029	100,00%	0,00%	0,00%	0,00%
oli-bor___0030	100,00%	0,00%	0,00%	0,00%

Total	30
Corretas	21
Incorretas	9
Sensibilidade	70,00%
Média de acerto	96,39%
Desvio padrão	10,04%

Tabela A.1: Análise da sensibilidade da classe Bor (carência de boro)

Análise da classe real Hea

	Classe prevista			
Imagem	**Bor**	**Hea**	**Pea**	**Pot**
oli-hea___0001	0,00%	100,00%	0,00%	0,00%
oli-hea___0002	0,01%	99,90%	0,00%	0,00%
oli-hea___0003	0,00%	99,98%	0,02%	0,00%
oli-hea___0004	0,00%	90,99%	9,01%	0,00%
oli-hea___0005	0,00%	99,92%	0,04%	0,03%
oli-hea___0006	0,00%	98,92%	1,01%	0,07%
oli-hea___0007	0,00%	99,60%	0,08%	0,30%
oli-hea___0008	0,00%	100,00%	0,00%	0,00%
oli-hea___0009	0,00%	99,77%	0,01%	0,21%
oli-hea___0010	0,01%	99,99%	0,00%	0,00%
oli-hea___0011	0,00%	64,08%	38,78%	0,07%
oli-hea___0012	60,89%	39,11%	0,00%	0,00%
oli-hea___0013	0,00%	99,99%	0,01%	0,00%
oli-hea___0014	2,27%	96,37%	0,87%	0,00%
oli-hea___0015	0,00%	0,35%	99,65%	0,00%
oli-hea___0016	0,00%	52,92%	24,88%	22,00%
oli-hea___0017	0,00%	99,96%	0,03%	0,01%
oli-hea___0018	0,06%	99,59%	0,00%	0,33%
oli-hea___0019	0,00%	72,31%	27,67%	0,01%
oli-hea___0020	0,00%	28,98%	71,02%	0,00%
oli-hea___0021	0,00%	99,96%	0,03%	0,01%
oli-hea___0022	0,00%	100,00%	0,00%	0,00%
oli-hea___0023	0,00%	99,99%	0,01%	0,00%
oli-hea___0024	0,00%	99,99%	0,00%	0,00%
oli-hea___0025	0,00%	99,81%	0,19%	0,00%
oli-hea___0026	0,00%	99,99%	0,00%	0,00%
oli-hea___0027	0,00%	4,04%	95,96%	0,00%
oli-hea___0028	0,00%	98,68%	1,31%	0,01%
oli-hea___0029	0,01%	99,98%	0,00%	0,00%
oli-hea___0030	0,00%	79,08%	20,92%	0,00%

Total	30
Corretas	26
Incorretas	4
Sensibilidade	86,67%
Média de acerto	94,30%
Desvio padrão	12,35%

Tabela A.2: Análise da sensibilidade da classe Hea (saudável)

Análise da classe real Pea

	Classe prevista			
Imagem	**Bor**	**Hea**	**Pea**	**Pot**
oli-pea___0001	0,00%	0,00%	100,00%	0,00%
oli-pea___0002	0,00%	0,00%	100,00%	0,00%
oli-pea___0003	0,00%	0,00%	100,00%	0,00%
oli-pea___0004	0,00%	0,00%	100,00%	0,00%
oli-pea___0005	0,00%	20,37%	79,61%	0,02%
oli-pea___0006	0,00%	0,00%	100,00%	0,00%
oli-pea___0007	0,00%	0,00%	100,00%	0,00%
oli-pea___0008	0,00%	40,07%	59,92%	0,01%
oli-pea___0009	0,00%	0,01%	99,99%	0,00%
oli-pea___0010	0,00%	0,00%	100,00%	0,00%
oli-pea___0011	0,00%	0,00%	100,00%	0,00%
oli-pea___0012	0,00%	0,00%	100,00%	0,00%
oli-pea___0013	0,00%	0,00%	100,00%	0,00%
oli-pea___0014	0,00%	0,00%	100,00%	0,00%
oli-pea___0015	0,00%	0,00%	100,00%	0,00%
oli-pea___0016	0,00%	0,00%	100,00%	0,00%
oli-pea___0017	0,00%	0,00%	100,00%	0,00%
oli-pea___0018	0,00%	0,00%	100,00%	0,00%
oli-pea___0019	0,00%	0,00%	100,00%	0,00%
oli-pea___0020	0,00%	98,38%	1,62%	0,00%
oli-pea___0021	0,00%	0,00%	100,00%	0,00%
oli-pea___0022	0,00%	0,00%	100,00%	0,00%
oli-pea___0023	0,00%	0,04%	99,96%	0,00%
oli-pea___0024	0,00%	0,00%	100,00%	0,00%
oli-pea___0025	0,00%	0,00%	100,00%	0,00%
oli-pea___0026	0,00%	0,00%	100,00%	0,00%
oli-pea___0027	0,00%	0,00%	100,00%	0,00%
oli-pea___0028	0,00%	0,00%	100,00%	0,00%
oli-pea___0029	0,00%	0,00%	100,00%	0,00%
oli-pea___0030	0,00%	0,00%	100,00%	0,00%

Total	30
Corretas	29
Incorretas	1
Sensibilidade	96,67%
Média de acerto	97,91%
Desvio padrão	8,09%

Tabela A.3: Análise da sensibilidade da classe Pea (olho de pavão)

Análise da classe real Pot

	Classe prevista			
Imagem	**Bor**	**Hea**	**Pea**	**Pot**
oli-pot___0001	0,00%	0,04%	6,66%	93,30%
oli-pot___0002	0,00%	0,00%	0,01%	99,99%
oli-pot___0003	0,00%	0,46%	55,58%	43,96%
oli-pot___0004	0,00%	0,00%	0,01%	99,99%
oli-pot___0005	0,00%	0,00%	0,00%	100,00%
oli-pot___0006	0,00%	0,00%	0,00%	100,00%
oli-pot___0007	0,00%	58,28%	18,61%	23,04%
oli-pot___0008	0,00%	0,00%	99,54%	0,46%
oli-pot___0009	0,00%	0,00%	0,00%	100,00%
oli-pot___0010	0,00%	0,00%	0,02%	99,96%
oli-pot___0011	0,01%	0,00%	0,00%	99,99%
oli-pot___0012	0,00%	0,00%	0,00%	100,00%
oli-pot___0013	0,00%	0,00%	0,00%	100,00%
oli-pot___0014	0,00%	0,00%	0,09%	99,91%
oli-pot___0015	0,00%	0,00%	0,28%	99,72%
oli-pot___0016	0,00%	0,00%	0,00%	100,00%
oli-pot___0017	0,00%	0,00%	0,00%	100,00%
oli-pot___0018	0,00%	0,00%	0,00%	100,00%
oli-pot___0019	0,00%	0,00%	0,00%	100,00%
oli-pot___0020	0,00%	0,00%	0,00%	99,99%
oli-pot___0021	1,67%	0,00%	0,00%	98,28%
oli-pot___0022	0,00%	0,00%	0,00%	100,00%
oli-pot___0023	0,60%	4,86%	0,00%	94,53%
oli-pot___0024	0,00%	0,13%	0,00%	99,86%
oli-pot___0025	0,00%	0,01%	0,10%	99,89%
oli-pot___0026	0,00%	0,00%	0,00%	100,00%
oli-pot___0027	0,00%	0,01%	0,00%	99,98%
oli-pot___0028	0,00%	0,00%	0,00%	100,00%
oli-pot___0029	0,00%	0,00%	0,00%	100,00%
oli-pot___0030	0,00%	0,00%	0,00%	100,00%

Total	30
Corretas	27
Incorretas	3
Sensibilidade	90,00%
Média de acerto	99,46%
Desvio padrão	1,61%

Tabela A.4: Análise da sensibilidade da classe Pot (carência de potássio)