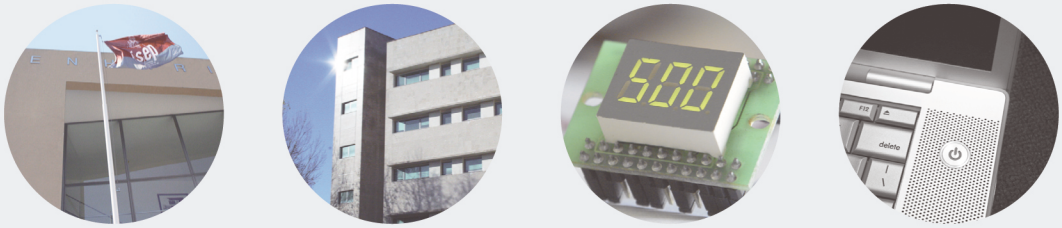
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de perceção visual com recurso a tecnologia de smartphones

SARA RAQUEL MONTEIRO DA SILVA PEREIRA
julho de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Sistema de perceção visual com recurso a tecnologia de smartphones

Sara Raquel Monteiro da Silva Pereira

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas Autónomos

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Julho, 2023

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Sistemas Autónomos.

Candidato: Sara Raquel Monteiro da Silva Pereira, Nº 1170515,
1170515@isep.ipp.pt

Coorientação Científica: André Dias, APD@isep.ipp.pt

Empresa: INESC TEC - Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

Orientador: Filipe Santos, filipe.n.santos@inesctec.pt

Co-Orientador: Luís Santos, luis.c.santos@inesctec.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Julho, 2023

Inicio os meus agradecimentos aos meus orientadores, ao Professor Doutor André Dias e ao Doutor Filipe Neves dos Santos, por todo o apoio dado ao longo da presente dissertação. Pretendo também realizar um agradecimento ao Luís Santos, colega de laboratório no INESC TEC, que me apoiou em todas as fases e ajudou a ultrapassar dificuldades para prosseguir com o trabalho até ao sucesso da dissertação. Agradeço também a todos os colegas do laboratório do CRIIS no INESC TEC, pelo acolhimento e apoio. De seguida pretendo deixar um agradecimento especial aos meus amigos e família, que estão sempre do meu lado, a apoiar, ouvir e saber o que dizer quando mais preciso. As recordações dos momentos passados no ISEP serão sempre lembradas com muito carinho.

Resumo

Na presente dissertação, desenvolvida no INESC TEC (Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência) nos laboratórios do CRIIS (*Center for Robotics in Industry and Intelligent Systems*), pretende-se explorar a capacidade de modelos *deep learning* no ambiente *Android*, através da comparação e avaliação de modelos YOLOv5 (YOLOv5s e YOLOv5n), YOLOv8 e *Single Shot Multibox Detector* (SSD) *MobileNetv2*. Este tópico encontra-se inserido no desenvolvimento do projeto Orioos – Solução robótica autónoma de baixo custo para a monitorização e a fenotipagem de culturas permanentes. Este consiste num robô em que o sistema de visão e perceção do ambiente é exclusivo a uma aplicação *Android*. Além de reduzir significativamente o custo do robô, esta abordagem permite uma reutilização/reciclagem de *smartphones* mais antigos. Para o treino dos modelos foram utilizados 2 *datasets* de uvas e troncos de videira, disponíveis *online*, e foi criado um *dataset* de *QR codes* envolvidos na vinha.

Como resultado dos treinos, utilizou-se a ferramenta *FittyOne* e um *dataset* de teste com imagens que os modelos nunca analisaram, de modo a comparar os resultados de *ground truth* com deteções efetuadas por cada modelo com uma confiança igual ou superior a 25 %. Verificou-se que os modelos YOLOv5n, YOLOv5s e YOLOv8n destacaram-se com os resultados mais positivos. A deteção de *QR codes* apresentou valores mais elevados de precisão, seguida da identificação de uvas. Na deteção de troncos, os modelos obtiveram valores menos positivos, sendo que o valor mais elevado de F1 *score* foi de 23 %. De todos os modelos, o modelo SSD *MobineNetv2* apresentou, nas deteções de uvas e de troncos, resultados menos satifastórios. Relativamente ao tempo de processamento no CPU de um computador, os modelos SSD *MobineNetv2*, YOLOv5n, YOLOV5s e YOLOv8n destacaram-se com valores inferiores a 600 milissegundos.

Após a análise num computador, foram realizados 3 testes de campo para avaliar os modelos YOLOv5s, YOLOv5n e SSD *MobineNetv2* num *smartphone*. Os modelos YOLO destacaram-se com o maior número de deteções correctas e as mais elevadas precisões. Na mesma área de análise, o modelo SSD *MobineNetv2* necessitou de uma maior proximidade do objeto para que este fosse detectado. Posteriormente, foram analisados os tempos de processamento no CPU de um *smartphone* e verificou-se que os valores mais baixos são do modelo SSD *MobineNetv2*. Este modelo revelou-se 33 % mais rápido do que o modelo YOLOv5n, o modelo mais rápido da rede YOLO.

O modelo YOLOv5s, apesar de ter mais precisão na deteção, tem um tempo de inferência mais longo, pelo que não consegue acompanhar as mudanças de perspetiva. Para a aplicação Orioos, foi adicionada a capacidade de descodificar os *QR codes* quando estes são detetados por uma rede neural. A identificação e descodificação de *QR Codes* permitirá ao robô ter informação externa, como a informação da *docking station* mais próxima, ou a localização precisa do local, de forma a eliminar erros incrementais que existem na navegação autónoma.

Palavras-Chave: Android, Aplicação móvel, Dataset, Deep learning, Deteção de frutos, Inferência, Redes neuronais, Robótica na agricultura, Visão por computador

Abstract

In this dissertation, developed at INESC TEC (Institute of Systems and Computer Engineering, Technology and Science) in the CRIIS (Center for Robotics in Industry and Intelligent Systems) laboratories, we intend to explore the capacity of deep learning models in the Android environment, through the comparison and evaluation of models YOLOv5 (YOLOv5s and YOLOv5n), YOLOv8 and Single Shot Multibox Detector (SSD) Mobilenet v2. This topic is inserted in the development of the Orioos project - Low cost autonomous robotic solution for monitoring and phenotyping of permanent cultures. This consists of a robot in which the vision and environment perception system is exclusive to a Android application. This approach significantly reduces the cost of the robot and allows for reutilization/recycling of older smartphones. For training the models, 2 datasets of grapes and vine trunks, available online, were used and a dataset of QR codes involved in the vineyard was created.

As a result of the training, we used the FittyOne tool and a test dataset with images that the models have never analyzed, in order to compare the results of ground truth with detections made by each model with a confidence equal to or greater than 25 %. It was found that the YOLOv5n, YOLOv5s and YOLOv8n models stood out with the most positive results. The detection of QR codes showed higher accuracy values, followed by the identification of grapes. In trunk detection, the models obtained less successful results, with the highest F1 score value being 23 %. Of all the models, the model MobineNetv2 presented, in the detections of grapes and trunks, less satisfactory results. Regarding the processing time on a laptop CPU, the models SSD MobineNetv2, YOLOv5n, YOLOV5s and YOLOv8n stood out with values below 600 milliseconds.

After the analysis on a computer, 3 field tests were performed to evaluate the YOLOv5s, YOLOv5n and SSD MobineNetv2 models on a smartphone. The YOLO models stood out with the highest number of correct detections and the highest accuracies. In the same analysis area, the SSD MobineNetv2 model required a greater proximity to the object for it to be detected. Subsequently, the processing times in the CPU of an smartphone were analyzed and it was found that the lowest values are from the SSD model MobineNetv2. This model proved to be 33 % faster than the YOLOv5n model, the fastest model in the YOLO network. The YOLOv5s model, despite having more accuracy in detection, has a longer inference time, so it cannot

keep up with perspective changes. For the Orioos application, the ability to decode QR codes was added when they are detected by a neural network. The identification and decoding of *QR Codes* will allow the robot to have external information, such as the information of the nearest *docking station*, or the precise location, in order to eliminate incremental errors that exist in autonomous navigation.

Keywords: Android, Computer vision, Dataset, Deep learning, Fruit detection, Inference, Mobile application, Neural networks, Robotics in agriculture

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

CNN *Convolutional Neural Network*

CPU *Central Process Unit*

FIP *Finding Patterns*

FN *False Negative*

FP *False Positive*

GNSS *Global Navigation Satellite System*

GPS *Global Positioning System*

GPU *Graphics Processing Unit*

HTTP *Hypertext Transfer Protocol*

ICT *Information and Communication technologies*

INESC TEC Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

IoT *Internet of Things*

IoU *Intersection over Union*

LiDAR *Light Detection and Ranging*

MQTT *Message Queuing Telemetry Transport*

RAM *Random Access Memory*

RF *Radio Frequency*

RFID *Radio Frequency Identification*

ROS *Robot Operating System*

RSS *Received Signal Strength*

SLAM *Simultaneous Localization and Mapping*

TP *True Positive*

UDP *User Datagram Protocol*

UGV *Unmanned Ground Vehicles*

WLAN *Wireless Local Area Network*

YOLO *You Only Look Once*

Capítulo 1

Introdução

1.1 Contextualização

Com o aumento da população mundial, o setor da agricultura deve acompanhar as necessidades de consumo e, ao mesmo tempo, ter em consideração problemas de nível económico e ambiental (escassez de água, seca, etc), sendo a sustentabilidade um tópico de extrema importância na atualidade. O agricultor deve efetuar tarefas como a identificação do fruto e monitorização do estado do mesmo, sendo que estas etapas representam uma grande importância para existir lucro. No entanto, a indústria alimentar ainda apresenta uma taxa relativamente alta de desperdício. Por exemplo, a Índia, um dos maiores produtores de fruta, desperdiça entre 30 % a 35 % da produção devido à subjetitividade da perceção dos seus trabalhadores [1].

Neste contexto, constatou-se que a aplicação de tecnologias tais como o uso de *smartphones* contribui para a perceção dos trabalhores e para a mudança do seu comportamento em relação a métodos mais conservadores [2]. O uso de novas tecnologias permite uma resposta a desafios do aumento de produtividade exigida pelo mercado, como também maximizar a rentabilidade tendo em mente a sustentabilidade do ambiente.

1.2 Caracterização de deteção de frutos

O processo de cultivo de frutos envolve múltiplas etapas que exigem um nível apropriado de conhecimento para a sua realização, nomeadamente a identificação do

fruto, monitorização do seu estado e contagem dos mesmos para uma correta gestão de recursos. Com uma quantidade menor de mão de obra, os obstáculos tornam-se ainda maiores visto que isso implica aumento de custos. Porém, os múltiplos problemas associados à agricultura abriram uma oportunidade para soluções tecnológicas.

O caso de estudo desta dissertação é a deteção autómatica de troncos de videira, de uvas e de *QR codes*. A rentabilidade de produção neste setor requer um custo elevado para máquinas e equipamentos, mão de obra e manutenção da vinha [3]. Durante a época de colheita, torna-se imperativo contratar mão de obra que possua as competências necessárias, o que se mostra cada vez mais desafiador em virtude da escassez de trabalhores qualificados e das condições precárias de trabalho. Os custos de mão de obra são uma parte significativa dos custos totais de produção, no caso de estudo de L. Araújo [4], em 2018, significaram 37,1 % do valor total. Para além disso, o reconhecimento da vinha para se ter conhecimento do rendimento da mesma é um processo trabalhoso dada a monitorização constante que é requerida, desde contagem de cachos por videira ou bagos por cacho. As uvas são uma fruta delicada pela sua constituição frágil, no entanto podem ser utilizadas em vários tipos de produtos (vinho, uvas de mesa, passas, entre outros). A produção de vinho entre 2020/2021 e 2021/2022 sofreu um crescimento de 18,6% o que implica um crescimento na produção de uvas [5]. Na campanha de 2022/2023 espera-se uma menor quantidade da produção de uvas traduzindo-se numa estimativa de diminuição de 9 % na produção de vinho relativamente a 2021/2022. Esta baixa deve-se à falta de água e ondas de calor verificadas em 2022, pelo que as condições de 2023 até à vindima são determinantes [6].

Neste sentido, é ainda mais crucial uma correta gestão dos recursos, monitorização do cultivo e eficiência na apanha do fruto, o que aumenta a procura e uso de automação nesta área. Um processo cada vez mais comum na produção agricola é o uso de visão por computador para que seja possível analisar o estado do fruto, desde o seu crescimento como também a densidade de produção [7]. De maneira a ser possível o uso de algoritmos de visão computacional para a deteção de um fruto é imprescindível o conhecimento de características do mesmo. A uva tem características heterogéneas no que diz respeito ao tamanho, formato e cor, sendo esta última característica também afetada pela luz solar. Estes são fatores que podem impactar o algoritmo de processamento de imagem. As imagens presentes na Figura 1.1 foram adquiridas por uma plataforma robótica e é possível observar diferentes estados do crescimento da uva. Verifica-se desde a imagem E até à G diferentes tonalidades e tamanhos, fatores afetados no processo de amadurecimento.

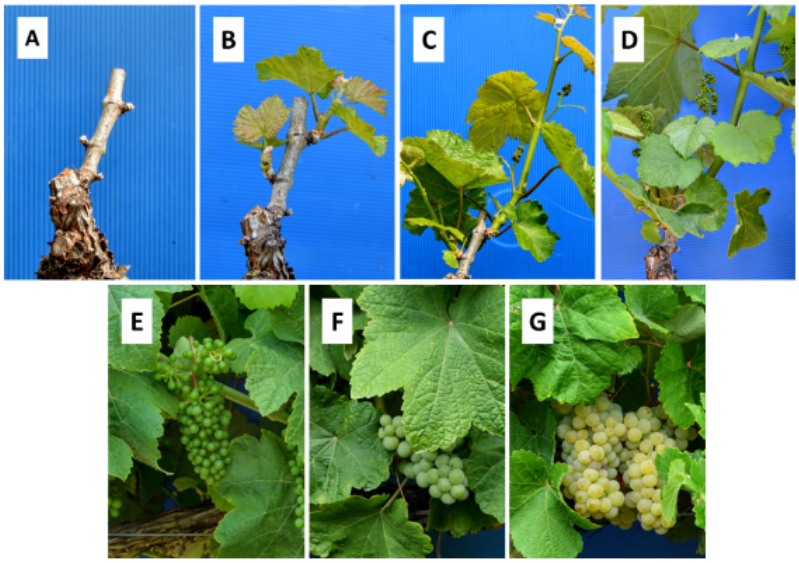


Figura 1.1: Diferentes estados do crescimento da uva. Retirada de [8].

1.3 Motivação

Na agricultura existe cada vez mais uma integração digital a métodos já existentes. Isto reflete-se no uso de inteligência artificial, sistemas de informação e outros tipo de *Information and Communication technologies* (ICT) para uma melhor gestão das diferentes etapas [9].

A tecnologia vai ser um fator cada vez mais presente dado que até 2027, espera-se um investimento global em inteligência artificial no mercado da agricultura de cerca de 5.17 mil milhões de dólares americanos [10]. Para além disso, com a inovação da tecnologia, os *smartphones* estão equipados com sensores e algoritmos de *machine learning*, o que permite uma nova camada de conhecimento e uma melhoria na resolução de problemas. Em 2022 foram instaladas 37 mil milhões de aplicações móveis, um aumento de 21 % em relação a 2021 e a maior parte em dispositivos *Android* [11]. Com aplicações nos *smartphones*, os agricultores podem recolher informação relevante, precisa e mais abrangente de forma a aumentar a produção de forma sustentável e a melhorar a subsistência do ambiente.

1.4 Objetivos

No laboratório de Robótica e IoT para Agricultura e Floresta de Precisão Inteligente do Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC) têm sido desenvolvidas soluções robotizadas e sistemas inteligentes

para aplicação em contexto de agricultura e floresta. Neste âmbito, existe a necessidade de desenvolver soluções de perceção visual para estes robôs, que sejam versáteis, de baixo custo, robustas em diferentes contextos de iluminação, modulares e reconfiguráveis. Com a realização da presente dissertação, o projeto Orioos – Solução robótica autónoma de baixo custo para a monitorização e a fenotipagem de culturas permanentes, com recurso a aplicação móvel desenvolvido pelo INESC TEC, pode ser melhorado. Este consiste num robô em que o sistema de visão e perceção do ambiente é exclusivo a uma aplicação *Android*. Como é possível observar na Figura 1.2, esta solução está centralizada num robô terrestre autónomo que utiliza um *smartphone* para adquirir dados GNSS e dados de imagem. Esta abordagem permite reduzir significativamente o custo do robô e permite uma reutilização/reciclagem de *smartphones* mais antigos. O robô Orioos é uma solução robótica de monitorização e fenotipagem, este consegue obter a localização exata do robô e, com recurso a redes neuronais, permitirá detetar e contar, por exemplo, frutos, animais, entre outros. A aplicação *Android* conecta-se a uma *cloud* através do protocolo MQTT.

Figura 1.2: Projeto Orioos.

A presente dissertação propõe um sistema automático em tempo real para deteção de componentes (*features*), especificamente uvas, troncos e *QR codes*, e a integração do mesmo numa aplicação móvel (Figura 1.3), em ambiente *Android*. Pretende-se obter a georreferenciação no momento da deteção, para que, com um bloco de navegação básico, seja possível controlar um robô por *waypoints* utilizando tópicos de *Robot Operating System* (ROS).

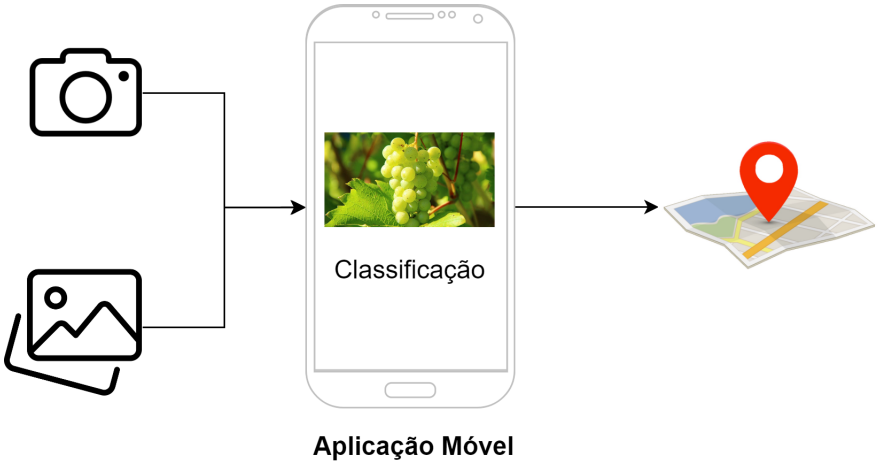


Figura 1.3: Diagrama de objetivo de deteção visual e georreferênciação.

A utilização de técnicas *deep learning* implica um *dataset* com grandes quantidades de dados, que serão fornecidos pelo INESC TEC e o respetivo laboratório. Propõe-se o desenvolvimento de uma versão robusta de deteção automática de uvas, troncos e *QR codes* baseada em redes neuronais e a posterior georreferênciação. Pretende-se treinar diferentes modelos, como o algoritmo *MobileNet* e YOLO, de modo a testar a sua *performance* em laboratório e na aplicação móvel. A robustez do sistema *deep learning* deve contar com a deteção em todas as condições de iluminação. Propõe-se a deteção e descodificação de *QR codes*, para que, como trabalho futuro, estes sejam considerados *landmarks* artificiais. Esta funcionalidade irá permitir que o robô tenha informação externa, como por exemplo, a informação da *docking station* mais próxima, ou a localização exata do local, de forma a eliminar erros incrementais que existem em navegação autónoma.

1.5 Plano de Trabalho

Na Figura 1.4 está descrito o plano de trabalho para a elaboração da presente dissertação. O projeto está divido em 3 macro tarefas: o estudo e treino de redes neuronais, o desenvolvimento da aplicação móvel e a escrita da dissertação.

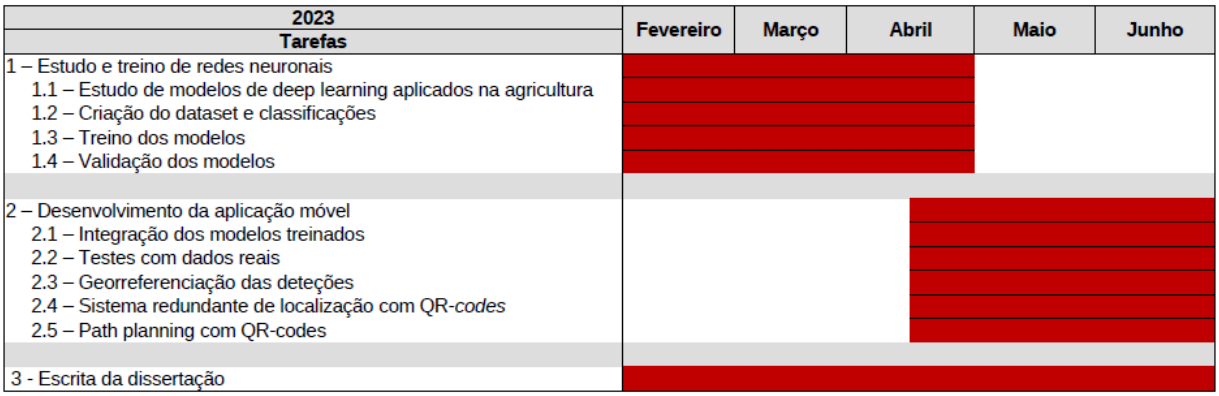


Figura 1.4: Calendarização.

1.6 Organização da Dissertação

A presente dissertação tem a seguinte organização. No primeiro capítulo é realizada uma introdução ao tema, onde é efetuada uma contextualização, seguida da caracterização da deteção de frutos e são abordados temas que fundamentam a elaboração da dissertação. Para além disso, são referidos os objetivos e o plano de trabalho.

No segundo capítulo, é realizada uma revisão do estado da arte com a apresentação de técnicas de *deep learning* aplicadas no presente contexto e são explicadas as métricas avaliadas no desempenho de modelos. Seguidamente, são especificados sensores presentes em *smartphones* e as respetivas funções, tendo em consideração o ambiente de estudo da dissertação. São descritos algoritmos de navegação autónoma e, por último são enumerados artigos cujo objetivo é o desenvolvimento de aplicações móveis na agricultura, de modo a se ter conhecimento da tecnologia e métodos utilizados.

No terceiro capítulo, é apresentada e descrita a implementação realizada e os respetivos resultados. São explicados e justificados os processos da etapa de treino de redes neuronais e é efetuada uma comparação dos mesmos, através de avaliação de métricas e deteção em imagens de teste. Relativamente à aplicação móvel, são descritos os algoritmos efetuados e é apresentada a diferença obtida com testes reais entre as redes neuronais treinadas e integradas no *smartphone*.

No quarto e último capítulo, são retiradas conclusões do trabalho desenvolvido na presente dissertação e propostas de trabalho futuro.

Capítulo 2

Revisão do Estado de Arte

Neste capítulo será apresentado e abordado um estudo do estado de arte relacionado com a deteção e classificação automática de frutos e a integração em aplicações móveis. Serão explicadas as métricas avaliadas para classificação do desempenho de um modelo *deep learning* para melhor compreensão dos métodos referidos no estado de arte. Serão apresentados exemplos de algoritmos *deep learning* aplicados neste contexto, com uma abordagem à sua arquitetura. Será apresentado o ecossistema de um telemóvel, em que são referidos os sistemas operativos mais utilizados e sensores inseridos em *smartphones* que podem ser aplicados no contexto da agricultura. De forma aos robôs, integrados com sistemas de visão, conseguirem operar no meio agrícola devem ter incorporados algoritmos e sensores que permitam uma localização precisa e um planeamento de trajetória que possibilite efetuar tarefas em segurança. Neste sentido, serão apresentadas diferentes abordagens utilizadas na área da navegação autónoma. Na quinta e última fase, são descritas propostas de diferentes autores de algoritmos de deteção de objetos no âmbito da agricultura e com recurso a aplicações móveis.

2.1 Agricultura de precisão

Agricultura de precisão é uma estratégia de gestão e análise de dados combinado com outras informações de modo a apoiar decisões de acordo com a maior eficiência na utilização dos recursos, produtividade, qualidade e sustentabilidade na produção agrícola [12]. Com a necessidade do aumento de produtividade e falta de mão

de obra, é imperativo que as empresas encontrem soluções para atingir as suas necessidades. A integração de tecnologia e de sistemas de informação na indústria, conhecida como indústria 4.0, permitiu que existisse um aumento da eficiência e do controlo de erros devido à existência de sensores entre outras tecnologias que fornecem informação precisa. No setor da agricultura são cada vez mais usadas tecnologias como sensores, *Internet of things* (IoT), inteligência artificial e robôs que permitem converter a agricultura tradicional numa agricultura precisa e sustentável. A implementação de ferramentas de agricultura de precisão permite que os agricultores pratiquem uma cultura com vantagens económicas devido a terem conhecimento de quando efetuar determinadas práticas [13].

Este tipo de agricultura adequa-se às necessidades de cada agricultor. Devem ser identificadas as prioridades com o utilizador de modo a que exista uma instalação de tecnologia de forma a cumprir os objetivos. Na Figura 2.1 pode-se observar que o ciclo de agricultura de precisão é dividido em 4 etapas [14].

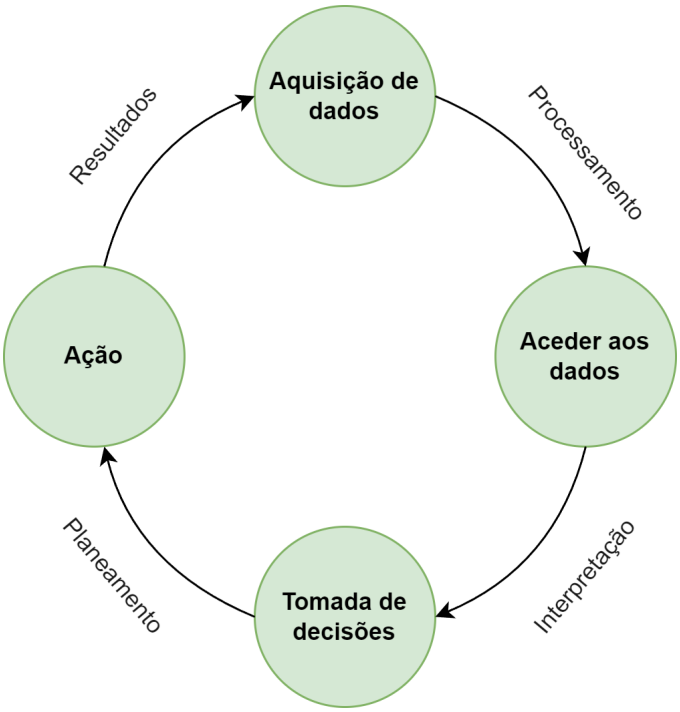


Figura 2.1: Ciclo de Agricultura de Precisão.

A primeira etapa consiste na aquisição de dados consoante o que se pretende analisar, desde a temperatura do ar, temperatura do solo, entre outras métricas. Esta informação pode ser adquirida com drones, sensores, robôs, entre outros meios. O próximo passo é o processamento de dados, que consiste em aceder à informação e interpretá-la. A aquisição pode ser realizada por diferentes meios e estar em diferentes formatos, como também pode existir informação incompleta e valores inválidos. Por isso pode ser necessário proceder a uma normalização dos dados

para um formato pré-definido e a identificação e eliminação de dados duplicados. Além disso, os dados recolhidos de diferentes fontes podem ser armazenados numa estrutura comum. Com os dados trabalhados é necessário traduzi-los em decisões ponderadas e inteligentes. Podem ser usados algoritmos matemáticos, modelos de estimação ou métodos de inteligência artificial (*machine learning*). Desta forma, são tomadas ações de acordo com a demanda da cultura e existe assim um uso eficiente dos recursos [14].

A Figura 2.2 representa um exemplo de diagrama de aplicação de agricultura de precisão baseada em IoT [13]. Primeiramente existe uma obtenção de dados através de sensores, drones ou robôs, seguindo-se o envio dos mesmos para a subsequente análise com técnicas de processamento de dados. O uso de computadores, *smartphones* ou aplicações móveis permite que a informação chegue aos agricultores de forma a que haja um planeamento das práticas de gestão da agricultura.

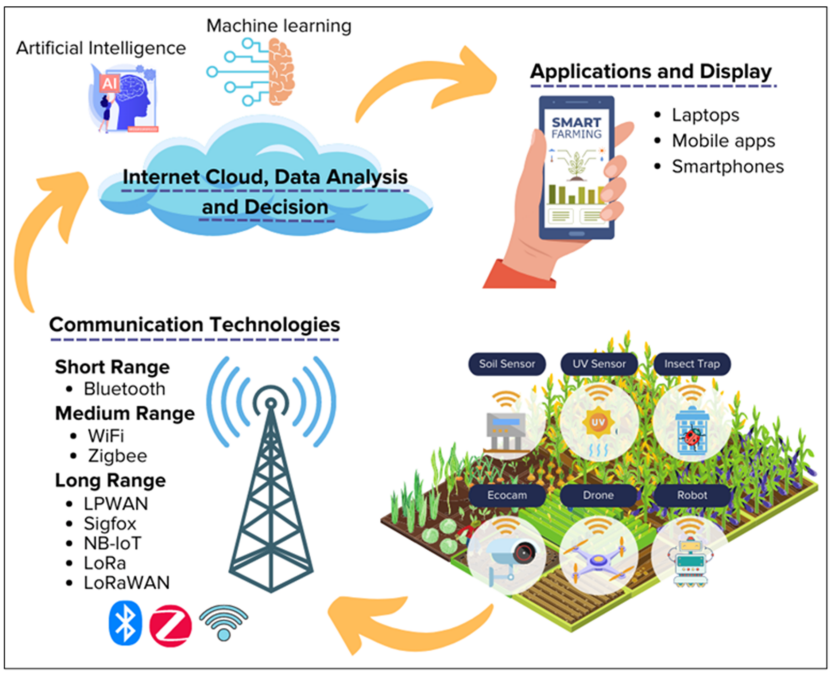


Figura 2.2: Diagrama de aplicação de agricultura de precisão. Retirada de [13].

2.2 ICT, Smartphones e Aplicações Móveis na agricultura

Information and Communication Technologies (ICT) é definido como um conjunto de tecnologias utilizadas para transmitir, armazenar e criar informação [15]. De acordo com *Kuhlmand and Berg* (2002) [16] , o desenvolvimento da agricultura foi

sumarizada como "...*it may be safely stated that modern large scale farming technology [...] controlled by state of the art information and communication technology has the potential for substantial reductions of the production costs for agricultural commodities*". O desenvolvimento de ICTs permite uma redução de custos de procura e uma diminuição da barreira de informação visto que podem fornecer dados personalizados. Em instalações agrícolas devem existir sistemas diferentes de monitorização, como sistemas de irrigação, controlo do solo e da iluminação e, por esta razão, existem tipicamente sistemas de controlo para cada elemento individual. Através destes meios tecnológicos passa a existir uma camada de suporte de decisão para os agricultores devido ao acesso constante a informação recente [17].

Na Figura 2.3 estão descritas algumas áreas em que ICTs têm um papel fundamental na agricultura. Como referido anteriormente, estas permitem um fornecimento de dados constante, facilitam a comunicação, entre outras vantagens [18].

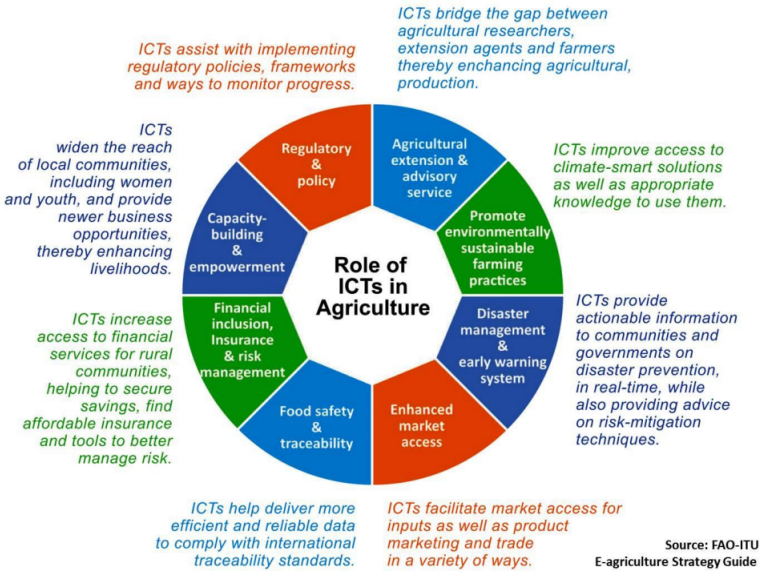


Figura 2.3: Papel de ICTs na agricultura. Retirada de [18].

As aplicações móveis são cada vez mais utilizadas na agricultura como meio de informação à medida que o uso de *smartphones* cresce continuadamente. Estes são pequenos, leves e possíveis de transportar para todo o lado significando acessibilidade de dados em qualquer situação. Contêm diversas funções que fornecem capacidades computacionais com possibilidade para monitorização, identificação de frutos ou até georeferênciação dos mesmos, permitindo uma análise da cultura.

2.3 Robótica na agricultura

A robótica na agricultura tem como objetivo tornar o trabalho mais eficiente, aumentando a produção e os lucros dos processos. O crescimento da demanda de produção e a falta de mão de obra nos períodos de colheita fez com que o uso de processos automatizados aumentasse [19]. O uso de *Unmanned Ground Vehicles* (UGV) está integrado na agricultura de precisão dado que estes têm integrado sensores/equipamento que possibilitam a execução de operações precisas e a recolha de medições exatas.

Estes robôs devem estar integrados com algoritmos de navegação autónoma e sistemas de perceção/localização, como um sensor *Light Detection and Ranging* (LiDAR) e *Global Navigation Satellite System* (GNSS). De modo a terem autonomia, devem existir sistemas de segurança responsáveis pela deteção de obstáculos e prevenção de colisões no trajeto do veículo. Para a recolha de dados e para o uso de algoritmos de visão computacional devem ter integrados sensores como câmaras, e para recolha/armazenamento de dados devem existir comunicações *wireless* baseadas em IoT [20].

Luiz Oliveira *et al.* [21] efetuaram um estudo de aplicações na agricultura que permitem efetuar tarefas com sistemas/máquinas autónomas. Os casos de uso incidem em diversas funções como a preparação da terra, semear e plantar, tratamento de plantas e apanha de fruto.

Figura 2.4: Exemplo de aplicação de robô na agricultura. Retirada de [22].

2.4 Deep Learning

A inteligência artificial é uma ferramenta cada vez mais presente em tarefas que o humano tem dificuldades de perceção ou em que seja necessária uma maior eficiência/rapidez na realização das mesmas. Uma das áreas de maior aplicação é a visão computacional, esta permite automatizar processos através da análise de imagens utilizando meios tradicionais. No entanto, em situações de imprevisibilidade e em que é necessária uma constante aprendizagem, são usados algoritmos que, com uma quantidade elevada de dados, conseguem extrair características dos mesmos, encontrar padrões e obter conclusões sem ser necessária a intervenção humana [23].

A arquitetura de *deep learning* consiste em múltiplas camadas (redes neuronais) com algoritmos em que cada uma interpreta a informação introduzida. As redes neuronais são tipicamente supervisionadas, o que exige que os dados iniciais para treinar o modelo sejam previamente anotados. É essencial que estes representem o que se pretende identificar mas que sejam variados de modo a representar de forma mais abrangente, a classe que se pretende identificar para que não ocorra *overfitting*, isto é quando a rede se torna demasiado especialista nos dados de treino. As redes neuronais podem ter diferentes tipos de arquiteturas conseguindo assim obter caraterísticas de aprendizagem específicas. Isto permite que as aplicações sejam muito variadas, especialmente ao nível de processamento de imagem. As *Neural Networks* são um conjunto de neurónios ligados por camadas (*layers)*, estando as mesmas representadas na Figura 2.5. Na camada representada a laranja (*input layer*) a informação é recebida, transitando posteriormente para as camadas de neurónios, representadas a azul (*hidden layers*). Estas camadas são o ponto principal de uma rede neuronal e é onde são efetuadas operações matemáticas nos dados de entrada. Finalmente, as classificações obtidas pela rede neuronal são apresentadas de acordo com a maior probabilidade. O resultado de uma rede neuronal depende da quantidade de informação introduzida, como também do número de camadas para produzir uma saída [24].

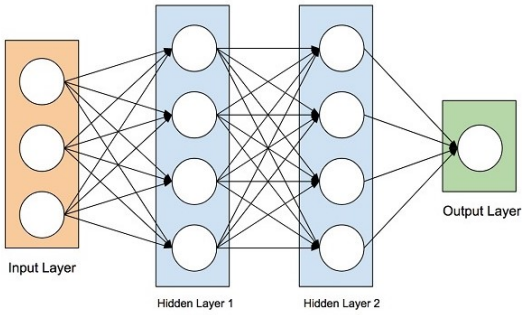


Figura 2.5: Exemplo de uma rede neuronal de duas camadas. Retirada de [25].

O interesse no *deep learning* pela parte das indústrias tem sido cada vez maior. O facto de existir uma aprendizagem com informação não estruturada com elevado grau de precisão torna esta aplicação uma grande vantagem. Por isso mesmo, este é uma tema com grande entusiasmo para a comunidade científica [26].

2.4.1 Avaliação dos Métodos

Para uma correta avaliação do comportamento do modelo é necessário ter conhecimento do cálculo e do significado de métricas importantes. Deve-se ter em consideração as siglas TP (*True Positive*), FP (*False Positive*) e FN (*False Negative*). Numa visão mais clara, TP refere-se às deteções corretas, FP às deteções incorretas e FN indica os falsos negativos [23].

Relativamente ao *recall* (equação 2.1), este permite analisar a relação entre as previsões positivas realizadas corretamente e todas as previsões que realmente são positivas (TP + FN). No caso da precisão (equação 2.1), esta representa a capacidade do modelo detetar corretamente os objetos pretendidos através da relação entre as previsões positivas realizadas corretamente e todas as previsões positivas (incluindo as falsas). O *recall* permite analisar a quantidade de FN e a precisão a quantidade de FP.

$$Recall = \frac{TP}{TP + FN} \tag{2.1}$$

$$Precisão = \frac{TP}{TP + FP} \tag{2.2}$$

Outro parâmetro que é comum ser analisado é o F1 (equação 2.3), este permite analisar o equilibrio entre o *recall* e a precisão [23].

$$F1 = 2 \times \frac{Recall \times Precisão}{Recall + Precisão} \tag{2.3}$$

Na avaliação dos resultados também se deve definir o valor percentual da interseção entre a área de deteção e a área onde se encontra a *bounding box* da *label* efetuada nas anotações (*groundtruth*). Na Figura 2.6 pode-se observar a forma de cálculo e um exemplo ilustrativo da determinação da *Intersection over Union* (IoU).

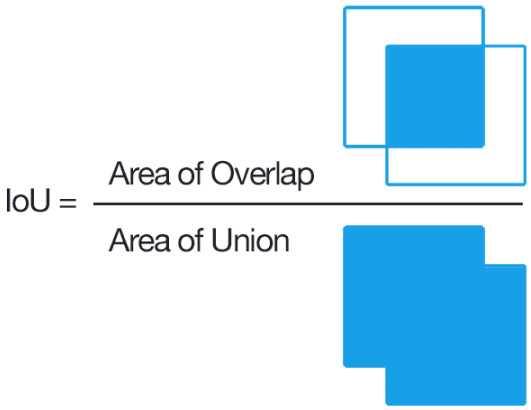


Figura 2.6: Exemplo da determinação de *Intersection over union*. Retirada de [27].

2.4.2 CNN

Uma rede neuronal convolucional (ConvoNet/CNN) é uma arquitetura de *deep learning*, inspirada pela perceção visual dos seres vivos, para deteção de objetos através da capacidade de interpretar e analisar imagens. Um modelo CNN consiste num conjunto de *processing layers* que têm capacidade de aprender características da imagem de entrada com diferentes níveis de abstração [28]. Na Figura 2.7 está descrito o conceito do modelo CNN com as respetivas *layers*.

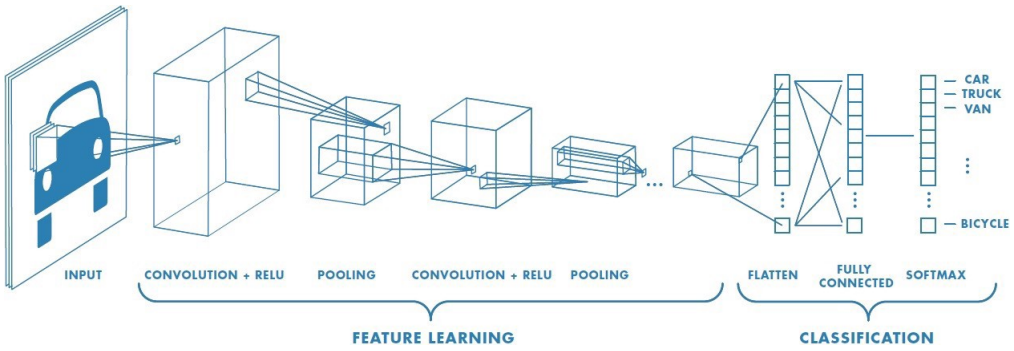


Figura 2.7: Conceito do modelo CNN. Retirada de [29].

Pode-se observar na Figura 2.7 que a imagem de entrada é dividida em várias camadas de *pixels* e é realizado um processo de aprendizagem em que são efetuadas duas operações: *convolution* e *pooling*. A camada de convolução é o compontente mais importante da arquitetura. Esta contém um filtro de convolução, conhecido como *Kernel* que, consoante os seus valores, permite que uma nova imagem seja criada por alteração dos valores dos *pixels* em relação ao seu valor original e o dos seus *pixels* vizinhos. Na Figura 2.8 pode-se verificar que a aplicação de um filtro é um produto de matrizes. No caso demonstrado, o *pixel* com valor de 4 é obtido

através da operação $1\times1+0\times0+0\times1+1\times0+1\times1+0\times0+1\times1+1\times0+1\times1=4$. [30].

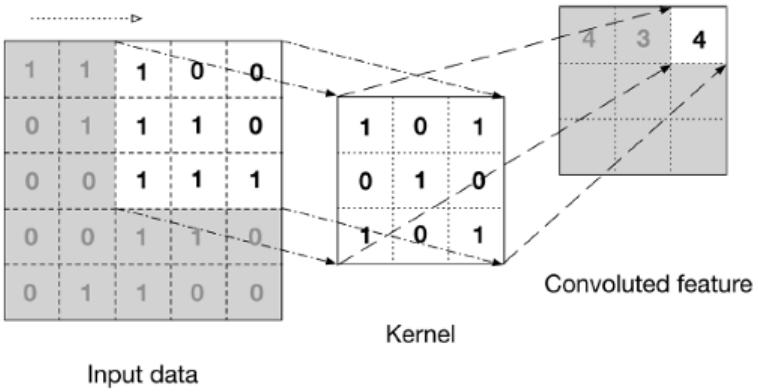


Figura 2.8: Exemplo de um filtro de *Kernel*. Retirada de [30].

Seguida da camada de convolução, existe sempre uma camada de *pooling*. O objetivo desta camada é reduzir a representação de dados, o controlo da sobreposição e a caracterização da área da imagem de entrada. O tipo de *pooling* mais utilizado é a operação max() para redimensionar espacialmente a entrada [30]. Na Figura 2.9 pode-se verificar que é selecionado o *pixel* com o maior valor considerando uma matriz de 4 *pixels*.

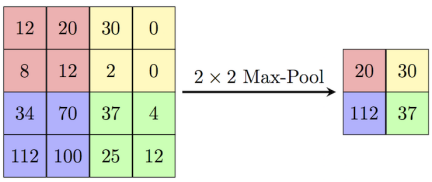


Figura 2.9: Exemplo de *Pooling*. Retirada de [31].

Por fim é utilizada uma camada (*Fully Connected Layer*) que calcula as probabilidades das classes obtidas a partir da análise da imagem de entrada.

2.4.3 MobileNet

MobileNet é uma arquitetura capaz de diminuir significativamente o tamanho do modelo e não afetar a sua precisão [32]. Esta característica permite que seja utilizada em aplicações móveis visto que estes dispositivos têm menor capacidade de memória e é preciso ter em consideração a bateria, pelo que um modelo mais leve torna-se mais eficiente [33]. Esta arquitetura, ao contrário de uma rede neuronal convolucional convencional, utiliza convoluções *depthwise separable*. Existem dois tipos de operações: *Depthwise convolution* e *Pointwise convolution* [32]. Numa *Depthwise*

convolution é efetuada uma convolução independente a cada canal da imagem de entrada (Figura 2.10).

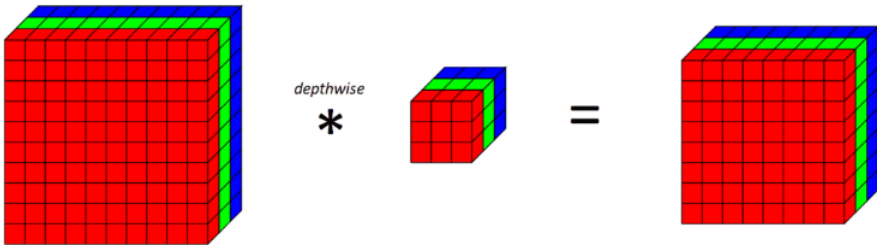


Figura 2.10: Exemplo de uma *Depthwise convolution*. Retirada de [34].

De seguida, na *Pointwise convolution*, é aplicada uma convolução 1x1 a todos os canais do *output* da primeira convolução (Figura 2.11).

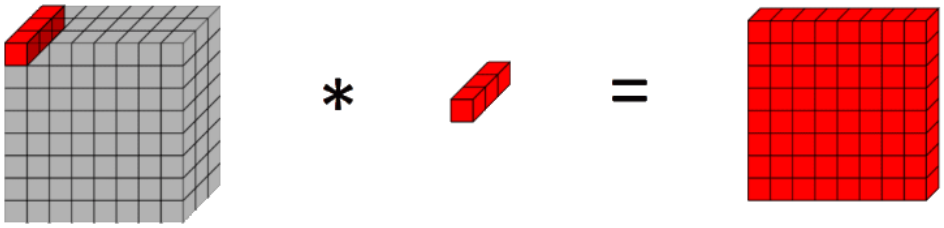


Figura 2.11: Exemplo de uma *Pointwise convolution*. Retirada de [34].

Estas convoluções permitem que existam menos parâmetros e cálculos, pelo que reduz o tempo de treino/inferências.

2.4.4 YOLO

You Only Look Once (YOLO) é um algoritmo de deteção de objetos em tempo real, baseado numa rede neuronal convolucional. Ao contrário de outros algoritmos, esta arquitetura analisa a imagem de entrada uma única vez, o que permite que a velocidade de deteção seja uma característica interessante. Como é possível observar na Figura 2.12, o modelo YOLO divide a imagem em pequenos quadrados. Em cada célula existe uma previsão do objeto e múltiplas *bounding boxes* com uma probabilidade de confiança na deteção do mesmo [35].

Cada *bounding box* contém 5 parâmetros de previsão: as coordenadas x e y do centro da *bounding box*, a largura, a altura e a confiança [36]. Esta arquitetura tem sofrido evoluções desde a sua criação em 2015, até à versão mais recente, a YOLOv8.

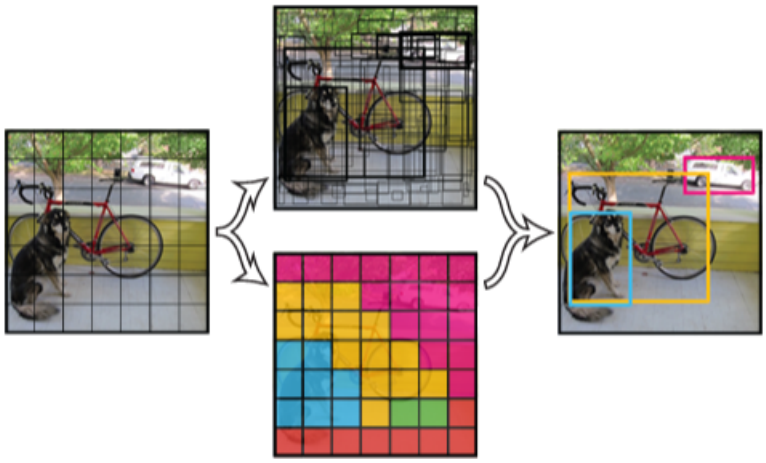


Figura 2.12: Análise de uma imagem com o modelo YOLO. Retirada de [35].

2.5 Mobile Ecosystem

Mobile Ecosystem é o conjunto de bens e serviços oferecidos, incluindo o *hardware*, o sistema operativo e a possibilidade de obter aplicações [37]. Atualmente a maior parte dos dispositivos móveis são os *smartphones*. Estes têm integrado uma unidade de processamento e outras características como um sistema operativo, *web browser* e capacidade de executar aplicações [38]. Existem dois sistemas operativos que são maioritariamente utilizados: *Android*, desenvolvido pela *Google*, e iOS, desenvolvido pela *Apple*.

O sistema operativo *Android* é baseado no sistema operativo *Linux*, tendo sido lançada a sua primeira versão em 2008 e a última versão, *Android 14*, em Fevereiro de 2023. O sistema operativo iOS é designado para dispositivos exclusivamente da *Apple*. Este teve a sua primeira versão em 2007 e a última versão foi lançada em Janeiro de 2023 (iOS 16.3). Ambos os sistemas operativos têm o respetivo *Web browser*, o *Android* disponibiliza o *Google Chrome* enquanto o iOS disponibiliza o *Safari* [39].

Os *smartphones* têm integrados sensores e capacidades de conectividade [40] que permitem auxiliar o utilizador nas mais variadas tarefas. Os sensores podem ser divididos em 5 categorias: sensores de movimento, de ambiente, de posição, de conectividade e imagem. Relativamente à presente dissertação, os sensores descritos na Tabela 2.1 [41] permitem o desenvolvimento de aplicações para uma agricultura de precisão.

Tabela 2.1: Sensores disponíveis num *smartphone* e respetivas aplicações na agricultura.

Sensores	**Função**	**Aplicação na Agricultura**
Acelerômetro	Medir a aceleração nos 3 eixos.	Assistência na navegação.
Giroscópio	Medir a velocidade angular.	Assistência na navegação.
Câmara	Gravar imagens e vídeos.	Dados para processamento de imagem.
Global Navigation Satellite System (GNSS)	Fornece geolocalização.	Assistência na navegação.
Cellular Network	Permite conexão a uma *network*.	Comunicar com um servidor para envio/receção de dados.
Wi-Fi	Permite conectar a uma *Wireless Local Area Network* (WLAN).	Comunicar com outros dispositivos/servidor.
Bluetooth	Permite trocar informação com outros dispositivos a uma curta distância.	Comunicar com outros dispositivos.

2.6 Navegação em robôs autónomos

Os robôs autónomos têm capacidade de aumentar a eficiência de execução de tarefas e estão presentes em diversas áreas. Para um correto funcionamento, deve-se ter em consideração a sua navegação, pelo que esta é uma área essencial de estudo na robótica. Estão equipados com sensores que permitem ter uma perceção do ambiente em que se encontram, como sonares, sensores inerciais, *encoders*, entre outros. O problema de navegação autónoma está dividida em 3 áreas: localização, mapeamento e planeamento de trajetória. A localização é o processo de determinar o ponto onde o robô está localizado. O mapeamento integra os obstáculos do ambiente em que o robô está inserido e o planeamento de trajetória determina o melhor caminho [42].

2.6.1 Métodos de localização

A navegação autónoma na agricultura é uma tarefa dificil devido às mudanças do ambiente, que são condicionadas pela altura do ano [42]. Deve-se ter em consideração a segurança do robô e do ambiente ao seu redor por isso é essencial uma alta precisão na localização do veículo. Atualmente, a maior parte da navegação baseia-se no uso de GNSS. Esta tecnologia refere-se ao conjunto de satélites que fornece informação sobre a posição e tempo aos recetores [43]. No caso de robôs na agricultura, estes operam frequentemente em ambientes agrícolas ou florestais onde os sinais de satélite sofrem um bloqueio e há impedimento de acesso ao sinal GNSS. Nesse sentido, é necessário os veículos estarem equipados com outros meios. As principais abordagens para resolver o problema de localização são as seguintes [44]:

- GPS: Este método fornece as coordenadas de localização do robô. A determinação da informação é obtida de forma precisa através de dados fornecidos por satélites.

- Odometria e sensores inerciais: Estes sensores determinam a localização do robô tendo sempre em consideração um ponto de localização prévio. O erro associado aumenta ao longo do tempo originando um problema de *dead reckoning*. Neste sentido, é fundamental a existência de infomação, contínua ou não, da localização absoluta de onde se encontra o robô para correção do erro.

- Determinação de distância e ângulo em relação a uma *landmark*: Alguns exemplos da tecnologia utilizada são *Radio Frequency* (RF) *beacons*, *Radio Frequency Identification* (RFID), entre outros.

- Odometria visual: Determinação da posição de *landmarks* ou implementação do método *Simultaneous Localization and Mapping* (SLAM) através do uso de uma câmara ou de um sistema *stereo*.

Lee et al. [45] desenvolveram um sistema de localização visual baseado em *QR codes* como *landmarks* artificiais. Cada *landmark* tem um *id* único que corresponde a uma coordenada x e y no mundo real e os *QR codes* são reconhecidos e descodificados a partir de um *smartphone*. Este tem as capacidades computacionais necessárias dado que tem integrado sensores como uma câmara e capacidades de comunicação que permitem a sua utilização no projeto referido. Na Figura 2.13 está descrita a arquitetura geral do trabalho desenvolvido. A aplicação *Android* identifica e descodifica o *QR -code* através da deteção de círculos presentes na área da *landmark*. Posto isto, através de comunicação *Wi-Fi* e acesso a um *database*, a informação presente na *landmark* é obtida. São efetuados cálculos para obter a localização exata do robô e esta informação é enviada por comunicação *bluetooth*.

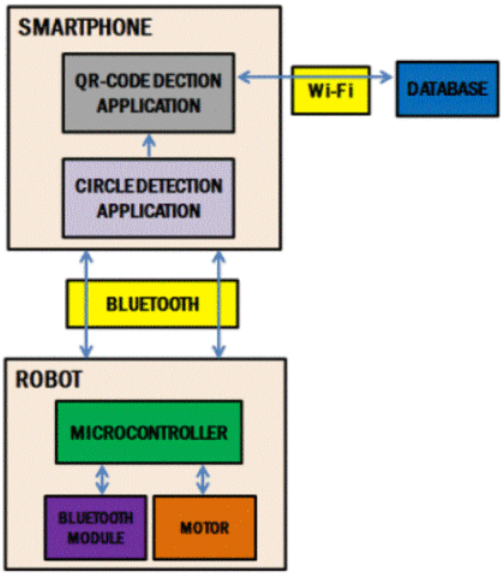


Figura 2.13: Diagrama da arquitetura do trabalho. Retirada de [45].

Quando o robô deteta e tem conhecimento da informação da *landmark*, são efetuados cálculos geométricos de modo a obter a posição do robô, de acordo com a Figura 2.14. Inicialmente é realizada a conversão da distância do centro do *QR code* ao centro da imagem (*Center of Image*) para uma medida real (*Dist_real*). A distância real (*Dist_real*) é decomposta na distância no eixo x (*Real_base*) e no eixo y (*Real_height*). Por último, ao subtrair a localização real da *landmark* (obtida com acesso ao *database*) à distância calculada do robô à mesma obtém-se a localização do robô em coordenadas x e y.

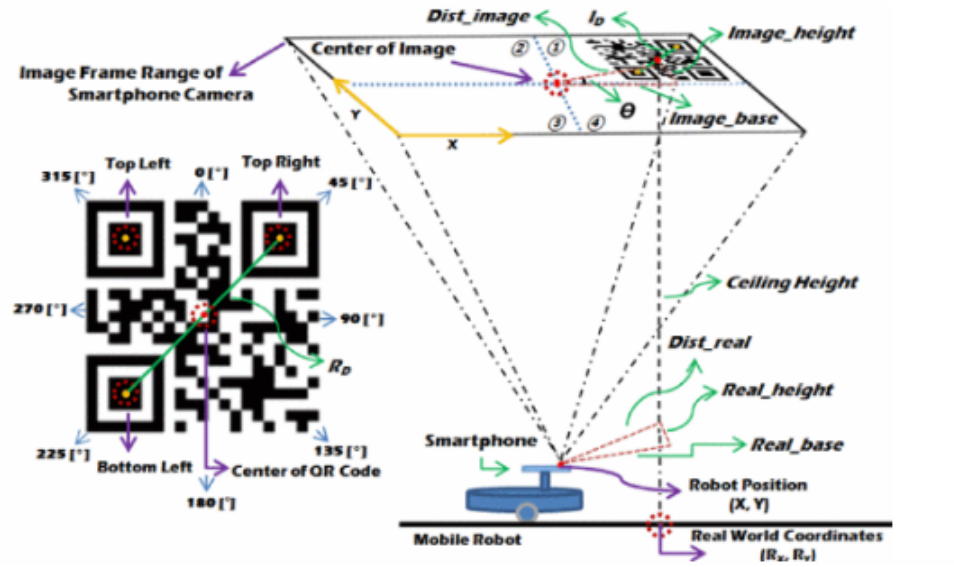


Figura 2.14: Localização do robô baseado em *QR code*. Retirada de [45].

O trabalho referido deteta o *QR code* através de visão computacional, no entanto é possível fazê-lo recorrendo a algoritmos de *deep learning*. *Blanger et al.* [46] realizaram uma comparação do desempenho de treino de diferentes arquiteturas. Os autores criaram um *dataset* (Figura 2.15), que agora se encontra disponível *online*, com 767 imagens de variados tamanhos contendo 1263 *QR codes*. Foram efetuadas *labels* dos *QR codes* e também das *Finding Patterns* (FIP), os quadrados presentes em 3 coordenadas de cada *landmark*. Para a avaliação, foram realizados treinos com dois cenários de aplicação: só *QR codes* e *QR codes* +FIP.

Figura 2.15: Exemplos de *QR codes* detetados. Retirada de [46].

Na Tabela 2.2 pode-se observar os resultados obtidos tendo em consideração a análise da precisão média. O modelo *MobileNet* apresentou resultados mais elevados para ambos os cenários de análise.

Tabela 2.2: Resultados da avaliação do desempenho dos diferentes modelos *MobileNet*, VGG16 e *ResNet50*, tendo em consideração a precisão média [46].

	QR codes	*QR codes* + FIPs
MobileNet	72.7 %	68.7 %
VGG16	67.7 %	66 %
ResNet50	67.1 %	64.1 %

Numa outra abordagem, utilizando um *smartphone*, os autores de *"Agricultural wireless sensor mapping for robot localization"* [47] desenvolveram um sistema de localização e mapeamento através da distância estimada obtida com um *iBeacon*. A distância e o ângulo entre um recetor e o *beacon* é determinada através de *Received Signal Strength* (RSS). Para permitir que o robô obtenha a informação dos *beacons*, foi desenvolvida uma aplicação *Android* e o *smartphone* está integrado no veículo. A aplicação obtém a informação de cada *beacon*, através de *bluetooth*, e com a biblioteca *Android Beacon* é calculada uma distância. A cada 1 segundo, esses dados são enviados através do protocolo *Datagram* (UDP) para o robô. Na Figura 2.16 pode-se observar o robô utilizado pelos autores, o local de testes e o posicionamento dos *beacons*.

Figura 2.16: Local de teste com *beacons*. Retirada de [47].

2.6.2 Métodos de planeamento de trajetória

O planeamento de trajetória consiste em encontrar uma sequência de movimentos que o robô deve realizar desde um ponto de partida até um destino, evitando ao mesmo tempo obstáculos no seu ambiente de trabalho. Ao contrário de ambientes *indoor*, os cenários de aplicação de robótica na agricultura enfrentam desafios. Estes são inconstantes, visto que são alterados ao longo do ano e estão condicionados

pelas condições metereológicas. Neste sentido, é necessário existir estratégias de planeamento de trajetórias adequadas. Nesta área existem duas sub-categorias: *Point-to-Point Path Planning* e *Coverage Path Planning* [48].

Num planeamento do tipo *Point-to-Point*, o objetivo consiste em determinar uma trajetória sem colisões desde o ponto de partida até ao ponto de destino, tendo em consideração parâmetros de otimização como o tempo, distância e energia. *Coverage path planning* é a tarefa de determinar um caminho que passe por todos os pontos da área em que o robô está inserido evitanto obstáculos. Este processo é definido pelos seguintes passos [48]:

1. O robô deve passar por toda a área sem sobreposição de caminhos.
2. O robô deve evitar obstáculos.
3. Uso de movimentos simples.
4. Deve ser realizada uma trajetória "ótima"nas condições possíveis.

Em [49] *Gentilini et al.* implementaram em ROS um algoritmo de planeamento de trajetórias capaz de gerar um caminho através de *waypoints*, tendo como referência as restrições cinemáticas do robô. Este estudo foi realizado no contexto do uso de UGVs na agricultura. Como é possível observar na Figura 2.17, o trajeto é obtido tendo em consideração os *waypoints* descritos a amarelo. O percurso é efetuado com recurso a uma navegação com *Global Positioning System* (GPS) e sem mapa do local.

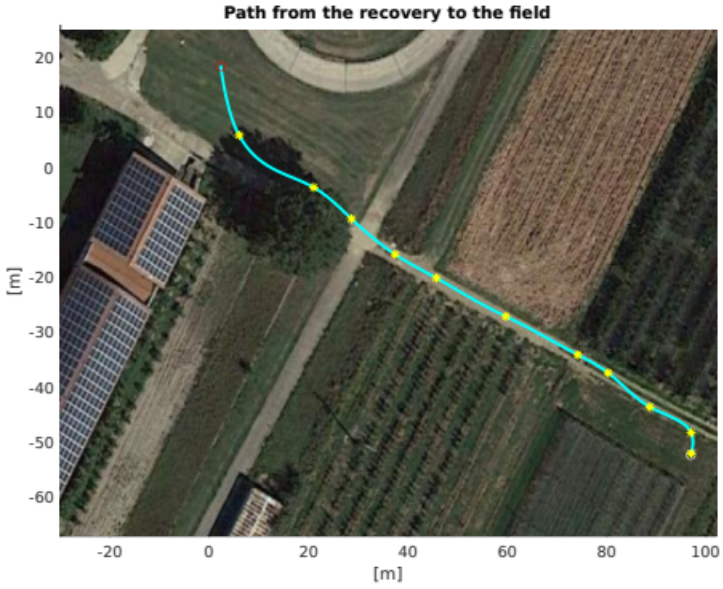


Figura 2.17: Trajetória efetuada recorrendo a *waypoints*. Retirada de [49].

Em [50] os autores melhoraram as capacidades de planeamento de trajetória em ambiente agrícola através da fusão de duas ferramentas desenvolvidas pelos mesmos, *AgRob Topologic* e *AgRob Path Planner*. *AgRobPP* é uma *framework open-source* criada em ROS. Esta está dividida em 3 ferramentas distintas:

- *AgRob Vineyard Detector* - Realiza segmentação de imagens de satélite para construção de mapas.
- *AgRob Topologic* - Uma ferramenta para contruir um mapa topológico a partir de um *occupancy grid map*.
- *AgRob Path Planner* - Abordagem de planeamento de trajetória para terrenos irregulares e com consideração do centro de massa do robô.

O *AgRob Path Planner* gera uma trajetória segura para o robô, no entanto uma limitação do mesmo é o excesso de uso de memória. Neste sentido, os autores propuseram utilizar a ferramenta *AgRob Topology* para gerar um *grid map*, em que a *AgRob Path Planner* é chamada para gerar uma trajetória em cada sub-mapa criado (abordagem local) e depois é utilizada uma abordagem de planeamento de trajetória global. De forma a encontrar o melhor caminho, é utilizado o algoritmo A* que tem em consideração a menor distância a ser percorrida. No planeamento de trajetória local é determinado o caminho mais curto entre *nodes*. Posto isto, numa abordagem global (Figura 2.18), é determinado o trajeto que assegura o menor número de *nodes* entre um ponto inicial e final.

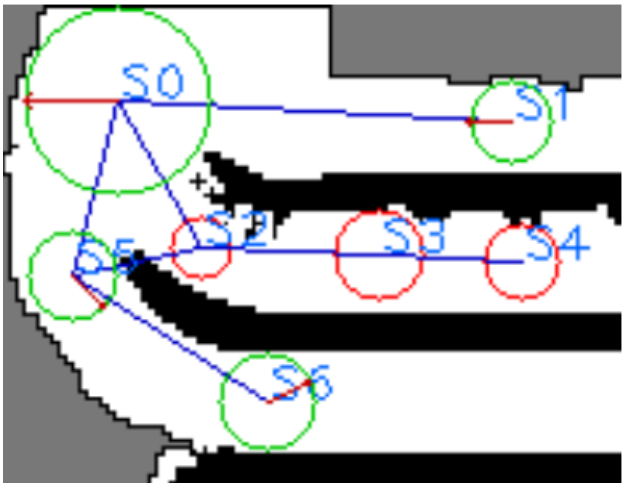


Figura 2.18: Exemplo do método de planeamento de trajetória global A*. Retirada de [50].

2.7 Aplicações móveis para a agricultura

Com a agricultura de precisão, aplicações integradas com algoritmos de inteligência artificial são cada vez mais usadas para resolver problemas, como identificação de frutos ou identificação de possíveis doenças.

Numa abordagem para detetar plantas e possibilidade de doenças, *Tembhurne et al.* [51] recorreram à arquitetura *MobileNet* como base e efetuaram uma adição de *hidden layers* para treinar um modelo e posteriormente integrá-lo numa aplicação *Android*, a *Plantscape*. O objetivo é identificar corretamente a doença, sendo que a imagem fornecida pode ser da cultura, da planta, folha ou flor. Para a constituição do *dataset* foram usadas 4 fontes *online* existindo no total 64 classes. Nos sistemas de deteção de imagens recorrendo a algoritmos de *deep learning* é comum a aplicação de variações nas imagens de forma a expandir o *dataset*. As técnicas utilizadas são geralmente a alteração de escala, posição e mudanças na imagem, como adição de ruído. Primeiramente, de forma a haver uniformidade, as imagems foram redimensionadas para um tamanho de 256 x 256 *pixels*. Posto isto, os autores decidiram, utilizando a biblioteca *Python ImageDataGenerator*, efetuar transformações como é possível observar na Figura 2.19.

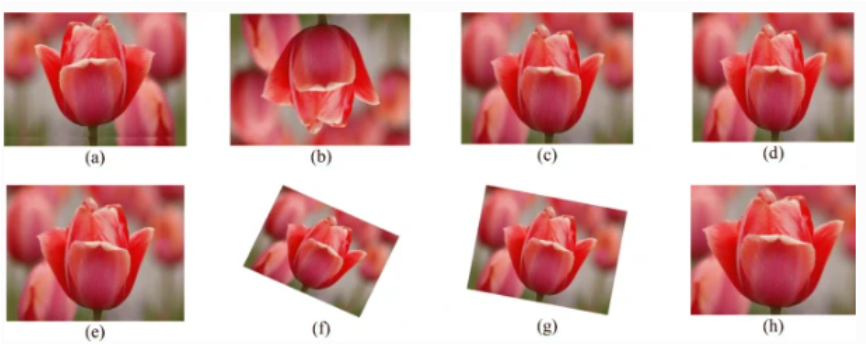


Figura 2.19: Exemplo de transformações realizadas. Retirada de [51].

O próximo passo consiste na deteção de objetos recorrendo a *bouding boxes* sob possíveis doenças identificadas nas plantas. Na realização de testes, foram comparadas diferentes arquiteturas a fim de atingir o melhor resultado. Os modelos avaliados foram o *Inceptionv2*, *Inceptionv3*, *ResNet50*, *Xception*, VGG-19, *NASNetMobile* e *MobileNetv1*. De forma a avaliar o desempenho dos modelos foram analisados os parâmetros de exatidão, precisão, *recall* e F1-*score* sendo que o modelo *MobileNetv1* teve melhores resultados com uma precisão de 94.4 %. Para além disso, esta arquitetura tem como vantagem a baixa latência e consumo. Para treinar o modelo, os autores utilizaram uma plataforma *cloud*, o *Microsoft Azure*. A *Virtual Machine* incluía 6 *Central Process Units* (CPU), 56 *Gigabytes* (GB) de *Random Access Memory* (RAM), 380 GBs de espaço no disco e uma *Graphics Processing Unit* (GPU)

Tesla K-80.

A aplicação *Android* desenvolvida tem diferentes funcionalidades para o utilizador (Figura 2.20). Tem integrada uma calculadora de fertilizante, informações sobre pragas e doenças, dicas para o cultivo e uma lista das plantas mais comuns. Relativamente à deteção em tempo real da planta e da doença, a aplicação permite que o utilizador tire uma fotografia utilizando a câmara do *smartphone* ou faça *upload* de uma imagem. De seguida é apresentado o nome da planta, a doença identificada e opções de tratamento no ecrã.

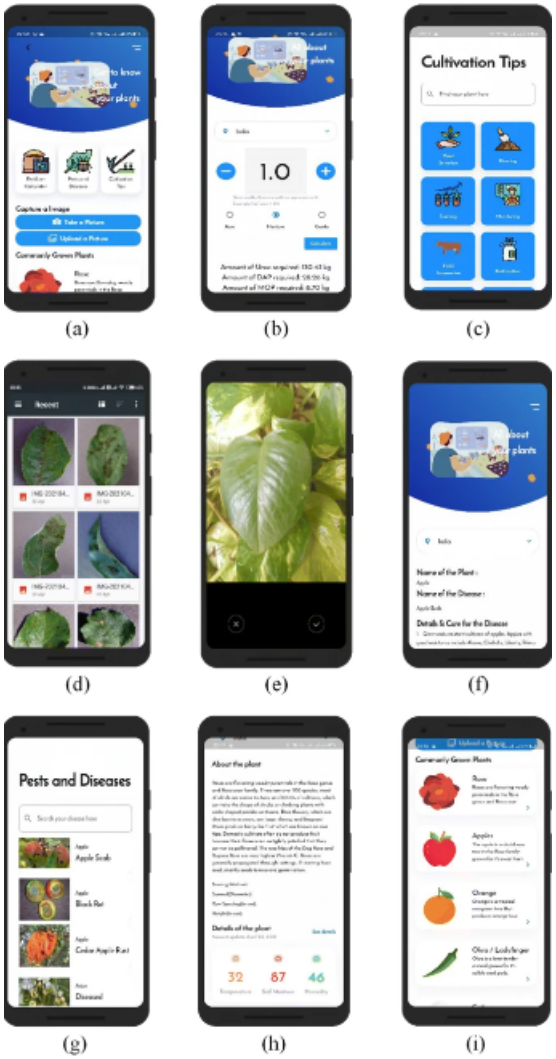


Figura 2.20: *User Interface* da Aplicação *Android* - *Plantscape* (a)*Home screen*, (b) Calculadora de fertilizante, (c) Dicas de cultivo, (d) Seleção da imagem, (e) Imagem capturada, (f) Doença detetada e detalhes, (g) Pragas e doenças, (h) Detalhes da planta, (i) Lista das plantas mais comuns. Retirada de [51].

Numa abordagem para classificação de frutas e vegetais, *Mukhiddinov et al.* [52] otimizaram o modelo YOLOv4 para, numa primeira instância, reconhecer o objeto

da imagem e depois classificá-lo como fresco ou podre. Para a constituição do *dataset* foram recolhidas imagens de 5 frutas (maçã, banana, laranja, morango e manga) e 5 vegetais (cenoura, batata, tomate, pepino e pimento) através do *Kaggle*, *Google*, *Bing images* e com o uso da câmara de um *smartphone*. No total existem 1200 imagens de dimensões 608 x 608 *pixels*. Neste trabalho foram efetuadas 2 tipos de *data augmentation*. Podem existir ajustes a nível dos *pixels* mas as *bounding boxes* não são alteradas, alguns exemplos são adição de *blur*, ajuste de brilho ou exposição, entre outros. Por outro lado, as imagens podem sofrer alterações que alteram os *pixels* e também as *bounding boxes*, como os exemplos da Figura 2.21. Este último tipo de transformações aumenta a *perfomance* do modelo na deteção de objetos.

Figura 2.21: Exemplo de métodos de augmentação: (a) imagem original, (b) 90º rotação, (c) 180º rotação, (d) 270º rotação. Retirada de [52].

Para além das alterações referidas, no modelo YOLOv4 foi ainda implementado o tipo de augmentação *Mosaic* (Figura 2.22). Este consiste na combinação de 4 imagens para gerar 1 imagem final. Inicialmente, as imagens são redimensionadas de maneira a que cada uma destas coincida com um canto da imagem final. De seguida, a imagem é cortada de forma aleatória podendo abranger qualquer área das imagens iniciais.

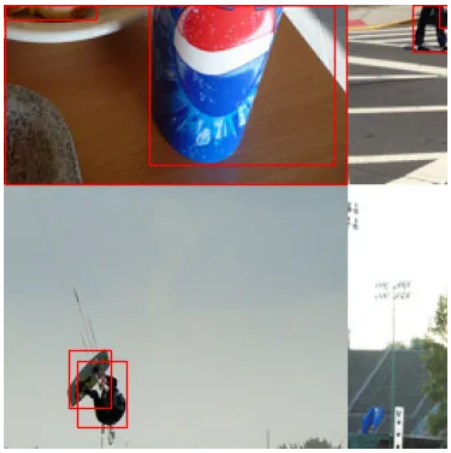


Figura 2.22: Exemplo de uma imagem criada a partir de *Mosaic Augmentation*. Retirada de [53].

Os autores realizaram testes num computador com um CPU 8-*core* 3.70 *Gigahertz* (GHz), 32 GB de RAM e uma gráfica *NVidia GeForce 1090 Ti*. Para os diferentes treinos realizados, utilizou-se um *batch size* de 32 e as imagens de entrada com dimensões de 416 x 416 *pixels*. O trabalho desenvolvido analisa o desempenho do modelo YOLOv3, YOLOv3-*tiny*, YOLOv4, YOLOv4-*tiny* e ainda o modelo YOLOv4 otimizado pelos autores. Os resultados presentes na Tabela 2.3 demonstram que, dos modelos YOLO sem otimização, o modelo YOLOv4 tem uma melhor precisão na deteção e classificação do objeto como também uma maior velocidade de treino em comparação com o YOLOv3.

Tabela 2.3: Resultados da avaliação do desempenho dos modelos YOLOv3, YOLOv3-tiny, YOLOv4 e YOLOv4-*tiny* [52].

Modelo	**Precisão média**	**Tempo Treino (h)**
YOLOv3	41.7 %	105
YOLOv3-*tiny*	23.6 %	12
YOLOv4	49.3 %	97
YOLOv4-*tiny*	28.5 %	10

De forma a implementar o sistema de classificação de fruta e legumes em tempo real, os autores desenvolveram um aplicação móvel. O sistema consiste numa arquitetura servidor-cliente. Esta baseia-se nos seguintes passos:

- O *smartphone* envia para o servidor a imagem a classificar.
- O servidor recebe, processa a imagem e prevê o resultado.
- Este projeto tem em consideração pessoas com problemas visuais, pelo que o resultado de texto é convertido para áudio.

- Os resultados, tanto áudio como visual, são enviados e apresentados no *smartphone*.

Na Figura 2.23 pode-se observar 2 exemplos de deteção de frutas.

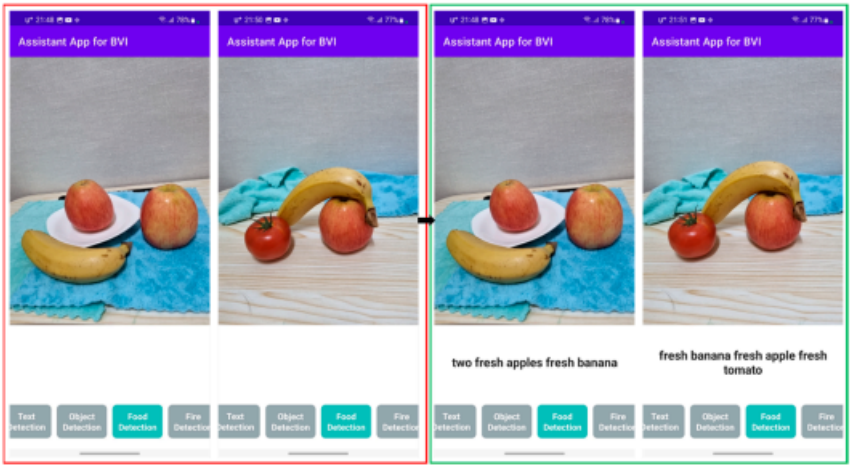


Figura 2.23: Aplicação móvel para classifcação de frutas e legumes. Retirada de [52].

Como limitações do projeto desenvolvido, os autores verificaram que algumas frutas e legumes têm *features* externas idênticas, como a cor, formato e textura, o que faz com que o sistema falhe na classificação.

Reda et al. [54] desenvolveram uma aplicação móvel, *AgroAld*, para classificação de espécies de plantas e doenças. O *dataset* utilizado foi o *PlantVillage*, que consiste em mais de 61000 imagens combinando 39 classes e cada classe representa uma combinação de espécie e doença. Este foi dividido com uma razão de 60:20:20 para treino, validação e teste, respetivamente. Este trabalho tem como foco o uso de *transfer learning* com um modelo CNN pré-treinado e investigar o efeito que tem a alteração do cenário de aprendizagem. Este é um método utilizado para treino de um modelo, tornando o mais eficiente, conferindo uma redução de recursos e de tempo. Foram testados 4 cenários distintos:

- Não utilizar as *convolution layers* e manter só as *classifier layers*.
- Não utilizar 80% das *convolution layers* e manter as restantes 20% das *convolution layers*.
- Não utilizar 50% das *convolution layers* e manter as restantes 50% das *convolution layers*.
- Utilizar todas as *layers*.

Como modelos de arquiteturas para estudo, os autores escolheram a *MobileNet*, *MobileNetv2*, *NasNetMobile* e *EfficientNetB0*. Considerando os 4 cenários referidos

e 2 conjuntos de parâmetros de treino, foram treinados 8 modelos por arquitetura. De modo a comparar os 4 melhores modelos, foi selecionado 1 modelo treinado por cada tipo de arquitetura. Os melhores modelos foram os seguintes:

- *MobileNetv2* com o 3º cenário.
- *NasNetMobile* com o 3º cenário.
- *MobileNet* com o 4º cenário.
- *EfficientNetB0* com o 4º cenário.

Na Tabela 2.4 estão descritos os resultados de cada modelo e cenário referido.

Tabela 2.4: Exatidão e F1-*scores* do melhor desempenho de cada arquitetura.

Modelo	**Exatidão**	***Mean F1-Score***
MobileNetV2	0.77	0.77
NasNetMobile	0.82	0.81
MobileNet	0.84	0.84
EfficientNetB0	0.99	0.99

Os autores concluíram que o melhor modelo como base de arquitetura foi o *EfficientNetB0* com o 4º cenário de *transfer learning*.

Relativamente à aplicação (*AgroAld*), foi desenvolvida em *Android* e a base de dados foi desenvolvida utilizando o *Cloud Firestone* para os dados serem armazenados de forma centralizada e segura. O modelo treinado está integrado na aplicação no formato de ficheiro *TensorFlow Lite*. Na Figura 2.24 pode-se observar o ecrã principal e as funcionalidades permitidas ao utilizador.

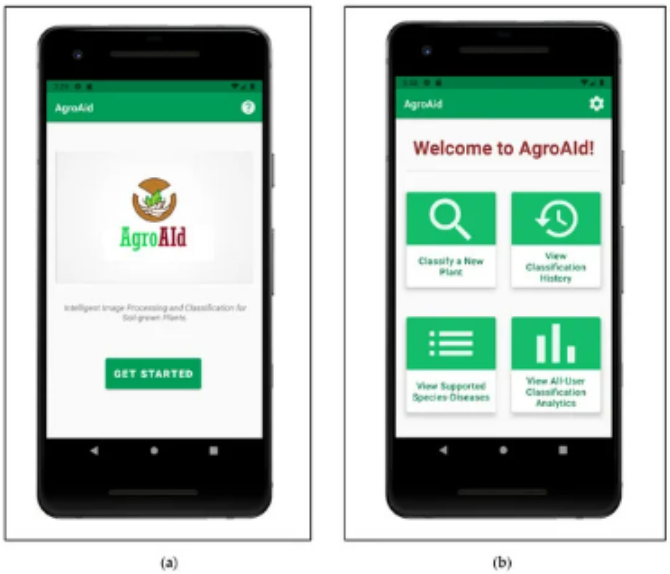


Figura 2.24: Aplicação móvel *AgroAid*: (a) Ecrã Principal e (b) Funcionalidades permitidas. Retirada de [54].

As funcionalidades implementadas incluem:

- Classificação de nova imagem.
- Tratamentos e conselhos para tratar da combinação espécie-doença detetada.
- Histórico de classificações.
- Apresentação de lista de todas as combinações suportadas pelo sistema.
- Obtenção e apresentação da análise espaço-temporal das espécies-doenças mais comuns.

Syamsury et al. [55] desenvolveram uma aplicação *Android* para deteção de doenças de plantas. Como o trabalho de *Reda et al.* [54], o *dataset* utilizado também foi o *PlantVillage*, com um *sub-set* de 32282 imagens para treino, 10756 imagens para validação e 10765 imagens para teste. De forma a realizar um estudo dos modelos com melhor desempenho, os autores selecionaram a arquitetura *MobileNetv2*, *MNasNet* e *Inceptionv3*. Depois do modelo treinado, este foi aplicado num computador e num *smartphone* para ser possível a comparação de resultados e a avalicação do modelo em diferentes contextos. De acordo com a Tabela 2.5, o modelo *MNasNet* teve a maior precisão (97 %).

Tabela 2.5: Classificação do desempenho dos modelos analisados.

Modelo	**Precisão**	***Recall***	***F1-Score***
MobileNet	95 %	94 %	95 %
MNasNet	97 %	96 %	96 %
Inceptionv3	96 %	96 %	96 %

Para testar o modelo numa aplicação móvel (Figura 2.25), os ficheiros gerados no processo de treino são convertidos para o formato *TensorFlow Lite*.

Figura 2.25: Aplicação móvel. Retirada de [55].

Para determinar os recursos do *smartphone* utilizados durante o uso da aplicação, são usadas ferramentas do *Android Studio*, nomeadamente a *Android profiler* para aceder à memória e ao uso do CPU. Para aceder ao estado da bateria foi utilizada a aplicação *AccuBatery*. O estudo foi efetuado em dispositivos móveis com as características descritas na Tabela 2.6.

Tabela 2.6: Especificações do dispositivo móvel.

Hardware	**Especificação**
Memória	4 GB
CPU	*Quad-core Snapdragon 820*
Sistema Operativo	*Android 8.9 Marsmallow*
Câmara	12 MP

No estudo dos recursos utilizados numa aplicação móvel, obtiveram-se os resultados descritos na Tabela 2.7. O modelo *Inceptionv3*, em todos os aspetos analisados,

apresenta quase o dobro do consumo comparativamente aos modelos *MobileNet* e *MNasNet*.

Tabela 2.7: Uso de recursos do *smartphone*.

Modelo	**Uso de CPU (Pico)**	***Power** (mAH)*
MobileNet	25.1 %	14.2
MNasNet	23.2 %	16.6
InceptionV3	43.5 %	34.3

Na Figura 2.26 pode-se analisar o consumo de bateria do modelo *MobileNet* no topo lateral esquerdo, *MNasNet* no topo direito e na parte inferior o modelo *Inceptionv3*.

Figura 2.26: Consumo de bateria com o modelo *MobileNet*, *MNasNet* e *Inceptionv3*. Retirada de [55].

Para uma análise de 100 imagens, o modelo *MobileNet* teve o menor consumo de 14,2 miliampere-hora (mAH). O modelo *MNasNet* teve também um consumo inferior (16.6 mAH) em relação ao *Inceptionv3*, em que se verificou um consumo cerca de duas vezes superior atingindo 34.4 mAH.

Dey et al. [56] propuseram o desenvolvimento de um modelo CNN, o *Fruit-VegCNN*, capaz de detetar frutas e vegetais e ser utilizado num dispositivo móvel. Para estudo foram selecionados os modelos *VGG (VGG19)*, *ResNet*, *MobileNetv2*, *NasNet* e *Inception-ResNet*. O *dataset* utilizado pelos autores é a combinação de 2 *datasets* de frutas e vegetais, num total de 132 classes. Os treinos foram realizados num computador com 4 CPU *Intel(R)Xeon(R)Gold6134* e uma GPU *Nvidia Tesla P100* com 12 GB de memória, de modo a otimizar o processo de treino. A partir da Tabela 2.8 pode-se verificar a exatidão obtida nos modelos analisados. O modelo *VGG19* obteve o valor mais alto, de 98.918 %, seguida da *MobileNetv2* e *Inception-ResNet* com 97.892 % e 98.432 %, respetivamente.

Tabela 2.8: Comparação de modelos tendo em consideração a exatidão.

Modelo CNN	Exatidão (%)
ResNet152v2	95.027
NASNetMobile	10.108
NASNetLarge	10.054
VGG19	98.918
MobileNetv2	97.892
Inception-ResNet	98.432

Posteriormente ao treino dos modelos, estes devem ser convertidos para o formato *TensorFlow Lite* de maneira a efetuar inferências em dispositivos móveis. A aplicação (Figura 2.27) foi desenvolvida usando a *framework Flutter* da *Google*.

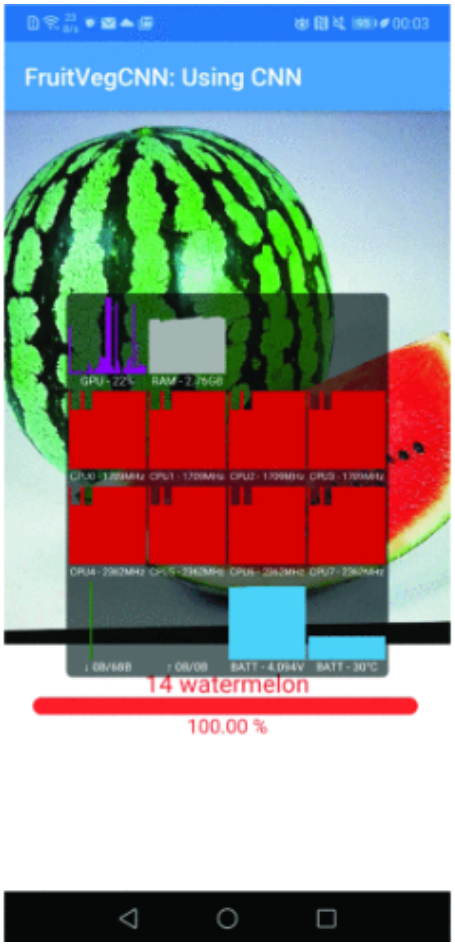


Figura 2.27: Aplicação FruitVegCNN. Retirada de [56].

Na Tabela 2.9 é efetuada uma comparação do tamanho dos ficheiros, em *megabytes* (MB), após conversão para o modelo pretendido. Pode-se verificar que os modelos *MobileNetv2* e *NASNetMobile* apresentam menor tamanho, dado que são

especificamente desenvolvidos para aplicações móveis.

Tabela 2.9: Comparação dos tamanhos dos ficheiros *TensorFlow Lite*.

Modelo CNN	**Tamanho (MB)**
ResNet152v2	335.5
NASNetMobile	20.9
NASNetLarge	352.5
VGG19	182.9
MobileNetv2	23.3
Inception-ResNet	340.8

Para analisar qual o modelo que apresenta melhor desempenho na integração numa aplicação móvel, foram avaliados os seguintes parâmetros: consumo da RAM, tempo de carregamento do modelo (*Loading Time)*, uso da CPU e da GPU e o consumo de energia em milliwatt (mW). Os resultados apresentados na Tabela 2.10 foram testados no *Huawei P20 Lite*.

Tabela 2.10: Comparação dos consumos do *smartphone* com diferentes modelos.

Modelo CNN	***Ld. Time (s)***	**RAM (GB)**	**CPU (%)**	**GPU (%)**	***Power (mW)***
ResNet152v2	16	0.65	64	21	2150
NASNetMobile	13	0.53	61	15	3302
NASNetLarge	14	0.69	68	11	3475
VGG19	17	0.78	72	22	3458
MobileNetv2	11	0.54	59	13	2746
Inception-ResNet	12	0.62	59	22	2880

Liu et al. [57] desenvolveram uma aplicação móvel para identificação da espécie de folhas de videira. O *dataset*, colhido manualmente pelos autores, contém 5091 imagens de 21 espécies diferentes (Figura 2.28).

Figura 2.28: Exemplos de imagens presentes no *dataset*. Retirada de [57].

Como pré-processamento, os autores transformaram as imagens originais para localizar as *features*. As imagens resultantes são obtidas através do complemento do canal correspondente na imagem original. Na imagem original as folhas são verdes, sendo uma mistura do canal vermelho e verde. Na imagem complementar as folhas são roxas, devido à quantidade de verde ser menor comparativamente ao canal vermelho e azul. De forma a aumentar a quantidade de *dataset* para treino, também foram aplicadas técnicas de augmentação, em particular transformações geométricas como rotação e *flipping*. Como resultado, o número total de imagens final foi de 33600. Na Tabela 2.11 pode-se verificar a precisão dos diferentes modelos CNN avaliados. Concluí-se que o uso de algoritmos de deteção CNN tem pelo menos uma precisão de 94 %.

Tabela 2.11: Precisão dos modelos avaliados.

Modelos CNN	Precisão (%)
Alexnet	94.70
VGG16	96.85
ResNet101	97.24
ResNet18	95.68
DenseNet	94.70
GoogleNet	99.66

A aplicação móvel, *VitisView*, foi desenvolvida no *Android studio*. O utilizador pode aceder à câmara do *smartphone* ou fazer *upload* de uma imagem. Existe uma comunicação entre o cliente *Android* e um servidor (Figura 2.29), por *Hypertext Transfer Protoco* (HTTP), para envio da imagem a detetar e a respetiva classificação. Foi utilizada um *java servlet*, responsável por processar as mensagens HTTP, através do *software Apache Tomcat*.

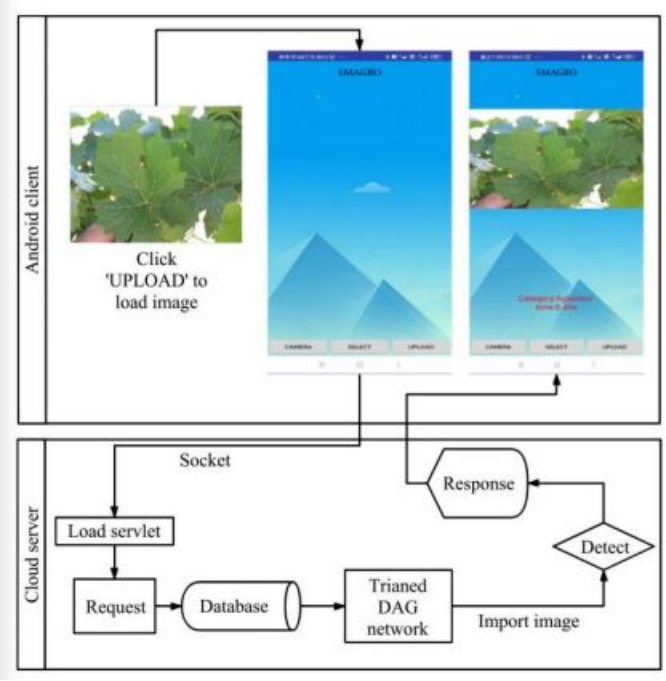


Figura 2.29: Diagrama de comunicação entre o cliente *Android* e o servidor. Retirada de [57].

Para testar o desempenho da aplicação foram utilizados dois *smartphones*. O Vivo X9 com o *Android* 7, 4 GB RAM e o processador *Qualcomm snapdragon 625* e o outro foi um MI 9 com o *Android* 10 , 6 GB RAM e o processor *Qualcomm snapdragon 855*. Ambos obtiveram uma precisão de deteção de 98 %, com 210 imagens que o modelo nunca analisou. O tempo médio para fazer o *upload* da imagem e obter resultados foi de 8.25 segundos (s) e 8.29 s, respetivamente.

2.8 Discussão do estado de arte

Numa análise aos trabalhos estudados, a Tabela 2.12 demonstra as conclusões obtidas relativamente aos algoritmos estudados pelos autores que apresentam melhor desempenho no respetivo cenário.

Tabela 2.12: Comparação de algoritmos propostos por diferentes autores, em sistemas de deteção automática integrados na agricultura. (n/d - não disponível)

Autor/Ano	**Algoritmo**	**Dataset/Img Treino**	**Precisão/Exatidão**
Tembhurne et al. 2023	*MobileNet*	12318/9854	94.4 % / n.d
Mukhiddinov et al. 2023	YOLOv4	12000/9600	49.3 % / n.d
Reda et al. 2023	*EfficientNet*	61486 / n.d	n.d / 99 %
Syamsuri et al. 2019	*MNasNet*	53899/32282	97 % / n.d
Dey et al. 2020	VGG19	93116/ n.d	n.d / 98.92 %
Liu et al. 2021	*GoogleNet*	33600/23520	n.d / 99.66 %

O objetivo dos artigos estudados é a integração do modelo *deep learning* numa aplicação móvel. Neste sentido, deve-se ter em consideração o desempenho do modelo mas também o tamanho do ficheiro do modelo e o consumo do *smartphone*. Na Tabela 2.13 pode-se verificar o modelo com melhor desempenho com uma aplicação *smartphone*, tópico estudado por alguns autores. *Syamsuri et al.* não especificou o tamanho do ficheiro do modelo mas concluíu que o do modelo *Inceptionv3* é quase quatro vezes superior comparativamente aos modelos *MobileNet* e *MNasNet*. Como é possível verificar na Tabela 2.12, *Dey et al.* determinou que o modelo VGG19 tem uma maior precisão para o *dataset* treinado, no entanto como é verificado na Tabela 2.13, o algoritmo *MobileNetv2* apresentou melhores condições relativamente ao consumo do *smartphone*.

Tabela 2.13: Comparação do consumo de recursos do *smartphone* para algoritmos propostos por diferentes autores, em sistemas de deteção automática integrados na agricultura.

Autor/Ano	**Algoritmo**	**CPU *load (%)***	***Power***	**Tamanho ficheiro**
Syamsuri et al. 2019	*MNasNet*	23.2	14.2 mAh	n/d
Dey et al. 2020	*MobileNetv2*	59	2746 mW	23.3 MB

As aplicações móveis estudadas apresentam diferentes funcionalidades, como é possível observar na Tabela 2.14.

Tabela 2.14: Comparação das funcionalidades das aplicações móveis propostas por diferentes autores, em sistemas de deteção automática integrados na agricultura.

Autor/Ano	**Funcionalidade**
Tembhurne et al. 2023	Identificação de espécies de plantas e doenças: Identificação da planta, da doença, sugestões de tratamento, cálculo de fertilizante e dicas de cultivo.
Mukhiddinov et al. 2023	Identificação de frutas e vegetais.
Reda et al. 2023	Identificação de espécies de plantas e doenças: Identificação da espécie e da doença, tratamentos, conselhos e histórico de classificações.
Syamsuri et al. 2019	Identificação de doenças de plantas.
Dey et al. 2020	Identificação de frutos e vegetais.
Liu et al. 2021	Identificação de espécies de videira.

Com o estudo do estado de arte, verificou-se a importância de um bom *dataset* e das transformações efetuadas nas imagens para que este se torne mais robusto. Percebeu-se as etapas necessárias antes do treino, a análise efetuada para classificar os modelos e o processo de inserir o modelo DL treinado numa aplicação móvel. Relativamente aos algoritmos *deep learning* utilizados no contexto de aplicações móveis, o modelo *MobileNet* é bastante comum nas análises efetuadas pelos autores, neste sentido é um modelo com interesse de estudo. Dos artigos apresentados, foi avaliado o modelo YOLOv4, no momento da presente dissertação, a arquitetura evoluiu até à versão YOLOv8, pelo que efetuar uma análise do desempenho desse modelo no contexto da agricultura vai permitir uma análise noutra perspetiva.

Capítulo 3

Implementação e Resultados

Os procedimentos realizados ao longo de um projeto são elementos fundamentais no seu desenvolvimento, dado que este é construído ao longo de várias etapas. Por esta razão neste capítulo são detalhadas as fases mais importantes, começando pela arquitetura do sistema, a formação do *dataset*, os procedimentos antes do treino, a avaliação dos modelos após o treino, a integração dos modelos DL na aplicação móvel e, por fim, a funcionalidade de descodificar *QR codes*.

3.1 Descrição da arquitetura

Na Figura 3.1 está descrita a arquitetura do sistema. Na aplicação móvel, o utilizador acede à câmara de modo a capturar o ambiente envolvente. Existe um bloco de processamento da imagem que, através de algoritmos de inferência dos modelos *deep learning* treinados e integrados no *smartphone*, deteta cachos de uvas, troncos de videira e *QR codes* e desenha *bounding boxes* como forma de visualização. Para se ter conhecimento da localização do objeto detetado, existe um bloco de localização com informação obtida através do sistema GNSS. Esta informação é enviada pelo protocolo de comunicação MQTT para uma base de dados (*Node-Red*).

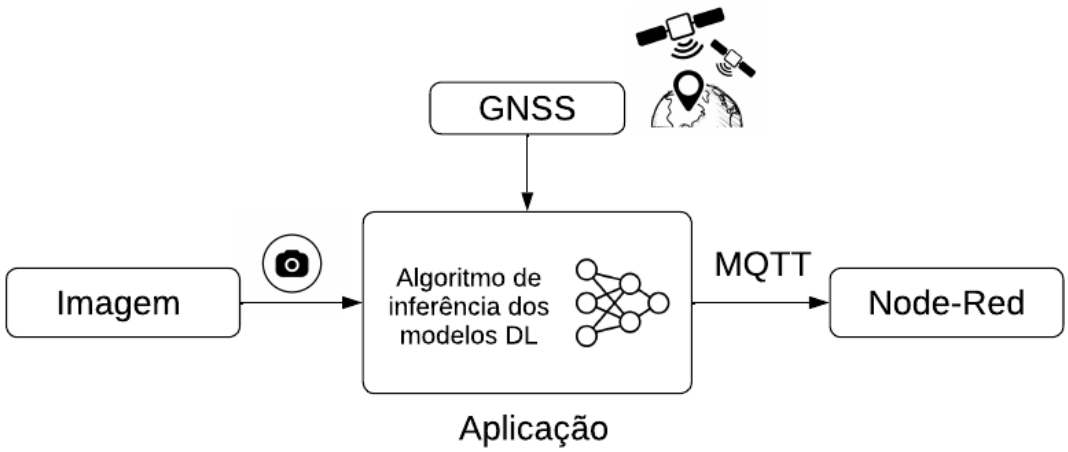


Figura 3.1: Diagrama da arquitetura do sistema.

3.2 Aquisição de Dados e Procedimentos

A recolha de dados é um dos processos mais importantes na preparação de algoritmos *deep learning*. Para a presente dissertação foram utilizados 3 *datasets* distintos. O *dataset* de cachos de videira [58], público no *Zenodo*, foi adquirido no Campus de Vairão, Vila de Conde, em 21 de julho e 11 de agosto de 2021. Foi também utilizado um *dataset* de cachos de videira e troncos [59], igualmente público no *Zenodo*. Por último, foram adquiridas imagens de *QR codes* dispostos em diferentes localizações da vinha, na Quinta do Seixo, Tabuaço, em março de 2023. Na Figura 3.2 pode-se observar 3 imagens, cada uma correspondente aos *datasets* referidos.

(a) *Dataset* de cachos de uvas - Resolução de 3000x4000 *pixels*.

(b) *Dataset* de *QR codes* - Resolução de 1536x2048 *pixels*.

(c) *Dataset* de cachos de uvas e troncos - Resolução de 1280x700 *pixels*, 1280 x 720 *pixels* e 1920 x 1080 *pixels*

Figura 3.2: Exemplo de imagens dos respetivos *datasets*.

A cada uma destas imagens foram realizadas anotações, através da ferramenta *open source* CVAT [60], considerando as classes *tiny_grape_bunch*, *medium_grape _bunch*, *trunk*, e *QRcode*. A diferença entre as classes *medium_grape _bunch* e *tiny_grape_bunch* é a fase de maturação da uva. Na Figura 3.3 é possível observar o ambiente do CVAT, no qual é necessário selecionar as regiões de interesse em cada imagem, identificando a respetiva classificação através de *bounding boxes*.

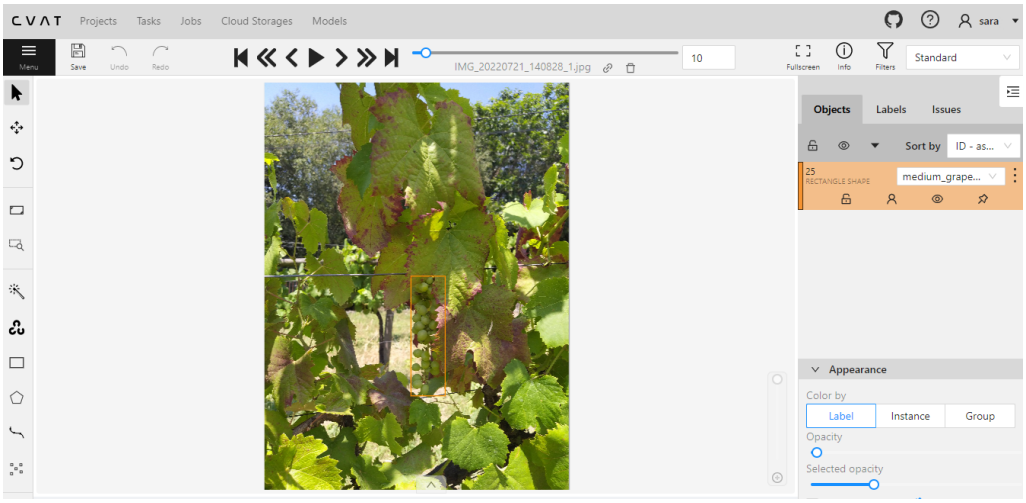


Figura 3.3: Ferramenta *open-source* CVAT.

Na tabela 3.1 está apresentado o número de anotações realizadas para cada classe.

Tabela 3.1: Quantidade de anotações para cada classe.

medium_grape	*tiny_grape*	*trunk*	*QRcode*
4615	2497	4381	264

A ferramenta CVAT permite fazer uma exportação das mesmas em diferentes formatos. Foi realizada a geração de ficheiros *.xml* (formato *Pascal VOC*) com as coordenadas dos objetos detetados. A preparação de um bom *dataset* é essencial para o sucesso do treino e desempenho do modelo DL e, neste sentido, procedeu-se à realização de um conjunto de transformações, através de um *script*.

A deteção de objetos em tempo real está associada a vários fatores imprevisíveis e não controláveis, especialmente em ambiente natural, como variações de iluminação, diferentes qualidades de imagem, sobreposição, entre outros. Para uma maior probabilidade de deteção, é necessário existir um *dataset* variado e que tenha em consideração os aspetos referidos. Na Figura 3.4 podem-se observar transformações efetuadas na mesma imagem (Figura 3.4a), como mudança no seu ângulo (Figura 3.4b), desfoque (Figura 3.4c), inversão horizontal e vertical (Figura 3.4d e 3.4e), saturação HUE (Figura 3.4f), ruído (Figura 3.4g), mudança de escala (Figura 3.4h) e translação (Figura 3.4i). As transformações referidas foram selecionadas devido a serem fatores que podem influenciar quando se efetua a deteção das classes em estudo. As anotações de cada imagem foram mantidas para cada transformação, sem penalizar as localizações das *bounding boxes*. Após todas as transformações, tendo em consideração os 3 *datasets*, foram obtidas 21632 imagens anotadas.

(a) Original. (b) Ângulo. (c) Desfocar.

(d) Inversão Horizontal. (e) Inversão Vertical. (f) HUE.

(g) Ruído. (h) Escala. (i) Translação.

Figura 3.4: Diferentes transformações realizadas ao *dataset*.

3.3 Treino e Avaliação de Redes Neuronais

A utilização de redes neuronais permite colmatar desvantagens nos métodos tradicionais de processamento de imagem para deteção e classificação de objetos, entre as

quais a imprevisibilidade inerente a imagens adquiridas em ambiente natural. Tendo isto em conta, a aprendizagem é a melhor abordagem para ultrapassar estes desafios.

O desempenho de uma rede neuronal depende do seu treino, dado que é a fase de aprendizagem. Posteriormente à preparação de um bom *dataset*, é necessário dividi-lo em treino e validação. Estas etapas são muito importantes para que a aprendizagem seja realizada com o melhor desempenho possível. O *dataset* foi dividido com uma proporção de 3:1, em que 75 % correspondem ao treino e os restantes 25 % à validação. Foram utilizadas 2 arquiteturas de *deep learning* distintas, a YOLO e o modelo SSD *MobileNetv2*, pré-treinado com o *dataset* da COCO [61]. Da rede YOLO, utilizou-se o modelo YOLOv5 e YOLOv8 em que foram selecionados 7 modelos pré-treinados. O modelo YOLOv5 e YOLOv8 são versões diferentes da mesma arquitetura, sendo o YOLOv8 mais recente. Para o mesmo tamanho de imagem, 640 *pixels*, cada versão disponibiliza 5 modelos (n, s, l, m, x). Estes diferem na velocidade e na precisão de análise, sendo que em ambas as versões, os modelos n e s são os mais rápidos [62] [63].

Todos os treinos foram realizados com recurso a uma GPU RTX 3090. O *batch size* refere-se à quantidade de imagens do *dataset* de treino que irá utilizar de forma sequencial. Por exemplo, se o seu valor for de 50, significa que em cada camada de treino serão utilizadas 50 amostras sucessivamente, até percorrer todas as camadas. Nos treinos dos modelos YOLO, com 300 *epochs* para o YOLOv8 e 150 *epochs* para o YOLOv5, foi utilizado um valor de *batch size* de 16. No treino da rede SSD *MobileNet*, o *batch size* foi de 8 para existir um menor consumo de memória e foram realizados 415000 *steps*, equivalente a 150 *epochs* [64].

Numa primeira abordagem ao *dataset*, foi necessário garantir uma organização de dados na estrutura *Pascal VOC*. Posteriormente, de forma a treinar as redes YOLO, utilizou-se um *script* para converter as anotações .xml para .txt e dividir o *dataset*. Na Figura 3.5 pode-se visualizar a organização de dados para treinar a rede YOLO.

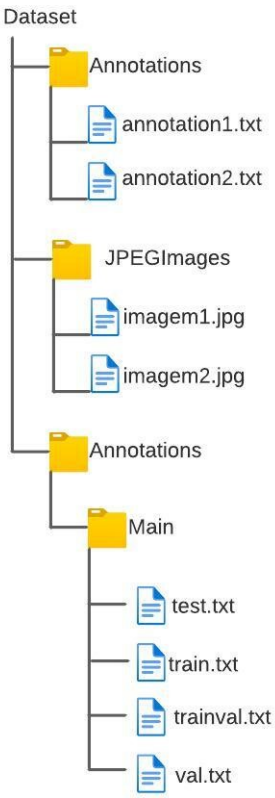


Figura 3.5: Organização de dados para treinar a rede YOLO.

Para o treino da rede *SSD MobileNetv2*, a partir do formato .txt, procedeu-se à conversão das anotações para um formato .csv, recorrendo a um *script* de conversão de formatos de ficheiros .txt para .csv. Por último, efetuou-se uma conversão para o formato compatível com *Tensorflow*, .record. Através da utilização de um *script* para a devida conversão, obteve-se a estrutura seguinte (Figura 3.6).

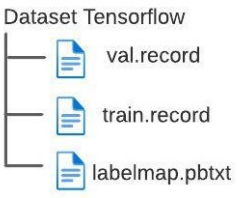


Figura 3.6: Organização de dados para treinar a rede *SSD MobileNetv2*.

Para a avaliação dos resultados, recorreu-se à análise de métricas, já referidas na Secção 2:

- *Recall*.

- Precisão.
- F1 *Score*.

Deve-se ter em consideração a particularidade de cada cultura para que seja possível adequar as metodologias num processo de automatização. Em diferentes fases do processo de maturação, as uvas têm características distintas. Na Figura 3.7 podem-se verificar 2 exemplos de imagens presentes nos *datasets* que demonstram a uva em estados diferentes de crescimento.

(a) Cacho de uvas em fase precoce de maturação.

(b) Cacho de uvas em fase mais avançada de maturação.

Figura 3.7: Características de cachos de uvas presentes no *dataset*.

A forma mais adequada de avaliar a viabilidade dos treinos é a partir da comparação dos resultados obtidos entre os diferentes modelos treinados. Com os dados descritos nas Tabelas 3.2 e 3.3 podem-se comparar os resultados relativamente à precisão, *recall* e F1 *score* para cada classe, tendo em consideração uma IoU de 50 % a 90 %. Para se obter estes resultados, utilizou-se um *dataset* de teste e a ferramenta *FittyOne* [65] de modo a comparar os resultados de *ground truth* com deteções efetuadas por cada modelo com uma confiança igual ou superior a 25 %. Este teste permite verificar a robustez do método, devido à análise ser efetuada em imagens com diferentes iluminações, uvas de diferentes tamanhos e vinhas distintas às imagens de treino.

Na Tabelas 3.2 e 3.3 observa-se que não existem resultados relativos à classe *tiny grape*. Isto deve-se ao facto de, no *dataset* de teste, não existirem imagens com *labels* desta classe.

Na análise à classe *medium grape*, verifica-se um valor relativamente elevado de precisão no modelo *MobileNetv2* (85 %) e um valor baixo de *recall* (11 %). Isto deve-se ao facto de que entre todas as deteções de *medium grape* efetuadas por este modelo, 85 % foram deteções corretas, no entanto, identificou incorretamente a maior parte das *labels* de *medium grape*. Um valor baixo de *recall*, devido a um elevado número de falsos negativos, traduz-se numa baixa capacidade de deteção da classe em questão. Relativamente aos modelos YOLO, observa-se que os modelos YOLOv5n e YOLOv5s destacam-se com os resultados mais equilibrados e elevados tendo em consideração tanto a precisão como o *recall*.

Na análise à classe de *QR code* verificam-se resultados bastante semelhantes. Os modelos que obtiveram valor de 75 % de *recall*, significa que tiveram falsos negativos.

Na deteção de troncos observa-se novamente uma elevada precisão e baixo valor de *recall* no modelo *MobileNetv2*. O modelo só identificou 1 tronco e de forma correta, pelo que de todas as deteções feitas, 100 % foram corretas. No entanto, de todas as *labels* de tronco só detetou 2 %, um valor bastante baixo. Relativamente aos modelos YOLO, o modelo YOLOv8n e YOLOv5n identificaram uma maior quantidade de troncos (17 %). No geral, os resultados foram baixos, pelo que uma análise futura é aumentar o *dataset* com imagens de troncos e mais variadas.

A métrica F1 *score* combina a métrica precisão e *recall* numa única avaliação, através de uma média harmónica. Na Tabela 3.3 verifica-se que os modelos YOLOv5n, YOLOv5s e YOLOv8s destacam-se os com melhores resultados. Como demonstrado anteriormente, o modelo *MobileNetv2* apresenta resultados baixos, pelo que o modelo deveria ter sido treinado com mais *steps* de modo a determinar se as métricas de avaliação e a precisão do modelo aumentavam.

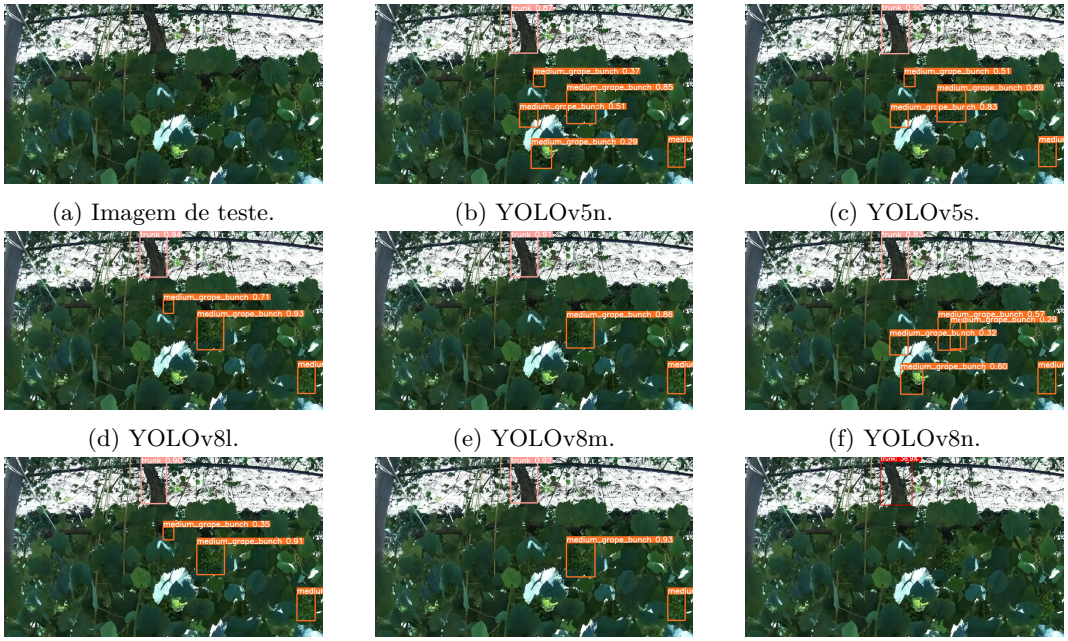
Tabela 3.2: Resultados percentuais da avaliação do desempenho dos modelos treinados, com um *dataset* de teste.

Modelo	*medium grape*		*QR code*		*trunk*	
	Precisão	*Recall*	Precisão	*Recall*	Precisão	*Recall*
YOLOv5n	75 %	71 %	100 %	75 %	36 %	17 %
YOLOv5s	79 %	69 %	100 %	100 %	33 %	15 %
YOLOv8l	79 %	48 %	100 %	100 %	27 %	5 %
YOLOv8m	77 %	50 %	100 %	75 %	47 %	13 %
YOLOv8n	69 %	68 %	100 %	100 %	33 %	17 %
YOLOv8s	78 %	53 %	100 %	75 %	35 %	10 %
YOLOv8x	79 %	52 %	100 %	75 %	40 %	10 %
SSD *MobileNetv2*	85 %	11 %	100 %	75 %	100 %	2 %

Tabela 3.3: Resultados da métrica F1 *score* dos modelos treinados, com um *dataset* de teste.

Modelo	*medium grape*	QR code	*trunk*
YOLOv5n	73 %	86 %	23 %
YOLOv5s	74 %	100 %	21 %
YOLOv8l	60 %	100 %	8 %
YOLOv8m	61 %	86 %	21 %
YOLOv8n	68 %	100 %	22 %
YOLOv8s	63 %	86 %	16 %
YOLOv8x	62 %	86 %	16 %
SSD *MobileNetv2*	20 %	86 %	3 %

Para se comparar os modelos, realizou-se inferências de deteção para imagens que os modelos nunca viram anteriormente. Na Figura 3.8, observa-se as deteções e precisões obtidas com cada modelo treinado, numa imagem de teste.



(a) Imagem de teste. (b) YOLOv5n. (c) YOLOv5s.

(d) YOLOv8l. (e) YOLOv8m. (f) YOLOv8n.

(g) YOLOv8s. (h) YOLOv8x. (i) *MobileNetv2*.

Figura 3.8: Comparação de resultados entre as redes neuronais treinadas, com a inferência de deteção da imagem de teste 1.

Na Figura 3.8, pode-se observar que relativamente aos modelos YOLOv5, o YOLOv5n destaca-se com mais deteções efetuadas corretamente. Da mesma forma, no modelo YOLOv8, o modelo YOLOv8n detetou uma maior quantidade correta de uvas e tronco. Como seria de prever, devido aos resultados de treino do modelo *MobileNetv2*, este só detetou uma classe de forma correta.

A Figura 3.9 representa os resultados obtidos na imagem de teste 2. Pode-se verificar que, relativamente aos modelos YOLO, todos detetaram corretamente os cachos de uvas. A diferença que se verifica é que o modelo YOLOv8n detetou incorretamente a classe referida, no entanto com uma confiança inferior (55 %). O modelo *MobileNetv2* detetou um cacho de uvas com confiança de 58.9 %, valor significativamente inferior comparativamente aos modelos YOLO.

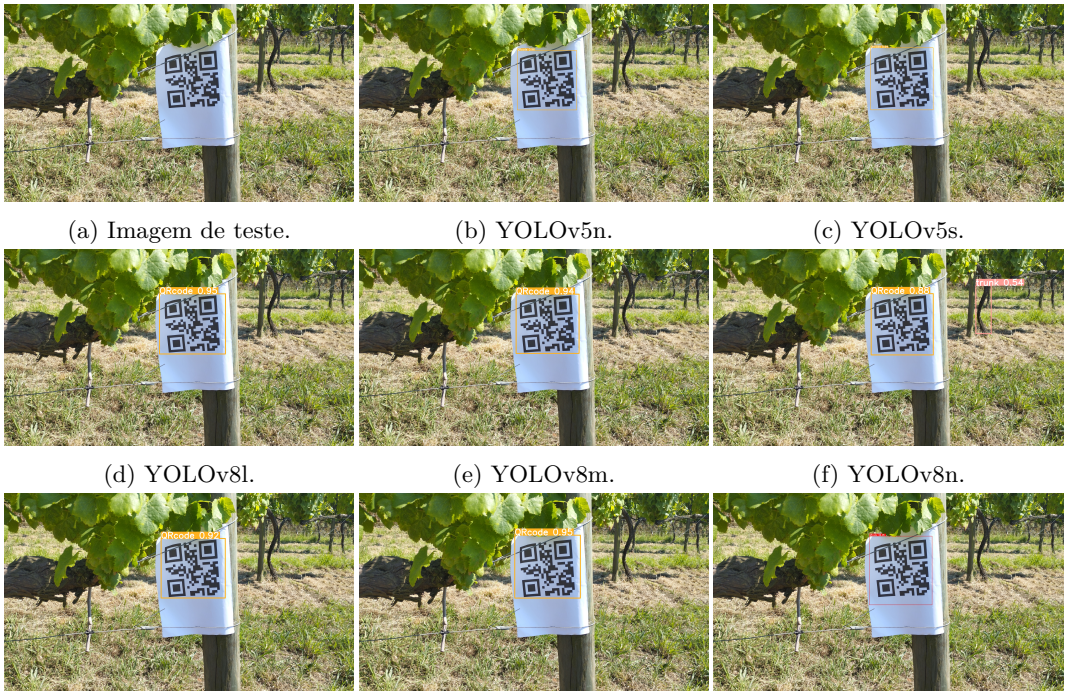
(a) Imagem de teste. (b) YOLOv5n. (c) YOLOv5s.

(d) YOLOv8l. (e) YOLOv8m. (f) YOLOv8n.

(g) YOLOv8s. (h) YOLOv8x. (i) *MobileNetv2*.

Figura 3.9: Comparação de resultados entre as redes neuronais treinadas, com a inferência de deteção da imagem de teste 2.

Na última imagem de teste, Figura 3.10, verifica-se uma correta deteção de *QR codes* por todos os modelos e com elevadas percentagens de confiança. O modelo YOLOv8n também detetou um tronco, enquanto nenhum dos restantes modelos o fez.



(a) Imagem de teste. (b) YOLOv5n. (c) YOLOv5s.

(d) YOLOv8l. (e) YOLOv8m. (f) YOLOv8n.

(g) YOLOv8s. (h) YOLOv8x. (i) *MobileNetv2*.

Figura 3.10: Comparação de resultados entre as redes neuronais treinadas, com a inferência de deteção da imagem de teste 3.

Para além das precisões de deteção, é fundamental ter em consideração o tempo de processamento, especialmente quando se pretende que a aplicação seja em tempo real. Os tempos apresentados na Tabela 3.4 foram obtidos utilizando o CPU i5 - 6200U de um portátil.

Tabela 3.4: Resultados do tempo de processamento dos diferentes modelos, no CPU de um portátil, em milissegundos.

Modelo	Figura 3.8	Figura 3.9	Figura 3.10
YOLOv5n	226.7	244.0	158.0
YOLOv5s	403.1	497.5	356.0
YOLOv8l	2122.9	3706.9	1808.3
YOLOv8m	1372.6	1541.2	1161.1
YOLOv8n	237.4	244.4	181.5
YOLOv8s	634.0	687.0	406.8
YOLOv8x	3783.6	2946.3	2665.0
SSD *MobileNetv2*	336.3	544.2	263.9

Comparando os tempos apresentados na Tabela 3.4, concluiu-se que os modelos

YOLOv5n, YOLOv5s e YOLOv8n destacam-se com uma baixa discrepância de tempos. O modelo SSD *MobileNetv2* apresentou igualmente um valor baixo de tempo de processamento, pelo que, neste parâmetro, apresenta uma característica essencial.

3.4 Aplicação Android

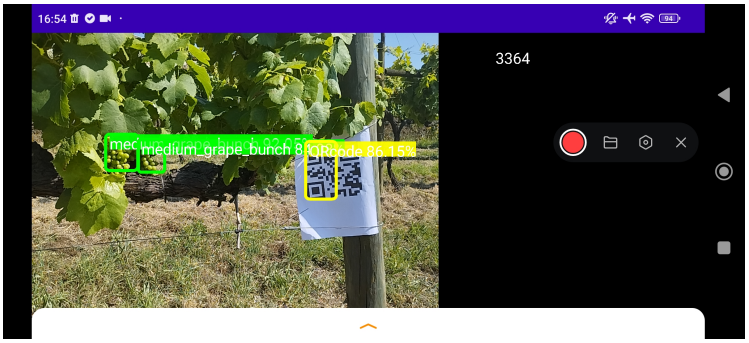
Para testar os modelos de DL treinados no contexto do uso de *smartphones*, foi utilizada a aplicação Orioos e outra aplicação desenvolvida com o objetivo de testar as redes YOLOv5. Nesta secção está descrito o desenvolvimento efetuado na aplicação *Android* Orioos e os resultados obtidos na comparação dos modelos treinados.

3.4.1 Integração de Modelos Deep-Learning

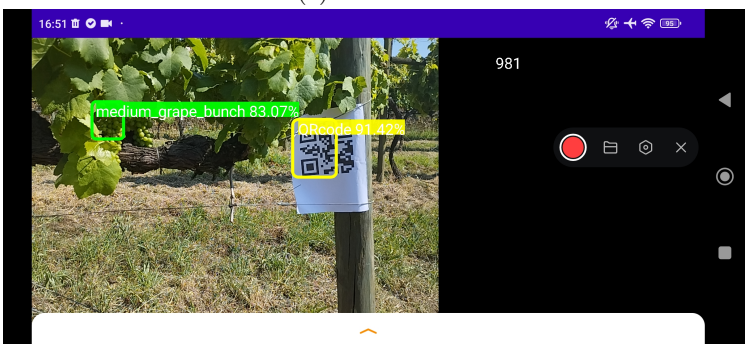
De forma a integrar os modelos de DL numa aplicação *Android*, recorreu-se a um *script* para efetuar a conversão dos mesmos para um ficheiro com o formato *TensorFlow Lite*. Foram treinados 5 modelos YOLOv8, no entanto surgiram complicações a integrá-los na aplicação pelo que estes não foram utilizados para comparação dos modelos no ambiente *Android*. Pelo contrário, os modelos YOLOv5 conseguem facilmente ser integrados, por isso foram treinados 2 modelos, o YOLOv5s e YOLOv5n. Esta escolha foi efetuada devido a serem os modelos mais rápidos dos 5 disponíveis [62].

A estrutura de inferência de modelos na aplicação *Android* é distinta para a rede YOLO e *MobileNet* devido a terem arquiteturas diferentes. Neste sentido, utilizou-se a aplicação Orioos para testar o modelo *MobileNet* e uma aplicação desenvolvida para testar os modelos YOLO. Em ambas as aplicações é possível observar a deteção efetuada, com o nome da classe e respetiva precisão de deteção. Para além disso, está também disponível o tempo de inferência de forma a comparar a velocidade dos modelos.

Foram efetuados testes em 3 localizações distintas com o *smartphone* Redmi Mi 11 com *Octa-core* (1x2.84 GHz *Cortex-X1*, 3x2.42 GHz *Cortex-A78* e 4x1.80 GHz *Cortex-A55*), 128 GB de memória e 6 GB de RAM. Em junho de 2023, no Campus Agrário de Vairão - Universidade do Porto, foram recolhidos resultados de inferências de deteção com as 2 aplicações referidas e com a integração dos modelos YOLOv5n, YOLOv5s e SSD emphMobileNetv2. Na Figura 3.11, verifica-se o comportamento dos diferentes modelos no ambiente *Android* para a mesma zona de deteção.



(a) YOLOv5s.



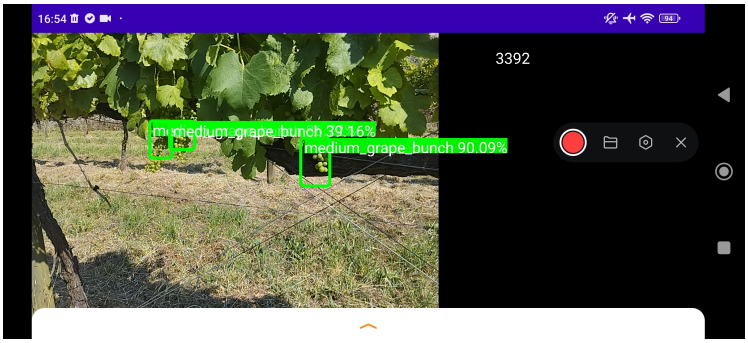
(b) YOLOv5n.

(c) SSD *MobileNetv2*.

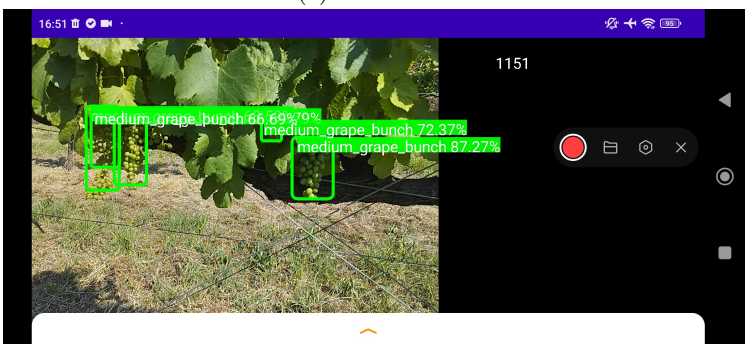
Figura 3.11: Comparação de resultados entre as redes neuronais em ambiente *Android*, com a inferência de deteção da imagem de teste 1.

Todos os modelos detetaram o *QR code* com uma confiança de 86.15 %, 91.42 % e 90 %, respetivamente. No entanto, só os modelos YOLO é que detetaram corretamente cachos de uvas e só o modelo YOLOv5s detetou os dois cachos presentes na imagem. Relativamente à deteção de troncos, nenhum modelo conseguiu detetar.

Na Figura 3.12 observa-se os resultados da inferência num local diferente da vinha. Os modelos YOLO detetaram corretamente os cachos de uvas, sendo que o modelo YOLOv5n efetuou mais deteções corretas. Pelo contrário, verifica-se que o modelo *MobileNetv2* não conseguiu efetuar nenhuma deteção.



(a) YOLOv5s.



(b) YOLOv5n.

(c) SSD *MobileNetv2*.

Figura 3.12: Comparação de resultados entre as redes neuronais em ambiente *Android*, com a inferência de deteção da imagem de teste 2.

Na mesma zona de análise, com o modelo SSD *MobileNetv2*, verifica-se na Figura 3.13 que este deteta um cacho de uvas quando há uma maior proximidade ao mesmo. No entanto, apresenta uma baixa confiança (52 %) e não detetou todos os cachos da imagem.

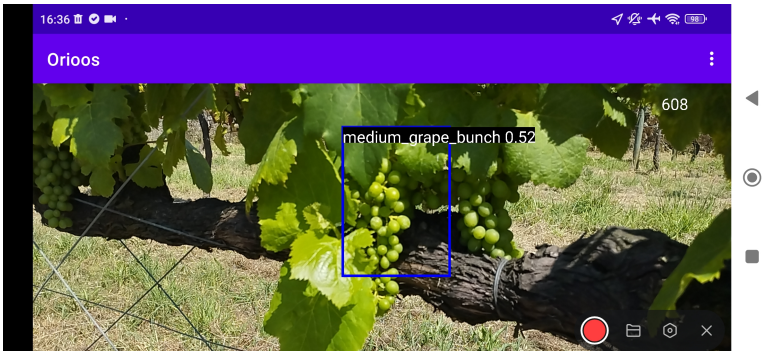
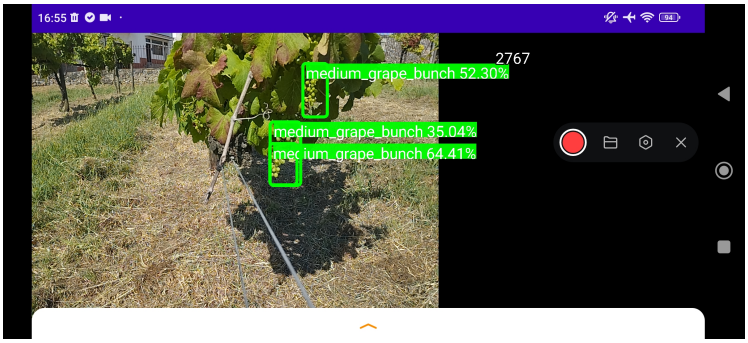
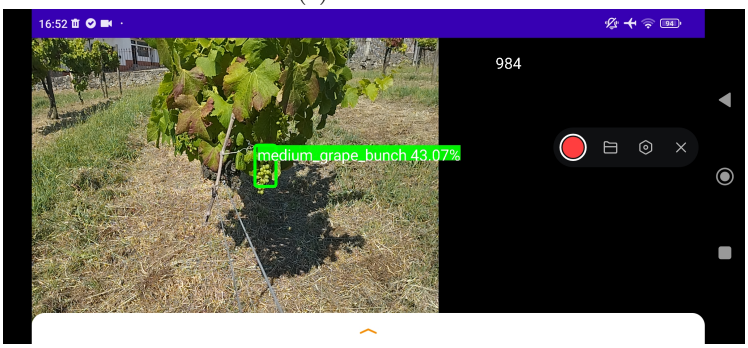


Figura 3.13: Inferência de deteção com o modelo SSD *MobileNetv2* na zona de análise da imagem de teste 2.

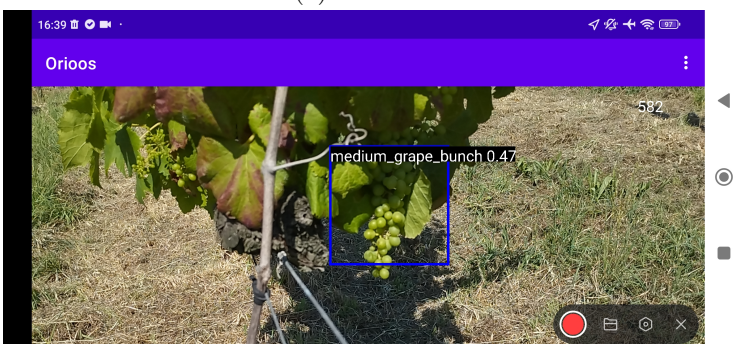
Na Figura 3.14 verifica-se novamente uma correta deteção de cachos de uvas pelos modelos YOLOv5, sendo que o modelo YOLOv5s detetou mais objetos de forma correta. Com a proximidade ao objeto da análise com os modelos YOLO, o modelo SSD *MobileNetv2* não obteve sucesso na deteção. No entanto, como foi referido anteriormente, devido a existir mais *zoom* do objeto, o modelo conseguiu detetar corretamente um cacho de uvas.



(a) YOLOv5s.



(b) YOLOv5n.



(c) SSD *MobileNetv2*.

Figura 3.14: Comparação de resultados entre as redes neuronais no ambiente *Android*, com a inferência de deteção da imagem de teste 3.

Na Figura 3.15 pode-se verificar a zona de análise onde se efetuou inferências de deteção de troncos e nenhum dos modelos conseguiu detetar troncos no ambiente. Realizou-se inferências no computador, de forma a verificar se o ambiente *Android* influenciava e os resultados foram os mesmos.

Figura 3.15: Zona de análise com troncos.

Para existir um registo da localização das diferentes classes detetadas pela aplicação Orioos, é enviado para o *Node-Red*, por MQTT, a georreferênciação das mesmas. A localização e envio de dados pela aplicação já estava previamente desenvolvida e é obtida através do GNSS do telemóvel. Na presente dissertação acrescentou-se informação a ser enviada caso alguma das classes dos modelos treinados fosse detetada, de modo a aparecer um ícone no mapa da respetiva deteção. Na Figura 3.16 observa-se a *interface* do *Node-Red* com uma deteção de um cacho de uvas.

Figura 3.16: *Interface* do *Node-Red* com deteção de um cacho de uvas.

De forma a verificar se existe diferença na inferência caso esta seja realizada no *smartphone* ou no computador, efetuou-se deteções na mesma zona de análise da Figura 3.11 e 3.12, num portátil com o CPU i5 - 6200U. Na Figura 3.17 verifica-se que os resultados são praticamente iguais à Figura 3.11. A diferença notória é no modelo YOLOv5n em que foi detetado um cacho de uvas incorretamente, no entanto, o modelo foi capaz de detetar um tronco.

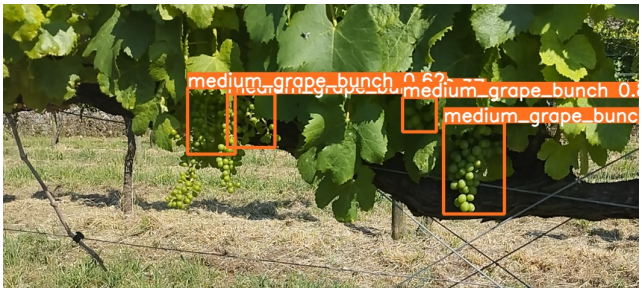
(a) YOLOv5s.

(b) YOLOv5n.

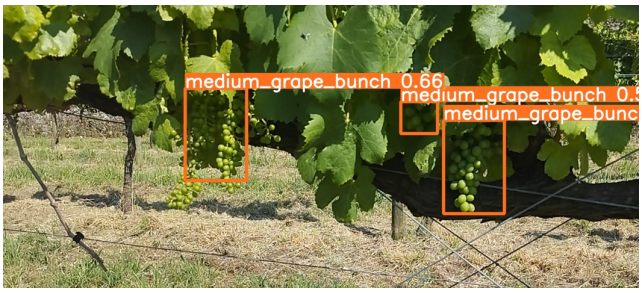
(c) SSD *MobileNetv2*.

Figura 3.17: Comparação de resultados entre as redes neuronais no computador, com a inferência de deteção da imagem de teste 1.

Realizou-se outra inferência no computador (Figura 3.18) para comparar a deteção com a Figura 3.12. Verifica-se que as deteções são bastante semelhantes e que o modelo *MobileNetv2* continuou sem ser capaz de fazer uma deteção correta.



(a) YOLOv5s.



(b) YOLOv5n.

(c) SSD *MobileNetv2*.

Figura 3.18: Comparação de resultados entre as redes neuronais no computador, com a inferência de deteção da imagem de teste 2.

Na Quinta do Seixo, em junho de 2023, obtiveram-se os resultados apresentados na Figura 3.19 e 3.20. Relativamente ao modelo SSD *MobileNetv2* (Figura 3.19), algumas das deteções foram um *QR code* com uma precisão de 100 % e 4 deteções de cachos de uvas com uma precisão máxima de 63 % e as restantes numa média de 30 %. O tempo de inferência é, no geral, de 400 milissegundos, valor bastante inferior aos valores previamente analisados da rede YOLO.

(a) Deteção de cacho de uvas.

(b) Deteção de cacho de uvas.

(c) Deteção de cacho de uvas.

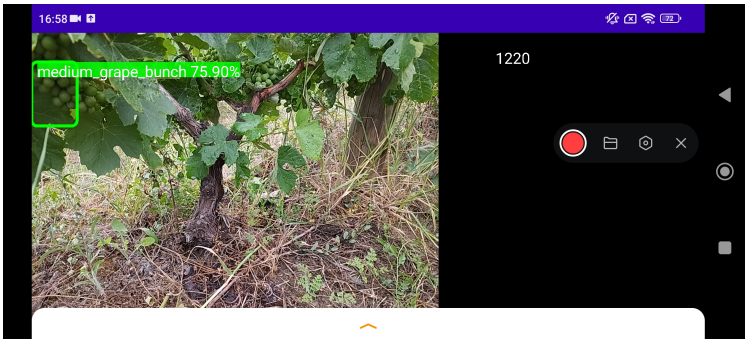
(d) Deteção de *QR code*.

(e) Deteção de cacho de uvas.

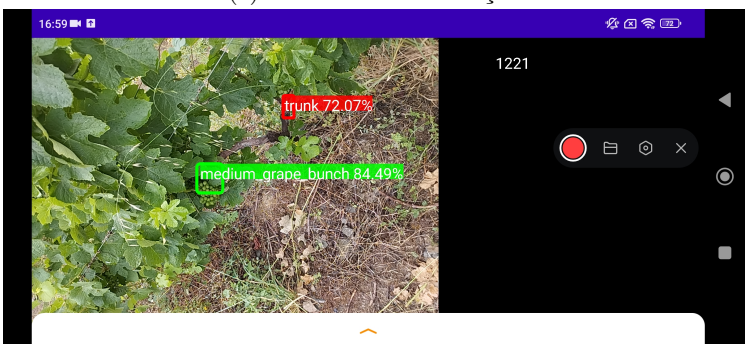
(f) Deteção de cacho de uvas.

Figura 3.19: Resultados da aplicação Orioos com o modelo SSD *MobileNetv2* - Quinta do Seixo.

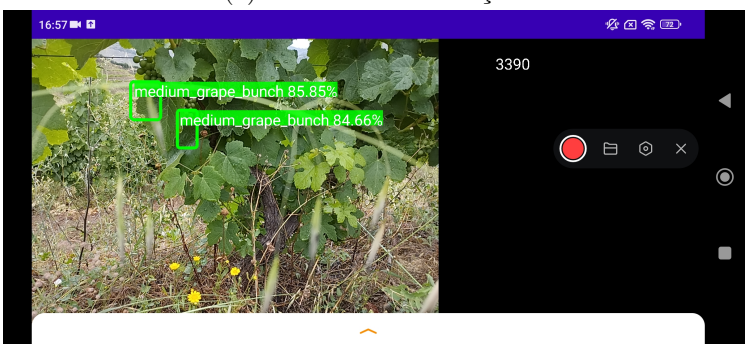
De seguida, testou-se os modelos YOLO e obteve-se os resultados da Figura 3.20. Com o modelo YOLOv5n obteve-se deteções de cachos de uvas e de um tronco com precisões relativamente elevadas. No modelo YOLOv5s observam-se 2 deteções de cachos de uvas com as *bounding boxes* ligeiramente fora do local real. Isto acontece devido ao tempo de inferência ser bastante elevado (3390 milissegundos), pelo que o movimento do telemóvel não é acompanhado pelas deteções em tempo real.



(a) YOLOv5n - 1º Deteção



(b) YOLOv5n - 2º Deteção.

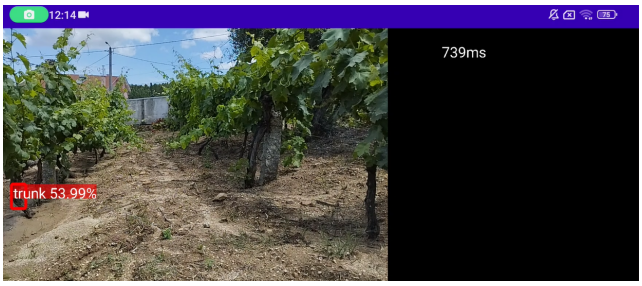


(c) YOLOv5s.

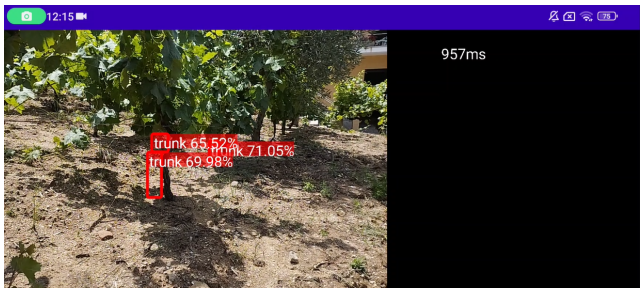
Figura 3.20: Resultados da aplicação com os modelos YOLO - Quinta do Seixo.

Por último, no início de junho de 2023, testou-se os modelos YOLO noutra vinha. Na Figura 3.21 observam-se alguns dos resultados obtidos. O modelo YOLOv5n detetou *QR codes*, troncos e cachos de uvas. No entanto, com o modelo YOLOv5s só se detetou um *QR code*.

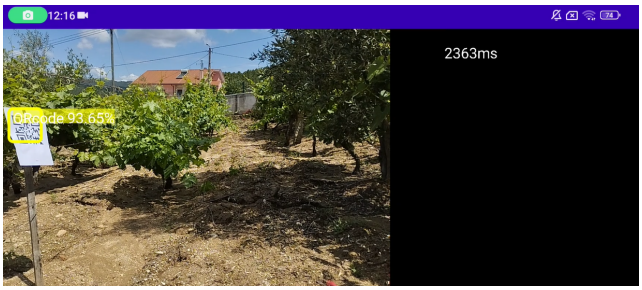
(a) YOLOv5n - 1º Deteção.



(b) YOLOv5n - 2º Deteção.



(c) YOLOv5n - 3º Deteção.



(d) YOLOv5s.

Figura 3.21: Resultados da aplicação com os modelos YOLO - Vinha.

Na Figura 3.22 está representada a *interface* do *Node-Red* com a deteção de um tronco e um ícone do mesmo representado no mapa. O ícone de veículo representa a localização do telemóvel.

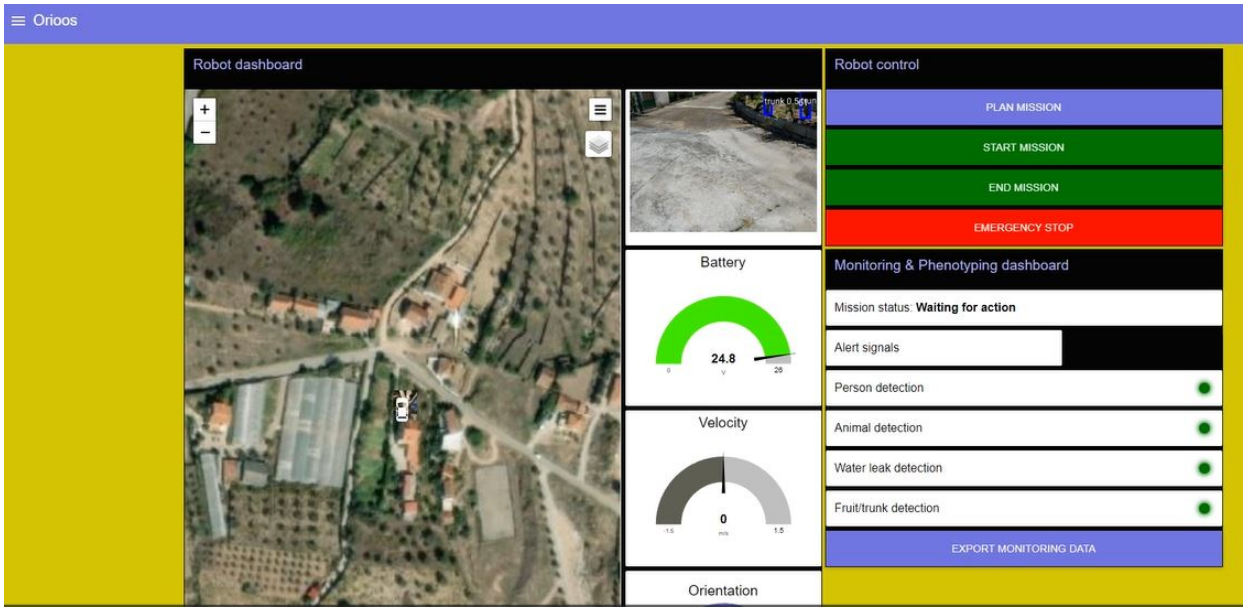


Figura 3.22: *Interface* do *Node-Red* com deteção de um tronco.

Nos testes da Figura 3.21 e na Quinta do Seixo, relativamene ao modelo YOLO, verificou-se mais deteções com o modelo YOLOv5n. Isto deve-se ao facto de ter sido

gravado um vídeo em andamento e o modelo YOLOv5s, como referido anteriormente, tem um tempo de inferência bastante elevado pelo que não funciona bem em deteções em tempo real.

Ambas as aplicação fornecem o tempo de inferência de modo a ser possível analisar qual o modelo mais rápido. Na Tabela 3.5 pode-se verificar a média de tempos de inferência entre 10 *frames*, para os modelo YOLOv5n, YOLOv5s e SSD *MobileNetv2*.

Tabela 3.5: Resultados do tempo de processamento dos diferentes modelos, na CPU de um *smartphone*, em milissegundos.

Modelo	YOLOv5n	YOLOv5s	SSD *MobileNetv2*
Tempo de Inferência	954.3	3012.3	639.3

Comparando os modelos YOLO, o modelo YOLOv5n é o mais rápido, sendo que existe um aumento de 215 % em comparação com o modelo YOLOv5s. Pode-se verificar que, apesar do modelo SSD *MobileNetv2* apresentar melhores tempos de inferência (33 % mais rápido comparando com o modelo YOLOv5n), este tem falhas na deteção das classes em estudo.

3.4.2 Descodificação de QR codes

Integrou-se na aplicação Orioos a descodificação de *QR codes*, quando estes são detetados pelas redes treinadas. Esta funcionalidade permite que o robô tenha capacidade de ter informação exterior e que estes sirvam como *landmarks* artificiais. A informação obtida e a localização do mesmo são enviadas para o *Node-Red*. Na Figura 3.23 é visível o comportamento da aplicação quando um *QR code* é detetado. Foi estabelecido um *time-out* de 5 segundos e, se a câmara deixar de detetar o *QR code* depois do tempo definido, é continuada a deteção dos objetos.

Figura 3.23: Descodificação de um *QR code* na aplicação Orioos.

Esta funcionalidade permite que, como trabalho futuro, a leitura dos *QR codes* possibilitem a obtenção de informação pelo robô. Alguns exemplos de utilização são

informações sobre a vinha ou a informação de localização da *docking station* mais próxima, de forma ao robô ter conhecimento para onde se deve deslocar.

Capítulo 4

Conclusões

Na presente dissertação, o objetivo consistia em desenvolver um sistema automático em tempo real para deteção de cachos de uvas, troncos de videira e *QR codes* numa aplicação móvel, em ambiente *Android*. Pretendia-se obter a georreferenciação no momento de deteção para que fosse possível controlar um robô por *waypoints*.

Com o desenvolvimento da dissertação concluiu-se que a qualidade e dimensão de um *dataset* são aspetos fundamentais para um modelo de *deep learning* robusto. Por esta razão, na Secção 3 foram destacadas as alterações realizadas nas imagens de modo a melhorar o *dataset*. Os resultados dos modelos YOLO revelaram-se os mais positivos, especialmente os modelos YOLOv5n, YOLOv5s e YOLOv8n. Para a deteção de uvas, verificou-se os melhores resultados para os modelos YOLOv5n e YOLOv5s, com um valor de F1 *score* de 73 % e 74 %, respetivamente. Na deteção de *QR codes*, todos os modelos apresentaram resultados elevados, entre valores de F1 *score* de 86 % e 100 %. Por outro lado, na deteção de troncos, os modelos obtiveram resultados menos positivos, sendo o valor mais elevado de F1 *score* de 23 %.

Com as imagens de teste e resultados apresentados foi possível verificar que o modelo SSD *Mobilenetv2* precisa de um maior número de *steps* para aumentar a sua precisão e melhorar a deteção, no entanto, isto não foi possível de realizar devido a dificuldades no uso do modelo SSD *MobileNet* e gestão de recursos. Relativamente ao tempo de inferência em CPU num computador, os modelos YOLOv5s, YOLOv5n, YOLOV8n e SSD *Mobilenetv2* apresentaram os resultados mais interessantes.

Com os testes de campo efetuados, verificou-se que os modelos YOLO apresentam uma melhor capacidade de deteção em distâncias mais longas relativamente aos

objetos a detetar, como também uma confiança mais elevada. Na mesma área de análise, o modelo SSD *MobineNetv2* necessitava de uma maior proximidade ao objeto para que este fosse detetado, em especial na deteção de cachos de uvas. Com uma das análises efetuadas no teste no Campus Agrário de Vairão - Universidade do Porto, foi demonstrado que nenhum modelo detetou corretamente troncos, pelo que se pode concluir que não há anotações suficientes ou variadas. No geral, os *QR codes* destacam-se como o melhor objeto detetado e com precisões mais elevadas. Os cachos de uvas são detetados pelos modelos, no entanto, devido ao ambiente envolvente ser semelhante, por vezes não são detetados todos os objetos devidos.

Relativamente ao tempo de inferência num *smartphone*, o modelo SSD *Mobilenetv2* demonstrou valores mais baixos de tempo de processamento, ao contrário dos modelos YOLO, especialmente o modelo YOLOv5s, que era lento e não conseguia acompanhar as mudanças de perspetiva. Em resumo, foi possível comparar diferentes modelos de *deep learning* com capacidade para detetar cachos de uvas, troncos e *QR codes*. Demonstrou-se um sistema integrado num *smartphone* capaz de detetar, com maior precisão, cachos de uvas e *QR codes*, com variações de iluminação e com frutos de diferentes dimensões.

No que diz respeito ao trabalho futuro proposto na presente dissertação destacam-se:

- Melhorar a qualidade do *dataset*, essencialmente uma melhoria nas anotações de troncos;
- Treino da rede SSD *MobileNetv2* com mais *steps* para verificar se existem melhorias;
- Explorar técnicas de navegação com recurso à informação obtida com os *QR codes* e a posição GNSS.

Contribuições

Os resultados obtidos na presente dissertação permitiram a realização de um artigo científico de conferência, submetido para a *Sixth Iberian Robotics Conference*.

Referências

[1] S. K. Behera, A. K. Rath, A. Mahapatra, and P. K. Sethy, "Identification, classification & grading of fruits using machine learning & computer intelligence: a review," [Citado na página 1]

[2] F. Li, P. Yang, K. Zhang, Y. Yin, Y. Zhang, and C. Yin, "The influence of smartphone use on conservation agricultural practice: Evidence from the extension of rice-green manure rotation system in china," vol. 813. [Citado na página 1]

[3] "Custos associados a uma viticultura sustentável PDF | PDF | irrigação | administração de terras." [Citado na página 2]

[4] L. M. G. F. Araújo, "Modelo de apuramento de custos de produção para a tomada de decisão: vinho branco e vinho tinto na região determinada do douro, estudo de caso." [Citado na página 2]

[5] "Evolução da produção nacional de vinho por região vitivinícola." `http://www.ivv.gov.pt/np4/home/163.html`, acedido a 23/02/2023. [Citado na página 2]

[6] Agroportal, "Previsão de colheita - campanha 2022/2023." `https://www.agroportal.pt/previsao-de-colheita-campanha-2022-2023/`, acedido a 23/02/2023. [Citado na página 2]

[7] M. Javaid, A. Haleem, I. H. Khan, and R. Suman, "Understanding the potential applications of artificial intelligence in agriculture sector," [Citado na página 2]

[8] G. F. Victorino, R. Braga, J. Santos-Victor, and C. M. Lopes, "Yield components detection and image-based indicators for non-invasive grapevine yield prediction at different phenological phases," vol. 54. [Citado nas páginas vii e 3]

[9] S. Coulibaly, B. Kamsu-Foguem, D. Kamissoko, and D. Traore, "Deep learning for precision agriculture: A bibliometric analysis," [Citado na página 3]

[10] R. a. M. ltd, "Global artificial intelligence in agriculture market (2022-2027) by offerings, technology, applications, geography, competitive analysis and the impact of covid-19 with ansoff analysis." `https://www.researchandmarkets.com/reports/5585680/`

global-artificial-intelligence-in-agriculture, acedido a 7/02/2023. [Citado na página 3]

[11] J. Koetsier, "Top apps of 2022 by installs, spend, and active users: Report." https://www.forbes.com/sites/johnkoetsier/2022/03/23/top-apps-of-2022-by-installs-spend-and-active-users-report/, acedido a 7/02/2023. [Citado na página 3]

[12] "Precision agriculture definition | international society of precision agriculture." https://www.ispag.org/about/definition, acedido a 20/02/2023. [Citado na página 7]

[13] A. Ali, T. Hussain, N. Tantashutikun, N. Hussain, and G. Cocetta, "Application of smart techniques, internet of things and data mining for resource use efficient and sustainable crop production," [Citado nas páginas vii, 8 e 9]

[14] "Precision farming metrics and benefits." https://www.farm21.com/understanding-precision-farming-metrics-and-how-you-can-benefit-from-them/, acedido a 20/02/2023. [Citado nas páginas 8 e 9]

[15] "Information and communication technologies (ICT) | unesco IIEP learning portal." https://learningportal.iiep.unesco.org/en/glossary/information-and-communication-technologies-ict, acedido a 23/02/2023. [Citado na página 9]

[16] F. Kuhlmann and E. Berg, "The farm as an enterprise – the european perspective." [Citado na página 9]

[17] "ICT in agriculture & digital farming." https://www.cropin.com/ict-in-agriculture, acedido a 23/02/2023. [Citado na página 10]

[18] G. Rapsomanikis, "Information and communication technology (ICT) in agriculture," [Citado nas páginas vii e 10]

[19] "Application of robotics in agriculture." https://robotnik.eu/robotics-applications-in-agriculture/. [Citado na página 11]

[20] P. Gonzalez-De-Santos, R. Fernández, D. Sepúlveda, E. Navas, and M. Armada, "Unmanned ground vehicles for smart farms," in *Agronomy - Climate Change and Food Security* (Amanullah, ed.), IntechOpen. [Citado na página 11]

[21] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in agriculture robotics: A state-of-the-art review and challenges ahead," vol. 10. [Citado na página 11]

[22] "Laboratory of robotics and IoT for smart precision agriculture and forestry | INESC TEC." https://www.inesctec.pt/en/laboratories/laboratory-of-robotics-and-iot-for-smart-precision-agriculture-and-forestry. [Citado nas páginas vii e 11]

[23] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," p. 53. [Citado nas páginas 12 e 13]

[24] A. Mishra, "Deep learning fundamental- important concepts." https://medium.datadriveninvestor.com/deep-learning-fundamental-important-concepts-59d7ae90901b, acedido a 5/03/2023. [Citado na página 12]

[25] "Uma abordagem intuitiva às redes neurais - LAMFO." https://lamfo-unb.github.io/2017/06/18/itro-ao-deep-learning/, acedido a 5/03/2023. [Citado nas páginas vii e 12]

[26] "Future of deep learning according to top AI experts of 2023." https://research.aimultiple.com/future-of-deep-learning/. [Citado na página 13]

[27] A. Rosebrock, "Intersection over union (IoU) for object detection." https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, acedido a 5/03/2023. [Citado nas páginas vii e 14]

[28] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, "Fundamental concepts of convolutional neural network," pp. 519–567. [Citado na página 14]

[29] S. Saha, "A comprehensive guide to convolutional neural networks — the ELI5 way." [Citado nas páginas vii e 14]

[30] I. Aziz, "Deep learning: An overview of convolutional neural network(CNN)," [Citado nas páginas vii e 15]

[31] "Max-pooling / pooling - computer science wiki." https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. [Citado nas páginas vii e 15]

[32] Y. Harjoseputro, I. P. Yuda, and K. P. Danukusumo, "MobileNets: Efficient convolutional neural network for identification of protected birds," vol. 10, no. 6, p. 2290. [Citado na página 15]

[33] J. Fan, "Designing light-weight networks for mobile applications," [Citado na página 15]

[34] "Using depthwise separable convolutions in tensorflow." https://machinelearningmastery.com/using-depthwise-separable-convolutions-in-tensorflow/, acedido a 5/03/2023. [Citado nas páginas vii e 16]

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. [Citado nas páginas vii, 16 e 17]

[36] A. Sharma, "Understanding a real-time object detection network: You only look once (YOLOv1)." [Citado na página 16]

[37] "What do we mean by mobile ecosystems?." https://rankingdigitalrights.org/2016/09/15/what-are-mobile-ecosystems/, acedido a 5/03/2023. [Citado na página 17]

[38] "What is a smartphone." https://www.techtarget.com/searchmobilecomputing/definition/smartphone, acedido a 5/03/2023. [Citado na página 17]

[39] T. Agarwal, "Android operating system : Introduction, features and applications." https://www.elprocus.com/what-is-android-introduction-features-applications/, acedido a 5/03/2023). [Citado na página 17]

[40] "Conectividade | desenvolvedores android | android developers." https://developer.android.com/guide/topics/connectivity, acedido a 5/03/2023. [Citado na página 17]

[41] "Sensores | desenvolvedores android | android developers." https://developer.android.com/guide/topics/sensors, acedido a 5/03/2023. [Citado na página 17]

[42] M. Pattinson, S. Tiwari, Y. Zheng, M. Campo-Cossio, R. Arnau, D. Obregón, A. Ansuategui, C. Tubio, I. Lluvia, O. Rey, J. Verschoore, L. Lenza, and J. Reyes, "Annual baška GNSS conference," [Citado nas páginas 18 e 19]

[43] "What is GNSS?." https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss. [Citado na página 19]

[44] R. Reis, J. Mendes, F. Neves dos Santos, R. Morais, N. Ferraz, L. Santos, and A. Sousa, "Redundant robot localization system based in wireless sensor

network," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 154–159. [Citado na página 19]

[45] S.-J. Lee, G. Tewolde, J. Lim, and J. Kwon, "QR-code based localization for indoor mobile robot with validation using a 3d optical tracking instrument," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 965–970. [Citado nas páginas vii, 19, 20 e 21]

[46] L. Blanger and N. S. T. Hirata, "An evaluation of deep learning techniques for qr code detection," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1625–1629. [Citado nas páginas vii, xi, 21 e 22]

[47] F. Neves Dos Santos, "Agricultural wireless sensor mapping for robot localization," vol. 417. [Citado nas páginas vii e 22]

[48] L. C. Santos, F. N. Santos, E. J. Solteiro Pires, A. Valente, P. Costa, and S. Magalhães, "Path planning for ground robots in agriculture: a short review," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 61–66. [Citado na página 23]

[49] L. Gentilini, S. Rossi, D. Mengoli, A. Eusebi, and L. Marconi, "Trajectory planning ROS service for an autonomous agricultural robot," [Citado nas páginas vii e 23]

[50] L. C. Santos, F. N. Santos, A. S. Aguiar, A. Valente, and P. Costa, "Path planning with hybrid maps for processing and memory usage optimisation," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 27–33. [Citado nas páginas vii e 24]

[51] J. V. Tembhurne, S. M. Gajbhiye, V. R. Gannarpwar, H. R. Khandait, P. R. Goydani, and T. Diwan, "Plant disease detection using deep learning based mobile application," [Citado nas páginas vii, 25 e 26]

[52] M. Mukhiddinov, A. Muminov, and J. Cho, "Improved classification approach for fruits and vegetables freshness based on deep learning," vol. 22, no. 21, p. 8192. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute. [Citado nas páginas vii, viii, xi, 26, 27, 28 e 29]

[53] G. Mongaras, "YOLOX explanation — mosaic and mixup for data augmentation." `https://medium.com/mlearning-ai/yolox-explanation-mosaic-and-mixup-for-data-augmentation-3839465a3adf`, acedido a 21/02/2023. [Citado nas páginas viii e 28]

[54] M. Reda, R. Suwwan, S. Alkafri, Y. Rashed, and T. Shanableh, "AgroAId: A mobile app system for visual classification of plant species and diseases using

deep learning and TensorFlow lite," vol. 9, no. 3, p. 55. [Citado nas páginas viii, 29 e 31]

[55] B. Syamsuri and I. Negara, "Plant disease classification using lite pretrained deep convolutional neural network on android mobile device," [Citado nas páginas viii, 31, 32 e 33]

[56] S. Dey, S. Saha, A. Singh, and K. McDonald-Maier, "FruitVegCNN: Power- and memory-efficient classification of fruits & vegetables using CNN in mobile MP-SoC," in *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1–7. [Citado nas páginas viii, 33 e 34]

[57] Y. Liu, J. Su, L. Shen, N. Lu, Y. Fang, F. Liu, Y. Song, and B. Su, "Development of a mobile application for identification of grapevine (vitis vinifera l.) cultivars via deep learning," no. 5, pp. 172–179. Number: 5 Publisher: Association of Overseas Chinese Agricultural, Biological and Food Engineers(AOCABFE), Chinese Society of Agricultural Engineering(CSAE). [Citado nas páginas viii, 35, 36 e 37]

[58] G. Moreira, M. Cunha, and F. N. dos Santos, "GVxmi | grapevine bunches dataset." [Citado na página 42]

[59] A. S. Aguiar and S. Magalhães, "Grape bunch and vine trunk dataset for deep learning object detection.." [Citado na página 42]

[60] "CVAT." `https://www.cvat.ai/`. [Citado na página 43]

[61] "COCO - common objects in context." `https://cocodataset.org`. [Citado na página 46]

[62] G. Jocher, "YOLOv5 by ultralytics." `https://github.com/ultralytics/yolov5`. [Citado nas páginas 46 e 54]

[63] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by ultralytics." [Citado na página 46]

[64] "Steps-VS-epochs - machine-learning." `https://dengking.github.io/machine-learning/Theory/Deep-learning/Guide/Batch-epoch-step/Steps-VS-epochs/`, acedido a 1/06/2023. [Citado na página 46]

[65] "Voxel51 // open source computer vision tools for machine learning." [Citado na página 48]