UNIVERSIDADE D
COIMBRA

Bruno António dos Santos Cabanas Cardoso

AGRICULTURA INTELIGENTE

DETEÇÃO INTELIGENTE DE PRAGAS EM IMAGENS

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes orientado pelas Professora Doutora Catarina Helena Branco Simões da Silva, Professora Doutora Joana Madeira Martins Costa e Professora Doutora Bernardete Martins Ribeiro e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2021

Esta página foi deixada intencionalmente em branco

Resumo

No nosso dia-a-dia somos dependentes das plantas para nos alimentarmos, produzir medicamentos ou cosméticos. A quantidade e qualidade das plantas cultivadas é afetada por pragas que restringem o crescimento de plantas e se propagam pelas plantações agrícolas. A monitorização e controlo destas pragas são importantes para maximizar uma produção agrícola de qualidade. Este controlo pode ser feito quimicamente, usando pesticidas ou biologicamente, recorrendo a armadilhas para contabilizar a população da praga e libertando insetos predadores para controlar essa população.

O controlo de pragas, através da identificação de insetos presos em armadilhas, depende da correta identificação da espécie dos insetos. A precisão com que se deteta e classifica o tipo de inseto, ou seja, o tipo de praga é fundamental para evitar perdas em plantações agrícolas. O âmbito desta dissertação é o estudo de métodos inteligentes para deteção de pragas utilizando visão por computador e a implementação de vários modelos de deteção de objetos para deteção da praga mosca-branca presa em armadilhas amarelas colocadas numa plantação de tomate.

O desenvolvimento da solução proposta passou pelo estudo da arquitetura de modelos de deteção de objetos e de classificação de imagens, escolha e pré-processamento de um conjunto de dados contendo imagens de armadilhas com a praga mosca-branca, redução e aumento desse conjunto de dados, implementação de diferentes modelos de deteção de objetos para deteção da praga mosca-branca, comparação do desempenho dos modelos, validação das deteções feitas pelos modelos e desenvolvimento de um protótipo. O aumento do conjunto de dados foi feito em colaboração com um aluno de licenciatura que ficou, sob minha orientação, responsável por gerar novas imagens de armadilhas utilizando transformações geométricas e por gerar novas imagens de moscas-brancas usando redes adversárias generativas.

O sistema proposto para detetar moscas-brancas em armadilhas pegajosas amarelas apresenta uma precisão até 89,7%. O seu desempenho compete com o dos técnicos especializados, permitindo identificar a mosca-branca com elevada precisão, rapidamente e numa fase inicial do seu desenvolvimento. A validação por parte dos técnicos, das previsões feitas confirmou que a solução proposta consegue detetar até 39,25% mais moscas-brancas do que os anotadores do conjunto de dados utilizado, com um tempo médio de deteção por imagem de 0,13 segundos.

O protótipo desenvolvido permite demonstrar que é possível tornar acessível a deteção da mosca-branca, utilizando visão por computador, através do *upload* de imagens existentes ou usando a câmara do dispositivo para obter uma deteção imediata.

Palavras-Chave

Armadilhas amarelas pegajosas, Classificação de imagens, Controlo integrado de pragas, Deep learning, Deteção de objetos, Deteção de pragas, Mosca-branca, R-CNN, Redes neuronais, Visão por computador, YOLO.

Abstract

On a daily basis we are dependent on plants to eat, do photosynthesis, or produce medicines and cosmetics. The quantity and quality of these cultivated plants is affected by pests that restrict their growth and spread through agricultural crops. Thus, monitoring and control these pests is essential to maximize the quality of agricultural production. This control can be done chemically (through pesticides) or biologically (counting the pest population in traps and releasing predatory insects to control that population).

Biological pest control, through the identification of insects in traps, depends on the identification of their species. Thus, the precision with which it detects and classifies the type of insect, that is, the type of pest, is fundamental to avoid crop losses.

The objectives of this internship are the study of intelligent methods for pest detection using computer vision and the implementation of a pest detection model for detection of whiteflies in yellow sticky traps placed in a tomato crop.

The development of the proposed solution involved the study of the architecture of object detection and image classification models, choice and pre-processing of a dataset containing images of yellow sticky traps, dataset reduction and dataset augmentation, implement different object detection models for whitefly detection, comparison of model performance, validation of the detections made by the models and development of a prototype. The dataset augmentation was done in collaboration with a graduate student who was, under my guidance, responsible for generating new images of traps using geometric transformations and new images of whiteflies using generative adversary networks.

The proposed system, to detect whiteflies on yellow sticky traps, is competitive with the performance of technicians, allowing to identify the whitefly with high precision, quickly and at a earliest stage of its development. The validation, by the technicians, of the predictions made by the object detection models, confirmed that the proposed solution can detect up to 39.25% more whiteflies than the data set annotators with an average image detection time of 0.13 seconds.

The prototype allows to demonstrate that it is possible to easily use computer vision to detect whiteflies, by uploading existing images or using the device's camera to obtain immediate detection.

Keywords

Computer vision, Deep learning, Image classification, Integrated pest management, Neural network, Object detection, Pest detection, R-CNN, Whitefly, Yellow sticky traps, YOLO.

Agradecimentos

Agradeço à Professora Doutora Catarina Helena Branco Simões da Silva, Professora Doutora Joana Madeira Martins Costa e Professora Doutora Bernardete Martins Ribeiro, orientadoras desta dissertação, pela disponibilidade e pelas correções, sugestões e melhoramentos sugeridos nas diferentes etapas do trabalho e que contribuíram para melhorar o resultado final apresentado.

Esta página foi deixada intencionalmente em branco

Índice

Acrónimos

Adam – Adaptive moment estimation

ANN – Artificial Neural Network

AUC – Area Under the Curve

CNN – Convolutional Neural Network

CSPN – Cross Stage Partial Network

Faster R-CNN – Faster Region-based CNN

Fast R-CNN – Fast Region-based CNN

FPN – Feature Pyramid Networks

GAN – Generative Adversarial Network

GLCM – Grey Level Co-occurrence Matrice

HSI – Hue, Saturation, Intensity

HSV – Hue, Saturation, Value

IoU – Intersection-over-Union

KNN – K Nearest Neighbors

LBP – Local Binary Patterns

mAP – mean Average Precion

MDC – Minimum Distance Classifier

NMS - Non-Maximum Suppression

OAA – One Against All

OAO – One Against One

PANet – Path Aggregation Network

PCA – Principal Component Analysis

R-CNN – Region-based CNN

RGB – Red, Green, Blue

RoI – Region of Interest

RPN – Region Proposal Network

SPP – Spatial Pyramid Pooling

SVM – Support Vector Machine

TL – Transfer Learning

WF – Whitefly

YOLO – You Only Look Once

Esta página foi deixada intencionalmente em branco

Lista de Figuras

1 Introdução

1.1 Enquadramento do tema

No início da década 90, um surto da mosca-branca nos Estados Unidos causou 500 milhões de dólares em prejuízos. Na América do Sul, em regiões que já adotavam sistemas de combate a pragas, os efeitos da mosca-branca não eram tão sentidos, havendo um benefício líquido anual de 2400 dólares por hectare para as plantações de tomate que adotavam técnicas de controlo (Oliveira *et al.*, 2001; Ortiz, 2007).

Atualmente, a evolução da tecnologia deu à sociedade a capacidade de produzir alimentos suficientes para satisfazer mais de 7,8 mil milhões de consumidores. No entanto, esta capacidade de produção e segurança alimentar continua ameaçada por fatores como as alterações climáticas, o declínio de espécies polinizadoras, as doenças em plantas e pragas. As pragas, para além de serem uma ameaça à segurança alimentar à escala global, também podem ter consequências para os pequenos agricultores que dependem de pequenas plantações como meio de subsistência. Cerca de 50% do rendimento potencial da produção agrícola mundial é perdido (em parte devido a pragas) e 80% dessa produção é gerada por pequenos agricultores.

No futuro, devido ao crescimento da população mundial em 30% até 2050 terá de se aumentar o tamanho das plantações agrícolas e a produção terá de crescer 70% para continuar a satisfazer a procura (Mohanty *et al*., 2016; Hughes e Salathe, 2020).

A indústria agrícola responde à praga mosca-branca utilizando 3 técnicas para evitar prejuízos: a prevenção, o controlo químico e o controlo biológico. Como a prevenção e o controlo biológico não são 100% eficazes e a utilização exagerada de pesticidas prejudica a qualidade da plantação e é dispendioso, torna-se necessário conjugar as 3 técnicas utilizando o controlo integrado de pragas. Se a eficácia do controlo biológico for maximizada poder-se-á melhorar o sistema integrado de pragas, reduzindo a necessidade de pesticidas e assim, ajudar o sector agrícola a reduzir substancialmente os prejuízos sem prejudicar a qualidade das plantações e diminuindo custos.

Por estas razões têm sido desenvolvidos vários esforços para evitar a perda de colheitas agrícolas devido a pragas e, mais recentemente surgiu a agricultura inteligente como forma de enfrentar os desafios de produtividade, impacto ambiental, segurança alimentar e sustentabilidade.

Nesse âmbito podem ser desenvolvidos sistemas automáticos de monitorização e controlo biológico de pragas compostos por câmaras fotográficas e modelos de classificação ou deteção de objetos (sendo os objetos os insetos que se pretendem detetar).

Atualmente, o controlo biológico é feito por técnicos que se deslocam regularmente às plantações para identificar os insetos presentes em armadilhas e fazer a sua contabilização, para desse modo controlar a população das pragas.

A utilização de sistemas automáticos de monitorização e controlo evitam que técnicos tenham de se deslocar tão frequentemente às plantações agrícolas para identificar e verificar a quantidade de pragas presas em armadilhas, tornando o processo de

identificação e contabilização de pragas menos dispendioso, mais rápido e, em alguns casos, mais preciso. (Mohanty *et al*., 2016; Hughes e Salathe, 2020).

Nesta dissertação estudam-se metodologias para detetar e identificar pragas em plantações agrícolas. Começa-se por descrever os tipos de pragas existentes, de seguida descrevem-se os métodos utilizados para deteção de pragas, posteriormente identificam-se as limitações dos métodos existentes e por último propõe-se um método de deteção de pragas recorrendo à visão por computador.

1.2 Objetivos e Relevância

Esta tese tem como **objetivo** a implementação de um modelo de deteção de objetos para detetar e contabilizar a praga mosca-branca em imagens de armadilhas pegajosas amarelas colocadas numa plantação de tomate.

A inclusão de modelos de deteção de objetos em sistemas automáticos de monitorização e controlo de pragas é **relevante,** pois permite fazer o controlo integrado de pragas de forma mais barata e eficiente. Isto permite aos agricultores diminuir a quantidade de pesticidas utilizada, melhorando a qualidade dos alimentos e também aumentar o aproveitamento da plantação agrícola, enquanto diminui a necessidade de recursos humanos (técnicos especializados).

O objetivo a alcançar é **significativo**, porque apenas recentemente se começou a desenvolver a deteção de pragas através de modelos de deteção de objetos e porque tem potencial para obter um desempenho superior à utilização de modelos de classificação de imagens.

As etapas que contribuem para a prossecução dos objetivos são:

- Pesquisa do estado da arte de deteção de pragas agrícolas. Nesta pesquisa inclui-se o tipo de pragas existentes, diferentes tipos de controlo de pragas, metodologias tradicionais e inteligentes de deteção de pragas;
- Pesquisa de conjuntos de dados com imagens de pragas agrícolas. Esta etapa é relevante, porque a falta de conjuntos de dados públicos anotados de imagens de armadilhas é um dos maiores obstáculos à resolução do problema;
- Estudo dos dois principais modelos de deteção de objetos. Abordam-se também os modelos de classificação de imagens e como podem ser adaptados à resolução do problema;
- Implementação de modelos de deteção de objetos para detetar a praga mosca-branca em imagens de armadilhas pegajosas amarelas. Esta fase inclui o processamento do conjunto de dados, a criação de novos conjuntos de dados a partir do conjunto de dados original, a implementação de modelos de deteção de objetos, a comparação desses modelos e a escolha do modelo com melhor desempenho para criar um protótipo de deteção da mosca-branca.

1.3 Plano de trabalhos

O plano de trabalhos divide-se por dois semestres. No primeiro semestre estudou-se o estado da arte e iniciou-se a implementação dos modelos de deteção de objetos. No segundo semestre continuou-se a implementação dos modelos de deteção de objetos e desenvolveu-se um protótipo.

1.3.1 Primeiro semestre

No primeiro semestre começou por se estudar o estado da arte da deteção de pragas em plantações agrícolas. De seguida pesquisaram-se os métodos usados para detetar pragas em armadilhas. Identificaram-se os principais problemas da utilização de sistemas inteligentes para deteção de pragas e os métodos utilizados para os resolver. De seguida pesquisaram-se os conjuntos de dados públicos para utilização no desenvolvimento dos modelos de deteção de objetos e estudou-se a arquitetura desses modelos de deteção de objetos. Por último, modificou-se o conjunto de dados escolhido para que possa ser utilizado pelos diferentes modelos de deteção de objetos, implementaram-se 2 modelos de deteção de objetos (*R-CNN* e *YOLOv5 (Xlarge)*) e obtiveram-se os primeiros resultados provisórios.

1.3.2 Segundo semestre

No segundo semestre implementaram-se outros 2 modelos de deteção de objetos (*Scaled-YOLOv4* e *YOLOv5 (Small)*) e criaram-se vários conjuntos de dados (aumentados, reduzido e diferentes divisões treino/teste) a partir do conjunto de dados original. Os resultados dos diferentes modelos, no conjunto de dados escolhido, foram avaliados tendo em conta vários parâmetros: precisão (mAP), tempo médio de deteção por imagem, número de moscas-brancas detetado e a memória ocupada. De seguida as deteções do modelo escolhido foram enviadas para um técnico (especialista em deteção de moscas-brancas) validar os resultados. Após a validação dos resultados anotaram-se as instâncias de mosca-branca que o modelo estava a detetar corretamente, mas não estavam anotadas e avaliou-se novamente o desempenho desse modelo. Após obter a validação dos técnicos geraram-se novas imagens de armadilhas utilizando uma rede adversária generativa para tentar melhorar o desempenho do modelo escolhido. De seguida desenvolveu-se um protótipo de uma aplicação para deteção da mosca-branca fazendo o carregamento de imagens ou acedendo à câmara do dispositivo.

1.4 Organização da Dissertação

A dissertação é composta por 6 capítulos. O primeiro capítulo destina-se aos objetivos propostos para o trabalho, organização da dissertação e enquadramento do tema onde é explicada a importância da deteção de pragas em plantações agrícolas. No segundo capítulo expõem-se alguns dos tipos de pragas, os diferentes controlos de pragas e as principais metodologias tradicionais e inteligentes utilizadas no combate a pragas. No terceiro capítulo descrevem-se os modelos de classificação de imagens e deteção de objetos que podem ser usados para fazer a deteção automática de pragas. No quarto capítulo descreve-se a solução proposta para deteção de pragas e a arquitetura dos modelos de deteção de objetos que fazem parte dessa solução. No quinto capítulo descreve-se o trabalho realizado e os resultados obtidos. Neste capítulo descrevem-se as iterações efetuadas para criar o modelo de deteção da mosca-branca em armadilhas amarelas: definição e alteração feitas ao conjunto de dados escolhido, implementação de 4 modelos de deteção de objetos, comparação dos resultados obtidos pelos 4 modelos de deteção de objetos, validação dos resultados por técnicos e desenvolvimento de um protótipo. No sexto capítulo apresentam-se as conclusões e o trabalho futuro.

Esta página foi deixada intencionalmente em branco

2 Controlo de pragas

Nos últimos anos a utilização de pesticidas está a ser complementada com outros métodos de controlo de pragas. Para estes métodos resultarem é necessário identificar corretamente e o mais cedo possível as pragas. De seguida identificam-se alguns tipos de pragas, os diferentes tipos de controlo de pragas e os métodos tradicionais e inteligentes utilizados no combate a pragas.

2.1 Pragas em plantações agrícolas

O tipo de pragas a combater depende, entre outros fatores, do tipo de plantação agrícola e da estação do ano. Estas pragas passam por diversas fases de desenvolvimento (ovo, infância e adultas) e quanto mais cedo forem detetadas mais rapidamente podem ser combatidas.

2.1.1 Pragas

Nesta dissertação pretende-se detetar a praga mosca-branca que afeta as plantações de tomate. Algumas espécies mais comuns de moscas-brancas são a *Trialeurodes vaporariorum* e a *Bemisia tabaci* (Figura 1). Estes dois tipos de mosca-branca são muito semelhantes e apenas um técnico especializado (por exemplo biólogo) após formação consegue distingui-las facilmente. Nesta dissertação apenas interessa identificar o inseto como sendo mosca branca (e não a sua espécie) nas suas diversas fases de desenvolvimento.

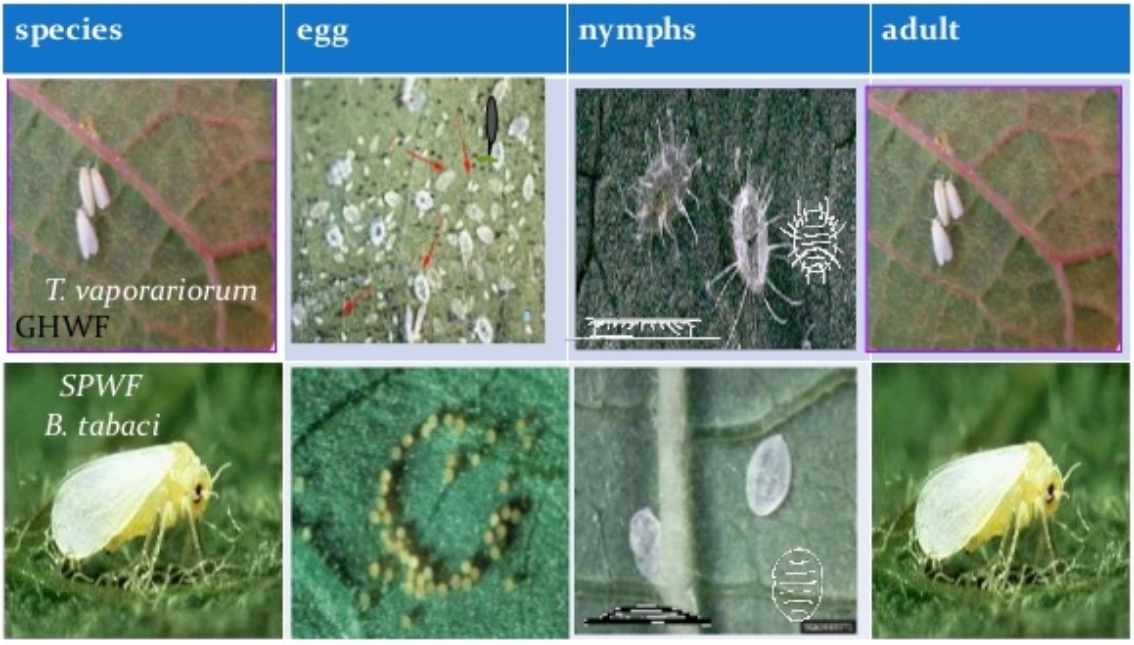


Figura 1 – Diferentes fases de dois tipos da praga mosca-branca *Trialeurodes vaporariorum* e *Bemisia tabaci* (Borude, *et al.*, 2017).

Outras pragas comuns que atacam plantações de fruta são a *Ceratitis capitata*, *Grapholita* e a *Drosophila suzukii* (Figura 2). Estes insetos nalgumas localidades já estão habituados a meios urbanos e rurais, o que dificulta o seu controlo.

a) b) c)

Figura 2 – Exemplos de pragas de fruta: a) *Ceratitis capitata*, b) *Grapholita* e c) *Drosophila suzukii* (Martins *et al.*, 2019).

A produção de arroz é afetada por pragas como a *Chilo suppressalis*, *Sesamia inferens*, *Cnaphalocrocis medinalis* e a *Pararaguttata bremerert* (Figura 3).

a) b) c) d)

Figura 3 – Exemplos de pragas de arroz na Asia: a) *Chilo suppressalis*, b) *Sesamia inferens*, c) *Cnaphalocrocis medinalis* e d) *Pararaguttata bremerert* (Qing *et al.*, 2012).

2.1.2 Controlo de pragas

Com o objetivo de manter as populações das pragas abaixo de um nível que cause dano económico podem utilizar-se as seguintes técnicas:

- **Prevenção:** A prevenção das pragas pode ser feita mantendo a plantação limpa, mantendo uma exposição solar adequada, utilizando um sistema de rega que debite apenas a quantidade de água necessária e cultivando na plantação agrícola, ervas que sejam dissuasoras de insetos propagadores de pragas;
- **Controlo químico:** O controlo químico é feito através da escolha e utilização de pesticidas adequados ao tipo de praga a combater;
- **Controlo biológico:** O controlo biológico é feito libertando insetos predadores para atacarem as pragas e usando armadilhas para detetar e controlar a população das pragas;
- **Controlo integrado de pragas:** Combinação da prevenção com os produtos químicos e controlo biológico.

O controlo integrado de pragas (Figura 4) permite fazer uma monitorização, deteção e combate mais eficaz do que a utilização individual da prevenção, controlo químico e

controlo biológico. Também permite a diminuição da quantidade de pesticidas usados no controlo químico. Nesta dissertação pretende-se tornar o controlo biológico mais eficaz, de modo a ser possível diminuir a componente química do controlo integrado de pragas, sem perder eficácia e diminuindo os custos.

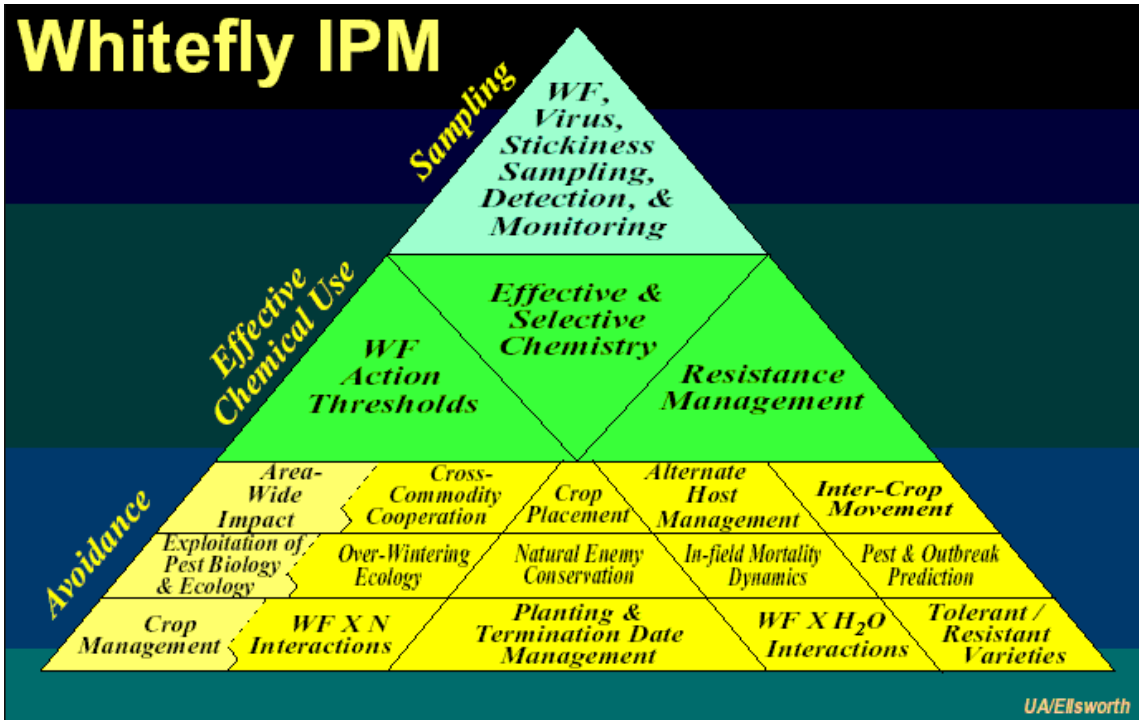


Figura 4 – Controlo integrado de pragas (Palumbo, 2002).

2.2 Armadilhas utilizadas no controlo biológico

As armadilhas são utilizadas para capturar os insetos propagadores de pragas. O tipo de armadilha utilizado depende da praga que se pretende controlar. Podem ser usadas, entre outras, armadilhas adesivas, pegajosas e de luz. Estas armadilhas para atraírem os tipos de insetos pretendidos, podem conter uma cor, feromonas ou alimentos que sejam atrativos para os insetos.

2.2.1 Armadilhas pegajosas

As armadilhas utilizadas normalmente no controlo biológico da praga mosca-branca são compostas por retângulos amarelos (cor que atrai a mosca-branca) que contêm um material pegajoso na sua superfície para reter os insetos (Figura 5a). Este tipo de armadilhas, dependendo do tipo de plantação pode ter ao fim de alguns dias centenas de moscas-brancas retidas. Nesta dissertação pretendia-se usar imagens destas armadilhas pegajosas amarelas, obtidas em estufas de plantações de tomate.

As armadilhas pegajosas vermelhas (Figura 5b), devido à sua cor são utilizadas para atrair outro tipo de pragas como a *Drosophila suzukii* (Roosjen et *al.*, 2020).

a) b)

Figura 5 – Armadilhas pegajosas: a) Amarelas e b) Vermelha (Greenvass, 2020 e Roosjen *et al.*, 2020).

2.2.2 Armadilhas adesivas e de luz

As armadilhas adesivas brancas podem ser colocadas no chão das plantações (Figura 6a) e conter sobre a armadilha feromonas ou alimentos para atrair os insetos de interesse. Entre outros insetos, estas armadilhas são utilizadas no controlo das pragas *Ceratitis capitata* e *Grapholita* (Martins et *al.*, 2019).

O controlo biológico das plantações de arroz que são afetadas pelas pragas *Chilo suppressalis*, *Sesamia inferens*, *Cnaphalocrocis medinalis* e *Pararaguttata bremerert* normalmente é feito utilizando uma armadilha composta por lâmpadas de luz negra (Figura 6b) que são utilizadas para atrair e prender os insetos (Qing e*t al.*, 2012).

a) b)

Figura 6 – a) Armadilha adesiva branca e b) Armadilha luminosa (Martins *et al.*, 2019, Roosjen *et al.*, 2020 e Qing *et al.*, 2012).

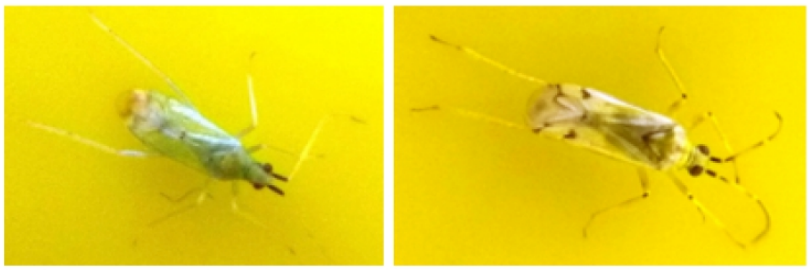
2.3 Métodos de combate a pragas e suas limitações

O controlo biológico de pragas é composto por insetos propagadores de pragas, insetos predadores, técnicos especializados e armadilhas.

2.3.1 Métodos tradicionais de controlo biológico

As armadilhas são espalhadas pela plantação agrícola que, dependendo do seu tamanho, pode conter centenas de armadilhas. As pragas são atraídas pela cor, feromonas ou alimentos das armadilhas e deslocam-se para a sua superfície acabando por ficar retidas devido à superfície pegajosa da armadilha. Um técnico especializado escolhe uma amostra da totalidade das armadilhas para fazer o controlo e monitorização das pragas. Esse técnico desloca-se à plantação agrícola semanalmente (dependendo do tamanho da plantação) para identificar os insetos presentes na amostra escolhida e contabilizar os insetos que são pragas. Se a evolução, de uma visita para a seguinte ultrapassar um limite definido são libertados insetos predadores para atacar as pragas e assim fazer um controlo biológico. Os

insetos predadores que normalmente são libertados para atacar a praga mosca-branca são o *Macrolophus pygmaeus* (Figura 7a) e o *Nesidiocori* (Figura 7b).



a) b)

Figura 7 – Insetos predadores da mosca-branca: a) *Macrolophus pygmaeus* e b) *Nesidiocori* (Nieuwenhuizen *et al.*, 2018).

Este método apresenta algumas desvantagens:

- **Trabalho excessivo e demorado:** Apesar de apenas ser necessário uma amostra das armadilhas para fazer o controlo das pragas, uma plantação pode conter centenas de armadilhas e cada armadilha conter centenas de insetos retidos.
- **Dispendioso:** Como alguns insetos são muito semelhantes é necessário a intervenção de um técnico especializado para identificar os insetos presentes nas armadilhas e fazer a contabilização das pragas.
- **Não é possível obter informação sincronizada:** Quando o técnico acaba de contabilizar o número de insetos presente na última armadilha da amostra o número de insetos presente nas armadilhas contabilizadas anteriormente já está desatualizado. Portanto esta técnica não consegue refletir com exatidão o comportamento real da uma população de pragas.
- **Propenso ao erro:** É um trabalho demasiado demorado, que exige concentração e a presença de diversas espécies de insetos semelhantes pode provocar erros.
- **Deteção feita demasiado tarde:** É mais fácil identificar visualmente os insetos na fase adulta do que em ovo ou na infância, o que pode levar a que apenas sejam contabilizadas as pragas na fase adulta.
- **Falta de precisão:** O técnico escolhe uma área da armadilha (normalmente 1 faixa nos 2 lados maiores da armadilha) para contabilizar a presença de pragas e extrapola essa quantidade para a restante área da armadilha.

2.3.2 Métodos inteligentes de controlo biológico

Nesta dissertação pretende-se utilizar sistemas automáticos de monitorização para minimizar os problemas identificados no método tradicional de controlo biológico. Nos métodos inteligentes (Figura 8 e Figura 9) usam-se câmaras para capturar imagens do topo das armadilhas, por exemplo semanalmente e, posteriormente utilizam-se modelos de classificação ou de deteção de objetos para identificar e contabilizar automaticamente os insetos presentes nas imagens capturadas. Se a evolução semanal do número de pragas presente nas imagens das armadilhas ultrapassar um valor limite são libertados insetos predadores para atacar as pragas.

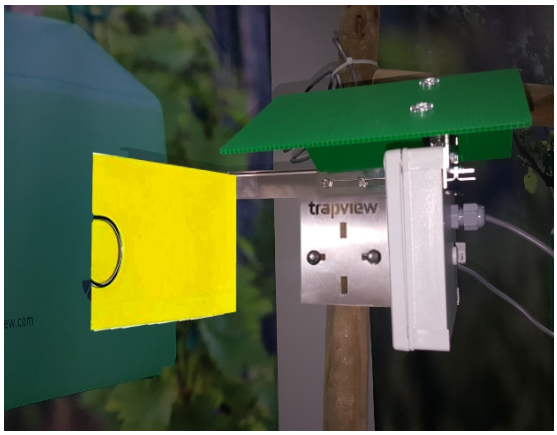


Figura 8 – Sistema automático de monitorização de armadilhas pegajosas amarelas (Trapview, 2020).

O método inteligente permite diminuir os custos do controlo biológico porque diminui a necessidade de técnicos especializados para identificar e contabilizar os insetos presentes nas armadilhas, permite fazer um controlo sincronizado e dependendo da precisão do modelo de classificação ou deteção de objetos pode melhorar a precisão da deteção de pragas (não é necessário fazer a extrapolação do número de moscas-brancas em cada armadilha, tornando a contagem mais exata).

Figura 9 – Sistema automático de monitorização de armadilhas pegajosas amarelas (AgroCares, 2020).

Apesar das vantagens em relação aos métodos tradicionais de controlo biológico, os métodos inteligentes têm algumas limitações:

- **Necessário grande conjunto de dados:** Atualmente existem muito poucos conjuntos de dados públicos com imagens de armadilhas anotadas e todos eles contêm problemas (imagens mal anotadas e falta de anotações). A anotação do conjunto de dados é uma tarefa demorada que tem de ser feita por um técnico especializado devido à semelhança entre insetos.
- **Pragas evoluem ao longo do tempo:** É necessário que o modelo de classificação ou deteção de objetos consiga identificar os insetos propagadores de pragas nas suas várias fases (ovo, infância e adulto).
- **Grande variedade e concentração de insetos:** Embora as armadilhas tenham feromonas ou alimentos para capturar apenas os insetos de interesse, na prática pode ser capturado uma grande variedade de espécies que podem ser identificadas como pragas pelos modelos de classificação ou deteção de objetos.
- **Sobreposição:** As imagens das armadilhas podem conter por exemplo folhas de plantas que impossibilitem a deteção de insetos que estejam debaixo dessas folhas (Martins *et al.*, 2019).

2.4 Conclusão

Com a utilização das imagens do topo das armadilhas é possível construir um sistema de monitorização automático que permita diminuir a necessidade dos técnicos especializados que normalmente fazem o controlo biológico de pragas, através da identificação e contabilização das pragas presentes nas armadilhas.

As imagens anotadas pelos técnicos podem ser utilizadas para treinar modelos de classificação ou deteção de objetos que depois de treinados podem ser usados para fazer uma monitorização biológica e um controlo automático de pragas agrícolas.

Nesta dissertação utilizam-se imagens de armadilhas pegajosas amarelas para detetar a praga mosca-branca. A utilização destas imagens é relevante porque permite integrar o controlo biológico no controlo integrado de pragas de forma mais fácil, eficiente e económica e também diminuir o uso de pesticidas para proteger a saúde ambiental e melhorar a qualidade da plantação agrícola.

Esta página foi deixada intencionalmente em branco

3 Estado da Arte

Os sistemas de monitorização automáticos de pragas podem ser compostos por modelos de classificação de imagens ou modelos de deteção de objetos. Neste capítulo descrevem-se os diferentes modelos de deteção de objetos e as etapas necessárias para adaptar a classificação de imagens à deteção de pragas.

3.1 Classificação de imagens

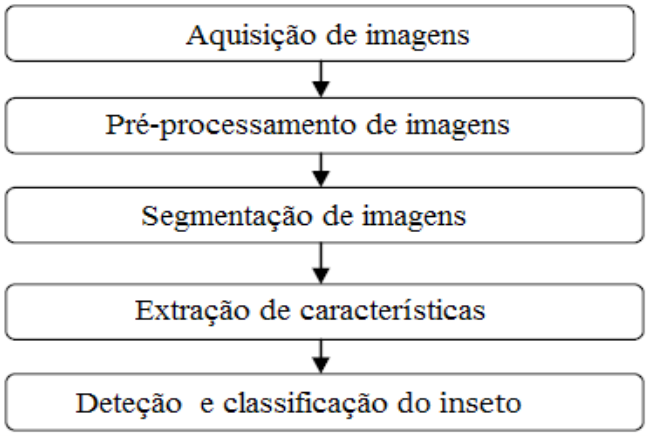


Figura 10 – Esquema dos passos envolvidos na deteção de pragas em armadilhas (Patel e Joshi, 2017).

A classificação de imagens pode ser usada para deteção de pragas, segmentando todos os insetos presentes numa imagem de uma armadilha e posteriormente fazer a classificação individual de cada objeto, ou seja, de cada inseto.

A classificação de imagens envolve algumas etapas de processamento da imagem (Figura 10). Essas etapas são a aquisição de imagens através da Internet ou câmara digital, pré-processamento das imagens para aprimorar os dados, por exemplo removendo distorções, segmentação da imagem para obter as partes mais relevantes de cada imagem, extração das características mais importantes e por último a deteção e classificação dos insetos (Gavhale e Gawande, 2014; Patel e Joshi, 2017).

3.1.1 Aquisição de imagens

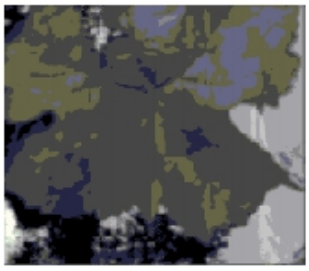
O primeiro passo de qualquer sistema de visão por computador é a aquisição de imagens utilizando uma câmara fotográfica. No caso da deteção de pragas, este passo envolve fotografar com alta resolução as armadilhas colocadas nas plantações agrícolas. Estas imagens podem ser obtidas na Internet ou nos campos agrícolas e o desempenho do sistema de classificação vai depender da qualidade do conjunto de imagens que se consegue construir. Após se obterem das imagens das armadilhas, os insetos contidos nessas imagens têm de ser manualmente anotados por um especialista para identificar as pragas (Patel e Joshi, 2017).

3.1.2 Pré-processamento

A qualidade da imagem é melhorada removendo ruído, distorções indesejadas e realçando as características mais importantes. Esta fase inclui a conversão do sistema de cores, aumento do contraste entre elementos e "suavização" da imagem.

Um dos métodos para melhorar o contraste entre cores é fazer a conversão do sistema da imagem de RGB para HSV (Hue, saturation, value) ou HSI (Hue, Saturation, Intensity). A remoção do fundo da imagem, neste caso dos píxeis correspondentes à armadilha pode ser feita utilizando um modelo de mistura gaussiana (*Gaussian mixture models*), onde se determina estatisticamente as intensidades mais prováveis para o fundo e os píxeis que não correspondem a essas intensidades são chamados de primeiro-plano (Gavhale e Gawande, 2014; Patel e Joshi, 2017).

a)



b)

Para remover ruído pode-se "branquear" o fundo das imagens identificando a percentagem da cor de fundo (normalmente amarelo) presente em cada píxel. Se tiver menos de uma percentagem limite da cor de fundo é lhe atribuída a cor branca. Com este método consegue-se aumentar a rapidez de processamento e a precisão.

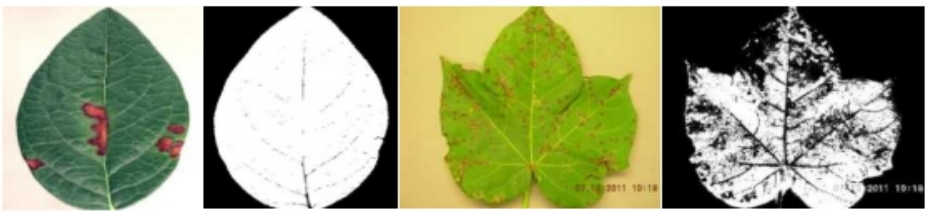
Pode-se melhorar a qualidade da imagem usando a difusão anisotrópica (Figura 11) para remover o ruído das imagens, preservando os elementos importantes como bordas e linhas. Isso é feito criando um conjunto de imagens em que cada imagem resultante é a combinação entre a imagem original e um filtro que depende do conteúdo local da imagem original. As imagens resultantes vão ficando cada vez mais "suaves". Podem-se utilizar filtros médios, medianos, máximos, etc. que percorrem cada píxel da imagem e substituem o seu valor, por exemplo pela média dos valores vizinhos (Khirade e Patil, 2015).

Figura 11 – Aplicação do método difusão anisotrópica: a) imagem original. b) imagem final (Khirade e Patil, 2015).

3.1.3 Segmentação

O processo de segmentação é feito de forma a repartir cada imagem em segmentos de acordo com as características de cada imagem. Alguns dos métodos mais utilizados são baseados em regiões, onde os píxeis relacionados com um objeto da imagem são agrupados, baseados em bordas, onde são identificadas bordas dos objetos para identificar descontinuidades nas imagens, baseados num valor limite ou baseados em agrupamento (Khirade e Patil, 2015; Patel e Joshi, 2017).

O método Otsu (Figura 12) baseado em valores limites, é um dos mais utilizados. Este método tem como objetivo dividir uma imagem em tons de cinzento determinando um valor limite que separe os elementos do fundo e da frente da imagem em dois grupos e posteriormente é atribuída a cor preto ou branco a cada um deles. Este método é particularmente adequado para imagens com histogramas bimodais (Kutty *et al.*, 2013; Torok, 2015).

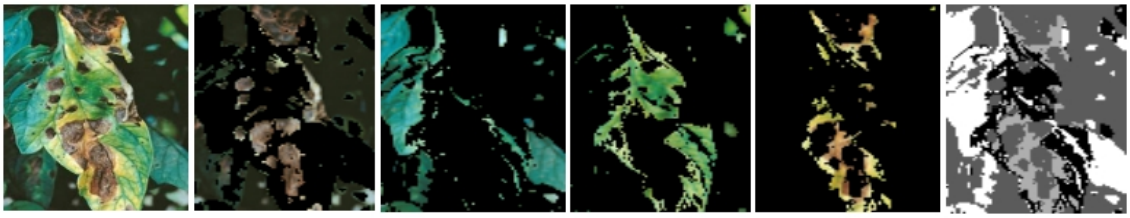


a) b)

Figura 12 – Método de Otsu aplicado a folhas de: a) mirtilo e b) algodão (Chaudhary *et al.*, 2012).

Outra possibilidade é a utilização do *adaptative thresholding* para segmentação da imagem. Enquanto no método Otsu a segmentação é feita assumindo que a imagem tem 2 classes de píxeis: de fundo e de primeiro plano, com o *adaptative thresholding* a imagem é dividida em diversas regiões (por ex. uma grelha) e cada região é tratada separadamente (por ex. aplicando o método de Otsu a cada região) de modo a ter em conta as diferenças de iluminação, sombra, etc**.**

No método baseado em agrupamento (Figura 13) usa-se frequentemente o algoritmo *k-means*. Com este algoritmo agrupam-se os píxeis em *k* grupos com cada píxel a pertencer ao grupo com o valor médio/centroide mais próximo (Khirade e Patil, 2015; Patel e Joshi, 2017).



a) b) c) d) e) f)

Figura 13 – Aplicação do k-means (*k*=4) a uma imagem de uma folha infetada: a) imagem original, (b,c,d,e) píxeis dos 4 clusters, f) Imagem com uma escala de cinzentos baseada no índex dos clusters (Al-Hiary *et al.*, 2011).

Para segmentar objetos sobrepostos (por exemplo os insetos presos nas armadilhas) pode combinar-se a segmentação baseada em contorno com a segmentação por regiões para detetar individualmente estes objetos.

A segmentação por contorno tem como objetivo estimar a forma dos objetos, mas este método falha na deteção de objetos sobrepostos ou que se tocam porque apenas é detetada uma forma/contorno.

A segmentação por regiões tem como objetivo dividir a imagem em diversas regiões tendo em conta a semelhança (cor, etc.) de píxeis adjacentes. Esta região pode começar em 1 píxel e ir adicionando recursivamente píxeis adjacentes que satisfaçam o critério de semelhança. A segmentação por regiões pode ter o problema de criar regiões desnecessárias, segmentando assim, objetos demasiados pequenos que não são importantes para o problema.

Com a combinação dos 2 métodos (segmentação baseada em contorno e segmentação por regiões) começa-se por aplicar a segmentação por regiões para identificar os contornos existentes, de seguida utiliza-se o k-means para agrupar os diferentes contornos em categorias (k=3, ruído, objetos individuais e objetos sobrepostos) e por fim separam-se os objetos sobrepostos utilizando a segmentação por regiões (Bakkay *et al.*, 2018a).

3.1.4 Extração de características

Após a segmentação das características, são extraídas as mais relevantes para a identificação e classificação dos insetos. Estas características incluem a cor (vários tipos de insetos têm cores diferentes), forma e textura (como os padrões de cores estão espalhados na imagem).

Para extrair o tipo de textura, habitualmente utiliza-se um método baseado na análise de texturas que tem em conta os píxeis individualmente e a relação entre eles. Neste método, a extração das características de cada imagem é feita usando uma matriz de co-ocorrência (Figura 14). A imagem é transformada numa matriz de escala de cinzentos (GLCM) e é atribuído um valor a cada cor presente em cada píxel da imagem. De seguida é somado o número de vezes que píxeis adjacentes com as mesmas cores ocorrem em conjunto. A matriz que resulta dessas somas pode ser interpretada como o conjunto de valores das características mais relevantes (Gavhale e Gawande, 2014; Patel e Joshi. 2017).

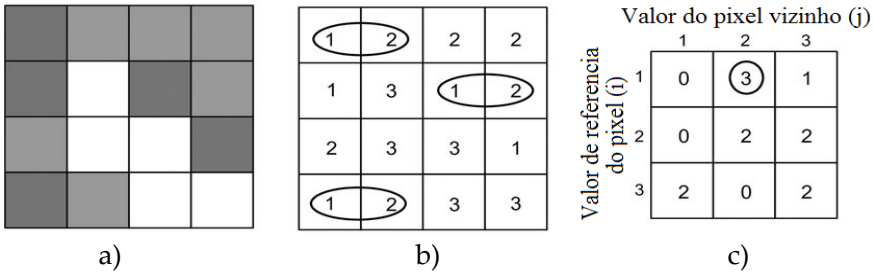


Figura 14 – Matriz de co-ocorrência (GLCM): a) imagem numa escala de cinzentos, b) valores numéricos da escala, c) gray-level co-occurrence matrix (GLCM) (Do *et al.*, 2019).

3.1.5 Classificadores (*KNN, SVM, ANN* e *CNN*)

Na última etapa utilizam-se os classificadores para treinar e testar os conjuntos de dados/imagens. Os classificadores mais usados são os *k-nearest neighbour*, *support vector machine*, redes neuronais artificiais e redes neuronais convolucionais.

Os principais fatores para a escolha de um desses classificadores são a complexidade do problema a resolver e o poder computacional disponível. Como resultado pode-se ter de testar vários classificadores até encontrar um que tenha uma relação satisfatória entre o seu desempenho e a sua complexidade. O classificador *k-nearest neighbour (KNN*) é o mais simples e funciona bem em pequenos conjuntos de dados. É sensível a *outliers* e o poder computacional necessário aumenta significativamente com o aumento do conjunto de dados, porque é necessário calcular a distância entre cada instância e os restantes pontos. O classificador *support vector machine* (*SVM*) permite uma fácil interpretação dos resultados e apresenta resultados robustos, mas o seu processo de treino é lento. As redes neuronais artificiais (*ANN*) são fáceis de implementar e formam relações não lineares facilmente. São aplicáveis a um grande número de problemas, mas o processo de treino é lento e não é possível interpretar o processo efetuado para obter os resultados (modelo *black-box*). As redes neuronais convolucionais podem evitar o pré-processamento manual de imagens, detetam e extraem automaticamente as características mais importantes, mas o seu processo de treino é lento e é um modelo *black-box*. Para melhorar a precisão da deteção de pragas em problemas com subclasses pode utilizar-se o *hierarchical learning*. Começa-se por prever apenas as classes principais (por exemplo praga ou não praga) e de seguida prevêem-se as suas subclasses (tipo de praga) (Gavhale e Gawande, 2014; Patel e Joshi, 2017).

k-nearest neighbours (*KNN*)

O algoritmo *KNN* calcula a distância (Euclidiana, Manhattan, etc.) de um ponto *p* não classificado a todos os outros pontos classificados do subconjunto de treino. Posteriormente escolhe *k* pontos vizinhos (com menor distância a *p*), verifica a classe desses pontos e contabiliza o número de pontos de cada classe. A classe atribuída ao ponto *p* é aquela que estiver em maioria entre pontos vizinhos (Gavhale e Gawande, 2014). Normalmente utiliza-se um valor ímpar de *k* para evitar um empate na classificação do ponto *p* ou atribuem-se pesos à contribuição dos pontos vizinhos *x* de modo que os pontos mais próximos contribuam mais. O peso mais usado consiste em atribuir a cada ponto vizinho o inverso da sua distância até *p*.

Support vector machine (*SVM*)

O *SVM* é um classificador não linear que foi projetado para trabalhar com apenas duas classes. O conjunto de dados de treino é dividido encontrando o hiperplano que maximiza a margem entre classes. Os dados/pontos mais próximos da margem que forem escolhidos para determinar o hiperplano são chamados de vetores de suporte. A classificação de problemas não lineares é feita calculando o produto interno entre vetores (função de kernel) e assim evitar mapear os dados explicitamente para um espaço de maior dimensão. A classificação multi-classe é feita construindo vários SVM de duas classes e utilizando a metodologia um contra todos (OAA) ou um contra um (OAO) (Gavhale e Gawande, 2014).

Redes neuronais artificiais (*ANN*)

As *ANN* podem ser utilizadas para problemas de visão por computador transformando a imagem (matriz de píxeis) num vetor. São compostas por camada de entrada, camadas ocultas e camada de saída (Figura 15). Cada camada é composta por um número predefinido de neurónios que estão anexados a funções de ativação (sigmoide, ReLu, etc.). As ligações entre neurónios contêm pesos que são atualizados durante a fase de treino que é efetuado normalmente através de *backpropagation* usando a regra da cadeia. Para encontrar a combinação de pesos em que o erro é o menor possível, normalmente utiliza-se o método da descida do gradiente. Começa-se por atribuir pesos aleatórios e os cálculos são efetuados partindo da camada de entrada até à camada de saída. Calcula-se o erro obtido na camada de saída e modificam-se os pesos da iteração anterior partindo da camada de saída até à camada de entrada. De seguida, e com os pesos obtidos na iteração anterior, são efetuados os mesmos cálculos até se atingir um número de iterações previamente definidas (Hassoun, 1995; Abraham, 2011).

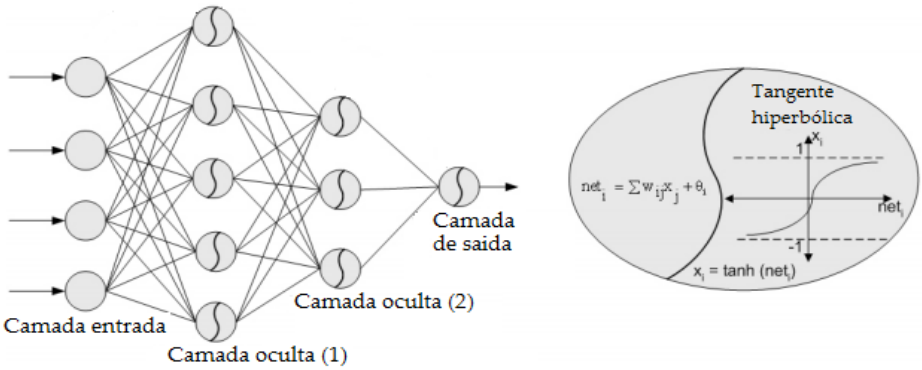


Figura 15 – Exemplo de uma rede neuronal artificial com 2 camadas ocultas e tangentes hiperbólicas como funções de ativação (Ghiassia *et al.* 2005).

Redes neuronais convolucionais(*CNN*)

Para os algoritmos anteriores é necessário identificar as características mais importantes de cada imagem, desenvolver um extrator dessas características que percorra as imagens píxel a píxel e posteriormente construir uma base de dados de características que será submetida à rede neural artificial ou algoritmo de *machine learning*. As *CNN* (Figura 17) são usadas em visão computacional para solucionar o problema da extração de características e o problema do tratamento de imagens com muitos píxeis. Com as *CNN* pode não ser necessário fazer a extração de características manualmente e ao contrário dos algoritmos anteriores não utiliza todas as entradas (píxeis), isto é, descobre as características mais importantes. Com a *CNN* usam-se as primeiras camadas para pré-processamento das imagens e posteriormente uma rede neuronal densa para classificação dessas imagens. Para este pré-processamento utilizam-se camadas de convolução, *pooling* e achatamento (Al-Saffar *et al.*, 2017; Traorea *et al.*, 2018).

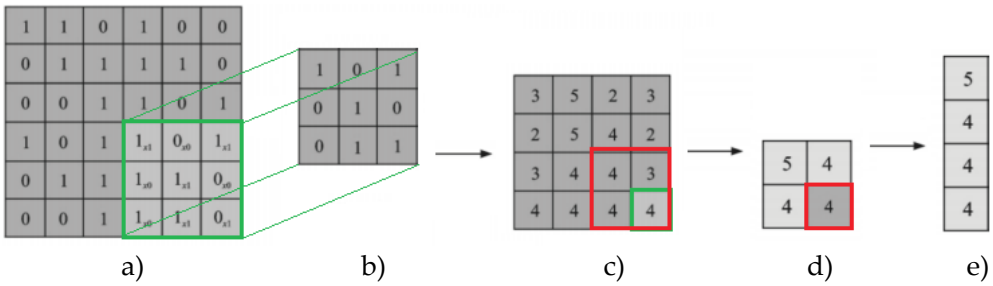


Figura 16 – Fases principais do pré-processamento de uma *CNN*: a) imagem, b) filtro c) mapa de características, d) matriz de pooling, e) camada de achatamento (Al-Saffar *et al.*, 2017).

A camada de convolução é usada para detetar as partes mais importantes das imagens. Isso é feito multiplicando cada píxel de cada imagem por um filtro (Figura 16b) e resultará numa nova matriz de menores dimensões chamada mapa de características (Figura 16c). Quanto maiores forem os valores do mapa de características mais importantes são para classificar corretamente a imagem. Nesta camada ainda se pode aplicar a função de ativação ReLu a cada elemento do mapa de características para transformar os valores negativos em zero e assim detetar melhor os padrões (transformando partes mais escuras em zonas mais claras). Esta camada é composta pelo conjunto de mapa de características obtido para cada imagem. Normalmente são utilizados vários tipos de filtros predefinidos com valores diferentes (para manter a identidade da imagem, para deteção das bordas das figuras, para obter um efeito de nevoeiro, etc.) e durante o treino do modelo para além de se descobrir os melhores pesos (feito de forma igual às *ANN*) também se descobre qual é o melhor filtro.

De seguida o mapa de características é utilizado na camada de *pooling* (Figura 16d) em conjunto com um filtro máximo, médio, etc. a percorrer todos os seus valores. Deste processo, caso se use o filtro máximo, resulta uma matriz de menores dimensões chamada matriz de *pooling* que contem as características ainda mais importantes após redução do ruído. Esta camada é composta pelo conjunto de matrizes de *pooling* resultantes dos mapas de características de cada imagem. A camada de achatamento (Figura 16e) transforma a matriz de *pooling* num vetor que servirá de entrada à rede neuronal densa (Al-Saffar *et al.*, 2017; Traorea *et al.*, 2018).

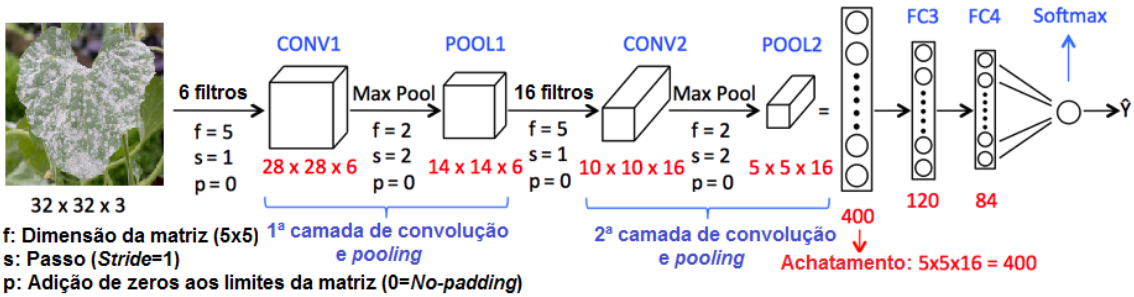


Figura 17 – Esquema de uma rede neuronal convolucional. Adaptado de (Cavaioni, 2018).

Os modelos de classificação normalmente são utilizados quando existe apenas uma classe associada a uma imagem. Por exemplo quando se pretende classificar uma imagem como sendo carro, casa, pessoa, entre outros. No caso da deteção de pragas não se pretende classificar a imagem da armadilha como contendo pragas ou não, mas sim identificar os vários insetos na mesma armadilha e contabilizar aqueles que são pragas. Nesse caso é necessário adaptar os modelos de classificação. Ou seja, é necessário obter a imagem do topo da armadilha, anotar manualmente os insetos presentes nas imagens, fazer o pré-processamento para remover o fundo da armadilha, fazer a segmentação dos insetos e escolher as suas características mais relevantes para classificar cada inseto presente na armadilha e fazer a sua contabilização. Esse processo pode ser demasiado complexo e desnecessário, caso sejam utilizados modelos de deteção de objetos que fazem automaticamente o processamento das imagens das armadilhas, considerando os insetos que se pretendem detetar como objetos.

3.2 Deteção de objetos

No controlo de pragas por visão por computador, para além da classificação de imagens também se pode utilizar a deteção de objetos. Ao contrário dos problemas de classificação onde a cada imagem está associada uma classe, com a deteção de objetos é possível classificar vários objetos (ou seja, vários insetos) na mesma imagem. Esta característica é útil para detetar e contabilizar em cada imagem das armadilhas os insetos que são pragas, os insetos predadores e os inofensivos. Os 2 modelos mais usados são a rede neuronal convolucional por regiões (*R-CNN*) e a *YOLO* (Bochkovskiy *et al.*, 2020).

3.2.1 Redes neuronais convolucionais por regiões

A rede neuronal convolucional por regiões é um modelo de deteção de objetos que foi evoluindo ao longo dos anos. Em 2014, o autor de (Girshick *et al.*, 2014) propôs o modelo *R-CNN*, um ano mais tarde o mesmo autor propôs o modelo *Fast R-CNN* (Girshick, 2015) que demora menos tempo a treinar. Em 2016 o autor de (Ren *et al.*, 2016) propôs o modelo *Faster R-CNN* que ainda é mais rápido. Em 2017 o autor de (He *et al.*, 2017) propôs o modelo *Mask R-CNN* que para alem da deteção de objetos permite também a sua segmentação.

Rede neuronal por regiões (*R-CNN*)

O modelo rede neuronal convolucional por regiões (Figura 18) é composto por 4 fases. Selecionar as regiões de cada imagem, obter o vetor de características dessas regiões, classificar cada região e por fim voltar a unir essas regiões classificadas na imagem original.

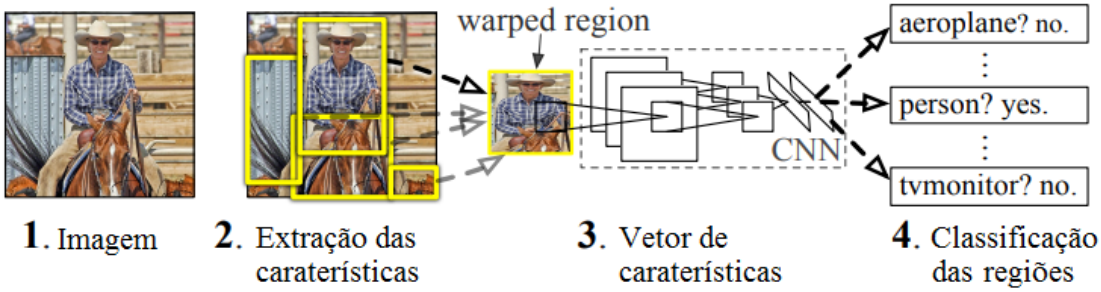


Figura 18 – Esquema de uma Rede neuronal convolucional por regiões (*R-CNN*) (Girshick *et al.,* 2014).

Começa por se utilizar a escolha seletiva para formar regiões da imagem que podem conter objetos. Este método foi criado para substituir o método da pirâmide, onde a partir de cada imagem eram criadas novas imagens de diferentes escalas e cada uma dessas novas imagens era totalmente percorrida por uma janela/matriz para tentar encontrar objetos de escalas/tamanhos diferentes. O método da pirâmide (Adelson *et al.,* 1983) é demasiado lento, depende da escolha da escala e é muito caro computacionalmente. Tendo isso em conta foi criada a escolha seletiva onde cada região é identificada através da semelhança de cor, textura, tamanho e forma. Um dos algoritmos existentes de escolha seletiva é o "superpíxel" (Uijlings *et al.,* 2013) onde a semelhança de cor e textura é verificada através das regiões de intersecção de um histograma formado pela cor e textura dos 3 canais de cada píxel (quanto maior a intersecção mais semelhante é a cor e textura). A semelhança entre tamanhos é obtida agrupando regiões através de um algoritmo de agrupamento aglomerativo hierárquico começando das regiões menores para as maiores. As formas das regiões são consideradas semelhantes/compatíveis se encaixarem umas nas outras sem deixar espaços. O resultado final destas métricas é ponderado para tomar a decisão de formar região ou não.

De seguida usa-se cada região da imagem para criar um vetor de características que irá representar essa região. Para isso é utilizada uma rede neuronal convolucional (*CNN*). No artigo original (Girshick *et al.,* 2014) escolheram a rede AlexNet que é composta por 5 camadas convolucionais, algumas delas seguidas por camadas de max-pooling e no final 3 camadas totalmente ligadas (*fully connected*). Esta rede neuronal foi treinada para classificar imagens recebendo as regiões no formato 227x227x3. Após o treino removeram a camada de classificação para obter o vetor de características no formato 4096x1.

De seguida é feita a classificação das regiões utilizando o vetor de características obtido. No artigo original (Girshick *et al.,* 2014) usaram o *support vector machine* (*SVM*) como classificador. Existe um *SVM* para cada classe de objetos. Cada vetor de características é avaliado por todos os *SVM* para obter a confiança do objeto pertencer a cada classe. No fim é atribuída a classe de maior confiança. O treino destes SVM é feito com o vetor de características obtido da *CNN*, o que significa que só pode ser feito após terminar o treino da *CNN*.

Tendo as propostas de classificação de cada região é necessário unir essas classificações na imagem original. Isso é feito utilizando a técnica *non-maximum suppression* (NMS) que tem como princípio rejeitar as regiões que intersetam (*intersection-over-union* – IoU) outras regiões com maior confiança e manter apenas as regiões de maior confiança.

Rede neuronal por regiões rápida (Fast R-CNN)

O mesmo autor de (Girshick *et al.*, 2014) para tentar combater alguns problemas do modelo *R-CNN* como o excessivo consumo de memoria e tempo de treino, propôs o modelo *Fast R-CNN* (Figura 19).

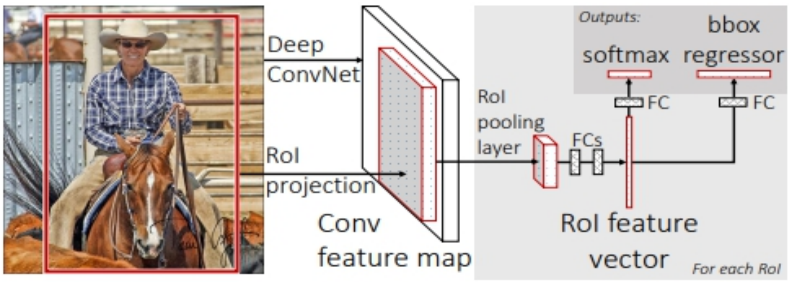


Figura 19 – Esquema de uma Rede neuronal convolucional por regiões rápida (*Fast R-CNN*) (Girshick, 2015).

Neste modelo apenas existe um sistema que é necessário treinar. A rede neuronal convolucional em vez de ter como dados de entrada regiões tem a imagem original. Destas imagens retira-se o vetor de características após a imagem passar pelas camadas convolucionais e de *max-pooling*. De seguida através da escolha seletiva retiram-se as regiões de interesse de cada mapa de características. Posteriormente todas as regiões de interesse identificadas são envolvidas em retângulos e utiliza-se a camada *RoI pooling* para transformar esses retângulos em vetores de tamanho fixo. Isso é feito dividindo os retângulos em regiões e escolhendo os píxeis de maior valor dentro de cada região. Esses vetores vão servir de entrada à parte densa da rede neuronal (*fully-connected*) que tem como saídas a classe de cada região e os valores limites dos retângulos. Este modelo é mais rápido do que a *R-CNN* porque para cada imagem não é necessário alimentar a rede neuronal convolucional com todas as regiões obtidas, mas apenas com essa imagem (Girshick, 2015).

Rede neuronal por regiões mais rápida (*Faster R-CNN*)

Apesar do modelo *Fast R-CNN* ser mais rápido do que o *R-CNN* a utilização da escolha seletiva de regiões, que é um algoritmo lento e que consome bastante memoria, impossibilita-o de obter melhores resultados. Por esse motivo foi criado o modelo *Faster R-CNN* (Figura 20) que utiliza uma rede neuronal para aprender as regiões de interesse e assim evitar a utilização da escolha seletiva.

Neste modelo, tal como no *Fast R-CNN*, as imagens servem de dados de entrada à rede neuronal convolucional. O vetor de características é obtido após a passagem da imagem pelas camadas convolucionais e de *max-pooling*. De seguida, em vez de se usar a escolha seletiva, utiliza-se outra rede neuronal em separado (treinada de forma independente) para prever as regiões de interesse de cada vetor de características. Posteriormente estas regiões de interesse são retiradas de cada vetor de características e colocadas todas do mesmo tamanho utilizando uma camada de *RoI pooling*. Os vetores resultantes desta operação são usados como entrada da parte densa da rede, de modo a classificar cada região que compõe a imagem e prever os valores limites dos retângulos que enquadram estas regiões de interesse (Ren *et al.*, 2016).

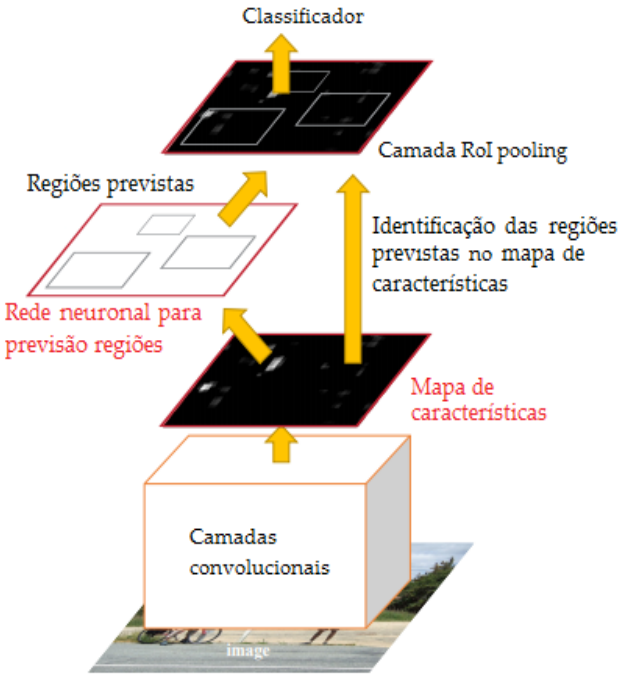


Figura 20 – Esquema de uma Rede neuronal convolucional por regiões mais rápida (*Faster R-CNN*) (Ren *et al.*, 2016).

Rede neuronal por regiões com máscara (*Mask R-CNN*)

A rede neuronal *Mask R-CNN* (Figura 21) para além de detetar objetos faz em simultâneo a sua segmentação gerando máscaras que indicam os píxeis pertencentes ao objeto detetado. Combina assim a deteção de objetos que tem como objetivo classificar objetos individuais e localiza-os, utilizando os retângulos delimitadores e a segmentação que tem como objetivo classificar cada píxel num conjunto predefinido de categorias, sem ter em conta a que objeto pertence. Este modelo estende a *Faster R-CNN* adicionando a cada região de interesse um ramo para prever a máscara de segmentação de cada objeto. Esse ramo é composto por uma rede neuronal densa (*fully connected*) aplicada a cada região de interesse, que retorna a classificação de cada píxel prevendo assim a máscara de segmentação de píxel a píxel (He *et al.*, 2017).

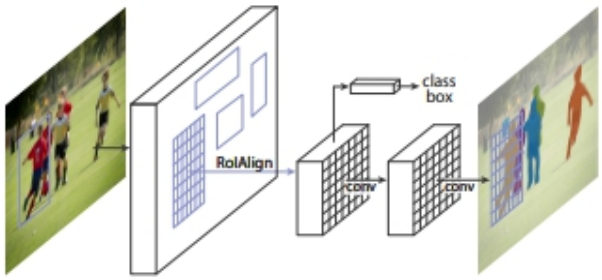
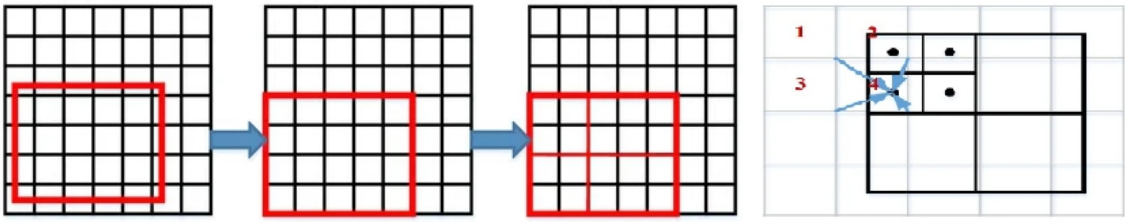


Figura 21 – Modelo da *Mask R-CNN* para deteção e segmentação de objetos (He *et al.*, 2017).

O modelo *Faster R-CNN* não foi construído para preservar os valores originais dos píxeis, já que utiliza a camada *RoI pooling* (Figura 22a) onde se escolhem os píxeis de maior valor dentro de cada região, enquanto o modelo *Mask R-CNN* necessita dos valores exatos das coordenadas dos píxeis para construir a máscara. Por esse motivo pode-se substituir a camada *RoI pooling* pela camada *RoIAlign* (Figura 22b) que preserva a localização espacial de cada píxel, utilizando a interpolação para calcular o valor de cada coordenada, evitando assim os arredondamentos feitos pela camada *RoIAlign* no mapeamento entre a imagem e o

mapa de características e entre o mapa de características e a camada de *RoI pooling*. Com a utilização do *RoI pooling* faz-se o arredondamento para baixo da largura, altura e coordenadas da região de interesse, causando a perda de informação e a alteração das suas coordenadas, mas garantindo que a região se sobrepõe aos limites das células do mapa de características (Bai *et al.*, 2020).



a) b)

Figura 22 – a) *RoI Pooling* e b) RoI Align(Bai *et al.*, 2020).

Com o *RoI Align* divide-se a região de interesse, por exemplo numa matriz 2x2 atribuem-se 4 pontos amostra a cada região e tendo em conta a altura e largura da região de interesse calculam-se as coordenadas de cada ponto amostra e as coordenadas do centro de cada célula do mapa de características. De seguida calculam-se os valores de cada ponto amostra através de interpolação dos 4 valores mais próximos do mapa de características e escolhe-se o valor maior para cada célula da matriz *RoI pooling* (Bai *et al.*, 2020). Na secção 4.1.1. desta dissertação apresenta-se a arquitetura detalhada do modelo.

3.2.2 *You Only Look Once (YOLO)*

O modelo *YOLO* foi criado para fazer a deteção de objetos em tempo real. Este modelo tem a vantagem de ser mais rápido do que o modelo *R-CNN*, mas em alguns casos tem menor precisão (*accuracy*) especialmente em objetos mais pequenos. A previsão das classes dos objetos e a sua localização é feita com apenas uma rede neuronal, o que permite treinar o sistema de forma mais fácil. Os resultados apresentam menos falsos positivos na deteção de objetos no fundo das imagens (Redmon *et al.*, 2015).

O modelo *YOLO* (Redmon *et al.*, 2015) foi proposto em 2015. O mesmo autor propôs em 2016 o modelo *YOLOv2* (Redmon e Farhadi, 2016) que consegue obter uma maior rapidez e precisão. Em 2018 fez algumas alterações a esse modelo e propôs o *YOLOv3* (Redmon e Farhadi, 2018). Em abril de 2020 é publicado um artigo com o modelo *YOLOv4* (Bochkovskiy *et al.*, 2020) e em junho do mesmo ano é proposto o modelo *YOLOv5* (ainda sem artigo publicado) (Jocher, 2020) que mantém a mesma arquitetura da *YOLOv4*, mas utiliza um *framework* diferente para efetuar o treino do modelo (converteram *framework* da rede neuronal Darknet originalmente escrito em C na versão *YOLOv3* para PyThorch). Em novembro de 2020 é publicado o artigo da *Scaled-YOLOv4* (Wang, 2020) onde se afirma obter os melhores resultados publicados no conjunto de dados COCO.

You Only Look Once – version 1 (YOLO)

O modelo *YOLO* (Figura 23) só olha uma vez para a imagem porque apenas é requerida uma propagação da imagem desde a camada inicial até a final para fazer a previsão. O modelo começa por redimensionar as imagens utilizadas durante o treino, para terem todas o mesmo tamanho e servirem de dados de entrada de uma rede neuronal convolucional que no artigo original (Redmon *et al.*, 2015) é composta por 24 camadas convolucionais e de max-pooling, seguidas de 2 camadas totalmente ligadas.

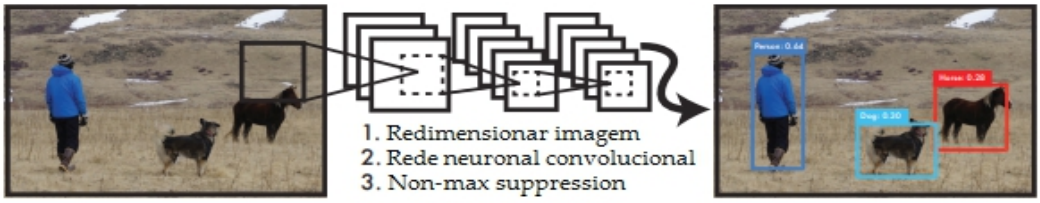


Figura 23 – Esquema do modelo You Only Look Once (*YOLO*) (Redmon *et al.,* 2015).

Numa segunda fase as imagens são divididas em regiões com o objetivo de a rede neuronal obter:

1) A probabilidade de o objeto dentro de cada região pertencer a uma determinada classe. 2) Os retângulos que delimitam os objetos que pertencem a essas regiões e a confiança nesses retângulos.

No artigo original (Redmon *et al.,* 2015) a imagem é dividida numa grelha 7x7 (Figura 24) e cada região/píxel dessa grelha contem 2 retângulos delimitadores. O vetor resultante dessa rede neuronal tem a dimensão (7, 7, B*5*C). Com 7x7 a ser o número de regiões, B o número de retângulos delimitadores que cada região prevê. 5 representa as dimensões desses retângulos (x, y, height, width) e a confiança com que cada retângulo é previsto. C é o número de classes.

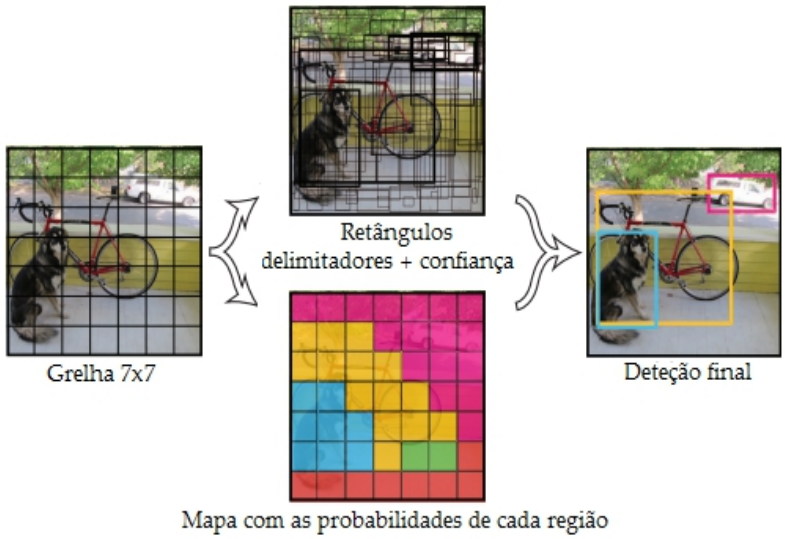


Figura 24 – Etapas do modelo *You Only Look Once* (*YOLO*) (Redmon *et al.,* 2015).

Por fim, tendo o vetor com as probabilidades associadas a cada região e a confiança dos retângulos delimitadores de cada objeto, é aplicada a técnica *non-max suppression* para remover previsões duplicadas para o mesmo objeto. O treino deste modelo é composto por 2 fases. Em primeiro a rede é treinada para classificar os objetos do dataset ImageNet e guardam-se os pesos. De seguida, utilizando esses pesos pré-treinados (*transfer learning*) converte-se o modelo para deteção de objetos. Isso é feito adicionando 4 camadas convolucionais e 2 totalmente ligadas (*fully connected*) que são inicializadas com pesos aleatórios. A camada final prevê a classe de cada objeto e as coordenadas dos retângulos delimitadores (Redmon *et al.,* 2015).

You Only Look Once – version 2 (*YOLOv2*)

Neste novo modelo foram adicionadas camadas de normalização para transformar as saídas das camadas convolucionais em valores entre 0 e 1. No modelo inicial *YOLO* o treino do classificador é feito com imagens 227x227 e de seguida utiliza imagens 447x447 para

fazer o treino da deteção de objetos. O modelo *YOLOv2* é quase sempre treinado com imagens 447x447 o que permite diminuir o número de épocas utilizadas durante esta fase. O modelo *YOLO* começa o treino por atribuir retângulos delimitadores de dimensão aleatória aos diferentes objetos. O modelo *YOLOv2* utiliza diversos retângulos que se adaptam bem a algumas formas mais conhecidas (carros, pessoas, etc.). A escolha dessas formas é feita no início do treino recorrendo ao algoritmo k-means (k=5 formas) para identificar aquelas que melhor agrupam as diferentes classes. O modelo *YOLO* usa sempre imagens do mesmo tamanho para treinar a deteção de objetos, enquanto o modelo *YOLOv2* de x em x épocas reduz essa dimensão para treinar a deteção de objetos menores (Redmon e Farhadi, 2016).

You Only Look Once – version 3 (*YOLOv3*)

A alteração mais significativa entre a versão *YOLOv3* e as anteriores encontra-se no método utilizado na classificação de objetos. O modelo *YOLO* usa a função softmax para obter a probabilidade associada a cada classe, somando 1 no total e atribuindo a classe àquela com maior probabilidade. O modelo *YOLOv3* utiliza funções logísticas independentes para obter a probabilidade de cada classe. Se essa probabilidade for maior do que um valor limite o objeto é classificado como pertencendo a determinada classe. Esta alteração permite atribuir mais de uma classe dentro de cada retângulo delimitador.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Esta versão usa como extrator de características a rede neuronal convolucional Darknet-53 (Figura 25) (na versão v2 utiliza a Darknet-18), composta por 5 blocos (que se repetem 1, 2, 8, 8 e 4 vezes respetivamente) de 2 camadas convolucionais seguidas de uma camada de resíduos. Tal como na rede ResNet as camadas de resíduos servem para evitar o treino de algumas das camadas/blocos da rede de modo a evitar o problema do desaparecimento do gradiente (*vanish gradient*) onde o gradiente vai desaparecendo à medida que se adicionam mais camadas à rede ou a explosão do gradiente (*exploding gradient*), não se conseguindo encontrar o mínimo global da função (Redmon e Farhadi, 2018).

Figura 25 – Darknet-53 (Redmon e Farhadi, 2018).

Para treinar o reconhecimento de objetos a escalas diferentes a *YOLOv3*, em vez de usar imagens de diferentes tamanhos como na versão 2, utiliza uma *Feature Pyramid Network* (*FPN*) (Figura 26) composta pela rede Darknet-53 onde são retornados mapas de características de diferentes níveis da sua arquitetura, em vez de apenas da camada final e onde os mapas de características finais de cada camada são obtidos através da junção do mapa de características da última camada com os mapas de características da camada adjacente (Lin *et al.*, 2017).

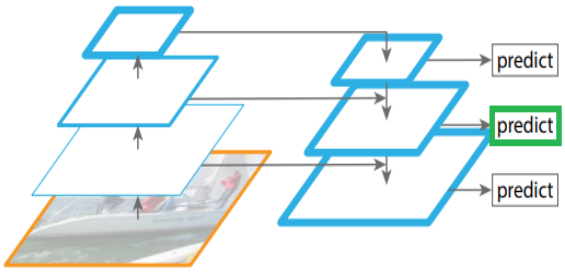


Figura 26 – Esquema da *Feature Pyramid Networks* (Lin *et al.*, 2017).

You Only Look Once – version 4 and version 5 (*YOLOv4*, *Scaled-YOLOv4* e *YOLOv5*)

A versão 4 e 5 da *YOLO* são recentes (abril e junho de 2020) e ainda existe pouca informação disponível. No entanto sabe-se que diferem das versões anteriores por usarem como extrator de características uma rede neuronal *Cross Stage Partial Network* (*CSP*), uma camada de *patial pyramid pooling* (*SPP*) para permitir a utilização de imagens de diferentes tamanhos e uma *Path Aggregation Network* (*PANet*) para combinar toda a informação dos mapas de características. A *Scaled-YOLOv4* ainda é mais recente (novembro 2020) e permite implementar várias versões da arquitetura da *YOLOv4*, alterando o número de camadas e de neurónios. A arquitetura destes modelos é apresentada na secção 4.1.2.

3.3 Conclusão

O âmbito desta dissertação foca-se especialmente na utilização de modelos de deteção de objetos para deteção da praga mosca-branca em imagens de armadilhas. No entanto, na secção 3.1 descreve-se como os modelos de classificação de objetos poderiam ser usados para o mesmo propósito. Como os modelos de classificação foram criados para classificar imagens e não vários objetos na mesma imagem, seria necessário obter as imagens das armadilhas, anotar as pragas nas imagens, fazer o pré-processamento da imagem para remover os píxeis correspondentes ao fundo da armadilha, de seguida fazer a segmentação dos píxeis correspondentes aos insetos e por último escolher as características mais relevantes para classificar cada inseto individualmente e fazer a sua contabilização.

A utilização de modelos de deteção de objetos torna a resolução do problema de deteção de pragas menos complexo porque estes modelos, ao contrário dos modelos de classificação, foram criados para detetar diferentes objetos na mesma imagem e existem exemplos da sua utilização com sucesso em áreas como sistemas de condução autónomos ou sistemas de segurança onde é necessário identificar por exemplo pessoas ou sinais de trânsito na mesma imagem ou *frame*. Para treinar estes modelos para a deteção de pragas é necessário obter as imagens das armadilhas e anotar as pragas nas imagens. Após se ter treinado o modelo, este pode ser usado para identificar os insetos nas imagens das armadilhas.

Nesta dissertação não foram definidos requisitos acerca dos modelos de deteção de objetos a utilizar. Assim, a opção entre um dos dois modelos de deteção de objetos (*R-CNN* e *YOLO*) a usar tem de ter em conta a complexidade do problema a resolver e o poder computacional disponível. Nesse sentido sabe-se que a *YOLO* tem um custo computacional mais baixo e a *R-CNN*, para alguns conjuntos de dados, obtém uma precisão superior à *YOLO*, especialmente em objetos pequenos como os insetos que se pretendem detetar (Redmon *et al.*, 2015).

No conjunto de dados COCO, utilizado como *bentchmark* para problemas de deteção de objetos, a *Mask R-CNN* obtém uma precisão média de 31,7%, a *YOLOv4* de 43,5%, a *Scaled-*

Yolov4 de 56,0% e a *YOLOv5* ainda não tem resultados publicados. No entanto não é possível saber qual dos dois modelos (*R-CNN* ou *YOLO*) é mais indicado para resolver o problema da deteção de pragas sem os implementar e testar, porque o desempenho de cada um dos modelos depende do conjunto de dados utilizado.

Os modelos escolhidos para implementar foram a *R-CNN* e os modelos *Scaled-YOLOv4* e *YOLOv5*. Com estes modelos conseguem-se testar duas arquiteturas diferentes (*R-CNN* vs *YOLO*) e modelos que ocupam pouco espaço de memória, mas que podem ser menos precisos e, em princípio, mais indicados para dispositivos móveis.

Esta página foi deixada intencionalmente em branco

4 Abordagem Proposta

A abordagem proposta consiste na utilização da visão por computador para detetar e contabilizar a praga mosca-branca em armadilhas amarelas pegajosas. Para desenvolver esta abordagem recorreu-se a uma metodologia que é aplicada no capítulo 5 para implementar um modelo de deteção da praga mosca-branca.

4.1 Metodologia

A abordagem proposta pretende responder às seguintes **questões**:

- Quais são as limitações dos métodos tradicionais?
- Como podem ser utilizados os modelos inteligentes para ultrapassar as limitações dos métodos tradicionais?
- Quais são métodos inteligentes existentes para deteção de objetos em imagens?
- Quais são as limitações dos métodos inteligentes?
- Qual o método inteligente mais adequado ao objetivo proposto?
    - Quais os modelos de deteção de objetos mais relevantes?
- Qual o desempenho dos métodos inteligentes comparado com o dos técnicos?

A **metodologia** a usar, para responder às questões e desenvolver a abordagem proposta (Figura 27), consiste em fazer testes com diferentes modelos de deteção de objetos e diferentes transformações do mesmo conjunto de dados. Para desenvolver esta metodologia utilizaram-se dados públicos e *software* de livre utilização.

As **fases** da metodologia a desenvolver consistem em:

- **Obtenção dos dados:** Propõe-se efetuar pesquisas de conjuntos de dados públicos contendo imagens de armadilhas amarelas pegajosas com a praga mosca-branca.
- **Processamento:** Propõe-se aumentar o conjunto de dados escolhido através de transformações geométricas e usando redes adversárias generativas.
- **Definição dos modelos:** Propõe-se estudar o estado da arte dos modelos de deteção de objetos e selecionar para implementação os mais recentes e que permitam comparar diferentes arquiteturas.

  Avaliação: Propõe-se utilizar métricas que permitam comparar a precisão, memória ocupada e o tempo médio de cada deteção de cada modelo de deteção de objetos.
- **Validação:** Propõe-se recorrer a um técnico para que este confirme a precisão das deteções feitas pelos modelos de deteção de objetos e valide as anotações acrescentadas ao conjunto de dados escolhido.

- **Protótipo:** Para integrar a deteção das moscas-brancas, a sua contabilização e a evolução do número de moscas-brancas ao longo do tempo propõe-se desenvolver um protótipo e fazer a sua prova de conceito em ambiente real.

Os maiores **desafios** da metodologia proposta são a falta de imagens de armadilhas amarelas com a praga mosca-branca, a falta de anotações nessas imagens e o tempo necessário para treinar os modelos de deteção de objetos.

Devido à falta de conjuntos de dados públicos e como modelos de deteção de objetos normalmente requerem um grande conjunto de dados para obterem bons resultados, propõe-se, na fase **Processamento**, aumentar o conjunto de dados escolhido.

Devido à falta de anotações, e como esta influencia a precisão dos modelos de deteção de objetos, propõe-se, na fase **Validação**, completar a anotação do conjunto de dados escolhido e validar essas anotações recorrendo a um técnico. Estas anotações só foram feitas nesta fase devido à disponibilidade dos técnicos e também por não ser possível anotar o conjunto de dados sem a participação destes técnicos, pela semelhança das moscas-brancas com outros insetos.

Devido às condições controladas (luz, fundo, etc.) nas quais as imagens do conjunto de dados possam ter sido obtidas propõe-se, na fase **Protótipo**, fazer a prova de conceito do protótipo em ambiente real (numa plantação de tomate). Devido às condições climatéricas desfavoráveis ao aparecimento da praga mosca-branca e à disponibilidade dos técnicos não foi possível fazer a prova de conceito.

O processo de treino dos modelos de deteção de objetos pode ser lento e por isso propõe-se, na fase **Definição dos modelos**, a utilização de t*ransfer learning* (*TL*) de modelos pré-treinados em outros conjuntos de dados (como o Microsoft COCO dataset). Esta ferramenta utiliza, para acelerar o processo de treino, os pesos obtidos de um modelo já treinado (para prever um conjunto de classes) para prever um novo/diferente conjunto de classes.

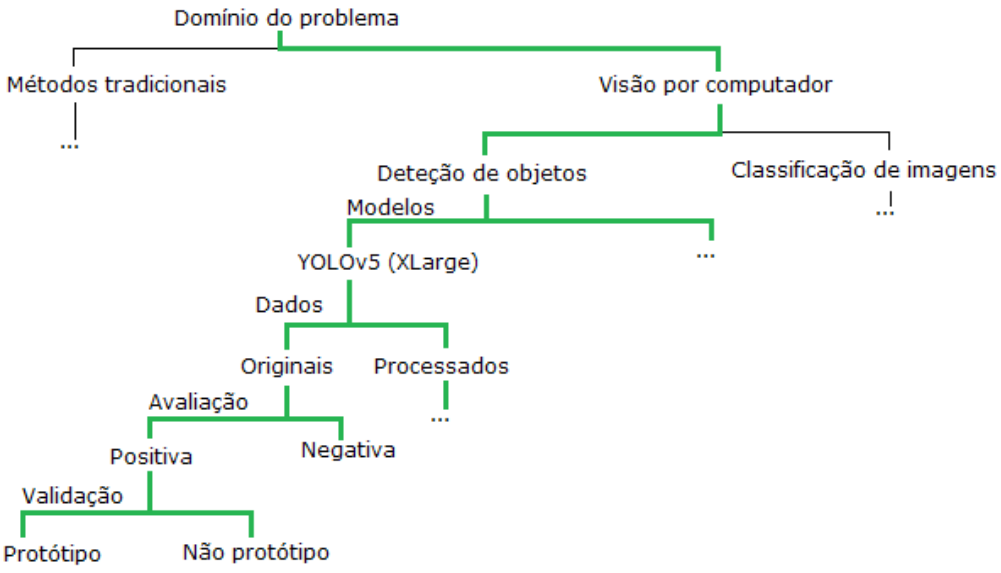


Figura 27 – Estrutura da metodologia proposta.

4.2 Modelos propostos

Para fazer a deteção de pragas em armadilhas propõe-se a utilização dos modelos de deteção de objetos: *R-CNN*, *Scaled-YOLOv4* e *YOLOv5* (*Xlarge* e *Small*).

Estes modelos permitem testar duas arquiteturas diferentes (*R-CNN* vs *YOLO*) e usar, na deteção da praga mosca-branca, o último modelo de deteção de objetos com artigo publicado (*Scaled-YOLOv4*) e outro modelo ainda sem artigo publicado (*YOLOv5*).

O desenvolvimento dos quatro modelos de deteção de objetos é relevante porque a *YOLOv5 (small)* é mais rápida (tem um custo computacional mais baixo) que a *YOLOv5 (Xlarge)*. A *Scaled-YOLOv4* é o modelo de deteção de objetos com resultados publicados, com melhor precisão e a *R-CNN*, para alguns conjuntos de dados obtém uma precisão superior aos modelos *YOLO* (Redmon et al., 2015).

4.2.1 Arquitetura da rede neuronal convolucional por regiões

Este modelo é composto por 3 etapas (Figura 28), a primeira etapa é composta pela **base do modelo (*backbone*)** e a ***Feature Pyramid Network (FPN)*** que são utilizadas para obter os mapas de características das imagens, a segunda etapa é composta pela ***Region Proposal Network (RPN)*** que é usada para classificar cada retângulo delimitador previsto como contendo ou não objetos e prever as coordenadas desses retângulos e a fase final é composta pelo ***RoI Classifier and Bounding Box Regressor*** que é utilizada para atribuir uma classe a cada retângulo delimitador e continuar a refinar as dimensões destes retângulos (Abdulla, 2017; He *et al.*, 2017; Brownlee, 2020;).

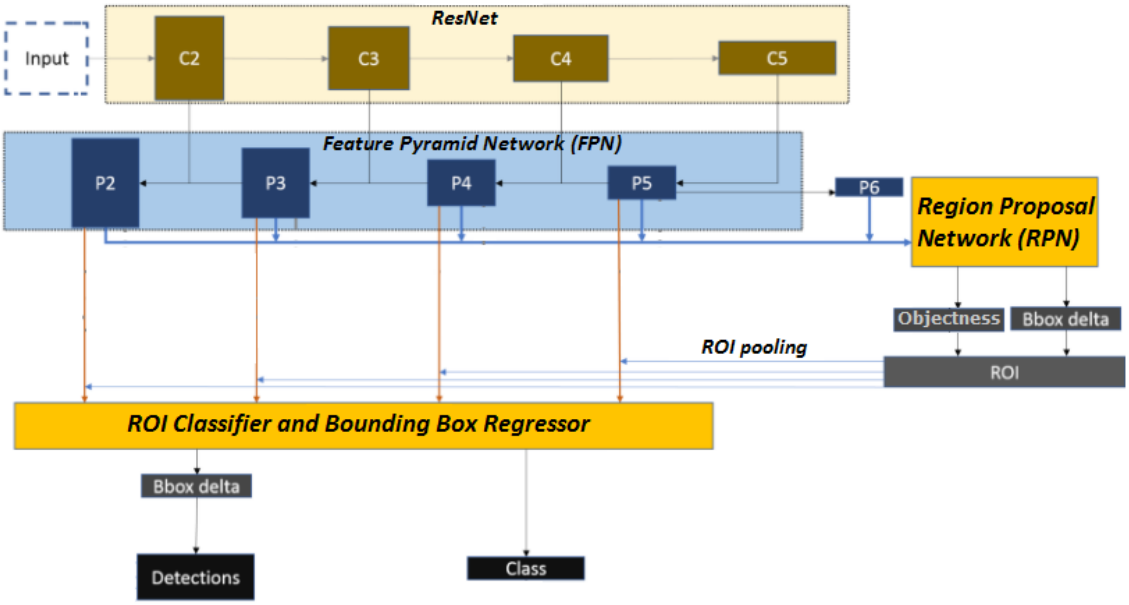


Figura 28 – Arquitetura da *Mask R-CNN*. Adaptado de (Deng et al., 2020).

Base do modelo (*backbone*)

A implementação é feita utilizando uma rede neuronal convolucional ResNet-101 (Figura 29) para extração de características. Esta rede neuronal é composta por 99 blocos de 3 camadas de convolução de diferentes tamanhos e número de filtros aplicados. As primeiras camadas começam por detetar características de baixo nível como bordas e cantos e, com o avançar do processo de convolução, as camadas seguintes começam a detetar sucessivamente características de maior nível como pessoas ou carros. As imagens de

entrada da ResNet-101 estão no formato 1024x1024 e com a passagem pelas diferentes camadas desta rede vão-se obter, para cada imagem 2048 mapas de características no formato 32x32 (Abdulla, 2017).

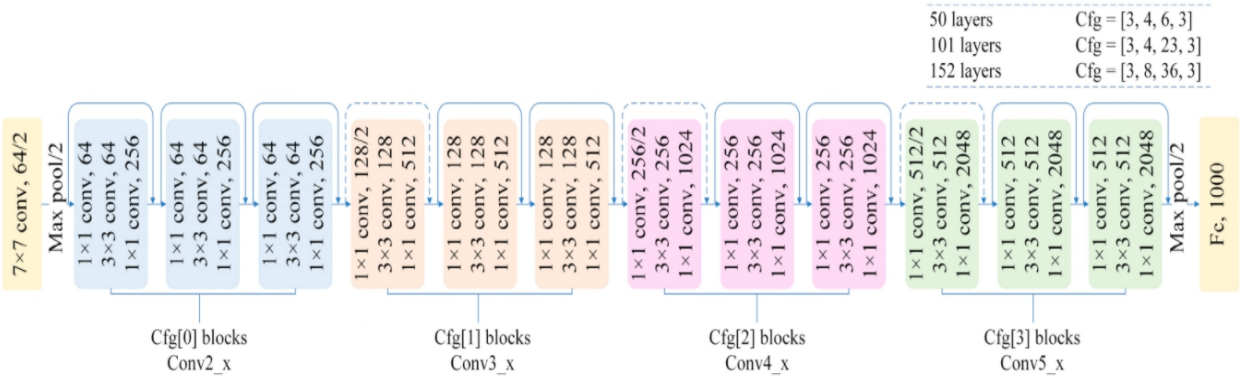


Figura 29 – Arquitetura das redes neuronais convolucionais ResNet-50, ResNet-101 e ResNet-152 (Deng *et al.*, 2020).

Feature Pyramid Network (FPN)

De seguida utiliza-se a *Feature Pyramid Network* (*FPN*) (Figura 30) para melhorar os mapas de características obtidos apenas com a ResNet-101 e conseguir detetar melhor objetos de diferentes escalas e mais pequenos.

As *FPN* são compostas pela rede neuronal ResNet-101 que, em vez de retornar apenas os mapas de características da última camada como descrito acima, retorna mapas de características em diferentes níveis da sua arquitetura (4 níveis da pirâmide correspondentes aos 4 blocos da ResNet). Nas primeiras camadas retornam os mapas de características para detetar características de baixo nível da imagem (de menor valor semântico/significado, mas de maior resolução), enquanto nas últimas camadas retornam mapas de características para detetar características de alto nível da mesma imagem (de maior significado, mas de menor resolução/mais filtradas). De seguida os mapas de características de maior nível são unidos aos mapas de características de menor nível da camada adjacente, adicionando os seus elementos e usando uma camada de convolução 1x1 com 256 filtros para diminuir o número de canais final, tornando o processamento mais rápido. Por fim, estes mapas de características passam por uma camada de convolução 3x3 com 256 filtros para reduzir/filtrar o efeito provocado pelo aumento de dimensão devido à soma dos mapas. Dependendo do tamanho do objeto escolhe-se um dos mapas de características dos diferentes níveis como entrada da fase seguinte do modelo da *Mask R-CNN* (Abdulla, 2017; Lin *et al.*, 2017).

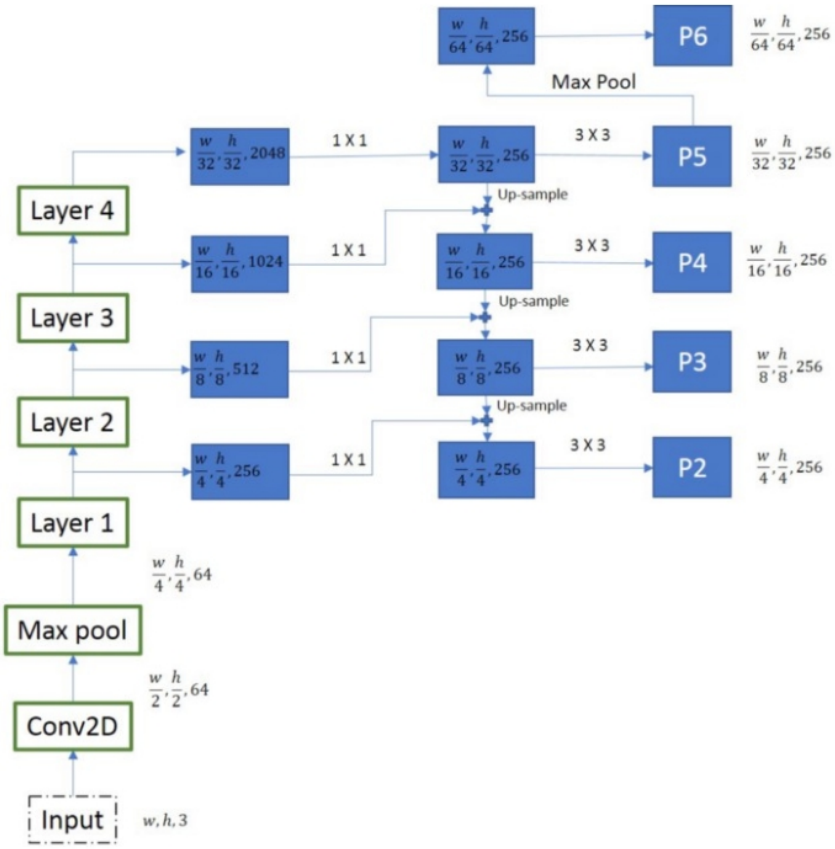


Figura 30 – Arquitetura da *Feature Pyramid Networks* (Pande, 2019).

Region Proposal Network (RPN)

A fase seguinte utiliza uma rede neuronal para propor regiões de interesse (*Region Proposal Network (RPN)*) de forma a diminuir o esforço computacional, identificando as regiões das imagens que têm objetos contidos em retângulos delimitadores de diferentes formas (Figura 31). Esta rede neuronal aprende as regiões de interesse das imagens a partir dos mapas de características obtidos na etapa anterior. Começa por considerar cada ponto do mapa de características um ponto âncora (*anchor point*) e gera por defeito 10 retângulos delimitadores âncora (*anchor boxes*) para cada um desses pontos. Os retângulos são gerados com base em 4 escalas (32x32, 64x64, 128x128 e 256x256) e 3 proporções/*ratio (0.5, 1 e 2)* e têm todos o mesmo centro/coordenadas para cada ponto âncora (caso se utilize mapas de características de diferentes níveis pode-se ignorar as escalas e usar apenas 3 retângulos delimitadores âncoras criados com base em 3 proporções). As coordenadas são referentes à imagem original e são obtidas considerando o número de vezes que a imagem foi reduzida para obter o mapa de características.

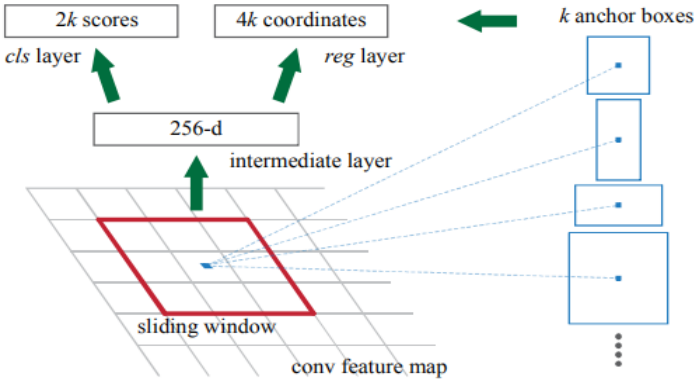


Figura 31 – Retângulos delimitadores obtidos com diferentes escalas e proporções (Ren *et al.*, 2016).

De seguida é aplicado um filtro 3x3 com 256 canais a cada mapa de características e o seu resultado serve de entrada em 2 ramos do modelo das regiões de interesse (Figura 32). Um dos ramos tem como dimensões de entrada a relação entre altura/largura da imagem original e o mapa de características, com o número de canais a corresponder à quantidade de retângulos delimitadores utilizados. A camada final desse ramo retorna a probabilidade de os retângulos delimitadores âncora conterem objetos ou não (*background* (0) *or foreground* (1)). O outro ramo tem como dimensões de entrada a relação entre altura/ largura da imagem original e o mapa de características, com o número de canais a corresponder ao número de retângulos delimitadores âncoras multiplicados pelas 4 coordenadas usadas para definir esses retângulos. A camada final retorna a variação das coordenadas (dy, dx, log(dh), log(dw)) do retângulo delimitador âncora para o do retângulo delimitador que melhor enquadra o objeto nele contido (ajusta o retângulo delimitador âncora ao objeto).

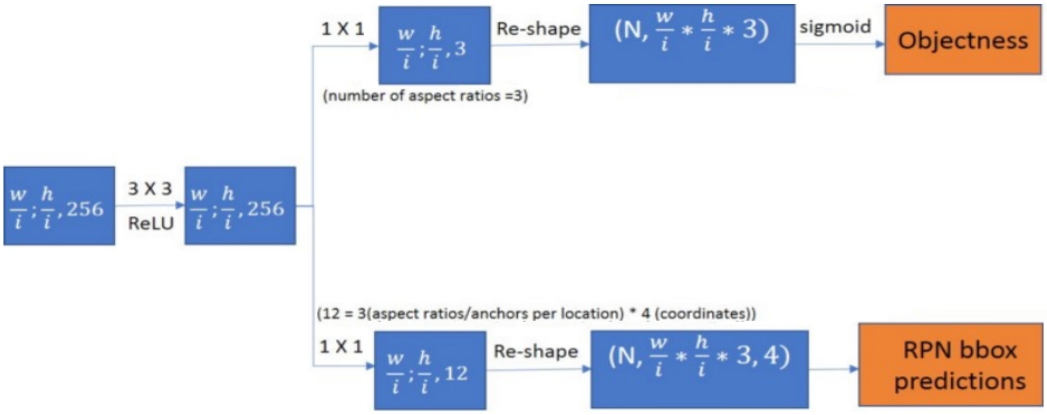


Figura 32 – Arquitetura da *Region Proposal Network* com 3 *anchor boxes* (Pande, 2019).

Para treinar a *Region Proposal Network* (presença de objetos e coordenadas) é atribuído a cada retângulo delimitador âncora um valor p associado à sua sobreposição a um retângulo delimitador real. Com a âncora a ter uma classificação positiva se $p = 1$ (IoU > 0.7), negativa se $p = -1$ (IoU < 0.3) e a não contribuir para o treino se $p = 0$ ($0.7 \geq$ IoU ≥ 0.3). O valor da *intersection over union* (IoU) (Figura 33) é calculado dividindo a área de sobreposição entre o retângulo delimitador real e âncora pela área de união entre eles.

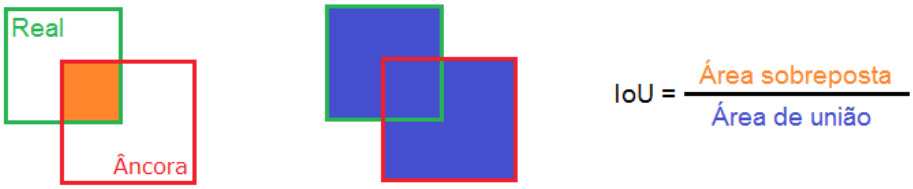


Figura 33 – Cálculo da IoU: a) Área de sobreposição, b) Área de união e c) fórmula.

Considerando o mapa de características obtido na última camada da Res-Net com a dimensão 2048 existem inicialmente 2048x10 = 20480 retângulos delimitadores âncoras para cada imagem. Para diminuir essa quantidade ignoram-se as âncoras que não contribuem para o treino e seleciona-se apenas as 6000 que têm melhor desempenho durante o treino. De seguida ajustam-se as coordenadas das âncoras de acordo com a variação de coordenadas aprendida. Por fim aplica-se o *non-maximum suppression* (NMS) para remover as âncoras sobrepostas (IoU > 0.7) com pior classificação e escolhem-se as 2000 âncoras que obtiveram melhor desempenho (Figura 34). Dessas 2000 regiões de interesse escolhem-se aleatoriamente 512 (256 positivas e negativas) para cada amostra (*mini-batch*). Para teste escolhem-se outras 1000 regiões de interesse (Ren *et al.*, 2016).

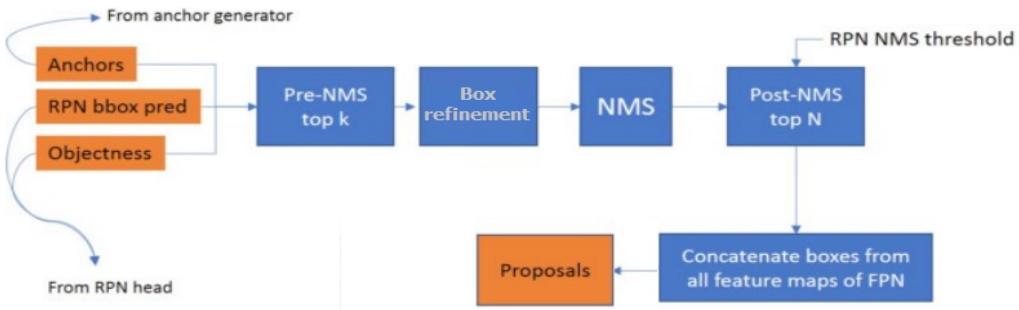
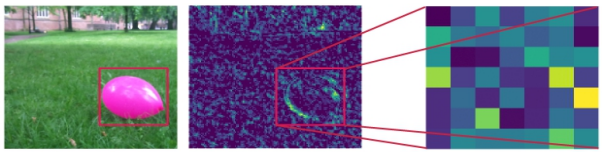


Figura 34 – Arquitetura do pós-processamento das regiões de interesse propostas. Adaptado de (Pande, 2019).

RoI Classifier and Bounding Box Regressor

Como as regiões de interesse têm diferentes dimensões começa-se por extrair essas regiões e redimensionar o seu tamanho para 7x7 (com 256 canais) utilizando uma camada de *RoI pooling* (Figura 35) (seleciona a zona de interesse no mapa de características, divide-a numa grelha 7x7 e extrai as características mais importantes, ou seja, os píxeis com valores mais altos de cada zona dessa grelha).



a) b) c)

Figura 35 – a) Exemplo de *RoI pooling*: Imagem original contendo uma região de interesse, b) Mapa de características com a região de interesse e c) Extração e redimensionamento da região de interesse (Abdulla, 2020).

De seguida a matriz resultante é achatada para servir de entrada a uma rede neuronal densa com 2 camadas *fully connected* do mesmo tamanho que se dividem posteriormente em 2 ramos, contendo outra camada *fully connected* que originam duas saídas (como na etapa anterior): a classificação das zonas de interesse e a continuação do refinamento das dimensões dos retângulos delimitadores previstos (Figura 36). Como a rede neuronal desta fase é mais profunda, permite classificar as regiões em diferentes classes (ao contrário da anterior que apenas fazia a distinção entre ter objeto ou não). Caso a região de interesse não tenha nenhum objeto é descartada. O refinamento das dimensões e coordenadas dos retângulos é feito de forma semelhante à etapa anterior.

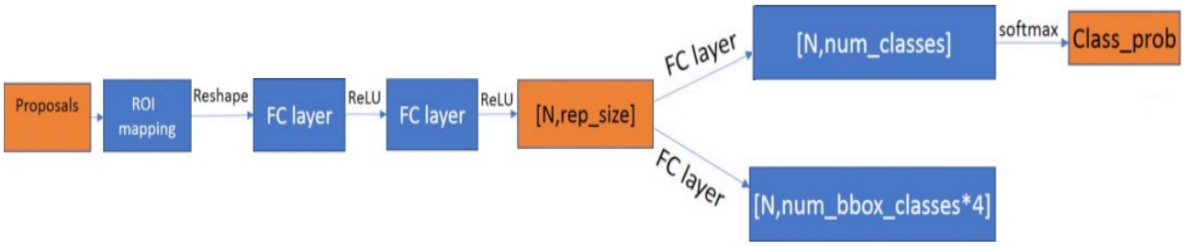


Figura 36 – Arquitetura da *RoI Classifier and Bounding Box Regressor.* Adaptado de (Pande, 2019).

Tal como na etapa anterior, removem-se os retângulos delimitadores previstos que tiverem uma classificação inferior a 0.5 e ajustam-se as dimensões e coordenadas dos retângulos

restantes tendo em conta as variações de coordenadas previstas. De seguida, para cada classe aplica-se o *non-maximum suppression* (*NMS*) para remover as regiões sobrepostas (IoU > 0.5) com pior classificação e no fim escolhem-se as 100 melhores regiões/retângulos previstos para cada imagem (Figura 37) (Pande, 2019; Brownlee, 2020).

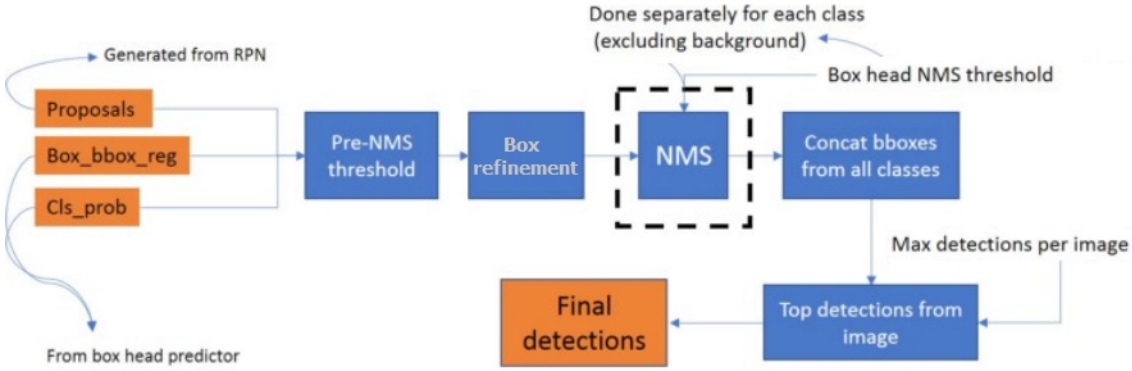


Figura 37 – Arquitetura do pós-processamento das regiões de interesse previstas. Adaptado de (Pande, 2019).

Nas camadas ocultas o modelo usa a função de ativação Relu e na camada final de classificação a função de ativação Softmax. A função de perda é composta por duas partes: a perda no ramo de classificação e a perda no ramo de regressão (prever as dimensões e localização dos retângulos). Para a perda na classificação utiliza-se a entropia cruzada categórica (*categorical cross entropy*) (Figura 38), que é uma variação da entropia cruzada para problemas multi-classe (que normalmente utilizam a função softmax), onde $\hat{y}$ é o valor previsto e y o seu valor real (1 se pertence à classe e 0 caso contrário).

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

Figura 38 – Função de custo – *Categorical Cross-Entropy*.

Na regressão usa-se a *Smoth L1 loss* (Figura 39) que penaliza linearmente as diferenças (x) entre os valores previstos e reais (altura, largura e coordenadas) quando são superiores a 1 e beneficia-os (menores perdas) quando são inferiores a 1, ou seja, quando o gradiente se está a aproximar do mínimo da função (Chen *et al.*, 2018).

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Figura 39 – Função de custo – *Smoth L1 loss*.

A avaliação final do desempenho do modelo é feita utilizando a métrica média da precisão média (mAP), no subconjunto de teste que pode ser obtida calculando a área debaixo da curva do gráfico *precision* vs *recall*. Considera-se uma previsão correta sempre que o valor da IoU é superior a 0.5. Este valor é calculado dividindo a área de sobreposição entre o retângulo delimitador real e previsto pela área de união entre eles (*intersection over union*). A precisão refere-se à percentagem de retângulos corretamente previstos (verdadeiros positivos) entre todos os retângulos previstos (verdadeiros positivos + falsos positivos). Para calcular o *recall* usa-se a percentagem de retângulos corretamente previstos (IoU>0.5) entre todos os retângulos reais (verdadeiros positivos + falsos negativos) (Hui, 2018).

4.2.2 Arquitetura da *Scaled-YOLOv4* e *YOLOv5*

A arquitetura dos modelos *Scaled-YOLOv4* e *YOLOv5* é semelhante diferindo apenas a forma como são implementados (por exemplo número de camadas, de neurónios, número e dimensões dos retângulos delimitadores, etc.). Estes modelos são compostos por 3 etapas, a primeira etapa é composta pelo ***backbone*** que é utilizado para extrair as características mais importantes de cada imagem através de uma *Cross Stage Partial Network* (*CSPN*), de seguida o ***model neck*** é usado para gerar *features pyramids* onde se obtém o mesmo objeto com escalas diferentes através da PANet e de seguida no *model head* são aplicados os retângulos delimitadores âncora (a*nchor boxes*) para obter os retângulos delimitadores finais, a probabilidade dos objetivos detetados e a probabilidade de um retângulo delimitador conter um objeto (para ajudar a definir as regiões de interesse – aquelas que têm objetos) (Figura 40).

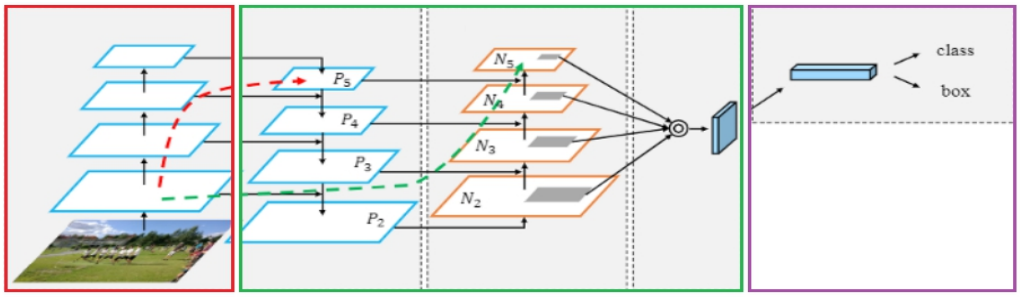


Figura 40 – Arquitetura da *YOLOv5*: a) *Backbone* (*Cross Stage Partial* Darknet-53), b) *Neck* (*PANet*) e c) *Head*. Adaptado de (Lui *et al.*, 2018).

Base do modelo (*backbone*)

Para reduzir os custos computacionais a *YOLOv5* utiliza como extrator de características a rede neuronal CSPDarknet-53 (*Cross Stage Partial* Darknet-53) (Figura 41) que divide o mapa de características obtido na camada base da Darknet-53 em 2 partes, com apenas uma dessas partes a ser submetida às camadas de convolução seguintes da rede Darknet-53, enquanto a outra parte é combinada através de uma camada de transição, com o mapa de características resultante do processo de convolução (Wang et al., 2019; Bochkovskiy et al., 2020). Esta rede neuronal utiliza a função de ativação Leaky Relu nas camadas ocultas e a Sigmoide na camada final de classificação. A função Relu ajuda a evitar o problema do gradiente desaparecendo (permite valores superiores a 1), é menos cara computacionalmente do que a tangente hiperbólica e Sigmoide e em alguns casos apresenta melhores resultados. A função Sigmoide (que varia entre 0 e 1) é usada para retornar a probabilidade associada à classe prevista (Pedamonti, 2018; Feng e Lu, 2019).

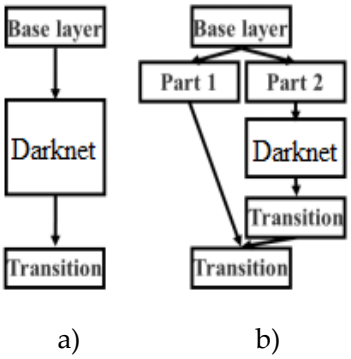


Figura 41 – a) Darknet-53 vs b) CSPDarknet-53. Adaptado de (Wang, 2019).

Para permitir a utilização de imagens de diferentes tamanhos a *YOLOv5* usa uma camada de *spatial pyramid pooling* (*SPP*) (Figura 42) em vez da camada de *pooling* original da Darknet-53, que aplica uma série de camadas de *pooling* ao mapa de características para que este resulte num vetor de tamanho fixo que sirva para alimentar a primeira camada da rede neuronal artificial. Por exemplo, a primeira camada divide o mapa de características numa grelha 4x4, a segunda camada divide o mapa numa grelha 2x2 e a terceira camada numa grelha 1x1, de seguida escolhem-se os valores maiores de cada elemento da grelha (*max-pooling*) e transformam-se as matrizes resultantes (matrizes de *pooling*) num vetor que terá as mesmas dimensões independentemente do tamanho do mapa de características (uma vez que as grelhas são de tamanho fixo) (He *et al.*, 2015).

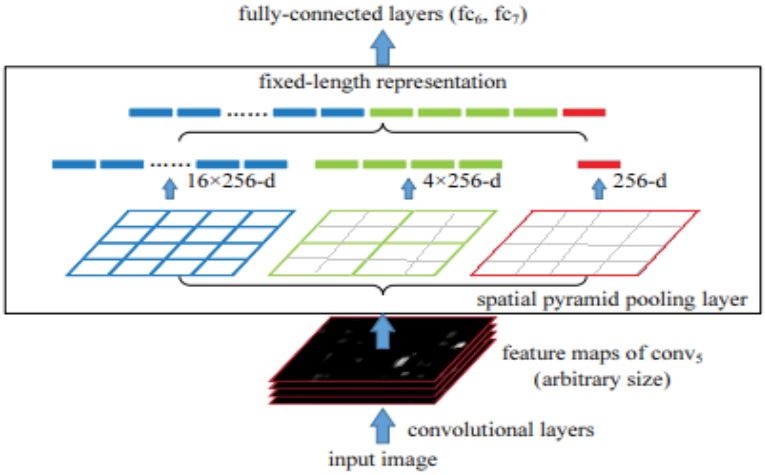


Figura 42 – a) *Spatial Pyramid Pooling*. Adaptado de (He *et al.*, 2015).

Model Neck

Na Feature *Pyramid Network* utilizada na versão 3 os mapas de características resultantes são usados independentemente para fazer deteções provocando duplicação de previsões e a não utilização de informação contida nos mapas de características que não são escolhidos. Na versão 5 da *YOLO* utiliza-se a PANet (*Path Aggregation Network*) (Figura 43) que combina a informação dos mapas de características (32x32, 16x16 e 8x8) obtidos das camadas de convolução da CSPDarknet-53.

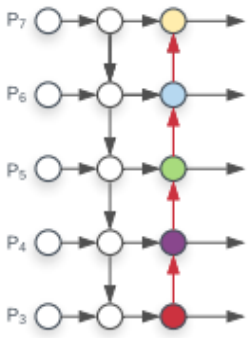


Figura 43 – a) PANet, com Pi a representar os mapas de características obtidos da CSPDarknet-53 (Tan *et al.*, 2020).

Com a *PANet* começa-se por obter novos mapas de características através da junção, de cima (da última camada) para baixo (para a primeira camada), dos mapas de características de diferentes camadas (como na *FPN*), de seguida obtêm-se os mapas de características finais através da junção de baixo (do último mapa de características obtido pela *FPN*) para

cima (para o primeiro mapa de características da *FPN*). Por último utiliza-se a operação *RoI Align* em cada mapa de características e une-se o resultado de cada matiz de *RoI Align*, resultante de cada mapa de características, escolhendo os valores mais altos de cada posição da matriz para obter o mapa de característica final (Lui *et al.*, 2018; Bochkovskiy *et al.*, 2020).

Model Head

Os mapas de características resultantes da fase anterior servem de entrada para a fase final do modelo que se divide em 2 partes para prever as dimensões dos retângulos delimitadores e a sua classe (e se contêm ou não um objeto). Esses mapas de características são divididos numa matriz 7x7 e a cada célula são atribuídos 3 retângulos delimitadores âncora (*anchor boxes*) de tamanhos diferentes. De seguida, para cada célula os retângulos âncoras dessa célula são comparados com os retângulos delimitadores reais que sobrepõem a célula. Para cada retângulo âncora prevê-se a diferença entre as suas coordenadas e as coordenadas do retângulo real (*dy, dx, dh, dw*), o *objectiviness score* que indica se o retângulo tem um objeto presente e a *class probabilities* que retorna a probabilidade de o objeto pertencer a uma classe. Para reduzir o número de retângulos âncoras utilizados durante o treino define-se um limite de confiança (*class probabilities**IoU) onde se ignoram os retângulos âncora com um valor abaixo desse limite (*confidence threshold* = 0.001). Para eliminar os retângulos previstos que sobrepõem o mesmo retângulo delimitador real usa-se o *non maximum supression* (*NMS*).

Com os retângulos previstos calcula-se a perda em relação aos retângulos reais. A função perda (*loss function*) contem 4 partes: a perda em relação ao centro, em relação à altura e largura, ao objeto (ter objeto ou não) e à classificação do objeto. Na primeira e segunda parte como é um problema de regressão (quer-se encontrar os valores que minimizam a diferença entre o centro, altura e largura do retângulo âncora e real) utiliza-se a função perda média do quadrado do erro (*Mean Square Error).* Na segunda e terceira parte como é um problema binário de classificação (a célula tem um objeto ou não) ou pode ser transformado em múltiplos problemas binários (o objeto pertence à classe A ou não e assim sucessivamente para todas as classes) usa-se a entropia cruzada binária (*binary cross entropy)* (Figura 44), que é uma variação da entropia cruzada para problemas de 2 classes (que utilizam normalmente a função Sigmoid), para cada classe e soma-se o resultado obtido (Li, 2019).

$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figura 44 – Função de custo – *Binary Cross-Entropy*.

A entropia cruzada binária usa o logaritmo do valor previsto ($\hat{y}$), obtido com a função de ativação Sigmoide, e o seu valor real (y) no cálculo da entropia, tendo em conta o tamanho do conjunto da amostra (*output size*). Caso a previsão seja igual ao valor real a entropia é zero.

4.3 Conclusão

Na deteção de pragas em armadilhas é necessário classificar e identificar vários insetos na mesma imagem da armadilha e contabilizá-los. Durante a fase de treino da classificação de

imagens são necessárias uma série de etapas para que se possa adaptar este método à deteção de pragas: obter a imagem do topo da armadilha, anotar manualmente os insetos presentes nas imagens, fazer o pré-processamento para remover o fundo da armadilha, fazer a segmentação de todos os objetos, ou seja insetos, extrair as características mais relevantes para classificar cada tipo de inseto presente na armadilha e, por fim classificar cada tipo de inseto individualmente.

Com os modelos de deteção de objetos o processo é mais simples e apenas é necessário obter a imagem, anotar manualmente os insetos presentes nas imagens e utilizar essas imagens anotadas para treinar os modelos de deteção de objetos.

Por esse motivo a abordagem proposta inclui o desenvolvimento de modelos de deteção de objetos (*R-CNN*, *Scaled-YOLOv4*, *YOLOv5 (Small)* e *YOLOv5 (XLarge)*) para deteção da praga mosca-branca.

Como não é possível saber qual dos modelos tem um melhor desempenho com o conjunto de dados utilizado sem os implementar, é necessário que a abordagem proposta inclua a implementação dos 4 modelos.

No entanto o modelo *R-CNN* e os modelos *YOLO* (*Scaled-YOLOv4* e *YOLOv5*) têm algumas diferenças que os podem tornar mais adequados à resolução do problema em estudo.

Durante a fase de treino, a *R-CNN* olha duas vezes para a imagem, uma para prever as regiões de interesse (píxeis que contêm insetos e suas coordenadas) e outra para identificar os objetos dentro dessas zonas (classificar os insetos e prever as suas coordenadas). Os modelos *YOLO* olham apenas uma vez para a imagem porque fazem logo a classificação dos insetos e a previsão das suas coordenadas.

Na base do modelo, a *R-CNN* utiliza uma rede neuronal convolucional Resnet101 para obter os mapas de características e os modelos *YOLO* usam uma rede neuronal convolucional *CSPN* – Darknet-53. A *R-CNN* utiliza as quatro camadas de convolução para retornar os mapas de características e os modelos *YOLO* usam apenas as três últimas camadas de convolução.

Após obtenção dos mapas de características a *R-CNN* utiliza a *Feature Pyramid Network* (*FPN*) para fazer o processamento destes mapas e os modelos *YOLO* usam a *Path Aggregation Network* (*PAN*). A *FPN* da *R-CNN* retorna quatro mapas de características finais (mapas de características iniciais após processamento) onde a cada época apenas é escolhido um desses mapas de características finais para fazer as previsões, o que pode provocar a não utilização de informação contida nos mapas que não forem escolhidos.

Para validar as anotações acrescentadas ao conjunto de dados e as previsões feitas pelos modelos de deteção de objetos inclui-se, na abordagem proposta, o contributo de um técnico para confirmar os resultados obtidos.

O protótipo incluído na abordagem proposta permite demonstrar que a visão por computador pode ser utilizada no combate a pragas, de forma fácil e económica, através do *upload* de imagens já existentes ou permitindo que a aplicação aceda à câmara do dispositivo.

5 Implementação e Resultados

A implementação do modelo de deteção de pragas começou pela definição do conjunto de dados a usar, de seguida implementaram-se os modelos de deteção de objetos descritos na secção 4.2, foi feito o pré-processamento dos dados para tentar melhorar os resultados, avaliaram-se e validaram-se esses resultados, geraram-se imagens artificiais e por último desenvolveu-se um protótipo. Esta implementação foi dividida em 7 fases (Figura 45):

- **Obtenção dos dados:** Definir um "*Conjunto de dados base*" e alterar o formato das anotações para que possam ser utilizadas por todos os modelos de deteção de objetos.
- **Definição dos modelos:** Definir os modelos de deteção de objetos a usar. Nesta fase testam-se todos os modelos de deteção de objetos no "*Conjunto de dados base*" para encontrar o modelo com maior precisão (mAP).
- **Pré-processamento:** Criar novos conjuntos de dados, aumentados, reduzidos e com várias divisões treino/teste. Nesta fase utiliza-se o modelo de deteção de objetos escolhido na fase anterior para encontrar o conjunto de dados com o qual se obtém a precisão (mAP) mais elevada.
- **Avaliação:** Avaliar os melhores modelos de deteção de objetos no conjunto de dados escolhido na fase anterior, considerando como critério de avaliação a mAP, número de moscas-brancas detetadas, tempo de previsão de uma imagem e memória ocupada pelo modelo. Nesta fase pretende-se escolher o modelo com melhor desempenho.
- **Validação:** Validar o modelo escolhido na fase anterior. Nesta fase pretende-se que os técnicos especialistas em deteção de pragas validem o modelo escolhido para avançar para a próxima fase ou reiniciar o fluxo da implementação caso os técnicos não validem o modelo.
- **Pós-processamento:** Gerar imagens artificiais usando uma rede neuronal generativa (GAN) para aumentar o conjunto de dados do modelo escolhido e assim tentar melhorar o seu desempenho. As imagens artificiais obtidas não foram suficientemente naturais e por isso esta fase não foi incluída no fluxo da implementação.
- **Protótipo:** Fazer o *deployment* do modelo de deteção de objetos escolhido, desenvolver um protótipo e fazer a sua prova de conceito.

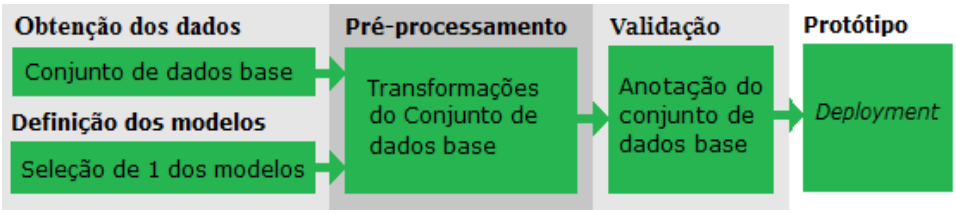


Figura 45 - Fluxo da implementação.

5.1 Obtenção dos dados

Nesta fase construiu-se o "*Conjunto de dados base"* e alterou-se o formato das anotações para que pudessem ser utilizadas por todos os modelos de deteção de objetos.

5.1.1 Imagens do Conjunto de dados base

As imagens usadas pertencem ao conjunto de dados de (Nieuwenhuizen *et al.*, 2018) que contém 284 imagens contendo dois tipos de moscas-brancas/*Whiteflies* (ambas com a anotação WF) e os insetos predadores *Macrolophus* (MR) e *Nesidiocoris* (NC). O conjunto de dados contém 5611 instâncias de anotadas de *Whiteflies, 1*341 instâncias anotadas de *Macrolophus* e 511 instâncias anotadas de *Nesidiocori.* Para obter o "*Conjunto de dados base*" removeram-se as anotações de MR e NC, não se utilizaram imagens que continham anotações erradas (as anotações não se sobreponham aos objetos/moscas-brancas), não se utilizaram as imagens que não continham moscas-brancas e removeram-se as imagens que aparentemente continham mais moscas-brancas não anotadas. O "*Conjunto de dados base*" ficou reduzido a 174 imagens (Figura 46) com 4940 anotações de WF.

Figura 46 – Exemplo de imagem pertencente ao "*Conjunto de dados base*" (Nieuwenhuizen *et al.*, 2018).

5.1.2 Anotações do Conjunto de dados base

A anotação do conjunto de dados de (Nieuwenhuizen *et al.*, 2018) foi efetuada no formato PASCAL VOC XML, usando o *software* LabelImg (Tzutalin, 2015). Com este tipo de *software* define-se o rótulo de cada objeto (*Whiteflie*) posicionando uma janela retangular rotulada sobre cada objeto a classificar. As anotações das moscas-brancas correspondem aos retângulos definidos pelos círculos verdes (Figura 47). A informação que a etiqueta ("264-7") pode fornecer não foi utilizada porque não permite identificar o número da armadilha nem a sua posição numa sequência temporal de imagens.

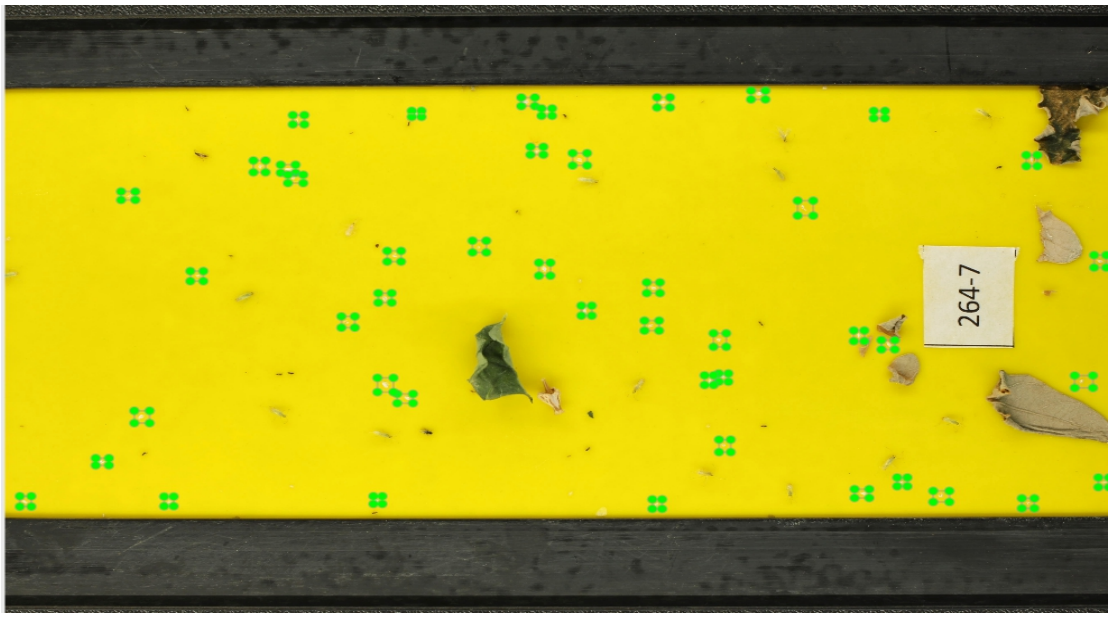
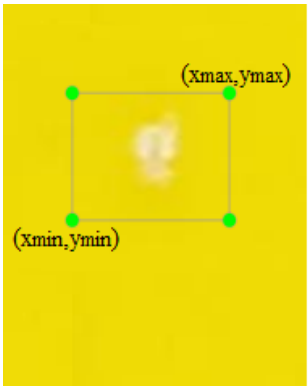


Figura 47 – Exemplo de uma imagem anotada pertencente ao "Conjunto de dados base" (Nieuwenhuizen et al., 2018).

Deste processo resulta, para cada imagem, um ficheiro XML contendo a dimensão da imagem e a descrição de cada janela retangular (coordenadas e rotulo, situando-se a coordenada (0,0) no canto superior esquerdo da imagem). Do ficheiro XML retiram-se as dimensões da imagem do diretório <size>, as coordenadas dos retângulos delimitadores do diretório <bndbox> e o nome da anotação do objeto <name> (Brownlee, 2020). As coordenadas (*xmin*,*ymin*) referem-se ao canto inferior esquerdo da anotação e (*xmax*,*ymax*) ao canto superior direito (Figura 48).




```
<object>
        <name>WF</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>60</xmin>
                <ymin>692</ymin>
                <xmax>98</xmax>
                <ymax>737</ymax>
        </bndbox>
</object>
```

Figura 48 – Exemplo da anotação de um inseto no formato PASCAL VOC utilizado no modelo de deteção de objetos *R-CNN*.

Como o modelo *YOLO* usa as anotações num formato diferente (*YOLO* Darknet) do utilizado pela rede neuronal por regiões (PASCAL VOC XML), converteram-se as anotações originais resultando, para cada imagem, um ficheiro txt contendo a *id* das anotações (0 = *Whiteflie*), as coordenadas (*xcenter*, *ycenter*) referentes ao centro das anotações normalizadas e as larguras (*width*) e alturas (*height*) normalizadas de cada anotação (Figura 49).

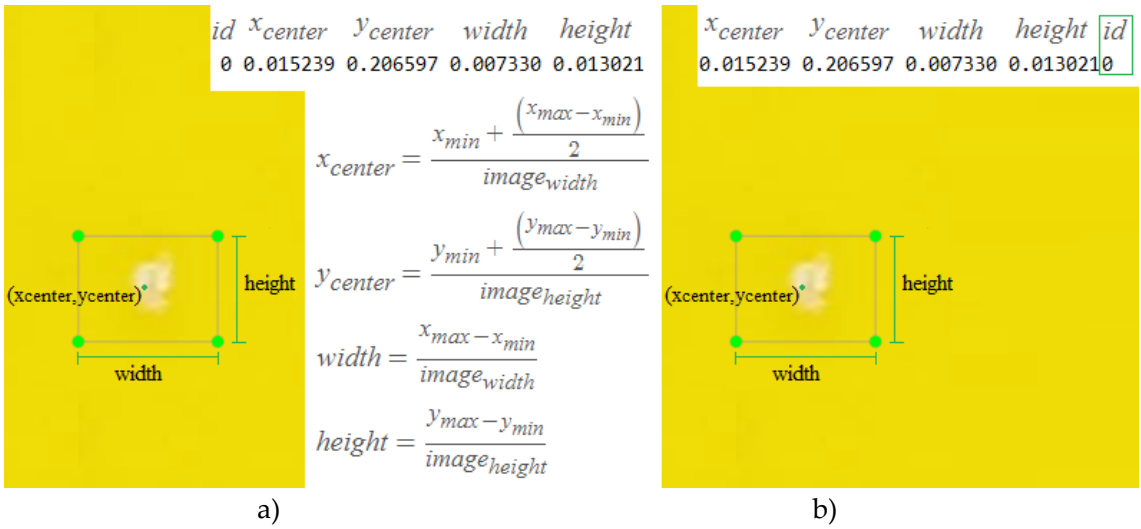


Figura 49 – Exemplo da anotação de um inseto no formato *YOLO*: a) *YOLOv5* e b) *Scaled-YOLOv4*.

5.2 Definição dos modelos

Nesta fase definiram-se os modelos de deteção de objetos a usar. No fim testaram-se todos os modelos no "*Conjunto de dados base*" e obtiveram-se os primeiros resultados.

Os modelos de deteção de objetos implementados são a rede neuronal convolucional por regiões com a arquitetura descrita na secção 3.2.1. e a *YOLOv5* e *Scaled-YOLOv4* com a arquitetura descrita na secção 3.2.2.

5.2.1 *R-CNN*

A implementação da rede neuronal por regiões é feita utilizando a biblioteca mrcnn (v2.1) (Abdulla, 2017) que permite usar e treinar redes neuronais por regiões para deteção de objetos, utilizando a versão 1 do TensorFlow e 2.2 do Keras. Com esta biblioteca evita-se a criação de modelos de raiz e acelera-se o processo de treino usando *transfer learning* de modelos pré-treinados em conjuntos de dados como o Microsoft COCO *dataset* (composto por 91 tipos de objetos com mais de 2.5 milhões de instâncias catalogadas e 328 mil imagens).

Esta biblioteca utiliza o modelo *Mask R-CNN* para deteção de objetos e sua segmentação, mas como o conjunto de dados usado não possuí máscaras, apenas se utilizou este modelo para deteção de imagens e não se tirou partido das suas capacidades de segmentação (Brownlee, 2020).

5.2.2 *Scaled-YOLOv4*

A *Scaled-YOLOv4* é o modelo de deteção de objetos com artigo publicado, que obtém os melhores resultados no conjunto de dados COCO (usado como referência para avaliar modelos de deteção de objetos). De forma semelhante à *YOLOv5*, a *Scaled-YOLOv4* permite escalar o modelo *YOLOv4* e implementar vários modelos da mesma arquitetura (desde o *Tiny* ao *Large*), variando o número de camadas e de neurónios por camada utilizando os parâmetros *depth_multiple* e *width_multiple* (Wang *et al.*, 2020).

A implementação da *Scaled-YOLOv4* foi feita utilizando o modelo *Large*, *Scaled-YOLOv4 (Large)*, da biblioteca *ScaledYOLOv4* disponibilizada em PyThorch (King-Yiu, 2020). O

treino foi inicializado usando os valores dos pesos pré-treinados com o conjunto de dados COCO e posteriormente treinaram-se todas as camadas do modelo.

5.2.3 *YOLOv5*

A implementação da *YOLOv5* é feita utilizando a versão v3.1 (Jocher, 2020b) da biblioteca yolov5 (disponibilizada pela ultralytics em junho de 2020) implementada usando o PyThorch (v >= 1.6.0) e que também permite utilizar *transfer learning* de modelos pré-treinados no conjunto de dados Microsoft COCO. Esta biblioteca com base nos parâmetros *depth_multiple* e *width_multiple* permite implementar 4 arquiteturas do mesmo modelo, variando o número de camadas e de neurónios por camada.

Estes parâmetros foram primeiramente utilizados no modelo EfficientDet para encontrar, através de *grid search*, a conjugação ótima de camadas e neurónios que permite à rede neuronal crescer, mantendo o melhor desempenho computacional possível enquanto melhora a precisão do modelo.

Para a implementação escolheram-se dois modelos: o mais profundo dos quatro, *YOLOv5 (XLarge)* que usa *depth_multiple*=1.33 e *width_multiple*=1.25 (multiplica o número de camadas do modelo original por 1.33 e o número de neurónios por 1.25) e o modelo *YOLOv5 (Small)* que é o menos profundo (Tan *et al.,* 2020). O treino foi inicializado utilizando os valores dos pesos pré-treinados com o conjunto de dados COCO e posteriormente treinaram-se todas as camadas do modelo.

Parâmetros dos modelos

A melhor conjugação de parâmetros de valores decimais dos modelos *YOLO* foi encontrada utilizando o algoritmo genético disponibilizado pela biblioteca yolov5 (Jocher, 2020b) e da biblioteca *ScaledYOLOv4* (King-Yiu, 2020).

Nesse algoritmo começa-se por inicializar os parâmetros com os seus valores predefinidos e definindo a mAP como função de ajuste. De seguida, a evolução dos parâmetros é feita criando um cenário base que se pretende evoluir ao longo de 100 gerações. Definiu-se como cenário base, melhorar o desempenho (mAP) da deteção de mosca-branca ao fim de 80 épocas.

O cenário base é evoluído escolhendo os melhores indivíduos (maior mAP) das gerações anteriores (pais) para criar a próxima geração (filhos). Os novos indivíduos (filhos) são criados aplicando a mutação uniforme com uma probabilidade de 90%. A cada geração, os filhos vão competir com os candidatos mais antigos (pais) por um lugar na próxima geração com base no seu ajuste (mAP). Este processo é repetido durante 100 gerações para encontrar a melhor conjugação de hiper-parâmetros (Jocher, 2020a).

A escolha dos pais para criar a geração seguinte é feita selecionando no máximo os 5 melhores indivíduos (pais) das gerações anteriores. Não se utiliza o operador de recombinação (*crossover*) por ser demasiado destrutivo e utiliza-se a mutação uniforme por se pretender evoluir valores decimais. Na mutação uniforme substitui-se o valor de um gene por um valor aleatório uniforme selecionado entre os limites superior e inferiores definidos para esse gene (definiu-se como valores limites os valores das mutações das épocas anteriores). Os melhores parâmetros para o modelo *R-CNN* foram encontrados usando a *grid search*, partindo dos valores pré-definidos e testando novas combinações de parâmetros dentro de um conjunto de valores definidos. Após definição dos melhores parâmetros, os modelos *YOLO* foram treinados durante 300 épocas e a *R-CNN* foi treinada durante 100 épocas.

O objetivo dos modelos de deteção de objetos é aprender a prever anotações semelhantes às reais, tanto na classe dos objetos como na dimensão dos retângulos que delimitam esses objetos. Assim a forma de inicializar as anotações previstas (*anchor boxes*) influencia o desempenho do modelo. Para encontrar as dimensões iniciais (largura e altura) das anotações previstas pode utilizar-se o algoritmo k-means para agrupar em 9 *clusters* as dimensões (largura e altura) das anotações reais e usar os centros desses *clusters* para inicializar as anotações previstas.

Utilizando o k-means definem-se 9 *clusters* (Figura 50), de seguida de entre todas as anotações escolhe-se aleatoriamente a largura e altura de 9 anotações para definir o centro desses *clusters.* A largura e altura das anotações reais restantes são atribuídas ao *cluster* mais próximo, usando como métrica de distância a distância euclidiana. Calcula-se novamente o centro dos *clusters* e atualiza-se o seu valor. Estes passos são repetidos até o valor dos centros dos *clusters* não se alterarem de uma iteração para a seguinte.

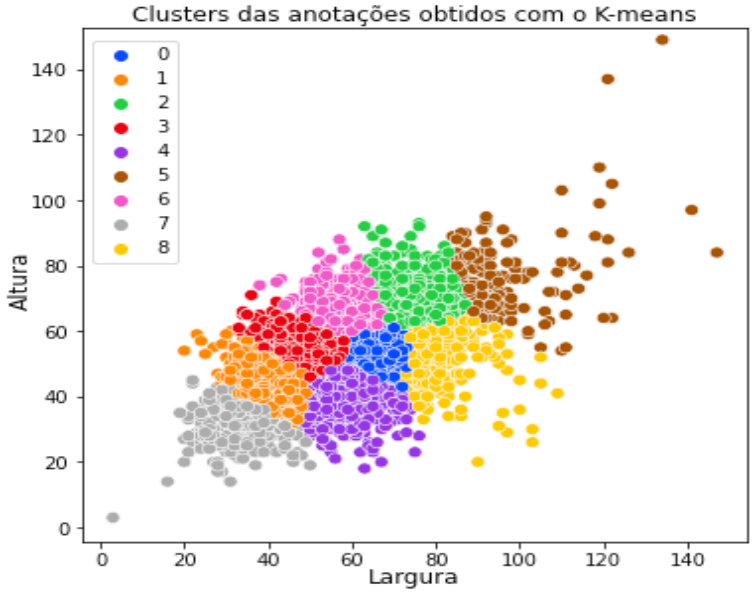


Figura 50 – Altura e largura das anotações (Nº de píxeis).

5.2.4 Resultados

Nesta fase testaram-se os modelos de deteção de objetos escolhidos no "*Conjunto de dados base*". Os resultados apresentam o modelo *YOLOv5 (XLarge)* com melhor desempenho (mAP=82,70%) e 17,81% superior em relação ao modelo com pior desempenho (*R-CNN*) (Figura 51).

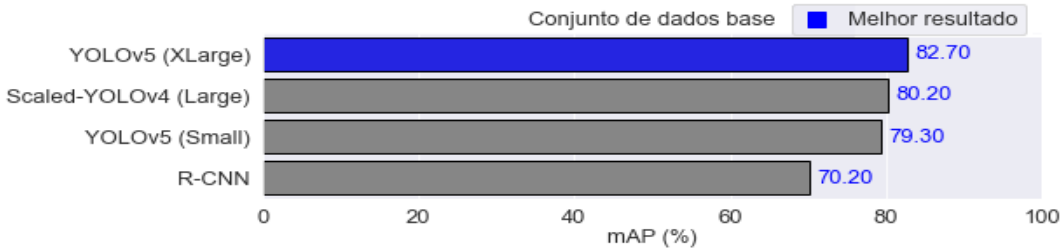


Figura 51 – Avaliação dos modelos de deteção de objetos no "Conjunto de dados base".

5.3 Pré-processamento

Nesta fase obtiveram-se novos conjuntos de dados a partir do "*Conjunto de dados base*" e testou-se o modelo que obteve melhores resultados na fase anterior, *YOLOv5 (XLarge)*, em todos os novos conjuntos de dados.

5.3.1 Conjuntos de dados aumentados

A partir do conjunto de dados de treino do "*Conjunto de dados base*" foram obtidas novas imagens e as respetivas anotações, através de 7 tipos de transformações:

Desfocado, onde os píxeis são desfocados, **Escala**, onde as imagens são aumentadas ou reduzidas (Figuras 52), **Iluminação**, onde se aumenta ou se reduz a luz de cada imagem, **Inversão**, onde se invertem horizontalmente as imagens (Figuras 53), **Rotação**, onde se roda a imagem *x* graus (Figuras 54), **Ruído**, onde se introduz um efeito "granulado", **Translação**, onde se efetua translações horizontais e verticais (Figuras 55).

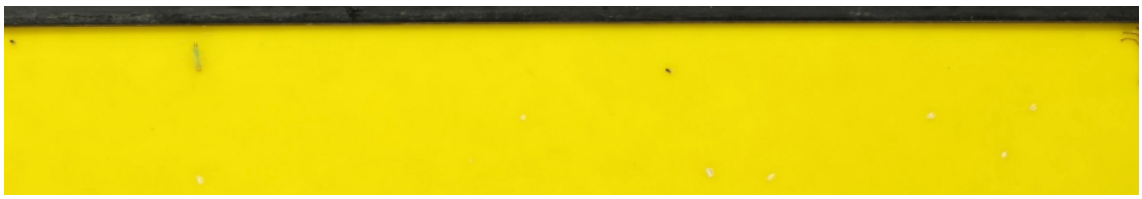


Figura 52 – Parte de uma imagem do "Conjunto de dados base" após Zoom (Escala).

Figura 53 – Parte de uma imagem do "Conjunto de dados base" após Inversão.

Figura 54 – Parte de uma imagem do "Conjunto de dados base" após Rotação.

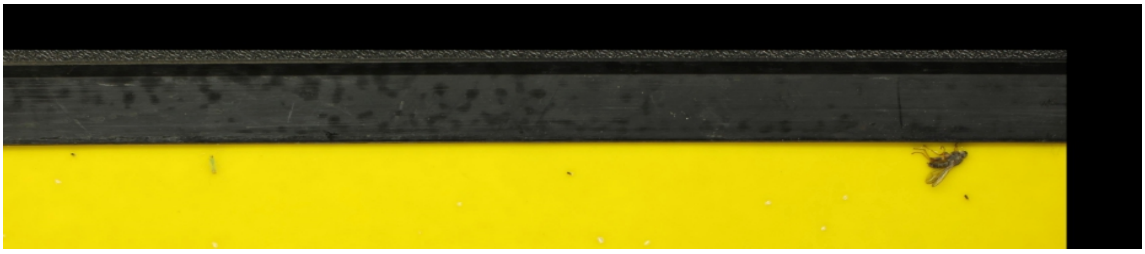


Figura 55 – Parte de uma imagem do "Conjunto de dados base" após Translação.

Cada transformação deu origem a um novo subconjunto de treino contendo as 139 imagens originais mais 139 imagens resultantes do tipo de transformação escolhida (duplicou-se o subconjunto de treino). O subconjunto de teste manteve-se igual para todos os 8 novos conjuntos de dados e contém apenas as 35 imagens do subconjunto de teste base. Os novos conjuntos de dados foram utilizados para treinar o modelo de deteção de objetos *YOLOv5*.

Para testar a adição de imagens sem moscas-brancas aumentou-se o "*Conjunto de dados base*" com 139 imagens de armadilhas amarelas sem instâncias de moscas-brancas anotadas. As 139 imagens foram adicionadas ao subconjunto de treino ficando com 279 imagens e 3565 instâncias anotadas de WF. O subconjunto de teste manteve as mesmas 35 imagens e com 1374 instâncias anotadas de WF (Figura 56). Os resultados obtidos com estes conjuntos de dados aumentados são apresentados na Figura 59.

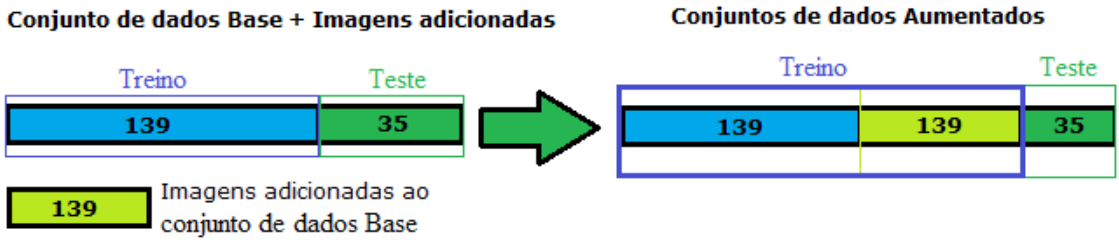


Figura 56 – Transformação do "Conjunto de dados base" para os diferentes conjuntos de dados aumentados.

5.3.2 Diferentes subconjuntos de treino e teste

O "*Conjunto de dados base*" usado contém 174 imagens com 4940 instâncias anotadas de moscas-brancas. Para fazer a avaliação preliminar do desempenho dos modelos de deteção de objetos construíram-se 5 conjuntos de dados utilizando 139 imagens para treino (80%) e 35 imagens (20%) para teste. No "*Conjunto de dados Teste 1*" usam-se as primeiras 35 imagens para teste e as restantes para treino. No "*Conjunto de dados de dados Teste 2*" utilizam-se as segundas 35 imagens para teste e as restantes para treino. No "*Conjunto de dados de dados Teste 3*" usam-se as terceiras 35 imagens para teste, no "*Conjunto de dados Teste 4*" utilizam-se as quatro 35 imagens para teste e no "*Conjunto de dados base*" usam-se as últimas 35 imagens para teste e as restantes para treino (Figura 57). Os resultados obtidos com estes diferentes subconjuntos de dados são apresentados na Figura 60.

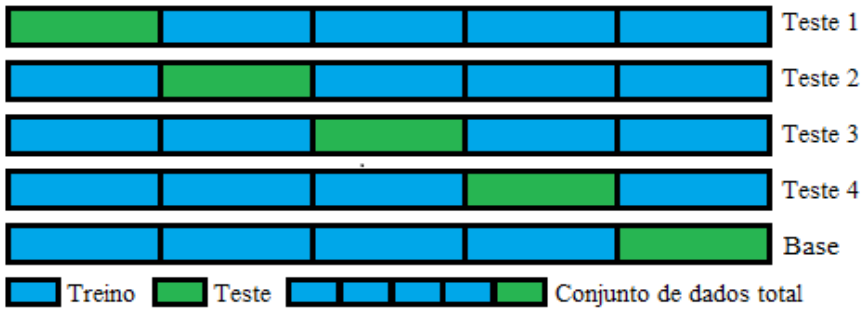


Figura 57 – Diferentes repartições do conjunto de dados.

5.3.3 Conjunto de dados reduzido

Nesta fase não era possível completar a anotação do *"Conjunto de dados base"* com a correção necessária e o contributo dos técnicos para essa tarefa seria mais valioso numa fase mais adiantada, onde os modelos de deteção de objetos teriam um melhor desempenho. Por esses motivos e para verificar o efeito da falta de anotações, removeram-se algumas das imagens com moscas brancas não anotadas. Entre as imagens que contêm moscas-brancas não anotadas optou-se por remover algumas das que fornecem a mesma informação (fase da mosca-branca, cor de fundo, outro tipo de insetos) que outras imagens com anotação mais completa.

O "*Conjunto de dados base*" foi reduzido removendo 27 imagens que contêm instâncias não anotadas de WF. A avaliação foi feita utilizando as primeiras 117 imagens para treino e as

restantes 30 imagens para teste. As 27 imagens removidas, como contêm as imagens onde existe a maior diferença entre as instâncias previstas e anotadas (contém mais moscas-brancas não anotadas), foram usadas para fazer a validação das previsões pelos técnicos na próxima fase (Figura 58). Os resultados obtidos com o "*Conjunto de dados reduzido*" é apresentado na Figura 61.

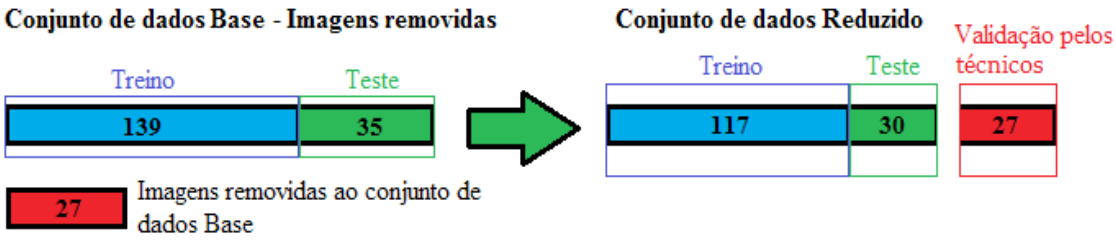


Figura 58 – Transformação do "Conjunto de dados base" para "Conjunto de dados reduzido".

5.3.4 Resultados

Os resultados obtidos (considerando a mAP) são semelhantes para os conjuntos de dados desfocado, iluminação, rotação e inversão e as diferenças de desempenho podem resultar da natureza estocástica do modelo de deteção de objetos.

O pior desempenho do "*Conjunto de dados aumentado através de ruído*" (mAP=79,60) pode dever-se à alteração dos píxeis dos objetos que se alteram da cor branca para pequenas manchas cinzentas. O pior desempenho do "*Conjunto de dados aumentado através de escalas*" (mAP=79,20%) pode dever-se às imagens transformadas com diminuição da escala e consequentemente dos objetos. A adição de imagens de armadilhas sem instâncias de moscas-brancas, ao "*Conjunto de dados base*" não melhora o desempenho do modelo, ou seja, não o ajuda a identificar melhor as instâncias que são moscas-brancas (Figura 59).

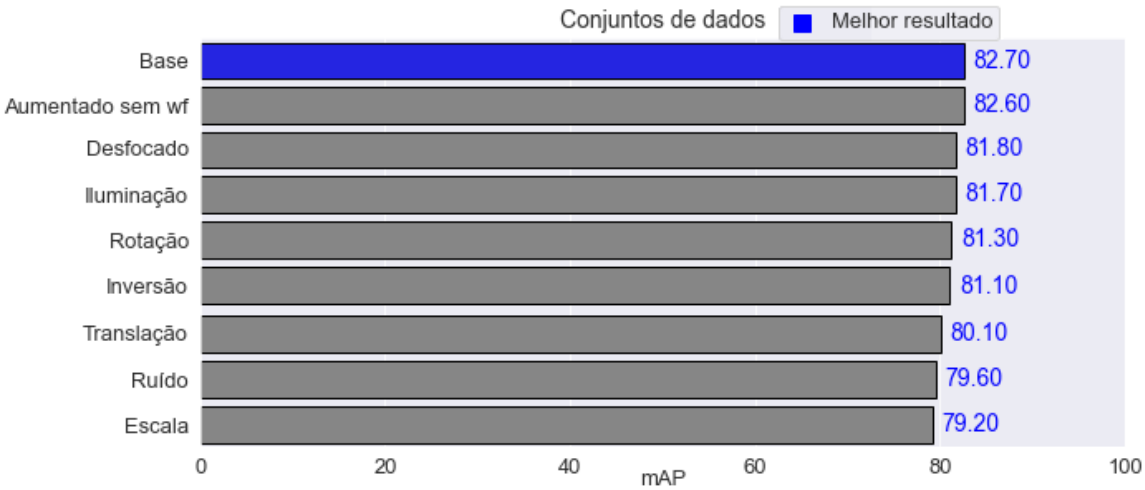


Figura 59 – Testes do modelo *YOLOv5 (XLarge)* nos conjuntos de dados aumentados.

A adição das imagens não melhora o desempenho do modelo. O único conjunto de dados onde o resultado não piora significativamente em relação ao "*Conjunto de dados base*" é o conjunto de dados aumentado com imagens sem moscas-brancas ("*Conjunto de dados aumentado sem WF*").

Os resultados obtidos nos restantes conjuntos de dados aumentados parecem indicar que o desempenho do modelo *YOLOv5* não melhora com estes conjuntos de dados, porque ao aumentar o número de imagens também se aumenta o número de instâncias de moscas-brancas não anotadas (Figura 59). Apesar da remoção de algumas imagens com instâncias

de moscas-brancas não anotadas do "*Conjunto de dados base*" ainda ficaram presentes muitas instâncias não anotadas. Para verificar o efeito das moscas-brancas não anotadas testaram-se várias combinações de treino e teste (variou-se o número de instâncias não anotadas nos subconjuntos de treino e teste).

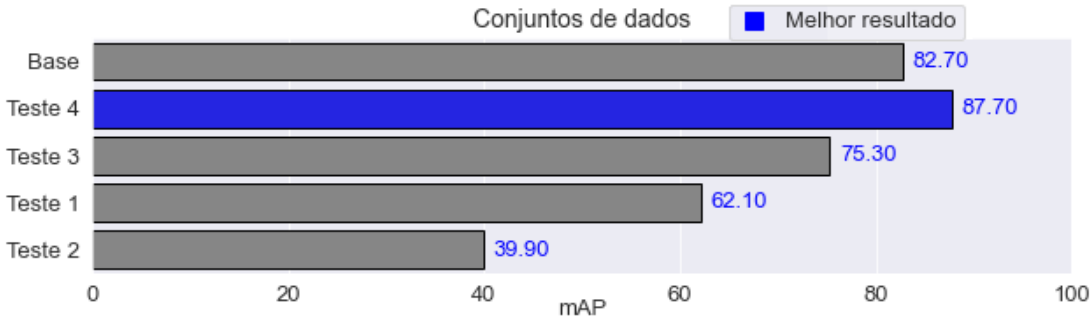


Figura 60 – Testes do modelo *YOLOv5 (XLarge)* nos diferentes subconjuntos de treino e teste.

Os resultados obtidos utilizando os conjuntos de dados *Teste 1*, *Teste 2* e *Teste 3* refletem a maior presença de instâncias de moscas-brancas não anotadas nos seus subconjuntos de teste. Pelo contrário, o "*Conjunto de dados Teste 4*" (mAP=87,70%) é aquele que tem o menor número de instâncias não anotadas no subconjunto de teste. Esse resultado não é válido porque se escolheu o subconjunto de teste para que este apresentasse o menor número de instâncias não anotadas possível (Figura 60). Como os resultados obtidos nos diferentes subconjuntos de treino e teste confirmam que as instâncias não anotadas influenciam significativamente o desempenho do modelo, procedeu-se à remoção de imagens com muitas instâncias não anotadas.

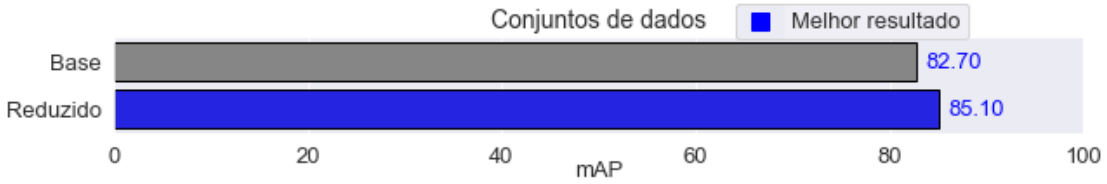


Figura 61 – Testes do modelo *YOLOv5 (XLarge)* no "Conjunto de dados reduzido".

A remoção de imagens com muitas instâncias não anotadas ao "*Conjunto de dados base*" melhora o desempenho do modelo (mAP=85,10%), porque existem menos instâncias não anotadas a serem detetadas como moscas-brancas (Figura 61).

5.4 Avaliação

Nesta fase foi avaliado o desempenho de todos os modelos de deteção de objetos no conjunto de dados para o qual se obteve o melhor resultado na fase anterior, o "*Conjunto de dados reduzido*". Para avaliar o desempenho dos modelos considerou-se a sua precisão, memória ocupada e rapidez. Estes critérios permitem avaliar qual é o modelo mais indicado para cada dispositivo e a troca (*trade-off*) entre precisão e rapidez ou memória.

5.4.1 Critérios

Para avaliar cada modelo consideraram-se 4 critérios:

- **Precisão (mAP):** Critério utilizado normalmente para avaliar a precisão dos modelos de deteção de objetos. Como existem muitas instâncias de moscas-brancas sem anotações também foi considerado o número de moscas-brancas detetadas.

- **Nº de moscas-brancas detetadas:** Pretende-se detetar o máximo de instâncias de moscas-brancas possível (anotadas e não anotadas).
- **Memória (MB):** Memória ocupada pelo modelo. Apesar de não ser um requisito, muitos modelos necessitam de ocupar pouca memória para poderem ser utilizados em dispositivos moveis.
- **Tempo (s):** Tempo médio que o modelo demora a detetar as instâncias de moscas-brancas numa imagem (utilizando o processador NVIDIA Tesla T4 GPU).

A precisão (mAP) permite comparar a precisão destes modelos com outros que venham a ser desenvolvidos em trabalhos futuros. O critério "*Nº de moscas-brancas detetadas*" permite contornar a falta de anotações em algumas instâncias e que pioram a precisão do modelo (mAP), apesar de não piorarem o desempenho do modelo. Também permite que os técnicos tenham uma métrica que possa ser utilizada para comparação com o número de moscas-brancas que se consegue detetar manualmente. A memória ocupada permite avaliar qual é o modelo mais indicado se existirem restrições de memória. O tempo médio de deteção de uma imagem permite avaliar se é plausível obter deteções em tempo real.

5.4.2 Resultados

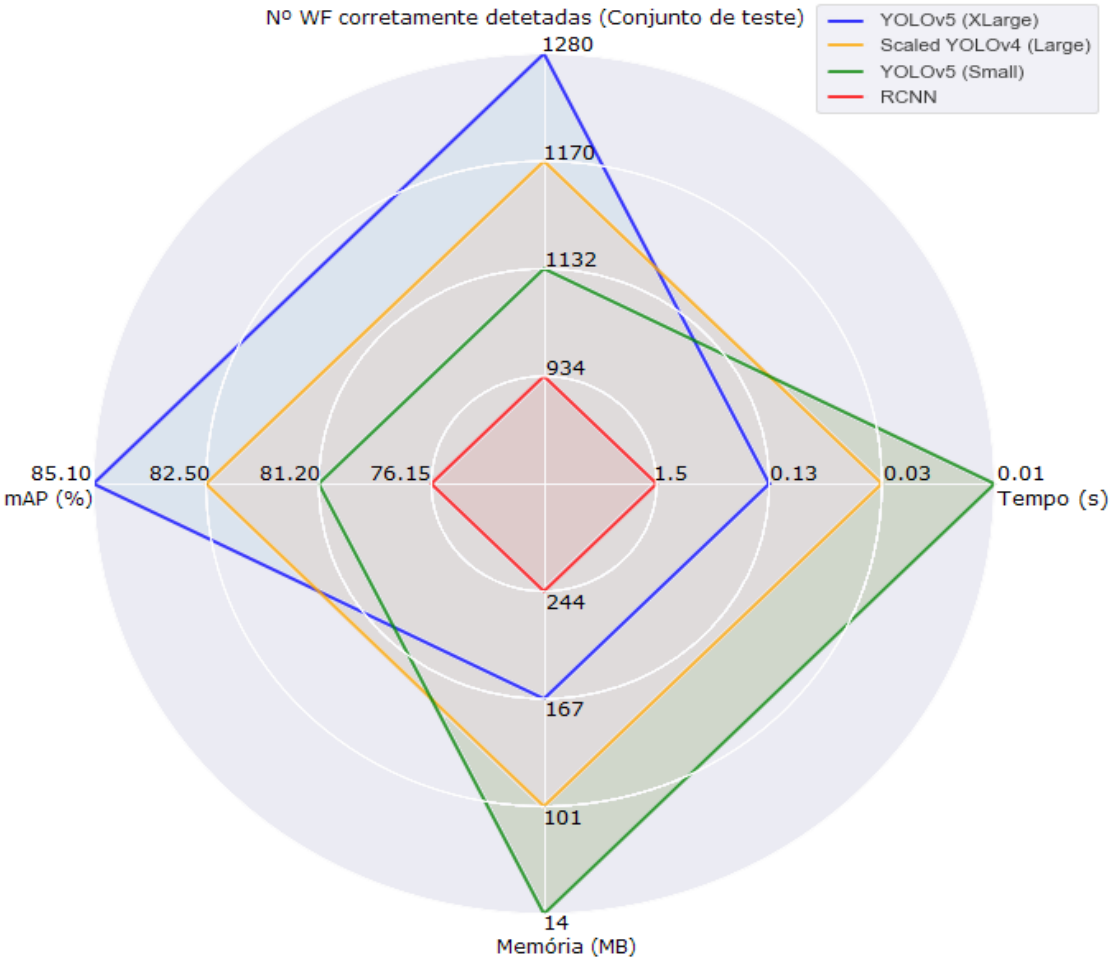


Figura 62 – Avaliação dos modelos de deteção de objetos no "Conjunto de dados reduzido".

O modelo que obteve a maior mAP para o "*Conjunto de dados reduzido*" foi a *YOLOv5 (XLarge)* com 85,10% (Figura 62). Em relação à *YOLOv5 (XLarge)*, a *Scaled-YOLOv4* obteve uma mAP inferior em -3,15% (deteta menos instâncias anotadas). A *YOLOv5 (Xlarge)*

detetou 1213 instâncias de moscas-brancas no subconjunto de teste (do "*Conjunto de dados reduzido*"), o que representa um incremento de 9,41% em relação à *YOLOv4 (Figura 62)*. No entanto a *YOLOv4* ocupa -39,5% de memória do que a *YOLOv5 (XLarge)* e têm aproximadamente o mesmo tempo médio de deteção para uma imagem de uma armadilha (Figura 62). Caso não existam restrições de memória ou tempo médio de deteção a *YOLOv5 (Xlarge)* é o melhor modelo. Caso contrário, como em dispositivos móveis a Scaled-*YOLOv4* ou a *YOLOv5 (Small)* podem ser os modelos mais indicados.

5.4.3 Interpretação dos resultados

Para interpretar os resultados foram escolhidas as deteções feitas pelo modelo *YOLOv5 (Xlarge)* que apresenta o melhor desempenho em precisão (mAP) e maior número de moscas-brancas detetadas. As deteções feitas, no subconjunto de teste e validação do "*Conjunto de dados reduzido*", são compostas por 5 elementos:

- **Verdadeiros positivos (TP):** Moscas-brancas que estão anotadas e são detetadas como moscas-brancas;
- **Verdadeiros positivos não anotados (TPNA):** Moscas-brancas que não estão anotadas e são detetadas;
- **Verdadeiros negativos (TN):** Outros insetos que não moscas-brancas e que não são detetados como sendo moscas-brancas;
- **Falsos positivos (FP):** Outros insetos que não moscas-brancas e que são detetados como sendo moscas-brancas;
- **Falsos negativos (FN):** Moscas-brancas que não são detetadas como sendo moscas-brancas.

Tabela I – Deteções do modelo *YOLOv5 (Xlarge)* no subconjunto de teste e validação do "*Conjunto de dados reduzido*".

Subconjunto	TP	TPNA	TN	FP	FN
Teste	1141	67	166	21	41
Validação	846	456	357	16	89

O modelo *YOLOv5 (Xlarge)* deteta 1208 moscas-brancas (TP+TPNA) moscas-brancas no subconjunto de teste do "*Conjunto de dados reduzido*" (Tabela I). Dessas moscas-brancas detetadas, 67 (TPNA) não foram detetadas pelos anotadores. Apenas 41 instâncias de moscas-brancas anotadas não são detetadas (FN) e 21 instâncias são detetadas erradamente como sendo moscas-brancas (FP).

No subconjunto de validação, a utilização do modelo de deteção de objetos melhora a deteção em 39,25% em relação aos técnicos (Tabela I), porque foram usadas imagens onde existem muitas instâncias não anotadas. Para além de detetar mais moscas-brancas do que os técnicos, o modelo *YOLOv5 (XLarge)* também deteta mais moscas-brancas na infância, ou seja, deteta mais cedo. As deteções erradas do modelo de deteção de objetos ocorrem sobretudo devido à deteção do fundo e da cauda de outros insetos como sendo mosca-branca (figura 63).

O técnico/anotador detetou as instâncias TP e FN como sendo moscas-brancas e o modelo de deteção de objetos as instâncias TP e TPNA (Figura 63). A utilização do modelo de deteção de objetos melhora a deteção de moscas-brancas em 2,20% no subconjunto de teste (Tabela I), porque contém imagens onde quase todas as instâncias de moscas-brancas estão anotadas.

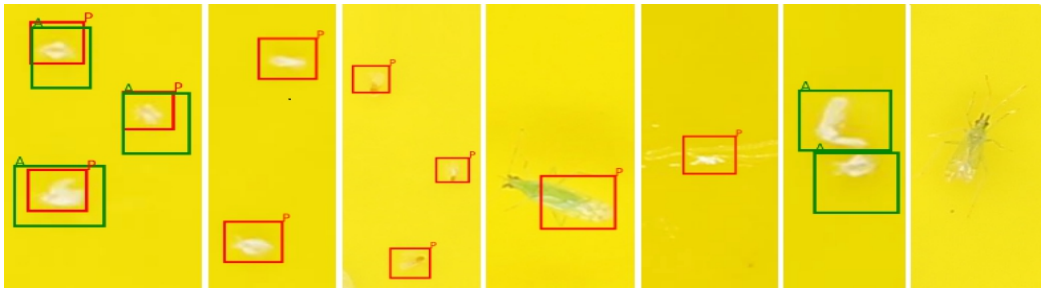


Figura 63 – Exemplos de deteções: a) WF anotadas e detetadas, b) WF adultas não anotadas mas detetadas, c) WF na infância não anotadas mas detetadas, d) Cauda do inseto Macrolophus detetado como WF, e) Fundo detetado como WF, f) WF anotadas mas não detetadas e g) inseto Macrolophus não detetado como WF.

5.5 Validação

Nesta fase, para validação das previsões dos modelos pelos técnicos, usou-se o subconjunto de validação do "*Conjunto de dados reduzido*" (composto por 27 imagens) e identificaram-se nessas imagens, as instâncias anotadas e as instâncias detetadas pelo modelo *YOLOv5 (XLarge)*. Com o objetivo de verificar se as previsões de instâncias não anotadas estão corretas (correspondem a moscas-brancas) submeteram-se estas imagens à validação dos técnicos.

5.5.1 Anotações

Após a confirmação dos técnicos de que o modelo de deteção de objetos estava a detetar corretamente instâncias de moscas-brancas não anotadas procedeu-se à anotação das instâncias não anotadas mais facilmente identificáveis. Essas instâncias foram principalmente as moscas-brancas na fase adulta, identificadas pela sua cor e instâncias da mosca-branca na infância, identificadas por estarem próximas de outras instâncias anotadas semelhantes (Figura 64).

Figura 64 – Anotação das WF na fase de pré-processamento. a) WF adulta posteriormente anotada, b) WF na infância posteriormente anotada, c) Possível WF adulta que se optou por manter não anotada e c) Possível WF na infância que se optou por manter não anotada.

Com as novas anotações o "*Conjunto de dados base*" (teste e treino) e o "*Conjunto de dados reduzido*" (teste, treino e validação) passaram de 4939 instâncias de moscas-brancas anotadas para 5863 (mais 18,71%) (Figura 65). O "*Conjunto de dados reduzido*" contém mais

388 instâncias anotadas no seu subconjunto de treino e mais 40 no seu subconjunto de teste (das possíveis 67 moscas-brancas detetadas e não anotadas (TPNA) anotaram-se 40). O subconjunto validado pelos técnicos contém mais 436 instâncias anotadas (das possíveis 456 moscas-brancas detetadas e não anotadas (TPNA) anotaram-se 436 instâncias).

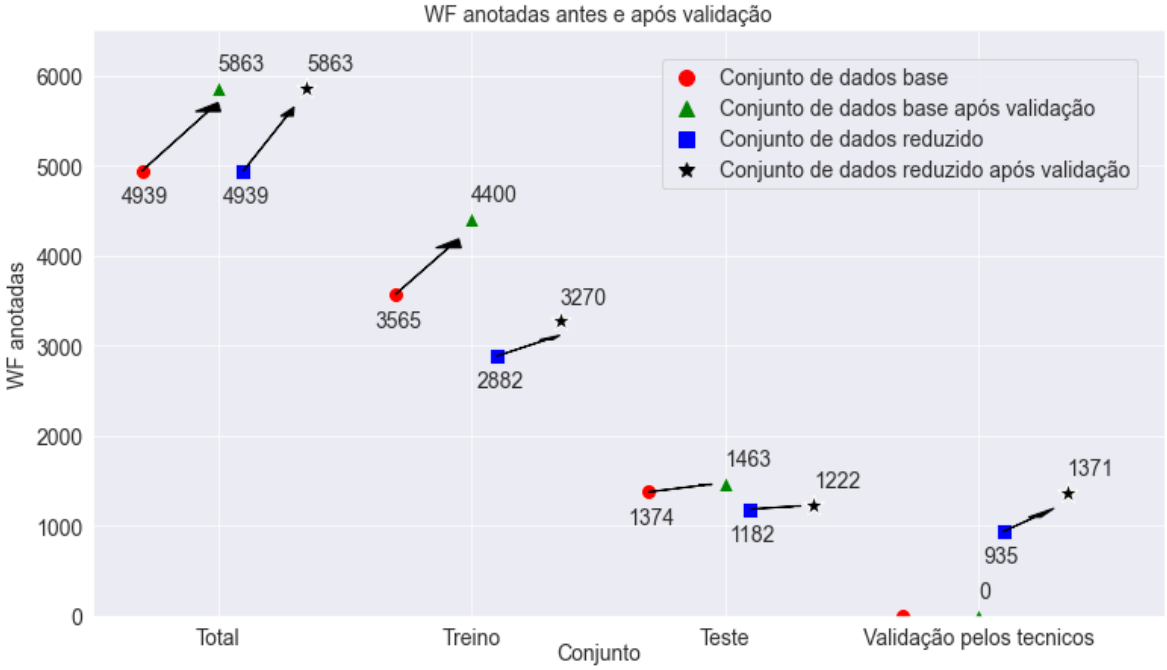


Figura 65 – Anotações, do "Conjunto de dados base" e do "Conjunto de dados reduzido", iniciais e após validação dos técnicos.

5.5.2 Resultados

O "*Conjunto de dados base após validação*" e o "*Conjunto de dados reduzido após validação*" foram utilizados para treinar e avaliar o modelo com melhor desempenho, *YOLOv5 (Xlarge)*, de forma a verificar se as novas anotações tinham reflexo na sua precisão (mAP).

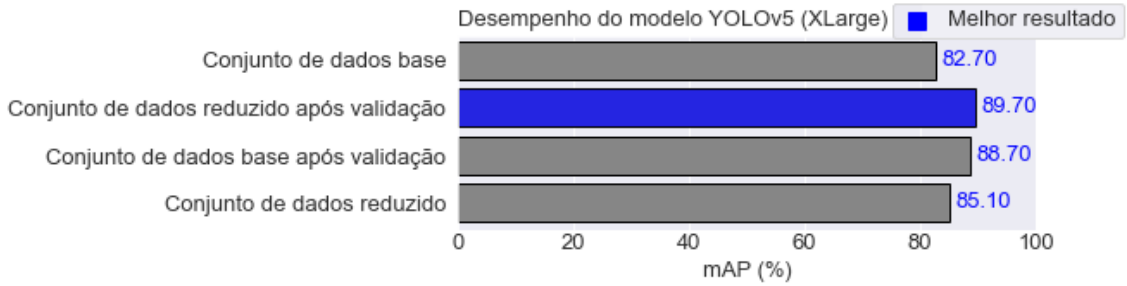


Figura 66 – Desempenho do modelo *YOLOv5 (Xlarge)* antes e após validação.

Com as novas anotações o desempenho do modelo *YOLOv5 (XLarge)* aumentou 7,25% no "*Conjunto de dados base após validação*" e 5.41% no "*Conjunto de dados reduzido após validação*" e a precisão (mAP) mais elevada passou de 85,10% para 89,70% (Figura 66).

O maior aumento de desempenho no "*Conjunto de dados base*" deve-se ao maior incremento do número de instâncias anotadas (835 novas instâncias anotadas no subconjunto de treino e 89 no subconjunto de teste) em relação ao "*Conjunto de dados reduzido*" (388 novas instâncias anotadas no subconjunto de treino e 40 no subconjunto de teste).

O modelo de deteção de objetos avalia as instâncias de moscas-brancas não anotadas (TPNA) como Falsos Positivos (FP), ou seja, moscas-brancas incorretamente detetadas.

Com a anotação dessas instâncias a precisão do modelo aumenta, porque passam a ser avaliadas como Verdadeiros Positivos (TP), ou seja, moscas-brancas corretamente detetadas.

5.6 Pós-processamento

Como se viu anteriormente o maior obstáculo à resolução do problema da deteção de moscas-brancas reside na falta de conjuntos de dados anotados. O "*Conjunto de dados base*" contém apenas 139 para treino e 35 para teste e o "*Conjunto de dados reduzido*" contém 117 imagens para treino e 30 para teste.

No capítulo 5.2 tentou-se aumentar o "*Conjunto de dados base*" utilizando transformações geométricas, como por exemplo a translação e a inversão das imagens originais. Utilizando este método a transformação é feita sobre a imagem original da armadilha (que contém moscas-brancas não anotadas) resultando em imagens novas com o mesmo número de instâncias não anotadas que as imagens originais.

Para gerar novas imagens de armadilhas amarelas onde todas as instâncias de moscas-brancas estejam anotadas, é necessário extrair as instâncias anotadas da armadilha original e de seguida posicionar essas instâncias (ou outras geradas artificialmente) em imagens de armadilhas amarelas que não contenham moscas-brancas por anotar.

5.6.1 Redes Adversárias Generativas

Nesta fase, com o propósito de criar novas imagens de armadilhas amarelas sem instâncias de moscas-brancas não anotadas, começou-se por extrair as instâncias de moscas-brancas das imagens originais das armadilhas, usando as coordenadas das suas anotações, e criaram-se 4940 imagens de moscas-brancas correspondentes ao número inicial de anotações disponíveis (Figura 67).

Figura 67 – Imagens originais das moscas-brancas.

De seguida utilizaram-se essas imagens originais das moscas-brancas para treinar uma rede adversária generativa (*Generative Adversarial Networks – GANs*) para gerar imagens artificias de moscas-brancas (com diferentes orientações, formas, etc.).

As *GANs* são compostas por 2 redes neuronais: a rede neuronal geradora e a rede neuronal discriminadora. A rede geradora foi utilizada para gerar as imagens das moscas-brancas artificias com base na distribuição dos píxeis das imagens originais (inverteu, desfocou, etc.). A rede discriminadora contém as imagens reais e as imagens geradas artificialmente (pela rede geradora) e tenta identificar quais são as imagens reais e artificiais. Este sistema teve como objetivo treinar a rede geradora a gerar imagens artificiais que a rede discriminadora não conseguisse destingir das imagens reais (Figura 68).

Figura 68 – Imagens artificiais das moscas-brancas obtidas utilizando *GANs*.

5.6.2 *Poisson Blending*

Por fim, usou-se o algoritmo *Poisson Blending,* para posicionar/mesclar as imagens artificiais das moscas-brancas sobre imagens de armadilhas amarelas vazias.

Para que as imagens das moscas-brancas artificiais se posicionem de forma natural sobre as imagens de armadilhas é necessário uniformizar a cor de fundo da armadilha com a cor de fundo das imagens das moscas-brancas artificiais (que contêm diversos tons de amarelo).

Utilizando o algoritmo *Poisson Blending*, definiu-se a região da imagem da mosca-branca que se pretendia copiar para a armadilha através de uma máscara a preto (1) e branco (0). Com os píxeis brancos a representarem a área da imagem da mosca-branca que se pretendia sobrepor à armadilha. Para cada píxel da imagem da mosca-branca, dentro da área que se pretendia sobrepor, limitou-se a uniformização desse píxel a uma escala que foi definida pelo valor máximo e mínimo dos seus 4 píxeis vizinhos. Caso o valor do píxel da armadilha amarela na mesma coordenada estivesse dentro dessa escala, escolhia-se o valor da escala mais próximo desse píxel para criar a sobreposição. Caso contrário escolhia-se o valor do píxel da imagem da mosca-branca para criar a sobreposição (Figura 69).

Este método permitiu criar uma imagem final mais uniforme ao misturar as cores semelhantes da imagem da mosca-branca e da armadilha e mantendo as cores originais da imagem da mosca branca onde as cores das duas imagens eram muito diferentes.

Figura 69 – Sobreposição de uma mosca-branca artificial numa imagem de armadilha vazia utilizando o algoritmo *Poisson Blending*: a) imagem a sobrepor, b) máscara c) imagem que recebe a sobreposição e d) imagem artificial final.

5.6.3 Resultados

O "*Conjunto de dados reduzido após validação*" foi aumentado com 100 imagens artificiais de armadilhas amarelas no subconjunto de treino e mantive o mesmo número de imagens no subconjunto de teste (Figura 70).

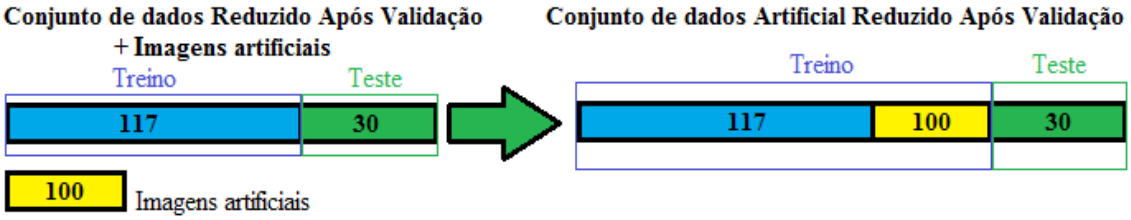


Figura 70 – Conjunto de dados obtidos com as imagens artificiais: "Conjunto de dados artificial reduzido após validação".

Para verificar se a adição das imagens artificias aumenta o desempenho do modelo *YOLOv5 (Xlarge)* utilizou-se o "*Conjunto de dados artificial reduzido após validação*" para treinar e avaliar esse modelo.

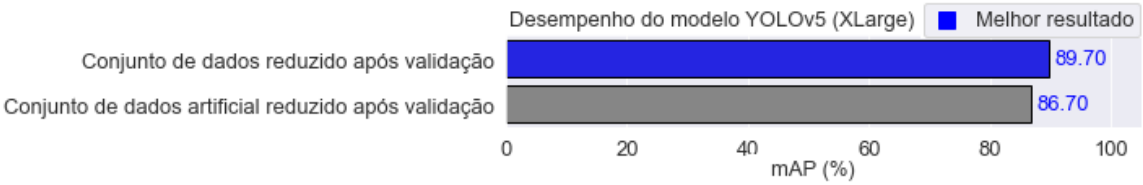


Figura 71 – Desempenho do modelo *YOLOv5 (Xlarge)* após adicionar imagens artificiais ao "Conjunto de dados reduzido após validação.

A adição de imagens artificiais piora o desempenho do modelo em -3.34% (Figura 71). Tal ocorre devido às dimensões das imagens usadas para treinar as *GANs*.

Para gerar as moscas-brancas artificias foi necessário transformar as dimensões das imagens originais das moscas-brancas (que têm as dimensões das respetivas anotações) para imagens quadrangulares de 64x64 píxeis.

Isso faz com que as moscas-brancas geradas artificialmente tenham dimensões diferentes das reais (são maiores e têm uma forma quadrada em vez de retangular) provocando a perda de precisão do modelo de deteção de objetos. O modelo está a aprender a detetar moscas-brancas de tamanho e forma diferente daquelas que existem no subconjunto de teste, que apenas contém moscas-brancas reais. Não se utilizam as imagens artificiais na próxima fase da implementação (protótipo) porque as imagens artificiais pioram o desempenho do modelo *YOLOv5 (Xlarge).*

5.7 Protótipo

Com o protótipo pretende-se integrar na mesma aplicação, a deteção das moscas-brancas e a sua evolução ao longo do tempo. Isso permite que os técnicos avaliem a evolução da população das moscas-brancas de imagem para imagem e ativem ou reajustem o controlo das moscas-brancas ao longo do tempo.

A componente *backend* do protótipo que contém o modelo de deteção de objetos, foi desenvolvida utilizando Python e a componente *frontend* utilizando JavaScript (biblioteca Chart.js), HTML e CSS. A ligação entre estas 2 componentes foi desenvolvida utilizando o *framework* Flask.

O protótipo permite ao utilizador optar entre tirar fotografias ("*Tirar fotografias*") ou usar imagens já existentes ("*Upload de imagens*") de armadilhas amarelas pegajosas.

Na opção *"Tirar fotografias"* o utilizador, através da componente *frontend*, acede à câmara do seu dispositivo e tira uma fotografia que é exibida na aplicação (e não é armazenada no seu dispositivo). Essa fotografia é transformada num *Binary Large Object* (BLOB) que, através do Flask, é enviado para a componente *backend*. Na componente *backend*, o BLOB é transformado numa imagem, onde é feita a deteção e contabilização de moscas-brancas. Por último essa imagem, com as deteções efetuadas e a respetiva contabilização, é enviada para a componente *frontend* onde se exibem as deteções e a contabilização do número de moscas-brancas.

Na opção "*Upload de imagens*" o utilizador, através da componente *frontend*, escolhe no seu dispositivo o conjunto de imagens nas quais pretende detetar e contabilizar a mosca-branca. De seguida essas imagens são enviadas para a componente *backend* onde é feita a deteção e contabilização. Por último as imagens, com as deteções efetuadas e a respetiva contabilização, são envidas para a componente *frontend* onde se exibem as deteções, a contabilização e a evolução do número de moscas-brancas.

5.7.1 Deployment

Nas fases anteriores melhorou-se o desempenho dos modelos de deteção de objetos para que eles pudessem ser utilizados de forma eficaz na deteção da mosca-branca. Para demonstrar essa eficácia, escolheu-se o modelo com melhor desempenho nas fases anteriores e este foi utilizado como base para o desenvolvimento de um protótipo.

Como não existiram requisitos para a implementação do protótipo (como memória ou tempo de deteção), escolheu-se o modelo *YOLO v5 (Xlarge)* que obteve a maior precisão e o maior número de moscas-brancas detetadas nas fases anteriores. O conjunto de dados escolhido foi aquele para o qual este modelo obteve o melhor desempenho, o "*Conjunto de dados reduzido após validação*". Para fazer o *deployment* treinou-se novamente o modelo *YOLOv5 (XLarge)* utilizando para treino todas as imagens do "*Conjunto de dados reduzido após validação*".

5.7.2 Diagrama de utilização

O protótipo é composto por dois caminhos. Pode-se escolher a opção *"Tirar fotografias"* ou a opção fazer o *"Upload de imagens"* (Figura 72).

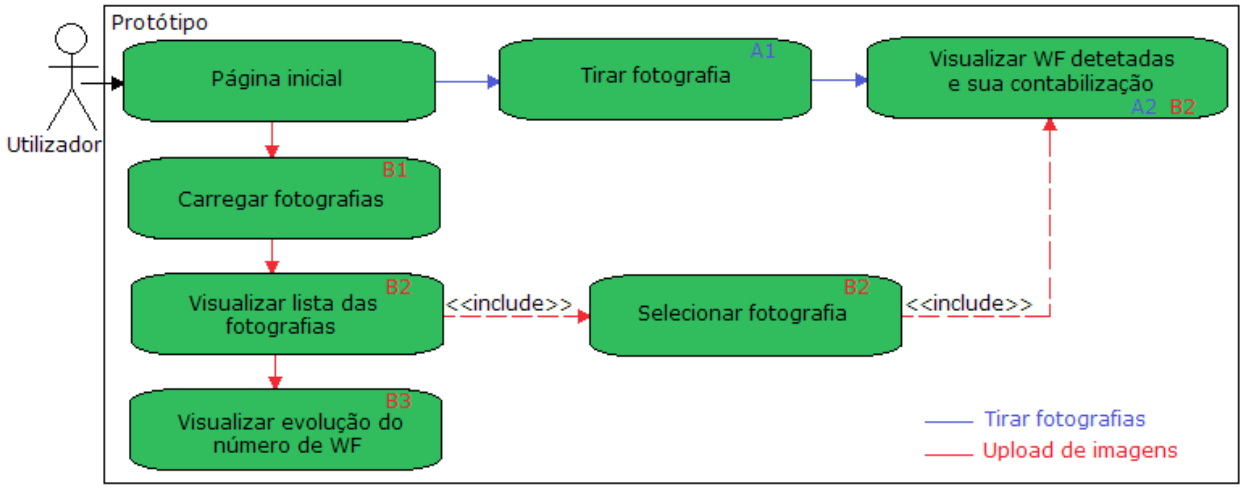


Figura 72 – Diagrama de utilização do protótipo.

"Tirar fotografias":

- **A1:** Tira-se uma nova fotografia através da câmara do dispositivo e faz-se a deteção das moscas-brancas utilizando o modelo *YOLOv5 (Xlarge)* (Figura 73);

Figura 73 – A1: Tirar fotografia.

- **A2:** De seguida é exibida a imagem da fotografia contendo as moscas-brancas detetadas e a correspondente contabilização dessas moscas-brancas (Figura 74).

Figura 74 – A2: Visualizar as WF detetadas.

"Upload de imagens":

- **B1:** Carregam-se as imagens das armadilhas e faz-se a deteção das moscas-brancas utilizando o modelo *YOLOv5 (Xlarge)* (Figura 75);

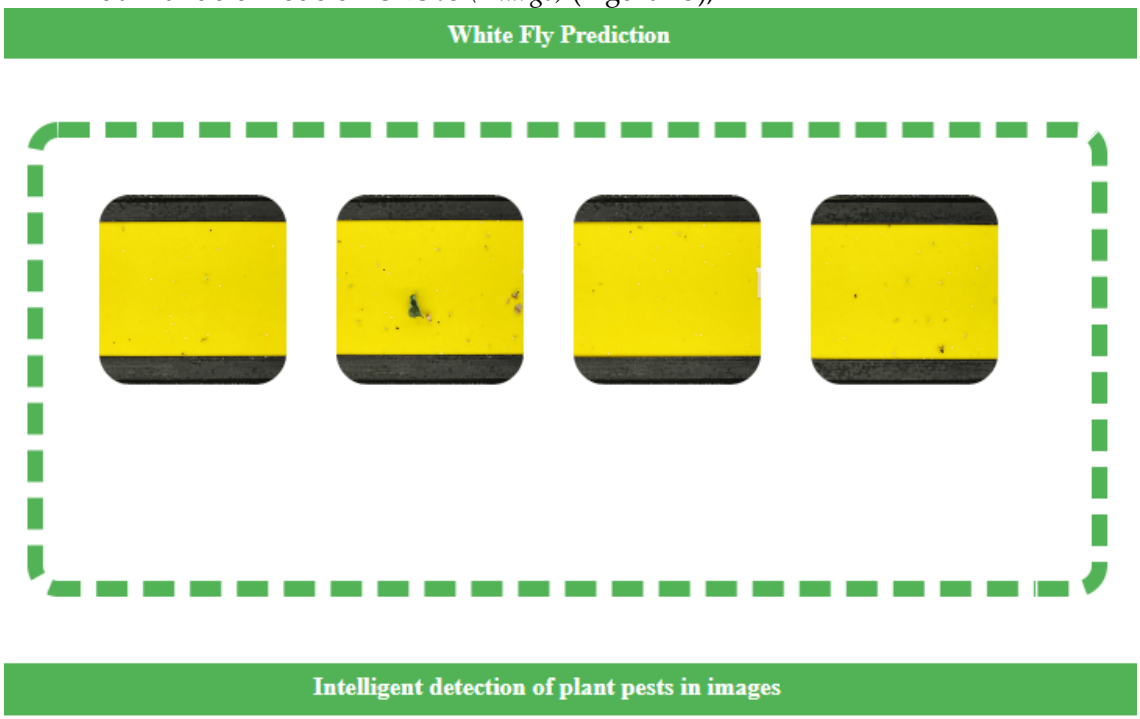


Figura 75 – B1: Upload das imagens.

- **B2:** De seguida é exibida uma lista das imagens carregadas e pode-se selecionar cada uma das imagens para visualizar as moscas-brancas detetadas e obter a sua contabilização (Figura 76);

Figura 76 – B2: Visualizar as WF detetadas.

- **B3:** Ainda se pode visualizar um gráfico com a evolução do número de moscas-brancas ao longo das imagens selecionadas (Figura 77).

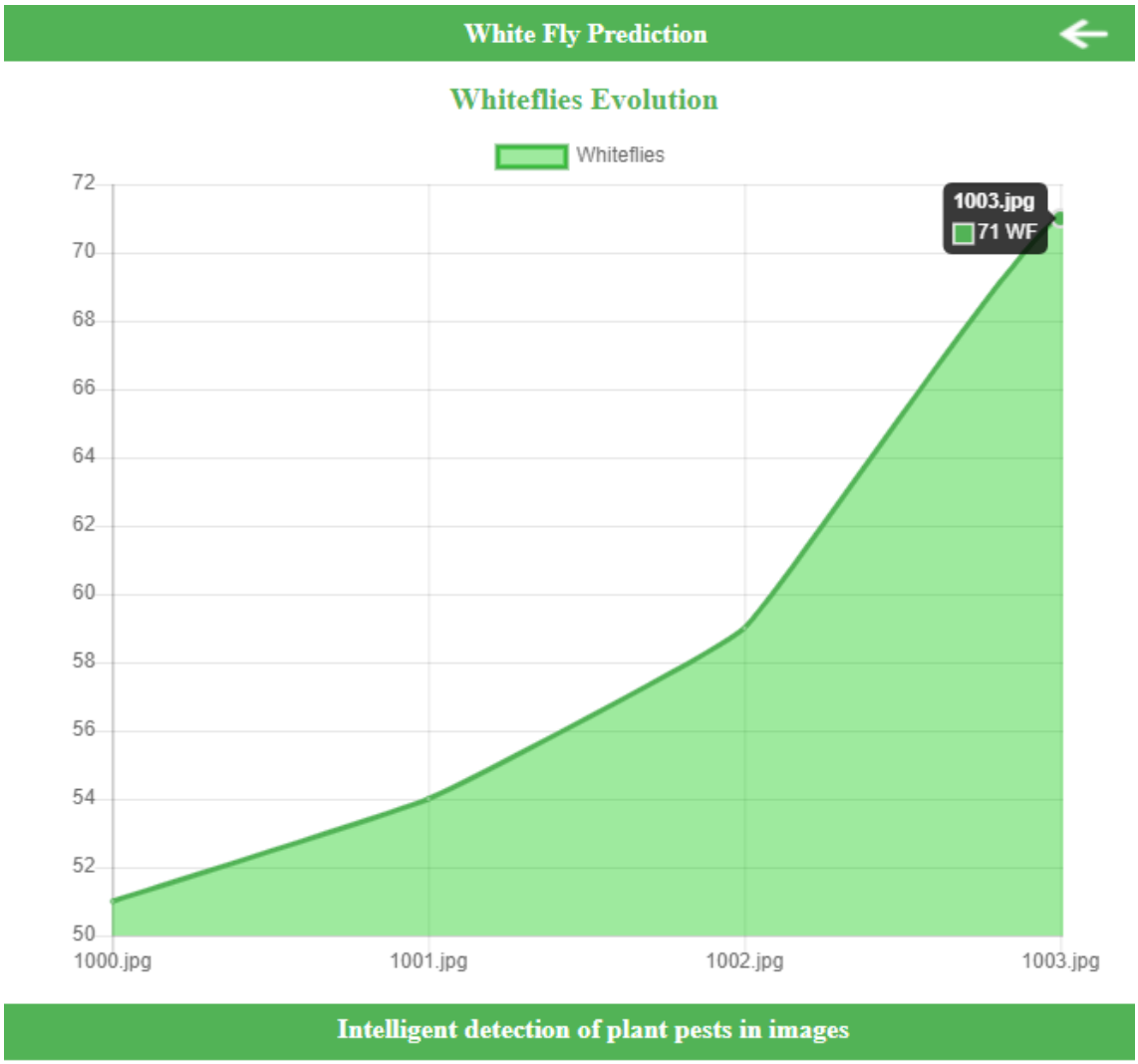


Figura 77 – B3: Evolução da mosca-branca.

Com a opção "T*irar fotografias*" é possível obter deteções em tempo real, mas possivelmente menos precisas devido à resolução das fotografias tiradas (1296x846). A opção "*Upload de imagens*" permite utilizar imagens obtidas através de dispositivos com câmaras de alta resolução (por exemplo 5184x3456). A visualização da evolução do número de moscas-brancas ao longo do tempo permite monitorizar a evolução do número de moscas-brancas e ativar um tipo de controlo (químico ou biológico), apenas quando se atinja um número definido de moscas-brancas na plantação.

Ao escolher-se o modelo *YOLOv5 (Xlarge)*, para implementar o protótipo, o tempo de deteção de cada imagem é, em média, de 0.13 segundos (utilizando o processador NVIDIA Tesla T4 GPU), o que pode tornar a utilização do protótipo demasiado lenta, caso se deseje detetar moscas-brancas num grande número de imagens. Nesse caso pode ser mais indicado utilizar o modelo *YOLOv5 (Small),* apesar da perda de precisão.

Para saber quais as características mais importantes para o protótipo (tempo de deteção vs precisão, evolução do número de moscas-brancas ao longo do tempo, etc.) é necessário obter a opinião de um técnico. Está prevista uma visita a uma plantação de tomate onde será possível obter a opinião de um técnico sobre o funcionamento do protótipo e recolher requisitos que poderão ser incluídos numa futura versão. A prova de conceito será feita durante essa visita, utilizando o protótipo em ambiente real para fazer deteções de moscas-brancas em armadilhas pegajosas amarelas.

6 Conclusão e trabalho futuro

Durante o primeiro semestre foram alcançadas todas as metas propostas para essa fase, como o estudo do estado da arte, os conjuntos de dados disponíveis, estudo dos modelos existentes de deteção de objetos, definir a arquitetura desses modelos e começar a sua implementação.

Durante o segundo semestre também foram alcançadas todas as metas propostas, como a continuação do desenvolvimento dos modelos de deteção de objetos, fazer testes em vários conjuntos de dados, avaliar o desempenho dos modelos de deteção de objetos nesses conjuntos de dados e afinar os seus desempenhos. Ainda se fizeram testes em conjuntos de dados aumentados com uma rede adversária generativa e desenvolveu-se um protótipo para demonstrar como a visão por computador pode ser aplicada à deteção de pragas.

Durante a dissertação implementaram-se os dois algoritmos de deteção de objetos mais recentes, a *YOLOv5* e *Scaled-YOLOv4* e outro modelo com uma arquitetura diferente, a *R-CNN* que poderia ser mais adequada ao problema a resolver.

Após encontrar os melhores parâmetros para cada um dos modelos de deteção de objetos testados, a melhoria de precisão dos modelos foi obtida através do pré-processamento do conjunto de dados original. Esse pré-processamento foi essencial para melhorar os resultados devido à falta de anotações em algumas imagens e à dificuldade em anotar essas imagens (as moscas-brancas não são facilmente identificáveis nas diversas fases).

Os modelos de deteção de objetos têm sido utilizados com sucesso em áreas onde é necessário identificar diferentes tipos de objetos na mesma imagem ou *frame* como por exemplo sistemas de condução autónomos ou sistemas de segurança. No entanto, o combate a pragas agrícolas utilizando estes modelos até ao momento tem muito poucos artigos publicados. Um dos motivos para tal pode ser a falta de conjuntos de dados anotados disponíveis. A falta de anotações pode estar relacionada com o método usado pelos técnicos para contabilizar moscas-brancas. Os técnicos apenas contabilizam as moscas presentes numa área da armadilha e extrapolam essa quantidade para obter uma estimativa. Embora essa extrapolação feita pelos técnicos permita que contabilizem as armadilhas mais rapidamente também retira precisão à contagem de moscas-brancas. Com os modelos de deteção de objetos utilizados nesta dissertação consegue-se obter rapidamente a contagem do número de moscas-brancas por armadilha sem comprometer a precisão da deteção.

Em todos os testes feitos os modelos *YOLO* obtiveram melhores resultados do que a *R-CNN*. Os modelos *YOLO* para além de terem melhor precisão, também fazem a deteção mais rápido e ocupam menos memória.

Nos testes efetuados utilizando a *YOLOv5 (XLarge)*, obteve-se um modelo que melhora a deteção da mosca-branca no conjunto de dados utilizado, entre 2,71% (subconjunto de treino) e 39,89% (subconjunto de validação) em relação aos anotadores/técnicos do conjunto de dados original. Este modelo permite que a deteção da mosca branca por armadilha seja feita em 0.13 segundo e ocupa 167MB de memória.

Caso seja necessário um modelo com um menor tempo de deteção ou que consuma menos memória pode-se optar por utilizar a *YOLOv5 (Small)* ou a *Scaled-YOLOv5* que obtiveram, respetivamente, uma precisão (mAP) de 76.15% e 81.20%. O modelo *R-CNN* é o menos preciso, com uma precisão (mAP) de 76,15% e o que ocupa mais memória e tem maior tempo de deteção. Devido à falta de anotações do conjunto de dados, para além do pré-processamento feito, foi importante obter a validação das deteções feitas pelos modelos de deteção de objetos por parte dos técnicos. A sua validação permitiu confirmar que os modelos, ao detetarem instâncias não anotadas, estavam a ter um desempenho superior aos anotadores/técnicos e utilizar essas deteções para anotar o conjunto de dados.

Após anotar as instâncias de moscas-brancas não anotadas, mas detetadas pelos modelos de deteção de objetos, obteve-se um resultado final (mAP) de 89,70% com o modelo *YOLOv5 (Xlarge)*.

O modelo *YOLOv5 (Xlarge)* foi usado no desenvolvimento de um protótipo que permite de forma fácil carregar fotografias de armadilhas ou aceder à câmara do dispositivo para detetar moscas-brancas de forma rápida e precisa. Para fazer o controlo da mosca-branca durante o tempo também é possível avaliar a evolução do número de moscas-brancas em cada imagem.

A prova de conceito do protótipo será feita durante uma visita a uma plantação de tomate. Durante essa visita será possível testar o protótipo fazendo deteções de moscas-brancas em ambiente real, obter a opinião de um técnico sobre o seu funcionamento e recolher as características mais relevantes a serem incluídas numa futura versão. Com esta solução proposta, utilizando modelos de deteção de objetos para detetar pragas em armadilhas, espera-se contribuir com uma solução que permita identificar automaticamente a praga mosca-branca em armadilhas pegajosas amarelas de forma rápida, precisa e fácil.

O maior obstáculo à resolução do problema da deteção de moscas-brancas reside na falta de conjuntos de dados anotados. A utilização de redes adversárias generativas, para gerar imagens artificiais de insetos pode ajudar a ultrapassar a falta de anotações e criar conjuntos de dados melhores. No entanto a geração de imagens artificiais terá de ser aprimorada, porque ainda não apresentam um aspeto suficientemente natural (próximo dos insetos reais) que as permita utilizar para melhorar o desempenho do modelo de deteção de pragas. No futuro, para melhorar o desempenho das *GANs* pode-se definir como dimensão única das imagens das moscas-brancas a maior largura e altura do conjunto das anotações ou como todas as anotações conseguem ser bem definidas por 9 dimensões (3 fases da mosca-branca e 3 posições sobre a armadilha) e pode-se utilizar isoladamente a *GAN* para essas dimensões.

No futuro, os modelos de deteção de objetos podem ser usados para identificar outros tipos de pragas que afetam plantações agrícolas ou em outros contextos (por exemplo em folhas em vez de armadilhas). Como não existem conjuntos de dados disponíveis, podem ser utilizadas as *GANs* (após melhorar o seu desempenho) para gerar novas imagens desses insetos que posteriormente podem ser sobrepostos/colocados em imagens de armadilhas vazias ou por exemplo em folhas.

Durante esta dissertação os modelos de deteção de objetos foram usados para detetar a mosca branca e fazer um controlo biológico desta praga reagindo em tempo útil aos danos que poderá provocar. No futuro pode-se incluir um módulo de previsão de pragas utilizando o número de moscas-brancas detetadas em cada imagem e em cada instante para construir uma série temporal. Esse módulo permitiria que fosse possível complementar o controlo biológico com técnicas de prevenção de forma mais eficiente.

Referências

Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Acedido em 2020, em: https://github.com/matterport/Mask_RCNN.

Abdulla, W. (2020). Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow. Acedido em 2020 em: https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46.

Abraham, A. (2011). Artificial Neural Networks. *Intelligent Systems Reference Library, vol 17. Springer.*

Adelson, E., Anderson, C., Bergen, J., Burt, P. e Ogden, J. (1983). Pyramid methods in image processing. *RCA Engineer.*

AgroCares. AgroCares Scoutbox. Acedido em 2021 em: https://www.agrocares.com/products /scoutbox/.

Al-Hiary, H., Bani-Ahmad, S., Reyalat, M., Braik, M. e ALRahamneh, Z. (2011). Fast and Accurate Detection and Classification of Plant Diseases. *International Journal of Computer Applications.*

Al-Saffar, A., Tao, H. e Talab, M. (2017). Review of Deep Convolution Neural Network in Image Classification. *International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications.*

Arivazhagan, S., Shebiah, R., Ananthi, S. e Varthini, S. (2013). Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. *Agric Eng Int: CIGR Journal.*

Arsenovic, M., Karanovic, M., Sladojevic, S., Anderla, A. e Stefanovic, D. (2019). Solving Current Limitations of Deep Learning Based Approaches for Plant Disease Detection. *Symmetry.*

Bai, T., Pang, Y., Wang, J., Han, K., Luo, J., Wang, H., Lin, J., Wu, J. e Zhang, H. (2020). An Optimized Faster R-CNN Method Based on DRNet and RoI Align for Building Detection in Remote Sensing Images. *Remote Sensing, vol. 12.*

Bakkay, M., Chambon, S., Rashwan, H., Lubat, C. e Barsotti, S. (2018a). Automatic detection of individual and touching moths from trap images by combining countour-based and region-based segmentation. *IET Computer Vision.*

Bakkay, M., Chambon, S., Rashwan, H., Lubat, C. e Barsotti, S. (2018b). Support Vector Machine (SVM) Recognition Approach adapted to Individual and Touching Moths Counting in Trap Images. *ArXiv: 1809.06663.*

Barbedo, J. (2020). Detecting and Classifying Pests in Crops Using Proximal Images and Machine Learning: A Review. *Atificial Intelligence, vol 1, pp. 312-328*

Bochkovskiy, A., Wang, C. e Liao, H. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv: 2004.10934.*

Borude, B., Bhalkare, S. e Undirwade, D. (2017). Seminar on Whitefly as a vector. Acedido em 2020, em: https://www.slideshare.net/bharatsborude/whitefly-as-vector.

Brownlee, J. (2020). Deep Learning for Computer Vision. Acedido em 2020, em: https://machinelearningmastery.com/how-to-train-an-object-detection-model-with-keras/.

Cavaioni, M. (2018). DeepLearning series: Convolutional Neural Networks. Acedido em 2020, em: https://medium.com/machine-learning-bites/deeplearning-seriesconvolutional-neural-networks-a9c2f2ee1524.

Chaudhary, P., Chaudhari, A., Cheeran, A. e Godara, S. (2012). Color Transform Based Approach for Disease Spot Detection on Plant Leaf. *International Journal of Computer Science and Telecommunications.*

Chen, Z., Zhang, T. e Ouyang, C. (2018). End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. *Remote Sensing, vol. 10.*

Costa, J., Silva, C. e Ribeiro, B. (2019). Hierarchical Deep Learning Approach for Plant Disease Detection. *Iberian Conference on Pattern Recognition and Image Analysis.*

Creswell, A., White, T., Dumoulin, V., Arulkumaran K., Sengupta, B. e Bharath, A. (2018). Generative Adversarial Networks: An Overview. *ArXiv: 1710.07035.*

Deng, L., Hu, H., Shi, P., Wang, W. e Kong, X. (2020). Region-Based CNN Method with Deformable Modules for Visually Classifying Concrete Cracks. *Applied sciences, vol. 10.*

Ding, W. e Taylor, G. (2016). Automatic moth detection from trap images for pest management. *Computers and Electronics in Agriculture, vol. 123.*

Do, Q., Lewis, M., Madhuranthakam, A. e Xi, Y. (2019). Texture analysis of magnetic resonance images of the human placenta throughout gestation: A feasibility study. *PLoS One.*

Esgario, J., Krohling, R. e Ventura, J. (2019). Deep Learning for Classification and Severity Estimation of Coffee Leaf Biotic Stress. *ArXiv: 1907.11561.*

Feng. J e Lu, S. (2019). Performance Analysis of Various Activation Functions in Artificial Neural Networks. *Journal of Physics: Conference Series.*

Gavhale, K. e Gawande, U. (2014). An Overview of the Research on Plant Leaves Disease detection using Image Processing Techniques. *IOSR Journal of Computer Engineering*.

Ghiassia, M., Saidaneb, H. e Zimbra, D. (2005). A dynamic artificial neural network model for forecasting time series events. *International Journal of Forecasting.*

Girshick, R., Donahue, J., Darrell T. e Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Computer Vision Foundation.*

Girshick, R. (2015). Fast R-CNN. *IEEE, International Conference on Computer Vision.*

Goswami, S. (2018). A deeper look at how Faster-RCNN works. Acedido em 2020, em: https://whatdhack.medium.com/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd.

Greenvass. technology & innovation in agriculture. Acedido em 2020, em: https://www.greenvass.es/en/traps/.

Hassoun, M. (1995). Fundamentals of Artificial Neural Networks. *MIT Press.*

He, K., Zhang, X., Ren, S. e Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *ECCV, European Conference on Computer Vision.*

He, K., Gkioxari, G., Dollar, P. e Girshick, R. (2017). Mask R-CNN. *ICCV, International Conference on Computer Vision.*

Hoffer, E., Hubara, I. e Soudry, D. (2018). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *NIPS, Neural Information Processing Systems.*

Hong, S., Kim, S., Kim, E., Lee, C., Lee, J., Lee, D., Bang, J. e Kim, G. (2020). Moth Detection from Pheromone Trap Images Using Deep Learning Object Detectors. *Agriculture, vol. 10.*

Hughes, D. e Salathé, M. (2020). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *ArXiv: 1511.08060.*

Hui, J. (2018). mAP (mean Average Precision) for Object Detection. Acedido em 2020, em: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detecti on- 45c121a31173.

Jocher, G. (2020a). Hyperparameter Evolution. Acedido em 2020, em: https://github.com/ ultralytics/yolov5/issues/607.

Jocher, G. (2020b). Yolov5. Acedido em 2020, em: https://github.com/ultralytics/ yolov5.

Khan, M., Lali, M., Sharif, M., Javed, K., Aurangzeb, K., Haider, S., Altamrah, A. e Akram, T. (2019). An Optimized Method for Segmentation and Classification of Apple Diseases Based on Strong Correlation and Genetic Algorithm Based Feature Selection. *IEEE Access, vol. 7.*

Khirade, S. e Patil, B. (2015). Plant Disease Detection Using Image Processing. *International Conference on Computing Communication Control and Automation* .

Kingma, D. e Ba, J. (2017). Adam: A Method for Stochastic Optimization. *ICLR, International Conference on Learning Representations.*

Kin-Yiu, Wong (2020). Scaled-YOLOv4. Acedido em 2021, em: https://github.com/Wong KinYiu/ScaledYOLOv4.

Kutty, S., Abdullah N., Hashim, H., Rahim, A., Kusim, A. e Sulinda, A. (2013). Classification of Watermelon Leaf Diseases Using Neural Network Analysis. *Business Engineering and Industrial Applications Colloquium*.

Li, E. (2019). Dive Really Deep into YOLO v3: A Beginner's Guide. Acedido em 2020, em: https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e.

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll´ar, P. e Zitnick, L. (2013). Microsoft COCO: Common Objects in Context. *ECCV, European Conference on Computer Vision.*

Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B. e Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *CVPR, Computer Vision and Pattern Recognition Conference.*

Liu, S., Qi, L., Qin, H., Shi, J. e Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *CVPR, Computer Vision and Pattern Recognition Conference.*

Martins, V., Freitas, L., Aguiar, M., Brisolara, L. e Ferreira, P. (2019). Deep Learning applied to the Identification of Fruit Fly in Intelligent Traps. *IX Brazilian Symposium on Computing Systems Engineering.*

Michaels, R. (2017) Automating sticky trap analysis. *BSc Research report, Biology, Leiden University.*

Mohanty, S., Hughes, D. e Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science.*

Mwebaze, E., Gebru, T., Frome, A., Nsumba, S. e Tusubira, J. (2019). iCassava 2019 Fine-Grained Visual Categorization Challenge. *ArXiv: 1908.02900.*

Nazri, A. (2018). *DATASET BPH and BENIGN.* Acedido em 2020 em: https://doi.org/10.6084/m9.figshare.7015658.v1.

Nguyen, T. (2018). Pest detection on Traps using Deep Convolutional Neural Networks. *ICSA, 8th International Conference on Software and Computer Applications.*

Nieuwenhuizen, A., Hemming J. e Suh H. (2018). Detection and classification of insects on stick-traps in a tomato crop using Faster R-CNN. *The Netherlands Conference on Computer Vision.*

Oliveira, M.R.V., T.J. Henneberry, e P. Anderson, (2001). History, current status, and collaborative research projects for Bemisia tabaci. *Crop Protection.*

Ortiz, O., (2007). Tropical Whitefly Project (TWFP) Progress Report, Impact Evaluation. *Centro Internacional de Agricultura Tropical (CIAT).*

Palumbo, J. e Ellsworth, P. (2002). Concerns for Whitefly Management in Multi-crop Communities . *College of Agriculture, The University of Arizona.*

Pande, S. (2019). Mask R-CNN Unmasked. Acedido em 2020, em: https://medium.com/@fractaldle/mask-r-cnn-unmasked-c029aa2f1296.

Patel, A. e Joshi, B. (2017). A Survey on the Plant Leaf Disease Detection Techniques. *International Journal of Advanced Research in Computer and Communication Engineering*.

Pedamonti, D. (2018). Comparison of non-linear activation functions for deep neural networks on MNIST classification task. *ArXiv: 1804.02763.*

Pérez, P., Gangnet, M. e Black., A. (2003). Poisson Image Editing. Poisson Image Editing. *ACM Transactions and Graphics.*

Prajapati, H., Shah, J. e Dabhi, V. (2017). Detection and classification of rice plant diseases. *Intelligent Decision Technologies.*

Qing, Y., Jun, L., Qing-jie, L., Guang-qiang, D., Bao-jun, Y., Hong-ming, C. e Jian, T. (2012). An Insect Imaging System to Automate Rice Light-Trap Pest Identification. *Journal of Integrative Agriculture.*

Ramakishnan, M. e Sahaya, A. (2015). Groundnut leaf disease detection and classification by using back probagation algorithm. *ICCSP, International Conference on Communication and Signal Processing.*

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J. e Hughes, D. (2017). Deep Learning for Image-Based Cassava Disease Detection. *Frontiers Plant Sci., vol. 8.*

Redmon, J., Divvala, S., Girshick, R. e Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *Computer Vision Foundation.*

Redmon, J. e Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *CVPR, Computer Vision and Pattern Recognition Conference.*

Redmon, J. e Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv: 1804.02767.*

Ren, S., He, K., Girshick, R. e Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NIPS, Neural Information Processing Systems Conference.*

Roosjen, P., Kellenberger, B., Kooistra, M., Green, D. e Fahrentrapp, J. (2020). Deep learning for automated detection of Drosophila suzukii: potential for UAV-based monitoring. *Pest Management Science.*

Ruder, S. (2015). An overview of gradient descent optimization algorithms. *ArXiv: 1609.04747.*

Sannakki, S., Rajpurohit, V., Nargund, V. e Kulkarni P. (2013). Diagnosis and classification of grape leaf diseases using neural networks. *Computing, communications and Networking Technologies*.

Shah, J., Prajapati, H. e Dabhi, V. (2015). A Survey on Detection and Classification of Rice Plant Diseases. *IEEE International Conference.*

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S. e Batra, N. (2020). PlantDoc: A Dataset for Visual Plant Disease Detection. *CoDS COMAD 2020.*

Smith, L. (2018a). A Disciplined approach to neural network hyper-parameters: part 1 – learning rate, batch size, momentum and weight decay. *US Naval Research Laboratory Technical Report 5510-026*.

Smith, S., Kindermans, P., Ying, C. e Le, Q. (2018b). Don`t decay the learning rate, increase the batch size. *ArXiv: 1711.00489.*

Solawetz, J. (2020). YOLOv5 New Version - Improvements And Evaluation. Acedido em 2020, em: https://blog.roboflow.com/yolov5-improvements-and-evaluation.

Tan, M., Pang, R. e Le, Q. (2020). EfficientDet: Scalable and Efficient Object Detection. *IEEE, International Conference on Computer Vision.*

Thapa, R., Snavely, N., Belongie, S. e Khan, A. (2020). The Plant Pathology 2020 challenge dataset to classify foliar disease of apples. *ArXiv: 2004.11958.*

Torok, L. (2015). Método Otsu. *Instituto de Computação – Universidade Federal Fluminense.*

Traorea, B., Kamsu-Foguema, B. e Tangarab, F. (2018). Deep convolution neural network for image recognition. *Ecological Informatics.*

Trapview. Automated pest monitoring. Acedido em 2020, em: http://www. Trapview.co m /v2/en/.

Tzutalin. LabelImg. (2015). Acedido em 2020, em: https://github.com/tzutalin/ labelImg.

Uijlings, J., Sande, K., Gevers, T. e Smeulders, A. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision.*

Wang, C., Liao, H., Yeh, I., Wu, Y., Chen, P. e Hsieh, J. (2019). CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN. *IEEE, International Conference on Computer Vision.*

Wang, C., Bochkovskiy, A., Liao, H. (2020). Scaled-YOLOv4: Scaling Cross Stage Partial Network. *ArXiv: 2011.08036.*

Wiesner-Hanks, T., Kaczmar, N., Stewart, E. e DeChant, C. (2018). Image set for deep learning: Field images of maize annotated with disease symptoms. *BMC Res Notes, vol. 11.*

Wilson, A., Roelofs, R., Stern, M., Srebro, N. e Recht, B. (2018). The Marginal Value of Adaptive Gradient Methods in Machine Learning. *NIPS, Neural Information Processing Systems.*

Yalcin, H. (2015). Vision based Automatic Inspection of Insects in Pheromone Traps. *Proceedings of the International Conference on Agro-Geoinformatics.*

Yusuken, Y. Object Detction #1: NMS. (2018). Acedido em 2021, em: https:// medium.com/ @yusuken/object-detction-1-nms-ed00d16fdcf9.

Zhong, Y., Gao, J., Lei, Q. e Zhou, Y. (2018). A Vision-Based Counting and Recognition System for Flying Insects in Intelligent Agriculture. *Sensors.*

Zhou, P., Feng, J., Ma, C., Xiong, C., HOI, S. e Weinan, E. (2020). Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning. *NIPS, Neural Information Processing Systems.*