



ESCOLA NAVAL

ta sainte & bief faire



André Couto Astorga Batista Pinto

Deteção de alvos terrestres a partir de um UAV,
em apoio a Operações Anfíbias

Dissertação para obtenção do Grau de Mestre em
Ciências Militares Navais, na especialidade de Fuzileiro

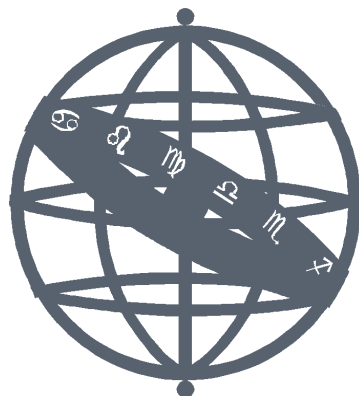


Alfeite
2022



ESCOLA NAVAL

talant de bi-faire



André Couto Astorga Batista Pinto

*Deteção de alvos terrestres a partir de um UAV, em apoio
a Operações Anfíbias*

Dissertação para obtenção do Grau de Mestre em
Ciências Militares Navais, na especialidade de Fuzileiro

Orientação de: Doutor Bruno Duarte Damas

Co-orientação de: Doutor Nuno Alexandre Antunes Martins Pessanha Santos
Mestre Vitor Manuel Borges Rodrigues

O Aluno Mestrando,

O Orientador,

Assinado por: **ANDRÉ COUTO ASTORGA BATISTA
PINTO**
Num. de Identificação: 30333256
Data: 2022.09.20 10:03:08+01'00'



André Pinto

Bruno Damas

Alfeite
2022

War is the ultimate realization of modern technology.

Don DeLillo

Agradecimentos

No decorrer da realização deste trabalho de investigação, foram várias as pessoas e instituições que se disponibilizaram a apoiar-me, ensinar-me e incentivar do início ao final deste projeto. Deixo aqui, desde já, os meus maiores agradecimentos.

Começo por agradecer ao CMOR FZ Matos Lopes, pela sua disponibilidade e pela sua amizade, contribuindo imensamente para a extensa recolha de dados, todos eles importantíssimos para o desempenho do meu trabalho.

Ao 1TEN FZ Ribeiro Ferreira, um apreço pela disponibilidade e ao exemplo que ofereceu, que mesmo sem obrigação ou fora do seu horário de trabalho, esteve sempre pronto a ajudar-me.

Ao CFR M Pereira de Castro, que sem a sua autorização, não seria possível realizar as demais gravações no Exercício Tróia, sem estas o desenvolvimento e conclusão desta dissertação seriam muito diferentes.

Um agradecimento exaustivo ao meu orientador PROF Bruno Damas e coorientadores, 1TEN EN-AEL Pessanha Santos e 1TEN FZ Borges Rodrigues, pela acessibilidade, preocupação, compreensão, partilha de experiências, transferência de conhecimentos, sugestões e disponibilidade sempre que algum obstáculo ou dúvida surgisse ao longo do trabalho.

Ao curso CMG Fernando Amor Monteiro de Barros, pelas experiências partilhadas e entreaajuda no decorrer destes longos cinco anos. De salientar os cinco camaradas na minha especialidade, que em todos os momentos estiveram presentes, ASPOF FZ Silva Duarte, APOF FZ Jaime Ricardo, ASPOF FZ Silvestre Rodrigues e ASPOF FZ Nobre Serôdio.

À Escola Naval, Escola de Fuzileiros e Corpo de Fuzileiros, pelo tempo, acesso, experiências e aprendizagens proporcionadas.

Por último, mas não menos importante, um extenso agradecimento à minha família e amigos, por tudo o que estas palavras significam.

A todos os referidos e aos que não foram mas estiveram presentes, a minha maior gratidão. Este apoio e estímulo a nível profissional e pessoal foram incomparáveis.

Resumo

Nas últimas décadas, os confrontos militares têm dependido, a uma larga escala, de novas e avançadas tecnologias, conduzindo a uma mudança de paradigma ao que respeita as operações militares. Esta mudança deve-se, essencialmente, à tentativa de mitigar erros que se verificaram em operações passadas, otimizar recursos e salvaguardar vidas humanas, garantindo ao mesmo tempo o sucesso e estado final desejado de qualquer operação. É neste espectro que são introduzidos os *Unmanned Aerial Vehicles* (UAVs) e a significativa vantagem que os mesmos possuem em prol de qualquer operação, desde a redução do custo de vidas humanas, ao apoio direto e indireto no campo de batalha, ao aumento do conhecimento situacional do terreno e do inimigo. Neste trabalho, é apresentado como, a partir de uma câmara integrada de um UAV, é possível fazer a deteção automática de alvos terrestres (militares) e a sua diferenciação com civis, em tempo real, usando uma rede neuronal de deteção de objetos. Partindo da aquisição de uma base de dados, grande e variada, capturada por drones comerciais e imagens extraídas da Internet. Após isto, foi necessário fazer a escolha da rede neuronal para este fim, que recaiu sobre o YOLOv4-tiny, sendo uma rede rápida na deteção de objetos em tempo real e com bons resultados de precisão. De seguida, foi necessário realizar a anotação de todos os militares em cada imagem, utilizando um programa gratuito de anotação de imagens (DarkLabel 2.4). Após isto, treinámos, validámos e testámos a rede, aferindo quais os melhores resultados e pesos a serem utilizados na deteção de militares. Por fim, olhámos para as limitações e trabalhos futuros a serem desenvolvidos na continuidade deste trabalho.

Palavras-chave: Operações Militares, UAVs, Deteção de Objetos, Redes Neurais, YOLO

Abstract

In recent decades, military clashes have depended on a wide range of new and advanced technologies, leading to a paradigm shift in military operations. This change is mainly due to the attempt to mitigate errors that have occurred in past operations, optimize resources and safeguard human lives, while ensuring the success and desired end state of any operation. It is in this spectrum that Unmanned Aerial Vehicles (UAVs) are introduced and their significant advantage in favor of any operation, from the reduction of the cost of human lives, to direct and indirect support on the battlefield, to the increase of situational awareness of the terrain and the enemy. In this paper, it is presented how, from an UAV camera, it is possible to make the automatic detection of ground targets (military) and its differentiation with civilians, in real time, using a neural network of object detection. Starting with the acquisition of a database, large and varied, captured by commercial drones and images extracted from the Internet. After this, it was necessary to choose the neural network, which fell on YOLOv4-tiny, being a fast network in detecting objects in real time and with good accuracy results. Next, it was necessary to perform the annotation of all the military objects in each image, using a free image annotation program (DarkLabel 2.4). After this, we trained, validated and tested the network, to know which results and weights were best to use for military detection. Finally, we looked at the limitations and future work to be done in continuing this work.

Keywords: Military Operations, UAVs, Object Detection, Neural Networks, YOLO

Índice

1	Introdução	1
2	Revisão da Literatura e Enquadramento Conceptual	5
2.1	UAVs em Apoio a Operações Militares	5
2.1.1	UAVs nos Fuzileiros	6
2.2	Inteligência Artificial (IA)	9
2.2.1	Visão Computacional	10
2.2.2	<i>Machine Learning</i>	10
2.2.3	<i>Deep Learning</i>	12
2.3	Convolutional neural network (CNN)	14
2.3.1	Inícios de estudo e criação da CNN	14
2.3.2	O crescimento das CNNs	20
2.3.3	YOLO	23
2.3.4	YOLOv4	25
2.4	Trabalhos Relacionados	28
3	Metodologia	35
3.1	YOLOv4-Tiny	35
3.2	Base de Dados	37
3.3	Métricas na avaliação do desempenho da rede	44
3.4	Treino	47
3.4.1	Base de Dados Militar Experimental	50
3.4.2	Base de Dados Militar	51
3.4.3	Base de Dados Militar/Civil	52
4	Apresentação, Tratamento e Discussão dos Resultados	55
4.1	Validação	55
4.1.1	Base de Dados Militar Experimental	56
4.1.2	Base de Dados Militar	58
4.1.3	Base de Dados Militar/Civil	62

4.1.4	Exposição final dos vários treinos e os seus mAP	64
4.2	Teste	65
4.2.1	Teste a Militares	65
4.2.2	Teste a Militares e Civis	68
4.2.3	Teste a Civis	71
4.2.4	Os Melhores Pesos para o Modelo	74
4.2.5	Confirmação final dos vários pesos com vídeos adquiridos por um drone	75
	Pesos F	76
	Pesos H	77
	Pesos I	78
5	Conclusão	79
5.1	Conclusões Finais	79
5.2	Limitações	80
5.2.1	Base de dados e respetiva anotação	80
5.2.2	YOLOv4-tiny e as suas limitações	81
5.3	Trabalhos Futuros	82
	Bibliografia	85

Lista de Figuras

2.1	Relação entre a Visão Computacional, <i>Machine Learning</i> , <i>Deep Learning</i> e a visão humana, adaptado de (Khan, Rahmani, Shah e Benamoun 2018).	10
2.2	A performance da ML altera conforme a dimensão dos dados de treino. A performance de modelos ML a um determinado ponto fica saturada, enquanto a performance da DL continua a aumentar, retirado de (Xiaogang Wang 2016).	11
2.3	Diferenciação entre classificação e detecção de objetos, retirado de (Hulstaert 2018).	12
2.4	Segmentação de objetos, retirado de (Agarwal 2019).	12
2.5	Comparação entre a <i>Machine Learning</i> e <i>Deep Learning</i> , adaptado de (Kaur 2020).	13
2.6	Estrutura do modelo <i>Perceptron</i> de Rosenblatt, retirado de (Kurenkov 2020).	14
2.7	Ilustração da estrutura típica de uma CNN, retirado de (Alfarzaei, Qiang, J. Zhao, Eshaq e E. Hu 2020).	16
2.8	Operação de convolução, retirado de (Paper 2021)	16
2.9	Diferentes filtros em diferentes camadas numa CNN, retirado de (Dickson 2019).	17
2.10	Diferença entre as operações <i>max-pooling</i> e <i>average-pooling</i> quando aplicado um filtro 2x2 num <i>input</i> 4x4, retirado de (Yani, Irawan e Setianingsih 2019)	18
2.11	Imagem original com uma matriz de pixels 200x200, adaptado de (Ayyadevara e Reddy 2020)	19
2.12	Imagem com redução de pixels de 200x200 para 25x25, adaptado de (Ayyadevara e Reddy 2020)	19
2.13	Visualização da matriz de pixels 25x25 numa folha de Excel, adaptado de (Ayyadevara e Reddy 2020)	20
2.14	Linha cronológica do desenvolvimento de CNNs para a detecção de objetos, retirado de (Zou, Z. Shi, Guo e Ye 2019).	21

2.15	Arquitetura do R-CNN, retirado de (Murthy, Hashmi, Bokde e Geem 2020).	21
2.16	Divisão da imagem <i>input</i> numa grelha $S \times S$, onde posteriormente o modelo faz a previsão das diferentes classes, retirado de (Redmon, Divvala, Girshick e Farhadi 2016).	24
2.17	Como o computador processa as diferentes células da grelha $S \times S$, retirado de (Yamashita, Nishio, Do e Togashi 2018).	24
2.18	Arquitetura de um detetor de objetos, retirado de (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020).	25
2.19	Estrutura de uma rede com uma camada SPP. Aqui, 256 corresponde ao número do filtro da camada conv5, esta sendo a última camada convolucional, retirado de (He, Zhang, Ren e Sun 2015).	26
2.20	Arquitetura da PANet, inspirando a arquitetura da <i>Neck</i> no YOLOv4. (a) FPN <i>backbone</i> ; (b) <i>bottom-up path augmentation</i> ; (c) <i>adaptive feature pooling</i> ; (d) <i>box branch</i> ; (e) <i>fully-connected fusion (concatenation is done instead of addition for YOLOv4)</i> , retirado de (Parico e Ahamed 2021).	27
2.21	Arquitetura do modelo proposto por Shuo Liu e Zheng Liu em 2017, retirado de (Shuo Liu e Z. Liu 2017).	29
2.22	Ilustração da arquitetura de rede proposta por Hua et al., retirado de (Hua, Xinqing Wang, D. Wang, Huang e X. Hu 2018).	30
2.23	Arquitetura da CapsNet para a deteção de objetos militares, retirado de (Janakiramaiah, Kalyani, Karuna, Prasad e Krishna 2021). . . .	32
3.1	Comparação do YOLOv4 com o YOLOv4-Tiny, retirado de (Aleksey Bochkovskiy 2020).	36
3.2	Primeira deteção com o YOLOv4-tiny no computador, detetando uma mochila e uma pessoa, recolhida pelo drone pessoal DJI Mavic Air 2S.	37
3.3	Corte da primeira deteção de um vídeo com militares no YOLOv4-tiny, no computador.	37
3.4	Amostras de imagens da base de dados militar específica 1, retiradas da Internet.	38
3.5	Amostras de imagens da base de dados militar específica 2, provenientes de cenários de treino realizados pelos Fuzileiros Portugueses na Lituânia.	39
3.6	Amostras de imagens da base de dados militar específica 3, provenientes de cenários do exercício Tróia 2021, realizado pelos cadetes da Escola Naval.	40

3.7	Amostras de imagens da base de dados militar específica 4, provenientes de cenários do exercício Tróia 2022, realizado pelos cadetes da Escola Naval.	41
3.8	Amostras de imagens base de dados militar/civil específica, retiradas da Internet.	42
3.9	Amostras de imagens da base de dados civil específica, retiradas da Internet.	42
3.10	Três atributos em deteção de objetos (Classes, <i>bounding box</i> e <i>confidence score</i> [0 a 1], retirado de (Team 2021).	45
3.11	Representação da <i>Intersection over Union</i> (IOU), retirado de (Pardilla, Passos, Dias, Netto e Silva 2021).	45
3.12	Diferenciação entre TP, FP e FN, retirado de (Holleman 2018). . .	46
3.13	Estrutura de treino, validação e teste da rede.	48
4.1	Teste a uma imagem militar, pesos F.	67
4.2	Teste a uma imagem militar, pesos H.	68
4.3	Teste a uma imagem militar, pesos I.	68
4.4	Teste a uma imagem militar/civil, pesos F.	70
4.5	Teste a uma imagem militar/civil, pesos H.	70
4.6	Teste a uma imagem militar/civil, pesos I.	71
4.7	Teste a uma imagem civil, pesos F.	73
4.8	Teste a uma imagem civil, pesos H.	73
4.9	Teste a uma imagem civil, pesos I.	74
4.10	Amostras de imagens das gravações realizadas com o drone DJI Air 2S para os testes finais, à esquerda correspondem ao Teste2 e as da direita ao Teste1.	76
4.11	Amostras de imagens do Teste1 e do Teste2 com os pesos F.	77
4.12	Amostras de imagens do Teste1 e do Teste2 com os pesos H.	77
4.13	Amostras de imagens do Teste1 e do Teste2 com os pesos I.	78
5.1	Trabalhos Futuros.	82

Lista de Tabelas

2.1	Comparação dos resultados do algoritmo de Hua et al. com diferentes métodos de detecção em duas datasets diferentes, adaptado de (Hua, Xinqing Wang, D. Wang, Huang e X. Hu 2018).	30
2.2	Comparação do modelo Jiang et al. com outros modelos, adaptado de (Jiang, L. Zhao, Li e Y. Jia 2020).	31
2.3	Comparação das métricas <i>recall</i> e <i>precision</i> do algoritmo de Janakiramaiah et al. com dois outros métodos diferentes, adaptado de (Janakiramaiah, Kalyani, Karuna, Prasad e Krishna 2021).	33
3.1	Especificações da câmara do drone Mavic Air 2 FLIR usado na Lituânia.	39
3.2	Especificações da câmara do drone DJI Mini 2, usado no exercício Tróia da Escola Naval, em 2021.	40
3.3	Especificações da câmara do drone DJI Air 2S, usado no exercício Tróia da Escola Naval, em 2022.	41
3.4	Constituição da base de dados total.	43
3.5	Divisão e número das imagens anotadas para treino e para teste.	44
3.6	Fase de treino e a sua constituição.	49
3.7	Treinos a realizar.	50
3.8	Treino experimental da BD Militar com pesos pré-treinados.	51
3.9	Treino da BD Militar com pesos pré-treinados.	52
3.10	Treino da BD Militar com os pesos adquiridos de treinos anteriores.	52
3.11	Treino da BD Militar/Civil com os pesos pré-treinados.	52
3.12	Treino da BD Militar/Civil com os pesos pré-treinados, utilizando a BD Militar e a BD Militar/Civil em conjunto.	53
4.1	Número de imagens utilizadas para cada treino em específico, desde a fase do treino até à fase da validação.	56
4.2	Matriz de confusão dos melhores pesos do Treino F, <i>threshold</i> 0.5.	61
4.3	Matriz de confusão dos melhores pesos do Treino H, <i>threshold</i> 0.5.	63
4.4	Matriz de confusão dos melhores pesos do Treino I, <i>threshold</i> 0.5.	64

4.5	Exposição dos vários treinos e os seus mAP.	64
4.6	Nº de imagens e proveniência na fase de teste, relativa a cada secção de estudo.	65
4.7	Comparação da <i>precision</i> e <i>recall</i> dos melhores pesos escolhidos, relativamente ao melhor <i>threshold</i> , , em imagens com militares.	67
4.8	Comparação da <i>precision</i> e <i>recall</i> dos melhores pesos escolhidos, relativamente ao melhor <i>threshold</i> , em imagens com militares/civis.	70
4.9	Teste dos pesos F no grupo de teste apenas com civis.	71
4.10	Comparação da <i>precision</i> e <i>recall</i> dos melhores pesos escolhidos, relativamente ao melhor <i>threshold</i> , em imagens com civis.	72
4.11	Especificidades das gravações finais de teste, gravadas pelo drone DJI Air 2S.	75

Lista de Equações

2.1	Equação simplificada dos modelos <i>Machine Learning</i>	10
3.1	Equação da <i>precision</i>	46
3.2	Equação do <i>recall</i>	46
3.3	Equação da <i>mean Average Precision</i>	47
3.4	Filtros	51

Lista de Acrónimos

ANEPC	Autoridade Nacional de Emergência e Proteção Civil
AP	<i>Average Precision</i>
AUC	<i>Area Under the Curve</i>
CapsNet	<i>Capsule Network</i>
CBRNE	<i>Chemical, Biological, Radiological, Nuclear, and Explosives</i>
CEOV	Célula de Experimentação Operacional de Veículos Não Tripulados
CF	Corpo de Fuzileiros
CNN	<i>Convolutional Neural Network</i>
CPLP	Comunidade dos Países de Língua Portuguesa
CSAR	<i>Combat Search and Rescue</i>
DAE	Destacamento de Ações Especiais
DIoU	<i>Distance- Intersection over union</i>
DL	<i>Deep Learning</i>
DSSD	<i>Deconvolutional Single Shot Detector</i>
ELM	<i>Extreme Learning Machine</i>
FLIR	<i>Forward Looking InfraRed</i>
FN	<i>False Negative</i>
FOV	<i>Field of View</i>
FP	<i>False Positive</i>
FPN	<i>Feature Pyramid Network</i>
FPS	<i>Frame Per Second</i>
FRI	Força de Reação Imediata
GE	Guerra Eletrónica
GPU	<i>Graphics Processing Unit</i>
IA	Inteligência Artificial

IED	<i>Improvised Explosive Device</i>
ILSVRC	<i>ImageNet Large Scale Visual Recognition Challenge</i>
IOU	<i>Intesection over Union</i>
ISR	<i>Intelligence, Surveillance and Reconnaissance</i>
ISTAR	<i>Intelligence, Surveillance, Target Acquisition, and Reconnaissance</i>
mAP	<i>mean Average Precision</i>
MIFA	Missões das Forças Armadas
ML	<i>Machine Learning</i>
MS COCCO	<i>Microsoft Common Objects in Context</i>
NATO	<i>North Atlantic Treaty Organization</i>
NBQR	Nuclear, Biológico, Químico e Radiológico
ONU	Organização das Nações Unidas
OpenCV	<i>Open Source Computer Vision Library</i>
PANet	<i>Path Aggregation Network</i>
RCNN	<i>Region-based Convolutional Neural Network</i>
RFB	<i>Radial Basis Function</i>
SENSIAC	<i>Sensing Information Analysis Centre</i>
SPP	<i>Spatial Pyramid Pooling</i>
SSD	<i>Single Shot MultiBox Detector</i>
SVM	<i>Support vector machine</i>
TL	<i>Transfer Learning</i>
TP	<i>True Positive</i>
UAV	<i>Unmanned Aerial Vehicle</i>
UE	União europeia
UGV	<i>Unmanned Ground Vehicles</i>
VC	Visão Computacional
YOLO	<i>You Only Look Once</i>

Capítulo 1

Introdução

O conflito e a guerra é algo que é e sempre será inerente ao ser humano, uma matéria de extrema importância para um estado, país ou organização. A diferença entre a vida e a morte, um caminho para a sobrevivência ou destruição, é algo a ser obrigatoriamente estudado e examinado (Griffith 1963).

Ao longo do tempo, do decorrer de inúmeras operações militares e das lições retiradas em cada conflito, este estudo tem sido empenhado no desenvolvimento de capacidades militares e tecnológicas, conduzindo a uma mudança de paradigma quando nos referimos a operações militares no terreno. Esta mudança deve-se, essencialmente, à tentativa de mitigar os erros e lacunas que aconteceram em operações passadas, garantindo o sucesso e estado final desejado de uma determinada missão ou operação.

É aqui que são introduzidos os *Unmanned Aerial Vehicles* (UAVs), estes têm a capacidade de apoiar direta ou indiretamente o campo de batalha, aumentar o conhecimento situacional do terreno e do inimigo, das suas capacidades, ao sobrevoar o cenário de operações, atualizando e recolhendo a informação necessária para que o sucesso da operação seja alcançado (Glade 2000). Estes UAVs conseguem recolher e transmitir imagens capturadas em tempo real para um operador, no entanto, isso requer um operacional dedicado em exclusivo à pilotagem do UAV e à análise das imagens adquiridas. Com vista a automatizar e agilizar este processo, criando um sistema de apoio à decisão e facilitando ao operador a sua tarefa, pretende-se neste trabalho estudar, aplicar e analisar técnicas de deteção automática de militares a partir de uma câmara integrada num UAV.

No âmbito da deteção automática de militares, inserem-se os conceitos de Inteligência Artificial (IA), *Machine Learning*, *Deep Learning* e Visão Computacional. A IA, é um campo que combina a ciência computacional com bases de dados robustas, permitindo que uma máquina resolva determinados problemas que,

se fossem resolvidos pelo ser humano, exigiriam inteligência (Campesato 2020). A *Machine Learning*, um subcampo da IA, tenta responder a esta necessidade de fazer com que uma máquina processe informação como um ser humano mas, as estruturas dos seus algoritmos são simples e suportam poucos dados, originando uma máquina com falhas consideráveis quando falamos em muita informação a ser processada. Aqui entra o conceito de *Deep Learning* e Visão Computacional, a primeira foi criada como um subcampo da *Machine Learning*, possuindo algoritmos de inteligência artificial, capazes de suportar uma grande quantidade de informação e dados para que, a Visão Computacional, área da *Deep Learning*, aplicada ao conceito de processamento de imagens, seguisse o princípio da IA, tentando reproduzir características humanas ao nível do reconhecimento e compreensão de imagens (Goodfellow 2016).

Para que a *Deep Learning* consiga responder aos objetivos da Visão Computacional nos dias de hoje, com mais poder computacional e uma grande dimensão de dados, usa redes neurais profundas, inspiradas no funcionamento do córtex cerebral do ser humano, fazendo com que uma determinada máquina aprenda com a informação que lhe é administrada. No caso de imagens, estas redes são usadas em três tarefas específicas, segmentação, classificação e deteção de objetos (Khan, Rahmani, Shah e Bennamoun 2018). É na deteção de objetos que o nosso trabalho vai ser focado, permitindo classificar um determinado objeto, dizendo o que é em específico e localizá-lo na imagem, com uma *bounding box*.

O objetivo deste trabalho é, a partir de uma câmara integrada de um UAV, realizar a deteção automática de alvos militares, em tempo real, usando uma rede neuronal profunda de deteção de objetos. Para isso, será necessário a aquisição de uma boa base de dados, grande e variada o suficiente. Após isto, escolher a rede neuronal profunda, uma rede rápida na deteção de objetos em tempo real e com bons resultados. Realizar a anotação de todos os militares, em cada imagem presente na base de dados e treinar a rede. No final, pretende-se alcançar uma deteção automática de alvos militares em tempo real, a partir da rede neuronal profunda escolhida e treinada, validando e testando a mesma, requerendo um processamento baixo de análise de imagem, permitindo integrar a mesma num UAV, otimizando recursos, mão de obra e alcançando excelentes resultados.

A estrutura da tese é a seguinte: no Capítulo 2, irá ser abordada a revisão de literatura e enquadramento conceptual, assente nos diversos conceitos importantes para a realização deste trabalho, como a importância, utilização e emprego dos UAVS em operações militares e nos Fuzileiros, a inteligência artificial, *Convolutional Neural Networks* e trabalhos realizados. No Capítulo 3, será discutida a metodologia

do trabalho, a escolha concreta da rede neuronal profunda, a aquisição da base de dados, as métricas usadas na avaliação do desempenho da rede, a personalização da rede e o treino da mesma. No Capítulo 4, iremos abordar a validação e o teste da rede, com base nos melhores pesos provenientes do treino. No último capítulo, vamos expor as conclusões finais, limitações e sugestões para estudos e trabalhos futuros.

Capítulo 2

Revisão da Literatura e Enquadramento Conceptual

Neste capítulo, iremos abordar os diversos temas e conceitos importantes na realização e compreensão do nosso trabalho. Primeiramente, o apoio que os UAVs têm nas operações militares e as missões que podem desempenhar, bem como os UAVs nos Fuzileiros. De seguida, iremos falar sobre algoritmos de deteção automática e os diversos conceitos englobados, desde a inteligência artificial a redes neuronais profundas em específico. Por último, vão ser apresentados alguns trabalhos de investigação, baseados na correlação entre deteção de objetos e redes neuronais profundas.

2.1 UAVs em Apoio a Operações Militares

A ascensão do poder tecnológico do mundo é uma realidade nos dias de hoje. Nas últimas décadas, a tecnologia tem evoluído significativamente em diversos setores, entre os quais, o setor militar. No mundo militar, uma notória inovação tecnológica foi o emprego de UAVs. O desenvolvimento e utilização destes meios, abriu portas a novas possibilidades na condução de operações militares (Udeanu, Dobrescu e Oltean 2016).

As missões que os UAVs executam podem divergir relativamente às suas características, como função, alcance, peso, durabilidade, altitude máxima, tipo de motor, etc. Devido a estes fatores, estas missões podem ser variadas a um nível conjunto (Patrício 2011), tais como:

- *Intelligence, Surveillance, Target Acquisition, and Reconnaissance* (ISTAR);
- Missões de combate e apoio ao combate;

- Relé de Comunicações;
- Guerra Eletrónica (GE);
- *Combat Search and Rescue* (CSAR);
- Detecção *Chemical, Biological, Radiological, Nuclear, and Explosives* (CBRNE);
- Apoio Logístico;
- Detecção de *Improvised Explosive Device* (IED).

Dentro destas áreas e possibilidades que os UAVs conseguem trazer a nível militar, neste trabalho, o foco da sua utilização cairá sobre a definição de ISTAR.

Várias foram as operações, onde os UAVs foram utilizados na área ISTAR, como por exemplo na Guerra do Golfo, onde os EUA usavam regularmente UAVs de forma a fazerem vigilância e patrulha de aéreas (Warrell 2020). Nos Balcãs (1995-1999), onde desempenharam funções desde vigilância marítima ao reconhecimento do avanço das forças terrestres no Kosovo (Becker 1999).

De forma a especificar o emprego de UAVs no âmbito desta dissertação, iremos olhar para a Marinha, mais concretamente a sua Força Especial, os Fuzileiros. Com o objetivo de fornecer a esta força, mais uma valência no desempenho das suas missões.

2.1.1 UAVs nos Fuzileiros

Com o objetivo de perceber a importância da contribuição, caracterização e edificação da valência *Intelligence, Surveillance and Reconnaissance* (ISR) e utilização de UAVs nas missões atribuídas ao Corpo de Fuzileiros (CF), olhamos para trabalho de (Ferreira 2021) nesta área.

Recentemente, o CF tem participado em diversos projetos de desenvolvimento na área dos UAVs. De acordo com (Ferreira 2021), o desenvolvimento da aplicação desta capacidade no âmbito de diversas ações de treino e exercícios, bem como empenhamentos reais de cariz nacional e internacional toma um espectro alargado. Desde atividades de formação de operadores do Destacamento de Ações Especiais (DAE), Exercício *Flaming Sword* 18, *North Atlantic Treaty Organization* (NATO) *Assurance Measures*, apoio à Autoridade Nacional de Emergência e Proteção Civil (ANEPC), aprontamento para República Centro-Africana, apoio a Forças no CF, apoio à unidade de polícia Naval, apoio à Força de Reação Imediata (FRI) e apoio ao Comando da Zona marítima dos Açores. Todas estas atividades foram desde

abril de 2018 a Outubro de 2019, em locais como Portugal Continental, Lituânia, Moçambique, Açores e São Tomé e Príncipe.

Este conjunto de atividades pode-se dividir em duas vertentes distintas de emprego tático dos UAVs (Ferreira 2021). Um emprego focado no mapeamento, para apoio ao planeamento, recolha, análise e disseminação de produtos operacionais de informação, para operações futuras. Um emprego tático dinâmico, em tempo (quase) real, para apoio da manobra de uma força num campo de batalha através da captação de imagens e vídeo, no espectro diurno e noturno.

De uma forma geral, é importante saber e delimitar as tarefas associadas ao CF, para que possamos dar o contributo da valência ISR, através do que é previsto nas Missões das Forças Armadas (MIFA) (Ferreira 2021):

- Empregar forças e meios militares a fim de assegurar a satisfação dos compromissos internacionais assumidos por Portugal no quadro mútuo da defesa coletiva, quer no âmbito da NATO quer da UE;
- Participar em operações militares (anfíbias e terrestres), integrando forças nacionais e multinacionais, na defesa do Território Nacional ou dos interesses nacionais e estrangeiro;
- Participar no enquadramento militar da população, contribuindo para criar as condições militares para a resistência passiva e ativa, com o desiderato do controlo do território nacional e da reposição da autoridade do Estado em caso de agressão externa;
- Participar em operações e missões no âmbito da NATO, ONU, UE e CPLP, ou noutro que venha a ser estabelecido, a fim de contribuir para a promoção da paz e da segurança internacional;
- Participar em operações de paz autónomas ou integrado em forças conjuntas e/ou combinadas;
- Participar em operações de proteção de forças do âmbito Nacional, NATO ou quando integradas em forças conjuntas e/ou combinadas;
- Reforçar, quando o grau de ameaça justificar, os meios de vigilância e defesa das instalações críticas de Marinha e de outros pontos sensíveis;
- Apoiar forças e serviços de segurança e de outros ramos a fim de contribuir para a proteção de pessoas e bens, para o combate ao terrorismo e criminalidade organizada;

- Apoiar o emprego das operações especiais no cumprimento das missões que lhe são inerentes.

Após isto, é delimitada a contribuição da valência de ISR nas missões atribuídas ao CF, presentes em (Ferreira 2021):

- Recolha de informação (compilação sistemática de informação para apoio à decisão através de da deteção de táticas, procedimentos, armamento, bases/apoio, tendências/evolução, hábitos, vulnerabilidades, posição e número de forças opositoras);
- Aviso antecipado (incrementando o processo de tomada de decisão para a retenção da iniciativa ou a fim de evitar potenciais emboscadas);
- Investigação de contacto (posição suspeita, ou após contacto para a identificação, classificação, avaliação do estado, análise de movimentos, organização e intenções das forças adversárias);
- Perseguição de contacto (exploração de sucesso, flagelação e recolha de informação);
- Correção e direção de fogo indireto (correção de tiro de morteiro e ou fogo indireto sobre posição estabelecida das forças opositoras);
- Avaliação de danos infligidos às forças opositoras (análise rápida do estado e remanescente de forças opositoras após contacto);
- Deteção e monitorização em ambiente nuclear, biológico, químico e radiológico (NBQR);
- Deteção de minas e dispositivos explosivos improvisados;
- Recuperação de pessoal, incluindo busca e salvamento;
- Extensão da porta de comunicações;
- Guerra eletrónica;
- Apoio a operações psicológicas;
- Informação geospacial e reconstituição de infraestruturas;
- Identificação de embarcações marítimas e perigos à navegação;
- Monitorização de meteorologia e condições oceanográficas.

De forma a edificar a capacidade dos UAVs e as suas valências de ISR nas missões atribuídas ao CF, R. Ferreira acentua o objetivo de implementar esta capacidade no CF. Segundo os vetores de desenvolvimento Doutrina, Organização, Treino, Material, Liderança, Pessoal, Infraestruturas e Interoperabilidade (DOTMLPII). Um processo complexo e moroso com as suas potencialidades e desafios.

As potencialidades, presentes na qualidade e disponibilidade de informação, estreitando a relação de uma interoperabilidade conjunta e combinada, com o objetivo de facilitar e aprimorar o processo de tomada de decisão, minimizar o risco de perda de vidas humanas, otimização de recursos e oportunidade de haver uma transformação organizacional positiva (várias e novas valências adquiridas). Por outro lado, como desafios, com uma maior incidência ao nível da liderança, a escassez de recursos materiais, pessoais e sustentação técnica dos mesmos (Ferreira 2021).

Atualmente, a componente de utilização de UAVs no âmbito de ISR no CF, ainda se encontra limitada pelas barreiras presentes nos vetores de desenvolvimento DOTMLPII. No entanto, as vantagens e força que esta componente pode trazer para o CF, é demasiado significativa para ser ignorada. Apesar do foco do CF estar afastado quando nos referimos a Inteligência Artificial, quer porque o desenvolvimento da componente está numa fase inicial ou por falta de verbas suficientes para objetivos ambiciosos, não podemos parar no tempo e ignorar o progresso. Neste seguimento, iremos perceber o que é a Inteligência Artificial e a forma como conseguimos utilizá-la com o intuito de aprimorar a valência ISR através de um UAV, realizando a deteção automática de militares.

2.2 Inteligência Artificial (IA)

De modo a entender como a Visão Computacional (VC) faz uso de redes neuronais profundas para detetar militares, neste subcapítulo iremos abordar a relação da IA, uma ciência e engenharia com o objetivo de construir máquinas inteligentes, tendo capacidade de concretizar tarefas semelhantes às que são realizadas pela inteligência humana ou animal (McCarthy 2007), com a definição de VC. Interligando os subcampos da IA (*Machine Learning* (ML) e *Deep Learning*) (DL), no progresso da VC e na criação de redes neuronais profundas, podemos ver esta relação na Figura 2.1.

2.2.1 Visão Computacional

Estamos rodeados de informação, câmaras nos telemóveis, computadores, informação registada na Internet, nunca foi tão fácil recolher informação e publicá-la, ter acesso a ela. É aqui que é inserida a VC, um setor de estudo que se empenha no problema de ajudar os computadores a 'ver', a perceber o conteúdo de imagens digitais (Brownlee, 2019). Num nível abstrato, o objetivo dos problemas da VC é utilizar informação recolhida por imagens, de modo a inferir algo sobre o mundo, semelhante ao que a visão humana faz (Prince 2012).

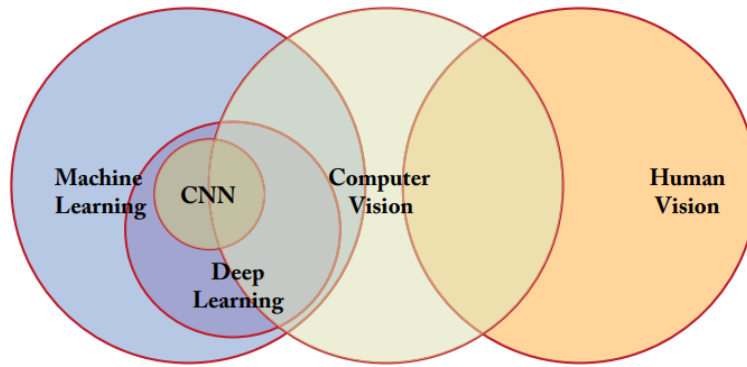


FIGURA 2.1: Realção entre a Visão Computacional, *Machine Learning*, *Deep Learning* e a visão humana, adaptado de (Khan, Rahmani, Shah e Bennamoun 2018).

2.2.2 *Machine Learning*

Para que a VC consiga perceber o conteúdo das imagens, numa tentativa de reproduzir a capacidade humana, muitos modelos, no início desta 'luta', apoiaram-se sobre a ML, funcionando de uma forma muito simples, como a expressão 2.1:

$$Y = f(X) \quad (2.1)$$

Onde após fornecidos determinados dados $\mathbf{X}$, o objetivo da ML é fazer a previsão dos resultados $\mathbf{Y}$.

O objetivo da VC é extrair informação útil através de imagens, o que se verificou ser uma tarefa extremamente desafiadora, ocupando milhares de mentes na procura de soluções para este problema, apesar deste esforço, a ML ficava ainda muito desatualizada na capacidade de construção de uma '*general-purpose seeing machine*' (Prince 2012).

Nos últimos anos, com o aumento tecnológico e a aquisição de uma grande quantidade de dados, foi observado que a performance de f nos dados de teste, aumentava proporcionalmente quando a dimensão dos dados de treino aumentava. Mas, devido à sua capacidade de aprendizagem, como é possível observar na Figura 2.2, a performance de estruturas de ML ficavam saturadas quando a dimensão do grupo de treino aumentava até um certo ponto, sendo os resultados insatisfatórios, enquanto os modelos DL tinham um melhor desempenho (Xiaogang Wang 2016).

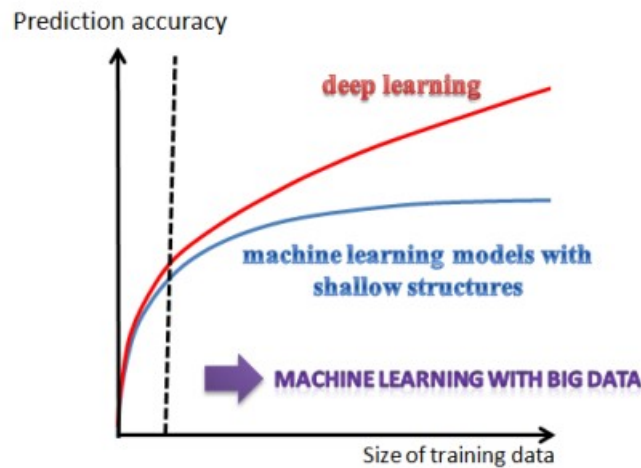


FIGURA 2.2: A performance da ML altera conforme a dimensão dos dados de treino. A performance de modelos ML a um determinado ponto fica saturada, enquanto a performance da DL continua a aumentar, retirado de (Xiaogang Wang 2016).

Algumas das tarefas mais importantes da VC que a ML tentava dar resposta, são as seguintes:

- Classificação de Objetos (Qual é a principal classe do objeto na imagem?);
- Detecção de Objetos (Onde estão os objetos na imagem e qual a classe deles?);
- Segmentação de Objetos (Que pixels pertencem ao objeto na imagem?).

A classificação de objetos determina que objetos em específico, estão contidos numa imagem, mas não os localiza no espaço. Contrariamente, a deteção de objetos, combina esta classificação e localização, determinando que objetos estão presentes na imagem e em que sítio da imagem, faz esta localização usando *bounding boxes* referentes a cada objeto individualmente (P. Sharma 2019), na Figura 2.3 podemos ver esta comparação entre classificação e deteção de objetos.

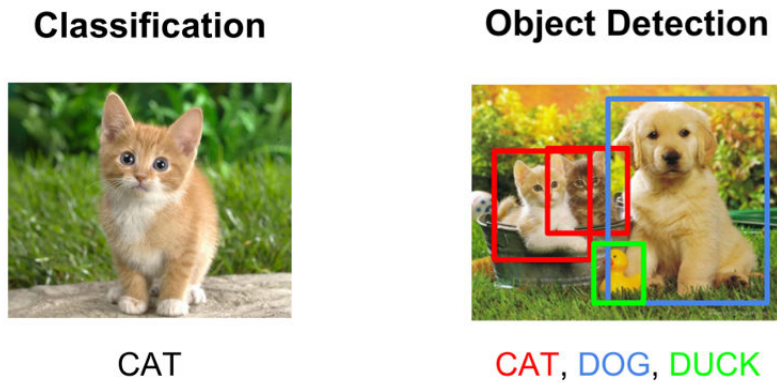


FIGURA 2.3: Diferenciação entre classificação e detecção de objetos, retirado de (Hulstaert 2018).

A segmentação de objetos consiste na separação dos objetos presentes na imagem em regiões, cada uma com a sua forma e fronteiras específicas, delineando áreas potencialmente significativas para processamento. A segmentação 'destaca' os objetos e facilita a sua avaliação, fornece detalhadamente os seus pixéis, diferentemente à classificação e detecção de objetos (Yuheng e Hao 2017). Na Figura 2.4 conseguimos ter uma ideia desta tarefa.

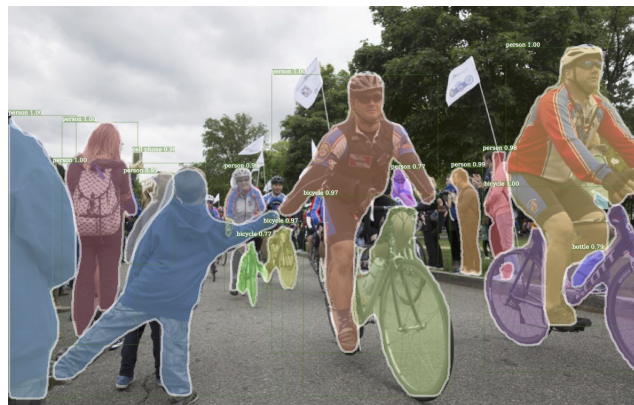


FIGURA 2.4: Segmentação de objetos, retirado de (Agarwal 2019).

2.2.3 *Deep Learning*

Com o decorrer do tempo, e na tentativa de satisfazer as tarefas da VC que a ML por si só não conseguia, a DL, inserida na ML, começou a ser desenvolvida. A DL é um mecanismo pelo qual um determinado algoritmo consegue aprender pelo exemplo, com o princípio de obter uma configuração desejada para um determinado

modelo, para que os dados de saída desejados possam ser obtidos a partir de um conjunto de dados de entrada (Ugail 2022). Esta tecnologia tem evoluído rapidamente e está a proliferar pelo mundo em áreas como a saúde, o entretenimento ou a robótica. A razão pela qual tem resultados superiores à ML, é devido à sua capacidade de aprendizagem com maior potencial, um elevado número de parâmetros e arquiteturas.

A grande diferença entre a ML e a DL, reside no facto da ML usar algoritmos construídos para tarefas específicas, com uma dimensão reduzida de dados, no conjunto de dados de entrada. A DL, por outro lado, consegue trabalhar com uma maior dimensão de informação, baseada num sistema com múltiplas camadas, onde cada camada usa o conjunto de dados de saída da camada anterior, permitindo aos modelos converter esta informação de grandes dimensões, em informação que consiga ser processada e analisada (Sperling 2018). Como podemos ver pela Figura 2.5, usando um exemplo de classificação, a ML necessita de que o programador faça a seleção específica das características que quer analisar, enquanto a DL possui modelos que fazem esta extração de caraterísticas automaticamente.

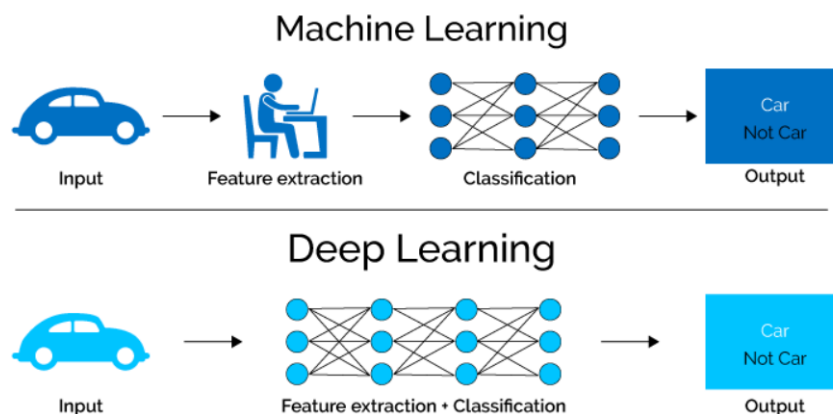


FIGURA 2.5: Comparação entre a *Machine Learning* e *Deep Learning*, adaptado de (Kaur 2020).

Para que a DL consiga responder a estes requisitos, de trabalhar com uma grande dimensão de dados, realizar a extração de caraterísticas automaticamente, processar muita informação e no final conseguir fornecer os dados de saída desejados, usa redes neuronais profundas.

Estas redes neuronais representam um paradigma computacional alternativo, na medida que a solução para um determinado problema, é dado com base no treino que o modelo teve através de um conjunto de exemplos. Esta inspiração

vem, originalmente, de estudos com base nos mecanismos que processam informação nos sistemas nervosos biológicos (Bishop 1994). No âmbito do nosso trabalho, vamos abordar um tipo de rede neuronal profunda, atualmente utilizada pelos seus bons resultados no processamento de imagens, as *Convolutional Neural Networks* (CNNs).

2.3 Convolutional neural network (CNN)

2.3.1 Inícios de estudo e criação da CNN

A história das redes neurais retoma o ano de 1943, quando Warren McCulloch e Walter Pitts escreveram um trabalho sobre como os neurónios funcionavam, criando uma rede neuronal simples usando circuitos elétricos (Abraham 2002). Em 1949, Donald Hebb apontou, no seu trabalho (Hebb 2005), para o facto das ligações neuronais serem fortificadas cada vez que fossem usadas, um conceito essencial quando se fala de como o ser humano aprende. Por esta altura, sem se saber, já se desvendava a base do que são as redes neurais.

Em 1957, Frank Rosenblatt, apoiou-se nas obras de McCulloch, Pitts e Hebb e concebeu o chamado *Perceptron* como um modelo matemático simples, representativo de como os neurónios trabalham. Como é possível ver pela Figura 2.6, este modelo dá apenas um conjunto de dados de saída com duas opções possíveis, 1 e 0, dependendo da soma dos dados de entrada (x) com os pesos (w).

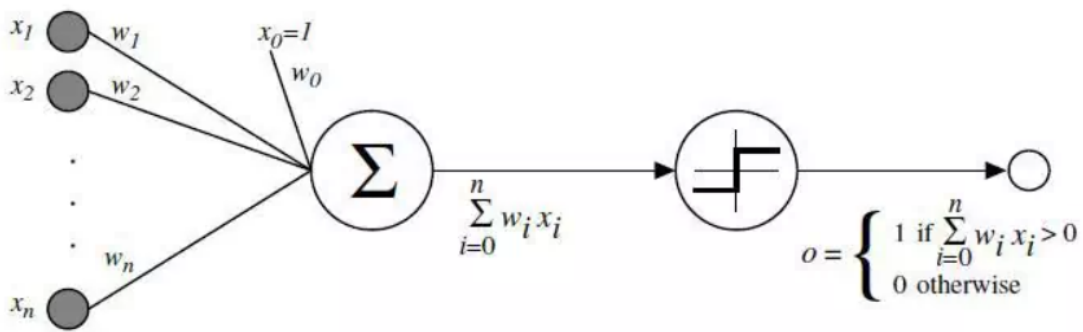


FIGURA 2.6: Estrutura do modelo *Perceptron* de Rosenblatt, retirado de (Kurenkov 2020).

Mais tarde, foi aferido que este modelo *Perceptron* não era possível de ser utilizado quando se fala de múltiplas camadas e extração de características específicas dos dados de entrada. Em 1980, por sua vez, foi criada a primeira arquitetura

tipo CNN, chamada Neocognitron. Esta conteve o que arquiteturas passadas não possuíam. Nesta estrutura, os conceitos de *feature extraction*, *pooling layers*, a utilização de convoluções numa rede neuronal, reconhecimento e classificação foram abordados (Fukushima e Miyake 1982).

Entre 1989 e 1998 a denominação de *Convolution Neural Network* formou-se a partir da criação da LeNet por Yann Lecun (LeCun, Bottou, Bengio e Haffner 1998), utilizada no reconhecimento de escrita à mão. No entanto, foram encontradas dificuldades nas redes neuronais entre o período de 1980 a 2000, nomeadamente a realização do treino dos vários parâmetros que as mesmas possuem. Era fácil de haver um *overfit* nos dados de treino, ou seja, os modelos tinham bons resultados quando testados com dados parecidos ou iguais aos utilizados no treino mas, quando testados com dados completamente diferentes, apresentavam maus resultados. Isto por não haver bases de dados de grandes dimensões e pela limitação a nível computacional na altura. Por esse motivo, o foco no estudo e utilização de redes neuronais perdeu-se, dando mais atenção a ferramentas da ML com capacidade de treino, deteção e previsão como a *Support vector machine* (SVM), Boosting e K-Nearest Neighbor (Xiaogang Wang 2016).

As CNNs são arquiteturas muito usadas quando se fala em análise de imagens, na deteção de objetos referida anteriormente e em arquiteturas mais usadas da *deep neural networks*. Além destas características, as CNNs têm uma significativa importância em segmentação de imagens, temos o exemplo das *Generative Adversial Networks* (GANs), estas permitem gerar uma imagem através de múltiplas imagens. Essencialmente, as CNNs são utilizada sempre que se usam imagens para algum fim em específico (Ayyadevara e Reddy 2020). CNNs são redes semelhantes aos nervos do cérebro humano, têm vários estágios em que conseguem ser treinadas. Sendo k o número de estágios. Os dados de entrada e de saída de cada estágio são chamados *feature maps*. Para o primeiro estágio, os *inputs features maps* são imagens. Os dados de saída de cada j estágio são as características extraídas de todas as localizações das *input features maps* desse estágio. Cada j estágio consiste em três tipos de camadas, *convolutional layer*, *pooling layer* e um último estágio, *fully connected layer* que dá as previsões dos dados de saída das classes de determinado objeto (Xiaogang Wang 2016). É possível observar esta arquitetura descrita pela Figura 2.7.

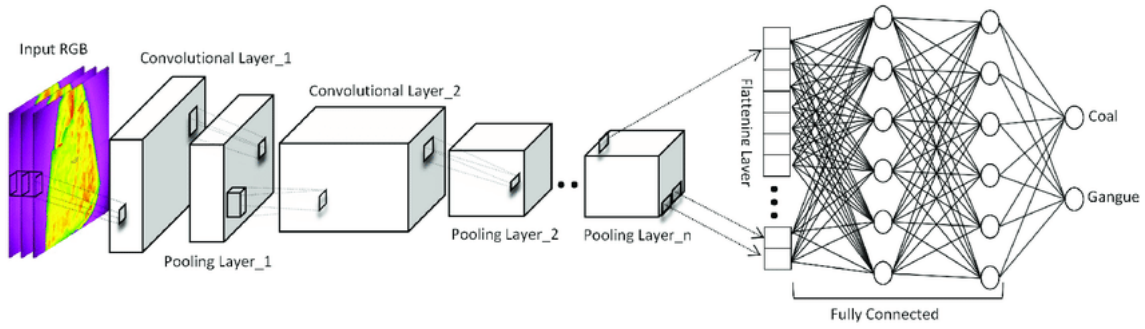


FIGURA 2.7: Ilustração da estrutura típica de uma CNN, retirado de (Alfarzaei, Qiang, J. Zhao, Eshaq e E. Hu 2020).

A convolução que se realizada na primeira camada, não é mais do que uma 'multiplicação' entre duas matrizes, Figura 2.8. Esta convolução é a chave no treino das redes neuronais, conduzida quando se calcula valores de camadas ocultas. Os resultados destes valores são apresentados após a 'multiplicação' da matriz dos valores de entrada, extraídos na multiplicação dos dados da imagem de entrada por filtros, denominados de filtros *kernel*, com os valores de peso ligados aos dados de entrada da camada oculta (Ayyadevara e Reddy 2020).

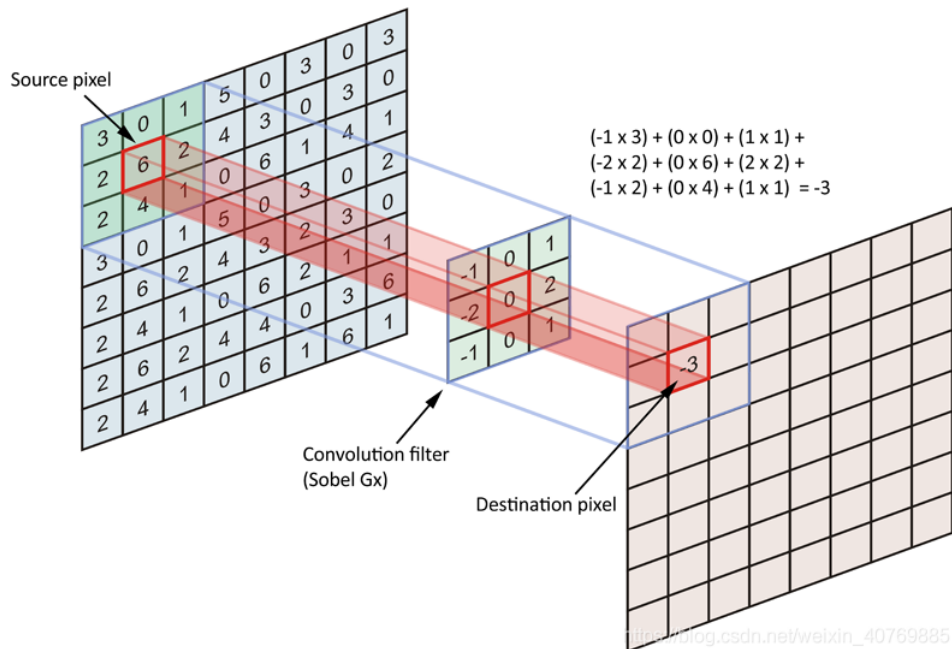


FIGURA 2.8: Operação de convolução, retirado de (Paper 2021)

Este filtro, é uma matriz de pesos que é iniciada aleatoriamente do início da análise da imagem. O modelo aprende os valores de peso ideais de um filtro

2.3. Convolutional neural network (CNN)

quando aumenta as épocas (Ayyadevara e Reddy 2020), ou seja, quando aumenta o número de passagens completas no conjunto de dados de treino pelo algoritmo da rede. Em geral, quanto mais filtros uma CNN possui, mais características de uma imagem conseguem ser reconhecidas e aprendidas pelo modelo. Por exemplo, um certo filtro poderá aprender sobre a matrícula de um carro e resultar numa ativação elevada (valor de 'multiplicação' de matrizes) quando na parte da imagem em que a rede está a realizar a convolução, é apresentada a matrícula de um carro.



FIGURA 2.9: Diferentes filtros em diferentes camadas numa CNN, retirado de (Dickson 2019).

Como podemos observar pela Figura 2.9, nas primeiras camadas de uma CNN, são aplicados filtros que detetam características específicas como arestas, cantos, círculos e cores, à medida que a informação se desloca para as camadas mais profundas, estes filtros começam a detetar características gerais, como olhos, rostos e objetos inteiros (Dickson 2019). Quanto mais filtros, mais informação é recolhida, visto que cada filtro aprende a analisar uma característica específica (Ugail 2022).

Esta função de convolução nas CNNs, possibilita a redução da dimensionalidade dos resultados. No caso de uma imagem, a operação de convolução analisa pequenas partes da imagem e extrai um conjunto de dados. A operação de *pooling*, por sua vez, reduz esse conjunto de dados para um número que consegue ser melhor analisado pelas restantes camadas da rede, até originar um resultado concreto, focando a análise em padrões relevantes, mantendo, ao mesmo tempo, informação crucial (Ugail 2022).

Max-pooling e *average-pooling* são as funções *pooling* mais usadas nesta operação. Na figura 2.10 podemos observar a diferença entre uma e outra, usado um filtro de dimensão 2x2 com um *input* e 4x4. Na *max-pooling*, cada filtro 'escolhe' o valor máximo e de seguida organiza a informação num *output* com dimensão de 2x2. Na *average-pooling*, por sua vez, é tirado o valor médio quando aplicado o filtro.

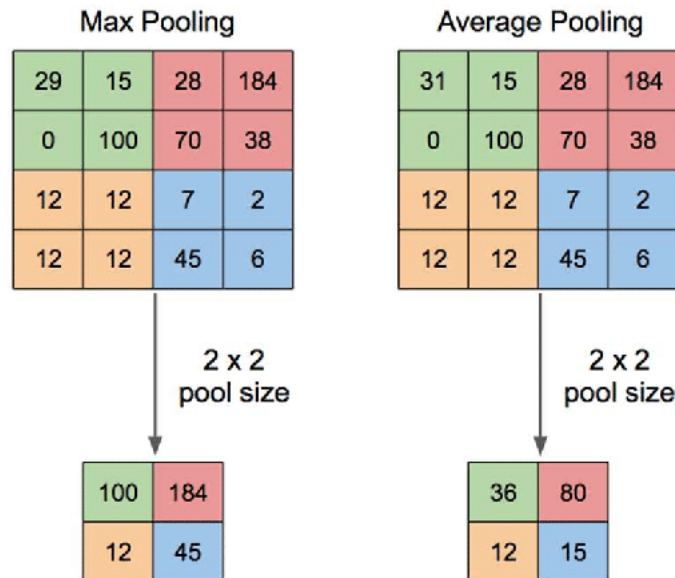


FIGURA 2.10: Diferença entre as operações *max-pooling* e *average-pooling* quando aplicado um filtro 2x2 num *input* 4x4, retirado de (Yani, Irawan e Setianingsih 2019)

2.3. Convolutional neural network (CNN)

A *Fully Connected Layer*, última camada de uma CNN, realiza o achatamento dos dados (matrizes) *output*, derivados das camadas anteriores, de modo que esses dados sejam possíveis de serem trabalhados e analisados, como é representado no exemplo abaixo, como exemplificado em (Ayyadevara e Reddy 2020):

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \{1, 2, 3, 4\}$$

Com este processo, transformaram-se os dados de saída da camada *pooling* em informação semelhante ao da camada de dados de entrada, para melhor explicar esta operação, iremos olhar para as figuras abaixo (Ayyadevara e Reddy 2020).

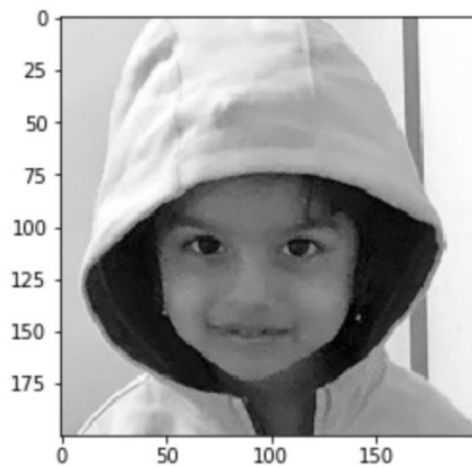


FIGURA 2.11: Imagem original com uma matriz de pixels 200x200, adaptado de (Ayyadevara e Reddy 2020)

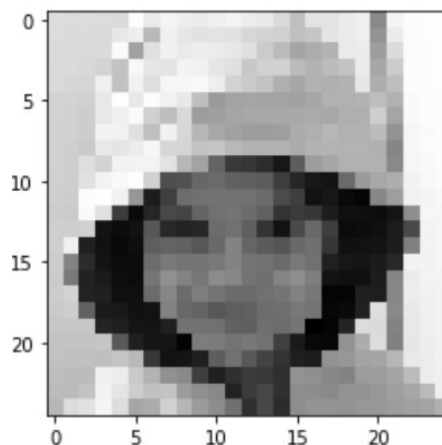


FIGURA 2.12: Imagem com redução de pixels de 200x200 para 25x25, adaptado de (Ayyadevara e Reddy 2020)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	222	220	221	220	218	253	234	245	238	235	239	243	236	232	218	193	228	228	234	239	139	245	252	253	253
2	221	219	219	218	232	239	186	240	231	226	227	226	215	212	209	193	199	229	234	239	150	236	252	253	253
3	219	218	218	218	251	165	224	241	234	238	236	231	224	204	180	166	173	180	234	236	159	219	252	252	253
4	218	219	216	211	196	240	231	228	243	241	229	224	201	209	210	189	181	189	196	255	168	204	252	252	253
5	218	214	213	240	189	242	223	246	246	249	238	211	203	196	177	168	179	176	178	211	175	191	252	252	253
6	212	212	208	232	254	232	252	241	232	192	155	164	166	165	164	163	166	178	178	181	190	178	250	252	251
7	211	209	205	232	240	251	208	191	217	158	161	166	169	169	170	170	171	169	176	177	206	166	250	252	251
8	209	209	205	243	242	225	198	241	215	184	169	163	159	158	160	173	176	184	178	179	189	156	246	250	252
9	210	207	203	232	229	236	246	214	213	196	199	185	179	179	181	172	179	180	180	181	177	136	246	251	252
10	209	206	222	212	242	243	244	226	184	165	104	61	57	48	27	97	158	167	178	178	178	136	246	249	252
11	208	206	225	243	249	254	209	82	85	105	109	100	98	95	65	43	28	24	109	156	169	175	242	248	251
12	208	205	252	255	242	153	53	66	111	116	117	116	115	109	78	66	22	27	14	9	117	156	241	245	249
13	205	204	250	225	63	15	42	77	71	104	115	118	110	101	56	64	40	34	20	20	17	25	138	246	246
14	208	206	205	23	16	22	90	45	39	43	110	115	99	56	23	78	107	65	15	17	20	32	70	244	246
15	208	219	37	22	14	15	97	102	100	90	108	133	104	94	88	108	114	57	21	22	23	33	138	243	246
16	205	133	48	24	15	17	110	124	118	119	124	134	119	116	109	123	116	36	27	25	31	42	242	243	246
17	204	124	38	30	19	16	120	146	133	121	142	138	118	114	135	145	120	24	20	21	33	136	236	243	244
18	205	212	39	37	20	20	110	137	110	128	119	109	109	115	119	133	121	8	9	36	34	137	237	241	243
19	204	209	101	31	29	21	22	127	108	102	91	97	101	111	113	122	118	11	14	18	222	139	233	240	242
20	200	200	200	41	36	25	24	37	117	117	116	101	99	114	111	118	4	8	41	220	219	134	232	239	244
21	197	196	196	199	92	37	26	25	8	127	125	118	122	116	67	31	13	11	150	173	220	131	231	238	242
22	195	193	193	192	198	187	56	22	25	27	97	115	93	70	55	36	33	140	153	165	166	183	233	236	242
23	192	190	189	240	237	202	180	187	140	66	36	52	64	61	51	150	146	138	134	157	159	166	189	237	240
24	189	188	197	244	229	206	194	196	157	146	138	39	63	73	53	144	143	139	150	148	153	161	167	187	239
25	184	220	225	245	221	187	178	209	183	157	143	116	51	78	40	144	150	150	148	153	153	158	162	165	178

FIGURA 2.13: Visualização da matriz de pixels 25x25 numa folha de Excel, adaptado de (Ayyadevara e Reddy 2020)

Esta informação, presente nas Figuras 2.11, 2.12 e 2.13, para ser observada concretamente pelos nossos olhos, resulta de informação viável de ser trabalhada, esta sendo:

array([[222, 220, 221, 220, 218, 253, 234, 245, 238, 235, 239, 243, 236, 232, 218, 193, 228, 228, 234, 239, 139, 245, 252, 253, 253], [221, 219, (...)] (...)]

Neste exemplo reduzido da matriz de pixels, que por sua vez define a Figura 2.12, consegue-se agora entender melhor o que esta *fully connected layer* reproduz no fluxo de uma CNN. Após a análise desta informação, nesta última camada, o modelo faz a previsão dos modelos de saída com base em probabilidades, detetando um determinado objeto.

2.3.2 O crescimento das CNNs

Foi em 2012 que, devido ao desafio de classificação da base de dados ImageNet, denominado por 'ImageNet large scale visual recognition challenge' (ILSVRC), novas e melhoradas arquiteturas de CNN foram criadas, destacando-se a AlexNet por Alex Krizhevsky (Krizhevsky, Sutskever e Hinton 2012). Com isto, houve uma evolução no desenvolvimento de CNNs, como podemos ver pela Figura 2.14.

2.3. Convolutional neural network (CNN)

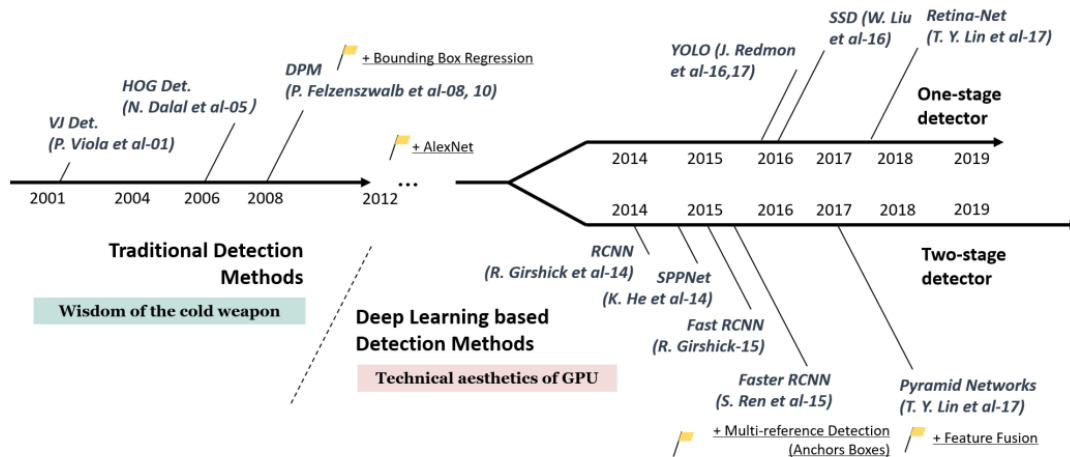


FIGURA 2.14: Linha cronológica do desenvolvimento de CNNs para a detecção de objetos, retirado de (Zou, Z. Shi, Guo e Ye 2019).

Posteriormente à criação da AlexNet, há uma diferenciação entre um detetor de objetos de um estágio e de dois estágios. Essencialmente, um detetor de objetos de dois estágios, como o *Region-based Convolutional Neural Network* (R-CNN), começa com a extração de um conjunto de regiões propostas na imagem de entrada através de um algoritmo de procura seletiva, como é possível identificar na Figura 2.15. Na fase seguinte, cada proposta é redimensionada para um tamanho fixo e colocada numa CNN pré-treinada de modo a extrair características. Finalmente, classificadores SVM são usados, de forma a prever a presença de um objeto dentro de cada região e classificar classes. Apesar do R-CNN ter alcançado um grande sucesso na altura, este primeiro estágio que realiza a extração de regiões propostas faz como que o modelo tenha uma velocidade muito lenta (Zou, Z. Shi, Guo e Ye 2019).

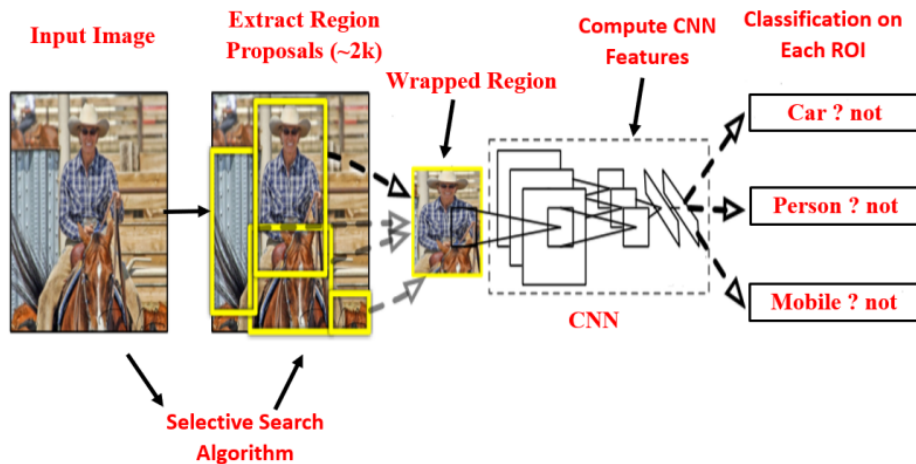


FIGURA 2.15: Arquitetura do R-CNN, retirado de (Murthy, Hashmi, Bokde e Geem 2020).

De modo a melhorar o desempenho dos detetores de dois estágios, outros modelos foram criados, tais como o Fast R-CNN e o Faster R-CNN, ambos apresentam melhorias relativamente ao R-CNN mas, quando comparados aos detetores de objetos de um estágio, usados para deteção de objetos em tempo real, a velocidade de processamento de imagem em tempo real é mais reduzida (Zou, Z. Shi, Guo e Ye 2019).

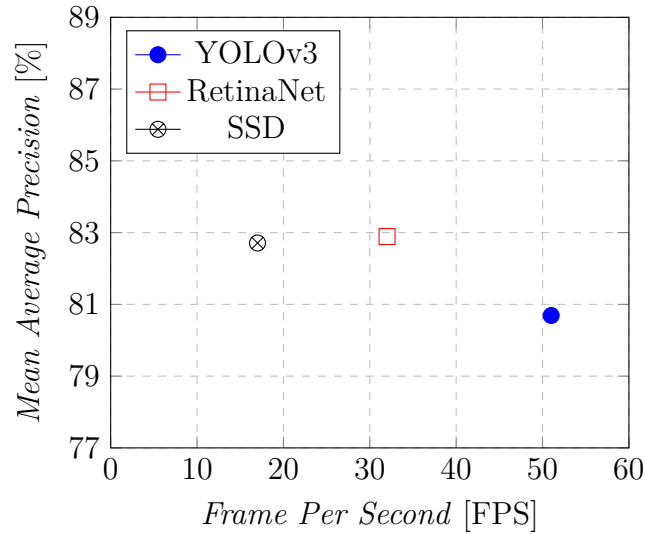
Os detetores de um estágio, não contêm o estágio das regiões propostas, estes detetores 'correm' diretamente sobre uma amostra densa de localizações. São exemplo desde modelos o *You Only Look Once* (YOLO), o *Single Shot MultiBox Detector* (SSD) e a RetinaNet (Zou, Z. Shi, Guo e Ye 2019). O YOLO aplica uma única rede neuronal à imagem completa. Esta rede divide a imagem em regiões e faz a previsão das *bounding boxes* e probabilidades para cada região em simultâneo. Apesar da boa velocidade de deteção, o YOLO sofre de uma queda de precisão na localização, comparativamente aos detetores de dois estágios, especialmente quando se trata de objetos de dimensão reduzida. Com o intuito de resolver este problema do YOLO, foi apresentado o SSD. A diferença deste para os outros detetores, é realizar a deteção de objetos de diferentes escalas em diferentes camadas da sua arquitetura, enquanto os detetores passados apenas faziam a deteção nas suas camadas superiores.

Apesar da velocidade elevada na deteção e simplicidade, estas redes de um estágio continuavam a perder com as redes de dois estágios, quando se falava em precisão. Para combater esta lacuna, foi construída a RetinaNet, esta rede sofreu a introdução de uma nova função chamada '*focal loss*', após ser afirmado que a causa derivaria de um desequilíbrio de classes de fundo (*foreground-background class imbalance*), encontrado durante o treino dos detetores.

Tendo em conta que este trabalho tem, como um dos objetivos finais, colocar uma rede neuronal num sistema embebido e realizar a deteção de objetos em tempo real, a escolha da rede neuronal recai sobre estes modelos de um estágio.

Olhando para o trabalho sobre a comparação destes três modelos na identificação em tempo real de pílulas (Tan, Huangfu, Wu e Chen 2021), obtiveram-se alguns resultados no que toca à sua performance, no Gráfico 1.1 é possível aferir estes resultados onde, em termos de precisão média, os modelos variam pouco, o YOLOv3 com o valor mais baixo (80.69%) e a RetinaNet com o mais elevado (82.69%) mas, olhando para a velocidade, *Frame Per Second* (FPS), o YOLOv3 confere uma vantagem significativa na sua velocidade em comparação aos outros dois modelos.

Gáfico 1.1. Comparação entre modelos de um estágio



2.3.3 YOLO

YOLO, como o nome indica, *You Only Look Once*, é um sistema de detecção de objetos que pode ser aplicado em tempo real, conseguindo reconhecer múltiplos objetos numa imagem apenas. Usa, portanto, uma abordagem diferente relativamente a outras redes neurais, tais como o R-CNN ou o Mask R-CNN, como já foi referido. Esta rede é relativamente simples, uma única rede neuronal convolucional que prevê, simultaneamente, múltiplas *bounding boxes* e probabilidades de classes para cada uma dessas caixas, tentando encontrar rapidamente os centros dos objetos. Esta rede tem inúmeros benefícios, desde já a sua rapidez, sendo possível aplicar a rede num vídeo *streaming* em tempo real. Adicionando a esta qualidade, YOLO alcança mais de metade da precisão média de outros sistemas de processamento de imagem em tempo real (Redmon, Divvala, Girshick e Farhadi 2016).

A função básica do YOLO é demonstrada na Figura 2.16, onde a rede divide a imagem *input* numa grelha $S \times S$, e cada célula da grelha é responsável por fazer a previsão do objeto centrado na mesma.

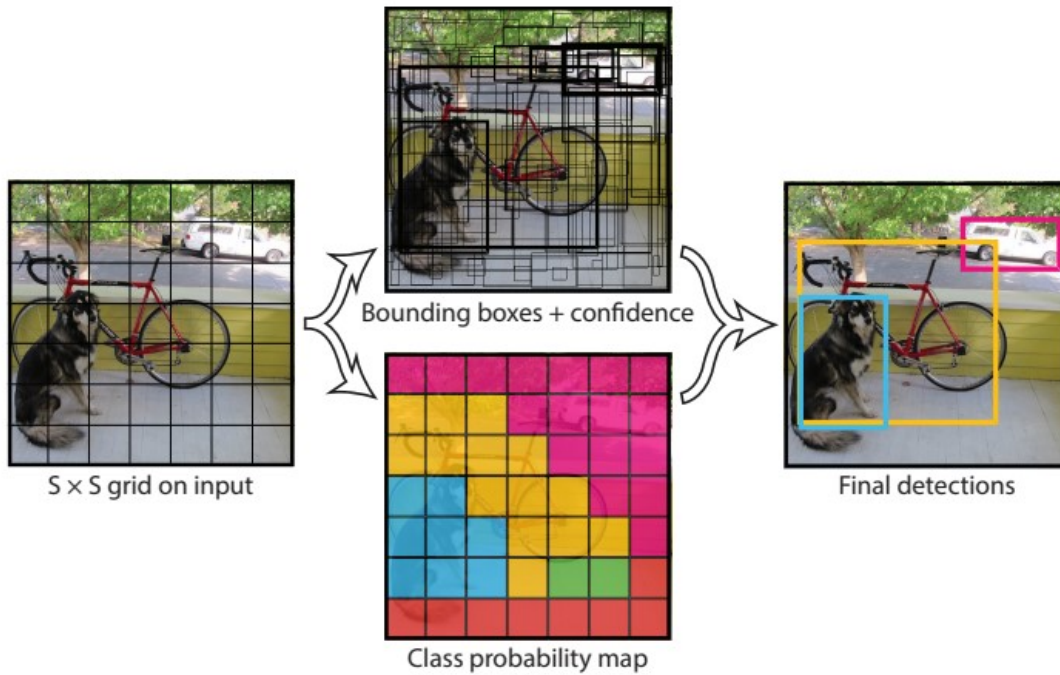


FIGURA 2.16: Divisão da imagem *input* numa grelha $S \times S$, onde posteriormente o modelo faz a previsão das diferentes classes, retirado de (Redmon, Divvala, Girshick e Farhadi 2016).

Cada célula da grelha faz a previsão de um determinado número de *bounding boxes* e define percentagens/probabilidades para cada uma dessas caixas, baseada na percentagem de cada célula da grelha, como a Figura 2.17 demonstra.

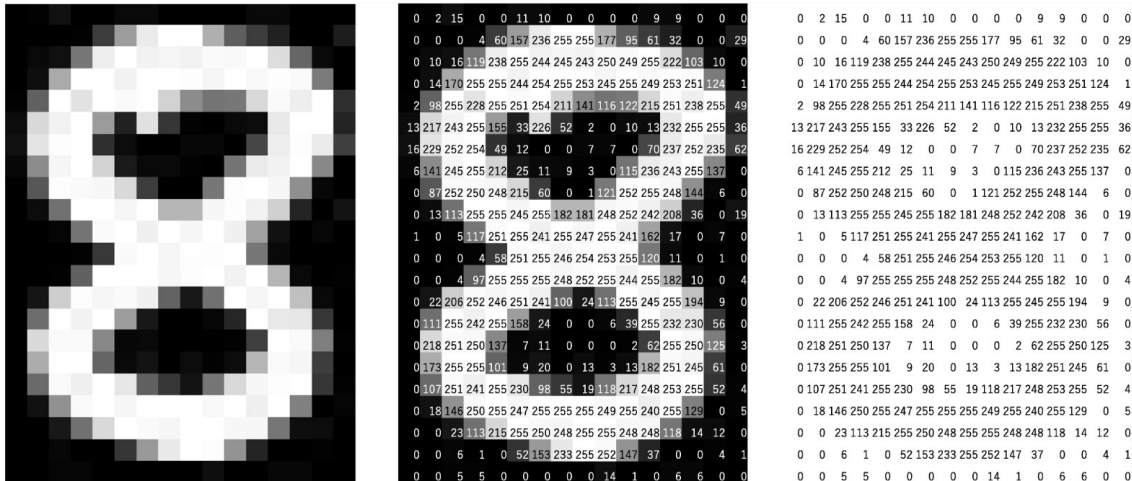


FIGURA 2.17: Como o computador processa as diferentes células da grelha $S \times S$, retirado de (Yamashita, Nishio, Do e Togashi 2018).

A escolha do YOLOv4 para este trabalho, criado por Alexey Bochkovskiy, Chien-Yao Wang e Hong-Yuan Mark Liao, deve-se à sua evolução comparativamente

ao YOLOv3, tornando-se mais rápido e mais eficiente, aumentando a *Average Precision* (AP) e os FPS em 10% e 12% respetivamente (Techzizou 2021b).

2.3.4 YOLOv4

De modo a justificar a escolha da rede neuronal YOLOv4, temos de responder a algumas questões, nomeadamente qual a sua estrutura, como foi criada, para que fim, o porquê do uso de certos métodos e qual é a sua comparação relativamente a outras redes.

O objetivo da criação do YOLOv4, teve por base a ideia de alcançar uma rede de deteção de objetos com uma velocidade elevada, para sistemas de produção e otimização em computação paralela (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020). Esta criação ambiciona o fácil treino do sistema e o uso do mesmo para que, mesmo usando um *Graphics Processing Unit* (GPU) local, se consigam alcançar resultados consideráveis em tempo real.

A arquitetura de um detetor de objetos segue os constituintes presentes na Figura 2.18, nomeadamente, *Input*, *Backbone* e *Neck*. O *Input* refere-se à fase onde se insere a imagem de entrada. Seguidamente, temos o *Backbone*, referente à rede que 'agarra' na imagem de entrada e extrai as *feature maps* (e.g. VGG16, Resnet-50, Darknet52 e RESNEX50). Imediatamente a esta fase, o *Neck*, e *Head*, subconjuntos do *Backbone*. O *Neck* serve para realçar a robustez da *feature map* extraída do *Backbone* usando (e.g. FPN, PAN e RFB). O *Head* lida com as previsões, dando uso a um detetor de objetos de um estágio (*Dense Prediction*) (e.g. YOLO, SSD e RetinaNet), ou utilizando um detetor de objetos com dois estágios (*Sparse Prediction*) (e.g. Fast R-CNN, Faster R-CNN).

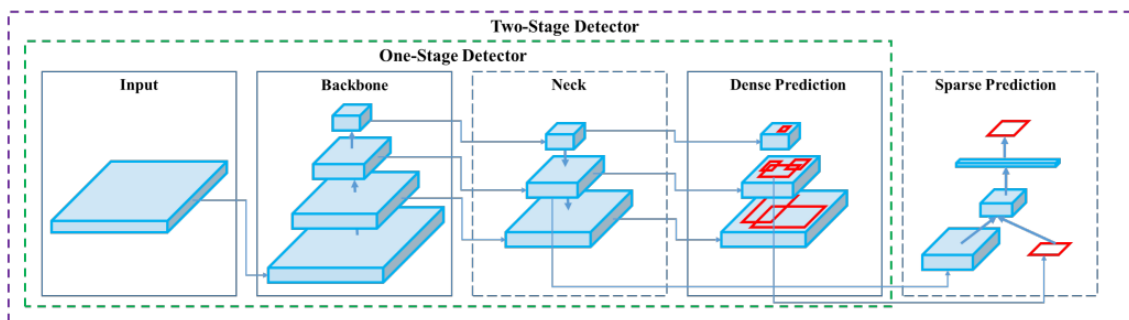


FIGURA 2.18: Arquitetura de um detetor de objetos, retirado de (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020).

Há imensas variedades de combinações de arquiteturas que podem ser feitas de forma a criar um bom detetor de objetos. Conforme a estrutura presente na Figura 2.18, Bonchkovskiy, Wang e Liao escolheram a melhor combinação para aplicar no YOLOv4.

No *Backbone* a escolha recaiu sobre o modelo CSPDARKNET53, isto derivado da justificação teórica e de vários estudos realizados com outros modelos (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020). No *Neck*, foi adicionado o bloco *Spatial Pyramid Pooling* (SPP) à CSPDarknet53, visto que aumentava significativamente o campo recetivo, ou seja, separava as caraterísticas mais significativas e causava quase nenhuma redução na velocidade de operação da rede. Como é possível visualizar na Figura 2.19, com a SPP, a imagem *input* pode ter qualquer tamanho, permitindo proporções arbitrárias e admitindo tamanhos arbitrários. A SPP divide as *features maps* em divisões mais finas, já com tamanhos definidos independentemente do tamanho da imagem de entrada, agregando características locais em cada divisão (He, Zhang, Ren e Sun 2015).

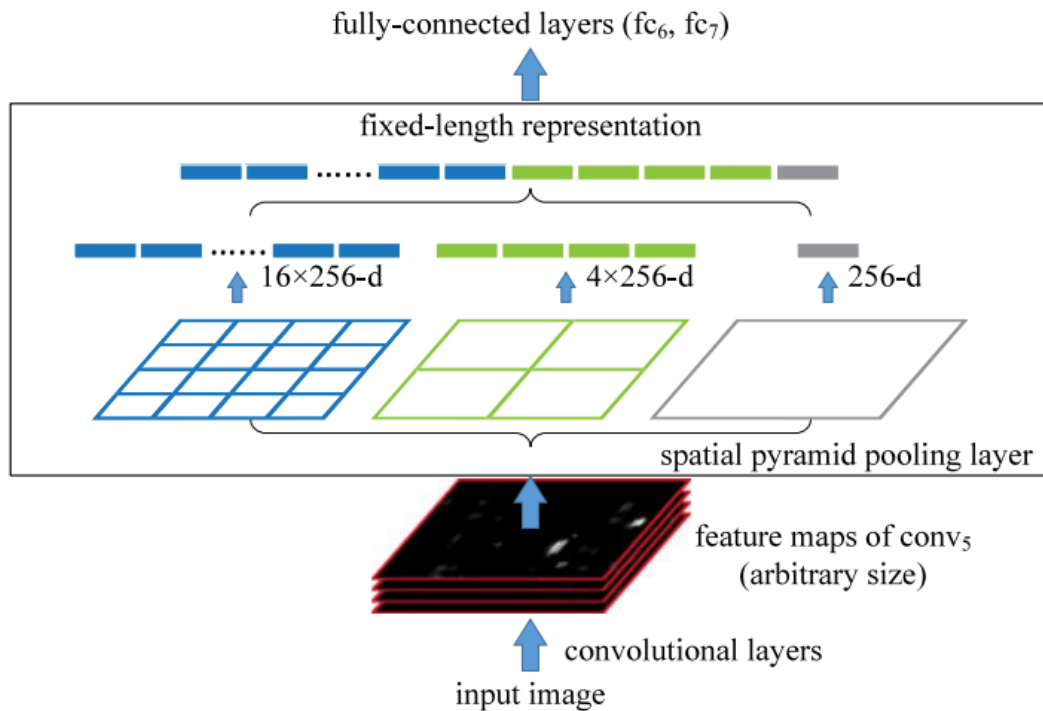


FIGURA 2.19: Estrutura de uma rede com uma camada SPP. Aqui, 256 corresponde ao número do filtro da camada conv5, esta sendo a última camada convolucional, retirado de (He, Zhang, Ren e Sun 2015).

2.3. Convolutional neural network (CNN)

Ainda no *Neck*, foi usado a *Path Aggregation Network* (PANet) como método de parâmetro de agregação de vários níveis da *backbone* para diferentes níveis de detetores, este método substitui o *Feature Pyramid Network* (FPN), utilizado no YOLOv3. O PANet, originalmente proposto por (Shu Liu, Qi, Qin, J. Shi e J. Jia 2018), faz esta agregação através da *augmentation*, fenómeno que garante um aumento de dados de treino com técnicas que permitem um aumento da quantidade de dados adicionando cópias ligeiramente modificadas ou dados sintéticos recém-criados a partir de dados existentes, e *pooling* como conseguimos visualizar na Figura 2.20, (b) e (c) respetivamente. Isto através da diminuição do caminho percorrido da informação, realçando a informação refinada nas camadas superiores (os classificadores) e recuperando a informação perdida, PANet garante que importantes características não são perdidas (Parico e Ahamed 2021).

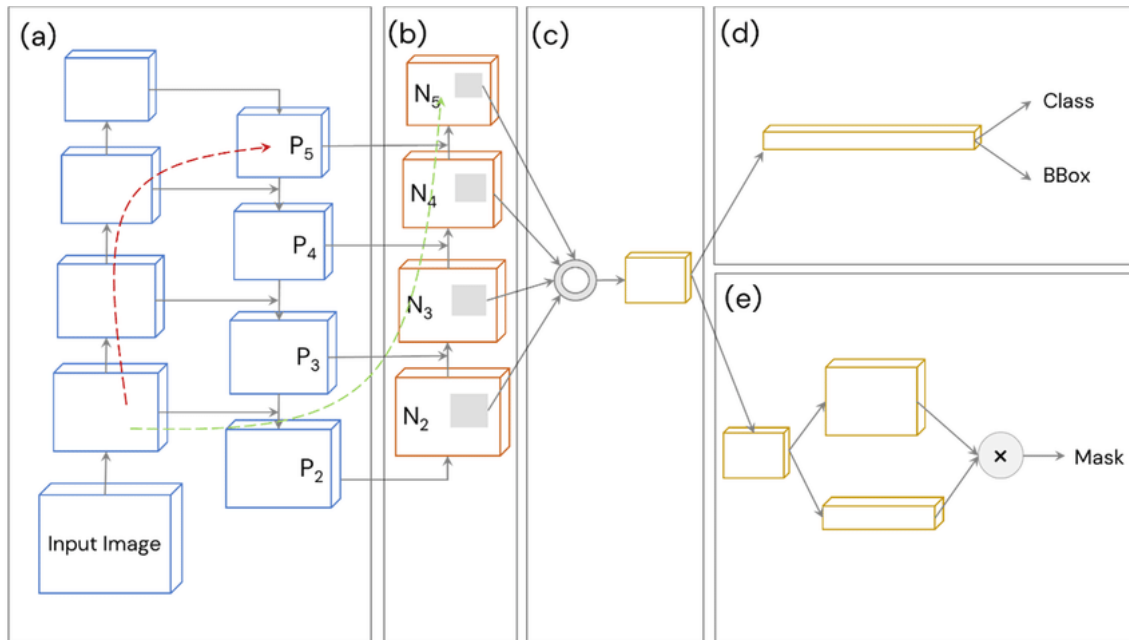


FIGURA 2.20: Arquitetura da PANet, inspirando a arquitetura da *Neck* no YOLOv4. (a) FPN backbone; (b) bottom-up path augmentation; (c) adaptive feature pooling; (d) box branch; (e) fully-connected fusion (concatenation is done instead of addition for YOLOv4), retirado de (Parico e Ahamed 2021).

Finalmente, foi escolhida a *Head* do YOLOv4, sendo esta a mesma do YOLOv3, usada para fazer a previsão de classes e *bounding boxes*.

Além desta arquitetura e com o objetivo de otimizar o treino, Bonchkovskiy, Wang e Liao aplicaram diferentes métodos e estratégias com a finalidade de aumentar a precisão do modelo e dos resultados no YOLOv4, denominados como *bag of freebies*

e *bag of specials*. Na *bag of freebies*, usaram métodos como *Data Augmentation* (e.g. Droplock regularization e Class label Smoothing), ou métodos para lidar com problemas de *data imbalance* existentes entre várias classes (e.g. focal loss). Na *bag of specials*, usaram métodos com o intuito de aumentar o custo de inferência, ampliando positivamente a precisão na deteção de objetos (e.g. SPP e DIoU).

Em termos de resultados, quando Bonchkovskiy, Wang e Liao compararam o YOLOv4 a outros detetores de objetos (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020), usando arquiteturas de GPU diferentes (Maxwell, Pascal e Volta), o YOLOv4 apresentou resultados superiores de acordo com os parâmetros velocidade (FPS) e precisão (AP), tendo por base a MS COCO *dataset*.

2.4 Trabalhos Relacionados

A deteção de objetos é um conceito importantíssimo quando se trata de operações militares e missões de reconhecimento. Algoritmos de deteção podem ser colocados em UAVs ou *Unmanned Ground Vehicles* (UGVs) de modo a recolher informações cruciais para a missão. No entanto, algoritmos de deteção só recentemente começaram a combater as lacunas que possuíam, em termos de cenários militares. Estas lacunas provinham da dificuldade em detetar objetos devido à sua dimensão, complexidade de cenários, distância do objeto aos sensores de deteção e uma fraca base de dados militar. Nesta secção vamos analisar alguns dos trabalhos e estudos realizados sobre este tema.

O trabalho publicado por Shuo Liu e Zheng Liu em 2017, *Multi-Channel CNN-Based Object Detection for Enhanced Situation Awareness* (Shuo Liu e Z. Liu 2017), realça o facto que naquela altura, algoritmos como o *Fast R-CNN*, melhoraram significativamente a performance na deteção de objetos quando treinadas e testadas em base de dados genéricas. Mas, a sua performance na deteção de objetos dimensionalmente reduzidos ou indistinguíveis era insuficiente. Com isto, os autores combinaram informação espacial, temporal e térmica através da implementação de vários canais de imagens, criando um algoritmo com o *Fast R-CNN* como *backbone* e a utilização da técnica de aprendizagem *cross-domain transfer* para aperfeiçoar o modelo na criação de múltiplos canais de imagens, arquitetura presente na Figura 2.21. No trabalho, os autores validam o algoritmo através de imagens da base de dados *Military Sensing Information Analysis Centre* (SENSIAC). Com resultados de 98.34% de AP e 98.90% de *accuracy*, pode-se dizer que o modelo tem um excelente desempenho. No entanto, verificou-se que o tempo de processamento

era consideravelmente longo para a utilização do modelo, na detecção de objetos em tempo real.

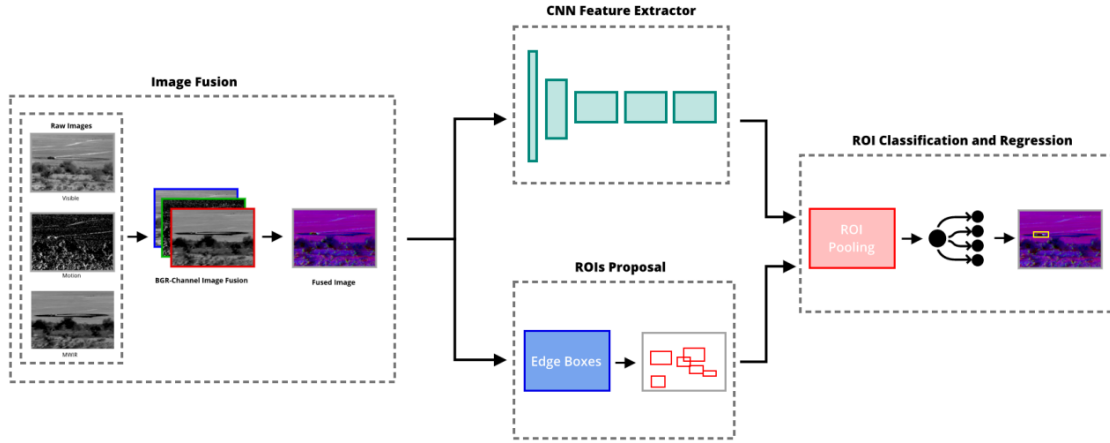


FIGURA 2.21: Arquitetura do modelo proposto por Shuo Liu e Zheng Liu em 2017, retirado de (Shuo Liu e Z. Liu 2017).

Em 2018, Hua et al. abordou o tema *Military Object Real-Time Detection Technology Combined with Visual Saliency and Psychology* (Hua, Xinqing Wang, D. Wang, Huang e X. Hu 2018), neste trabalho foi apresentado um método de detecção de objetos militares, que permitia alcançar uma detecção rápida e precisa, pela combinação da definição de saliência visual humana e psicologia visual. Para isto, Hua et al. estabeleceram um modelo de detecção de região saliente, baseado em canais duplos e fusão de características. O primeiro canal processa a informação relativa à posição e contraste das imagens e o segundo canal faz a integração de informação nas características visuais primárias, após isto, os dois canais interligam-se a fim de determinarem a região saliente. Podemos observar melhor esta arquitetura proposta pelos autores na Figura 2.22.

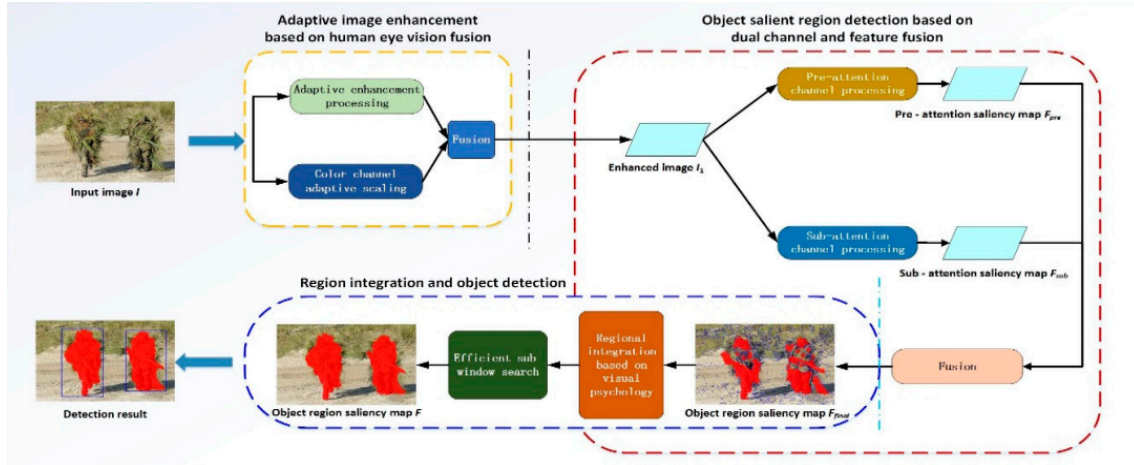


FIGURA 2.22: Ilustração da arquitetura de rede proposta por Hua et al., retirado de (Hua, Xinqing Wang, D. Wang, Huang e X. Hu 2018).

Os resultados experimentais mostram que, comparativamente a outros algoritmos, este modelo tem vantagens na precisão, eficácia e simplicidade, onde não só reduz a camuflagem do campo de batalha para uma melhor detecção do objeto, como alcança uma detecção rápida e precisa. Como é possível observar pela Tabela 2.1, a negrito estão os melhores resultados para cada coluna e *dataset*, onde este algoritmo superou os outros, em quatro dos cinco parâmetros, mas, novamente, fraquejando quando se trata em detecção em tempo real. Sento KITTI e MOD duas bases de dados diferentes.

Métodos	<i>Dataset</i>	Pf(%)	Pm(%)	Pd(%)	Pe(%)	Ptime(s)
Faster R-CNN	KITTI	11.21	13.34	60.32	15.13	0.076
	MOD	16.25	15.38	50.83	17.54	0.086
DSOD300	KITTI	14.48	15.91	63.38	6.23	0.017
	MOD	18.95	19.28	51.42	10.35	0.021
YOLO V2 544	KITTI	13.31	12.29	59.84	14.56	0.022
	MOD	15.17	15.49	49.45	19.89	0.026
DSSD	KITTI	9.53	10.69	66.25	13.53	0.018
	MOD	16.24	13.19	55.16	15.41	0.028
Hua et al. algoritmo	KITTI	4.19	3.13	90.47	2.21	0.106
	MOD	6.16	5.37	82.05	6.42	0.185

TABELA 2.1: Comparação dos resultados do algoritmo de Hua et al. com diferentes métodos de detecção em duas datasets diferentes, adaptado de (Hua, Xinqing Wang, D. Wang, Huang e X. Hu 2018).

Em 2019, Ramalho explorou no seu trabalho, *Feature Extraction and Object Classification in Video Sequences for Military Surveillance* (Ramalho 2019), o desenvolvimento de um sistema altamente especializado para identificar classes limitadas e particulares, neste caso, focando-se no uso militar, carros de combate, armas, pessoas e veículos, alcançando vigilância territorial. Este trabalho apoiou-se na utilização de um detetor de um estágio, SSD, combinado com a utilização da biblioteca TensorFlow. Apesar da mitigação de alguns erros ao longo do estudo do trabalho, o modelo final apresentou diversidades ao objetivo proposto inicialmente, nomeadamente, objetos que o modelo detetava deixaram de ser detetados, isto porque, quando treinou a rede singularmente com imagens de objetos militares, a mesma apenas começou a detetar estas classes. Verificou um *Overfitting* no modelo, onde complexidade da arquitetura da rede usada não se interligava com a dimensão da base de dados, sendo esta muito reduzida.

Em 2020, Jiang, Zhaq, Li e Jia, focaram o seu trabalho no tema *Real-time object detection method for embedded devices* (Jiang, L. Zhao, Li e Y. Jia 2020). Neste trabalho, abordaram um tema muito relacionado com um dos objetivos do nosso trabalho, propondo um método rápido na deteção de objetos tendo como estrutura a rede YOLOv4-tiny, estrutura simplificada do YOLOv4, adequada para ser integrada em sistemas embebidos ou móveis. Nos resultados, melhoraram o método YOLOv4-tiny em termos de estrutura da rede, que em comparação com outras redes, teve resultados mais rápidos e quase o mesmo valor de mAP do YOLOv4-tiny inicial, como é possível visualizar na Tabela 2.2.

Métodos	FPS	mAP(%)
YOLOv3 R-CNN	49	52.5
YOLOv4	41	64.9
YOLOv3-tiny	277	30.5
YOLOv4-tiny	270	38.1
Método Proposto	294	38.0

TABELA 2.2: Comparação do modelo Jiang et al. com outros modelos, adaptado de (Jiang, L. Zhao, Li e Y. Jia 2020).

No mesmo ano, Surrisyad e Wahyono abordaram o tema *A Fast Military Object Recognition using Extreme Learning Approach on CNN* (Surrisyad et al. 2020), onde propuseram a combinação do uso de uma CNN e *Extreme Learning Machine* (ELM), de forma a colmatarem o longo tempo de treino de uma CNN, posteriormente treinaram e testaram o algoritmo com objetos militares.

No ano de 2021, Akbulut e Khalaf publicaram um trabalho, correlacionando a ideia de controlo, monitorização de armamento e a utilização do algoritmo YOLOv3 (Akbulut e Khalaf 2021). Neste estudo os autores recolheram 825 imagens da arma AK-47 e fizeram a sua respetiva anotação (*bounding boxes*). Os resultados obtidos foram satisfatórios quando testaram a rede com imagens extraídas do Google, imagens claras da arma. No entanto, para que um sistema inteligente de deteção de armas seja eficaz, os autores têm a perceção que será necessário instituições, departamentos, companhias de segurança etc., 'agarrem' nesta simples ideia e a ampliem.

No mesmo ano Janakiramaiah et al. propuseram o trabalho *Military object detection in defense using multi-level capsule networks* (Janakiramaiah, Kalyani, Karuna, Prasad e Krishna 2021). Aqui, apontaram o problema de que CNNs, estruturas boas para realizarem a deteção de objetos, apresentavam recorrentemente um problema de invariável de localização, apoiavam-se muito na dimensão da base de dados para treino. Onde, geralmente, há pouca informação e imagens militares disponíveis graças à sua confidencialidade e problemas de segurança. O projeto baseou-se numa arquitetura de rede neuronal, recente, a *Capsule Network* (CapsNet) e numa base de dados reduzida, retirada da Internet, conseguimos ver a arquitetura desta rede pela Figura 2.23.

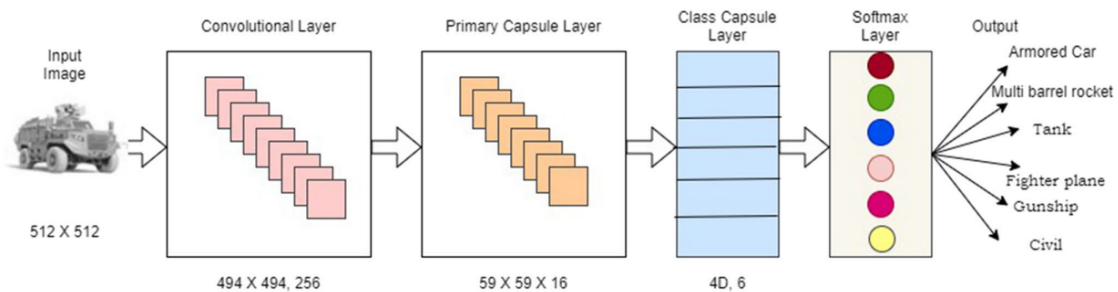


FIGURA 2.23: Arquitetura da CapsNet para a deteção de objetos militares, retirado de (Janakiramaiah, Kalyani, Karuna, Prasad e Krishna 2021).

Alcançou bons resultados em termos de precisão, na Tabela 2.3 é possível aferir estes resultados quando comparado a outros dois métodos, SVM e CNN+TL.

Classes	Métricas (%)	SVM	CNN+TL	Proposto
Armored car	Precision	69.89	87.68	91.94
	Recall	54.17	72.33	83.67
Multi-barrel Rocket	Precision	61.79	73.16	89.02
	Recall	50.67	59.5	86.5
Tank	Precision	66.12	77.02	89.33
	Recall	60.83	82.67	90.67
Fighter Plane	Precision	61.09	73.82	86.91
	Recall	56	65.33	83
Gunship	Precision	64.41	86.06	88.57
	Recall	60.33	76.17	81.33
Civil	Precision	33.9	54.62	71
	Recall	59.6	88.6	90.6

TABELA 2.3: Comparação das métricas *recall* e *precision* do algoritmo de Janakiramaiah et al. com dois outros métodos diferentes, adaptado de (Janakiramaiah, Kalyani, Karuna, Prasad e Krishna 2021).

Capítulo 3

Metodologia

De forma a realizar a deteção automática de alvos terrestres a partir de uma rede neuronal, primeiramente temos que a escolher. Como foi possível constatar no capítulo anterior, diversas redes poderiam ter sido a nomeada para este efeito, quer pela sua precisão, simplicidade, rapidez, etc. Mas, olhando para o panorama geral, limitações, recursos computacionais e o objetivo do trabalho, a escolha recaiu sobre a rede YOLO, mais propriamente a versão YOLOv4-tiny, que mostrou ter uma relação velocidade/precisão/simplicidade superior, quando comparada com outras redes do mesmo estilo. Neste capítulo, portanto, iremos abordar a razão pela qual decidimos alterar a rede YOLOv4 para a versão mais reduzida, YOLOv4-tiny, a instalação da rede e uso da mesma no nosso computador, como adquirimos as imagens para a nossa base de dados, as métricas que vamos utilizar na avaliação do nosso modelo e por fim, como procedemos ao treino da rede.

3.1 YOLOv4-Tiny

Como uma das metas a alcançar do trabalho é, no final, usar esta rede treinada e integrá-la numa plataforma tipo UAV, chegou-se ao consenso de mudar a rede neuronal escolhida, YOLOv4 para a YOLOv4-Tiny.

A razão desta mudança, teve presente o facto de o YOLOv4-Tiny ser uma versão reduzida do YOLOv4, criado para ser usado em máquinas com menos potência computacional, como é o caso do computador pessoal ou UAVs. Permitindo realizar o treino da rede a uma velocidade mais elevada e quando utilizada efetivamente na deteção de objetos, fazer uso de menos potência computacional. A sua arquitetura apresenta algumas diferenças para que haja este ganho de velocidade, nomeadamente o número de camadas convolucionais pré-treinadas, tendo apenas 29 camadas. Adicionalmente, o número de camadas YOLO foram reduzidas a duas, contrariamente a três do YOLOv4.

Como é possível verificar pela Figura 3.1, estas alterações permitiram aumentar substancialmente os FPS da rede, e mesmo alcançar uma *mean Average Precision* (mAP) de 40 (Techzizou 2021b).

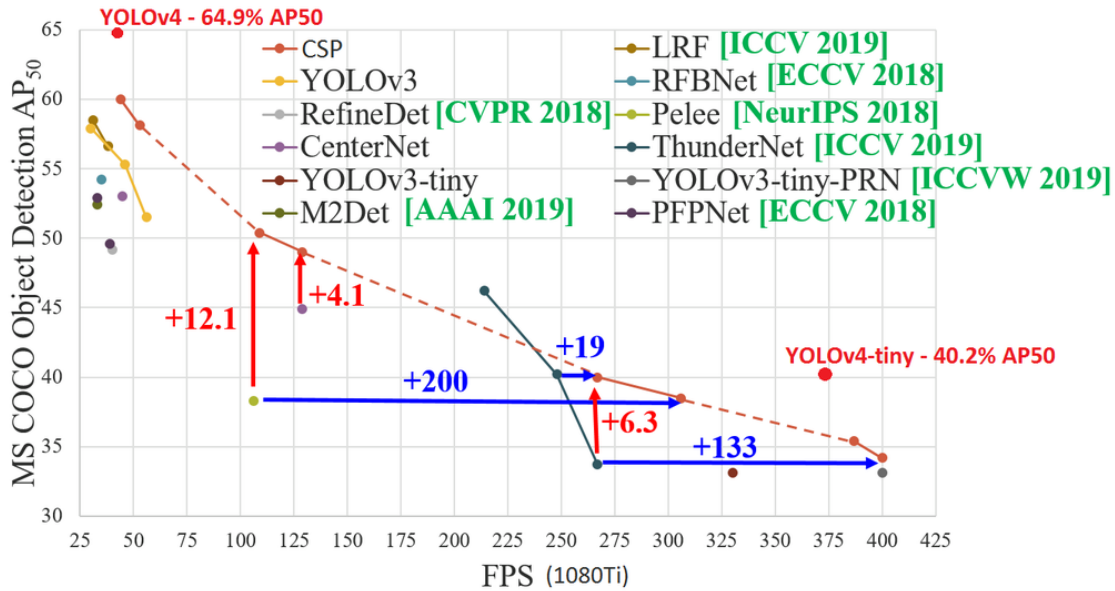


FIGURA 3.1: Comparação do YOLOv4 com o YOLOv4-Tiny, retirado de (Aleksey Bochkovskiy 2020).

Para realizar a detecção de alvos terrestres, necessitamos de implementar a rede de detecção de objetos YOLOv4-tiny no nosso computador local.

De modo a tal ser possível, utilizamos a biblioteca *Open Source Computer Vision Library* (OpenCV), livre ao acesso académico e comercial no âmbito da Visão Computacional, *Machine Learning* e processamento de imagens, permitindo aplicar a rede, às imagens e vídeos que iremos utilizar no teste da mesma. Para o treino da nossa rede personalizada, implementámos a Darknet de Alexey Bochkovskiy et al. (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020), uma estrutura de rede neuronal, de livre acesso igualmente, com capacidade de suporte para CPU e GPU, como é o caso do nosso computador local.

Após a instalação bem sucedida do YOLOv4-tiny, utilizamos a base de dados *Microsoft Common Objects in Context* (MS COCO), com mais de 200 000 imagens anotadas e 80 classes distintas, de modo a identificar a performance da rede YOLOv4-tiny no nosso computador. Na Figura 3.2, recolhida pelo drone pessoal DJI Mavic Air 2S, a rede detetou a pessoa e a mochila. Na Figura 3.3, *frame* de um vídeo gravado na Lituânia pelos Fuzileiros, foram detetadas pessoas e outras pessoas como cavalos, provavelmente pelo equipamento que transportam.



FIGURA 3.2: Primeira deteção com o YOLOv4-tiny no computador, detetando uma mochila e uma pessoa, recolhida pelo drone pessoal DJI Mavic Air 2S.



FIGURA 3.3: Corte da primeira deteção de um vídeo com militares no YOLOv4-tiny, no computador.

3.2 Base de Dados

Antes de começarmos a realizar o treino da nossa rede personalizada, como pretendemos alcançar bons resultados de teste, combatendo o *overfitting*, temos de recolher e adquirir uma extensiva compilação de vídeos e imagens de militares, para

que, manualmente, seja definido o que é caracterizado como um militar, desenhas *bounding boxes* em cada objeto individualmente, extrair a informação de cada anotação e posteriormente colocar na pasta da rede para o treino ser realizado.

Para a anotação das imagens, utilizamos uma ferramenta gratuita de anotação de vídeos e imagens, DarkLabel2.4¹, este modelo contém vários formatos de extração de imagens anotadas, incluindo a nossa de interesse, em formato YOLO.

Nesta fase inicial do trabalho, recolha de imagens e vídeos, o total de imagens recolhidas são de 732 imagens provenientes da Internet e 76 minutos de vídeo total, capturados por drones comerciais. Podemos diferenciar esta base de dados em seis bases de dados mais específicas distintas: Base de dados militar específica 1 (BDME1), base de dados militar específica 2 (BDME2), base de dados militar específica 3 (BDME3), base de dados militar específica 4 (BDME4), base de dados militar/civil específica (BDMCE) e base de dados civil específica (BDCE).

A BDME1, são imagens de militares retiradas da Internet, um total de 181 imagens, contendo em média 5 objetos por imagem. Podemos observar alguns exemplos do seu conteúdo pela Figura 3.4.



FIGURA 3.4: Amostras de imagens da base de dados militar específica 1, retiradas da Internet.

¹<https://github.com/darkpgmr/DarkLabel>

A BDME2, são imagens de cenários de treino, realizadas pelos Fuzileiros Portugueses na Lituânia, recolhidas pelo drone Mavic Air 2 FLIR, contendo em média 12 objetos por imagem, num total de 40 minutos de vídeo. As especificações da sua câmara estão representadas na Tabela 3.1. Podemos observar alguns exemplos, do conteúdo desta base de dados, pela Figura 3.5.



FIGURA 3.5: Amostras de imagens da base de dados militar específica 2, provenientes de cenários de treino realizados pelos Fuzileiros Portugueses na Lituânia.

Mavic Air 2 FLIR	
Sensor	1/2"CMOS
Field of View (FOV)	84°
Aperture	f/2.8
Shooting Range	1 m to ∞
ISO Range	100-6400
Video Resolution	4K Ultra HD: 3840×2160 24/25/30/48/50/60 fps
Observations	Forward Looking InfraRed (FLIR)

TABELA 3.1: Especificações da câmara do drone Mavic Air 2 FLIR usado na Lituânia.

A BDME3, são imagens de cenários do exercício Tróia, realizado pelos cadetes da Escola Naval em 2021, recolhidas pelo drone DJI Mini 2, contendo em média

31 objetos por imagem, num total de 10 minutos de vídeo. As especificações da sua câmara estão representadas na Tabela 3.2. Podemos observar alguns exemplos, do conteúdo desta base de dados, pela Figura 3.6.



FIGURA 3.6: Amostras de imagens da base de dados militar específica 3, provenientes de cenários do exercício Tróia 2021, realizado pelos cadetes da Escola Naval.

DJI Mini 2	
Sensor	1/2.3"CMOS
Field of View (FOV)	83°
Aperture	f/2.8
Shooting Range	1 m to ∞
ISO Range	100-3200
Video Resolution	4K: 3840×2160 @ 24/25/30 fps

TABELA 3.2: Especificações da câmara do drone DJI Mini 2, usado no exercício Tróia da Escola Naval, em 2021.

A BDME4, são imagens de cenários do exercício Tróia, realizado pelos cadetes da Escola Naval em 2022, recolhidas pelo drone DJI Air 2S, contendo em média 13 objetos por imagem, num total de 26 minutos de vídeo. As especificações da sua câmara estão representadas na Tabela 3.3. Podemos observar alguns exemplos, do conteúdo desta base de dados, pela Figura 3.7.



FIGURA 3.7: Amostras de imagens da base de dados militar específica 4, provenientes de cenários do exercício Tróia 2022, realizado pelos cadetes da Escola Naval.

DJI Air 2S	
Sensor	1"CMOS
Field of View (FOV)	88°
Aperture	f/2.8
Shooting Range	0.6 m to ∞
ISO Range	100-6400
Video Resolution	5.4K: 5472×3078 @ 24/25/30 fps

TABELA 3.3: Especificações da câmara do drone DJI Air 2S, usado no exercício Tróia da Escola Naval, em 2022.

A BDMCE, contém imagens de militares juntamente com civis retiradas da Internet, um total de 499 imagens, contendo em média 7 objetos por imagem. Podemos observar alguns exemplos do seu conteúdo pela Figura 3.8.



FIGURA 3.8: Amostras de imagens base de dados militar/civil específica, retiradas da Internet.

A BDCE, contém imagens de civis retiradas da Internet, um total de 52 imagens, contendo em média 2 objetos por imagem. Podemos observar alguns exemplos do seu conteúdo pela Figura 3.9.

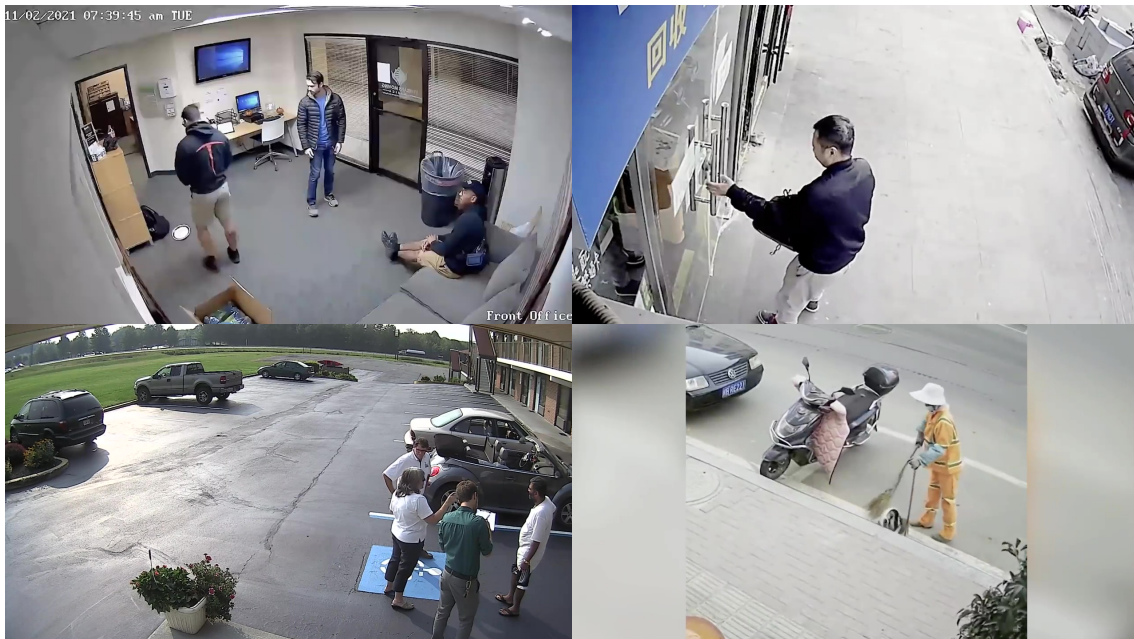


FIGURA 3.9: Amostras de imagens da base de dados civil específica, retiradas da Internet.

De referir que a BDME1, BDMCE e BDCE apresentam um problema, não são vistas aéreas, provenientes de um drone. Este defeito irá afetar bastante o desempenho da rede se inserirmos a mesma num UAV com o propósito de apoiar forças militares. No entanto, devido à dificuldade e limitação de encontrar imagens variadas, gravadas via aérea, inserimos igualmente estas imagens na nossa base de dados, num objetivo de prova de conceito.

Resumidamente, a nossa base de dados, num todo, é composta por bases de dados específicas, de acordo com a Tabela 3.4.

Base de Dados			
BD específica	Plataforma	Nº médio de objetos	Imagens/vídeo
BDME1	Internet	5	181 imagens
BDME2	Mavic Air 2 FLIR	12	40min de vídeo
BDME3	DJI Mini 2	31	10min de vídeo
BDME4	DJI Air 2S	13	26min de vídeo
BDMCE	Internet	7	499 imagens
BDCE	Internet	2	52 imagens

TABELA 3.4: Constituição da base de dados total.

Com a nossa base de dados já materializada, o próximo passo será realizar a respetiva anotação de cada imagem, configurar a rede de modo a que consigamos treiná-la e obter resultados. No decorrer deste processo verificámos que não conseguíamos, nem seria vantajoso realizar a anotação de todos os vídeos da nossa base de dados. A Tabela 3.5 apresenta a informação relativa ao número de imagens anotadas para cada base de dados específica, quer no conjunto de treino e no de teste.

Imagens anotadas		
BD específica	Treino	Teste
BDME1	116	65
BDME2	1 726	-
BDME3	-	-
BDME4	1 594	-
BDMCE	447	52
BDCE	-	52

TABELA 3.5: Divisão e número das imagens anotadas para treino e para teste.

Pela Tabela 3.5, conseguimos notar alguns pontos importantes a serem mencionados. Não utilizámos a BDME3 neste trabalho pois os objetos nela presentes eram militares com mochilas grandes e, como tínhamos poucos vídeos e imagens de objetos semelhantes, decidimos não implementar no nosso trabalho para não influenciar negativamente a rede.

Notamos também que não temos imagens de teste da BDME2 e BDME4, as bases de dados mais importantes neste trabalho, preferimos colocar as imagens das mesmas quase todas na secção do treino, as outras que sobraram mesmo em circunstâncias de operação diferentes, eram bastante semelhantes às utilizados no treino, optámos por não as utilizar. No entanto, está planeado ainda no final deste trabalho, conseguirmos realizar duas gravações distintas com o drone, a utilizar na fase de teste. Uma com elementos militares e civis, e outra com um elemento civil e outro militar.

Vemos também que não utilizamos a BDCE para treino da rede, isto porque o foco do nosso trabalho não está assente na deteção de civis, mas sim na deteção de militares.

Nas próximas secções, iremos abordar as métricas na avaliação do desempenho da rede, bem como a constituição verdadeiramente de três bases de dados de estudo, provenientes das bases de dados específicas mencionadas na Tabela 3.4.

3.3 Métricas na avaliação do desempenho da rede

De modo a validar e perceber certos parâmetros que serão apresentados e discutidos nas próximas secções e capítulos do trabalho, nesta secção examinamos

diferentes métricas de performance de um detetor de objetos.

Uma detecção, é representada por um conjunto de três atributos, a classe do objeto, a *bounding box* correspondente e o índice de confiança, valor entre 0 e 1.

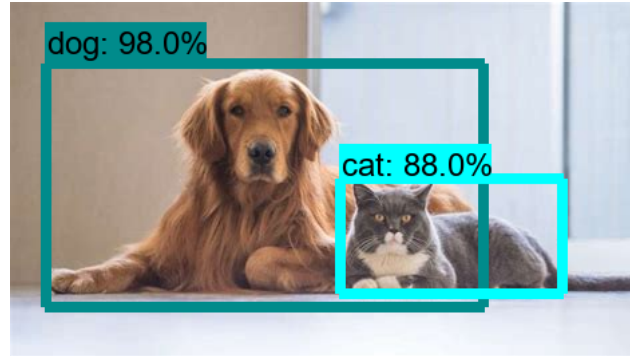


FIGURA 3.10: Três atributos em detecção de objetos (Classes, *bounding box* e *confidence score* [0 a 1], retirado de (Team 2021).

A primeira métrica a ser apresentada, neste campo, é a *Intersection over Union* (IOU), visível na Figura 3.11.

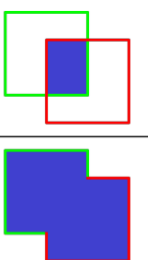
$$\text{IOU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{área de sobreposição}}{\text{área da união}}$$
A diagram illustrating the IOU metric. It shows two overlapping rectangles: a blue one and a red one. The intersection of the two rectangles is shaded in green. Below the rectangles, a large green shape represents the union of the two rectangles.

FIGURA 3.11: Representação da *Intersection over Union* (IOU), retirado de (Padilla, Passos, Dias, Netto e Silva 2021).

IOU é igual à área de sobreposição entre a *bounding box* prevista e a *bounding box* verdadeira, dividida pela área total da sua união. A combinação perfeita ocorre quando $\text{IOU} = 1$, se as duas *bounding boxes* não se intercedarem, $\text{IOU} = 0$. Quanto mais próximo o valor do IOU for do 1, melhor é considerada a detecção.

Outras duas métricas são a *precision* e o *recall*. De modo a perceber os seus valores, existem três definições que temos de abranger, *True Positive* (TP), *False Positive* (FP) e *False Negative* (FN). Como se verifica na Figura 3.12, um TP define-se como uma detecção correta de uma *bounding box* verdadeira. Um FP como uma detecção incorreta de um objeto não presente ou uma detecção mal posicionada de um objeto presente, isto dependendo do *threshold* determinado, como é o caso

de 0.5 na imagem. Por último, um FN corresponde à não detecção de uma *bounding box* verdadeira (Padilla, Netto e Da Silva 2020).

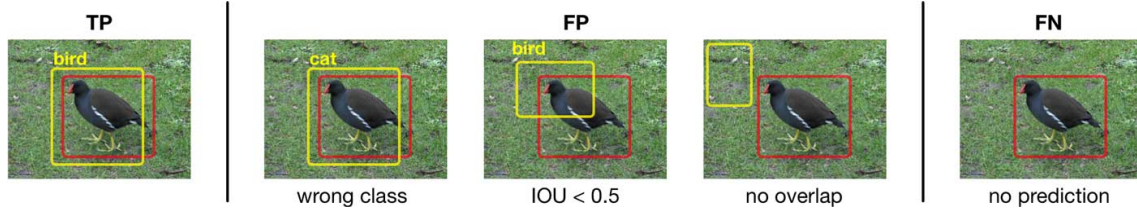


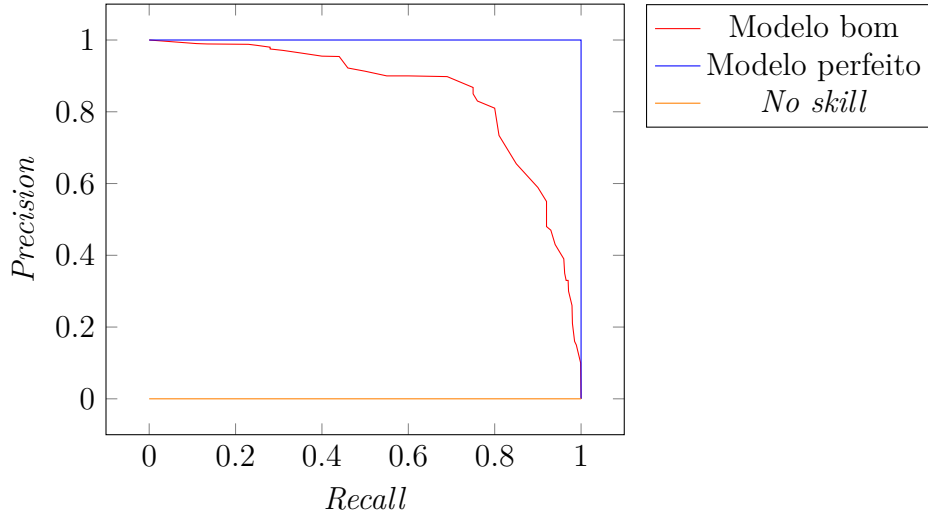
FIGURA 3.12: Diferenciação entre TP, FP e FN, retirado de (Hollemans 2018).

A *precision*, representada pela equação 3.1, faz a divisão dos TP com a soma dos TP mais os FP. Enquanto o *recall*, equação 3.2, divide os TP com a soma dos TP mais os FN.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \quad (3.1)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \quad (3.2)$$

A curva exemplificada no Gráfico 2.1, é uma boa forma de avaliar o desempenho de um detetor de objetos. Um detetor de objetos é considerado bom quando a *precision* se mantém alta enquanto o *recall* aumenta (modelo bom), o que significa que conseguindo fornecer um determinado *threshold* ideal à rede, ela consegue balancear a *precision* e o *recall* para detetar o mínimo de FP e FN.

Gráfico 2.1. Curva *Precision VS Recall* relativa ao *threshold*

Com base na curva *Precision x Recall*, outra métrica importante é retirada, a *Average Precision* (AP), esta é calculada pela *Area Under the Curve* (AUC) da curva referida anteriormente. Na prática, os valores para a AP variam de 0 a 1.

O cálculo da AP refere-se apenas a uma classe, no entanto, na detecção de objetos, geralmente existem K classes. Aqui é inserida outra métrica, a *mean Average Precision* (mAP), que calcula a precisão do detetor de objetos sobre todas as classes (Padilla, Netto e Da Silva 2020). Sendo AP_i a AP nas i classes e N o número total de classes a serem avaliadas, a expressão 3.3, retirada de (Padilla, Netto e Da Silva 2020), define a mAP.

$$mAP = \frac{1}{K} \sum_{i=1}^K AP_i \quad (3.3)$$

3.4 Treino

Com as bases de dados distintas adquiridas e a anotação feita das imagens, procedemos ao treino da rede. Para isto, juntámos as bases de dados diferentes e delimitamos três grandes bases de dados para realizarmos os nossos treinos, obter os melhores resultados e testarmos esses resultados, estas são: a base de dados experimental (BDMEX), a base de dados militar (BDM) e a base de dados militar/civil (BDMC). Conseguimos observar esta divisão na Figura 3.13.

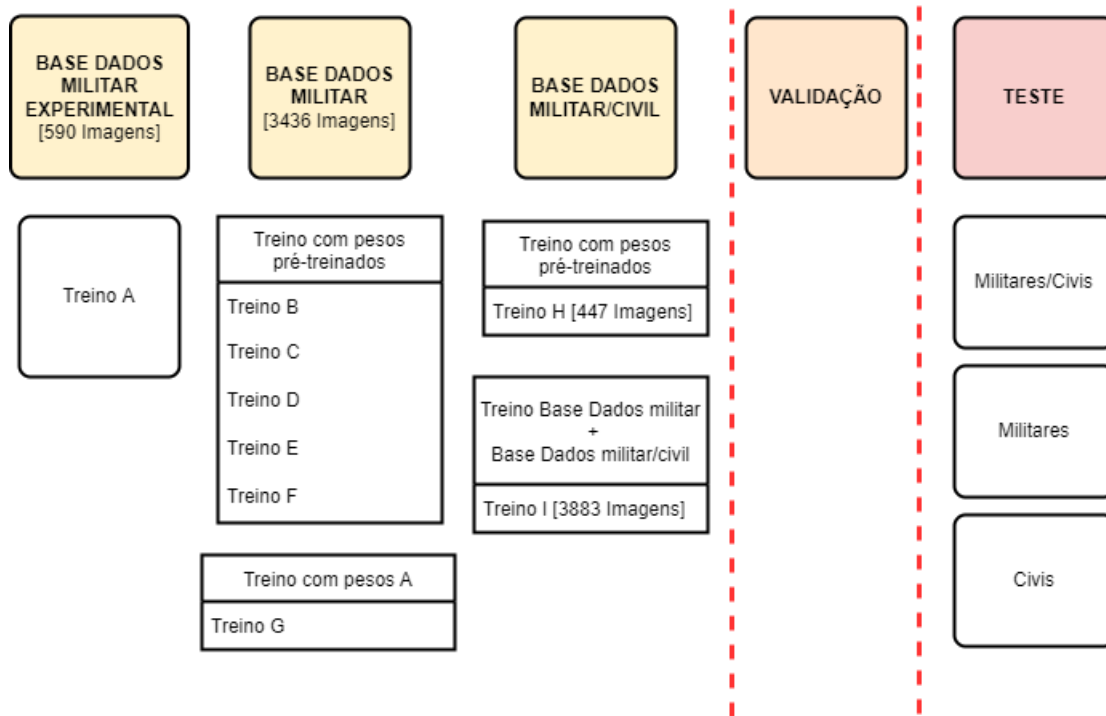


FIGURA 3.13: Estrutura de treino, validação e teste da rede.

A Tabela 3.6 caracteriza pormenorizadamente a divisão das bases de dados específicas entre as bases de dados descritas na Figura 3.13, bem como o número de imagens contidas em cada uma delas. Onde a BDMEX é usada como primeiros resultados, ver previamente como é que o treino está a correr, mesmo com poucas imagens. A BDM já com objetivo de ter um bom desempenho na deteção de militares. A BDMC com a finalidade de fazer com que o modelo diferencie militares de civis. Esta organização das bases de dados, os dados de validação e de teste são possíveis de ver e extrair num link para a Drive².

²<https://drive.google.com/drive/folders/1fF-YESD4565NmvpdaBgvkDcNlFQG-LK5?usp=sharing>

Fase de Treino		
Base Dados	Base Dados Específica	Imagens
Militar Experimental	BDME1	116
	BDME2	475
Militar	BDME1	116
	BDME2	1 726
	BDME4	1 594
Militar/Civil	BDMCE	447
Militar/Civil + Militar	BDME1	116
	BDMCE	447
	BDME2	1 727
	BDME4	1 594

TABELA 3.6: Fase de treino e a sua constituição.

Após termos as bases de dados materializadas, organizámos os diferentes treinos para as mesmas. Esta organização, de uma forma mais pormenorizada, está contida na Tabela 3.7. Aqui vemos que além das diferentes bases de dados, também alterámos diversos hiperparâmetros da nossa rede, com o objetivo de encontrar a melhor configuração para alcançar o melhor desempenho.

Importante referir que, para não treinar o modelo de raiz, fizemos o que se designa por aprendizagem por transferência. Para tal, transferimos e utilizámos os pesos YOLOv4-tiny pré-treinados até 29 camadas convolucionais, treinados com a base de dados MS COCO, de (Techzizou 2021a). À exceção do treino G, onde utilizámos os pesos provenientes do treino A.

Treino	<i>Batch</i>	Subsivisões	Iterações	Base Dados	Pesos
A	64	16	6000	Experimental	Pré-treinados
B	64	16	6000	Militar	Pré-treinados
C	32	08	7000	Militar	Pré-treinados
D	96	16	6000	Militar	Pré-treinados
E	96	32	6000	Militar	Pré-treinados
F	128	16	6000	Militar	Pré-treinados
G	128	16	6000	Militar	Pesos A
H	128	16	6000	Militar/Civil	Pré-treinados
I	128	16	6000	Militar + Civil	Pré-treinados

TABELA 3.7: Treinos a realizar.

A partir disto, iremos expor como realizámos a configuração da nossa rede personalizada YOLOv4-tiny para obter os primeiros resultados. Seguidamente, as alterações para a base de dados militar. Por fim, como procedemos ao treino da rede com a base de dados contendo militares e civis, de forma a conseguir distinguir estas duas classes na mesma *frame* de vídeo.

3.4.1 Base de Dados Militar Experimental

Após termos um número considerável de imagens anotadas para obter resultados de treino, nomeadamente 590 imagens com as suas respetivas coordenadas, procedeu-se à configuração da nossa rede, seguindo um tutorial de como treinar um detetor de objetos personalizado no Windows (Techzizou 2021a).

Esta configuração, após alguns passos, centrou-se no importante para a realização do treino da rede, nomeadamente a alteração de alguns dos hiperparâmetros do ficheiro yolov4-tiny-custom na pasta *cfg* da *darknet*, deixando muitos destes valores padrão. Os hiperparâmetros foram os *batches*, as subdivisões, *network size width* e *height*, o número de iterações, os *steps* e os filtros.

O valor do *batch* alterado para 64, as subdivisões para 16, a *network size width* e *height* para 416, o número máximo de *batches*, ou iterações, para 6000 e os

steps para 4800 e 5400 respetivamente, 80% e 90% do número máximo de *batches*. O número de filtros das camadas convolucionais, modificados, de acordo com a expressão retirada de (Techzizou 2021a):

$$Filtros = (classes + 5) \times 3 \quad (3.4)$$

Como a nossa classe em estudo para esta base de dados experimental é ‘militar’, uma classe, o valor dos filtros foi alterado para 18. Alterámos a linha das classes de 80, provenientes da base de dados MS COCO, para o número das nossas classes, que é 1, antes das duas camadas YOLO. Após realizarmos estas alterações, procedemos ao treino da nossa base de dados experimental.

É de referir que um importante hiperparâmetro a ter em consideração é a *data augmentation*, esta técnica apresenta-se como uma solução para o problema de dados limitados. A *data augmentation* realiza uma ampliação dos dados e engloba um conjunto de técnicas que melhoram o tamanho e a qualidade dos conjuntos de dados, com o objetivo de capacitar o modelo com um maior e mais variado número de dados e consequentemente informação, resultando numa melhor aprendizagem pelo modelo e desempenho na sua utilização (Shorten e Khoshgoftaar 2019). Nesta fase inicial de experimentação, deixámos por defeito o YOLOv4-tiny realizar por si a *data augmentation* que está pré-definido a fazer. Modificando e ampliando as imagens dados no conjunto de treino em saturação, exposição e tonalidade de imagem, com os valores de 1.5, 1.5 e 0.10 respetivamente.

Finalmente, o treino experimental da nossa base de dados, representado na Tabela 3.8, deu-se. Os resultados são possíveis de serem observados no próximo capítulo.

Treino	Batch	Subsivisões	Iterações	Steps	Pesos
A	64	16	6000	4800/5400	Pré-treinados

TABELA 3.8: Treino experimental da BD Militar com pesos pré-treinados.

3.4.2 Base de Dados Militar

Para completar esta base de dados, foram anotadas ao todo 3436 imagens de militares. Após a colocação das *bounding boxes* nos respetivos militares, seguiu-se

a divisão dos vários treinos como as tabelas 3.9 e 3.10 evidenciam. O treino G teve como intuito perceber se a rede com mais imagens a treinar com base nos pesos do treino A, teria melhores resultados do que com os pesos pré-treinados.

Treino	Batch	Subsivisões	Iterações	Steps	Pesos
B	64	16	6000	4800/5400	Pré-treinados
C	32	08	7000	5600/6300	Pré-treinados
D	96	16	6000	4800/5400	Pré-treinados
E	96	32	6000	4800/5400	Pré-treinados
F	128	16	6000	4800/5400	Pré-treinados

TABELA 3.9: Treino da BD Militar com pesos pré-treinados.

Treino	hiperparâmetros conf.	Pesos
G	Iguais ao treino F	Adquiridos do treino A

TABELA 3.10: Treino da BD Militar com os pesos adquiridos de treinos anteriores.

3.4.3 Base de Dados Militar/Civil

Com as imagens anotadas da base de dados militar/civil, utilizaram-se as melhores configurações aferidas no treino da base de dados militar. Dividiu-se o treino em duas fases distintas. A primeira, exposta na Tabela 3.11, é apresentado o treino H, com a configuração dos hiperparâmetros iguais ao treino F e 447 imagens. Foram utilizados os pesos pré-treinados.

Treino	hiperparâmetros conf.	Observações
H	Iguais ao treino F	BD militar/civil (447 imagens)

TABELA 3.11: Treino da BD Militar/Civil com os pesos pré-treinados.

A segunda fase do treino, representada na Tabela 3.12, pelo treino I, junta o total de imagens da base de dados militar e da base de dados militar/civil, com a configuração dos hiperparâmetros iguais ao treino F.

3.4. Treino

Treino	hiperparâmetros conf.	Observações
I	Iguais ao treino F	BD Militar + BD Militar/Civil (3883 imagens)

TABELA 3.12: Treino da BD Militar/Civil com os pesos pré-treinados, utilizando a BD Militar e a BD Militar/Civil em conjunto.

Os resultados provenientes destes treinos serão apresentados no Capítulo 3.

Capítulo 4

Apresentação, Tratamento e Discussão dos Resultados

O potencial do modelo proposto é demonstrado neste capítulo, aqui vamos apresentar os resultados derivados do treino da rede na fase de validação. Seguidamente, iremos aferir a credibilidade e a performance do modelo quando testado com imagens e/ou vídeos, todos estes distintos dos que foram usados na fase de validação, acontecendo no subcapítulo Teste. O mesmo será dividido em três partes, o teste a militares e civis, apenas a militares e apenas a civis. No final, iremos discutir estes resultados, limitações e trabalhos futuros.

4.1 Validação

Após realizarmos o treino descrito na secção anterior, obtivemos os resultados de validação do modelo. Nesta secção, vamos expor e analisar esses resultados, o impacto da configuração de certos hiperparâmetros no treino, o desempenho do modelo ao longo do treino e o impacto do aumento e diferenciação da base de dados. O objetivo é aferir e escolher os melhores pesos a utilizar para o conjunto de teste, delimitando o melhor modelo para conseguirmos detetar militares ou diferenciar militares de civis.

Com o objetivo de percebermos a divisão das imagens por cada treino nesta fase de validação, a Tabela 4.1 apresenta esta divisão. Bem como a multiplicação automática das imagens feita pela rede na fase do treino, devido à *data augmentation* pré-definida, como já mencionado anteriormente.

Fase de Validação			
Treino	Imagens	Treino	Validação (10% das Imagens)
A	590	12 459	59
B - G	3436	76 486	344
H	447	10 075	45
I	3883	86 536	389

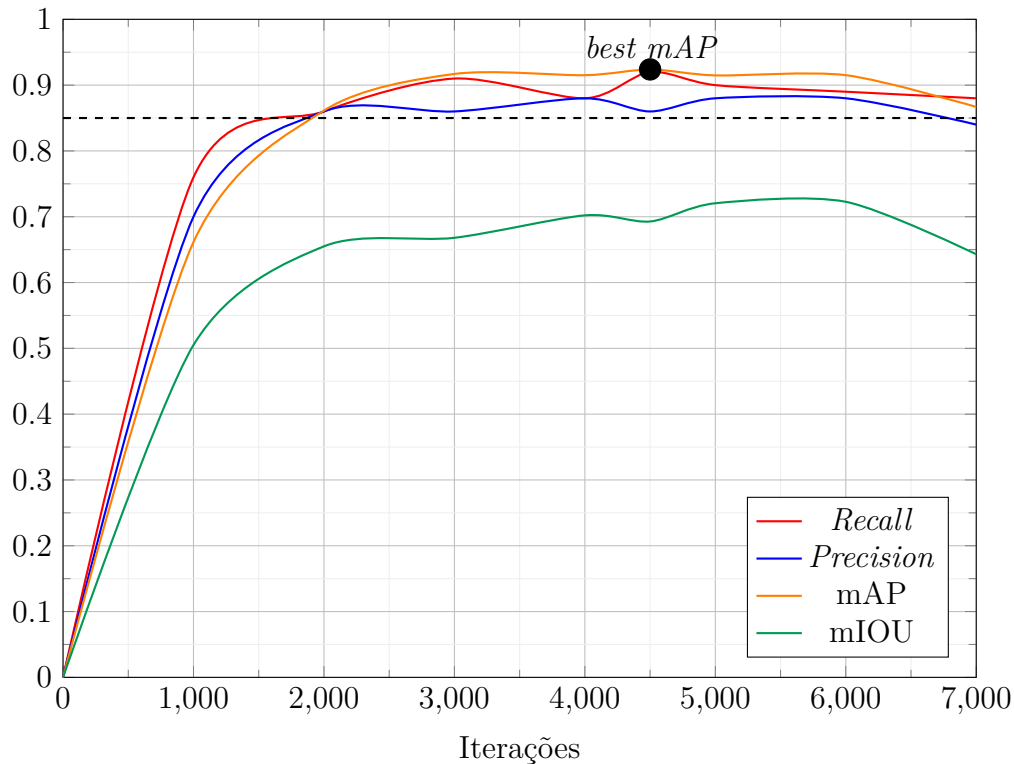
TABELA 4.1: Número de imagens utilizadas para cada treino em específico, desde a fase do treino até à fase da validação.

4.1.1 Base de Dados Militar Experimental

Como já foi referido, esta base de dados foi definida como a experimental, pouco diferenciada no seu conteúdo, apenas com 590 imagens. Usada unicamente para observar o comportamento da rede com poucas imagens.

Devido à *data augmentation* automática do YOLOv4-tiny, as 590 imagens anotadas foram multiplicadas para 13 824, dividindo-as em 12 459 de treino, e 59 (10% das 590 imagens dadas à rede) para validação. O gráfico 3.1 contém quatro parâmetros tirados do treino e validação do treino A. Já mencionados na secção 2.3, estes são o *recall*, *precision*, mAP e mIOU. No eixo do x estão representadas as iterações ao longo do treino e no eixo do y os valores do *recall*, *precision*, mAP e mIOU.

Gráfico 3.1. Validação do Treino A



Como é observável pelo Gráfico 3.1, visualizamos que a rede teve um bom desempenho no treino e validação da base de dados. No entanto, foi observado um problema de *overfitting*. Apesar das métricas alcançarem valores acima dos 0.85 e o seu melhor valor mAP ser de 0.92, a quebra e estabilização do valor do *recall* deu-se muito antes dos outros parâmetros começarem a decrescer (6000 iterações). A presença de *overfitting* no modelo também é notada pelos valores altos das quatro métricas, onde é comum o modelo ter um bom desempenho no treino e na validação, e um mau desempenho no grupo de teste (I. Bilbao e J. Bilbao 2017).

Quando testámos os melhores pesos do treino A com um vídeo *streaming*, com imagens de militares diferentes das que foram treinadas e validadas na rede, observámos que os resultados³ não eram de todo satisfatórios. Isto acontece devido à complexidade do modelo, sendo demasiado complexo para a reduzida e não representativa base de dados fornecida para o treino do mesmo.

³<https://youtu.be/fnq-f0DZ6uY>

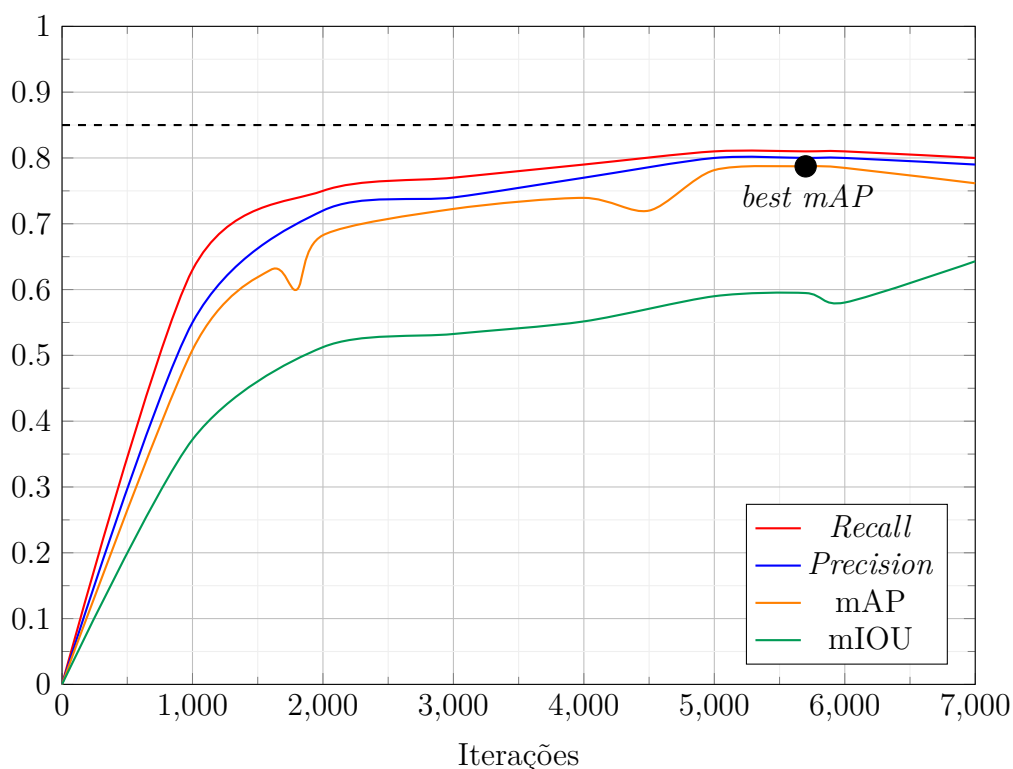
4.1.2 Base de Dados Militar

Com base nos resultados insatisfatórios do treino experimental, aumentámos a nossa base de dados militar para um total de 3436 imagens. Como já anteriormente referido, realizámos seis treinos distintos de forma a extrair quais as melhores configurações e pesos dados à rede, para a mesma ter um melhor desempenho nos dados de teste.

Devido à *data augmentation* automática do YOLOv4-tiny, as 3436 imagens anotadas foram multiplicadas para 76 486 de treino, e 344 (10% das 3436 imagens dadas à rede) para validação.

Com os mesmos hiperparâmetros da subsecção anterior, realizamos o treino da rede, o treino B, com a única diferença do número de imagens anotadas ser maior.

Gráfico 3.2. Validação do Treino B



Como podemos ver pelo Gráfico 3.2, o treino B apresentou um decréscimo nos quatro parâmetros, relativamente ao treino A com probabilidades altas, aqui o B ficou abaixo da linha a tracejado dos 0.85.

Apesar dos valores das métricas, dos melhores pesos do treino B, decrescerem quando foram comparados com os melhores pesos do treino A. O *recall* de

0.92 para 0.81, a *precision* de 0.86 para 0.80, a mAP de 0.92 para 0.79 e a mIOU de 0.7 para 0.59, quando testámos igualmente no mesmo vídeo que foi referido no treino A, os resultados⁴ foram muito mais satisfatórios. Isto porque, como o modelo treinou com mais dados, captou mais ruído e mais características dos objetos, o que fez com que tivesse um pior desempenho nos dados de validação, mas um melhor desempenho nos dados de teste.

Após termos uma boa base para prosseguir o trabalho, olhámos para os principais hiperparâmetros que podemos alterar no ficheiro de configuração da rede, de modo a alcançar melhores resultados. Esses foram o *batch*, as subdivisões e o número máximo de iterações.

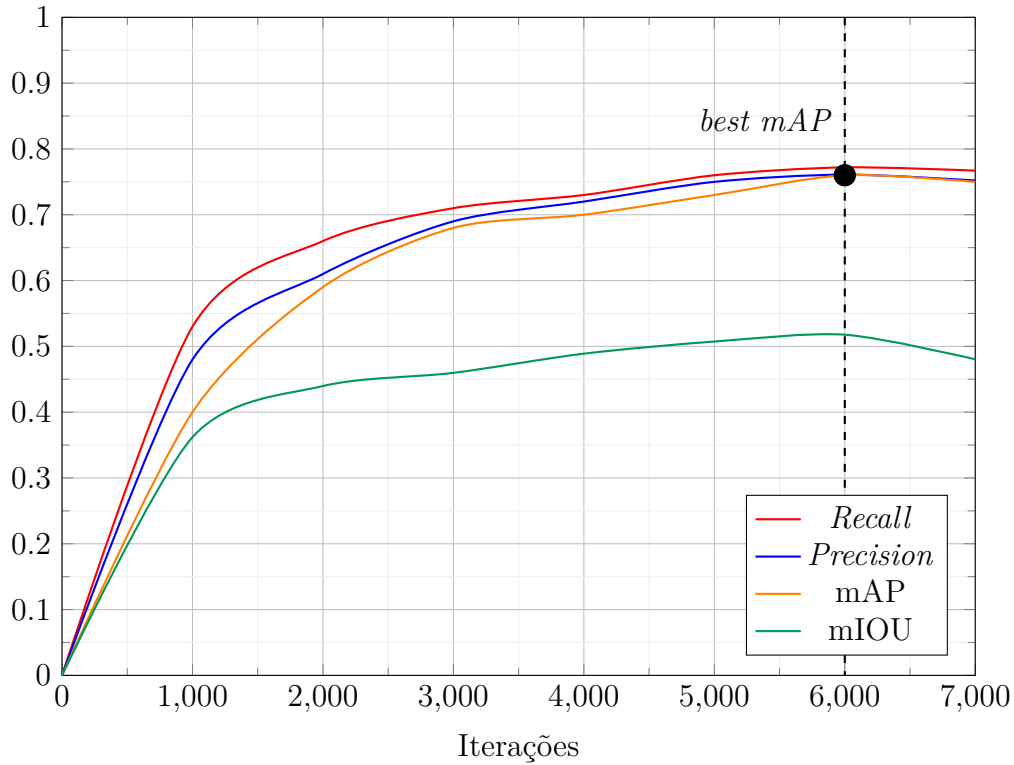
Estudo relacionados entre a relação do tamanho de *batch* com subdivisões não são comuns. No entanto, sabem-se as vantagens e desvantagens da escolha do tamanho do *batch*. Um *batch* grande possibilita a redução do tempo de treino da rede, em contrapartida, resulta em menos interações na atualização dos pesos, acentuando o problema do *overfitting* (Radiuk 2017). Experiências e testes realizados neste contexto mostram que aumentar o tamanho do *batch* vai resultar no gradual decréscimo da taxa de aprendizagem. Em contraste, pequenos tamanhos de *batch* resultam num maior gradiente computacional, verificando-se resultados mais estáveis e de confiança (R. Lin 2022). Concluindo, não há uma resposta específica para qual o número ideal de tamanho de *batch*, a sua escolha resulta num equilíbrio entre os benefícios de *batches* grande e pequenos.

Relativamente às subdivisões, ou *mini-batches*, são importantíssimos no treino da rede. Permitem ao modelo uma maior flexibilidade na atualização dos seus parâmetros ao longo do treino (Barreto 2021).

Com isto em mente, realizámos o treino C, diminuindo o número de *batch* para 32, subdivisões para 8 e iterações para 7000.

⁴<https://youtu.be/MyirYyZ0FbQ>

Gráfico 3.3. Validação do Treino C



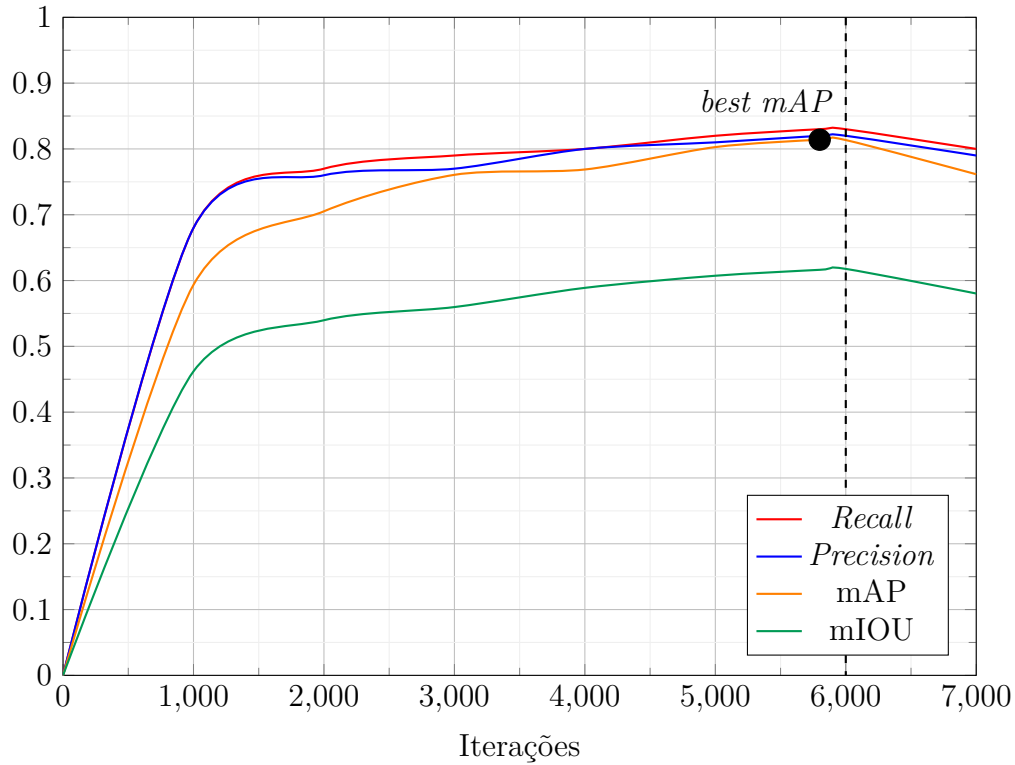
No Gráfico 3.3, conseguimos ver alguns impactos destas mudanças. O treino apresentou-se mais consistente, foi possível observar que 7000 iterações era em demasia, pois a partir das 6000 as probabilidades dos parâmetros começavam a decrescer. O tempo de treino também aumentou, de 4.5 horas para 5.8 horas devido ao decréscimo de imagens usadas numa época. Relativamente aos resultados de validação, o treino C conseguiu detetar mais ruído, diminuindo o mAP de 0.79 no treino B, para 0.76 no treino C. É observado um mIOU quase inferior a 0.5, possível de ser retificado nos próximos treinos.

Após este resultado, realizámos os restantes treinos previstos nesta subsecção, o treino D, E, F e G.

Os treinos D, E, F não variaram muito os valores das métricas. Podemos fazer uma breve comparação analisando os seus valores de mAP e tempo de treino. O treino D teve uma mAP de 0.80 com um tempo de treino de 7.5 horas. O treino E, uma mAP de 0.79 e um tempo de treino de 6.5 horas. O treino F, uma mAP de 0.81 com um tempo de treino de 8 horas.

Notou-se que o treino F, apesar das 8 horas que levava a realizar o treino, tendo um tamanho de *batch* maior mas subdivisões de 16, apresentou os melhores resultados de validação.

Gráfico 3.4. Validação do Treino F



No Gráfico 3.4, conseguimos visualizar o treino F, com a melhor mAP a situar-se na iteração 5800, com valor de 81.40. Também conseguimos notar um decréscimo substancial após as 6000 iterações.

Pela matriz de confusão representada na tabela 4.2, derivada do treino e validação de F, com um *threshold* de 0.5, vemos que das 344 imagens dadas para validação da rede, foram registados 2057 verdadeiros positivos, 456 falsos positivos e 428 falsos negativos.

		Classe esperada	
		Sim	Não
Classe prevista	Sim	2057	456
	Não	428	—

TABELA 4.2: Matriz de confusão dos melhores pesos do Treino F, *threshold* 0.5.

Relativamente ao treino G, foram usados os pesos do treino A para treinar a rede. Verificou-se uma menor mAP, esta sendo de 0.80, não ficando muito longe, mesmo assim, do treino F.

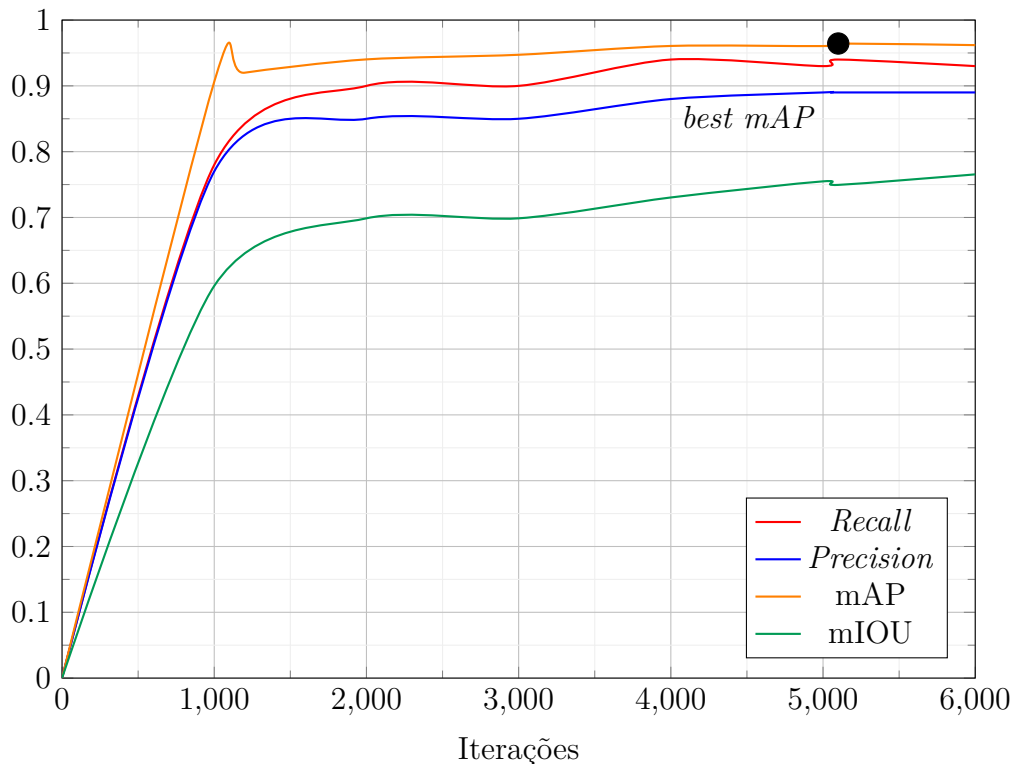
Derivado dos resultados de validação dos diversos treinos, os melhores pesos finais para a base de dados militar, relativos aos hiperparâmetros escolhidos, caíram sobre os melhores pesos do treino F.

4.1.3 Base de Dados Militar/Civil

O desafio seguinte para o trabalho, não se limitava apenas na identificação de militares, pois numa situação real de guerra, poderão existir tanto militares como civis no terreno. As diferenças entre estes dois elementos, visualmente, são poucas. Nesta subsecção, são apresentados os resultados do treino da rede com duas classes distintas, civil e militar. Para isto, tivemos de alterar novamente alguns dos hiperparâmetros da rede, nomeadamente as classes para 2 e os filtros para 21.

O primeiro treino, o treino H, com 447 imagens, sofreu *data augmentation* e a rede aumentou o seu número de imagens para 10 075 de treino e 44 (10% das imagens inseridas na rede) para a fase da validação. No gráfico 3.5 podemos observar o desempenho da rede ao longo do treino.

Gáfico 3.5. Validação do Treino H



Mais uma vez, semelhante ao treino A, a rede é demasiado complexa relativamente à base de dados pouco representativa e reduzida. Fazendo com que os

valores das métricas na validação sejam altíssimos, com uma mAP de 0.96.

Pela matriz de confusão representada na tabela 4.4, vemos que nas 44 imagens dadas para validação da rede, com *threshold* de 0.5, foram registados 118 verdadeiros positivos, 9 falsos positivos e 7 falsos negativos.

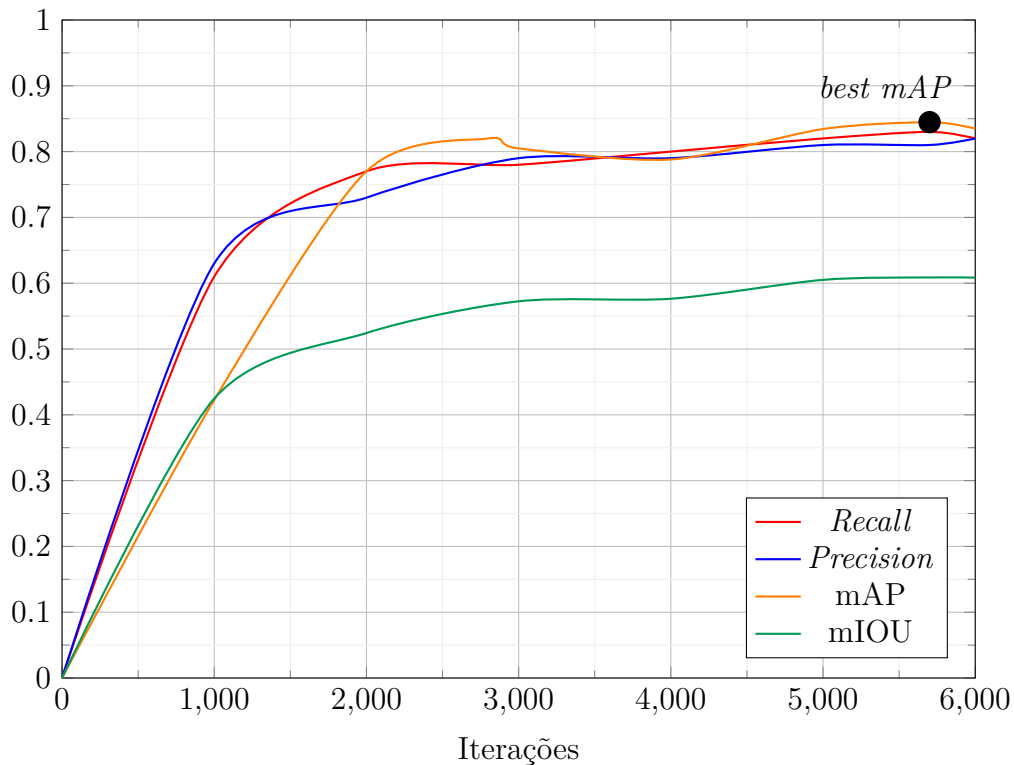
		Classe esperada	
		Sim	Não
Classe prevista	Sim	118	9
	Não	7	—

TABELA 4.3: Matriz de confusão dos melhores pesos do Treino H, *threshold* 0.5.

Individualmente, foram registados da classe militar, 65 verdadeiros positivos e 9 falsos positivos. Da classe civil, 53 verdadeiros positivos e 5 falsos positivos.

Relativamente ao treino I, como já foi referido, juntou-se a base de dados militar com a base de dados militar/civil, dando um total de 3 883 imagens. A *data augmentation* multiplicou este valor para 86 536 imagens de treino e 389 de validação. Conseguimos ver o progresso ao longo do treino I no Gráfico 3.6.

Gráfico 3.6. Validação do Treino I



É possível observar a redução das probabilidades dos quatro parâmetros, em comparação ao treino I. Reduzindo a mAP de 0.96 para 0.84.

Pela matriz de confusão representada na tabela 4.4, vemos que das 389 imagens dadas para validação da rede, com *threshold* de 0.5, foram registados 2162 verdadeiros positivos, 513 falsos positivos e 447 falsos negativos.

		Classe esperada	
		Sim	Não
Classe prevista	Sim	2162	513
	Não	447	—

TABELA 4.4: Matriz de confusão dos melhores pesos do Treino I, *threshold* 0.5.

Individualmente, foram registados da classe militar, 2119 verdadeiros positivos e 505 falsos positivos. Da classe civil, 43 verdadeiros positivos e 8 falsos positivos.

4.1.4 Exposição final dos vários treinos e os seus mAP

A Tabela 4.5 apresenta os vários treinos e as respetivas mAP referentes aos melhor pesos de cada um. De referir os pesos que vão ser utilizados na fase do teste, estes sendo os pesos F com mAP de 81.40%, os pesos H com mAP de 96.43% e os pesos I com mAP de 96.43%.

Treino	Hiperparâmetros	BD	mAP	Obs.
A	<i>64Batch / 16Sub / 6kMaxBatch</i>	Experimental	92.39%	<i>Overfitting</i>
B	<i>64Batch / 16Sub / 6kMaxBatch</i>	Militar	78.72%	-
C	<i>32Batch / 8Sub / 7kMaxBatch</i>	Militar	76.12%	-
D	<i>96Batch / 16Sub / 6kMaxBatch</i>	Militar	80.21%	-
E	<i>96Batch / 32Sub / 6kMaxBatch</i>	Militar	79.13%	-
F	<i>128Batch / 16Sub / 6kMaxBatch</i>	Militar	81.40%	-
G	<i>128Batch / 16Sub / 6kMaxBatch</i>	Militar	79.60%	Pesos A
H	<i>128Batch / 16Sub / 6kMaxBatch</i>	Militar/Civil	96.43%	<i>Overfitting</i>
I	<i>128Batch / 16Sub / 6kMaxBatch</i>	Militar+Civil	84.47%	-

TABELA 4.5: Exposição dos vários treinos e os seus mAP.

4.2 Teste

Apesar dos bons resultados na validação da rede (treinos F, H e I principalmente), apenas saberemos se a rede está a ter realmente um bom desempenho, quando a testarmos em vídeos ou imagens nunca antes vistas pela mesma. Nesta subsecção iremos realizar isso. Expondo três vídeos/imagens distintas aos melhores pesos, F, H e I. Dividindo o estudo em três partes e chegando à conclusão de quais são os melhores pesos.

A Tabela 4.6 apresenta o número total de imagens e a sua proveniência, utilizadas para esta fase de teste, para cada secção de estudo (Militar, Militar/Civil e Civil). As bases de dados onde foram extraídas as imagens para teste, poderão apresentar-se como uma barreira na conclusão do trabalho, visto que não foram capturadas por drones, mas sim extraídas da Internet, capturadas por câmaras de vigilância, máquinas fotográficas ou de filmagem. Favorecendo os pesos H e I, que contêm mais imagens desta natureza. No entanto, foram as únicas imagens conseguidas nesta fase, diferenciadas para cada secção de estudo.

Fase de Teste		
Secção de Teste	NºImagens	Base Dados Específica
Militar	65	BDME1
Militar/Civil	52	BDMCE
Civil	52	BDCE

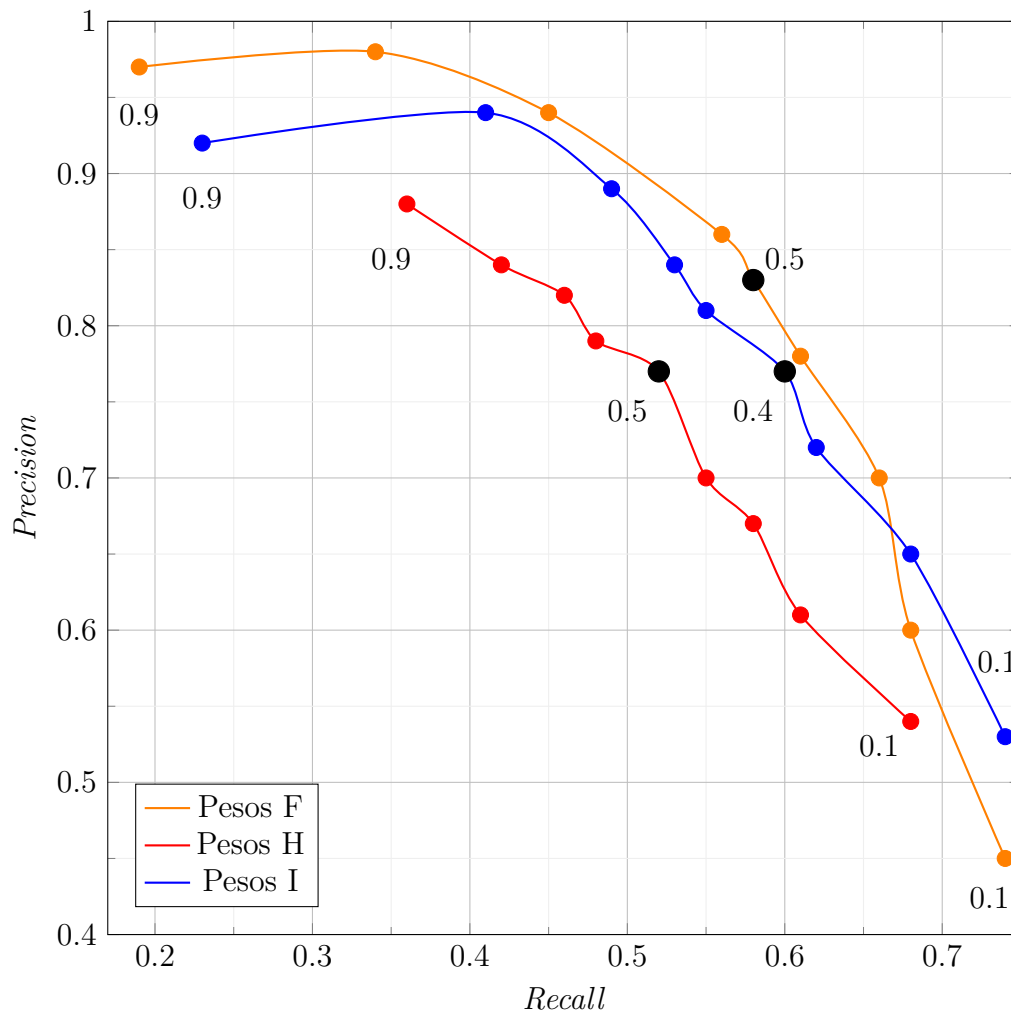
TABELA 4.6: N° de imagens e proveniência na fase de teste, relativa a cada secção de estudo.

4.2.1 Teste a Militares

De modo a realizar o teste dos pesos F, H e I em apenas militares, foi recolhido um vídeo com apenas esta classe presente, ao todo 65 imagens anotadas, com 1 até 8 objetos a variar entre imagens, provenientes da BDME1.

Realizado o teste, desenvolvemos o gráfico 3.7, relacionando a *precision* com o *recall* do modelo, que nos dá uma comparação dos vários desempenhos de teste dos pesos.

Gráfico 3.7. Curva *Precision VS Recall* - Pesos F, H e I com militares



Conseguimos aferir, do Gráfico 3.7, que os pesos F são os que apresentam um melhor desempenho de teste em militares. O melhor *threshold* escolhido destes pesos foi o 0.5, permitindo um balanço entre a *precision*, de 83%, com o *recall*, de 58%. Notamos claramente o problema de *overfitting* nos pesos H, derivado do seu treino com poucos e não variados dados.

Os vários pontos dos diferentes pesos, presentes no gráfico, representam os diferentes *thresholds* recolhidos neste estudo da *precision* e *recall*. Desde o *threshold* de 10% a 90%. Para os restantes pesos foram escolhidos os seus respetivos melhores *thresholds*, a Tabela 4.7 faz a comparação entre ambos.

Pesos	<i>Threshold</i>	<i>Precision</i>	<i>Recall</i>
F	0.5	83%	58%
H	0.4	77%	52%
I	0.5	77%	60%

TABELA 4.7: Comparação da *precision* e *recall* dos melhores pesos escolhidos, relativamente ao melhor *threshold*, em imagens com militares.

Observamos igualmente, pelos pesos F e I que, quando a rede é treinada com imagens de civis e militares, começa a procurar especificidades entre estas duas classes. Mesmo com as imagens do grupo de teste, iguais no teste dos dois pesos, a rede fica 'baralhada' em algumas das imagens. Isto devido aos pesos pré-treinados com a base de dados MS COCO, onde um militar/civil é apenas identificado como 'pessoa'. A rede, como usou a transferência de aprendizagem no treino, aprendeu rapidamente que um determinado objeto é um militar ou civil, diferenciando-o em caraterísticas específicas (e.g. cor da roupa, armamento), no entanto, para este desempenho ser mais elevado, iria necessitar de muitas mais imagens anotadas das duas classes.

As Figuras 4.1, 4.2 e 4.3 demonstram testes feitos a uma imagem militar com os respetivos pesos.

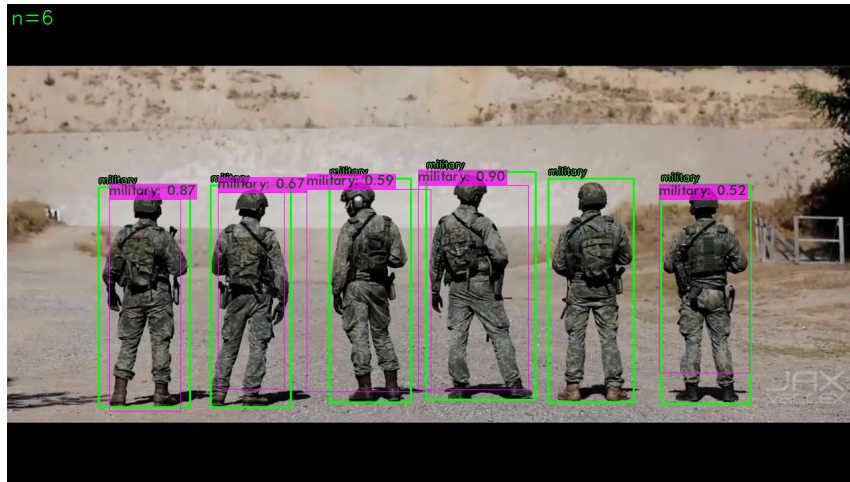


FIGURA 4.1: Teste a uma imagem militar, pesos F.



FIGURA 4.2: Teste a uma imagem militar, pesos H.

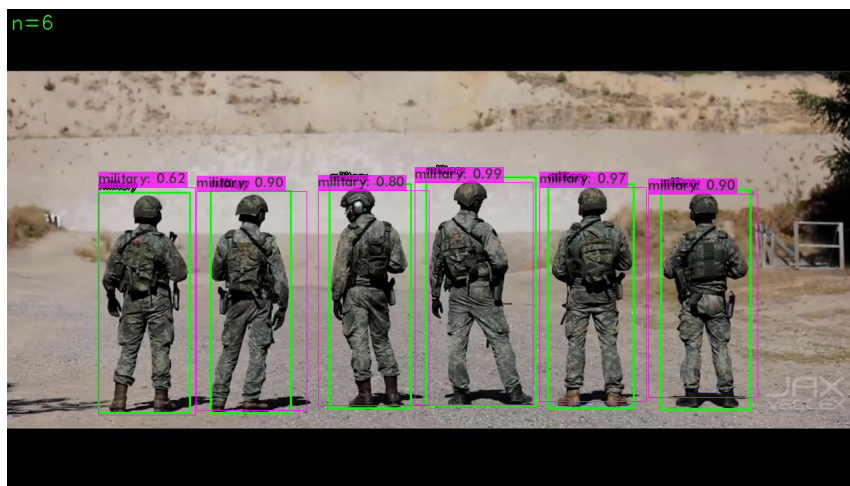
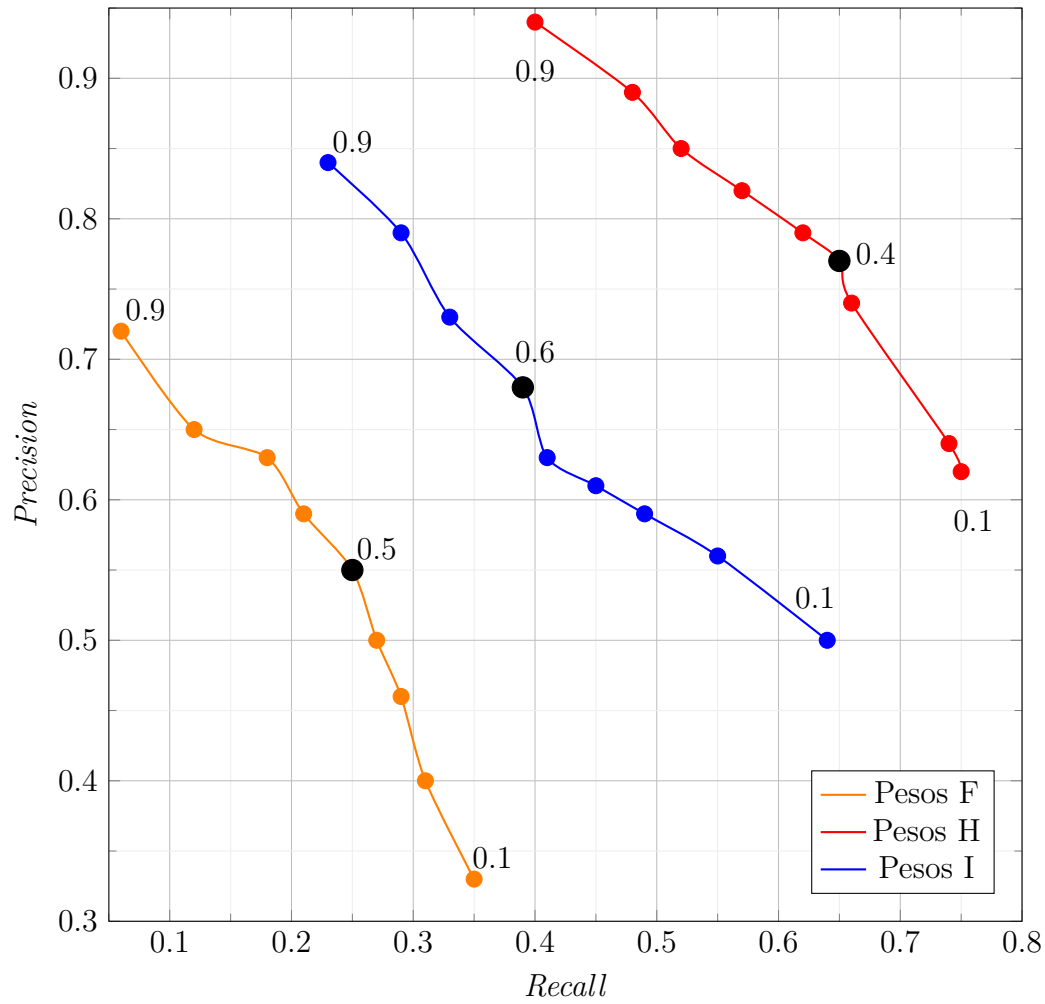


FIGURA 4.3: Teste a uma imagem militar, pesos I.

4.2.2 Teste a Militares e Civis

Para este teste foram recolhidas diferentes imagens e vídeos, um total de 52 imagens anotadas, com 1 até 13 objetos a variar entre as imagens, provenientes da BDMCE.

No gráfico 3.8, as respetivas curvas dos pesos F, H e I, que relacionam a *precision* e *recall* do modelo, são apresentadas.

Gráfico 3.8. Curva *Precision* VS *Recall* - Pesos F, H e I com militares/civis

Pelo Gráfico 3.8 percebemos que, quando se trata de civis e militares na mesma imagem, os pesos F não apresentam uma boa performance quando comparada ao estudo anterior. Neste caso, os pesos do treino H, apresentam uma maior proximidade ao modelo perfeito. Com o melhor *threshold* escolhido, de 40%, os pesos H apresentam uma *precision* de 77% e um *recall* de 65%. No entanto, este bom desempenho pode coincidir com o facto de as imagens utilizadas no teste, serem gravadas em posições e ângulos semelhantes às utilizadas no treino H.

As outras curvas, dos pesos F e I, comparativamente aos pesos H, possuem uma baixo desempenho na relação *precision/recall*, como é exposto na Tabela 4.8.

Pesos	<i>Threshold</i>	<i>Precision</i>	<i>Recall</i>
F	0.5	55%	25%
H	0.4	77%	65%
I	0.6	68%	39%

TABELA 4.8: Comparação da *precision* e *recall* dos melhores pesos escolhidos, relativamente ao melhor *threshold*, em imagens com militares/civis.

As Figuras 4.4, 4.5 e 4.6 demonstram testes feitos a uma imagem militar/-civil com os respetivos pesos.

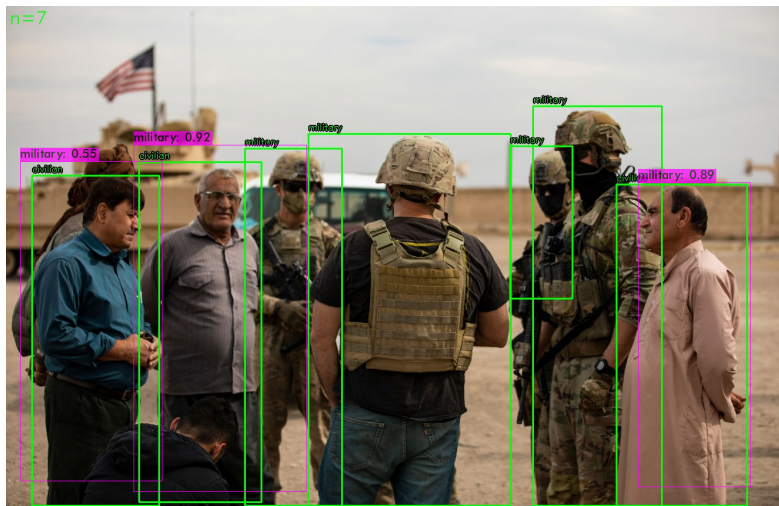


FIGURA 4.4: Teste a uma imagem militar/civil, pesos F.

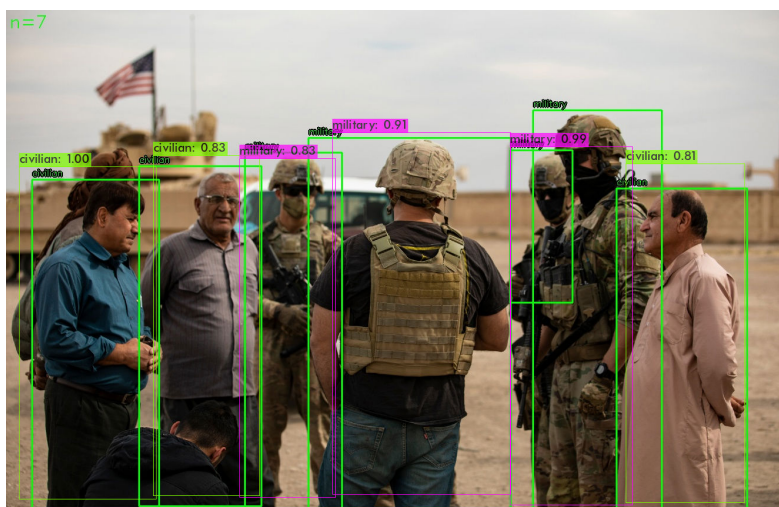


FIGURA 4.5: Teste a uma imagem militar/civil, pesos H.



FIGURA 4.6: Teste a uma imagem militar/civil, pesos I.

4.2.3 Teste a Civis

O último teste, foi feito em apenas imagens de civis. Recolheram-se várias fotos e vídeos, um total de 52 imagens, com 1 até 3 objetos a variar entre as imagens, pertencentes à BDCE.

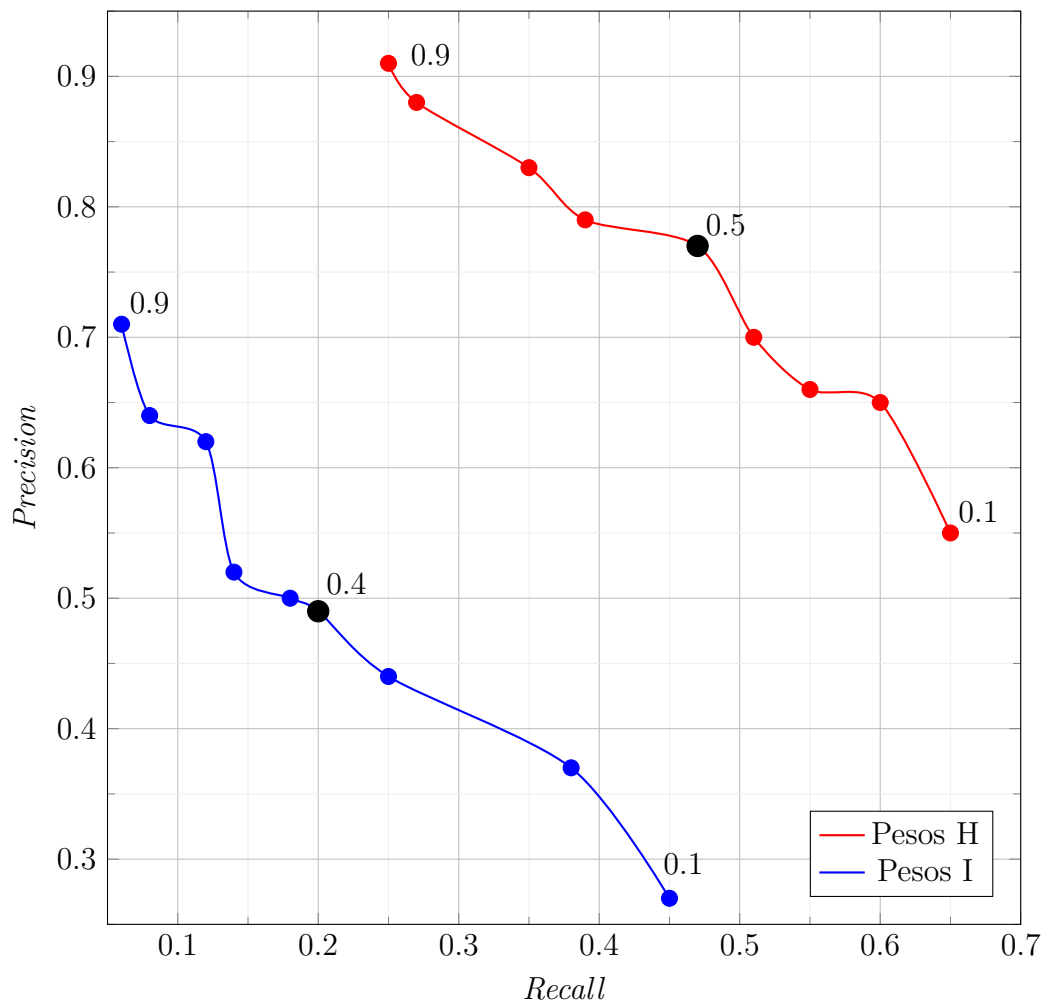
No Gráfico 3.9, as respetivas curvas dos pesos H e I, que relacionam a *precision* e *recall* do modelo, são apresentadas. Importante referir que não se fez a curva dos pesos F porque não haveriam verdadeiros positivos, sendo que quer a *precision* ou o *recall* seriam de valor 0.

No entanto, realizámos o teste com os pesos F nesta base de dados. Tivemos como resultados os valores presentes da Tabela 4.9, onde foram detetado falsos positivos (militares que não o são) com vários *thresholds*. Concluímos que se utilizados os pesos F na deteção de militares com civis misturados, irão detetados civis como militares efetivamente.

Teste dos pesos F em civis		
<i>Threshold</i>	NºCivis	Falsos Positivos
0.3	85	57
0.4	85	37
0.5	85	17
0.6	85	24
0.7	85	17
0.8	85	13

TABELA 4.9: Teste dos pesos F no grupo de teste apenas com civis.

Gráfico 3.9. Curva *Precision VS Recall* - Pesos H e I com civis



Igualmente ao caso anterior, os pesos H obtiveram um melhor desempenho no teste da rede. Com um *threshold* de 50%, o modelo alcançou uma *precision* de e um *recall* de 77% e 47% respetivamente. No entanto, como já referido, estes resultados podem ficar comprometidos em videos de teste gravados por um drone, sendo uma prespetiva diferente à base de dados dada para grupo de tese nesta secção. É possível realizar a comparação com os pesos I, mais detalhadamente, pela Tabela 4.10.

Pesos	<i>Threshold</i>	<i>Precision</i>	<i>Recall</i>
H	0.5	77%	47%
I	0.4	49%	20%

TABELA 4.10: Comparação da *precision* e *recall* dos melhores pesos escolhidos, relativamente ao melhor *threshold*, em imagens com civis.

4.2. Teste

As Figuras 4.7, 4.8 e 4.9 demonstram testes feitos a uma imagem civil com os respectivos pesos.



FIGURA 4.7: Teste a uma imagem civil, pesos F.



FIGURA 4.8: Teste a uma imagem civil, pesos H.

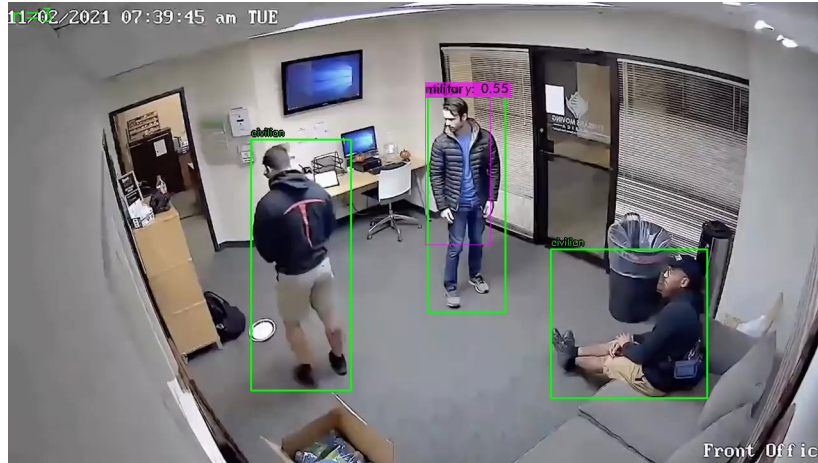


FIGURA 4.9: Teste a uma imagem civil, pesos I.

4.2.4 Os Melhores Pesos para o Modelo

Em suma, para este trabalho, não é possível escolher um dos melhores pesos em específico. Ou seja, temos dois pesos que podem ser escolhidos quando nos referimos a diferentes campos de estudo. Estes são os pesos F e os pesos H.

Os pesos F, quando falamos em deteção puramente militar, com uma relação *precision/recall* de 83%/58%. Conseguimos perceber, através do *recall*, que o modelo com estes pesos irá detetar uma maior quantidade de falsos negativos.

Por outro lado, quando olhamos para a deteção de militares e civis na mesma *frame* de vídeo, a melhor decisão de escolha dos pesos ideais recai sobre os pesos H. Estes com uma relação *precision/recall* de 77%/65%. Com uma *precision* mais baixa comparativamente aos pesos F na deteção de militares, mas com um *recall* mais elevado.

No entanto, sendo estes *thresholds* os melhores teoricamente, no nosso trabalho não seriam os escolhidos. A razão é pelo simples facto de que numa operação, é mais favoráveis termos falsos positivos do que falsos negativos, ou seja, é preferível detetarmos civis que não são militares e realizarmos a confirmação se são ou não, do que não identificarmos sequer militares, comprometendo a missão. Com isto, os possíveis *thresholds* selecionados terão uma *precision* menor e um *recall* maior.

Apesar dos ótimos valores que estes pesos apresentam para as duas situações descritas, estes resultados não são tão simples quanto aparentam. Quando nos focamos efetivamente na deteção de militares, e a sua distinção entre civis, a partir de um UAV em operações militares, o treino realizado neste trabalho não é suficiente

e apresenta inúmeras limitações. Neste seguimento, iremos realizar diferentes testes, mas agora com vídeos derivados de gravações com um drone.

4.2.5 Confirmação final dos vários pesos com vídeos adquiridos por um drone

Como já anteriormente referido, o nosso conjunto de dados de teste era limitativo, na medida que não continha imagens provenientes de um drone. O que tornou os resultados pouco fiáveis quando relacionados com o objetivo do nosso trabalho.

Neste seguimento, foram realizadas gravações a partir de um drone onde continham tanto militares como civis. A especificidades destas gravações estão contidas na Tabela 4.11.

Testes Finais			
Teste	Nº médio objetos	Tempo vídeo (s)	Plataforma
1	1 civil / 3 militares	19	DJI Air 2S
2	1 civil / 1 militar	50	DJI Air 2S

TABELA 4.11: Especificidades das gravações finais de teste, gravadas pelo drone DJI Air 2S.

Na Figura 4.10 conseguimos visualizar algumas *frames* destas gravações. Onde as imagens à esquerda correspondem ao Teste2 e as da direita ao Teste1.



FIGURA 4.10: Amostras de imagens das gravações realizadas com o drone DJI Air 2S para os testes finais, à esquerda correspondem ao Teste2 e as da direita ao Teste1.

Após esta recolha de imagens, realizamos os teste utilizando novamente os pesos F, H e I individualmente nos dois vídeos, com um *threshold* de 0.5.

Pesos F

Utilizando os pesos F, apenas com militares, sabíamos imediatamente que a rede não iria detetar civis. No entanto, utilizámos com o objetivo de averiguar se obtinha muitos falsos positivos e usar como comparação aos pesos I.

Os resultados desta deteção podem ser visualizados nos seguintes vídeos no Youtube: Teste1⁵ e Teste2⁶. Também é possível observar duas *frames* de cada vídeo na Figura 4.11, notando a confusão da rede na deteção, visto que não foi treinada para diferenciar civis de militares. Importante de referir igualmente, que a rede teve dificuldades em detetar militares ou civis de branco, cor não predominante na base de dados.

⁵<https://youtu.be/LWKY4H7nEI0>

⁶<https://youtu.be/rpnHMffCXbw>



FIGURA 4.11: Amostras de imagens do Teste1 e do Teste2 com os pesos F.

Pesos H

Utilizando os pesos H, já com as classes militar e civil, esperamos já ter o *feedback* do treino para esta diferenciação de uma forma mais credível. Ainda mais, quando na avaliação dos melhores resultados entre estes pesos, os pesos H prevaleceram.

Os resultados desta detecção podem ser visualizados nos seguintes vídeos no Youtube: Teste1⁷ e Teste2⁸. Também é possível observar duas *frames* de cada vídeo na Figura 4.12. Nota-se a dificuldade da rede em diferenciar militares de civis numa perspectiva aérea, confirmando-se a observação de que a rede com estes pesos se encontrava em *overfit*. Chegamos à conclusão que afinal os pesos H não serão os melhores quando queremos detetar e diferenciar militares de civis a partir de um drone, contrariamente ao referido anteriormente na secção de teste.



FIGURA 4.12: Amostras de imagens do Teste1 e do Teste2 com os pesos H.

⁷<https://youtu.be/gIIBYKmAbeg>

⁸<https://youtu.be/97r1840AmVM>

Pesos I

Utilizando os pesos I, será interessante averiguar se terá efetivamente melhor desempenho neste conjunto de dados de teste, em comparação aos pesos H. Visto que na secção de teste ficou abaixo do desempenho dos pesos H. Apesar de já termos observado que os pesos H estão em *overfitting*.

Os resultados desta deteção podem ser visualizados nos seguintes vídeos no Youtube: Teste1⁹ e Teste2¹⁰. Também é possível observar duas *frames* de cada vídeo na Figura 4.13.



FIGURA 4.13: Amostras de imagens do Teste1 e do Teste2 com os pesos I.

Com este último teste, conseguimos ver que os melhores pesos na diferenciação de militares e civis recaem sobre os pesos I, contrariamente à conclusão na fase de teste anterior. Apesar deste bom desempenho, nota-se que a rede apresenta falhas significativas, nomeadamente a não deteção do elemento civil quando está deitado no chão ou a deteção do mesmo como militar, a não deteção de militares fardados com uniformes brancos ou de diferentes cores daquelas a que a imagem treinou.

Estas falhas são derivadas de algumas limitações que fomos absorvendo ao longo do trabalho. No capítulo final, iremos olhar para estas limitações. Bem como para trabalhos futuros e conclusões retiradas deste trabalho.

⁹<https://youtu.be/9Dl29q62ikM>

¹⁰https://youtu.be/83y4fg_8DMY

Capítulo 5

Conclusão

5.1 Conclusões Finais

Nesta dissertação, o objeto de estudo recaiu no treino de uma rede neuronal personalizada, capaz de detetar e diferenciar militares de civis. Com o objetivo final de implementar a mesma num UAV em apoio a operações militares. Para isto, foi escolhida a rede YOLOv4-tiny, uma rede com uma boa relação entre a precisão na deteção de objetos e velocidade no processamento de imagem.

O passo seguinte foi construir uma base de dados e realizar a anotação de cada objeto em cada imagem ou *frame* de vídeo. Conseguimos adquirir um total de 732 imagens e 76 minutos de vídeo a partir de 5 plataformas diferentes: Três drones diferentes, YouTube e Google Images.

Imediatamente, dividimos o nosso estudo em três partes cruciais para alcançarmos os melhores resultados: treino, validação e teste. No treino, repartimos o mesmo em três secções, um treino experimental, um treino apenas com imagens de militares e o outro treino com imagens de militares e civis. Treinámos a rede e validámos os resultados. Alcançando 3 melhores valores de mAP em 3 treinos distintos, de 81.40%, 84.47% e 96.43%. Com estes ótimos valores na fase de validação, procedemos à fase de teste.

No teste com cada um destes pesos, em três vídeos diferentes, um com militares, outro com militares e civis, e o último apenas com civis, concluímos que não havia um melhor peso, mas sim dois, para diferentes circunstâncias. Se apenas quiséssemos detetar militares, os pesos F seriam os melhores. Se o objetivo fosse diferenciar militares de civis, os pesos H prevaleceriam. Os valores para estes resultados relacionavam o *threshold*, a *precision* e o *recall*. Isto num nível teórico, porque a nível prático e real, teremos de dar primazia a uma baixa *precision* e a um alto *recall*, com o objetivo de mesmo que consigamos detetar militares como civis, temos

uma área de manobra para confirmar se a detecção está correta ou não. Contrariamente, se tivermos uma *precision* alta e baixo *recall*, irão haver bastantes falsos negativos, ou seja, militares a não serem detetados sequer.

No entanto, estes resultados foram com o primeiro conjunto de dados de teste, quando inserimos o segundo conjunto, com gravações provenientes de um drone, concluímos que os pesos H se encontravam em *overfit*, e como as imagens no primeiro grupo de testes eram diferentes mas ao mesmo tempo semelhantes (em ângulo e equipamento) às imagens usadas para o treino H, este problema não se manifestou. No segundo grupo de teste, os pesos que prevaleceram foram os I, na diferenciação de militares e civis, mesmo com as suas limitações, a rede apresentou um melhor desempenho em comparação com a utilização dos outros pesos.

Em suma, este trabalho serviu como prova de conceito inicialmente e, numa fase mais avançada, foi validado experimentalmente. Demonstrando a aplicabilidade da implementação da rede YOLOv4-tiny na detecção de militares, bem como a diferenciação entre militares e civis. Importante em Operações militares, reduzindo o custo de vidas humanas, apoiando direta e indiretamente o campo de batalha, aumentando o conhecimento situacional do terreno e do inimigo, entre outras vantagens. Ainda que com algumas limitações, como serão descritas na próxima secção, os resultados foram deveras satisfatórios, resultados estes que podem ser melhorados com trabalhos futuros.

5.2 Limitações

5.2.1 Base de dados e respetiva anotação

Uma das limitações primordiais é a base de dados. Para se conseguir realizar o treino e posteriormente uma boa detecção de objetos é necessário uma base de dados extensa e representativa. Tendo um ponto de referência, a base de dados MS COCO mesmo com 330 000 imagens, mais de 200 000 anotadas (T.-Y. Lin et al. 2014), quando implementada no YOLOv4 atinge uma AP máxima de aproximadamente 44% (Alexey Bochkovskiy, C.-Y. Wang e Liao 2020).

Com isto em mente, e olhando para o nosso trabalho, notamos que as mAP dos melhores pesos tiveram valores muito superiores ao esperado, devido ao *overfit* no conjunto de treino. Em confirmação, quando realizávamos o teste em vídeos completamente diferentes, que poderiam 'confundir' a rede ou com múltiplos objetos juntos a uma distância considerável, esta tinha um desempenho insatisfatório.

Mais ainda, a diferenciação entre um militar e um civil numa operação militar é de extrema importância e complexidade. Muitas das imagens e vídeos da nossa base de dados referentes a militares, eram semelhantes em termos de cor, fardamento, armamento ou ambiente. Continham pequenas mudanças mas, numa perspectiva militar, onde o militar opera em todo o ambiente, condições, luminosidade, com diferentes uniformes, camuflagem, as imagens eram pouco representativas.

Outra limitação é a anotação de imagens. Como mencionado anteriormente, usámos um programa gratuito de anotação de imagens (*DarkLabel* s.d.). Acontecendo inúmeras vezes, com objetos em profundidade, ter de realizar a ampliação da imagem, resultando na distorção do objeto e consequentemente uma fraca anotação.

Por fim, o último aspeto a referir é o ângulo das gravações dos vídeos. Apesar de se tentar realizar as gravações de vários ângulos, muitas vezes não o foi possível fazer. Além disso, quando usados os pesos pré-treinados relativos à base de dados MS COCO, no treino da nossa rede personalizada, soube-se que esta base de dados tem poucas imagens gravadas de distâncias longas ou de uma perspectiva vertical, distâncias e situações em que é necessário operar um UAV, para o mesmo não ser detetado pelo inimigo. esta limitação apresentou-se como um obstáculo no desenvolvimento e objetivo final do trabalho.

5.2.2 YOLOv4-tiny e as suas limitações

Apesar da rede escolhida ser, como já referido, uma das mais competitivas quando se fala em rapidez *versus* precisão, por este motivo apresenta limitações em termos de precisão relativamente a outras redes. Numa operação militar, este erro de precisão pode custar vidas humanas, ser uma quebra de segurança.

Igualmente, por ser uma rede que dá primazia à rapidez, descarta a aptidão de diferenciar pequenos objetos na imagem. O algoritmo do YOLOv4-tiny apresenta uma capacidade insuficiente de extrair características de pequenas dimensões quando realiza deteção de objetos, não conseguindo tirar vantagens de características multi-escalares (Q. Liu, Fan, Xi, Yin e Yang 2022).

Outro ponto importante de referir, é a capacidade do YOLOv4-tiny realizar *data augmentation*. Uma qualidade de extrema importância e vantajosa no treino da rede. No entanto, o mesmo só realiza *data augmentation* em três modos, saturação, exposição e tonalidade de imagem. Com o que já referimos na subsecção anterior, a nossa base de dados continha cores muitos semelhantes em objetos e muitas vezes, os mesmos eram de pequenas dimensões. Algo bom para combater

este problema é realizar, à parte, outras *data augmentations*, como por exemplo a *mosaic data augmentation*, esta permite ao modelo aprender e identificar, com uma maior precisão, objetos de dimensões inferiores (A. Sharma 2022), a ideia por trás desta *augmentation* é simples. Primeiramente, redimensiona as imagens e combina quatro imagens numa só, cada uma num canto diferente. De seguida, as *bounding boxes* são colocadas nos objetos a detetar. Após isto, a imagem é cortada aleatoriamente e as *bounding boxes* que ficaram sem objetos são retiradas. Por fim, a imagem é redimensionada novamente e as *bounding boxes* também redimensionadas em conjunto com a imagem (Mongaras 2022).

5.3 Trabalhos Futuros

Com o intuito de combater as limitações referidas anteriormente, melhorar o modelo e utilizá-lo efetivamente em apoio a operações militares ou até mesmo em circuito fechado de televisão, nesta secção são apresentadas melhorias e trabalhos futuros a ter em consideração, como podemos observar pela Figura 5.1. Em seguida, falaremos das ideias e propostas mais importantes e cruciais na continuidade deste trabalho.



FIGURA 5.1: Trabalhos Futuros.

Numa fase inicial, é necessário ter a noção que quantas mais imagens forem anotadas, com ambientes distintos, ângulos, distância, luminosidade, etc, maior será a capacidade da rede a aprender e melhorar o seu desempenho.

Numa segunda fase, combater o problema principal do YOLO, a dificuldade em extrair características de pequenas dimensões quando realiza a detecção de objetos. Esta limitação desafia a diferenciação de um militar para um civil em determinadas circunstâncias.

Para tal ser executado, olhamos para um possível modelo híbrido, implementando o YOLO com o Mask R-CNN (T. Wang 2019). A ideia parte na absorção do bom das duas redes. O Mask R-CNN, ao nível da precisão, com a capacidade de realizar classificação e segmentação de instância. Ou seja, detetar, classificar e delinear cada objeto distinto numa determinada imagem (Hafiz e Bhat 2020). O YOLO, ao nível da velocidade, com a sua capacidade de processamento de imagem elevado, muito utilizado na detecção de objetos em vídeo *streaming*.

Este modelo híbrido parte da utilização da *framework open source* Keras (Gulli e Pal 2017), gratuita e fácil de operar. Aqui, o objetivo utilizar as duas redes mencionadas anteriormente, YOLO com o Mask R-CNN. O conceito passa por, primeiramente, utilizar o YOLO a fim de detetar civis e militares rapidamente num determinado vídeo, se esta detecção for positiva para civis, o *output* das *bounding boxes* dessa detecção, é dada como *input* à rede Mask R-CNN, onde a mesma, treinada para isso, conseguirá detetar objetos característicos de um militar, quer o armamento, colete tático, mochila, capacete, etc. Com isto, a rapidez do modelo é cumprida, não descurando a precisão na detecção de objetos. Outra opção viável, misturando a detecção com a segmentação, recai sobre o YOLACT++ (Bolya, Zhou, Xiao e Lee 2020), um modelo convolucional produzido para detecção de objetos em tempo real, usando igualmente segmentação de instância. Apesar de perder com o Mask R-CNN em AP (29.8 contra 35.7), ganha em velocidade (33.5 FPS para 8.6 FPS).

A última proposta para trabalhos futuros neste âmbito é, eventualmente, implementar o modelo treinado num UAV e testá-lo em diferentes exercícios e condições.

Bibliografia

- Abraham, Tara H (2002). «(Physio) logical circuits: The intellectual origins of the McCulloch–Pitts neural networks». Em: *Journal of the History of the Behavioral Sciences* 38.1, pp. 3–25.
- Agarwal, Rahul (2019). *Demystifying Object Detection and Instance Segmentation for Data Scientists*. URL: <https://towardsdatascience.com/a-hitchhikers-guide-to-object-detection-and-instance-segmentation-ac0146fe8e11> (acedido em 19/06/2022).
- Akbulut, Yaman e Rayan Khalaf (2021). «Smart Arms Detection System Using YOLO Algorithm and OpenCV Libraries». Em: *Turkish Journal of Science and Technology* 16, pp. 129–136.
- Alfarzaei, Murad, Niu Qiang, Jiaqi Zhao, Refat Eshaq e Eryi Hu (abr. de 2020). «Coal/Gangue Recognition Using Convolutional Neural Networks and Thermal Images». Em: *IEEE Access* PP, pp. 1–1. DOI: 10.1109/ACCESS.2020.2990200.
- Ayyadevara, V Kishore e Yeshwanth Reddy (2020). *Modern Computer Vision with PyTorch: Explore deep learning concepts and implement over 50 real-world image applications*. Packt Publishing Ltd.
- Barreto, Saulo (2021). *Why Mini-Batch Size Is Better Than One Single “Batch” With All Training Data*. URL: <https://www.baeldung.com/cs/mini-batch-vs-single-batch-training-data> (acedido em 15/06/2022).
- Becker, Elizabeth (1999). «CRISIS IN THE BALKANS: THE DRONES; They’re Unmanned, They Fly Low, and They Get the Picture». Em: *The New York Times*.
- Bilbao, Imanol e Javier Bilbao (2017). «Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks». Em: *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 173–177. DOI: 10.1109/INTELCIS.2017.8260032.
- Bishop, Chris M (1994). «Neural networks and their applications». Em: *Review of scientific instruments* 65.6, pp. 1803–1832.
- Bochkovskiy, Aleksey (2020). *YOLOv4 the most accurate real-time neural network on MS COCO dataset*. URL: <https://alexeyab84.medium.com/yolov4->

- the-most-accurate-real-time-neural-network-on-ms-coco-dataset-73adfd3602fe (acedido em 26/03/2022).
- Bochkovskiy, Alexey, Chien-Yao Wang e Hong-Yuan Mark Liao (2020). «Yolov4: Optimal speed and accuracy of object detection». Em: *arXiv preprint arXiv:2004.10934*.
- Bolya, Daniel, Chong Zhou, Fanyi Xiao e Yong Jae Lee (2020). «Yolact++: Better real-time instance segmentation». Em: *IEEE transactions on pattern analysis and machine intelligence*.
- Campesato, Oswald (2020). *Artificial intelligence, machine learning, and deep learning*. Mercury Learning e Information.
- DarkLabel (s.d.). <https://github.com/darkpgmr/DarkLabel>. Accessed: 2022-06-10.
- Dickson, Ben (2019). *What is transfer learning?* URL: <https://bdtechtalks.com/2019/06/10/what-is-transfer-learning/> (acedido em 19/06/2022).
- Ferreira, Carlos Manuel Ribeiro (2021). «Edificação do Destacamento de *Intelligence, Surveillance and Reconnaissance* no Corpo de Fuzileiros». Tese de doutoramento. Instituto Universitário Militar.
- Fukushima, Kunihiko e Sei Miyake (1982). «Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition». Em: *Competition and cooperation in neural nets*. Springer, pp. 267–285.
- Glade, David (2000). *Unmanned aerial vehicles: Implications for military operations*. 16. Center for Strategy e Technology, Air War College, Air University.
- Goodfellow, Ian (2016). *Deep Learning-Ian Goodfellow, Yoshua Bengio, Aaron Courville-Google Books*.
- Griffith, Samuel B (1963). *Sun Tzu: The art of war*. Vol. 39. Oxford University Press London.
- Gulli, Antonio e Sujit Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.
- Hafiz, Abdul Mueed e Ghulam Mohiuddin Bhat (2020). «A survey on instance segmentation: state of the art». Em: *International journal of multimedia information retrieval* 9.3, pp. 171–189.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun (2015). «Spatial pyramid pooling in deep convolutional networks for visual recognition». Em: *IEEE transactions on pattern analysis and machine intelligence* 37.9, pp. 1904–1916.
- Hebb, Donald Olding (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

- Holleman, Matthisjs (2018). *One-stage object detection*. URL: <https://machinethink.net/blog/object-detection/> (acedido em 08/06/2022).
- Hua, Xia, Xinqing Wang, Dong Wang, Jie Huang e Xiaodong Hu (2018). «Military object real-time detection technology combined with visual salience and psychology». Em: *Electronics* 7.10, p. 216.
- Hulstaert, Lars (2018). *A Beginner's Guide to Object Detection*. URL: <https://www.datacamp.com/tutorial/object-detection-guide> (acedido em 19/06/2022).
- Janakiramaiah, B, G Kalyani, A Karuna, LV Prasad e M Krishna (2021). «Military object detection in defense using multi-level capsule networks». Em: *Soft Computing*, pp. 1–15.
- Jiang, Zicong, Liquan Zhao, Shuaiyang Li e Yanfei Jia (2020). «Real-time object detection method for embedded devices». Em: *computer vision and pattern recognition*.
- Kaur, Jagreet (2020). *Automatic Log Analysis using Deep Learning and AI*. URL: <https://www.xenonstack.com/blog/log-analytics-deep-machine-learning> (acedido em 19/06/2022).
- Khan, Salman, Hossein Rahmani, Syed Afaq Ali Shah e Mohammed Bennamoun (2018). «A guide to convolutional neural networks for computer vision». Em: *Synthesis lectures on computer vision* 8.1, pp. 1–207.
- Krizhevsky, Alex, Ilya Sutskever e Geoffrey E Hinton (2012). «Imagenet classification with deep convolutional neural networks». Em: *Advances in neural information processing systems* 25.
- Kurenkov, Andrey (2020). «A Brief History of Neural Nets and Deep Learning». Em: *Skynet Today*.
- LeCun, Yann, Léon Bottou, Yoshua Bengio e Patrick Haffner (1998). «Gradient-based learning applied to document recognition». Em: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
- Lin, Runze (2022). «Analysis on the Selection of the Appropriate Batch Size in CNN Neural Network». Em: *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*. IEEE, pp. 106–109.
- Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár e C Lawrence Zitnick (2014). «Microsoft coco: Common objects in context». Em: *European conference on computer vision*. Springer, pp. 740–755.
- Liu, Qi, Xiaoyu Fan, Zhipeng Xi, Zhijian Yin e Zhen Yang (2022). «Object detection based on Yolov4-Tiny and Improved Bidirectional feature pyramid

- network». Em: *Journal of Physics: Conference Series*. Vol. 2209. 1. IOP Publishing, p. 012023.
- Liu, Shu, Lu Qi, Haifang Qin, Jianping Shi e Jiaya Jia (2018). «Path aggregation network for instance segmentation». Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768.
- Liu, Shuo e Zheng Liu (2017). «Multi-channel CNN-based object detection for enhanced situation awareness». Em: *arXiv preprint arXiv:1712.00075*.
- McCarthy, John (2007). «What is artificial intelligence?» Em.
- Mongaras, Gabriel (2022). *YOLOX Explanation — Mosaic and Mixup For Data Augmentation*. URL: <https://medium.com/mllearning-ai/yolox-explanation-mosaic-and-mixup-for-data-augmentation-3839465a3adf> (acedido em 29/08/2022).
- Murthy, Ch, Mohammad Farukh Hashmi, Neeraj Bokde e Zong Woo Geem (mai. de 2020). «Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms—A Comprehensive Review». Em: *Applied Sciences*. DOI: 10.3390/app10093280.
- Padilla, Rafael, Sergio L Netto e Eduardo AB Da Silva (2020). «A survey on performance metrics for object-detection algorithms». Em: *2020 international conference on systems, signals and image processing (IWSSIP)*. IEEE, pp. 237–242.
- Padilla, Rafael, Wesley L Passos, Thadeu LB Dias, Sergio L Netto e Eduardo AB da Silva (2021). «A comparative analysis of object detection metrics with a companion open-source toolkit». Em: *Electronics* 10.3, p. 279.
- Paper, Develop (2021). *how does the convolution kernel in convolution neural network (CNN) extract image features*. URL: <https://developpaper.com/image-algorithm-how-does-the-convolution-kernel-in-convolution-neural-network-cnn-extract-image-features-python-realizes-image-convolution-operation/> (acedido em 19/06/2022).
- Parico, Addie Ira Borja e Tofael Ahamed (2021). «Real time pear fruit detection and counting using YOLOv4 models and deep SORT». Em: *Sensors* 21.14, p. 4803.
- Patrício, Hélio Ferreira (2011). *O Emprego de "UNMANNED AERIAL SYSTEMS" em Operações Militares e outras missões - Desafios para as Forças Armadas*. Lisboa.
- Prince, Simon JD (2012). *Computer vision: models, learning, and inference*. Cambridge University Press, p. 16.

- Radiuk, Pavlo M (2017). «Impact of training set batch size on the performance of convolutional neural networks for diverse datasets». Em:
- Ramalho, Pedro José Ascensão (2019). «Feature Extraction and Object Classification in Video Sequences for Military Surveillance». Em: *Faculdade de Engenharia da Universidade do Porto*.
- Redmon, Joseph, Santosh Divvala, Ross Girshick e Ali Farhadi (2016). «You only look once: Unified, real-time object detection». Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.
- Sharma, Aditya (2022). *Achieving Optimal Speed and Accuracy in Object Detection (YOLOv4)*. URL: <https://pyimagesearch.com/2022/05/16/achieving-optimal-speed-and-accuracy-in-object-detection-yolov4/> (acedido em 10/08/2022).
- Sharma, Pulkit (2019). *Image Classification vs. Object Detection vs. Image Segmentation*. URL: <https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81> (acedido em 19/06/2022).
- Shorten, Connor e Taghi M Khoshgoftaar (2019). «A survey on image data augmentation for deep learning». Em: *Journal of big data* 6.1, pp. 1–48.
- Sperling (2018). *Deep Learning Spreads*. URL: <https://semiengineering.com/deep-learning-spreads/> (acedido em 19/06/2022).
- Surrisyad, Hari et al. (2020). «A Fast Military Object Recognition using Extreme Learning Approach on CNN». Em: *International Journal of Advanced Computer Science and Applications* 11.12.
- Tan, Lu, Tianran Huangfu, Liyao Wu e Wenying Chen (2021). «Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification». Em: *BMC medical informatics and decision making* 21.1, pp. 1–11.
- Team, Great Learning (2021). *Real-Time Object Detection Using TensorFlow*. URL: <https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/> (acedido em 08/06/2022).
- Techzizou (2021a). *Train a custom YOLOv4-tiny object detector on Windows*. URL: <https://medium.com/analytics-vidhya/train-a-custom-yolov4-tiny-object-detector-on-windows-5ec436100433#065e> (acedido em 21/03/2022).
- (2021b). *Yolov4 vs Yolov4-tiny Training YOLO for Object Detection*. URL: <https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec> (acedido em 21/03/2022).

- Udeanu, Gheorghe, Alexandra Dobrescu e Mihaela Oltean (2016). «Unmanned aerial vehicle in military operations». Em: *Sci. Res. Educ. Air Force* 18.1, pp. 199–206.
- Ugail, Hassan (2022). *Deep Learning in Visual Computing: Explanations and Examples*. CRC Press.
- Wang, Tony (2019). *Recognizing Firearms from Images and Videos in Real-Time with Deep Learning and Computer Vision*. URL: <https://medium.com/@tont/recognizing-firearms-from-images-and-videos-in-real-time-with-deep-learning-and-computer-vision-661498f45278> (acedido em 17/06/2022).
- Wang, Xiaogang (2016). «Deep learning in object recognition, detection, and segmentation». Em: *Foundations and Trends in Signal Processing* 8.4, pp. 217–382.
- Warrell, Helen (2020). «From Desert Storm to Soleimani: how US drone warfare has evolved». Em: *Financial Times*.
- Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do e Kaori Togashi (2018). «Convolutional neural networks: an overview and application in radiology». Em: *Insights into imaging* 9.4, pp. 611–629.
- Yani, Muhamad, S Irawan e Casi Setianingsih (mai. de 2019). «Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail». Em: *Journal of Physics: Conference Series* 1201, p. 012052. DOI: 10.1088/1742-6596/1201/1/012052.
- Yuheng, Song e Yan Hao (2017). «Image segmentation algorithms overview». Em: *arXiv preprint arXiv:1707.02051*.
- Zou, Zhengxia, Zhenwei Shi, Yuhong Guo e Jieping Ye (2019). «Object detection in 20 years: A survey». Em: *arXiv preprint arXiv:1905.05055*.