INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Estimativa do Peso de Corvinas e Deteção de Períodos de Alimentação

JOÃO LEAL MADUREIRA DIAS
Junho de 2022

POLITÉCNICO
DO PORTO

Estimativa do Peso de Corvinas
e
Deteção de Períodos de Alimentação

João Leal Madureira Dias

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Ana Madureira
Coorientador: João Ferreira

Porto, Junho 2022

“So, remember to look up at the stars and not down at your feet. Try to make sense of what you see and wonder about what makes the universe exist. Be curious. And however difficult life may seem, there is always something you can do and succeed at. It matters that you don't just give up. Unleash your imagination. Shape the future.”

- Stephen Hawking

Dedicatória

Todo o trabalho desenvolvido e descrito ao longo do presente documento é dedicado aos meus avós.

Resumo

O presente trabalho de investigação tem como objetivo explorar a aplicação de modelos de *machine learning* e *deep learning* a imagens obtidas em tanques que agregam múltiplos peixes (*fish farms*).

O correto desenvolvimento dos seres vivos presentes nestes tanques envolve processos de controlo minuciosos, não só das condições do meio como também das características dos próprios animais. O peso é uma destas características e o seu controlo fornece informações importantes relativamente ao processo de crescimento e à saúde dos animais.

É frequente que os processos de controlo utilizados periodicamente pelas instituições responsáveis pela criação e desenvolvimento de determinados seres vivos sejam realizados de forma manual, o que implica não só um consumo de tempo significativo como também poderá colocar em risco o bem-estar do ser vivo e do individuo responsável.

Na tentativa de reduzir a janela temporal necessária para o recolha de dados relativos ao peso de corvinas que habitam as *fish farms* da empresa SEAentia é proposta a utilização de um procedimento, composto por um modelo YOLOv4, por um *script* em *Python* e por um modelo de regressão linear simples, capaz de realizar estimativas de peso para cada ser vivo.

Adicionalmente, é proposta também a utilização do mesmo modelo de visão por computador e de um *script* de pós-processamento para identificação de períodos de alimentação, caracterizados pelo agrupamento das corvinas numa determinada região das *fish farms*.

Palavras-chave: Deteção de objetos; Estimativa de pesos; Aquacultura; Modelo de Visão por Computador; Modelo de Regressão Linear simples;

Abstract

This research work aims to explore the application of machine learning and deep learning models to images obtained from tanks that aggregate multiple fish (fish farms).

The correct development of the living beings present in these tanks involves detailed control processes, not only of the environmental conditions but also of the characteristics of the animals themselves. Weight is one of these characteristics and its control provides important information regarding the growth process and the health of the animals.

It is common for monitoring processes used periodically by institutions responsible for the breeding and development of certain living creatures to be done manually, which not only implies a significant consumption of time but may also put at risk the welfare of the living being and the individual responsible.

In an attempt to reduce the time window required to collect data on the weight of croakers present in SEAentia's fish farms, it is proposed to use a procedure, composed of a YOLOv4 model, a Python script, and a simple linear regression model, capable of making weight estimates for each living creature.

Additionally, it is also proposed to use the same computer vision model and a post-processing script to identify feeding periods, which are characterized by the existence of groups of meagres in a certain region of the fish farms.

Keywords: Object Detection; Weight Estimation; Aquaculture; Computer Vision Model; Simple Linear Regression Model;

Agradecimentos

Em primeiro lugar, gostaria de agradecer à Doutora Ana Madureira e ao professor João Ferreira pela orientação providenciada durante o processo de escrita e elaboração da presente tese e pela sua disponibilidade para responder às minhas questões. Agradeço também à organização proponente, a SEAentia, pelo fornecimento de imagens e dados essenciais à realização de todo o projeto.

Em seguida, deixo o meu agradecimento à minha família pelo apoio que sempre me deram, sem hesitar, em períodos de maior dificuldade que atravessei, principalmente durante o passado ano letivo, onde foi necessária uma adaptação significativa a uma nova rotina de trabalho académico pós-laboral.

Em quarto lugar, agradeço ao João Samões e ao Diogo Barbosa, colegas que me acompanham desde a licenciatura em Engenharia Informática, não só pela sua amizade mas também por toda a ajuda que sempre se mostraram dispostos a dar. Foram vitais para o meu bom desempenho no decorrer do presente mestrado.

Por fim, agradeço também meus amigos André Madureira, João Fonseca, Gustavo Ferreira e José Ferreira, pelo papel importante que protagonizaram, principalmente no primeiro semestre do segundo ano do mestrado.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Equações

Lista de Acrónimos e Símbolos

Lista de Acrónimos

AHQ Analytic Hierarchy Process
AP Average Precision
BFLOP/s Billion Floating Point Operations per second
CNN Convolutional Neural Network
COCO Common Objects in Context
FPS Frames per second
GPU Graphics Processing Unit
IA Inteligência Artificial
ILSVRC ImageNet Large Scale Visual Recognition Challenge
IoU Intersection over Union
mAP Mean Average Precision
MDP Markov Decision Process
MiT Massachusetts Institute of Technology
PRC Precision-Recall Curve
RPN Region Proposal Network
SIFT Scale Invariant Feature Transform
SSD Single Shot Multi-Box Detector
SVM Super Vector Machine
TOPSIS Technique for Order of Preference of Similarity to Ideal Solution
VOC Visual Object Challenge
YOLO You Only Look Once

Lista de Símbolos

λ_{max} Valor Próprio (utilizado na aplicação do método AHP)
A Matriz de comparação (utilizada na aplicação do método AHP)
A' Conjunto de Valores associados a cada Critério para a Solução Menos Ideal (utilizado na aplicação do método TOPSIS)
A* Conjunto de Valores associados a cada Critério para a Solução Ideal (utilizado na aplicação do método TOPSIS)
C_i^* Proximidade da Solução Ideal (utilizado na aplicação do método TOPSIS)
IA Índice Aleatório
IC Índice de Consistência
J Conjunto de Critérios Positivos (utilizado na aplicação do método TOPSIS)
L Conjunto de Critérios Negativos (utilizado na aplicação do método TOPSIS)
m Número de Alternativas (utilizado na aplicação do método TOPSIS)
n Número de Critérios (utilizado na aplicação do método TOPSIS)
RC Razão de Consistência
S_i' Separação da Solução Ideal (utilizado na aplicação do método TOPSIS)
S_i^* Separação da Solução Ideal (utilizado na aplicação do método TOPSIS)
x Vetor de Prioridades (utilizado na aplicação do método AHP)

1 Introdução

1.1 Contexto

O peixe é um dos produtos mais comercializados a nível mundial, sendo que é a principal e única fonte de proteína e nutrientes disponível em muitas partes do mundo (E.H. Allison, 2011).

O grande aumento da população mundial e o consequente aumento da procura de alimento (nomeadamente peixe e seus derivados) entram em conflito direto com a crescente estagnação da indústria de pesca selvagem, colocando em risco o futuro deste setor a médio-longo prazo (Ottinger et al., 2016).

Como medida de combate às dificuldades sentidas pelo setor piscatório por todo o globo e na tentativa de tornar mais sustentáveis as atividades associadas a este setor, tem aumentado, nas últimas décadas, a procura e investimento no setor da zootecnia que estuda a produção racional de organismos aquáticos, com recurso a intervenção humana, prática também conhecida por aquacultura (E.H. Allison, 2011).

Estima-se que cerca de 50% do peixe consumido atualmente seja produzido com recurso a esta técnica e que a aquacultura seja cada vez mais preponderante no que diz respeito à saúde humana. Tendo em conta o duplicar do consumo de peixe *per capita* num curto espaço de 50 anos (entre 1960, onde esse mesmo consumo se situava em valores a rondar os 9.9 quilogramas, e 2010 onde disparou para 18.9 quilogramas) e a relevância que o consumo deste tipo de alimento tem no total de proteína consumida pela população mundial (cerca de 17% da proteína animal e 6.5% de proteína total consumida globalmente é proveniente de animais aquáticos, nomeadamente peixes), é possível afirmar que o investimento e desenvolvimento de técnicas de aquacultura é fulcral para o contínuo crescimento da nossa população (Troell et al., 2009).

Dados provenientes de instituições que controlam informações relativas à produção e pesca de peixes a nível mundial garantem que o desenvolvimento da indústria piscatória tradicional, medido através da produção de milhões de toneladas de peixe, no intervalo de tempo entre 1983 e 2013, viu um incremento de cerca de 20 milhões de toneladas (passando de cerca de 71.1 para 92.6 milhões de toneladas) (FAO, 2014). Enquanto, no mesmo período, a produção via aquacultura cresceu em cerca de 64 milhões de toneladas (aumento de 6.2 para 70.2 milhões de toneladas), o que representa um crescimento anual na ordem dos 8.6%, superando indústrias de renome como a de produção pecuária (4.6%) ou a indústria suína (2.2%) (Troell et al., 2009).

A versatilidade dos ecossistemas associados à aquacultura garante a produção e continuidade de mais de seis mil espécies de animais aquáticos, a nível mundial (Ottinger et al., 2016). O grande volume de animais a serem produzidos com recurso a este tipo de técnica origina aquele que é o principal ponto negativo da sua aplicação: a necessidade de grandes quantidades de animais aquáticos selvagens para servirem de alimento aos seres vivos criados em ambiente controlado. Desta forma, o crescimento da produção via aquacultura agrava a necessidade já existente de obtenção de peixe através de pesca tradicional, nomeadamente peixe de pequenas dimensões (não utilizado para consumo humano) e derivados deste produto, como óleo de peixe e farinha de peixe, utilizados na confeção de alimento para os animais em tanques e *fish farms* (Troell et al., 2009).

Na perspetiva de tornar a produção de peixe em aquacultura o mais sustentável possível surgiu a SEAentia (*SEAentia-Food, Lda.*), uma *start-up* portuguesa fundada em setembro de 2017 em Cantanhede, Coimbra. Esta empresa é pioneira na produção de corvinas (*Argyrosomus regius*) de alta qualidade, utilizando técnicas científicas que garantem o desenvolvimento sustentável da espécie, tanto a nível económico como a nível ambiental (SEAentia, 2017).

Os animais ao cuidado desta empresa são mantidos num sistema RAS (*Recirculating Aquaculture System*) que reduz o desperdício de água através da sua constante purificação e circulação, evitando também a utilização de produtos químicos que poderiam potencialmente prejudicar o desenvolvimento dos animais. Os resíduos e dejetos ou são removidos através de processos automáticos ou são neutralizados, garantindo o bem-estar de todos os seres vivos criados neste sistema (Halvorson and Smolowitz, 2009).

1.2 Definição do Problema

A SEAentia (SEAentia, 2017) pretende estabelecer-se como uma referência internacional em aquacultura ao explorar novas espécies com elevado potencial de comercialização e aceitação por parte do consumidor final. Este objetivo só será atingível se a empresa continuar a combinar novos métodos de cultura com tecnologia de vanguarda e conhecimento científico, o que garantirá uma produção sustentável com produtos rastreáveis, bio seguros e de elevada qualidade.

Hoje em dia, uma das principais atividades de controlo associadas à manutenção dos tanques e crescimento das espécies são as pesagens de peixes, realizadas periodicamente. Estas pesagens permitem tirar conclusões sobre o bem-estar dos animais, sobre o seu desenvolvimento e sobre as condições do seu habitat atual. O processo de realização das pesagens é manual e demorado, uma vez que exige a remoção, pesagem e reinserção de cada peixe no seu respetivo tanque.

A ineficiência deste processo é o problema que serve de base à presente tese e poderá ser parcial ou totalmente combatido com recurso a modelos de visão por computador e a modelos de regressão (*machine learning*), tal como será descrito no decorrer do documento.

De igual modo, o controlo dos períodos de alimentação dos animais é igualmente relevante. A presente tese propõe ainda um mecanismo de deteção de períodos de alimentação de forma automática, na perspetiva de poder fornecer informação relevante relacionada com estes períodos.

Por estas razões espera-se que ambas as soluções desenvolvidas apresentem alguma aplicabilidade ao mundo real e aos problemas enfrentados regularmente pelos colaboradores da SEAentia.

1.3 Objetivos

Os principais objetivos do trabalho descrito neste documento são a realização de uma estimativa do peso de cada corvina presente nos diversos tanques ao cuidado da SEAentia e a deteção automática de períodos de alimentação dos peixes presentes nos referidos tanques.

Para atingir o primeiro objetivo será realizado um processo que pode ser dividido em três fases: uma primeira fase onde será treinado e aplicado um modelo de visão por computador capaz de segmentar instâncias (de forma a distinguir individualmente cada

peixe no seu respetivo tanque), uma segunda fase responsável pelo cálculo do comprimento real de cada corvina detetada e uma terceira fase, onde será aplicado um modelo de regressão, previamente treinado com dados relativos às dimensões e pesos de diferentes corvinas, na perspetiva de estimar o peso de cada animal.

O segundo objetivo aproveitará o modelo de visão por computador treinado previamente e utilizará ainda um *script* em *Python* que permitirá interpretar os grupos de peixes encontrados em cada imagem, de modo a atingir uma conclusão relativamente à existência ou não de períodos de alimentação.

1.4 Abordagem

Tendo em conta o problema apresentado e os objetivos propostos, a abordagem relativa à construção de uma solução será composta pelas seguintes fases:

- Pesquisa bibliográfica relativa ao estado da arte, incluindo tópicos como: inteligência artificial, *machine learning*, *deep learning*, redes neuronais convolucionais, introdução histórica à área de *visão por computador*, análise de métodos de classificação de imagem e deteção de objetos de última geração e análise de trabalhos já realizados na área;

- Realização da análise de valor da solução proposta;

- Experimentação e Avaliação: análise das hipóteses, metodologias de avaliação, parâmetros de avaliação, estratégias de recolha de dados e métricas de avaliação;

- Análise e descrição detalhada do processo de implementação e estudo computacional;

- Conclusões, contribuições, limitações e trabalho futuro.

1.5 Organização do Documento

O presente documento é composto por sete capítulos e tem como objetivo fornecer ao leitor uma descrição detalhada sobre o processo de desenvolvimento da solução para o problema proposto.

O primeiro capítulo descreve a introdução da dissertação. Nesta secção é introduzida a motivação e o problema a resolver. São também enumerados os objetivos do trabalho, assim como as etapas da abordagem escolhida.

O segundo capítulo é composto pela revisão da literatura e estado da arte relativos ao tema em questão. Esta secção engloba conceitos relevantes na área de estudo, tais como: inteligência artificial, *machine learning*, *deep learning*, redes neuronais convolucionais e *visão por computador*. A descrição destes temas tem como objetivo contextualizar o estudo e desenvolvimentos que serão realizados nos capítulos seguintes, agregando informações de várias fontes. Neste capítulo são também descritas múltiplas versões de métodos de visão por computador de três famílias distintas. O segundo capítulo é terminado com uma análise de uma revisão de literatura realizada na área de estimativa de peso de seres vivos.

O capítulo 3 introduz a análise de valor da solução proposta que descreve de que forma é que o trabalho realizado neste documento poderá gerar valor e enriquecer a área de estudo. Esta análise é suportada por métodos de decisão multicritério como AHQ e TOPSIS.

Segue-se o quarto capítulo, onde, além de serem descritas as hipóteses a avaliar, são também descritas as metodologias e parâmetros de avaliação, as estratégias de recolha de dados e ainda as métricas utilizadas para avaliação dos modelos utilizados no decorrer das soluções implementadas.

O quinto capítulo contém uma descrição detalhada das configurações, características e particularidades do processo de desenvolvimento dos procedimentos aplicados para criação das soluções para os problemas propostos.

O penúltimo capítulo diz respeito à conclusão do documento, onde é realizado um balanço geral do trabalho desenvolvido durante os capítulos anteriores e onde são destacadas as contribuições e limitações do trabalho desenvolvido, assim como potenciais melhorias a realizar em trabalhos futuros.

O capítulo final contém a bibliografia, onde são listadas todas as referências utilizadas para a realização do presente documento.

2 Revisão de Literatura

No presente capítulo será apresentada informação relevante relacionada com a área de estudo. Será realizada uma contextualização relativamente a conceitos como inteligência artificial, *machine learning* e *deep learning*, assim como uma análise arquitetural a um modelo de rede neuronal convolucional. Nesta secção estará, posteriormente, incluída uma breve introdução histórica ao universo de visão por computador que será seguida por uma análise a três tipos de modelos utilizados frequentemente em tarefas de deteção de objetos e classificação de imagens. Estes modelos serão comparados entre si ao nível de desempenho e ao nível arquitetural.

O capítulo é finalizado com uma análise ao estado da arte da aplicação de modelos de visão por computador e *machine learning* a problemas relacionados com a estimativa de peso de determinados seres vivos.

2.1 Inteligência Artificial, Machine Learning e Deep Learning

Na tentativa de descrever software que se comporta de forma inteligente é frequente a utilização de termos como *deep learning*, *machine learning* e inteligência artificial como se de sinónimos se tratassem (Garbade, 2018). É, por isso, importante salientar que todos estes termos possuem, entre eles, diferenças significativas.

Inteligência artificial descreve a habilidade de uma máquina, robot ou computador reproduzir comportamentos comumente associados a seres humanos. Este tipo de capacidade pode ser desenvolvida através de processos de *machine learning*, que não necessitam de ser explicitamente programados por um agente humano e que conferem ao sistema a capacidade de se adaptar às circunstâncias onde se encontra inserido, respondendo de forma adequada (Copeland, 2014). *Deep learning* é uma ramificação de *machine learning* que descreve a utilização de redes neuronais com várias camadas, capazes de interpretar informação passada como *input* e de tomar decisões com base na informação obtida (Grieve, 2020). As três áreas de estudo encontram-se enquadradas na Figura 1.

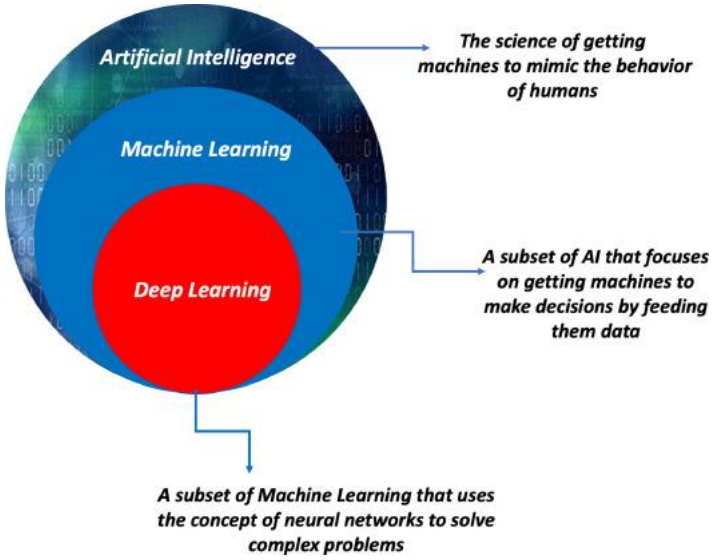

Figura 1 - Inteligência Artificial e suas ramificações (Leonardi et al., 2021a)

2.1.1 Inteligência Artificial

O conceito de Inteligência Artificial tem evoluído de forma exponencial nas últimas décadas (C. Huang et al., 2019) e está diretamente relacionado com a habilidade que um computador possui de realizar tarefas normalmente desempenhadas pelo ser humano (Voulodimos et al., 2018). É um conceito transversal a várias áreas incluído filosofia, ficção, medicina, engenharia, música e biologia. É também um conceito complexo e extenso que se desdobra em várias ramificações, componentes e tipos, tal como é ilustrado na Figura 2 (Abioye et al., 2021).

Um dos principais marcos históricos relacionados com este domínio foi a obra de Alan Turing, atualmente considerado um dos fundadores da Computação e Inteligência Artificial, que, em 1950, concebeu um teste capaz de desafiar as tradicionais posições teológicas da época. Este teste – Teste de Turing (Turing, 1950) - redefiniu todas as conclusões que tinham sido, até então, obtidas, sobre a possibilidade replicar o processo cognitivo de um ser inteligente numa máquina.

A definição de "Inteligência Artificial" viria a ser proposta pela primeira vez por John McCarthy, em 1956, numa conferência na universidade de Darmouth (McCarthy et al., 1956). Desde então, este conceito passou a estar interligado com a capacidade de um computador reproduzir e assimilar comportamentos humanos, tais como o processo de aprendizagem, o processo de realização de escolhas, o processo de tomada de decisão e o processo de interpretação de emoções (Zhang and Lu, 2021).

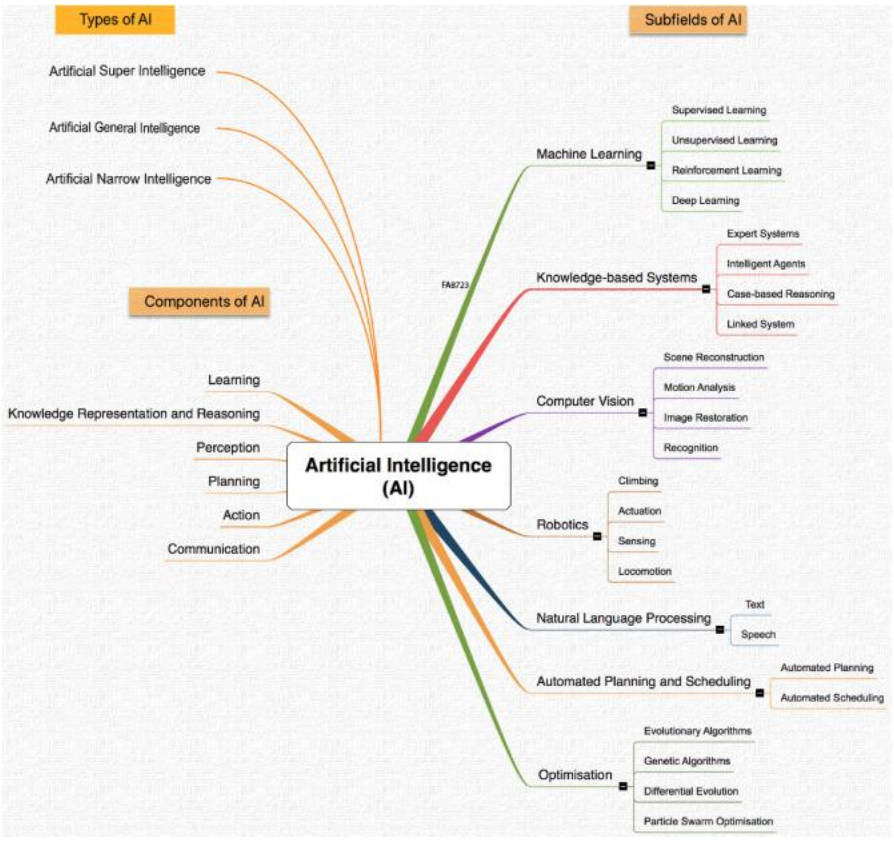

Figura 2 - Ramificações, componentes e tipos de Inteligência Artificial (Abioye et al., 2021)

Atualmente, devido a um aumento da capacidade de computação, a uma melhoria substancial do *hardware* e *software* integrados no desempenho deste tipo de funções e usufruindo dos avanços realizados em outras áreas (tais como *Big Data*), a utilização fiável de sistemas inteligentes encontra-se, cada vez mais, envolvida e integrada em aplicações utilizadas diariamente por milhões de pessoas e é uma tecnologia que afeta, de diversas formas e em grande escala, a nossa sociedade (Zhang and Lu, 2021).

2.1.2 Machine Learning

Machine learning é uma ramificação da área de inteligência artificial (Figura 2) e engloba o estudo, design e utilização de *softwares* capazes de simular o processo de aprendizagem e de realizar determinados comportamentos baseado em dados fornecidos previamente ou em experiências passadas (Abioye et al., 2021).

Algoritmos de *machine learning* podem ser divididos em quatro tipos, dependendo do seu método de aprendizagem: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço (*reinforcement learning*) (Zhang and Lu, 2021).

2.1.2.1 Aprendizagem Supervisionada

Ocorre quando são utilizados dados etiquetados no processo de treino do algoritmo que o levarão a prever e categorizar novos dados. Este tipo de aprendizagem pode ser subdividido em duas categorias: classificação ou regressão.

Problemas de classificação estão normalmente relacionados com a obtenção de resultados discretos, tais como a deteção de um determinado objeto numa imagem. Um dos métodos mais comuns de classificação por aprendizagem supervisionada denomina-se SVM (*Super Vector Machine*) e, como tal, pode ser aplicado em problemas relacionados com reconhecimento de imagens, categorização de texto, diagnósticos médicos e classificação de movimentos (Yuan et al., 2010).

Problemas de regressão têm como objetivo encontrar uma relação entre duas ou mais variáveis e permitem a realização de previsões matemáticas (Maulud and Mohsin Abdulazeez, 2020). Uma das aplicações mais comuns deste tipo de algoritmo é denominada regressão linear, conceito que foi proposto pela primeira vez por Sir Francis Galton, em 1984 (DOW et al., 1984), e desde então tem vindo a ser aplicado em áreas relacionadas com investigações matemáticas que permitam a previsão de valores de *output* através da modelação de vários valores de *input*. O processo de aprendizagem e treino deste tipo de algoritmo consiste na criação de relações lineares entre variáveis dependentes e independentes (Maulud and Mohsin Abdulazeez, 2020).

Por norma, aplicações de algoritmos de regressão linear são subdivididas em dois grupos: regressão linear simples e regressão linear múltipla.

A regressão linear simples consiste na previsão de uma variável dependente, *y*, com recurso a uma única variável independente, *x*. Este tipo de regressão linear pode ser descrito através da expressão $y = \beta 0 + \beta 1 x + \varepsilon$ (Maulud and Mohsin Abdulazeez, 2020).

Por outro lado, regressão linear múltipla é uma técnica utilizada para prever uma variável dependente, *y*, utilizando múltiplas variáveis independentes, *x*. Pode ser descrita através da expressão $y = \beta 0 + \beta 1 x 1 + \cdots + \beta m x m + \varepsilon$ (Maulud and Mohsin Abdulazeez, 2020).

2.1.2.2 Aprendizagem Não Supervisionada

Pressupõe precisamente o oposto de aprendizagem supervisionada e ocorre quando um algoritmo é treinado sem a utilização de dados etiquetados. É esperado que o algoritmo consiga detetar informações relevantes sobre os dados através de uma

análise exaustiva das sua estrutura e características (Leonardi et al., 2021b). É um tipo de aprendizagem utilizado, por exemplo, em tarefas como *clustering.*

Tarefas de *clustering* requerem que o sistema decida, com base em determinados critérios, de que forma é que os dados fornecidos podem ser agrupados em classes (Al-Omary and Jamil, 2006). Por outras palavras, o processo de *clustering* refere-se à divisão de dados em grupos onde cada elemento de cada grupo partilha características semelhantes com os restantes elementos e, ao mesmo tempo, pode ser facilmente distinguido de elementos de outros grupos (Gan, 2013).

2.1.2.3 Aprendizagem Semi-Supervisionada

O método de aprendizagem semi-supervisionada resulta de uma combinação entre aprendizagem supervisionada e aprendizagem não supervisionada. Por consequência, o funcionamento de um algoritmo sujeito a este tipo de aprendizagem é exponenciado em situações em que existam dados etiquetados e não etiquetados. Em circunstâncias normais, a quantidade de dados não etiquetados é superior à quantidade de dados etiquetados, uma vez que, o grande objetivo deste tipo de aprendizagem é perceber como determinados algoritmos reagem à presença de dois tipos de dados e, em seguida, tirar o máximo de proveito dessa mesma condição (Zhu and Goldberg, 2009).

2.1.2.4 Aprendizagem Por Reforço

Este tipo de aprendizagem descreve um processo de sucessivas interações entre um agente e o ambiente onde este se enquadra. O agente deve aprender, de forma progressiva, a comportar-se, de modo a maximizar as recompensas que pode obter. O próprio agente pode avaliar as suas ações baseando-se na existência e qualidade da recompensa e, consequentemente, possui a possibilidade de aprender tendo em conta a sua experiência.

Recorrendo à Figura 3 é possível descrever o estado inicial do agente pela variável S_t num determinado momento t. O agente, através do seu processo de aprendizagem, realiza uma ação A_t que o move para um novo estado, S_{t+1}, e retorna uma recompensa, R_{t+1}. Esta interação pode ser apelidada de "transição" e é a unidade nuclear de um sistema de aprendizagem por reforço.

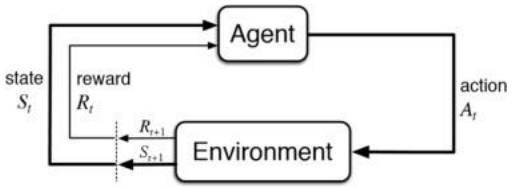

Figura 3 - Ilustração do processo de Aprendizagem por Reforço (Sutton and Barto, 1998)

Este tipo de aprendizagem contempla ainda, de forma explicita, uma suposição que descreve qualquer transição futura como sendo independente do passado tendo em conta o presente, ou seja, qualquer estado futuro é dependente apenas do estado que o antecedeu. Este tipo de pressuposto é conhecido como uma suposição de Markov, o que implica que o procedimento que suporta um processo de aprendizagem por reforço seja conhecido como um *Markov Decision Process* (MDP).

2.1.3 Deep Learning

Deep learning é uma ramificação de *machine learning* e tem como objetivo providenciar a modelos computacionais, com várias camadas de processamento, a capacidade de representar informação com recurso a múltiplos níveis de abstração, simulando o funcionamento do cérebro humano quando confrontado com objetos ou situações que requerem a sua interpretação (Voulodimos et al., 2018).

Os modelos de *deep learning* permitem que determinados sistemas adquiram conhecimento através de experiência, enquanto estabelecem uma visão hierárquica de um determinado problema, isto é, enquanto encaram o problema de forma sucessivamente mais complexa (Ian Goodfellow, Yoshua Bengio, 2017). Estes tipos de modelos usufruem de uma estrutura de camadas e a sua utilização tem sido cada vez mais frequente, por exemplo, na área de visão por computador.

O conceito de *deep learning* deriva do conceito de "redes neuronais". A investigação associada a esta área de estudo atingiu um marco histórico na década de 1940, após a proposta de um modelo composto por análises e observações das características dos neurónios presentes no cérebro de um ser inteligente, o modelo de McCulloch-Pitts (*MP*), cujo principal objetivo residia na tentativa de resolver problemas de aprendizagem de forma sistematizada (Borges Oliveira et al., 2021).

Cerca de 40 anos mais tarde, na década de 1980, foi popularizada outra proposta de grande relevância para o enriquecimento desta área de estudo, a proposta do algoritmo de retro propagação (*back-propagation algorithm*) por (Rumelhart et al, 1986).

A utilização deste algoritmo permite uma rede neuronal ajustar automaticamente os seus pesos e parâmetros, através da aplicação da *chain rule* (expressão matemática

resultante na função derivada de duas funções compostas). Após cada transição de informação entre camadas no sentido terminal da rede, é realizada uma transição em sentido contrário, denominada de retro propagação, onde os valores dos parâmetros supramencionados são ajustados. Este ajuste permitirá que, ao longo do processo de treino, os resultados do vetor de *output* se aproximem dos valores presentes no vetor de resultados desejados (Rumelhart et al., 1986).

No entanto, apesar de graduais avanços, tal como aqueles descritos anteriormente, a evolução do conceito de redes neuronais e do conceito de *deep learning*, sofreu de alguma estagnação, devido, principalmente, à capacidade de processamento limitada por parte das unidades computacionais da época e à falta de quantidade e qualidade de dados para propósitos de treino.

Já na década de 2000, mais precisamente em 2006, deu-se um novo acontecimento que reavivou o estudo deste domínio. Este avanço, ligado à área de reconhecimento de discurso foi descrito em (Hinton et al., 2012), e aliado ao aumento significativo da capacidade de processamento de unidades computacionais (por exemplo unidades de processamento gráfico, *GPU*), à redução substancial dos custos de *hardware* com grandes potencialidades e ainda às melhorias consideráveis no domínio de *machine learning* (Guo et al., 2016), justificam muito do crescimento em utilização, eficácia e assertividade dos modelos de *deep learning* nos últimos anos.

2.1.4 Redes Neuronais Convolucionais

Uma rede neuronal convolucional (ou, em inglês, *convolutional neural network, CNN*) é uma das representações mais comuns e avançadas de um modelo de *deep learning*. Modelos deste tipo estão, normalmente, associados à receção de imagens como *input*, o que permite a introdução de adaptações arquiteturais que melhoram o seu desempenho.

Na sua forma mais conceptual, uma *CNN* assemelha-se a uma rede neuronal. Ou seja, trata-se de um sistema de neurónios interconectados, divididos em camadas, que realizam um mapeamento não linear entre um vetor de *input* e um vetor de *output.* Cada neurónio, ou nó, possui um peso associado à sua conexão com outros nós e produz um *output* resultante de uma função aplicada à soma dos *inputs* que recebe de neurónios em camadas anteriores (Gardner and Dorling, 1998).

Existem três tipos de camadas distintas que compõem a arquitetura tradicional de uma rede neuronal convolucional: camadas convolucionais, camadas de *pooling* e camadas

totalmente conectadas (ou *fully-connected layers*) (Figura 4). Cada uma destas camadas realiza diferentes funções que se complementam mutuamente.

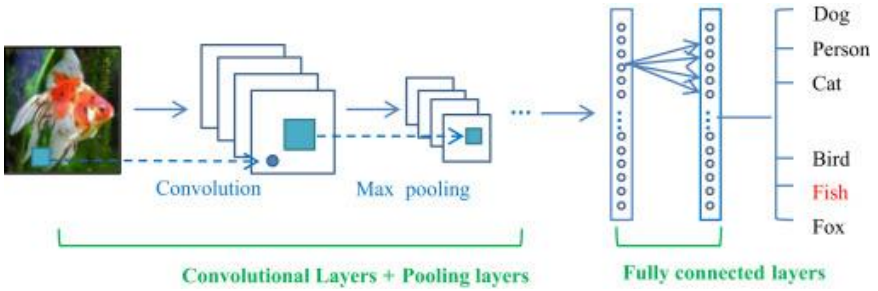

Figura 4 - Arquitetura tradicional de uma CNN (Guo et al., 2016)

2.1.4.1 Camada Convolucional

A camada convolucional é constituída por um conjunto de filtros, também denominados *kernels*. As unidades base desta camada apresentam a capacidade de "aprendizagem", garantida pela aplicação de operações matemáticas no decorrer do funcionamento da rede onde estão inseridas.

As dimensões de cada neurónio são reduzidas ao nível da sua altura e comprimento. No entanto, a dimensão associada à sua profundidade é equivalente à profundidade da imagem de *input*. A título de exemplo, um neurónio poderá apresentar como dimensões 4 pixéis de altura, 4 pixéis de comprimento e 3 de profundidade, devido aos 3 canais de cor existentes numa determinada imagem.

Durante a análise inicial da imagem de *input*, cada neurónio é movido ao longo de uma determinada região da imagem (cuja dimensão é definida através de um híper parâmetro denominado *receptive field*) e é calculado o produto entre o valor associado ao filtro e o valor proveniente do *input*. Finalizada a análise da imagem é produzido um mapa de duas dimensões que contém todos os resultados dos produtos previamente especificados, situados espacialmente na posição de cada *input* (Figura 5). Os valores deste mapa, produzido para cada neurónio, são também denominados de **ativações** (Fei-Fei, 2016).

Diferentes ativações são produzidas quando os neurónios são expostos a diferentes objetos e/ou imagens. Esta variação permite à rede neuronal distinguir *features* de *input* que poderão ser bastante simples, tais como cantos ou bordas de um determinado objeto ou, eventualmente, bastante complexos, como seres humanos ou animais.

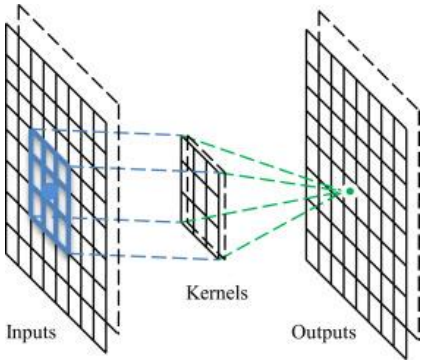

Figura 5 - Operação numa Camada Convolucional (Guo et al., 2016)

Em problemas de classificação de imagens, os procedimentos de treino aplicados a redes neuronais convolucionais têm como grande objetivo encontrar conjuntos de *kernels* que consigam extrair *features* distintivos e de alta qualidade, de modo a provocar um aumento na taxa de acerto do processo de classificação de inputs (Li et al., 2014).

2.1.4.2 Camadas de *Pooling*

Camadas de *pooling* são, por norma, inseridas entre camadas convolucionais e são responsáveis pela diminuição das dimensões de inputs que serão depois utilizados por restantes camadas. Em problemas de classificação de imagens, estas camadas são utilizadas para reajustar a altura e comprimento de cada *input*.

As operações realizadas nestas camadas são denominadas de *subsampling* ou *downsampling*, uma vez que a redução das dimensões que é realizada tem um impacto direto na perda de informação (considerada descartável) de cada imagem (Voulodimos et al., 2018).

Esta perda de informação é justificada, uma vez que, a alteração das componentes dimensionais de uma imagem permite reduzir o impacto de problemas como *overfitting* (Chin et al., 2017) e *overhead* computacional.

As duas estratégias de *pooling* mais utilizadas são *average pooling*[1] e *max pooling*[2] (Figura 6). Ambas as estratégias foram analisadas detalhadamente relativamente ao seu desempenho por (Boureau et al, 2010) e, seguidamente, foi comprovado por (Scherer et al., 2010) que a estratégia de *max pooling* pode levar a melhores resultados ao nível de

[1] Consiste no cálculo da média de todos os valores de uma determinada zona de mapa de *features;*
[2] Consiste na escolha dos maiores valores das múltiplas zonas que compõe de um mapa de *features;*

rapidez de convergência, seleção de *features* e capacidade de generalização de uma *CNN*.

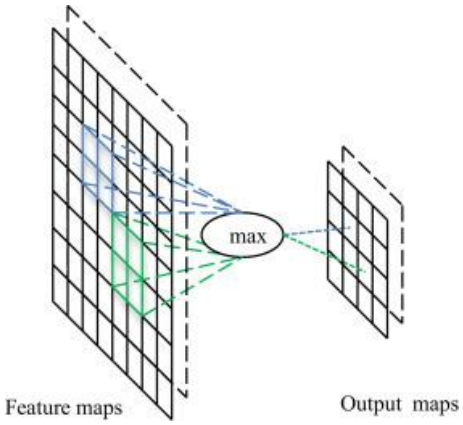

Figura 6 - Operação de Max Pooling numa Camada de Pooling (Guo et al., 2016)

As camadas de *pooling* são, das três camadas constituintes de uma rede neuronal convolucional, aquelas que têm vindo a ser estudadas em maior detalhe ao longo dos anos. Existem três tipos de abordagens relacionadas com as operações aplicadas nestas camadas, que decorrem das estratégias acima mencionadas, cada uma com propósitos e objetivos diferentes (Guo et al., 2016):

- ***Stochastic pooling*** – Abordagem proposta por (Zeiler and Fergus, 2013) na tentativa de colmatar lacunas ao nível da generalização dos dados de treino, identificadas na aplicação da estratégia de *max pooling*. As tradicionais operações determinísticas de *pooling* são substituídas por procedimentos estocásticos que consistem na escolha aleatória de pontos de ativação em cada região de acordo com uma distribuição multinominal. O seu funcionamento é semelhante ao funcionamento do procedimento de *max pooling*, no entanto, são utilizadas várias cópias dos dados de *input*, cada uma com ligeiras deformações. A natureza estocástica deste método contribui para a redução do *overfitting*.

- ***Spatial Pyramid Pooling (SPP)*** – Esta abordagem tem como objetivo proporcionar maior flexibilidade a métodos baseados em redes neuronais convolucionais (He et al., 2015). É frequente este tipo de métodos utilizarem *inputs* de dimensões fixas, no entanto, esta restrição pode ter influência negativa na capacidade de reconhecimento de dados de *input* com dimensões variadas. Deste modo, foi proposta a adição de uma camada de *spatial pyramid pooling* com a capacidade de extrair representações de dimensões fixas de dados/regiões com dimensões arbitrárias. Quando aplicada com sucesso, esta camada adicional confere um grau de robustez adicional a toda a *CNN*, permitindo-lhe interpretar imagens com diferentes escalas ou dimensões.

- ***Def-pooling*** – Embora as estratégias de *max pooling* e *average pooling* sejam capazes de dar resposta a problemas de deformação nos dados de *input*, não são capazes de realizar uma aprendizagem relativamente às restrições de deformação e modelo geométrico de determinadas partes de objetos, em problemas do domínio de visão por computador. Este tipo de abordagem tem como objetivo melhorar a forma como as deformações de diversos tipos de dados são tratadas, através da adição de uma camada de *def-pooling* especializada na aprendizagem de padrões de deformação. Esta camada tem a vantagem de poder atuar em múltiplos níveis de abstração e foi proposta por (Ouyang et al., 2014).

Tendo em conta os diferentes objetivos e propósitos associados ao design de cada uma destas abordagens, é possível combinar várias estratégias de *pooling* de modo a elevar o desempenho de uma rede neuronal convolucional.

2.1.4.3 Camada Totalmente Conectada

A camada totalmente conectada (ou *fully connected layer*) é a terceira e última componente da arquitetura de uma *CNN* (Figura 7). É nesta camada que são simulados "raciocínios" de alto nível, cada neurónio convolucional presente nesta secção está interligado com todos os neurónios da camada imediatamente seguinte (Chin et al., 2017).

Os mapas de *features* de duas dimensões que são utilizados como *input* inicial de camadas deste tipo são gradualmente transformados em vetores de *features* de apenas uma dimensão. O vetor resultante pode depois ser utilizado de acordo com um determinado número de categorias para propósitos de classificação de imagens, tal como em (Krizhevsky et al., 2012a), ou, alternativamente, pode ser interpretado como *input* para posterior processamento, tal como descrito em (Girshick et al., 2014a).

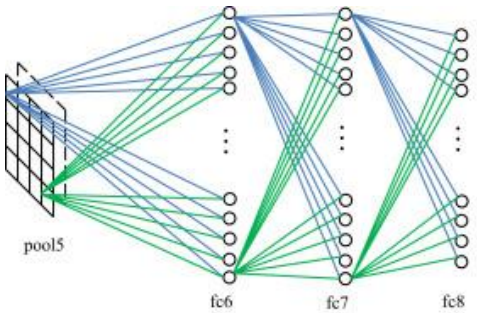

Figura 7 - Operação numa Fully Connected Layer (Guo et al., 2016)

Por norma, a estrutura deste tipo de camadas não é alterada frequentemente, no entanto, este tipo de alteração pode existir, tal como demonstrado em (Oquab et al., 2014)

onde a última *fully connected layer* foi substituída por duas outras camadas semelhantes de modo que a rede neuronal utilizada se adaptasse com sucesso às tarefas de reconhecimento visual que lhe foram propostas.

Um dos principais inconvenientes desta componente arquitetural reside no facto de serem necessários bastantes parâmetros para o seu normal funcionamento (o que implica grande esforço computacional, principalmente em períodos de treino). Esta desvantagem foi reconhecida e foram realizadas implementações com o objetivo de diminuir o número de ligações entre neurónios em camadas totalmente conectadas (Guo et al., 2016).

Cada componente das camadas convolucionais recebe informações relacionadas com o seu campo recetivo local, provenientes de camadas anteriores. Esta transição de dados permite a progressiva deteção de *features* simples tais como cantos ou bordas, que irão ser progressivamente agregados de modo a formarem composições mais completas e de mais alto nível.

De um modo geral, redes neuronais convolucionais demonstram-se capazes de obter bons desempenhos em tarefas relacionadas com reconhecimento de padrões e visão por computador, especialmente quando comparadas com métodos tradicionais de *machine learning* (Yoshua, 2009). Além do seu desempenho melhorado, o processo de treino de uma rede neuronal convolucional encontra-se bastante otimizado, devido à utilização de processos automatizados associados a variações de gradientes e retro propagação (Yoo, 2015). Estes dois fatores contribuem em grande medida para o aumento significativo da utilização deste tipo de algoritmo no passado recente (Voulodimos et al., 2018).

2.2 Visão por Computador: Breve Introdução Histórica

A génese da área de estudo denominada "visão por computador" deu-se no ano de **1966** quando foi formado um laboratório de estudo de inteligência artificial, integrado num projeto chamado *The Summer Vision Project*, no *Massachusetts Institute of Technology* (*MiT*), pelo professor Marvin Minsky (Papert, 1966). Este grupo de estudo estabeleceu o objetivo de compreender e reproduzir uma parte significativa de um sistema visual. Esta tarefa, embora complexa, foi escolhida devido ao facto de poder ser dividida em problemas mais simples, permitindo assim aos vários investigadores trabalhar de forma autónoma, com vista à criação de um sistema complexo que

representasse um avanço histórico na área de reconhecimento de padrões por unidades computacionais (Papert, 2004).

Embora o seu objetivo não tenha sido totalmente atingido, esta iniciativa pioneira marca o começo de uma área que é, atualmente, a que tem um grau de crescimento mais significativo no âmbito da inteligência artificial.

Na década de **1970** foi escrita, por David Marr, uma das mais influentes obras sobre perceção visual e sobre a forma como o cérebro humano lida com a capacidade visual. Esta obra, publicada em 1982 e intitulada "*Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information*" (Marr, 1982), é baseada em conceitos derivados da área de ciência e neurociência e estabeleceu ideias de alto nível muito relevantes no que diz respeito à estrutura hierárquica sobre a qual a visão humana é baseada.

Para (Marr, 1982), o processo visual está dividido em diversas camadas (Figura 8), a primeira camada, também conhecida por *primal sketch*, entra em ação imediatamente após a perceção de uma determinada imagem ou objeto, desconstrói o seu *input* e representa-o de forma delineada. Em seguida, a imagem inicialmente interpretada é convertida para 2 ½ dimensões, preservando a orientação dos elementos da imagem e contornando descontinuidades de acordo com o ponto de vista do visualizador. A terceira camada, denominada *3-D Model Representation*, centra-se no objeto presente na imagem e descreve a sua forma e a organização espacial (Noë, 2002).

Após a divulgação do modelo descrito por David Marr iniciou-se o desenvolvimento de modelos de reconhecimento visual suportados, principalmente, pela última fase de representação visual, uma vez que, a capacidade de conseguir representar um modelo 3D de um determinado objeto tornaria mais fácil o reconhecimento do mesmo.

Um destes modelos, conhecido por *Generalized Cylinder* e desenvolvido na universidade de *Stanford* por (Brooks and Binford, em 1981), consistia na ideia de que o nosso planeta é composto por formas simples (tais como cilindros ou blocos) e que qualquer objeto presente no mundo real é apenas uma combinação mais ou menos elaborada destas mesmas formas simples, a partir de um determinado ângulo de visão.

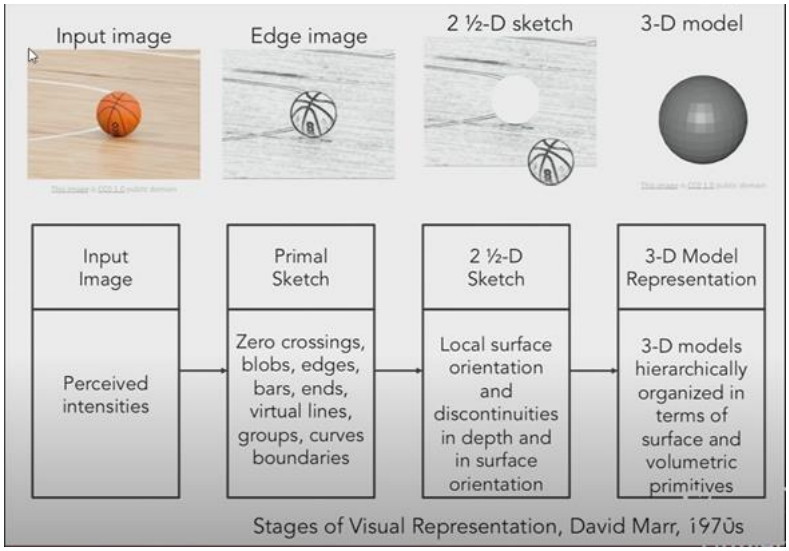

Figura 8 - Fases da Representação Visual segundo David Marr (Ebrahimpour, 2018)

Em **1997** verifica-se outro grande avanço no âmbito do agrupamento percetual (*perceptual grouping*), com a introdução do modelo *Normalized Cut* por (Shi and Malik, 2000). Este modelo distingue-se dos restantes apresentados até à data, não só pelas suas capacidades, mas também por apresentar a particularidade de interpretar fotografias e registos do mundo real, a cores. A capacidade de agrupamento percetual permite aos seres humanos categorizarem objetos no seu campo de visão de uma maneira instantânea e é uma das componentes visuais mais importantes. O modelo proposto por (Shi and Malik, 2000) representa um avanço significativo no conhecimento adquirido em relação a este conceito, no entanto, o agrupamento percetual é uma problemática que ainda não se encontra totalmente dominada nos dias de hoje.

Em **1999**, é desconstruída de forma mais aprofundada a capacidade de classificação de objetos. (David Lowe, 1999) apresenta o seu projeto *Scale Invariant Feature Transform (SIFT) & Object Recognition*, baseado na perspetiva evolutiva de que o reconhecimento de objetos não ocorre através da identificação do objeto no seu todo, mas sim através da identificação de *features* que distinguem esse mesmo objeto. A constatação da importância de *features* na tarefa de reconhecimento de objetos permitiu superar obstáculos como a rotação, translação e escala das imagens ou a presença de objetos em cenários desorganizados e com múltiplos elementos (Figura 9).

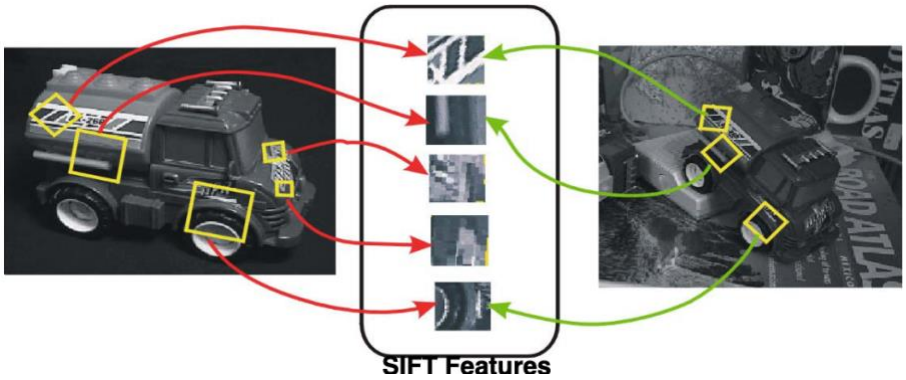

Figura 9 - Extração de features em objetos de orientação e colocação diferente (Lowe, 2004)

Posteriormente, no começo do século XXI, em **2001**, foi apresentado um novo trabalho de desenvolvimento e investigação, realizado por Viola & Jones, no âmbito da deteção facial. Este trabalho, intitulado *Robust Real-time Face Detection* (Jones and Viola, 2001), deu a conhecer o primeiro algoritmo de deteção facial em tempo real e viria a ser utilizado, em 2006, pela empresa japonesa *Fujifilm*, na criação da primeira câmara fotográfica digital com reconhecimento facial. Este trabalho é composto por três principais componentes. O primeiro componente, *Integral Image,* está relacionado com a representação e avaliação de *features* de uma imagem de forma extremamente rápida. Em seguida, dá-se a construção de um classificador simples e eficiente através da seleção de *features* relevantes, provenientes de uma biblioteca extensa, com recurso a um algoritmo de aprendizagem automática chamado *AdaBoost* (Freund and Schapire, 1999). A componente final deste modelo encontra-se associada à agregação sucessiva de classificadores gradualmente mais complexos, o que leva à seleção de *features* progressivamente mais relevantes e, consequentemente, aumenta a capacidade de deteção de faces (Viola and Jones, 2004).

(Viola and Jones, 2004), desenvolveram um algoritmo que apresentava capacidades de aprendizagem automatizada, nomeadamente ao nível da deteção de *features*, e, embora não fosse uma ferramenta de *deep learning*, a sua criação indiciava grandes avanços no estudo da visão realizada por unidades computacionais.

No seguimento dos avanços progressivos anteriormente detalhados e, numa tentativa de criar um standard global que permitisse categorizar e avaliar os vários trabalhos e modelos desenvolvidos no âmbito da área de *visão por computador*, dá-se o aparecimento do *PASCAL Visual Object Challenge*, criado por (Everingham et al., 2005) e que teve a sua primeira edição no ano de **2005**. Este desafio, constituído por milhares de imagens divididas em vinte diferentes categorias, tinha como objetivo avaliar o grau de exatidão com que diferentes algoritmos identificavam cada um dos objetos que lhes eram apresentados.

Durante os anos que se seguiram, foi possível verificar um aumento progressivo no sucesso associado à identificação das diferentes categorias de objetos, tal como é possível verificar na através Figura 10 (Everingham et al., 2015).

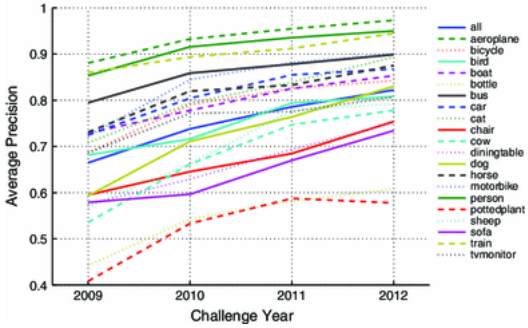

Figura 10 - Resultados do "PASCAL VOC" entre os anos de 2009 e 2012 (Everingham et al., 2015)

Não obstante a utilidade e importância do *PASCAL VOC*, as características deste desafio levaram vários investigadores a questionarem o número de classes disponíveis neste dataset, argumentando que os objetos existentes no mundo real não se cingem apenas a vinte diferentes categorias. Tendo em conta este processo de pensamento, em **2009**, surge o projeto *ImageNet* (Deng et al., 2010), com cerca de 14 milhões de imagens manualmente etiquetadas e 22 mil categorias diferentes. A partir deste projeto surgiu, em **2010**, um desafio de classificação de imagens chamado *ImageNet Large Scale Visual Recognition Challenge* (*ILSVRC*) com cerca de 1,461,406 imagens distribuídas por, aproximadamente, 1000 categorias (Russakovsky et al., 2015).

Desde o começo deste desafio que é possível verificar progressivas melhorias associadas ao erro nas tarefas de classificação (Figura 11).

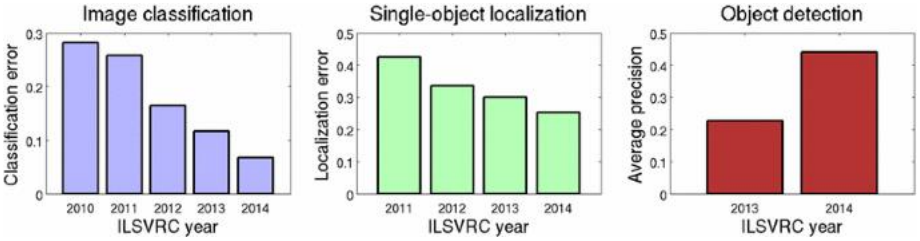

Figura 11 – Erro associado à classificação de imagens presentes no dataset ILSVRC, entre 2010 e 2014 (Everingham et al., 2015)

A equipa vencedora da primeira edição do *ILSVRC* era conhecida por *NEC* e utilizou *features* provenientes do método *SIFT* e ainda uma *Support Vector Machine* estocástica. A edição mais relevante deste desafio ocorreu no ano de **2012**, quando a equipa vencedora, *SuperVision*, utilizou, pela primeira vez, uma rede neuronal convolucional de grande escala, chamada *AlexNet* (Figura 12), para resolver o problema de reconhecimento de objetos (Krizhevsky et al., 2012b) e ultrapassou a sua concorrência por uma larga margem. Esta edição ficou marcada por uma redução significativa da taxa de erro (de 26% para 16%).

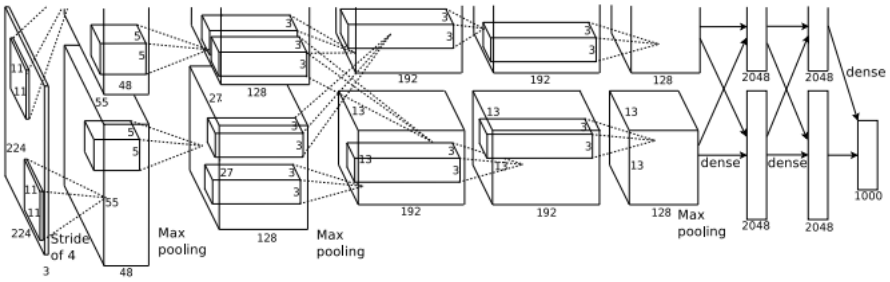

Figura 12 - Arquitetura da AlexNet (Krizhevsky et al., 2012a)

A arquitetura da rede neuronal convolucional utilizada para vencer o *ILSVRC2012* foi largamente influenciada pelo trabalho de (LeCun et al., 1998), no âmbito de algoritmos de retro propagação, gradientes e de modelos de reconhecimento de caracteres desenhados à mão (LeCun et al., 1998) (Figura 13).

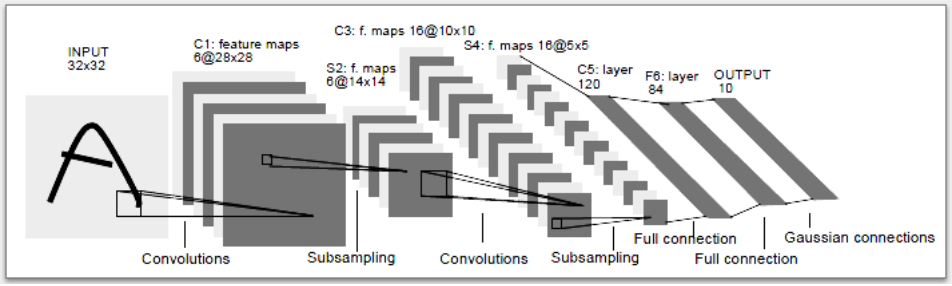

Figura 13 - Arquitetura do modelo utilizado por (LeCun et al., 1998)

Desta forma, o desafio do ano de 2012 assumiu uma **relevância histórica** na importância associada a metodologias de ***deep learning***. As edições seguintes foram compostas, e vencidas, na sua larga maioria, por submissões que integravam redes neuronais convolucionais, o que comprova a eficácia e o poder desse tipo de modelo em tarefas relacionadas com classificação de imagens e deteção de objetos (Russakovsky et al., 2015).

Desde então, a popularidade de detetores possuidores de uma arquitetura baseada em redes neuronais convolucionais tem aumentado, sendo possível classificar os métodos que utilizam esta estrutura em dois grupos (Li et al., 2020):

- **Detetores de duas fases**, tais como métodos da família *R-CNN,* assim apelidados devido ao seu modo de funcionamento, composto, em primeiro lugar, por algoritmos de proposta de regiões de interesse e, numa segunda instância, por uma rede neuronal convolucional capaz de realizar a classificação das *bounding boxes* candidatas;

- **Detetores de uma só fase**, tais como métodos da família *YOLO* (*You Only Look Once*) ou *Single Shot Multi-box Detectors (SSD)*, possuem a capacidade de prever

bounding boxes e classificar imagens diretamente e de forma síncrona, a partir de *inputs* de dimensões originais.

Na secção que se segue serão analisados e descritos exemplos de detetores de ambos os grupos acima mencionados.

2.3 Sistemas de Deteção de Objetos

Nas subsecções que se seguem serão descritas três arquiteturas distintas que representam o estado da arte de modelos de visão por computador. Cada uma destas arquiteturas apresenta diferentes características que conferem diferentes vantagens aos modelos que as aplicam.

2.3.1 You Only Look Once

Detetores que fazem uso da arquitetura *You Only Look* Once, ou, alternativamente, *YOLO*, apresentam a particularidade de realizarem classificações e definições de *bounding boxes* através de um processo que envolve apenas uma análise da imagem de *input* – daí a sua designação, *You Only Look Once*. A capacidade de produzir *outputs* através desta única e rápida análise garante o funcionamento deste tipo de algoritmos em tempo real.

Os métodos *YOLO* interpretam de uma nova forma o problema de deteção de objetos, encarando-o como um problema de regressão e produzindo *outputs* em forma de probabilidades diretamente a partir da análise dos pixéis de uma imagem, com recurso a regiões resultantes do delineamento de vários perímetros (*bounding boxes*) pertencentes a diferentes classes (Redmon et al., 2015). As previsões e classificações são realizadas com recurso a uma única rede neuronal convolucional (Figura 14).

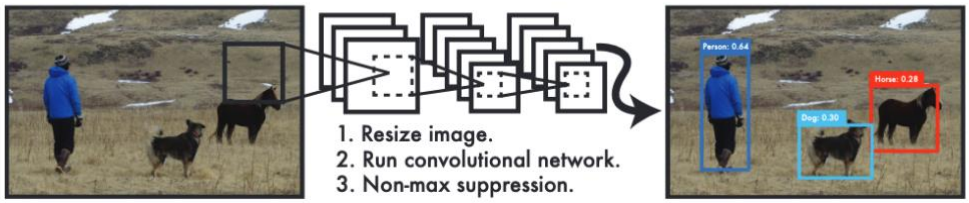

Figura 14 - Sistema de Deteção de métodos da família YOLO (Redmon et al., 2015)

O desenvolvimento de modelos utilizando este tipo de estrutura base tem vindo a ser realizado de forma iterativa, com sucessivas versões que apresentam melhorias face às suas predecessoras. As subsecções que se seguem apresentam uma breve descrição das funcionalidades e principais pontos de interesse de cada uma destas versões.

2.3.1.1 YOLO version 1

O processo de funcionamento de um detetor desta categoria inicia-se pela divisão da imagem de input numa grelha de dimensões S x S. Em cada célula desta grelha são definidas B bounding boxes, associadas a um valor numérico, resultante da expressão representada na Equação 1, que permite o cálculo da confiança associada à previsão.

Equação 1 - Expressão para cálculo da confiança da previsão de uma *bounding box*

$$C = \Pr(Object) * IOU_{pred}^{truth}$$

A variável IoU (*intersection over union*) pode apresentar valores entre 0 e 1 e diz respeito a dois conceitos complementares: interseção entre a área de uma determinada *bounding box* e a área onde se encontra presente o objeto e união entre ambas essas áreas.

Durante a conceção das *bounding boxes*, cada célula encontra-se também responsável por prever a categoria associada ao objeto que pode (ou não) estar presente no seu perímetro de análise (R. Huang et al., 2019).

A capacidade de aprendizagem deste modelo é introduzida com recurso a uma função de perda (*loss function*), cujo *output* permite extrair conclusões relativamente ao desempenho do detetor. Um maior valor de perda está associado a níveis de desempenho indesejados.

Além de poderem ser treinados com imagens de tamanho original, os métodos desta família possuem outras vantagens:

- São extremamente rápidos uma vez que as deteções são realizadas com base numa única análise do *input* por parte da rede neuronal;
- São capazes de analisar o contexto completo da imagem e de guardar e utilizar essa informação de forma eficiente, devido ao facto de terem acesso à imagem de *input* completa durante as fases de treino e de teste;
- São capazes de aprender e realizar generalizações que lhes conferem maior robustez e maior grau de adaptabilidade quando confrontados com situações novas e inesperadas, por exemplo, quando obrigados a generalizar de imagens naturais para obras de arte.

Independentemente das vantagens identificadas anteriormente, métodos da família *YOLO* apresentam, na sua versão inicial, resultados abaixo da concorrência ao nível da taxa de acerto na localização espacial de objetos, principalmente quando esses objetos apresentam reduzidas dimensões. Além disso, estes algoritmos apresentam dificuldades em processos de generalização que envolvem cenários específicos, tais como generalizações que têm como resultado imagens com *aspect ratios* [3] ou configurações invulgares (Chen et al., 2018).

2.3.1.2 YOLO version 2

A segunda versão de detetores deste grupo apresenta melhorias relacionadas com as lacunas apresentadas pela versão anterior, tanto em tarefas de deteção de objetos de reduzidas dimensões como o nível da precisão da correta localização de determinados objetos.

O método YOLOv2 introduz a capacidade de reprocessamento dos dados de input através da normalização por lotes (batch normalization) nas camadas convolucionais e utiliza perímetros predefinidos (anchor boxes) modelados de acordo com objetos existentes no mundo real, para facilitar a previsão das bounding boxes. A

Tabela 1 apresenta dados relativos à aplicação de diferentes detetores ao desafio PASCAL VOC 2012 e comprova a influência que os aperfeiçoamentos realizados nesta versão tiveram na precisão média (mean average precision, mAP) da classificação de diferentes classes de objetos (Redmon and Farhadi, 2016).

Tabela 1 - Resultados do desafio PASCAL VOC 2012

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

A diminuição no âmbito do tempo de execução resulta da substituição do extrator de *features VGG-16* por um novo sistema, baseado na arquitetura *Googlenet*, com níveis de precisão ligeiramente inferiores, mas com notórias melhorias ao nível da quantidade de operações necessárias para extração de características distintivas. Este novo sistema, denominado *Darknet-19*, confere aos métodos desta versão uma velocidade de processamento superior.

[3] Relação entre a altura e comprimento de uma imagem.

2.3.1.3 YOLO version 3

A terceira versão de detetores *YOLO* apresenta uma nova substituição do sistema de extração de *features*, desta vez para uma instância mais avançada do sistema utilizado anteriormente, o *Darknet-53*. Este sistema apresenta, como o próprio nome indica, 53 camadas convolucionais, que o tornam mais poderoso que o seu predecessor (*Darknet-19*) e mais eficiente do que modelos concorrentes como o *ResNet-101* ou o *ResNet-152*. Na Tabela 2 é possível visualizar uma comparação entre estes sistemas, quando avaliados com recurso ao *dataset ImageNet*. A coluna "BFLOP/s" (*billion floating point operations per second*) constata o poderio e eficácia do sistema *Darknet-53*, uma vez que simboliza a sua capacidade de processamento e eficiência na utilização de recursos computacionais e na realização de operações convolucionais (Redmon and Farhadi, 2018).

Tabela 2 - Comparação de sistemas de extração de features em termos de taxa de acerto, billion of operations, billion floating point operations per second e frames por segundo (Redmon and Farhadi, 2018)

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

A desempenho de detetores *YOLOv3*, quando avaliado com recurso ao *dataset COCO* (Lin et al., 2015), indica resultados positivos de precisão média para um *threshold IoU* de 0.5 (AP_{50} na Tabela 3), no entanto, estes resultados pioram à medida que o *threshold* é aumentado, o que revela lacunas ao nível da capacidade de localização deste tipo de algoritmo. É também possível verificar melhorias ao nível da deteção de objetos de reduzidas e médias dimensões, principalmente comparando com detetores *YOLOv2* (colunas "AP_S" e "AP_M").

Tabela 3 - Desempenho de vários métodos com diferentes sistemas de extração de *features* quando testados com recurso ao *dataset COCO* (Redmon and Farhadi, 2018)

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Apesar dos detetores *RetinaNet* apresentarem valores de precisão média superiores quando comparados com qualquer outra gama de detetores, é relevante salientar que o tempo de processamento destes métodos é bastante superior ao de algoritmos como *YOLOv3*, tal como é possível constatar através da análise da Figura 15.

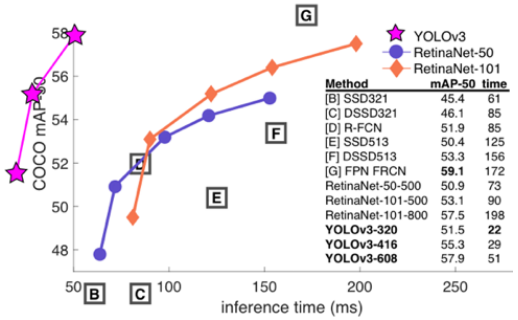

Figura 15 - Velocidade de execução em milissegundos e taxa de acerto (IoU = 0.5) de diferentes detetores (Redmon and Farhadi, 2018)

2.3.1.4 YOLO version 4

A quarta versão deste tipo de modelo exige menos potência computacional ao nível das unidades de processamento gráfico e apresenta melhorias significativas no que diz respeito à *mean average precision* e a rapidez de execução (cerca de 10% e 12% respetivamente) (Bochkovskiy et al., 2020a). O processo de treino dos modelos desta versão foi especialmente delineado para ser realizado com recurso a um único *GPU*, de modo que seja possível realizar deteções num número mais alargado de ambientes.

A versão *YOLOv4* implementa um conjunto de estratégias denominado "*Bag of Freebies*", cujo nome deriva da expressão "*freebies*" (termo utilizado para descrever a obtenção de um produto ou serviço de forma gratuita) devido ao facto de melhorar o desempenho do algoritmo sem provocar um aumento significativo no tempo de inferência. Algumas destas estratégias, exemplificadas na Figura 16, já faziam parte do arsenal de ferramentas utilizado por diferentes modelos, como é o caso do processo de *data augmentation*, através do qual se altera o conjunto de treino de modo a expor o modelo a imagens que, de outra forma, não seriam analisadas (Bochkovskiy et al., 2020a).

Por outro lado, este conjunto apresenta também estratégias inovadoras, como por exemplo *Self-Adversarial Training (SAT)*, que consiste na realização de alterações subtis nas áreas das imagens mais vezes analisadas pelo modelo e ainda *mosaic data augmentation* (Figura 17) que utiliza colagens de quatro imagens numa tentativa de refinar a granularidade das deteções realizadas (Solawetz, 2020).

De modo a compreender quais conjuntos de estratégias funcionavam melhor em conjunto na tentativa de elevar o desempenho do modelo, os seus criadores realizaram diversas sessões de treino e teste com diferentes conjuntos de estratégias, utilizando o *dataset COCO* (Tabela 4).

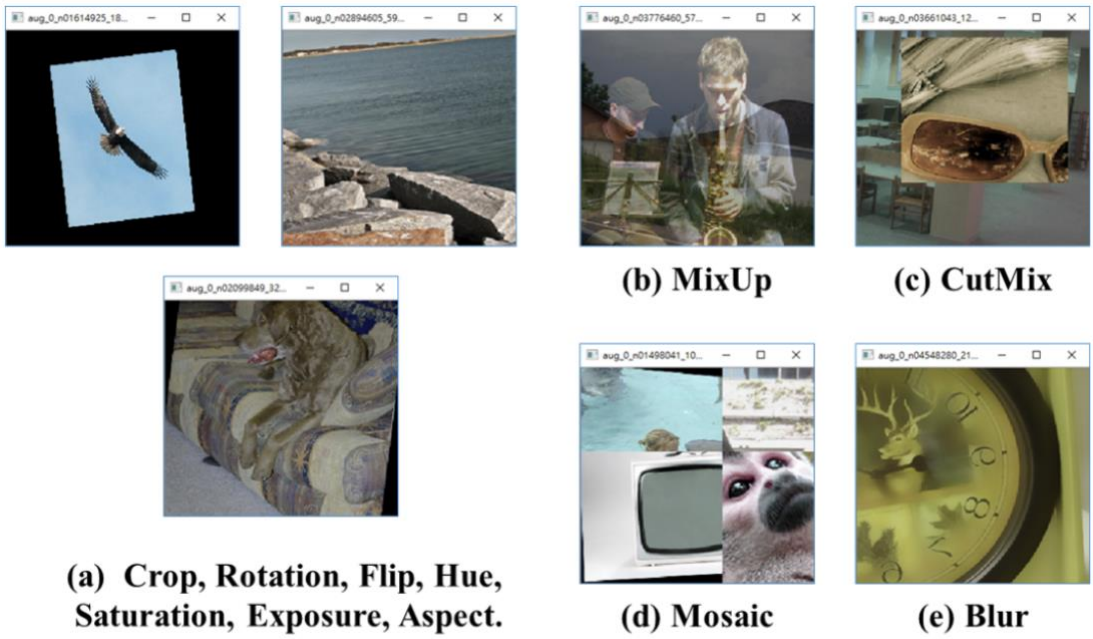

Figura 16 - Diferentes estratégias de Data Augmentation (Bochkovskiy et al., 2020a)

Figura 17 - Exemplo de imagens resultantes do processo de mosaic data augmentation (Bochkovskiy et al., 2020a)

Tabela 4 - Resultados da utilização de diferentes conjuntos de estratégias pertencetes ao "Bag of Freebies" (Bochkovskiy et al., 2020a)

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	**94.0%**
	✓						**78.0%**	**94.3%**
		✓					**78.1%**	**94.5%**
			✓				77.5%	93.8%
				✓			**78.1%**	**94.4%**
					✓		64.5%	86.0%
						✓	**78.9%**	**94.5%**
	✓	✓		✓			**78.5%**	**94.8%**
	✓	✓		✓		✓	**79.8%**	**95.2%**

De modo semelhante, são também aplicadas técnicas que aumentam marginalmente o tempo de inferência enquanto melhoram consideravelmente o desempenho do modelo. Algumas destas técnicas são, por exemplo, a função de ativação *"mish"*, o sistema de escolha de *bounding boxes "DIoU NMS"*, o sistema de *"cross mini-batch normalization (CmBN)"* (otimizado para execução em máquinas com um único GPU) e, por fim, a técnica *"DropBlock regularization"* (Figura 18), que oculta determinadas secções das imagens de treino de modo a forçar o modelo a reconhecer diferentes *features*. Este conjunto de estratégias constitui o chamado "*Bag of Specials*" (Solawetz, 2020).

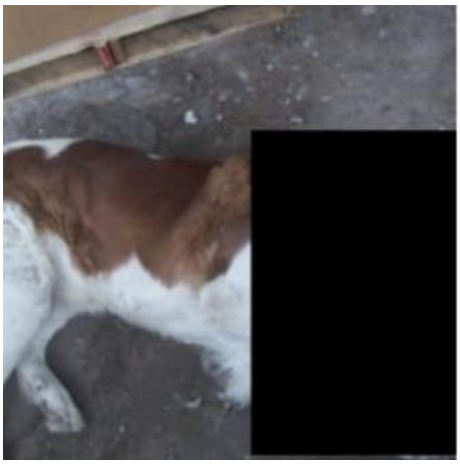

Figura 18 - Exemplo de DropBlock regularization(Bochkovskiy et al., 2020a)

Ambos estes conjuntos de estratégias, utilizados para elevar o funcionamento e eficácia da quarta versão dos modelos *YOLO*, contribuem diretamente para a sua versatilidade e proliferação, garantindo-lhe uma combinação de *frames per second* e *mean average*

precision acima da média, quando avaliado, por exemplo, com recurso ao *dataset COCO* (Figura 19).

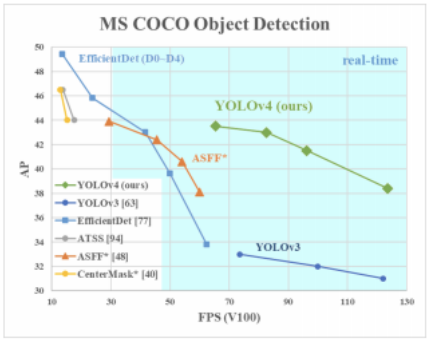

Figura 19 - Desempenho do modelo YOLOv4 no *dataset COCO* (Bochkovskiy et al., 2020a)

2.3.2 Convolutional Neural Network

Métodos da família *CNN* estão na base dos grandes avanços realizados num curto espaço de tempo na área de deteção de objetos e classificação de imagens (Shetty, 2016). Modelos como *Fast/Faster R-CNN* (Girshick, 2015; Ren et al., 2016) são conceptualmente intuitivos e oferecem elevado grau de robustez e flexibilidade enquanto mantêm níveis de desempenho aceitáveis.

As múltiplas versões de modelos deste grupo representam sucessivas melhorias, tanto a nível arquitetural como a nível de funcionamento, e encontram-se descritas sucintamente nas secções seguintes.

2.3.2.1 R-CNN

A primeira versão de detetores deste tipo, denominada *R-CNN*, tinha com o objetivo o aperfeiçoamento da taxa de acerto em tarefas de deteção de objetos. Estes detetores eram compostos por três módulos distintos (Figura 20): um primeiro módulo com a função de propor regiões de interesse na imagem (independentemente da sua categoria), um segundo módulo composto por uma rede neuronal convolucional com a função de extrair *features* das regiões sugeridas e um terceiro componente que integrava várias *SVM's* com a capacidade de classificar cada região (Girshick et al., 2014b).

Embora capaz de excelentes resultados ao nível da taxa de acerto na deteção de objetos, o método *R-CNN* apresentava lacunas relacionadas com a demora do tempo de deteção e com necessidade significativa de recursos computacionais.

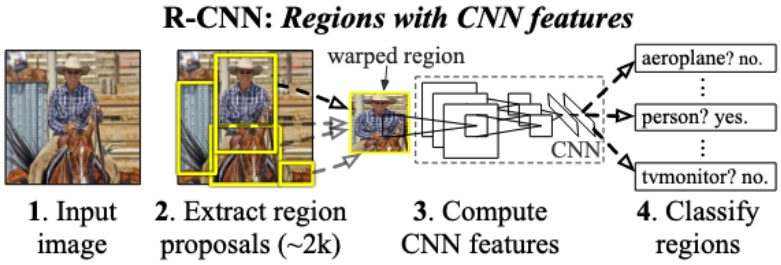

Figura 20 - Deteção de Objetos utilizando método R-CNN (Girshick et al., 2014b)

2.3.2.2 Fast R-CNN

Como proposta para mitigar os efeitos das lacunas acima mencionadas, surgiu o método *Fast R-CNN*. Este método propõe uma nova camada, chamada *Region of Interest Pooling Layer* (Figura 21) cuja função é extrair vetores de *features* de todas as regiões de interesse propostas. Esta nova camada permite que os recursos computacionais sejam partilhados e que o processo de deteção de objetos seja mais rápido, uma vez que se torna possível analisar múltiplas regiões de interesse ao mesmo tempo (funcionalidade não disponível em algoritmos *R-CNN*). Métodos deste tipo também dispensam a utilização tão intensa de memória da unidade computacional, uma vez que não recorrem ao armazenamento de *features* através da cache (técnica de armazenamento utilizada no sistema arquitetural *R-CNN*) (Girshick, 2015).

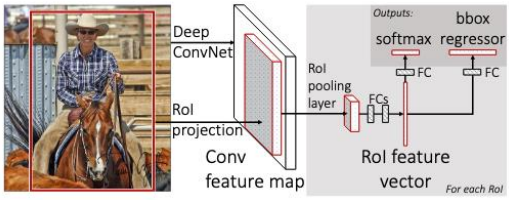

Figura 21 - Deteção de Objetos utilizando Fast R-CNN, com enfâse na camada RoI Pooling (Girshick, 2015)

2.3.2.3 Faster R-CNN

Este método propõe um modelo composto por dois módulos: um módulo de proposta de regiões de interesse através da utilização de uma *fully convolutional network* (chamado *Region Proposal Network, RPN*) e um outro módulo que consiste num detetor *Fast R-CNN.* O módulo *RPN* auxilia o módulo *Fast R-CNN,* indicando-lhe onde deve procurar determinados *features*.

Este tipo de detetores (Figura 22) possuem a capacidade de dividir os recursos computacionais entre os cálculos realizados nos módulos, o que contribui para uma maior eficiência na execução (Ren et al., 2016).

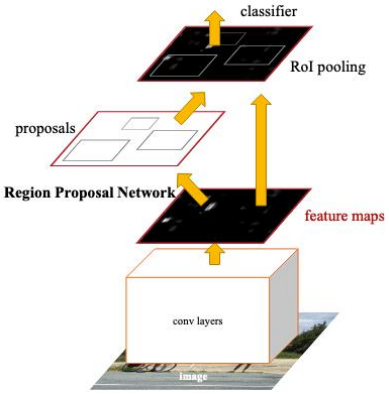

Figura 22 - Deteção de Objetos utilizando Faster R-CNN - destaque para o módulo RPN (Ren et al., 2016)

2.3.2.4 Mask R-CNN

Este algoritmo é construído com base no seu predecessor – *Faster R-CNN*. Enquanto este último apresenta como *output* uma categoria e uma *bounding box* para cada imagem de *input*, os métodos *Mask R-CNN* apresentam ainda um terceiro valor de *output*: a máscara do(s) objeto(s) presentes na imagem. Esta máscara é obtida através de dois tipos de segmentação de imagem, realizados numa ramificação arquitetural adicionada ao esquema base deste método (Odemakinde, 2021). Estes tipos de segmentação são conhecidos por (Figura 23):

- **Segmentação Semântica –** Enquadra cada pixel num determinado conjunto de categorias, sem realizar diferenciação entre as várias instâncias de objetos presentes nessa categoria;

- **Segmentação por Instância** – Deteta e diferencia cada instância de objetos existentes na imagem.

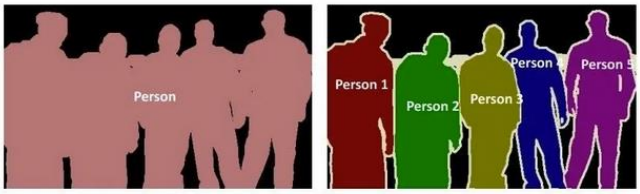

Figura 23 - Comparação entre segmentação semântica e segmentação por instâncias (Odemakinde, 2021)

Funcionalmente, o ramo que cria as máscaras de segmentação é constituído por uma *Fully Convolutional Network* aplicada a cada região de interesse, *pixel* a *pixel* (He et al., 2017).

A camada *RolAlign,* representada na Figura 24, tem como objetivo a preservação espacial da localização de cada *pixel*, de modo que a imagem de *input* permaneça espacialmente alinhada com a imagem de *output*. Esta camada representa também uma melhoria face a métodos da *framework Faster R-CNN*, que não apresentavam a capacidade de preservar este alinhamento.

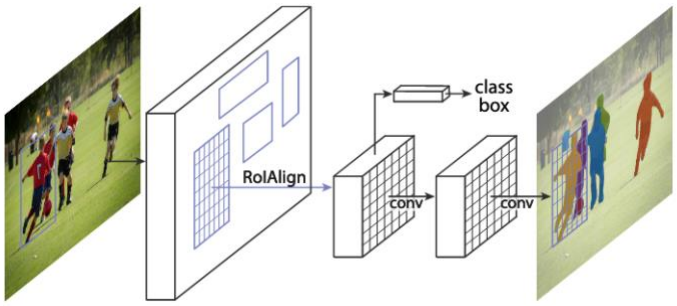

Figura 24 - Arquitetura de um método Mask R-CNN (Odemakinde, 2021)

A aplicação de detetores *Mask R-CNN* resulta na obtenção de uma máscara e de um *score* relativo à classificação de cada objeto.

2.3.3 Single Shot Multi-Box Detector

A arquitetura *Single Shot Multi-box Detector* foi proposta por (Liu et al., 2016) em 2016 (Figura 25). Este tipo de detetor, caracterizado pela necessidade de percorrer toda a rede convolucional apenas uma vez, surge através da combinação de funcionalidades pertencentes a métodos da família *CNN* e da família *YOLO*. A utilização de *anchor boxes* provenientes do primeiro grupo é complementada pela interpretação do tema de classificação de imagens como um problema de regressão (Li et al., 2020).

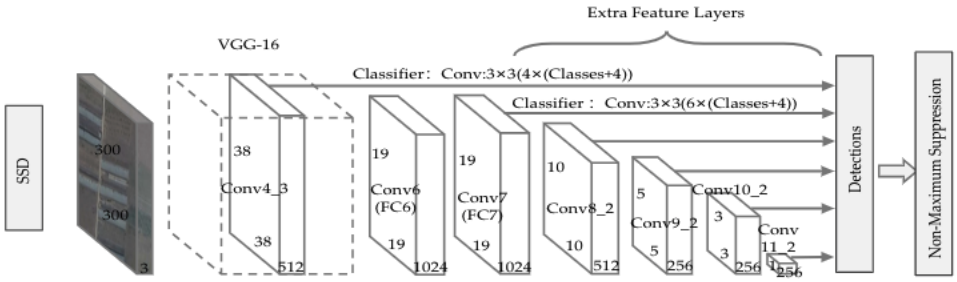

Figura 25 - Arquitetura de um detetor SSD (Li et al., 2020)

O sistema de extração de *features* utilizado é o *VGG-16*, presente, por exemplo, em detetores *Faster R-CNN*. Este sistema compõe a rede base que inicia a estrutura em pirâmide característica de métodos *SSD* (Morera et al., 2020).

São utilizadas técnicas de deteção multireferência e multiresolução que permitem a definição de múltiplas *anchor boxes* de diferentes tamanhos e resoluções por toda a imagem de *input.* Estes perímetros são a base da criação de *bounding boxes* e serão posteriormente utilizados por diferentes camadas para deteção e classificação de objetos. A adoção de mapas de *features* multidimensionais permite a este tipo de rede detetar objetos de diferentes tamanhos ao mesmo tempo, uma vez que, à medida que a resolução espacial dos mapas de *features* diminui, os detalhes associados a cada *feature* adquirem um grau de abstração superior (Li et al., 2020).

Durante o processo de treino inicial de algoritmos *SSD*, as *bounding boxes* de diferentes resoluções e tamanhos são comparadas com as regiões onde se encontram presentes os objetos a classificar na imagem de *input*. Este processo ocorre na tentativa da obtenção de uma *bounding box* específica com o maior grau de *IoU (intersection over union)* para cada objeto. As restantes *bounding boxes* são comparadas com as restantes regiões delimitadoras dos objetos e marcadas como exemplos negativos, caso o seu grau de *IoU* não seja superior a um determinado *threshold*, através de um procedimento denominado *non-maximum suppression* (Neubeck and van Gool, 2006) . É recomendado que sejam utilizados exemplos negativos e exemplos positivos de *bounding boxes* num rácio de 3:1 de modo a fornecer informação suficiente ao algoritmo relativamente aos locais onde deve (ou não) tentar classificar objetos (Forson, 2017).

2.3.4 Sistematização das Características dos Detetores

Os métodos acima descritos de forma breve apresentam modos de funcionamento diferentes, embora possam ser aplicados a múltiplos problemas semelhantes. Nas secções que se seguem, serão comparadas as características arquiteturais e o

desempenho dos modelos estudados previamente. Os dados utilizados para este estudo comparativo encontram-se disponíveis, por exemplo, nos documentos de apresentação de cada versão de cada algoritmo, como via de constatação das melhorias e novas capacidades apresentadas.

2.3.4.1 Arquitetura

Uma substancial parte das diferenças existentes entre os detetores da família *YOLO* e detetores da família *CNN* reside nas diferentes arquiteturas de cada uma das *frameworks*.

Os métodos da família *CNN* apresentam *pipelines* complexas que integram vários tipos de componentes, incluindo: algoritmos de *Selective Search* (Girshick, 2014) (Uijlings et al., 2013) que geram potenciais regiões delimitadoras, redes neuronais convolucionais que extraem *features* e *SVM* que classificam cada uma das regiões previamente delimitadas. A complexidade e a necessidade de realizar operações computacionalmente exigentes resulta, frequentemente, em lentidão no processo de treino deste tipo de detetores.

Na tentativa de colmatar esta lacuna surgiram métodos como *Faster R-CNN* caracterizados por uma partilha de recursos computacionais mais eficiente e que substituem os algoritmos de *Selective Search* por redes neuronais. Embora esta restruturação apresente um efeito positivo no que diz respeito ao tempo necessário para treinar os detetores desta família, estes ainda apresentam dificuldades em produzir resultados em tempo real (Girshick, 2015).

Os métodos baseados na *framework YOLO* partilham algumas similaridades com os procedimentos acima descritos, nomeadamente ao nível da classificação de *bounding boxes* utilizando procedimentos convolucionais. No entanto, estes tipos de sistema impõem constrangimentos espaciais ao nível da divisão da imagem de *input* (diminuindo a possibilidade de ocorrerem deteções duplicadas) e impõem também constrangimentos ao nível do número de regiões delimitadoras criadas. Estas condicionantes, quando unificadas num único modelo, criam uma *pipeline* cuja eficácia é potencializada pelo seu *design* (Redmon et al., 2015).

É relevante salientar que as restrições aplicadas pela *framework YOLO* apresentam efeitos negativos ao nível da exatidão das suas classificações. Uma vez que cada pequena região da imagem pode apenas resultar em duas *bounding boxes* e que essas *bounding boxes* só poderão estar associadas a uma categoria, este tipo de método demonstra dificuldades a classificar objetos distintos e de reduzidas dimensões, principalmente quando estes se encontram agrupados.

Os detetores *SSD* não possuem *fully connected layers* e usufruem da utilização de *anchor boxes* e de mecanismos de obtenção de *bounding boxes* com um nível de granularidade superior, o que lhes permite colmatar lacunas existentes em métodos *YOLO*, principalmente ao nível da deteção de objetos de variadas dimensões (Figura 26). A utilização de um sistema de extração de *features VGG-16* estabelece uma crucial diferença arquitetural entre estes detetores de uma única fase. *Single Shot Multi-box Detectors* apresentam ainda uma arquitetura em pirâmide, composta por vários mapas de *features* multidimensionais, peças fundamentais na classificação correta de objetos de diferentes dimensões em imagens de várias resoluções (Tsang, 2018).

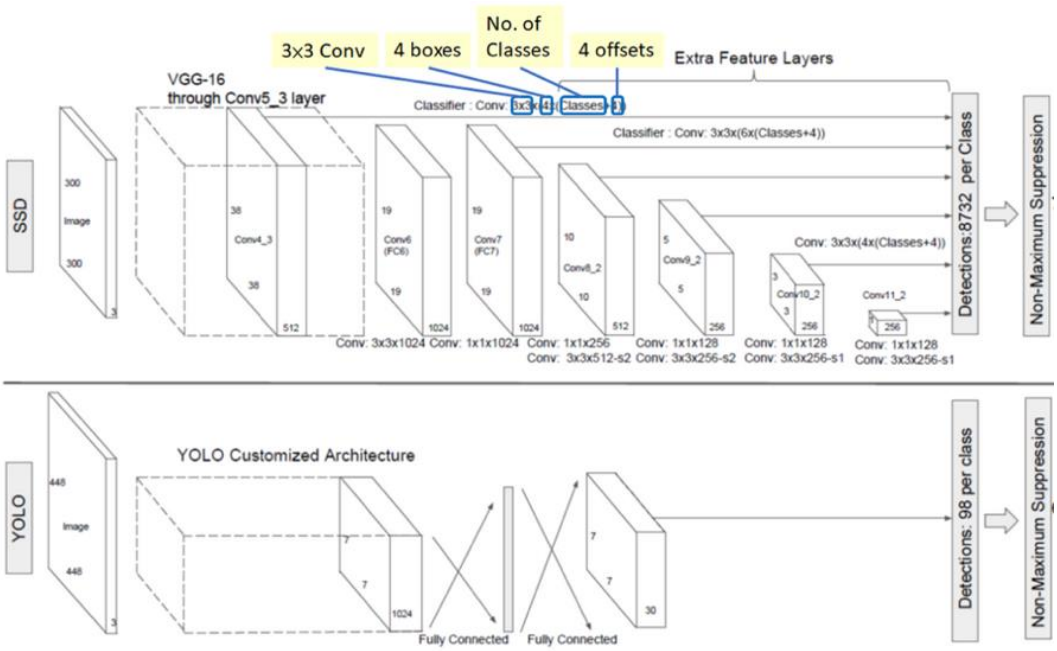

Figura 26 - Diferenças arquiteturais entre métodos SSD e YOLO

2.3.4.2 Desempenho

Analisando estes métodos ao nível do seu desempenho nas edições de 2007 e 2012 do desafio *Pascal Visual Object Classes* é possível extrair conclusões relativamente às condições ideias de aplicabilidade de ambos os algoritmos (Redmon et al., 2015).

A Tabela 5 apresenta uma distinção entre detetores que atuam em tempo real e detetores que não atuam em tempo real e, para cada um desses detetores, discrimina qual o dataset utilizado para o seu treino, a precisão média (*mean average precision, mAP*) e o número de frames por segundo (*frames per second, FPS*). Com recurso a estes dados é possível verificar que os modelos *YOLO* são, de forma inequívoca, os detetores em tempo real mais precisos, o método *YOLO*, em especial, apresenta um mAP de

63.4% (uma melhoria de mais de 10 unidades percentuais relativamente ao seu predecessor – *Fast YOLO*).

Ao nível de métodos que não atuam em tempo real, algoritmos da família *CNN* apresentam variações significativas ao nível da precisão média.

O método *Fast R-CNN* atinge um nível de precisão de 70% que, embora positivo quando comparado com grande parte dos restantes métodos desta categoria, é contrastado por um tempo de execução elevado, comprovado ao número extremamente reduzido de *frames* por segundo que são analisadas – 0.5.

A versão seguinte, *Faster R-CNN*, é representada na tabela através de duas variações: uma variação treinada com recurso ao algoritmo de classificação de imagem *VGG-16* e uma outra variação conhecida por *Zeiler-Fergus Faster R-CNN* (Ren et al., 2016).

A primeira variação – *Faster R-CNN VGG-16* – é mais precisa do que o método YOLO (diferença de precisão de cerca de 10 unidades percentuais), embora apresente um tempo de execução seis vezes superior. Já a segunda variação – *Faster R-CNN ZF* – é ligeiramente menos precisa quando comparada ao método YOLO (diferença de cerca de uma unidade percentual) e é também 2.5 vezes mais lenta.

Tabela 5 - Tabela de resultados de vários detetores quando submetidos ao desafio PASCAL VOC 2007 (Redmon et al., 2015)

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Analisando a Figura 27, que sintetiza os diferentes tipos de erros cometidos pelos métodos *Fast R-CNN* e YOLO durante o teste *Pascal VOC 2007,* é possível confirmar determinadas características salientadas durante a descrição dos algoritmos destas duas famílias. Detetores *YOLO* demonstram dificuldade em localizar objetos corretamente – este tipo de erro representa mais de metade da totalidade dos erros cometidos, acumulando um total de 19% das suas deteções. Por outro lado, detetores *Fast R-CNN* apresentam melhores resultados em tarefas de localização, que contrastam

com a sua taxa de 13.6% na deteção de falsos positivos (classificação de objetos que não existem na imagem de *input*).

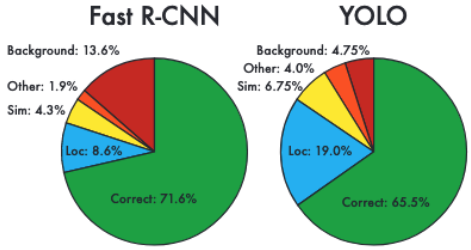

Figura 27 - Fast R-CNN vs YOLO: Percentagem de Deteções no desafio PASCAL VOC 2007 (Redmon et al., 2015)

Analisando os resultados do *PASCAL VOC 2012* (Tabela 6) que têm apenas em conta a precisão média (desprezando o tempo de execução), é possível verificar que o detetor *YOLO* atinge uma precisão média de 57.9% que o coloca na metade inferior da tabela classificativa. Este resultado é influenciado negativamente pela dificuldade que o método apresenta na classificação de objetos de reduzidas dimensões, tais como garrafas (coluna "*bottle*"), ovelhas (coluna "*sheep*") ou televisões (coluna "*tv*") e é influenciado positivamente pelos resultados extremamente positivos na deteção de objetos como comboios (coluna "*train*") e gatos (coluna "*cat*").

Na mesma tabela é possível verificar que a precisão média de detetores *Faster R-CNN* é de cerca 70.7%, o que representa um crescimento de aproximadamente 13% face ao método *YOLO*. Este detetor apresenta a maior taxa de acerto de toda a tabela em objetos como gatos (coluna "*cat*"), cães (coluna "*dog*") e comboios (coluna "*train*"), demonstrando fragilidades na deteção de imagens com garrafas (coluna "*bottle*") e plantas (coluna "*plant*").

Tabela 6 - Tabela Classificativa do desafio PASCAL VOC 2012 (Redmon et al., 2015)

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Apesar dos resultados deste desafio parecerem favorecer claramente os detetores *Faster R-CNN*, é importante salientar novamente que não é contabilizado o tempo de

execução relativo à obtenção de resultados. Tendo em conta a *pipeline* da estrutura arquitetural complexa que suporta este tipo de métodos, é seguro afirmar que o tempo de execução do algoritmo *Faster R-CNN* é comprometido em prol da precisão. Este tipo de comprometimento, realizado no sentido inverso por detetores *YOLO*, pode ou não ser vantajoso, dependendo das características do problema ao qual os detetores são aplicados.

O mesmo *dataset* foi aplicado a detetores *Single Shot Multi-box Detector*, na perspetiva de avaliar o seu desempenho comparativamente aos resultados obtidos por outros detetores (de única ou dupla fases). A Tabela 7 ilustra os resultados obtidos. Detetores *SDD* treinados com recurso a imagens de dimensões 300 x 300 atingem resultados de precisão média superiores aos obtidos por detetores *Fast/Faster R-CNN*. Aumentando as dimensões das imagens para 512 x 512, os resultados obtidos são ainda mais esclarecedores, métodos *SSD* apresentam uma diferença de mais de vinte pontos percentuais ao nível de *mAP* face a métodos YOLO e uma diferença de 4.1% na mesma medida de avaliação, face a métodos *Faster R-CNN* treinados com imagens provenientes dos *datasets PASCAL VOC2007, PASCAL VOC2012* e *COCO* (Liu et al., 2016).

Tabela 7 - Resultados do desafio PASCAL VOC2012. Comparação entre métodos RCNN, YOLO e SSD (Liu et al., 2016)

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast[6]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster[2]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Faster[2]	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO[5]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	88.4	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

Avaliações realizadas com recurso unicamente ao *dataset COCO* que, por norma, apresenta objetos de dimensões mais reduzidas face aos presentes em imagens provenientes de *datasets PASCAL* resultaram nos dados presentes na Tabela 8. A adaptação de métodos *SSD* a variações de dimensões dos objetos a classificar, através da diminuição das dimensões das *anchor boxes* utilizadas, permite a obtenção de resultados superiores aos apresentados pelos restantes sistemas de deteção, principalmente quando o tamanho das imagens de *input* é de 512 x 512.

Tabela 8 - Resultados referentes ao teste COCO. Comparação de detetores Fast/Faster R-CNN, YOLO e SSD ao nível de mAP (Liu et al., 2016)

Method	Data	Mean average precision		
		0.5	0.75	0.5:0.95
Fast R-CNN [6]	train	35.9	-	19.7
Fast R-CNN [21]	train	39.9	20.5	19.4
Faster R-CNN [2]	train	42.1	-	21.5
Faster R-CNN [2]	trainval	42.7	-	21.9
Faster R-CNN [22]	trainval	45.3	24.2	23.5
ION [21]	train	42.0	23.0	23.0
SSD300	trainval35k	41.2	23.2	23.4
SSD512	trainval35k	**46.4**	**26.7**	**27.7**

De modo a avaliar comparativamente o tempo de execução de detetores *SSD* face a detetores *Faster R-CNN* e *YOLO*, estes métodos foram testados com recurso ao *dataset PASCAL VOC2007,* obtendo-se assim dados relativos à precisão média e *frames* por segundo apresentados por cada algoritmo. Através da análise da Tabela 9, é possível verificar que tanto o modelo *SSD300* como o modelo *SSD512* (cuja nomenclatura faz referência às dimensões das imagens utilizadas durante o período de treino) apresentam valores de precisão média e de *frames* por segundo superiores aos apresentados por métodos *Faster R-CNN* (independentemente do sistema de extração de *features* incorporado, *VGG-16* ou *ZF*). O modelo *Fast YOLO*, embora mais eficiente, atingindo 155 *FPS*, apresenta lacunas ao nível do parâmetro *mAP*, demonstrando um desvio de cerca de 22% face ao modelo *SSD* com maior número de *FPS.*

Tabela 9 - Resultados referentes ao teste PASCAL VOC2007. Comparação de detetores Faster R-CNN, YOLO e SSD ao nível de mAP e FPS (Liu et al., 2016)

Method	*mAP*	FPS	Test batch size	# Boxes
Faster R-CNN [2] (VGG16)	73.2	7	1	300
Faster R-CNN [2] (ZF)	62.1	17	1	300
YOLO [5]	63.4	45	1	98
Fast YOLO [5]	52.7	155	1	98
SSD300	74.3	46	1	8732
SSD512	76.8	19	1	24564
SSD300	74.3	59	8	8732
SSD512	76.8	22	8	24564

De acordo com (Liu et al., 2016), o método *SSD300* é o primeiro detetor em tempo real a atingir valores de precisão média acima de 70%, podendo ainda ser salientado que cerca de 80% do tempo utilizado em tarefas de classificação é gasto pelo sistema de

extração de *features* (neste caso, *VGG-16*), o que indica que a utilização de um sistema mais eficiente e mais rápido poderá permitir ao modelo *SSD512* realizar previsões e classificações em tempo real.

2.4 Visão por Computador e Estimativa de Peso

Embora sejam escassas as informações relativas ao estado da arte da aplicação de técnicas de visão por computador à tarefa de estimativa do peso de peixes, em particular no que diz respeito a peixes como a corvina, existem documentos relevantes que sintetizam a aplicação deste tipo de técnica ao problema descrito, particularmente no que diz respeito à estimativa de peso de suínos ou de gado (Dohmen et al., 2022).

De acordo com as características dos trabalhos realizados nesta área, a implementação deste tipo de solução tem como objetivo colmatar as lacunas dos processos de medições e controlo de peso realizados frequentemente nos locais que albergam grandes quantidades de animais. Além de permitir reduzir a quantidade de tempo utilizado na realização deste tipo de processos, a automatização do cálculo do peso dos seres vivos permite também o controlo dos seus fatores vitais, tais como o bom funcionamento do sistema digestivo (Dohmen et al., 2022).

Um dos principais procedimentos aplicados na realização deste tipo de estimativa reside no estudo de uma relação empírica entre características morfológicas dos animais analisados (por exemplo, o seu comprimento) e o seu peso corporal (Heinrichs and Losinger, 1998). No entanto, a medição dos animais, de forma manual, para obtenção dos valores associados a este tipo de características pode ser uma atividade perigosa que poderá colocar o ser vivo sob uma grande quantidade de *stress* e prejudicar o seu desenvolvimento (Heinrichs et al., 1992).

O estudo da aplicação de modelos de visão por computador a este tipo de processo apresenta, por isso, uma relevância extrema no contexto do problema estudado. Além de garantirem maior eficiência a nível temporal, a sua utilização garante que todos os processos de controlo decorrem de forma mais segura. Adicionalmente, a aplicação de técnicas de *machine learning* pode ser igualmente vantajosa durante este tipo de procedimento (Mollah et al., 2010).

De acordo com a revisão de literatura para estimativa de peso de suínos e gado, realizada por (Dohmen et al., 2022) e na qual foram analisados 26 trabalhos de investigação, a característica física mais utilizada, a partir da qual é estabelecida uma relação empírica com o peso de um determinado animal, é o seu comprimento corporal. Esta característica foi utilizada em quase 50% dos estudos analisados no

âmbito daquele trabalho. A segunda característica com mais tendência para ser utilizada foi a altura do animal.

No seguimento da análise dessa mesma revisão de literatura é também possível verificar uma tendência na utilização de modelos de regressão linear para obtenção de valores relativos à estimativa do peso dos animais (Figura 28). A métrica de avaliação relativa ao coeficiente de determinação, r^2, que indica a proporção de variação da variável dependente que pode ser explicada pela variável independente de um modelo de regressão, apresenta também uma elevada utilização no decorrer dos *papers* estudados (Figura 29).

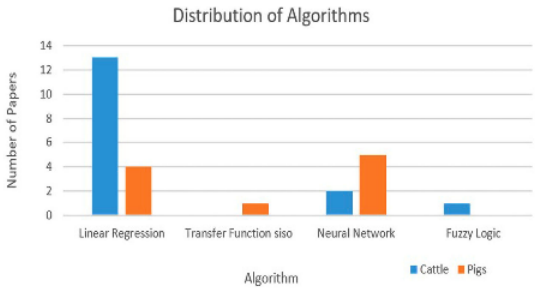

Figura 28 - Distribuição dos algoritmos de *machine learning* utilizados para estimativas de peso nos 26 trabalhos analisados (Dohmen et al., 2022)

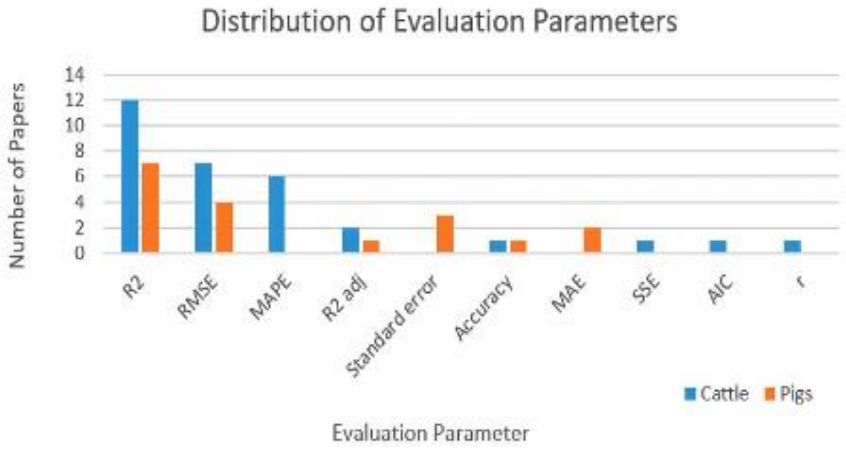

Figura 29 - Distribuição das métricas de avaliação utilizadas para estimativas de peso nos 26 trabalhos analisados (Dohmen et al., 2022)

Ao nível das limitações encontradas durante a realização dos trabalhos analisados, grande parte deles apontam para dificuldades na obtenção de dados de qualidade e na sua preparação para utilização durante os processos de treino e teste. São igualmente referidos desafios no que diz respeito à qualidade e resolução das imagens utilizadas, às condições de iluminação dos ambientes onde as imagens foram capturadas e à postura dos animais no momento da captação dos registos.

As informações fornecidas pela revisão de literatura realizada indicam também a existência de condicionantes em praticamente todos os trabalhos analisados, principalmente no que diz respeito a utilização de máquinas fotográficas configuradas *à priori*, que impactaram a exatidão dos modelos utilizados, especialmente após sofrerem alterações posicionais durante o processo de treino. De acordo com os autores, (Dohmen et al., 2022), seria ideal que pudessem vir a ser utilizadas câmaras dinâmicas ou drones para realização de estimativas de peso, de modo a mitigar potenciais falhas causadas pela má configuração dos dispositivos ou pela movimentação espacial dos animais.

Através dessas medições mais frequentes e menos propicias a erros, seria possível ter acesso a uma maior quantidade de dados estimados e, dessa forma, controlar mais eficazmente as variações do peso de cada animal ou de cada espécie, o que facilitaria, por exemplo, a deteção de algumas doenças ou condições vitais adversas (Segerkvist et al., 2020). A aplicação de modelos de *deep learning* a este tipo de problemas apresenta também grande potencial, segundo os autores, uma vez que estes modelos são capazes de realizar tarefas de segmentação de instâncias e identificação de objetos e podem ser combinados com modelos de *machine learning*, de modo a formarem processos automatizados de estimativa de peso.

A revisão de literatura realizada por (Dohmen et al., 2022) salienta, não só, a escassez de abordagens baseadas em modelos de visão por computador para realização de tarefas como a estimativa do peso de seres vivos, como também, a importância que este tipo de algoritmos assentes em redes neuronais convolucionais poderão vir a ter nessa mesma área de estudo, devido, principalmente, às suas competências e grande aplicabilidade.

2.5 Sinopse do Capítulo

O presente capítulo tem início com a descrição e distinção dos conceitos de inteligência artificial, *machine learning* e *deep learning.* É conferido especial destaque, ao nível de design e funcionamento, ao conceito de redes neuronais convolucionais, utilizadas frequentemente em problemas de identificação e classificação de imagens.

Segue-se um subcapítulo introdutório à história e evolução da área de estudo associada à visão por computador. Durante esta subsecção são descritos e situados temporalmente múltiplos avanços que permitiram o desenvolvimento funcional e arquitetural de vários modelos, capazes de atingir resultados impressionantes ao nível da deteção e classificação de imagens e objetos.

De entre esses modelos, são destacados três representantes do estado da arte de sistemas de deteção de objetos: modelos da família YOLO, modelos da família CNN e modelos da família SSD. Cada um destes métodos é descrito durante capítulo, onde também são apresentados dados comparativos, obtidos através da aplicação de diferentes *datasets*, que permitem extrair conclusões relativamente ao modo de funcionamento, vantagens e desvantagens de cada algoritmo.

O capítulo dois é finalizado com uma descrição do estado da arte da aplicação de modelos de *deep* e *machine learning* à tarefa de estimativa de peso de diferentes seres vivos.

3 Análise de Valor

Um pré-requisito fundamental para a realização de uma análise do valor associada a um determinado projeto, ideia ou solução é a compreensão do conceito de valor (Kenton, 2021).

O conceito de valor descreve a importância material ou imaterial conferida a um determinado objeto, serviço ou bem (Kenton, 2021).

Independentemente da indústria ou do setor em que se insere, a criação de valor é, indubitavelmente, um fator determinante para o crescimento e sucesso de uma empresa. Este processo decorre muitas vezes da exploração de novos projetos e ideias. No entanto, não é, de todo, incomum, que os procedimentos de implementação de promissores novos projetos sejam interrompidos precocemente devido a planeamento ineficaz e falta de recursos adequados ao desenvolvimento de uma ideia.

É também possível que um determinado conceito que, numa primeira instância, aparentasse ser adequado ao mercado onde se insere acabe por não acrescentar qualquer tipo de valor à empresa que o desenvolveu.

Nas secções que se seguem serão descritas e aplicadas técnicas relevantes que permitam extrair conclusões sobre o real valor da solução proposta neste documento.

3.1 Identificação de Oportunidades

O desenvolvimento e estudo de uma solução para identificação automatizada de zonas de vegetação através de imagens de satélite é facultado por múltiplas oportunidades existentes na área de *Deep Learning*.

A primeira oportunidade está diretamente relacionada com a importância que tem sido conferida, ano após ano, a áreas relacionadas com inteligência artificial. Desde 2012 que a ramificação de IA conhecida por *Deep Learning* tem crescido de forma exponencial, particularmente em métodos relevantes para o tema proposto, com sucessivas adaptações e melhorias implementadas em algoritmos construídos com recurso a uma arquitetura neuronal convolucional (Shetty, 2016). Exemplos destes mesmos avanços são os métodos da família CNN e da família YOLO, métodos estes que representam o estado da arte no âmbito de deteção de objetos e classificação de imagens e que, apesar do seu aparecimento relativamente recente e das suas

diferenças substanciais ao nível funcional e arquitetural, apresentam resultados extremamente promissores (Du, 2018).

Tendo em conta estes desenvolvimentos, surge a chance de comparar métodos de visão por computador, no contexto da identificação de zonas de vegetação através de imagens de satélite, e, de concluir, com base nos resultados obtidos, que algoritmo representante do estado da arte desta área mais se adequa ao problema em questão.

Em segundo lugar, uma oportunidade que deve ser aproveitada e que decorre dos avanços do poder computacional e da facilidade com que esse mesmo poder computacional é distribuído através de serviços existentes na *cloud*.

A execução de tarefas de processamento de imagens é bastante exigente ao nível gráfico e computacional, este tipo de processamento, aliado ao processo de treino de algoritmos de *Deep Learning,* não é, muitas vezes, suportado pelo *hardware* existentes dentro de máquinas de uso quotidiano. Com esta limitação em mente, foram criadas ferramentas, acessíveis através da *internet*, que disponibilizam recursos presentes no servidor onde estão hospedadas e que, por norma, são muito mais capazes do que os recursos locais de um investigador comum. Produtos como o *Google Colaboratory*, que não passam de *Jupyter notebooks* hospedados na *web*, oferecem aos seus utilizadores a possibilidade de escrever e executar código (em *Python*) tendo sido especialmente criados para o desenvolvimento de projetos na área de *Machine Learning* e IA ("Google Colab," 2017);

O crescimento da indústria da inteligência artificial e, em particular, da indústria *visão por computador* é um fator que também deve ser tido em conta como uma oportunidade da qual se deve tirar partido. O mercado para desenvolvimentos na área de *visão por computador* foi avaliado em cerca de 14.82 biliões de dólares americanos no ano de 2020 (aproximadamente 13 mil milhões de euros) e é esperado que apresente uma percentagem de crescimento anual de cerca de 7.3% até ao ano de 2028.

Os progressos recentes nesta área têm alargado os horizontes relativamente às aplicações deste tipo de modelos. Atualmente, sistemas utilizados em indústrias como a educação, a saúde, a robótica, o retalho e a segurança usufruem, não só dos progressos acima mencionados, como também de desenvolvimentos associados a melhorias de sensores e de tecnologias de captação de imagem.

Um dos setores que mais tira partido das melhorias nas técnicas de visão por computador é o setor automóvel, que tem vindo a ser alvo de uma mudança significativa no seu paradigma, relacionada com a crescente implementação de

automóveis munidos com a capacidade de condução autónoma. Tendo em conta que automóveis com esta capacidade necessitam de possuir algum tipo de visão por computador que lhes permita reproduzir (e até superar) a condução humana, é esperado que exista um investimento significativo nesta área de estudo, que se deverá traduzir numa melhoria, não só dos métodos de *Deep Learning* que lhe estão associados, como também de todas as componentes de *hardware* que a complementam ("Visão por computador Market Size & Share Report, 2021-2028," 2021).

Por fim, é importante considerar a oportunidade de automatizar e, consequentemente, melhorar, processos que já existem. Com a avaliação do desempenho mais do que um método de reconhecimento de imagem e deteção de zonas de vegetação através de imagens de alta resolução será possível disponibilizar *feedback* relevante que poderá ser usado na implementação de sistemas autónomos que tenham a capacidade de substituir seres humanos na realização deste tipo de tarefas (que, por norma, são maçadoras e requerem bastante tempo para serem completadas).

No seguimento das oportunidades acima identificadas, considerou-se que a comparação e análise de resultados obtidos através da aplicação de métodos de nova geração de visão por computador seria um processo que acrescentaria valor à solução desenvolvida.

Assim, serão treinados, testados e comparados métodos da família *YOLO*, métodos *Single Shot Multi-box Detector (SSD)* e métodos da família *CNN* de modo que seja possível concluir qual o mais adequado à tarefa subjacente. Os métodos pertencentes a estes grupos apresentam diferenças ao nível funcional e arquitetural e, de modo a contextualizar o leitor, são descritos de forma breve nas secções que se seguem.

3.1.1 YOLO

Algoritmos YOLO são executados em tempo real e têm como objetivo prever e as suas regiões delimitadoras (também denominadas *bounding boxes*) e a categoria à qual um ou vários objetos presentes numa imagem estão associados. O seu nome – "You Only Look Once" – faz referência ao facto de o detetor necessitar apenas de uma propagação pela sua rede neuronal para realizar uma previsão.

Desta forma, este tipo de método divide a imagem que recebe como *input* em várias células, cada célula é responsável por prever cinco *bounding boxes* (caso exista mais do que um objeto numa determinada célula). Uma vez que, por norma, existem várias células que não irão conter qualquer tipo de objeto que seja útil ao nível de

classificação, é computado um valor probabilístico que, quando não é atingido, serve como critério de exclusão para células que apresentem baixa probabilidade de conterem um objeto de interesse. Este processo de filtragem de células e de obtenção da *bounding box* com maior área útil é denominado *non-max supression* (Figura 30) (Świeżewski, 2020).

Figura 30 - Aplicação de non-max supression (Świeżewski, 2020)

Usufruindo de uma única rede neuronal convolucional, estes detetores podem ser treinados recorrendo a imagens em tamanho original (que são ajustadas em tempo de execução), o que lhes permite maior facilidade em reconhecer e detetar determinados *features*. Possuem também benefícios ao nível da sua rapidez de execução, uma vez que podem ser utilizados em *real-time*, e ao nível da sua capacidade de aprendizagem associada à generalização de determinadas representações de objetos (Redmon and Farhadi, 2016).

3.1.2 Single Shot Multi-Box Detector (SSD)

Detetores do tipo *SSD* realizam previsões em tempo real. Ao nível arquitetural, estes modelos apresentam uma estrutura em pirâmide composta por um sistema *VGG-16* de extração de *features* (sem *fully connected layers*) e por múltiplas camadas convolucionais capazes de classificar *features* de diversas dimensões.

A utilização de *anchor boxes* pré-definidas manualmente (perímetros que se aproximam da localização real de objetos presentes na imagem de *input*), permite que o modelo aplique operações de regressão até identificar a localização exata de cada objeto. As *bounding boxes* resultantes da regressão anteriormente mencionada apresentam várias dimensões e resoluções, o que aumenta a probabilidade da produção de previsões corretas, independentemente do tamanho e *aspect ratio* dos objetos (Liu et al., 2016).

Durante o período de treino deste tipo de detetores, várias *bounding boxes* serão geradas para cada *anchor box*, muitas delas englobando regiões de pouca utilidade para a tarefa de classificação (baixo índice de *IoU* face à localização real de um determinado objeto). Embora aparentemente irrelevantes, estas seleções não devem ser

descartadas, uma vez que, a sua utilização durante o processo de treino do modelo permite-lhe acumular conhecimento relativamente a como evitar decisões erradas ou inconsistentes. Com o objetivo de tornar o algoritmo mais robusto, são também aplicados processos de *data augmentation* que fazem variar a resolução e orientação de cada imagem de *input* (Forson, 2017).

A seleção final de uma *bounding box* para cada objeto é realizada através de um processo de *non-max supression*, de onde resultam *N* perímetros associados a valores de confiança e índices de *IoU* superiores a determinados *thresholds* pré-estabelecidos.

3.1.3 CNN

A família CNN incluí métodos como *R-CNN, Fast R-CNN, Faster R-CNN* e *Mask R-CNN*. As diferentes nomenclaturas utilizadas para descrever cada detetor deste grupo fazem referência às graduais evoluções e modificações que foram sendo implementadas tanto ao nível arquitetural como funcional, desde o começo do estudo de *Region-Based Convolutional Neuronal Networks*.

R-CNN – Representa das primeiras aplicações de uma rede neuronal convolucional ao problema de deteção de objetos e classificação de imagens. É composto por uma arquitetura dividida em três módulos:

- Módulo de proposta de regiões – responsável por gerar e extrair regiões candidatas a serem classificadas/categorizadas;
- Módulo de extração de *features* – usufrui de uma rede neuronal convolucional para extrair *features* de uma região candidata;
- Módulo de classificação – responsável por classificar *features* de acordo com as categorias previamente estabelecidas;

Embora competente e relativamente simples, este detetor apresenta uma lenta velocidade de treino e execução, uma vez que cada região candidata deve ser analisada pelo módulo de extração de *features* de forma sequencial (Brownlee, 2019).

Faster R-CNN – Modelo proposto com o objetivo de atenuar as dificuldades apresentadas pelo método *R-CNN*. Ao nível arquitetural é composto por uma única *pipeline* (contrastando com a *pipeline* complexa e modular do seu predecessor).

Este detetor recebe como *input* uma imagem e uma série de regiões candidatas. O modulo de extração de features é composto por uma rede neuronal convolucional pré-

treinada e a extração ocorre, mais especificamente, numa nova camada dessa mesma rede, a camada de *Region of Interest Pooling*.

Os *features* extraídos são, posteriormente, analisados por uma *fully connected layer* e, por fim, o método resulta em dois *outputs*: a previsão da categoria da imagem e a previsão da sua região delimitadora.

Devido à sua única *pipeline*, este modelo apresenta melhorias significativas ao nível da velocidade de treino e execução, no entanto, requer que lhe sejam fornecidas como *input* as regiões de interesse candidatas (Brownlee, 2019).

Faster R-CNN & Mask R-CNN – O método *Faster R-CNN*, também alicerçado numa única *pipeline*, é composto por dois módulos:

- *Region Proposal Network* – Rede neuronal convolucional com o objetivo de propor regiões de interesse e sugestões relativas ao tipo de objetos presentes nessas regiões;

- *Fast R-CNN* – Rede neuronal convolucional com a função de extrair *features* das regiões propostas. Módulo responsável pelo *output* de regiões delimitadoras e de classificações de objetos;

O primeiro módulo age como um "mecanismo de atenção", fornecendo informação sobre a localização de determinados *features*, à rede neuronal convolucional que o sucede (Brownlee, 2019).

O modelo *Mask R-CNN* é construído na base do modelo *Faster R-CNN*, apresentando uma modificação arquitetural que lhe confere a capacidade de produzir três *outputs* distintos: regiões delimitadoras, classificações de objetos e à rede neuronal convolucional que o sucede associadas aos mesmos.

A produção das máscaras associadas aos objetos é garantida por uma nova camada, composta por uma *fully connected layer* e responsável pela segmentação de imagens (Odemakinde, 2021).

3.2 AHP & TOPSIS

Métodos multicritério de apoio à decisão podem ser aplicados em várias áreas, desde a área da saúde à área da engenharia. Este tipo de métodos analisam um determinado número de critérios, um grau de importância, ou peso, a cada um deles associado e um

conjunto de alternativas propostas que servem como solução para um determinado problema.

O seu principal objetivo é categorizar este conjunto de alternativas de acordo com o quão em linha com os critérios previamente estabelecidos estas se encontram, procurando dar a conhecer a solução que apresente o menor grau de compromisso face aos critérios selecionados (Jahan et al., 2016).

Dois dos métodos multicritério de apoio à decisão mais amplamente utilizados são conhecidos por *Analytic Hierarchy Process (AHP)* e *Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)*. Ambos serão descritos e aplicados nas secções que se seguem (Widianta et al., 2018).

3.2.1 Analytic Hierarchy Process (AHP)

Esta metodologia, desenvolvida por Tomas L. Saaty em 1980, encontra-se alicerçada no método de pensamento newtoniano e cartesiano, cujo objetivo passa por decompor um problema em fatores e subfactores sucessivamente mais simples de modo a facilitar a compreensão do domínio em que o problema se insere e a, posterior, construção de uma solução (Souza Marins et al., 2009). Apresentando a possibilidade de ser decomposto em sete fases, o AHP encontra-se descrito e aplicado ao problema tratado nas secções que seguem.

Fase 1 – Divisão Hierárquica - Uma divisão hierárquica do problema, de onde resultam, por norma, um objetivo e um determinado número de critérios e alternativas.

Da aplicação desta divisão ao tema descrito neste documento resulta um diagrama como o apresentado na

Figura 31. Traduzindo a informação contida no diagrama, é possível afirmar que o principal objetivo do trabalho descrito é encontrar uma ferramenta de visão por computador capaz de detetar zonas de vegetação através de imagens de satélite, tendo em conta a sua rapidez de execução (a), a sua facilidade em encontrar e corretamente identificar objetos de diferentes dimensões (b), a sua facilidade em interpretar imagens de diferentes dimensões (c), o seu grau de dificuldade ao nível da implementação (d) e a sua intensidade ao nível do consumo de recursos computacionais (e). As principais alternativas a serem estudadas são os detetores da família YOLO, os detetores da família CNN e detetores da família SSD.

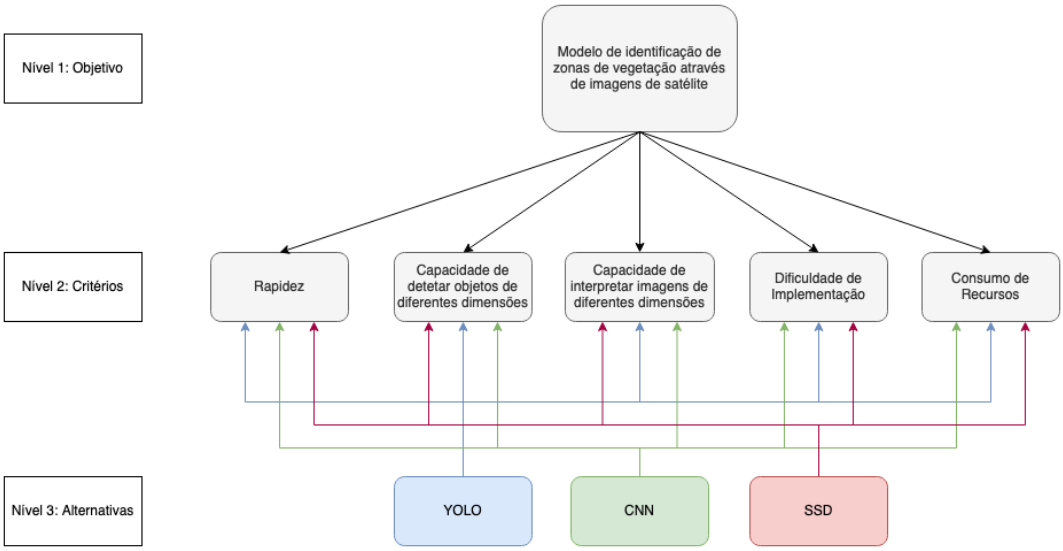

Figura 31 - Divisão Hierárquica do problema em objetivo, critérios e alternativas

Fase 2 – Comparação de critérios – Para a comparação de alternativas e critérios, Saaty definiu uma escala fundamental, com valores de um a nove. Esta escala encontra-se representada na Tabela 10.

Tabela 10 - Escala fundamental de Saaty

Nivel de importância	Definição	Explicação
1	Igual importânica	As duas atividades contribuem igualmente para o objetivo
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2,4,6,8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

Aplicando a escala de Saaty a cada par de critérios previamente estabelecidos é possível obter uma tabela de comparação, que pode, posteriormente, ser convertida numa matriz de comparação (Tabela 11). Nesta matriz encontra-se representado o grau de importância relativo de cada critério face aos restantes. Cada um destes valores foi definido com base no estudo dos vários modelos.

Tabela 11 - Matriz de comparação

	(a)	(b)	(c)	(d)	(e)
(a)	1	2	2	5	5
(b)	1/2	1	2	4	5
(c)	1/2	1/2	1	1	1
(d)	1/5	1/4	1	1	2
(e)	1/5	1/5	1	1/2	1

Fase 3 – Obtenção de prioridades relativas – O passo seguinte consiste na utilização da matriz de comparação, obtida na fase anterior, para a obtenção do vetor de prioridades relativas de cada critério. Para isso, é primeiramente necessário normalizar a suprarreferida matriz, dividindo cada valor pelo total da soma da sua respetiva coluna. O processo da normalização encontra-se demonstrado em seguida (Tabela 12, Tabela 13).

Tabela 12 - Matriz intermédia para obtenção do vetor de prioridades relativas

	(a)	(b)	(c)	(d)	(e)
(a)	1	2	2	5	5
(b)	1/2	1	2	4	5
(c)	1/2	1/2	1	1	1
(d)	1/5	1/4	1	1	2
(e)	1/5	1/5	1	1/2	1
Soma	2,4	6,35	7	11,5	14

Procedendo à divisão de cada valor de cada coluna pela respetiva soma de todos os valores dessa mesma coluna obtemos:

Tabela 13 - Matriz intermédia para a obtenção do vetor de prioridades (após divisão)

	(a)	(b)	(c)	(d)	(e)
(a)	0,42	0,31	0,29	0,43	0,36

(b)	0,21	0,16	0,29	0,35	0,36
(c)	0,21	0,08	0,14	0,09	0,07
(d)	0,08	0,04	0,14	0,09	0,14
(e)	0,08	0,03	0,14	0,04	0,07

O vetor de prioridades relativas pode ser obtido através da média aritmética de cada linha da matriz normalizada.

Tabela 14 - Vetor de prioridades

	Vetor de Prioridades
(a)	0,362
(b)	0,274
(c)	0,118
(d)	0,098
(e)	0,072

Através do vetor calculado, é possível concluir que o critério **(a)**, que representa a rapidez de execução de cada modelo, apresenta uma importância superior face aos restantes critérios.

Fase 4 – Avaliação da consistência – A presente fase engloba uma verificação da consistência dos julgamentos anteriores, através do cálculo da razão de consistência (RC). Valores de razão de consistência superiores a 0.1 não são, por norma, aceitáveis, uma vez que indicam que os julgamentos realizados na comparação de critérios se encontram demasiado próximos da aleatoriedade. A razão de consistência pode ser calculada através da aplicação da Equação 2.

Equação 2 - Fórmula de cálculo da razão de consistência

$$RC = \frac{IC}{IA}$$

Uma vez que o valor do índice de consistência (IC) não é, ainda, conhecido, o seu cálculo é o primeiro passo para a obtenção da razão de consistência. No entanto, este cálculo encontra-se dependente de uma outra variável, λ_{max}, que pode ser obtida através da

expressão $Ax = \lambda_{max}x$, onde *A* representa a matriz de comparação e *x* representa o vetor próprio obtido na fase 3 (Figura 32 e Figura 33). O cálculo deste valor encontra-se demonstrado em seguida:

	(a)	(b)	(c)	(d)	(e)
(a)	1	2	2	5	5
(b)	1/2	1	2	4	5
(c)	1/2	1/2	1	1	1
(d)	1/5	1/4	1	1	2
(e)	1/5	1/5	1	1/2	1
Soma	2,4	6,35	7	11,5	14

Vetor de Prioridades
0,362
0,274
0,118
0,098
0,072

Figura 32 - Processo de multiplicação da matriz de comaparação pelo vetor próprio

	1,996	Vetor de Prioridades		5,51
=	1,443	0,362	=	5,27
	0,606	0,274		5,14
	0,501	0,118		5,11
	0,366	0,098		5,08
		0,072		

Figura 33 - Processo de obtenção de valores para cálculo do $\lambda\max$

Equação 3 - Fórmula para cálculo do λ_{max}

$$\lambda \max = \frac{5{,}51 + 5{,}27 + 5{,}14 + 5{,}11 + 5{,}08}{5} = 5{,}22$$

Sabendo o valor de λ_{max}, torna-se possível calcular o índice de consistência. A fórmula para obtenção do IC, assim como a sua aplicação, encontra-se representada naEquação 4.

Equação 4 - Fórmula e cálculo do índice de consistência

$$IC = \frac{\lambda \max - n}{n - 1} = \frac{5{,}22 - 5}{5 - 1} = 0{,}055$$

O valor do índice aleatório (IA), referente a um grande número de comparações de pares de critérios, pré-calculado pelo Laboratório Nacional de Oak Ridge (Ribeiro and

Alves, 2016), encontra-se tabelado (Tabela 15), de acordo com o número de critérios definidos durante a aplicação do método.

Tabela 15 - Tabela de valores referentes ao índice aleatório

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

De acordo com a tabela acima apresentada e tendo em conta a definição de cinco critérios, é possível concluir que o valor do IA a ser usado é 1,12. Assim, a razão de consistência dos julgamentos realizados pode ser obtida através da aplicação da Equação 5.

Equação 5 - Fórmula para o cálculo da razão de consistência

$$RC = \frac{IC}{IA} = \frac{0{,}055}{1{,}12} = 0{,}049$$

A obtenção de um valor inferior a 0.1 indica que os valores das prioridades relativas atribuídas a cada par de critérios são consistentes.

Fase 5 – Comparação de critérios por alternativa – Na presente etapa do método AHP, os procedimentos realizados para a elaboração da matriz de comparação de critérios devem ser repetidos, com as convenientes alterações, para a elaboração de matrizes de comparação de alternativas por critério. Deve ainda ser calculado o vetor de prioridades de cada uma das matrizes elaboradas.

	YOLO	CNN	SSD
YOLO	1	5	1
CNN	1/5	1	1/5
SSD	1	5	1
Soma	2,2	11	2,2

	YOLO	CNN	SSD
YOLO	0,45	0,45	0,45
CNN	0,09	0,09	0,09
SSD	0,45	0,45	0,45

Vetor de Prioridades
0,45
0,09
0,45

Figura 34 - Matriz de comparação e vetor de prioridades para o critério "Capacidade de interpretar imagens de diferentes dimensões"

	YOLO	CNN	SSD
YOLO	1	5	1/3
CNN	1/5	1	5
SSD	3	1/5	1
Soma	4,2	6,2	6,3

	YOLO	CNN	SSD
YOLO	0,24	0,81	0,053
CNN	0,048	0,16	0,79
SSD	0,71	0,03	0,16

Vetor de Prioridades
0,38
0,33
0,3

Figura 35 - Matriz de comparação e vetor de prioridades para o critério "Rapidez"

	YOLO	CNN	SSD
YOLO	1	1/3	1/2
CNN	3	1	2
SSD	2	1/2	1
Soma	6	1,83	3,5

	YOLO	CNN	SSD
YOLO	0,17	0,18	0,14
CNN	0,5	0,55	0,57
SSD	0,33	0,27	0,29

Vetor de Prioridades
0,16
0,54
0,3

Figura 36 - Matriz de comparação e vetor de prioridades para o critério "Dificuldade de implementação"

	YOLO	CNN	SSD
YOLO	1	1/3	2
CNN	3	1	4
SSD	1/4	1/2	1
Soma	4,25	1,83	7

	YOLO	CNN	SSD
YOLO	0,24	0,18	0,29
CNN	0,71	0,55	0,57
SSD	0,06	0,27	0,14

Vetor de Prioridades
0,24
0,61
0,16

Figura 37 - Matriz de comparação e vetor de prioridades para o critério "Utilização de recursos"

	YOLO	CNN	SSD
YOLO	1	1/2	1/3
CNN	2	1	1/2
SSD	3	2	1
Soma	6	3,5	1,83

	YOLO	CNN	SSD
YOLO	0,17	0,14	0,18
CNN	0,33	0,29	0,27
SSD	0,5	0,57	0,55

Vetor de Prioridades
0,16
0,30
0,54

Figura 38 - Matriz de comparação e vetor de prioridades para o critério "Capacidade de detetar objetos de diferentes dimensões"

Após a obtenção de todos os vetores de prioridades, estes devem ser unidos numa única matriz que agrega os resultados das comparações paritárias entre cada critério e cada uma das alternativas (Tabela 16).

Tabela 16 - Matriz de agregação dos resultados das comparações paritárias

0,38	0,16	0,45	0,16	0,24
0,33	0,30	0,09	0,54	0,61
0,30	0,54	0,45	0,30	0,16

Fase 6 – Obtenção da prioridade composta para as alternativas – Após o registo das matrizes de comparação elaboradas na fase anterior, o passo seguinte reside no cálculo do vetor de prioridades compostas das alternativas (Figura 39). Este vetor pode ser obtido através da multiplicação da matriz composta pela agregação dos vetores prioridade obtidos na Fase 5 pelo vetor prioridade dos critérios, obtido na Fase 3.

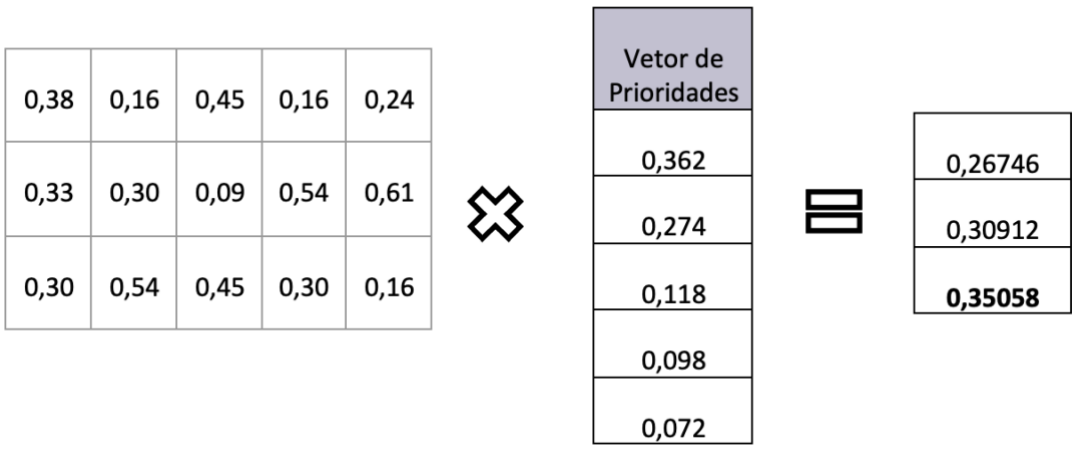

Figura 39 - Processo de obtenção do vetor de prioridade composta para as alternativas

Fase 7 – Escolha da alternativa – A última fase do método AHP é composta pela escolha da alternativa mais adequada, através da análise dos resultados obtidos na fase anterior. Assim, em função dos critérios definidos e das suas prioridades, é possível concluir que a alternativa composta pela aplicação de métodos da família SSD, com um grau de prioridade 0.35058, é a mais adequada de entre todas as apresentadas.

3.2.2 Technique of Order Preference by Similarity to Ideal Solution (TOPSIS)

O método TOPSIS é uma ferramenta de tomada de decisão multicritério que, quando aplicada a um determinado problema, apresenta como principal objetivo a seleção de uma alternativa, de entre um conjunto de alternativas, que simbolize o menor distanciamento possível da solução ideal e o maior distanciamento possível da pior solução.

Este método pressupõe a existência de *m* alternativas, *n* atributos/critérios e pressupõe ainda a existência de uma pontuação associada a cada alternativa para cada critério. As soluções são categorizadas de acordo com os valores dos seus atributos, assim, a alternativa ideal deverá apresentar o maior número de atributos benéficos e, logicamente, a pior alternativa deverá apresentar o maior número de atributos prejudiciais.

Desta forma, se x_{ij} representar a pontuação associada à alternativa i de acordo com o critério j, é possível construir uma matriz X de dimensão m por n. É também possível definir um conjunto J de critérios benéficos e um conjunto L de critérios prejudiciais.

O método TOPSIS é composto por cinco fases distintas e encontra-se descrito e aplicado ao problema subjacente a este documento nas secções que se seguem.

Fase 1 – Construção da matriz e obtenção do r_{ij} – Na primeira fase deste método é necessário construir a matriz de pontuações entre critérios e alternativas (Tabela 17) e, posteriormente, proceder à normalização dos seus valores, através da divisão de cada valor pelo resultado da expressão (Σx^2_{ij}) (Tabela 18). Cada valor resultante da divisão é representado pela variável r_{ij}. Este processo, assim como a especificação das variáveis *m, n, J, L* e do respetivo peso de cada critério, encontram-se demonstrados em seguida.

m = 3 (YOLOv3, CNN, SSD);

n = 5;

J = **(a)** rapidez, **(b)** capacidade de interpretar objetos de diferentes dimensões, **(c)** capacidade de interpretar imagens de grandes dimensões;

L = **(d)** dificuldade de implementação, **(e)** utilização de recursos;

Pesos = **(a)** – 0,2; **(b)** – 0,3; **(c)** – 0,2; **(d)** – 0,2; **(e)** – 0,1;

Tabela 17 - Matriz de pontuações entre critérios e alternativas

	(a)	(b)	(c)	(d)	(e)
YOLO	7	4	8	5	2
CNN	3	5	4	6	3
SSD	8	8	8	6	2

Tabela 18 - Matriz para cálculo de Σx^2_{ij}

	(a)	(b)	(c)	(d)	(e)
YOLO	70	45	80	50	20
CNN	30	55	40	60	30
SSD	80	80	80	60	20
$\sum(x_{ij})^2$	180	180	200	170	70
$\sum(x_{ij})^{1/2}$	13,42	13,42	14,14	13,04	8,37

Após o cálculo de (Σx^2_{ij}) para cada coluna, é possível utilizar esses valores para obter a matriz normalizada, utilizando os mesmos como denominadores respetivos na operação da divisão de cada valor de cada coluna. A matriz normalizada encontra-se apresentada na Tabela 19.

Tabela 19 - Matriz normalizada

	(a)	(b)	(c)	(d)	(e)
YOLO	0,52	0,30	0,57	0,38	0,24
CNN	0,22	0,37	0,28	0,46	0,36
SSD	0,60	0,60	0,57	0,46	0,24

Fase 2 – Construção da matriz de decisão – A segunda fase consiste na construção da matriz de decisão (Tabela 20). Partindo da matriz obtida na fase anterior, devem ser utilizados os pesos para cada critério, w_j. Cada valor de cada coluna da matriz deve ser

multiplicado pelo peso associado ao seu critério correspondente, obtendo v_{ij} ($v_{ij} = w_j r_{ij}$).

Tabela 20 - Matriz de decisão

	(a)	(b)	(c)	(d)	(e)
YOLO	0,104	0,09	0,114	0,076	0,024
CNN	0,044	0,111	0,056	0,092	0,036
SSD	0,12	0,18	0,114	0,092	0,024

Fase 3 – Determinação da solução ideal por critério – A determinação da solução ideal, por critério, é realizada utilizando a matriz obtida na fase 2 (Tabela 21). Devem ser selecionados os maiores valores de cada coluna associada a um critério benéfico e os menores valores pertencentes a colunas associadas a critérios prejudiciais. A seleção destes valores resultará na criação de um vetor *A*,* tal como demonstrado abaixo.

Tabela 21 - Matriz para obtenção da solução ideal por critério

	(a)	(b)	(c)	(d)	(e)
YOLO	0,104	0,09	0,114	0,076	0,024
CNN	0,044	0,111	0,056	0,092	0,036
SSD	0,12	0,118	0,114	0,092	0,024

*A** = {0,12; 0,118; 0,114; 0,076; 0,024}

De forma semelhante, é necessária a obtenção do vetor *A'*, que deve ser formado com recurso ao inverso do processo descrito acima. Devem ser selecionados os menores valores de cada coluna associada a um critério benéfico e os maiores valores pertencentes a colunas associadas a critérios prejudiciais (Tabela 22).

Tabela 22 - Matriz para obtenção da solução menos ideal por critério

	(a)	(b)	(c)	(d)	(e)
YOLO	0,104	0,09	0,114	0,076	0,024
CNN	0,044	0,111	0,056	0,092	0,036
SSD	0,12	0,18	0,114	0,092	0,024

A' = {0,044; 0,09; 0,056; 0,092; 0,036}

Fase 4 – Determinar a separação da solução ideal – Na quarta fase do método TOPSIS deve ser determinada a separação da solução ideal, S_i^*, para cada coluna da matriz obtida na segunda fase. Este valor pode ser obtido através de uma expressão $S_i^* = [\Sigma (v_j^* - v_{ij})^2]^{½}$, este processo encontra-se descrito em seguida e é iniciado pela obtenção do somatório do quadrado da diferença entre cada valor da matriz e o valor do vetor A^* associado a cada critério (e, consequentemente, a cada coluna da matriz). Os valores resultantes deste passo intermédio são, posteriormente, elevados a um meio, resultando em S_I^* (Tabela 23).

Tabela 23 - Determinação da separação da solução ideal

	(a)	(b)	(c)	(d)	(e)	S_i^*
YOLO	$(0,104 - 0,12)^2 = 2,56 \times 10^{-4}$	$7,84 \times 10^{-4}$	0	0	0	0,032
CNN	$5,78 \times 10^{-3}$	$4,9 \times 10^{-5}$	$3,36 \times 10^{-3}$	$2,56 \times 10^{-4}$	$1,44 \times 10^{-4}$	0,098
SSD	0	0	0	$2,56 \times 10^{-4}$	0	0,016

Esta fase inclui ainda a determinação da solução menos ideal, com recurso a um processo similar ao descrito acima. Nesta caso, além dos valores da matriz, são utilizados os dados referentes ao vetor A', obtido na fase anterior, para cálculo de $S_i' = [\Sigma (v_j' - v_{ij})^2]^{½}$ (Tabela 24).

Tabela 24 - Determinação da separação da solução menos ideal

	(a)	(b)	(c)	(d)	(e)	S_i'
YOLO	$(0,104 - 0,044)^2 = 3,6 \times 10^{-3}$	0	$3,36 \times 10^{-3}$	$2,56 \times 10^{-4}$	$1,44 \times 10^{-4}$	0,087
CNN	0	$4,41 \times 10^{-4}$	0	0	0	0,021
SSD	$5,78 \times 10^{-3}$	$8,1 \times 10^{-3}$	$3,36 \times 10^{-3}$	0	$1,44 \times 10^{-4}$	0,132

Fase 5 – Cálculo da proximidade relativa à solução ideal – A última fase engloba o cálculo da proximidade relativa à solução ideal (Tabela 25). Este valor pode ser obtido através da aplicação da expressão $C_i^* = S'_i / (S_i^* + S'_i)$. A alternativa que estiver associada

ao maior valor de C_i^* deve ser considerada a melhor solução para o problema equacionado.

Tabela 25 - Vetor de proximidade à solução ideal

	C_i^*
YOLO	0,73
CNN	0,18
SSD	0,89

Assim, em função da aplicação do método TOPSIS demonstrada acima, é possível concluir que a alternativa que contempla a aplicação do modelo *Single Shot Multi-Box Detector (SSD)*, associada a um valor de C_i^* de 0,89, é a mais adequada de entre todas as apresentadas.

3.3 Sinopse do Capítulo

O presente capítulo foi introduzido através da definição o conceito de valor, quando associado a um determinado projeto ou negócio. Em seguida, procedeu-se à clarificação e enunciação das oportunidades identificadas, tendo em conta a sua relação com o tópico subjacente ao problema a resolver.

A identificação de oportunidades seguida por uma breve introdução teórica a cada um dos tipos de algoritmos de *Deep Learning* propostos para análise no âmbito da deteção de zonas de vegetação através de imagens de satélite.

Finalmente, foi realizada uma análise das alternativas e critérios definidos de acordo com o problema, com recurso a dois métodos multicritério de apoio à decisão: AHP e TOPSIS. Ambos estes métodos conduziram ao mesmo resultado, indicativo de que modelos *Single Shot Multi-Box Detector* parece, teoricamente, ser a solução mais adequada, de acordo com as características do problema proposto.

4 Experimentação e Avaliação de Resultados

A qualidade de um determinado sistema está diretamente relacionada com a exatidão e aplicabilidade das técnicas de verificação que compõem o processo de avaliação e testagem desse mesmo *software*. Programas cujo mau funcionamento possa originar consequências que, por exemplo, coloquem em risco uma vida humana, requerem um processo de experimentação e avaliação mais minucioso e exaustivo do que sistemas que acarretem consequências menos relevantes aquando de uma falha (Rathore and Kumar, 2019).

É essencial definir os requisitos de validação no início de cada projeto, de modo que as estratégias de validação, verificação e teste sejam escolhidas de forma coerente e de acordo com os requisitos iniciais (Rathore and Kumar, 2019).

No contexto do problema atual, a definição dos requisitos de validação e das rotinas de testagem são fulcrais para garantir a confiabilidade e exatidão dos processos desenvolvidos. Deste modo, nas secções que se seguem serão descritas as hipóteses a validar, as metodologias de avaliação e as métricas utilizadas para validação dos resultados obtidos.

4.1 Hipótese

Tendo em conta os objetivos propostos no âmbito desta dissertação de mestrado e reconhecendo a necessidade de implementação de processos iterativos que permitam atingir esses mesmos objetivos, é possível definir duas hipóteses que fazem referência a cada uma das metas a atingir.

Para o problema de cálculo de estimativas de peso de diferentes corvinas é possível estabelecer uma hipótese como a seguinte: "É viável a estimativa do peso de um animal como a corvina, com recurso a um modelo de regressão linear simples e utilizando dados relativos ao seu comprimento, obtidos através do *output* de um modelo de visão por computador *YOLOv4*?".

Para a identificação de períodos de alimentação, a hipótese estudada pode ser descrita da seguinte forma: "É possível identificar corretamente a existência de períodos de

alimentação através da deteção de grupos de peixes e da comparação da área dessa mesma deteção com a área da figura originalmente analisada?".

As hipóteses colocadas na presente subsecção indicam o estudo e análise de um modelo *YOLOv4* e de um modelo de regressão linear simples na tentativa de obtenção de informação relevante que permita a realização de estimativas de peso para sujeitos individuais e que permita a identificação de períodos de alimentação de grupos de sujeitos.

Deste modo, as soluções propostas apresentarão, como principal objetivo, a resposta aos diferentes critérios/questões de validação definidas no contexto do problema atual, nomeadamente:

1. O sistema de estimativa de pesos opera com um nível de velocidade de acordo com o esperado?

2. O desvio percentual entre as estimativas de peso realizadas e o peso real de cada corvina apresenta valores aceitáveis?

3. O sistema de estimativa de pesos é adaptável a diferentes condições de utilização e/ou à utilização de diferente *hardware*?

4. O sistema de deteção de períodos de alimentação opera com um nível de velocidade de acordo com o esperado?

5. O sistema de deteção de períodos de alimentação é capaz de produzir conclusões relevantes utilizando apenas informações relativas à área das deteções e da imagem original?

4.2 Metodologia de Avaliação

Uma metodologia de avaliação é uma ferramenta que permite uma melhor compreensão das etapas necessárias para a realização de uma avaliação qualitativa de um determinado produto. O seguimento de uma metodologia de avaliação sistemática permite ao leitor obter a perceção dos passos necessários para determinar a qualidade de um objeto de estudo (Rudd, 2019).

A metodologia de avaliação utilizada na presente tese consistiu em três etapas fundamentais, definidas com recurso à estratégia implementada por (Baehr, 2004):

- Definição dos parâmetros de avaliação;

- Definição de estratégias para obtenção de dados;
- Definição das métricas de avaliação.

Cada uma das etapas previamente listadas será descrita e enquadrada no contexto do projeto atual durante as subsecções seguintes.

4.2.1 Parâmetros de Avaliação

A necessidade de definição de parâmetros de avaliação está diretamente relacionada com a utilização dos resultados produzidos e com a forma como esses resultados impactarão a tomada de decisões por parte de quem os irá utilizar.

No contexto da presente tese, os parâmetros de avaliação podem ser traduzidos em questões ou frases que descrevam os objetivos a atingir ou as características que os resultados obtidos devem possuir. No seguimento desta ideia, encontram-se, em seguida, enumerados, os parâmetros de avaliação identificados:

1. É possível obter uma estimativa de peso de uma corvina através de uma deteção realizada numa fotografia?
2. É possível obter uma estimativa de peso de uma corvina através de uma deteção realizada numa fotografia, com um desvio médio inferior a 15%?
3. É possível identificar a existência de um período de alimentação, em imagens com vários peixes, através da análise e comparação de áreas?
4. É possível aplicar os procedimentos desenvolvidos a filmagens?
5. É possível transpor os procedimentos desenvolvidos para um ambiente real?

Os resultados obtidos através da testagem e avaliação dos procedimentos desenvolvidos serão utilizados para dar resposta às questões acima enumeradas. Estas mesmas respostas serão fatores relevantes na utilidade e aplicabilidade futura das soluções desenvolvidas.

4.2.2 Estratégias de Obtenção de Dados

Para o desenvolvimento dos procedimentos necessários à obtenção de uma solução para os problemas propostos serão necessários quatro conjuntos de dados distintos: o primeiro conjunto de dados será utilizado para treinar o modelo *YOLOv4*, o segundo *dataset* será utilizado para avaliar esse mesmo modelo na perspetiva de realizar

estimativas de pesos, o terceiro conjunto de dados agregará imagens a utilizar na tarefa de identificação de períodos de alimentação e o último *dataset* será utilizado para o processo de treino do modelo de regressão linear simples.

Tal como mencionado acima, o primeiro *dataset* foi utilizado para treinar o modelo de visão por computador. A principal função deste modelo é a deteção e identificação de uma ou mais corvinas em imagens. Dada a escassez de uma grande quantidade de imagens de corvinas devidamente etiquetadas, optou-se por uma solução mais genérica, recorrendo ao treino do modelo através de imagens de peixes, independentemente da sua espécie.

Todas as imagens utilizadas durante o processo de treino foram obtidas com recurso ao *Open Images Dataset V6* (Kuznetsova et al., 2018), disponibilizado pela *Google*. Este *website* disponibiliza cerca de nove milhões de imagens etiquetadas e pertencentes a múltiplas categorias. Um total de oitocentas imagens da categoria "Fish" foram descarregadas deste repositório, utilizando um software denominado *Open Images Dataset V4 Toolkit (OIDv4_Toolkit)* (Vittorio, 2018). As anotações dessas mesmas imagens foram, em seguida, convertidas para o formato aceite pela arquitetura *YOLOv4*, com recurso a um *software* resultante de um *fork* do projeto *OIDv4_Toolkit*. Na Figura 40 e na

Figura 41, respetivamente pode ser visualizados exemplos de uma imagem utilizada durante o processo de treino e das respetivas coordenadas da *bounding box* que diz respeito ao objeto presente na imagem.

Figura 40 - Exemplo de imagem utilizada no processo de treino do modelo YOLOv4

```
0 0.4883203125 0.5670666829268292 0.839375 0.5395760000000001
```

Figura 41 - Exemplo de coordenadas no formato utilizado pelo modelo YOLOv4

O segundo conjunto de imagens, utilizado para avaliar o procedimento de estimativa de pesos, foi composto por imagens de corvinas. Uma vez que os recursos fornecidos pela empresa proponente não eram utilizáveis, principalmente devido à sua resolução, as imagens utilizadas para esta avaliação foram capturadas em superfícies comerciais, com recurso a uma máquina fotográfica *Canon PowerShot G9X Mark II*, cujas características se encontram sistematizadas na Tabela 26.

Tabela 26 - Características da máquina fotográfica Canon PowerShot G9X Mark II

Características	Dimensão
Largura do sensor	13.2 milímetros
Altura do sensor	8.8 milímetros
Focal Length	10.2 milímetros

Além das imagens, foram também recolhidos dados relativos à distância entre o peixe e a máquina fotográfica no momento da foto, ao peso e ao comprimento de cada sujeito, para que pudessem ser estabelecidas comparações com as estimativas realizadas. Na Figura 42 é possível visualizar uma das imagens utlizadas no processo de avaliação onde se encontra presente uma corvina com cerca de 65 centímetros de comprimento e com um peso de, aproximadamente, 3 quilogramas e 243 gramas. Esta imagem foi capturada a uma distância de 60 centímetros do animal.

Figura 42 - Exemplo de imagem utilizada no processo de avaliação do modelo YOLOv4

O terceiro *dataset*, foi utilizado no processo de avaliação do modelo *YOLOv4* no contexto da identificação de períodos de alimentação. Este conjunto de dados foi composto por imagens de conjuntos de peixes, em grupo (Figura 43) ou dispersos (Figura 44). Todas as imagens foram obtidas com recurso a uma pesquisa no motor de busca *Google*, através da utilização da expressão "*fish groups being fed*".

Figura 43 - Exemplo de imagem utilizada no processo de avaliação do modelo YOLOv4 no contexto de identificação de zonas de alimentação (grupo de peixes)

Figura 44 - Exemplo de imagem utilizada no processo de avaliação do modelo YOLOv4 no contexto de identificação de zonas de alimentação (peixes dispersos)

Fatores como o comprimento ou altura das imagens utilizadas em processos de avaliação não foram considerados durante o processo de recolha, uma vez que as dimensões de todas as imagens são calculadas e utilizadas de forma dinâmica em todos os *scripts* executados, após as deteções por parte do modelo de visão por computador.

O *dataset* utilizado para a realização do processo de treino do modelo de regressão linear simples é composto por linhas de informação presentes num ficheiro de texto. Cada linha contém dados sobre um determinado individuo, nomeadamente a sua identificação, o seu peso, em gramas, e o seu comprimento, em centímetros. O formato de cada linha encontra-se ilustrado na Tabela 27.

Tabela 27 - Formato da informação contida no ficheiro de texto utilizado para treino do modelo de regressão simples

Indivíduo	Peso (g)	Comprimento (cm)
1	1400	51.5
2	1000	47.5
3	1200	49
4	1060	46
5	1380	48.5

Todas as informações contidas neste conjunto de dados foram fornecidas pela SEAentia, após serem registadas durante um dos processos de pesagens e medições de controlo das corvinas presentes nos seus tanques (Figura 45).

Figura 45 - Registo fotográfico do processo de medição realizado pelos responsáveis da SEAentia

4.2.3 Métricas de Avaliação

Durante as subsecções que se seguem serão analisadas as métricas selecionadas para a realização de uma avaliação dos diferentes modelos selecionados. Para avaliação do modelo de regressão linear simples serão utilizadas métricas como o coeficiente de determinação (r^2), o erro médio absoluto e o erro médio quadrático. Para avaliação do modelo *YOLOv4* serão utilizadas métricas como o *recall,* precisão, *F1-score* e *mean average precision.*

Para que as definições de cada métrica sejam compreendidas de forma intuitiva é necessário abordar, em primeiro lugar, o significado de conceitos como "verdadeiros/falsos positivos" e "verdadeiros/falsos negativos" (Figura 46).

Um resultado denominado "verdadeiro positivo" subentende que tanto o resultado esperado como o resultado da previsão são ambos positivos. De modo semelhante, um resultado "verdadeiro negativo" engloba um resultado esperado negativo e um resultado da previsão igualmente negativo (Dilmegani, 2019).

Um resultado denominado "falso positivo" ocorre quando o resultado esperado e o resultado da previsão diferem, sendo que o resultado esperado é negativo e o resultado da previsão é positivo. Por fim, um resultado "falso negativo" é também caracterizado pela diferença entre o resultado esperado e o resultado da previsão, no entanto, neste caso, o resultado esperado é positivo e o resultado da previsão é positivo (Dilmegani, 2019).

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figura 46 - Esquematização da relação entre valores reais e valores previstos no contexto de Verdadeiros/Falsos Positivos/Negativos (Dilmegani, 2019)

Os resultados obtidos através da aplicação das métricas a cada modelo serão analisados com o objetivo de compreender de que forma as condições de realização do projeto e de treino dos diferentes modelos impactaram o seu desempenho, na tentativa de obtenção de soluções para os problemas estudados.

4.2.3.1 Coeficiente de Determinação

O coeficiente de determinação indica a proporção de variação da variável dependente que pode ser explicada pela variável independente de um modelo de regressão. O valor máximo que o coeficiente de determinação pode atingir é 1, não possuindo limite mínimo, uma vez que o modelo poderá ser sempre arbitrariamente pior (Glen, 2012).

Esta mesma métrica pode também convertida numa percentagem que indica o valor percentual de pontos que coincidem com a reta de regressão de um determinado modelo.

O utilidade do coeficiente de determinação reside, principalmente, na sua capacidade de ajudar a prever a probabilidade de eventos futuros apresentarem resultados expectáveis (Glen, 2012).

4.2.3.2 Erro Médio Absoluto e Erro Médio Quadrático

O erro médio absoluto pode ser definido como a média de todos os erros absolutos, obtidos através do cálculo da média de todas as diferenças absolutas entre o valor da previsão e o valor real (Fürnkranz et al., 2011a). A Equação 6, onde *n* representa o número total de previsões, x_i um valor real e *x* o valor da respetiva previsão, ilustra o cálculo do MAE.

Equação 6 - Expressão para cálculo do erro médio absoluto

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

O erro quadrático médio pode ser obtido através do cálculo da média de todos os erros absolutos ao quadrado (Fürnkranz et al., 2011b), tal como traduzido na Equação 7, onde *n* representa o número total de previsões, x_i um valor real e *x* o valor da respetiva previsão.

Equação 7 - Expressão para cálculo do erro médio quadrático

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|^2$$

Emboras estas duas métricas estejam relacionadas com o erro do modelo, ambas apresentam valores substancialmente diferentes, devido às características das funções que permitem o seu cálculo. Para aplicação ao problema em questão, é possível que a métrica mais relevante seja o erro quadrático médio, uma vez que é dada uma enfâse superior a grandes diferenças entre valores reais e valores resultantes da previsão.

4.2.3.3 Recall e Precisão

No contexto de *machine learning*, a precisão pode ser descrita como o rácio entre o número de verdadeiros positivos e a soma do número de verdadeiros positivos com o número de falsos positivos. Esta métrica mede a exatidão do modelo na classificação de previsões positivas e pode ser escrita, formalmente, de acordo com a Equação 8 (Gad, 2020).

Equação 8 - Expressão para o cálculo da Precisão

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Poistivos}$$

O recall, por outro lado, é uma métrica que descreve o rácio entre o número de previsões positivas corretamente classificadas e o número total de classificações positivas. Com esta métrica é possível interpretar a capacidade que o modelo possuí de realizar previsões positivas, uma vez que, quanto mais alto o recall, maior o número de previsões positivas realizadas. A expressão para o cálculo do recall encontra-se descrita na Equação 9 (Gad, 2020).

Equação 9 - Expressão para o cálculo do *Recall*

$$Recall = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

Embora calculadas com recursos a expressões semelhantes, a precisão e o *recall* descrevem de forma bastante diferente as previsões de um modelo. A precisão representa a confiabilidade do modelo na realização de previsões positivas enquanto o *recall* interpreta essas mesmas previsões de forma diferente, na tentativa de a compreender quantas delas são corretas.

Assim, num modelo com elevado *recall* e baixa precisão, é possível constatar que grande parte das previsões positivas são, de facto, corretas, mas que se verifica a existência de um elevado número de falsos positivos. Paralelamente, um modelo com elevada precisão e baixo *recall*, apesar de ser capaz de classificar apenas um reduzido número de previsões positivas, apresenta uma taxa de acerto elevada nessa mesma classificação (Gad, 2020).

4.2.3.4 F1-Score

A métrica *f1-score* é utilizada, em sistemas de classificação binários, para medir a exatidão de um modelo aplicado a um determinado conjunto de dados (Sasaki, 2007). O F1-Score pode ser calculado através da média harmónica entre a precisão e o *recall* de um determinado modelo, tal como demonstrado na Equação 10.

Equação 10 - Expressão para o cálculo do *F1-Score*

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

4.2.3.5 Average Precision e Mean Average Precision

A precisão média (do inglês *average precision*) é utilizada, neste documento, como um sinónimo do conceito *mean average precision (mAP)*, uma vez que existem apenas duas classes relevantes para o problema colocado, referentes à existência ou não de um peixe/grupo de peixes, independentemente da sua espécie.

A presente métrica pode ser descrita como a área localizada abaixo da *precision-recall curve* e pode ser calculada através da divisão da média de todos os valores de precisão pelo valor 11 (que representa a segmentação dos valores de *recall* em intervalos de 0.1 unidade – 0, 0.1, 0.2, ..., 0.9 e 1). De modo a exemplificar a lógica descrita anteriormente, são apresentadas abaixo as figuras Figura 47 e Figura 48 que descrevem,

respetivamente, uma *precision-recall curve* e o processo de cálculo da precisão média para um determinado problema (Hui, 2018).

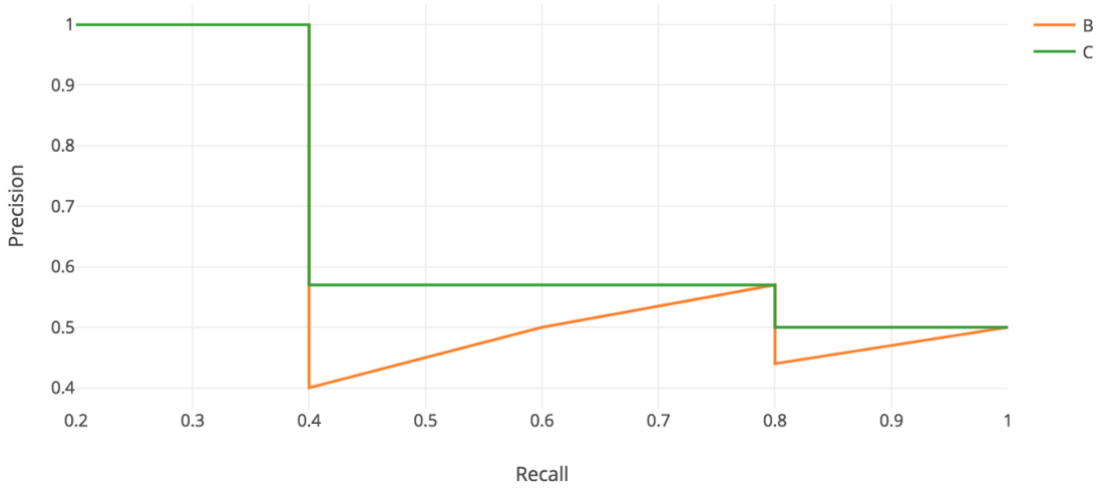

Figura 47 - Exemplo de uma Precision-Recall Curve (Hui, 2018)

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.., 0.9, 1\}} p_{inter\_p}(r)$$
$$= \frac{1}{11}(1 + 1 + 1 + 1 + 0.67 + 0.67 + 0.67 + 0.5 + 0.5 + 0.5 + 0.5)$$
$$\approx 0.728$$

Figura 48 - Exemplo do cálculo da precisão média para uma determinada Precision-Recall Curve (Jie Tan, 2019)

A construção de *datasets* como o *Microsoft COCO (Common Objects in Context),* introduziu novas formas de proceder ao cálculo da precisão média através de variações do índice de *IoU* ou através da análise e agrupamento de objetos dependendo sua área (Lin et al., 2015):

- AP @ *IoU* = 0.50:0.05:0.95 – Principal métrica utilizada para compração de resultados obtidos utilizando do *dataset COCO*. Descreve dez variações de 0.5 unidades do índice de *IoU*, com começo no valor 0.50 e fim no valor 0.95. O valor final da precisão média de cada detetor é computado através do cálculo da média aritmética de todos os valores obtidos para cada índice de *IoU;*

- AP @ *IoU* = 0.75 – Cálculo da precisão média é realizado tendo em conta necessidade do índice de *IoU* de uma previsão ter de ser superior ao valor 0.75 (substituindo o tradicional *threshold* de 0.5);

- AP_S – Métrica para deteção de objetos com área inferior a 32^2 pixéis;

- AP_M – Métrica para deteção de objetos com área compreendida entre 32^2 e 96^2 pixéis;

- AP_L – Métrica para deteção de objetos com área superior a 96^2 pixéis.

As variações no procedimento do cálculo da precisão média permitem distinguir de forma mais eficaz e acertada as características de diferentes modelos, fornecendo aos utilizadores a capacidade de escolher de forma mais informada um determinado detetor para aplicação num determinado problema.

4.3 Sinopse do Capítulo

Durante o presente capítulo foram apresentadas as hipóteses em estudo, com base nos objetivos que se pretendem atingir através do trabalho desenvolvido. Em seguida, foi introduzido o conceito de metodologia de avaliação, conceito este que foi abordado através da sua desconstrução em: parâmetros de avaliação, que descrevem os objetivos a atingir, estratégias de obtenção de dados, que clarificam de forma os *datasets* utilizados foram obtidos e, por fim, métricas de avaliação, utilizadas para avaliação dos modelos selecionados.

5 Desenvolvimento e Estudo Computacional

Tal como referido em secções anteriores, o presente trabalho de investigação apresenta dois objetivos. O primeiro objetivo contempla a realização de uma estimativa do peso de corvinas através de análise e interpretação de uma (ou mais) fotografias. Para este efeito foi desenvolvido um processo semiautomático composto por três fases. O segundo objetivo é referente à identificação de períodos de alimentação através da análise de imagens onde estejam presentes várias corvinas, tendo em conta que um período de alimentação é caracterizado pela existência de um agrupamento de vários peixes numa região da imagem.

Inicialmente, o primeiro objetivo seria atingido através da utilização e análise de vídeos dos tanques geridos pela SEAentia, onde habitam dezenas de corvinas. No entanto, as condições de filmagem e a resolução dos vídeos impossibilitaram a sua correta utilização. Após ponderação, em conjunto com a empresa proponente, foi dada continuidade ao desenvolvimento, ainda que, com recurso a imagens provenientes, na sua grande maioria, de repositórios *online*, na perspetiva de que o trabalho realizado pudesse vir a ter utilidade prática e acrescentar valor aos procedimentos realizados atualmente pela SEAentia.

Tendo em conta as considerações iniciais acima descritas, as subsecções que se seguem descrevem os processos de desenvolvimento das soluções para os problemas propostos. Em primeiro lugar, serão analisadas as configurações iniciais utilizadas durante o processo de desenvolvimento. Em seguida, serão descritos, em pormenor, os procedimentos realizados para realização de estimativas de pesos e, por fim, será apresentada a solução desenvolvida para identificação de períodos de alimentação.

5.1 Configuração do Ambiente de Desenvolvimento & Modelos

5.1.1 Configuração do Ambiente de Desenvolvimento

Nas últimas décadas, o desenvolvimento de unidades de processamento gráfico (*GPU*) e a sua grande capacidade computacional tem causado grande impacto, especialmente

em campos de estudo que beneficiem diretamente da sua utilização, como é o caso da área de visão por computador (Fassold, 2016).

Tradicionalmente, tarefas como a deteção e extração de *features*, características dos modelos desta área, eram realizadas no *CPU* (unidade de processamento central). No entanto, tendo em conta os avanços tecnológicos realizados no passado recente, a execução deste tipo de operações passou a ser realizada com recurso às unidades de processamento gráfico, o que as torna cinco a dez vezes mais eficientes (Fassold, 2016).

De modo a tirar o maior proveito dos benefícios associados à utilização de *GPU's* em tarefas de visão por computador, todos os procedimentos relativos ao desenvolvimento das soluções relatadas neste documento foram realizado com recurso a um ambiente de desenvolvimento virtual.

A plataforma de desenvolvimento selecionada foi o *Google Colaboratory*, onde é possível utilizar *Jupyter Notebooks* para executar código arbitrário na linguagem *Python*. Esta plataforma é acessível através de um navegador (*browser*) e é especialmente indicada para o desenvolvimento e teste de ferramentas de *machine learning* (Bisong, 2019).

O nível de subscrição gratuito desta plataforma garante o acesso a unidades de processamento gráfico *NVIDIA Tesla K80"* que podem ser utilizadas no âmbito do treino e teste de modelos de visão por computador. No entanto, este acesso é limitado a nível temporal, de modo a garantir que existem sempre recursos disponíveis para todos os utilizadores da plataforma. Por norma, o acesso a um *GPU* é garantido durante cerca de oito a dez horas, findo esse período, um *GPU* utilizável só volta a ficar disponível aproximadamente doze horas depois.

Além do *GPU* acima mencionado, o ambiente utilizado, à data dos desenvolvimentos, encontrava-se equipado com *Python* na versão 3.7.13 (van Rossum and Drake Jr, 1995) e com *CUDA* (Vingelmann et al., 2020), um ambiente de desenvolvimento criado pela *NVIDIA* que potencia as funcionalidades das suas unidades de processamento gráfico, na versão 11.1.

5.1.2 Configuração do Modelo YOLOv4

O modelo *YOLOv4* utilizado foi obtido com recurso ao repositório *Github* elaborado por (Bochkovskiy, 2020). Este modelo é baseado no repositório original do mesmo autor (Bochkovskiy et al., 2020b) e apresenta modificações que permitem a utilização de algoritmos *YOLOv2*, *YOLOv3* e *YOLOv4* em ambientes *Windows* e *Linux*.

Além de todos os ficheiros necessários à compilação da *darknet*, uma *framework open-source* que permite a utilização de um rede neuronal convolucional de alta rapidez e relativa simplicidade ao nível da instalação e configuração, a base de código utilizada garante também o acesso a ferramentas como:

- *CUDNN* (Chetlur et al., 2014), uma biblioteca de funções primitivas utilizadas no contexto de *deep learning* e de redes neuronais convolucionais. Incluí vários algoritmos relacionados com convoluções regressivas e progressivas e disponibiliza, também, implementações de conceitos comumente utilizados como camadas de *pooling* e funções de ativação *ReLU* (Fassold, 2016).

- *OpenCV* (Bradski, 2000), uma biblioteca específica para problemas de visão por computador que facilita, por exemplo, a migração de processos de processamento de imagem para o *GPU*, garantindo a sua mais eficiente execução (Fassold, 2016).

- *CMake*, uma ferramenta *open-source* e *cross-platform*, que oferece a possibilidade de definição de uma lógica de compilação relativamente abstrata que é, posteriormente, traduzida para vários formatos utlizados por diferentes ferramentas de compilação de *software* (Yapp, 2011).

A própria *darknet* requer, também, configurações adicionais para que esteja adaptada ao problema em questão no momento do treino. Estas alterações podem ser realizadas num ficheiro de configuração específico que, no caso do projeto desenvolvido, se denomina "yolov4-obj.cfg". A Tabela 28 esquematiza as alterações realizadas no ficheiro de configuração da *darknet*, tendo em conta que o objetivo principal seria preparar esta rede neuronal para a deteção de uma única classe, a classe "Peixe", em múltiplas imagens ou vídeos.

Adicionalmente, são necessárias alterações a dois ficheiros, obj.names e obj.data, utilizados para dar a conhecer ao modelo a nomenclatura das classes a identificar, a localização dos datasets de treino e teste, e a localização onde devem ser guardadas as cópias de segurança do progresso do treino, realizadas periodicamente. Na tabelas abaixo (Tabela 29 e Tabela 30) encontra-se descrito o conteúdo dos ficheiros obj.names e obj.data, respetivamente.

Tabela 28 - Parametrização do ficheiro *yolov4-obj.cfg*

Parâmetro	Valor Definido	Observações
Batch	64	Número de imagens a processar em cada iteração

Subdivisions	16	Número de divisões de cada *batch*. No contexto atual, um *batch* de 64 imagens seria processado de 16 em 16 imagens
Max_batches	6000	Número de classes * 2000, no entanto, não pode assumir um valor inferior a 600
Steps	4800, 5400	80% e 90% do valor de max_batches, respetivamente
Width	416	Pode tomar o valor de qualquer múltiplo de 32, no entanto, o valor predefinido 416 poderá ser suficiente para obtenção dos resultados desejados
Heigth	416	Mesma lógica aplicada ao parâmetro “Width”
Classes	1	Apenas uma classe a ser detetada, classe “Peixe”
Filters	18	(Número de classes + 5) * 3

Tabela 29 - Conteúdo do ficheiro *obj.names*

Parâmetro	Valor Definido
[sem valor]	Fish

Tabela 30 - Conteúdo do ficheiro *obj.data*

Parâmetro	Valor Definido
Classes	1
Train	data/train.txt
Valid	data/test.txt
Names	data/obj.names
Backup	/mydrive/yolov4/backup

Tal como é possível verificar a partir da análise da Tabela 30, a segunda e terceira linhas, que dizem respeito aos parâmetros "Train" e "Valid", respetivamente, apontam para dois ficheiros de texto distintos. Cada um destes ficheiros contém todos os caminhos relativos para as imagens a utilizar nos processos de treino ("Train") e teste ("Valid"). Na Figura 49 encontra-se ilustrado um excerto do conteúdo do ficheiro *train.txt*.

```
1    data/obj/2aa4fa5fdd0a3e59.jpg
2    data/obj/fb9cb6278983183f.jpg
3    data/obj/689bb2ab73c9bb03.jpg
4    data/obj/9bbb4ac122fa3398.jpg
5    data/obj/d76f6ea7017f4ffe.jpg
```

Figura 49 - Excerto do conteúdo do ficheiro *train.txt*

Tendo em conta a grande quantidade de imagens a utilizar, especialmente no processo de treino, foi implementado um *script*, em *Python*, capaz de automatizar o processo de escrita de caminhos relativos em cada ficheiro de texto. Um excerto desse mesmo *script* está representado na Figura 50.

```
 1 image_files = []
 2 os.chdir(os.path.join("data", "obj"))
 3 for filename in os.listdir(os.getcwd()):
 4     if filename.endswith(".jpg"):
 5         image_files.append("data/obj/" + filename)
 6 os.chdir("..")
 7 with open("train.txt", "w") as outfile:
 8     for image in image_files:
 9         outfile.write(image)
10         outfile.write("\n")
11     outfile.close()
12 os.chdir("..")
```

Figura 50 - *Script* implementado para geração do ficheiro *train.txt*

5.1.3 Configuração do Modelo de Regressão Linear Simples

O modelo de regressão linear simples foi treinado e testado no mesmo ambiente virtual *Google Colaboratory* mencionado na secção anterior.

De um modo geral, as configurações necessárias para utilização deste modelo residiram na instalação de bibliotecas para simplificação do processo de desenvolvimento. Os principais *softwares* externos utilizados foram: a biblioteca *Scikit-Learn,* a biblioteca *Matplotlib* e a biblioteca *Numpy*.

A biblioteca *Scikit-Learn* disponibiliza múltiplos algoritmos de *machine learning* que podem ser utilizados no desenvolvimento de soluções para problemas de aprendizagem supervisionada e não supervisionada (Pedregosa FABIANPEDREGOSA et al., 2011). São também disponibilizadas funções de cálculo de métricas associadas aos diferentes algoritmos, como é o caso do erro médio absoluto e do erro médio quadrático.

Já a biblioteca *Matplotlib* (Hunter, 2007) agrega um conjunto de funcionalidades úteis em tarefas de visualização de dados. E, por fim, a biblioteca *Numpy* (Harris et al., 2020), utilizada massivamente no desenvolvimento de algoritmos matemáticos em *Python*, disponibiliza múltiplas funções matemáticas de alto nível e oferece a capacidade de utilização de listas n-dimensionais de uma forma mais eficiente do que a tradicionalmente utilizada pela linguagem.

5.2 Estimativa de Peso

Durante a presente subsecção serão descritos os procedimentos implementados para realização da estimativa do peso de um determinado peixe.

O desenvolvimento desta solução é composto por três fases, cada uma com características distintas. A **primeira fase** é relativa ao **modelo *YOLOv4*** e corresponde ao processo de treino e avaliação deste modelo, para que ele esteja preparado para detetar peixes de várias espécies, a **segunda fase** diz respeito ao **cálculo do comprimento real do objeto**, fazendo uso da teoria de semelhança de triângulos e aplicando-a à fotografia analisada. Por fim, na **terceira** e última **fase**, é calculada uma **estimativa do peso do sujeito**, com recurso à previsão do seu comprimento real, usada como *input* no modelo de regressão linear simples.

5.2.1 1ª Fase – Modelo YOLOv4 (Treino e Avaliação)

5.2.1.1 Treino do Modelo YOLOv4

O processo de treino de modelo YOLOv4 foi realizado num ambiente virtual *Google Colaboratory*, tal como mencionado em secções anteriores. Para dar início ao processo de treino do modelo foi utilizado o comando *bash* descrito na Figura 51. Este comando invoca a *darknet* (*framework* base para o funcionamento de todo o procedimento), indica o tipo de trabalho a realizar (através do parâmetro *train*), fornece os caminhos relativos para os ficheiros de configuração a utilizar, tais como o ficheiro *obj.data* e o ficheiro *yolov4-obj.cfg* e, por fim, referencia o nome do ficheiro de pesos pré-treinados que reduzirá a duração de todo o processo (*yolov4.conv.137*).

```
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137
```

Figura 51 - Comando *bash* para dar início ao treino do modelo YOLOv4

Após o fim do treino foi utilizado o comando presente na Figura 52 para o cálculo de métricas com a precisão, o *recall*, o F1-*score* e a *mean average precision*. As principais diferenças entre o comando utilizado para obtenção dos valores das métricas e o comando utilizado para dar início ao processo de treino residem no parâmetro que indica a função a realizar (no caso da Figura 52, este parâmetro é denominado *map*) e no parâmetro que indica o ficheiro de pesos a utilizar (neste caso, o ficheiro utilizado é o que foi gerado após o fim do treino, *yolov4-obj_6000.weights*).

```
!./darknet detector map data/obj.data cfg/yolov4-obj.cfg
/mydrive/Tese/yolov4/backup/yolov4-obj_6000.weights
```

Figura 52 - Comando *bash* utilizado para cálculo das métricas

Um excerto do *output* produzido pela execução do comando acima apresentado pode ser visualizado na Figura 53. Na linha 9 e 13 dessa mesma figura encontram-se os valores obtidos para as diferentes métricas: precisão (0.91 para um *threshold* de confiança de 0.25), *recall* (0.92 para um *threshold* de confiança de 0.25), F1-*score* (0.91 para um *threshold* de confiança de 0.25) e *mean average precision* (93.67% para um *threshold intersection over union* de 0.5).

```
 1 calculation mAP (mean average precision)...
 2  Detection layer: 139 - type = 28
 3  Detection layer: 150 - type = 28
 4  Detection layer: 161 - type = 28
 5 1000
 6  detections_count = 11076, unique_truth_count = 4289
 7 class_id = 0, name = Fish, ap = 93.67%           (TP = 3939, FP = 407)
 8
 9  for conf_thresh = 0.25, precision = 0.91, recall = 0.92, F1-score = 0.91
10  for conf_thresh = 0.25, TP = 3939, FP = 407, FN = 350, average IoU = 75.02 %
11
12  IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
13  mean average precision (mAP@0.50) = 0.936734, or 93.67 %
14 Total Detection Time: 75 Seconds
```

Figura 53 - *Output* do comando presente na Figura 52

Tendo em conta as condicionantes estabelecidas pela utilização de um nível de subscrição gratuito nesta plataforma da *Google*, é estimado que o processo de treino tenha demorado cerca de seis dias até estar completo na sua totalidade. O facto de o modelo utilizado possuir a capacidade de realização periódica de *backups* (ficheiros com informações relativas ao peso de cada neurónio num determinado momento) facilitou bastante o treino do algoritmo, uma vez que o progresso realizado num determinado dia não era simplesmente descartado no momento do término do tempo de execução associado à subscrição.

5.2.1.2 Avaliação do Modelo YOLOv4

Para a realização dos procedimentos de teste do modelo *YOLOv4* foram utilizados dois comandos *bash* distintos.

O primeiro comando, representado na Figura 54, descreve a realização de uma deteção numa imagem individual, passada por parâmetro, com recurso à sua localização relativa e com o nome *60_2.jpg*. Além da utilização do caminho relativo da imagem a avaliar, este comando apresenta ainda uma outra característica que o distingue dos restantes analisados anteriormente, a utilização do parâmetro *test*, que indica a realização de um período de avaliação. A *flag -ext_output* indica que as coordenadas dos objetos detetados devem ser escritas na consola.

```
  # deteção numa imagem específica
1 !./darknet detector test data/obj.data cfg/yolov4-obj.cfg
2 /mydrive/Tese/yolov4/backup/yolov4-obj_last.weights -ext_output
  60_2.jpg
```

Figura 54 - Comando *bash* utilizado para realização de uma deteção num imagem específica

Ao realizar deteções individuais é possível ter acesso à imagem resultante dessa mesma deteção, imagem essa que incluí a *bounding box* traçada no decorrer da execução (Figura 55).

Figura 55 - Imagem resultante da realização de uma deteção individual

O segundo comando utilizado para a realização de testes, ilustrado na Figura 56, permite a realização de deteções em múltiplas imagens de uma só vez. Este comportamento deve-se à utilização da *flag -out* e da passagem de um parâmetro com um nome de um ficheiro em formato *json* (*result.json*). Tal como mencionando em secções anteriores, o ficheiro *test.txt* contém todos os caminhos relativos para as imagens a utilizar na deteção.

```
  # deteção num conjunto de imagens e escrita do output para um
  ficheiro JSON
1 !./darknet detector test data/obj.data cfg/yolov4-obj.cfg
2 /mydrive/Tese/yolov4/backup/yolov4-obj_last.weights -ext_output -out
  result.json < data/test.txt
```

Figura 56 - Comando *bash* utilizado para realização de deteções num conjunto de imagens

O ficheiro *result.json*, produzido no final da avaliação, possuí uma estrutura semelhante àquela que pode ser visualizada no excerto presente na Figura 57.

Para cada imagem analisada é criado um objeto *json* que contém:

- *Frame_id* – Número de ordem da imagem analisada;
- *Filename* – Caminho relativo para o diretório onde se encontra guardada a imagem;
- Lista de *objects* detetados na imagem, com as seguintes características:
    - *Class_id* – Identificador da classe detetada (no caso atual apenas existe uma classe, a classe *Fish*, cujo identificador é 0);
    - *Name* – Nome associado à classe;
    - *Center_x, center_y, width, height* – Coordenadas da *bounding box* traçada após a deteção (exemplificado na Figura 58);
    - *Confidence* – Grau de confiança da deteção.


```
 1 [
 2 {
 3  "frame_id":1,
 4  "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_1.jpg",
 5  "objects": [
 6   {"class_id":0, "name":"Fish", "relative_coordinates":
 7
 8 {"center_x":0.473018, "center_y":0.504056, "width":0.543378, "height":0.278948},
 9 "confidence":0.996137}
10  ]
11 },
12 {
13  "frame_id":2,
14  "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_4.jpg",
15  "objects": [
16   {"class_id":0, "name":"Fish", "relative_coordinates":
17
18 {"center_x":0.488616, "center_y":0.484681, "width":0.920438, "height":0.712567},
19 "confidence":0.989885}
20  ]
21 }
22 ]
```


Figura 57 - Excerto do ficheiro *result.json*

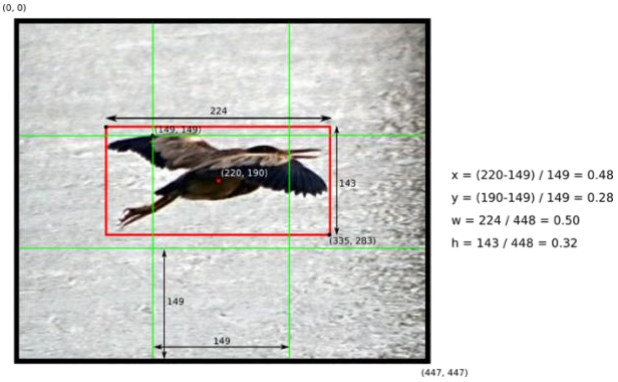

Figura 58 - Exemplo das coordenadas de uma deteção

Os campos de maior interesse para a elaboração do restante procedimento de estimativa de pesos serão as coordenadas da *bounding box* e o grau de confiança da deteção. Ambos serão utilizados na secção seguinte para o cálculo do comprimento real de um exemplo de um objeto analisado.

5.2.2 2ª Fase – Cálculo do Comprimento Real

Após a utilização do modelo de visão por computador para realização de uma deteção e posterior obtenção das coordenadas do objeto na imagem, por via do ficheiro *json* produzido pelo modelo, é necessário calcular o tamanho real do sujeito, para que este seja utilizado como *input* num modelo de regressão linear simples.

Através da aplicação do princípio da Semelhança de Triângulos, ilustrado na Figura 59 para o contexto da captura de uma fotografia, é possível obter o valor do comprimento do objeto fotografado ("*Field dimension*") em metros.

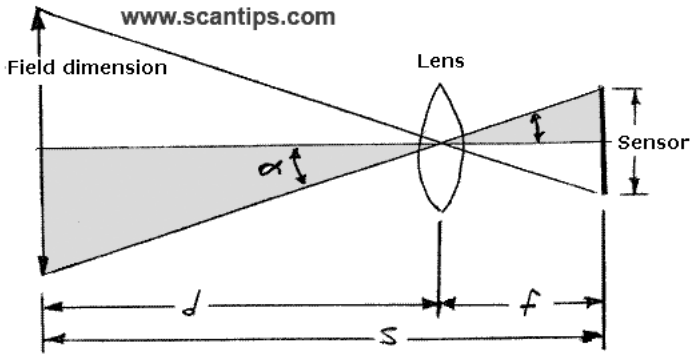

Figura 59 - Princípio da Semelhança de Triângulos no contexto da captura de uma fotografia (Fulton, 2015)

A Equação 112 traduz a semelhança entre os dois triângulos sombreados a cinza na Figura 59.

Equação 11 - Expressão de semelhança de triângulos aplicada a uma fotografia

$$\frac{Dimensão\ Sensor\ (mm)}{Focal\ Length, f\ (mm)} = \frac{Comprimento\ real\ do\ objeto\ (m)}{Distância\ ao\ objeto, d\ (m)}$$

Pressupondo que, para um determinado exemplo, a distância a partir da qual a fotografia é capturada e a focal length da máquina fotográfica utilizada são conhecidas, torna-se necessário calcular o comprimento do objeto no sensor para que seja possível escrever a Equação 11, acima apresentada, em função da variável que se pretende obter, CRO (ou "comprimento real do objeto"), obtendo assim a Equação 12.

Equação 12 - Expressão que traduz o cálculo do Comprimento Real de um objeto numa fotografia, em metros

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

Para o cálculo do comprimento do objeto no sensor é necessário conhecer o tamanho do sensor pertencente à máquina fotográfica utilizada, em milímetros, o comprimento da fotografia capturada, em pixéis, e o comprimento do objeto na fotografia, também em pixéis.

Embora os primeiros dois elementos sejam conhecidos *à priori*, com recurso ao estudo das especificações da câmara utilizada e das informações digitais da imagem capturada, o terceiro elemento, comprimento do objeto na fotografia, é ainda desconhecido. Para o seu cálculo deve ser utilizado o campo *width*, presente no ficheiro *json*.

Com recurso à Figura 58, é possível observar que o comprimento do objeto na imagem, COI, pode ser alternativamente definido como comprimento da *bounding box* associada à sua deteção e, consequentemente, pode ser calculado através da multiplicação entre o valor do campo *width* e o tamanho da imagem capturada, em pixéis, tal como descrito na Equação 13.

Equação 13 - Expressão que traduz o cálculo do Comprimento do Objeto na Imagem, em pixéis

$$COI\ (pixels) = Largura\ da\ Imagem\ (pixels) * width\ (YOLOv4\ output)$$

Conhecidos estes dados, é possível obter o comprimento do objeto no sensor, COS, através da Equação 14.

Equação 14 - Expressão que traduz o cálculo do Comprimento do Objeto no sensor, em milímetros

$$COS\ (mm) = \frac{\text{Comprimento } do\ Sensor\ (mm) * Comprimento\ Objeto\ na\ Imagem\ (pixels)}{Largura\ da\ Imagem\ (pixels)}$$

Após a obtenção do comprimento do objeto no sensor, em milímetros, estão disponíveis todos os dados necessários ao cálculo do comprimento real do objeto, CRO, em metros, recorrendo a Equação 15.

Equação 15 - Expressão que traduz o cálculo do Comprimento Real do Objeto, em metros

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

A obtenção do valor de comprimento real do peixe analisado encerra a segunda fase do procedimento de estimativa de peso. Em seguida, na subseção 5.2.2.1, é exemplificado o cálculo analítico do comprimento real de uma corvina. Já na subseção 5.2.2.2, é demonstrado o processo de implementação deste mesmo cálculo na pipeline.

5.2.2.1 Exemplificação do Processo de Cálculo Analítico do Comprimento Real

De modo a exemplificar o processo descrito anteriormente, será calculado, em seguida o comprimento real da corvina presente na Figura 60, utilizando as coordenadas da sua *bounding box*, obtidas após a sua deteção, através do ficheiro *json* produzido pelo modelo *YOLOv4* (excerto presente na Figura 61). À data da fotografia, a corvina apresentava um comprimento real de, aproximadamente, 63 centímetros.

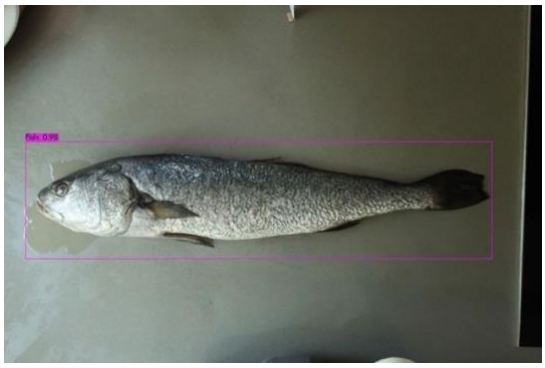

Figura 60 - Deteção utilizada a título de exemplo para cálculo do comprimento real de um sujeito presente numa fotografia

```
{
 "frame_id":3,
 "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_2.jpg",
 "objects": [
  {"class_id":0, "name":"Fish", "relative_coordinates":
{"center_x":0.473614, "center_y":0.546175, "width":0.867367, "height":0.328135},
"confidence":0.982146}
 ]
}
```


Figura 61 - Excerto do ficheiro *json* resultante da deteção realizada na Figura 60

O primeiro passo para o cálculo do comprimento real do peixe detetado é a utilização da Equação 12, com o objetivo de obter o comprimento do objeto na imagem. Para o cálculo deste valor, descrito na Figura 62, foi utilizado o comprimento da imagem original, em pixéis, obtido através da análise das propriedades da fotografia e a *width* da *bounding* box resultante da deteção.

$$COI\ (pixéis) = Largura\ da\ Imagem\ (pixéis) * width\ (YOLOv4\ output)$$

$$COI\ (pixéis) = 2048 * 0.867367$$

$$COI\ (pixéis) = 1776.37\ px$$

Figura 62 - Cálculo do comprimento de um objeto numa imagem

Conhecido o comprimento do objeto na imagem, é possível proceder ao cálculo do comprimento do objeto no sensor através da aplicação da Equação 13. Para a utilização desta expressão é necessário conhecer o comprimento do sensor a partir do qual a imagem foi capturada. Este valor não é fixo e poderá variar entre diferentes modelos de máquinas fotográficas, no entanto, para a máquina utilizada, *Canon PowerShot G9X Mark II*, o comprimento do sensor é de 13.2 milímetros.

$$COS\ (mm) = \frac{\text{Comprimento}\ do\ Sensor\ (mm) * Comprimento\ Objeto\ na\ Imagem\ (pixéis)}{Largura\ da\ Imagem\ (pixéis)}$$

$$COS\ (mm) = \frac{13.2 * 1776.37}{2048}$$

$$COS\ (mm) = 11{,}45\ mm$$

Figura 63 - Utilização da Equação 14, a título de exemplo, para cálculo do comprimento de um objeto no sensor

Sabendo o comprimento do objeto no sensor, resta apenas aplicar a Equação 15 para conhecer a estimativa de comprimento do animal fotografado (Figura 64). Para a utilização desta expressão é necessário conhecer duas variáveis: a distância entre o animal e a lente (aproximadamente 60 centímetros) e a *focal length* da máquina fotográfica utilizada (10.2 milímetros).

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

$$CRO\ (m) = \frac{0.6 * 11.45}{10.2}$$

$$CRO\ (m) = 0{,}67\ m$$

Figura 64 - Utilização da Equação 15, a título de exemplo, para cálculo do comprimento real de um objeto

De acordo com a estimativa realizada, o comprimento da corvina presente na Figura 60 é de, aproximadamente, 0.67 metros, ou, 67 centímetros. Tendo em conta que o seu comprimento real seria de, sensivelmente, 63 centímetros, é possível verificar que o valor estimado representa um desvio percentual de cerca de 6.3% face ao valor esperado.

5.2.2.2 Implementação do Processo de Cálculo do Comprimento Real

Dado o caracter relevante dos cálculos anteriormente mencionados para o projeto em questão, foi realizada uma implementação que segue a lógica de obtenção do comprimento real de um objeto para cada deteção realizada pelo modelo de visão por computador.

A Figura 65 ilustra um excerto do código implementado. Neste excerto é lido o ficheiro *json*, resultante da deteção, e o seu conteúdo é guardado na variável *resultJSON*. Cada elemento da lista de objetos *json* é percorrido em seguida e são obtidas, dinamicamente, as dimensões das imagens, com recurso ao parâmetro *filename* (que indica o caminho relativo para cada ficheiro).

```
#### CONSTANTES

WIDTH_IMGS = []
# focal length da camera utilizada (mm)
FOCAL_LENGTH = 10.2
# distância do objeto à camera (cm)
DISTANCIA_40 = 40
DISTANCIA_60 = 60
# tamanho do sensor em largura e altura (mm)
TAMANHO_SENSOR_LARGURA = 13.2
TAMANHO_SENSOR_ALTURA = 8.8
#### CONSTANTES

f = open("/content/darknet/result.json")
resultJSON = json.load(f)
confianca = 0
j = 0
lista_filename_width = []

for i in resultJSON:
  filename = i['filename']
  image = PIL.Image.open(str(filename))
  # width e height da imagem em pixeis
  width_img, height_img = image.size
  WIDTH_IMGS.append(width_img)
  if len(i['objects']) > 1:
    confianca = 0
    for j in i['objects']:
      if j['confidence'] > confianca:
        confianca = j['confidence']
        indice = i['objects'].index(j)
    comprimento_objeto_na_imagem = str(i['objects'][indice]['relative_coordinates']
['width'] * width_img)
    lista_filename_width.append(i['filename'] + ";" + comprimento_objeto_na_imagem
+ ";" + str(width_img))
  else:
    comprimento_objeto_na_imagem = str(i['objects'][0]['relative_coordinates']
['width'] * width_img)
    lista_filename_width.append(i['filename'] + ";" + comprimento_objeto_na_imagem
+ ";" + str(width_img))
```

Figura 65 - Implementação do cálculo do comprimento do objeto na imagem

Para imagens onde tenha sido detetado mais do que um animal, é preservada apenas a deteção com maior grau de confiança.

Na lista *lista_filename_width* são guardados, para cada imagem, o seu caminho relativo, o valor do comprimento do objeto na imagem e o comprimento da imagem.

Para a realização dos restantes cálculos é necessário estabelecer um procedimento para que a distância da máquina fotográfica ao objeto seja obtida dinamicamente. Como solução para esta problemática estabeleceu-se que a nomenclatura de cada imagem deveria incluir o valor dessa mesma distância e o número de ordem da imagem. Assim, uma ficheiro de imagem com o nome “60_1.jpg”, por exemplo, indica que aquela é a primeira fotografia captada a uma distância de, aproximadamente, 60 centímetros.

Tendo definido este pré-requisito, torna-se possível implementar o restante procedimento. Na Figura 66 encontra-se representada essa mesma implementação, com comentários que ilustram possíveis valores das variáveis de modo a facilitar a

compreensão. O valor do comprimento real do objeto detetado será guardado na variável *real_object_width_centimeters*.

```
for filename_width in lista_filename_width:
  # filename_width =
  # data/custom_val/mydrive/Tese/yolov4/validation_custom/60_2.jpg;1776.37;2048
  # comprimento_obj_na_imagem = 1776.37
  comprimento_obj_na_imagem = filename_width.split(';')[1]
  # filename_path = data/custom_val/mydrive/Tese/yolov4/validation_custom/60_2.jpg
  file_path = filename_width.split(';')[0]
  # filename_img = 60_2.jpg
  filename_img = file_path.split('/')[-1]
  # distancia = 60
  distancia = filename_img..split('_')[0]

  object_width_on_sensor = (TAMANHO_SENSOR_LARGURA * float(comprimento))
                                                    / float(i.split(';')[2])

  real_object_width_centimeters = (distancia_camera * object_width_on_sensor)
                                                                / FOCAL_LENGTH
```

Figura 66 - Implementação do cálculo do comprimento real do objeto na imagem

5.2.3 3ª Fase – Modelo de Regressão Linear & Estimativa do Peso

A fase final do procedimento de estimativa de peso de uma corvina reside no treino e avaliação de um modelo de regressão linear simples capaz de realizar previsões de peso (em gramas) para um determinado comprimento (em centímetros). Nas subsecções que se seguem serão descritos os procedimentos de treino, avaliação e integração do modelo no projeto final.

5.2.3.1 Treino e Avaliação do Modelo de Regressão Linear

Os dados utilizados para o treino do modelo foram fornecidos, no formato de ficheiro *csv* (*comma separated values*), pela SEAentia, tal como descrito em secções anteriores. A estrutura deste ficheiro pode ser visualizada na Tabela 27, mas, por conveniência, é também exemplificada na Tabela 31.

Tabela 31 - Conteúdo e estrutura do ficheiro utilizado para treino do modelo de regressão linear simples (adaptação da Tabela 27)

Indivíduo	Peso (g)	Comprimento (cm)
1	1400	51.5
2	1000	47.5
3	1200	49

A leitura do ficheiro foi realizada com recuso ao método *read_csv()* disponibilizado pela biblioteca *Pandas*. Na Figura 67 é possível visualizar um excerto da implementação que, além de ler o ficheiro (chamado "Pesos_Medidas_Sandbox.csv"), procede à divisão do *dataset* em conjuntos de treino e teste. Esta divisão, regrada pelo parâmetro *test_size*, presente na função *train_test_split()*, foi realizada com um rácio de 75% dados de treino e 25% dados de teste, o que equivale a 444 linhas de informação de treino e 148 linhas de dados de teste. A representação gráfica da distribuição dos valores de treino e teste encontra-se representada na Figura 68 e foi construída com recurso à biblioteca *Matplotlib*.

```
# Leitura do ficheiro de dados fornecido pela SEAentia
dados = pd.read_csv('/mydrive/Tese/regression/Pesos_Medidas_Sandbox.csv', sep=';')

# Conversão dos valores das colunas para o tipo de dados "float"
dados['Peso']=dados['Peso'].astype(float)
dados['Comprimento']=dados['Comprimento'].astype(float)

# Divisão do dataset em conjunto de treino e de teste
x_train, x_test, y_train, y_test =
        train_test_split(dados.Comprimento, dados.Peso, test_size = 0.1)
```

Figura 67 - Implementação do processo de leitura do ficheiro de dados e divisão do *dataset*

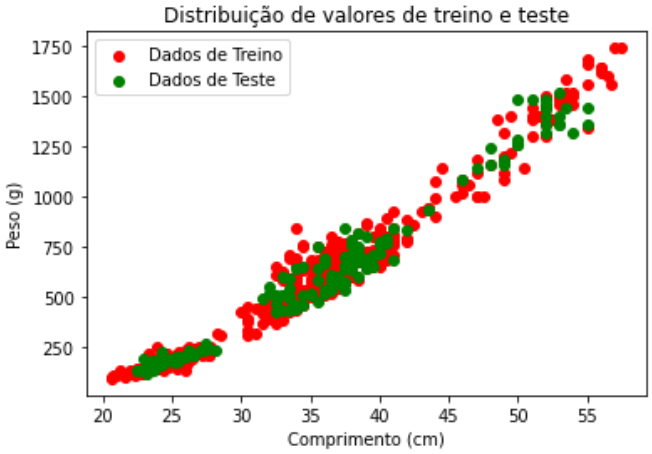

Figura 68 - Representação gráfica da distribuição dos valores de treino e teste

O modelo de regressão linear utilizado foi importado da biblioteca *Scikit-learn* e instanciado, com recurso ao método *LinearRegression()*, de modo a ser utilizável durante a execução. O treino do modelo foi realizado com recurso à função *fit()*, que pode ser invocada a partir do objeto anteriormente instanciado. Esta função utiliza, como parâmetros, as listas de dados de treino, criadas após a divisão do *dataset*. De modo que todos os dados se encontrem nos formatos necessários para a realização do treino, foi utilizada a biblioteca *Numpy* para conversão da variável *x_train* numa lista de listas, cada uma com apenas um elemento. Os procedimentos descritos neste parágrafo encontram-se ilustrados na Figura 69.

```
1 # Instanciação do modelo de regressão linear simples
2 linear_regression = LinearRegression()
3
4 # Treino do modelo de regressão linear, com recurso à função fit()
5 linear_regression.fit(np.array(x_train).reshape(-1,1), y_train)
```

Figura 69 - Instanciação e treino do modelo de regressão linear simples

O processo de avaliação do modelo de regressão linear simples pode ser realizado com recurso a uma única função, a função *predict()*, invocada através do objeto que contém o modelo. Esta função necessita apenas de um parâmetro, a lista de valores de teste para os quais é necessário realizar a previsão. Na Figura 70 encontra-se a implementação realizada para a previsão de pesos, tendo em conta os comprimentos presentes na lista *x_test*.

```
1 # Previsão de pesos para a lista x_test (contém valores de comprimento)
2 weight_predictions = linear_regression.predict(np.array(x_test).reshape(-1,1))
```

Figura 70 - Implementação da utilização da função *predict()* para previsão de um conjunto de pesos a partir de uma lista de comprimentos

O diagrama de dispersão entre os valores de teste e a reta de regressão, resultante das previsões realizadas, pode ser visualizado na Figura 71. À semelhança da representação gráfica da distribuição de valores de treino e teste, este diagrama também foi construído com recurso à biblioteca *Matplotlib*.

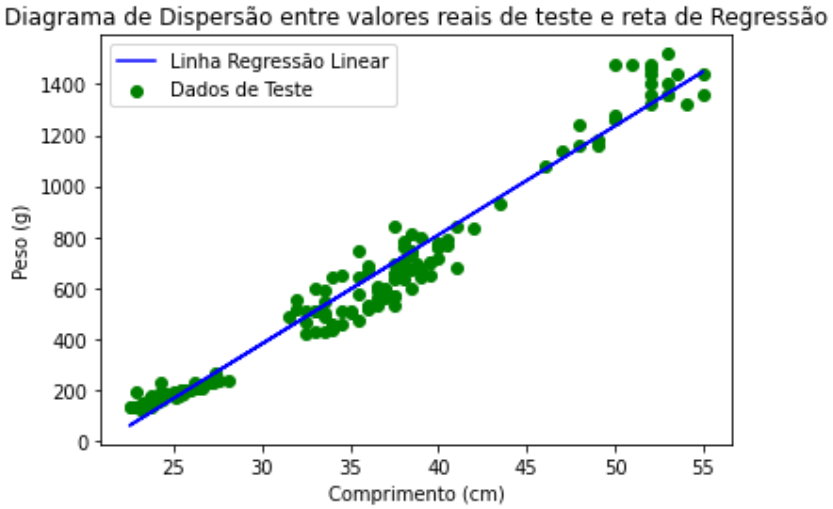

Figura 71 - Diagrama de dispersão entre os valores de teste e a reta de Regressão

Métricas como o coeficiente de determinação das previsões (r^2), erro médio absoluto e erro médio quadrático podem ser calculadas com recurso a métodos também disponibilizados pelo *package metrics* da biblioteca *Scikit-learn*. Na Figura 72 é possível visualizar a invocação destes métodos assim como o valor de cada métrica.

```
from sklearn import metrics

# r2_score = 0.9641
r2_score = metrics.r2_score(y_test, weigth_predictions)

# erro médio absoluto, mae = 59.7121
mae = metrics.mean_absolute_error(y_test, weigth_predictions)

# erro médio quadrático, mse = 5679.0812
mse = metrics.mean_squared_error(y_test, weigth_predictions)
```

Figura 72 - Invocação de funções para cálculo de métricas e respetivos valores

De acordo com a definição de cada métrica, é possível proceder à interpretação dos valores obtidos, tal como clarificado em seguida:

- Coeficiente de Determinação (r^2) – Esta métrica estatística indica a proporção de variação da variável dependente que pode ser explicada pela variável independente de um modelo de regressão. O valor máximo que o coeficiente de determinação pode atingir é 1, não possuindo limite mínimo, uma vez que o modelo poderá ser sempre arbitrariamente pior. O valor obtido, 0.9641, indica que uma percentagem muito grande da variação dos valores associados à variável "peso" estão dependentes da variação dos valores da variável "comprimento";

- Erro Médio Absoluto – O erro médio absoluto representa a diferença entre os valores previstos e os valores esperados. No modelo estudado, esta métrica tem o valor de 59.7121, o que significa que, em média, o valor do peso resultante da previsão do modelo e o valor do peso real do sujeito diferem em aproximadamente 60 gramas.

- Erro Médio Quadrático – Esta métrica é calculada de forma semelhante ao erro médio absoluto, no entanto, o facto de a diferença entre valor esperado e valor previsto ser elevada ao quadrado proporciona uma enfâse superior aos desfasamentos com valores mais elevados. O valor obtido, 5679.0812, indica que, em determinados locais, a linha de regressão se encontra bastante distante dos valores esperados, tal como pode ser visualizado com recurso à Figura 71.

O valor das métricas erro médio absoluto e erro médio quadrático poderão estar relacionados com a falta de exatidão no momento das medições e pesagens das corvinas. Tal como é possível verificar a partir da análise da Figura 73, capturada durante a extração dos dados utilizados no treino do modelo de regressão linear, a medição do comprimento do animal foi realizada de forma pouco precisa, não havendo sequer garantias de que o peixe se encontrava exibido em todo o seu comprimento no momento da medição. Além disso, a ferramenta de medição utilizada não se encontra

corretamente posicionada, o que também terá causado represálias na obtenção do valor do comprimento.

Figura 73 - Exemplo de medição de uma corvina

5.2.3.2 Integração no Projeto

O momento de integração do modelo de regressão linear simples na *pipeline* de realização da estimativa de peso ocorre após o cálculo do valor do comprimento real do sujeito a analisar. Este valor é utilizado como parâmetro da função *predict()*, de modo a ser obtida uma estimativa do peso do animal, ficando assim completo o procedimento que serve de solução ao primeiro problema estudado neste documento (Figura 74).

```
1 # Previsão utilizando o valor estimado do comprimento real do sujeito estudado
2
3 prediction = regr.predict(np.array([real_object_width_centimeters]).reshape(-1,1))
```

Figura 74 - Implementação da previsão de peso para um determinado valor de comprimento

Para demonstração do procedimento completo foi dada continuidade à utilização da Figura 60 que apresentou um valor de comprimento previsto de, aproximadamente, 67 centímetros, tal como comprovado na secção 5.2.2.1. A utilização deste valor como parâmetro na função que realiza a previsão de peso originou o *output* que se encontra presente na Figura 75.

```
 1 ################################################
 2
 3 imagem com largura de 2048 pixéis
 4
 5 imagem: data/custom_val/mydrive/Tese/yolov4/validation_custom/60_2.jpg
 6 capturada a uma distância de 60cm
 7
 8 comprimento em cm: 67.34849647058824cm
 9
10 estimativa de peso em g: 1968.9904649870205g
```

Figura 75 - *Output* da estimativa de peso realizada para a corvina presente na Figura 60

A estimativa de peso produziu o valor 1968.9905 gramas. Tendo em conta que o peso real do animal estudado era de, aproximadamente, 2 quilogramas e 66 gramas, a previsão realizada representa um desvio percentual de, sensivelmente, 4.70%.

5.2.4 Aplicação do Procedimento Desenvolvido

O procedimento de estimativa de peso foi aplicado a mais três exemplos, de modo a analisar a exatidão das previsões. Todas as imagens foram capturadas com recurso à mesma câmara fotográfica anteriormente descrita, a uma distância de, sensivelmente, 60 centímetros do sujeito fotografado. O processo de obtenção dos resultados encontra-se descrito em seguida e é concluído com recurso à Tabela 32, onde é realizada uma sistematização comparativa dos resultados esperados e dos resultados obtidos.

5.2.4.1 Exemplo #1

A Figura 76 foi utilizada como exemplo número 1. Nesta imagem encontra-se presente uma corvina com aproximadamente 63 centímetros de comprimento e com peso na ordem dos 2 quilogramas e 308 gramas.

Figura 76 - Registo fotográfico utilizado como exemplo #1 para testagem do processo de estimativa de peso

O modelo de visão por computador identificou corretamente o ser vivo na figura e, por consequência, produziu a *bounding box* presente na Figura 77, assim como as coordenadas relativas à deteção, que podem ser visualizadas no excerto do ficheiro *json* apresentado na Figura 78.

Figura 77 - *Bounding box* resultante da deteção realizada na Figura 76

```
{
 "frame_id":5,
 "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_3.jpg",
 "objects": [
  {"class_id":0, "name":"Fish", "relative_coordinates":
{"center_x":0.450587, "center_y":0.348311, "width":0.855585, "height":0.615387},
"confidence":0.983320},
  {"class_id":0, "name":"Fish", "relative_coordinates":
{"center_x":0.433298, "center_y":0.057051, "width":0.692916, "height":0.106799},
"confidence":0.385166}
 ]
}
```


Figura 78 - Coordenadas da deteção realizada na Figura 76

Aplicando o mesmo procedimento utilizado anteriormente, é necessário proceder ao cálculo do comprimento do objeto na imagem, através da aplicação da Equação 13, tal como demonstrado na Figura 76.

$$COI\ (pixéis) = \ Largura\ da\ Imagem\ (pixéis) * width\ (YOLOv4\ output)$$

$$COI\ (pixéis) = 638\ * 0.855585$$

$$COI\ (pixéis) = 545.86\ px$$

Figura 79 - Demonstração do cálculo do comprimento do objeto na Figura 76

Sabendo o comprimento do objeto na imagem é possível proceder ao cálculo do comprimento do objeto no sensor – processo descrito na Equação 14 e demonstrado na Figura 80.

$$COS\ (mm) = \frac{Largura\ do\ Sensor\ (mm) * Comprimento\ Objeto\ na\ Imagem\ (pixéis)}{Largura\ da\ Imagem\ (pixéis)}$$

$$COS\ (mm) = \frac{13.2 * 545.86}{638}$$

$$COS\ (mm) = 11.29\ mm$$

Figura 80 - Demonstração do cálculo do comprimento do objeto presente na Figura 76 no sensor da máquina fotográfica utilizada

Por fim, é possível aplicara a Equação 15 para determinar a estimativa do comprimento real do sujeito fotografado (demonstrado na Figura 81).

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

$$CRO\ (m) = \frac{0.6 * 11.29}{10.2}$$

$$CRO\ (m) = 0{,}66\ m$$

Figura 81 - Demonstração do cálculo de uma estimativa do comprimento real do objeto presente na Figura 76

Concluindo e de acordo com a estimativa realizada, o comprimento da corvina presente na Figura 76 é de, aproximadamente, 0.66 metros, ou, 66 centímetros. Este valor representa um desvio percentual de cerca de 4.76% face ao comprimento real do animal.

A descoberta de um valor aproximado do comprimento do sujeito fotografado permite a utilização desse mesmo valor como *input* do modelo de regressão linear simples, através de processos semelhantes aos descritos anteriormente na secções 5.2.3.1 e 5.2.3.2. O *output* produzido pelo modelo de regressão linear encontra-se apresentado na Figura 82.

```
 1 ################################################
 2
 3 imagem com largura de 638 pixéis
 4
 5 imagem: data/custom_val/mydrive/Tese/yolov4/validation_custom/60_3.jpg capturada a
 6 uma distância de 60cm
 7
 8 comprimento em cm: 66.43365882352941cm
 9
10 estimativa de peso em g: 1930.039842071239g
```

Figura 82 - Output da estimativa de peso realizada para a corvina presente na Figura 76

A estimativa de peso resultante, aproximadamente 1930.0398 gramas, representa uma desvio percentual de 16.38% face ao peso real da corvina analisada.

5.2.4.2 Exemplo #2

Para o segundo exemplo foi utilizada a Figura 83 onde se encontra representada uma corvina com aproximadamente 2 quilogramas e 472 gramas e 65 centímetros de comprimento.

Figura 83 - Registo fotográfico utilizado como exemplo #2 para testagem do processo de estimativa de peso

A partir desta imagem foi obtida a *bounding box* presente na Figura 84, cujas coordenadas se encontram descritas no excerto do ficheiro *json* representado na Figura 85.

Figura 84 - Bounding box resultante da deteção realizada na Figura 83


```
{
 "frame_id":2,
 "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_4.jpg",
 "objects": [
  {"class_id":0, "name":"Fish",
"relative_coordinates":{"center_x":0.488616, "center_y":0.484681,
"width":0.920438, "height":0.712567},
"confidence":0.989885}
 ]
}
```


Figura 85 - Coordenadas da deteção realizada na Figura 83

A partir das coordenadas da deteção é possível realizar o mesmo procedimento descrito anteriormente, até ser obtida uma estimativa do comprimento real do objeto, em centímetros, tal como descrito em seguida, na Figura 86.

$$COI\ (pixéis) = \ Largura\ da\ Imagem\ (pixéis) * width\ (YOLOv4\ output)$$

$$COI\ (pixéis) = 636\ * 0.920438$$

$$COI\ (pixéis) = 585.50\ px$$

$$COS\ (mm) = \frac{Largura\ do\ Sensor\ (mm) * Comprimento\ Objeto\ na\ Imagem\ (pixéis)}{Largura\ da\ Imagem\ (pixéis)}$$

$$COS\ (mm) = \frac{13.2 * 585.40}{636}$$

$$COS\ (mm) = 12.15\ mm$$

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

$$CRO\ (m) = \frac{0.6 * 12.15}{10.2}$$

$$CRO\ (m) = 0{,}71\ m$$

Figura 86 - Processo de obtenção de uma estimativa do comprimento real do objeto presente na Figura 83, tendo em conta as coordenadas da *bounding box* da sua deteção

O valor obtido para o comprimento real do foi de 0.71 metros, ou, alternativamente, 71 centímetros. Este valor representa desvio percentual de, aproximadamente, 9.23%, face ao peso real da corvina fotografada. Quando utilizado como *input* do modelo de regressão linear simples, o comprimento calculado da corvina resulta num valor de peso estimado de, aproximadamente, 2 quilogramas e 144 gramas (Figura 87), o que representa um desvio percentual de cerca de 13.27%, comparativamente ao peso real da corvina.

```
1 ################################################
2
3 imagem com largura de 636 pixéis
4
5 imagem: data/custom_val/mydrive/Tese/yolov4/validation_custom/60_4.jpg
6 capturada a uma distância de 60cm
7
8 comprimento em cm: 71.46930352941176cm
9
10 estimativa de peso em g: 2144.4401771550242g
```

Figura 87 - Output da estimativa de peso realizada para a corvina presente na Figura 83

5.2.4.3 Exemplo #3

No terceiro exemplo foi utilizada a Figura 88. Nesta imagem é possível visualizar uma corvina com comprimento de, aproximadamente, 41 centímetros e com um peso de cerca de 813 gramas.

Figura 88 - Registo fotográfico utilizado como exemplo #3 para testagem do processo de estimativa de peso

A deteção realizada nesta imagem resultou na obtenção das coordenadas presentes na Figura 89 que permitiram o cálculo da estimativa do comprimento real do objeto, demonstrado na Figura 90.


```
{
 "frame_id":1,
 "filename":"data/custom_val/mydrive/Tese/yolov4/validation_custom/60_1.jpg",
 "objects": [
  {"class_id":0, "name":"Fish",
"relative_coordinates":{"center_x":0.473018, "center_y":0.504056,
"width":0.543378, "height":0.278948}, "confidence":0.996137}
 ]
}
```


Figura 89 - Coordenadas da deteção realizada na Figura 88

$$COI\ (pixéis) = \ Largura\ da\ Imagem\ (pixéis) * width\ (YOLOv4\ output)$$

$$COI\ (pixéis) = 2048\ * 0.543378$$

$$COI\ (pixéis) = 1112.84\ px$$

$$COS\ (mm) = \frac{Largura\ do\ Sensor\ (mm) * Comprimento\ Objeto\ na\ Imagem\ (pixéis)}{Largura\ da\ Imagem\ (pixéis)}$$

$$COS\ (mm) = \frac{13.2 * 1112.84}{2048}$$

$$COS\ (mm) = 7.17\ \ mm$$

$$CRO\ (m) = \frac{Distância\ ao\ objeto\ (m) * Comprimento\ do\ Objeto\ no\ Sensor\ (mm)}{Focal\ Length\ (mm)}$$

$$CRO\ (m) = \frac{0.6 * 7.17}{10.2}$$

$$CRO\ (m) = 0{,}42\ m$$

Figura 90 - Processo de obtenção de uma estimativa do comprimento real do objeto presente na Figura 88, tendo em conta as coordenadas da *bounding box* da sua deteção

O cálculo do valor do comprimento real da corvina analisada resultou no valor 0.42 metros, ou, alternativamente, 42 centímetros. Este resultado representa um desvio percentual de, sensivelmente, 2.44% comparativamente ao valor do comprimento real do sujeito estudado (41 centímetros).

A utilização do valor de comprimento calculado previamente como *input* do modelo de regressão linear simples resultou na obtenção do valor de peso estimado de, aproximadamente, 897.9012 gramas, tal como é possível verificar através do *output* do procedimento, presente na Figura 91. Este valor representa um desvio percentual de, sensivelmente, 10.44% face ao valor real do peso da corvina presente na Figura 88.

```
 1 ################################################
 2
 3 imagem com largura de 2048 pixéis
 4
 5 imagem: data/custom_val/mydrive/Tese/yolov4/validation_custom/60_1.jpg
 6 capturada a uma distância de 60cm
 7
 8 comprimento em cm: 42.19170352941177cm
 9
10 estimativa de peso em g: 897.9012298944161g
```

Figura 91 - Output da estimativa de peso realizada para a corvina presente na Figura 88

5.2.4.4 Sistematização dos Resultados Obtidos

A Tabela 32 representa uma sistematização dos resultados obtidos para os três diferentes exemplos analisados.

Tabela 32 - Sistematização dos resultados obtidos durante os testes do processo de estimativa de peso

Imagem Analisada	Comprimento Real (cm)	Peso Real (g)	Estimativa Comprimento (cm)	Estimativa Peso (g)	Desvio Comprimento (%)	Desvio Peso (%)
Figura 60	63	2066	67	1968.9905	6.30	4.70
Figura 76	63	2308	66	1930.0398	4.76	16.38
Figura 83	65	2472	71	2144.4402	9.23	13.72
Figura 88	41	813	42	897.9012	2.44	10.44

De acordo com os resultados dos procedimentos aplicados, é possível verificar que a desvio médio entre o valor do peso de cada corvina e o valor estimado pelo procedimento desenvolvido é, aproximadamente, 11.31%.

No gráfico representado na Figura 92 estão contidos dados relativos à nuvem de dispersão utilizada durante o período de teste do modelo de regressão linear, a linha de regressão resultante da aplicação do modelo e os valores reais e estimados para cada um dos exemplos presentes na Tabela 32.

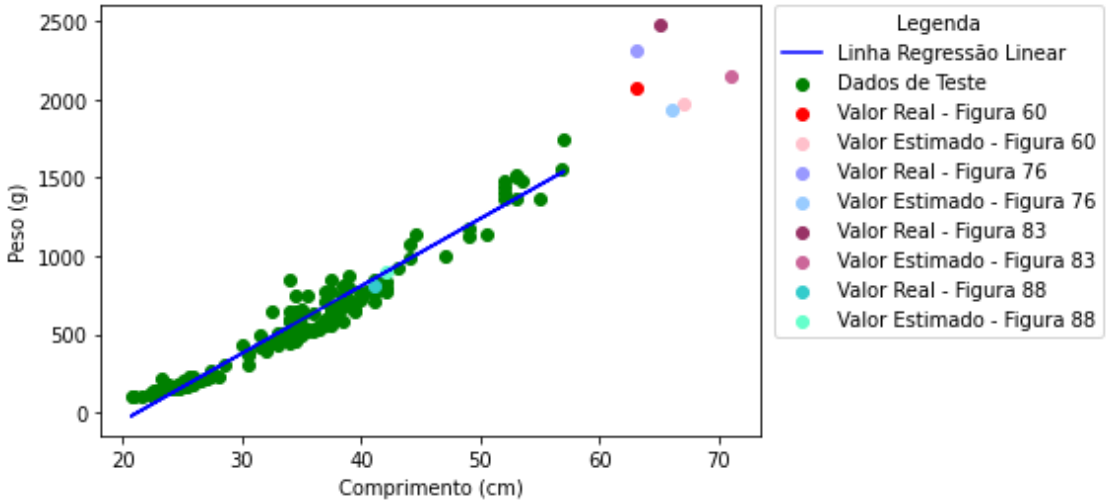

Figura 92 - Nuvem de Dispersão entre valores reais de teste, reta de Regressão, Valores Reais e Estimativas

Através da análise do gráfico é possível verificar que os valores estimados se encontram em linha com a reta de regressão. No entanto, comparando os valores reais e os respetivos valores obtidos através da aplicação do procedimento desenvolvido, é notório algum distanciamento, principalmente no que diz respeito a estimativas realizadas para valores de comprimento muito diferentes daqueles que se encontravam no conjunto de treino do modelo de regressão. Esta disparidade entre o valor real e o valor estimado poderá ser explicada pelo elevado valor de erro médio quadrático obtido na avaliação do modelo.

5.3 Identificação de Períodos de Alimentação

O procedimento implementado para identificação de períodos de alimentação em imagens com vários peixes é composto por duas fase: a **primeira fase** usufrui do **modelo *YOLOv4*** utilizado no problema anterior para deteção de grupos ou instâncias individuais de peixes em imagens. A **segunda fase** diz respeito à utilização de um ***script* de pós-processamento**, em *Python*, capaz de calcular e comparar a área da maior

bounding box detetada com a área da imagem utilizada. Esta comparação ditará o *output* final, fornecendo informação relativamente à existência de períodos de alimentação.

5.3.1 1ª Fase – Modelo YOLOv4

5.3.1.1 Treino do Modelo YOLOv4

O principal requisito exigido ao modelo *YOLOv4* utilizado para implementação da solução delineada foi a sua capacidade de detetar grupos ou instâncias individuais de peixes de várias espécies. Para isso, seria apenas necessário treinar este mesmo modelo com recurso a imagens desse tipo de ser vivo.

Uma vez que todo esse processo já foi realizado para o desenvolvimento do procedimento de obtenção de estimativas de peso, não foi necessária qualquer alteração ao processo de treino do modelo de visão por computador, pelo que, não existe nada a acrescentar na presente subsecção.

5.3.1.2 Avaliação do Modelo YOLOv4

De um modo geral, a solução implementada está baseada no cálculo da área da imagem utilizada e na comparação deste valor com a área da maior *bounding box* detetada, de forma percentual. Se esta percentagem for superior a um *threshold* estabelecido *à priori,* então é possível considerar que se encontra a decorrer um período de alimentação.

As imagens de períodos de alimentação fornecidas pela SEAentia não apresentavam a resolução e nitidez necessárias para a sua utilização (como é possível verificar através da análise da Figura 93), pelo que, todas as imagens utilizadas no período de avaliação foram recolhidas com recurso ao motor de busca *Google* e aos termos de pesquisa "*fish groups being fed*". Através dessa mesma pesquisa foram recolhidos dois tipos de imagens: imagens representativas de grupos de peixes a serem alimentados e imagens que representassem múltiplas instâncias de peixes dispersos, de modo a avaliar o procedimento tendo em conta dois cenários distintos (Figura 94).

Embora este procedimento não seja aplicável a qualquer cenário que pode ser encontrado no mundo real, o objetivo do seu desenvolvimento reside na sua futura aplicabilidade a cenários semelhantes aos que ocorrem diariamente na SEAentia. As condições de gravação e captação de imagem desta empresa permitem desconsiderar o impacto de fatores como a escala da imagem e a proximidade da câmara à superfície

da água, fatores estes que poderiam, em alguns casos, restringir a aplicabilidade do procedimento desenvolvido.

Figura 93 - Exemplo de representação de um período de alimentação fornecido pela SEAentia

Figura 94 - Imagens que simulam um período de alimentação e um período de não alimentação, respetivamente

A aplicação do modelo *YOLOv4* às imagens utilizadas resultou nas *bounding boxes* descritas nas Figura 95 e Figura 96. Tal como é possível visualizar, a fotografia que retrata um período de alimentação apresenta uma *bounding box* com uma área bastante superior a qualquer *bounding box* detetada na fotografia que não retrata um período de alimentação.

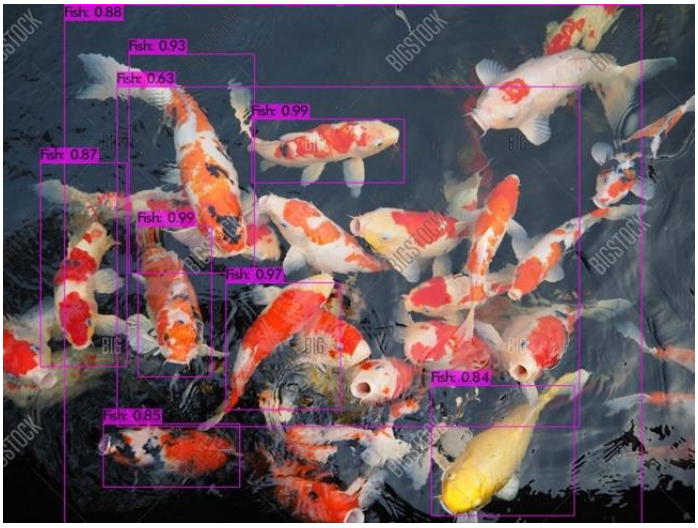

Figura 95 - Deteção realizada numa imagem que representa um período de alimentação

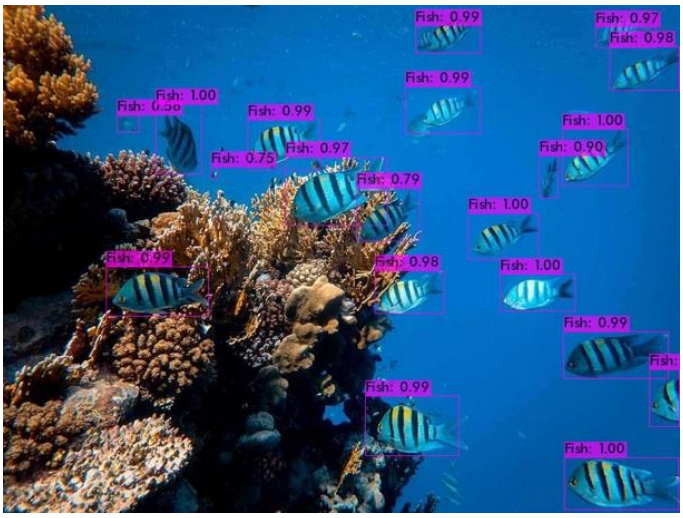

Figura 96 - Deteção realizada numa imagem que representa a inêxistencia de um período de alimentação

5.3.2 2ª Fase – Aplicação do *Script* de pós-processamento

Havendo realizado a deteção torna-se possível aceder ao ficheiro *json* produzido no fim da execução do modelo *YOLOv4*. Este ficheiro foi utilizado por um *script* em *Python* cujo comportamento se encontra descrito em seguida e, posteriormente, ilustrado na Figura 97.

- Em primeiro lugar, foi realizado o cálculo da área de cada imagem, em pixéis.
- Em seguida, para cada imagem analisada, foi realizado o cálculo da área de todas as *bounding boxes* detetadas nessa mesma imagem, após converter os seus valores de *width* e *heigth* em pixéis.

- De forma iterativa foi encontrada a área da maior *bounding box* associada a cada imagem.

- Procedeu-se ao cálculo do valor percentual da área ocupada pela maior *bounding box* na imagem original.

- O valor percentual foi comparado com o *threshold* previamente definido, com o valor de 0.55, caso o seu valor fosse superior foi considerada a existência de um período de alimentação na imagem analisada.

```
# Threshold a utilizar para comparação com o valor percentual
THRESHOLD = 0.55

for ficheiroJSON in resultJSON:
  filename = ficheiroJSON['filename']
  image = PIL.Image.open(str(filename))
  # width e height da imagem em pixéis
  width_img, height_img = image.size
  filename_dimensions[filename] = width_img, height_img
  # Cálculo da área da imagem original
  area_img = width_img * height_img

  area_maior_bb = 0

  for j in ficheiroJSON['objects']:
    width_pred = float(j['relative_coordinates']['width'])
    height_pred = float(j['relative_coordinates']['height'])

    # Conversão das dimensões da bounding box em pixéis
    width_pred_px = width_pred * width_img
    height_pred_px = height_pred * height_img

    # Cálculo da área da bounding box
    area_bb = width_pred_px * height_pred_px

    # Obtenção da área da maior bounding box
    if area_bb > area_maior_bb:
      area_maior_bb = area_bb

  # Valor percentual da área ocupada pela maior bounding box na imagem original
  percentagem_area_bb = (area_maior_bb * 100) / area_img

  print("\n#######################################################\n")
  print("Imagem " + filename + " com área " + str(area_img) + " px^2\n")
  print("A maior bounding box nesta imagem tem área de " + str(area_maior_bb)
        + " px^2\n")
  # Comparação do valor percentual com o threshold previamente definido
  if percentagem_area_bb > THRESHOLD:
    print("A área desta bounding box representa uma percentagem superior a "
        + str(THRESHOLD)
        + "% da imagem utilizada (aprox." + str(percentagem_area_bb) + "%)\n")
    print("É possível afirmar que se encontra a decorrer um período de alimentação "
        + "na imagem utilizada\n")
    print("---> ALIMENTAÇÃO OK\n\n")
  else:
    print("A área desta bounding box representa uma percentagem inferior a "
        + str(THRESHOLD)
        + "% da imagem utilizada (aprox." + str(percentagem_area_bb) + "%)\n")
    print("É possível afirmar que não se encontra a decorrer um período de "
        + "alimentação na imagem utilizada\n")
    print("---> ALIMENTAÇÃO KO\n\n")
```

Figura 97 - Implementação do *script* utilizado para deteção da existência de períodos de alimentação

Para ambas as imagens presentes na Figura 94, os *outputs* produzidos pelo *script* foram encontram-se apresentados, respetivamente, nas Figura 98 e Figura 99.

```
1 ######################################################
2
3 Imagem data/alimentacao/mydrive/Tese/yolov4/alimentacao/alimentacao_2.jpg
4 com área 1867500 px^2
5
6 A maior bounding box nesta imagem tem área de 1599727.2999210448 px^2
7
8 A área desta bounding box representa uma percentagem superior a 55% da imagem
9 utilizada (aprox.85.66143506939999%)
10
11 É possível afirmar que se encontra a decorrer um período de alimentação
12 na imagem utilizada
13
14 ---> ALIMENTAÇÃO OK
```

Figura 98 - *Output* produzido pelo *script* para a imagem presente na Figura 94 que ilustra a existência de um período de alimentação

```
1 ######################################################
2
3 Imagem data/alimentacao/mydrive/Tese/yolov4/alimentacao/alimentacao_4_not.jpg com
4 área 750000 px^2
5
6 A maior bounding box nesta imagem tem área de 15235.04424 px^2
7
8 A área desta bounding box representa uma percentagem inferior a 55% da imagem
9 utilizada (aprox.2.0313392319999997%)
10
11 É possível afirmar que não se encontra a decorrer um período de alimentação na
12 imagem utilizada
13
14 ---> ALIMENTAÇÃO KO
```

Figura 99 - Output produzido pelo script para a imagem presente na Figura 69 que ilustra a inexistência de um período de alimentação. A previsão reforça a inexistência deste período

Tal como é possível observar, os resultados obtidos encontram-se de acordo com o *output* esperado, por isso, esta técnica para deteção de períodos de alimentação parece ser aplicável ao ecossistema da empresa proponente, no entanto, será necessário um estudo mais aprofundado, por exemplo, relativamente à definição de um *threshold* adequado, para que esta solução possa a vir a ter utilidade do ponto de vista prático.

5.3.3 Aplicação do Procedimento Desenvolvido

De modo semelhante aos testes realizados para o procedimento de obtenção de estimativas de peso, foram também selecionados exemplos adicionais para testar as capacidades do processo de identificação de períodos de alimentação.

5.3.3.1 Exemplo #1

Para análise do primeiro exemplo foi utilizada a Figura 100. Nesta imagem é possível visualizar um agregado de peixes em alto mar. Embora estes animais não estejam a ser alimentados, esta imagem foi utilizada na tentativa de satisfazer a condição principal

que permite a identificação de um período de alimentação no contexto específico do problema abordado.

Tal como descrito anteriormente, a presença de um aglomerado significativo de peixes num dos tanques ao cuidado da SEAentia será um forte indicador da existência de um período de alimentação. A utilização da Figura 100 tem, por isso, como objetivo, compreender se os grupos de peixes são corretamente interpretados pelo modelo *YOLOv4* e pelo *script* de pós-processamento.

Figura 100 - Registo fotográfico utilizado como exemplo #1 para testagem do processo de identificação de períodos de alimentação

Na imagem utilizada foram detetadas duas *bounding* boxes de dimensões substancialmente diferentes. A *bounding box* de área superior, cerca de 257959.8397 pixéis quadrados foi selecionada e, quando comparada com a área total da imagem original (360000 pixéis quadrados), permitiu inferir a existência de um período de alimentação, uma vez que a área da deteção representava cerca de 71.66% da imagem original - valor percentual superior ao *threshold* de 55% definido inicialmente (Figura 101).

```
1 ######################################################
2
3 Imagem data/alimentacao/mydrive/Tese/yolov4/alimentacao/alimentacao_3.jpg
4 com área 360000 px^2
5
6 A maior bounding box nesta imagem tem área de 257959.83968496 px^2
7
8 A área desta bounding box representa uma percentagem superior a 55% da imagem
9 utilizada (aprox.71.6555110236%)
10
11 É possível afirmar que se encontra a decorrer um período de alimentação na
12 imagem utilizada
13
14 ---> ALIMENTAÇÃO OK
```

Figura 101 - Output produzido pelo script para a imagem presente na Figura 100 que ilustra a existência de um período de alimentação

5.3.3.2 Exemplo #2

No segundo exemplo foi aplicada uma imagem, capturada num ambiente aquático, com várias instâncias de peixes não agrupados (Figura 102). As diferentes *bounding boxes* identificadas apresentam áreas semelhantes, embora sejam todas manifestamente reduzidas, principalmente quando comparadas à área total da imagem, que apresenta o valor de 2457000 pixéis quadrados.

A área da maior instância identificada (aproximadamente 17784.37 pixéis quadrados) representa cerca de 0.72% da totalidade da figura original 89. Este valor permite supor a inexistência de um período de alimentação na Figura 102, , devido à sua inferioridade ao *threshold* estabelecido inicialmente.

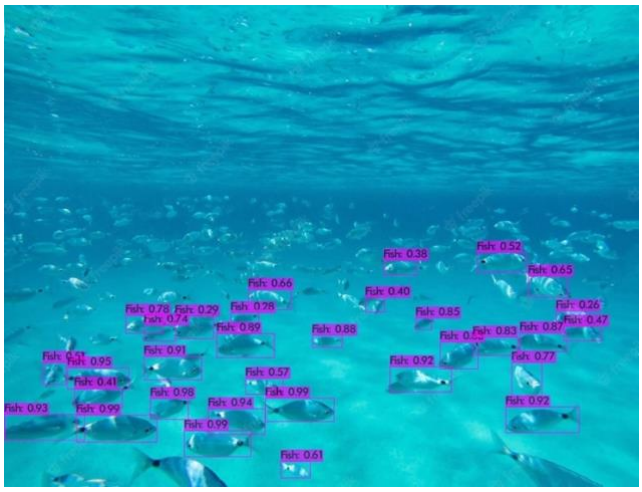

Figura 102 - Registo fotográfico utilizado como exemplo #2 para testagem do processo de identificação de períodos de alimentação

```
1 ######################################################
2
3 Imagem data/alimentacao/mydrive/Tese/yolov4/alimentacao/alimentacao_7.jpg
4 com área 2457000 px^2
5
6 A maior bounding box nesta imagem tem área de 17784.3743532 px^2
7
8 A área desta bounding box representa uma percentagem inferior a 55% da imagem
9 utilizada (aprox.0.7238247600000001%)
10
11 É possível afirmar que não se encontra a decorrer um período de alimentação na imagem
12 utilizada
13
14 ---> ALIMENTAÇÃO KO
```

Figura 103 - Output produzido pelo script para a imagem presente na Figura 102 que ilustra a existência de um período de alimentação

5.3.3.3 Exemplo #3

O terceiro exemplo foi realizado com recurso à Figura 104. Nesta imagem é possível visualizar, novamente, um conjunto de peixes. À semelhança do comportamento esperado no exemplo #1, no exemplo atual o objetivo seria que o procedimento identificasse o conjunto de peixes e o enquadrasse numa única *bounding box*, de modo que estivessem reunidas as condições necessárias para inferir a existência de um período de alimentação.

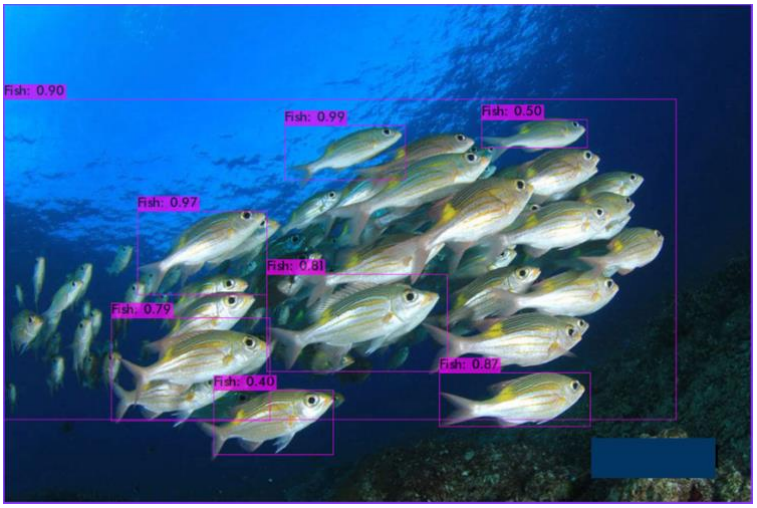

Figura 104 - Registo fotográfico utilizado como exemplo #3 para testagem do processo de identificação de períodos de alimentação

Tal como é possível verificar através da análise da Figura 104, o grupo de seres vivos foi corretamente identificado, sendo que, de acordo com o *output* do procedimento (Figura 105), a área da *bounding box* correspondente à sua deteção representava cerca de 58.05% da área total da imagem. Uma vez que este valor percentual é superior ao *threshold* definido *à priori*, é possível afirmar que, no contexto do problema atual, a imagem analisada representa um período de alimentação.

```
 1 ########################################################
 2
 3 Imagem data/alimentacao/mydrive/Tese/yolov4/alimentacao/alimentacao_1.jpg com
 4 área 912600 px^2
 5
 6 A maior bounding box nesta imagem tem área de 529780.1330231981 px^2
 7
 8 A área desta bounding box representa uma percentagem superior a 55% da imagem
 9  utilizada (aprox.58.05173493569999%)
10
11 É possível afirmar que se encontra a decorrer um período de alimentação na
12  imagem utilizada
13
14 ---> ALIMENTAÇÃO OK
```

Figura 105 - Output produzido pelo script para a imagem presente na Figura 104 que ilustra a existência de um período de alimentação

5.3.3.4 Sistematização dos Resultados Obtidos

No seguimento dos resultados obtidos para os exemplos utilizados na avaliação do procedimento de identificação de períodos de alimentação, foi construída a Tabela 33, que agrega as informações mais relevantes de cada teste.

Tabela 33 - Sistematização dos resultados obtidos durante os testes do processo de identificação de períodos de alimentação

Figura	Área da Figura (px^2)	Área da maior *bounding box* (px^2)	Relação Percentual entre as áreas (%)	Resultado Obtido
Figura 95	1867500	1599727.30	85.66	Existência de Período de Alimentação
Figura 96	750000	15235.04	2.03	Inexistência de Período de Alimentação
Figura 100	360000	257959.84	71.66	Existência de Período de Alimentação
Figura 102	2457000	17784.37	0.72	Inexistência de Período de Alimentação
Figura 104	912600	529700.13	58.05	Existência de Período de Alimentação

De um modo geral e tendo em conta as condicionantes existentes em cada teste, é possível concluir que o procedimento implementado se comporta de forma linear, isto é, em imagens que apresentam grupos de peixes, esses grupos são corretamente identificados e a comparação da sua área com a área total da figura resulta na identificação de um período de alimentação. Por outro lado, em imagens onde estejam contidas várias instâncias de peixes isolados, o modelo de visão por computador é capaz de distinguir cada instância e, por consequência, ao comparar a área da maior instância detetada com a área total da figura, concluí a inexistência de um período de alimentação.

Se, no futuro, existir a possibilidade de obter imagens com um nível de resolução aceitável dos tanques monitorizados pela SEAentia, será necessária uma revisão do valor de *threshold* a aplicar, tendo em conta a posição das câmaras utilizadas e a sua

distância à superfície da água, para que os resultados obtidos sejam coerentes com o comportamento das corvinas.

5.4 Sinopse do Capítulo

O presente capítulo é iniciado com um esclarecimento relativamente aos objetivos a atingir através do trabalho desenvolvido.

Segue-se a secção que introduz as configurações e parametrizações realizadas ao nível do ambiente de desenvolvimento e modelos utilizados. São também referenciadas ferramentas adicionais utilizadas.

A secção seguinte aborda a temática da realização de estimativas de peso. O procedimento aplicado é, brevemente, descrito e é fornecida informação sobre o processo de treino e avaliação do modelo *YOLOv4*. São demonstrados os comandos utilizados e o formato do output produzido pelo modelo de visão por computador. A obtenção deste mesmo output leva à descrição do processo de cálculo do comprimento real, em centímetros, de um determinado objeto. Este processo foi integrado na pipeline desenvolvida e permite a posterior aplicação do modelo de regressão linear simples que, após ser treinado, é capaz de produzir uma estimativa de peso, em gramas, para um determinado comprimento, em centímetros.

Esta mesma secção é finalizada com a demonstração da aplicação do procedimento desenvolvido a algumas imagens, a título de exemplo. Os resultados obtidos são, em seguida, agregados e analisados, tendo em conta o objetivo inicial.
Por fim, a última secção é dedicada à identificação de períodos de alimentação, onde é realçada a utilização do mesmo modelo *YOLOv4* configurado anteriormente e descrito em maior detalhe o script de pós processamento responsável pelo cálculo da área das imagens e *bounding boxes* resultantes das deteções. É, igualmente, demonstrada a aplicação do procedimento a outros exemplos e é realizada uma sistematização dos resultados obtidos, de modo que possam ser retiradas conclusões sobre o trabalho desenvolvido.

6 Conclusão

No presente capítulo serão descritas, em primeiro lugar, as contribuições resultantes do trabalho desenvolvido, ao nível do negócio e ao nível académico. Em seguida, são mencionadas as principais limitações e condicionantes enfrentadas durante todo o processo de desenvolvimento e de escrita e, por fim, é apresentada uma subseção com informação relativa a possíveis melhorias que possam ser efetuadas com recurso a trabalho futuro.

6.1 Contribuições

Nesta subsecção serão destacadas as contribuições facultadas pelo trabalho desenvolvido durante a elaboração do presente documento. Estas contribuições serão divididas em dois grupos: contribuições para o negócio da empresa proponente e contribuições académicas.

Ao nível do negócio, são propostas duas soluções para dois problemas recorrentes nos procedimentos realizados pelos colaboradores da SEAentia. A solução desenvolvida para a realização de estimativas de peso, ainda que esteja desenhada para aplicação individual, ou seja, aplicação a instâncias singulares de corvinas, não deixa de ser um contributo valioso que pode vir a simplificar os processos de pesagens realizados mensalmente. A aplicação do procedimento desenvolvido, num ambiente real, evitará a utilização de réguas ou ferramentas de medição e tornará o processo de medição e pesagem significativamente mais rápido e mais seguro, desde que sejam conhecidas as especificações da câmara fotográfica utilizada e a distância entre essa mesma câmara e a corvina fotografada.

Já a solução implementada para a identificação de períodos de alimentação também poderá ser bastante útil, principalmente se forem adquiridos dispositivos capazes de atingir melhores resoluções durante a gravação e monitorização da atividades nos tanques e se for realizado um estudo relativamente ao *threshold* ideal a aplicar em diversas situações.

O trabalho realizado representa também uma contribuição académica ao nível da aplicação de técnicas de visão por computador e de análise das propriedades de diferentes fotografias no que diz respeito à determinação do peso de corvinas e ao estudo do seu comportamento em períodos de alimentação.

De salientar que os avanços nesta área de estudo, bastante específica, são escassos e que as condicionantes encontradas, num contexto praticamente desconhecido, não impediram a implementação de duas soluções funcionais e que cumprem os objetivos traçados.

Não obstante, estas soluções não são perfeitas e apresentam espaço para melhorias futuras que as poderão utilizar como base para desenvolvimento de procedimentos mais complexos e de maior utilidade prática.

6.2 Limitações

A principal condicionante enfrentada durante o desenvolvimento da presente dissertação residiu nas limitações temporais impostas pela mudança do tema da mesma, numa fase já tardia do desenvolvimento. Esta alteração, aliada à inexperiência na área de visão por computador, resultou em obstáculos que necessitaram de bastante esforço para serem ultrapassados.

Além disso, existiram outras condicionantes enfrentadas, nomeadamente, durante o desenvolvimento, no que diz respeito à falta de imagens de qualidade que pudessem ser utilizadas durante o processo de treino e teste do modelo de visão por computador. Esta limitação levou à utilização de imagens genéricas, obtidas a partir de repositórios *online* ou a partir de pesquisas em motores de busca.

Tal como mencionado no capítulo quinto, os dados relativos às medições das corvinas, utilizados para o treino do modelo de regressão também não apresentavam a exatidão desejada, o que limitou as capacidades das previsões desse mesmo modelo.

Ao nível do processo de treino do modelo *YOLOv4*, existiu também uma limitação relacionada com a falta de recursos locais para utilização deste algoritmo. Estas condicionantes levaram a utilização de um ambiente virtual que implicou consequências, especialmente na velocidade de execução do processo de treino.

6.3 Trabalho Futuro

O trabalho futuro aplicado à temática estudada durante esta tese terá em vista a melhoria dos procedimentos elaborados. Idealmente, o modelo *YOLOv4* seria adaptado para ser capaz de reconhecer e identificar peixes em vídeos e não só imagens, seria também relevante solicitar e obter dados de melhor qualidade que permitissem a

construção de soluções mais robustas e com uma maior aplicabilidade a diferentes problemas.

No que diz respeito à solução implementada para realização das estimativas de peso, seria importante a obtenção de maior número de imagens e informações para treino de ambos os modelos. Deste modo, tornar possível utilizar um modelo *YOLOv4* pré-treinado com imagens de peixes de várias espécies que seria posteriormente orientado para a deteção específica de corvinas, através da utilização de uma quantidade moderada de imagens desse tipo de animal. A finalização do processo de treino com recurso a estas imagens em particular tornaria o algoritmo mais eficaz, quando confrontado com situações reais.

De igual forma, com recurso a um conjunto de dados alargado de comprimento e peso de várias corvinas, seria possível construir e treinar um modelo de regressão de forma mais completa, originando estimativas muito mais próximas da realidade.

Adicionalmente, seria vantajoso estudar o desempenho de diferentes modelos de visão por computador no desempenho das mesmas tarefas de modo a obter o máximo de proveito das capacidades deste tipo de algoritmo.

Por fim, o estudo de uma solução que fosse independente de fatores como a distância da câmara utilizada ao sujeito detetado, facilitaria a implementação de todos os restantes procedimentos. O estudo das corvinas presentes em tanques de grandes dimensões não garante que a distância de cada peixe à câmara que vigia o seu habitat seja linear, e, por esta razão, a utilização de soluções dependentes desta mesma distância acaba por restringir a aplicabilidade dos procedimentos desenvolvidos.

7 Referências

Abioye, S.O., Oyedele, L.O., Akanbi, L., Ajayi, A., Davila Delgado, J.M., Bilal, M., Akinade, O.O., Ahmed, A., 2021. Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges. Journal of Building Engineering 44, 103299. https://doi.org/10.1016/J.JOBE.2021.103299

Adrion, W.R., Branstad, M.A., Cherniavsky, J.C., 1982. Validation, Verification, and Testing of Computer Software.

Al-Omary, A.Y., Jamil, M.S., 2006. A new approach of clustering based machine-learning algorithm. Knowledge-Based Systems 19, 248–258. https://doi.org/10.1016/J.KNOSYS.2005.10.011

Baehr, M., 2004. Faculty Development Series Evaluation Methodology.

Bisong, E., 2019. Google Colaboratory. Building Machine Learning and Deep Learning Models on Google Cloud Platform 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7

Bochkovskiy, A., 2020. Darknet [WWW Document]. URL https://github.com/AlexeyAB/darknet#readme (accessed 5.9.22).

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020a. YOLOv4: Optimal Speed and Accuracy of Object Detection.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020b. YOLOv4: Optimal Speed and Accuracy of Object Detection.

Borges Oliveira, D.A., Ribeiro Pereira, L.G., Bresolin, T., Pontes Ferreira, R.E., Reboucas Dorea, J.R., 2021. A review of deep learning algorithms for computer vision systems in livestock. Livestock Science 253, 104700. https://doi.org/10.1016/J.LIVSCI.2021.104700

Boureau, Y.-L., Ponce, J., Fr, J.P., Lecun, Y., 2010. A Theoretical Analysis of Feature Pooling in Visual Recognition.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Brooks, R.A., Binford, T.O., 1981. Geometric modeling in vision for manufacturing, in: Techniques and Applications of Image Understanding. pp. 141–159.

Brownlee, J., 2019. A Gentle Introduction to Object Recognition With Deep Learning [WWW Document]. URL https://machinelearningmastery.com/object-recognition-with-deep-learning/ (accessed 1.18.22).

Chen, Y., Goorden, M.C., Beekman, F.J., Shang, L., Sui, L., Wang, S., Du, J., 2018. Understanding of Object Detection Based on CNN Family and YOLO You may also like Convolutional neural network based attenuation correction for 123 I-FP-CIT SPECT with focused striatum imaging Analysis of Object Detection Performance Based on Faster R-CNN Wenze Li-Sentiment analysis of film reviews based on CNN-BLSTM-Attention Understanding of Object Detection Based on CNN Family and YOLO. J. Phys 12029. https://doi.org/10.1088/1742-6596/1004/1/012029

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E., 2014. cuDNN: Efficient Primitives for Deep Learning.

Chin, C.L., Lin, B.J., Wu, G.R., Weng, T.C., Yang, C.S., Su, R.C., Pan, Y.J., 2017. An automated early ischemic stroke detection system using CNN deep learning algorithm. Proceedings - 2017 IEEE 8th International Conference on Awareness Science and Technology, iCAST 2017 2018-January, 368–372. https://doi.org/10.1109/ICAWST.2017.8256481

Computer Vision Market Size & Share Report, 2021-2028 [WWW Document], 2021. URL https://www.grandviewresearch.com/industry-analysis/computer-vision-market (accessed 1.7.22).

Copeland, B.J., 2014. artificial intelligence | Definition, Examples, Types, Applications, Companies, & Facts | Britannica [WWW Document]. URL https://www.britannica.com/technology/artificial-intelligence (accessed 2.10.22).

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, Li Fei-Fei, 2010. ImageNet: A large-scale hierarchical image database 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dilmegani, C., 2019. Machine Learning Accuracy: True vs. False Positive/Negative [WWW Document]. URL https://research.aimultiple.com/machine-learning-accuracy/ (accessed 5.26.22).

Dohmen, R., Catal, C., Liu, Q., 2022. Computer vision-based weight estimation of livestock: a systematic literature review. New Zealand Journal of Agricultural Research 65, 227–247. https://doi.org/10.1080/00288233.2021.1876107

DOW, M.M., BURTON, M.L., WHITE, D.R., REITZ, K.P., 1984. Galton's Problem as network autocorrelation. American Ethnologist 11, 754–770. https://doi.org/10.1525/AE.1984.11.4.02A00080

Du, J., 2018. Understanding of Object Detection Based on CNN Family and YOLO. Journal of Physics: Conference Series 1004, 012029. https://doi.org/10.1088/1742-6596/1004/1/012029

Ebrahimpour, M., 2018. Introduction to Artificial Intelligence: An Introduction to Computer Vision [WWW Document]. URL (accessed 1.6.22).

E.H. Allison, 2011. Aquaculture, Fisheries, Poverty and Food Security 62.

Everingham, M., Eslami, S.M.A., van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2015. The Pascal Visual Object Classes Challenge: A Retrospective. International Journal of Computer Vision 111, 98–136. https://doi.org/10.1007/S11263-014-0733-5/FIGURES/27

Everingham, M., Zisserman, A., Williams, C.K.I., Gool, L. van, Allan, M., Bishop, C.M., Chapelle, O., Dalal, N., Deselaers, T., Dorkó, G., Duffner, S., Eichhorn, J., Farquhar, J.D.R., Fritz, M., Garcia, C., Griffiths, T., Jurie, F., Keysers, D., Koskela, M., Laaksonen, J., Larlus, D., Leibe, B., Meng, H., Ney, H., Schiele, B., Schmid, C., Seemann, E., Shawe-Taylor, J., Storkey, A., Szedmak, S., Triggs, B., Ulusoy, I., Viitaniemi, V., Zhang, J., 2005. The 2005 PASCAL Visual Object Classes Challenge.

FAO, 2014. The State of World Fisheries and Aquaculture.

Fassold, H., 2016. Computer vision on the GPU-Tools, algorithms and frameworks. INES 2016 - 20th Jubilee IEEE International Conference on Intelligent Engineering Systems, Proceedings 245–250. https://doi.org/10.1109/INES.2016.7555129

Fei-Fei, L., 2016. CS231n Convolutional Neural Networks for Visual Recognition [WWW Document]. URL https://cs231n.github.io/convolutional-networks/#conv (accessed 1.21.22).

Forson, E., 2017. Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning | by Eddie Forson | Towards Data Science [WWW Document]. URL https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab (accessed 1.26.22).

Freund, Y., Schapire, R.E., 1999. A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence 14, 771–780.

Fulton, W., 2015. Calculator to compute the Distance or Size of an Object in a photo Image [WWW Document]. URL https://www.scantips.com/lights/subjectdistance.html (accessed 5.9.22).

Fürnkranz, J., Chan, P.K., Craw, S., Sammut, C., Uther, W., Ratnaparkhi, A., Jin, X., Han, J., Yang, Y., Morik, K., Dorigo, M., Birattari, M., Stützle, T., Brazdil, P., Vilalta, R., Giraud-Carrier, C., Soares, C., Rissanen, J., Baxter, R.A., Bruha, I., Baxter, R.A., Webb, G.I., Torgo, L., Banerjee, A., Shan, H., Ray, S., Tadepalli, P., Shoham, Y., Powers, R., Shoham, Y., Powers, R., Webb, G.I., Ray, S., Scott, S., Blockeel, H., De Raedt, L., 2011a. Mean Absolute Error. Encyclopedia of Machine Learning 652–652. https://doi.org/10.1007/978-0-387-30164-8_525

Fürnkranz, J., Chan, P.K., Craw, S., Sammut, C., Uther, W., Ratnaparkhi, A., Jin, X., Han, J., Yang, Y., Morik, K., Dorigo, M., Birattari, M., Stützle, T., Brazdil, P., Vilalta, R., Giraud-Carrier, C., Soares, C., Rissanen, J., Baxter, R.A., Bruha, I., Baxter, R.A., Webb, G.I., Torgo, L., Banerjee, A., Shan, H., Ray, S., Tadepalli, P., Shoham, Y., Powers, R., Shoham, Y., Powers, R., Webb, G.I., Ray, S., Scott, S., Blockeel, H., De Raedt, L., 2011b. Mean Squared Error. Encyclopedia of Machine Learning 653–653. https://doi.org/10.1007/978-0-387-30164-8_528

Gad, A.F., 2020. Accuracy, Precision, and Recall in Deep Learning [WWW Document]. URL https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/ (accessed 5.26.22).

Gan, G., 2013. Application of data clustering and machine learning in variable annuity valuation. Insurance: Mathematics and Economics 53, 795–801. https://doi.org/10.1016/J.INSMATHECO.2013.09.021

Garbade, M., 2018. Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences | by Dr. Michael J. Garbade | Towards Data Science [WWW Document]. URL https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb (accessed 3.27.22).

Gardner, M.W., Dorling, S.R., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment 32, 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Girshick, R., 2015. Fast R-CNN.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014a. Rich feature hierarchies for accurate object detection and semantic segmentation.

Girshick, R., Donahue, J., Darrell, T., Malik, J., Berkeley, U.C., Malik, J., 2014b. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1, 5000. https://doi.org/10.1109/CVPR.2014.81

Glen, S., 2012. Coefficient of Determination (R Squared): Definition, Calculation [WWW Document]. URL https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared/ (accessed 5.29.22).

Google Colab [WWW Document], 2017. URL https://research.google.com/colaboratory/faq.html (accessed 1.5.22).

Grieve, P., 2020. Deep learning vs. machine learning: What's the difference? [WWW Document]. URL https://www.zendesk.com/blog/machine-learning-and-deep-learning/ (accessed 2.10.22).

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S., 2016. Deep learning for visual understanding: A review. Neurocomputing 187, 27–48. https://doi.org/10.1016/J.NEUCOM.2015.09.116

Halvorson, H.O., Smolowitz, R., 2009. Aquaculture. Encyclopedia of Microbiology 17–22. https://doi.org/10.1016/B978-012373944-5.00116-4

Harris, C.R., Millman, K.J., der Walt, S.J. van, Gommers, R., Virtanen, P., David Cournapeau, Wieser, E., Taylor, J., Sebastian Berg, Smith, N.J., Kern, R., Hoyer, M.P. and S., van Kerkwijk, M.H., Matthew Brett, Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Pierre Gérard-Marchant, Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern

Analysis and Machine Intelligence 37, 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

Heinrichs, A.J., Losinger, W.C., 1998. Growth of Holstein dairy heifers in the United States. Journal of Animal Science 76, 1254–1260. https://doi.org/10.2527/1998.7651254X

Heinrichs, A.J., Rogers, G.W., Cooper, J.B., 1992. Predicting Body Weight and Wither Height in Holstein Heifers Using Body Measurements. Journal of Dairy Science 75, 3576–3581. https://doi.org/10.3168/JDS.S0022-0302(92)78134-X

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 82–97. https://doi.org/10.1109/MSP.2012.2205597

Huang, C., Cai, H., Xu, L., Xu, B., Gu, Y., Jiang, L., 2019. Data-driven ontology generation and evolution towards intelligent service in manufacturing systems. Future Generation Computer Systems 101, 197–207. https://doi.org/10.1016/J.FUTURE.2019.05.075

Huang, R., Pedoeem, J., Chen, C., 2019. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018 2503–2510. https://doi.org/10.1109/BIGDATA.2018.8621865

Hui, J., 2018. mAP (mean Average Precision) for Object Detection [WWW Document]. URL https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (accessed 1.28.22).

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Computing in Science & Engineering 9, 90–95. https://doi.org/10.1109/MCSE.2007.55

Ian Goodfellow, Yoshua Bengio, and A.C., 2017. Deep lLearning. Genetic Programming and Evolvable Machines 19, 305–307.

Jahan, A., Edwards, K.L., Bahraminasab, M., 2016. Multi-criteria decision-making for materials selection. Multi-criteria Decision Analysis for Supporting the Selection of Engineering Materials in Product Design 63–80. https://doi.org/10.1016/B978-0-08-100536-1.00004-7

Jie Tan, R., 2019. Breaking Down Mean Average Precision (mAP) [WWW Document]. URL https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (accessed 1.28.22).

Jones, M., Viola, P., 2001. Robust Real-time Face Detection, in: Proceedings Eighth IEEE International Conference on Computer Vision. IEEE Computer Society, Los Alamitos, CA, USA, p. 747. https://doi.org/10.1109/ICCV.2001.937709

Kenton, W., 2021. Value Definition [WWW Document]. URL https://www.investopedia.com/terms/v/value.asp (accessed 1.5.22).

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012a. ImageNet Classification with Deep Convolutional Neural Networks.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012b. ImageNet Classification with Deep Convolutional Neural Networks.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V., 2018. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. International Journal of Computer Vision 128, 1956–1981. https://doi.org/10.1007/s11263-020-01316-z

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2323. https://doi.org/10.1109/5.726791

Leonardi, R., Giudice, A. lo, Isola, G., Spampinato, C., 2021a. Deep learning and computer vision: Two promising pillars, powering the future in orthodontics. Seminars in Orthodontics 27, 62–68. https://doi.org/10.1053/J.SODO.2021.05.002

Leonardi, R., Giudice, A. lo, Isola, G., Spampinato, C., 2021b. Deep learning and computer vision: Two promising pillars, powering the future in orthodontics. Seminars in Orthodontics 27, 62–68. https://doi.org/10.1053/J.SODO.2021.05.002

Li, M., Zhang, Z., Lei, L., Wang, X., Guo, X., 2020. Agricultural Greenhouses Detection in High-Resolution Satellite Images Based on Convolutional Neural Networks: Comparison of Faster R-CNN, YOLO v3 and SSD. https://doi.org/10.3390/s20174938

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D.D., Chen, M., 2014. Medical image classification with convolutional neural network. 2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014 844–848. https://doi.org/10.1109/ICARCV.2014.7064414

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dolí, P., 2015. Microsoft COCO: Common Objects in Context.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9905 LNCS, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Lowe, D., 2004. The SIFT (Scale Invariant Feature Transform) Detector and Descriptor.

Lowe, D.G., 1999. Object recognition from local scale-invariant features. Proceedings of the IEEE International Conference on Computer Vision 2, 1150–1157. https://doi.org/10.1109/ICCV.1999.790410

Marr, D., 1982. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Henry Holt and Co., Inc., New York, NY, USA.

Maulud, D.H., Mohsin Abdulazeez, A., 2020. A Review on Linear Regression Comprehensive in Machine Learning. Journal of Applied Science and Technology Trends 01, 140–147. https://doi.org/10.38094/jastt1457

McCarthy, J., Minsky, M., Rochester, N., Shannon, C.L., 1956. The dartmouth summer research project on artificial intelligence. Artificial intelligence: past, present, and future.

Mollah, M.B.R., Hasan, M.A., Salam, M.A., Ali, M.A., 2010. Digital image analysis to estimate the live weight of broiler. Computers and Electronics in Agriculture 72, 48–52. https://doi.org/10.1016/J.COMPAG.2010.02.002

Morera, Á., Sánchez, Á., Moreno, A.B., Sappa, Á.D., Vélez, J.F., 2020. SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities. Sensors 2020, Vol. 20, Page 4587 20, 4587. https://doi.org/10.3390/S20164587

Neubeck, A., van Gool, L., 2006. Efficient non-maximum suppression. Proceedings - International Conference on Pattern Recognition 3, 850–855. https://doi.org/10.1109/ICPR.2006.479

Noë, A., 2002. Vision and Mind: Selected Readings in the Philosophy of Perception - Google Books [WWW Document]. URL https://books.google.pt/books?hl=en&lr=&id=r70IEreOL-wC&oi=fnd&pg=PA229&dq=david+marr+vision&ots=wi-ILXZLy9&sig=yNF5vyv_LlD1J6vXpKS1vnkRK-E&redir_esc=y#v=snippet&q=D%20sketch&f=false (accessed 12.31.21).

Odemakinde, E., 2021. Mask R-CNN: A Beginner's Guide - viso.ai [WWW Document]. URL https://viso.ai/deep-learning/mask-r-cnn/ (accessed 1.10.22).

Oquab, M., Bottou, L., Laptev, I., Sivic, J., 2014. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks.

Ottinger, M., Clauss, K., Kuenzer, C., 2016. Aquaculture: Relevance, distribution, impacts and spatial assessments - A review. Ocean and Coastal Management 119, 244–266. https://doi.org/10.1016/J.OCECOAMAN.2015.10.015

Ouyang, W., Luo, P., Zeng, X., Qiu, S., Tian, Y., Li, H., Yang, S., Wang, Z., Xiong, Y., Qian, C., Zhu, Z., Wang, R., Loy, C.-C., Wang, X., Tang, X., 2014. DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection.

Papert, S., 2004. The Summer Vision Project.

Papert, S.A., 1966. The Summer Vision Project.

Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot andÉdouardand, M., Duchesnay, andÉdouard, Duchesnay EDOUARDDUCHESNAY, Fré., 2011. Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. Journal of Machine Learning Research 12, 2825–2830.

Rathore, S.S., Kumar, S., 2019. A study on software fault prediction techniques. Artificial Intelligence Review 51, 255–327. https://doi.org/10.1007/S10462-017-9563-5/FIGURES/8

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2015. You Only Look Once: Unified, Real-Time Object Detection.

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement.

Redmon, J., Farhadi, A., 2016. YOLO9000: Better, Faster, Stronger.

Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

Ribeiro, M.C. de C.R., Alves, A. da S., 2016. Aplicação do método Analytic Hierarchy Process (AHP) com a mensuração absoluta num problema de seleção qualitativa. Sistemas & Gestão 11, 270–281. https://doi.org/10.20985/1980-5160.2016.V11N3.988

Rudd, A., 2019. What is the evaluation of methodology? [WWW Document]. URL https://www.nbccomedyplayground.com/what-is-the-evaluation-of-methodology/ (accessed 5.25.22).

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 1986 323:6088 323, 533–536. https://doi.org/10.1038/323533a0

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115, 211–252. https://doi.org/10.1007/S11263-015-0816-Y/FIGURES/16

Sasaki, Y., 2007. The truth of the F-measure [WWW Document]. URL (accessed 5.26.22).

Scherer, D., Müller, A., Behnke, S., 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6354 LNCS, 92–101. https://doi.org/10.1007/978-3-642-15825-4_10

SEAentia, 2017. About us – SEAentia [WWW Document]. URL https://www.seaentia.pt/about-us/ (accessed 4.18.22).

Segerkvist, K.A., Höglund, J., Österlund, H., Wik, C., Högberg, N., Hessle, A., 2020. Automatic weighing as an animal health monitoring tool on pasture. Livestock Science 240, 104157. https://doi.org/10.1016/J.LIVSCI.2020.104157

Shetty, S., 2016. Application of Convolutional Neural Network for Image Classification on Pascal VOC Challenge 2012 dataset.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905. https://doi.org/10.1109/34.868688

Solawetz, J., 2020. YOLOv4 - An explanation of how it works [WWW Document]. URL https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/ (accessed 4.9.22).

Souza Marins, C., de Oliveira Souza, D., da Silva Barros, M., 2009. O USO DO MÉTODO DE ANÁLISE HIERÁRQUICA (AHP) NA TOMADA DE DECISÕES GERENCIAIS – UM ESTUDO DE CASO.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. IEEE Transactions on Neural Networks 9, 1054–1054. https://doi.org/10.1109/TNN.1998.712192

Świeżewski, J., 2020. YOLO Algorithm and YOLO Object Detection - Appsilon | Enterprise R Shiny Dashboards [WWW Document]. URL https://appsilon.com/object-detection-yolo-algorithm/ (accessed 1.17.22).

Troell, M., Joyce, A., Chopin, T., Neori, A., Buschmann, A.H., Fang, J.G., 2009. Ecological engineering in aquaculture - Potential for integrated multi-trophic aquaculture (IMTA) in marine offshore systems. Aquaculture 297, 1–9. https://doi.org/10.1016/J.AQUACULTURE.2009.09.010

Tsang, S.-H., 2018. Review: SSD — Single Shot Detector (Object Detection) | by Sik-Ho Tsang | Towards Data Science [WWW Document]. URL https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11 (accessed 1.26.22).

Turing, A., 1950. The Turing Test: Verbal Behavior as the Hallmark of Intelligence - Google Books [WWW Document]. URL https://books.google.pt/books?hl=en&lr=&id=CEMYUU_HFMAC&oi=fnd&pg=PA67&dq=Turing,+A.+M.,+1950,+Computing+machinery+and+intelligence&ots=dQhiPW1-bC&sig=9zDz24EeWAixU89SgLLl92qt07E&redir_esc=y#v=onepage&q=Turing%2C%20A.%20M.%2C%201950%2C%20Computing%20machinery%20and%20intelligence&f=false (accessed 3.27.22).

Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M., 2013. Selective search for object recognition. International Journal of Computer Vision 104, 154–171. https://doi.org/10.1007/S11263-013-0620-5/FIGURES/9

van Rossum, G., Drake Jr, F.L., 1995. Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

Vingelmann, P., NVIDIA, Fitzek, F.H.P., 2020. CUDA, release: 10.2.89.

Viola, P., Jones, M.J., 2004. Robust Real-Time Face Detection. International Journal of Computer Vision 57, 137–154.

Vittorio, A., 2018. Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset. GitHub repository.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., 2018. Deep Learning for Computer Vision: A Brief Review. https://doi.org/10.1155/2018/7068349

Widianta, M.M.D., Rizaldi, T., Setyohadi, D.P.S., Riskiawan, H.Y., 2018. Comparison of Multi-Criteria Decision Support Methods (AHP, TOPSIS, SAW & PROMENTHEE) for Employee Placement. Journal of Physics: Conference Series 953. https://doi.org/10.1088/1742-6596/953/1/012116

Yapp, C., 2011. A CMake-Based Cross Platform Build System for Tcl/Tk.

Yoo, H.-J., 2015. Deep Convolution Neural Networks in Computer Vision: a Review. IEIE Transactions on Smart Processing and Computing 4. https://doi.org/10.5573/IEIESPC.2015.4.1.035

Yoshua, B., 2009. Learning Deep Architectures for AI - [WWW Document]. URL https://books.google.pt/books?hl=en&lr=&id=cq5ewg7FniMC&oi=fnd&pg=PA1&ots=Kpi5JXskIA&sig=zm76nPcp_bl__LV4F0nRaEr0f_c&redir_esc=y#v=onepage&q&f=false (accessed 12.31.21).

Yuan, R., Li, Z., Guan, X., Xu, L., 2010. An SVM-based machine learning method for accurate Internet traffic classification. Information Systems Frontiers 12, 149–156. https://doi.org/10.1007/S10796-008-9131-2/TABLES/9

Zeiler, M.D., Fergus, R., 2013. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. 1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings.

Zhang, C., Lu, Y., 2021a. Study on artificial intelligence: The state of the art and future prospects. J Ind Inf Integr 23, 100224. https://doi.org/10.1016/J.JII.2021.100224

Zhang, C., Lu, Y., 2021b. Study on artificial intelligence: The state of the art and future prospects. J Ind Inf Integr 23, 100224. https://doi.org/10.1016/J.JII.2021.100224

Zhu, X., Goldberg, A.B., 2009. Introduction to Semi-Supervised Learning. https://doi.org/10.2200/S00196ED1V01Y200906AIM006 6, 1–116. https://doi.org/10.2200/S00196ED1V01Y200906AIM006