Universidade do Minho
Escola de Engenharia

David Miguel Duarte Rodrigues Alves

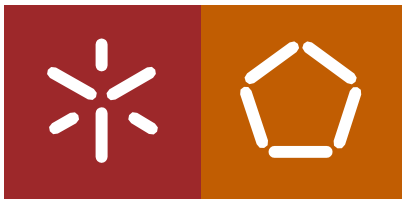
Desenvolvimento de um Sistema de Ensino para Crianças com Perturbação do Espectro do Autismo com Recurso a um Robô Humanoide

janeiro de 2023

Desenvolvimento de um Sistema de Ensino para Crianças com Perturbação do Espetro do Autismo com Recurso a um Robô Humanoide

David Miguel Duarte Rodrigues Alves

UMinho | 2023



Universidade do Minho
Escola de Engenharia

David Miguel Duarte Rodrigues Alves

Desenvolvimento de um Sistema de Ensino para Crianças com Perturbação do Espetro do Autismo com Recurso a um Robô Humanoide

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação do
Professor Doutor Cesar Analide de Freitas e Silva da Costa Rodrigues
Professora Doutora Filomena Maria Rocha Menezes Oliveira Soares

janeiro de 2023s

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicado.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositórioUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho

Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

Assinado por: **David Miguel Duarte Rodrigues Alves**
Num. de Identificação: 13578447
Data: 2023.01.30 17:44:49+00'00'

AGRADECIMENTOS

Aproveito esta oportunidade para expressar a minha gratidão a todas as pessoas que me ajudaram ao longo do desenvolvimento e escrita desta dissertação. Em particular, gostaria de agradecer:

— Ao meu orientador, Professor Doutor Cesar Analide de Freitas e Silva da Costa Rodrigues, pelos conhecimentos transmitidos, pelo feedback e pela supervisão;

— À minha coorientadora, Professora Doutora Filomena Maria Rocha Menezes Oliveira Soares, pelo apoio, disponibilidade, interesse, conhecimentos transmitidos e pela dedicação. Estou extremamente grato à Professora pelo tempo e energia investidos em mim e neste trabalho;

— À minha amada esposa, Manuela Silva, pelo apoio e incentivo constantes durante todo o processo de escrita desta dissertação. O seu amor incondicional e paciência cruciais ao longo deste processo. Sem ela ao meu lado eu não conseguiria ter finalizado esta etapa da minha vida;

— Ao meu amigo, Vinícius Silva, por me ter ajudado a dar início a este processo, pelo acompanhamento, pelas ideias e opiniões pois sem ele este tema não teria surgido;

— À PRIMAVERA BSS, ao meus colegas de equipa, mas, principalmente, ao meu chefe Hugo Lourenço, por me incentivar e ter disponibilizado o tempo necessário para a concretização desta dissertação;

— Por fim, mas igualmente importante, à minha companheira peluda Fox, por estar sempre presente oferecendo amor e conforto durante todas as horas de pesquisa e escrita.

Obrigado a todos pelas suas contribuições ao longo deste processo.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Progressivamente, os robôs tendem a ser mais cooperativos e inteligentes, para que possam interagir de forma natural com os seres humanos. Uma tipologia de robôs frequentemente utilizada na interação com os seres humanos são os robôs humanoides, uma vez que se assemelham fisicamente à aparência do corpo humano e são capazes de prestar auxílio na realização de diversas tarefas do quotidiano. Ao longo da última década, investigadores têm estudado a viabilidade do uso de robôs humanoides no estímulo da interação social em crianças com Perturbação do Espectro do Autismo (PEA). O autismo é um transtorno do neurodesenvolvimento que se manifesta precocemente, onde o comportamento dos indivíduos é caracterizado por padrões repetitivos, atividades ou interesses restritos e pela dificuldade na comunicação/ interação social.

Neste contexto, a presente dissertação tem como objetivo o desenvolvimento de um sistema de ensino interativo, capaz de promover o desenvolvimento sócio emocional em crianças com PEA utilizando, como mediador, um robô humanoide. O sistema desenvolvido possui uma arquitetura capaz de incorporar diferentes atividades de ensino e utiliza o algoritmo *You Only Look Once* (YOLO) para deteção de objetos em imagens. Com a finalidade de testar o sistema desenvolvido, foram desenvolvidas duas atividades de ensino, nomeadamente, o ensino de figuras geométricas e cores.

Finalmente, o sistema desenvolvido foi testado de duas formas distintas: a) sem o robô humanoide, sendo este substituído pela voz do computador; b) com robô humanoide, sendo este responsável pela interação robô-humano. Os resultados revelam que o sistema de ensino desenvolvido é capaz de detetar, com elevada percentagem de precisão, as figuras geométricas e as cores pré-definidas na lista de atividades, podendo assim ser uma ferramenta promissora no ensino de crianças com dificuldades de aprendizagem.

Palavras-Chave: Inteligência Artificial; Reconhecimento de Objetos; Robô ZECA; Perturbação do Espectro do Autismo (PEA); *You Only Look Once* (YOLO)

ABSTRACT

Progressively, robots tend to be more cooperative and intelligent, so they can interact naturally with humans. A typology of robots frequently used in the interaction with humans are humanoid robots, once they physically resemble of the human body and they are able to help in various daily tasks. Over the past decade, researchers have studied the feasibility of using humanoid robots to stimulate social interaction in children with Autism Spectrum Disorder (ASD). Autism is a neurodevelopmental disorder that manifests itself early, where the behavior of individuals is characterized by repetitive patterns, restricted activities or interests, and difficulty in communication/social interaction.

In this context, the present dissertation aims to develop an interactive teaching system, capable of promoting socio-emotional development in children with ASD using, as a mediator, a humanoid robot. The developed system has an architecture capable of incorporating different teaching activities and uses the You Only Look Once (YOLO) algorithm to detect objects in images. In order to test the developed system, two teaching activities were developed, namely, the teaching of geometric figures and colors.

Finally, the developed system was tested in two different ways: a) without the humanoid robot, which was replaced by the computer voice; b) with humanoid robot, which is responsible for robot-human interaction. The results show that the developed education system is able to detect, with a high percentage of accuracy, the pre-defined geometric figures and colors in the list of activities, showing that the developed system is a promising tool able to teach children with learning difficulties.

Keywords: Artificial intelligence; Image Recognition; ZECA robot; Autism Spectrum Disorder (ASD); You Only Look Once (YOLO)

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÓNIMOS

ABA	*Applied Behavior Analysis*
AIBO	*Artificial Intelligence roBOt*
AP	*Average Precision*
API	*Application Programming Interface*
BiFPN	*Weighted Bi-directional Feature Pyramid Network*
CDC	Centro de Controle e Prevenção de Doença
CNN	Rede Neuronal Convolucional
DIR	*Developmental, Individual Difference*
FN	Falso negativo
FP	Falso positivo
FPS	*Frames per second*
GPU	Unidade de processamento gráfica
GUI	Interface Gráfica
HOG	*Histogram of Oriented Gradients*
HTTP	*Hypertext Transfer Protocol*
IoU	Interseção sobre União
JDK	*Java Development Kit*
mAP	*Mean Average Precision*
MS COCO	*Microsoft Common Objects in Context*
PEA	Perturbação do Espectro do Autismo
R-CNN	*Region-based Convolutional Neural Networks*
RGB	*Red, Green, Blue*
RNA	Rede Neuronal Artificial
RNR	Rede Neuronal Recorrente
SSD	*Single Shot Detector*
SVM	*Support Vector Machine*
VN	Verdadeiro negativo
VP	Verdadeiro positivo
WPF	*Windows Presentation Foundation*
YOLO	*You Only Look Once*
ZECA	*Zeno Engaging Children with Autism*

1. INTRODUÇÃO

Primeiramente, será apresentada uma primeira abordagem do tema a tratar no Enquadramento, incluindo o uso dos robôs-humanoides e interação com o ser humano e a Perturbação do Espectro do Autismo. De seguida, serão apresentados as motivações e os objetivos deste trabalho. Por fim, será apresentada a estrutura da presente dissertação.

1.1 Enquadramento

Uma das tipologias de robôs frequentemente utilizadas na interação com o ser humano é o robô humanoide, cuja aparência física assemelha-se ao corpo humano comunicando através de gestos e fala. A capacidade de interagir com os seres humanos de maneira natural em contextos sociais, através da fala, gestos, expressões faciais e movimentos do corpo, é um fator fundamental para garantir a aceitação de robôs na sociedade.

A principal característica das crianças com Perturbação do Espectro do Autismo (PEA) é a dificuldade de interação social, isto é, dificuldade na capacidade de interagir e de se comunicar e na reciprocidade social. Adicionalmente, estas crianças apresentam comportamentos estereotipados ou rígidos, associadas a comportamentos repetitivos e/ou de interesses marcados por objetos ou temas específicos (Robins et al., 2004). A designação de espectro foi atribuída pela variabilidade dos sintomas, desde as formas mais leves até às formas mais graves.

Atualmente, a investigação existente sobre a interação humano-robô tem demonstrado benefícios no uso de robôs na educação de crianças autistas, nomeadamente no melhoramento do nível de resposta, de envolvimento e de interesse das crianças e também na promoção de novos comportamentos sociais (Alcorn et al., 2019; Tapus et al., 2007).

Nos estudos realizados por Alnajjar et al., 2020 e Ricks & Colton, 2010, os autores referiram que as intervenções utilizando robôs são mais eficazes no diagnóstico PEA e que os robôs podem, efetivamente, melhorar as habilidades de interação e atenção dessas crianças, mesmo em indivíduos que não estejam dispostos a interagir com terapeutas humanos. Costa, 2014 verificou

que as crianças com autismo tendem a desenvolver habilidades socio emocionais ao lidar com robôs humanoides. Alnajjar et al., 2020 referiu que, apesar das abordagens metodológicas de ensino variarem significativamente nos estudos realizados, os investigadores tendem a desenvolver soluções semelhantes com base em premissas conceituais e práticas semelhantes.

Tuna & Tuna, 2019 observam que, independentemente do conteúdo e da quantidade de erros que as crianças cometessem durante o processo de aprendizagem de uma determinada tarefa, os robôs não demonstravam fadiga, contrariamente a um educador humano. Adicionalmente, nos estudos realizados por Verner et al., 2016 verificou-se que o uso de robôs no processo de aprendizagem permite assegurar um ensino mais personalizado do que no ensino tradicional, uma vez que os robôs interagem de forma particular e diferenciada com as crianças, de acordo com as suas limitações e ritmo de aprendizagem.

Neste trabalho será abordado o sistema de ensino interativo desenvolvido, com a finalidade de promover o desenvolvimento sócio emocional em crianças com PEA utilizando, como mediador, um robô humanoide *Zeno R50 Robokind Robotic Platform*, designado como ZECA (*Zeno Engaging Children with Autism*). O robô ZECA tem sido utilizado em diversos estudos científicos relacionados com a interação social com crianças autistas (Feil-Seifer & Mataric, 2005) e com a deteção e promoção do reconhecimento de emoções (Costa et al., 2014; Pereira et al., 2017). Serão também abordados os dois módulos de atividades de ensino desenvolvidos para testar o sistema, nomeadamente, o módulo de reconhecimento de figuras geométricas e o módulo de cores e os resultados obtidos.

1.2 Motivação

Tal como referido anteriormente, a Perturbação do Espetro do Autismo (PEA) é um distúrbio neurológico que afeta a interação social, a capacidade de comunicação e o comportamento das pessoas. Contudo, estes distúrbios não devem ser um impedimento para que os indivíduos com PEA possam realizar as tarefas do seu quotidiano.

Os robôs humanoides estão em constante desenvolvimento e têm sido utilizados numa ampla variedade de áreas científicas, desde a saúde até ao ensino. Paralelamente, os estudos científicos têm mostrado que os indivíduos com PEA tendem a relacionar-se harmoniosamente com este tipo de robôs. Assim sendo, o uso de robôs humanoides poderá ser uma grande mais-valia no desenvolvimento das habilidades de comunicação e interação social destes indivíduos.

Outra temática em voga atualmente é o reconhecimento de objetos em imagens. Trata-se de uma área em constante evolução, com algoritmos cada vez mais eficazes e rápidos capazes de detetar objetos em imagens.

Face ao exposto anteriormente, a grande motivação desta dissertação passa a junção destas duas temáticas – Robôs Humanoides e Reconhecimento de Imagem – através do uso de robôs no instrução de atividades de ensino, como cores e formas geométricas, a crianças com PEA. Com a utilização de robôs humanoide na educação, pretende-se ajudar as crianças com PEA no processo de aprendizagem e nas suas habilidades de comunicação e interação com as pessoas.

1.3 Objetivos

O objetivo principal deste trabalho consiste no desenvolvimento de um sistema de ensino interativo capaz de incorporar diferentes módulos de atividades de ensino, para promover o desenvolvimento sócio emocional em crianças com PEA utilizando, como mediador, um robô humanoide, o ZECA. Com a finalidade de testar o sistema desenvolvido, desenvolveu-se dois módulos de atividades de ensino, nomeadamente, o módulo de reconhecimento de figuras geométricas e o módulo de cores.

De forma a atingir o objetivo principal, procedeu-se, inicialmente, ao desenvolvimento de uma Application Programming Interface (API) para a uniformização da comunicação entre o robô e o computador. Seguidamente, procedeu-se ao treino das redes neuronais para a deteção de objetos, nomeadamente para a deteção de cores (vermelho, azul e verde) e de figuras geométricas (quadrado, triângulo e círculo). Na fase seguinte, procedeu-se ao desenvolvimento da segunda API para a deteção de objetos em tempo real. Esta API utiliza as redes neuronais

treinadas previamente. Finalmente procedeu-se ao desenvolvimento da interface gráfica, cuja função é criar listas de instruções para o robô.

1.4 Organização do documento da dissertação

O presente documento é composto por 6 Capítulos, sendo o primeiro a introdução, onde é apresentado o enquadramento, a motivação, os objetivos e a estrutura do presente documento.

No Capítulo 2, é abordado o autismo e são citados estudos que demonstram os benefícios no uso de robôs no ensino de atividades educacionais a crianças com perturbação do espectro do autismo, nomeadamente no melhoramento do nível de resposta, de envolvimento e de interesse das crianças e também na promoção de novos comportamentos sociais. Neste Capítulo são também abordadas as técnicas de *Machine Learning* e os algoritmos utilizados para reconhecimento de objetos numa imagem.

No Capítulo 3 é apresentado o *hardware* e *software* relativos ao robô, a câmara utilizada para captação das imagens e os materiais utilizados. Seguidamente, é apresentado o algoritmo utilizado para treino das redes neuronais, as configurações adotadas no algoritmo e os procedimentos necessários para o treino das redes neuronais.

No Capítulo 4 são apresentados os desenvolvimentos realizados, nomeadamente as API e a interface gráfica. Neste capítulo também são apresentados os desenvolvimentos realizados para automatizar o treino de uma rede neuronal. Por fim é apresentada uma visão geral da arquitetura do sistema.

No Capítulo 5 são apresentados os principais resultados obtidos nos testes realizados de forma a avaliar a percentagem de precisão dos objetos detetados pela câmara. Os testes foram realizados sem o robô ZECA, sendo este substituído pelo computador e com robô ZECA, onde foi possível avaliar o desempenho do robô.

As principais conclusões deste trabalho são apresentadas no Capítulo 6, onde são também expostas algumas propostas para trabalhos futuros.

Por fim, encontram-se discriminadas as referências bibliográficas consultadas no âmbito desta dissertação.

2. ESTADO DE ARTE

Neste Capítulo é apresentado o conceito de Perturbação do Espetro do Autismo (PEA) e estudos científicos onde a utilização de robôs humanoides no ensino de atividades a crianças/indivíduos com PEA revelou-se uma mais-valia no desenvolvimento destes indivíduos. Por fim, será abordado a temática de Machine Learning e os algoritmos mais utilizados.

2.1 Perturbação do Espetro do Autismo e ensino de atividades educacionais com robô

A Perturbação do Espetro do Autismo (PEA) é um transtorno do neurodesenvolvimento que afeta a condição comportamental do ser humano. Os indivíduos com PEA apresentam défices em dois domínios distintos, nomeadamente (Hyman et al., 2020):

— Na comunicação e interação;

— Em padrões de comportamento, sendo estes restritivos e repetitivos.

Adicionalmente, as pessoas com PEA apresentam dificuldades em manter o contacto visual, no reconhecimento da linguagem corporal e em aplicar, socialmente, as competências adquiridas fora do contexto em que são aprendidas (Scassellati et al., 2012). Estudos revelam que a PEA afeta, aproximadamente, 1% da população mundial (van der Hallen et al., 2019) e 1 em cada 44 crianças de acordo com o Centro de Controle e Prevenção de Doença (CDC) (Maenner et al., 2021). Normalmente, a PEA é diagnosticada até aos 2 anos de idade. Nos estudos realizados por Esther Ben Itzchak & Ditza A. Zachor, 2011 verificou-se que o tratamento personalizado e antecipado dos indivíduos com PEA ajuda a melhorar a qualidade de vida dos mesmos no quotidiano.

Por norma, as crianças com PEA demonstram pouco interesse na realização das tarefas, o que, consequentemente, dificulta a aprendizagem das crianças e a tarefa do lecionador (Kumazaki et al., 2018). Com o avanço da inteligência artificial, os robôs tendem a apresentar habilidades sociais mais sofisticadas e mais próximas do comportamento real dos seres humanos. A utilização de robôs no quotidiano dos seres humanos sido cada vez maior e com diferentes finalidades, entre as quais, a reabilitação, a educação e a companhia.

Uma das tipologias de robôs frequentemente utilizada na interação com o ser humano é o robô humanoide, cuja aparência física assemelha-se ao corpo humano. Os robôs humanoides podem estar equipados com câmaras, sensores e mecanismos que os possibilitam de se comunicarem através de gestos e fala e de se moverem autonomamente. No tratamento de crianças com PEA, estes robôs auxiliam o trabalho dos lecionadores/terapeutas através do contato visual ou físico, na captação de atenção, na participação em atividades interativas, na imitação, na comunicação verbal e na expressão de emoções (Diehl et al., 2012). Devido à semelhança com os seres humanos, estes robôs conseguem desencadear emoções nos seres humanos, o que permite estabelecer uma relação mais próxima humano-robô (Henschel et al., 2020). Nos estudos realizados por Irfan et al., 2019 verificou-se que as interações humano-robô podem tornar-se repetitivas ao longo do tempo, o que conduz a uma redução da captação da atenção das crianças devido ao desaparecimento do efeito '*novidade*' provocada pelo robô. Contudo, e principalmente em crianças com PEA com défice de atenção, é fundamental conseguir a atenção do utilizador no desenvolvimento de tarefas.

Atualmente, a investigação existente sobre a interação humano-robô tem demonstrado o benefício do uso de robôs no melhoramento da interação social e da atenção nas crianças com perturbação do espectro do autismo (Alnajjar et al., 2020; Ricks & Colton, 2010). Nos estudos realizados por Wood et al., 2021 verificou-se que as crianças com PEA tendencialmente interagem com maior facilidade com objetos não sociais como, por exemplo robôs, tendo estes a capacidade de estimular e de manter estas crianças mais focadas nas tarefas. Rakhymbayeva et al., 2021 refere ainda que o uso a longo prazo de um robô humanoide na terapia de crianças com autismo pode ser conseguido através do ensino de atividades de aprendizagem de acordo com a faixa etária das crianças.

Em modo de conclusão, as tecnologias robóticas trouxeram desenvolvimentos na educação e na terapia em crianças com PEA (Saleh et al., 2021), e estas tecnologias, podem ser utilizadas como tutores no ensino, sendo eficazes em aumentar comportamentos cognitivos e afetivos, alcançando resultados semelhantes aos tutores humanos (Belpaeme et al., 2018).

Na Tabela 1 encontram-se sumarizados alguns estudos relevantes acerca do ensino de tarefas a crianças com PEA utilizando, como mediador, várias tipologias de robôs.

Tabela 1 – Estudos publicados ao longo das últimas duas décadas acerca do ensino de tarefas a crianças com PEA utilizando, como mediador, várias tipologias de robôs

Publicações	**Nº participantes**	**Faixa etária**	**Principal Objetivo**	**Resultados**
Stanton et al., 2008	11	5-8	Analisar, se um robô semelhante a um cão, pode auxiliar no desenvolvimento da interação social em crianças com PEA	Com o auxílio do robô AIBO, as crianças aprenderam a articular um maior número de palavras e mostraram-se mais envolvidas na fala e no comportamento de interação
Marti & Giusti, 2010	5	6–11	Com o auxílio do robô Iromec, ajudar as crianças com PEA a aprender diferentes tipos de jogos de forma a melhorar a interação social	Todas as crianças revelaram interesse na aprendizagem de todos os jogos
Nikolopoulos et al., 2010	4	8-19	Desenvolver um meio econômico, prático e eficiente de ajudar a ensinar comportamentos social em indivíduos com PEA com o auxílio do robô LEGO NEX. Foi incluído modelo de intervenção DIR/Floortime	Foi verificada a viabilidade do uso de robôs no ensino para crianças com autismo
Costa et al., 2014	2	14 e 16	Promover o reconhecimento de emoções em crianças com PEA com o auxílio de um robô humanoide	Os participantes mostraram, com sucesso, a capacidade de reconhecer emoções
Kose et al., 2015	6	6-8	Estimular as crianças com PEA a compreender e a imitar os sinais ensinados	O desempenho de aprendizagem das crianças foi melhorado. Adicionalmente, foi possível ensinar palavras abstratas com o auxílio de contos de histórias
Hawon Lee & Eunja Hyun, 2015	4	4-5	Utilização de um *script* para ensinar elementos educacionais divertidos	Os resultados mostram uma otimização nos diálogos das crianças com o robô e com a troca de expressões emocionais
Barakova et al., 2015	6	8–12	Utilização de um robô humanoide LEGO no tratamento de crianças com PEA com a finalidade de melhorar as suas habilidades sociais	A interação com Robô LEGO aumentou a interação social das crianças
Shamsuddin et al., 2015	12	5–12	Ensinar habilidades de comunicação a crianças com PEA com o auxílio da técnica de análise de comportamento (ABA) e um robô	Os resultados demonstraram que o envolvimento das crianças aumentou ao longo das sessões de aprendizagem com o robô
Nunez et al., 2015	10	23	Melhorar *"enganement"* em crianças com PEA durante as sessões de aprendizagem com o auxílio de um robô humanoide	Os participantes revelaram bastante interesse nos robôs
Zheng et al., 2016	4	4.61	Melhorar as habilidades de crianças com PEA através da imitação e com recurso ao robô	O sistema proposto foi capaz de estimular a imitação de gestos em crianças com PEA

(V. Silva et al., 2017)	6	8-9	Promover a imitação e reconhecimento de expressões faciais em crianças com PEA com o auxílio do robô RoboKind Zeno R50	O robô interagiu de forma natural e confortável com as crianças com PEA, especialmente no reconhecimento de emoções e imitação. Os resultados sugerem que um robô humanoide pode ser utilizado como um mediador elegível em atividades de reconhecimento de emoções
Golestan et al., 2017	4	4-6	Utilizar o robô Sphero para desenvolver habilidades sociais e de comunicação em crianças com PEA	As crianças foram estimuladas a falar mais palavras, a praticar "amizade" com o robô e a compartilhar emoções. Verificou-se também uma maior interação destas crianças com os pais/terapeutas
Gelsomini et al., 2017	2	7-9	Utilização do robô Puffy como uma ferramenta de aprendizagem e brincadeira para crianças	As crianças mostraram melhoramento no processo de aprendizagem
Bharatharaj et al., 2017	6	6–16	Melhoramento das habilidades de interação social de crianças com PEA com o auxílio de um robô semelhante a um papagaio	Os intervenientes demonstraram atração e alegria na interação com o robô
Moorthy & Pugazhenthi, 2017	2	Até 10	Ensinamento das habilidades psicomotoras a crianças com PEA com recurso a um kit robótico *joystick*	A aprendizagem foi positiva e, com o uso do kit na transmissão de direções, verificou-se a coordenação olho-mão e preensão palmar nas crianças
Taheri et al., 2017	4	6	Ensinamento dos fundamentos da música e melhoramento das habilidades sociais das crianças como o auxílio de um robô xilofone/baterista	Os resultados apontam benefícios no uso do robô no ensino da música e mostram um melhoramento das habilidades sociais nas crianças
Pereira et al., 2017	15	3-5	Utilização do robô RoboKind Zeno R50 no ensinamento de cores e figuras geométricas	Os resultados demonstraram que as crianças interagem com o robô de forma natural
Sakka et al., 2018	6	11-15	Utilização de robôs no melhoramento das habilidades sociais associando a voz a gestos em crianças com PEA	As crianças mostraram melhoramentos na comunicação
Fachantidis et al., 2020	22	-	Melhoramento da interação social em crianças com PEA com o auxílio de um robô educacional	Os resultados demonstram uma mudança na atitude dos alunos. Foram observados melhoramentos nas habilidades sociais, comunicacionais e emocionais
Jabar H. Yousif & Mohammed J. Yousif, 2020	-	-	Utilização do robô Nao no auxílio das crianças com PEA a ler histórias, a soletrar palavras e a corrigir respostas a perguntas de matemática	O uso de robôs ajudou as crianças na aquisição dos conhecimentos com maior facilidade
Efstratiou et al., 2021	-	-	Utilização do robô Pepper no melhoramento da memória a curto e a longo prazo, bem como na comunicação e nas habilidades sociais	A intervenção com o robô resultou num aumento da interação e da motivação para a comunicação nas crianças
Korneder et al., 2021	3	5	Utilização de um robô no ensinamento de uma linguagem e habilidades de comunicação a crianças com PEA	Os participantes revelaram um melhoramento nas habilidades de comunicação após sete sessões

2.2 Machine Learning

Reconhecer padrões ou objetos em imagens pode trazer várias vantagens em inúmeras áreas, desde a medicina (Iqbal et al., 2018) até ao ensino (Sun et al., 2018). *Machine learning* é uma inteligência artificial que possibilita aos computadores tomarem decisões com o auxílio de algoritmos. Estes algoritmos são treinados a partir de dados para reconhecerem padrões e tornam-se capazes de fazer previsões. A *Machine Learning* encontra-se dividida em três áreas principais, nomeadamente: classificação, a regressão e agrupamento. De forma geral, a classificação, a regressão e o agrupamento podem estar divididas em três tipos de aprendizagem, nomeadamente a aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço. A aprendizagem supervisionada, que inclui as áreas de classificação e regressão, utiliza dados já classificados e compara as suas previsões com esses dados. A aprendizagem não supervisionada, que inclui a área agrupamento, agrupa os dados conforme comportamentos similares. Por último, a aprendizagem por reforço, frequentemente utilizada em jogos e robótica, é uma aprendizagem com base no erro cometido, ou seja, o computador utiliza a tentativa erro para encontrar uma solução para o problema proposto.

2.2.1 Classificação

A classificação, a partir de um conjunto de dados de entrada, prevê e classifica em que classe os pertencem os dados.

A tarefa de classificação começa com um conjunto de dados no qual as atribuições de classe já são conhecidas. A título de exemplo, um modelo de classificação que prevê o risco de atribuir crédito a clientes pode ser desenvolvido com base na observação de vários processos de aplicações para crédito e os respetivos resultados ao longo de um período de tempo. O tipo mais simples de classificação é a binária, onde o atributo alvo só tem dois valores possíveis, por exemplo email *spam* ou email não *spam*. Os atributos com mais de dois valores possíveis são atributos *Multiclass*. No processo de treino (construção do modelo), o algoritmo de classificação encontra relações entre os valores dos atributos de previsão e os valores dos atributos objetivo.

Cada algoritmo de classificação pode usar diferentes técnicas para encontrar estas relações. Os modelos de classificação podem, então, ser testados comparando os valores de previsão com valores de um *dataset*.

A classificação possui diferentes tipologias de algoritmos, nomeadamente: Árvore de Decisão, Naive Bayes, *Support Vector Machine* (SVM) e Redes Neuronais Artificiais.

i. Árvore de Decisão

Geram automaticamente regras que são afirmações condicionais que utilizam, por base, a lógica de construção de uma árvore. A árvore é constituída por nós de decisão, onde são executados os testes com atributos específicos e por folhas, que indicam o respetivo valor final (Quinlan, 1986). Por exemplo, se o objetivo for decidir fazer surf, para tal é necessário ter em conta alguns o céu (sol, tempestade ou chuva), as ondas (grandes ou pequenas) e o vento (forte ou fraco). Para um dia de sol e com ondas grandes o resultado da árvore é SIM – Fazer Surf. A árvore de decisão encontra-se exemplificada na Figura 1.

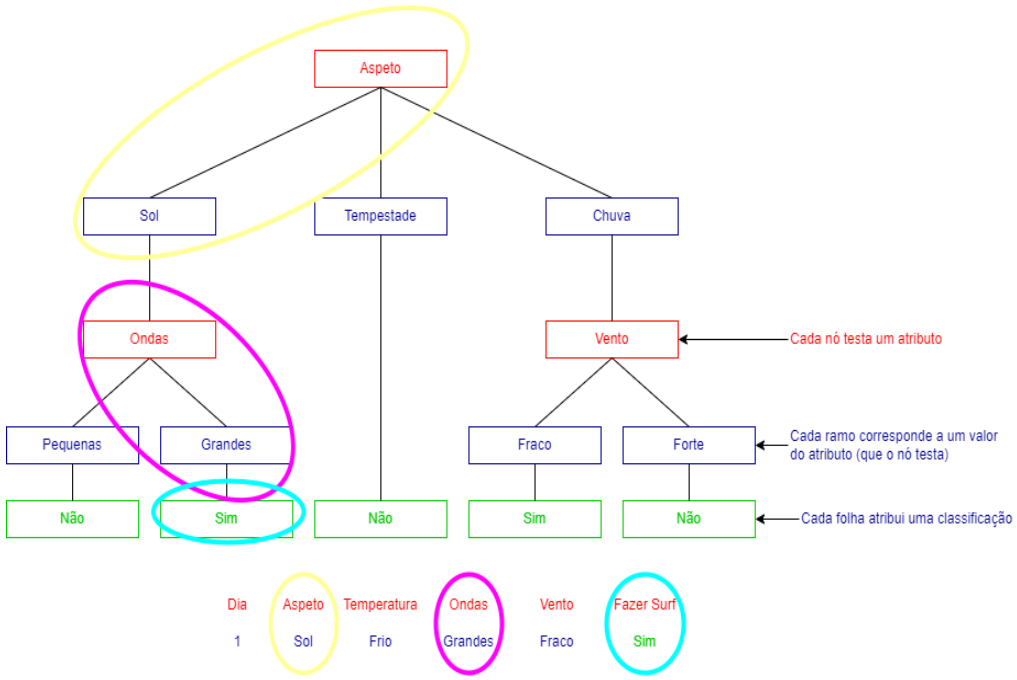


Figura 1 – Árvore de decisão para fazer Surf (adaptado Quinlan, 1986)

ii. Naive Bayes

Trata-se de uma técnica de classificação baseada no Teorema de Bayses, bastante conhecido e utilizado no universo das probabilidades, que utiliza a suposição de independência entre os atributos (Hand & Yu, 2001). Por exemplo, um fruto pode ser considerado como uma maçã se é vermelho, redondo e se tiver cerca de 3 polegadas de diâmetro. Mesmo que esses recursos dependam uns dos outros, todas estas propriedades contribuem de forma independente para a probabilidade de que este fruto é uma maçã. Este sistema de classificação apresenta as seguintes como vantagens:

— Simplicidade matemática, o que o torna um algoritmo "rápido";

— A eficiência em previsões multi-classes;

— Em problemas onde a suposição de independência entre os atributos é válida, este algoritmo consegue alcançar um desempenho melhor que outros métodos mais complexos.

As desvantagens associadas ao uso deste sistema de classificação são as seguintes:

— Na realidade, é muito pouco provável a existência de um conjunto de indicadores que sejam completamente independentes, isto é, normalmente existe correlação entre as variáveis;

— As probabilidades calculadas pelo algoritmo não devem ser consideradas precisas.

As aplicações práticas deste sistema de classificação são:

— Classificação de texto e filtros de SPAM;

— Previsões multi-classe;

— Previsões em tempo real, dado o rápido desempenho do modelo.

iii. Support Vector Machine (SVM)

Support Vector Machines (SVM) é um algoritmo que é utilizado para problemas de classificação. De forma sumarizada, SVM define uma linha de separação, designada como hiperplano, entre dados pertencentes de duas classes arbitrárias (Cortes & Vapnik, 1995).

a. SVM Linear

Nesse algoritmo, cada item de dados corresponde a um ponto no espaço n-dimensional. O SVM é um algoritmo que busca uma linha de separação entre duas classes distintas analisando os dois pontos, um de cada grupo, mais próximos da outra classe (Burges, 1998). Isto é, SVM escolhe a reta (também designada por hiperplano, em maiores dimensões) entre dois grupos que se distância mais de cada um (no caso da Figura 2, a reta H_3).

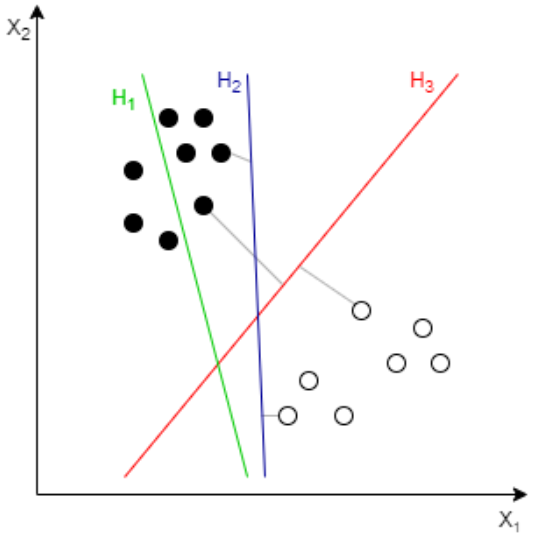


Figura 2 – Classificação linear de dados em duas classes (adaptado Lei, 2017)

Após a definição do hiperplano, o sistema conseguirá prever a qual classe pertence um dado novo ao verificar, em qual lado do hiperplano, o novo dado novo se encontra. O hiperplano tem

como principal objetivo maximizar a distância entre os pontos mais próximos de cada uma das classes.

b. SVM Multiclass

Quando é necessário proceder à classificação de dados em mais de duas classes. SVM *Multiclass*, a forma mais simples e comum é dividir problema *Multiclass* em várias classificações binárias, que podem ser *one vs one* ou *one vs all*, também designado por *one vs rest*.

A divisão em *one vs one* consiste em dividir o problema *Multiclass* em várias classificações binárias. Por exemplo, se tivermos as três classes, A, B e C, as classificações binárias serão (A, B), (A, C) e (B, C) e a classe mais votada será a escolhida. Se o SVM escolher A em (A, B), C em (A, C) e C em (B, C), o resultado será (A, C, C) e a classe escolhida será a C.

A divisão em *one vs all* consiste em comparar cada classe com todo o resto. Ou seja, se tivermos as classes A, B e C, o problema será divido nas comparações (A, B+C), (B, A+C) e (C, A+B).

c. SVM Não Linear

Existem grupos de dados que não podem ser separados apenas por hiperplano (Figura 3). Nesses casos, é utilizada o SVM Não Linear na delimitação de duas classes, onde são traçados uma ou mais linhas, retas ou curvas, para a separação das classes.

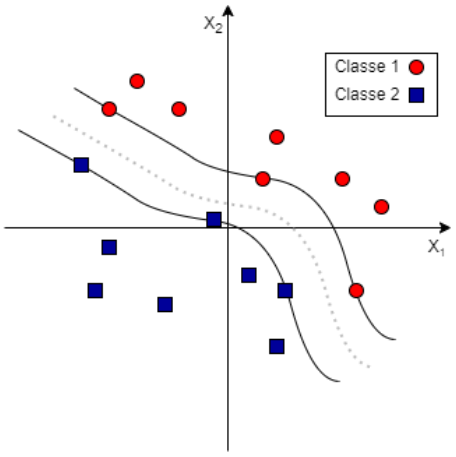


Figura 3 – SVM não linear (adaptado Mohammadi & Minaei, 2019)

Para a separação das classes, inicialmente, o algoritmo faz uma transformação não-linear do espaço para, de seguida, separar os grupos com um SVM linear (Figura 4). Dessa forma, apesar da separação ser um hiperplano no espaço das *features*: designação do espaço depois da transformação, no espaço das entradas - designação do espaço inicial, a separação é não-linear.

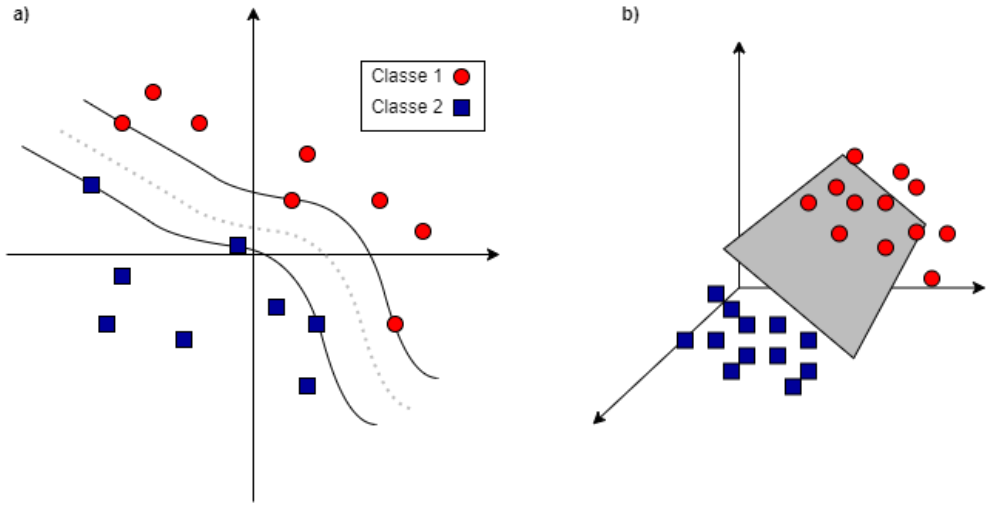


Figura 4 – Transformação não linear entre o espaço das estradas, a) e o espaço das features , b) (adaptado Mohammadi & Minaei, 2019)

d. Regressão por SVM

Apesar de, frequentemente, o SVM ser mais utilizado para classificação, o SVM também pode ser utilizado para regressão, isto é, para prever valores contínuos com base nos dados em vez de prever as classes às quais os dados pertencem.

iv. Redes Neuronal Artificial (RNA)

Uma rede neuronal artificial (RNA) é um sistema computacional concebido com base num modelo simplificado do sistema nervoso central dos seres humanos. É definida por uma estrutura

interligada de unidades computacionais, designadas por neurónios. A propriedade mais importante da RNA é a capacidade de aprender e, com isso, melhorar o seu desempenho.

Os neurônios (Figura 5) comunicam-se através de sinapses. A sinapse é a região onde dois neurônios entram em contato e, através da qual, os impulsos nervosos são transmitidos. Os impulsos recebidos por um neurônio A através dos dentritos (*inputs*) são processados e o neurônio A dispara produzindo uma substância neurotransmissor, que flui do corpo celular para o axônio (*outputs*), que pode estar conectado a um dendrito de um outro neurônio B.

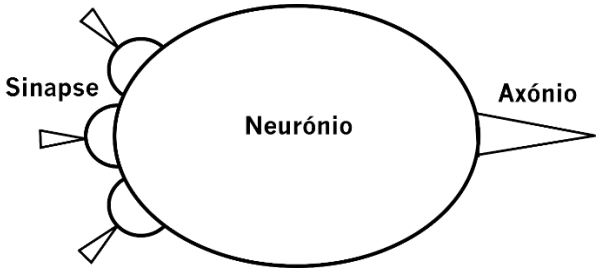


Figura 5 – Neurónio

Uma RNA (Figura 6) é composta por várias unidades de processamento. As unidades de processamento geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O funcionamento das RNA pode ser sumarizado da seguinte forma:

— Sinais são apresentados à entrada;

— Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;

— É feita a soma ponderada dos sinais que produz um nível de atividade;

— Se este nível de atividade exceder um certo limite (*threshold*), a unidade produz uma determinada resposta de saída.

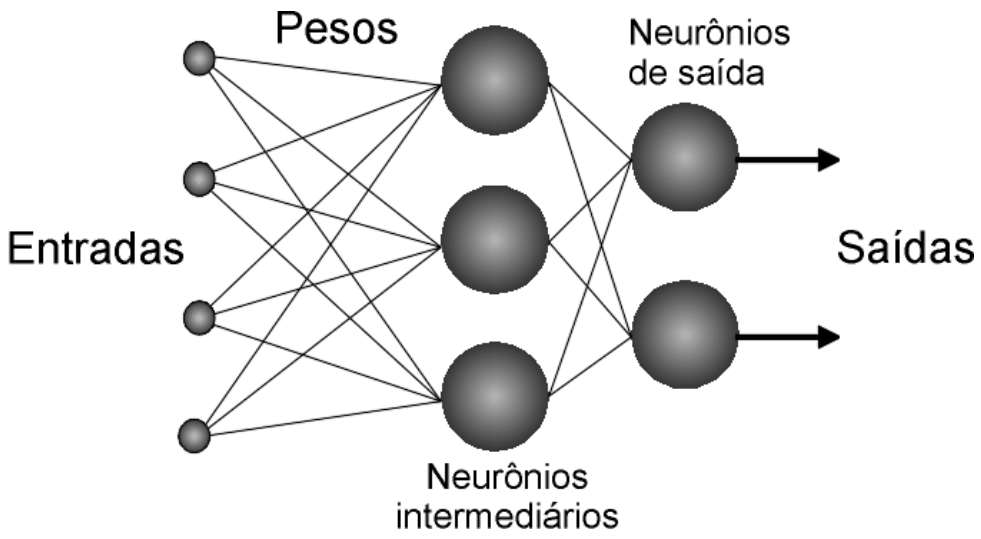


Figura 6 – Funcionamento das RNAs

Dentro das RNA, encontra-se a Rede Neuronal Convolucional (CNN) e a Rede Neuronal Recorrente (RNR).

a. Rede Neuronal Convolucional (CNN)

A CNN é uma rede neuronal artificial do tipo *feedforward* utilizada no processamento e análise de imagens. A partir de uma imagem de entrada, a CNN é capaz de diferenciar aspetos e reconhecer os objetos presentes na mesma. A arquitetura CNN utiliza um algoritmo de *deep learning*, que permite que o treino da rede seja feito de uma forma simples e rápida.

De forma sumarizada, a CNN utiliza uma variação de perceptrons multicamadas e contém uma ou mais camadas convolucionais, que podem estar completamente conectadas ou agrupadas. Estas camadas geram mapas que registam uma área da imagem, que é dividida em retângulos. A imagem dividida é então enviada para o processamento. A CNN tem como vantagens:

— Alta precisão em problemas de reconhecimento de imagem;

— Deteção automática de objetos sem qualquer supervisão humana.

As desvantagens do uso da CNN são as seguintes:

— A CNN não codifica a posição e orientação do objeto;

— São necessários a utilização de muitos dados de treino.

b. Rede Neuronal Recorrente (RNN)

Uma RNN é uma rede do tipo de RNA desenvolvida para reconhecer padrões em sequências de dados. Os algoritmos que utilizam este tipo de redes consideram o tempo e uma sequência de dados para deteção de padrões, sendo estes particularmente adequados para a análise de séries temporais. Esta rede utiliza conexões cíclicas, Figura 7, entre as suas camadas que permite manter informações de estado.

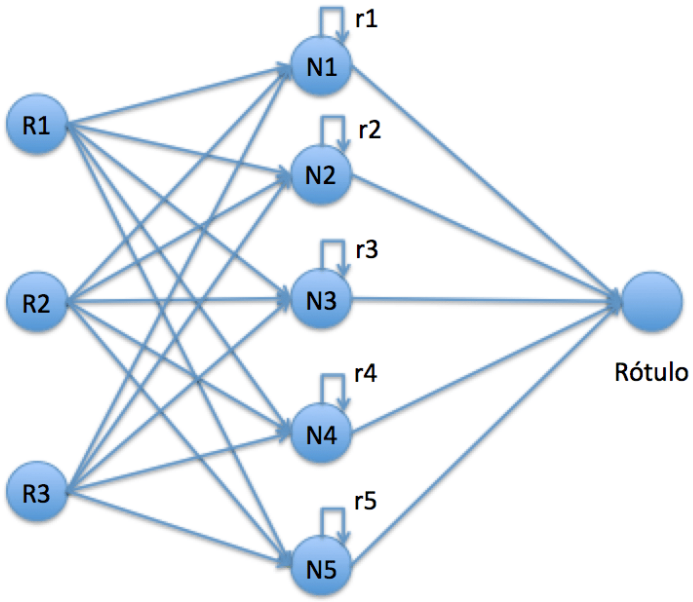


Figura 7 – Rede Neuronal Recorrente (Carvalho et al., 2016)

2.2.2 Regressão

A regressão, similarmente à classificação, utiliza os dados de entrada já classificados para prever uma resposta. A grande diferença entre estas duas áreas do *Machine Learning* é que, na regressão, o objetivo é estimar um valor numérico e não uma classificação de uma observação. Um exemplo da regressão é um modelo que utilize a idade e os anos de escolaridade de um indivíduo não classificado anteriormente para tentar estimar o seu salário. Nestes casos, são utilizados como base do modelo as idades, os anos de escolaridades e os salários dos indivíduos já classificados previamente.

2.2.3 Agrupamento

O agrupamento, também conhecido como *"Clustering"*, tem como objetivo agrupar dados em grupos conhecidos como *"clusters"*. Estes dados apresentam similaridades dentro de seu *cluster*. Um exemplo do agrupamento é agrupar fotos de animais similares em *clusters*, sem ter o conhecimento prévio de qual animal está a ser apresentado.

2.2.4 Algoritmos de deteção de objetos

A primeira abordagem com a finalidade de resolver problemas de deteção de objetos utiliza a técnica desenvolvida por Harr Cascades (Viola & Jones, 2001). Desde então, vários algoritmos têm vindo a ser desenvolvidos com esta finalidade, entre os quais:

— ***Histogram of Oriented Gradients*** **(HOG)** – É um método de extração de informação de imagens que foi introduzido em 2005 pelos investigadores Dalal e Triggs (Dalal & Triggs, 2005). Este método tem como objetivo extrair informações referentes à orientação das arestas existentes numa imagem. Trata-se de uma técnica que pode ser utilizada para auxiliar o reconhecimento facial, compensando erros provocados por fatores como oclusão parcial, pose e mudanças de iluminação;

— ***Region-based Convolutional Neural Networks*** **(Fast R-CNN)** – No algoritmo anterior, R-CNN, cada região proposta na imagem é processada de um modo sequencial, isto é, só pode ser processada uma região de cada vez da imagem. O Fast R-CNN é mais rápido porque consegue processar várias regiões de uma imagem de uma só vez. Contudo, o modelo Fast R-CNN é dependente do algoritmo *Selective Search* para a escolha de regiões da imagem;

— ***Faster*** **R-CNN** – Trata-se de uma extensão do Fast R-CNN. Este algoritmo é mais rápido do que o Fast R-CNN devido à implementação da *Region Proposal Network* (RPN). A RPN é responsável pela escolha das propostas das regiões da imagem e o Fast R-CNN deteta os objetos nas regiões propostas. No início é inserida uma imagem na RPN, que retorna uma lista de regiões onde a probabilidade de existirem objetos de interesse seja elevada. As regiões escolhidas são então introduzidas à camada de *pooling de RoI* que, por sua vez, irá devolver as *features* de cada objeto de interesse. De seguida, o Fast R-CNN classifica o conteúdo das *bounding boxes* como o objeto que acha ser ou *background* (no caso de querer ignorar) e, por último, ajusta as coordenadas das *bounding boxes*, de modo a destacar melhor o objeto detetado (Resende, 2022);

— ***Single Shot MultiBox Detector*** **(SSD)** – A rede SSD é uma rede neuronal que gera regiões de interesse em imagens, nas quais realiza a localização de objetos e um classificador para detetar os tipos de objetos nas regiões de interesse (Liu et al., 2016). O termo *Single Shot* refere-se à passagem única da rede pela imagem para localizar e classificar os objetos. O termo *Multibox* está relacionado com a técnica de regressão da caixa delimitadora para a deteção de objetos, em que as coordenadas de um objeto detetado na região de interesse são regredidas para as coordenadas reais do objeto (*ground truth*);

— **RetinaNet** – Esta rede faz uso de diversas técnicas modernas, como conexões residuais e *feature pyramids*, em conjunto com uma *loss function* conhecida como *focal loss* (Lin et al., 2017). O treinamento desta rede é um processo demorado. Contudo e devido à disponibilidade de modelos pré-treinados, é possível acelerar o treinamento ao otimizar apenas os parâmetros das camadas finais da rede (G. R. Silva, 2018);

— ***EfficientDet*** – Este algoritmo utiliza uma rede de espinha dorsal (*backbone*), à qual através das suas otimizações, com o uso de uma rede de pirâmide de recursos bidirecional ponderada (BiFPN) e um método de dimensionamento, dimensiona uniformemente a resolução, profundidade e largura de todos os *backpones* e faz a previsão de classes em simultâneo (Tan et al., 2019);

— ***You Only Look Once*** **(YOLO)** – Este algoritmo, descrito por Redmon et al., 2015, é um método para deteção de objetos onde, através de uma rede convolucional, é realizado em paralelo, a previsão das caixas delimitadoras e suas respetivas classes, permitindo uma avaliação única da imagem. Outra vantagem associada a este método é o fato de dispor de informações do contexto, uma vez que faz a análise da imagem como um todo, reduzindo assim os erros de falsos positivos em relação ao plano de fundo. Adicionalmente, esse tipo de abordagem oferece uma boa generalização, permitindo ser utilizado em domínios diferentes do qual foi treinado. É conhecido pela sua rapidez e eficácia para deteção de objetos. A deteção de objetos é feita como um problema de regressão e fornece as probabilidades de classe das imagens detetadas.

2.3 Imagem digital

Uma imagem é defina por uma função computacional f (x,y) em que x e y são as coordenadas de um ponto num sistema bidimensional. A amplitude da imagem corresponde à intensidade de tom de cinza da mesma. Assim sendo, a imagem digital é então definida por conjunto finito de pontos com as respetivas coordenadas e intensidades, denominando-se estes pontos por pixéis (Edward R. Dougherty, 2020). As imagens podem estar representadas de três formas distintas:

— Preto e branco, designada por imagem binária. O valor de cada pixel é representado por 1 ou 0;

— Tons de cinza, em que os valores de cada pixel são representados por valores entre 0 e 255;

— Imagem a cores, em que é utilizado o sistema *red, green, blue* (RGB) e cada pixel é definido por valores entre 0 e 255 das cores vermelho, verde e azul.

A partir de vários tipos de dispositivos, entre as quais, máquinas fotográficas ou câmaras de vídeo, é possível captar imagens tridimensionais, transformando-a em imagens bidimensionais.

2.4 You Only Look Once

O *You Only Look Once* (YOLO), descrito por Redmon et al., 2015, é um método para deteção de objetos onde, por intermédio de uma rede convolucional é feita, em simultâneo, a predição das caixas delimitadoras e as suas respetivas classes, permitindo uma única avaliação da imagem. A rede neuronal convolucional é utilizada para extrair as características (*features*) do objeto a classificar. Trata-se de um algoritmo com arquitetura CNN, com uma precisão de deteção elevada comparativamente a outros métodos de deteção de objetos, mas com processamento de imagem bastante mais rápido. Outra mais-valia deste algoritmo é dispor de informações de contexto, isto é, o algoritmo analisa a imagem na sua totalidade, reduzindo assim os falsos positivos em relação ao plano de fundo. Adicionalmente, este tipo de análise oferece uma boa generalização, o que permite a sua utilização em domínios diferentes do qual foi treinado. Na primeira versão do YOLO, é possível o processamento de imagem através da deteção de objetos a 45 *frames* por segundo, com menos falsos positivos no fundo da imagem e com uma grande precisão na deteção apesar da baixa precisão da localização do objeto (Redmon et al., 2015).

De forma sucinta, o algoritmo utiliza uma caixa delimitadora, designada como *bounding box*, para identificar características do objeto a detetar, como a altura, a largura, centro e classe. Este algoritmo prevê qual a classe do objeto a identificar e a sua localização na imagem, tal como é demonstrado na Figura 8.

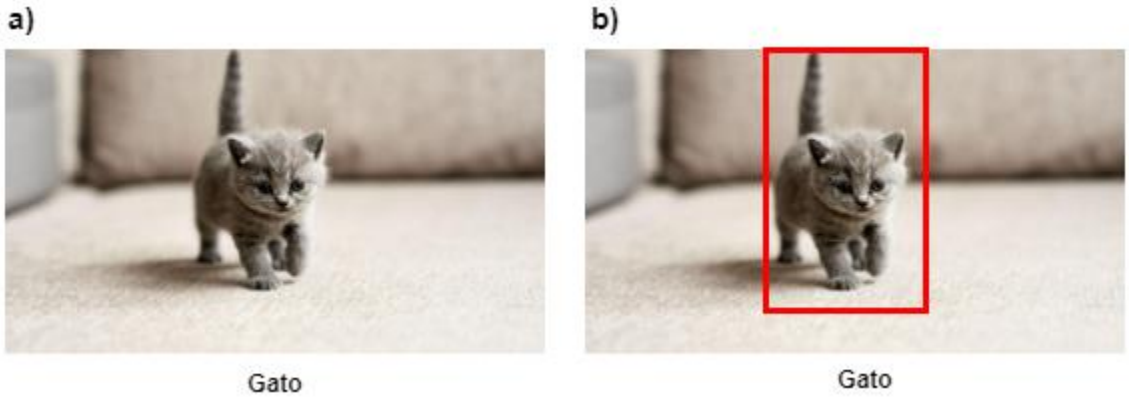


Figura 8 – YOLO: a) classificação do objeto; b) classificação e localização do objeto na imagem com a *bounding box*

Por exemplo, no treino do algoritmo para detenção de gatos e de pessoas, o resultado do algoritmo será a classe do objeto detetado, isto é, se se trata de um gato ou de uma pessoa, e a precisão da detenção, ou seja, o grau de certeza da previsão, tal com é ilustrado na Figura 9.

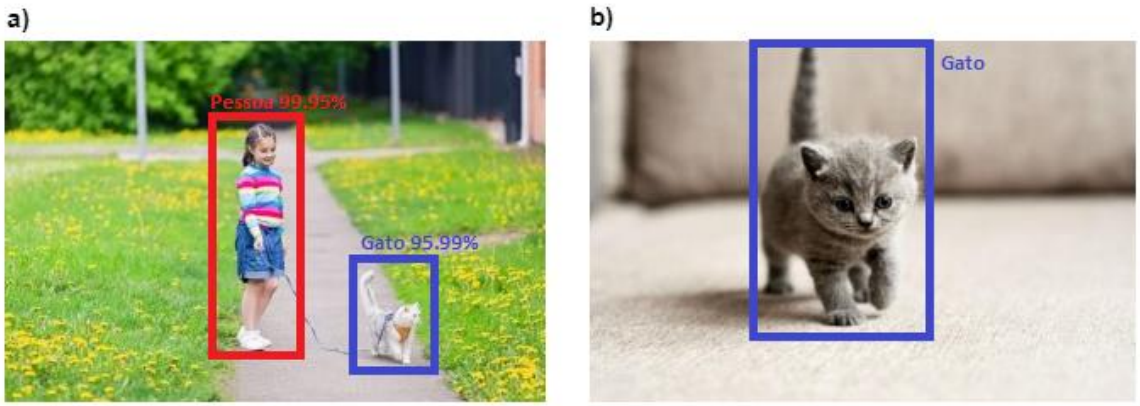


Figura 9 – YOLO: a) deteção e localização na imagem dos objetos “gato” e “pessoa”; b) detenção e localização na imagem do objeto “gato”

Com este algoritmo é possível detetar um objeto numa imagem, bem como saber o grau de precisão do mesmo.

2.4.1 Versões YOLO

Nos anos posteriores à publicação do algoritmo YOLO em 2015, foram publicadas versões mais avançadas deste algoritmo, nomeadamente:

— **YOLOv2** – Lançada em 2017, nesta versão conseguiram-se melhorias significativas ao nível da resolução da classificação, sendo esta mais precisa e mais rápida (Redmon & Farhadi, 2017). Esta versão também apresenta melhorias nas caixas delimitadoras;

— **YOLOv3** – Lançada em 2018, esta versão apresenta melhorias na eficácia da previsão (Redmon & Farhadi, 2018). Esta versão também solucionou o problema do reconhecimento de objetos de menor dimensão, com a previsão sendo efetuada em 3 níveis diferentes de granularidade. Em contrapartida, o uso de 3 níveis de granularidade conduziu a uma diminuição da rapidez do algoritmo;

— **YOLOv4** – Trata-se de uma versão mais rápida e mais precisa comparativamente ao YOLOv2 e YOLOv3. Esta versão destaca-se pela melhoria na velocidade de inerência, pela precisão e pela otimização do uso da memória de unidade de processamento gráfica (GPU) (Bochkovskiy et al., 2020);

— **YOLOv5** – Lançada em maio de 2021, trata-se da versão mais recente do YOLO (Zhu et al., 2021). Esta versão foi desenvolvida na framework *Pythorch*. É cerca de 90% mais pequena, em tamanho, comparativamente ao YOLOv4. O YOLOv5, e apresenta uma precisão bastante semelhante à versão anterior.

2.4.2 Modo de funcionamento

A deteção de objetos com recurso ao algoritmo YOLO baseia-se em três técnicas fundamentais (Figura 10):

— ***Residual blocks*** – O algoritmo divide a imagem em regiões de *grades* ou quadrados de igual dimensão e prevê a caixa delimitadora de cada região da *grade*;

— ***Bounding box regression*** – Os objetos em cada quadrado são destacados por uma caixa delimitadora ou *bounding box* e cada caixa possui as seguintes propriedades: peso, altura, classe e centro. Em cada caixa é apresentado o grau de confiança do objeto constar na caixa delimitadora. Em simultâneo o algoritmo prevê a probabilidade da classe;

— **Interseção sobre União (IoU)** – É uma medida baseada no índice Jaccard, estatística utilizada para medir a similaridade e diversidade de conjuntos de amostras, que avalia a sobreposição entre duas caixas delimitadoras. De seguida, a probabilidade da caixa e da classe são combinadas e o objeto é encontrado. Finalmente são mantidas as interseções cujo grau de confiança seja superior a um valor que, por defeito, é de 30%. Este valor é denominado de *threshold*.

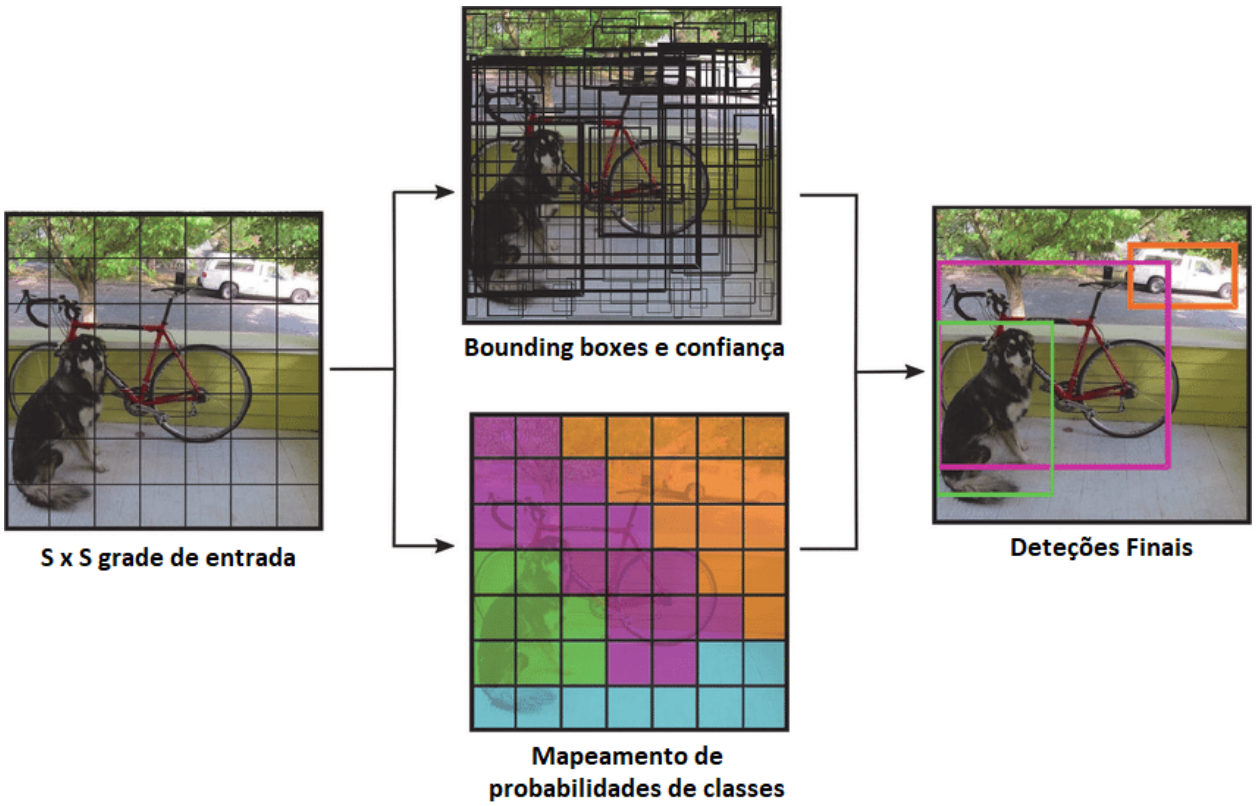


Figura 10 – Processamento de uma imagem com a rede YOLO (adaptado Santiago Teles de Menezes et al., 2020)

Depois de processada uma imagem, cada caixa do YOLO, é possível obter os atributos:

— **Coordenadas**: localização da caixa delimitadora;

— **Precisão**: probabilidade do objeto detetado.

2.4.3 Mean Average Precision (mAP), Average Precision (AP) e Loss

A principal métrica que é utilizada para avaliação de desempenho de diferentes modelos detetores de objetos, tais como R-CNN, SSD, *EfficientDet* e YOLO, é a *Mean Average Precision* (mAP). A mAP compara a caixa delimitadora verdadeira com a caixa detetada prevista pelo modelo e retorna uma pontuação. Quanto maior é a percentagem de retorno, mais preciso é o modelo. A mAP é tem como base os seguintes conceitos: matriz confusão, precisão e revocação e Interseção sobre União (IoU).

A matriz de confusão, apresentada na Tabela 2, indica a contagem dos valores previstos e reais. Para a criação desta matriz são necessários quatro atributos:

— Verdadeiro positivo (VP) – indica o número de exemplos positivos classificados com precisão;

— **Verdadeiro negativo (VN)** – indica o número de exemplos negativos classificados com precisão;

— **Falso positivo (FP)** – indica o número de exemplos negativos reais classificados como positivos;

— **Falso negativo (FN)** – indica o número de exemplos positivos reais classificados como negativos.

Tabela 2 – Matriz de confusão

		Classificação Prevista	
		Negativo	Positivo
Classificação Real	Negativo	VN	FP
	Positivo	FN	VP

A precisão ou *precision* calcula a precisão das previsões. A partir da matriz de confusão, a precisão é calculada pela divisão dos VP com a soma dos VP e os FP, equação (1)

$$Precisão = \frac{VP}{VP + FP} \tag{1}$$

A revocação ou *Recall* é a divisão dos VP com a soma dos VP e FN, equação (2).

$$Revocação = \frac{VP}{VP + FN} \tag{2}$$

A Interseção sobre União (IoU), ilustrada na Figura 11, avalia a sobreposição entre dois limites e é utilizado para medir qual a porção da área prevista se sobrepõe à verdadeira. Por outras palavras, a IoU indica a sobreposição das coordenadas da caixa delimitadora prevista em relação à caixa verdadeira. A IoU é determinada através da divisão entre a área sobreposta e a união das duas áreas.

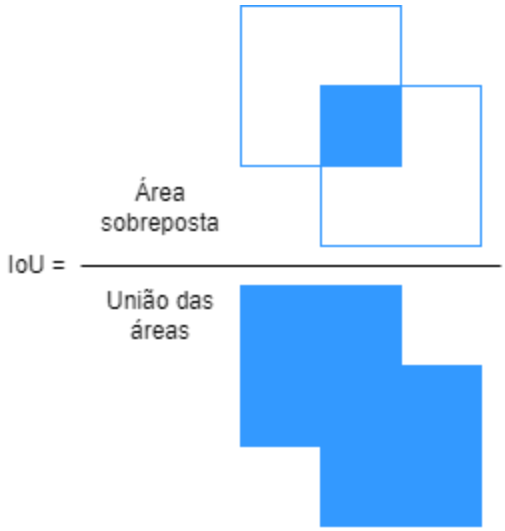


Figura 11 – Intereseção sobre União (IoU)

A curva *Precision* x *Recall* avalia a performance de um detetor de objeto em função da confiança para cada classe de objeto. Um detetor de objeto de uma determinada classe deve permanecer com alta precisão à medida que a revocação aumenta, isto é, com a variação do grau de confiança ou *threshold*, a precisão e a revocação ainda serão altas. Outra forma de se avaliar é através da verificação dos FP, isto é, quanto menor os FP maior a precisão e quanto menor os FN maior é a revocação. Por norma, a curva *Precision* x *Recall* tende a iniciar com valores de alta precisão diminuindo à medida que a *recall* aumenta.

A *Average Precision* (AP) é outra métrica utilizada para comparar a avaliação de desempenho dos detetores de objetos e corresponde à área acima da curva *Precision* x *Recall*.

Durante o treino de uma rede com o algoritmo YOLO, é possível a obtenção dos valores mAP, isto é, são obtidos os valores de AP para cada objeto quando o IoU é superior a 50%. Após o treino de uma rede, é possível obter o *Loss*, que corresponde à perda do treino, isto é, a perda do IoU. O *Loss* traduz quão bem o algoritmo pode localizar o centro de um objeto e quão bem a caixa delimitadora prevista cobre um objeto.

2.4.4 Microsoft Common Objects in Context (MS COCO)

O *dataset Microsoft Common Objects in Context* (MS COCO), publicado pela Microsoft em 2014 (Lin et al., 2014), é um conjunto alargado de dados de deteção, segmentação, deteção de pontos-chave e *label* de objetos em grande escala. Este *dataset* é frequentemente utilizado em testes em tempo real e para avaliação de desempenho de diferentes modelos detetores de objetos.

No lançamento do YOLOv4 foi realizado um estudo comparativo do desempenho do YOLOv4 em relação a outros sistemas de deteção de objetos utilizando o *dataset* MS COCO. Uma das principais conclusões do estudo foi que, o YOLOv4 demonstrou uma maior precisão (AP - *average precision*) e velocidade (FPS – *frames* por segundo) duas vezes superior comparativamente ao EfficentDet (Figura 12). O YOLOv4 apresentou também uma melhoria da velocidade e precisão face ao YOLOv3 (Figura 12).

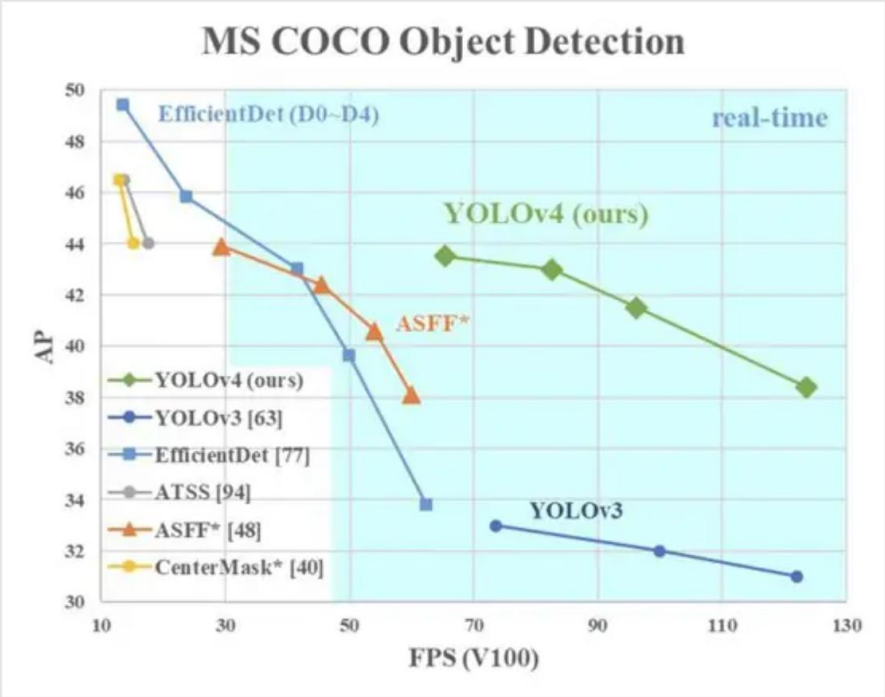


Figura 12 – Comparação entre vários sistemas de deteção de objetos (Bochkovskiy et al., 2020)

No âmbito deste trabalho, o algoritmo para reconhecimento de objetos utilizado é o YOLO na versão 4, uma vez que se trata de um algoritmo rápido, eficaz e que não necessita de uma grande quantidade de dados para treino da rede. Aquando do início do desenvolvimento deste trabalho, tratava-se da versão mais recente lançada do YOLO.

3. MATERIAIS E METODOLOGIA

Neste capítulo, encontram-se descritos os materiais utilizados e os procedimentos gerais adotados no presente trabalho, juntamente com detalhes acerca do *hardware* e *software* utilizados. Encontram-se também descritos os métodos necessários para o treino de uma rede neuronal para deteção de objetos a partir de uma imagem.

3.1 Hardware e Software

O hardware utilizado no presente trabalho é composto por um robô humanoide, o Zeno R-50 da *Robokind Robotic Platform*, e por duas câmaras externas para deteção dos objetos.

3.1.1 Robô Zeno R50 Robokind (ZECA)

O robô Zeno R50 *Robokind robotic platform* (Figura 13), designado como ZECA (*Zeno Engaging Children with Autism*), assemelha-se fisicamente a uma criança do sexo masculino. Possui capacidade de expressar emoções e emitir sons, isto é, consegue falar e gesticular. A face deste robô é construída com *Frubber*, que é um material semelhante à pele humana, tanto no aspeto visual como ao toque. Na Tabela 3 encontra-se sintetizado as principais características do robô Zeca.

O robô ZECA possui várias bibliotecas disponíveis em JAVA, que permitem estabelecer a comunicação entre o computador e o robô. Estas bibliotecas podem ser utilizadas para o envio de instruções ao robô, como a fala e os gestos. As bibliotecas podem ser obtidas no repositório Maven através do link *https://mvnrepository.com/artifact/org.robokind*.

Figura 13 – Zeno R-50

Tabela 3 – Principais características do robô Zeca

Características Físicas		
	Peso	5.7 kg
	Altura	69 cm
Multimédia		
	1 coluna de som	
	3 microfones	
	2 câmaras (CMOS Digital Hi-Def (720p))	
Conectividade		
	Wi-Fi (IEEE 802.11a/b/g/n)	
	Conexão Ethernet (Gigabit/100/10)	
	Porta HDMI	
Hardware		
	Motherboard	Intel Atom x86 Z530 1,6 GHz
	Memoria RAM	1GB DDR2
	Outros	Memória Flash 4 GB
		16 GB micro SD
	Sensores	
		1 Giroscópio
		1 Acelerómetro
		1 Bussola
Software		
	Sistema Operativo	Ubuntu Linux 32 bit
	Linguagem programação	JAVA

3.1.2 Suporte (raquete), formas e cores utilizadas no treino das redes

Para o treino da rede foram selecionadas três cores, o verde, o vermelho e o azul, e três formas geométricas, o triângulo, o círculo e o quadrado. Estes elementos foram impressos e, posteriormente, colados numa cartolina, de forma a aumentar a espessura dos mesmos funcionando assim como cartões. Durante o treino das redes, estes cartões foram posicionados numa raquete de *ping pong* (Figura 14 e Tabela 4), que serviu como pega e suporte para os mesmos.

Figura 14 – Raquete *ping pong* (suporte)

Tabela 4 – Raquete *ping pong* com objetos para deteção e classificação: a) objeto; b) objeto posicionado a mostrar ao robô

(a)	(b)
	
	
	
	
	
	

3.1.3 Câmaras

No âmbito deste trabalho foram utilizadas duas câmaras com propriedades distintas, nomeadamente:

— Para fotografar objetos (Tabela 4) para treino da rede: telemóvel Samsung Galaxy S8, com câmara com resolução de 4032 x 3024 pixels (Figura 15)

Figura 15 – Camara utilizada para fotografar objetos para treino das redes

— Para deteção de objetos tem tempo real: HP Webcam HD 4310, com resolução de 1920x1080 pixels (Figura 16).

Figura 16 – Camara utilizada para deteção de objetos em tempo real

3.2 Google Colab

O *Google Colab* é um serviço da Google na *cloud* que possibilita a criação de um *notebook*, onde é possível executar sequências de código e/ou programas e estabelecer a conexão com o *Google Drive*. O sistema operativo utilizado no *Google Colab* é o Linux, o que permite compilar o YOLO, e o código a executar é normalmente escrito na linguagem *Python*. Com o *Google Colab* também é possível utilizar a unidade de processamento gráfica (GPU) sem que seja necessário a realização de quaisquer configurações de sistema. No que diz respeito ao presente trabalho, o serviço *Google Colab* permite a compilação e posterior execução do YOLO para treino de uma rede neuronal de forma gratuita, com a limitação do tempo de utilização do serviço.

3.3 YOLO – Configurações/Parâmetros

O YOLO pode ser descarregado através do link *https://github.com/AlexeyAB/darknet* onde contém as instruções/ferramentas necessárias para a compilação do YOLO. Para a compilação do YOLO, existe um ficheiro, designado como *makefile*, para o efeito. Primeiramente, neste ficheiro, existem vários parâmetros de configuração que devem ser alterados para o valor 1 indicados na Tabela 5.

Tabela 5 – Parâmetros de configuração do ficheiro Makefile de compilação do YOLO

Parâmetro	Descrição
GPU	Compilação com *Compute Unified Device Architecture* (CUDA) utilizando da GPU
CUDNN	Aceleração do treino da rede utilizando a GPU
OPENCV	Compilação com *OpenCV* 4.x/3.x/2.4 permitindo a deteção de objetos em vídeos ou webcams

De seguida, para a compilação YOLO é necessário executar o comando *make* para gerar a aplicação YOLO.

Após a compilação do YOLO é necessário proceder ao treino das redes. Para o treino das redes utilizado a aplicação YOLO anteriormente compilada, é necessário passar como argumento os seguintes ficheiros:

— Ficheiro com extensão *".data"* que contém as informações sobre (Figura 17):

a. O número de classes da rede, linha 1 da Figura 17;
b. A designação do ficheiro com a lista de imagens para treino, linha 2 da Figura 17;
c. A designação do ficheiro com a lista de imagens para teste, linha 3 da Figura 17;
d. A designação do ficheiro com a lista do nome das classes, linha 4 da Figura 17;
e. A diretoria onde vai ser guardado o *dataset* na linha 5 da Figura 17.

```
1  classes = 3
2  train = data/train.txt
3  valid = data/test.txt
4  names = data/classes.names
5  backup = train
```

Figura 17 – Ficheiro *".data"*

— Ficheiro de configuração da rede neuronal com extensão *".config"*. Este ficheiro de configuração contém a estrutura da rede neuronal. Devido ao limite no uso de recursos de forma gratuita na plataforma *Google Colab* utilizou-se, para treino das redes, o ficheiro de configuração *yolov4-tiny-custom.cfg* disponibilizado no repositório do YOLO que e possibilita uma maior rapidez no treino.

Existem várias secções no ficheiro de configuração (Figura 18) que têm de ser alteradas em função da rede que pretende treinar. As alterações realizadas encontram-se mostradas na Tabela 6.

```
[net]
#Testing
#batch=1
#subdivisions=1
#Training
batch=32
subdivisions=12
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation=1.5
exposure=1.5
hue=.1

learning_rate=0.00261
burn_in=1000
max_batches=6000
policy=steps
steps=4800,5400
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=2
pad=1
activation=leaky
```

```
[yolo]
mask=3,4,5
anchors=10,14,23,27,37,58,81,82,135,169,344,319
classes=3
num=6
jitter=.3
scale_x_y=1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh=.7
```

Figura 18 – Ficheiro de configuração da rede neuronal do YOLO

Tabela 6 – Secções e parâmetros alterados no ficheiro configuração

Secção	Parâmetro	Descrição
net	*batch*	Número de imagens por iteração
	subdivisions	Número de fragmentos em que o *batch* é dividido para a memória da GPU.
	width	Largura da imagem a redimensionar
	height	Altura da imagem a redimensionar
	max_batches	2000 * número de classes
	steps	80% a 90% do número de *batches*
yolo	*classes*	Número de classes
convolutional	*filters*	8*número de classes (todas as secções anteriores da yolo)

Por defeito, o valor do parâmetro "*batch"* é 64, representado na linha 6 da Figura 18, mas, devido a limitação no uso de recursos da plataforma *Google Colab*, o valor deste parâmetro foi alterado para 32.

Neste ficheiro existem várias secções "*yolo"*, exemplificadas nas linhas 17 a 227, e para cada uma delas é necessário alterar o parâmetro *classes*. Anteriormente à secção "*yolo"* existe a secção "*convolutional"*, representado na linha 25 da Figura 18, onde deve ser alterado o parâmetro "*filters"*, ou seja, na linha 27 da Figura 18

Para o treino da rede, para além dos dois ficheiros referidos anteriormente, é ainda possível passar por argumento um terceiro ficheiro, uma rede pré-treinada. Esta rede pré-treinada é uma rede que foi previamente treinada com um grande conjunto de dados. Para isso é necessário descarregar o ficheiro *"yolov4-tiny.conv.29"*, que contém as primeiras 29 camadas convolucionais do *yolov4-tiny*.

Para além dos argumentos referidos acima, é necessário passar o argumento *"-dont_show"* para desativar o *OpenCV* de forma a não mostrar os resultados de inferência, uma vez que o trabalho está a ser executado na plataforma *Google Colab*.

Por fim, também é possível, à medida que a rede é treinada, passando como argumento o *"-map"* obter o gráfico *Loss x Precision*. Este gráfico, Figura 19, é guardado na diretoria onde é executado o YOLO com a denominação de *"data.png"*.

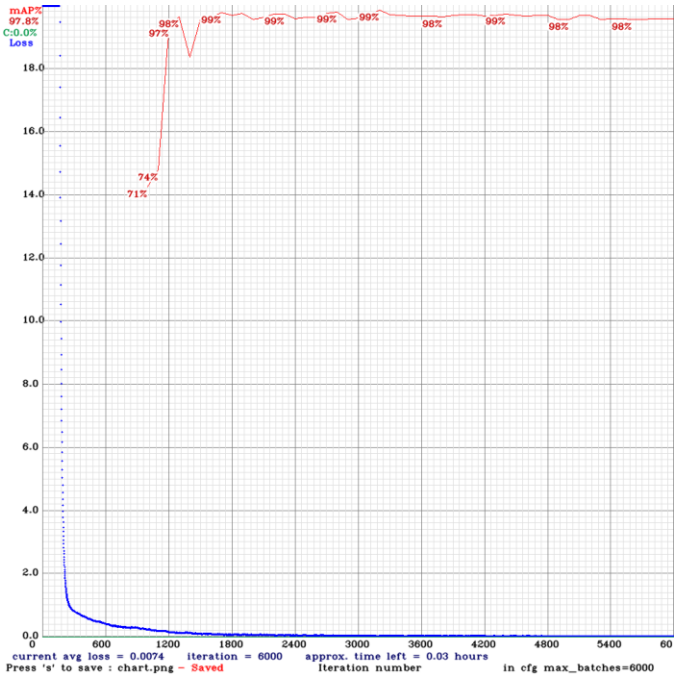


Figura 19 – Gráfico de *Loss* x *Precision* do treino de uma rede com o YOLO

Este ficheiro gerado pode ser guardado para futura comparação entre outras redes treinadas.

3.4 Label das imagens

Umas das tarefas mais importantes para o treino de uma rede neuronal é a *label* ou a etiquetagem das imagens. Para isso, é necessário obter um número significativo de fotografias dos objetos que se pretende treinar a rede. As imagens devem ser etiquetadas, isto é, para cada imagem deve existir um ficheiro com as coordenadas do objeto, a altura e a largura. As imagens e suas etiquetas correspondentes são utilizadas para treino e para o teste da rede. Para a etiquetagem das imagens utilizou-se aplicação *Yolo_Label* disponível no link *https://github.com/developer0hye/Yolo_Label*. Com esta aplicação, a etiquetagem das imagens,

é feito tal como exemplificado na Figura 20, implica o posicionamento de uma *bounding box* à volta do objeto a classificar e resulta num ficheiro com extensão *".txt"* apresentado na Figura 21.

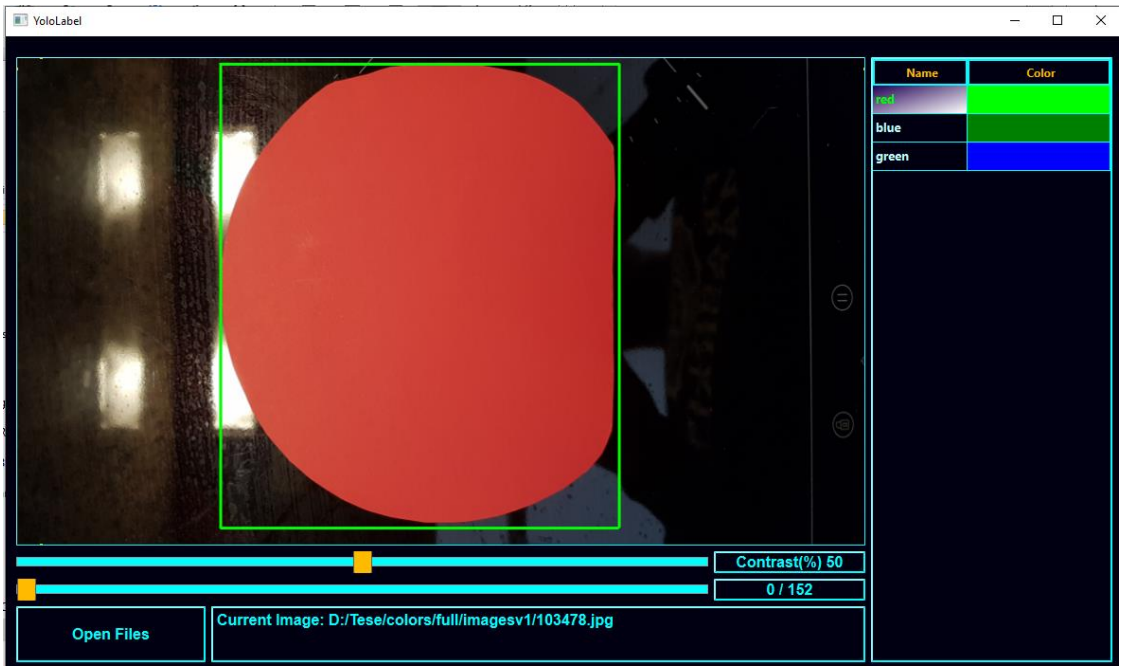


Figura 20 – Processo de etiquetagem de uma imagem

```
1 0 0.474826 0.487755 0.470486 0.954082
```

Figura 21 – Exemplo de um ficheiro resultante da etiquetagem de uma imagem

Esta aplicação necessita também de um ficheiro *"classes.txt"* que possui uma lista dos objetos a etiquetar. Este ficheiro deve ser carregado juntamente com as imagens.

4. DESENVOLVIMENTO DO TABALHO

Neste Capítulo encontram-se descritos as aplicações e os automatismos desenvolvidos no âmbito deste trabalho para treino de uma rede neuronal. É também mostrado uma visão global da implementação de todo o sistema, a interface gráfica e a arquitetura do software desenvolvido.

4.1 Prova de Conceito

Para o desenvolvimento das tarefas de ensino de cores e formas geométricas, numa primeira fase foi desenvolvido um script em *Python* para o reconhecimento de raquetes de *ping pong*. Este script foi utilizado como prova de conceito do algoritmo YOLO.

Para deteção das raquetes de *ping pong*, procedeu-se ao treino de uma rede neuronal adotando a seguinte metodologia:

1- Descarregamento de 125 fotografias de raquetes de *ping pong* disponíveis na internet (Figura 22);

2- Para o treinamento da rede de YOLO, foi necessário etiquetar as imagens descarregadas com a localização de cada raquete de *ping pong*. Para a etiquetagem das imagens foi utilizado o software *Yolo_label*;

3- Complicação do YOLO e treino da rede.

Figura 22 – Imagem da raquete de *ping pong* extraída da internet

Nesta prova de conceito foi possível, com o *script* desenvolvido, a deteção das raquetes de *ping pong* em vídeo e a olho nu com uma percentagem de certeza de 80 a 90%. Adicionalmente, aquando da criação desta rede, verificou-se a existência de diversos passos manuais que poderiam ser automatizados. Nesse sentido, foram desenvolvidos alguns *scripts*, que se encontram descritos nos Capítulos 4.3 e 4.4, de forma a automatizar os passos manuais.

4.2 Criação de Datasets de Cores e Figuras Geométricas

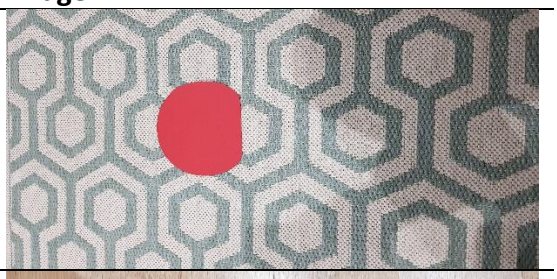
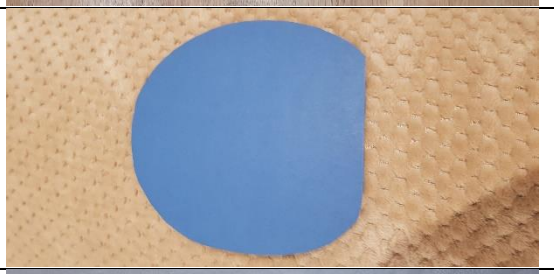
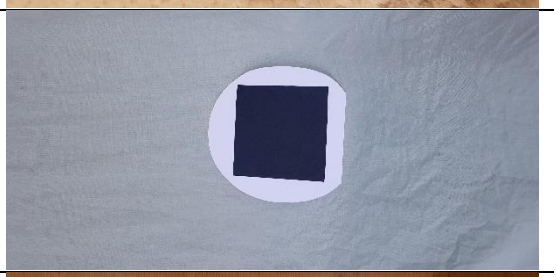
Para o treino de uma rede é necessário a existência de um *dataset* constituído por um conjunto de imagens etiquetadas. Para a criação dos *datasets* foram utilizados os seguintes 6 objetos:

— Cores: vermelho, azul e verde;

— Figuras geométricas: triângulo, círculo e quadrado.

Com vista a diversificar as imagens utilizadas, os objetos foram fotografados com diferentes cenários de fundo e com distâncias compreendidas entre 0.5 metros a 1 metro (Tabela 7). Para cada objeto foram tiradas 200 fotografias, o que perfaz um total $\approx$ 1200 fotografias.

Tabela 7 – Exemplos de fotografias tiradas para criação dos *datasets*

Objeto	**Imagem**
Vermelho	
Verde	
Azul	
Quadrado	
Triângulo	
Círculo	

4.3 Automatização para criação de redes neuronais

Com o objetivo de automatizar a criação das redes neuronais a partir do YOLO foram desenvolvidos dois automatismos, nomeadamente a geração automática dos ficheiros necessários e a criação de um *notebook* com as sequências de instruções para treino das redes.

As fotografias retiradas aos objetos foram carregadas no *Google Drive.* A partir do *Google Colab* é possível estabelecer a conexão com o Google Drive e utilizar as fotografias carregadas no treino da rede.

Para uma melhor organização das pastas na raiz do Google Drive, foi necessário a criação de uma estrutura de pastas (ver Figura 23):

— Pasta principal, designada *Dataset Generator*;

— Dentro da pasta principal foram criadas as pastas *colors*, *shapes*, *scripts* e *darknet*. A pasta *scripts* contém os *scripts* para a criação do *dataset*. A pasta *darknet* contém a aplicação do YOLO descarregada e compilada;

— Dentro das pastas *colors* e *shapes* foi criada a pasta *images,* onde são carregadas as fotografias etiquetadas. Foram também criadas as pastas *data*, onde são gerados os ficheiros de configuração a utilizar pelo YOLO, e a pasta *train,* onde são geradas as redes treinadas pelo YOLO;

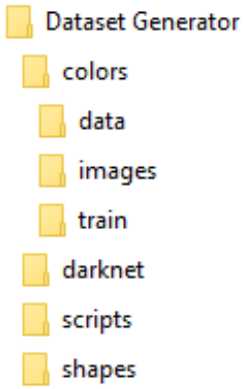


Figura 23 – Organização das pastas na raiz do Google Drive

Como referido no Capítulo 3.4, para a treino de uma rede a partir da aplicação do YOLO, é necessário a existência dos 4 ficheiros que se encontram listados na Tabela 8.

Tabela 8 – Ficheiros necessários para treino de uma rede a partir da aplicação YOLO

Descrição	**Denominação do ficheiro**
Ficheiro de configuração da rede	*config.config*
Ficheiro que contém as designações das classes	*classes.names*
Ficheiro que contém a designação das imagens para treino	*train.txt*
Ficheiro que contém a designação das imagens para teste	*test.txt*
Ficheiro que contém a designação dos ficheiros de treino, teste e número de classes	*data.data*

Para a criação dos ficheiros listados na Tabela 8 foi desenvolvido um *script* em linguagem Python, denominado "create-files.py". O *script* create-files.py foi introduzido na pasta *scripts* para posterior execução e criação dos ficheiros listados na Tabela 8. O *script* create-files.py recebe como argumento a designação da pasta referente à rede a treinar, por exemplo, *colors*. Este *script* pode ser consultado no link https://github.com/davidmiguelalves/teaching-system/blob/main/Dataset%20Generator/scripts/create-files.py e a sua execução segue a seguinte sequência:

1. Dentro da pasta com a designação passada como argumento, são criadas as pastas "*train"* e "*data"*;
2. Geração de uma lista com as imagens existentes na pasta "*images"*;
3. Na pasta *data*, é gerado um ficheiro designado *"test.txt,* com 30% da lista referida no Ponto 2;
4. Na pasta *data*, é gerado um ficheiro denominado *"train.txt",* com 70% da lista referida no Ponto 2;

5. É copiado para a pasta *data* o ficheiro *"classes.txt"* existente na pasta "*images",* renomeando o ficheiro para *"classes.names";*
6. Geração de um ficheiro denominado por *"data.data"*;
7. É copiado o ficheiro de configuração *"yolov4-tiny-custom.cfg"* para a pasta *data*. O ficheiro *"yolov4-tiny-custom.cfg"* é renomeado para "*config.config"* e são alteradas as configurações referidas no Capítulo 3.4.

Depois da criação do *script*, foi criado um *notebook* no *Google Colab* denominado *"generate_dataset.ipynb".* Tal como referido anteriormente, este *notebook* contém a sequência de todas as instruções e passos necessários para a geração de uma rede neuronal. Este *notebook* pode ser consultado no link *https://github.com/davidmiquelalves/teaching-system/blob/main/Dataset%20Generator/scripts/generate_dataset.ipynb*. O *notebook* foi desenvolvido para executar 7 passos, de acordo com a ordem das seguintes instruções:

1. Estabelecer a conexão com o *Google Drive* para aceder à estrutura de pastas, imagens e ficheiros para treino da rede

```
# connect to google drive
from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/MyDrive/Dataset Generator'
```

Lista 1 – Conexão ao *Google Drive*

2. Descarregamento da aplicação do YOLO

```
# get YOLO
!git clone https://github.com/AlexeyAB/darknet.git
```

Lista 2 – Descarregar YOLO

3. Alteração do conteúdo do ficheiro *makefile* e compilação do YOLO

```
# change makefile parameters and compile
%cd '/content/drive/MyDrive/Dataset Generator/darknet'
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!make
```

Lista 3 – Alteração ficheiro *makefile*

4. Dar permissões à aplicação do YOLO

```
# give YOLO permssions
!chmod +x '/content/drive/MyDrive/Dataset Generator/darknet/darknet'
```

Lista 4 – Adicionar permissões ao YOLO

5. Descarregamento da rede pré-treinada

```
# download the last released yolov4-tiny ConvNet weights
%cd '/content/drive/MyDrive/Dataset Generator/darknet'
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yol
o_v4_pre/yolov4-tiny.conv.29
```

Lista 5 – Descarregar rede pré-treinada

6. Criação dos ficheiros necessários para o treino da rede com recurso ao *script* descrito anteriormente

```
# create files (config.cfg\names.names\train.txt\test.txt\data.data)
# change datasetname
%cd '/content/drive/MyDrive/Dataset Generator/scripts'
datasetname = "CHANGE_HERE" #CHANGE_HERE
!python create-files.py --dataset $datasetname
```

Lista 6 – Criação de ficheiros para o YOLO

7. Treino da rede neuronal

```
# train network
# change datasetname
%cd '/content/drive/MyDrive/Dataset Generator/'
datasetname = "CHANGE_HERE" #CHANGE_HERE
!darknet/darknet detector train $datasetname/data/data.data $dataset-
name/data/config.config darknet/yolov4-tiny.conv.29 -dont_show -map
```

Lista 7 – Treinar rede neuronal

Uma vez que este *notebook* tem o como finalidade o treino de mais do que uma rede neuronal, os passos 2, 3 e 4 são apenas executados uma única vez, visto que a aplicação do YOLO e rede pré-treinada já se encontram descarregadas e aplicação do YOLO compilada.

Relativamente ao passo 7, quando este é executado à medida que o treino da rede é realizado, são criados, a cada 1000 iterações, ficheiros *".weights"* na pasta "*train*", contendo no nome do ficheiro o número da iteração correspondente. Quando o treino da rede é finalizada, é criado um ficheiro *"_best.weights"*, que contém a melhor percentagem de mAP de todos os treinos realizados. Para a obtenção de mais dados para além do mAP é possível executar um último passo

```
# calculate mAP
# change datasetname
%cd '/content/drive/MyDrive/Dataset Generator/'
datasetname = "CHANGE_HERE" #CHANGE_HERE
!darknet/darknet detector map $datasetname/data/data.data $dataset-
name/data/config.config $datasetname/train/config_best.weights
```

Lista 8 – Calcular valores de mAP da rede treinada

Este passo possibilita a obtenção do valor de mAP global, dos valores de AP para cada objeto, do valor da IoU e do valor de precisão (Figura 24). Estes dados servem de apoio para a comparação da precisão entre todas as redes treinadas.

```
class_id = 0, name = red, ap = 98.31%           (TP = 58, FP = 0)
class_id = 1, name = blue, ap = 94.99%          (TP = 55, FP = 2)
class_id = 2, name = green, ap = 100.00%        (TP = 65, FP = 1)

 for conf_thresh = 0.25, precision = 0.98, recall = 0.98, F1-score = 0.98
 for conf_thresh = 0.25, TP = 178, FP = 3, FN = 3, average IoU = 91.03 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.977642, or 97.76 %
```

Figura 24 – Resultado da percentagem de mAP global, AP de cada objeto, IoU e precisão de uma rede treinada

Em síntese, com o desenvolvimento do *notebook* é possível o treinamento de qualquer rede neuronal. Para tal, é necessário a criação de uma pasta no *Google Drive* com a estrutura de pastas acima referida, a realização do *upload* das imagens etiquetadas e a execução de todos os passos do *notebook.* Depois da execução do *notebook*, a rede treinada ficará disponível na pasta *"train"*, num ficheiro denominado *"config_best.weights"*.

4.3.1 Geração das redes neuronais

Para a obtenção do número de imagens necessárias para uma boa deteção dos objetos, realizou-se o treino das redes neuronais com 50, 100, 150 e 200 fotografias para cada forma geométrica (triângulo, quadrado e círculo) e cor (vermelho, verde e azul), com pré-treino e sem pré-treino, o que perfaz um total de 16 treinos. Após cada treino, o YOLO procedeu à geração dos gráficos *Loss* x *Precision*, que serão discutidos no Capítulo 5.

4.3.2 Scripts para a deteção de objetos numa imagem

Após a criação das redes neuronais, foram desenvolvidos dois *scripts* em linguagem *Python* com o objetivo de verificar a correta deteção dos objetos:

— Um *script* denominado por "*camara_detect.py"* que, em tempo real e a partir de uma câmara, deteta os objetos presentes numa imagem;

— Um *script* denominado por "*save_video.py*" que, em vídeo, deteta os objetos presentes em cada *frame*. Os resultados das deteções são guardados num ficheiro "*csv*" com a designação do objeto detetado, a precisão e a *frame* correspondente. O segundo *script* permite a comparação dos resultados obtidos nas redes treinadas. Este *script* gera um vídeo a partir do vídeo original posicionando no objeto detetado uma *bounding box,* com a precisão e o nome do objeto detetado.

Estes *scripts* foram desenvolvidos em *Python*, pois existe uma biblioteca denominada "cv2", *OpenCV*, onde é possível a leitura de imagem a partir de uma câmara ou vídeo. Nesta biblioteca existe um módulo denominado "*dnn*", *Deep Neural Networks*, onde é possível a leitura de uma rede neuronal treinada pelo YOLO e a deteção de objetos numa imagem:

```
net = cv2.dnn.readNetFromDarknet('config.config', 'weights.weights')
model = cv2.dnn_DetectionModel(net)
model.setInputParams(scale = 1 / 255, size = (416, 416), swapRB = True)
ret, frame = camera.read()
frame = cv2.resize(frame, (416, 416), interpolation = cv2.INTER_AREA)
classIds, scores, boxes = model.detect(frame, confThreshold = 0.6, nmsThreshold = 0.04)
```

Lista 9 – Deteção de objetos a partir de uma rede treinada com o YOLO

Estes *scripts* podem ser consultados no link *https://github.com/davidmiguelalves/teaching-system/tree/main/ObjectDetectionAPI*.

4.3.3 Validações das redes neuronais

Após os primeiros treinos de rede foram realizados testes preliminares de forma a avaliar a precisão da deteção dos objetos. Nestes testes com recurso a aplicação de deteção de objetos

em tempo real, verificou-se que a precisão era reduzida, na ordem dos 50% a 60%, o que estava abaixo do expectável. Com a finalidade melhorar a validação da etiquetagem das imagens e de aumentar a precisão da deteção dos objetos, foi desenvolvido um automatismo em *Powershell,* pode ser consultado no link *https://github.com/davidmiguelalves/teaching-system/blob/main/Dataset%20Generator/scripts/verifyimages.ps1*, onde cada imagem etiquetada é colocada numa pasta intitulada com a mesma designação da etiqueta. Por exemplo, as imagens etiquetadas a vermelho foram colocadas numa pasta intitulada como *"vermelho"*. Desta forma antes do treino de uma rede é possível verificar, visualmente, verificar se todas as imagens foram etiquetadas corretamente.

Após a realização destes testes preliminares verificou-se também a existência de imagens na horizontal e na vertical, fazendo com que, após o redimensionamento do YOLO, as imagens fossem mal etiquetadas. Para contornar esse problema, todas as imagens foram uniformizadas, isto é, as imagens foram posicionadas na horizontal e redimensionadas para 416x416. Por fim e de forma a aumentar a precisão da deteção do objeto, optou-se por realizar a etiquetagem ao objeto a treinar em vez do objeto inteiro (raquete + cartões com o objeto) reduzindo assim a área de possível erro.

Nestes testes, na rede neuronal deteção de formas geométricas foram detetadas algumas anomalias, nomeadamente, a deteção do triângulo quando a figura mostrada era um quadrado rodado. Posto isso, foi validado que as fotografias aos objetos foram tinham sempre a mesma orientação, consequentemente, para validação desta anomalia, em fase de testes foram tiradas novas 40 fotografias de cada objeto de diferentes ângulos e treinada uma nova rede. Com isto obtiveram-se resultados mais satisfatórios, apesar de ainda haver inconsistências no quadrado. Finalmente, foi treinada uma rede nova, com fotografias retiradas de vários ângulos, com 50, 100, 150 e 200 imagens por objeto.

4.4 API ZECA

Para o presente trabalho foi desenvolvido uma API REST em JAVA. Esta API utiliza *Spring*, que é uma *framework open source* para o JAVA. Esta API tem por objetivo a uniformização de toda a comunicação realizada entre o robô ZECA e o computador, de forma que, qualquer *software* desenvolvido consiga enviar instruções para o robô ZECA executar. Por motivos de compatibilidade das bibliotecas, a API desenvolvida utiliza o *Java Development Kit* (JDK) na versão 8.

As possíveis instruções a serem enviadas para o robô ZECA são a emissão de fala e emissão de produção de gestos. Para o envio de ordens de instruções ao robot, é necessário estabelecer a comunicação com o mesmo através de um pedido de conexão.

Na API foram disponibilizados vários pedidos de requisição *Hypertext Transfer Protocol* (HTTP):

- ***Connect***: Este é um pedido do tipo *POST* que tem como objetivo estabelecer a conexão com o robô. O IP é enviado no corpo do pedido;
- ***Disconnect***: Este é um pedido do tipo *POST* que tem como objetivo estabelecer a desconexão do robô;
- ***Speak***: Este é um pedido do tipo *POST* que tem como objetivo o envio de instrução de fala para o robô falar. Essa frase é enviada no corpo do pedido;
- ***PlayAnimation***: Este é um pedido do tipo *POST*, em que, a partir de uma lista pré-definida, é enviado um comando para o robô executar animações. As animações disponíveis neste trabalho são: a) resposta correta, em que o ZECA sorri e agita os braços; b) resposta incorreta, em que o ZECA agita a cabeça e reproduz uma cara triste. O comando é enviado no corpo do pedido.
- ***Probe***: Este é um pedido do tipo *GET* em que é validado se a API está operacional.

No caso de alguns pedidos falharem, a API retorna uma resposta *Bad Request* a quem realiza o pedido de requisição. Uma das vantagens do desenvolvimento desta API é que a mesma funciona como módulo e de forma independente. Assim sendo, caso seja necessário substituir o robô ZECA por outro robô, é apenas necessário modificar os 5 pedidos acima listados ou, em alternativa, desenvolver uma API nova com estes 5 pedidos, mantendo assim todo o restante sistema operacional. A API desenvolvida pode ser consultada no link *https://github.com/davidmiguelalves/teaching-system/tree/main/ZecaAPI*.

4.5 API de Deteção de Objetos

Para o reconhecimento de objetos foi desenvolvida outra API desenvolvida na linguagem *Python*. Os objetivos principais desta API são: a) consultar as atividades de ensino disponíveis; b) para cada atividade de ensino consultar quais os objetos disponíveis; c) para cada atividade de ensino, é possível o envio de imagens para a API e a API retorna os objetos identificados.

Para deteção dos objetos são utilizadas as redes neuronais treinadas para o efeito. Esta API utiliza uma base de dados em ficheiro *".json"*, onde se encontra toda a informação de todas as atividades de ensino disponíveis. Para o funcionamento da API foram desenvolvidos quatro pedidos de requisição HTTP:

— ***Activities***: Este é um pedido do tipo *GET*, que tem como objetivo a obtenção de todas as atividades de ensino disponíveis, que neste caso são as cores e as figuras geométricas;

— ***Objects:*** Este é um pedido do tipo *GET*, em que dada uma atividade de ensino, que é passada no corpo do pedido, retorna a lista dos objetos disponíveis. Por exemplo, no caso da atividade de ensino das cores, é retornado a lista vermelho, verde e azul;

— ***Detection:*** Este é um pedido do tipo *PUT*, em que dada uma atividade de ensino e uma imagem, que é passada no corpo do pedido, deteta qual o objeto presente na imagem. Neste

pedido são retornadas as coordenadas do objeto detetado, a percentagem de deteção e qual o nome do objeto detetado;

— ***Probe:*** Este é um pedido do tipo *GET* que tem como objetivo validar se a API está operacional.

À semelhança da API referida anteriormente, esta API funciona como módulo e de forma independente. Assim sendo, é possível adicionar novas atividades de ensino sem que seja necessária a alteração da interface. Caso seja adicionada novas atividades de ensino, estas atividades ficam automaticamente disponíveis na interface gráfica.

Esta API pode ser consultada no link *https://github.com/davidmiguelalves/teaching-system/blob/main/ObjectDetectionAPI/api.py*.

4.6 Interface Gráfica

Para o utilizador ter a possibilidade de ordenação das sequências de instrução a enviar para o robô ZECA, foi desenvolvida a interface gráfica ilustrada na Figura 25. Esta interface foi desenvolvida na linguagem c# com *Windows Presentation Foundation* (WPF) e utiliza as APIs desenvolvidas anteriormente para o robô ZECA e para o reconhecimento de objetos.

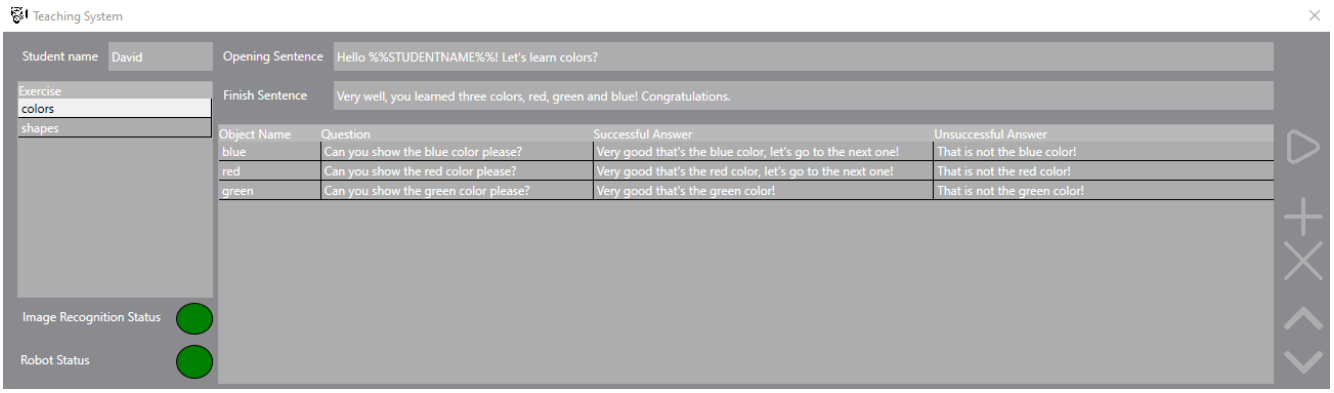


Figura 25 – Interface gráfica

Quando se inicia a interface gráfica, a interface faz um pedido à API de deteção de objetos para obter as atividades de ensino existentes. Após a obtenção de lista de atividades, o utilizador tem a possibilidade de criar sequências de instruções para o robô. As sequências de instruções são compostas por uma pergunta e 2 respostas, uma para o caso de sucesso e outra para o caso de insucesso, e essas instruções são referentes a cada objeto. A lista de instruções pode conter o mesmo objeto mais do que uma vez caso o utilizador o defina.

As listas de instruções, de cores e formas geométricas definidas pelo utilizador, são guardadas numa base de dados, num ficheiro com extensão *".xml"*. O utilizador também tem a possibilidade de editar as instruções e de trocar a ordem das mesmas nas listas. O utilizador também tem a possibilidade de adicionar o nome do aluno e de definir uma frase de iniciação e de finalização da atividade a ensinar.

O procedimento da interface gráfica que ocorre durante o ensino de uma atividade pode ser descrito da seguinte forma:

1. Conexão à câmara para obtenção da imagem;
2. Envio de um pedido à API ZECA para emissão da fala de iniciação do exercício;
3. Envio constantemente da imagem obtida em tempo real pela câmara externa para a API para deteção de objetos;
4. Para cada objeto definido na ordem de instruções na atividade de ensino: a) Envio de um pedido à API ZECA com a pergunta definida na instrução do objeto; b) caso seja detetado o objeto proposto, é enviado um pedido à API do ZECA com a fala de sucesso. Caso seja o objeto detetado não seja o pedido, é enviado para o robô a fala de insucesso
5. Envio da fala de instrução de finalização do exercício.

Durante o desenvolvimento da interface gráfica, e de forma a reduzir a percentagem de deteções inválidas, foi definido que um objeto só é detetado após 2 segundos.

A interface, desenvolvida em módulos, encontra-se ilustrada na Figura 26.

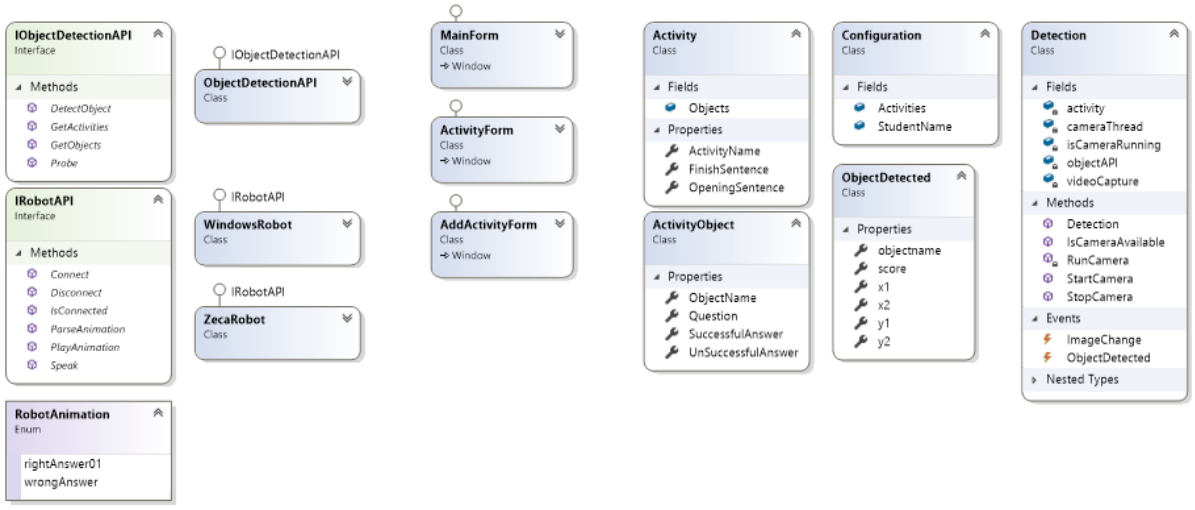


Figura 26 – Diagrama de classes da interface gráfica

O desenvolvimento da interface por módulos possibilita a alteração das API e do robô sem que seja necessário a alteração da interface gráfica total, sendo apenas necessário o desenvolvimento de classes com as Interfaces correspondentes. O desenvolvimento da interface gráfica pode ser consultado no link *https://github.com/davidmiguelalves/teaching-system/tree/main/ObjectDetectionAPP*.

4.7 Visão Geral do Sistema

O sistema global é composto por três componentes que foram desenvolvidas no âmbito da presente dissertação e detalhadas nos Capítulos anteriores, nomeadamente, por duas API e uma interface gráfica. A API ZECA é responsável pela uniformização da comunicação entre o robô ZECA e o computador. A API de deteção de objetos é responsável pela disponibilização da lista de atividades de ensino. Esta API tem também como finalidade a deteção dos objetos presentes numa imagem. A interface gráfica possibilita ao utilizador a definição das sequências de instruções a transmitir ao robô, que por sua vez transmitirá à criança. O objetivo das sequências de instruções é o ensino à criança da atividade de ensino.

Na API ZECA estão disponíveis todas as instruções possíveis que o ZECA é capaz de executar. Desta forma, o ZECA é capaz de receber todas as instruções de fala e gestos e executá-las corretamente.

A API de deteção de objetos é responsável pelo reconhecimento de objetos numa imagem. Neste caso, tendo em conta que a lista de atividades de ensino é constituída por cores e formas geométricas, os objetos mostrados pela criança ao robô são detetados na API.

A interface gráfica toma decisões com base na deteção dos objetos obtidos pela API de deteção de objetos e, procede à emissão de feedback tanto ao utilizador como à criança. O *feedback* enviado à criança será realizado pela interface com recurso à API ZECA, enquanto o *feedback* ao utilizador será realizado graficamente na interface gráfica.

Em síntese, na interface gráfica, o utilizador pode selecionar qual a atividade de ensino que deseja ensinar à criança (cores e formas geométricas). Cada atividade de ensino possui uma lista de instruções ordenadas e sequenciadas. Cada atividade de ensino está representada na interface, com a respetiva sequência de instruções para que o utilizador as possa criar, consultar e editar.

4.7.1 Arquitetura do sistema

A arquitetura do sistema implementado encontra-se esquematizada na Figura 27. A arquitetura do sistema é composta pelos seguintes elementos:

— Robô ZECA;

— Câmara para captação em tempo real;

— Interface gráfica desenvolvida em c# WPF;

— API ZECA desenvolvida em JAVA;

— API de deteção de objetos desenvolvida em Python.

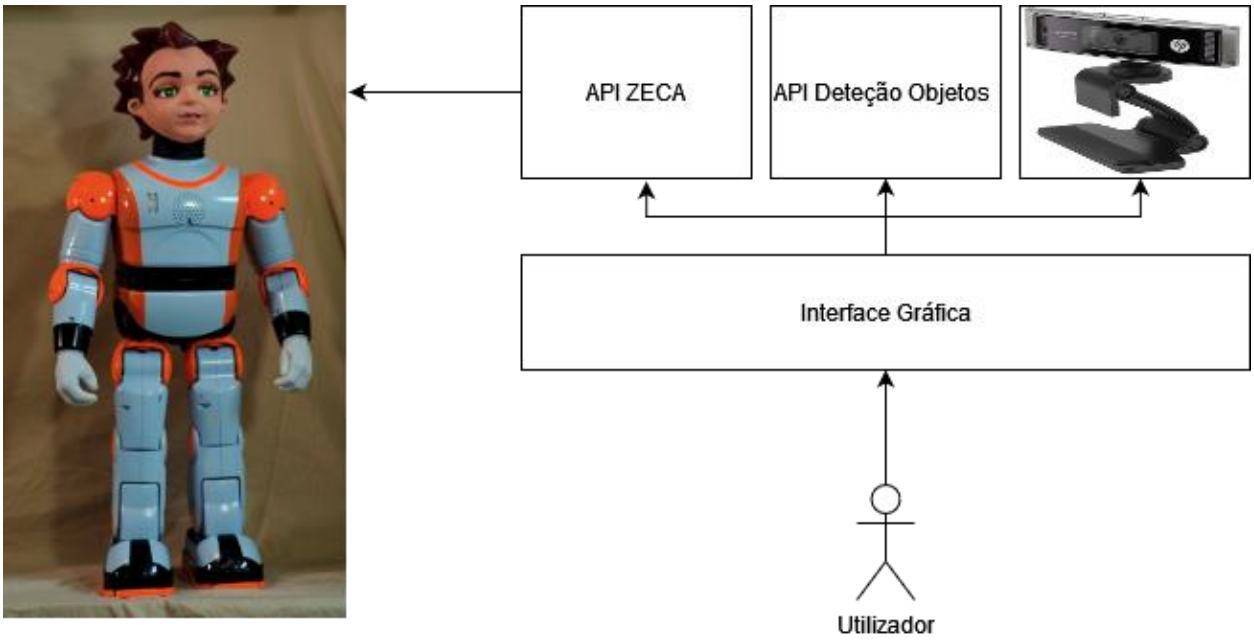


Figura 27 – Arquitetura do sistema desenvolvido

Na interface gráfica, o utilizador tem a possibilidade criar sequências de instruções com o objetivo de ensinar uma atividade à criança. Depois do utilizador ter a sequência de instruções finalizada, este pode então dar início à atividade de ensino. Uma atividade de ensino, representada na Figura 28, é iniciada com o envio de uma instrução de fala ao robô, definida na frase de inicialização. A atividade é finalizada quando não existem mais instruções na lista definida pelo utilizador, enviando ao robô a instrução de fala, definida na frase de finalização. Depois da frase de inicialização a interface percorre todas as instruções definidas pelo utilizador, durante a atividade de ensino e a interface está constantemente a obter a imagem da câmara em tempo real, enviando-a para a API de deteção de objetos.

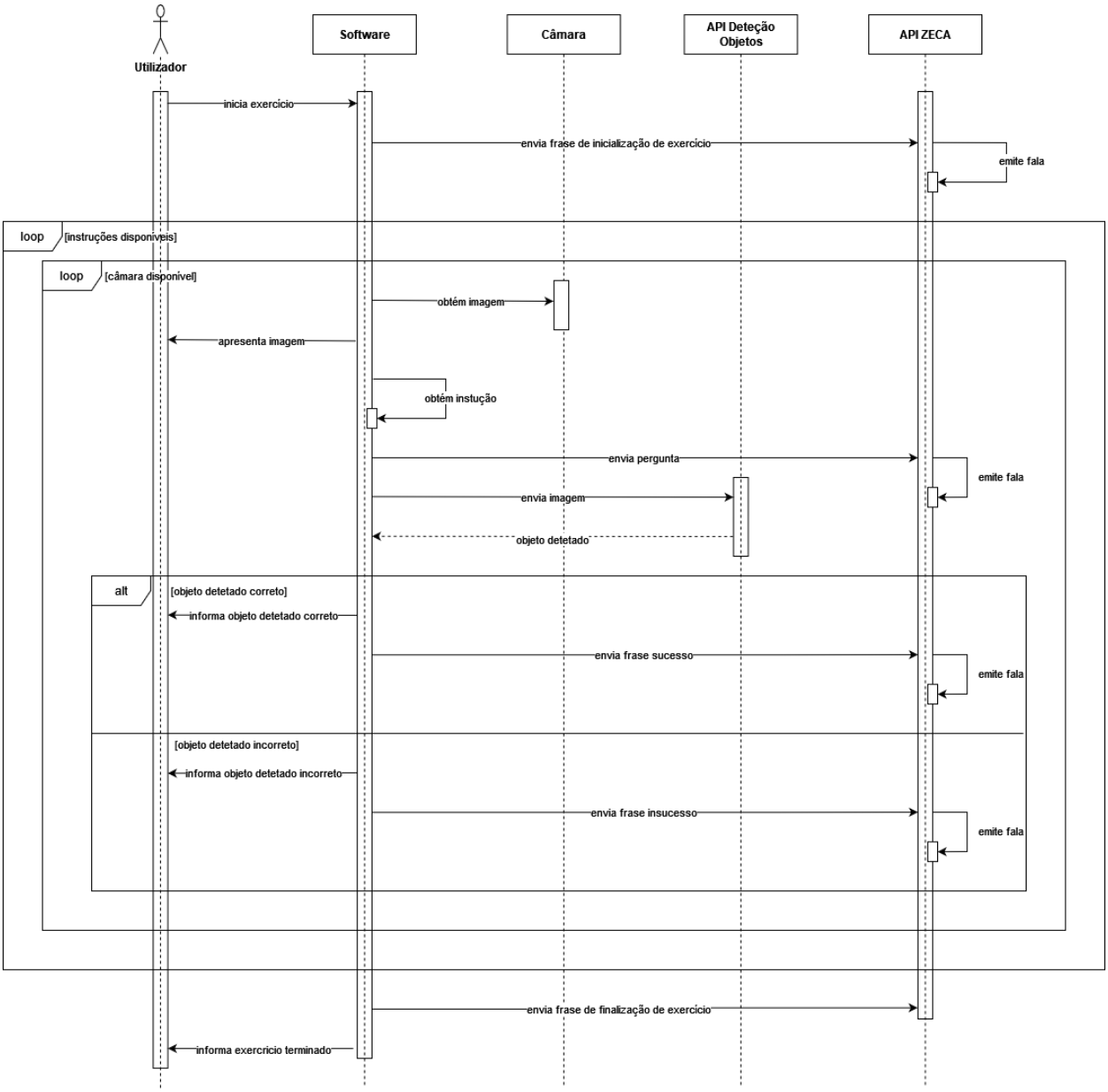


Figura 28 – Diagrama de sequência de uma atividade de ensino

Depois de finalizada a atividade de ensino, o utilizador é redirecionado para a interface inicial onde pode dar início a outra atividade de ensino.

5. RESULTADOS

Neste Capítulo são descritos os procedimentos utilizados para avaliação do desempenho do sistema desenvolvido. De seguida, serão apresentados e discutidos os resultados obtidos na deteção de cores e formas geométricas nas duas redes neuronais treinadas.

5.1 Avaliação do Sistema

Para a avaliação da precisão das redes treinadas e verificação dos objetos detetados, foram utilizados os *scripts* descritos no Capítulo 4.3.2: a) *script* para deteção de objetos em tempo real; b) *script* para deteção de objetos a partir de um vídeo.

A avaliação do sistema desenvolvido, isto é, a percentagem da precisão dos objetos detetados, foi realizada de duas formas distintas:

a) De forma parcial, isto é, sem o robô ZECA, sendo este substituído pela voz do computador. Esta metodologia possibilitou testar o funcionamento do todo o sistema sem o ZECA, numa fase preliminar dos trabalhos (fase de testes);

b) De forma global, isto é, com robô ZECA, onde foi possível avaliar o desempenho do robô e do sistema global. Nesta fase, todo o sistema desenvolvido já tinha sido previamente testado no ponto a) e já estava em funcionamento. No ponto b) foi testado se o robô executava corretamente as ordens de instruções e a comunicação entre o robô e o computador. Estes testes foram realizados por dois adultos.

Para cada uma das atividades de ensino (cores e formas geométricas) foram configuradas ordens de instruções.

5.1.1 Avaliação da deteção das cores

A partir dos gráficos *Loss* x *Precision*, gerados no treino de cada rede de cores, foi possível a comparação da evolução do treino das redes ao longo do tempo. Para o treino da rede das cores foram treinadas 8 redes, nomeadamente com 50, 100, 150 e 200 fotografias para cada objeto, com e sem pré-treino. Os resultados encontram-se ilustrados na Figura 29 (onde foram utilizadas 50 fotografias por objeto), Figura 30 (onde foram utilizadas 100 fotografias por objeto), Figura 31 (onde foram utilizadas 150 fotografias por objeto) e Figura 32 (onde foram utilizadas 200 fotografias por objeto). Na Tabela 9 encontram-se sumarizados os valores de mAP global retiradas das Figuras referidas, assim como os valores de *Loss*.

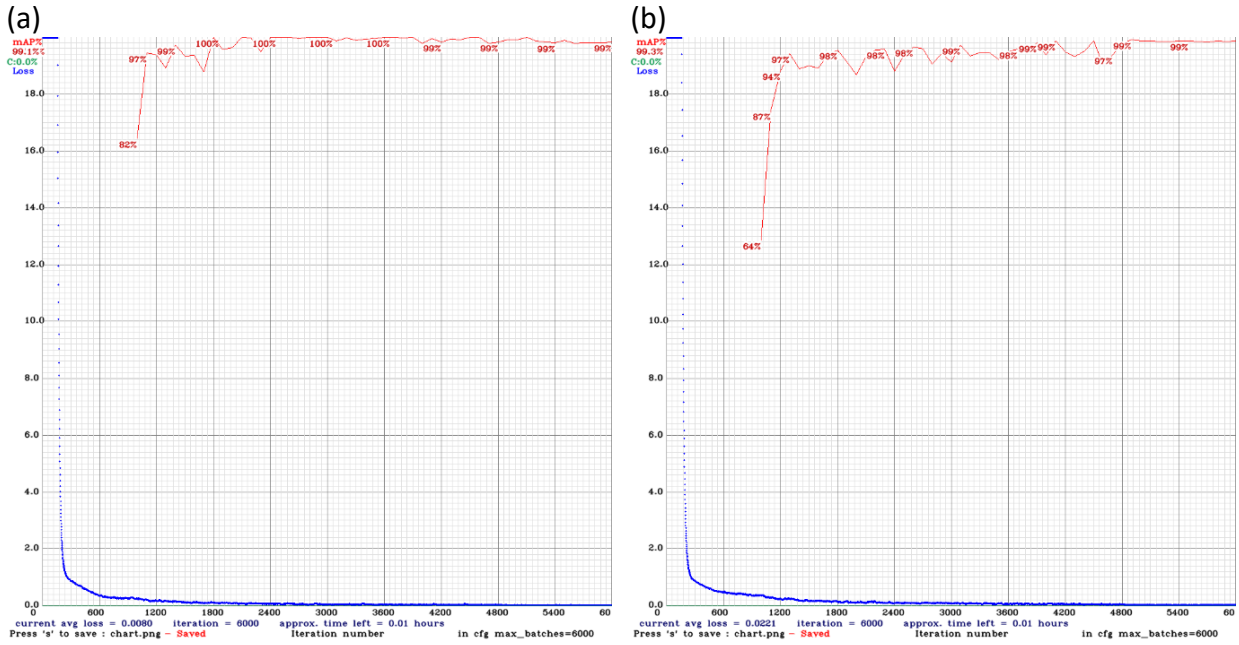


Figura 29 – Gráficos *Loss x Precision* obtidos no treino de cores com 50 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

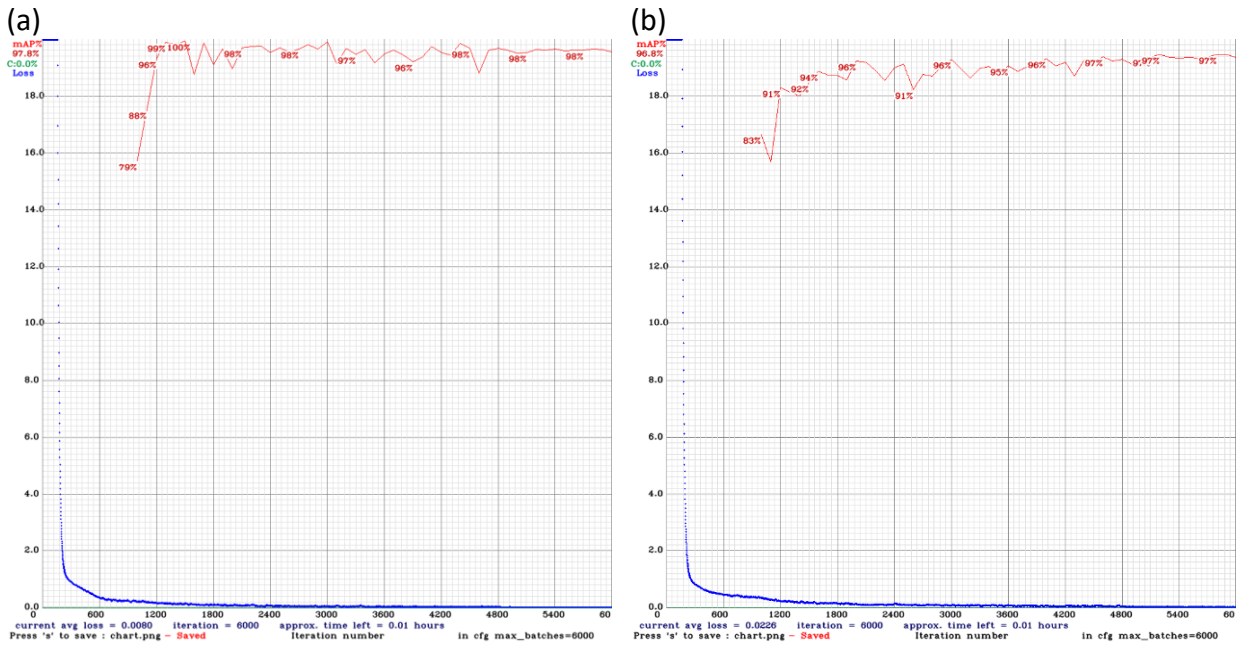


Figura 30 – Gráficos *Loss x Precision* obtidos no treino de cores com 100 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

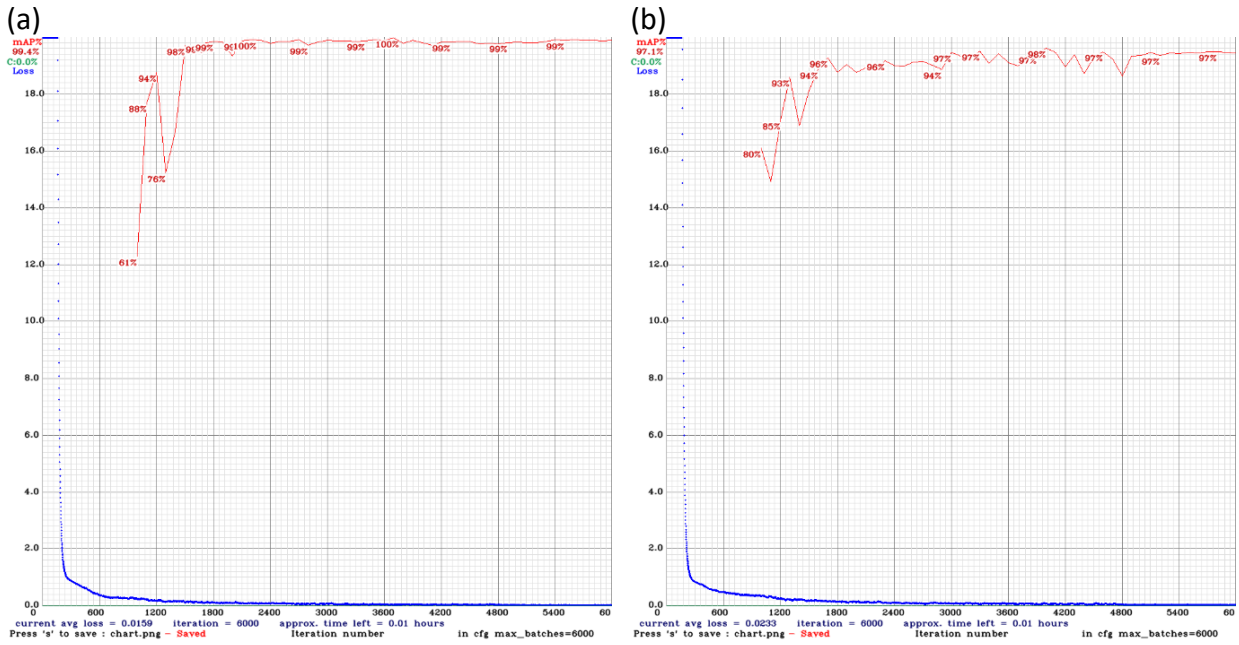


Figura 31 – Gráficos *Loss x Precision* obtidos no treino de cores com 150 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

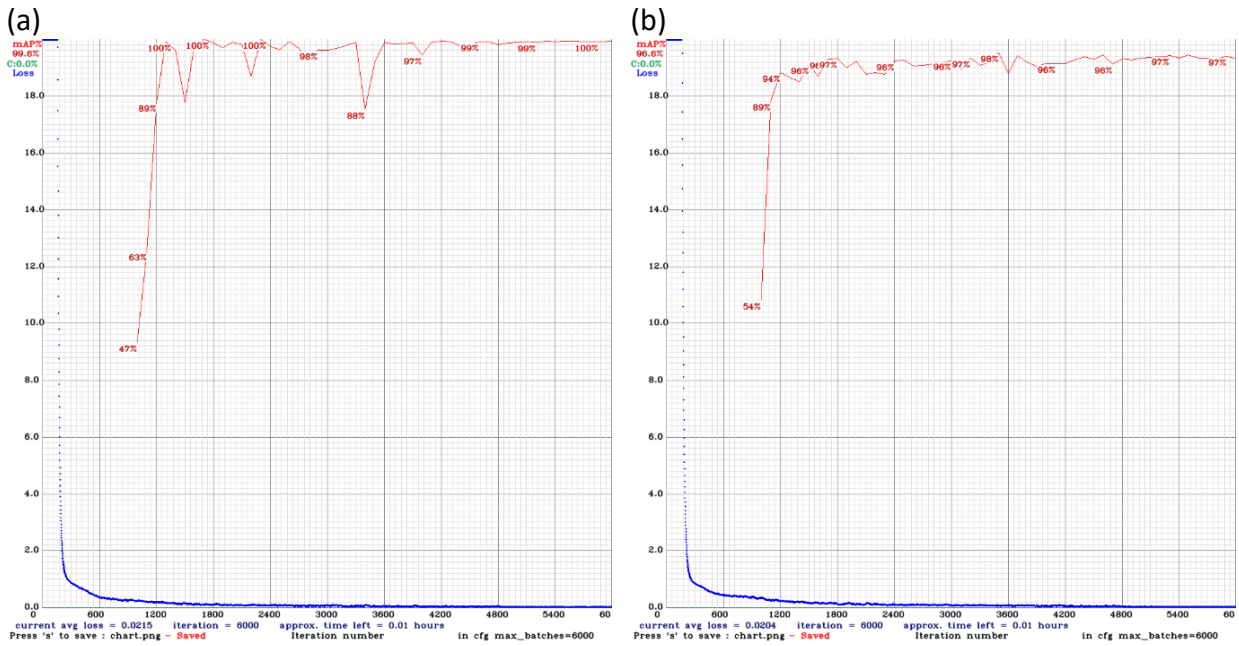


Figura 32 – Gráficos *Loss x Precision* obtidos no treino de cores com 200 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

Tabela 9 – Valores de mAP global e *Loss* obtidos durante o treino das redes de cores

	Com pré-treino		**Sem pré-treino**	
Imagens por objeto	mAP global	*Loss*	mAP global	*Loss*
50	99.21%	0.0080	99.30%	0.0221
100	97.80%	0.0080	96.80%	0.0226
150	99.40%	0.0159	97.10%	0.0233
200	99.60%	0.0215	96.60%	0.0204

Nas Figuras 29, 29, 30, 31, 32 e Tabela 9, verificou-se que os valores de mAP para redes treinadas com pré-treino são superiores comparativamente às redes treinadas sem pré-treino. Verificou-se também que as redes treinadas com pré-treino e sem pré-treino apresentam valores de *Loss* baixos e que estes valores diminuem à medida que o treino da rede é realizado, o que indica que a rede está a ser treinada corretamente.

Para cada rede treinada, com pré-treino e sem pré-treino, foi executado o comando “darknet detector map” com a finalidade de obter os seguintes valores: a) mAP; b) AP para cada objeto da

rede; c) IoU; d) precisão; e) número de verdadeiros e falsos positivos para cada objeto. Estes resultados encontram-se sumarizados nas Tabelas 10, 11, 12 e 13

Tabela 10 – Resultados de mAP, IoU e precisao da rede de cores treinada com pré-treino

Imagens por objeto	mAP	IoU	Precisão
50	99.76%	71.76%	0.90
100	99.82%	81.19%	0.96
150	100.00%	82.30%	0.97
200	99.98%	81.74%	0.98

Tabela 11 – Resultados de AP, verdadeiros e falsos positivos por objeto da rede de cores treinada com pré-treino

Imagens por objeto	AP (Vermelho)	VP	FP	AP (Azul)	VP	FP	AP (Verde)	VP	FP
50	99.53%	53	6	99.76%	59	14	100.00%	69	0
100	100.00%	53	2	99.45%	59	4	100.00%	69	2
150	100.00%	53	1	100.00%	59	4	100.00%	69	1
200	100.00%	53	1	99.94%	59	2	100.00%	69	0

Tabela 12 – Resultados de mAP, IoU e precisão da rede de cores treinada sem pré-treino

Imagens por objeto	mAP	IoU	Precisão
50	99.58%	81.95%	0.92
100	98.80%	86.31%	0.94
150	99.15%	78.18%	0.93
200	98.16%	81.27%	0.95

Tabela 13 – Resultados de AP, verdadeiros e falsos positivos por objeto da rede de cores treinada sem pre-treino

Imagens por objeto	AP (Vermelho)	VP	FP	AP (Azul)	VP	FP	AP (Verde)	VP	FP
50	99.53%	52	3	99.76%	59	13	100.00%	68	0
100	100.00%	52	6	99.45%	59	4	100.00%	69	1
150	100.00%	52	9	100.00%	59	4	100.00%	69	0
200	100.00%	51	8	99.94%	59	1	100.00%	69	0

Os resultados sumarizados das Tabelas 10, 11, 12 e 13 mostram que:

a) Por norma, o treino de uma rede com uma rede já pré-treinada, Tabela 10 e Tabela 11, conduz a valores de mAP, IoU e precisão superiores do que os valores obtidos nas redes treinadas sem pré-treino;

b) Quanto maior o número de fotografias utilizadas no treino da rede, maior é o valor de AP de deteção do objeto, menor é o número de falsos positivos e maior é o número de verdadeiros positivos.

Os resultados mostram também que as redes treinadas com 150 ou 200 fotografias por objeto a partir de uma rede pré-treinada apresentam resultados bastante favoráveis, com valores superiores a 99% de mAP, 81% de IoU e 0.97 de precisão. Os resultados mostram que a rede treinada com 200 fotografias com pré-treino apresenta os valores mais elevados de mAP, IoU e precisão.

Para a validação da deteção dos objetos em tempo real foi utilizada a rede treinada com 200 fotografias por objeto. Com o auxílio do *script* desenvolvido para deteção de objetos em tempo real, foi possível a deteção das cores (vermelho, verde e azul). Para demostração dos resultados obtidos, foi gravado um vídeo para cada objeto e em torno do mesmo, que pode ser consultado na Tabela 14.

Tabela 14 – Vídeos de cores para cada objeto e em torno do mesmo

Objeto	**Vídeo**
Vermelho	https://youtu.be/Yn2cIdak7ew
Verde	https://youtu.be/dl5nCAlU1H0
Azul	https://youtu.be/iiJBQYNdCQg

A finalidade de gravar um vídeo em torno do próprio objeto foi de validar se o objeto detetado é o correto. Estes vídeos foram gravados com a mesma câmara que foi utilizada para fotografar os objetos para o treino das redes. Para cada vídeo, com o auxílio do *script* desenvolvido para a deteção dos objetos em vídeo, foi possível a deteção dos objetos que aparecem nos vídeos a), b) e c) e estes resultados encontra-se sumarizados na Tabela 16. Uma vez que cada vídeo possui durações distintas, o número de *frames* existentes em cada vídeo é diferente, o que consequentemente faz com que o número de deteções também seja diferente. Os vídeos gerados com os objetos detetados podem ser consultados na Tabela 15.

Tabela 15 – Vídeos gerados com os objetos de cores detetados

Objeto	**Vídeo**
Vermelho	https://youtu.be/mhQ4j2mUuWk
Verde	https://youtu.be/WHfCyeM6yYk
Azul	https://youtu.be/lqe6jQKmxi4

Tabela 16 – Número de deteções para cada objeto para o treino da rede de cores

		***Frames* detetados**					
Objeto a detetar	**Total de *frames***	Vermelho	mAP	Azul	mAP	Verde	mAP
Vermelho	596	580	94.71%	0	0	0	0
Azul	612	0	0	507	83.49%	0	0
Verde	589	0	0	0	0	547	74.37%

Os resultados sumarizados na Tabela 16 mostram que todas as cores foram detetadas corretamente. Os resultados mostram que a rede treinada apenas não detetou: a) o vermelho

em 16 *frames*, ou seja, 2.7% do número total de *frames*; b) o azul em 105 *frames*, ou seja, em 17.2% do número total de *frames*; c) o verde em 42 *frames*, ou seja, em 7.1% do número total de *frames*, apesar de ter uma mAP de 74%. Assim sendo, a não deteção da cor vermelha e verde é praticamente impercetível apesar da percentagem da precisão da cor verde ser menor. Na deteção da cor azul é possível, por vezes, verificar que em alguns *frames* a cor não é detetada e a precisão é mais baixa que a da cor vermelha.

5.1.2 Avaliação parcial do sistema de deteção de cores

Para a avaliação parcial do sistema para a atividade de ensino de cores, foram criadas na interface as ordens de instruções sumarizadas na Tabela 17. Nesta atividade de ensino foi utilizada a rede pré-treinada com 200 fotografias por objeto.

Tabela 17 – Ordens de instruções criadas para validação do funcionamento do sistema parcial de deteção de cores

Instrução 1		
	Objeto	Vermelho
	Pergunta	Podes mostrar a cor vermelha?
	Resposta em caso deteção correta	Muito bem, essa é a cor vermelha!
	Resposta em caso deteção incorreta	Essa não é a cor vermelha!
Instrução 2		
	Objeto	Azul
	Pergunta	Podes mostrar a cor azul?
	Resposta em caso deteção correta	Muito bem, essa é a cor azul!
	Resposta em caso deteção incorreta	Essa não é a cor azul!
Instrução 3		
	Objeto	Verde
	Pergunta	Podes mostrar a cor verde?
	Resposta em caso deteção correta	Muito bem, essa é a cor verde!
	Resposta em caso deteção incorreta	Essa não é a cor verde!

Como é possível a utilização de uma frase de iniciação e uma de finalização da atividade de ensino, foram utilizadas as frases: “Olá, vamos aprender cores?” e “Muito bem, aprendeste três

cores, o vermelho, o azul e o verde. Parabéns!" respetivamente. Com o auxílio das ordens de instruções da Tabela 15, foi avaliado se os objetos foram detetados corretamente e, para cada instrução, se a pergunta e as respostas de deteção correta e incorreta são emitidas corretamente. Para tal, foi definida a lista de validações conforme a Tabela 18.

Tabela 18 – Lista de validações a verificar na deteção de cores

Instrução nº	**Validação**
1	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar o objeto correto, ou seja, o vermelho
	Validar se é emitida a resposta de deteção correta
2	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar um objeto que não o azul
	Validar se é emitida a resposta de deteção incorreta
	Validar se a pergunta é emitida novamente
	Mostrar o objeto correto, ou seja, o azul
3	
	Validar se é emitida a resposta de deteção correta
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar o objeto correto, ou seja, o verde
	Validar se é emitida a resposta de deteção correta

No link *https://youtu.be/mOAqlI-GFJ4* é possível visualizar o desempenho do sistema parcial. Neste vídeo é possível observar que todas as validações ocorreram com sucesso, ou seja, as ordens de instruções foram percorridas na ordem correta. Verificou-se também a correta emissão de feedback, tanto do lado do computador como da interface, assim como a correta deteção dos objetos propostos (azul, vermelho e verde).

Em síntese, os resultados demonstram que a deteção de cores e as ordens de instruções foram executadas com sucesso.

5.1.3 Avaliação de deteção de formas geométricas

A partir dos gráficos *Loss x Precision*, gerados no treino de cada rede de formas geométricas, foi possível a comparação da evolução do treino das redes ao longo do tempo. Para o treino da rede das formas geométricas foram treinadas 8 redes, nomeadamente com 50, 100, 150 e 200 fotografias para cada objeto, com e sem pré-treino. Os resultados encontram-se ilustrados na Figura 33 (onde foram utilizadas 50 fotografias por objeto), Figura 34 (onde foram utilizadas 100 fotografias por objeto), Figura 35 (onde foram utilizadas 150 fotografias por objeto) e Figura 36 (onde foram utilizadas 200 fotografias por objeto). Na Tabela 16 encontram-se sumarizados os valores de mAP globais retiradas das Figuras referidas, assim como os valores de *Loss*.

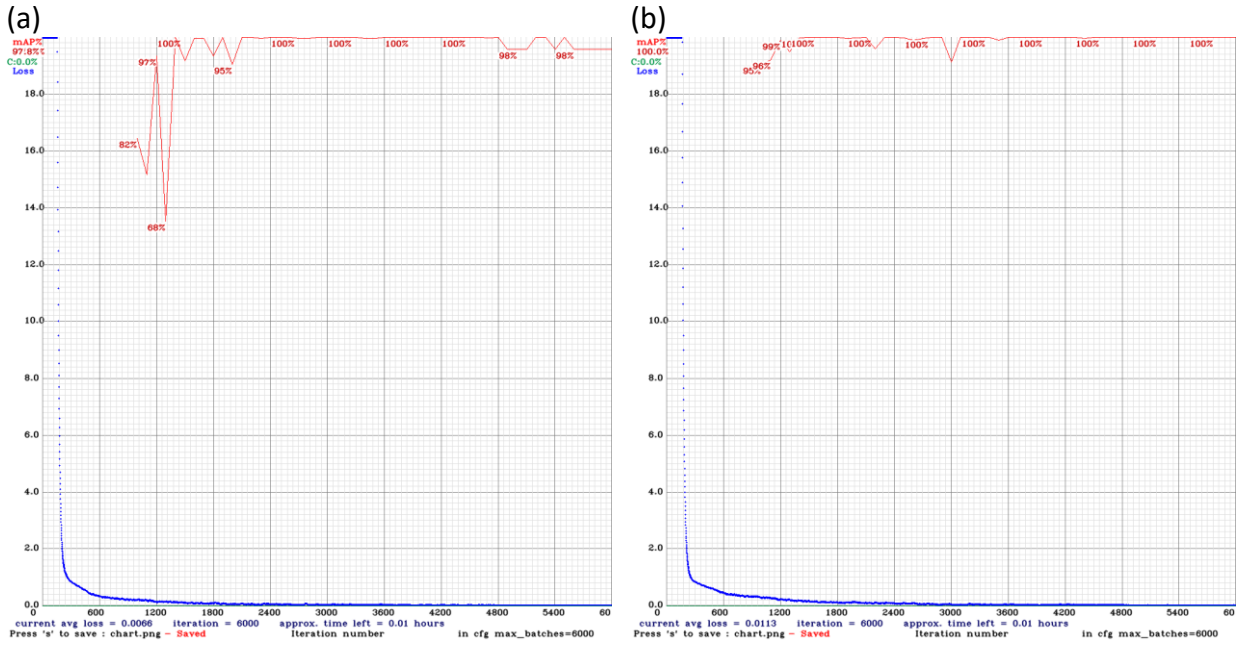
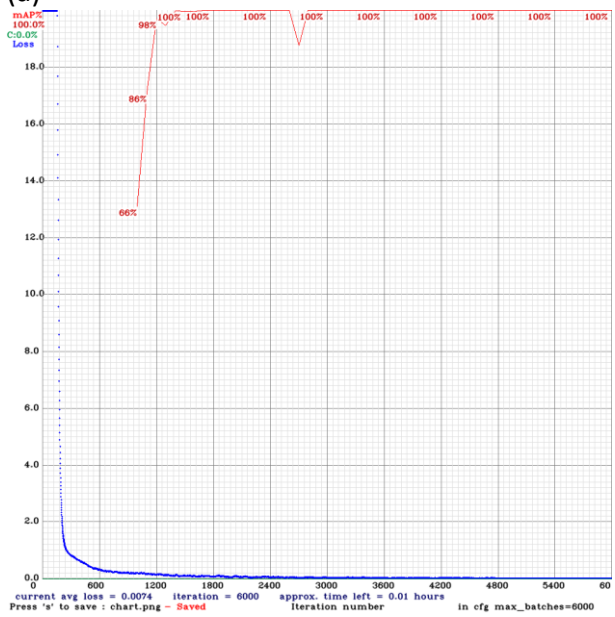


Figura 33 – Gráficos *Loss x Precision* obtidos no treino de formas geométricas com 50 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

(a)



(b)

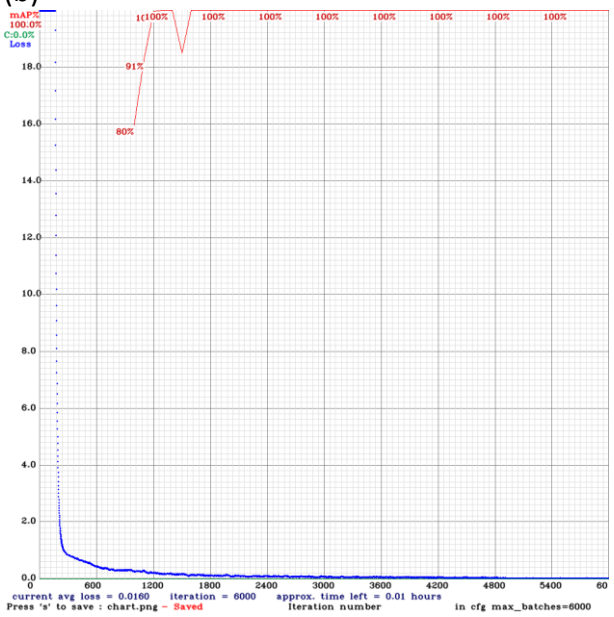
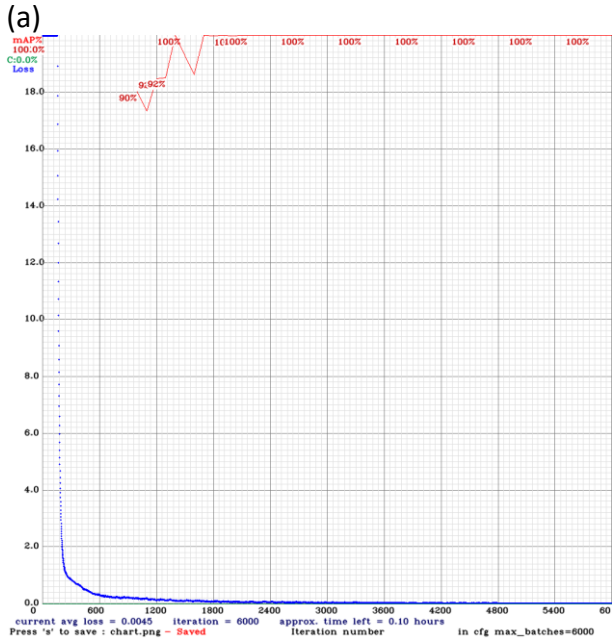


Figura 34 – Gráficos *Loss x Precision* obtidos no treino de formas geométricas com 100 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

(a)



(b)

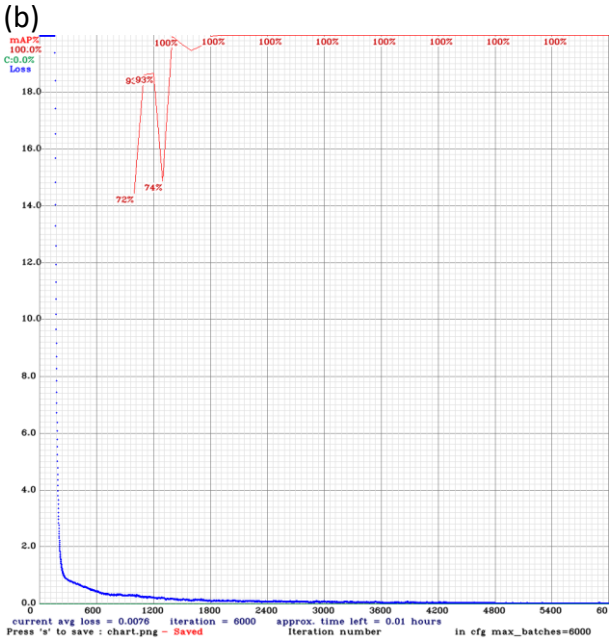
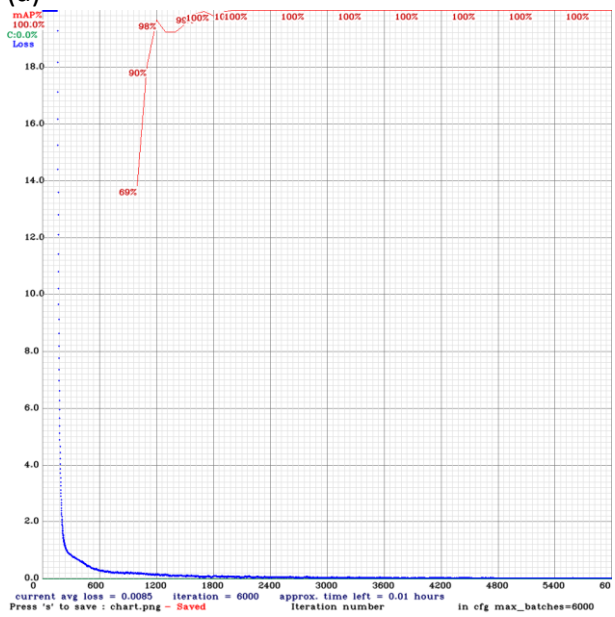


Figura 35 – Gráficos *Loss x Precision* obtidos no treino de formas geométricas com 150 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

(a)



(b)

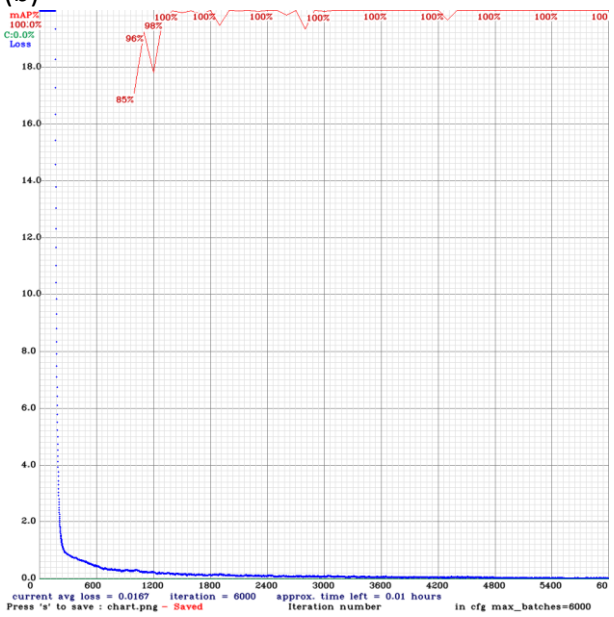


Figura 36 – Gráficos *Loss x Precision* obtidos no treino de formas geométricas com 200 fotografias por objeto: (a) com pré-treino (b) sem pré-treino

Tabela 19 – Valores de mAP global e *Loss* dos gráficos obtidos no treino da rede de figuras geométricas

	Com pré-treino		**Sem pré-treino**	
Imagens por objeto	mAP global	*Loss*	mAP global	*Loss*
50	97.80%	0.0066	100.00%	0.0113
100	100.00%	0.0074	100.00%	0.0160
150	100.00%	0.0045	100.00%	0.0076
200	100.00%	0.0085	100.00%	0.0167

Nas Figuras 33, 34, 35, 36 e na Tabela 19, verificou-se que os valores de mAP para redes treinadas com pré-treino e sem pré-treino são de 100.00%. Ta como verificado no treino da rede para deteção das cores, verificou-se que as redes treinadas com pré-treino e sem pré-treino apresentam valores de *Loss* baixos e que estes valores diminuem à medida que o treino da rede é realizado.

Para cada rede treinada, com pré-treino e sem pré-treino, foi executado o comando “darknet detector map” com a finalidade de obter os seguintes valores: a) mAP; b) AP para cada objeto da

rede; c) IoU; d) precisão; e) número de verdadeiros e falsos positivos para cada objeto. Estes resultados encontram-se sumarizados nas Tabelas 20, 21, 22, 23.

Tabela 20 – Resultados de mAP, IoU e precisao da rede de figuras geométricas treinada com pré-treino

Imagens por objeto	mAP	IoU	Precisão
50	71.18%	55.71%	0.74
100	68.36%	63.41%	0.78
150	72.72%	61.57%	0.75
200	71.81%	66.59%	0.78

Tabela 21 – Resultados de AP e verdadeiros e falsos positivos por objeto da rede de figuras geométricas treinada com pré-treino

Imagens por objeto	AP (Triângulo)	VP	FP	AP (Círculo)	VP	FP	AP (Quadrado)	VP	FP
50	81.23%	31	21	97.95%	36	6	34.38%	11	1
100	71.58%	31	20	99.12%	36	2	34.38%	11	0
150	84.10%	31	21	99.68%	37	5	34.38%	11	0
200	81.06 %	31	21	100.00%	37	1	34.38%	11	0

Tabela 22 – Resultados de mAP, IoU e precisão da rede de figuras geométricas treinada sem pré-treino

Imagens por objeto	mAP	IoU	Precisão
50	68.33%	49.47%	0.64
100	69.18%	47.49%	0.63
150	72.83%	57.88%	0.72
200	73.36%	52.21%	0.66

Tabela 23 – Resultados de AP e verdadeiros e falsos positivos por objeto da rede de figuras geométricas treinada sem pré-treino

Imagens por objeto	AP (Triângulo)	VP	FP	AP (Círculo)	VP	FP	AP (Quadrado)	VP	FP
50	85.90%	29	24	84.72%	34	18	34.38%	11	0
100	88.10%	29	19	86.33%	36	23	33.10%	11	2
150	96.42%	31	12	87.70%	35	18	34.38%	11	0
200	97.10%	30	19	88.60%	37	20	34.38%	11	1

Os resultados mostram que, apesar de graficamente os valores de mAP global serem de 100.00%, quando são obtidos os valores de mAP da melhor rede treinada com 200 fotografias por objeto, os resultados não são satisfatórios. Mesmo utilizando 200 fotografias por objeto, os valores de mAP $\approx$ 72%, IoU $\approx$ 66% e precisão $\approx$ 0.78.

Com a utilização do *script* para a deteção de objetos em tempo real, foi detetada uma inconsistência durante a deteção do quadrado. Esta inconsistência ocorria quando o quadrado era girado, tanto para a direita como para a esquerda, e a deteção em grande parte das vezes resultava num triângulo. Essa inconsistência pode ser observada no vídeo *https://youtu.be/kmIDW7WFXkw*. Verificou-se que esta inconsistência devia-se ao facto das fotografias dos objetos terem sido tiradas sempre com a mesma orientação (Figura 37), fazendo com que o quadrado girado a 45 graus, na parte superior ou na parte inferior (Figura 38), ficasse semelhante com o triângulo. Esta inconsistência é a razão pela qual os valores de AP do quadrado sejam de, aproximadamente, 34%.

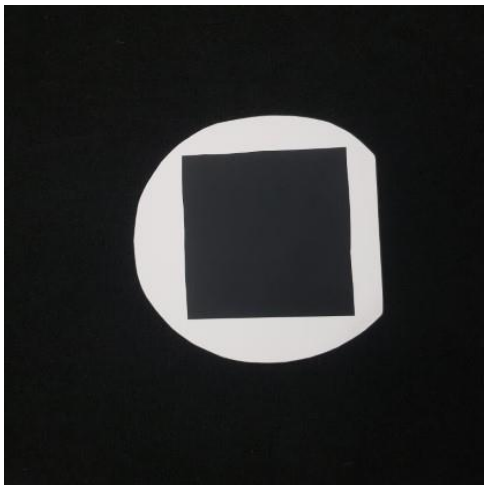


Figura 37 – Fotografia retirada ao quadrado

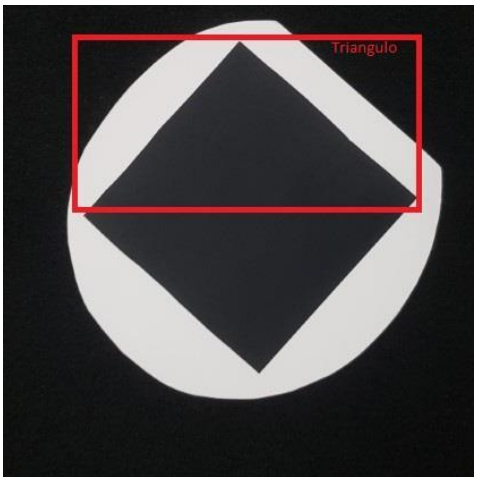
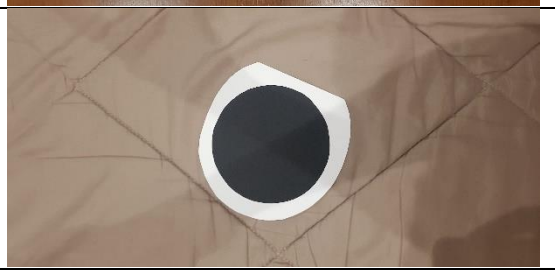


Figura 38 – Fotografia retirada ao quadrado com uma rotação de ≈ 45º

Para retificar esta inconsistência, foram tiradas 40 fotografias de cada objeto (triângulo, círculo e quadrado), com diferentes rotações e com distâncias compreendidas entre 0.5 e 1 metro (Tabela 24). Das 600 fotografias existentes, foram retiradas 120 fotografias e adicionadas 120 fotografia novas etiquetadas. De seguida procedeu-se ao treino de uma nova rede com pré-treino.

Tabela 24 – Fotografias retiradas com diferentes ângulos

Objeto	**Imagem**
Quadrado	
Triângulo	
Círculo	

Depois do treino da nova rede e com a o auxílio do *script* para deteção de objetos em tempo real, foi observado uma diferença significativa na deteção do quadrado quando este era rodado. Assim sendo e apesar das melhorias implementadas, com alguma frequência ainda foi detetado o triângulo em vez do quadrado. Posto isso, foi então treinada uma nova rede com 50, 100, 150 e 200 fotografias por objeto retiradas com diversos ângulos e com distâncias compreendidas entre 0.5 metros a 1 metro. A rede foi treinada com uma rede pré-treinada. Os resultados dos gráficos *Loss x Precision* encontram-se ilustrados na Figura 39 e sumarizados na Tabela 25.

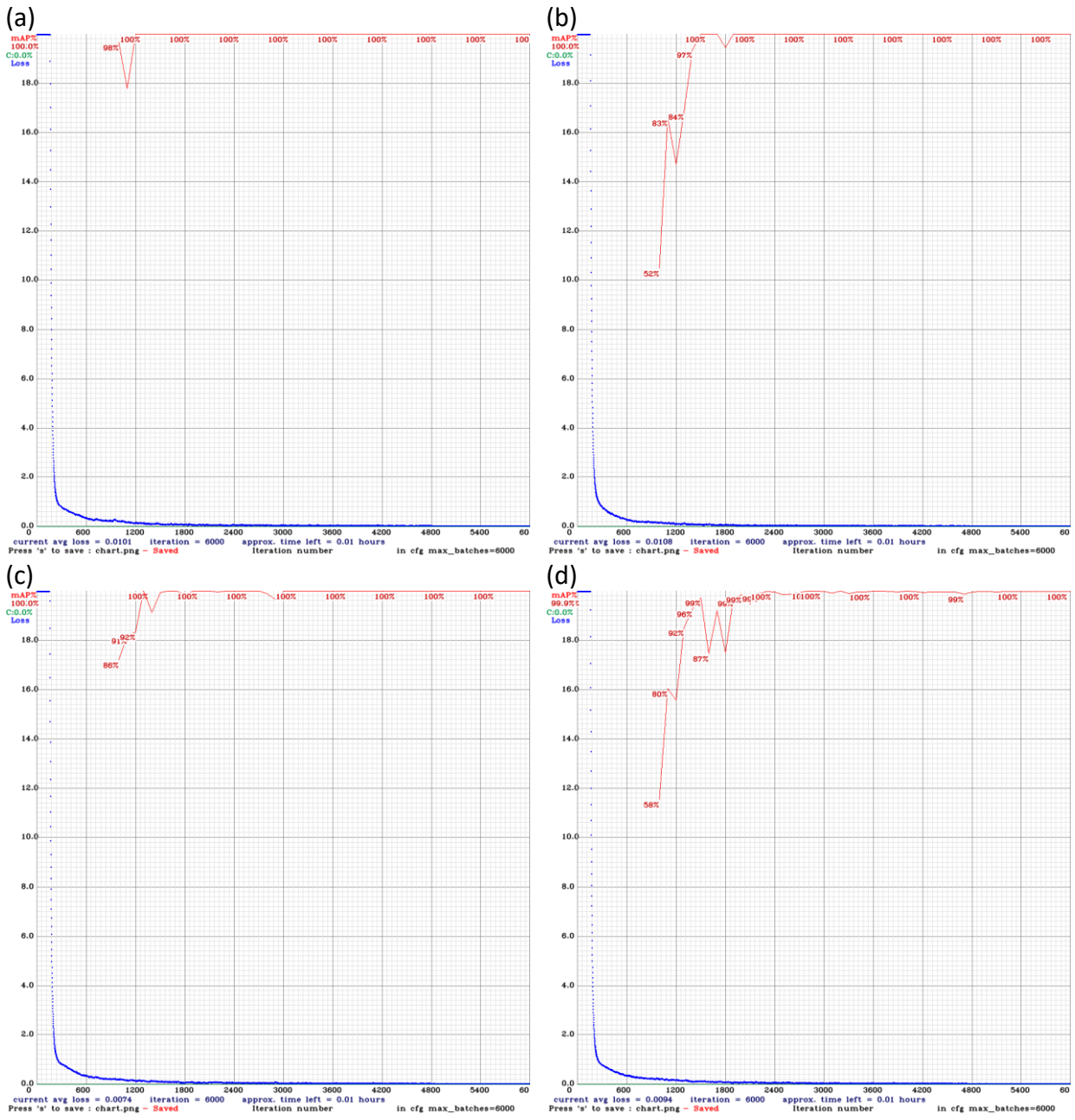


Figura 39 – Gráficos *Loss x Precision* obtidos no treino da nova rede de figuras geométricas: (a) 50 fotografias por objeto (b) 100 fotografias por objeto (c) 150 fotografias por objeto (d) 200 fotografias por objeto

Tabela 25 – Valores de mAP global e *Loss* obtidos no novo treino da rede de figuras geométricas

Imagens por objeto	mAP global	*Loss*
50	100.00%	0.0101
100	100.00%	0.0108
150	100.00%	0.0074
200	99.99%	0.0094

Os resultados mostram que a nova rede treinada possui valores de mAP de ≈ 100.00%, com valores de *Loss* reduzidos. Similarmente ao procedimento adotado na avaliação de desempenho da rede anterior, foram calculados os valores de mAP, AP para cada objeto, IoU, precisão e o número de verdadeiros e falsos positivos para a nova rede. Os resultados obtidos encontram-se sintetizados nas Tabelas 26 e 27.

Tabela 26 – Resultados de mAP, IoU e precisão obtidos após o treino da nova rede de figuras geométricas

Imagens por objeto	mAP	IoU	Precisão
50	71.14%	40.62%	0.55
100	88.89%	50.36%	0.65
150	93.91%	62.52%	0.91
200	98.52%	82.11%	0.96

Tabela 27 – Resultados de AP, verdadeiros e falsos positivos por objeto obtidos após o treino da nova rede de figuras geométricas

Imagens por objeto	AP (Triângulo)	VP	FP	AP (Círculo)	VP	FP	AP (Quadrado)	VP	FP
50	87.97%	27	24	75.82%	30	25	49.62%	14	9
100	90.13%	30	33	83.21%	31	8	93.33%	31	9
150	99.36%	31	2	94.22%	34	3	88.14%	30	4
200	95.80%	31	2	99.76%	35	2	100.00%	32	0

Os resultados obtidos na nova rede treinada de formas geométricas (Tabela 25 e 23) são similares, em ordem de grandeza, aos resultados observados na rede treinada de cores. A utilização de 200 fotografias por objeto permitiu a obtenção de valores acima de 98% de mAP, 82% de IoU e 0.96 de precisão. Comparativamente à rede de figuras geométricas treinada previamente (inexistência de fotografias retiradas com diversos ângulos), é possível observar na Tabela 27 que valores de AP para cada objeto são superiores a 95% e a forma geométrica quadrado passou de 34% para 100.00%.

Para a validação da deteção dos objetos em tempo real foi utilizada a rede treinada com 200 fotografias por objeto, uma vez que é a rede que apresenta maiores valores de mAP e IoU tal como observado aquando do treino de rede de cores. Com o auxílio do *script* desenvolvido para deteção de objetos em tempo real, foi possível a deteção das formas geométricas (quadrado, círculo e triângulo). Para demostração dos resultados obtidos, foi gravado um vídeo para cada objeto e em torno do mesmo. Estes vídeos podem ser consultados na Tabela 28.

Tabela 28 – Vídeos de figuras geométricas para cada objeto e em torno do mesmo

Objeto	**Vídeo**
Triângulo	https://youtu.be/66ZAwiMWd_c
Quadrado	https://youtu.be/hgqH0mqvldw
Círculo	https://youtu.be/v5ECx0M6xGY

Para cada vídeo, da Tabela 28, com o auxílio do script desenvolvido para a deteção dos objetos em vídeo, foi possível a deteção dos objetos que aparecem nos vídeos e estes resultados encontra-se sumarizados nas Tabelas 29 e 30, para a primeira e segunda rede de formas geométricas, respetivamente.

Tabela 29 – Número de deteções para cada objeto e precisão média obtido com a primeira rede de figuras geométricas

		***Frames* detetados**					
Objeto a detetar	**Total de *frames***	Quadrado	Precisão	Círculo	Precisão	Triângulo	Precisão
Quadrado	596	238	97.20%	7	69.06%	272	93.94%
Círculo	532	0	0%	532	98.20%	0	0%
Triângulo	582	0	0%	0	0%	536	89.95%

Tabela 30 – Número de deteções para cada objeto e precisão média obtido com a segunda rede de figuras geométricas

		***Frames* detetados**					
Objeto a detetar	**Total de *frames***	Quadrado	Precisão	Círculo	Precisão	Triângulo	Precisão
Quadrado	596	596	99.57%	0%	0%	0%	0%
Círculo	532	0	0%	532	99.83%	0%	0%
Triângulo	582	0	0%	0%	0%	582	98.99%

Os vídeos gerados com os objetos detetados a partir da primeira rede treinada, sem fotografias retiradas com diversos ângulos, podem ser consultados na Tabela 31 e os vídeos gerados com os objetos detetados a partir da segunda rede treinada, com fotografias retiradas com diversos ângulos, podem ser consultados na Tabela 32.

Tabela 31 – Vídeos gerados a partir da primeira rede treinada com os objetos de figuras geométricas detetados

Objeto	**Vídeo**
Triângulo	https://youtu.be/AuDcm2gH_Gw
Quadrado	https://youtu.be/Mpqs9EmTCUQ
Círculo	https://youtu.be/F8pvYIGruq0

Tabela 32 – Vídeos gerados a partir da segunda rede treinada com os objetos de figuras geométricas detetados

Objeto	**Vídeo**
Triângulo	https://youtu.be/xNx3mnX0dJ4
Quadrado	https://youtu.be/n4CmtZDDaHs
Círculo	https://youtu.be/Tr7kvPXF-i0

Os resultados mostram que, no vídeo do quadrado (596 *frames*) a utilização da primeira rede de treino de figuras geométricas, conduziu a deteções da forma triângulo (272 *frames*) e círculo (7 *frames*) incorretamente conforme a Tabela 29. Os resultados observados na Tabela 30, obtidos com a utilização da segunda rede treinada de formas geométricas, mostram uma melhoria bastante significativa dos resultados, com a inexistência de incoerência na deteção de objetos. A nova rede treinada detetou os objetos, em todos os vídeos de todas as formas geométricas e em todos os *frames,* com precisão superior a 99%.

Em síntese, verificou-se que, quanto maior o número de imagens fornecidas à rede, maior a AP de deteção do objeto, menor o número de falsos positivos, maior é número de verdadeiros positivos e maior é o valor de mAP. Adicionalmente, para uma maior precisão, as fotografias devem ser tiradas com vários ângulos distintos.

5.1.4 Avaliação parcial do sistema para deteção de formas geométricas

Para a avaliação parcial do sistema para a atividade de ensino de formas geométricas, foram criadas na interface as ordens de instruções listadas na Tabela 33. Nesta atividade de ensino foi utilizada a nova rede pré-treinada com 200 fotografias por objeto.

Tabela 33 – Ordens de instruções criadas para validação do funcionamento do sistema parcial de deteção de figuras geométricas

Instrução 1		
	Objeto	Quadrado
	Pergunta	Podes mostrar o quadrado?
	Resposta em caso deteção correta	Muito bem, esse é o quadrado!
	Resposta em caso deteção incorreta	Esse não é o quadrado!
Instrução 2		
	Objeto	Círculo
	Pergunta	Podes mostrar o círculo?
	Resposta em caso deteção correta	Muito bem, esse é o círculo!
	Resposta em caso deteção incorreta	Esse não é o círculo!
Instrução 3		
	Objeto	Triângulo
	Pergunta	Podes mostrar o triângulo?
	Resposta em caso deteção correta	Muito bem, esse é o triângulo!
	Resposta em caso deteção incorreta	Esse não é o triângulo!

Similarmente ao adotado na rede de cores, as frases de inicialização e finalização utilizadas foram: “Olá, vamos aprender figuras geométricas?” e “Muito bem, aprendeste três formas geométricas, o quadrado, o círculo e o triângulo. Parabéns!” respetivamente. Com o auxílio das ordens de instruções da Tabela 27, foi avaliado se os objetos foram detetados corretamente e, para cada instrução, se a pergunta e as respostas de deteção correta e incorreta são emitidas corretamente. Para tal, foi definida a lista de validações conforme a Tabela 34.

Tabela 34 – Lista de validações a verificar na deteção de figuras geométricas

Instrução nº	Validação
1	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar o objeto correto, ou seja, o quadrado
	Validar se é emitida a resposta de deteção correta
2	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar um objeto que não o círculo
	Validar se é emitida a resposta de deteção incorreta
	Validar se a pergunta é emitida novamente
	Mostra o objeto correto, ou seja, o círculo
3	
	Validar se é emitida a resposta de deteção correta
	Validar se a pergunta é emitida conforme o pretendido
	Mostra o objeto correto, ou seja, o triângulo
	Validar se é emitida a resposta de deteção correta

No link *https://youtu.be/XnPvyAPM1wI* é possível visualizar o desempenho do sistema parcial. Neste vídeo é possível observar que todas as validações ocorreram com sucesso, ou seja, as ordens de instruções foram percorridas na ordem correta. Verificou-se também a correta emissão de feedback, tanto do lado do computador como da interface, assim como a correta deteção dos objetos propostos (quadrado, triângulo e círculo).

Em síntese, os resultados demonstram que a deteção de formas geométricas e as ordens de instruções foram executadas com sucesso.

5.1.5 Avaliação global do sistema

Para avaliação global do desempenho do sistema, é necessária a validação entre a interação entre todos os intervenientes do sistema, nomeadamente: a) API do ZECA; b) API de reconhecimento de objetos; c) interface gráfica; d) robô ZECA. Com as três aplicações a funcionar em simultâneo, juntamente com o robô ZECA e com as instruções utilizadas anteriormente na avaliação do sistema parcial para as atividades de ensino de cores e formas geométricas, foram simuladas as duas atividades de ensino. Para a atividade de ensino de cores, procedeu-se à

gravação de um vídeo. Para a atividade de ensino de formas geométricas, procedeu-se à gravação de dois vídeos. A interação com o robô ZECA foi testada com dois adultos. Os links para os vídeos podem ser consultados na Tabela 35.

Tabela 35 – Vídeos com simulações das atividades de ensinos de cores e formas geométricas

Atividade de Ensino	**Vídeo**
Cores	https://youtu.be/7Zb7pBJVqRI
Figuras Geométricas (adulto 1)	https://youtu.be/sUCmBo-imIU
Figuras Geométricas (adulto 2)	https://youtu.be/XFScSsyxz2U

Para cada atividade de ensino, procedeu-se à realização do mesmo procedimento realizado na avaliação do sistema parcial, ou seja:

Tabela 36 – Lista de validações a verificar no sistema global

Instrução nº	**Validação**
1	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar o objeto correto
	Validar se é emitida a resposta de deteção correta
2	
	Validar se a pergunta é emitida conforme o pretendido
	Mostrar um objeto que não o pedido
	Validar se é emitida a resposta de deteção incorreta
	Validar se a pergunta é emitida novamente
	Mostra o objeto correto
3	
	Validar se é emitida a resposta de deteção correta
	Validar se a pergunta é emitida conforme o pretendido
	Mostra o objeto correto
	Validar se é emitida a resposta de deteção correta

Nos vídeos referidos acima é possível observar que todas as validações ocorreram com sucesso, ou seja, as ordens de instruções foram percorridas na ordem correta. Verificou-se também a correta emissão de feedback, tanto do lado do robô ZECA como da interface, assim como a

correta deteção dos objetos propostos em cada atividade de ensino, ou seja, para as cores (vermelho, verde e azul) e formas geométricas (quadrado, triângulo e círculo).

Em síntese, os resultados demonstram que o sistema global encontra-se em correto funcionamento para as atividades de ensino propostas.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste Capítulo serão apresentadas as principais conclusões deste trabalho e propostas de trabalhos futuros no sentido de dar continuidade ao trabalho desenvolvido.

6.1 Conclusões gerais

A utilização de robôs humanoides na educação pode melhorar as habilidades de interação, atenção e motivação das crianças com défice de atenção. Isto deve-se ao facto de os robôs humanoide possuírem a capacidade de exprimir emoções, permitindo-lhes assim estabelecer uma melhor conexão com as crianças. Neste trabalho foi abordada a utilização de um robô humanoide, ZECA, na instrução de duas atividades de ensino, cores e formas geométricas. O sistema foi desenvolvido com vista a ser utilizado em crianças com PEA.

Para a criação do sistema referido, foram desenvolvidas 3 aplicações, nomeadamente: a) API para o robô ZECA, responsável pela comunicação entre o computador e o robô; b) API de reconhecimento de objetos, responsável pelo reconhecimento de objetos presentes em uma imagem; c) interface gráfica, responsável pela configuração e execução das atividades de ensino. A avaliação do sistema desenvolvido ocorreu de duas formas distintas: a) parcial, isto é, a presença do robô ZECA foi substituída pela voz do computador. Esta avaliação possibilitou a validação de quase todo o sistema. Neste caso foi possível a avaliação do feedback emitido pelo computador (voz) e a deteção dos objetos; b) global, com o robô ZECA. Esta avaliação possibilitou, adicionalmente, a avaliação de emissão de feedback do robô (voz e gestos) e a comunicação entre o computador e o robô. Os resultados mostram que:

— Relativamente ao algoritmo de deteção utilizado, o YOLO, verificou-se que as redes treinadas com 50, 100, 150 e 200 fotografias por objeto apresentaram precisões de deteções distintas. Verificou-se que os melhores resultados de mPA e IoU foram conseguidos com 200

fotografias por objeto numa rede pré-treinada. Estas fotografias foram retiradas com diferentes rotações e distâncias ao objeto, de forma a minimizar os erros de deteção;

— Nos vídeos gravados com rotações em torno do próprio objeto, os objetos foram detetados com precisão superior a 99% e 74% na rede treinada para deteção de formas geométricas e cores, respetivamente;

— Na avaliação parcial do sistema verificou-se que a interface gráfica e a API de deteção de objetos comunicam corretamente entre si e que a interface procede ao envio correto de feedback positivo e negativo em concordância com o objeto detetado;

— Na avaliação global do sistema, testado em ambiente laboratorial, verificou-se que o sistema desenvolvido se encontra em correto funcionamento para as atividades de ensino propostas. O robô ZECA é capaz de emitir *feedback* positivo e negativo à pessoa que está a aprender a atividade de ensino, gesticulado os braços e a cabeça em concordância com o *feedback* que prende emitir.

Assim sendo, o sistema desenvolvido no âmbito desta dissertação para o ensino de cores e formas geométricas encontram-se operacional e pronto a ser testado com crianças com PEA.

6.2 Trabalhos futuros

Tendo em consideração que a interface gráfica desenvolvida no âmbito desta dissertação possibilita a inserção de novas atividades de ensino, os trabalhos futuros propostos são:

— Inserção de novas atividades de ensino como, por exemplo, o ensino de onomatopeias de animais, cão, gato, macaco. Outro aspeto bastante relevante é a adequação das atividades de ensino a lecionar à idade atual da criança e ao ritmo de aprendizagem da mesma;

— A utilização de versões mais atuais do YOLO, de forma a melhorar e velocidade e a percentagem de deteção dos objetos;

— Melhoramento da interface gráfica através da programação por blocos, *blokly*, permitindo ao utilizador ter mais possibilidades ao nível das instruções. Esta programação possibilita ao utilizador o fornecimento de mais sequências de instrução, permitindo assim um maior grau de personalização da atividade de ensino;

— Ainda no que diz respeito à interface gráfica, seria interessante que o utilizador pudesse desenvolver as próprias atividades de ensino através da etiquetagem e *uplod* das fotografias para o treino da rede;

— Teste do sistema desenvolvido em crianças com PEA. Seria interessante a realização de um estudo do sistema desenvolvido em crianças com PEA para avaliação da aprendizagem de cores e formas geométricas, da interação robô-criança e o nível de atenção das crianças nas atividades em questão. Paralelamente, seria interessante compreender se a utilização do robô ZECA conduz a um melhoramento de outras *skills* à parte das atividades de ensino lecionadas, nomeadamente a interação e comunicação com pessoas.

REFERÊNCIAS BIBLIOGRÁFICAS

Alcorn, A. M., Ainger, E., Charisi, V., Mantinioti, S., Petrović, S., Schadenberg, B. R., Tavassoli, T., & Pellicano, E. (2019). Educators' Views on Using Humanoid Robots With Autistic Learners in Special Education Settings in England. *Frontiers in Robotics and AI*, *6*. https://doi.org/10.3389/frobt.2019.00107

Alnajjar, F., Cappuccio, M. L., Mubin, O., Arshad, R., & Shahid, S. (2020). Humanoid Robots and Autistic Children: A Review on Technological Tools to Assess Social Attention and Engagement. *International Journal of Humanoid Robotics*, *17*(06). https://doi.org/10.1142/S0219843620300019

Barakova, E. I., Bajracharya, P., Willemsen, M., Lourens, T., & Huskens, B. (2015). Long-term LEGO therapy with humanoid robot for children with ASD. *Expert Systems*, *32*(6), 698–709. https://doi.org/10.1111/exsy.12098

Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B., & Tanaka, F. (2018). Social robots for education: A review. *Science Robotics*, *3*(21). https://doi.org/10.1126/scirobotics.aat5954

Bharatharaj, J., Huang, L., Mohan, R., Al-Jumaily, A., & Krägeloh, C. (2017). Robot-Assisted Therapy for Learning and Social Interaction of Children with Autism Spectrum Disorder. *Robotics*, *6*(1), 4. https://doi.org/10.3390/robotics6010004

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

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. https://doi.org/10.1023/A:1009715923555

Carvalho, E., Ferreira, B., Ferreira, M., Filho, G., Ueyama, J., & Pessin, G. (2016). Uso de Redes Neurais Recorrentes para Localização de Agentes em Ambientes Internos. *Symposium on Knowledge Discovery, Mining and Learning (KDMiLe)*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Costa, S. (2014). *Affective robotics for socio-emotional development in children with autism spectrum disorders, Tese de Doutoramento, Escola de Engenharia da Universidade do Minho* [Tese de Doutoramento]. Universidade do Minho.

Costa, S., Soares, F., Pereira, A. P., Santos, C., & Hiolle, A. (2014a). Building a game scenario to encourage children with autism to recognize and label emotions using a humanoid robot. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, 820–825. https://doi.org/10.1109/ROMAN.2014.6926354

Costa, S., Soares, F., Pereira, A. P., Santos, C., & Hiolle, A. (2014b). A pilot study using imitation and storytelling scenarios as activities for labelling emotions by children with autism using a humanoid robot. *4th International Conference on Development and Learning and on Epigenetic Robotics*, 299–304. https://doi.org/10.1109/DEVLRN.2014.6982997

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Diehl, J. J., Schmitt, L. M., Villano, M., & Crowell, C. R. (2012). The clinical use of robots for individuals with Autism Spectrum Disorders: A critical review. *Research in Autism Spectrum Disorders*, *6*(1), 249–262. https://doi.org/10.1016/j.rasd.2011.05.006

Edward R. Dougherty. (2020). *Digital Image Processing Methods* (CRC Press).

Efstratiou, R., Karatsioras, C., Papadopoulou, M., Papadopoulou, C., Lytridis, C., Bazinas, C., Papakostas, G. A., & Kaburlasos, V. G. (2021). *Teaching Daily Life Skills in Autism Spectrum Disorder (ASD) Interventions Using the Social Robot Pepper* (pp. 86–97). https://doi.org/10.1007/978-3-030-67411-3_8

Esther Ben Itzchak, & Ditza A. Zachor. (2011). Who benefits from early intervention in autism spectrum disorders? *Research in Autism Spectrum Disorders*, *5*(1), 345–350. https://doi.org/10.1016/j.rasd.2010.04.018

Fachantidis, N., Syriopoulou-Delli, C. K., Vezyrtzis, I., & Zygopoulou, M. (2020). Beneficial effects of robot-mediated class activities on a child with ASD and his typical classmates. *International Journal of Developmental Disabilities*, *66*(3), 245–253. https://doi.org/10.1080/20473869.2019.1565725

Feil-Seifer, D., & Mataric, M. J. (2005). Socially Assistive Robotics. *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, *ICORR 2005*(465–468), 465–468. https://doi.org/10.1109/ICORR.2005.1501143

Gelsomini, M., Leonardi, G., Degiorgi, M., Garzotto, F., Penati, S., Silvestri, J., Ramuzat, N., & Clasadonte, F. (2017). Puffy - an Inflatable Mobile Interactive Companion for Children with Neurodevelopmental Disorders. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2599–2606. https://doi.org/10.1145/3027063.3053245

Golestan, S., Soleiman, P., & Moradi, H. (2017). Feasibility of using sphero in rehabilitation of children with autism in social and communication skills. *2017 International Conference on Rehabilitation Robotics (ICORR)*, 989–994. https://doi.org/10.1109/ICORR.2017.8009378

Hand, D. J., & Yu, K. (2001). Idiot's Bayes?Not So Stupid After All? *International Statistical Review*, *69*(3), 385–398. https://doi.org/10.1111/j.1751-5823.2001.tb00465.x

Hawon Lee, & Eunja Hyun. (2015). The Intelligent Robot Contents for Children with Speech-Language Disorder. *Educational Technology & Society*.

Henschel, A., Hortensius, R., & Cross, E. S. (2020). Social Cognition in the Age of Human–Robot Interaction. *Trends in Neurosciences*, *43*(6), 373–384. https://doi.org/10.1016/j.tins.2020.03.013

Hyman, S. L., Levy, S. E., Myers, S. M., Kuo, D. Z., Apkon, S., Davidson, L. F., Ellerbeck, K. A., Foster, J. E. A., Noritz, G. H., Leppert, M. O., Saunders, B. S., Stille, C., Yin, L., Weitzman, C. C., Childers, D. O., Levine, J. M., Peralta-Carcelen, A. M., Poon, J. K., Smith, P. J., … Bridgemohan, C. (2020). Identification, Evaluation, and Management of Children With Autism Spectrum Disorder. *Pediatrics*, *145*(1). https://doi.org/10.1542/peds.2019-3447

Iqbal, Z., Khan, M. A., Sharif, M., Shah, J. H., ur Rehman, M. H., & Javed, K. (2018). An automated detection and classification of citrus plant diseases using image processing techniques: A review. *Computers and Electronics in Agriculture*, *153*, 12–32. https://doi.org/10.1016/j.compag.2018.07.032

Irfan, B., Ramachandran, A., Spaulding, S., Glas, D. F., Leite, I., & Koay, K. L. (2019). Personalization in Long-Term Human-Robot Interaction. *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 685–686. https://doi.org/10.1109/HRI.2019.8673076

Jabar H. Yousif, & Mohammed J. Yousif. (2020). Humanoid Robot as Assistant Tutor for Autistic Children. *International Journal of Computation and Applied Sciences*.

Korneder, J., Louie, W.-Y. G., Pawluk, C. M., Abbas, I., Brys, M., & Rooney, F. (2021). Robot-mediated interventions for teaching children with ASD: a new intraverbal skill. *Assistive Technology*, 1–10. https://doi.org/10.1080/10400435.2021.1930284

Kose, H., Akalin, N., Yorganci, R., Ertugrul, B. S., Kivrak, H., Kavak, S., Ozkul, A., Gurpinar, C., Uluer, P., & Ince, G. (2015). *iSign: An Architecture for Humanoid Assisted Sign Language Tutoring* (pp. 157–184). https://doi.org/10.1007/978-3-319-12922-8_6

Kumazaki, H., Yoshikawa, Y., Yoshimura, Y., Ikeda, T., Hasegawa, C., Saito, D. N., Tomiyama, S., An, K., Shimaya, J., Ishiguro, H., Matsumoto, Y., Minabe, Y., & Kikuchi, M. (2018). The impact of robotic intervention on joint attention in children with autism spectrum disorders. *Molecular Autism*, *9*(1), 46. https://doi.org/10.1186/s13229-018-0230-8

Lei, Y. (2017). Individual intelligent method-based fault diagnosis. *Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery*, 67–174. https://doi.org/10.1016/B978-0-12-811534-3.00003-2

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). *Focal Loss for Dense Object Detection*.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014). *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* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

Maenner, M. J., Shaw, K. A., Bakian, A. v., Bilder, D. A., Durkin, M. S., Esler, A., Furnier, S. M., Hallas, L., Hall-Lande, J., Hudson, A., Hughes, M. M., Patrick, M., Pierce, K., Poynter, J. N., Salinas, A., Shenouda, J., Vehorn, A., Warren, Z., Constantino, J. N., … Cogswell, M. E. (2021). Prevalence and Characteristics of Autism Spectrum Disorder Among Children Aged 8 Years — Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2018. *MMWR. Surveillance Summaries*, *70*(11), 1–16. https://doi.org/10.15585/mmwr.ss7011a1

Marti, P., & Giusti, L. (2010). A robot companion for inclusive games: A user-centred design perspective. *2010 IEEE International Conference on Robotics and Automation*, 4348–4353. https://doi.org/10.1109/ROBOT.2010.5509385

Mohammadi, V., & Minaei, S. (2019). Artificial Intelligence in the Production Process. *Engineering Tools in the Beverage Industry: Volume 3: The Science of Beverages*, 27–63. https://doi.org/10.1016/B978-0-12-815258-4.00002-0

Moorthy, R. S., & Pugazhenthi, S. (2017). Teaching Psychomotor Skills to Autistic Children by Employing a Robotic Training Kit: A Pilot Study. *International Journal of Social Robotics*, *9*(1), 97–108. https://doi.org/10.1007/s12369-016-0375-6

Nikolopoulos, C., Kuester, D., Sheehan, M., Dhanya, S., Herring, W., Becker, A., & Bogart, L. (2010). Socially Assistive Robots and Autism. *Solid State Phenomena*, *166–167*, 315–320. https://doi.org/10.4028/www.scientific.net/SSP.166-167.315

Nunez, E., Matsuda, S., Hirokawa, M., & Suzuki, K. (2015). *Humanoid Robot Assisted Training for Facial Expressions Recognition Based on Affective Feedback* (pp. 492–501). https://doi.org/10.1007/978-3-319-25554-5_49

Pereira, A. P. S., Freitas, H., Costa, P., Silva, V. C., Soares, F., & Esteves, J. S. (2017). Using a humanoid robot as the promoter of interaction with children in the context of educational games. *International Journal of Mechatronics and Applied Mechanics*.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Rakhymbayeva, N., Amirova, A., & Sandygulova, A. (2021). A Long-Term Engagement with a Social Robot for Autism Therapy. *Frontiers in Robotics and AI*, *8*. https://doi.org/10.3389/frobt.2021.669972

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

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*.

Resende, D. A. (2022). *Nº 125 - 2021: Fashion Analysis from Surveillance Data taken from UAVs (Tese Mestrado)*.

Ricks, D. J., & Colton, M. B. (2010). Trends and considerations in robot-assisted autism therapy. *2010 IEEE International Conference on Robotics and Automation*, 4354–4359. https://doi.org/10.1109/ROBOT.2010.5509327

Robins, B., Dautenhahn, K., te Boekhorst, R., & Billard, A. (2004). Effects of Repeated Exposure to a Humanoid Robot on Children with Autism. In *Designing a More Inclusive World* (pp. 225–236). Springer London. https://doi.org/10.1007/978-0-85729-372-5_23

Sakka, S., Gaboriau, R., Picard, J., Redois, E., Parchantour, G., Sarfaty, L., Navarro, S., & Barreau, A. (2018). *Rob'Autism: How to Change Autistic Social Skills in 20 Weeks* (pp. 261–274). https://doi.org/10.1007/978-3-319-59972-4_19

Saleh, M. A., Hanapiah, F. A., & Hashim, H. (2021). Robot applications for autism: a comprehensive review. *Disability and Rehabilitation: Assistive Technology*, *16*(6), 580–602. https://doi.org/10.1080/17483107.2019.1685016

Santiago Teles de Menezes, R., Marrocos Magalhaes, R., & Maia, H. (2020). Object Recognition Using Convolutional Neural Networks. In *Recent Trends in Artificial Neural Networks - from Training to Prediction*. IntechOpen. https://doi.org/10.5772/intechopen.89726

Scassellati, B., Henny Admoni, & Matarić, M. (2012). Robots for Use in Autism Research. *Annual Review of Biomedical Engineering*, *14*(1), 275–294. https://doi.org/10.1146/annurev-bioeng-071811-150036

Shamsuddin, S., Yussof, H., Hanapiah, F. A., Mohamed, S., Jamil, N. F. F., & Yunus, F. W. (2015). Robot-assisted learning for communication-care in autism intervention. *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*, 822–827. https://doi.org/10.1109/ICORR.2015.7281304

Silva, G. R. (2018). *DETECÇÃO DE OBJETOS EM IMAGENS UTILIZANDO TÉCNICAS DE APRENDIZAGEM PROFUNDA (Tese Mestrado)*.

Silva, V., Soares, F., & Sena Esteves, J. (2017). Mirroring and recognizing emotions through facial expressions for a RoboKind platform. *2017 IEEE 5th Portuguese Meeting on Bioengineering (ENBENG)*, 1–4. https://doi.org/10.1109/ENBENG.2017.7889480

Stanton, C. M., Kahn Jr., P. H., Severson, R. L., Ruckert, J. H., & Gill, B. T. (2008). Robotic animals might aid in the social development of children with autism. *Proceedings of the 3rd International Conference on Human Robot Interaction - HRI '08*, 271. https://doi.org/10.1145/1349822.1349858

Sun, A., Li, Y., Huang, Y.-M., & Li, Q. (2018). *The Exploration of Facial Expression Recognition in Distance Education Learning System* (pp. 111–121). https://doi.org/10.1007/978-3-319-99737-7_11

Taheri, A., Meghdari, A., Alemi, M., & Pouretemad, H. (2017). Teaching Music to Children with Autism: A Social Robotics Challenge. *Scientia Iranica*, *0*(0), 0–0. https://doi.org/10.24200/sci.2017.4608

Tan, M., Pang, R., & Le, Q. v. (2019). *EfficientDet: Scalable and Efficient Object Detection*.

Tapus, A., Mataric, M., & Scassellati, B. (2007). Socially assistive robotics [Grand Challenges of Robotics]. *IEEE Robotics & Automation Magazine*, *14*(1), 35–42. https://doi.org/10.1109/MRA.2007.339605

Tuna, A., & Tuna, G. (2019). The Use of Humanoid Robots with Multilingual Interaction Skills in Teaching a Foreign Language: Opportunities, Research Challenges and Future Research Directions. *Center for Educational Policy Studies Journal*, *9*(3). https://doi.org/10.26529/cepsj.679

van der Hallen, R., Manning, C., Evers, K., & Wagemans, J. (2019). Global Motion Perception in Autism Spectrum Disorder: A Meta-Analysis. *Journal of Autism and Developmental Disorders*, *49*(12), 4901–4918. https://doi.org/10.1007/s10803-019-04194-8

Verner, I. M., Polishuk, A., & Krayner, N. (2016). Science Class with RoboThespian: Using a Robot Teacher to Make Science Fun and Engage Students. *IEEE Robotics and Automation Magazine*, *23*(2), 74–80. https://doi.org/10.1109/MRA.2016.2515018

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, I-511-I–518. https://doi.org/10.1109/CVPR.2001.990517

Wood, L. J., Zaraki, A., Robins, B., & Dautenhahn, K. (2021). Developing Kaspar: A Humanoid Robot for Children with Autism. *International Journal of Social Robotics*, *13*(3), 491–508. https://doi.org/10.1007/s12369-019-00563-6

Zheng, Z., Young, E. M., Swanson, A. R., Weitlauf, A. S., Warren, Z. E., & Sarkar, N. (2016). Robot-Mediated Imitation Skill Training for Children With Autism. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *24*(6), 682–691. https://doi.org/10.1109/TNSRE.2015.2475724

Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). *TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios*.