UNIVERSIDADE
BEIRA INTERIOR

# SeeMyHome: Sistema de Visão Computacional assente numa Aplicação Móvel destinada a pessoas invisuais

**(Versão final após defesa)**

**João Pedro Afonso Branco**

Dissertação para obtenção de Grau de Mestre em
**Engenharia Informática**
(2º Ciclo de Estudos)

Orientador: Prof. Doutor Paulo André Pais Fazendeiro

**29 de dezembro de 2022**

## Declaração de Integridade

Eu, João Pedro Afonso Branco, que abaixo assino, estudante com o número de inscrição M11019, do Curso de Mestrado em Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, e que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assim assumo na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 29/12/2022

Assinado por: João Pedro
Afonso Branco
Identificação: BI15663253
Data: 2022-12-29 às 12:45:51

# Agradecimentos

Apesar de todo o caminho percorrido durante a minha vida académica, incluindo a realização deste projeto, ser fruto da minha responsabilidade, dedicação, paciência, esforço e vontade própria, na verdade nada se faz sozinho e por de trás de cada barreira ultrapassada, houve sempre alguém a apoiar e ajudar-me. Por isso, a conclusão deste projeto, bem como da grande maior parte da minha vida académica não seria possível sem o apoio e colaboração de todos aqueles que me acompanharam, a quem eu agradeço profundamente:

Primeiramente a Deus pela determinação e saúde que sempre me deu, especialmente na realização deste projeto durante a situação pandémica que ainda se está a viver.

Aos meus pais e à minha irmã pelo amor, dedicação e apoio que me deram ao longo deste percurso universitário.

Ao meu orientador Professor Doutor Paulo André Pais Fazendeiro pelos conselhos, conhecimento, apoio e orientação que me deu ao longo do desenvolvimento deste projeto. Pela sua total disponibilidade, dedicação e confiança neste trabalho, promovendo o meu crescimento pessoal, académico e profissional.

A todos os docentes do Departamento de Informática pelo conhecimento valioso que me facultaram durante todo o meu percurso universitário, enriquecendo os meus conhecimentos e preparando-me para a minha vida profissional.

Este trabalho foi apoiado pela ”Fundação para a Ciência e a Tecnologia” (FCT) no âmbito do projecto CPCA/A0/429960/2021 e desenvolvido em parte no Centro de Computação de Alto Desempenho da Universidade de Évora (HPC-UÉ).

# Resumo

A falta de visão é uma condição que geralmente impõe muitas barreiras e dificuldades na vida de uma pessoa. Uma simples tarefa como distinguir determinados objetos ou obter informação sobre o meio envolvente, acaba por representar uma grande adversidade. O principal objetivo deste trabalho é projetar uma solução proficiente e que seja capaz de oferecer apoio a pessoas com dificuldades moderadas de visão. A mesma traduz-se na criação de um sistema de visão computacional, mais especificamente uma aplicação móvel capaz de identificar objetos presentes no ambiente que rodeia o utilizador.

A primeira fase deste projeto de dissertação consistiu na realização de um estudo sistemático sobre o que atualmente está a ser desenvolvido ou já foi disponibilizado acerca de aplicações móveis destinadas a pessoas invisuais. Foi feito um levantamento de várias propostas de diversos autores, a fim de analisar as suas diferentes perspetivas de trabalho. Como resultado dessa análise, estabeleceram-se alguns desafios e estratégias inovadoras que irão permitir desenvolver uma aplicação móvel capaz de dar resposta ao problema em estudo. A descrição da solução aqui reportada elenca todas as técnicas e procedimentos necessários para organizar o respetivo trabalho. Apresenta também os fundamentos de uma abordagem conexionista genérica aplicada ao reconhecimento de objetos.

Numa segunda e última fase procedeu-se à construção do sistema proposto para esta dissertação. Construi-se uma aplicação móvel, denominada por SeeMyHome, que recorre às áreas da Visão Computacional e Inteligência Artificial para prestar apoio a pessoas invisuais, proporcionando-lhes uma maior independência e uma melhor qualidade de vida. A nossa abordagem consistiu na construção de um sistema eficiente de deteção de objetos em tempo real, que procura replicar a função do olho humano por meio do processamento de imagens obtidas por um smartphone e de algoritmos capazes de reconhecer e identificar objetos nas mesmas. A solução aqui divulgada utiliza uma rede neuronal convolucional (CNN) para classificar imagens e tecnologia *Text-To-Speech* para informar o utilizador invisual sobre tudo o que se encontra ao seu redor. SeeMyHome consegue destacar-se das demais aplicações atualmente disponibilizadas e propostas, devido à adição de um novo recurso que concede a oportunidade de o utilizador, através de um assistente visual, poder adicionar qualquer objeto novo que deseje na aplicação. A Aprendizagem por Transferência e o Aumento de dados demonstraram que é possível treinar uma rede neuronal, que anteriormente aprendeu a detetar características relevantes num grande conjunto de dados de treino, para reconhecer novas classes de objetos num conjunto de dados reduzido, onde os resultados de precisão e de classificação atingem valores superiores aos métodos tradicionais.

# Palavras-chave

Pessoas Invisuais, Aplicação *Android*, Visão Computacional, Rede Neuronal Artificial, Rede Neural Convolucional, Reconhecimento de Objetos, *Text-To-Speech*, Aprendizagem por Transferência, Aumento de Dados.

# Abstract

A visual impairment is a condition that usually imposes several barriers and difficulties in a person's life. A simple task, such as distinguishing certain objects or obtaining information about their surroundings, ends up being very problematic. The main objective of this work is to design a proficient solution capable of offering support to people with moderate vision difficulties. This will be achieved through the creation of a computer vision system, more specifically a mobile application capable of identifying objects present in the environment that surrounds the user.

The first phase of this dissertation project consisted in conducting a systematic study to identify what has already been made available and what is currently being developed in the area of mobile applications for visually impaired people. An initial research of several proposals from several authors was carried out in order to analyse their different work perspectives. As a result of this analysis, some challenges and innovative strategies were established, allowing for the development of a mobile application capable of answering to this problem. The solution description here reported contains all the techniques and procedures needed to organise this work. Finally, a connectionist approach applied to object recognition is presented.

The second and final phase, the proposed system for this dissertation was constructed. A mobile application, called SeeMyHome, is built using Computer Vision and Artificial Intelligence to provide support to blind people, giving them greater independence and a better quality of life. Our approach was to build an efficient real-time object detection system that seeks to replicate the function of the human eye by processing images taken by a smartphone and algorithms capable of recognizing and identifying objects in them. The solution disclosed here uses a convolutional neural network (CNN) to classify images and Text-to-speech technology to inform the blind user about everything around him. SeeMyHome manages to stand out from other applications currently available and proposed, due to the addition of a new feature that grants the user, through a visual assistant, the opportunity to add any new object they wish in the application.Transfer learning and Data Augmentation have shown that it is possible to train a neural network, which previously learned to detect relevant features on a large training dataset, to recognize new classes of objects on a small dataset, where the accuracy and classification results are higher than traditional methods.

# Keywords

Visual Impairment, Android Application, Artificial Neural Network, Convolutional Neural Network, Object Recognition, *Text-To-Speech*, Transfer Learning, Data Augmentation.

# Índice

# Lista de Figuras

# Lista de Tabelas

# Lista de Acrónimos

**AI** Artificial Intelligence

**API** Application Programming Interface

**AT** Assistive Technology

**CIFAR** Canadian Institute for Advanced Research

**CMOS** Complementary Metal-Oxide-Semiconductor

**CNN** Convolutional Neural network

**COCO** Common Objects in Context

**CUDA** Compute Unified Device Architecture

**DTMF** Dual-Tone Multi-Frequency

**EAN** European Article Number

**FFNN** Feed Forward Neural Network

**FPS** Frames Per Second

**GPL** General Public License

**GPS** Global Positioning System

**GPU** Graphics Processing Unit

**HTTP** Hyper-Text Transfer Protocol

**ID** Identification

**IEEE** Institute of Electrical and Electronics Engineers

**IP** Internet Protocol

**JPG** Joint Photographic Experts Group

**JSON** JavaScript Object Notation

**mAP** Mean Average Precision

**MDPI** Multidisciplinary Digital Publishing Institute

**MLP** Multi-layer perceptron

**ML** Machine Learning

**PHP** Hypertext Preprocessor

**QR** Quick Response

**REST** Representational State Transfer

**REST** Representational State Transfer

**RFID** Radio Frequency IDentification

**ROI** Region of Interest

**RPN** Region Proposal Network

**ReLU** Rectified Linear Unit

**SCP** Secure Copy Protocol

**SDK** Software Development kit

**SOM** System On Module

**SQL** Structured Query Language

**SSD** Single Shot Detector

**SSH** Secure Shell Protocol

**SVM** Support Vector Machines

**TB** Terabyte

**UBI** Universidade da Beira Interior

**URI** Uniform Resource Identifie

**URL** Uniform Resource Locator

**VGG** Visual Geometry Grou

**VOC** Visual Object Classes

**WHO** World Health Organization

**XML** Extensible Markup Language

**YOLO** You Only Look Once

**ZIP** Compressed File

# Capítulo 1

# Introdução

## 1.1 Enquadramento e motivação

Nos últimos anos, a tecnologia móvel sofreu uma enorme evolução, derivada da progressiva massificação e popularidade dos *smartphones* atuais. Não há dúvidas que a relação dos dispositivos móveis com a sociedade está cada vez mais sólida, assumindo parte integrante da nossa vida diária. A grande flexibilidade, disponibilidade e utilidade que eles nos oferecem determinou uma crescente procura por estes aparelhos eletrónicos, e consequentemente uma maior dependência gerada pelo uso dos mesmos. Prova disso, é a pesquisa publicada pela Strategy Analytics, em que se estima que a base global de utilizadores de *smartphones* nos últimos anos aumentou acentuadamente, atingindo um recorde de 3,95 mil milhões em junho de 2021. Nessa mesma data, o planeta contabilizava um total de cerca de 7,90 mil milhões de pessoas, o que significa que 50% da população mundial tem um *smartphone* [19].

Atualmente, é impressionante e indiscutível a forma como um simples telemóvel se tornou uma ferramenta tão preciosa na nossa vida. Esta vertente, é impulsionada pelas grandes vantagens da sua utilização. Por exemplo, comunicar com familiares e amigos distantes e usufruir de diversas aplicações tecnológicas muito úteis. Para a maior parte das pessoas, algumas dessas aplicações representam apenas uma forma de facilitar as suas tarefas diárias, como por exemplo: efetuar compras (alimentação, peças de roupa, entre outras), obter informações acerca de transportes públicos, etc. No entanto, para pessoas com dificuldades visuais, o desenvolvimento de aplicações projetadas para atender a essa necessidade específica acaba por ter um impacto e uma diferença ainda maior na sua vida, tornando possível a sua independência e uma maior conexão com o mundo ao seu redor.

Segundo o relatório mundial sobre visão, publicado em 2019 pela World Health Organization (WHO), existem pelo menos 2,2 mil milhões de pessoas com deficiência visual no mundo [20]. A visão é um dos órgãos mais importantes para o ser humano. É através dele, que conseguimos ter a perceção do ambiente que se encontra a nossa volta. Para uma pessoa invisual, a falta desta capacidade, geralmente, traz bastantes limitações nas suas tarefas diárias. Obter informação sobre possíveis obstáculos que se encontrem à sua frente, torna-se um desafio bastante rigoroso. De acordo com o estudo [21], numa entrevista a trezentos indivíduos portadores de deficiência visual, 13% dos participantes sofreram acidentes ao bater com a cabeça num obstáculo inesperado e 7% sofreram quedas enquanto caminham, pelo menos uma vez por mês. Sendo assim, para uma pessoa invisual, é útil ter ao seu dispor um sistema que identifique obstáculos que estão à sua frente, de forma a alertá-la de diversos perigos que possam aparecer no seu caminho. Para além da sua segurança, também é muito importante para o utilizador, que o sistema seja capaz de reconhecer os objetos e indicar-lhe o seu respetivo nome, uma vez que isso permitirá que ele seja capaz de realizar determina-

das tarefas de forma mais autónoma. Portanto, desenvolver aplicações móveis eficientes e frutíferas que concedam recursos de apoio a esta causa, torna-se bastante motivador e um grande desafio, de modo a contribuir para uma melhor qualidade de vida de um segmento significativo da nossa população.

## 1.2 Descrição do problema

O problema proposto para esta dissertação traduz-se na construção de um sistema, concretizado com uma aplicação móvel, que servirá de apoio a pessoas invisuais. A aplicação deverá incorporar um recurso que seja capaz de detetar e classificar objetos presentes em imagens recolhidas do ambiente que rodeia o utilizador. Os resultados do reconhecimento serão transmitidos por meio de indicações sonoras. Esta funcionalidade promete oferecer uma maior confiança sobre o ambiente envolvente, e de certa forma, permitir uma maior autonomia e segurança nas deslocações.

Na resolução deste problema, os dispositivos móveis apresentam potencial elevado na procura por uma solução eficiente, uma vez que reúnem todas as ferramentas necessárias para concretização do objetivo proposto. Através da câmara incorporada no telemóvel, é possível simular o sistema ótico humano e receber informação sobre o meio envolvente, mediante a recolha de imagens. Para além disso, os *smartphones* possuem recursos (*Text-To-Speech* e *Speech-To-Text*) que permitem a comunicação entre o utilizador e o dispositivo.

De uma forma mais detalhada, o sistema que se pretende desenvolver nesta dissertação deverá contar com os seguintes requisitos funcionais:

1. Recolha de imagens do meio envolvente;
2. Deteção e classificação dos objetos presentes na cena;
3. A interface deve ser pensada atendendo aos previsíveis utilizadores e privilegiar as indicações sonoras;
4. Utilização (recolha, tratamento e carregamento) de conhecimento cooperativo advindo de diversas fontes;
5. Transição/teste simples entre/de modelos alternativos de classificação.

## 1.3 Objetivos e hipóteses de investigação

O principal objetivo desta dissertação é desenvolver um sistema para assistência a pessoas invisuais que permita expandir a sua perceção do meio envolvente. Primeiramente, pretende-se projetar uma solução para a resolução do problema proposto e desenvolver um protótipo. No final, o mesmo deverá ser testado, com o objetivo de ser disponibilizado, inicialmente, para a plataforma *Android*. Propomos começar pelo sistema operativo *Android*, pelo facto de este ser o mais utilizado em todo mundo. Segundo o site de estatísticas Statcounter Global Stats, em setembro de 2022, 71,55% dos *smartphones* em todo o mundo funcionam com o

sistema operativo *Android* [22]. Tal não inviabiliza a futura distribuição para outras plataformas, de modo a abranger o máximo possível de utilizadores. Neste trabalho, tendo em conta o problema proposto, foram formuladas algumas hipóteses de investigação:

1. Com o sistema proposto é possível auxiliar um invisual a localizar um determinado objeto no seu ambiente envolvente ainda que em diferentes condições de iluminação e parcialmente oculto do campo de visão.

2. A combinação de características de *edge computing* e de *cloud computing* permite treinar ou efetuar a reaprendizagem de um modelo de rede neuronal capaz de classificar novos objetos de uma forma transparente para o utilizador final.

3. A implementação de um ambiente colaborativo entre utilizadores permite criar modelos para reconhecimento de objetos com um melhor desempenho.

## 1.4 Organização do documento

De modo a refletir o trabalho que foi feito, este projeto de dissertação encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, o enquadramento para o mesmo, a motivação para a sua escolha, os seus objetivos, hipóteses de investigação e a respetiva organização do documento.

2. O segundo capítulo – **Estado da Arte** – apresenta um estudo bibliográfico referente à analise de diferentes propostas e resultados obtidos de diferentes autores. Para além disso, inclui também uma revisão de aplicações móveis atualmente disponíveis para o publico alvo, bem como alguns sistemas diferentes que abordam a funcionalidade de detetar objetos. Por fim, é apresentada uma análise critica de todo o trabalho realizado e alguns desafios que se encontram em aberto.

3. O terceiro capítulo – **Abordagem conexionista aplicada ao reconhecimento de objetos** – discute todos os conceitos necessários para adquirir um conhecimento amplo e produtivo sobre como é feito o reconhecimento de objetos através de um modelo de rede neuronal artificial. Primeiro será introduzida a noção de rede neuronal artificial. Mais a frente, falar-se-á um pouco sobre redes neuronais profundas, especialmente as redes neuronais convolucionais (CNN). De seguida, será explicado como é realizada a aprendizagem e o reconhecimento de padrões por meio de um modelo de rede neuronal. Por fim, divulgar-se-á um estudo comparativo de diferentes algoritmos de deteção de objetos.

4. O quarto capítulo – **Descrição da solução proposta** – descreve a solução projetada para a resolução do problema proposto nesta dissertação. Fazem parte deste capítulo, a arquitetura da solução, o conceito de servidor em nuvem aplicado, a calendarização do projeto e os resultados que se pretendem alcançar.

5. O sexto capítulo – **Detalhes de Implementação** – procura revelar alguns aspetos de implementação fundamentais para uma compreensão mais aprofundada de todo o sistema desenvolvido.
6. O sétimo capítulo – **Testes e Resultados Experimentais** – divulga os resultados obtidos após a realização de experiências e testes ao *software* desenvolvido.
7. O oitavo capítulo – **Conclusão e Trabalho Futuro** – apresenta as conclusões resultantes do trabalho realizado, bem como o trabalho futuro.

# Capítulo 2

# Estado de Arte

## 2.1 Introdução

Nos tempos atuais, um telemóvel é visto como um grande aliado na acessibilidade a pessoas invisuais, dado que é dispositivo leve, movível e com forte poder tecnológico. Com base neste conhecimento, vários criadores começaram a investigar ferramentas proficientes e capazes de proporcionar um maior conforto e qualidade de vida a pessoas invisuais. É o caso das aplicações móveis que apresentam recursos inovadores. Muitos investigadores têm usado a tecnologia móvel para desenvolver aplicações com funcionalidades úteis, como por exemplo, detetar objetos, identificar dinheiro, reconhecer cores, entre outras. Todas essas atividades representam uma nova forma de contornar a falta de visão, e disponibilizar uma abordagem alternativa e eficaz na realização de uma determinada tarefa que só poderia ser feita com base na visão. Posto isto, é evidente que a construção de aplicações móveis com este propósito não seja um assunto novo, mas sim, um tema que tem vindo a ser trabalhado e discutido ao longo destes últimos anos. No entanto, apesar dos avanços significativos desta tecnologia, é importante notar que ainda temos muitos passos a dar, para conseguirmos chegar a um sistema que seja fortemente capaz de atender a todas as necessidades dos utilizadores.

Uma revisão de literatura, demonstrou que o desenvolvimento de trabalhos dentro do contexto desta dissertação tem sido alvo de grande investigação. Neste capítulo, apresentamos alguns projetos e sistemas já publicados, examinando as várias propostas e resultados obtidos de diferentes autores. De seguida, incluímos uma revisão de algumas aplicações móveis já existentes, bem como alguns sistemas que adotam tecnologias diferentes para detetar objetos. Por fim, concluímos este capítulo com uma análise critica sobre tudo o que nele foi abordado e indicamos os desafios que se encontram em aberto.

É importante mencionar que, para a elaboração deste trabalho bibliográfico, foram utilizadas as seguintes bases de dados científicas: Institute of Electrical and Electronics Engineers (IEEE), Multidisciplinary Digital Publishing Institute (MDPI), ScienceDirect e Google Scholar. Como forma de pesquisa, recorremos às seguintes palavras-chave para identificar artigos de interesse: “Visually Impaired + Object Detection + mobile application”. As mesmas permitiram garantir que as publicações científicas encontradas se baseiam no desenvolvimento de aplicações móveis que empregam claramente o recurso de detetar objetos com a finalidade principal de apoiar pessoas invisuais.

## 2.2 Revisão Bibliográfica

O reconhecimento de objetos é apresentado em vários estudos, através da implementação de uma aplicação móvel com mecanismos de deteção e classificação de imagens. Esta atividade

é muito útil na assistência a pessoas invisuais, e, portanto, foi adotada pela maior parte dos autores que utilizam a tecnologia móvel para criar recursos que sejam de interesse a pessoas com dificuldades visuais. Alguns projetos, incluem ainda mais funcionalidades (reconhecimento de texto, de dinheiro, de rosto, etc.), no entanto, nesta revisão bibliográfica, demos especial atenção à deteção e ao reconhecimento de objetos. Consideramos que a chave de sucesso para obter um sistema robusto e eficiente centra-se, na forma como o reconhecimento foi implementado, na quantidade de objetos identificados pela aplicação (base de dados), na precisão de classificação e, mais importante, na *interface* criada que deve ser desenhada e implementada com especial atenção, a fim de vir a ser utilizada por pessoas invisuais.

Recentemente, foram publicadas várias propostas de autores que fazem uso da tecnologia móvel para ajudar pessoas com problemas de visão. É o caso de [23] (2021), onde os autores propõem um sistema simples que promete ajudar pessoas invisuais a compreender o que os rodeia. A sua aplicação móvel foi desenvolvida com o objetivo de tentar replicar o órgão responsável pela visão do ser humano, ou seja, os nossos olhos, através da ajuda da câmera do dispositivo móvel e do recurso de deteção de objetos, em tempo real. Este sistema foi desenvolvido no ambiente de trabalho Android Studio e conta com duas funcionalidades: reconhecimento de objetos e de texto. O nome dos objetos identificados e o texto reconhecido, são informados ao utilizador por meio de áudio, utilizando técnicas de *Text-To-Speech*.

Nesta proposta, os autores fizeram um estudo comparativo de diferentes algoritmos para detetar objetos. Entre R-CNN [24], Fast R-CNN [25], Faster R-CNN [26], You Only Look Once (YOLO) [27] e Single Shot Detector (SSD) [28], chegaram à conclusão de que o mais adequado a implementar no seu projeto era o SSD, uma vez que é o melhor algoritmo que equilibra velocidade e precisão, com um valor de 74,3 Mean Average Precision (mAP), o mais alto entre os demais modelos. Sendo assim, construíram a sua aplicação utilizando a Application Programming Interface (API) do TensorFlow , juntamente com o modelo Common Objects in Context (COCO) SSD MobileNet v1. Atualmente, o mesmo foi treinado para detetar e localizar noventa categorias de objetos diferentes. Sempre que uma imagem for enviada, como entrada, para este modelo, ele irá produzir: uma lista com os objetos detetados, uma caixa delimitadora de cada um e uma pontuação que indica a confiança de cada uma das classes detetadas. Num recurso, em que o processamento de imagens é realizado em tempo real, o tempo de resposta deve ser estudado com muita atenção. Nos casos em que o utilizador aponta a câmara para mesma cena durante algum tempo, se a funcionalidade de detetar objetos foi programada de forma a informar o objeto identificado sempre que for processada uma imagem, acontece que o mesmo pode ser informado repetidamente. Para resolver este problema, os autores desta proposta optaram por definir um tempo curto em que aplicação deixa de informa as classes detetadas. Ou seja, sempre que um objeto é detetado, ele é informado uma única vez ao utilizador e só quando passarem cinco segundos é que a aplicação volta novamente a informar.

Concluído esta proposta, os autores anunciam que o seu sistema apresenta uma precisão geral de aproximadamente 90%. Divulgam ainda que, a sua aplicação é de fácil utilização e que apenas os objetos com um limite de confiança superior ao estabelecido serão informados. Futuramente, procuram aumentar a eficiência do modelo, treinando um grande número

de imagens, com um maior número de etapas para obter melhores resultados.

Nos sistemas desenvolvidos em [29, 30] (2020), [31] (2019) [32] (2018), os autores projetaram soluções que partilham a mesma ideia que abordagem anterior. À exceção do projeto apresentado em [30], os restantes mencionados implementaram uma aplicação móvel onde é possível verificar que contém uma interface apropriada a pessoas invisuais. Para além disso, os botões e o texto possuem tamanhos suficientemente grandes, de modo a facilitar a navegação por parte do utilizador, o que permite classificá-la de fácil utilização. No que diz respeito ao conjunto de dados utilizado para treinar o modelo, os autores de [31] e [32], em vez do COCO, utilizaram o Canadian Institute for Advanced Research (CIFAR)-10 [33] e CraftAR [34], respetivamente. O primeiro contém imagens de dez tipos de objetos distintos, o que significa que o modelo consegue apenas detetar dez classes diferentes de objetos. Isso é relativamente pouco em comparação com a quantidade de objetos que a proposta anterior consegue reconhecer. Noutro ponto de vista, o Craft AR Dataset é uma biblioteca que aproveita a vantagem da Cloud Image Recognition Software Development kit (SDK) para obter grandes conjuntos de dados de imagens. Os autores que utilizaram este formato, desenvolveram o seu próprio modelo de rede neuronal, e sendo assim não é possível saber a quantidade de objetos que o sistema é capaz de detetar, uma vez que tal não foi referido na sua publicação.

No trabalho apresentado em [35] (2020), Akshay Chuttergoon e Leckiaj Nagowah construíram uma aplicação móvel, designada por Eye Guide, com o objetivo principal de apoiar pessoas invisuais nas suas tarefas diárias. O sistema desenvolvido dispõe de três funcionalidades: reconhecimento de objetos, de texto e reconhecimento facial. A aplicação faz uso da funcionalidade *talkback* para facilitar a interação entre o utilizador e o *software*. *Talkback* é um recurso exclusivo do sistema operativo *Android*, que serve para dar suporte de voz a pessoas com baixa ou perda total de visão [36]. Permite informar ao utilizador, através de voz sintetizada, tudo que está a acontecer no ecrã do dispositivo móvel, como por exemplo, indicando-lhe as alternativas disponíveis para realizar uma determinada ação. Para reconhecer os objetos, os autores da aplicação Eye Guide, utilizaram modelos pré-treinados da plataforma FritzAI. Essa plataforma recorre à Aprendizagem Automática, em inglês Machine Learning, de modo a disponibilizar soluções para quem procura desenvolver aplicações móveis que pretendam detetar objetos [37]. Tal como acontece nas propostas anteriores, o seu modelo de reconhecimento de objetos foi treinado utilizando a API TensorFlow.

No final, Akshay Chuttergoon e Leckiaj Nagowah, testaram a sua aplicação durante três meses na organização Lois Lagesse Trust Fund. O objetivo desta última é fornecer educação e emprego a pessoas invisuais, oferecendo também recursos que as ajudem nas suas tarefas diárias [38]. Relativamente à funcionalidade de reconhecer objetos, a mesma foi aprovada por 90% dos participantes. A opinião sobre este recurso ficou divida entre os vários participantes, alguns descrevem-no como uma funcionalidade interessante a longo prazo. Por outro lado, os restantes concordaram que essa abordagem pode ser perigosa, nos casos em que os objetos pontiagudos são encontrados nas proximidades [23].

Para além da funcionalidade de detetar e reconhecer objetos, alguns autores incluíram, nos seus projetos, um método para calcular a distância e a direção dos objetos ao utilizador, a

fim de melhorar a experiência de reconhecimento. Algumas propostas que adotam este novo recurso podem ser observadas em [39, 40, 41](2020). Para calcular a distância, a abordagem mais comum foi utilizar a lei da similaridade do triângulo [42]. Para tal, os autores, referem que a câmara do dispositivo móvel deve estar calibrada e o modulo de calibração, deve ser executado sempre que aplicação for iniciada pela primeira vez. O objetivo da calibração é determinar a distância focal da lente da câmara. Segundo [40], a similaridade do triângulo prova que tanto o triângulo em tempo real, como o capturado entre o utilizador e o objeto são semelhantes. Sendo assim, uma vez que a distância é diretamente proporcional à largura do pixel do objeto, é possível deduzir uma fórmula que permita calcular a distância do objeto ao utilizador. A largura do pixel é, normalmente, determinada através das coordenadas das caixas delimitadores de cada objeto (uma caixa retangular desenhada em torno do objeto detetado na imagem). No que diz respeito à direção dos objetos, em [39], os autores, utilizaram uma técnica que consiste em dividir a imagem em três partes ao longo do seu comprimento, numa proporção 3:4:3. Logo, os primeiros 30% da imagem caracterizam o lado esquerdo, os seguintes 40% referem-se ao centro (frente do utilizador), e por último, os restantes 30% dizem respeito ao lado direito. Deste modo, por exemplo, se o centro do objeto estiver na primeira parte da imagem, então a sua direção será à esquerda.

Em [43] (2020), Sunit Vaidya, Naisha Shah, Niti Shah e Prof. Radha Shankarmani projetaram um sistema de deteção de objetos em tempo real, que em vez usar o modelo COCO SSD MobileNet v1, utilizam os modelos Tiny YOLOv3 e YOLOv3. Os autores desta proposta, também fizeram uma análise comparativa entre diferentes algoritmos. No entanto, ao contrário do que acontece em [23], no seu estudo, não incluíram o algoritmo SSD. Entre os algoritmos R-CNN, Fast R-CNN, Faster R-CNN e YOLO, decidiram escolher o YOLO para seu trabalho. Eles explicam que o motivo da sua preferência foi o facto de este funcionar de maneira diferente, visto que em uma única instância, pega na imagem inteira e processa-a. Para além disso, dentro dos quatro algoritmos, é o que apresenta maior velocidade de processamento, podendo processar quarenta e cinco *frames* por segundo. Porém, eles não incluíram o SSD, mas este consegue processar cinquenta e nove *frames* por segundo, mais catorze que o YOLO [23]. O trabalho desenvolvido por estes autores é muito semelhante ao que já se viu até agora. A sua aplicação móvel é iniciada e, automaticamente, o mesmo acontece à câmara do dispositivo móvel. De seguida, ao pressionar um botão com a designação "Iniciar/Parar Yolo", a visualização, em tempo real, é processada. Por fim, os objetos são detetados e informados ao utilizador por meio de saída de voz. Com base nos três modelos YOLO disponíveis YOLOv1, YOLOv2 e YOLOv3, os autores, mencionam que o último é o mais rápido e preciso, pois apresenta melhores resultados de velocidade e precisão em comparação com os outros. Desta forma, optaram por executar o Tiny YOLOv3. O conjunto de dados (*datset*), que os autores utilizaram consegue detetar, no total, oitenta classes de objetos diferentes.

Em [44] (2019), Selman Tosun e Enis Karaarslan, propõem uma aplicação móvel para reconhecimento de objetos, designada por Third Eye, que de igual forma emprega o modelo YOLO. Neste projeto, os autores, utilizaram o modelo YOLOv2 para detetar objetos. Eles explicam que, no seu estudo, como o processamento de imagem é um processo de alto desempenho, tiveram que trabalhar com um conjunto de dados que incluísse menos objetos.

Sendo assim, recorreram ao conjunto de dados Tiny YOLO trainval, que, atualmente, contém vinte classes de objetos diferentes. No entanto, como plano futuro, eles pretendem aumentar o número de classes neste conjunto, de modo que aplicação consiga reconhecer mais do que vinte objetos. Para além disso, da mesma forma que acontece em [39], Selman Tosun e Enis Karaarslan também tencionam incluir um método que permita ao sistema notificar a distância e a direção dos objetos detetados em relação ao utilizador. O código fonte do seu trabalho foi disponibilizado com General Public License (GPL) na plataforma GitHub e pode ser acedido em [45].

Uma perspetiva diferente pode ser observada em [46] (2019), onde foi projetado um sistema com a mesma finalidade, mas utilizando o algoritmo SSD. Os autores, no início do seu trabalho, referem que decidiram implementar o modelo YOLO treinado. Entretanto, com o objetivo de obter uma visão detalhada de como funcionava a deteção em diferentes modelos, implementaram também o algoritmo SSD. Verificaram que este último obteve melhores resultados de precisão e de desempenho, em comparação com o YOLO. Posto isto, escolheram o SSD como modelo principal da aplicação. A mesma não é muito diferente das apresentadas nas várias propostas mencionadas anteriormente. De igual forma, ela foi desenvolvida para detetar e reconhecer objetos próximos do utilizador e informar-lhe o nome deles através de produção artificial de fala humana, *Text-To-Speech*.

De modo a contornar o número reduzido de objetos reconhecidos pelos modelos já existentes, a construção de conjuntos de dados próprios é outra solução discutida por vários autores. É uma forma de incluir novos objetos e oferecer uma lista mais ampla e variada. Acontece que os modelos, normalmente, são pré-treinados com um conjunto de dados já definido. Muitos deles incluem objetos insuficientes e alguns podem até nem ser muito úteis para objetivo de trabalho em questão. Já que estamos a desenvolver um recurso destinado a pessoas invisuais, é importante que a aplicação consiga detetar objetos que estão frequentemente presentes no seu dia a dia, e quanto maior for a quantidade de classes diferentes, mais eficiente é o sistema. No projeto desenvolvido em [47] (2019) é possível encontrar uma proposta de trabalho que se baseia neste critério. Os autores desenvolveram uma aplicação móvel com o objetivo de identificar possíveis perigos ou obstáculos que possam aparecer no caminho do utilizador (pessoa invisual), proporcionando-lhe uma caminhada mais segura em espaços urbanos. Como o cenário é uma cidade, uma zona ao ar livre (por exemplo uma rua), os objetos implementados no sistema têm de ser determinantes. O modelo base escolhido pelos autores, foi o Inception-v3. Inception-v3, é um modelo CNN de deteção de objetos desenvolvido pela Google, capaz de detetar mil classes de objetos diferentes. Segundo os dados relatados pela própria Google, este modelo foi treinado com sucesso e atingiu uma precisão superior a 78,1% [48]. À partida, este modelo, parece ser profícuo, devido à quantidade de objetos implementados. No entanto, analisando a sua lista de objetos, é possível verificar que a maioria deles não são úteis para este objetivo. Grande parte deles correspondem a raças de animais. Posto isto, os autores decidiram preparar um conjunto de dados com imagens suas, que incluem objetos mais adequados ao seu objetivo, como escadas, trilhos, calçadas e passagem para peões. De seguida, eles fizeram uma reaprendizagem, ou seja, através dessas imagens de treino e com base no modelo Inception-v3, obtiveram um novo modelo de

identificação. Esse novo modelo, é então utilizado na sua aplicação.

A aplicação foi testada num espaço urbano durante a realização de uma caminhada. O teste foi dividido em duas experiências. Na primeira prática, treinaram um modelo, ao qual deram o nome de Modelo 1, com seiscentas e trinta e seis imagens de obstáculos como escadas e bicicletas, e trezentas e noventa e três imagens de outras situações perigosas como trilhos, calçadas e passadeiras. Tal como mencionado no parágrafo anterior, o Inception-v3 foi reaprendizado com essas imagens para ser capaz de reconhecer esses novos objetos. No final, os autores mencionam que, apesar do número de amostras dos objetos ser pequena, os resultados foram promissores. Conseguiram atingir 90% ou mais de precisão em todas as categorias de objetos desejados. No entanto, o Modelo 1 apresentou algumas falhas de classificação, como por exemplo, confundiu uma vedação com uma escada.

Na segunda e última experiência, para além das imagens reunidas pelos próprios autores, a aprendizagem conteve também imagens recolhidas da *Internet*. Porém, algumas delas, não são adequadas para a aprendizagem e, sendo assim tiveram de aplicar recortes às imagens baixadas. O seu conjunto de dados, que contém entre quatrocentas e duas mil imagens, foi dividido em dois grupos: 80% das imagens para o treino e 20% para o teste. Com este novo critério de avaliação, voltaram a analisar a precisão de classificação do Modelo 1 e, obtiveram resultados com menor precisão comparando com os obtidos na primeira experiência, onde duas categorias ficaram abaixo dos 70% de precisão. Por outro lado, no Modelo 2 (Segunda experiência), os resultados já foram um pouco melhores, onde todas as categorias de objetos conseguiram chegar aos 80% ou mais de precisão. Finalizando esta proposta, os criadores da aplicação concluíram que o Modelo 2, que incluía imagens tiradas da *Internet*, tem maior capacidade de classificação em comparação ao Modelo 1. Contudo, eles revelam que a precisão do modelo pode reduzir à medida que forem adicionadas mais categorias de objetos para serem detetadas.

A construção de sistemas equipados com servidores que realizam a parte do reconhecimento objetos, é uma área que também está a ser estudada por alguns autores. É o caso de [49], onde Neel Parikh, Ishita Shah e Safvan Vahora desenvolveram uma aplicação móvel que faz uso de um *smartphone* Android e de um servidor. A ideia desta proposta é recolher imagens, em tempo real, através da câmara do dispositivo móvel, e enviá-las para o servidor que irá fazer o reconhecimento visual de objetos. Desta forma, o telemóvel não ficará subcarregado com essa tarefa. O servidor utilizado neste trabalho, foi equipado com Graphics Processing Unit (GPU)(NVIDIA TESLA K80), e pode executar Compute Unified Device Architecture (CUDA) para realizar computação profunda. Assim que uma *frame* for recebida pelo servidor, ocorrerá a segmentação de imagem e todos os objetos serão detetados. De seguida, os mesmos serão enviados para um modelo de rede neuronal pré-treinado, que é capaz de reconhecer onze classes de objetos diferentes. Os resultados do reconhecimento, são enviados do servidor para aplicação e informados ao utilizador através de áudio.

Para utilizar esta aplicação, o *smartphone* do utilizador precisa de estar conectado a uma rede *Wi-fi* para poder enviar e receber informações do servidor, o que é um problema no caso de instabilidade da rede. Para além disso, apresentação dos resultados pode demorar algum tempo se a conexão estiver muito lenta, ou até mesmo não serem mostrados, se houver

uma quebra do serviço de *Internet*.

Noutro ponto de vista, em [40] (2020), os autores propuseram uma aplicação móvel que utiliza um servidor em nuvem para treinar modelos personalizados com objetos dos próprios utilizadores. Como ponto de partida, o sistema desenvolvido oferece o reconhecimento de oitenta tipos de objetos diferentes, através da implementação do modelo MobileNet SSD v1 e do conjunto de dados COCO. Para além disso, a aplicação permite ainda que o utilizador acrescente novos objetos, ou seja, qualquer objeto que ele desejar reconhecer. Para isso, um assistente visual do utilizador, terá de recolher pelo menos dez imagens do objeto desejado, em vários ângulos. Outra forma é ele gravar um pequeno vídeo, de trinta segundos, com o objeto de interesse. De seguida, recorrendo à aprendizagem por transferência [50], os autores propõem a utilização de um modelo já treinado (SSD MobileNet v2) para treinar os novos dados dos objetos requeridos pelo utilizador. Segundo eles, é uma forma de acelerar o processo de treino, bem como de obter melhores resultados. Por fim, por meio da plataforma TensorFlow Lite, o modelo treinado é então convertido para o formato tflite, com o objetivo de poder ser utilizado naAPI do TensorFlow.

Apesar desta última abordagem disponibilizar um novo método para incluir novos objetos, os autores, na sua publicação mencionam que o processo de rotular imagens é feito com base no *software* LabelImg [51]. O mesmo é uma ferramenta de anotação de imagem gráfica, onde cada anotação é guardada num ficheiro Extensible Markup Language (XML) de forma manual. Isso significa que processo de anotação de imagens não é realizado no próprio telemóvel. Para além disso, o assistente do utilizador necessita de ter algum conhecimento sobre como é construído um modelo de rede neuronal artificial. Para avaliar a deteção personalizada de novos objetos, os autores criaram dados para dois objetos diferentes. Com base nos resultados obtidos no final da experiência, os objetos alcançaram uma precisão geral de 92% e 87%, respetivamente. Porém, os autores explicam que se as categorias a serem detetadas aumentarem, a precisão geral do sistema poderá diminuir.

Por último, os autores de [52] (2020), propõem uma aplicação móvel chamada ReCog com o mesmo objetivo que a anterior. Nesta aplicação o utilizador também pode treinar os seus próprios objetos, e ao contrário do que acontece na anterior, a anotação de imagens é realizada no telemóvel. É apresentada uma *interface* que começa por rotular o novo objeto, pedindo ao assistente do utilizador que indique o nome do mesmo. Após este passo, o assistente terá de recolher dez imagens do objeto desejado, de modo a submetê-las num servidor que irá treinar um modelo de rede neuronal para classificar os objetos solicitados. No final, por meio de notificações, a aplicação ReCog avisa o utilizador quando o modelo estiver concluído e pronto a ser utilizado. Na Tabela 2.1, é possível observar, com mais detalhe, todos as propostas mencionadas nesta secção. A tabela encontra-se estruturada em vários temas que correspondem aos critérios de analise utilizados neste estudo.

| | Propostas | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [23] | [29] | [30] | [31] | [32] | [35] | [39] | [40] | [41] | [43] | [44] | [46] | [47] | [49] | [52] |
| *Interface* | | | | | | | | | | | | | | | |
| Texto e botões com tamanho adequado | – | ✓ | – | ✓ | ✓ | ✓ | – | – | – | ✓ | – | – | – | – | × |
| Fácil utilização/navegação | – | ✓ | – | ✓ | ✓ | ✓ | – | – | | ✓ | ✓ | – | – | – | – |
| Acesso às funcionalidades através de comandos de voz | × | ✓ | × | × | × | ✓ | × | ✓ | – | × | × | × | × | × | × |
| Utiliza técnicas de *Text-To-Speech* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Funcionalidades oferecidas pela aplicação | | | | | | | | | | | | | | | |
| Deteção de objetos | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Distância e direção dos objetos | × | × | × | × | × | × | ✓ | ✓ | ✓ | × | × | × | × | × | × |
| Reconhecimento de texto | ✓ | × | × | ✓ | × | ✓ | × | × | × | × | × | × | × | × | × |
| Reconhecimento de rosto | × | × | × | × | × | ✓ | × | ✓ | × | × | × | × | × | × | × |
| Metodologia de Trabalho - Modelo utilizado para detetar objetos | | | | | | | | | | | | | | | |
| COCO SSD MobileNet v1 | ✓ | ✓ | ✓ | – | × | ✓ | ✓ | ✓ | × | × | × | ✓ | × | ✓ | × |
| COCO SSD MobileNet v2 | × | × | × | – | × | × | × | ✓ | ✓ | × | × | × | × | × | × |
| YOLO v2 | × | × | × | – | × | × | × | × | × | × | ✓ | × | × | × | × |
| YOLO V3 | × | × | × | – | × | × | × | × | × | ✓ | × | × | × | × | × |
| Tiny YOLO v3 | × | × | × | – | × | × | × | × | × | ✓ | × | × | × | × | × |
| Inception v3 | × | × | × | – | × | × | × | × | × | × | × | × | ✓ | ✓ | × |
| Modelo elaborado pelos autores | × | × | × | × | ✓ | × | × | ✓ | × | × | × | × | × | × | ✓ |
| Metodologia de Trabalho - Conjunto de dados | | | | | | | | | | | | | | | |
| COCO | ✓ | ✓ | ✓ | × | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | × |
| CIFAR 10 | × | × | × | ✓ | × | × | × | × | × | × | × | × | × | × | × |
| CraftAR | × | × | × | ✓ | ✓ | × | × | × | × | × | × | × | × | × | × |
| ImageNet 2012 | × | × | × | | × | × | × | × | × | × | × | × | × | ✓ | × |
| Conjunto adicional personalizado | × | × | × | × | × | × | × | ✓ | × | × | × | × | ✓ | ✓ | ✓ |
| Capaz de detetar 80 ou mais objetos | ✓ | ✓ | ✓ | × | – | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | × | ✓ |
| Plataformas Disponíveis | | | | | | | | | | | | | | | |
| Android | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| *Web* | × | × | × | × | × | × | × | × | × | ✓ | × | × | × | × | × |
| Necessita de ligação à *Internet* | × | ✓ | ✓ | × | × | × | × | – | – | × | × | × | × | ✓ | × |
| Resultados Obtidos | | | | | | | | | | | | | | | |
| Taxa de aprovação ou Precisão geral do sistema | 90% | – | – | – | 80,4% | 90% | 87% | 92% e 87% | – | 85,5% | 23,7% | – | 80% | 96,4% | – |

Tabela 2.1: Resumo das propostas de aplicações móveis apresentadas na Secção 2.2. O símbolo "✓" significa que o sistema proposto compre determinado requisito, "×" que não o satisfaz e "–-" que não existe informação suficiente no artigo.

## 2.3 Aplicações atualmente disponíveis para o público alvo

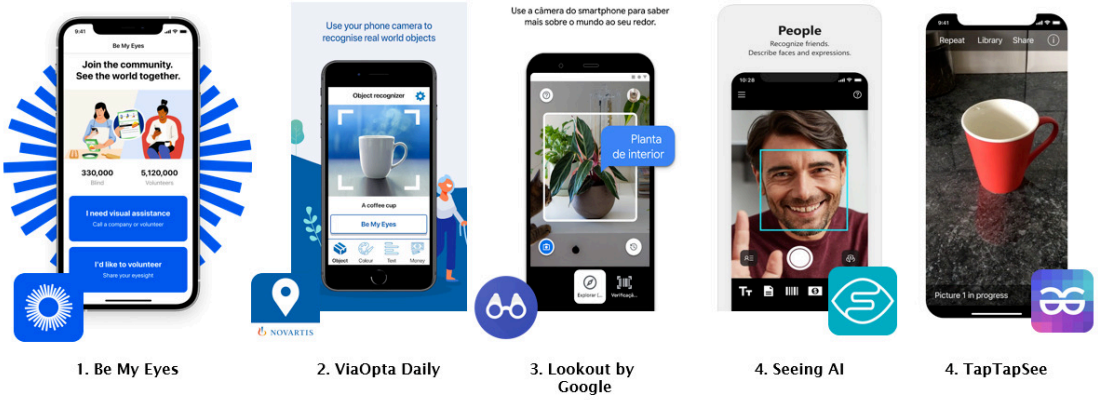



Figura 2.1: Algumas aplicações móveis atualmente disponíveis para as plataformas *Android* e iOS. Imagem adaptada de [1], [2], [3], [4] e [5] respetivamente.

No nosso telemóvel, se abrirmos o serviço de distribuição digital de aplicações móveis (Play Store, no caso do sistema operativo *android*, ou App Store, no caso do iOS) e pesquisarmos por aplicações de apoio a pessoas invisuais, facilmente encontramos uma lista variada com as mais recentes tecnologias. Tudo isso indica que estamos perante uma evolução e um desenvolvimento crescente, no diz que respeito a esta matéria. Assim sendo, o objetivo desta secção é reunir aplicações que se enquadrem nesta dissertação, de modo a compará-las e analisá-las, a fim de obter uma visão geral do que já se encontra disponível.

Comecemos pela aplicação Be My Eyes, fundada por Hans Jørgen Wiberg num evento "startup" na Dinamarca. Be My Eyes, é uma aplicação capaz de conectar pessoas invisuais a voluntários com visão e representantes da empresa [1]. O seu objetivo é oferecer assistência visual, através de uma simples chamada de vídeo. Desta forma, o utilizador poderá solicitar auxílio para resolver qualquer tipo de problema, como por exemplo, verificar datas de validade ou apoio para navegar em ambientes que lhe são desconhecidos. Durante a transmissão ao vivo, o voluntário tem como missão ajudá-lo a orientar a câmara do dispositivo móvel e avisar quando deve ligar a lanterna, de modo a responder com sucesso ao seu pedido. De forma sucinta, esta aplicação funciona do seguinte modo: o utilizador começa por escolher entre duas opções, se precisa de assistência visual ou se gostaria de ser voluntário; selecionando a primeira alternativa, ele terá de aguardar até que o pedido seja solicitado e que se encontre disponível um voluntário; após esse passo, finalmente será iniciada uma videochamada que permite que os dois possam realizar uma tarefa em conjunto. Be My Eyes foi lançada a 15 de janeiro de 2015, e conta já com mais de cinco milhões de voluntários e trezentas mil pessoas invisuais ou com visão reduzida em todo o mundo. Para além disso, encontra-se disponível em mais de cento e oitenta línguas e em cento e cinquenta países.

Como segunda perspetiva, temos a aplicação ViaOpta Daily oferecida pela empresa Novartis Pharmaceuticals Corporation. ViaOpta Daily, foi desenvolvida com funcionalidades que prometem oferecer apoio a pessoas com dificuldades visuais nas suas tarefas diárias. Permite o reconhecimento de objetos, de dinheiro, de texto e de cores. Para além disso, apresenta uma *interface* adequada a pessoas invisuais, uma vez que utiliza um guia de voz

e cada secção contém um tutorial auditivo [2]. Esta aplicação funciona como um assistente pessoal, onde o utilizador terá apenas de escolher a funcionalidade que deseja, e ela ajudar-lhe-á a resolver uma determinada tarefa especifica. Por exemplo, ao selecionar a opção de reconhecimento de objetos, ViaOpta Daily pede ao utilizador que toque no centro do ecrã para obter uma fotografia do objeto alvo, de forma a descrevê-lo em voz alta. ViaOpta Daily também dá a oportunidade de selecionar uma fotografia presente na biblioteca do utilizador para fazer o reconhecimento da mesma. Esta aplicação encontra-se, atualmente, disponível nas plataformas *Android* e iOS, em oito idiomas.

Lookout by Google é mais uma aplicação que foi desenhada com objetivo de ajudar pessoas invisuais. Com recurso à câmara do *smartphone*, ela facilita o acesso a uma informação variada sobre o ambiente que rodeia o utilizador [3]. Em termos de funcionalidades, Lookout by Google, presenteia o utilizador como cinco modos diferentes: identificação de produtos (disponível em mais de 20 países), leitura de textos e documentos em voz alta, identificação de moedas (dólares americanos) e deteção e reconhecimento de objetos que rodeiam o utilizador. Esta aplicação encontra-se apenas disponível para o sistema operativo Android, em mais de vinte idiomas.

Desenvolvida pela Microsoft, Seeing Artificial Intelligence (AI), é outra aplicação móvel que partilha o mesmo objetivo das anteriores mencionadas. Seeing AI, foi desenvolvida especialmente para o sistema operativo iOS, com o objetivo de disponibilizar recursos úteis a pessoas invisuais, transformando o mundo visual numa experiência audível [53]. Ela é capaz de reconhecer familiares e amigos do utilizador, através de reconhecimento facial, bem como as suas emoções e expressões faciais. Para além disso, por meio da AI, oferece a possibilidade de o utilizador invisual saber o que se encontra e o que está a acontecer à sua volta como, por exemplo, o nome de objetos e ações humanas. Devido ao seu enorme sucesso, Seeing AI, foi crescendo e introduzindo muitas mais funcionalidades, como é caso do reconhecimento de moedas e de texto, identificação de produtos, entre outras. Sem fins lucrativos, esta aplicação foi lançada em 2017 em setenta países. Uma das principais preocupações da Microsoft, é expandir a sua aplicação em mais idiomas, de forma a ficar mais acessível para milhões de pessoas em todo mundo. Ela começou com uma versão em inglês, e neste momento conta já com dezoito novos idiomas (francês, alemão, japonês, etc.) [4].

TapTapSee é uma aplicação móvel projetada especificamente para oferecer assistência a pessoas com perda de visão ou deficiência visual. Ela foi desenhada unicamente para reconhecer imagens, a partir da utilização da CloudSight Image Recognition API. TapTapSee utiliza a câmara e o *VoiceOver* do *smatphone* para tirar uma foto ou gravar um vídeo de dez segundos de um determinado objeto e informá-lo ao utilizador em voz alta [5]. Da mesma forma que acontece na aplicação anterior, o *flash* do telemóvel é ativado quando for detetada pouca luz, de modo a ser possível realizar o reconhecimento de objetos em ambientes de fraca iluminação. Atualmente, TapTapSee está disponível para as plataformas *Android* e iOS, em pelo menos treze idiomas diferentes.

Noutro ponto de vista temos a aplicação Supersense, que foi construída com o objetivo de introduzir a possibilidade de pessoas invisuais poderem reconhecer texto, dinheiro, produtos e objetos, através do poder da AI e da Visão Computacional [54]. A deteção e o re-

conhecimento de objetos são feitos em tempo real, ou seja, o utilizador invisual não necessita de tirar uma fotografia do objeto alvo e aguardar que a imagem seja processada, em vez disso, ele apenas terá de mover a câmara e os objetos serão identificados e informados em voz alta constantemente. Surpersense foi desenvolvida pela Mediate, uma empresa que desenvolve sistemas inovadores de visão computacional [55]. Para além da aplicação apresentar um suporte de acessibilidade *VoiceOver* e serviço *TalkBack*, todos os recursos funcionam sem a necessidade de haver uma ligação à *Internet*. Em breve, Supersense promete disponibilizar um recurso que permite ao utilizador saber a distância a que se encontra dos objetos reconhecidos pelo sistema. Apesar desta aplicação trazer funcionalidades agradáveis, uma das suas grandes desvantagens é o facto de não ser totalmente gratuita, ou seja, o acesso às funcionalidades disponibilizadas é limitado. Para obter um acesso sem restrições, o utilizador terá de aderir a um plano de assinatura pago.

Aipoly Vision também inclui uma funcionalidade que concede o reconhecimento de objetos em tempo real. Esta aplicação móvel é capaz de identificar mais de cinco mil objetos, novecentos pratos de alimentação em todo mundo (incluindo alimentos crus) e quase duas mil espécies de plantas e animais [56]. Dentro dos cinco mil objetos, Aipoly Vision aprendeu a reconhecer mil objetos universais. A deteção e identificação é realiza de forma rápida e sem necessidade de conexão à *Internet*, o que torna esta aplicação muito apelativa. Aipoly Vision consegue ainda ler texto em diferentes alfabetos e reconhecer cores. Esta aplicação foi disponibilizada de forma gratuita com suporte para vinte e seis idiomas distintos, no entanto, existe ainda uma versão paga, Aipoly Premium, onde o utilizador poderá desbloquear mais objetos identificáveis e ler texto em qualquer idioma.

Por último, terminamos esta revisão com a aplicação Envision AI, que foi vencedora do prémio Google Play Awards de 2019 [57] como melhor aplicação de experiência de acessibilidade. Desenvolvida pela Envision Technologies BV, esta aplicação capacita utilizadores invisuais ou com baixa visão a viverem uma vida mais independente, mediante a descrição audível sobre o mundo visual ao seu redor [58]. Neste momento, Envision AI, oferece as seguintes funcionalidades úteis: reconhecimento de qualquer tipo de texto em qualquer superfície, narração da cena que rodeia o utilizador (deteção e reconhecimento de objetos) e identificação de produtos por meio dos seus códigos de barras ou Quick Response (QR) Codes. Adicionalmente, concede ainda um recurso que tem como objetivo encontrar pessoas (o nome de familiares e amigos do utilizador são ditos em voz alta quando estiverem à sua volta) e objetos, através da seleção do objeto desejado numa lista definida pela aplicação. Envision AI pode ser baixada nas plataformas Android e iOS, em mais de sessenta línguas diferentes.

De forma a adquirir um visão abrangente acerca desta secção, na Tabela 2.2, é possível observar um resumo que compara as características mais revelantes das aplicações móveis analisadas em cima. É importante referir que existem muitas mais aplicações que oferecem apoio a pessoas com dificuldades visuais, porém, nesta secção, destacamos aquelas que achamos mais notáveis e que vão de encontro com o trabalho proposto nesta dissertação.

| Recursos | Aplicações Móveis | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Be My Eyes | ViaOpta Daily | Lookout by Google | Seeing AI | TapTapSee | Supersense | Aipoly Vision | Envision AI |
| Interface bem construída e fácil de utilizar | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Notificação de resultados através de *Text-To-Speech* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Necessita de uma conexão *Wi-Fi* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ |
| Aplicação móvel disponível em vários idiomas | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Assistente de Voz (chamada de vídeo) | ✓ | ✓ | × | × | × | × | × | × |
| Deteção e reconhecemento objetos | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Deteção e reconhecimento de objetos em tempo real | × | × | ✓ | × | × | ✓ | ✓ | ✓ |
| Ativação de *flash* em ambientes com fraca iluminação | ✓ | × | ✓ | ✓ | ✓ | × | ✓ | ✓ |
| Identificação de dinheiro | × | ✓ | ✓ | ✓ | × | ✓ | × | × |
| Reconhecimento texto | × | ✓ | ✓ | ✓ | × | ✓ | × | ✓ |
| Leitura de texto manuscrito | × | × | × | ✓ | × | × | × | ✓ |
| Reconhecimento facial | × | × | × | ✓ | × | × | × | ✓ |
| Descrição da cor percebida | × | ✓ | × | ✓ | × | × | ✓ | ✓ |
| Identificação produtos (Códigos de barras) | × | × | ✓ | ✓ | × | ✓ | × | × |
| Disponível no sistema operativo *Android* | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |
| Disponível no sistema operativo iOS | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Aplicação disponibilizada gratuitamente | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × |

Tabela 2.2: Tabela de resumo das diversas aplicações móveis atualmente disponíveis.

## 2.4 Análise de sistemas noutros campos

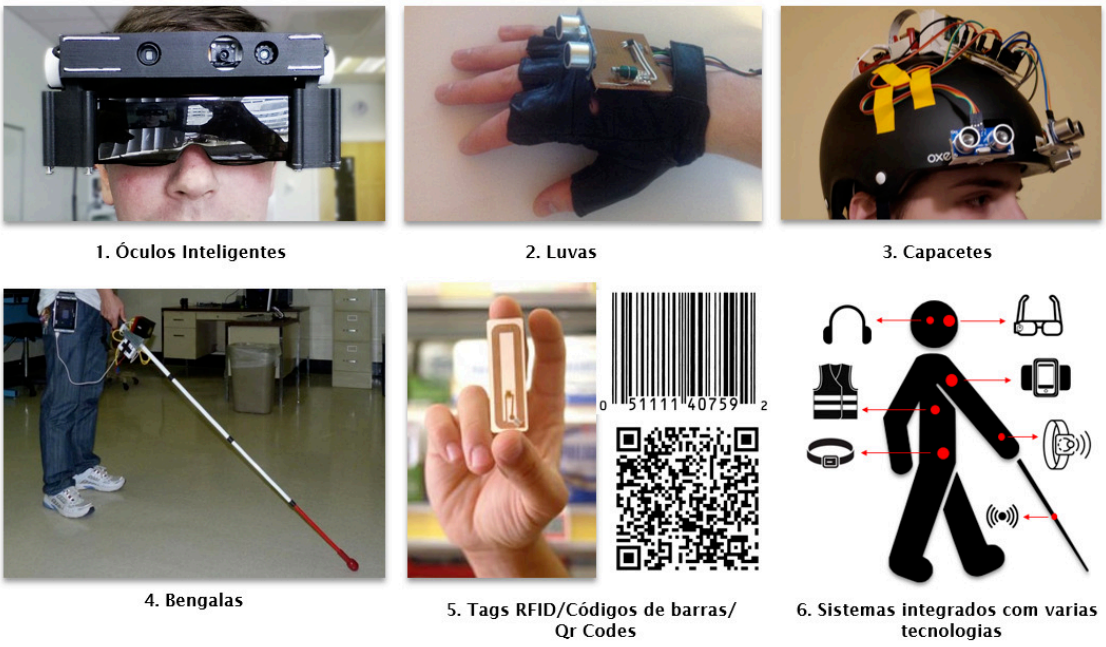



Figura 2.2: Diferentes dispositivos tecnológicos de assistência a pessoas invisuais, equipados com módulo de deteção de objetos. Imagem adaptada de [6], [7], [8], [9, 10, 11], respetivamente.

Existem muitas formas de desenvolver um sistema que seja capaz de detetar e reconhecer objetos. Até aqui, vimos várias propostas que se centram no desenvolvimento de aplicações móveis. No entanto, na literatura, é possível encontrar projetos que abordem outras metodologias, isto é, que não se baseiam unicamente em aplicações, mas sim no estudo de outros meios ou ferramentas, que de igual forma empregam o recurso de detetar objetos. Esta secção é muito fundamental para a compreensão de um estudo mais aprofundado sobre tudo aquilo que esta a ser desenvolvido no campo desta dissertação. Por isso, de seguida destacamos alguns trabalhos interessantes e diferentes daquilo que já se viu até agora.

Muitos criadores estudam a possibilidade de expandir o recurso de deteção de objetos para um simples acessório do nosso quotidiano, um par de óculos. Existem várias propostas que partilham esta realidade [59, 60, 61, 62, 63]. Este acessório, muitas vezes chamado de Óculos Inteligentes ou Vidro Inteligente, geralmente encontra-se equipado com processadores, câmaras e sensores ultrassónicos, localizados na parte frontal do dispositivo. Estes mecanismos permitem processar imagens do ambiente envolvente e detetar objetos. Os objetos, normalmente, são notificados por voz ou por meio de vibrações. Um exemplo desta nova vertente é o sistema desenvolvido em [59] (2020), onde os autores, com o apoio de um *smartphone*, desenvolveram uns óculos portáteis e fáceis de utilizar. A sua ideia é que o seu estudo sirva de auxílio a pessoas com dificuldades visuais e as ajude a navegar em ambientes internos e externos com mais segurança. O seu trabalho envolveu a implementação do modelo YOLO v3 para detetar objetos, treinado com o conjunto de dados COCO. Um sensor, montado nos óculos, é utilizado para transmitir imagens para esse modelo e opera a uma velocidade de transferência de trinta *frames* por segundo. Para treinar o modelo, os autores

utilizaram a API Google Vision e o Google Cloud. Neste trabalho, a distância do objeto em relação ao utilizador também é calculada. Ela é informada ao utilizador a partir de vibração, isto é, à medida que a distância do objeto ao utilizador diminui, a frequência de vibração aumenta e passa a ser continua. Por fim, é importante referir que o *smartphone* utilizado neste sistema, serviu para fornecer uma *interface* de voz ao utilizador.

Uma abordagem diferente pode ser observada em [7], onde os autores projetaram uma luva inteligente que promete ajudar pessoas invisuais a identificar objetos em ambientes internos, através da utilização da estrutura TensorFlow e do algoritmo de deteção SSD. O seu protótipo pode ser observado no ponto 2 da Figura 2.2. A luva funciona como um sistema independente e foi equipada com uma câmara Raspberry Pi, um módulo de reconhecimento de voz e outro de deteção de objetos, sensores ultrassônicos e motores de micro-vibração. O TensorFlow foi implementado na Raspberry Pi, com a ajuda da biblioteca Python OpenCV. Para detetar e reconhecer objetos, os autores utilizaram a API do TensorFlow e o seu modelo base, que foi treinado no conjunto de dados COCO. O protótipo desenvolvido contém também um microfone integrado no modelo de reconhecimento de voz, que permite ao utilizador, por meio comandos definidos, indicar a tarefa que pretende realizar. Por exemplo, ele pode falar para luva e informá-la que pretende encontrar um determinado objeto. Se o mesmo estiver na lista dos noventa objetos que o sistema consegue identificar, então ela guiá-lo-á até esse objeto desejado.

No trabalho apresentado em [64], Salman Shaikh, Saiyed Farhan Jafar, Karan Sosa e Pratap Nair propuseram um dispositivo diferente para apoiar pessoas invisuais, chamado Capacete Inteligente. O objetivo do seu trabalho é ajudar o utilizador a identificar objetos em tempo real. Para isso, o capacete encontra-se equipado com uma câmara Internet Protocol (IP) (Raspberry PI) para detetar objetos, sensores ultrassônicos para calcular a distância dos mesmos em relação ao utilizador, e por fim, um módulo de Global Positioning System (GPS) e SOS. O sistema é capaz de reconhecer objetos recorrendo a um algoritmo de processamento de imagens, e consegue detetar até quinze objetos por *frame*. O modulo de GPS serve para ajudar o utilizador a navegar até casa e o SOS para enviar uma mensagem a um contacto registado no sistema, em caso de emergência. No final, os autores, mencionaram algumas limitações no seu trabalho. Uma delas é que o sistema apresenta uma latência de 2 a 4 segundos, devido às limitações de processamento do Raspberry Pi. E por último, o módulo GPS Neo-6Mt apresenta uma precisão de 2,5-10m dependendo do ambiente.

Etiquetas Radio Frequency IDentification (RFID), códigos de barras e QR Codes são outros métodos escolhidos por alguns autores, em alternativa aos algoritmos ou modelos para detetar de objetos. RFID, é uma tecnologia que pode ser utilizado para identificar objetos através de sinais de radio (ondas de radiofrequência). Para isso, são usadas *tags* RFID, etiquetas constituídas, normalmente, por uma antena e um microchip. Elas podem conter informação armazenada (mensagens legíveis), com por exemplo, neste caso, o nome do objeto alvo. Sendo assim, quando uma etiqueta estiver dentro do alcance de um leitor RFID, é transmitido um sinal eletromagnético para sua antena. Isso vai fazer com que o seu circuito seja alimentado, permitindo enviar um outro sinal, chamado sinal de retorno, ao leitor [65]. Esse sinal contém a informação que estava gravada na *tag* RFID (nome do objeto), e desta

forma é possível então saber qual o objeto que rodeia o utilizador. Um exemplo da utilização de RFID para detetar objetos pode ser observado em [66] (2019), onde os autores propuseram um sistema, denominado por NavCane, capaz de ajudar pessoas invisuais a reconhecer objetos num ambiente interno. NavCane é dispositivo eletrónico, semelhante a uma bengala, equipado com cinco sensores ultrassónicos, dos quais um diz respeito a um leitor RFID. No seu estudo atribuíram etiquetas RFID a objetos, normalmente presentes em ambientes internos, como sofás, mesas, cadeiras, etc. O leitor ao encontrar essas etiquetas lê a informação do rotulo do objeto e informa-o ao utilizador por meio de áudio. Mais propostas que adotam esta tecnologia, podem ser observadas em [67, 68, 69].

No que diz respeito aos códigos de barras e QR Codes, eles funcionam de forma muito semelhante às *tags* RFID. Porém, a grande desvantagem dos leitores desse tipo de códigos é que só são capazes de ler etiquetas, com representações gráficas, a uma curta distância. Isso torna os leitores RFID muito mais vantajosos e viáveis, uma vez que eles conseguem realizar leituras de até dez metros de distância e de centenas de *tags* por segundo [70]. Portanto, utilizar sistemas equipados com leitores de códigos de barras ou de QR Codes para identificar objetos (como sofás, mesas, cadeiras, etc), não se torna muito útil e eficaz. Posto isto, eles passaram a ser utilizados para identificar outros tipos de objetos, como por exemplo, produtos alimentares, medicamentos, etc. Para uma pessoa invisual, saber distinguir embalagens de produtos diferentes é uma tarefa bastante importante na sua rotina diária. Isso é possível porque cada produto tem o seu próprio código de barras, o que permite diferenciá-los. Geralmente por meio de um telemóvel ou de um leitor adequado e equipado num sistema eletrónico, é possível saber o nome de um determinado produto e informá-lo ao utilizador através de áudio. Um exemplo desta abordagem pode ser observado em [71] (2016), onde os autores desenvolveram um sistema de identificação de produtos, propondo uma nova técnica para detetar códigos de barras European Article Number (EAN).

Por último e não menos importante, neste estudo bibliográfico, foi possível encontrar alguns projetos que reúnem vários componentes tecnológicos num sistema complexo. Isto é, em vez de estarem todos integramos num só dispositivo, como é caso dos óculos ou dos capacetes inteligentes, eles encontram-se divididos em vários módulos (observar ponto 6 da Figura 2.2). Quando falamos em sistema complexo, referimo-nos a um sistema que resulta da junção de várias ideias ou partes, e que de alguma forma estão interligadas entre si. Por exemplo, em [72] (2020) os autores apresentaram um novo dispositivo de Assistive Technology (AT) com estruturas de aprendizagem profunda em hardware vestível. O mesmo consiste na junção de três elementos: uma câmara de baixo custo (sensor Complementary Metal-Oxide-Semiconductor (CMOS) RGB), uma unidade de computação System On Module (SOM) e um sensor ultrassônico. A câmara foi implementa nuns óculos e permite recolher imagens em tempo real. O componente SOM e o sensor ultrassônico foram inseridos num cinto, sendo que o primeiro é utilizado para processar as imagens enviadas pela câmara e o segundo para fornecer a distância do objeto ao utilizador.

Na Tabela 2.3 é apresentado um resumo de todas as propostas mencionadas nesta secção. Alguns dados, considerados como importantes, são destacados, tais como: resultados obtidos, metodologia de trabalho aplicada, conjunto de dados e ferramentas utilizadas.

| Ano | Proposta | Resultados | Metodologia | Conjunto de dados | Ferramentas Utilizadas |
|---|---|---|---|---|---|
| 2020 | *"Design and Implementation of Voice Assisted Smart Glasses for Visually Impaired People Using Google Vision" [59]* | Construção de um sistema: Óculos Inteligentes, com processadores, sensores ultrassônicos e um GPS para fornecer informações de localização. | YOLO v3 | COCO (81 Objetos) | API Google Vision; API Google Text-to-speech; Google Cloud; Android Studio; |
| 2020 | *"Smart Glove for Blind using Tensorflow" [7]* | Luva Inteligente que utiliza Tensorflow e um algoritmo de deteção de objetos em ambiente internos. | COCO SSD MobileNet v1 | COCO (90 Objetos) | Tensorflow; Raspberry pi 3B; Sensor Ultrassônico; Micro Vibrating; |
| 2020 | *"Solar-Powered Deep Learning-Based Recognition System of Daily Used Objects and Human Faces for Assistance of the Visually Impaired" [72]* | Dispositivo de AT vestível movido a energia solar para detetar e reconhecer objetos; | Algoritmo R-CNN | COCO + Conjunto personalizado (103 Objetos no total) | Sensor CMOS RGB; Sensor Ultrassônico; SOM; Paineis solares; |
| 2019 | *"Smart Helmet for Visually Impaired" [64]* | Implementação de um protótipo, chamado Capacete Inteligente, com o objetivo de ajudar o utilizador a identificar objetos em tempo real; | Modelo não mencionado | *Dataset* não informado (15 objetos por *frame*) | Raspberry pi 3B; microcontrolador AT89C52; Google Cloud Text-to-Speech; GPS Neo-6M; |
| 2019 | *"An Astute Assistive Device for Mobility and Object Recognition for Visually Impaired People" [66]* | Construção de um dispositivo semelhante a uma bengala: NavCane; NavCane é capaz de identificar objetos através de etiquetas e de um leitor RFID equipado no sitema. | *Tags* RFID | Atribuição de etiquetas a objetos (como sofás, cadeiras, mesas, etc.) (Quantidade de objetos não mencionada) | Sensores ultrassônicos; Leitor RFID; |
| 2016 | *"Low-computation egocentric barcode detector for the blind" [71]* | Proposta de um nova técnica para detetar códigos de barras com baixa computação. | Códigos de barras | ArteLab Rotated Barcode (365 cód. de barras EAN) WWU Muenster Barcode (1000 códigos de barras) | Técnica LSD; |

Tabela 2.3: Tabela de resumo dos diversos dispositivos eletrónicos apresentados na Secção 2.4.

## 2.5 Análise Crítica

O presente capítulo deu-nos a conhecer a evolução e a adaptação de diferentes perspetivas de pesquisas científicas sobre o desenvolvimento de sistemas dedicados a pessoas com dificuldades mais ou menos graves de visão. Desde a apresentação de um estudo detalhado sobre recentes propostas de aplicações móveis publicadas por vários autores, a uma análise profunda de sistemas relacionados, foi possível obter uma visão extensa e produtiva sobre tudo o que está a ser desenvolvido no âmbito desta dissertação. Após este grande passo, é importante agora avaliar de forma crítica e rigorosa todo estudo realizado, a fim de obter uma opinião bem fundamentada sobre o conhecimento adquirido, e identificar novos desafios e oportunidades de trabalho.

Em primeiro lugar, consideramos que a solução mais eficaz para implementação de um sistema que permita a deteção e reconhecimento de objetos é a construção de uma aplicação móvel. Anteriormente, na Secção 2.4, investigamos sistemas que utilizam metodologias diferentes, no entanto, o utilizador teria de investir para os requisitar. Tal como foi mencionando no início desta dissertação, atualmente, a maior parte da população possui um telemóvel, e ele disponibiliza muitos recursos úteis que ajudam a concretizar o objetivo proposto. Sendo assim, acreditamos que não há muitas vantagens em o utilizador adquirir diversos equipamentos (óculos inteligentes, bengalas, etc.), uma vez que a tecnologia móvel oferece, de igual forma, todas as ferramentas necessárias. Para além disso, um telemóvel é um aparelho leve, portátil e não causa fadiga ao utilizador. Por exemplo, o sistema [72], apresentado na Secção 2.4, exige que o utilizador esteja constantemente equipado com muitos artefactos, o que pode originar cansaço e dificuldade em continuar com a realização da tarefa.

No que diz respeito à criação de um recurso de deteção e reconhecimento de objetos, na Secção 2.2, vimos que vários autores propõem a implementação de um modelo de rede neuronal artificial. Na Tabela 2.1, facilmente observamos que a maior parte dos modelos requeridos por eles, são capazes de detetar mais ou menos oitenta objetos diferentes. Foi ainda encontrada uma exceção, o trabalho desenvolvido em [47] que utilizou o Inception v3, que é um modelo que consegue reconhecer mil classes diferentes de objetos. Porém, muitos deles, não são úteis para objetivo em questão. É importante reforçar a ideia de que, quanto maior for a quantidade de objetos reconhecidos pelo modelo, mais eficiente e útil se torna a aplicação para o utilizador invisual. Por outro lado, também não é viável disponibilizar um sistema que não tenha aprendido a reconhecer objetos que estejam frequentemente presentes no dia a dia do utilizador. Apesar dos oitenta objetos ser um número considerado suficiente para iniciar um recurso de deteção de objetos, o mesmo ainda está muito longe de atingir o objetivo pretendido. A adição de um conjunto personalizado com objetos novos e treinados pelos autores, foi uma outra solução encontrada no estudo bibliográfico realizado. Efetuar a reaprendizagem de um modelo já existente acaba por ser uma abordagem melhor, quando comparada com o treino de uma rede neuronal a partir do zero, isto porque é possível aproveitar o conhecimento que a rede adquiriu na primeira vez que foi treinada. Noutro ponto de vista, no final da revisão, analisamos o trabalho que foi proposto em [52], onde foi desenvolvida uma solução que se acredita ser o melhor caminho a seguir. A utilização de um servidor em nuvem para treinar um modelo personalizado com objetos do utilizador pode

ser um bom ponto de partida.

Para terminar esta análise critica, é revelante refletir um pouco sobre as diversas aplicações moveis atualmente disponíveis, que foram divulgadas na Secção 2.3. Como pudemos observar na mesma, é possível confirmar o desenvolvimento crescente no que diz respeito à matéria abordada nesta dissertação. Apesar disso, ainda que sejam muitas as vantagens oferecidas por elas, durante o estudo das mesmas encontrámos alguns pontos que achamos menos apelativos. Be My Eyes é muito útil, uma vez que com uma simples chamada de vídeo é possível solucionar qualquer problema, no entanto, o utilizador depende da disponibilidade e da boa-fé de um voluntario que não conhece. Sobre outra perspetiva, ao experimentarmos as aplicações mencionadas, verificamos que existem algumas situações em que a classificação de objetos não foi realizada de forma correta, o que levanta questões de precisão.

## 2.6 Desafios em aberto

Com base na análise critica realizada na secção anterior, encontrarmos novos desafios de investigação para a realização do projeto que pretendemos desenvolver nesta dissertação. É de notar que os mesmos, permitem a construção de um sistema que consideramos mais robusto e com mais oportunidades em relação aos já existentes.

A quantidade de objetos reconhecidos pela aplicação e a precisão de classificação, podem ser melhoradas através da introdução de um mecanismo que permita ao utilizador reconhecer os seus próprios pertences. Desta forma a aplicação passaria a identificar qualquer objeto que o utilizador desejasse e na quantidade que ele quisesse. Uma vez que o treino de um modelo de rede neuronal pode ser feito por meio de aprendizagem supervisionada, o objeto observado seria o mesmo que foi utilizado para treinar a rede e, sendo assim, a precisão de classificação aumentaria significativamente. Ora, posto isto, facilmente obteríamos uma aplicação muito mais personalizável e mais poderosa no que diz respeito à precisão no reconhecimento de objetos. Para além disso, a aplicação não estaria tão restrita aos modelos já treinados e disponibilizados, onde a quantidade e qualidade de objetos é insuficiente.

Um outro novo desafio, é tirar partido das imagens recolhidas dos objetos dos vários utilizadores, de modo a reuni-las e criar um modelo mais amplo e evolutivo, isto é, que seja capaz de detetar e reconhecer não só objetos do próprio utilizador, como também de outros utilizadores. Esta nova abordagem promete trazer vantagens a qualquer um deles, dado que um objeto de um determinado utilizador pode ser útil para outro. Em temos de implementação, as imagens dos objetos dos utilizadores seriam enviadas para um servidor em nuvem, onde ocorrerá o treino ou reaprendizagem de modelo de rede neuronal. De seguida, o mesmo poderia ser baixado por qualquer utilizador, a partir da aplicação.

## 2.7 Considerações finais

Em conclusão, este capítulo serviu de base para obter um conhecimento científico alargado sobre tudo que está a ser implementado no âmbito desta dissertação. Através do mesmo foi possível fazer um levantamento sobre estado atual em que se encontra o desenvolvimento

de sistemas destinados a pessoas com dificuldades visuais. O estudo bibliográfico realizado ajudou a compreender as linhas de pesquisa existentes, o que já foi ou está a ser desenvolvido nesta área e procurar possíveis falhas que ainda precisem de ser trabalhadas. É de notar que o Estado de Arte foi um dos capítulos mais importantes no processo de desenvolvimento deste trabalho académico, uma vez que permitiu fazer referência a tudo que atualmente existe acerca deste tema, evitando desperdícios de tempo e contribuindo para o aperfeiçoamento de novas ideias de progresso.

# Capítulo 3

# Abordagem conexionista aplicada ao reconhecimento de objetos

## 3.1 Introdução

Detetar e reconhecer objetos em imagens recolhidas por um *smartphone*, em tempo real, é uma tarefa bastante desafiadora em áreas como a Visão Computacional e Processamento de Imagem. O objetivo do presente capítulo é dar a conhecer todos os conceitos e técnicas que estão por detrás deste útil recurso. Será feita uma descrição detalhada, de como funciona todo o processo de reconhecimento de imagens e quais são as bases necessárias para um sistema ser capaz de localizar objetos na cena que rodeia o utilizador.

Ao longo desta dissertação, foi mencionada várias vezes a ideia de que treinar um modelo de rede neuronal artificial para interpretar o mundo visual, é chave de sucesso que permite construir um sistema produtivo na identificação e classificação rápida de objetos. Sendo assim, este capítulo, começa por apresentar o conceito de rede neuronal artificial, de modo a compreender a sua essência e as suas funções. De seguida, falar-se-á um pouco sobre CNN, que é um tipo de rede neuronal artificial que tem vindo a ser aplicada com sucesso na análise de imagens digitais. Mais á frente, uma breve explicação sobre como é feita a interpretação do conhecimento através de uma rede neuronal e a classificação de imagens. Por fim, será divulgado um estudo comparativo de diferentes algoritmos de deteção de objetos.

## 3.2 Rede Neuronal Artificial

Em tempos antigos, muitos investigadores sentiam a necessidade de criar sistemas computacionais que os ajudassem a resolver determinadas tarefas complexas e que exigiam um comportamento inteligente. Muitos desses problemas representavam várias situações da vida real e que facilmente eram resolvidos pelo cérebro humano. Assim sendo, os investigadores inspiraram-se no funcionamento aprimorado do nosso cérebro, e desenvolveram, com sucesso, a chamada Rede Neuronal Artificial [73]. A mesma é vista como um modelo matemático que simula o comportamento de seres inteligentes e que é capaz de adquirir conhecimento à medida que vai apreendendo, ganhando experiência.

O estudo das redes neuronais artificiais ofereceu suporte a muitas áreas atualmente conhecidas, como Ciências da Computação, Medicina (por exemplo, na realização de diagnósticos médicos), Engenharia, entre outras. No campo da Engenharia, a utilização deste tipo de redes, apoiou a criação de estratégias que concedessem, por exemplo, a interpretação de imagens e vídeos digitais, reconhecimento de voz, texto e rosto, dando origem a áreas de investigação muito profícuas da AI e Visão Computacional [74].

### 3.2.1 O fascinante cérebro humano - Conceitos Introdutórios

Considerado o centro da inteligência, o cérebro é um dos órgãos do sistema nervoso mais importantes para o ser humano, sendo responsável por coordenar a maior parte das atividades do nosso corpo. Em apenas alguns milissegundos, ele é capaz de processar diferentes tipos de informação, proveniente de várias fontes. Além de tudo, devido às suas qualidades extraordinárias, está apto a angariar conhecimento à medida que aprende, raciocina e resolve problemas. Esta vertente é a prova da existência de inteligência humana. Atualmente não existe uma definição consensual de inteligência, porém muitas delas tendem a convergir para a mesma ideia [75]. Em [76], Jensen segue a observação de Carl Bereiter, que define inteligência como: "o que se usa quando não se sabe o que fazer". A mesma traduz-se na capacidade de o ser humano resolver um determinado problema quando este não se encontra preparado para uma dada situação específica. Em 1987, Snyderman e Rothman declaravam o raciocínio, a resolução de problemas e a aprendizagem como fatores determinantes na criação de inteligência [77].

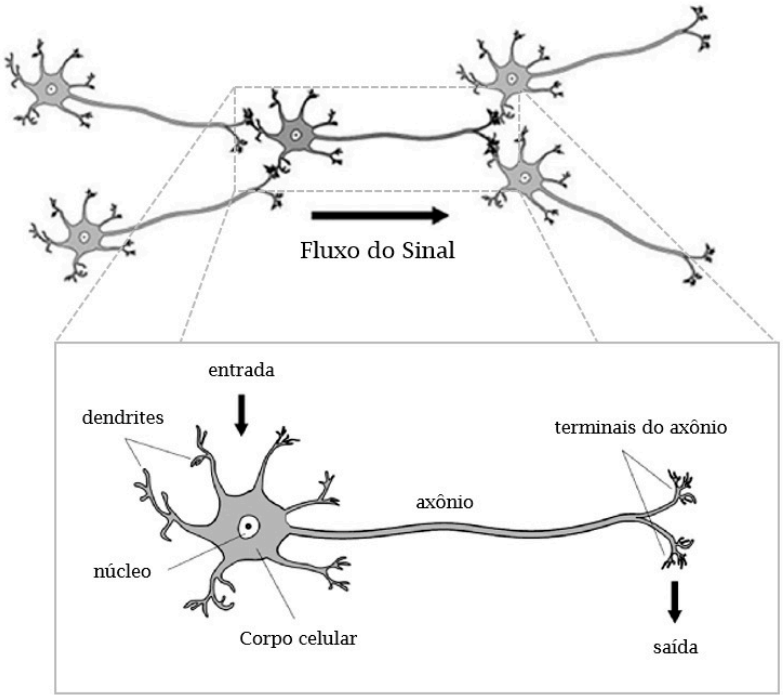



Figura 3.1: Figura ilustrativa de um neurónio biológico. Imagem adaptada de [12].

O nosso cérebro é um órgão complexo, e isso torna-o bastante exigente e competente na hora de administrar as suas diversas funções e elementos. Sendo ele o centro do sistema nervoso, uma das suas principais funções é processar informação, obtida pelos órgãos sensoriais. No sistema nervoso, os neurónios são considerados a unidade base do cérebro humano. Na Figura 3.1, é possível observar uma representação ilustrativa de um neurónio.

Um neurónio é uma célula biológica especializada em processar informações. Ele está dividido em três componentes principais: as dendrites, que são prolongamentos citoplasmáticos que permitam a receção de estímulos nervosos, a soma ou corpo celular, que é o centro metabólico do neurónio, e o axônio, que realiza a transmissão das informações geradas para

outros neurónios [78]. O corpo e as dendrites formam a superfície de entrada do neurónio, enquanto o axónio corresponde à saída do fluxo de informação (impulsos elétricos), sendo esta propagada sempre no mesmo sentido. No final de cada axónio estão presentes as sinapses. As mesmas, são vistas com o ponto de contacto entre a extremidade de um determinado axónio de um neurónio e a dendrite de outro. Elas estabelecem um método de comunicação entre os vários neurónios presentes no cérebro humano. É de notar que um neurónio não vive de forma isolada no nosso cérebro, antes pelo contrário, ele encontra-se ligado a outros neurónios (observar novamente a Figura 3.1). Quando um impulso nervoso atinge as terminações sinápticas, são libertadas substâncias químicas denominadas por neurotransmissores. Essas substâncias permitem controlar os impulsos elétricos, por exemplo, se houver uma diminuição na libertação de neurotransmissores, ocorrerá uma inibição sináptica, ou seja, um bloqueio na emissão de estímulos nervosos. Sendo assim, a eficácia da sinapse pode ser ajustada pelos sinais que passam por ela, de modo a aprender. [79]

### 3.2.2 Definição de neurónio artificial

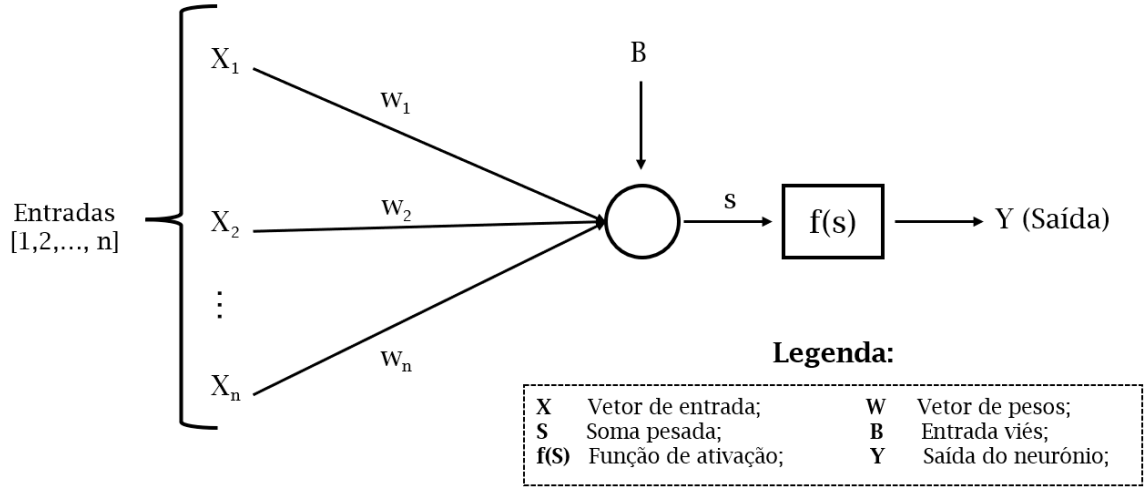



Figura 3.2: Modelo do neurónio artificial proposto por McCulloch e Pitts em 1943.

O primeiro modelo de um neurónio artificial foi proposto pelo neurofisiologista Warren McCulloch e pelo matemático Walter Pitts em 1943. O mesmo foi idealizado com base na estrutura e no funcionamento do neurónio real. McCulloch e Pitts, concentraram-se nos processos biológicos do nosso cérebro e desenvolveram um modelo que acaba por ser uma representação sobre o que na altura se sabia acerca do neurónio biológico. Na Figura 3.2, encontra-se esquematizado o seu modelo. Ao observá-lo e compará-lo com a Figura 3.1, que retrata os vários componentes de um neurónio biológico, facilmente se chega a conclusão que existem muitos pontos em comum.

No caso do neurónio artificial, existem muitas semelhanças com o neurónio biológico, porque a sua estrutura e funcionalidades são derivadas da observação deste último. Enquanto num neurónio biológico, a informação entra nele por meio das dendrites, de seguida é processada no corpo celular e depois novamente transmitida através do axónio, num neurónio artificial, ele recebe a informação a partir de entradas (*Inputs*), aplica um função e

passa o resultado a outro neurónio e por aí adiante [80]. Portanto, um neurónio artificial não é nada mais que uma função $g : \mathbb{R}^n \rightarrow \mathbb{R}$ que a cada vetor $X$ de entrada, de dimensão n, faz corresponder um valor real $y$ (observar Figura 3.2). Esta função $g(.)$ depende ainda de um vetor de pesos $[w_0 w_1 ... w_n]$ e da função de ativação escolhida $f(.)$. Num neurónio artificial, os pesos são utilizados para guardar a informação ou conhecimento da rede. Esses fatores multiplicativos pelas entradas permitem com que a rede neuronal aprenda, dado que a aprendizagem é ela conseguir determinar os pesos que façam com que resolva o problema em questão. Aliás, aprender é também saber ajustar os seus valores, de forma a minimizar os erros que possam existir. Posto isto, os valores presentes nas entradas do neurónio são então alvo de uma soma ponderada ($s$), que é obtida pela seguinte expressão matemática:

$$s = \sum_{i=0}^{n} x_i w_i. \quad (3.1)$$

Onde:

- $x_0 = 1$;
- $x_i$ é um valor de entrada associado ao índice $i$;
- $w_i$ é o peso associado à entrada de índice $i$;
- $n$ é a dimensão do vetor de entrada $X$;

O valor calculado na Equação 3.1 passa por uma função, chamada função de ativação, que irá produzir o valor de saída do neurónio $y = f(s)$. Um neurónio artificial é ainda constituído por uma entrada "especial" chamada viés (*bias* em inglês), que assume um valor fixo igual a um ($x_0 = 1$). A entrada viés tem como efeito deslocar a função de ativação. Segundo Haykin, o viés permite aumentar ou diminuir a entrada líquida dessa função, dependendo se a relação é positiva (excitatória) ou negativa (inibitória) [81, 82]. Por exemplo, se o viés for negativo, a soma pesada das entradas tem de superar o seu valor de modo que o neurónio produza um valor positivo na saída. Observando bem, esta abordagem representa o processo sináptico num neurónio biológico, onde vimos que a quantidade de neurotransmissores controla a emissão ou inibição de estímulos nervosos.

### 3.2.3 Funções de Ativação

A função de ativação pode assumir várias formas, no entanto as mais utilizadas são as seguintes [83, 84]:

- **Função Linear**, com saída em (-$\infty$,$\infty$):

$$f(s) = \beta s \quad (3.2)$$

A função linear pode ser definida pela Equação 3.2, onde a variável $\beta$ representa um número real, diferente de zero. Ela é muito utilizada na resolução de problemas de regressão, onde se pretende efetuar a aproximação de funções.

- **Função Degrau (step ou Heaviside)**, com saída em $\beta_1$,$\beta_2$:

$$f(s) = \begin{cases} \beta_2 & , s \geq 0 \\ \beta_1 & , s < 0 \end{cases} \quad (3.3)$$

A função degrau, também designada por Step ou Heaviside, foi introduzida pelo matemático e engenheiro eletricista Oliver Heaviside [85] e pode ser definida pela expressão 3.3. A mesma é uma função descontinua com valor igual $\beta_2$ quando a soma pesada ($s$) é positiva, e com valor igual $\beta_1$, quando $s$ é negativo. Normalmente $\beta_1 = 0$ e $\beta_2 = 1$, embora também se utilize $\beta_1 = -1$.

- **Função Rectified Linear Unit (ReLU)**, com saída em (0,$\infty$):

$$f(s) = \max(0, s) \quad (3.4)$$

A função ReLU é uma outra função de ativação que pode ser definida pela equação matemática 3.4. A mesma produz resultados no intervalo de zero a infinito, retornando o valor zero quando $s$ é negativo e o próprio valor $s$ quando ele é positivo. Uma das grandes vantagens desta função é a de não deixar que todos os neurónios da rede sejam ativados ao mesmo tempo, uma vez que um neurónio só é ativado quando o valor de $s$ for positivo. Isso permite acelerar o processo de treino da rede, dado que muitos deles estarão desativados. Porém nem tudo são vantagens, por exemplo, se um neurónio nunca obtiver um valor positivo à saída, ele poderá ser esquecido e não aprender nada, ou seja, não adquirir conhecimento.

- **Função Sigmoide**, com saída em (0,1):

$$f(s) = \frac{1}{1 + \exp(-s)} \quad (3.5)$$

A função sigmoide poder ser definida pela expressão matemática 3.5. Esta função contínua e diferenciável é chamada de sigmoide porque a sua curva é muito semelhante à letra "S". Qualquer saída desta função assume sempre valores positivos, no intervalo de 0 a 1. Portanto, se a variável $s$ admitir valores negativos, a saída da função $f(s)$ tende a aproximar-se do valor zero, por outro lado, se admitir valores positivos, tende a aproximar-se de um.

- **Função Tangente Hiperbólica**, com saída em (-1,1):

$$f(s) = tanh(s) = \frac{sinh(s)}{cosh(s)} \quad (3.6)$$

Tal como o próprio nome indica, a Tangente Hiperbólica, é uma função hiperbólica, caracterizada por se basear em funções trigonométricas, como o seno e cosseno. Ela é chamada de tangente porque a sua expressão é dada pela divisão entre seno hiperbólico e o cosseno hiperbólico, como é possível observar em 3.6. Em termos de gráfico, a tangente hiperbólica tem uma forma muito semelhante à função sigmoide, com a dife-

rença de ser simétrica em torno da origem. Isso faz como que seja uma função continua e diferenciável, com valores de saída situados no intervalo de -1 a 1.

- **Função *Softmax***, com saída em (0,1):

$$f(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \quad (3.7)$$

Onde $i = 1, ..., K$ e $z = (z_1, ..., z_k) \in \mathbb{R}^K$.

Normalmente aplicada na última camada de uma rede neuronal artificial, a função *softmax*, é uma função de ativação muito utilizada para calcular as probabilidades previstas de cada classe de saída da rede. Ela pode ser definida pela expressão matemática 3.7. Os valores $z_i$ correspondem a valores do vetor de entrada e $K$ ao número de classes previstas. O somatório garante que a função *softmax* retorna apenas valores no intervalo de 0 a 1, de modo a obter uma distribuição probabilística valida.

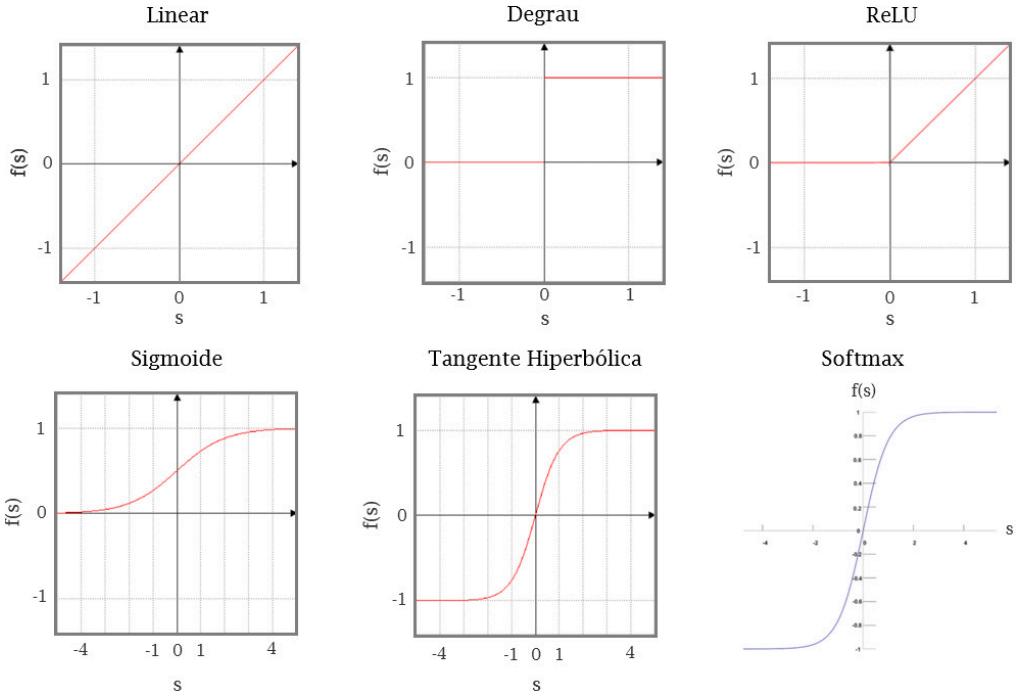



Figura 3.3: Funções de ativação. O gráfico da função Softmax foi adaptado de [13].

De modo a compreender melhor as funções de ativação que foram abordadas, a Figura 3.3 apresenta as suas representações gráficas. A escolha da função a utilizar é intrínseca ao problema a resolver, ou seja, depende da natureza do sistema que está a ser projetado. Por exemplo, na resolução de problemas de classificação onde para cada ponto de entrada se pretende fazer corresponder uma etiqueta chamada classe, á função que obteria melhores resultados seria a função Sigmoide [83]. Isto porque ela tende a impulsionar o seu resultado para as extremidades do intervalo de [0,1], e tirando partido deste comportamento é possível saber se uma dada entrada pertence ou não a uma determinada classe (classificação binária) [86].

### 3.2.4 Arquitetura de Redes Neuronais Artificiais

Um neurónio artificial, quando é ligado a outros neurónios, formando várias camadas, dá origem às tais redes neuronais artificiais. O primeiro modelo de rede neuronal artificial foi apresentado também, em 1943, por Warren McCulloch e Walter Pitts [87]. A sua estrutura é muito semelhante ao sistema nervoso biológico. Da mesma forma que o nosso cérebro é constituído aproximadamente por oitenta e seis mil milhões de neurónios [88], uma rede neuronal artificial pode ser formada por centenas ou milhares de unidades de processamento simples, interligados entre si.

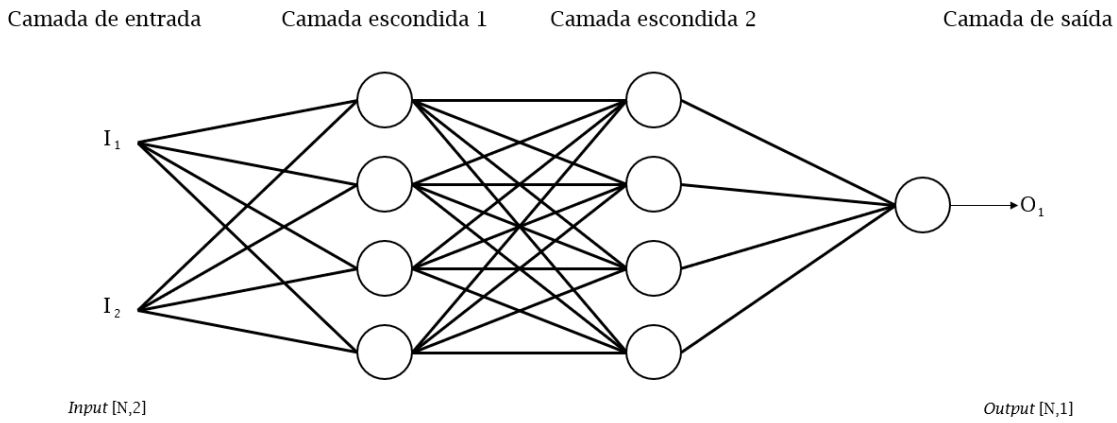



Figura 3.4: Rede Neuronal Artificial de múltiplas camadas e com uma única saída.

Na Figura 3.4, é possível observar um exemplo de um Multi-layer perceptron (MLP), que é um tipo de rede neuronal artificial em que os neurónios estão organizados em mais do que uma camada. Caso eles estivessem estruturados em apenas uma única camada, a rede neuronal passar-se-ia a chamar apenas de percetrão. Porém, um percetrão isolado contém algumas limitações. Só quando são colocados em várias camadas é que se consegue obter um sistema suficientemente capaz de resolver a maioria dos problemas. Num MLP, a informação é propagada sempre no mesmo sentido, o que faz dela uma Feed Forward Neural Network (FFNN). Para além disso, as saídas dos neurónios de uma determinada camada são as entradas dos neurónios da camada seguinte.

Qualquer MLP é formado por três tipos de camadas diferentes: uma camada de entrada (que não tem quaisquer neurónios), uma ou mais camadas escondidas ou intermédias e, por último, uma camada de saída. No exemplo apresentado, estão presentes duas camadas escondidas, cada uma com quatro neurónios artificiais. Apesar de duas camadas serem suficientes, atualmente, na área de Deep Learnig (ramo de aprendizagem automática) estudam-se redes neuronais artificiais com muitas mais camadas, designadas por redes neuronais profundas [89]. Isto porque, o fator mais importante na capacidade de aprendizagem de uma rede destas é número de neurónios na camada escondida. Quantos mais neurónios tiver a rede nas camadas escondidas, mais ligações terá, e como é nos pesos que é guardada a informação, automaticamente a rede terá a capacidade de aprender muitas mais coisas, ou seja, guardar muita mais informação. Claro está também, que quanto maior for o número de camadas intermédias, mais eficiente se torna a aprendizagem.

## 3.3 Redes Profundas - Convolutional Neural network (CNN)

Recentemente, várias evidências empíricas mostram que redes neuronais artificiais profundas, formadas por dezenas de camadas, são capazes de obter resultados mais promissores e com um melhor desempenho na realização de tarefas, como processar e analisar imagens digitais. No trabalho desenvolvido por Ronen Eldan e Ohad Shamir, em [90], ficou provado que a construção de redes com muitas camadas é uma técnica bastante vantajosa, a nível computacional. Noutra perspetiva, na investigação realizada em [91], os autores mostraram que as redes neuronais profundas são capazes de alcançar um forte desempenho de classificação, quando são otimizadas.

Atualmente, as redes neuronais profundas mais aplicadas no reconhecimento visual de objetos são as CNNs. Elas tiveram origem em trabalhos desenvolvidos pelo cientista Kunihiko Fukushima. Na década de 1970, o mesmo, desenvolveu o famoso Neocognitrão, que serviu de inspiração para a criação das redes CNNs [92]. Na altura pretendia-se simular o sistema visual humano, e o Neocognitrão veio apoiar essa ambição, uma vez que retrata uma CNN profunda capaz de reconhecer padrões visuais por meio de aprendizagem.

### 3.3.1 Noções fundamentais para a compreensão de uma CNN

O estudo e analise de uma CNN, obriga a compreensão de algumas noções base. Primeiro é importante ter conhecimento, que as redes de convolução são dividas em pelo menos dois estágios: inicialmente é feita a extração de características (*features*) e depois as mesmas são utilizadas para classificar a imagem de entrada. Na extração de características, são usados dois tipos de camadas diferentes: camadas de convolução e camadas de subamostragem. As suas definições podem ser descritas da seguinte forma [15]:

- **Camadas de Convolução** - são responsáveis por realizar o produto escalar entre duas matrizes. Elas aplicam filtros nos dados que recebem de entrada e geram os famosos mapas de recursos (*feature maps*). O *output* de uma camada deste tipo é, normalmente, uma matriz de características, que contém atributos importantes para realização da tarefa desejada;

- **Filtro** - também designado por matriz de convolução ou simplesmente *Kernel*, é um termo utilizado para descrever um conjunto de técnicas que servem para realçar determinadas particularidades na imagem de entrada. Mas à frente na Secção 3.3.3 falar-se-á um pouco mais sobre este notável recurso;

- **Camadas de Subamostragem** - têm como principal objetivo diminuir a complexidade computacional, através da redução do tamanho dos mapas de características gerados em cada camada de convolução. Quando uma característica é identificada, a camada de subamostragem torna irrelevante a zona ou região onde foi detetada. Para a finalidade em questão, não é importante saber o local exato dessa peculiaridade, mas sim se está ou não presente na imagem.

Com a ajuda dos conceitos aqui abordados, de seguida, iremos ver como é feito o processo de construção e modelagem de uma CNN, ou seja, a sua arquitetura.

### 3.3.2 Arquitetura CNN

Em 1998, Yann LeCun e outros autores, propuseram uma das primeiras arquiteturas de redes CNN desenvolvidas até hoje, chamada de LeNet5 [14]. Ela foi projetada para reconhecer caracteres manuscritos e impressos por máquina, tornando-se um modelo popular no campo da classificação de imagens. A arquitetura LeNet5 pode ser observada na Figura 3.5.

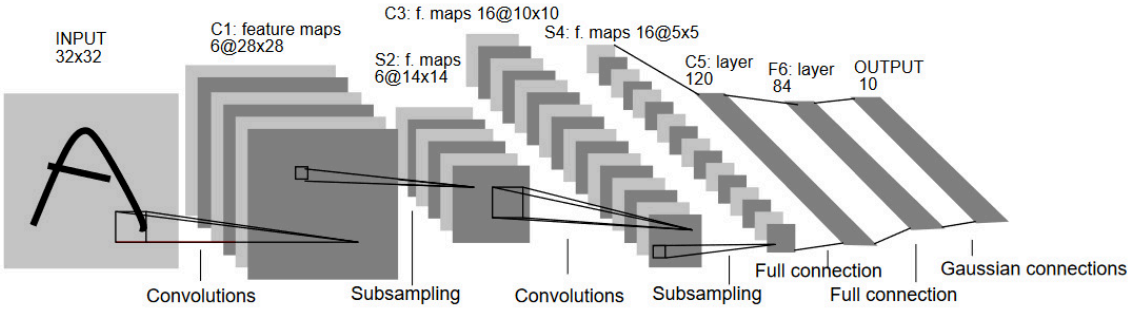



Figura 3.5: Arquitetura LeNet5. Imagem retirada de [14].

Sem contar com a camada de entrada, LeNet5 é constituída por sete camadas: três camadas de convolução, duas de subamostragem e duas totalmente conectadas. Tal como é possível observar na Figura 3.5, a entrada para o modelo apresentado corresponde a uma imagem, em tons de cinza, com uma dimensão de 32x32 *pixels*. De seguida ela passa pela primeira camada de convolução, onde são aplicados seis filtros, obtendo como resultado um mapa de recursos com o tamanho 28x28. Após essa etapa, o tamanho do mesmo é reduzido para metade (14x14) através da primeira camada de subamostragem. Depois passa novamente por mais uma camada de convolução com dezasseis filtros. Como os filtros desta segunda camada de convolução são diferentes da primeira, o mapa de recursos muda para 10x10. A sua dimensão é reduzida mais uma vez para metade (5x5) na segunda camada de subamostragem. Logo depois, entra na camada final de convolução com cento e vinte filtros. Finalmente, chega a primeira camada totalmente conectada com oitenta e quatro neurónios. Por fim, uma vez que o problema de Yann LeCun estava dividido em dez classes diferentes, a ultima camada totalmente conectada (*Output*) foi construída com dez neurónios, em que cada um retorna a probabilidade prevista para cada classe.

| Nome da Camada | Nº de Filtros | Tamanho do filtro | Tamanho do mapa de recursos | Função de ativação utilizada |
|---|---|---|---|---|
| Camada de entrada | - | - | 32 x 32 | - |
| Camada de convolução 1 | 6 | 5 x 5 | 28 x 28 | Tang. Hiperbólica |
| Camada de subamostragem 1 | - | 2 x 2 | 14 x 14 | - |
| Camada de convolução 2 | 16 | 5 x 5 | 10 x 10 | Tang. Hiperbólica |
| Camada de subamostragem 2 | - | 2 x2 | 5 x 5 | - |
| Camada de convolução 3 | 120 | 5 x 5 | 120 | Tang. Hiperbólica |
| Camada totalmente conectada 1 | - | - | 84 | Tang. Hiperbólica |
| Camada de saída t. conectada 2 | - | - | 10 | *Softmax* |

Tabela 3.1: Detalhes da arquitetura LeNet5. Dados facultados de [14].

Com o objetivo de obter uma visão mais alargada, a Tabela 3.1, apresenta alguns detalhes mais profundos sobre a arquitetura LeNet5.

### 3.3.3 Matrizes de Convolução

Na arquitetura CNN, vimos que os filtros utilizados em cada camada de convolução diferem de camada para camada. Existe uma razão lógica pela qual as matrizes de convolução não serem sempre as mesmas. A ideia é que cada camada seja capaz de aprender apenas determinadas características ou recursos, e o grau de complexidade vá aumentando, de modo a detetar características cada vez mais profundas. Por exemplo, a primeira camada de convolução (conv1) pode aprender a detetar bordas e cantos, a segunda formas geométricas, e a terceira, e por ai adiante, procuram recursos de nível superior. Na Figura 3.7 encontra-se representada uma imagem ilustrativa que reflete a complexidade de características aprendidas em varias camadas de convolução.


Figura 3.6: Visualização dos recursos CNN aprendidos nas camadas de convolusão. Imagem adaptada de [15].

As matrizes de convolução são muito importantes no processamento de imagens, pois permitem extrair características úteis para classificar as imagens de entrada. A matriz a utilizar depende da característica que se pretende estudar, por exemplo, existem filtros que permitem detetar bordas, aplicar nitidez ou desfocagem, entre outras gamas de efeitos. Na Figura 3.7 pode-se observar a execução de alguns exemplos de matrizes de convolução com dimensão 3x3. Um filtro pode ser visto como uma pequena região que se desloca ao longo da imagem, de modo a recolher traços que sejam marcantes.

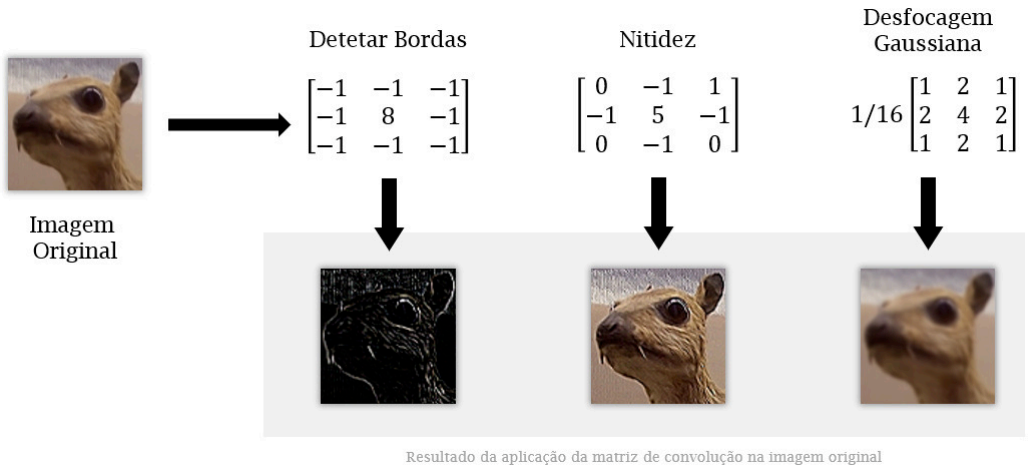



Figura 3.7: Exemplos da aplicação de matrizes de convolução. Imagem adaptada de [16].

## 3.4 Aprendizagem e reconhecimento de padrões

### 3.4.1 Interpretação do conhecimento

Uma rede neuronal artificial tem como principal função reconhecer padrões no ambiente em que está a atuar, de forma a interpretar o conhecimento que vai adquirindo. Para isso, na maior parte das vezes, ela recorre à Aprendizagem Supervisionada, onde uma entrada está associada a uma saída. Entradas semelhantes que vêm de classes com critérios semelhantes, levam a rede neuronal a classificá-las como pertencentes à mesma categoria. No reconhecimento de objetos, acontece isso mesmo, um objeto pode ser muito idêntico a uma classe de entrada e, sendo assim, ele é classificado como pertencente a essa categoria.

### 3.4.2 Classificação de imagens

A classificação de imagens é um método muito popular, que utiliza redes neuronais artificias para reconhecer elementos visuais numa imagem. A classificação é feita com base em padrões identificados e agrupados por temas. Neste caso, o papel da rede é conseguir distinguir, a partir do conjunto de treino, os dados que pertencem a cada classe, de modo a conseguir futuramente generalizar, ou seja, identificar corretamente dados que não estejam nesse conjunto.

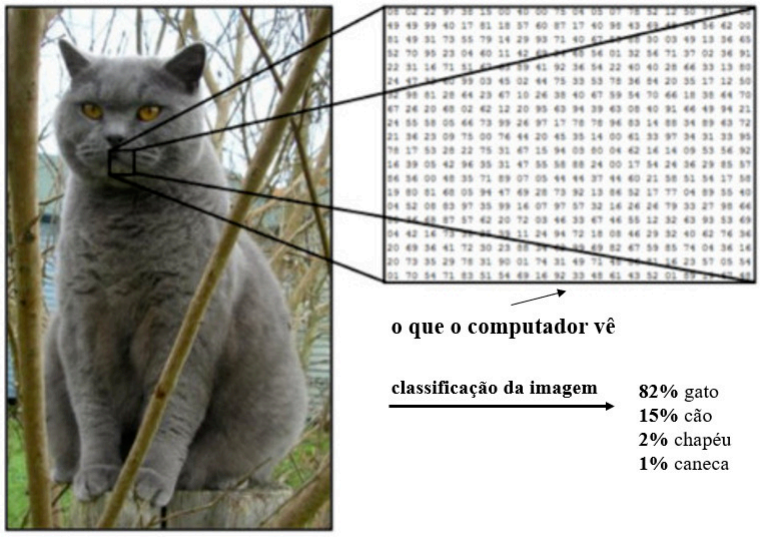



Figura 3.8: Classificação de imagem. Imagem adaptada de [17].

Através de uma câmara de um dispositivo móvel e de um modelo de rede neuronal é possível recolher e classificar imagens, de modo a reconhecer objetos na cena do utilizador. Um modelo de rede neuronal treinado é usado para realizar a classificação de imagens. Um exemplo de uma classificação de imagem pode ser observado na Figura 3.8. Como se pode observar na mesma, um modelo treinado para classificar animais e objetos, gera um número definido de resultados de classificação (82 por cento gato, 15 por cento cão, entre outros), de acordo com o que é reconhecido na imagem de entrada. Neste caso é recebido como entrada,

uma imagem de um gato, então o modelo classifica-a como gato, indicando a confiança que tem no resultado obtido. Outros resultados podem ser apresentados, no entanto destaca-se aquele com maior percentagem de confiança.

## 3.5 Estudo comparativo de diferentes algoritmos de deteção de objetos

A escolha do algoritmo a utilizar no reconhecimento de objetos, é uma tarefa bastante influente e indispensável para alcançar um sistema robusto. Características que os tornam diferentes, como a precisão de classificação e rapidez com que as imagens são processadas por segundo, são aspetos decisivos de seleção. Na Secção 2.2 do Capítulo 2 vimos que a preferência dos autores divide-se entre o algoritmo SSD e YOLO. Da mesma forma que observamos em [23] e [43], de seguida, apresentamos um estudo comparativo dos algoritmos mais populares e utilizados até ao momento. A nossa análise incluiu os seguintes algoritmos: YOLO, R-CNN, Fast R-CNN, Faster R-CNN e SSD. Este estudo teve como objetivo encontrar uma solução eficiente e adequada para a construção do sistema de deteção de objetos desejado neste projeto de dissertação.

| Algoritmo de deteção | Precisão mAP (%) | | FPS | GPU utilizada |
|---|---|---|---|---|
| | VOC 2007 | VOC 2012 | | |
| SSD | **74,3** | **73,1** | **59** | NVIDIA Corporation (nome da GPU não mencionada) |
| YOLO | 53,4 | 57,9 | 45 | NVIDIA Titan X |
| Faster R-CNN | 73,2 | 70,4 | 7 | NVIDIA K40 |
| Fast R-CNN | 66,9 | 65,7 | 5 | NVIDIA K40 |
| R-CNN | 58,9 | 53,3 | <1 | NVIDIA Corporation (nome da GPU não mencionada) |

Tabela 3.2: Características dos algoritmos de deteção de objetos.

Como ponto de partida, comecemos por observar duas variáveis de estudo importantes: mAP e Frames Per Second (FPS). A primeira, é uma métrica de avaliação muito popular e utilizada por vários algoritmos de deteção de objetos, a fim de medir a precisão dos seus modelos. Neste caso, permite-nos saber o quão bom o algoritmo ou modelo é capaz de detetar e reconhecer objetos. Outra variável notável é a FPS, que nos indica o número de *frames* processadas por segundo. Isto é, define a rapidez com que cada um consegue processar imagens e gerar a saída desejada. Na Tabela 3.2 são apresentados os valores de cada variável para cada um dos algoritmos estudos, bem como algumas características importantes.

### 3.5.1 R-CNN

Em 2014, numa conferência IEEE sobre visão computacional e reconhecimento de padrões, Ross Girshick, Jeff Donahue, Trevor Darrell e Jitendra Malik propuseram um algoritmo de deteção de objetos chamado R-CNN. Em [24], os autores explicam o seu trabalho começando por descrever que, o seu sistema, utiliza a estratégia "reconhecimento usando regiões", defendida por Chunhui Gu e outros autores em [93], para extrair cerca de duas mil propostas

de região na imagem. De seguida, para cada uma delas, calcula recursos recorrendo a uma grande CNN. Por último, classifica cada região. No conjunto de teste Pascal Visual Object Classes (VOC) 2007, o R-CNN obteve uma pontuação de 58,9% mAP e no VOC 2012 53,3%.

### 3.5.2 Fast R-CNN

Em 2015, uma nova versão do R-CNN foi sugerida por Ross Girshick, designada por Fast R-CNN. O novo algoritmo de treino foi desenvolvido com o objetivo de superar alguns problemas encontrados no R-CNN, bem como melhorar a sua velocidade de treino e teste, e aumentar a precisão de deteção. Tal como o próprio nome indica, Fast R-CNN, em português, R-CNN rápido, foi desenvolvido com o argumento de ser mais veloz em relação ao seu antecessor. No trabalho apresentado em [25], Ross Girshick, aponta as várias vantagens do algoritmo Fast R-CNN:

1. Fast R-CNN é mais rápida e apresenta maior qualidade de deteção, ou seja, um valor mais alto de precisão mAP, em comparação com o R-CNN;

2. Neste novo algoritmo, foi incluída uma nova camada de Region of Interest (ROI) *Pooling*, que permite a extração de varias propostas de região de interesse (ROIs), na mesma imagem de entrada. Para cada proposta de objeto, ROI *Pooling*, garante que a saída seja sempre do mesmo tamanho. Ou seja, é capaz de extrair vetores de recursos de comprimento igual, em todas as propostas. Isto concede uma maior velocidade de processamento, sem afetar a precisão de deteção. Para além disso, uma vez que o treino é caro em termos de tempo, esta abordagem permite economizar muito tempo;

3. Fast R-CNN propõem um treino mais eficiente, ao efetuar a partilha de recursos (computação e memória) entre todas as ROIs da mesma imagem.

4. Relativamente à arquitetura, ao contrário dos três estágios apresentados pelo R-CNN (extração de propostas de região na imagem, cálculo de recursos e classificação utilizando Support Vector Machines (SVM)), na Fast R-CNN, o treino é feito apenas num único estágio;

5. Na R-CNN, para cada imagem, os recursos extraídos de cada proposta de objeto são gravados no disco. Eles requerem centenas de *gigabytes* de armazenamento. Em contrapartida, o Fast R-CNN não os armazena, e, sendo assim, não necessita de tanto armazenamento em disco, permitindo poupar muito espaço.

Quanto ao valor de precisão, este algoritmo obteve 66,9% de mAP no conjunto de teste Pascal VOC 2007 e 65,7% no VOC 2012. Ainda que as vantagens deste algoritmo sejam fortemente favoráveis, ele apresenta uma desvantagem que o torna menos apelativo. Acontece que o Fast R-CNN, tal como o R-CNN, dependem do algoritmo de busca seletiva para gerar propostas de região. Isso acaba por tornar o processo mais demorado. Este problema foi resolvido com a construção de uma nova versão do Fast R-CNN.

### 3.5.3 Faster R-CNN

No mesmo ano, em 2015, o mesmo grupo de investigadores da Microsoft, apresentou o seu trabalho mais recente, um novo sucessor do Fast R-CNN, chamada Faster R-CNN. Esta nova arquitetura acaba por ser uma extensão da sua antecessora, e tal como nome indica, foi desenvolvida para ser ainda mais rápido que o Fast R-CNN. Nesta nova versão, foi introduzida uma nova mudança no algoritmo, através da implementação de uma Region Proposal Network (RPN). Em [26], os autores, descrevema RPN como uma rede totalmente convolucional, capaz de prever simultaneamente os limites dos objetos e as pontuações em cada deteção. Referem ainda que, o treino desta rede é feita de ponta a ponta (imagem completa), permitindo gerar propostas de região de alta qualidade. Umas das suas finalidades é a partilha de recursos (cálculos) convolucionais entre a RPN e a Fast R-CNN. Esta técnica permite uma redução significativa do tempo computacional.

A sua arquitetura divide-se em duas etapas: primeiro, por meio da RPN, são geradas propostas de região, e por último, com a contribuição da Fast R-CNN, são detetados objetos nas mesmas. De forma sucinta, o sistema Faster R-CNN funciona da seguinte forma [94]:

1. Dada uma imagem de entrada, o RPN gera varias propostas de região;
2. Como este algoritmo é um acrescento do Fast R-CNN, através da camada de ROI Poling, ele extrai um vetor de recurso de comprimento fixo para todas as propostas e classificao;
3. No final, obtém as pontuações de cada classe de objetos detetada, bem como as caixas delimitadoras;

A Faster R-CNN, com a introdução do RPN, tornou-se numa arquitetura mais poderoso, melhorando a qualidade das propostas de região e a precisão geral de deteção de objetos. No conjunto de teste Pascal VOC 2007, obteve um valor mAP de 73,2%, e no VOC 2012 70,4%. Contudo, apesar das vantagens e resultados promissores obtidos por parte do Fast R-CNN, o facto de este e dos seus antecessores apresentarem um FPS muito baixo, acaba por não ser suficiente para serem equipados num sistema de deteção em tempo real. De seguida descrevemos dois algoritmos que oferecem a possibilidade de detetar objetos em tempo real, uma vez que apresentam uma taxa de FPS suficientemente alta.

### 3.5.4 YOLO

YOLO, é outra abordagem publicada por Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhad em 2016. Em [27], eles afirmam que a sua arquitetura é extremamente rápida, com o seu modelo YOLO básico capaz de processar, em tempo real, quarenta e cinco quadrados de imagens por segundo. Para além disso, apresenta uma latência baixa, com menos de vinte e cinco milissegundos. Os autores descrevem a técnica baseada em proposta de região, adotada pelo R-CNN, como lenta e difícil de otimizar, porque cada componente deve ser treinado separadamente. Em contrapartida, o YOLO, vê a imagem como inteira. Tal como o próprio nome indica, ele olha apenas uma vez para cada imagem, durante o processo de

treino e teste. Portanto, este algoritmo treina imagens completas e otimiza diretamente o desempenho de deteção. Este sistema funciona de forma simples, em três etapas:

1. A imagem de entrada é redimensionada para 448x448;
2. De seguida, uma única CNN é executada na imagem;
3. Por fim, as deteções resultantes são limitadas pela confiança do modelo.

### 3.5.5 SSD

No final de novembro de 2016, foi apresentado um novo algoritmo de deteção de objetos chamado SSD. Nessa época, ele conquistou o valor mais alto de precisão de deteção, com uma percentagem mAP de 74,3 % no conjunto de teste VOC 2007 e 73,1% no VOC 2012. Para além de apresentar um vantajoso valor mAP, ele também é significativamente o mais rápido em comparação com os restantes. Tal afirmação deve-se ao facto de ele ser capaz de processar cinquenta e nove *frames* por segundo.

Tal como o algoritmo YOLO, no SSD, as tarefas de localizar e classificar objetos são feitas numa única instância, através das propriedades *MultiBox* resultantes do trabalho desenvolvido em [95]. No que diz respeito a arquitectura, o algoritmo SSD, baseia-se na arquitetura da rede Visual Geometry Grou (VGG)-16. A rede VGG-16 é utilizada para extrair características nas imagens de entrada. Em vez de utilizar o fenómeno da janela deslizante, que percorre a imagem de entrada, o SSD divide a imagem como se fosse uma grelha e cada célula fica responsável por detetar objetos naquela região.

### 3.5.6 Apreciação do estudo realizado

Perante o estudo realizado, facilmente se chega a conclusão que os algoritmos YOLO e SSD são os mais eficazes na classificação de imagens, em tempo real. Isto porque, para construir um sistema que inclua um recurso de deteção de objetos com essa característica, eles são os únicos que apresentam uma taxa de FPS elevada. (observar Tabela 3.2). Para além disso, eles processam a imagem de entrada num único estágio, promovendo um treino muito mais hábil. Em termos de precisão mAP, o algoritmo SSD obteve melhores resultados em comparação com o YOLO, o que indica que provavelmente seja a melhor escolha a seguir.

## 3.6 Considerações finais

Em suma, este capítulo reuniu todas as técnicas, abordagens e conceitos fundamentais para adquirir um conhecimento enriquecedor sobre como é realizado o reconhecimento de objetos em imagens digitais. Inicialmente foi apresentado todo o material teórico necessário para compreender a natureza de um modelo de rede neuronal artificial. Por último, o presente capítulo, terminou com um estudo aprofundado de diferentes algoritmos de deteção de objetos, a fim de encontrar um bom caminho a seguir no projeto prático proposto nesta dissertação.

# Capítulo 4

# Descrição da solução proposta

## 4.1 Introdução

Neste capítulo será apresentada a solução proposta para a construção da aplicação móvel desejada nesta dissertação. O mesmo tem como objetivo apresentar todas as abordagens de desenvolvimento necessárias para orientar, acompanhar e monitorizar todo o trabalho a realizar, de modo a alcançar com sucesso todos resultados que são esperados. Em primeiro lugar, será mostrada a arquitetura projetada para a solução do problema, ou seja, o modelo que descreve a ideia do sistema a ser desenvolvido. Logo de seguida, a calendarização programada para a elaboração da aplicação móvel. Por fim, serão nomeados os resultados que se pretendem atingir no final do processo de produção do sistema.

## 4.2 Arquitetura da solução

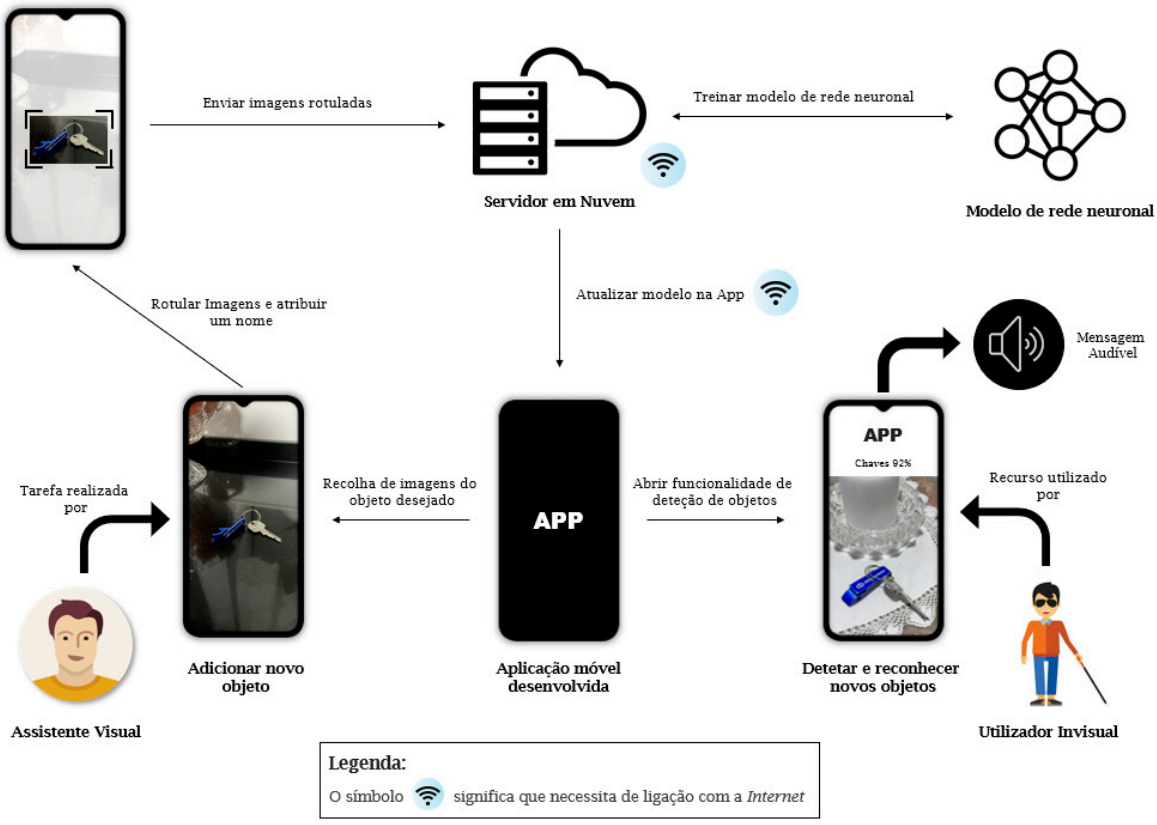



Figura 4.1: Arquitetura proposta para o desenvolvimento da aplicação móvel em estudo.

Na Figura 4.1 encontra-se representada a arquitetura proposta para o desenvolvimento

da aplicação móvel destinada a pessoas invisuais. A tarefa de adicionar novos objetos começa por ser realizada na própria aplicação, com a ajuda de uma assistente visual, que pode ser um familiar ou amigo do utilizador. Ele apenas terá de recolher algumas imagens que sejam representativas do objeto que o utilizador deseje que a aplicação reconheça e, de seguida rotulá-las. O processo de rotular imagens é fundamental no treino e na aprendizagem de uma rede neuronal, uma vez que permite prever e reconhecer padrões. Após esse passo ser concluído, as imagens originais e as anotadas serão enviadas para um servidor em nuvem, onde ocorrerá o treino de um modelo de rede neuronal. O mesmo ficará apto a detetar e reconhecer todos objetos que o utilizador invisual desejar. No final, ele terá apenas de baixar o modelo na aplicação e abrir a funcionalidade correspondente para começar a identificar os seus novos objetos. A aplicação fornecerá uma mensagem audível com informação sobre o nome dos objetos detetados nas imagens recolhidas em tempo real.

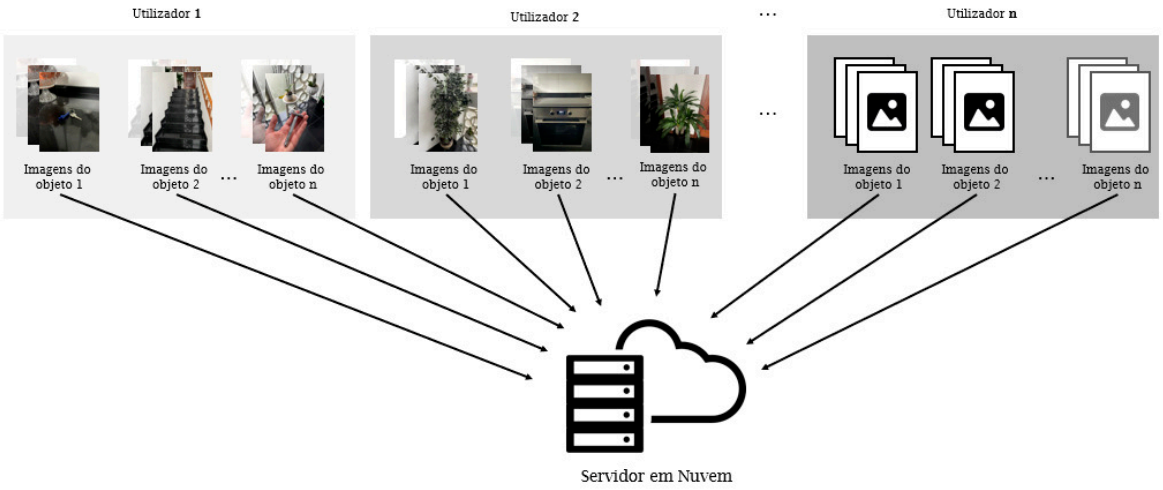



Figura 4.2: Imagem ilustrativa da funcionalidade que permitirá enviar as imagens dos objetos de vários utilizadores para o mesmo servidor em nuvem.

Na Secção 2.6 do Capítulo 2, foi ainda mencionado o desafio de investigar a hipótese dos vários utilizadores da aplicação móvel poderem enviar todas as imagens dos seus novos objetos para o mesmo servidor, com o objetivo de criar um modelo de rede neuronal mais ágil e proveitoso. Para além disso, o esforço e o tempo necessários para concretizar um modelo mais evolutivo é menor, em comparação com os modelos personalizados, que exigem muita dedicação. Não esquecendo também que o modelo obteria uma gama mais variada de objetos, sendo todos úteis para a finalidade em questão. Na Figura 4.2, é possível observar um esquema que reflete a ideia proposta.

Por último, é importante evitar que a rede neuronal classifique dois objetos de dois utilizadores como diferentes, e na verdade eles serem o mesmo. Por exemplo, imaginemos que um determinado utilizador regista o seu novo objeto com nome “Televisão” e vêm outro utilizador e atribui o nome “TV”. Ambos se referem ao mesmo, no entanto, o modelo de rede neuronal passa a dois objetos distintos, uma vez que existem duas anotações ou rótulos diferentes. Isso pode provocar dificuldades no reconhecimento e tornar menos eficaz o processo de treino da rede. Sendo assim, tem que haver um consentimento universal.

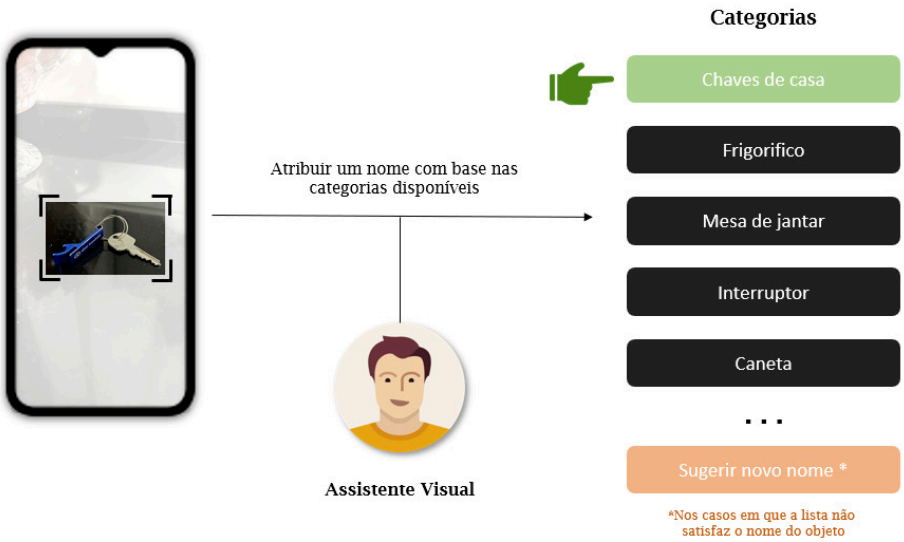



Figura 4.3: Esquema que reflete a atividade de atribuir um nome ao objeto desejado, com base em categorias.

Uma saída que permite solucionar a referida situação, é definir uma lista de categorias e atribuir um nome ao objeto com base numa delas. Em vez do assistente visual digitar o nome, ele passa a ter de selecionar, na lista, o item correspondente. Caso nenhum dos nomes apresentados forem adequados ao objeto em questão, a aplicação oferece a oportunidade de o assistente sugerir um novo nome. O mesmo passará a ser visto também pelos outros utilizadores, de forma a não criar ambiguidades. Esta solução proposta pode ser observada, de forma esquemática, na Figura 4.3.

## 4.3 Servidor em Nuvem

Neste trabalho, a ideia inicial seria utilizar um único servidor para o armazenamento de imagens e treino de um modelo de rede neuronal, como demonstrado na arquitetura divulgada na secção anterior. Porém, o servidor que foi requisitado, denominado por Oblivion, é uma máquina que se encontra isolada do mundo exterior, podendo apenas ser acedida através do protocolo Secure Shell Protocol (SSH) e do respetivo endereço. Sendo assim, não foi possível instalar um servidor Web no servidor Oblivion a fim de enviar os dados da aplicação móvel para o mesmo, uma vez que as portas não se encontram abertas. Posto isto, a solução para resolver esta situação passou por usar dois servidores: um servidor que realiza o treino e um servidor que recebe os dados enviados pela aplicação.

A Figura 4.4 apresenta um esquema que retrata toda solução pensada para ser possível enviar os dados da aplicação móvel para o servidor Oblivion, e o modelo de rede neuronal para aplicação. Tal será feito através da introdução de um servidor Apache entre essas duas entidades. Numa máquina física, localizada na Universidade da Beira Interior (UBI), foi instalado um servidor Apache que permite compartilhar dados entre o dispositivo móvel de cada utilizador e o servidor Web. Essa etapa é realizada por meio dos métodos POST e GET, que serão explicados mais a frente na Secção 5.3.5. Após os dados terem sido armazenados no servidor Apache, os mesmos serão novamente enviados para o servidor Oblivion via SSH,

com objetivo de treinar um novo modelo de rede neuronal nesse conjunto de dados. Por fim, na ordem inversa, o servidor Apache recupera o modelo treinado por meio do comando Secure Copy Protocol (SCP), e envia-o para aplicação móvel.

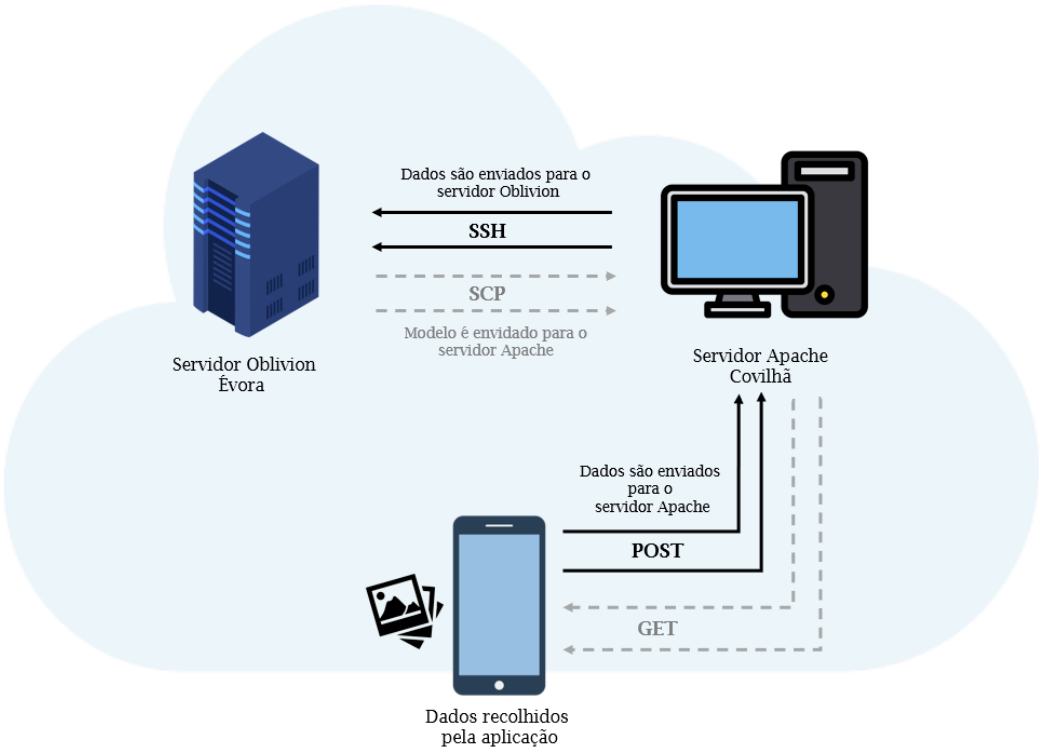



Figura 4.4: Esquema que reflete como são enviados os dados da aplicação móvel para o servidor Oblivion, e na ordem inversa, como é enviado o modelo de rede neuronal para aplicação, mediante a utilização de um servidor Apache.

### 4.3.1 Servidor Oblivion

Oblivion é um supercomputador adquirido pela Universidade de Évora, capaz de processar grandes volumes de dados (na ordem de *Petabytes*, o equivalente a milhões de *Gigabytes*) [96]. A máquina dispõe de recursos computacionais e do *software Vision*, que inclui características especialmente adaptadas a trabalhos de inteligência artificial e *Deep Learning*. O sistema *Vision* oferece ambientes Python e Anaconda, permitindo que os utilizadores possam desenvolver o seu próprio ambiente de execução. Para além disso, disponibiliza ainda módulos de *software* que podem ser carregados de modo a satisfazer as dependências das suas aplicações [97]. Neste projeto, Oblivion foi utilizado como servidor central para realizar os cálculos necessários para o treino de um modelo de rede neuronal. Na tabela apresenta em 4.1, é possível observar as características e recursos de *Hardware* utilizados pelo supercomputador Oblivion.

| Oblivion *Hardware* | |
|---|---|
| Graphics Processing | Utilização máxima de 4 GPUs em simultâneo. (NVIDIA A100 Tensor Core GPUs (40gb per GPU) |
| CPU | Dual AMD Rome 7742 |
| System Memory | 1TB |
| Storage Capacity | 10 TB |

Tabela 4.1: Características e recursos computacionais do servidor Oblivion.

### 4.3.2 Servidor Apache

O servidor Apache instalado numa máquina física da UBI, concedeu a alternativa de partilhar informação (dados) com o dispositivo móvel de cada utilizador. Apache é um servidor Web rápido, confiável e seguro, que permite manusear solicitações do tipo Cliente/Servidor. Mais à frente na Secção 5.3.1 do Capítulo 5 falar-se-á um pouco mais sobre este servidor.

## 4.4 Calendarização

Com o objetivo de organizar todo o trabalho a realizar, foi desenvolvido um cronograma que define uma data prévia para a execução de cada tarefa estabelecida no início do projeto. Este controlo e gestão do programa de produção do sistema em estudo, pode ser observado por meio de um diagrama de *Gantt*, apresentado na Figura 4.5.

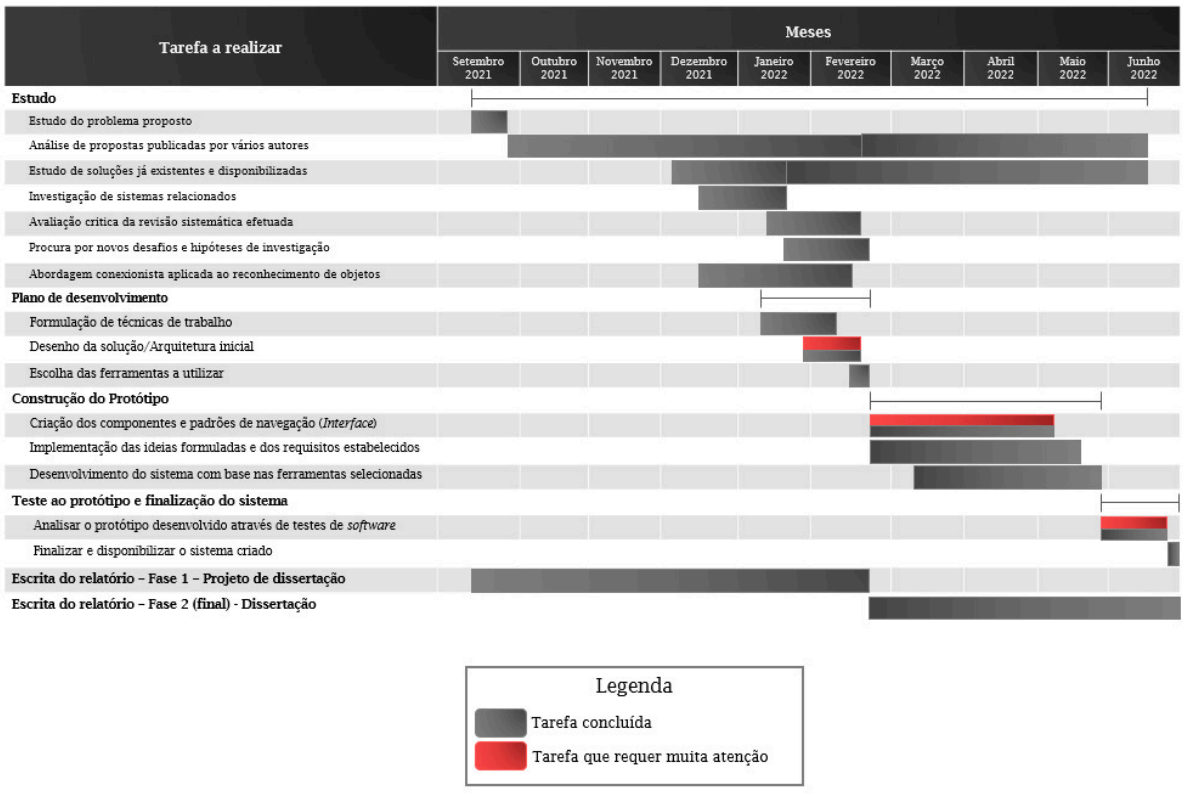



Figura 4.5: Diagrama de *Gantt* que ilustra o cronograma das diferentes etapas de construção do sistema.

## 4.5 Resultados esperados

Em geral, no final do processo de produção do sistema, espera-se obter como resultado uma aplicação móvel bem-sucedida, onde todas as etapas de desenvolvimento foram executadas e finalizadas com sucesso. É importante que todos desafios que foram definidos na Secção 2.6 sejam conseguidos, bem como todo o plano de desenvolvimento abordado neste capítulo. De forma mais metódica, espera-se obter os seguintes resultados:

1. Criar uma aplicação móvel capaz de detetar e reconhecer objetos presentes no ambiente que rodeia o utilizador;
2. Conseguir oferecer uma funcionalidade que permita que um assistente visual do utilizador possa adicionar qualquer objeto ao sistema;
3. Obter resultados promissores de reconhecimento, ou seja, qualquer objeto que tenha sido implementado deve atingir uma boa precisão de classificação (acima de 90%);
4. Espera-se ainda que a junção dos vários objetos de todos os utilizadores da aplicação num único modelo de rede neuronal seja efetuado com êxito. Para além disso, é importante que a adição de novas categorias de objetos não enfraqueça os bons resultados de precisão no reconhecimento.

## 4.6 Considerações finais

Em suma, este capítulo apresentou, de forma sucinta, todo o plano de desenvolvimento que será utilizado posteriormente na fase de produção do sistema. Através do mesmo, foi possível adquirir uma visão futura sobre como será construída e implementada a aplicação móvel desejada nesta dissertação. Para além disso, o plano de desenvolvimento elaborado permitiu definir metas e técnicas de preparação para alcançar todos os desafios em aberto, mencionados na Secção 2.6.

# Capítulo 5

# Trabalho Desenvolvido

## 5.1 Introdução

O presente capítulo tem como objetivo anunciar todo o trabalho que foi realizado, com base na solução proposta e conforme os objetivos que foram estabelecidos no início deste projeto. Uma vez que o mesmo é um trabalho de grande dimensão, este capítulo encontra-se dividido em várias secções. Em primeiro lugar, serão descritas as tecnologias e ferramentas utilizadas na concretização do sistema, isto é, todos os recursos que foram necessários para a construção da aplicação móvel desejada e para a correta utilização dos servidores que foram requisitados. De seguida, serão delineados todos os requisitos funcionais e não funcionais do sistema desenvolvido, incluindo os diagramas de casos de uso e de atividade correspondentes às funcionalidades implementadas na aplicação. Por fim, sendo esta uma plataforma que se destina a pessoas invisuais, o desenho e a conceção da respetiva *interface* foi um dos atributos mais notável neste trabalho. Posto isto, no final deste capítulo é dada a conhecer como foram projetadas todas as atividades desenvolvidas, de modo que o utilizador final seja capaz de interagir com o sistema de forma fácil e autónoma.

## 5.2 Tecnologias e Ferramentas Utilizadas - Aplicação móvel

Para a criação da aplicação móvel esperada, alguns instrumentos de trabalho foram cruciais para atingir o objetivo pretendido. De seguida, nesta secção, serão apresentados os meios utilizados na elaboração da aplicação, começando pela ferramenta base, o Android Studio.

### 5.2.1 Android Studio

O Android Studio permitiu planear e construir a aplicação móvel, visto que é um ambiente de desenvolvimento integrado, que oferece ferramentas proficientes para criar aplicações móveis compatíveis com todos os dispositivos que utilizam o sistema operativo *Android* [98]. Edição de código inteligente, suporte de compilação baseado em *gradle*, emuladores que concedem a oportunidade de testar o projeto idealizado em várias resoluções de ecrãs de telemóveis, são alguns dos recursos atraentes que motivaram a sua escolha para este projeto.

### 5.2.2 Java

Atualmente, as linguagens de programação suportadas pelo Android Studio na criação de aplicações *Android* são a Kotlin e Java. Neste projeto, recorreu-se à linguagem Java para escrever todo o código necessário para a criação do sistema pretendido. A mesma é descrita como uma linguagem de programação orientada a objetos, sendo amplamente utilizada na

construção de aplicações *Android*. Para além disso, é também uma plataforma de computação que oferece recursos úteis no desenvolvimento de qualquer tipo de aplicação móvel.

### 5.2.3 Retrofit Android

O Retrofit é uma biblioteca muito popular de Hyper-Text Transfer Protocol (HTTP) Cliente, desenvolvida pela empresa Square para Java e Android. Uma das vantagens principais do Retrofit é a sua simplicidade, sendo capaz de facilitar o processo de receber e fazer o *upload* de ficheiros no formato JavaScript Object Notation (JSON) por meio de *Web service Representational State Transfer (REST)*. Neste trabalho, esta biblioteca foi implementada na aplicação Android com intuito de enviar os dados de novos objetos para o Servidor Apache e baixar o novo modelo de rede neuronal que foi treinado no Servidor Oblivion. A ferramenta Retrofit é utilizada para fazer solicitações de API, uma vez que é responsável por gerir o processo de recebimento, envio e criação de solicitações e respostas HTTP [99].

### 5.2.4 TensorFlow

O TensorFlow é uma plataforma que disponibiliza gratuitamente ferramentas, bibliotecas e recursos que apoiam o desenvolvimento e a implementação de aplicações móveis que envolvam a tecnologia de aprendizagem automática, em inglês Machine Learning (ML) [100]. Esta estrutura de aprendizagem oferece a possibilidade de criar e treinar redes neuronais para detetar e reconhecer objetos. A API do TensorFlow foi utilizada, neste projeto, para construir o recurso de identificação de objetos.

### 5.2.5 *Text-To-Speech*

*Text-To-Speech* é um termo utilizado na área da tecnologia assistiva para descreve a ação de converter texto digital em voz sintetizada [101]. Qualquer sistema que inclua *Text-To-Speech* é capaz de simular a voz de um ser humano, por meio da entrada de uma representação textual. Esta tecnologia é fundamental no apoio a pessoas invisuais, pois concede um método viável e eficiente de comunicação entre o dispositivo móvel e o utilizador invisual. Através da mesma é possível mantê-lo a par de tudo o que está a acontecer ao seu redor. Neste projeto, *Text-To-Speech* é utilizada parca converter o nome dos objetos reconhecidos em uma mensagem audível, de modo que utilizador final da aplicação móvel consiga saber os objetos que se encontram a sua frente.

### 5.2.6 *Speech-To-Text*

Na ordem inversa da tecnologia anterior, *Speech-To-Text* é uma técnica que é capaz de transformar voz audível em texto editável. Todo esse processo é normalmente conhecido como reconhecimento de voz e baseia-se em algoritmos que procuram classificar sinais auditivos de palavras faladas e convertê-los em texto, usando caracteres *Unicode* [102]. O *Unicode* permite atribuir um código (número) a um determinado caráter, independentemente do dispositivo utilizado ou idioma falado, concedendo assim uma forma viável de converter voz em

texto a partir de diferentes línguas [103]. Neste projeto, a tecnologia *Speech-To-Text* é usada num recurso que permita reconhecer o nome do objeto falado pelo utilizador, com o objetivo de encontrar esse objeto especifico no ambiente que o rodeia.

## 5.3 Tecnologias e Ferramentas Utilizadas - Servidor Apache

No que diz respeito ao servidor Apache empregue neste projeto, algumas tecnologias e ferramentas tiveram de ser devidamente instaladas no mesmo, de modo que ele pudesse cumprir com a sua função, que é receber os dados (imagens e ficheiros XML) do sistema, e enviar para aplicação móvel o modelo treinado no novo conjunto de dados. De seguida, abordamos todos os artefactos necessários, começando primeiro pelo Apache Server.

### 5.3.1 Apache Server

O Apache Server é considerado o servidor *web* mais popular que existe atualmente, capaz de executar código em Hypertext Preprocessor (PHP) e processar solicitações HTTP, o protocolo que permite a comunicação e a transferência de informação através da *Internet*. Uma vez que o servidor e o cliente foram conectados mediante o protocolo HTTP, neste trabalho o Apache Server foi responsável por estabelecer uma comunicação segura entre essas duas entidades. Desta forma, a aplicação móvel poderá decretar uma ligação com o servidor Apache criado, para enviar e receber dados.

### 5.3.2 MySQL

MySQL é um *software* desenvolvido pela empresa Oracle, que utiliza a linguagem Structured Query Language (SQL) para criar, organizar e gerir bases de dados [104]. Através do mesmo, é possível armazenar dados em uma variedade de tabelas. Neste trabalho, o MySQL foi instalado no servidor Apache, com objetivo de armazenar as imagens e os ficheiros XML numa pasta localizada no servidor e guardar o caminho dos mesmos numa base de dados. Foi criada uma única base de dados, com duas tabelas, uma destinada as imagens e a outra aos ficheiros XML. Portanto, sempre que um dado for recebido pelo servidor, o caminho do mesmo será anexado na tabela correspondente. Mais a frente, no Capítulo 6, é dada a conhecer, com mais detalhe, a importância destas tabelas na organização dos dados e na atribuição de um Identification (ID) a cada imagem enviada para o servidor Apache.

### 5.3.3 PHP

O PHP é uma linguagem de programação amplamente utilizada no desenvolvimento de aplicações para a plataforma *Web*, bem com na criação de *Websites* [105]. Neste projeto, a linguagem PHP foi utilizada para construir a API necessária para realizar a ação de *upload* de imagens e de ficheiros XML para o servidor. A mesma contém todos os passos e informação necessária para que o envio dos dados seja feito de forma competente.

### 5.3.4 phpMyAdmin

O pypMyAdmin é um *software* escrito em PHP, que foi desenvolvido com o objetivo de lidar com a administração do MySQL através da *Internet* [106]. Esta plataforma é capaz de tornar todo o processo de gestão de base dados menos trabalhoso, uma vez que foi construída com uma interface simples e de fácil utilização. Através da mesma é possível criar e editar base dados e tabelas, sem a necessidade de recorrer a linhas de comandos de programação, o que torna a realização da tarefa muito mais rápida. Neste trabalho, a ferramenta pypMyAdmin serviu de base para acompanhar e controlar a base dados criada.

### 5.3.5 REST

O REST é um estilo de arquitetura de *software* que segue um conjunto de diretrizes a serem utilizadas na prestação de comunicações de serviços *Web*. O acrónimo REST define-se da seguinte forma: *Representational State Transfer*. O termo *State Transfer* significa que, num modelo de comunicação Cliente-Servidor, o servidor não necessita de saber o estado do cliente e vice-versa [107]. No universo REST, as solicitações HTTP são equivalentes a uma chamada que define uma operação a ser realizada em um objeto (também denominado por recurso) presente no servidor. Os métodos HTTP (GET, POST, PUT, DELETE) são normalmente utilizados para indicar a ação que se pretende efetuar num determinado recurso. No projeto desta dissertação, serão usados os métodos POST e GET, sendo que o primeiro permite que o cliente (neste caso, a aplicação móvel) insira ou crie um objeto no servidor (neste caso, uma imagem e um ficheiro XML). Já o método GET, procura obter/ler um recurso do servidor, ou seja, concede a aplicação uma forma segura de adquirir o modelo de rede neuronal que foi treinado. O diagrama apresentado na Figura 5.1 ajuda a compreender melhor todo o processo de comunicação entre a aplicação móvel e o servidor Apache, mediante a utilização dos serviços POST e GET.

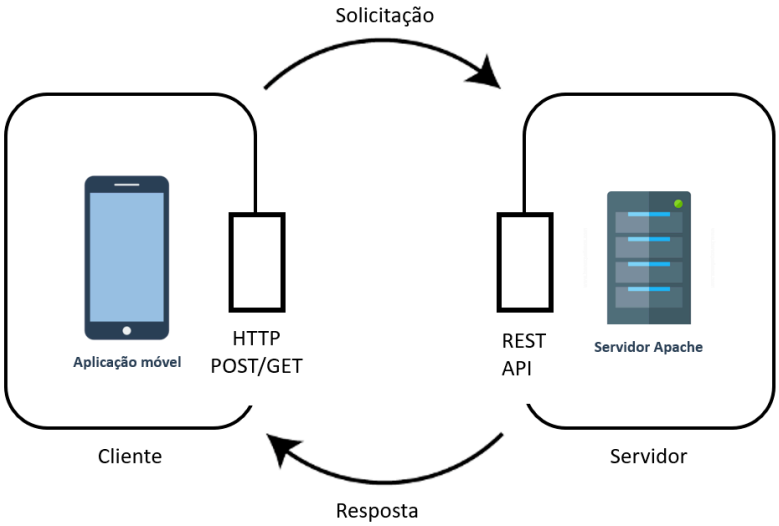



Figura 5.1: Arquitetura Cliente/Servidor. Imagem adaptada de [18].

Numa solicitação HTTP, o recurso é indicado pelo seu respetivo protocolo de *Internet* Uniform Resource Identifie (URI). Este descreve uma sequência de caracteres que identi-

ficam um recurso presente na *Internet*. Por sua vez, o Uniform Resource Locator (URL) engloba o caminho completo, ou seja, não só denomina o recurso como também indica a localização do mesmo na *Internet*, por meio do protocolo HTTP. Por exemplo, no caso de um pedido GET por parte do cliente, o respetivo URL seguiria a seguinte nomenclatura:

- **GET** https://dominio/Api.php?apicall=recurso

Na própria URL também podem ser passados parâmetros no corpo da solicitação. Isso é útil quando a tarefa desejada é inserir dados no servidor. Um exemplo de um pedido do tipo POST, pode ser o seguinte:

- **POST** https://dominio/Api.php?apicall=upload - recurso

A API indicada no respetivo URL é escrita na linguagem PHP e construída com base nas funcionalidades necessárias para as respetivas solicitações, seguindo todas as regras de programação.

### 5.3.6 Ubuntu 18.04.6

O Ubuntu é um sistema operativo gratuito, que foi disponibilizado em 2004 pela empresa Britânica Canonical. Atualmente é um sistema popular de distribuição Linux, adequado para computação em Nuvem e instalação de servidores [108]. Nesta parte do trabalho, o sistema operativo Ubuntu, na versão 18.06.6, foi utilizado para instalar um servidor Apache, bem como todas as ferramentas que foram abordadas nesta secção.

## 5.4 Tecnologias e Ferramentas Utilizadas - Servidor Oblivion

Durante a utilização do servidor central do sistema, Oblivion, várias ferramentas e tecnologias tiveram de ser introduzidas a fim de alcançar os resultados esperados. Tal como foi mencionado no capítulo anterior, este servidor ficou encarregue de realizar todos os cálculos computacionais necessários para treinar um modelo de rede neuronal artificial. Assim sendo, de seguida, serão apresentadas as diversas tecnologias e ferramentas que foram necessárias para que ele pode-se cumprir essa tarefa. É importante mencionar que o servidor Oblivion também foi trabalhado no sistema operativo Ubuntu, descrito na Subsecção 5.3.6 da Secção 5.3.

### 5.4.1 Tensorflow 2.9.0

Desenvolvida pela empresa Google em 2015, o TensorFlow é uma ferramenta imprescindível no desenvolvimento e criação de modelos de ML . Ele oferece um vasto conjunto de recursos inovadores que permitem produzir sistemas que sejam capazes de extrair padrões num grande conjunto de dados [109]. Por meio dos seus algoritmos eficientes, é possível construir e treinar modelos de rede neuronal artificial com sucesso. No projeto desta dissertação, a versão 2.9.0 do TensorFlow foi instalada no servidor Oblivion com o objetivo de implementar e criar o modelo de rede neuronal a ser usado na deteção de objetos.

### 5.4.2 Python 3.9.5

O Python é uma linguagem de programação de alto nível, muito utilizada em áreas de Aprendizagem Profunda e em estruturas de ML. Atualmente, tornou-se uma linguagem indispensável na implementação e treino de redes neuronais de multicamadas, devido às suas grandes bibliotecas, como o Tensorflow, OpenCV, Pandas, etc. Neste trabalho, no servidor Oblivion, o modulo 3.9.5 do Python foi instalado juntamente com os principais pacotes [110]:

1. **Pillow –** uma biblioteca de Python, que oferece suporte para abrir e gravar vários formatos de imagens;
2. **Matplotlib –** uma biblioteca de Python, que apoia a criação de gráficos e visualizações de dados. ;
3. **Numpy –** uma biblioteca popular no processamento de grandes matrizes em Python. Opera juntamente com várias funções matemáticas de alto nível para gerir, de forma rápida e eficiente, essas mesmas matrizes;
4. **OpenCV –** uma biblioteca multiplataforma que é normalmente aplicada no processamento de imagens e na execução de modelos de Aprendizagem Automática. Fornece determinadas funções que são muito úteis para o desenvolvimento de projetos de Visão Computacional;
5. **Pandas –** uma biblioteca de *software* desenvolvida para a linguagem Ptython, que oferece uma estrutura de dados intuitiva para apoiar a análise e manipulação de diversos tipos de dados;
6. **Libxml2 –** uma biblioteca de *software* que procura analisar e manipular documentos XML. A mesma foi implementada na linguagem C, porém fornece ligações com a linguagem Python;
7. **Keras –** uma poderosa biblioteca de rede neural desenvolvida em Python, capaz de executar TensorFlow e de fornecer ferramentas úteis para a construção de modelos de Aprendizagem Profunda.

### 5.4.3 SSD Mobilenet V2

O SSD Mobilenet v2 é um modelo de deteção de objetos capaz de determinar as caixas delimitadoras e a categoria de um objeto quando uma imagem for recebida como entrada [111]. Este modelo baseia-se no algoritmo SSD e foi treinado no conjunto de dados COCO. Atualmente, consegue detetar noventa classes diferentes de objetos. Neste trabalho, o modelo SSD Mobilenet v2 foi aplicado na aplicação móvel, com o objetivo de oferecer ao utilizador um sistema inicial de deteção de objetos, ou seja, uma aplicação móvel que já esteja preparada para o reconhecimento de alguns objetos. É de notar que, quando aplicação móvel for disponibilizada ao público, o modelo personalizado pelos vários utilizadores do sistema ainda está numa fase inicial de crescimento, o que significa que ainda não estará preparado para identificar muitas classes de objetos. Logo, o modelo SSD Mobilenet v2 vem apoiar o modelo personalizado, na sua fase inicial de desenvolvimento.

## 5.5 Análise de Requisitos

A secção que se segue procura descrever todos os requisitos funcionais e não funcionais que definem a aplicação móvel desenvolvida. A Engenharia de Requisitos é considerada uma das áreas mais importantes no desenvolvimento de um *software* e na análise de um sistema, uma vez que reúne todos os esforços necessários para que sejam atingidas boas práticas no final das atividades propostas. O seu conceito centra-se no processo de criação de um documento, chamado documento de requisitos, que recolhe e estabelece todos os serviços que são requisitados por um sistema. Geralmente, um requisito define-se como uma característica que deve estar presente no trabalho implementado, a fim de este alcançar determinados objetivos e dar resposta à solução projetada para o problema em questão. Todas as propriedades indispensáveis num sistema, como por exemplo restrições de funcionamento, requisitos de portabilidade, entre outros, devem ser interpretadas como partes fundamentais na estrutura e comportamento de um software, tornando a tarefa inicial de especificação de requisitos muito importante no processo de desenvolvimento da aplicação. Os requisitos podem-se dividir em dois tipos diferentes: requisitos funcionais e requisitos não funcionais.

### 5.5.1 Requisitos Funcionais

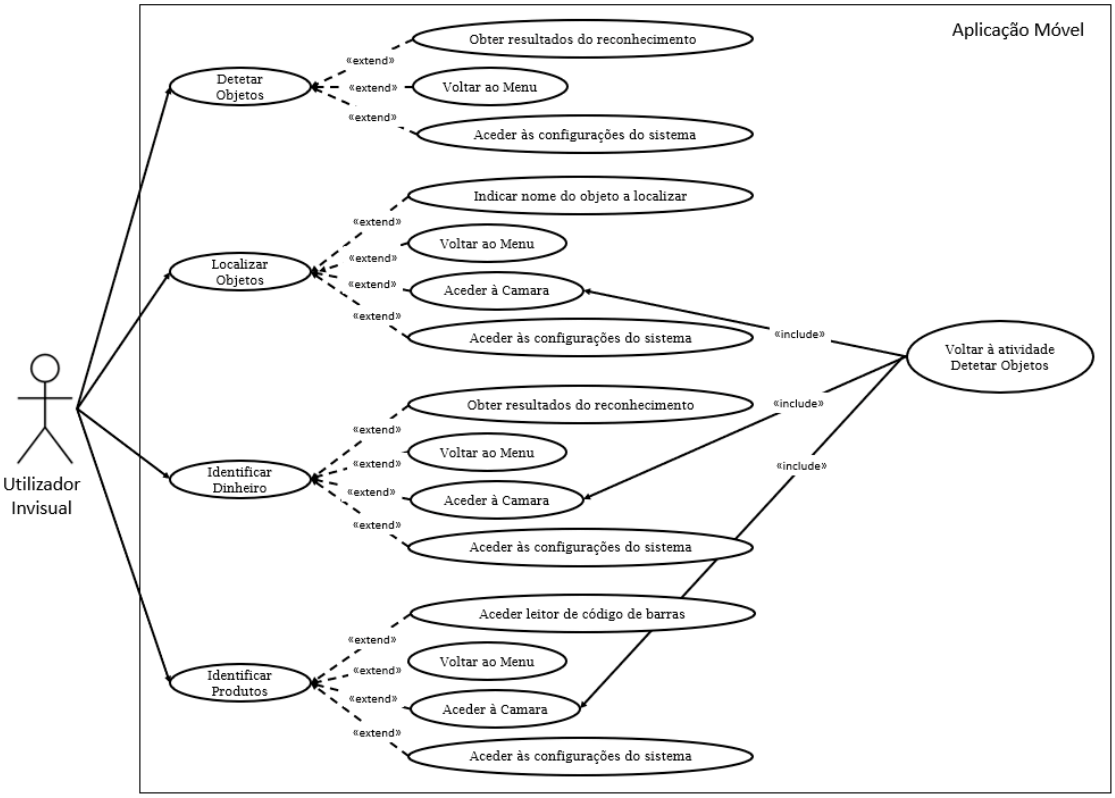



Figura 5.2: Caso de uso - Funcionalidades concedidas ao utilizador invisual.

Na Engenharia de Requisitos, um requisito funcional especifica uma determinada função ou atividade que o software deverá suportar. Desta forma, o conjunto de requisitos funcionais inclui todas as funcionalidades oferecidas por um sistema, descrevendo o comportamento de todos os seus componentes e serviços. Este exercício detalhado de como a aplicação

se encontra estruturada, a nível de tarefas que o consumidor final espera que venham a ser realizadas, serve de introdução para compreender o trabalho que foi implementado.

Normalmente, o material de documentação utilizado na especificação de requisitos funcionais assenta em diagramas de caso de uso, de atividade, de classes, entre outros. Neste trabalho recorremos a diagramas de casos de uso e de atividade para fazer o levantamento dos requisitos funcionais do sistema e demonstrar todas a funcionalidades disponíveis no mesmo. Começando pelos diagramas de caso de uso, a solução proposta completa dois tipos de atores: utilizador invisual e assistente visual. O diagrama apresentado na Figura 5.2 está associado ao utilizador invisual e o da Figura 5.3 ao assistente visual.


Figura 5.3: Caso de uso - Funcionalidades concedidas ao assistente visual.

Visto que um diagrama de caso de uso descreve todas as funcionalidades propostas por um sistema, do ponto de vista do utilizador, os diagramas apresentados resumem todas as atividades disponíveis dentro do cenário da aplicação móvel e de acordo com o tipo de ator ou utilizador da aplicação. Assim sendo, o sistema desenvolvido apoia-se nas seguintes seis funcionalidades principais, sendo que as quatro primeiras se referem ao utilizador invisual e as duas últimas ao assistente visual:

1. Detetar Objetos;
2. Localizar Objetos;
3. Identificar Dinheiro;
4. Identificar Produtos;
5. Adicionar um novo objeto ao sistema;
6. Adicionar um novo produto ao sistema.

De seguida, por meio de diagramas de atividade será mostrado o fluxo de controle de cada uma das seis funcionalidades mencionadas anteriormente, assim como os aspetos dinâmicos do sistema. Um diagrama de atividade apoia a compreensão de um determinado caso de uso, a um nível mais detalhado, e descreve como as atividades são coordenadas segundo uma sequência de ações que são realizadas pelo sistema. [112]

#### 5.5.1.1 Detetar Objetos

A funcionalidade Detetar Objetos foi desenvolvida no sistema para que o utilizador invisual seja capaz de adquirir conhecimento claro sobre determinados objetos que estejam presentes à sua volta, dando resposta a solução projetada para resolução do problema proposto nesta dissertação. Na Tabela 5.1 apresentada é possível observar informações mais pormenorizadas acerca deste primeiro requisito funcional.

| **Detetar Objetos** | | |
|---|---|---|
| **ID do Requisito:** 1 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Utilizador Invisual. | | |
| **Pré-condições:** Modelo de rede neuronal artificial para reconhecer objetos. | | |
| **Pós-condições:** Resultados serão apresentados se forem detetados objetos nas imagens recolhidas pelo sistema, em tempo real. Aquele que apresentar maior percentagem de confiança será divulgado. | | |
| **Descrição do requisito:** O utilizador invisual deverá ser informado sobre o nome dos objetos que forem detetados pela aplicação. | | |
| **Material de suporte:** Diagrama de atividade 5.4. | | |

Tabela 5.1: Requisito Funcional 1 - Descrição da funcionalidade Detetar Objetos.

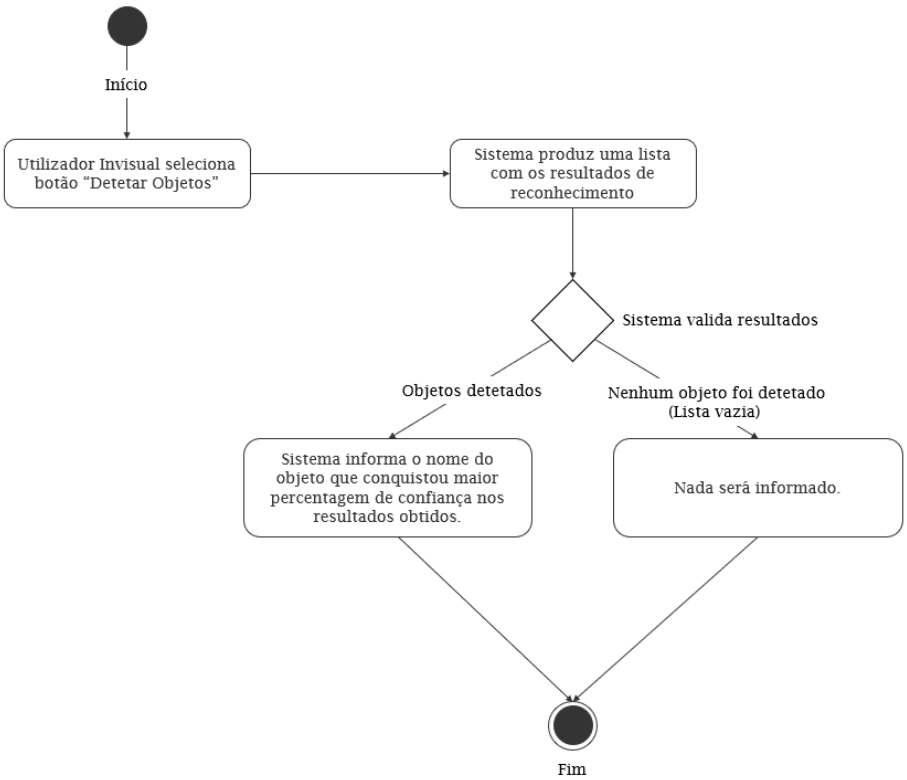



Figura 5.4: Diagrama de atividade associado à funcionalidade Detetar Objetos.

#### 5.5.1.2 Localizar Objetos

A segunda funcionalidade do sistema construído, Localizar Objetos, foi pensada de modo a oferecer ao utilizador invisual a possibilidade de localizar um objeto que tenha sido implementado no sistema. Este recurso extra foi desenvolvido com o intuito de tornar mais rápido e direto o processo de encontrar um objeto que seja solicitado pelo utilizador. Desta forma, o processo de reconhecimento é direcionado somente para aquele objeto específico, independentemente de ser identificados outros objetos na cena. A Tabela 5.2 apresenta, com mais detalhe, toda a informação associada a este segundo requisito funcional.

| **Localizar Objetos** | | |
|---|---|---|
| **ID do Requisito:** 2 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Utilizador Invisual. | | |
| **Pré-condições:** Acesso à *Internet* | | |
| **Pós-condições:** Objeto será localizado se o mesmo tiver sido implementado no sistema. Sinal sonoro será ativado quando o objeto for detetado pelo modelo de rede neuronal. | | |
| **Descrição do requisito:** O utilizador invisual poderá localizar um determinado objeto do sistema que deseje encontrar, indicando o nome do mesmo à aplicação por meio de *Speech-To-Text*. | | |
| **Material de suporte:** Diagrama de atividade 5.5. | | |

Tabela 5.2: Requisito Funcional 2 - Descrição da funcionalidade Localizar Objetos.

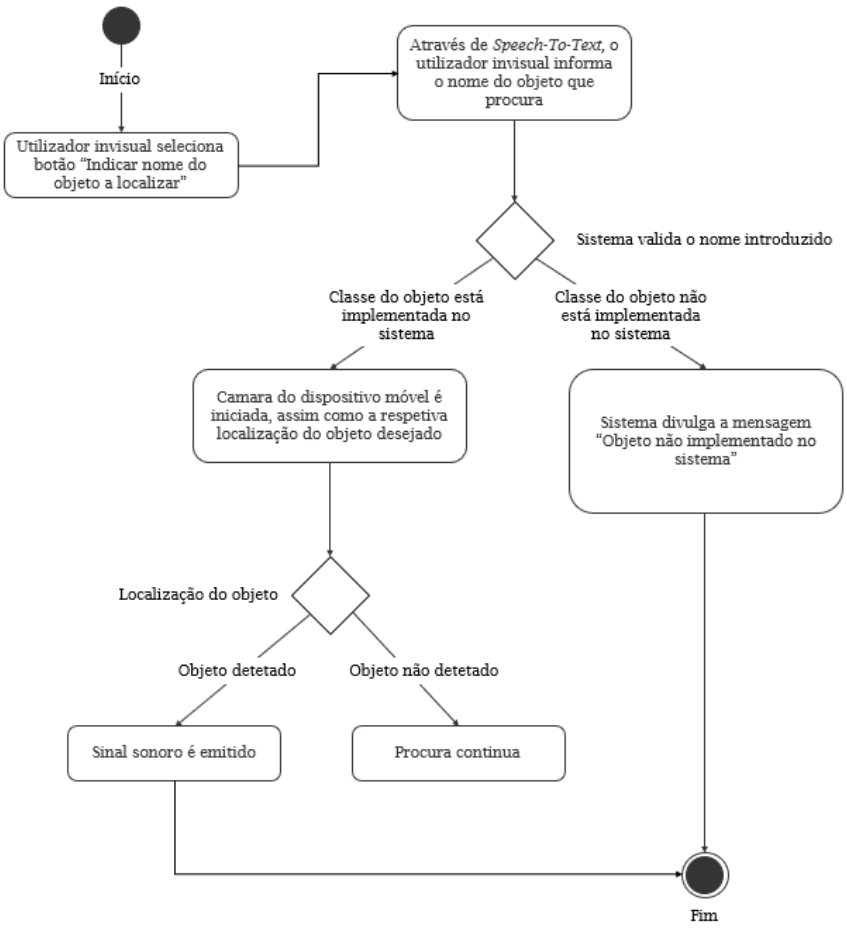



Figura 5.5: Diagrama de atividade associado à funcionalidade Localizar Objetos.

#### 5.5.1.3 Identificar Dinheiro

Identificar dinheiro é mais uma funcionalidade útil e extra que promete complementar o sistema e oferecer uma maior acessibilidade e qualidade de vida a pessoas invisuais. Este recurso é fundamental na realização de inúmeras tarefas na vida diária de um invisual como, por exemplo, verificar o valor de notas e moedas para realizar uma compra num determinado negócio. Apesar da evolução tecnológica estar a trazer novos sistemas de pagamento, como é o caso de compras digitais através do nosso *smartphone*, o dinheiro físico ainda está muito presente no nosso dia a dia. Para uma pessoa invisual, não saber qual o valor da nota ou moeda que está a tocar torna-se uma tarefa difícil, ou quase impossível. Sendo assim, a atividade identificar dinheiro, vem trazer uma maior independência e sucesso na vida diárias destas pessoas. Na Tabela 5.3 é possível visualizar todo o conhecimento necessário para uma melhor compreensão desta funcionalidade.

| **Identificar Dinheiro** | | |
|---|---|---|
| **ID do Requisito:** 3 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Utilizador Invisual. | | |
| **Pré-condições:** Modelo de rede neuronal artificial para detetar notas e moedas. | | |
| **Pós-condições:** Resultados serão apresentados se forem detetadas notas/moedas nas imagens recolhidas pelo sistema, em tempo real. Aquela que apresentar maior percentagem de confiança será divulgada. | | |
| **Descrição do requisito:** O utilizador invisual deverá poder ser informado sobre o valor da nota/moeda que foi detetada pelo sistema. | | |
| **Material de suporte:** Diagrama de atividade 5.6. | | |

Tabela 5.3: Requisito Funcional 3 - Descrição da funcionalidade Identificar Dinheiro.

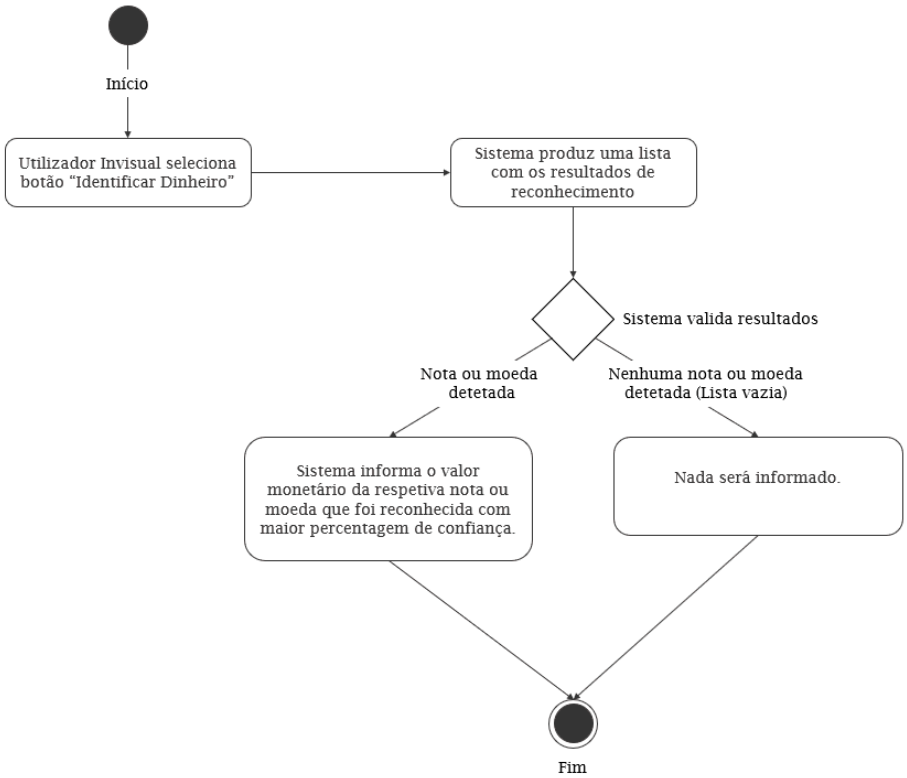



Figura 5.6: Diagrama de atividade associado à funcionalidade Identificar Dinheiro.

#### 5.5.1.4 Identificar Produtos

A última funcionalidade concedida ao utilizador invisual oferece-lhe a experiência de saber o nome de uma enorme variedade produtos que estão inseridos em diversas categorias, tais como: alimentação, saúde, higiene, etc. Este recurso vem no sentido de aumentar a sua autonomia dentro e fora de casa. Atualmente, qualquer produto disponibilizado numa loja de comércio possui um código de barras, que facilita a sua identificação e distinção entre os demais. Sendo assim, através de uma tecnologia que envolva a leitura de códigos de barras, é possível determinar o nome do produto que está a ser utilizado. A funcionalidade Identificar Produtos baseia-se exatamente numa ferramenta desse tipo. A Tabela 5.4 mostra todos os detalhes importantes que caracterizam esta funcionalidade.

| **Identificar Produtos** | | |
|---|---|---|
| **ID do Requisito:** 4 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Utilizador Invisual. | | |
| **Pré-condições:** Nada a assinalar. | | |
| **Pós-condições:** Nome do produto só será informado se o mesmo estiver incluído na lista de produtos reconhecidos pela aplicação móvel. | | |
| **Descrição do requisito:** O utilizador invisual deverá poder identificar qualquer produto que tinha sido previamente implementado no sistema. | | |
| **Material de suporte:** Diagrama de atividade 5.7. | | |

Tabela 5.4: Requisito Funcional 4 - Descrição da funcionalidade Identificar Produtos.

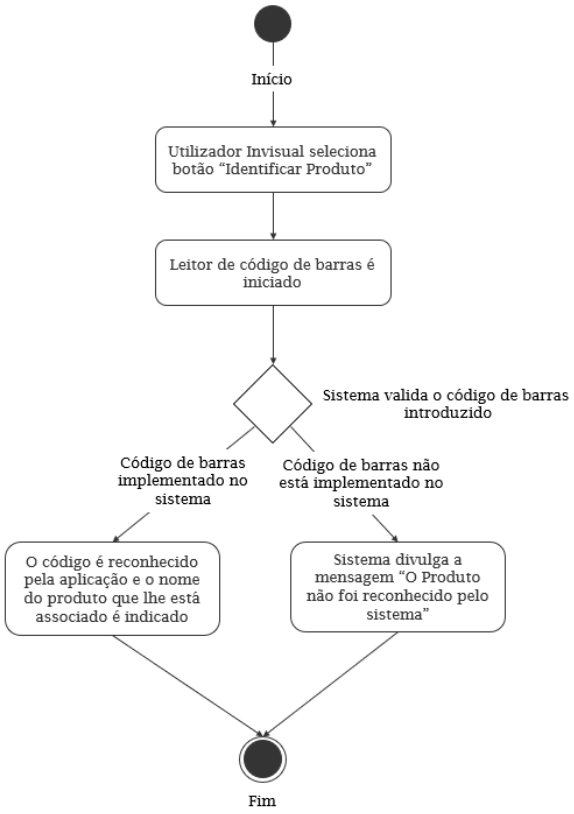



Figura 5.7: Diagrama de atividade associado à funcionalidade Identificar Produtos.

#### 5.5.1.5 Adicionar um novo objeto ao sistema

Até aqui foram abordadas todas funcionalidades que poderão ser requisitados pelo utilizador invisual. Resta abordar as duas funcionalidades que foram construídas unicamente para serem trabalhadas por um assistente visual. A primeira tem como função adicionar novos objetos ao sistema, com o principal objetivo de aumentar a sua quantidade, oferecendo assim a possibilidade de expandir o recurso de deteção e torná-lo ainda mais produtivo e eficaz. Uma vez que esta atividade requer o uso da visão, ela foi projetada de forma a ser utilizada exclusivamente por uma pessoa visual, por exemplo um familiar ou amigo do utilizador. Na Tabela 5.5 é possível observar as particularidades mais revelantes desta funcionalidade.

| **Adicionar Novo Objeto** | | |
|---|---|---|
| **ID do Requisito:** 5 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Assistente Visual. | | |
| **Pré-condições:** Acesso à *Internet*. | | |
| **Pós-condições:** Objeto será adicionado ao sistema se o assistente visual validar os dados obtidos (nome do objeto, imagens recolhidas e anotações de imagem). | | |
| **Descrição do requisito:** O Assistente Visual deverá poder adicionar qualquer objeto que deseje ao sistema, com o objetivo de este vir a ser detetado com sucesso no futuro. | | |
| **Material de suporte:** Diagrama de atividade 5.8. | | |

Tabela 5.5: Requisito Funcional 5 - Descrição da funcionalidade Adicionar Novo Objeto.

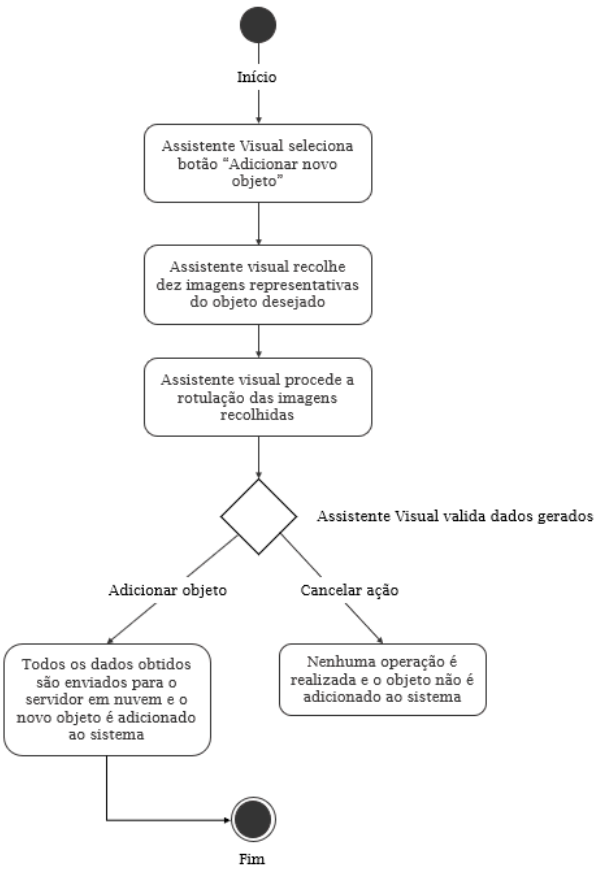



Figura 5.8: Diagrama de atividade associado à funcionalidade Adicionar Novo Objeto.

#### 5.5.1.6 Adicionar um novo produto ao sistema

A segunda e última funcionalidade que foi confiada ao assistente visual libera a oportunidade de ser adicionado ao sistema qualquer produto que possua um código de barras legível. Esta ferramenta espera trazer mais liberdade e controlo da funcionalidade Identificar Produtos, que foi desenvolvida na aplicação móvel para o utilizador invisual. Neste momento, a mesma conta com x códigos de barras disponíveis para identificação, sendo que todos eles representam produtos português começados pelos três dígitos 560. Tal como referido, independentemente da categoria ou da nacionalidade, qualquer produto pode ser incluído na aplicação. Informações adicionais sobre este requisito podem ser encontradas na Tabela 5.6.

| **Adicionar novo produto** | | |
|---|---|---|
| **ID do Requisito:** 6 | **Tipo de Requisito:** Funcional | **Data de criação:** 19/06/2022 |
| **Atores:** Assistente Visual. | | |
| **Pré-condições:** Nada a assinalar. | | |
| **Pós-condições:** Produto será adicionado ao sistema se o assistente visual validar os dados fornecidos (nome do produto e código de barras). | | |
| **Descrição do requisito:** O Assistente Visual deverá poder adicionar qualquer produto que deseje ao sistema, independentemente da categoria ou nacionalidade. O código barras do novo produto terá de ser reconhecido pela aplicação no futuro com sucesso. | | |
| **Material de suporte:** Diagrama de atividade 5.9. | | |

Tabela 5.6: Requisito Funcional 5 - Descrição da funcionalidade Adicionar novo produto.

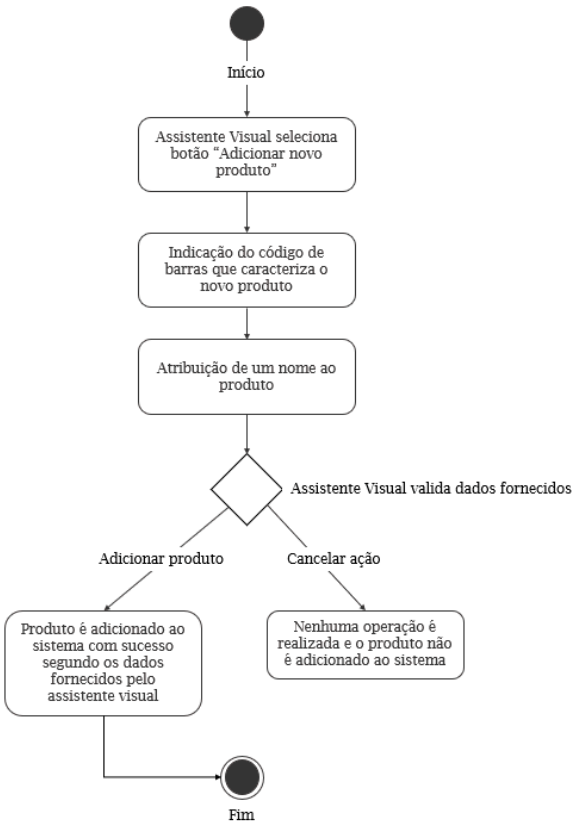



Figura 5.9: Diagrama de atividade associado à funcionalidade Adicionar Novo Produto.

### 5.5.2 Requisitos não funcionais

Ao contrário do que acontece num requisito funcional, o não funcional, tal como o próprio nome indica, não descreve uma funcionalidade do sistema, mas sim os critérios que são necessários para que a mesma seja executada de forma correta e eficiente pelo software. Portanto, qualquer requisito não funcional encontra-se relacionado com questões de capacidade, desempenho, disponibilidade, tecnologias envolvidas, confiabilidade, usabilidade, entre outras. Em vez de este tipo de requisito se focar na compreensão das atividades que o sistema deve fazer, ele preocupa-se apenas em como pode fazê-las e com que recursos, assegurando que o produto final cumpre todas as expectativas do utilizador. [113]

No presente trabalho, foram quantificados os seguintes requisitos não funcionais: Acessibilidade, Usabilidade, Confiabilidade, Disponibilidade, Desempenho, Latência, Instalação, Adaptabilidade, Capacidade, Armazenamento, Segurança e Manutenção. Nas Tabelas de 5.7 a 5.14 é possível observar, respetivamente, como se encontram organizados os requisitos não funcionais implementados.

| **Acessibilidade** | | |
|---|---|---|
| **ID do Requisito:** 7 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** O Produto final cumpre as necessidades exigidas por pessoas invisuais. | | |
| **Base lógica:** Facilidade na utilização do sistema, ou seja, o mesmo deverá atender aos previsíveis utilizadores. Uma vez que o consumidor final da aplicação será uma pessoa invisual, o software desenvolvido deverá permitir que a mesma consiga realizar as tarefas de forma autónoma, coerente e sem constrangimentos. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.7: Requisito não funcional 1 - Acessibilidade.

| **Usabilidade** | | |
|---|---|---|
| **ID do Requisito:** 8 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Informação de resultados e validação de dados. | | |
| **Base lógica:** Funcionalidades que apresentem resultados de reconhecimento (Detetar Objetos, Identificar Dinheiro, Identificar Produtos) ao utilizador invisual, devem adotar mecanismos de *Text-To-Speech*, isto é, fornecimento de informação através de voz sintetizada. Nos recursos Localizar Objeto e Identificar produto, a validação do nome do objeto a localizar e do código de barras introduzido, em caso de falha, deve ser anunciada ao utilizador com base em áudio e da forma mais cómoda possível. Para além disso, tais atividades terão de conter instruções precisas de como devem ser executadas. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.8: Requisito não funcional 2 - Usabilidade.

| **Disponibilidade** | | |
|---|---|---|
| **ID do Requisito:** 9 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Disponibilidade do produto. | | |
| **Base lógica:** O software desenvolvido e os servidores que fazem parte da aplicação móvel deverão estar disponíveis 24 horas por dia, durante todo o ano. É de notar também que apesar do novo modelo de rede neuronal não estar imediatamente disponível, o assistente visual poderá adicionar um novo objeto ou produto ao sistema sempre que o desejar, sem restrições de horário. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.9: Requisito não funcional 3 - Disponibilidade.

| **Desempenho e Latência** | | |
|---|---|---|
| **ID do Requisito:** 10 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Desempenho e tempo de resposta do produto. | | |
| **Base lógica:** Qualquer ação ou tarefa desempenhada pelo sistema deverá ser efetuada habilmente e o mais rapidamente possível, uma vez que o utilizador invisual pode interpretar um resposta tardia como uma falha da aplicação ou um pedido que não está a ser solicitado (por exemplo ao selecionar um botão e a ação demorar mais que o esperado). | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.10: Requisito não funcional 4 - Desempenho e Latência.

| **Instalação e Adaptabilidade** | | |
|---|---|---|
| **ID do Requisito:** 11 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** *Download* do software e compatibilidade de dispositivos. | | |
| **Base lógica:** No serviço de distribuição digital de aplicações, o *Download* da aplicação móvel desenvolvida deverá ser realizado de forma simples e tendo em conta os utilizadores inexperientes. O software terá também de ser compatível com todos os dispositivos móveis que apresentam o sistema operativo *Android*. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.11: Requisito não funcional 5 - Instalação e Adaptabilidade.

| **Capacidade e Armazenamento** | | |
|---|---|---|
| **ID do Requisito:** 12 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Capacidade de utilizadores e limites de armanezamento. | | |
| **Base lógica:** O sistema deverá ser capaz de processar, em simultâneo, dados (imagens e anotações de imagem) provenientes de um número ilimitado de utilizadores diferentes. Todos os dados gerados deverão ser enviados e guardados no servidor Oblivion, com capacidade de armazenamento máximo de 10 Terabyte (TB). | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.12: Requisito não funcional 6 - Capacidade e Armazenamento.

| **Segurança e Confiabilidade** | | |
|---|---|---|
| **ID do Requisito:** 13 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Confiança e segurança no produto entregue. | | |
| **Base lógica:** O software deverá garantir que os dados do utilizador estão protegidos de acessos indesejados, ou seja, que não sejam manuseados por terceiros, assegurando a total confiança do utilizador no sistema que está a usar, independentemente da funcionalidade que está a ser executada. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.13: Requisito não funcional 7 - Segurança e Confiabilidade

| **Manutenção** | | |
|---|---|---|
| **ID do Requisito:** 14 | **Tipo de Requisito:** Não Funcional | **Data de criação:** 21/06/2022 |
| **Descrição:** Melhoria e otimização do produto. | | |
| **Base lógica:** A aplicação móvel desenvolvida deverá ser submetida a manutenção sempre que necessário, permitindo incluir novos avanços e ideias, bem como correção de defeitos que não tenham sido detetados durante a realização de testes ao software disponibilizado. | | |
| **Material de suporte:** Nada a assinalar. | | |

Tabela 5.14: Requisito não funcional 8 - Manutenção.

## 5.6 Conceção da *Interface* e da Interação com o Utilizador

A criação de um conceito de interface é um processo que proporciona o desenvolvimento de competências, características e de ideias que permitam projetar um protótipo que define o produto final esperado. Todas as técnicas de prototipagem, esboços e desenhos com inter-

faces próximas do projeto real, servem para planear e demonstrar todo o trabalho a realizar. Assim sendo, o objetivo desta última secção é apresentar o protótipo que foi construído durante a fase de desenvolvimento e mostrar o resultado que foi alcançado, de modo a acompanhar a evolução do desenvolvimento da aplicação móvel.

### 5.6.1 Atividade Principal - Menu

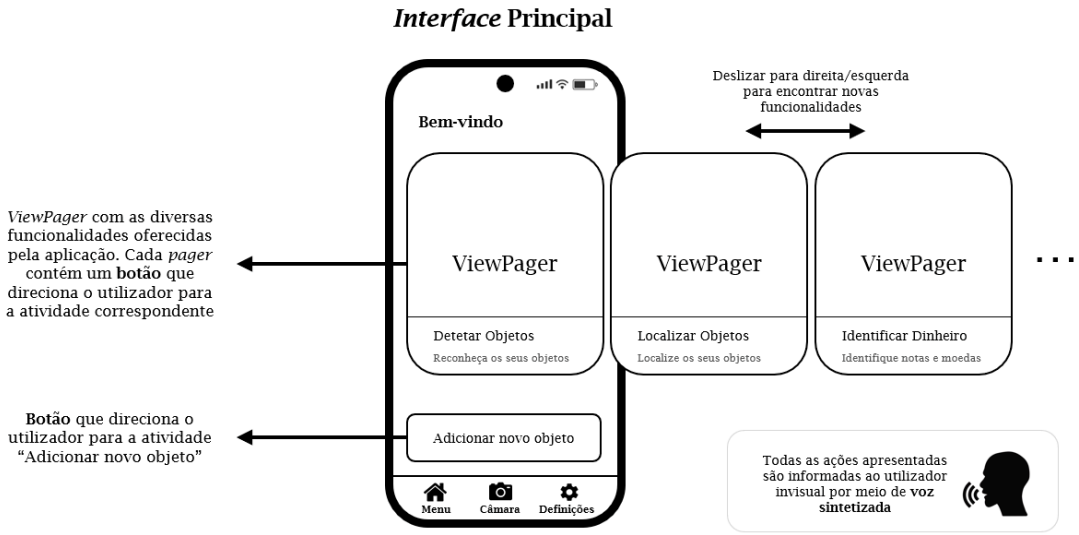



Figura 5.10: Esboço desenvolvido para a *interface* Atividade Principal - Menu.

Assim que o utilizador iniciar a aplicação móvel, a primeira *interface* que vai encontrar corresponde à atividade principal do sistema, ou seja, ao Menu. Na Figura 5.10 é possível observar um esboço que descreve todas as interações com botões e funcionalidades que estão disponíveis nesta *interface*. Pensando no consumidor final da aplicação, neste caso uma pessoa invisual, o protótipo foi desenhado de forma que o utilizador possa interagir com o sistema o mais comodamente possível. Logo, foi estabelecida a ideia de criar um *ViewPager*.

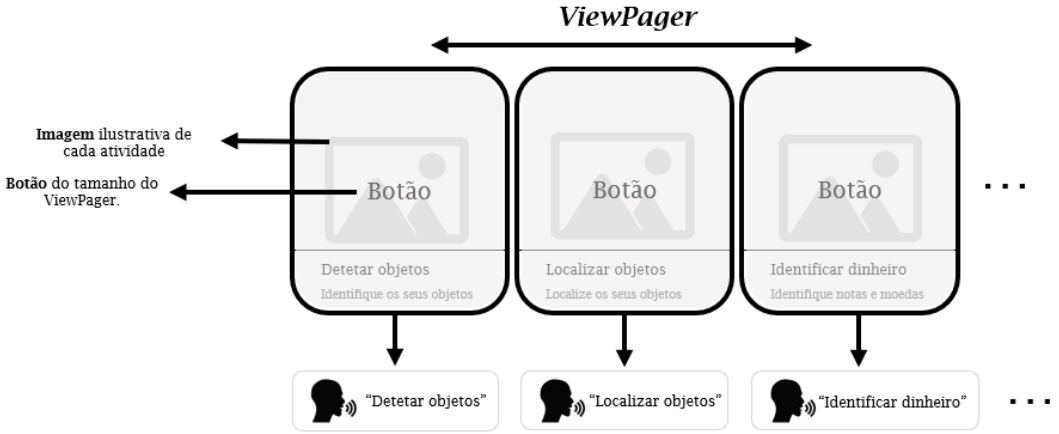



Figura 5.11: Esboço desenvolvido para o *ViewPager* presente no menu da aplicação.

Um *ViewPager* consiste em visualizações deslizáveis de tela, ou seja, transições de uma determinada página para outra. Através do mesmo, com um simples gesto, o utilizador invi-

sual poderá deslizar de botão para botão, e obter uma navegação muito mais prática e intuitiva. Na Figura 5.11, encontra-se representado um esboço que reflete todo o funcionamento do *ViewPager* projetado. Cada página, também designada por fragmento, contém o nome da respetiva funcionalidade, uma breve descrição sobre a mesma, uma imagem ilustrativa da atividade em questão e um botão que direciona o utilizador para a respetiva atividade. Para uma pessoa invisual, o mais importante numa aplicação móvel é a forma como foi implementado o acesso a botões e funcionalidades. Assim sendo, todos os botões desenvolvidos foram desenhados de forma a ocupar uma grande parte do ecrã do dispositivo móvel, de modo que o utilizador não sinta dificuldade em selecioná-los e não cause constrangimentos. Para tal, cada botão do *ViewPager* foi planeado de modo a abranger o máximo de tamanho possível dentro de cada fragmento. Por outro lado, como é possível observar na Figura 5.10, o *ViewPager* é também a peça que apresenta maior dimensão no ecrã do telemóvel.

Um outro detalhe importante na iteração entre o utilizador invisual e a aplicação móvel, é a comunicação entre ambos, ou seja, o sistema deverá dispor de um meio que permita informá-lo sobre qual foi a ação que selecionou ao clicar num botão. Para isso, a solução projetada foi indicar ao utilizador o nome da funcionalidade em voz alta, através de voz sintetizada. Sempre que ele deslizar sobre a tecnologia do ViewPager, da esquerda para direita ou vice-versa, será emitido um áudio, a partir do telemóvel que informa o nome da atividade que está a aparecer. Claro está também que cada vez que for selecionada uma funcionalidade, é lhe informado que entrou na respetiva atividade.

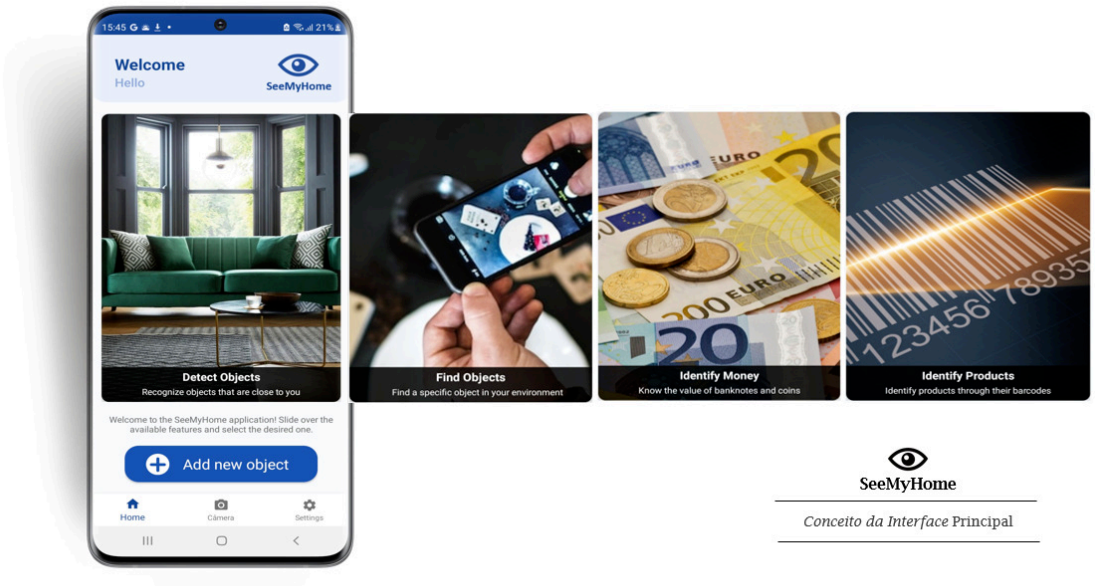



Figura 5.12: Conceito final da atividade principal do sistema.

A Figura 5.12 apresenta o conceito final da interface Detetar Objetos que foi introduzido no processo final de produção do sistema.

### 5.6.2 Detetar Objetos

A primeira funcionalidade disponibilizada pela aplicação diz respeito à ação de detetar objetos. A mesma permite que o utilizador invisual seja capaz de obter uma informação ampla sobre tudo que se encontra à sua volta, através do fornecimento de indicações sonoras. Logo, sempre que o utilizador selecionar o primeiro fragmento do *ViewPager* será imediatamente encaminhado para esta respetiva atividade Detetar Objetos. Na Figura 5.13, é possível observar um esboço que, inicialmente, foi desenhado para planear a construção da mesma. Ela encontra-se dividida em três componentes principais: um painel informativo, que indica o nome dos objetos que vão sendo reconhecidos pela aplicação, um painel de visualização que mostra o que está a ser capturado pela câmara do dispositivo móvel e, por fim, um botão que ativa um áudio informativo com o nome do objeto que foi detetado.

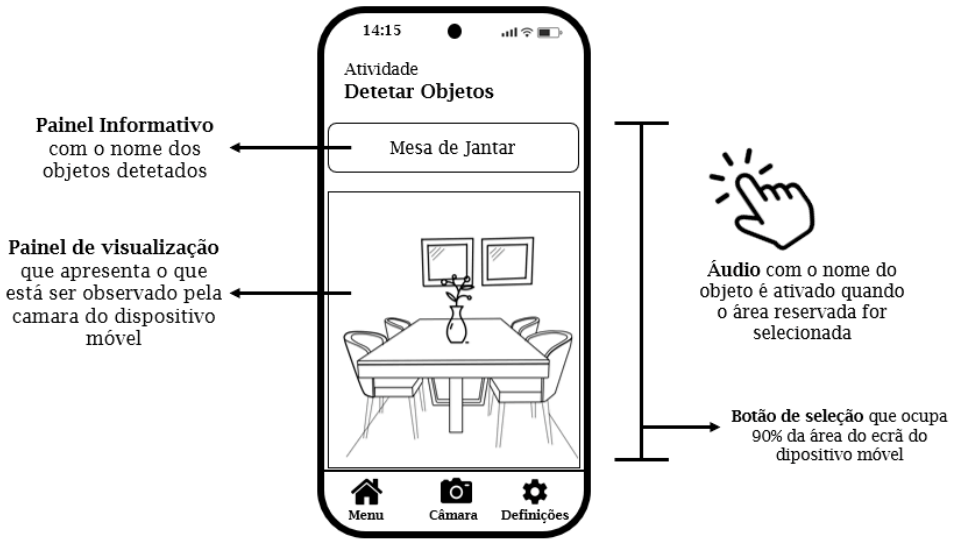



Figura 5.13: Esboço desenvolvido para a *interface* da funcionalidade Detetar Objetos.

Para o utilizador invisual, o recurso mais importante nesta funcionalidade é selecionar o botão que lhe indica, em voz sintetizada, o nome do objeto que foi reconhecido. Assim sendo, este componente obrigou a uma forte atenção e destaque no momento da sua construção na *interface* Detetar Objetos. Com o objetivo de não haver dificuldade, por parte do utilizador, em selecioná-lo, ele foi implementado de modo a ocupar a maior parte da atividade. Em termos de design, este botão foi criado com uma cor transparente, logo ele não é visto na *interface*, mas está presente em 90% da sua área. Portanto, quando o utilizador clica no centro ecrã, o mesmo será ativado e o respetivo áudio informativo será anunciado.

A Figura 5.14 apresenta o conceito final da interface Detetar Objetos. Tal como foi mencionado em cima, o botão que emite o áudio com o nome do objeto detetado não é visível na interface criada. No entanto, ele está presente e ocupa uma grande parte da área do ecrã do dispositivo móvel.

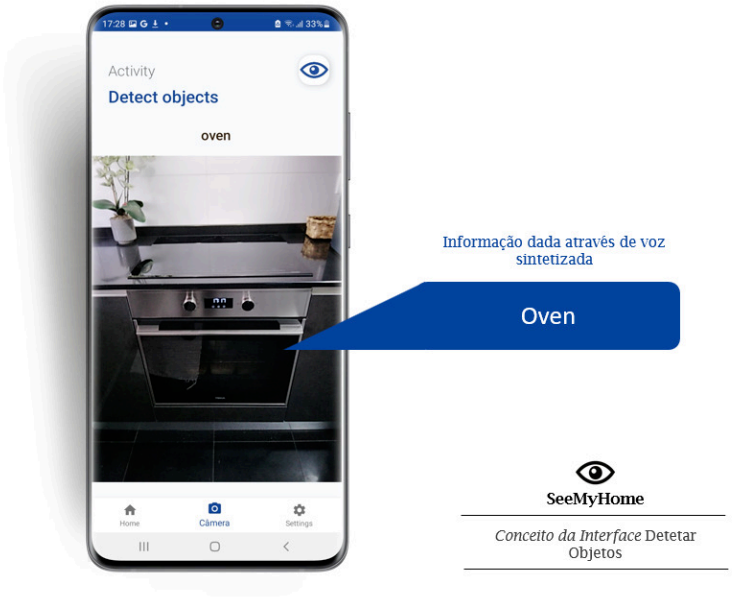



Figura 5.14: Conceito final da funcionalidade Detetar Objetos.

### 5.6.3 Localizar Objetos

No que diz respeito à segunda funcionalidade do sistema desenvolvido, Localizar Objetos, a sua *interface* foi projetada a fim de incluir instruções simples e dinâmicas que guiem o utilizador invisual durante a sua utilização, cumprindo assim o requisito não funcional 5.8.

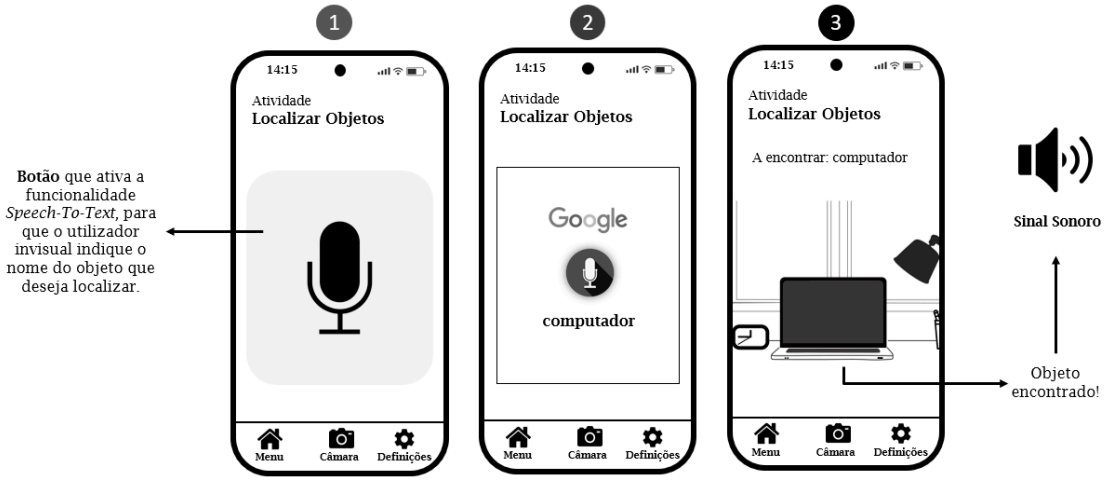



Figura 5.15: Esboço desenvolvido para a *interface* da funcionalidade Localizar Objetos.

Na Figura 5.15, encontra-se representado o esboço da interface Localizar Objetos. Como é possível observar, a mesma encontra-se dívida em três partes, sendo que a primeira corresponde à atividade de abertura desta funcionalidade. Quando o utilizador invisual selecionar este recurso no menu principal do sistema, uma mensagem de áudio será emitida com a informação de que está presente um botão no qual ele deverá selecionar para indicar o nome do objeto que deseja localizar. Em termos de dimensão e acessibilidade, este botão foi desenhado de forma a estar presente no centro do ecrã do dispositivo móvel e ser suficiente

grande, para que o utilizador não tenha dificuldades em selecioná-lo (requisito não funcional 5.7). Logo que o mesmo seja carregado, por meio da API da Google, o passo seguinte procura ativar a funcionalidade Speech-To-Text, a fim de efetuar o reconhecimento de fala (observar ponto 2 da Figura 5.15). Depois de ter sido obtido o nome do objeto e verificado se este está na lista de objetos reconhecidos pela aplicação, a mesma irá avançar para a sua localização. O esquema planeado para esta etapa final pode ser observado no ponto 3 da Figura 5.15. No momento que for detetado o objeto desejado, um sinal sonoro contínuo será emitido, com a finalidade de avisar o utilizador da sua existência.

O recurso Localizar Objetos é muito benéfico para o utilizador, pois não necessita de estar constantemente a ouvir os reconhecimentos que estão a ser obtidos pelo sistema, quando apenas deseja encontrar um objeto específico. Ele somente tem a tarefa de apontar a câmara do telemóvel sobre a zona que o rodeia, não se preocupando com o que está a ser detetado.

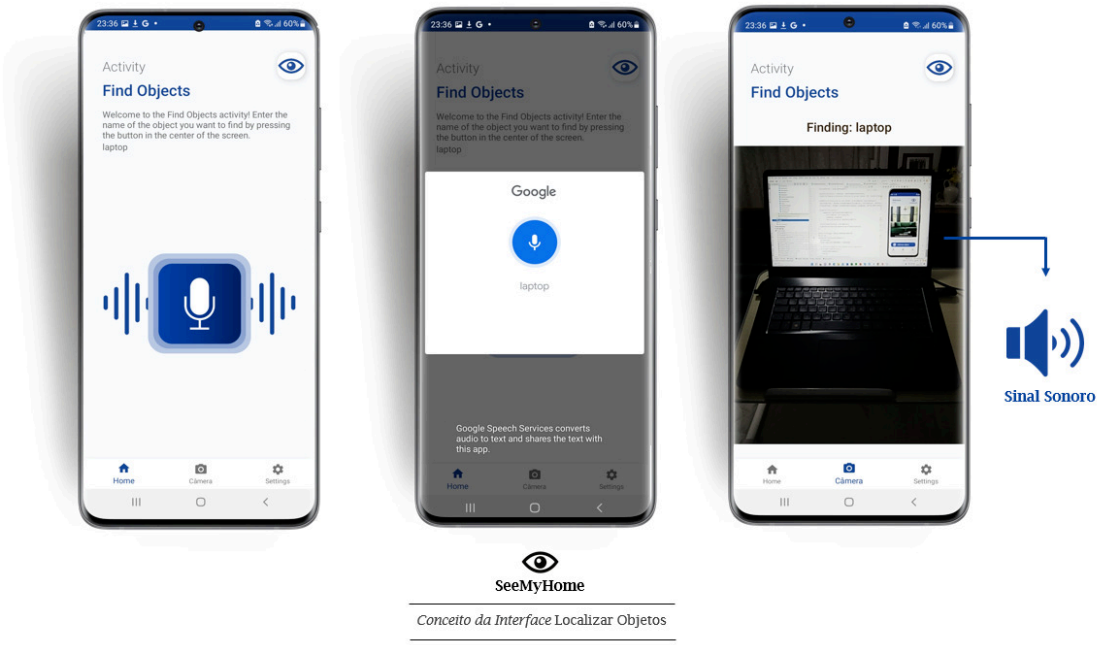



Figura 5.16: Conceito final da funcionalidade Localizar Objetos.

O resultado final de produção da interface Localizar Objetos pode ser observado na Figura 5.16. A ideia projetada foi totalmente implementada, e o recurso ficou exatamente igual ao esboço apresentado na Figura 5.15.

### 5.6.4 Identificar Dinheiro

Na aplicação móvel construída, a funcionalidade Identificar Dinheiro foi implementada de forma muito semelhante à atividade Detetar Objetos, justificando assim a parecença existente entre ambas, tanto a nível prático como a nível de desenho. Na Figura 5.17 é possível verificar como foi desenvolvido o esboço inicial para interface Identificar Dinheiro.

Para este caso, foi construído um novo modelo com o objetivo de reconhecer apenas o valor de notas e moedas. Ele foi aplicado no sistema da mesma forma que foi incluído o modelo que permite detetar objetos. Sendo assim, o mecanismo de reconhecimento mantém-se

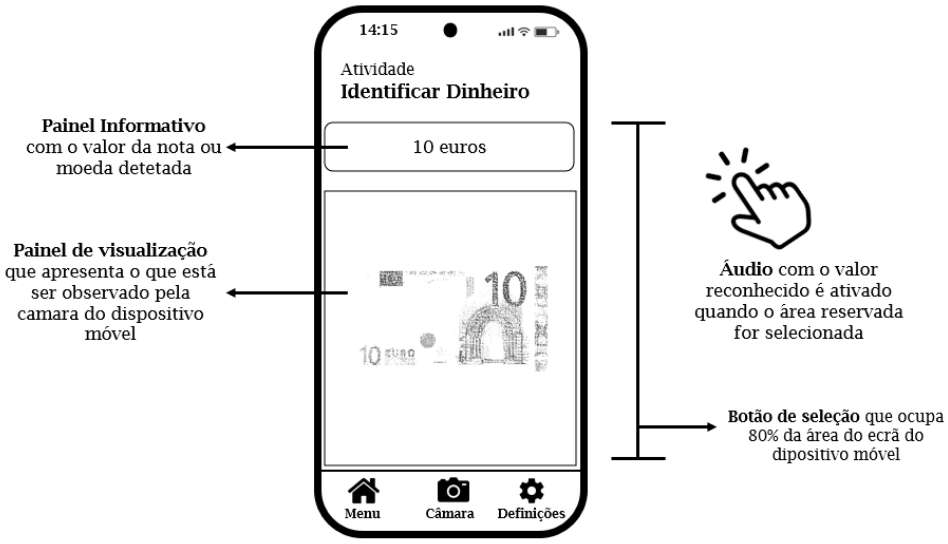



Figura 5.17: Esboço desenvolvido para a *interface* da funcionalidade Identificar Dinheiro.

igual no recurso Identificar Dinheiro. O utilizador terá apenas de pressionar o botão, que também foi programado de modo a ocupar 80% do ecrã, para saber qual o valor da nota ou moeda que está a ser detetada pelo sistema.

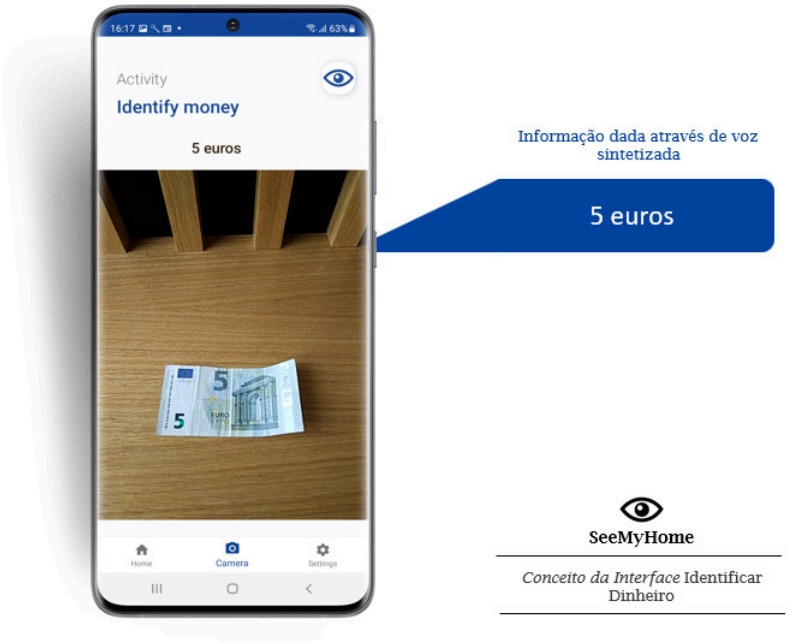



Figura 5.18: Conceito final da funcionalidade Identificar Dinheiro.

O conceito final da referente *interface* Identificar Dinheiro é mostrado na Figura 5.18.

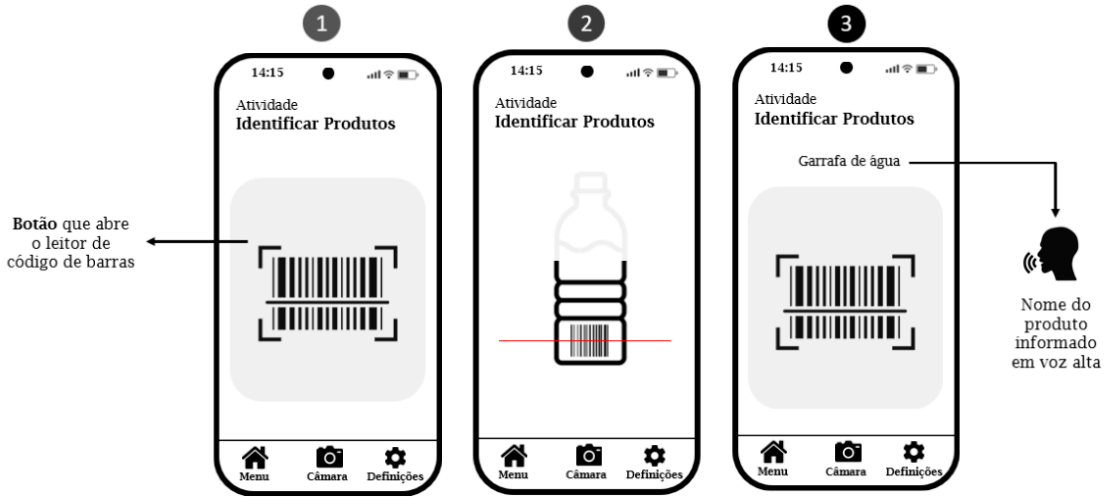



Figura 5.19: Esboço desenvolvido para a *interface* da funcionalidade Identificar Produtos.

### 5.6.5 Identificar Produtos

Por fim, o esboço divulgado na Figura 5.19 apresenta o conceito da *interface* destinada ao reconhecimento de produtos. Assim que o utilizador selecione a funcionalidade Identificar Produtos no *ViewPager* da interface principal, ele será encaminhado para uma primeira tarefa onde se encontra um botão que permite iniciar o processo de leitura de códigos de barras, a fim de alcançar o objetivo pretendido. Ao selecionar o botão, o utilizador deverá rodar o produto desejado em várias posições, até que a aplicação encontre o respetivo código de barras. Por último, após o mesmo ser reconhecido, o utilizador é enviado novamente para o ponto inicial da funcionalidade Identificar Produtos, mas já com o nome do produto determinado. O resultado será informado por meio de voz sintetizada.

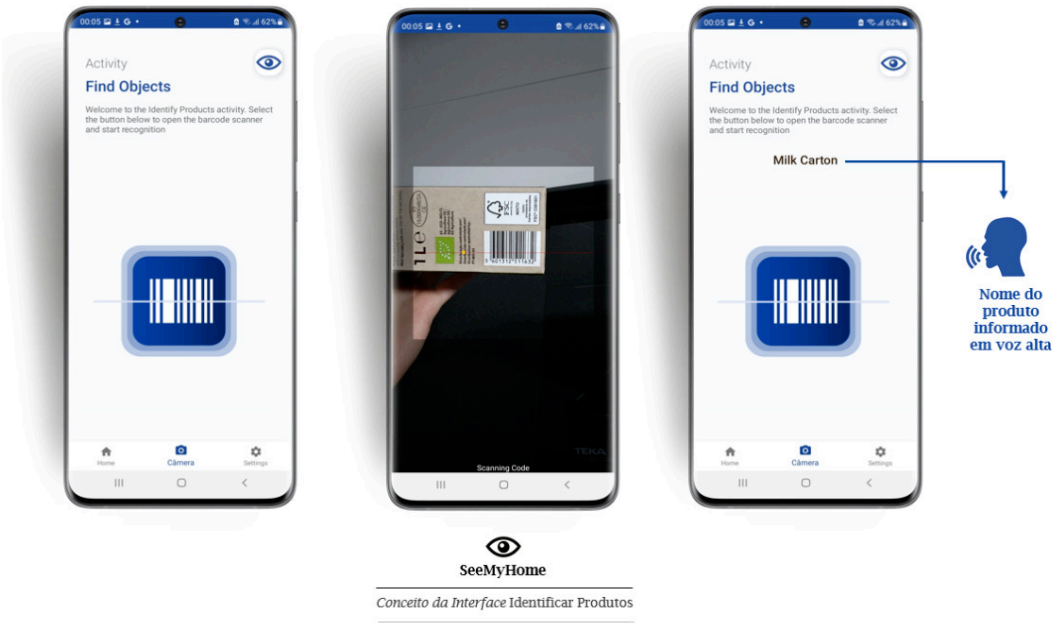



Figura 5.20: Conceito final da funcionalidade Identificar Produtos.

O conceito final da *interface* Identificar Produtos é mostrado na Figura 5.20.

### 5.6.6 Adicionar Novos Objetos

Até aqui foram abordados todos recursos implementados no sistema que poderão ser requisitados pelo utilizador invisual, porém dentro da atividade principal da aplicação está ainda presente uma funcionalidade que foi construída unicamente para ser trabalhada por um assistente visual. A mesma tem como função adicionar novos objetos ao sistema, com o principal objetivo de aumentar a sua quantidade, oferecendo assim a possibilidade de expandir o recurso de deteção e torná-lo ainda mais produtivo e eficaz. Uma vez que esta atividade requer o uso da visão, ela foi projetada de forma a ser utilizada exclusivamente por uma pessoa visual, por exemplo um familiar ou amigo do utilizador.

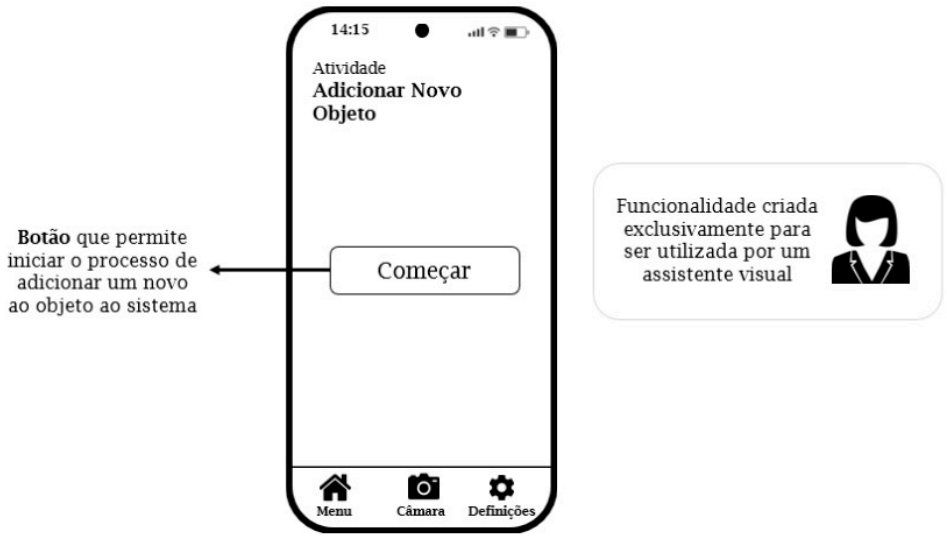



Figura 5.21: Esboço desenvolvido para a *interface* de abertura da funcionalidade Adicionar Novo Objeto.

Tal como foi mencionado na Subsecção 5.6.1 desta secção, o menu da aplicação móvel desenvolvida contém no final um botão destinado à atividade Adicionar Novo Objeto. O assistente visual ao selecioná-lo será encaminhado para a respetiva *interface* desta funcionalidade, cujo seu esboço pode ser visualizado na Figura 5.21. Ela foi desenhada de modo a incluir um botão que permite começar esta atividade. O mecanismo que faculta a adição de um novo objeto ao sistema envolve muitos passos, o que torna este recurso, em termos de estrutura, o mais complexo da aplicação móvel. O esboço apresentado representa apenas a interface de abertura desta atividade, no entanto existem muitas mais desencadeadas a partir desta e, sendo assim, comecemos por partes. Primeiro é importante saber quais são os passos necessários para a correta realização deste recurso, que são os seguintes:

1. Atribuir um nome ao objeto;
2. Recolher dez imagens que sejam representativas do objeto em questão;
3. Rotular o objeto nas imagens capturadas;

4. Enviar todos os dados obtidos para o servidor em nuvem.


Figura 5.22: Esboço desenvolvido para a *interface* de atribuição de um nome ao novo objeto.

O primeiro passo encontra-se esquematizado na Figura 5.22. Assim que o assistente visual selecione o botão começar, a interface mostrada no primeiro ponto da mesma figura será divulgada. A mesma contém um breve texto informativo sobre o funcionamento do primeiro passo a ser realizado, e um botão que permite realizar a tarefa. Este último dirige o assistente para uma nova atividade que é caracterizada inicialmente por uma lista, onde poderá selecionar um nome ao objeto que deseja incluir no sistema. Caso nenhum dos nomes apresentados seja adequado ao objeto em questão, ele poderá sugerir e atribuir um novo nome através do botão apresentado no final dessa atividade (observar ponto 2 da Figura 5.22). Para conceder um novo nome o objeto, o assistente terá de digitá-lo no campo destinado a esse efeito (observar o ponto 3 da mesma figura), e de seguida pressionar o botão com a descrição "Ok" para concluir e avançar para o passo seguinte.

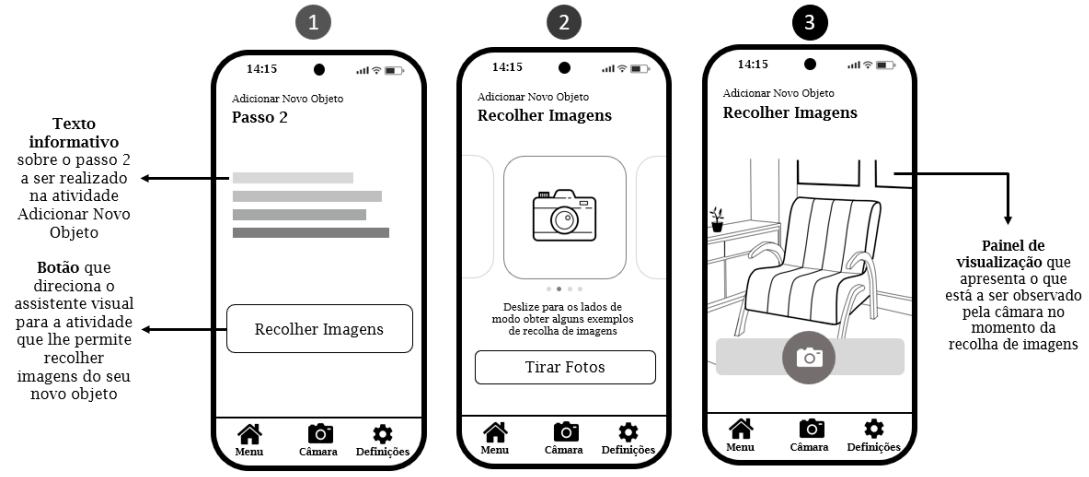



Figura 5.23: Esboço desenvolvido para a *interface* de recolha de imagens do objeto a adicionar ao sistema.

O segundo passo, cujo esboço encontra-se representado na Figura 5.23, destina-se à recolha de imagens do objeto a ser adicionado no sistema. Da mesma forma que acontece

no passo anterior, esta tarefa também começa por ter uma atividade onde é explicado todo o mecanismo necessário para a sua realização (observar ponto 1 da Figura 5.23). De seguida, ao selecionar o botão com a descrição "Recolher Imagens", uma nova atividade é desencadeada partir da primeira, com a finalidade de mostrar ao assistente visual um breve guia que o irá ajudar a saber como deve tirar fotos, isto é, em que tipo de cenários, condições de iluminação e de oclusão (imagens com objetos parcialmente ocultos). Com o objetivo de tornar claro esse guia, foi projetado um *ViewPager* nessa atividade, que apresenta alguns exemplos de imagens que devem ser recolhidas (observar ponto 2 da Figura 5.23). Por fim, para começar a tirar fotografias do seu objeto, o assistente visual deverá selecionar o botão com a descrição "Tirar Fotos" e, imediatamente, a aplicação da câmara do dispositivo móvel será iniciada (observar ponto 3 da Figura 5.23).


Figura 5.24: Esboço desenvolvido para a *interface* de anotação/rotulação de imagens.

Depois de todas as imagens terem sido devidamente reunidas, a tarefa seguinte é rotulá-las, ou seja, desenhar um retângulo ou quadrado sobre o objeto na imagem recolhida, de modo que a rede neuronal, durante o treino, saiba o de se encontra o objeto pretendendo nas imagens que foram recolhidas. Esta etapa pode ser observada em formato de esboço na Figura 5.24. Para este terceiro passo, o assistente terá apenas de mover mover os quartos pontos da figura geométrica apresentada, de forma a incluir apenas nela o objeto que se deseja adicionar ao sistema. Assim que anotação esteja terminada, deverá ser acionado o botão com a descrição "Avançar" para concluir este passo e avançar para o seguinte.

O último passo da funcionalidade Adicionar Novo Objeto traduz-se no envio de todos os dados recolhidos, imagens e anotações, para o servidor em nuvem destinado ao treino de um novo modelo de rede neuronal. No desenho da direita da Figura 5.24 encontra-se o esboço deste quarto e último passo. Ele apenas representa um simples botão que permite ao assistente visual concluir todas etapas deste recurso e efetuar o upload dos dados.

Através das três figuras seguintes, é possível ver como ficou o conceito final da atividade Adicionar Novo Objeto. A Figura 5.25 destina-se à apresentação da primeira interface que é exibida imediatamente após o assistente visual pressionar o botão corresponde no menu

da aplicação. De seguida, a Figura 5.26 mostra a imagem final do processo de recolha de imagens, isto é, as três atividades que levam a realização desse passo. Por fim, na Figura 5.27 e 5.28 revelam o resultado do mecanismo de anotação de imagem e o envio dos dados para o servidor.

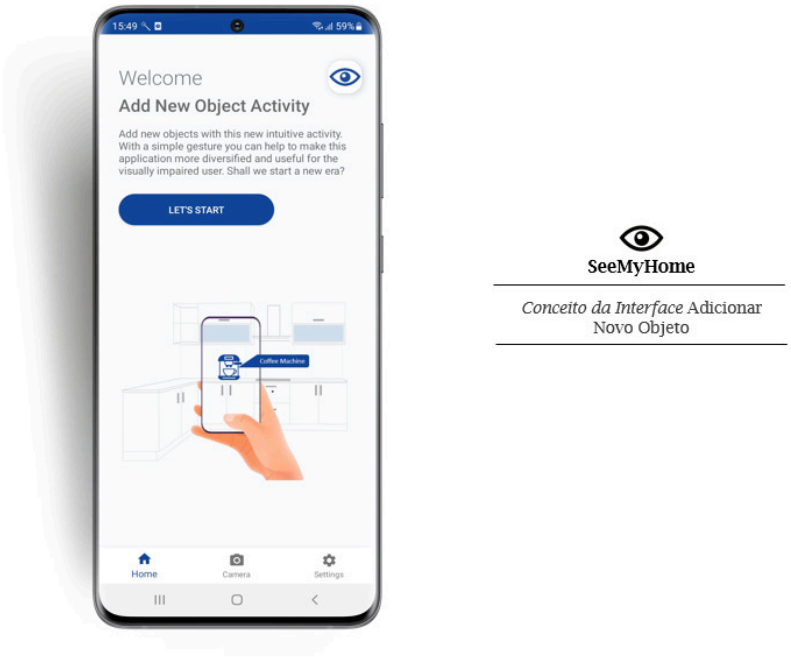



Figura 5.25: Conceito final da funcionalidade Adicionar Novo Objeto - Atividade Inicial.

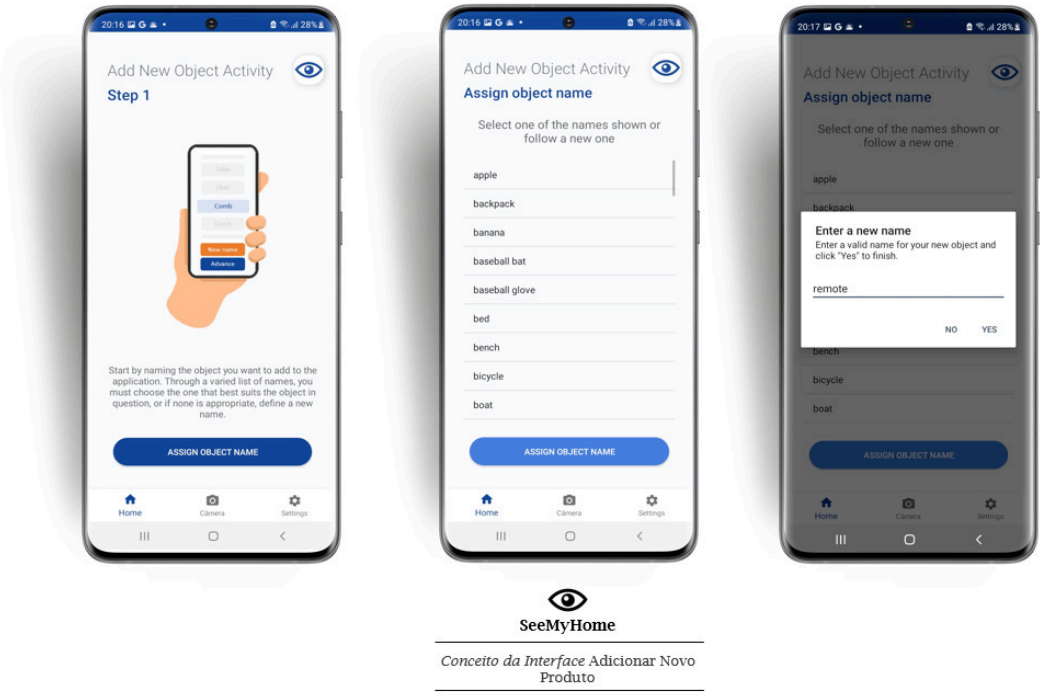



Figura 5.26: Conceito final da funcionalidade Adicionar Novo Objeto - Atribuição de um nome ao objeto.

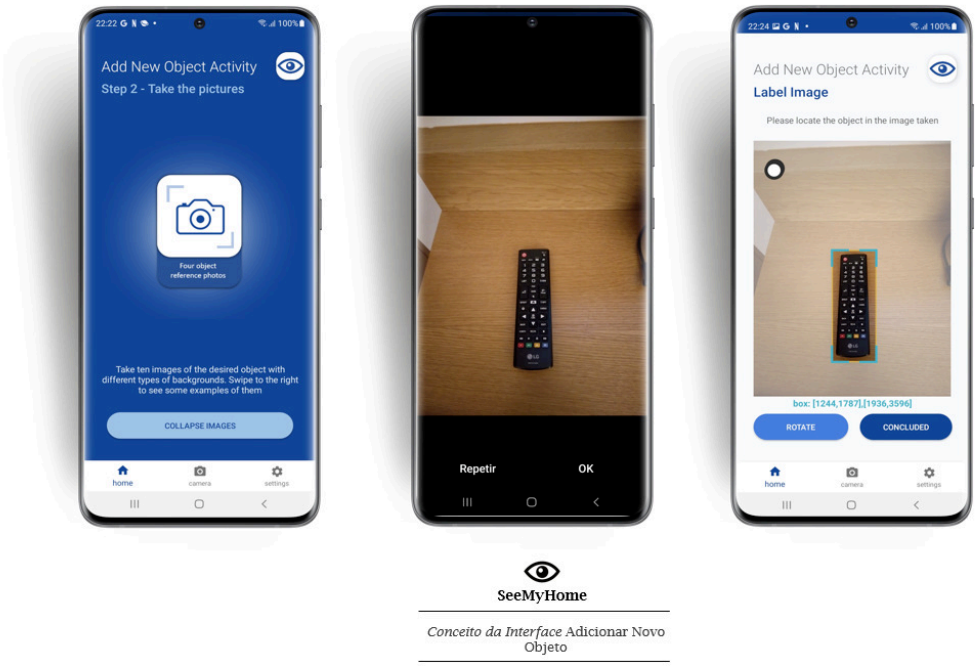



Figura 5.27: Conceito final da funcionalidade Adicionar Novo Objeto - Recolha de imagens.

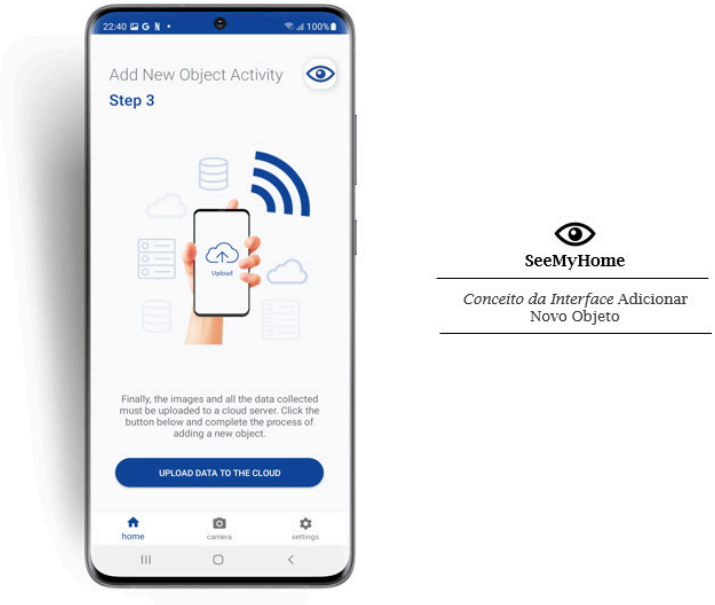



Figura 5.28: Conceito final da funcionalidade Adicionar Novo Objeto - Envio dos dados para o servidor.

Assim que o assistente visual terminar a atividade que lhe permite adicionar um novo objeto ao sistema, o passo que se segue é aguardar que o modelo de rede neuronal seja treinado com os novos dados adquiridos e que esteja disponível para *download* na aplicação. O assistente visual poderá acompanhar o surgimento de novos modelos acedendo às definições do sistema. O seu conceito foi implementado a fim de oferecer ao utilizador a oportunidade de fazer alguma alteração nas configurações do sistema como, por exemplo, alterar o idioma

da aplicação. Porém, neste momento, as definições da aplicação móvel foram programadas apenas para atualizar o software, ou seja, o modelo de deteção de objetos. Desta forma, quando o assistente visual se dirigir às definições do sistema, irá encontrar uma interface de atualização cujo esboço pode ser observado na Figura 5.29.

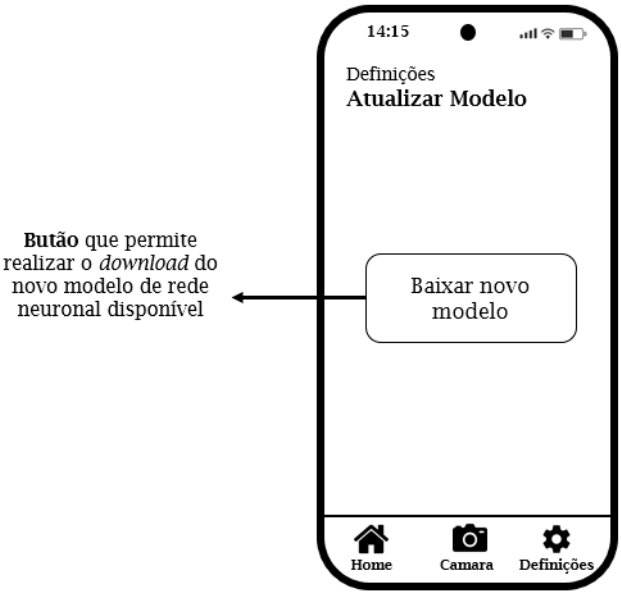



Figura 5.29: Esboço desenvolvido para a atividade Atualizar Software.

A interface destinada às definições foi desenhada de forma a incluir somente um botão que serve para carregar e atualizar o modelo da aplicação móvel. Este botão só estará presente na interface nos casos em que houver uma nova versão disponível e que esta seja diferente da que está a ser utilizada pelo sistema.

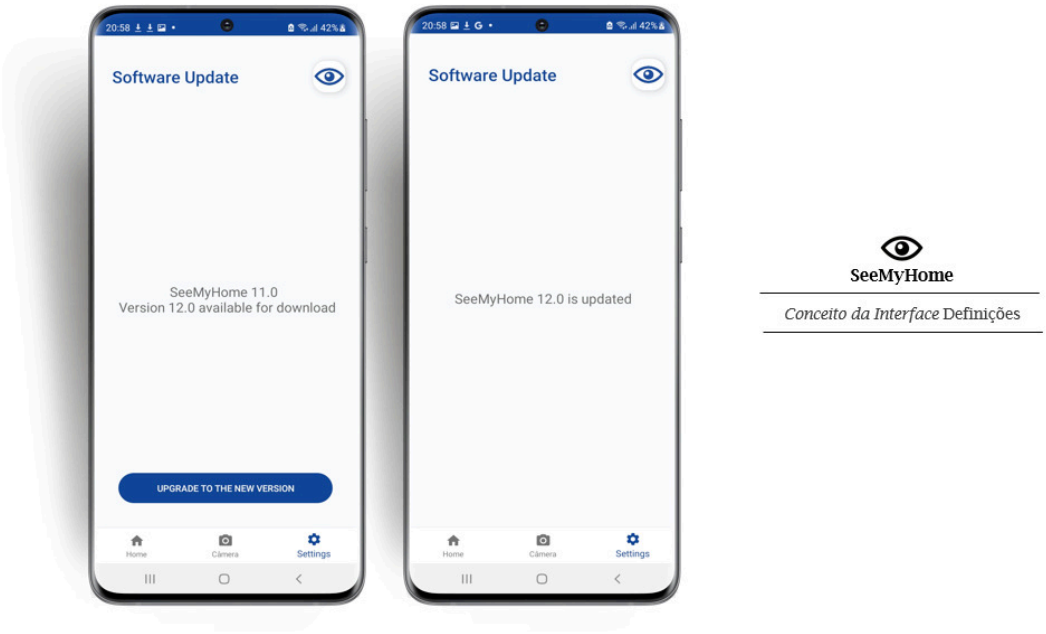



Figura 5.30: Conceito final da atividade Atualizar Software.

O conceito final da atividade Definições encontra-se representado na Figura 5.30. O

assistente visual ao selecionar o botão com a designação "Upgrade to the new version" vai fazer com que a aplicação móvel obtenha o modelo de deteção de objetos que está presente no servidor Apache, e que anteriormente foi treinado no servidor Oblivion. Este novo modelo vai substituir o atual e vai ser baixado para a respetiva pasta de *Downloads* do dispositivo móvel em uso. A atualização é rápida e imediata, e após estar concluída o assistente visual será encaminhada para uma nova interface que indica a versão atual do sistema.

### 5.6.7 Adicionar Novos Produtos

Adicionar novos produtos, é a segunda e última funcionalidade que se encontra disponível na aplicação e que foi exclusivamente criada para ser utilizada por um assistente visual. A mesma pode ser acedida no menu principal da aplicação, deslizando para a direita sobre o botão "Adicionar novo objeto". Esta ação pode ser compreendida observando a Figura 5.31.

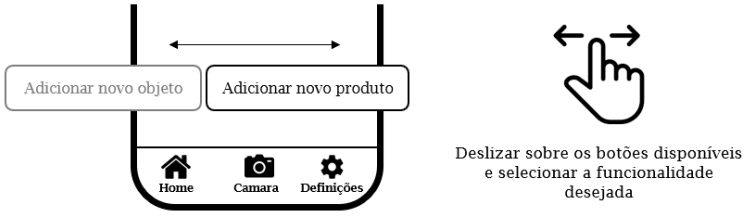



Figura 5.31: Botão "Adicionar novo produto" do Menu Principal.

O assistente visual ao selecionar o botão destinado a ação de adicionar um novo produto, será direcionado para uma atividade onde se encontra um botão que irá iniciar a leitura de um novo código de barras. O esboço da Figura 5.32 reflete isso mesmo.


Figura 5.32: Esboço desenvolvido para a atividade Adicionar Novo Produto.

A leitura do novo código barras é feita por um leitor que foi incorporado na aplicação móvel, o mesmo que é utilizado na funcionalidade Identificar Produtos. Os leitores de códigos de barras, também designados por scanners óticos, que normalmente vemos em lojas comerciais emitem um feixe luminoso de cor vermelha, percorrendo todos as barras presentes no código do produto. Na nossa aplicação, esse feixe é representado por uma linha reta, também de cor vermelha. Com a ajuda da câmara do dispositivo móvel, o assistente visual deverá apontar o código de barras do produto sobre essa linha. Logo que o leitor detete o código, a sequência de dígitos que está associado ao produto será guardado na aplicação móvel, em conjunto com um nome que será facultado pelo assistente. A indicação do nome, será a feita com base numa mensagem de diálogo que inclui uma caixa de inserção de texto. Por fim, o assistente visual poderá concluir a adição do novo produto ao sistema, selecionado o botão com a descrição "avançar" presente na mesma mensagem de diálogo. Na Figura 5.33 é possível observar todas estas ações mencionadas.

A funcionalidade Adicionar Novo Produto, incorpora a possibilidade de ser adicionado ao sistema qualquer tipo de produto que possua um código de barras. Alguns exemplos são: fármacos (caixas de medicação), bebidas e embalagens alimentares, entre outras variedades de produtos. Este recurso traduz-se num processo fundamental na vida de uma pessoa invisual, dado que oferece a oportunidade de tornar mais independentes as suas tarefas diárias.


Figura 5.33: Esboço desenvolvido para a atividade Registar Novo Produto.

O conceito final da presente funcionalidade é revelado nas Figuras 5.34 e 5.35. A primeira apresente o resultado da primeira atividade, isto é, a atividade de abertura da funcionalidade. A segunda figura mostra como ficaram implementadas as ações: leitura do novo código de barras e indicação do nome do novo produto através de uma mensagem de diálogo.

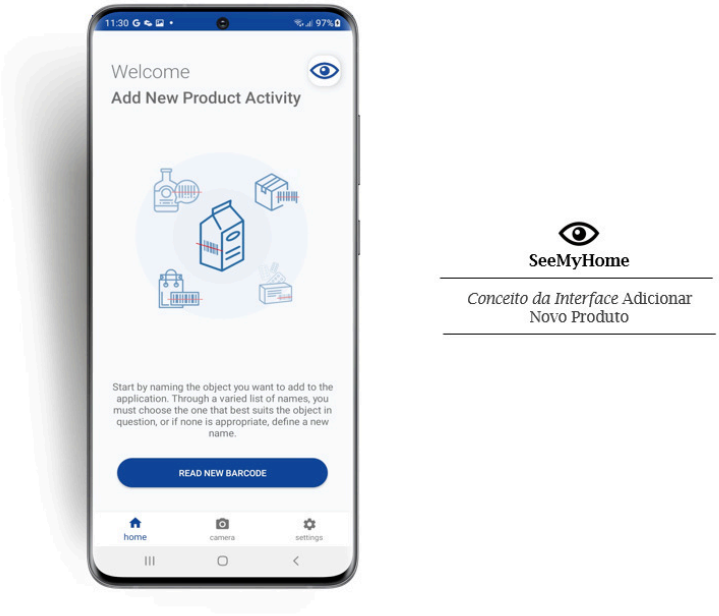



Figura 5.34: Conceito final da atividade Adicionar Novo Produto - Atividade Principal.

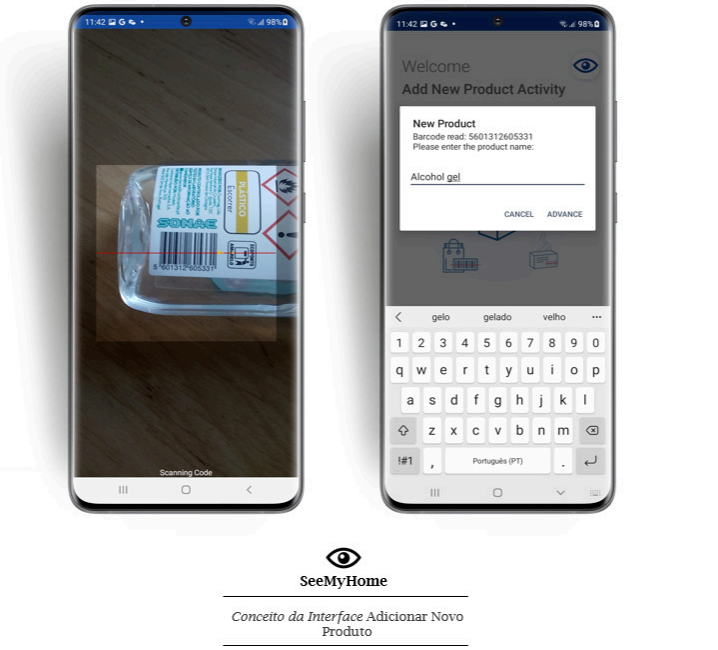



Figura 5.35: Conceito final da atividade Adicionar Novo Produto - Atividade Registar Produto.

## 5.7 Considerações finais

O presente capítulo deu a conhecer todo o trabalho que foi desenvolvido no âmbito desta dissertação. Nele foram anunciadas todas partes que reunidas traduzem todo o sistema construído. No final dele capítulo foram ainda apresentados os conceitos finais de todas as atividades introduzidas na aplicação móvel, obtendo assim visão global sobre como o sistema será disponibilizado.

# Capítulo 6

# Detalhes de Implementação

## 6.1 Introdução

Uma vez apresentado todo o trabalho que foi desenvolvido, resta mencionar como é que o mesmo foi implementado em termos práticos. Sendo assim, o presente capítulo retrata alguns detalhes mais técnicos, importantes para uma compreensão mais profunda sobre todo o sistema. Será explicado como é que o modelo de rede neuronal construído pelos vários utilizadores da aplicação móvel está implementado no servidor Oblivion e como é que este reage com a entrada de novos dados referentes a novos objetos. Para além disso, é feito também um relato geral sobre a implementação das restantes funcionalidades oferecidas pela aplicação móvel: Localizar Objetos, Identificar Dinheiro e, por fim, Identificar Produtos.

## 6.2 Deteção de objetos

Nesta secção será analisada a implementação da funcionalidade Detetar Objetos a um nível mais detalhado. Veremos como foi treinado o modelo de rede neuronal que é utilizado no sistema para reconhecer os novos objetos introduzidos pelos vários utilizadores da aplicação móvel. Pretende-se ainda mostrar como foi realizado o processo que permite enviar novos dados para o servidor e fazer o *Download* do novo modelo na aplicação.

### 6.2.1 Envio de novos dados para o Servidor Apache

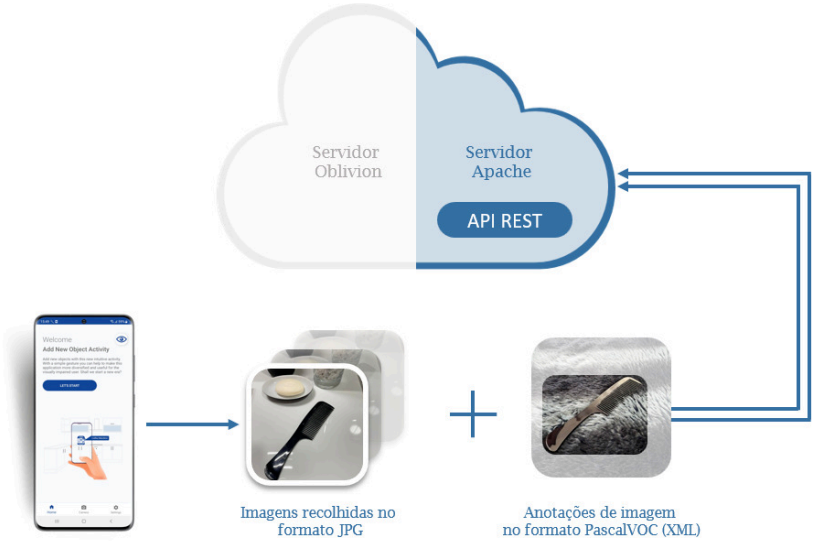



Figura 6.1: Envio de novos dados para o Servidor Apache.

A deteção de objetos é um recurso que requer muita atenção e esforço durante a sua implementação, a fim de ser alcançados bons resultados de reconhecimento. Na nossa aplicação, a adição de novos objetos ao sistema foi estudada com base num modelo de rede neuronal que vai evoluindo à medida que o conjunto de dados cresce. No descrição da solução proposta apresentada no Capítulo 4, vimos que este processo exige alguns passos técnicos importantes. A primeira etapa traduz-se na recolha de imagens de um novo objeto que será incluído no modelo de deteção. Ao longo desta dissertação, vimos que as imagens são obtidas pela aplicação móvel do utilizador e, que de seguida, serão enviadas para o servidor Apache. A Figura 6.1, ilustra esta tarefa com algum detalhe.

Na funcionalidade Adicionar Novo Objeto, o assistente recolhe as imagens e ao mesmo tempo procede à anotação das mesmas. As imagens são obtidas no formato Joint Photographic Experts Group (JPG) e os ficheiros XML seguem as métricas do formato PascalVoc. Através da aplicação móvel, os dados deverão ser enviados para o Servidor Apache. Para realizar esta ação, no Android Studio utilizou-se a classe ***MultipartUploadRequest()*** [114]. A partir da mesma é possível fazer o upload de ficheiros para um determinado servidor. Esta é maneira mais comum de realizar uma solicitação HTTP/Multipart em Java.

```
...

private static final String UPLOAD_URL = ``http://193.136.67.237:81/Api.php";

...

private void uploadMultipartPhoto(String pathImage) {

   try {
       String uploadId = UUID.randomUUID().toString();
       new MultipartUploadRequest(getApplicationContext(),uploadId,UPLOAD_URL)
               .addFileToUpload(pathImage,"image")
               .setMaxRetries(2)
               .startUpload();
   } catch (FileNotFoundException | MalformedURLException e) {
       e.printStackTrace();
   }
}

...
```

Excerto de Código 6.1: *Upload* de imagens para o Servidor Apache, a partir da Aplicação Móvel

O Excerto de Código 6.1 expõem o código que foi escrito para implementar a classe *MultipartUploadRequest()* no projeto *Android*. Um dos parâmetros de entrada desta classe é a variável estática "UPLOAD_URL", que guarda o URL do *script* associado à API REST que se encontra no servidor Apache. Na solicitação do *upload*, o caminho da imagem recolhida pelo assistente visual, é adicionado no método ***addFileToUpload()***. O envido dos dados, neste caso uma imagem, é iniciado chamando o método ***startUpload()***. Para enviar um ficheiro XML, o processo é exatamente o mesmo que o apresentado, só que vez de enviar o

caminho da imagem é enviado o caminho do ficheiro XML que foi construído pela aplicação. Para além disso, o segundo parâmetro do método ***addFileToUpload()*** é alterado consoante a natureza dos dados. Esse parâmetro ajuda a definir o tipo de conteúdo que é enviado na solicitação, ou seja, se for uma imagem é indicada uma *string* com a descrição "image", por outro lado, se for um ficheiro XML, uma *string* com a descrição "xml".

No servidor Apache foi programada uma API REST que vai tratar dos dados recebidos cada vez que forem feitas solicitações HTTP *multipart request*. A API foi construida a partir da linguagem PHP e contém toda informação necessária para que os dados sejam recebidos com sucesso e anexados nas respetivas pastas presentes no servidor que se encontra na UBI. No Excerto de Código 6.2 são mencionados alguns detalhes importantes sobre esta API.

```
<?php

 ...

 $upload_path = "images/";
 $server_ip = gethostbyname(gethostname());
 $upload_url = 'http://'.$server_ip.'/'.$upload_path;

 if($_SERVER['REQUEST_METHOD']=='POST'){

     ...

     $file_path = $upload_path . getFileName() . '.' . $extension;
     $img_name = getFileName() . '.' . $extension;

     ...

     move_uploaded_file($_FILES['image']['tmp_name'],$file_path);
     $sql = "INSERT INTO images (photo_name, photo_url,caption) ";
     $sql .= "VALUES ('{$img_name}','{$file_url}','{$caption}')";

     ...
 }
 function getFileName(){
     global $connection;
     $sql = "SELECT max(id) as id FROM images";
     $result = mysqli_fetch_array(mysqli_query($connection, $sql));
     if($result['id']==null)
      return 1;
     else
      return ++$result['id'];
     mysqli_close($connection);
 }

?>
```

Excerto de Código 6.2: API criada para a receção de imagens.

No início do excerto, é possível observar como foi formulado o local onde as imagens, neste caso, serão armazenadas. Mais à frente, na primeira condição apresentada é analisado o tipo de método que está a ser aplicado, ou seja, se estamos perante um tipo de solicitação POST. Caso essa condição se verifique, significa que a intenção é receber dados provenientes da aplicação móvel. Para tal, primeiro são declaradas duas variáveis, ”upload_url” e ”img_name”, que contêm, respetivamente, o caminho onde a imagem será armazenada e o nome da mesma. De seguida, por meio da função ***move_uploaded_file()***, a imagem recebida é guardada na pasta indicada pela variável ”upload_url”. O caminho da imagem e o seu nome, são depois anexados numa tabela, através da instrução SQL INSERT. Na Subsecção 5.3.2 do Capítulo 5 foi referido que foram construídas tabelas que servem de apoio na organização dos dados. Uma das tabelas criadas contém a informação relativa ao *upload* de imagens no servidor Apache. Os conteúdos das variáveis mencionadas são guardados nas respetivas colunas dessa tabela.

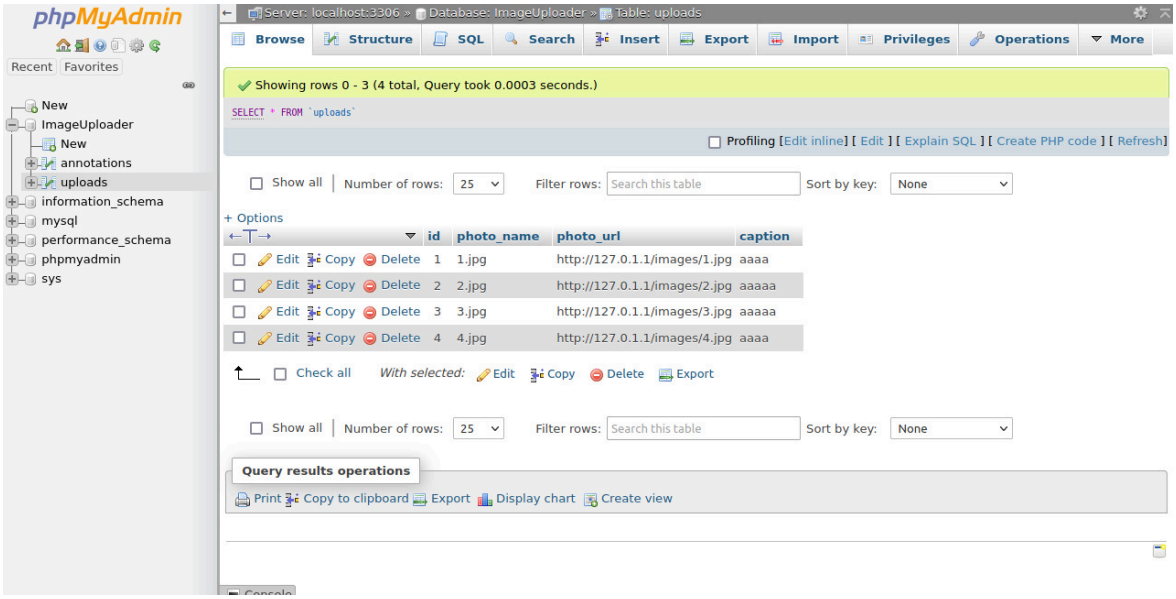



Figura 6.2: Exemplo da tabela ”imagens” na ferramenta phpMyAdmin.

A imagem da Figura 6.2 mostra um exemplo de uma tabela com dados de quatro imagens que foram recebidas pelo servidor Apache. Observando a coluna ”photo_name” vemos que os nomes das imagens respeitam uma ordem numérica crescente. Esta numeração sequencial é fundamental para uma boa gestão na receção de dados provenientes de diferentes utilizadores. A ideia é que, independentemente da quantidade de objetos adicionados e do número de utilizadores, a aplicação esteja preparada para receber muitas imagens. Posto isto, a cada imagem que chegar ao servidor é lhe associada um ID, que irá definir o seu nome. A função que calcula o valor do ID e atribui o nome à imagem pode ser observa no final do Excerto de Código 6.2, apresentado anteriormente. O bloco de instruções que se encontra dentro da função ***getFileName()*** tem como objetivo determinar o valor do ID máximo da tabela, para que uma nova imagem seja nomeada com o número que vem a seguir. Para esse efeito, é utilizada a instrução SQL SELECT, que vai procurar na tabela ”images” o maior ID que existir. Caso o resultado da pesquisa for um identificador nulo, significa que ainda

não temos imagens no servidor e, sendo assim, a primeira a chegar vai ter a atribuição do número/ID com um valor numérico igual a um. Por outro lado, se o identificador não for nulo, o seu valor máximo é incrementado uma unidade e a nova imagem é definida de acordo com o resultado dessa operação. Por fim, é importante mencionar que todo este processo se repete para os XMLs. Da mesma forma que a imagens são armazenadas, o mesmo acontece com os ficheiros XML. A única diferença é que eles são guardados numa pasta diferente e os dados numa tabela diferente. Alias, voltando a observar a Figura 6.2, no phpMyAdim, está presente uma tabela com a designação "annotations" que diz respeito aos ficheiros XML.

### 6.2.2 Treino de um novo modelo de rede neuronal no Servidor Oblivion


Figura 6.3: Envio de dados entre o Servidor Apache e o Servidor Oblivion.

A próxima fase traduz-se no envio dos dados que estão no Servidor Apache para o Oblivion. As novas imagens e anotações dos novos objetos irão fazer parte do conjunto de dados do sistema de deteção. Primeiro é importante referir que no Servidor Apache, foi criado um *script* que contém um ciclo *While* infinito. Dentro do mesmo, é averiguada a quantidade de novas imagens que chegaram ao servidor. Neste projeto, foi estabelecido o limite máximo de 100 imagens, até que seja iniciado um novo treino. Numa fase inicial de disponibilização do sistema, é importante que o limite não seja muito alto de modo a que o utilizador não tenha que esperar muito tempo para que o modelo seja treinado. Portanto, cada vez que forem enviadas 100 imagens e 100 ficheiros XML para o Servidor Apache, as pastas que contém os dois tipos de dados são zipadas e enviadas para o Servidor Oblivion, que contém poder computacional para iniciar o treino. O ciclo *While* foi programado de forma a ser infinito para que este processo esteja sempre a ser analisado e repetido. O esquema apresentado na Figura 6.3, reflete a ligação existente entre estes dois servidores.

Para que os dados sejam enviados de um servidor para outro, foi necessário estabelecer um acesso via SSH, entre ambos. O SSH, é um protocolo de comunicação de rede que permite que dois computadores se comuniquem e compartilhem dados entre si. Uma das características mais notáveis no SSH, é a sua forte criptografia que faz com que a informação seja intangível, exceto para os destinatários, oferecendo assim um acesso seguro e confiável.

```
sshpass -p "*********" scp /var/www/html/images.zip joaobranco@mn01.oblivion.
    uevora.pt:/mnt/beegfs/projects/cpca21a006/joaobranco/customTF2/data/images.
    zip

sshpass -p "*********" scp /var/www/html/annotations.zip joaobranco@mn01.oblivion
    .uevora.pt:/mnt/beegfs/projects/cpca21a006/joaobranco/customTF2/data/
    anotations.zip
```

Excerto de Código 6.3: Comandos de envio das pastas no formato ZIP para o Servidor Oblivion

O Excerto de Código 6.3 revela como foram construídos os dois comandos SSH que são utilizados para enviar as pastas com as imagens e as anotações para o respetivo servidor.

No Servidor Oblivion, está presente um *script* Python que verifica a chegada das pastas no formato Compressed File (ZIP). A ideia é que o treino de um novo modelo só aconteça quando surgirem novos dados. Para isso, o *script* criado contém uma condição que vai analisar o aparecimento das pastas "images" e "annotations" no servidor. Caso estejam presentes, o treino de uma nova rede neuronal é iniciado com o mais recente conjunto de dados. É de notar que, após começar o treino, as pastas zipadas são eliminadas do servidor para que possam ser detetadas outras mais à frente. No final do treino, o modelo é convertido para o formato TFLite, a fim de ser adicionado no sistema. Entretanto, para isso ser possível, tem primeiro de ser enviado para o Servidor Apache, uma vez que é este que comunica diretamente com a aplicação móvel. Para realizar esta ação é executado o comando Linux apresentado no Excerto de Código 6.4, que vai buscar o modelo "detect.tflite" ao servidor Oblivion.

```
scp joaobranco@mn01.oblivion.uevora.pt:/mnt/beegfs/projects/cpca21a006/joaobranco
    /customTF2/data/tflite/tflite_with_metadata/detect.tflite /home/joaobranco/
    Downloads/ModelTflite/detect.tflite
```

Excerto de Código 6.4: Comando que vai buscar o modelo TFLite ao Servidor Oblivion

### 6.2.3 *Download* do novo modelo TFLite

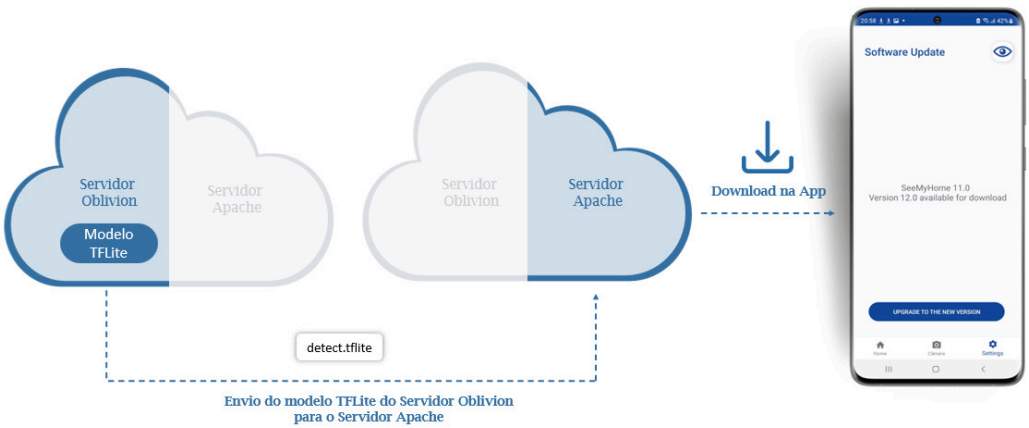



Figura 6.4: *Download* do novo modelo TFLite na Aplicação Móvel.

Assim que o ficheiro "detect.tflite" for movido para o Servidor Apache, o passo final consiste em baixá-lo na a aplicação móvel do utilizador, a fim de ser utilizado no recurso de deteção de objetos. A imagem ilustrativa da Figura 6.4, concede um esquema que retrata todo este processo. Uma curiosidade muito interessante: como é a aplicação móvel sabe que existe um novo modelo para *download*? Aliás, ela tem de verificar que esse novo modelo é diferente daquele que está a ser utilizado atualmente, só aí é que faz sentido atualizar para uma nova versão. O segredo está por trás de um ficheiro de texto que foi implementado no servidor Apache. Esse ficheiro contém informação sobre o número da última e mais recente versão que foi treinada no servidor Oblivion. O seguinte Excerto de Código 6.5, apresenta um exemplo do ficheiro que foi criado no servidor.

```
{
  ``AppName'':``SeeMyHome'',
  ``Version'':4,
  ...
}
```

Excerto de Código 6.5: Exemplo do ficheiro de texto "Version.txt".

Cada vez que for recebido um modelo TFLite no Servidor Apache, o mesmo vê isso como uma nova versão. Então quando isso se verificar, o número da versão que se encontra no ficheiro de texto é incrementado uma unidade. Para o exemplo apresentado, a mesma seria alterada de quatro para cinco. Passando para a aplicação móvel, quando o utilizador a baixa é gerado também um ficheiro de texto que vai acompanhar o sistema durante o ciclo de vida. Esse ficheiro contém o valor de uma variável que guarda a versão atual que o sistema está a utilizar. Quando esse valor for diferente do número que está ficheiro do Servidor Apache, então estamos perante uma nova versão de atualização. Por exemplo, se modelo da aplicação estiver na terceira versão e o que está no servidor na quarta, a interface projetada para a atualização do *software* indica que o sistema pode ser atualizado para quarta versão. Para poder comparar os dois valores, é necessário obter o número da versão que está no ficheiro de texto localizado no Servidor Apache. Para isso foi utilizado o método **HTTP GET**. Este método permite fazer uma solicitação para que sejam capturados dados que estejam no servidor. No cabeçalho da requisição GET, é passado um parâmetro que vai guardar o número da versão que está no servidor. No Excerto de Código 6.6, é possível observar como foi realizada a requisição no Android Studio.

```
    @GET(``version.txt'')
    Call<org.tensorflow.lite.examples.detection.AppFileData> getLatesAppInfo( ...
        , @Query(``Version'') String strExtraVersion);
```

Excerto de Código 6.6: Método GET aplicado na Aplicação Móvel

Por último, o *download* na aplicação é efetuado com a ajuda do serviço **DownloadManager**, disponibilizado pelo Android Studio. O DownloadManager foi desenvolvido com o objetivo de lidar com ***downloads* HTTP** [115]. O Cliente, que neste caso é a aplicação móvel, efetua uma solicitação que lhe permite baixar o ficheiro "detect.tflite" que está no Servidor Apache. No Excerto de Código divulgado em 6.7, são exibidos alguns detalhes do código escrito em Java no Android Studio e, que ajudam a compreender como foi feito este

pedido.

```
...
DownloadManager.Request request = new DownloadManager.Request(Uri.parse(
    strUrl));

...

request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS, ''
    detect.tflite");

...

DownloadManager manager = (DownloadManager) _context.getSystemService(Context
    .DOWNLOAD_SERVICE);
manager.enqueue(request);

strUrl = String.format(''http://%s:%d/%s", ''193.136.67.237",81, ''detect.
    tflite");

final Intent pIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(strUrl));
pIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
this._context.startActivity(pIntent);
...
```

Excerto de Código 6.7: *Download* do ficheiro "detect.tflite" na Aplicação Móvel.

O Download do modelo é realizado através do respetivo URL, que inclui o domínio e a porta do servidor Apache e o nome do ficheiro TFLITE, obtendo deste modo o caminho que a aplicação deve seguir para ir a buscar o ficheiro ao servidor. O mesmo será guardado na pasta de transferências do respetivo dispositivo móvel com o mesmo nome "detect.tflite".

### 6.2.4 Reconhecimento de objetos na Aplicação Móvel

Na aplicação móvel, o reconhecimento de objetos é feito com base em modelos que foram convertidos para o TensorFlow Lite da API de deteção de objetos do TensorFlow. O nosso modelo personalizado pelos vários utilizadores da aplicação móvel, numa fase inicial, como é de prever, vai começar por detetar um conjunto de classes de objetos reduzido. A evolução do modelo e da quantidade de objetos reconhecidos por ele, vai depender do uso que os vários utilizadores lhe deram, ou seja, da quantidade de imagens de objetos que eles foram adicionando ao sistema. Posto isto, decidiu-se juntar o modelo COCO SSD MobileNet v2 ao nosso recurso de deteção de objetos. Ele é capaz de detetar noventa e uma classes diferentes de objetos, permitindo assim reforçar o nosso modelo de deteção. Assim sendo, termos dois modelos a trabalhar e fornecer resultados na aplicação móvel, sendo que COCO SSD MobileNet v1 vai servir de entrada na funcionalidade Detetar Objetos, e o modelo personalizado é usado para adicionar novos objetos ao sistema. Dentro dos resultados fornecidos por ambos os modelos, aquele que tiver maior pontuação de confiança é apresentado ao utilizador.

## 6.3 Localizar Objetos

A funcionalidade Localizar Objetos foi construída de forma simples. Ela aproveita o recurso de deteção de objetos para pesquisar o objeto que o utilizador indicou com o apoio da tecnologia *Speech-To-Text*. Na lista de resultados de reconhecimentos, a atividade Localizar Objetos procura o nome que lhe foi informado. Para esse fim, foi projetada uma condição que verifica se o mesmo está ou não presente na lista. Numa situação de sucesso, um sinal sonoro será emitido e a funcionalidade termina a sua missão, caso contrário, a procura continua. Para ativar um som continuo assim que for reconhecido o objeto desejado na imagem de entrada, foi usado serviço **ToneGenerator**, também disponibilizado pelo Android Studio. ToneGenerator fornece métodos que produzem tons Dual-Tone Multi-Frequency (DTMF), que são sons de frequências geradas quando pressionamos os dígitos de um teclado do telefone [116]. O Excerto de Código 6.8, mostra como foi implementado o serviço ToneGenerator no Android Studio.

```
...

final List<Detector.Recognition> results = detector.recognizeImage(
    croppedBitmap);

for(int i = 0; i<results.size(); i++){

  if(results.get(i).getTitle().equals(object)){

    ToneGenerator toneGen = new ToneGenerator(AudioManager.STREAM_MUSIC, 200)
        ;
    toneGen.startTone(ToneGenerator.TONE_CDMA_PIP,50);
    break;
  }
}

...
```

Excerto de Código 6.8: Serviço ToneGenerator implementado no Android Studio

Por fim, é importante referir que o objeto indicado pelo utilizador invisual só será procurado se o mesmo estiver implementado no sistema de deteção de objetos. Não faz sentido o utilizador estar à espera que um determinado objeto seja encontrado, se a aplicação móvel não é capaz de o identificar. Sendo assim, antes de iniciar o processo de localização, o nome do objeto é validado primeiro pelo sistema.

## 6.4 Identificar Dinheiro

A funcionalidade Identificar Dinheiro foi implementa da mesma maneira que a atividade Detetar Objetos. A única diferença está no modelo utilizado no reconhecimento. Para ser possível detetar o valor de notas e moedas, foi treinada uma rede neuronal que aprendesse a reconhecer diferentes tipos notas e moedas. Para este trabalho, foram implementadas notas

de cinco, dez, vinte, cinquenta, cem e duzentos euros e moedas de um e dois euros e de dez, vinte e cinquenta cêntimos. O modelo construído seguiu a arquitetura e Single-Shot Detector e foi convertido para o formato TensorFlow Lite.

No Android Studio, usou-se o detetor da API do TensorFlow Lite para detetar todas as classes implementadas no modelo. Ele gera uma lista de resultados de recolhimento que inclui o nome de cada classe detetada e a pontuação de confiança. A classe que obtiver maior percentagem de confiança será divulgada, em voz sintetizada, ao utilizador invisual. É de notar que também foi estabelecido um limite de confiança de 50%, pelo que resultado apresentado tem uma confiança superior ou igual a esse valor. O Excerto de Código 6.9 mostra como foi construido o código necessário para encontrar e indicar a classe que foi detetada na imagem. Caso não haja resultados ou os mesmos não satisfaçam as condições, nada será informado.

```
...
detectorMoney = TFLiteObjectDetectionAPIModel.create(this, MODEL_MONEY,
    LABELS_FILE_MONEY, TF_OD_API_INPUT_SIZE, IS_QUANTIZED2);

...

final List<Detector.Recognition> resultsMoney = detector3.recognizeImage(
    croppedBitmap);

Detector.Recognition object_detected = null;

if(results != null){
  confidence_max = results.get(0).getConfidence();
}

for (final Detector.Recognition result : results) {
  if (result.getConfidence() >= confidence_max) {
    confidence_max = result.getConfidence();
    object_detected = result;
  }
}

if(object_detected != null && object_detected.getConfidence() >=
    minimo_confianca){
  textView.setText(object_detected.getTitle());
  ...
  speak(object_detected.getTitle());
}else{
  textView.setText("");
}
...
```

Excerto de Código 6.9: Implementação da funcionalidade Identificar Dinheiro na Aplicação Móvel.

## 6.5 Identificar Produtos

Nesta última secção pretende-se explicar como foi implementada a última funcionalidade do sistema, Identificar Produtos. Detalhes importantes como a construção de um leitor de código de barras e a identificação de produtos segundo uma lista com vários códigos serão explicados. Para além disso, será revelado como foi implementada a funcionalidade Adicionar Novos Produtos, que é manuseada por um assistente visual.

### 6.5.1 Leitura de códigos de barras

A leitura de códigos de barras foi possível com a ajuda da implementação da classe **IntentIntegrator**. A mesma foi escrita em Java para facilitar a integração com o *Barcode Scanner* via *Intents* [117]. No Excerto de Código 6.10 é possivel observar como foi implementada a classe IntentIntegrator.

```
...

IntentIntegrator integrator = new IntentIntegrator(this);
integrator.setCaptureActivity(CaptureAct.class);
integrator.setOrientationLocked(false);
integrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
integrator.setPrompt(``Scanning Code");
integrator.initiateScan();

...

IntentResult result = IntentIntegrator.parseActivityResult(requestCode,
    resultCode,data);

product = findProduct(result.getContents());

...
```

Excerto de Código 6.10: Construção do leitor de código de barras do sistema.

Apesar da leitura de código de barras ser sempre efetuada, a missão desta funcionalidade só será um sucesso se o código lido for associado a um determinado produto de uma lista que foi incluída na aplicação móvel. Essa lista é, na verdade, um ficheiro de texto que contém alguns códigos de barras de produtos que são atualmente comuns. Ao ser detetado um código barras, o sistema percorre esse ficheiro e tenta encontrar um código que seja igual ao que foi lido. A função ***findProduct()*** recebe como parâmetro o código de barras e retorna o nome do produto que está ligado a ele. Caso o código não esteja no ficheiro criado, a aplicação móvel informa o utilizador que o mesmo não foi implementado no sistema.

### 6.5.2 Adição de novos produtos ao sistema

Quando um produto não faz parte do sistema, a aplicação SeeMyHome oferece a solução de adicioná-lo. Esta tarefa foi desenvolvida na funcionalidade Adicionar Novo Produto. Para

incluir um novo produto, um assistente visual terá de apresentar o respetivo código de barras à aplicação móvel. O leitor utilizado na identificação de produtos serviu para esta atividade. Ao ler o novo código de barras, o assistente visual terá de indicar o nome do produto que lhe é correspondido. Para esta funcionalidade, não foi utilizado um servidor em nuvem para guardar os nomes e códigos de barras de novos produtos, de forma a virem a ser identificados por outros utilizadores como acontece na deteção de objetos. Isto porque, cada produto é caracterizado em diferentes aspetos, com por exemplo, a marca do fabricante. Sendo assim, demos liberdade ao utilizador de escolher o nome que achar mais adequado. Por fim, o novo produto é adicionado à lista de produtos que se encontra no ficheiro de texto da aplicação do utilizador, com o objetivo de vir a ser identificado futuramente.

## 6.6 Considerações finais

Em suma, este capítulo apresentou alguns detalhes mais técnicos que foram fundamentais para a elaboração do sistema projetado. Nele explicamos como foi implementada cada funcionalidade ou recurso que foi construido na aplicação móvel. Por fim, a nível de programação de código, foram apresentadas as abordagens utilizadas.

# Capítulo 7

# Testes e Resultados Experimentais

## 7.1 Introdução

Este capítulo apresenta os resultados de um estudo realizado ao *software* que foi construído. Determinadas metodologias de deteção de objetos foram aplicadas e diferentes modelos de rede neuronal foram testados e analisados a fim de encontrar a solução mais adequada para resolução do problema proposto. Veremos que o treino de redes neuronais com Aprendizagem por Transferência e Aumento Dados torna-se mais eficiente, sendo capaz de alcançar melhores resultados de deteção num conjunto de dados reduzido. Todas os testes e resultados experimentais obtidos serão aqui divulgados.

## 7.2 Detetor de objetos

O detetor de objetos implementado recorre a um modelo que é personalizado e desenvolvido por um número indeterminado de utilizadores. Quer isto dizer que o sistema produzido irá operar com uma rede neuronal que vai sendo treinada de forma incremental, onde a sua evolução depende exclusivamente do uso que os vários utilizadores lhe derem. É de observar que, neste caso não é possível saber de antemão quantas classes o modelo vai ter e em que quantidades de imagens cada uma vai ser treinada. Porém, o sistema tem de estar preparado para todas as situações possíveis. Por exemplo, uma vez que se propõem apenas dez imagens para iniciar a criação de uma nova classe, é importante que o seu reconhecimento não seja afetado por classes que já tenham um número bem desenvolvido de imagens e vice-versa. Nesta secção, será descrito o conjunto de dados que foi usado para realizar uma variedade de testes com modelos que adotam diferentes abordagens de treino de redes neuronais voltadas para o reconhecimento de objetos. Será explicado o procedimento empregue na distribuição dos dados em dois grupos: treino e teste. Por último, serão mencionadas todas as métricas de avaliação utilizadas para analisar os resultados que são fornecidos pelo detetor de objetos.

### 7.2.1 Conjunto de dados

O objetivo da presente subsecção é apresentar o conjunto de dados que foi especialmente desenvolvido para treinar e testar os modelos de rede neuronal que fazem parte das atividades experimentais realizadas neste trabalho. Como é de esperar, a distribuição dos dados foi feita em dois grupos: treino e teste. O primeiro é responsável pela aprendizagem da rede neuronal, ou seja, por treinar o modelo que irá prever resultados de objetos presentes nas imagens. Por outro lado, o conjunto de teste serve para testar o modelo e é frequentemente utilizado para medir o seu desempenho e a capacidade da rede alcançar bons resultados de reconhecimento. De seguida, veremos como foram elaborados estes dois grupos.

#### 7.2.1.1 Treino

O conjunto de dados de treino contém 1510 imagens que foram anotadas manualmente. Elas estão agrupadas em quatro classes de objetos com diferentes quantidades:

- Classe Tesoura - **750 imagens com 15 objetos**;
- Classe Pente - **500 imagens com 10 objetos**;
- Classe Gilete - **250 imagens com 5 objetos**;
- Classe Comando - **10 imagens com 1 objeto**.

Normalmente o treino de uma rede neuronal é aplicado a um conjunto de dados que inclui uma quantidade definida e igual de imagens para todas os objetos. No entanto, como foi mencionado anteriormente, o sistema desta dissertação pretende trabalhar com um conjunto indefinível de classes e imagens e, sendo assim, o conjunto de dados foi criado a pensar nesse novo avanço. De todas as classes de objetos, a primeira é que contém mais imagens, 750 imagens recolhidas de 15 objetos diferentes. Depois, duas classes com 500 e 250 imagens respetivamente e, por fim, uma classe com apenas 10 imagens. Esta última representa um novo objeto que foi adicionado ao sistema. A ideia é verificar como é que esta nova classe se comporta num conjunto de dados mais evoluído e com objetos em diferentes quantidades.


Figura 7.1: Amostra do conjunto de dados utilizado para efeito de treino.

A Figura 7.1 mostra uma parte do conjunto de dados construído. Por cada objeto incluído nas três primeiras classes, foram recolhidas cinquenta imagens em quatro tipos de ambientes. A ideia é que a rede aprenda a reconhecer objetos em cenários diversificados como, ambientes com excelente e fraca iluminação, ambientes em que o objeto se encontra cortado na imagem, ou seja a sua totalidade não é visível na mesma e ambientes onde os objetos implementados são sobrepostos ou juntos com outros que a rede neuronal não conhece. Portanto, por cada subconjunto de cinquenta imagens, 30% foram tiradas em ambientes luminosos, outros 30% em ambientes com fraca iluminação, 20% em ambientes com o objeto parcialmente oculto e, por fim, os últimos 20% em ambientes com o objeto sobreposto. Para o objeto Comando, o procedimento é o mesmo, mas somente para dez imagens.

#### 7.2.1.2 Teste

Na fase de validação do sistema, o conjunto de teste é fundamental para avaliar os resultados. Para isso, é feita uma simulação de previsões reais, onde são fornecidos à rede dados que nunca foram vistos. Esses dados formam o conjunto de teste. Eles iram comprovar se o modelo está de facto operacional e se é capaz de fazer uma boa generalização dos dados. O método mais comum na criação de um conjunto de teste, é distribuir o conjunto de dados original numa proporção de 80% para treino e 20% para teste. No entanto, para o sistema que foi desenvolvido, se formos atribuir 20% das dez imagens da classe comando para teste, significa que só serão testadas apenas duas imagens. Tal, não é suficiente para avaliar uma classe e ter a certeza que estão a ser produzidos bons resultados de deteção. Sendo assim, decidiu-se utilizar as 1510 imagens e anotações para treino e elaborar um novo conjunto só para teste. O mesmo contabiliza um total de 240 imagens das quatro classes treinadas, isto é, 60 por cada classe de objeto e dessas 60 imagens 25% para cada tipo de ambiente.

### 7.2.2 Métricas de avaliação utilizadas

Para avaliar a capacidade de generalização da rede, recorreu-se a uma métrica que considera a **taxa de acerto** de cada classe de objeto no respetivo conjunto de teste. A taxa de acerto permite saber a percentagem de sucesso no reconhecimento de cada objeto. Para a calcular utilizou-se a seguinte expressão matemática 7.1:

$$(n/t) * 100. \quad (7.1)$$

A variável n corresponde ao número total de acertos e a variável t ao número total de testes realizados. É de notar que, quanto maior for o valor da taxa de acerto, mais assertivo é o algoritmo de ML.

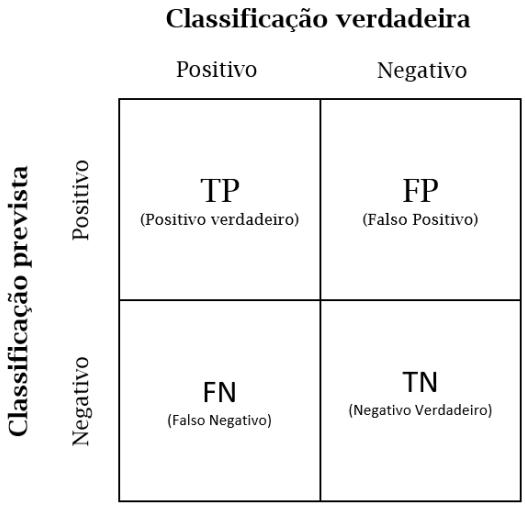



Figura 7.2: Exemplo genérico de uma Matriz de Confusão.

O desempenho do algoritmo de classificação foi medido através de uma **matriz de confusão**. Ela procura compreender a relação existente entre os acertos e erros que o modelo de rede neuronal apresentou [118]. A sua estrutura baseia-se numa tabela distribuída por linhas e colunas, em que o seu número depende do total de classes de objetos que o modelo possui. Na Figura 7.2 encontra-se esquematizado um exemplo genérico e popular de uma matriz de confusão associada a uma classificação binária, ou seja, com duas classes: Positivo e Negativo. Os resultados são agrupados na matriz com base em quatro variáveis, que se definem da seguinte forma:

- **Positivo Verdadeiro** (*True Positive* - **TP**): Quantidade de vezes que modelo classificou corretamente uma amostra positiva como positiva;
- **Falso Positivo** (*False Positive* - **FP**): Quantidade de vezes que modelo classificou erradamente uma amostra negativa como positiva;
- **Negativo Verdadeiro** (*true Negative* - **TN**): Quantas vezes que o modelo classificou corretamente uma amostra negativa como negativa;
- **Falso Negativo** (*False Negative* - **FN**): Quantas vezes que o modelo classificou erradamente uma amostra positiva como negativa;

Com base nos valores das variáveis apresentadas, é possível tirar partido da matriz de confusão e avaliar o desempenho do modelo de rede neuronal. Nesta dissertação, foram utilizadas as seguintes quatro métricas de medição de desempenho:

- **Acurácia** (*Acurary*): mede a *performance* geral do modelo. De todos os resultados de classificação, quantos o modelo classificou corretamente. Pode ser calculada usando a Expressão 7.2:

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (7.2)$$

- **Precisão** (*Precision*): Mede a exatidão do modelo em classificar amostras como positivas. Quanto maior for o seu valor, maior é a confiança depositada no modelo. Pode ser calculado através da Expressão 7.3:

$$\frac{TP}{TP + FP} \quad (7.3)$$

- ***Recall***: Mede a capacidade do modelo detetar amostras positivas. Quanto maior for o seu valor, maior é quantidade de amostras que foram detetadas corretamente. Ele é calculado a partir da Expressão 7.4:

$$\frac{TP}{TP + FN} \quad (7.4)$$

- ***F1 SCORE***: Mede a média harmônica entre Precisão e *Recall*. Para a calcular utiliza-se

a Expressão Matemática 7.5

$$\frac{2TP}{2TP + FP + FN} \quad (7.5)$$

## 7.3 Metodologia de teste e análise de resultados

O procedimento de testes ao *software* construído consistiu na realização de quatro experiências com modelos de rede neuronal que seguem metodologias de treino diferentes. A ideia é encontrar um sistema de deteção de objetos eficaz e capaz de responder de forma assertiva ao problema que se propõem nesta dissertação. É óbvio que objetivo principal é que os objetos que são adicionados pelo assistente visual de cada utilizador sejam bem detetados. Para isso, o conjunto de dados de treino e teste, elaborado no passo anterior, vai ajudar a compreender qual a melhor solução que permite que classes novas e já existentes se adaptem bem no reconhecimento. As experiências efetuadas traduziram-se na aplicação de diversos métodos voltados para o reconhecimento de objetos, tais como: avaliação de resultados fornecidos por um modelo de rede neuronal construído desde o zero, do mesmo modelo, mas utilizando a técnica de Aumento Dados, de um modelo pré-treinado num grande conjunto de dados onde as quatro classes são treinadas através de Aprendizagem por Transferência e, por fim, a aplicação novamente da técnica Aumento de dados neste último modelo.

### 7.3.1 Modelo de rede neuronal construído do zero

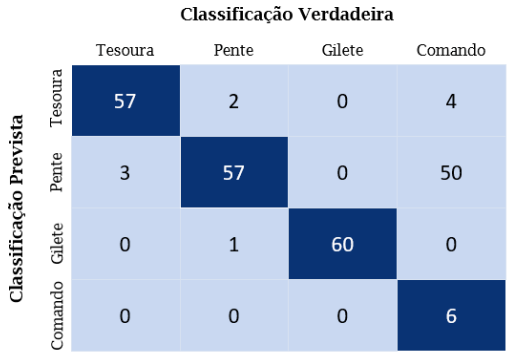
Numa primeira experiência, começou-se por criar e treinar uma rede neuronal a partir do zero, isto é, desde a fase onde aprende características relevantes nas imagens de treino até a produção de resultados de saída. A mesma foi construída com base na arquitetura SSD Mobilenet V1, com extrator de recursos FPN-Lite. O objetivo desta primeira prática é avaliar a capacidade de deteção de objetos num conjunto de dados reduzido e com uma rede neuronal que segue os princípios básicos.

| Classes de objetos | Taxa de acerto em % | | | | |
|---|---|---|---|---|---|
| | Ambiente luminoso | Ambiente com fraca iluminação | Ambiente com o objeto parcialmente oculto na imagem | Ambiente com o objeto sobreposto com outros | Precisão geral da classe |
| Tesoura | 100% | 100% | 93% | 86% | 95% |
| Pente | 100% | 100% | 100% | 80% | 95% |
| Gilete | 100% | 100% | 100% | 100% | 100% |
| Comando | 33% | 7% | 0% | 0% | 10% |

Tabela 7.1: Resultados obtidos com um modelo de rede neuronal construido do zero.

Após concluir o treino do modelo de rede neuronal, o mesmo foi submetido a testes no conjunto de dados que contém as imagens destinadas a esse fim. Os resultados obtidos podem ser observados na Tabela 7.1. Para este primeiro modelo, à exceção da classe comando, os resultados foram bons no geral. As quatro primeiras classes de objetos alcançaram uma taxa de acerto superior a 90%, sendo que a Gilete consegui acertar em todos os testes, obtendo uma percentagem máxima de 100%. Contudo, vale realçar que o objeto de teste mais

importante, Comando, não atingiu um bom resultado. Com somente dez imagens, esta classe só acertou seis vezes em sessenta testes realizados. É um resultado muito ruim, quando o objetivo é utilizar apenas uma pequena quantidade de imagens para treinar um objeto novo.




(a) Matriz de Confusão

| | Precision | Recall | F1 SCORE |
|---|---|---|---|
| Tesoura | 0.90 | 0.95 | 0.93 |
| Pente | 0.52 | 0.95 | 0.67 |
| Gilete | 0.98 | 1.0 | 0.99 |
| Comando | 1.0 | 0.10 | 0.18 |
| **No geral (%)** | **85%** | **75%** | **69%** |

Accuracy (%) : 75%

(b) Resultados das métricas de avaliação de desempenho

Figura 7.3: Matriz de confusão associado ao modelo de rede neuronal construido do zero.

Através da matriz de confusão e dos resultados de desempenho mostrados na Figura 7.4, é possível analisar com mais detalhe este primeiro modelo. Como é de prever, devido à baixa taxa de acerto da classe Comando, a métrica *Accuracy* e *Recall* ficou pelos 75%. Nos testes às restantes classes, nenhuma se confundiu com o objeto Comando, somente houve uma confusão mínima entras as classes Tesoura e Pente. Devido a isso, o mesmo assume o valor máximo de precisão. No entanto, o comando confundiu-se cinquenta vezes com o objeto Pente e quatro com Tesoura, o que origina a redução de precisão geral, para os 85%.

### 7.3.2 Modelo de rede neuronal construído do zero com aumento de dados

Para melhorar os resultados de deteção do objeto Comando, uma possível solução passa por aplicar a técnica *Data Augmentation*, em português, Aumento de dados no modelo construído desde o zero. Este método é caracterizado por um conjunto de procedimentos que visam aumentar artificialmente a quantidade de dados, daí o seu nome. A produção de dados sintéticos é, atualmente, o melhor método para adicionar mais dados ao treino de modelos de redes neuronais [119]. É de notar que, para o sistema desenvolvido a quantidade de imagens de determinadas classes de objetos, numa fase inicial, vai ser muito reduzida dado que são apenas recolhidas dez imagens por cada objeto adicionado. No entanto, o objetivo é que o assistente visual não tenha de recolher muitas mais, de modo que esta tarefa não se torne demasiada massuda e desmotivante. Para isso, a abordagem de Aumento de Dados é fundamental para o sistema desenvolvido.

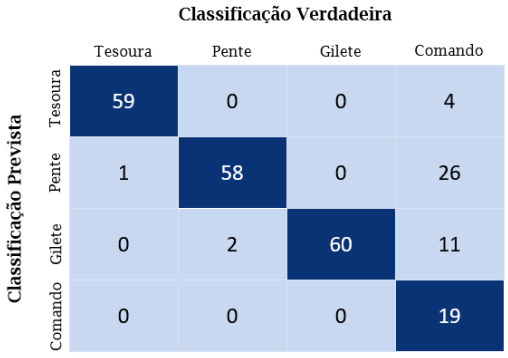
Para obter mais dados, a solução passa por realizar pequenas alterações nas imagens do conjunto de dados, tais como: rotação aleatória, redimensionamento, cortes, mudança de contraste, tradução (onde a imagem é movida ao longo da direção X, Y), entre outras. Apesar de serem as mesmas imagens, ao serem submetidas a essas técnicas, as novas imagens

produzidas artificialmente vão ser interpretadas pela rede neuronal como distintas. Desta forma, é possível então aumentar o *dataset* sem haver a necessidade de recolher novos dados. Para esta segunda experiência, o modelo anterior foi novamente treinado, mas desta vez com Aumento de Dados. Para aumentar o conjunto, foram aplicadas técnicas de mudança de contraste, inversão da imagem horizontalmente e rotação da imagem num ângulo de noventa graus. Os resultados obtidos deste segundo modelo encontram-se na seguinte Tabela 7.2.

| Classes de objetos | Taxa de acerto (%) | | | | |
|---|---|---|---|---|---|
| | Ambiente luminoso | Ambiente com fraca iluminação | Ambiente com o objeto parcialmente oculto na imagem | Ambiente com o objeto sobreposto com outros | Precisão geral da classe |
| Tesoura | 100% | 100% | 100% | 93% | 98% |
| Pente | 100% | 100% | 93% | 93% | 97% |
| Gilete | 100% | 100% | 100% | 100% | 100% |
| Comando | 27% | 47% | 40% | 13% | 32% |

Tabela 7.2: Resultados obtidos no modelo anterior aplicando a técnica de Aumento de Dados.

Segundo os valores apresentados, chegou-se à conclusão de que o Aumento de Dados veio melhorar ligeiramente os resultados. É de notar que a classe Comando ainda continua abaixo dos 50%. Ao contrário do que se verificou na experiência anterior, onde o objeto só foi detetado em ambientes com boa e fraca iluminação, neste novo método, o mesmo já é reconhecido nos restantes tipos de ambientes, ainda que com baixa taxa de acerto. Noutro ponto de vista, as duas primeiras classes de objetos atingiram valores melhores, passaram de 95% para 98% e 97%, respetivamente. Na Figura 7.4, é possível perceber o que aconteceu nos casos em que o detetar não foi capaz de acertar corretamente na classe verdadeira, mediante a observação da matriz de confusão e das respetivas medições de desempenho do modelo.




(a) Matriz de Confusão

| | *Precision* | *Recall* | *F1 SCORE* |
|---|---|---|---|
| Tesoura | 0.94 | 0.98 | 0.96 |
| Pente | 0.68 | 0.97 | 0.80 |
| Gilete | 0.82 | 1.0 | 0.90 |
| Comando | 1.0 | 0.32 | 0.48 |
| **No geral (%)** | **86.02%** | **81.67%** | **78.5%** |

**Accuracy (%) : 81,67%**

(b) Resultados das métricas de avaliação de desempenho

Figura 7.4: Matriz de confusão do modelo de rede neuronal construido do zero com Aumento de Dados.

Comparando a matriz de confusão apresentada com a obtida anteriormente, é possível ver que os objetos Tesoura e Pente confundiram-se menos vezes ao introduzir a metodologia de Aumento de dados, daí a obterem melhores resultados de *Recall*. A classe Comando também se confundiu muito menos com as outras implementadas e isso fez com que a precisão geral aumenta-se de 85% para 86.02%. Uma vez que houve menos confusão entre

classes e mais acertos, principalmente no objeto Comando, a *Accuracy* obteve um resultado de 81,67%, mais 6,67% que o modelo anterior que não usava a técnica de aumento de dados.

### 7.3.3 Modelo de rede neuronal pré-treinado com Aprendizagem por Transferência

É verdade que os resultados melhoram com a introdução da técnica Aumento de Dados, porém, ainda não é suficiente para obter uma taxa de acerto promissora na classe que contém apenas dez imagens de treino. Assim sendo, partiu-se para uma nova abordagem que se traduz na utilização de um modelo pré-treinado num grande conjunto de dados. Em vez de construir e treinar uma rede neuronal desde o seu começo, recorreu-se à Aprendizagem por Transferência do modelo pré-treinado SSD MobileNet V1, que aprendeu a reconhecer noventa classes diferentes de objetos num conjunto de dados com milhares de imagens. A Aprendizagem por Transferência é um método de ML muito popular, onde é possível aproveitar o conhecimento de uma rede pré-treinada para realizar uma tarefa semelhante e que já foi aprendida [120]. A ideia é utilizar um modelo que foi treinado num conjunto de dados em grande escala, tornando-se suficiente para servir de modelo genérico do mundo visual. Através dele é possível tirar partido de todo o conhecimento adquirido durante a fase de aprendizagem, sem haver a necessidade de treinar a rede completa.

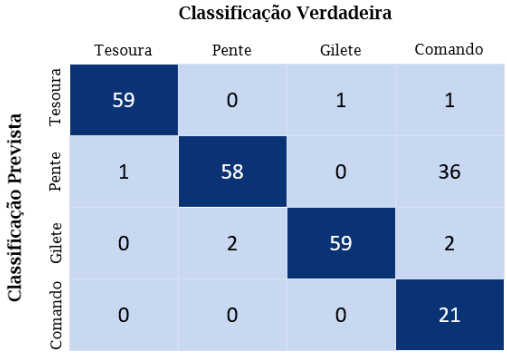
As redes neuronais profundas, como vimos no Capítulo 3, estão estruturadas em camadas que começam por extrair características que sejam importantes para realização da tarefa que lhe foi destinada, enquanto que, as últimas permitem obter os resultados que dão resposta ao problema em questão. Neste projeto, utilizou-se a aprendizagem de transferência congelando as camadas de extração de recursos do modelo SSD MobileNet V1, direcionando deste modo, o treino apenas para as últimas camadas da rede. Portanto, não são atualizados os seus pesos no treino de novos dados, uma vez que ela já apreendeu recursos muitos específicos e que são gerais para quase todos os tipos de dados. Desta forma, a Aprendizagem por Transferência é uma técnica aconselhável para conjuntos de dados com classes de objetos com poucas imagens de treino [121].

| Classes de objetos | Taxa de acerto (%) | | | | |
|---|---|---|---|---|---|
| | Ambiente luminoso | Ambiente com fraca iluminação | Ambiente com o objeto parcialmente oculto na imagem | Ambiente com o objeto sobreposto com outros | Precisão geral da classe |
| Tesoura | 100% | 93% | 100% | 100% | 98% |
| Pente | 100% | 93% | 93% | 100% | 97% |
| Gilete | 93% | 100% | 100% | 100% | 98% |
| Comando | 60% | 53% | 13% | 13% | 35% |

Tabela 7.3: Resultados obtidos aplicando Aprendizagem por Transferência num modelo pré-treinado.

A Tabela 7.3 apresenta os resultados obtidos com a intervenção da técnica Aprendizagem por Transferência no modelo pré-treinado. Para este caso, é provável que determinadas classes percam alguma percentagem de acerto, isto porque, a rede neuronal não aprendeu a extrair características exclusivamente para as imagens do conjunto de treino. É de notar que o extrator de recursos foi congelado. Isso é percetível na classe Gilete, que anteriormente obtinha uma percentagem de acerto de 100% e, agora com este método, passou para 98%.

Sobre outra perspetiva, as classes Tesoura e Pente, mantiveram a percentagem de acerto. Já a classe Comando continua a obter uma taxa inferior a pelo menos 50%. Porém, apesar do resultado houve alguns aspetos positivos que devem ser considerados. Nos ambientes com boa e fraca iluminação houve um progresso bastante apreciável, onde se verificou uma taxa de acerto de 60% e 50%, respetivamente. Isso significa que este objeto está num bom caminho para alcançar um resultado mais revelador e promissor.




(a) Matriz de Confusão

| | *Precision* | *Recall* | *F1 SCORE* |
|---|---|---|---|
| Tesoura | 0.97 | 0.98 | 0.89 |
| Pente | 0.61 | 0.97 | 0.75 |
| Gilete | 0.94 | 0.98 | 0.96 |
| Comando | 1.0 | 0.35 | 0.52 |
| **No geral (%)** | **88%** | **82%** | **78%** |

**Accuracy (%) : 82,08%**

(b) Resultados das métricas de avaliação de desempenho

Figura 7.5: Matriz de confusão associado ao modelo pré-treinado com Aprendizagem por Transferência.

Avaliando os resultados através da matriz de confusão e das métricas de desempenho mostradas na Figura 7.5, facilmente se chega à conclusão que houve um ligeiro aumento do valor de precisão, *Recall* e *Accuracy*, impulsionado por mais acertos da classe Comando.

### 7.3.4 Modelo de rede neuronal pré-treinado com Aprendizagem por Transferência e com Aumento de Dados

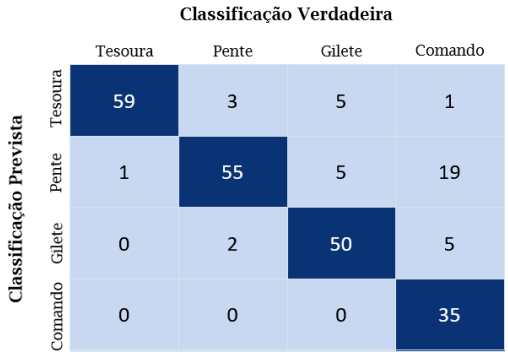
| Classes de objetos | Taxa de acerto (%) | | | | |
|---|---|---|---|---|---|
| | Ambiente luminoso | Ambiente com fraca iluminação | Ambiente com o objeto parcialmente oculto na imagem | Ambiente com o objeto sobreposto com outros | Precisão geral da classe |
| Tesoura | 100% | 93% | 100% | 100% | 98% |
| Pente | 100% | 93% | 86% | 86% | 92% |
| Gilete | 93% | 67% | 86% | 86% | 83% |
| Comando | 73% | 73% | 33% | 53% | 58% |

Tabela 7.4: Resultados obtidos aplicando Aprendizagem por Transferência e aumento de dados no modelo pré-treinado.

Numa última experiência, decidiu-se utilizar novamente a técnica de Aumento de Dados, mas desta vez no modelo pré-treinado. Como vimos anteriormente, os resultados melhoram ao aplicar o Aumento de Dados num modelo treinado desde o zero. Por outro lado, também houve melhorias com a introdução da Aprendizagem por Transferência num modelo pré-treinado. Logo, a finalidade desta última experiência é analisar os resultados obtidos quando essas duas abordagens trabalham em conjunto. A Tabela 7.4 mostra os resultados

da taxa de acerto obtidos para este novo procedimento de treino.

Ao praticar o aumento de dados no modelo pré-treinado da experiência anterior, finalmente conseguiu-se conquistar uma taxa de acerto positiva para a classe Comando. A mesma, que antes verificava um valor de 35%, passou para 58%. Contudo, apesar deste ser um dos sucessos mais esperados neste novo modelo, a classe Gilete acabou por sofreu uma perda na taxa de acerto. Antes constatava valores entre os 90% e 100%, e agora com esta abordagem só foi capaz de atingir uma percentagem de acerto de 83%. De seguida, através dos resultados de medição de desempenho da rede apresentados na Figura 7.6, é possível analisar com maior detalhe o comportamento do modelo pré-treinado com Aprendizagem por Transferências e Aumento de Dados.




(a) Matriz de Confusão

| | Precision | Recall | F1 SCORE |
|---|---|---|---|
| Tesoura | 0.87 | 0.98 | 0.92 |
| Pente | 0.69 | 0.92 | 0.79 |
| Gilete | 0.88 | 0.83 | 0.85 |
| Comando | 1.0 | 0.58 | 0.74 |
| **No geral (%)** | **86%** | **83%** | **82,5%** |

Accuracy (%) : 82,92%

(b) Resultados das métricas de avaliação de desempenho

Figura 7.6: Matriz de confusão do modelo com Aprendizagem por Transferência e Aumento de Dados.

Com o objeto Comando a superar mais resultados positivos, o *Recall* e *Accuracy* subiram de 82% para 83% e 82.08% para 82.92%, respetivamente. Já a precisão desceu para 86%, por causa das confusões existentes nas classes verdadeiras Gilete e Pente. No entanto, este foi o modelo que obteve melhores resultados de desempenho. Embora a confiança depositada no mesmo seja um pouco menor devido à precisão obtida, a sua *performance* geral e a sua capacidade de detetar amostras positivas é a mais alta entre as demais metodologias. Posto isto, tudo indica que as técnicas de Aprendizagem por Transferência e Aumento de Dados quando são combinadas, produzem resultados promissores e, sendo assim, este modelo é considerado o mais eficaz para dar resposta à solução do sistema de deteção de objetos que se pretende incluir neste trabalho.

## 7.4 Considerações finais

Neste capítulo foram testadas diferentes metodologias de treino de redes neuronais voltadas para o reconhecimento de objetos. A realização das experiências permitiu avaliar devidamente o sistema e encontrar uma solução que seja capaz de dar resposta ao problema proposto nesta dissertação. Concluiu-se que um modelo com Aprendizagem por Transferência e Aumento de dados é a via mais vantajosa para o detetor de objetos empregue neste trabalho.

# Capítulo 8

# Conclusão e Trabalho Futuro

A conclusão deste projeto traduz-se no desenvolvimento de um sistema que vem a revelar a sua utilidade na assistência a pessoas invisuais e que promete melhorar a sua qualidade de vida. É de sublinhar que os *smartphones* atuais são uma ferramenta valiosa, com enorme potencial para apoiar estas pessoas e ajudá-las a compreender melhor o mundo que as rodeia. A deteção de objetos em tempo real, provou ser um excelente recurso na conquista por uma vida mais independente. Segundo o estudo [35], analisado na Secção 2.2 do Capítulo 2, a funcionalidade de detetar objetos foi testada por pessoas invisuais (obteve uma taxa de aprovação de 90%) foi classificada como muito interessante, o que significa que é vista como um ótimo ponto de partida para futuros desenvolvimentos. O projeto idealizado nesta dissertação vem no sentido de procurar criar um sistema capaz de reunir um nível de consenso semelhante por parte do público alvo.

Este projeto de dissertação teve como principal objetivo planear a construção de uma aplicação móvel que seja capaz de prestar auxílio a pessoas com dificuldades mais ou menos graves de visão, através de um recurso de deteção de objetos, em tempo real. Em primeiro lugar, foi feito um estudo bibliográfico com o objetivo de conhecer o que está a ser desenvolvido no âmbito desta dissertação. De seguida, foi elaborado um plano de desenvolvimento que descreve todas as técnicas e procedimentos a serem utilizados na fase de produção do sistema desejado. Mais a frente, foi um apresentada uma abordagem conexionista aplicada ao reconhecimento de objetos, que inclui alguns conceitos importantes para obter um forte conhecimento sobre este tema e o consequente estudo de diferentes algoritmos de deteção de objetos. Finalmente, foi construido o protótipo da aplicação móvel esperada e demonstrado todo o trabalho que foi desenvolvido nesta dissertação. Alguns detalhes de implementação mais relevantes foram divulgados numa fase posterior, de modo a compreender como foi elaborado todo o conceito do sistema criado. Por fim, e não menos importante, são apresentados os testes e experiências realizadas ao produto final, onde foi possível encontrar abordagens mais eficazes para o treino de modelos de deteção de objetos com melhor desempenho.

Como trabalho futuro procura-se validar o sistema desenvolvido com pessoas invisuais. O objetivo é observar a reação dos consumidores finais da aplicação móvel fase ao produto construído e às funcionalidades implementadas. Desta forma, é possível obter uma informação mais precisa sobre o sucesso e progresso do sistema. É fundamental saber se as pessoas invisuais estão satisfeitas com a aplicação ou, em caso contrário, em que aspetos poderia ser melhorada. A sua opinião é muito importante para saber se o sistema pode seguir para uma fase de disponibilização ou se é necessário ainda reavaliar alguns aspetos. Para além disso, poderão ser apresentas algumas sugestões que levem a novas ideias de trabalho.

# Acknowledgement

This work was supported by the ”Fundação para a Ciência e a Tecnologia” (FCT) in the scope of the project CPCA/A0/429960/2021 and developed in part at the High Performance Computing Center of the University of Évora (HPC-UÉ).

# Bibliografia

[1] Be My Eyes. (2021) Bringing sight to blind and low-vision people. Last access to 6 December 2021. [Online]. Available: https://www.bemyeyes.com/ xiii, 13

[2] Novartis Pharmaceuticals Corporation. (2020) Viaopta daily. Last access to 6 December 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.viaopta.daily&hl=pt_PT&gl=US xiii, 13, 14

[3] Google LLC. (2021) Lookout de google. Last access to 6 December 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.reveal&hl=pt_BR&gl=US xiii, 13, 14

[4] Microsoft Corporation. (2021) Seeing ai: Talking camera for the blind. Last access to 6 December 2021. [Online]. Available: https://apps.apple.com/us/app/seeing-ai/id999062298 xiii, 13, 14

[5] CloudSight Inc. (2019) Taptapsee. Last access to 23 December 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.msearcher.taptapsee.android&hl=pt_PT&gl=US xiii, 13, 14

[6] The Engineer. (2014) These glasses will let near blind people see again. Last access to 28 November 2021. [Online]. Available: https://wonderfulengineering.com/these-glasses-will-let-near-blind-people-see-again/ xiii, 17

[7] S. V, M. R, A. M. S, P. V, and S. M, "Smart glove for blind using tensorflow," *International Research Journal of Engineering and Technology (IRJET)*, pp. 1–6, 2020. xiii, 17, 18, 20

[8] Babos Dávid . (2020) Blind assistance helmet. Last access to 18 November 2021. [Online]. Available: https://www.kickstarter.com/projects/blindhelmet/blind-assistance-helmet xiii, 17

[9] Paul Smith. (2013) Enginursday - rfid technology. Last access to 30 November 2021. [Online]. Available: https://www.sparkfun.com/news/1284 xiii, 17

[10] Britannica, The Editors of Encyclopaedia. (2021) Barcode. Last access to 30 November 2021. [Online]. Available: https://www.britannica.com/technology/barcod xiii, 17

[11] Erik Gregersen. (2021) Qr code. Last access to 30 November 2021. [Online]. Available: https://www.britannica.com/technology/QR-Code xiii, 17

[12] A. Neves, I. González, J. Leander, and R. Karoumi, "A new approach to damage detection in bridges using machine learning," *International Conference on Experimental Vibration Analysis for Civil Engineering Structures*, pp. 73–84, 2017. xiii, 26

[13] L. Shen, Q. Zhang, G. Cao, and H. Xu, "Fall detection system based on deep learning and image processing in cloud environment," *Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 590–598, 2018. xiii, 30

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, pp. 2278–2324, 1998. xiii, xv, 33

[15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET). IEEE*, pp. 1–6, 2017. xiii, 32, 34

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European conference on computer vision*, pp. 818–833, 2014. xiii, 34

[17] S. Ravindra. (2017) How convolutional neural networks accomplish image recognition? Last access to 27 January 2021. [Online]. Available: https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html xiii, 35

[18] S. Curtis. (2020) Rest vs. crud. Last access to 11 June 2022. [Online]. Available: https://stevenpcurtis.medium.com/rest-vs-crud-ca5522bf3fc3 xiii, 50

[19] Neil Mawston. (2021) Strategy analytics: Half the world owns a smartphone. Last access to 28 November 2021. [Online]. Available: https://news.strategyanalytics.com/press-releases/press-release-details/2021/Strategy-Analytics-Half-the-World-Owns-a-Smartphone/default.aspx 1

[20] W. H. Organization, "Aworld report on vision," pp. 1–180, 2019. 1

[21] R. Manduchi and S. Kurniawan, "Watch your head, mind your step: mobility-related accidents experienced by people with visual impairment." *Dept. Comp. Eng., Univ. California, Santa Cruz, Tech. Rep*, 2010. 1

[22] Statcounter Global Stats. (2021) Mobile operating system market share worldwide. Last access to 3 February 2021. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide 3

[23] A. Salunkhe, M. Raut, S. Santra, and M. S. Bhagwat, "Android-based object recognition application for visually impaired," *In ITM Web of Conferences (Vol. 40, p. 03001), EDP Sciences*, 2021. 6, 7, 8, 12, 36

[24] R. Girshick, "Ross girshick and jeff donahue and trevor darrell and jitendra malik," *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014. 6, 36

[25] ——, "Fast r-cnn," *In Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015. 6, 37

[26] R. G. Shaoqing Ren, Kaiming He and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, pp. 91–99, 2015. 6, 38

[27] R. G. Joseph Redmon, Santosh Divvala and A. Farhadi, "You only look once:unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016. 6, 38

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *In European conference on computer vision*, pp. 21–37, 2016. 6

[29] S. Suresh and Akhilaa, "Vision: Android application for the visually impaired," *In 2020 IEEE International Conference for Innovation in Technology (INOCON)*, pp. 1–6, 2020. 7, 12

[30] N. Boyko and B. Mandych, "Technologies of object recognition in space for visually impaired people," *In IDDM*, pp. 338–347, 2020. 7, 12

[31] F. M. H. Md. Amanat Khan Shishir, Shahariar Rashid Fahim and T. Farah, "Eye assistant: Using mobile application to help the visually impaired," *In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), IEEE*, pp. 1–4, 2019. 7, 12

[32] M. Awad, J. E. Haddad, T. Mahmoud, E. Yaacoub, and M. Malli, "Intelligent eye: A mobile application for assisting blind people," *In 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pp. 1–6, 2018. 7, 12

[33] Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. (2017) The cifar-10 dataset. Last access to 15 October 2021. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html 7

[34] Catchoom. (2020) Your out-of-the-box image recognition and ar solution. Last access to 8 December 2021. [Online]. Available: https://catchoom.com/image-recognition-ar-craftar/ 7

[35] A. Chuttergoon and L. Nagowah, "Eye guide: An assistive mobile technology for visually impaired persons," *In 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), IEEE*, pp. 198–203, 2020. 7, 12, 101

[36] Melissa Cruz Cossetti. (2018) O que é o talkback? Last access to 10 October 2021. [Online]. Available: https://tecnoblog.net/247247/o-que-e-o-talkback/ 7

[37] Jameson Toole and Dan Abdinoor and team Fritz Labs Incorporated. (2021) Serviço fritz ai. Last access to 12 October 2021. [Online]. Available: https://www.fritz.ai/ 7

[38] News on Sunday. (2018) Braille version of constitution of the republic of mauritius. Last access to 19 October 2021. [Online]. Available: https://defimedia.info/braille-version-constitution-republic-mauritius 7

[39] A. Badave, R. Jagtap, R. Kaovasia, S. Rahatwad, and S. Kulkarni, "Android based object detection system for visually impaired," *In 2020 International Conference on Industry 4.0 Technology (I4Tech), IEEE*, pp. 34–38, 2020. 8, 9, 12

[40] N. Khaled, S. Mohsen, K. E. El-Din, S. Akram, H. Metawie, and A. Mohamed, "In-door assistant mobile application using cnn and tensorflow," *In 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), IEEE*, pp. 1–6, 2020. 8, 11, 12

[41] B. Kuriakose and F. E. Sandnes, "Smartphone navigation support for blind and visually impaired people - a comprehensive analysis of potentials and opportunities," *In International Conference on Human-Computer Interaction*, pp. 568–583, 2020. 8, 12

[42] R. P. Prasetya and F. Utaminingrum, "Triangle similarity approach for detecting eyeball movement," *In 2017 5th International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 37–40, 2017. 8

[43] S. Vaidya, N. Shah, N. Shah, and P. R. Shankarmani, "Real-time object detection for visually challenged people," *In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE*, pp. 311–316, 2020. 8, 12, 36

[44] S. Tosun and E. Karaarslan, "Real-time object detection application for visually impaired people: Third eye," *In 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), IEEE*, pp. 1–6, 2019. 8, 12

[45] Selman Tosun and Enis Karaarslan. (2018) Thirdeye - real-time object detection for visually impaired people. Last access to 12 October 2021. [Online]. Available: https://github.com/selmantsn/Third-Eye-master 9

[46] S. A. Jakhete, A. Dorle, P. Pimplikar, P. Bagmar, and A. Rajurkar, "Object recognition app for visually impaired," *In 2019 IEEE Pune Section International Conference (PuneCon), IEEE*, pp. 1–4, 2019. 9, 12

[47] I. T. Manabu Shimakawa, Kosei Matsushita, C. Okuma, and K. Kiyota, "Smartphone apps of obstacle detection for visually impaired and its evaluation," *Proceedings of the 7th ACIS International Conference on Applied Computing and Information Technology*, pp. 1–6, 2019. 9, 12, 21

[48] Google team. (2021) Advanced guide to inception v3 on cloud tpu. Last access to 24 November 2021. [Online]. Available: https://cloud.google.com/tpu/docs/inception-v3-advanced 9

[49] I. S. Neel Parikh and S. Vahora, "Android smartphone based visual object recognition for visually impaired using deep learning," *In 2018 International Conference on Communication and Signal Processing (ICCSP), IEEE*, pp. 0420–0425, 2018. 10, 12

[50] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, pp. 1345–1359, 2009. 11

[51] Tzutalin. (2021) Labelimg. Last access to 30 January 2022. [Online]. Available: https://github.com/tzutalin/labelImg 11

[52] D. Ahmetovic, D. Sato, U. Oh, T. Ishihara, K. Kitani, and C. Asakawa, "Recog: Supporting blind people in recognizing personal objects," *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020. 11, 12, 21

[53] Microsoft Corporation. (2021) Seeing ai in new languages. Last access to 23 December 2021. [Online]. Available: https://www.microsoft.com/en-us/ai/seeing-ai 14

[54] V. C. R. Inc. (2020) Computer vision for the blind. Last access to 23 December 2021. [Online]. Available: https://www.supersense.app/ 14

[55] Mediate. (2020) Mediate is a research and innovation lab working at the intersection of computer vision and augmented reality. Last access to 23 December 2021. [Online]. Available: https://www.mediate.tech/ 15

[56] V. Labs. (2019) Vision ai for the blind and visually impaired. Last access to 26 December 2021. [Online]. Available: https://www.aipoly.com/ 15

[57] Google. (2019) Google play award winners 2019. Last access to 26 December 2021. [Online]. Available: https://play.google.com/store/apps/editorial_collection/promotion_topic_googleplayawards2019 15

[58] E. T. BV. (2021) Let your camera speak to you. Last access to 26 December 2021. [Online]. Available: https://www.letsenvision.com/envision-app 15

[59] P. S. Rajendran, P. Krishnan, and D. J. Aravindhar, "Design and implementation of voice assisted smart glasses for visually impaired people using google vision api," *In 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE*, pp. 1221–1224, 2020. 17, 20

[60] N. Satani, S. Patel, and S. Patel, "Ai powered glasses for visually impaired person," *nternational Journal of Recent Technology and Engineering (IJRTE)*, 2020. 17

[61] R. Bastomi, F. P. Ariatama, L. Y. A. T. Putri, S. W. Saputra, M. R. Maulana, M. Syai'in, I. Munadhif, A. Khumaidi, M. B. Rahmat, A. S. Setiyoko *et al.*, "Object detection and distance estimation tool for blind people using convolutional methods with stereovision," *In 2019 International Symposium on Electronics and Smart Devices (ISESD), IEEE*, pp. 1–5, 2019. 17

[62] W.-J. Chang, L.-B. Chen, C.-H. Hsu, J.-H. Chen, T.-C. Yang, and C.-P. Lin, "Medglasses: a wearable smart-glasses-based drug pill recognition system using deep learning for visually impaired chronic patients," *IEEE Access*, pp. 17 013–17 024, 2020. 17

[63] C.-H. Chen and M.-F. Shiu, "Smart guiding glasses with descriptive video service and spoken dialogue system for visually impaired," *In 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*, pp. 1–2, 2020. 17

[64] S. Shaikh1, S. F. J. ans Karan Sosa, and P. Nair, "Smart helmet for visually impaired," *International Research Journal of Engineering and Technology (IRJET) 06(04)*, pp. 1846–1849, 2019. 18, 20

[65] R. Weinstein, "Rfid: a technical overview and its application to the enterprise." *IT professional 7.3*, pp. 27–33, 2005. 18

[66] V. V. Meshram, K. Patil, V. A. Meshram, and F. C. Shu, "An astute assistive device for mobility and object recognition for visually impaired people," *IEEE Transactions on Human-Machine Systems 49.5*, pp. 449–460, 2019. 19, 20

[67] A. Dionisi, E. Sardini, and M. Serpelloni, "Wearable object detection system for the blind," *In 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pp. 1255–1258, 2012. 19

[68] M. Murad, A. Rehman, A. A. Shah, S. Ullah, M. Fahad, and K. M. Yahya, "Rfaide – an rfid based navigation and object recognition assistant for visually impaired people," *In 2011 7th International Conference on Emerging Technologies, IEEE*, pp. 1–4, 2011. 19

[69] M. K. Khine and T. T. Aung, "Rfid-based audio guidance cane for blind and visually impaired person," *International Journal of Engineering Research and Technology (IJER)*, pp. 1472–1477, 2014. 19

[70] Evanildo da Silveira. (2013) Identificação a distância. Last access to 30 November 2021. [Online]. Available: https://revistapesquisa.fapesp.br/identificacao-a-distancia/ 19

[71] C. Creusot and A. Munawar, "Low-computation egocentric barcode detector for the blind," *In 2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2856–2860, 2016. 19, 20

[72] B. Calabres, R. Velázquez, C. Del-Valle-Soto, R. de Fazio, N. I. Giannoccar, and P. Viscont, "Solar-powered deep learning-based recognition system of daily used objects and human faces for assistance of the visually impaired," *Energies, 13(22)*, p. 6104, 2020. 19, 20, 21

[73] M. M.Gupta, L. Jin, and N. Homma, *Static and dynamic neural networks: from fundamentals to advanced theory*, ser. ISBN 0-471-21948-7, 2004. 25

[74] M. Babini, "Reconhecimento de padrões lexicais por meio de redes neurais," *Universidade Estadual Paulista (UNESP)*, pp. 1–110, 2006. 25

[75] R. Colom, S. Karama, R. E. Jung, and R. J. Haier, "Human intelligence and brain networks," *Dialogues in clinical neuroscience*, p. 489, 2010. 26

[76] A. R. Jensen, "The g factor: The science of mental ability," *Psicothema, 11(2)*, pp. 445–446, 1999. 26

[77] M. Snyderman and S. Rothman, "Survey of expert opinion on intelligence and aptitude testing," *American Psychologist*, p. 137, 1987. 26

[78] J.-W. Lin, "Artificial neural network related to biological neuron network: a review," *Advanced Studies in Medical Sciences*, pp. 55–62, 2017. 27

[79] AnilK.Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer. IEEE*, pp. 31–44, 1996. 27

[80] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications*, ser. ISBN 978-953-307-243-2, 2011. 28

[81] S. Haykin, *Cognitive dynamic systems: perception-action cycle, radar and radio*, ser. ISBN 978-0-521-11436-3, 2012. 28

[82] I. W. Sandberg, J. T. Lo, C. L. Fancourt, J. C. Principe, S. Katagiri, and S. Haykin, *Nonlinear dynamical systems: feedforward neural network perspectives*, ser. ISBN 0-471-34911-9, 2001. 28

[83] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *towards data science*, pp. 310–316, 2017. 28, 30

[84] L. Zhan, "Implementation of fixed-point neuron models with threshold, ramp and sigmoid activation functions," *IOP Conference Series: Materials Science and Engineering*, p. 012054, 2017. 28

[85] H. Unz, "Oliver heaviside (1850-1925)," *IEEE Transactions on education*, pp. 30–33, 1963. 29

[86] D. Ceccon. (2020) Funções de ativação: definição, características, e quando usar cada uma. Last access to 23 January 2022. [Online]. Available: https://iaexpert.academy/2020/05/25/funcoes-de-ativacao-definicao-caracteristicas-e-quando-usar-cada-uma/ 30

[87] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, pp. 115–133, 1943. 31

[88] S. Herculano-Houzel, "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost," *Proceedings of the National Academy of Sciences*, pp. 10 661–10 668, 2012. 31

[89] Y. Bengio, *Learning deep architectures for AI*, 2009. 31

[90] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," *Conference on learning theory*, pp. 907–940, 2016. 32

[91] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *Journal of cheminformatics*, pp. 1–13, 2017. 32

[92] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE transactions on systems, man, and cybernetics*, pp. 826–834, 1983. 32

[93] P. A. Chunhui Gu, Joseph J. Lim and J. Malik, "Recognition using regions," *In CVPR*, 2019. 36

[94] Ahmed Fawzy Gad. (2020) Faster r-cnn explained for object detection tasks. Last access to 12 November 2021. [Online]. Available: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/ 38

[95] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, "Scalable, high-quality object detectio," *arXiv pré-impressão arXiv:1412.1441*, 2014. 39

[96] Oblivion. (2022) Oblivion about. Last access to 28 June 2022. [Online]. Available: https://www.oblivion.uevora.pt/inicio/ 44

[97] V. U. Vision. (2022) Vision software. Last access to 28 June 2022. [Online]. Available: https://vistalab-vision.readthedocs.io/en/latest/software/index.html 44

[98] Google. (2021) Android studio. Last access to 8 January 2022. [Online]. Available: https://developer.android.com/studio 47

[99] B. Nzivu. (2021) Simple get request using retrofit in android. Last access to 11 June 2022. [Online]. Available: https://www.section.io/engineering-education/making-api-requests-using-retrofit-android/ 48

[100] TensorFlow. (2021) An end-to-end open source machine learning platform. Last access to 11 January 2022. [Online]. Available: https://www.tensorflow.org/ 48

[101] U. Team and J. Martin. (2018) What is text-to-speech technology (tts)? Last access to 11 January 2022. [Online]. Available: https://www.understood.org/articles/en/text-to-speech-technology-what-it-is-and-how-it-works 48

[102] Amazon. (2022) What is speech to text? Last access to 12 February 2022. [Online]. Available: https://aws.amazon.com/pt/what-is/speech-to-text/ 48

[103] J. K. Korpela, *Unicode explained*, ser. 978-0-596-10121-3, 2006. 49

[104] B. Raharjo, *Belajar otodidak membuat database menggunakan MySQL*, ser. 978-602-8758-45-1, 2011. 49

[105] T. P. Group. (2022) What is php? Last access to 1 May 2022. [Online]. Available: https://www.php.net/manual/en/intro-whatis.php 49

[106] T. phpMyAdmin development team. (2022) Bringing mysql to the web. Last access to 1 May 2022. [Online]. Available: https://www.phpmyadmin.net/ 50

[107] C. Team. (2021) What is rest? Last access to 11 June 2022. [Online]. Available: https://www.codecademy.com/article/what-is-rest 50

[108] N. G. (2022) What is ubuntu? a quick beginner's guide. Last access to 12 June 2022. [Online]. Available: https://www.hostinger.com/tutorials/what-is-ubuntu 51

[109] S. Yegulalp. (2022) What is tensorflow? the machine learning library explained. Last access to 12 June 2022. [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html 51

[110] M. Hasan. (2021) The 30 best python libraries and packages for beginners. Last access to 12 June 2022. [Online]. Available: https://www.ubuntupit.com/best-python-libraries-and-packages-for-beginners/ 52

[111] D. Cochard. (2021) Mobilenetssd : A machine learning model for fast object detection. Last access to 27 June 2022. [Online]. Available: https://medium.com/axinc-ai/mobilenetssd-a-machine-learning-model-for-fast-object-detection-37352ce6da7d 52

[112] V. Paradigm. (2022) What is activity diagram? Last access to 19 June 2022. [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ 55

[113] P. Gorbachenko. (2021) What are functional and non-functional requirements and how to document these. Last access to 21 June 2022. [Online]. Available: https://enkonix.com/blog/functional-requirements-vs-non-functional/ 61

[114] JavaDoc. (2022) Class multipartuploadrequest. Last access to 29 September 2022. [Online]. Available: https://www.javadoc.io/doc/net.gotev/uploadservice/3.3/net/gotev/uploadservice/MultipartUploadRequest.html 80

[115] A. S. Documentation. (2022) Downloadmanager. Last access to 30 September 2022. [Online]. Available: https://developer.android.com/reference/android/app/DownloadManager 85

[116] ——. (2022) Tonegenerator. Last access to 30 September 2022. [Online]. Available: https://developer.android.com/reference/android/media/ToneGenerator 87

[117] ——. (2022) Class intentintegrator. Last access to 6 October 2022. [Online]. Available: https://zxing.github.io/zxing/apidocs/com/google/zxing/integration/android/IntentIntegrator.html 89

[118] D. Nogare. (2020) Performance de machine learning – matriz de confusão. Last access to 7 October 2022. [Online]. Available: https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/?fbclid=

IwAR1pqJUF3wGPYTobQMYRXSjtQWt14JNoKTRwq20bE8B3iRvgSNYWccPqfs8 94

[119] C. Dilmegani. (2022) What is data augmentation? techniques, examples and benefits. Last access to 8 October 2022. [Online]. Available: https://research.aimultiple.com/ data-augmentation/ 96

[120] L. Torrey and J. Shavlik, "Transfer learning," *IGI global*, pp. 242–264, 2010. 98

[121] P. Baheti. (2022) A newbie-friendly guide to transfer learning. Last access to 9 October 2022. [Online]. Available: https://www.v7labs.com/blog/transfer-learning-guide 98