Instituto Politécnico de Tomar

Escola Superior de Tecnologia

Rafael Tomé Gonçalves Escudeiro

Specchio – Espelho Inteligente

Relatório de Projeto de Mestrado

Orientador: Professor Luís Oliveira, Instituto Politécnico de Tomar

Co-orientador: Professor Luís Almeida, Instituto Politécnico de Tomar

Relatório apresentado ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática – Internet das Coisas

Dedicatória

Aos meus pais Carlos e Fernanda pela constante dedicação.

Ao meu irmão Duarte pela constante companhia.

Aos meus amigos pela força e coragem.

Aos meus orientadores pela sua ajuda e suporte.

Resumo

O atual mundo tecnológico disponibiliza uma grande quantidade de informação e de forma muito rápida, sendo fundamentais meios de interação simples, personalizados e eficientes. A evolução da Internet das Coisas (*IoT*), a computação na *cloud*, a massificação dos dispositivos móveis e a constante necessidade de estar *on-line* sugerem novos desafios tecnológicos.

Este trabalho apresenta uma solução tecnológica capaz de disponibilizar informação de forma natural, atualizada e personalizada a um utilizador enquanto este executa tarefas do quotidiano em frente a um espelho. Trata-se de um dispositivo que além de refletir a cara do utilizador, fornece diverso conteúdo informativo, como notícias, horas, informação meteorológica, emails, sugestões, etc. Distingue-se pelo facto de adaptar o conteúdo informativo ao utilizador presente, recorrendo a técnicas de identificação de imagem facial por aprendizagem (*machine learning*). Na presença de vários utilizadores em frente ao espelho, a informação apresentada é limitada à autorizada, evitando divulgação de conteúdos informativos sensíveis e privados (ex: *emails*, mensagens).

O trabalho demonstra como programadores sem conhecimentos avançados em aprendizagem podem ainda assim, desenvolver módulos baseados em *Machine Learning* recorrendo a ferramentas da Cloud e integrá-los nas suas aplicações como "caixa pretas". Descreve-se em particular, como treinar e utilizar de forma eficaz modelos específicos utilizando ferramentas como o Firebase, ML Kit e Cloud AutoML da Google. Adicionalmente demonstra-se como aplicar tecnologias de aprendizagem em dispositivos móveis cujo poder de processamento é reduzido. O resultado é um protótipo de "espelho inteligente", que integra tecnologias de reconhecimento baseada em *machine learning* para dispositivos móveis, com suporte numa plataforma de Internet das Coisas utilizando o sistema operativo, *Android Things*. A taxa de sucesso dos testes de reconhecimento facial ronda os 70%, e é considerada promissora, no entanto, sugere um treino com uma base de dados mais alargada e testes de usabilidade em trabalhos futuros.

Palavras-chave

Machine Learning, Reconhecimento Facial, Internet of Things, Android

Abstract

The current technological world provides a large amount of information and very quickly, thus, it is fundamental simple, personalized and efficient means of interaction. The evolution of the Internet of Things, the cloud computing, the massification of mobile devices and the constant need to be online bring new technological challenges.

This work presents a technological solution capable of providing natural, updated and personalized information to a user while performing daily tasks in front of a mirror. It is a device that not only reflects the user's face, but also provides various informative content such as news, hours, weather information, emails, suggestions, etc. It is distinguished by the fact that it adapts the informative content to the present user, using techniques of image's facial identification through machine learning. In the presence of several users in front of the mirror, the information presented is limited to the authorized, avoiding the exposure of sensitive and private informative content (e.g. emails, messages).

The work demonstrates how programmers without advanced learning skills can still develop Machine Learning-based modules using Cloud tools and integrate them into applications as "black box". It describes how to effectively train and use specific models using tools such as Google's Firebase, ML Kit and Cloud AutoML. Another contribution is to demonstrate how to apply learning technologies to mobile devices with reduced processing power. The result is a "smart mirror" prototype, which integrates machine learning based on recognition technologies for mobile devices, supported on an Internet of Things platform running Android Things operating system. The success rate of facial recognition tests is 70%, and is considered promising, however, suggests a training with a larger database and further usability tests in future work.

Keywords

Machine Learning, Facial Recognition, Internet of Things, Android

Índice

Índice de Figuras

Lista de Acrónimos e Siglas

IoT	*Internet of Things*
RGPD	Regulação Geral sobre a Proteção de Dados
MTCNN	*Multi-Task Cascaded Convolutional Neural Network*
P-Net	*Proposal Network*
R-Net	*Refine Network*
O-Net	*Output Network*
API	*Application Programming Interface*
REST	*Representational State Transfer*
AutoML	*Automated Machine Learning*
Kit ML	*Kit Machine Learning*
SDK	*Android Software Development Kit*
DNN	*Deep Neural Network*
CNN	*Convolutional Neural Network*
NAS	*Neural Architecture Search*

Capítulo 1

Introdução

Este trabalho tem como objetivo demonstrar a deteção e reconhecimento facial aplicados a uma solução de "espelho inteligente". Atualmente existem diversas tecnologias de reconhecimento de imagem (Gorodnichy, Granger, & Radtke, 2014) como também diversas soluções de espelho inteligente (Jose, Chakravarthy, Jacob, Ali, & D'souza, 2017) (A., T., D., & Dokhale, 2018), porém as aplicações existentes não integram ambas as componentes de forma eficiente e eficaz. O "espelho inteligente", designado de "Specchio", além de refletir a imagem da pessoa, propõe uma integração eficiente para apresentação de informação personalizada. O fator inovador reside no facto do Specchio adaptar o seu conteúdo informativo de acordo com o utilizador que se apresente em frente. Caso detete várias pessoas em simultâneo, a apresentação de informações sensíveis e privadas são limitadas.

Specchio utiliza um dispositivo de processamento e um monitor. O efeito espelho de Specchio baseia-se na utilização de uma película *Two Way Mirror Film*[1] aplicada sobre o monitor. Existem diversos projetos e bibliotecas de espelhos inteligentes possíveis de implementar (Paykar, s.d.), sendo o mais conhecido o MagicMirror2[2]. Esta plataforma tem como vantagens o facto de ser *open source* e mantida por uma grande comunidade de programadores. Outra característica que a distingue das restantes é o facto de ser modular e destes componentes poderem ser desenvolvidos e integrados pela comunidade.

O espelho inteligente adapta os seus conteúdos de acordo com o utilizador presente utilizando técnicas de reconhecimento facial baseadas em imagem. O reconhecimento de imagens pode ser realizado através de técnicas clássicas de visão por computador e mais recentemente a técnicas de aprendizagem (*Computer Vision* e *Machine Learning*).

Ambas estão intrinsecamente relacionadas, pois *computer vision* utiliza técnicas de *machine learning* e algumas técnicas de *machine learning* são desenhadas especificamente para *computer vision*. No entanto *Machine Learning* é mais abrangente,

[1] https://www.twowaymirrors.com/two-way-mirror-film/ [acedido em 13/02/2019]
[2] https://magicmirror.builders/ [acedido em 13/02/2019]

uma vez que os seus algoritmos e modelos estatísticos são utilizados na execução das mais variadas tarefas científicas. Estes algoritmos não dependem apenas das instruções, mas recorrem ao reconhecimento de padrões e inferência. Indicando um conjunto correto de *inputs*, *outputs*, classificadores e recursos, o algoritmo adapta-se de forma eficaz à tarefa pretendida em qualquer área tecnológica.

Este capítulo está organizado em cinco secções, Contextualização, Motivação, Objetivos, Proposta de Desenvolvimentos e Análise de Construção, Estrutura do Relatório.

1.1. Contextualização

Algumas das tarefas que maior parte das pessoas faz no decorrer do seu dia-a-dia é ler notícias, ver a meteorologia, usar o espelho para a sua higiene pessoal, entre muitas outras. E porque não juntar tais tarefas numa única, potencializando o tempo disponível da pessoa? O conceito é apresentar essas informações de notícias e/ou meteorologia no espelho. Trata-se de um "espelho inteligente", pois é possível interagir com ele e configurá-lo de forma natural de acordo com o perfil do utilizador.

Privacidade e segurança dos dados pessoais são atualmente preocupações da sociedade global, no entanto criam desafios no que diz respeito ao desenvolvimento de aplicações seguras, de fácil utilização e intuitiva.

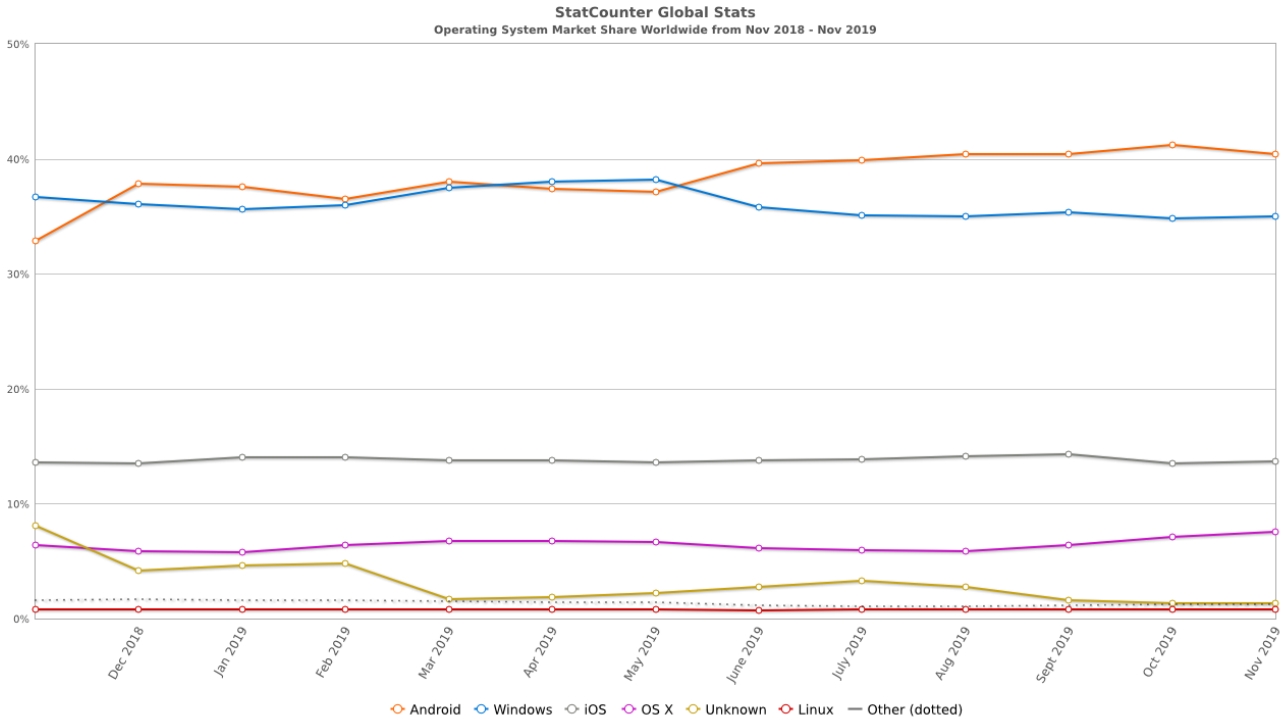
1.2. Motivação

Novas abordagens de transmissão de informações digitais estão agora a ser desenvolvidas face à enorme quantidade de informação que se transfere pela *web*. Uma delas é transmitir a informação focada num utilizador específico, utilizando para isso ferramentas de reconhecimento de imagem. Atualmente diversas são as áreas que se implementam tecnologias de reconhecimento facial, tais como segurança, publicidade, desporto, saúde e muitas outras. Presentemente o reconhecimento facial, já é utilizado para identificação de terroristas, aceder a conteúdo protegido, autenticação em sistemas, receção de publicidade direcionada, etc.

No caso da saúde, já existem exames que recorrem ao reconhecimento de imagem para deteção de determinadas doenças de difícil deteção, tais como as doenças cancerígenas (Kourou, Exarchos, Exarchos, Karamouzis, & Fotiadis, 2015). Desde o

lançamento do Face ID[3] no iPhone em 2017, a implementação desta funcionalidade nos dispositivos móveis tem vindo a crescer exponencialmente. Através dela, além de desbloquearmos o dispositivo móvel, conseguimos também realizar pagamentos e abrir determinadas aplicações. Porém, apesar de todas as aplicações e soluções já existentes de reconhecimento facial, ainda não foi eleita uma tecnologia de desenvolvimento preferencial para a criação destas soluções. Um dos propósitos deste trabalho será servir de estudo para chegar a uma possível conclusão de qual a tecnologia de identificação de utilizador recomendada para plataformas *IoT*. A constante evolução e utilização de *machine learning* em diversos projetos atuais, sugere desenvolver o Specchio com suporte de *machine learning*, porém existem outras opções que não recorrem ao *machine learning* para funcionalidades de tratamento de imagem, como é o caso das ferramentas de visão por computador, tais como o OpenCV abordado no tópico 2.3.7.

Atualmente, a maioria das pessoas do mundo civilizado possuem e utilizam no seu dia-a-dia um *smartphone*. Estima-se que existam 2.5 mil milhões de dispositivos Android ativos, segundo um tweet[4] da própria Android. Este sistema operativo é largamente utilizado pela comunidade, possui grande quota do mercado de sistemas operativos e no que refere a *mobile* (ver na Figura 1).



[3] https://support.apple.com/pt-pt/HT208109 [consultado em 07/05/2019]
[4] https://twitter.com/android/status/1125822326183014401 [consultado em 07/05/2019]

Figura 1 - Quota Sistemas Operativos[5]

Devido à elevada utilização de *smartphones* Android, o Specchio integra uma componente *mobile* Android de forma a que a acessibilidade ao Specchio se torne mais simples e prática. Como o espelho não possui mecanismos de configuração (botões, teclados, etc) de fácil acesso, recorre-se assim a uma aplicação móvel através de um *smartphone* para realizar essa tal configuração.

1.3. Objetivos

Com este trabalho, pretende-se desenvolver um espelho inteligente, cujo objetivo principal é facilitar as tarefas diárias que os utilizadores realizam no seu dia-a-dia. Este espelho além de refletir a cara do utilizador, fornece diverso conteúdo informativo, como notícias, dicas, meteorologia, entre muitos outros. Tem a especial característica de adaptar o seu conteúdo informativo consoante o utilizador que o está a usar, utilizando para isso técnicas de reconhecimento de imagem. Além disso, se tal conteúdo informativo for sensível e privado, como emails e/ou mensagens pessoais, e for detetado mais de um utilizador frente ao espelho, este adapta-se e fornece apenas informação autorizada para esses utilizadores, como as notícias e a meteorologia que não são informações sensíveis. Além do espelho, outro dos objetivos a desenvolver é a criação de uma aplicação *mobile* cuja finalidade é a configuração do espelho que o utilizador acede. A nível da arquitetura de sistema, pretende-se encontrar uma solução fiável para os dias de hoje, tanto a nível de software como a nível de hardware.

Em suma, os objetivos desta investigação consistem em:

- Criação de um *dashboard* informativo dinâmico (espelho)
- Criação de uma aplicação *mobile* para registo e configuração do espelho
- Implementação de uma tecnologia de reconhecimento facial no espelho
- Adaptação do conteúdo informativo ao utilizador identificado

1.4. Proposta de Desenvolvimentos e Análise de Construção

[5] Imagem retirada de https://gs.statcounter.com/os-market-share/ [consultado em 02/12/2019]

No desenvolvimento de Specchio, houve a necessidade de investigar a tecnologia de reconhecimento de imagem que mais se adequava ao âmbito do projeto. Vários parâmetros foram considerados no estudo da fiabilidade da tecnologia: um dos mais importantes é o tempo de treino para reconhecimento de determinada imagem. Outro parâmetro importante é a precisão, isto é, para uma dada pessoa registada, qual a percentagem de faces reconhecidas no total de faces testadas.

Nos projetos onde existem abordagens de *machine learning* e *computer vision*, a avaliação do modelo proposto depende do *dataset* [6] em estudo. A qualidade e quantidade de dados presentes nesse *dataset* são um fator essencial para que determinado algoritmo/tecnologia funcione de acordo com o esperado.

O desenvolvimento do espelho inteligente, Specchio, pressupõe um projeto tanto a nível de *hardware* como a nível de *software* (ver Figura 2).

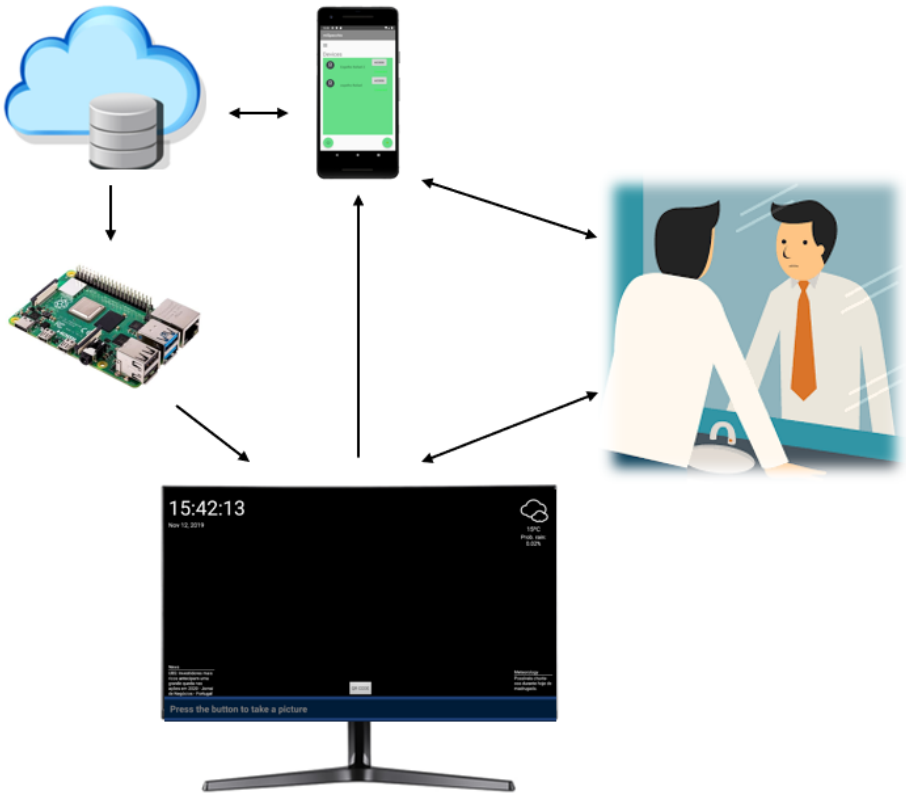


Figura 2 – Arquitetura simplificada de Specchio

Sendo um espelho inteligente, o dispositivo deve refletir a imagem pessoa. Esta funcionalidade é realizada através da colocação de uma película refletiva e adesiva sobre

[6] Conjunto de dados

o monitor, designada *Two Way Mirror*. Essa película tem como característica espelhar de um dos lados e do outro permitir que a luz atravesse.

O processamento de reconhecimento de imagem e visualização do *dashboard* dinâmico, utiliza um Raspberry Pi[7] ligado ao monitor. A opção pelo Raspberry Pi deve-se ao facto de ser um dispositivo barato, fiável e utilizado em diversos projetos *IoT*. Existem diversas alterativas de dispositivos *IoT* possíveis de utilizar no entanto para o sistema operativo escolhido só existem duas, sendo o Raspberry Pi a alternativa mais barata e acessível como abordado no tópico 2.4.

Na configuração do seu espelho inteligente, utiliza-se um *smartphone* com uma aplicação Android desenvolvida para o efeito.

Dado o domínio do ecossistema Android, a aplicação do smartphone foi desenvolvida em android, assim como o sistema operativo escolhido para Raspberry Pi é Android Things[8]. Estas opções tiveram em consideração a rápida prototipagem da solução em função dos recurso on-line.

1.5. Estrutura do Relatório

O presente relatório está organizado em seis capítulos, sendo eles: Introdução, Estado da Arte, Fundamentos Teóricos, Metodologia, Testes e resultados, Conclusões e trabalho futuro.

No segundo capítulo são apresentados diversos temas, começando por um descritivo de dados pessoais e privacidade. Depois disso é apresentado as fases de um reconhecimento de uma face como também os diversos serviços multiplataforma usados nos dias de hoje. Por fim, é abordado o sistema operativo Android que é utilizado em dispositivos *IoT*.

No terceiro capítulo estão expostos os diferentes fundamentos teóricos de uma forma detalhada pelo que a sua compreensão é fulcral para entender o projeto e metodologia desenvolvida.

No quarto capítulo está exposto a metodologia implementada neste projeto, as etapas de reconhecimento facial, o reconhecimento de objetos, as diferentes componentes do projeto e por fim a arquitetura geral e fluxos da aplicação e sistema.

[7] https://www.raspberrypi.org/ [consultado em 22/06/2019]
[8] https://developer.android.com/things [consultado em 22/06/2019]

O quinto capítulo é referente aos testes e resultados obtidos. São apresentados os detalhes de avaliação, a eficácia e a matriz de confusão do modelo obtido.

O último capítulo é referente às conclusões deste relatório. É abordado também os trabalhos futuros que podem ser feitos de forma a melhorar a investigação.

Capítulo 2

Estado da Arte

Atualmente, vivemos na "Era dos Ecrãs". Grande parte da informação chega às pessoas de forma digital, através de um clique ou de um comando de voz. Este paradigma tem revolucionado o dia-a-dia das pessoas, quer no trabalho, quer na sua vida pessoal. Porém, tal simplicidade de acesso originou também fragilidades na segurança e privacidade dos dados pessoais de cada pessoa, sendo este um objeto de estudo por diversos investigadores e profissionais de forma a colmatar esta situação. Apesar de já existirem diversos progressos na área, a verdade é que ainda existe muito desenvolvimento para realizar, especialmente a área de reconhecimento de imagem. Neste capítulo pretendo demonstrar o estado da arte. Este capítulo está organizado em quatro secções, Dados Pessoais e Privacidade, Face Recognition, Serviços Multiplataforma e Android Things. A secção 2.1 demonstra o que são os dados pessoais e privacidade e qual é a sua implicância. Na secção 2.2, está retratado o tema de Face Recognition. Na secção 2.3 abordam-se as diferentes tecnologias utilizadas pelos investigadores e na secção 2.4 aborda-se o sistema operativo Android Things.

2.1. Dados Pessoais e Privacidade

Segundo a Comissão Europeia[9], dados pessoais são informação relativa a uma pessoa viva, identificada ou identificável. Além disso, o conjunto de informações distintas que possibilitam a identificação de uma pessoa também são consideradas como dados pessoais. Os dados pessoais atualmente são dos recursos mais valiosos de uma pessoa ou entidade, pois facilmente podem causar problemas devido a utilização imprópria e incorreta. No nosso caso do espelho inteligente, a pessoa será identificada com recurso a tecnologias de reconhecimento de imagem. O conteúdo que será transmitido é adaptável consoante o utilizador, podendo este possuir dados de informação públicos, como notícias, ou mais sigilosos como é o caso do nome, email, dados de localização, entre outros. Durante o desenvolvimento deste trabalho estes diferentes tipos de dados serão características a ter em conta. Outro fator importante no desenvolvimento do Specchio para trabalho futuro será o cumprimento com o novo Regulamento Geral

[9] https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en [consultado em 09/02/2019]

sobre a Proteção de Dados (RGPD)[10], que indica como devem ser tratados os dados pessoais. Segundo o RGPD, o utilizador terá o direito de ser esquecido, de transferir os dados para outra entidade, de opor-se ao processamento de dados e adicionar e/ou corrigir os dados pessoais.

A privacidade do utilizador consiste na salvaguarda da sua identidade pessoal, tendo este o direito da reserva de informações e vida pessoal. Porém se analisarmos a sociedade de atualmente, concluímos que a privacidade é assunto discutível com vantagens e desvantagens, onde qualquer pessoa transporta um smartphone ligado à internet e com câmara cujo acesso pode ser remoto sem a pessoa reparar.

2.2. Face Recognition

Para que Specchio funcione de acordo com a expectativa do utilizador, Specchio necessita de realizar a tarefa de identificar a face de um utilizador perante uma coleção já existente de faces. Se houver uma correspondência entre a face detetada e a base de dados, é iniciada assim a autenticação. Face Recognition é hoje em dia uma das áreas de investigação mais complexas e promissoras do mercado, pois o seu leque de usabilidade é enorme. Existem diversos tipos de sistemas, funções e variáveis que tentam obter a eficácia máxima no reconhecimento.

Uma das vantagens da implementação de reconhecimento facial numa solução tecnológica é a automação. Graças ao rápido processamento e integração contínua, o reconhecimento facial torna agora determinadas tarefas automatizadas. O reconhecimento manual pode possuir falhas pois é realizado pelo ser humano, que devido ao cansaço ou distração pode reconhecer de forma errada, enquanto que uma máquina pode funcionar 24 horas por dia com uma reduzida probabilidade de erros. Velocidade e alta precisão são vantagens do reconhecimento facial automático, sendo resultado de um constante desenvolvimento por parte dos investigadores, tornando assim o reconhecimento de dia para dia mais rápido e preciso. O facto de o reconhecimento ser *contactless*[11] evita diversos problemas como o caso da sujidade em terminais de impressão, com vista à autenticação/identificação de pessoas. O reconhecimento facial possui algumas lacunas no seu âmbito. Além do elevado grau de processamento requerido para reconhecer uma face, existe também os problemas de ângulo da face e da sua

[10] https://www.sage.com/pt-pt/rgpd/ [consultado em 09/02/2019]
[11] Sem uso de contacto

luminosidade que pode dificultar a precisão como também a existência de dados faciais variáveis, como o caso da barba ou cabelo, que influenciam o reconhecimento de uma pessoa. Tais problemas (designados "ruídos" no processamento de imagem) podem ser resolvidos dependendo da *performance* do modelo para o reconhecimento de imagem.

Para que haja testes e comparações na área científica, muitos investigadores utilizam o *dataset* Labeled Faces in the Wild Home[12] (Figura 3) para compararem a eficácia dos seus algoritmos. Este *dataset* possui 13233 imagens com faces de 5749 pessoas distintas. Um dos modelos com maior taxa de sucesso neste *dataset* é o FaceNet (Schroff, Kalenichenko, & Philbin, 2015), sendo a sua precisão de correspondência de imagens neste *dataset*[13] de 99.63%. Face Recognition pode ser divido em duas etapas, Face Detection e Face Identification, descritas nas secções seguintes.

Figura 3 - Faces da Wild Home Dataset[14]

2.2.1. Face Detection

Face Detection é a primeira fase do Face Recogniton. O principal objetivo desta etapa é distinguir faces humanas numa imagem, relativamente a outros objetos que lá possam estar.

[12] http://vis-www.cs.umass.edu/lfw/ [consultado em 09/02/2019]
[13] https://paperswithcode.com/paper/facenet-a-unified-embedding-for-face [consultado em 09/02/2019]
[14] Imagem retirada de http://vis-www.cs.umass.edu/lfw/ [consultado em 08/03/2019]

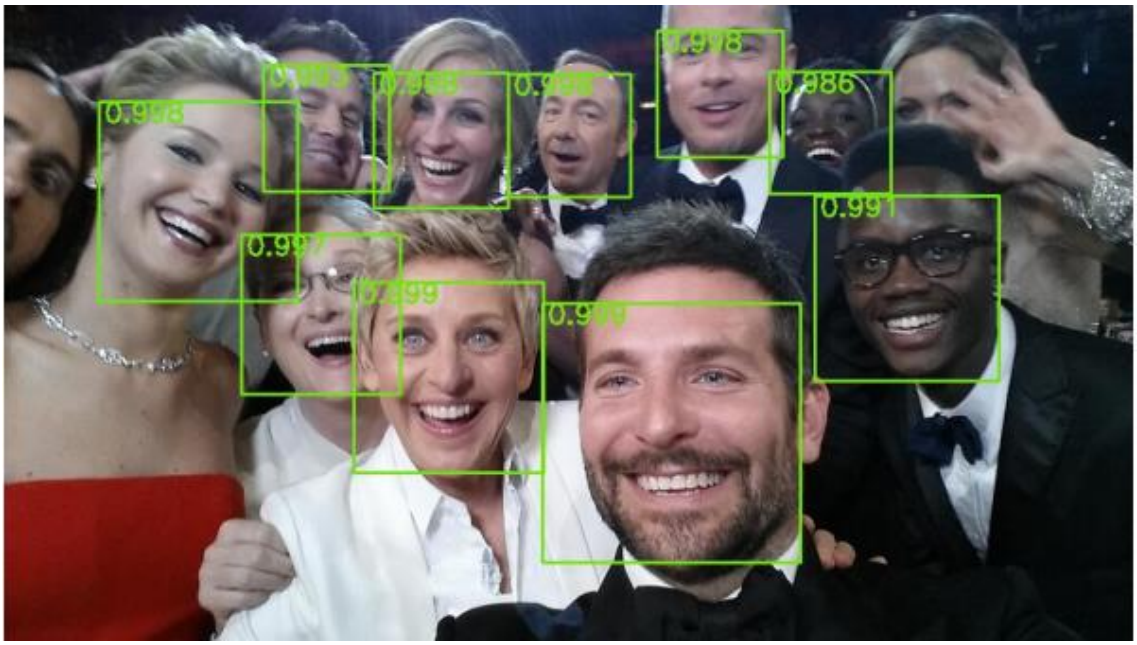


Figura 4 - Deteção de Caras[15]

Face Detection não é uma área recente de investigação, já há muitos anos que a comunidade científica tem realizado progressos neste domínio.

Existem duas abordagens dos métodos Face Detection, os métodos clássicos de visão por computador e processamento digital de imagem e, os métodos baseados em aprendizagem (*deep learning*). Nos clássicos estes dividem-se em dois tipos, os métodos baseados em *features* e os métodos baseados em *templates* de imagens.

Até aos meados de 2001, as tecnologias e algoritmos para deteção de caras eram computacionalmente exigentes e de precisão limitada. Paul Viola e Michael Jones desenvolveram um novo algoritmo que além de ter uma precisão elevada, também funcionava em *realtime* (Viola & Jones, 2001). Essa *framework* ficou conhecida como *The Viola-Jones Face Detection*. O artigo refere que as *features* são selecionadas através do algoritmo *AdaBoost* (Schapire, 1990), sendo os modelos organizados de forma hierárquica de acordo com a sua complexidade (como uma cascata). De forma resumida o algoritmo *AdaBoost* provém da palavra *Adaptive Boosting* e consiste no ajuste das classificações em relação a instâncias anteriormente classificadas, pelo que em cada iteração o peso de cada exemplo classificado incorretamente aumente. Esta *framework* coleciona pequenas *features* da face que juntas criam um classificador robusto. Segundo

[15] Imagem retirada de https://towardsdatascience.com/an-intro-to-deep-learning-for-face-recognition-aa8dfbbc51fb [consultado em 08/02/2019]

esse artigo, os classificadores resultantes mais simples focam-se nas áreas onde a probabilidade de existir uma face é reduzida enquanto que os classificadores mais complexos focam-se nas áreas prováveis de existir uma face. O detetor de faces desse método designa-se de *Detector Cascade* e consiste numa sequência de classificadores simples para complexos. Uma das características destes métodos baseados por *features* é o facto de serem rápidos e eficazes, sendo utilizados durante décadas.

Relativo aos métodos baseados em imagens, estes têm como foco localizar e extrair automaticamente as faces de uma imagem completa, utilizando para isso algoritmos baseados em correlação.

As abordagens mais recentes para deteção de caras recorrem a técnicas de aprendizagem (*deep learning)* entretanto aperfeiçoadas com base no atual poder de cálculo. A mais popular é a *Multi-Task Cascaded Convolutional Neural Network* (MTCNN) (Zhang, Zhang, Li, & Qiao) famosa pelos seus excelentes resultados e pelo facto de conseguir detetar determinados pontos de referência faciais, como os olhos e a boca. A rede neuronal utiliza uma estrutura em cascata dividida em três redes ou modelos. O primeiro passo é redimensionar a imagem para diferentes tamanhos, processo este chamado *Image Pyramid*. O primeiro modelo indica as regiões faciais candidatas (*Proposal Network* ou P-Net) e de seguida o segundo modelo filtra as *bounding boxes* (*Refine Network* ou R-Net), eliminando desta forma os possíveis falsos candidatos. Por último, o terceiro modelo propõe os pontos de referência faciais (*Output Network* ou O-Net), as pequenas regiões da face com mais detalhe, como a boca e os olhos. Assim, os métodos que recorrem ao *deep learning* baseados nesta abordagem são compostos por três etapas, como demonstrado na Figura 5.

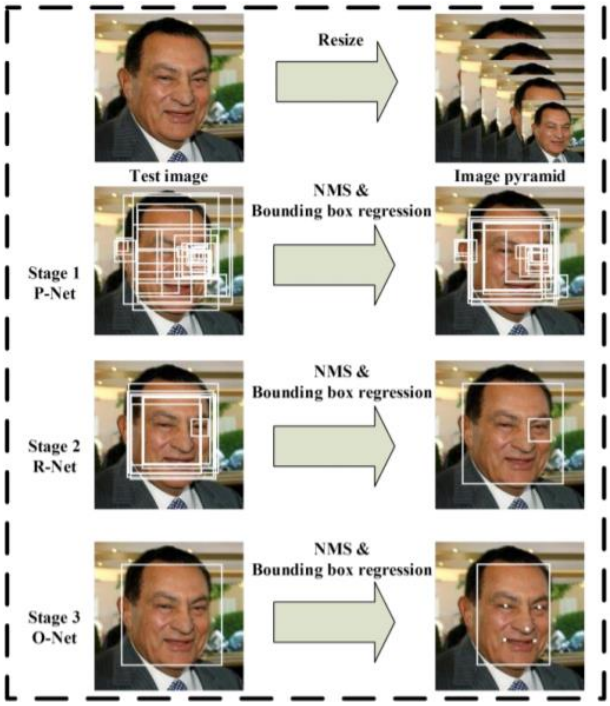


Figura 5 - Etapas do Multi-Task Cascaded Convolutional Neural Network[16]

Proveniente do mesmo artigo, a Figura 6 representa a arquitetura do modelo proposto, onde P-Net tem como saída as faces candidatas, o R-Net tem como saída a posição da cara e o O-Net tem como saída a posição dos elementos com mais detalhe, caso da boca e dos olhos.

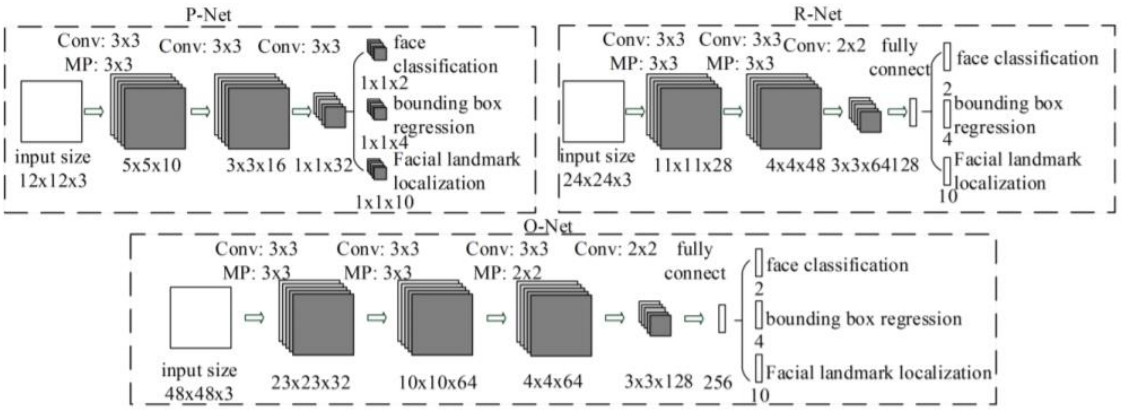


Figura 6 - Arquitetura da Multi-Task Cascade Convolutional Neural Network[17]

A comunidade científica realiza competições entre investigadores para descobrirem qual é o melhor método e algoritmo para a identificação de caras. Um dos desafios colocados à comunidade científica é o *ImageNet Large Scale Visual Recognition*

[16] Imagem retirada do artigo (Zhang, Zhang, Li, & Qiao)
[17] Imagem retirada do artigo (Zhang, Zhang, Li, & Qiao)

Challenge[18], que é um *dataset* de imagens devidamente estruturado para que os investigadores possam por à prova os seus desenvolvimentos. Esse desafio no ano de 2012 sofreu uma reviravolta, pois uma equipa de estudantes de Toronto apresentou uma nova forma de detetar caras, utilizando para isso *Deep Convolutional Neural Networks* (Krizhevsky, Sutskever, & Hinton), superando assim todas as outras metodologias utilizadas até então, tais como *Fisher Vectors* (Simonyan, Parkhi, Vedaldi, & Zisserman) e *Support Vector Machines* (Phillips). Uma *Deep Convolutional Neural Network*[19] é uma rede neuronal artificial profunda que tem como principal finalidade classificar imagens, agrupá-las por similaridade e reconhecer objetos numa dada imagem ou cenário (tópico abordado na secção 3.2). Pode ser aplicada ao processamento de textos, análise de espectrogramas de som, identificação de vírus, etc, sendo uma abordagem muito promissora. A comunidade científica e as empresas elegeram a tecnologia de *Deep Learning* como linha orientadora para futuros desenvolvimentos tecnológicos nas mais variadas áreas tecnológicas e aplicacionais.

2.2.2. Face Identification

Face Identification é a segunda fase do Face Recognition. O principal objetivo desta etapa é estabelecer uma correspondência entre uma dada imagem e conjunto de imagens relacionadas estandos estas integradas numa grande base de dados. Trata-se de processo de mapeamento. A face detetada é sujeita assim a um processo de correspondência com o *dataset* em utilização. Quanto maior a probabilidade dessa correspondência, maior é a certeza de a face pertencer à pessoa correspondida, sendo assim a pessoa identificada corretamente.

[18] https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/ [consultado em 10/02/2019]
[19] https://skymind.ai/wiki/convolutional-network [consultado em 10/02/2019]

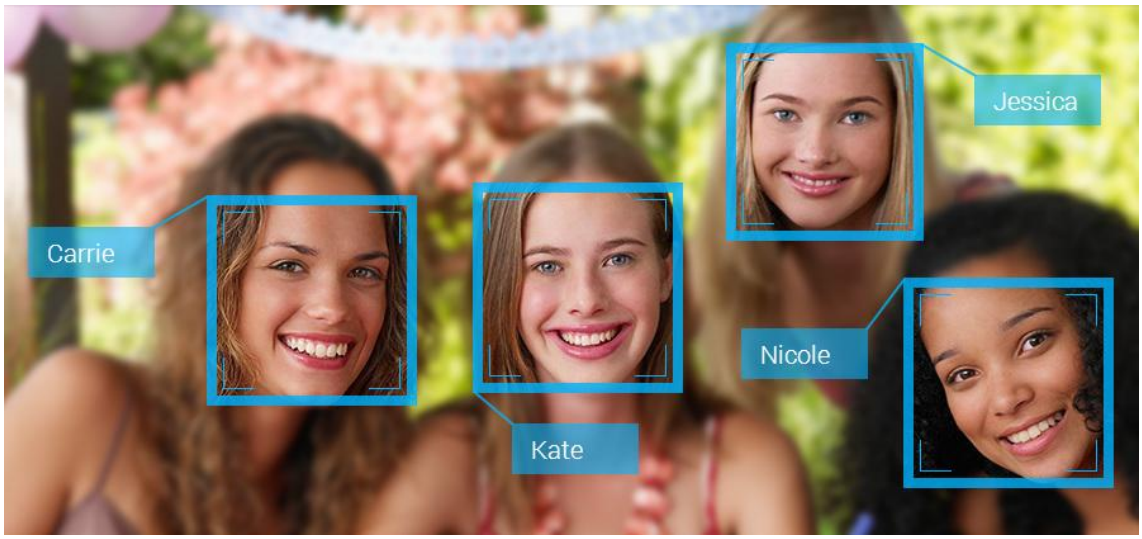


Figura 7 - Identificação de Pessoas[20]

2.3. Serviços Multiplataforma

Atualmente, existem diversas possibilidades de implementar um sistema de reconhecimento de imagem num projeto tecnológico. Alguns serviços são fornecidos por companhias tecnológicas de renome. Todas elas fornecem uma API (*Application Programming Interface*) através da qual os investigadores podem submeter as suas imagens e/ou vídeos para que a plataforma analise. Um dos objetivos do desenvolvimento do Specchio será utilizar uma dessas tecnologias de forma a que o reconhecimento seja eficaz. A maioria das empresas não revela detalhes das suas técnicas de reconhecimento facial no entanto pressupõe-se que recorram às *Deep Convolutional Neural Networks* pois são aquelas que publicamente demonstram os melhores resultados.

2.3.1. Amazon Rekognition

Proveniente da Amazon, Amazon Rekognition[21] facilita a implementação de análises de imagens e vídeos a projetos de software. Abrange diversas funcionalidades, tais como detetar e reconhecer texto em imagens, detetar conteúdo não seguro, reconhecer celebridades, análise facial, entre muitas outras, das quais se destaca uma em especial, o

[20] Imagem retirada de https://towardsdatascience.com/an-intro-to-deep-learning-for-face-recognition-aa8dfbbc51fb [consultado em 23/07/2019]
[21] https://aws.amazon.com/rekognition/ [consultado em 11/02/2019]

reconhecimento facial. Na Figura 8 está representado o fluxo do reconhecimento de um utilizador a partir de uma imagem obtida em *realtime*.

Figura 8 - Amazon Rekognition Verificação de Utilizadores[22]

2.3.2. Watson Visual Recognition

Oriundo da IBM, o Watson Visual Recognition[23], que além de ter a capacidade de classificar qualquer conteúdo visual, dá a possibilidade de criar os nossos próprios classificadores e *datasets*.

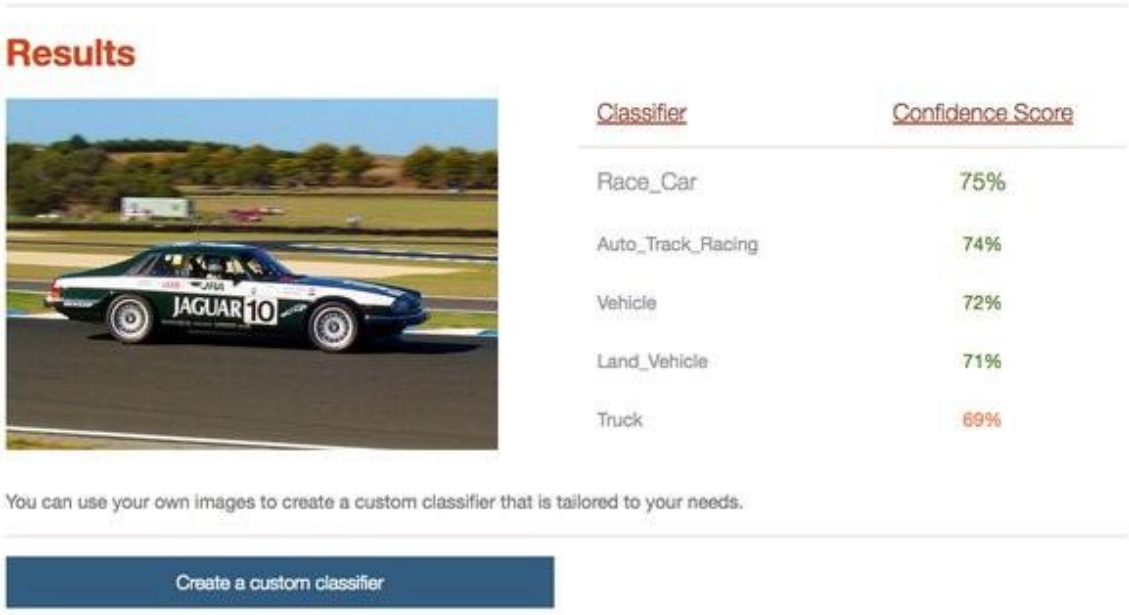


Figura 9 - Watson Online Demo[24]

[22] Imagem retirada de https://aws.amazon.com/pt/rekognition/ [consultado em 11/02/2019]
[23] https://www.ibm.com/watson/services/visual-recognition/ [consultado em 11/02/2019]
[24] Imagem retirada de https://www.ibm.com/watson/services/visual-recognition/ [consultado em 11/02/2019]

2.3.3. Azure Face API

A Microsoft apresenta Azure Face API[25]. Tem como intuito fornecer algoritmos para detetar, analisar e reconhecer caras em imagens. Algumas das funcionalidades possíveis nesta *API* é a deteção e verificação de caras, pesquisa de caras similares, identificação de pessoas e categorização de faces.

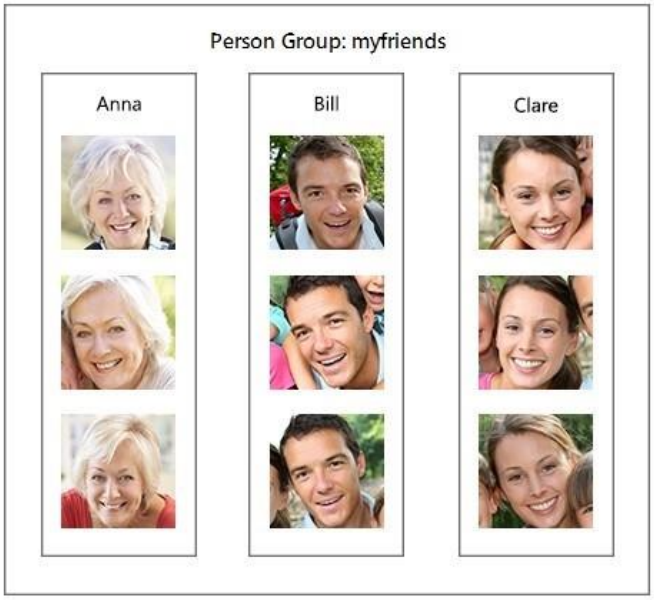


Figura 10 - Distribuição de caras[26]

2.3.4. Tensorflow

TensorFlow[27] é uma biblioteca *open source* de *machine learning* para investigação e produção suportada pela Google. Flexibilidade e rapidez são características inerentes a esta comunidade. Por ser *open source*, a sua taxa de implementação e desenvolvimento é bastante elevada, como também o tamanho da comunidade que a utiliza. A lista de funcionalidades é extensa, sendo possível realizar reconhecimento de imagens com ela. Graças à implementação do Keras[28] por parte do TensorFlow. Keras é uma biblioteca *open source* de uma rede neuronal, escrita na linguagem Python[29].

[25] https://azure.microsoft.com/pt-pt/services/cognitive-services/face/ [consultado em 11/02/2019]
[26] Imagem retirada de https://azure.microsoft.com/pt-pt/services/cognitive-services/face/ [consultado em 11/02/2019]
[27] https://www.tensorflow.org/ [consultado em 11/02/2019]
[28] https://www.tensorflow.org/guide/keras [consultado em 11/02/2019]
[29] https://www.python.org/ [consultado em 11/02/2019]

2.3.5. Google Cloud Vision

A Google também disponibiliza o Google Cloud Vision[30]. Esta framework encapsulando diversos modelos avançados de *machine learning* numa *API REST*, permite analisar o conteúdo das imagens de forma fácil e rápida. Através da ferramenta Cloud AutoML[31], os investigadores podem treinar os seus modelos de *machine learning* personalizados. Na Figura 11 apresentam-se os fluxos de análise de uma imagem segundo as diversas funcionalidades que a Google oferece.

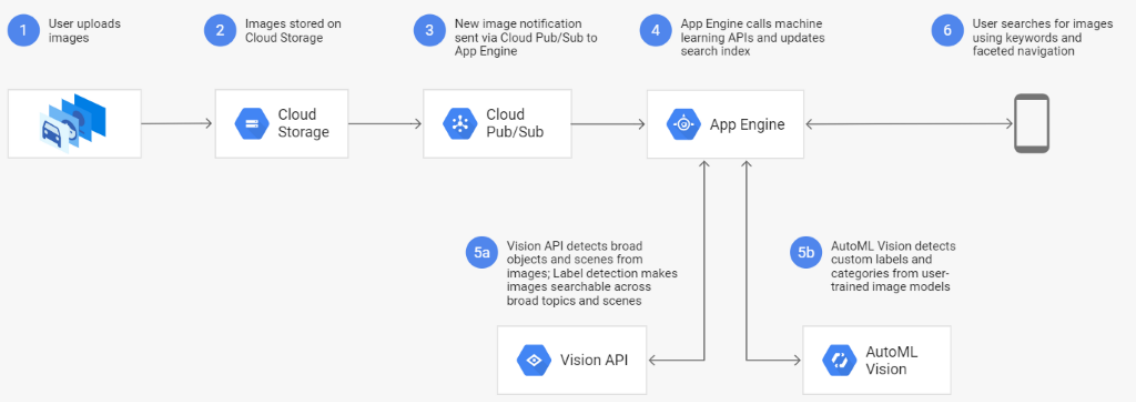


Figura 11 - Fluxo Google Vision API[32]

2.3.6. Firebase

A Google disponibiliza a Firebase[33] que consiste numa plataforma de desenvolvimento com diversas ferramentas que ajudam as equipas de desenvolvimento a implementarem uma solução aplicacional. É compatível com diversas tecnologias e ambientes, entre os quais iOS, Android, Web, Unity e C++.

Em termos de reconhecimento de imagem, possui uma ferramenta de *machine learning* para dispositivos móveis, designada Kit de ML[34], que está numa fase BETA de desenvolvimento. Esta ferramenta apenas está disponível para ambientes móveis, neste caso iOS e Android. Consoante as *APIs* escolhidas, o processamento pode ser executado tanto no dispositivo como na *cloud*. Além disso, podemos importar o nosso modelo personalizado, hospedá-lo e disponibilizá-lo através do Firebase.

[30] https://cloud.google.com/vision/ [consultado em 11/02/2019]
[31] https://cloud.google.com/automl/ [consultado em 11/02/2019]
[32] Imagem retirada de https://cloud.google.com/vision/ [consultado em 11/02/2019]
[33] https://firebase.google.com/ [consultado em 23/03/2019]
[34] https://developers.google.com/ml-kit/ [consultado em 23/03/2019]

2.3.7. OpenCV

OpenCV[35] (*Open Source Computer Vision Library*) é uma biblioteca *open-source* para desenvolvimento de aplicações de visão por computador com requisitos de tempo real. O OpenCV é desenvolvido em C++ assim como a sua *API* principal é em C++, no entanto disponibiliza *wrappers*[36] para Python, Java e Matlab/Octave.

2.4. Android Things

Em dezembro de 2016, a Google anunciou a sua nova plataforma de *Internet of Things*, denominada Android Things[37].

Usando essa plataforma, é agora possível criar dispositivos inteligentes com sistema Android que podem utilizar as Android *APIs* e os Google Services. Visto que o *Board Support Package*[38] (BSP) é gerido pela Google, não existe a preocupação de desenvolver o *kernel*[39] ou *firmware*[40]. A nível de *system image*[41] esse é fornecido e construído pelo Android Things Console[42], onde se configura os parâmetros do sistema. A instalação do sistema no dispositivo é finalizada com a colocação dessa *system image* na memória do dispositivo. Incorporados no Android Things, estão os diversos *Android Software Development Kit* (*SDK*), os Google Play Services e a Google Cloud Platform, como representado na Figura 12.

[35] https://opencv.org/ [consultado em 11/02/2019]
[36] Pacote de *software*
[37] https://developer.android.com/things [consultado em 22/06/2019]
[38] Camada de software que contém os *drivers* e rotinas
[39] Componente central do sistema operativo, estabelece as ligações entre as aplicações e o processamento
[40] Classe para controlo de baixo nível do *software*
[41] Conjunto de ficheiros estruturados com vista a instalar determinado sistema
[42] https://developer.android.com/things/console [consultado em 22/06/2019]

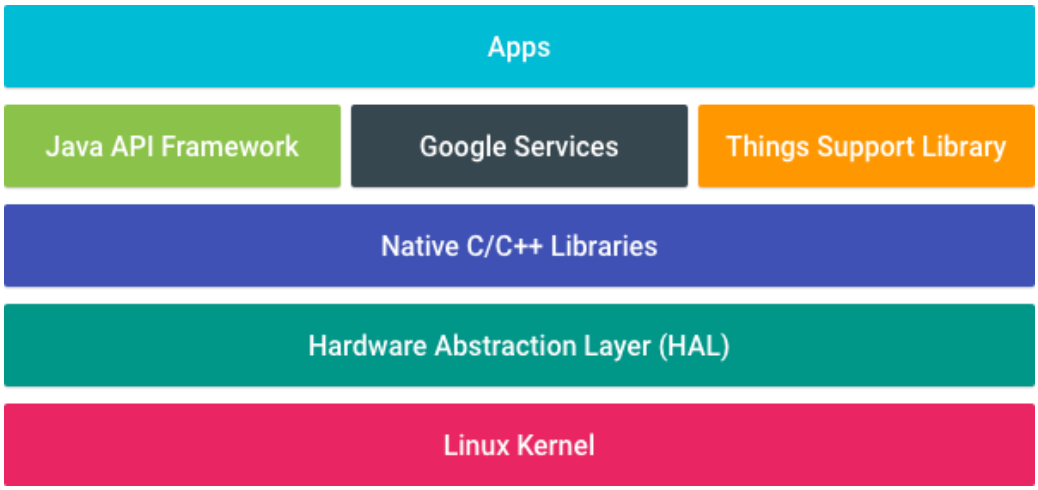


Figura 12 - Arquitetura Android Things[43]

O desenvolvimento de aplicações para dispositivos móveis ou para dispositivos *IoT* difere em alguns aspetos, tais como:

- Acesso a periféricos: em dispositivos *IoT* existe um maior acesso aos periféricos, graças às bibliotecas e *hardware* existentes, ao contrário dos móveis;
- Presença de aplicações: em dispositivos *IoT* só existe uma aplicação em funcionamento de forma a que a inicialização do sistema seja rápida e que o armazenamento não seja afetado, ao contrário dos móveis;
- Início automático: as aplicações *IoT* inicializam-se de forma automática com a ligação do sistema, ao contrário das aplicações móveis que requerem o *input* por parte dos utilizadores;

Considerada uma das principais vantagens face a outros sistemas operativos em *IoT*, tais como o Linux, é o facto de permitir atualizações diretas graças à parceria com a Google, que disponibiliza uma estrutura que permite aos fabricantes a distribuição de atualizações de sistema e *patches*[44] de segurança, dificultando assim o trabalho a ataques não desejados por parte de *hackers*[45] pois essas atualizações são realizadas remotamente.

Quanto ao hardware disponível para utilizar este sistema operativo, atualmente estão disponíveis dois equipamentos: Raspberry Pi 3 e o NXP i.MX7D, como representado na Figura 13.

[43] Imagem retirada de https://developer.android.com/things/get-started [consultado em 22/06/2019]
[44] Correção ou atualização de software
[45] Indivíduos com alto conhecimento em informática

	NXP i.MX7D	**Raspberry Pi 3**
		
Processador	NXP i.MX7 Dual	Broadcom BCM2837
Velocidade Processador	1GHz	1.2GHz
Arquitetura	ARM Cortex-A7 + M4	ARM Cortex-A53
Memória	512MB DDR3L	1GB SDRAM
Interface Debug	Micro USB debug	Micro USB debug

Figura 13 - Hardware compatível com Android Things

Capítulo 3

Fundamentos Teóricos

3.1. Machine Learning e Redes Neuronais Artificiais

Novos algoritmos permitem que máquinas possam realizar determinada tarefa com base em modelos estatísticos e inferência obtidos através da análise de dados de treino. Os sistemas de *machine learning* ao analisarem grandes quantidades de informação geram um modelo dinâmico que é refinado durante o progresso da aprendizagem. Existem diversos tipos de algoritmos passíveis de serem utilizados, como também diversos tipos de processos de aprendizagem, entre os quais se destacam a aprendizagem supervisionada, não-supervisionada e semi-supervisionada. Na supervisionada, os dados em teste já possuem os *inputs* e *outputs*. Na não-supervisionada, só existem dados de *input* e na semi-supervisionada existem muitos dados de *input* mas poucos dados de *output*. Por norma, é utilizada a aprendizagem supervisionada.

Os algoritmos de aprendizagem podem ser classificados em algoritmos de regressão, algoritmos de classificação, *deep learning* e árvore de decisão.

Uma rede neuronal artificial é a tentativa de simular ou replicar os processos que ocorrem no nosso cérebro durante a execução de uma determinada tarefa. É composta por um determinado número de *inputs*, um dado número de camadas escondidas e outro número de *outputs*, como representado na Figura 14.

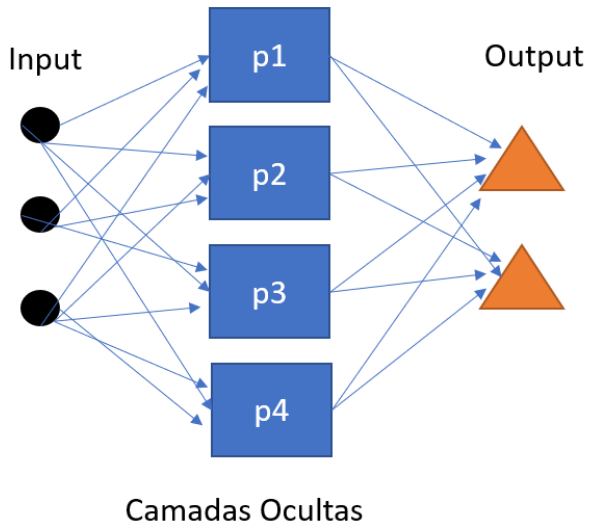


Figura 14 - Arquitetura de uma rede neuronal de 3 camadas

Estes diferentes nós podem ser considerados como os neurónios do cérebro ou da rede a criar. Estas redes podem integrar diversas camadas com um diferente número de neurónios, existindo sempre uma camada de entrada (*input*) e outra de saída (*output*). O objetivo desta rede é realizar uma dada tarefa com sucesso e tal sucesso é mais suscetível de acontecer quanto mais neurónios de entrada e saída integrarem a estrutura da rede. Os neurónios estão conectados entre si através de uma ligação, cuja contributo é ajustável através do seu peso (representado na camada oculta). Durante o processo de aprendizagem, os pesos de cada neurónio são atualizados e por meio dessas ligações existe uma propagação de informação. Caso os *outputs* de uma rede sejam os *inputs* de uma outra rede, criam-se assim diversas camadas ocultas, gerando uma DNN (*Deep Neural Network*). Os algoritmos de *machine learning* do tipo *deep learning* funcionam no domínio das redes DNN pois estas possuem várias camadas ocultas. Ao contrário dos restantes algoritmos que já possuem os *inputs* e *outputs* possíveis de determinar, no *Deep Learning* os modelos são inferidos através das camadas ocultas existentes e respetiva aprendizagem.

3.2. Convolutional Neural Network (CNN)

Para o processamento de imagem nas redes convencionais, a análise de uma imagem implica um elevado processamento por parte dos neurónios de cada camada. Se a rede receber no *input* uma imagem com o tamanho 150 x 150 pixels e se existirem 100 neurónios por cada camada, isto significa que cada neurónio irá receber a imagem fornecida e atribuir um peso por cada píxel. Portanto, cada neurónio irá processar 150 x 150 = 22500 pesos. Adicionalmente se multiplicarmos pelo número de neurónios de cada camada, neste caso 100, a rede irá processar mais de dois milhões de parâmetros para uma pequena imagem. Torna-se evidente que o tempo de treino desta rede será muito grande e que o consumo de processamento será elevado.

As análises de imagens através de redes neuronais tipicamente envolvem as *Convolutional Neural Networks* (CNN) (Lee, Grosse, Ranganath, & Ng, 2009), optimizando os requisitos de treino e poder de cálculo. Tais redes são bastante utilizadas em tarefas complexas que envolvem imagens, vídeos, textos e muitos outros. A principal diferença é que nestas redes as camadas ocultas não estão todas ligadas entre si na fase de aprendizagem, dependendo de menos parâmetros do que as redes neuronais convencionais que possuem as camadas ocultas totalmente interligadas.

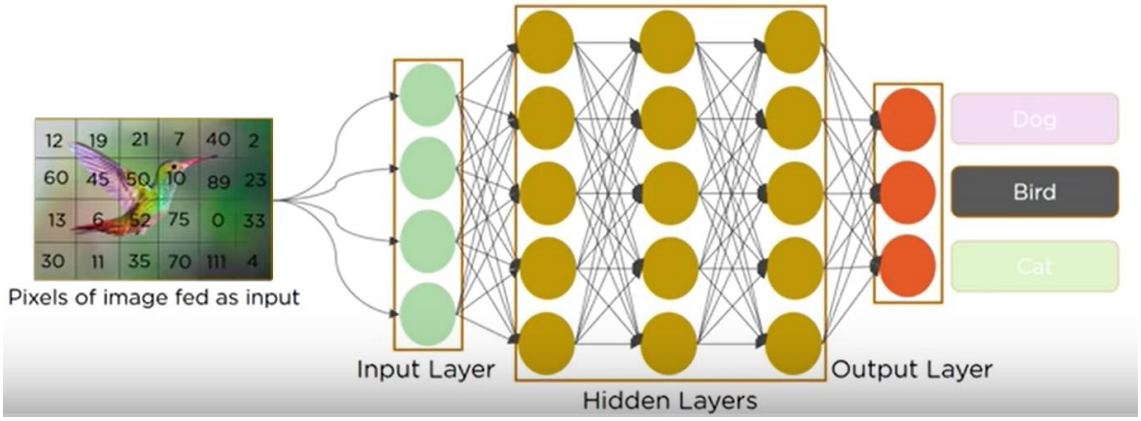


Figura 15 - Fluxo de uma Convolutional Neural Network[46]

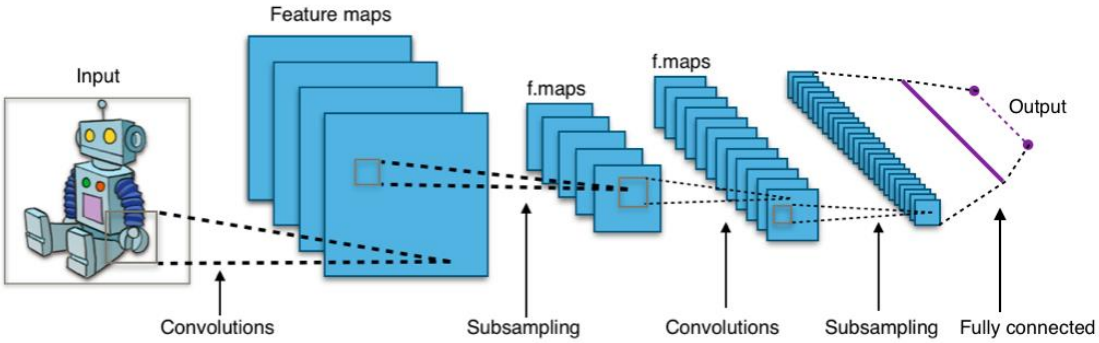


Figura 16 - Arquitetura de uma Convolutional Neural Network[47]

Além do processamento elevado, outro dos problemas no reconhecimento de imagens é a conversão da imagem de *input* em vetores. Nesse processo perdem-se detalhes e informações (pixéis adjacentes, etc) da imagem cruciais para o reconhecimento. A arquitetura de uma CNN procura resolver estas lacunas (ver Figura 15 e Figura 16). As redes CNN utilizam um algoritmo de convolução para o tratamento das imagens durante o processo de reconhecimento e aprendizagem. Nesse algoritmo utilizam-se filtros, em forma de matriz e operações de convolução que tem como função salientar as primitivas mais importantes (*features*) na imagem e descartar as restantes. As CNNs são tipicamente constituídas por 4 camadas (*Convolutional Layer*, *ReLU Layer*, *Pooling Layer* e *Fully Connected Layer*). Após as sucessivas convoluções e aprendizagens o *output* será gerado.

[46] Imagem retirada de https://www.youtube.com/watch?v=Jy9-aGMB_TE&feature=youtu.be [consultado em 05/12/2019]
[47] Imagem retirada de https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9 [consultado em 10/11/2019]

3.3. Arquiteturas e Cloud AutoML

Sobre a arquitetura das redes neurais utilizadas pelas plataformas da Google, podemos afirmar que estas estão em constante desenvolvimento. A implementação dessas redes tem sido um sucesso em diversas áreas, desde reconhecimento de texto a reconhecimento de imagens. Um dos grandes problemas que existe nos dias de hoje é o tempo e custo que esses modelos acarretam a serem desenvolvidos por equipas de engenheiros e cientistas altamente qualificados na área, porque apesar de serem conhecedores da matéria o número possível de modelos a implementar em determinada tarefa é muito vasto. Isto leva a que certas empresas, como a Google, invistam em mecanismos automáticos para desenvolvimento dos seus modelos de *machine learning*. A Google apostou em diferentes tipos de algoritmos dos quais se destacaram dois com grande potencial, *evolutionary algorithms* (Real, et al., 2017) e *reinforcement learning algorithms* (Le & Zoph, 2017). Ambos reforçam a ideia de que a conceção da estrutura de uma rede neuronal pelo ser humano pode tornar-se muito complexa e a afirmam que a conceção e pesquisa de forma automática é o caminho a seguir. Quanto ao primeiro, este baseia-se na repetibilidade de testes enquanto que o segundo refina a inferência através da geração iterativa de modelos.

No caso dos *reinforcement learning algorithms* a sua arquitetura consiste no treino de uma rede "filha" pelo controlador da rede neuronal, com vista a ser avaliada e testada para determinada vertente, como está representado na Figura 17.

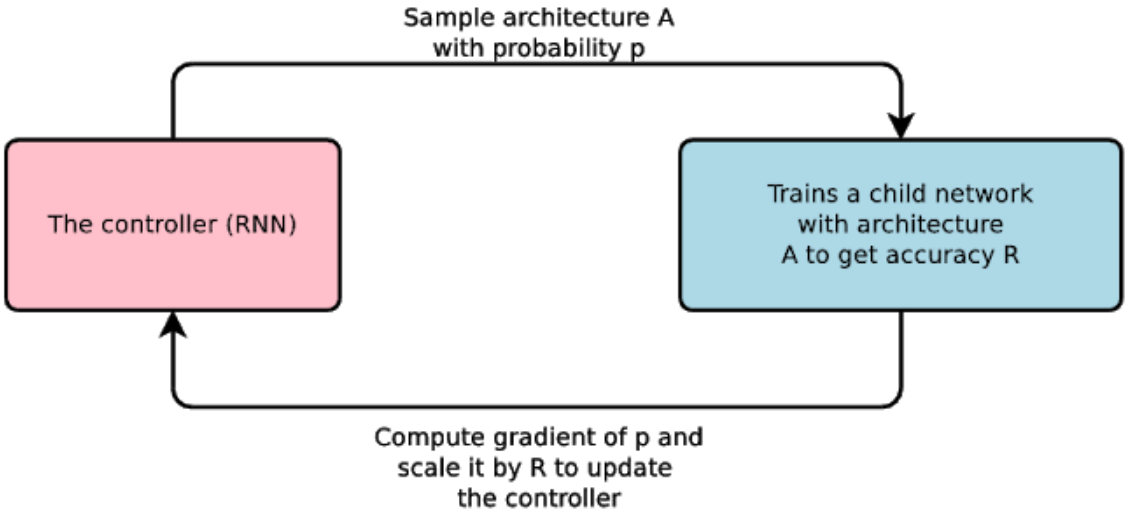


Figura 17 – Visão da Neural Architecture Search[48]

Através de uma rede recorrente, é possível atualizar o controlador com uma maior precisão graças à validação que foi obtida na rede "filha". Numa próxima iteração, o controlador dará maior enfâse a arquiteturas com maior precisão. Assim, o controlador aprende a melhorar a sua pesquisa ao longo do tempo.

Ao serem criadas redes neuronais de forma automática, permite aos programadores com conhecimento limitado na área de aprendizagem possam criar redes neuronais que funcionam de forma eficaz nas suas aplicações.

Utilizando a *Neural Architecture Search* (*NAS*) baseada nos *reinforcement learning algorithms*, é encontrado a melhor arquitetura de uma rede neuronal para determinado fim. Diferentes blocos são escolhidos e colocados juntos de forma a criar uma rede neuronal. Essa rede neuronal é testada e baseando-se nos resultados obtidos, esses blocos são ajustados de forma a otimizar a rede o máximo possível. Na Figura 18 estão representados alguns blocos recomendados (Zoph, Vasudevan, Shlens, & Le, 2018) para a criação de uma rede neuronal de reconhecimento de imagem.

[48] Imagem retirada do artigo (Le & Zoph, 2017)

- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

Figura 18 - Blocos para uma rede de reconhecimento de imagem

Os bons resultados obtidos através da solução *NAS* devem-se ao facto da arquitetura criada ser treinada e testada num pequeno conjunto de dados, porque se fosse treinada num grande conjunto de dados como é o ImageNet demoraria bastante tempo. O conceito destas redes neurais pode simplificar-se da seguinte maneira: se funcionam em pequenos conjuntos de dados de forma eficaz também irão funcionar de forma correta em conjuntos maiores.

Apesar da abordagem de *NAS* ser eficaz, não é eficiente pois necessita de bastante poder de computação como também de tempo. Hoje os investigadores tentam tornar esse mecanismo mais eficiente, através das novas arquiteturas: *Progressive Neural Architecture Search (PNAS)* (Liu, et al., 2018) e *Efficient Neural Architecture Search* (Pham, Guan, Zoph, Le, & Jeff, 2018). De forma resumida, as *PNAS* usam uma abordagem de otimização sequencial (*Sequential Model-Based Optimization*) ao contrário do *reinforcement learning* utilizado no NAS. Referente às *ENAS*, estas consistem em aproveitar todos pesos treinados utilizando o *transfer learning*, que se resume ao aproveitamento de forma eficaz do treino que já foi realizado para um novo conjunto de dados, como representado na Figura 19.

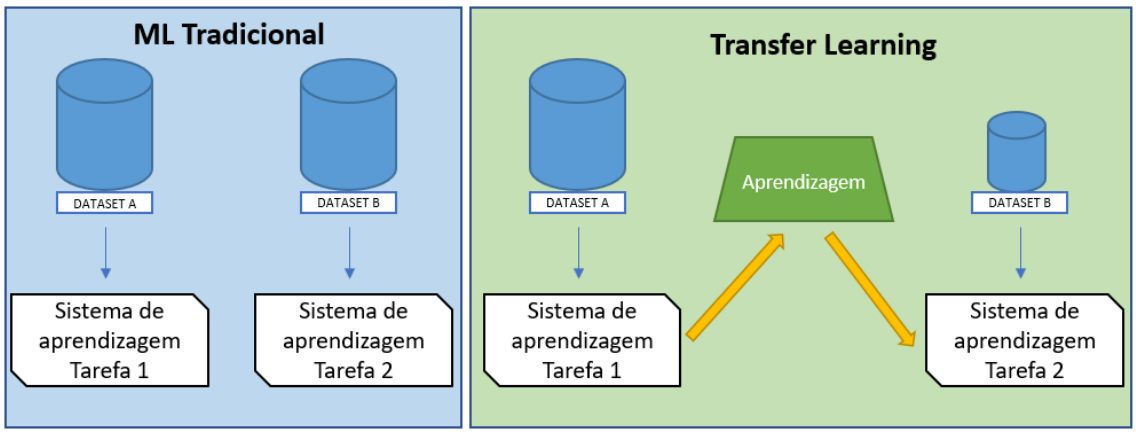


Figura 19 - Diferenças entre ML Tradicional e o Transfer Learning

Transfer learning ao contrário do *machine learning* tradicional, supera assim o ambiente de aprendizagem isolado e utiliza o conhecimento adquirido numa determinada tarefa de forma a resolver tarefas relacionadas. No tradicional, a aprendizagem é única e isolada, onde o conhecimento não é retido. No *transfer learning* a aprendizagem depende do conhecimento resultante das tarefas anteriores, pelo que o processo pode ser mais rápido e preciso, podendo mesmo necessitar de um *dataset* menor para o treino.

De forma a facilitar a utilização destas redes neurais, eis que a Google disponibiliza a plataforma Cloud AutoML (Figura 20) para a sociedade utilizar, interligada com a ferramenta ML Kit. O único passo que o investigador necessita de realizar é fornecer os dados que pretende testar, pois a complexidade do *deep learning* é realizado pela Google através do seu algoritmo *NAS*, encontrando para isso a melhor arquitetura aos dados fornecidos.

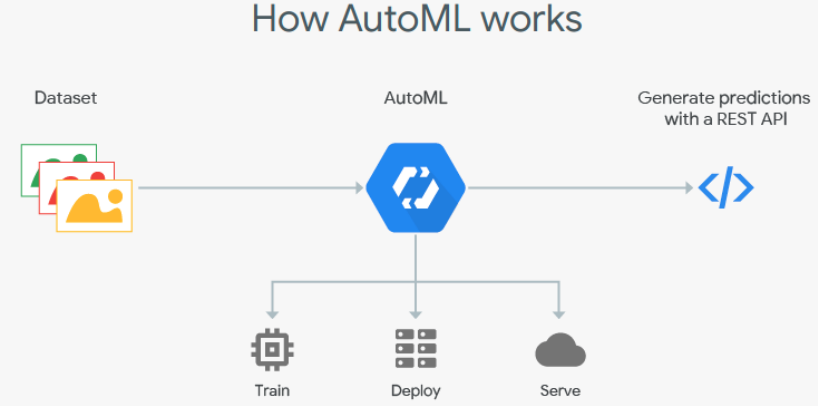


Figura 20 - Google Cloud AutoML[49]

3.4. Avaliação de resultados

Uma das formas utilizadas pela comunidade científica para analisar a performance de um algoritmo de classificação, neste caso de reconhecimento, é através de uma matriz de confusão. A matriz de confusão é uma tabela que tem como função descrever a performance de certo algoritmo num conjunto de dados, conhecendo dados verdadeiros das informações em estudo.

[49] Imagem retirada de https://cloud.google.com/automl/docs/ [consultado em 22/08/2019]

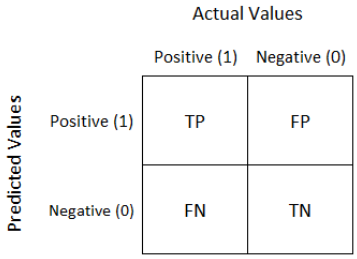


Figura 21 - Tabela de uma matriz de confusão[50]

Essa tabela ou matriz (Figura 21) é constituída por linhas e colunas. Nas linhas, está representado os valores previstos enquanto que nas colunas estão representados os valores reais. Na tabela constam o número de falsos positivos (FP), falsos negativos (FN), verdadeiros positivos (TP) e verdadeiros negativos (TN). Por exemplo, utilizando certo algoritmo para identificar/distinguir entre cavalos e zebras, os resultados podem ser apresentados sob a forma de uma tabela (ver Figura 22).

Matriz de confusão		**Valores Reais**	
		Zebra	Cavalo
Valores Previstos	Zebra	5	2
	Cavalo	3	3

Figura 22 - Tabela exemplo Zebras e Cavalos

Assumindo que eram treze animais em estudo, oito zebras e cinco cavalos, o algoritmo previu que das oito zebras reais, três eram cavalos. Da mesma forma, dos cinco cavalos, o tal algoritmo assumiu que dois eram zebras. Desta forma, é possível visualizar erros de previsão. Em suma, neste teste exemplo foram detetados cinco verdadeiros positivos, dois falsos positivos, três negativos falsos e três verdadeiros negativos.

Utilizando a matriz de confusão, é agora possível calcular a precisão e o *recall* desse algoritmo. O *recall* é calculado a partir da divisão entre os verdadeiros positivos e a soma dos verdadeiros positivos com os falsos negativos. No exemplo de cima, constatamos que a zebra foi identificada corretamente cinco vezes, porém foi identificada duas vezes com sendo um cavalo. Logo, o *recall* é calculado da seguinte

[50] Imagem retirada de https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [consultado em 27/07/2019]

forma: $\frac{5}{5+2} = 0.714$, sendo essa a probabilidade de o algoritmo identificar de forma correta uma zebra. Quanto à precisão, esta é calculada a partir da divisão entre os verdadeiros positivos e a soma dos verdadeiros positivos com os falsos positivos. No exemplo de cima, a probabilidade de o algoritmo reconhecer um animal como sendo mesmo uma zebra é calculada da seguinte forma: $\frac{5}{5+3} = 0.625$. Isto porque o algoritmo reconheceu cinco vezes de forma acertada, porém reconheceu três vezes uma zebra sendo um cavalo.

Capítulo 4

Metodologia

Presentemente as pessoas esperam que a tecnologia simplifique o seu dia-dia e recorrem cada vez mais ao seu *smartphone* para realizar as mais diversas tarefas. O "espelho inteligente", capaz de disponibilizar informação de forma natural, atualizada e personalizada a um utilizador enquanto este executa tarefas do quotidiano em frente a um espelho, é mais um contributo para a análise de interação simplificada com a informação. O desenvolvimento desta solução envolve tecnologias e ferramentas das áreas da Internet das Coisas, *Mobile* e *Machine Learning*.

A plataforma computacional do "espelho inteligente" pretende-se integrada no ecossistema Android dado o elevado número de utilizadores detentores de dispositivos móveis compatíveis e devido ao facto de a Google fornecer serviços de *Cloud* para reconhecimento de imagens para aplicações Android. O sistema operativo adoptado é o Android Things e corre num Raspberry Pi 3.

Sendo o Android um produto da Google, será utilizada como ferramenta de reconhecimento facial, o Firebase ML Kit, pertencente igualmente à Google. Tal escolha deve-se sobretudo à facilidade de integração e escalabilidade que é possível obter com esta arquitetura, de forma rápida e simples. O facto de poder ser executado no dispositivo e/ou na *cloud*, ser multiplataforma, processar os dados sem acesso à *internet* e facultar um local de *hosting* para o nosso modelo foram características tidas em conta na escolha. Outras ferramentas foram ponderadas, porém não ofereciam diversas funcionalidades orientadas para dispositivos móveis como o Firebase oferece de forma estável e a um preço acessível.

ML Kit tem como vantagens a facilidade de utilização pois a integração de inteligência artificial em aplicações móveis não é tarefa fácil, pois além de as bibliotecas de *machine learning* serem complexas o modelo tem de ser construído de forma a que o seu processamento seja executado de forma eficiente e leviana em dispositivos móveis. Através desta ferramenta o único requisito que o investigador tem de realizar é o envio de dados para o ML Kit e processar a resposta proveniente do *SDK*. De acordo com a Google, a privacidade dos dados está segura pois estes não são guardados nas chamadas à API, sendo imediatamente eliminados após o processo da chamada terminar. Outra das

razões principais de se ter escolhido o ML Kit face a outras tecnologias foi a disponibilização de *SDKs* e *APIs*, que são essenciais para o desenvolvimento de aplicações móveis. Tecnologias como a Microsoft Azure, o OpenCV e a Amazon ficaram de parte na escolha devido ao não fornecimento dessas valências.

O ML Kit também possui desvantagens, uma delas é o facto de o tamanho da aplicação crescer de forma proporcional com o tamanho do modelo inserido nela. Logo se o modelo de *machine learning* implementado de forma local no dispositivo for muito grande, o tamanho da aplicação aumentará. No entanto, para o desenvolvimento deste projeto essa desvantagem não será preocupante pois o modelo implementado é de tamanho reduzido. Outra das desvantagens é o estado da ferramenta atual, que se encontra numa fase *beta* e o suporte/documentação não é muito extenso. Além disso, a parametrização do algoritmo utilizado nas ferramentas do ML Kit é ainda indisponível face ao estado atual da plataforma.

4.1. Etapas do Reconhecimento Facial

O processo de reconhecimento de determinado utilizador é dividido por etapas, como referido no tópico 2.2. No âmbito de Specchio podemos considerar duas grandes etapas, Deteção e Reconhecimento.

4.1.1. Deteção

O primeiro passo para o reconhecimento de uma pessoa é a sua deteção. Após capturar a imagem, Specchio irá detetar quantas faces existem na imagem capturada pelo espelho recorrendo à tecnologia de Deteção Facial[51] do ML Kit do Firebase (Figura 23).

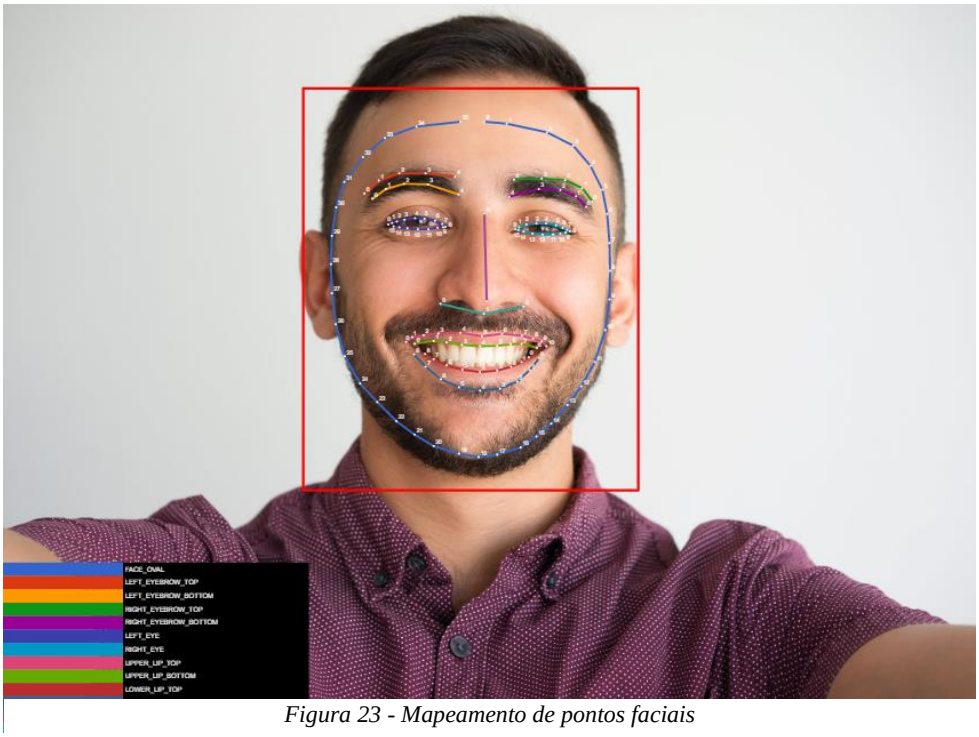


Figura 23 - Mapeamento de pontos faciais

Através dessa API é possível detetar rostos numa imagem, obter o seu contorno e os principais traços faciais. Outra das características é o reconhecimento de expressões faciais, tais como o sorriso e olhos abertos.

Tais funções são conseguidas pela utilização de pontos de referência e ângulos do rosto em relação à câmara. Quantos aos ângulos, são descritos da seguinte forma:

- Euler X: um rosto com um ângulo de Euler X positivo está voltado para cima
- Euler Y: um rosto com um ângulo de Euler Y positivo está voltado para o lado direito da câmara
- Euler Z: um rosto com ângulo de Euler Z positivo está girado no sentido anti-horário em relação à câmara

A ferramenta disponibiliza todos estes ângulos em diferentes modos, possíveis de selecionar dependendo do objetivo da deteção, rapidez ou eficácia. Dependendo do

[51] https://firebase.google.com/docs/ml-kit/detect-faces [consultado em 02/11/2019]

ângulo de Euler Y existem pontos de referência detetáveis na face do sujeito, dos quais se destacam:

- Olho esquerdo
- Olho direito
- Orelha esquerda
- Orelha direita
- Base do nariz
- Parte inferior da boca
- Etc…

No total, existem 133 pontos possíveis detetar numa face nesta plataforma. Tais pontos dividem-se em conjuntos retratando determinado contorno, como a sobrancelha, a bochecha, formato do rosto e por aí adiante. O maior intervalo de pontos, de 0 a 35, representa neste caso o formato do rosto.

Outra funcionalidade desta plataforma é a classificação de uma característica facial numa determinada imagem. Atualmente só é compatível com duas características, sorriso e olhos abertos. Tal classificação retorno um valor preciso, entre 0 e 1. Quanto maior o valor, maior é a probabilidade de a pessoa estar a sorrir e/ou com os olhos abertos.

Specchio ao analisar a imagem capturada, se detetar mais de uma face, irá mostrar o *dashboard dinâmico default*, com a configuração *default* implementada. Se detetar apenas uma face irá proceder à próxima fase que é o reconhecimento.

4.1.2. Identificação

Com uma face detetada, Specchio recorre agora à ferramenta AutoML Vision Edge[52] para a identificação de determinada face. Esta ferramenta de rotulação de imagens possui no seu portefólio cerca de 400 categorias distintas que abrangem diversos conceitos, no entanto a característica que torna esta ferramenta tão conhecida é o facto de possibilitar a criação e treino de um modelo de classificação personalizado com os nossos próprios dados. Outro dos principais recursos do AutoML Vision Edge é a hospedagem do modelo de classificação criado no Firebase, disponibilizando *SDKs* para futura utilização em dispositivos móveis, tanto iOS como Android.

[52] https://firebase.google.com/docs/ml-kit/automl-image-labeling [consultado em 03/11/2019]

4.1.2.1. Conjunto de dados

O primeiro passo consiste na criação de um projeto na consola[53] do Firebase. A etapa seguinte é a obtenção de dados e a sua organização. Como pretende-se reconhecer faces, é necessária uma coleção de imagens com faces de pessoas. Essa coleção de imagens é importada para o nosso modelo através da plataforma AutoML, havendo três formas de *upload* dessas imagens:

- Arquivo ZIP estruturado
- Cloud Storage[54] com índice CSV
- Imagens não rotuladas

Optou-se pela terceira opção pois a Cloud Storage necessita de um plano Firebase Blaze[55] (plano pago de acordo com a utilização) como também pelo facto de a quantidade de imagens a importar para uma primeira fase ser reduzida.

Para uma primeira fase foram importadas 20 imagens de cinco personalidades, Cristiano Ronaldo, Lionel Messi, Roger Federer, Rafael Nadal e eu próprio, perfazendo um conjunto de 100 imagens, como indicado na Figura 24. A cada imagem foi atribuído um rótulo, neste caso o nome de cada personalidade. Para testar o modelo ao máximo além de cada imagem ser distinta, houve a tentativa de incorporar imagens de diferentes ângulos de forma a que o modelo fosse o mais preciso possível. O reduzido número de imagens de cada personalidade deve-se ao facto do reduzido número de contas na aplicação registados. Isto é, cada personalidade tem uma conta registada e caso se importasse um *dataset* com muitas imagens e personalidades, teria de haver o mesmo número de contas na aplicação registadas. Como estamos num ambiente de protótipo, escolheu-se essa opção do *dataset* reduzido.

[53] https://console.firebase.google.com/u/0/ [consultado em 04/11/2019]
[54] https://firebase.google.com/docs/storage [consultado em 04/11/2019]
[55] https://firebase.google.com/pricing/?hl=pt-br#blaze-calculator [consultado em 04/11/2019]

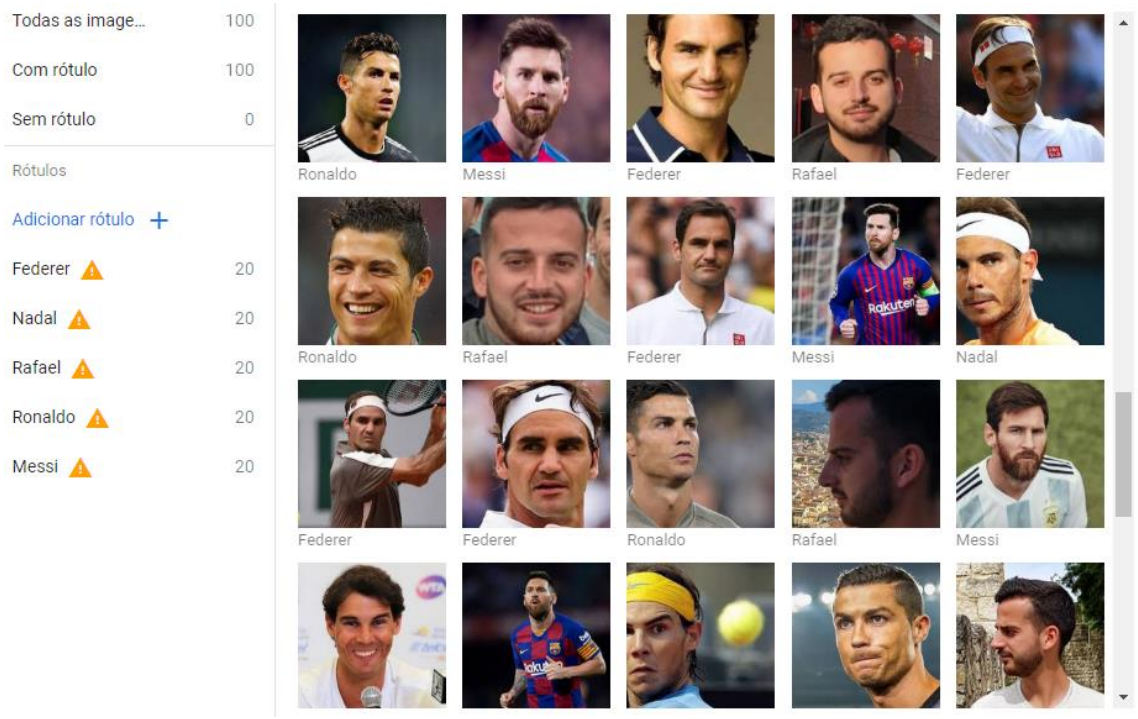


Figura 24 - Dataset de imagens utilizado

4.1.2.2. Treino

Após as imagens estarem com o respetivo rótulo, chega a fase do treino do modelo. O primeiro passo é dar um nome ao modelo a treinar, neste caso foi FaceDataset. De seguida é possível configurar dois parâmetros do treino, a latência/tamanho de pacote e o tempo de treinamento. Se a prioridade for a latência baixa ou tamanho de pacote reduzido, o modelo será menor e mais rápido. Se a precisão for prioritária, o modelo será maior e mais lento.

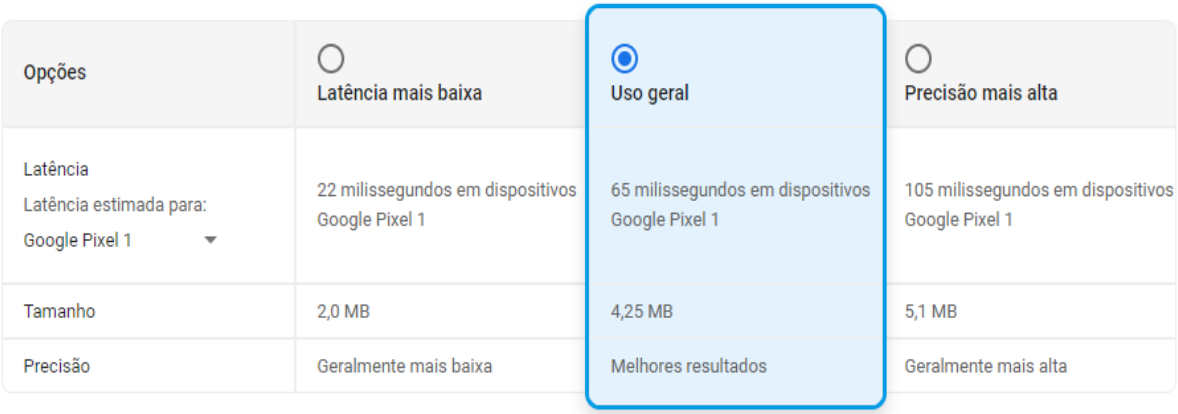


Figura 25 - Latência e tamanho do pacote[56]

[56] Imagem retirada de https://cloud.google.com/automl/ [consultado em 04/11/2019]

No caso do "espelho inteligente" escolhemos a solução intermédia uso geral (Figura 25) com vista a obter os melhores resultados na precisão face ao tamanho do pacote.

Quanto ao tempo de treino, optou-se pela duração de 1 hora pois é a indicada segundo a Figura 26 para conjunto de imagens pequenos, neste caso 100 imagens. Quanto maior o tempo maior é a precisão, por norma.

Tempos de treinamento típicos	
Conjuntos muito pequenos	1 hora
500 imagens	2 horas
1.000 imagens	3 horas
5.000 imagens	6 horas
10.000 imagens	7 horas
50.000 imagens	11 horas
100.000 imagens	13 horas
1.000.000 de imagens	18 horas

Figura 26 - Tempos de treino ML Kit[57]

4.1.2.3. Implementação

Por fim, estando o treino finalizado resta agora utilizar o modelo. AutoML fornece duas opções para a utilização do modelo: fazer o *download* do próprio ou publicar o modelo.

Ao fazer o download do modelo, este é colocado dentro do projeto da aplicação, consistindo em três ficheiros:

- **model.tflite**: modelo propriamente dito em formato TensorFlow Lite[58], indicado para utilização em dispositivos móveis, embebidos e de *IoT*
- **dict.txt**: ficheiro de texto que possui os rótulos das imagens
- **manifest.json**: ficheiro JSON com o mapeamento dos ficheiros

[57] Imagem retirada de https://cloud.google.com/automl/ [consultado em 04/11/2019]
[58] https://www.tensorflow.org/lite/guide/get_started [consultado em 03/11/2019]

Ao publicar o modelo, este fica alojado e disponível no Firebase. As principais vantagens deste modo é a possibilidade de atualizar o modelo sem ter de criar uma versão da aplicação que está a utilizar o mesmo. Além disso, torna o tamanho inicial da aplicação menor, a configuração remota é possível como também a exibição de diferentes modelos para diferentes conjuntos de utilizadores. De realçar que qualquer funcionalidade referente ao modelo não será possível até que este seja transferido para a aplicação, caso este não tenha sido importado localmente.

Numa primeira fase optou-se pelo modelo local pois não se espera alterações no modelo de forma remota nesta fase. No entanto, é possível conciliar ambas as alternativas simultaneamente para a utilização no modelo caso seja necessário.

Com o modelo pronto, é agora possível reconhecer numa dada imagem o utilizador que se encontra frente ao espelho e adaptar o conteúdo do espelho a si, conforme descrito no tópico 4.3.

4.1.2.4. Análise do modelo

Com o modelo gerado e treinado, podemos agora investigar o que está por dentro do modelo com a utilização de uma ferramenta designada Netron[59]. Esta ferramenta consiste em visualizar redes neurais, modelos *deep learning* e *machine learning*. Como parâmetro de análise recebe o modelo ou a rede neural a analisar, servindo de *input*. Na Figura 27 estão representadas algumas informações obtidas pela utilização do Netron. Podemos constatar as seguintes informações:

- O modelo é quantitizado visto que o AutoML cria por defeito modelos quantitizados (método de compressão para reduzir o tamanho dos modelos com vista a ser utilizados em dispositivos móveis)
- O formato do modelo é TensorFlow Lite v3
- O *input* do modelo tem dimensão 1x224x224x3 (imagens com tamanho 224x224 com 3 canais)
- A saída do modelo tem dimensão 1x5 (5 rótulos)

[59] https://github.com/lutzroeder/netron [consultado em 03/11/2019]

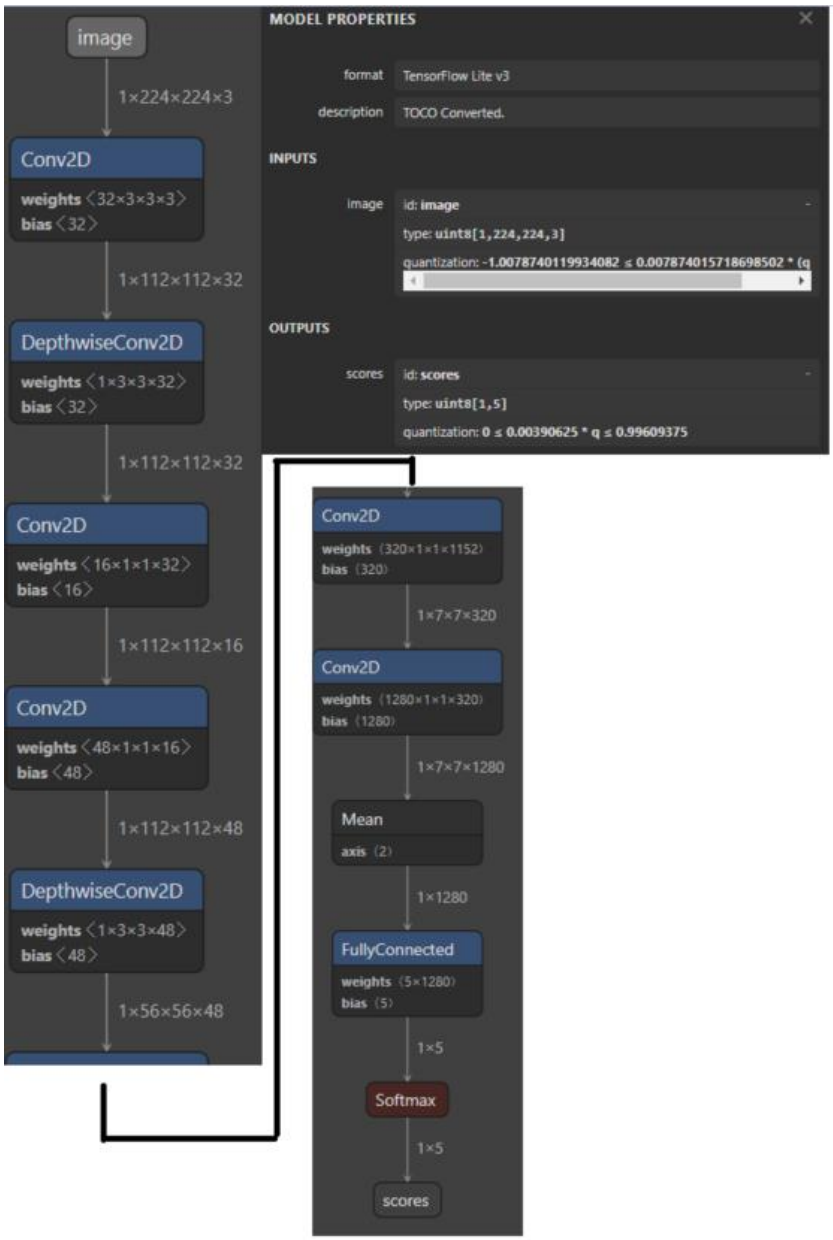


Figura 27 - Análise do modelo gerado[60]

4.2. Reconhecimento de objetos

De forma complementar ao projeto e não sendo este o objetivo primordial, foi implementado em Specchio uma tecnologia de reconhecimento de objetos detetados na imagem capturada. Para isso utilizou-se uma rede neuronal desenvolvida por investigadores da Google com o formato de TensorFlow Lite. A esta solução deram o nome de MobileNet[61] e treinaram-na com milhões de imagens provenientes da

[60] Imagem gerada na ferramenta *Netron*
[61] https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md [consultado em 23/08/2019]

ImageNet[62], resultando num modelo com cerca de 1000 categorias distintas. Visto que este modelo é pré-treinado (previamente treinado num extenso *dataset*), pode ser personalizado aplicando a técnica de *transfer learning*. Como o reconhecimento de objetos não é a temática principal de Specchio, decidimos utilizar o modelo pré-treinado sem ter de o alterar para novos dados. Tal como o modelo de AutoML para reconhecimento de utilizadores, este modelo está alojado no projeto em forma de dois ficheiros:

- **mobilenet_quant_v1_224.tflite**: modelo em formato Tensorflow Lite
- **labels.txt**: ficheiro de texto que possui o dicionário das categorias existentes

4.3. *Dashboard* dinâmico

O *dashboard* dinâmico consiste nas informações que são apresentadas ao utilizador que variam de acordo com o utilizador reconhecido. Caso este não seja reconhecido ou Specchio detete mais de um utilizador simultaneamente, serão apresentadas as informações por defeito sem qualquer configuração guardada.

Diversas são as informações apresentadas, como representado na Figura 28, entre as quais se destacam:

- **Meteorologia**: Specchio recorre ao Dark Sky API[63] para obter informações meteorológicas consoante a localização designada. A sua utilização é grátis para 1000 chamadas/dia a esse serviço, por cada chamada a mais desse limite a plataforma cobra 0,000091€. Estes são alguns dados dos quais é possível extrair através desta API:
  - Pressão atmosférica
  - Humidade
  - Tipo de precipitação
  - Nascer/Pôr do Sol
  - Velocidade do vento
  - Direção do vento
  - Temperatura aparente
  - Cobertura de nuvens
  - Etc...

[62] http://image-net.org/ [consultado em 23/08/2019]
[63] https://darksky.net/dev [consultado em 23/08/2019]

- **Notícias**: para apresentação de notícias Specchio utiliza a News API[64] que fornece manchete de notícias e artigos de mais de 30,000 fontes distintas presentes na internet de forma grátis, sendo possível filtrar o conteúdo recebido pelos seguintes critérios:
    - Palavras-chave
    - Data da publicação
    - Nome da fonte
    - Domínio da fonte
    - Linguagem
- **Frases inspiradoras**: relativamente às frases inspiradoras Specchio utiliza a TheySaidSo Famous Quotes API[65] onde a sua implementação é grátis e a filtragem do conteúdo devolvido é possível usando diferentes categorias, de acordo com o tipo de frase que é pretendido
- **Horas:** utiliza o tempo do sistema onde a aplicação está a correr
- **Número de faces detetadas:** como mencionado em 4.1.1, Specchio deteta quantas faces estão na imagem e apresenta o número de faces detetadas
- **QR Code:** após determinada ação, como um clique de um botão, será visualizado um QR Code no *dashboard* (no espelho) para o utilizador através da aplicação móvel registar o espelho na sua conta. Esse QR Code é gerado de forma aleatória e contém o endereço/identificação desse espelho na base de dados. Com o endereço do espelho, o utilizador guarda-o na sua conta, atribuindo um nome ao espelho. Após esse registo, é possível configurar o espelho através da aplicação *mobile* miSpecchio
- **Objeto detetado:** pela utilização do modelo referido em 4.2, é apresentado o objeto reconhecido na imagem capturada

[64] https://newsapi.org/ [consultado em 23/08/2019]
[65] https://theysaidso.com/api/ [consultado em 23/08/2019]

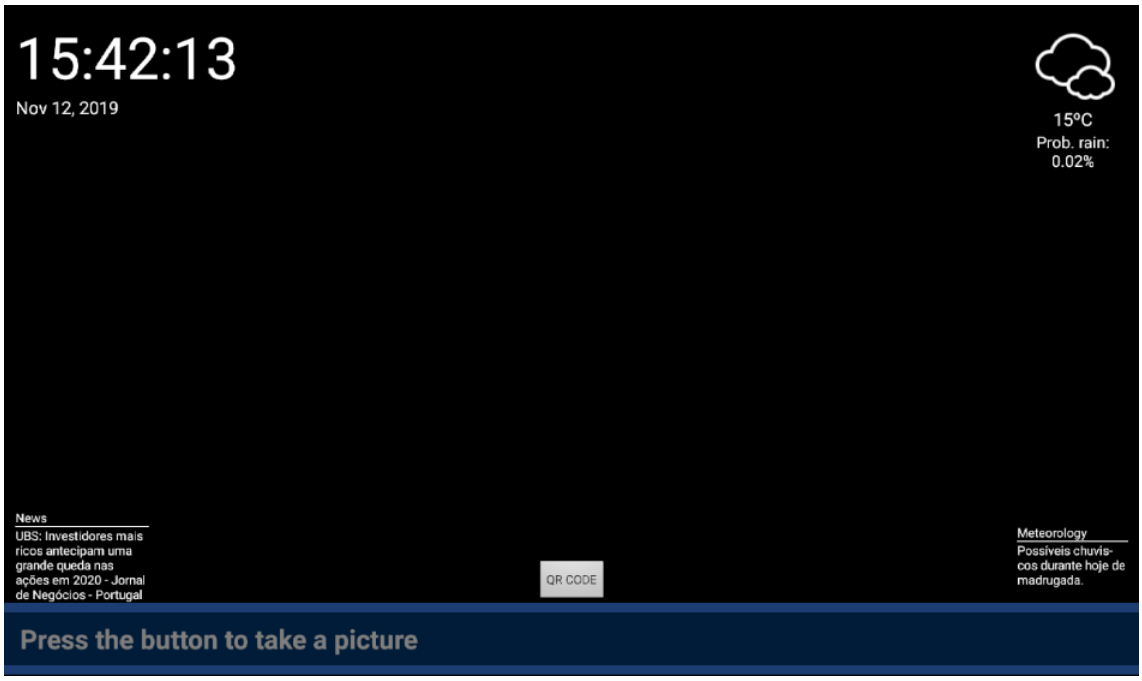


Figura 28 - Dashboard dinâmico

4.4. Plataforma Firebase

Além de facultar ferramentas de reconhecimento de imagem, a plataforma Firebase fornece também a hospedagem dos serviços e ferramentas necessárias para um projeto tecnológico decorrer sem problemas. Em Specchio utilizamos a Firebase Authentication[66] para autenticação na aplicação móvel, a Firebase Realtime Database[67] como base de dados e o Firebase ML Kit para reconhecimento como já referimos anteriormente.

Em relação à autenticação, esta é fulcral para que o sistema consiga reconhecer o utilizador em causa. A Firebase Authentication fornece serviços, bibliotecas e *SDKs* fáceis de implementar nas aplicações. Foi a tecnologia escolhida devido à simplicidade, gestão e segurança na criação de utilizadores. Com ela, é possível autenticar no sistema através de email e password, através de número de telefone e possui a funcionalidade de recuperação de contas incorporada nos seus serviços.

Quanto à base de dados, esta concebida com vista em registar os espelhos e interligar os espelhos e utilizadores. Empregando a ferramenta Realtime Database, os dados ficam armazenados e sincronizados numa base de dados NoSQL, que consiste numa base de dados não relacional. Todos os dados são sincronizados em todos os clientes

[66] https://firebase.google.com/docs/auth [consultado em 02/11/2019]
[67] https://firebase.google.com/docs/database [consultado em 02/11/2019]

em tempo real permanecendo disponíveis mesmo quando a aplicação se encontra no estado *offline* (sem acesso à internet).

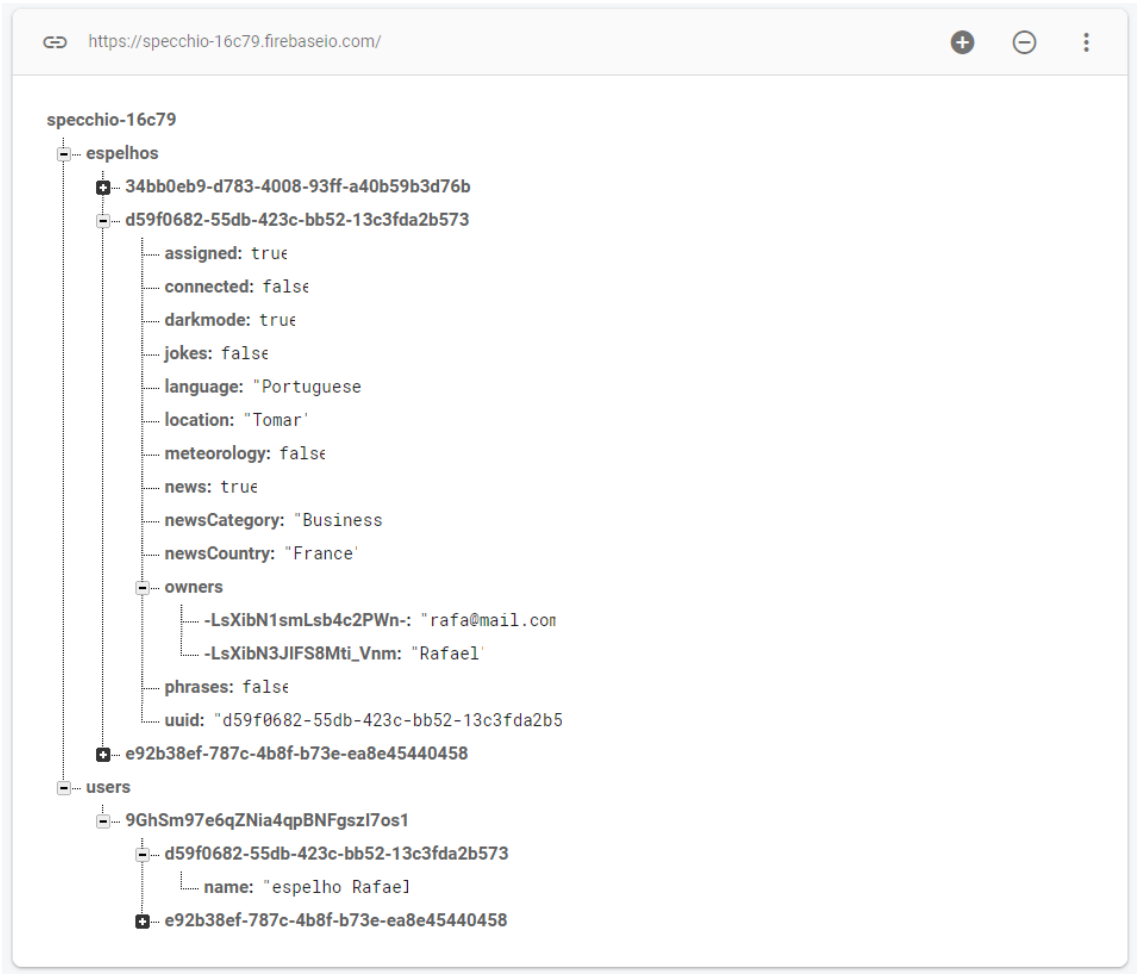


Figura 29 - Estrutura da Realtime Database

Na Figura 29 está demonstrada a estrutura da base de dados utilizada no projeto, que possui duas coleções, a coleção "espelhos" e a coleção "*users*". Na coleção espelhos são inseridos todos os espelhos criados na aplicação, isto é, quando a aplicação inicia pela primeira vez é criado um espelho automaticamente com os parâmetros por defeito. Um dos atributos dessa coleção é o atributo *owners*, indicando o "dono" do espelho, que é preenchido quando o utilizador regista o espelho como seu, através do QR Code anteriormente referido. A coleção users tem como intuito ser o registo dos espelhos que cada utilizador registou na aplicação, criando assim a ligação espelho-utilizador.

Apesar de todos os benefícios da plataforma Firebase, esta também possui limitações. Uma das suas limitações é a existência de planos distintos com diferentes funcionalidades e características. Um dos exemplos é o facto de no plano gratuito ser possível treinar um modelo apenas durante 1 hora e cujo conjunto de imagens não

ultrapasse as mil imagens, relativamente à ferramenta Firebase ML Kit. A execução de chamadas diretamente na *cloud* também implica custos nalgumas ferramentas. Como indica a Figura 30, existe o plano Spark, Flame e Blaze[68].

Figura 30 - Planos Firebase[69]

Outra das lacunas da plataforma Firebase é o estado atual das ferramentas, estando muitas ainda na fase *beta*[70]. Um dos exemplos dessas lacunas é o treino do modelo na ferramenta Cloud AutoML, cujo treino por via de uma interface ainda não é possível. Isto implica a não atualização do modelo de reconhecimento quando um novo utilizador é criado no sistema. Logo, por cada vez que se cria um novo utilizador, é necessário criar um novo modelo com as imagens da face do novo utilizador.

4.5. Aplicação *mobile*

Tal como indica o tópico 1.3 um dos objetivos planeados era a construção de uma aplicação *mobile* para registo e configuração de um espelho. Essa aplicação designada miSpecchio possui diversas componentes e funcionalidades. Ao entrar na aplicação, o utilizador depara-se com uma janela de autenticação, pela qual pode criar uma conta ou iniciar sessão, como constatamos na Figura 31.

[68] https://firebase.google.com/pricing [consultado em 12/09/2019]
[69] Imagem retirada de https://firebase.google.com/pricing [consultado em 12/09/2019]
[70] Fase de testes de usabilidade de um *software*

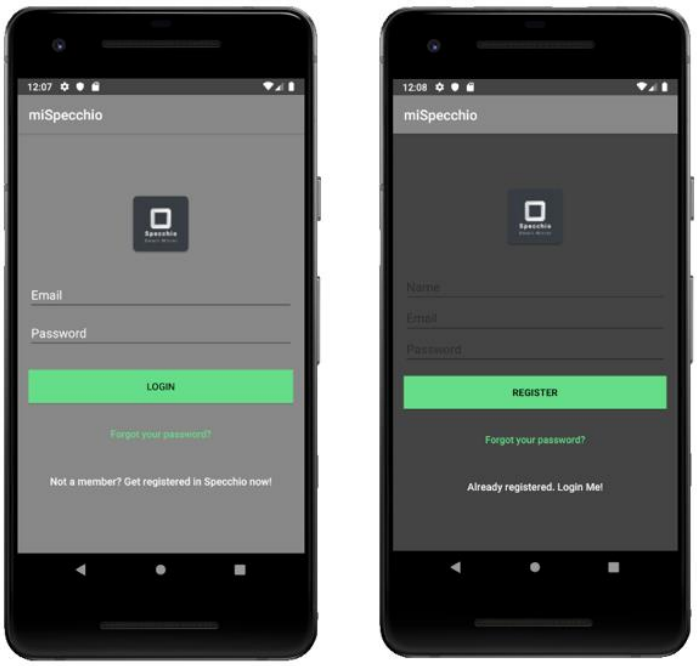


Figura 31 - miSpecchio Autenticação

Após iniciar sessão é apresentado uma lista de espelhos associados ao seu perfil, um botão para terminar a sessão e outro botão para registar um novo espelho (Figura 32).

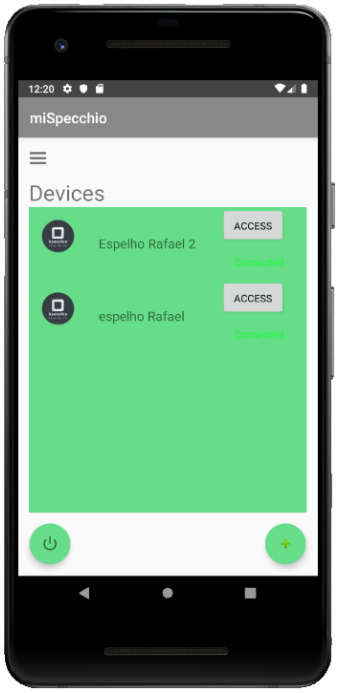


Figura 32 - miSpecchio Página Inicial

Em cada item da lista de espelhos, é possível aceder à configuração do espelho selecionado, por meio do botão "*ACCESS*". Na janela de configuração estão

representados os parâmetros das diversas componentes representadas no *dashboard* do espelho. É possível modificar as seguintes especificações:

- idioma do *dashboard*
- visualizar notícias ou não
- escolher o país da notícia
- escolher categoria da notícia
- visualizar a meteorologia ou não
- ativar modo *dark*
- visualizar frases inspiradoras ou não
- visualizar anedotas ou não

Se o utilizador pressionar no botão Guardar, as configurações desse espelho serão guardadas e atualizadas no Firebase em tempo real, como verificamos na Figura 33.

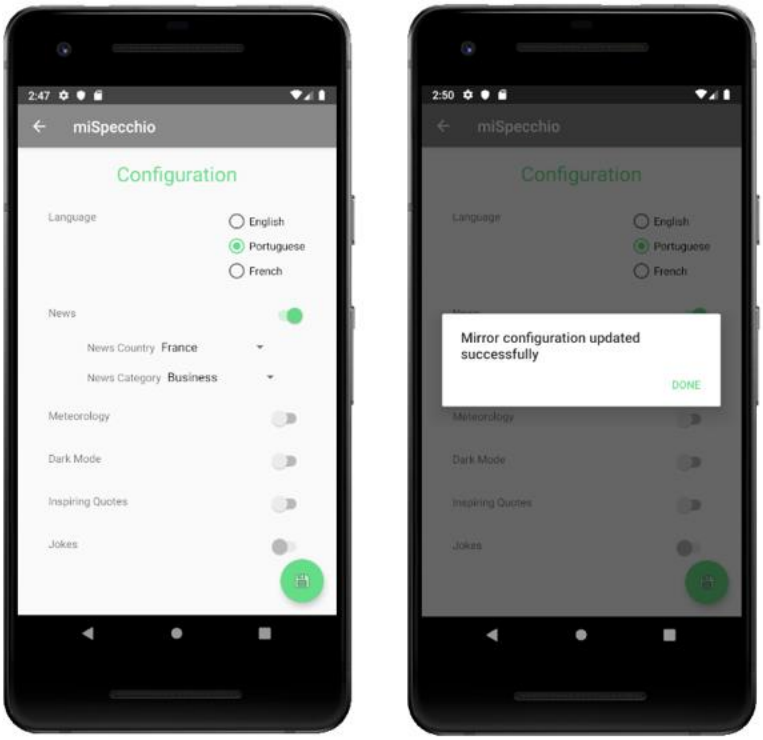


Figura 33 - miSpecchio Configuração de um espelho

Como referenciado no tópico anterior, miSpecchio regista um espelho por meio da leitura de um QR Code. Tal ação é executada a partir do botão de registo na página principal. Para tal efeito, basta pressionar o botão de registo e apontar o smartphone para o espelho para este detetar o QR Code apresentado no dashboard do espelho, como representado na Figura 34. Após ler o QR Code, é solicitado ao utilizador que insira um

nome para o dado espelho, de forma a guardar essa informação no Firebase. Após essa solicitação esse espelho é registado no seu perfil e através da aplicação miSpecchio é possível configurá-lo remotamente.

Figura 34 - miSpecchio Leitura de QR Code

4.6. Arquitetura Geral e Fluxos

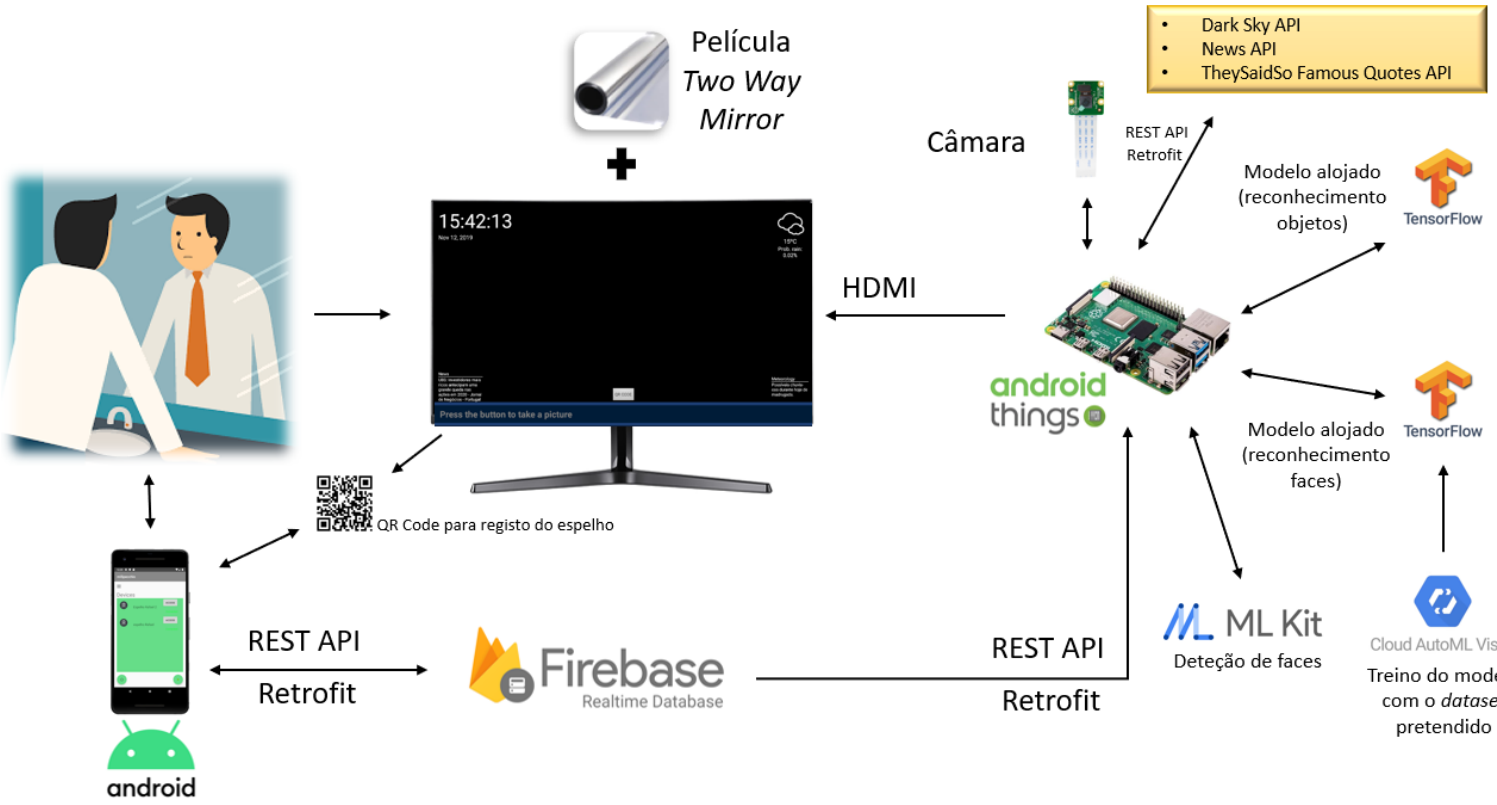


Figura 35 - Arquitetura geral e fluxos do sistema

Na Figura 35 está representado a arquitetura geral do sistema, com os seus respetivos fluxos. De seguida estão identificados os fluxos em geral, desde a utilização ao reconhecimento facial:

1. O utilizador depara-se com o espelho inteligente (um monitor que possui uma película refletora)
2. Através da aplicação móvel pode registar (através da leitura do QR Code) e configurar o espelho
3. A aplicação miSpecchio acede à base de dados Firebase e à Firebase Authentication para consultar e modificar os dados do utilizador e espelho, por meio de uma chamada *REST*
4. A aplicação Specchio que está no Raspberry Pi tira uma foto ao utilizador com a câmara que está acoplada (após o utilizador pressionar um botão situado no Raspberry Pi, para evitar que o sistema esteja a processar imagens (poupança de energia))

5. Utilizando a ferramenta ML Kit, Specchio verifica se existem e qual a quantidade de faces presentes na imagem capturada
    a. Ao mesmo tempo, através do modelo TensorFlow alojado na aplicação, realiza o reconhecimento de objetos na imagem capturada
6. Se não detetar ou detetar mais de 1 face na imagem:
    a. Specchio mostra as informações por defeito no monitor
7. Caso detete 1 face na imagem:
    a. Specchio recorre ao modelo Tensorflow alojado na aplicação criado e treinado através da ferramenta Cloud AutoML
    b. Caso a face detetada seja reconhecida no modelo, Specchio recebe da base de dados Firebase as configurações do utilizador e do espelho em específico
    c. Caso não seja reconhecido, Specchio mostras as informações por defeito no monitor
8. Specchio recorre agora às diferentes *APIs* de informação (meteorologia, notícias, frases, etc) e envia-as para o monitor por meio de HDMI

Na Figura 36 está representado o protótipo do sistema criado, onde é possível ver o Raspberry Pi com a aplicação Specchio (com a câmara acoplada), o monitor (espelho) sem a película espelhada e o smartphone com a aplicação miSpecchio. De realçar que durante este desenvolvimento foi utilizado um *Starter Kit* de Android Things que já possui diversas componentes (incluindo o Raspberry Pi), facilitando assim o desenvolvimento em todas as suas vertentes. A utilização deste Starter Kit teve influência na escolha entre Raspberry Pi e NXP i.MX7D, dispositivos legíveis para o sistema operativo Android Things, como abordado na secção 2.4.

Figura 36 - Protótipo sem a película inserida

Capítulo 5

Testes e resultados

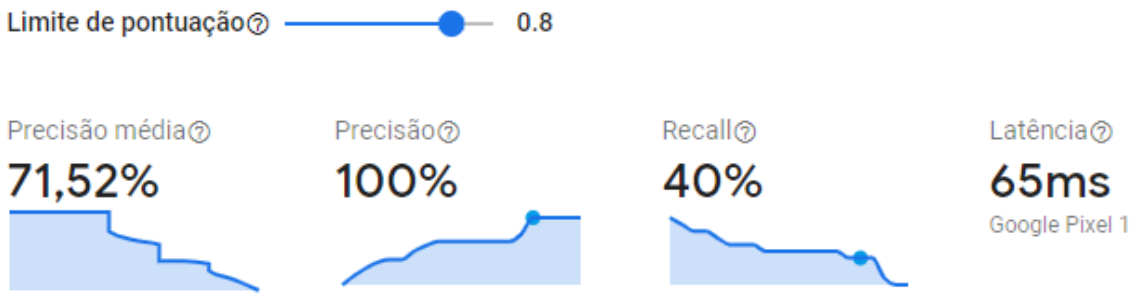


Figura 37 - Detalhes da avaliação[71]

Os testes e resultados do reconhecimento de imagem utilizando a ferramenta AutoML do Firebase estão representados na Figura 37 com os detalhes da avaliação correspondendo ao modelo em utilização. Esta ferramenta fornece um limite de pontuação que consiste em determinar o melhor valor da curva de compensação de precisão-*recall* em relação a este modelo. Por outras palavras é a confiança mínima que o modelo precisa de ter para atribuir um rótulo a uma imagem. O desempenho deste modelo é afetado por o valor dessa confiança mínima, sendo este medido através de duas métricas, precisão e *recall*, métricas estas abordadas no tópico 1.4. Nessa mesma figura, é demonstrado o valor de confiança utilizado no projeto, 0.8. Esse valor resulta numa precisão média de 71,52%, uma precisão de 100% (menos falsos positivos), um *recall* de 40% (menos falsos negativos) e uma latência de 65ms (tempo necessário de uma previsão) num dispositivo Google Pixel 1.

Na Figura 38 estão representados os detalhes de avaliação caso o limite de pontuação fosse 0.75 e 0.9, respetivamente.

[71] Imagem retirada da ferramenta AutoML

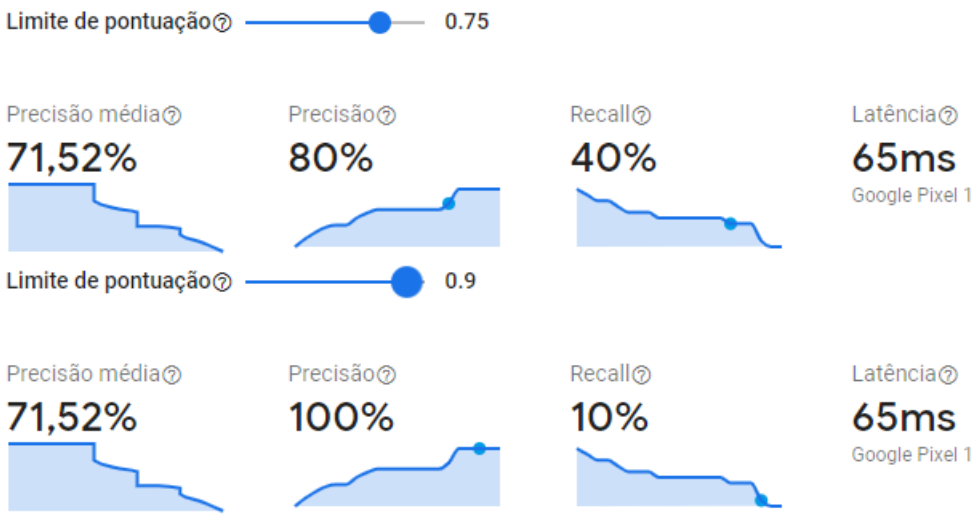


Figura 38 - Variância do limite de pontuação[72]

Aplicando a ferramenta ML Kit é possível testar a eficácia do modelo gerado arrastando uma nova imagem para o *dashboard* da plataforma Firebase. Desta forma, utilizou-se uma imagem que não se encontra no *dataset* de treino e na Figura 39 estão representados os resultados do reconhecimento de dita imagem. Com cerca de 83%, o modelo reconheceu na imagem fornecida o utilizador Rafael. Uma das desvantagens desta ferramenta é a impossibilidade de disponibilizar diferentes parâmetros relacionados com o treino.

Figura 39 - Teste de reconhecimento de imagem

Além dos detalhes de avaliação, a ferramenta AutoML fornece ainda a matriz de confusão do modelo utilizado, como representado na Figura 40. Ao analisar essa matriz, constatamos a frequência que o modelo classificou cada rótulo de forma correta e os rótulos mais confundidos.

[72] Imagem retirada da ferramenta AutoML

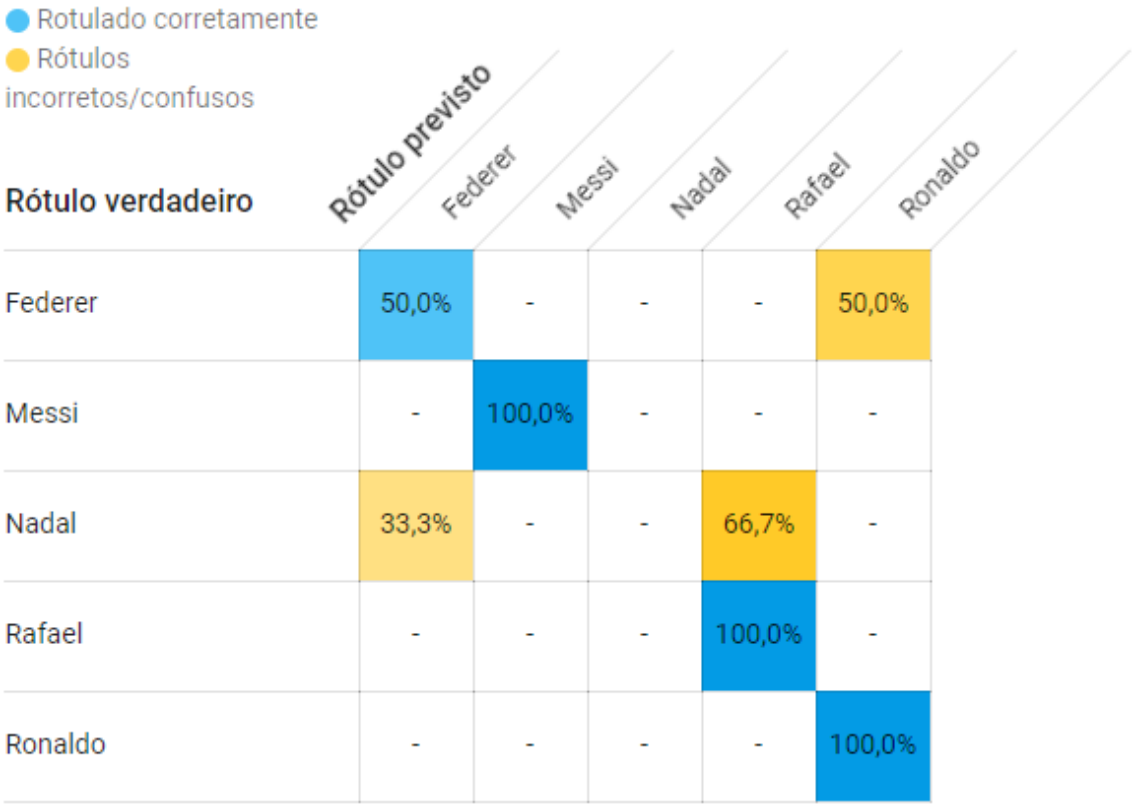


Figura 40 - Matriz de confusão de Specchio[73]

[73] Imagem retirada da ferramenta AutoML

Capítulo 6

Conclusões e trabalho futuro

User Experience é um termo técnico que hoje é moda no seio da sociedade. Uma boa experiência em termos de usabilidade e visual é o que as pessoas solicitam e é o que Specchio pretende oferecer.

Graças à funcionalidade de reconhecimento facial, Specchio destaca-se dos outros projetos de espelhos inteligentes devido ao conteúdo dinâmico que se adapta à pessoa que o acede. O acesso e a sua configuração por meio da aplicação *mobile* é uma característica que o distingue dos restantes projetos que neste momento se encontram no mercado e em desenvolvimento.

Por meio de uma aplicação móvel Android, uma aplicação de Android Things e uma plataforma proveniente da Google que possui ferramentas de deteção e reconhecimento de imagem conseguiu-se realizar o objetivo inicial com sucesso, que era criar uma solução tecnológica funcional que envolvesse estas três áreas, Internet das Coisas, *Mobile* e *Machine Learning*.

Neste trabalho, conseguimos comprovar que não é necessário ser dotado de um vasto conhecimento em *machine learning* e *deep learning* para implementar essas tecnologias nos diversos projetos que desenvolvemos. Tais tecnologias estão disponíveis em diversas plataformas e desenvolvidas para funcionar em dispositivos com baixo poder de processamento.

Para trabalho futuro fica o treino do modelo de forma dinâmica, isto é, quando o utilizador cria uma conta na aplicação miSpecchio é solicitado que registe diversas fotos da sua face (com diferentes ângulos faciais), inserindo-as diretamente no modelo de reconhecimento facial com a respetiva rotulagem. Além disso, seria interessante comparar a eficácia de reconhecimento facial utilizando para isso outras plataformas além da Google, utilizando para isso a mesma estrutura de dados.

Referências

A., J., T., P., D., S. S., & Dokhale, S. A. (2018). A Review Paper Design and Development of a Smart Mirror Using Raspberry Pi.

Gorodnichy, D. O., Granger, E., & Radtke, P. (2014). Survey of commercial technologies for face.

Jose, J., Chakravarthy, R., Jacob, J., Ali, M. M., & D'souza, S. M. (2017). Home Automated Smart Mirror as an Internet of Things (IoT) Implementation.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. pp. 8-17.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks.

Le, Q. V., & Zoph, B. (2017). Neural Architecture Search with Reinforcement Learning.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., . . . Murphy, K. (2018, Julho 26). Progressive Neural Architecture Search.

Paykar, M. (n.d.). *MakeUseOf*. Retrieved Dezembro 2018, from https://www.makeuseof.com/tag/6-best-raspberry-pi-smart-mirror-projects-weve-seen-far/

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Jeff, D. (2018, Fevereiro 12). Efficient Neural Architecture Search via Parameter Sharing.

Phillips, P. J. (n.d.). Support Vector Machines Applied to Face Recognition.

Real, E., Moore, S., Selle, A., Saxena , S., Suematsu , Y. L., Tan, J., . . . Kurakin, A. (2017). Large-Scale Evolution of Image Classifiers.

Schapire, R. E. (1990). *The Strength of Weak Learnability*.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering.

Simonyan, K., Parkhi, O. M., Vedaldi, A., & Zisserman, A. (n.d.). Fisher Vector Faces in the Wild.

Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (n.d.). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition.