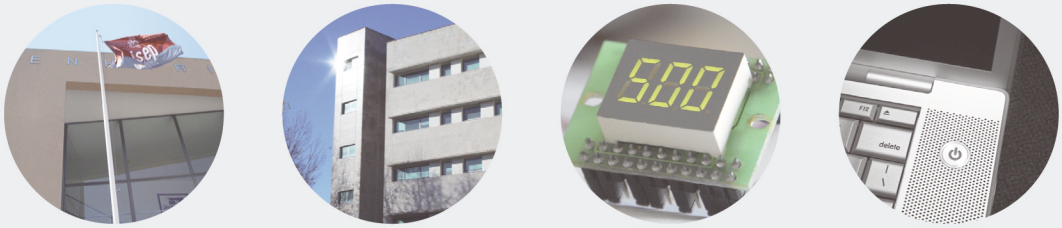
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de Apoio para Medicação de Idosos com Recurso ao Reconhecimento de Fala e Facial

RÚBEN IVAN RIBEIRO PINTO
novembro de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Sistema de Apoio para Medicação de Idosos com Recurso ao Reconhecimento de Fala e Facial

Ruben Ivan Ribeiro Pinto

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2023

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Ruben Ivan Ribeiro Pinto, Nº 1170839, 1170839@isep.ipp.pt

Orientação Científica: Lino Figueiredo, lbf@isep.ipp.pt

Coorientação Científica: Antonio Meireles, aem@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Novembro, 2023

Agradecimentos

Queria começar por agradecer ao engenheiro Antonio Meireles e Lino Figueiredo pelo interesse e apoio dado, o que foi, sem dúvida, uma chave essencial para a conclusão do meu projeto.

Devo congratular o Instituto Superior de Engenharia do Porto pelo excelente ensino que me foi proporcionado e pelas pessoas com quem partilhei várias experiências ao longo destes seis anos, tanto amigos como colegas de curso e todos eles contribuíram de alguma forma para o meu sucesso.

Quero agradecer a Raquel por todo o apoio e paciência proporcionada durante esta vida académica com especial atenção em momentos decisivos como a elaboração desta dissertação.

Mas, acima de todos, quero agradecer a minha mãe e irmão pelo esforço investido ao longo de toda a minha vida para que eu tivesse as melhores condições, porque nunca me faltaram com nada e sempre mostraram toda a preocupação e apoio, independentemente das minhas escolhas de carreira ou de estudo.

Resumo

Nesta dissertação, apresenta-se o desenvolvimento de um protótipo de um dispensador de comprimidos com o objetivo de combater o problema comum de não cumprimento e falta de adesão a regimes de medicação. De modo a tornar o dispositivo mais autónomo, foram exploradas as crescentes capacidades das redes neuronais e o poder computacional de microcomputadores como o Raspberry Pi.

Investigou-se, assim, a integração de modelos de Reconhecimento Facial e Reconhecimento de Fala nesses microcomputadores. Estas soluções oferecem aprimoramentos significativos, incluindo maior segurança e uma experiência de interação mais acessível para os utilizadores.

Durante o desenvolvimento deste projeto, foi efetuado o treino de modelos de Reconhecimento Facial e avaliado o seu desempenho por meio de métricas robustas, como o F1-Score e a curva ROC. Estudou-se, também, como estes modelos podem ser aplicados eficazmente num sistema de dispensação de medicamentos e em microcomputadores. Além disso, realizou-se uma análise abrangente da aplicação de modelos de Reconhecimento de Fala pré-treinados.

Este trabalho representa um passo importante na melhoria da eficácia e acessibilidade dos sistemas de dispensação de medicamentos, contribuindo para uma maior adesão dos pacientes aos seus regimes de medicação.

Palavras-Chave: IA, DPL, Reconhecimento Facial, Raspberry Pi, Reconhecimento de Fala

Abstract

This dissertation presents the development of a prototype pill dispenser with the aim of combating the common problem of non-compliance and lack of adherence to medication regimes. In order to make the device more autonomous, the growing capabilities of neural networks and the computing power of microcomputers such as the Raspberry Pi were explored.

The integration of Facial Recognition and Speech Recognition models into these microcomputers was investigated. These solutions offer significant improvements, including greater security and a more accessible interaction experience for users.

During the course of this project, Facial Recognition models were trained and their performance evaluated using robust metrics such as the F1-Score and the ROC curve. We also studied how these models can be effectively applied to a drug dispensing system and microcomputers. In addition, a comprehensive analysis of the application of pre-trained Speech Recognition models was carried out.

This work represents an important step towards improving the effectiveness and accessibility of drug dispensing systems, contributing to greater patient adherence to their medication regimes.

Keywords: IA, DPL, Facial Recognition, Raspberry Pi, Speech Recognition

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Acrónimos

ANN	*Artificial neural networks*
CNN	*Convolutional neural networks*
DPL	*Deep Learning*
FN	*False Negatives*
FP	*False Positives*
HMM	*Hidden Markov Model*
IA	*Inteligência Artificial*
LFW	*Labeled Faces in the Wild*
ML	*Machine Learning*
ONU	*Organização das Nações Unidas*
PCB	*Printed Circuit Board*
RFID	*Radio Frequency Identification*
RNN	*Recurrent Neural Networks*
ROC	*Receiver Operating Characteristic*
TN	*True Negatives*
TP	*True Positives*
UML	*Unified Modeling Language*
WAcc	*Word Accuracy*
WER	*Word Error Rate*

Capítulo 1

Introdução

1.1 Contextualização

Desde a década de 1950 tem-se assistido a um processo de envelhecimento populacional, ou seja, a percentagem de idosos mundialmente é cada vez mais elevada. Em 2022, a *Organização das Nações Unidas* (ONU) revelou as projeções populacionais mundiais, que podem ser observadas na Figura 1.1. Pode verificar-se que o processo de envelhecimento irá continuar, e que a sua fase mais acentuada acontecerá nos próximos anos. Estima-se que o número de idosos, com 60 anos ou mais, apresentará uma variação de 962 milhões em 2017, para 2,2 mil milhões em 2050 e 3,1 mil milhões em 2100 [1].

Este envelhecimento será uma das transformações mais significativas do século XXI e implicará todos os sectores da sociedade, tal como o setor financeiro, o dos transportes e o da proteção social [1].

Portugal não é exceção, segundo os últimos censos realizados em 2021 a percentagem de população idosa (65 anos ou mais) aumentou, sendo atualmente 23,4% [2]. Assim, Portugal é o terceiro país com o índice de envelhecimento mais elevado da União Europeia. O peso dos idosos na estrutura populacional, tem aumentado de forma significativa, devido por um lado à diminuição dos nascimentos e por outro ao aumento da esperança de vida [3].

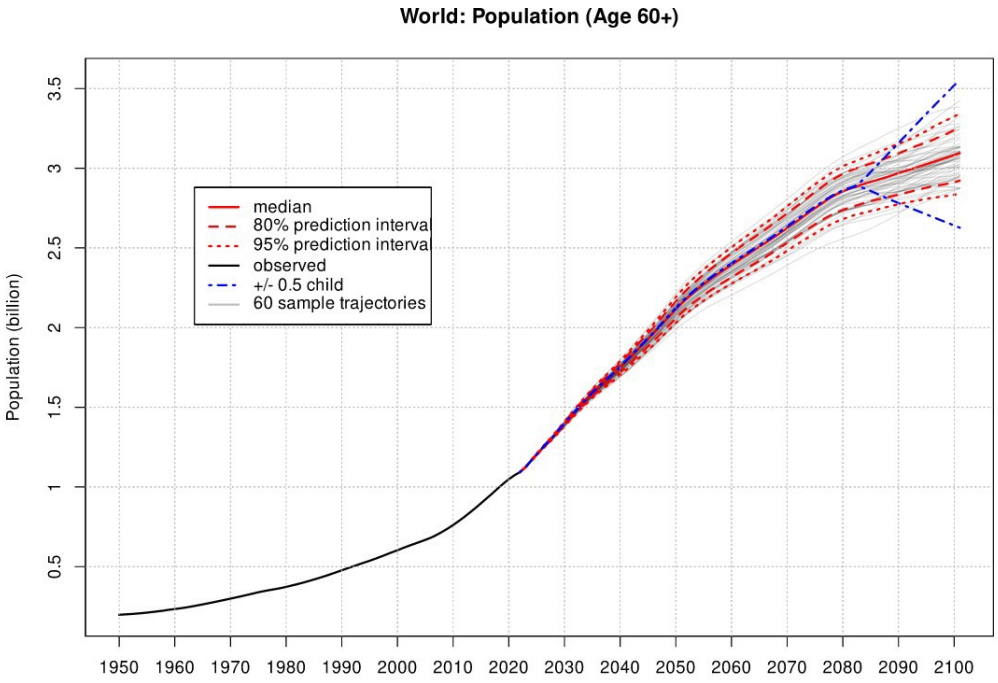


Figura 1.1: População absoluta de idosos de 60 anos e mais [4].

Devido à sua vulnerabilidade, as pessoas idosas apresentam maior prevalência de consumo de medicamentos, tanto prescritos quanto não prescritos. Essa prevalência deve-se a vários fatores, como doenças crónicas, doenças degenerativas, alterações na capacidade cognitiva e motora.

Para alguns pacientes torna-se difícil cumprir um regime de medicação complexo, especialmente para aqueles com uma vasta gama de medicamentos, com diferentes dosagens e horários de administração. Pacientes com Alzheimer, por exemplo, são muito propensos a descuidarem-se da medicação e muitas vezes esquecem-se completamente de tomá-la.

O não cumprimento e a não adesão a um regime de medicação pode trazer graves consequências para o paciente, como complicações no estado de saúde que muitas vezes levam à necessidade de hospitalização e até mesmo à morte [5].

Em 2023 a Deco proteste inquiriu 1151 portugueses (Figura 1.2), destes 29% sofreram um erro de medicação. O mais comum, foi não seguir o tratamento até ao fim, indicado por 38% dos inquiridos. Seguindo-se desrespeitar o horário das tomas (31%), abstenção das restrições na dieta recomendadas pelo médico (27%), tomar fármacos fora de prazo ou em condições suspeitas(26%), e não respeitar as regras da toma (29%) [6].

Figura 1.2: Erros na administração de medicação [6].

Estima-se que mais de 100.000 pessoas morrem anualmente nos Estados Unidos parcialmente devido à falta de adesão ao tratamento prescrito, um em cada cinco (21%) dos pacientes nunca segue a sua prescrição e um em cada vinte (6%) não consegue identificar os seus próprios medicamentos. E entre 12 e 20% tomam remédios de outros pacientes.

Comprova-se igualmente que a falta de adesão à medicação prescrita é clinicamente significativa em 50% dos pacientes. A falta de adesão deve-se à falta de compreensão de como tomar os medicamentos (via oral, várias vezes ao dia, antes/-depois das refeições), a não perceber a importância do tratamento para a melhoria da saúde, à necessidade de tomar vários comprimidos de uma só vez, a problemas de mobilidade, entre outros motivos [7].

Um estudo em 256 pacientes ingleses e 249 pacientes espanhóis, verificou que 15% destes não sabiam ler e interpretar instruções para a toma por via oral quatro vezes ao dia, e 37% não compreendiam a instrução de toma de um medicamento com estômago vazio [8].

Apesar de já existirem inúmeros dispositivos de auxílio na gestão e toma dos fármacos, observa-se que grande parte destes não são adequados, quer seja por questões relacionadas com a usabilidade ou por incompatibilidade com o nível de literacia em saúde [8].

Diante destes problemas, surge a necessidade de ajudar estes pacientes, por meio do desenvolvimento de um sistema *Inteligência Artificial* (IA) que dispense os medicamentos automaticamente num horário previamente programado por um cuidador, permitindo que o paciente não se preocupe com a gestão da sua própria medicação. Tudo isso visa melhorar a qualidade de vida das pessoas idosas com algumas dificuldades cognitivas e permitir uma vida menos dependente de terceiros [9].

1.2 Motivação

Estudos demonstram que as aptidões cognitivas atingem o seu pico por volta dos 30 anos, permanecem estáveis até à idade dos 50 – 60 anos e a partir desta tendem a diminuir, apresentando-se uma diminuição ainda mais substancial a partir dos 70 anos [10].

A complexidade dos esquemas terapêuticos, as alterações neuros sensoriais, como o deterioramento da visão, e o declínio cognitivo, tornam ainda mais complexo e difícil o uso apropriado dos medicamentos nesta população [5]. Tanto o abuso como a falta de um medicamento podem resultar em situações críticas e potencialmente fatais.

Com o crescimento acentuado da população idosa e a constante evolução nas aplicações de tecnologia no setor da saúde, surge a possibilidade de facilitar a organização e metodologia utilizada no consumo de medicação, nomeadamente de comprimidos.

Neste sentido esta dissertação aspira desenvolver um protótipo de um dispensador de medicamentos automático. Desta forma, foram estudadas diferentes abordagens de maneira a identificar quais os benefícios e diferenças entre as mesmas e como podem ser aplicadas para dar uma resposta aos problemas atuais.

1.2.1 Objetivos

O projeto, onde está inserida esta dissertação, visa criar um sistema de medicação inteligente, que deverá resultar da integração de três subsistemas principais:

- subsistema de reconhecimento e segurança;
- subsistema de interação com o utilizador;
- subsistema de ação.

O primeiro irá consistir num modelo de reconhecimento facial e comparação de imagens utilizando *Deep Learning* (DPL) de maneira a validar o utilizador que interage com o modelo. O desenvolvimento do subsistema terá em conta as especificidades dos utilizadores, assim sendo, não será necessária a recolha de uma elevada quantidade de fotos, não sobrecarregando, deste modo, o utilizador com configurações.

O segundo subsistema irá apresentar diversas funções de interação. Face às dificuldades que as pessoas demonstram cada vez mais em interagir com sistemas complexos, ações como saber as horas e efetuar cálculos simples podem ser executadas via comandos de voz.

Por fim, o terceiro subsistema será composto por um mecanismo que permite dispensar a medicação (comprimidos) que é fornecida ao utilizador.

1.2.2 Resultados esperados

Como resultado, é esperado um protótipo que consiga conectar estes 3 subsistemas. É, ainda, esperada uma taxa de sucesso nos modelos IA superior a 80%. Este prototipo será elaborado utilizando tecnologias *open-source* e tendo em conta a futura reprodução utilizará componentes de fácil acesso.

1.3 Organização da Dissertação

A presente dissertação está dividida em cinco capítulos, tal como descrito seguidamente.

O primeiro capítulo é dedicado à introdução da dissertação. É feita a contextualização do tema da dissertação, a motivação, os objetivos e o resultado esperado.

O segundo capítulo integra a metodologia, onde é feito o enquadramento teórico dos conceitos que suportam a dissertação, desde o reconhecimento facial até ao reconhecimento de fala.

O terceiro capítulo aborda o desenvolvimento efetuado ao nível de *software* na dissertação, onde são apresentados os modelos de *Inteligência Artificial* (IA) desenvolvidos, bem como toda a arquitetura de *software*. É, também, exposta a escolha de *hardware* e as razões que motivaram essa escolha. Apresentando-se, ainda, a construção do prototipo final e a interligação do *software* com o *hardware*.

Por último, no quarto capítulo são expostas as conclusões obtidas, sendo apresentada uma análise ao trabalho desenvolvido ao longo da dissertação, com uma avaliação crítica aos resultados obtidos e cumprimento dos objetivos inicialmente propostos.

Capítulo 2

Metodologia

Neste capítulo são apresentados os conceitos de reconhecimento de fala e reconhecimento facial, bem como o estado atual dos dispensadores automáticos de comprimidos.

2.1 Inteligência Artificial

A maioria dos estudos defende que a IA surgiu quando os primeiros robôs foram criados. A palavra robô surgiu pela primeira vez na peça "R.U. R" (*Rossum's Universal Robot*) escrita por *Karel Capek* em 1921. Na peça esta palavra retratava uma fábrica onde máquinas eram utilizadas para fazer o trabalho laboral dos funcionários. Já no meio do século XX Isaac Asimov imortaliza a palavra robô, quando a utiliza numa coleção de histórias de ficção cientifica. Embora estas tenham sido as primeiras menções da palavra robô já há muito tempo que a sociedade convivia com esta ideia. O primeiro autómato humanoide remonta ao terceiro século na China, quando o engenheiro mecânico Yan Shi apresenta ao imperador Mu of Zhou uma figura mecânica em forma de humano feita com couro, madeira e órgãos artificiais [11].

O termo IA surgiu em 1955 quando Jonh McCarthy referiu que o mesmo se traduz na ciência e engenharia de criar máquinas inteligentes. Isto permitiu a criação de um novo ramo de investigação e estudo que seria o grande começo da IA. Durante os anos que correram os computadores foram ficando cada vez mais fortes e conseguindo resolver cada vez mais problemas matemáticos complexos até que em maio

de 1997 Big Blue, Figura 2.1, é capaz de derrotar o campeão mundial de Xadrez Gary Kasparov.

Figura 2.1: Computador *deep blue* [12].

Nos dias correntes IA é considerada um ramo da engenharia que implementa novos conceitos e novas soluções para resolver problemas complexos. Tendo evoluído bastante e possuindo uma série de subcategorias que podem ser visíveis na Figura 2.2 em forma hierárquica.

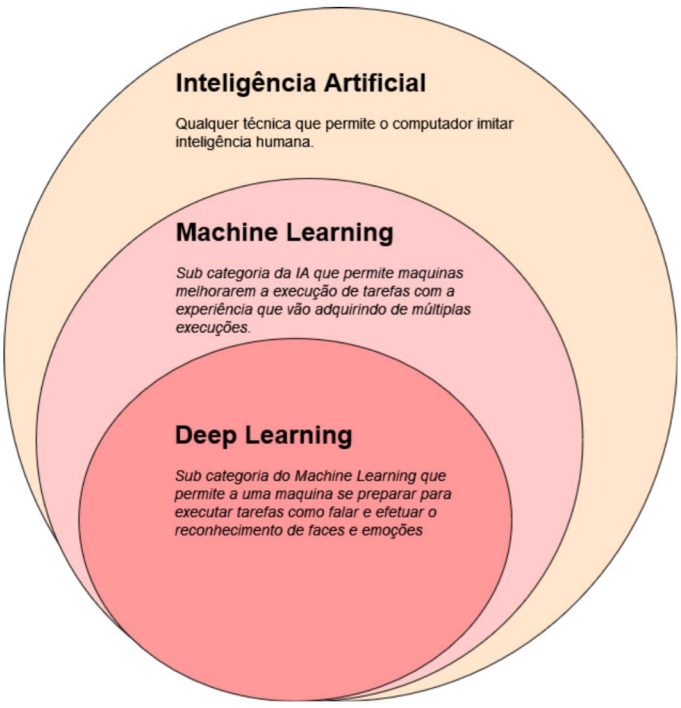


Figura 2.2: Camadas da IA [13].

2.1.1 Machine Learning

Machine Learning (ML) é o *subset* da IA que se foca no desenvolvimento de algoritmos que melhor representam um conjunto de dados. Observando a Figura 2.3 é possível comparar a diferença do desenvolvimento de um algoritmo clássico versos um algoritmo ML [14].

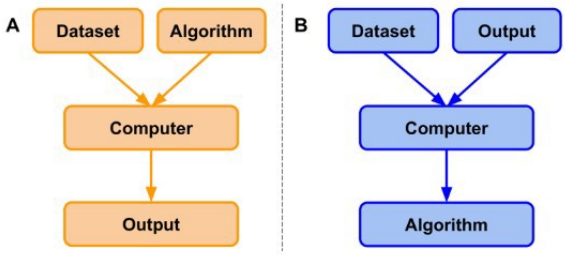


Figura 2.3: Comparação entre a programação clássica (A) e ML (B) [14].

Na programação clássica, Figura 2.3(A) o computador recebe um conjunto de dados e um algoritmo, que instruí o computador sobre como operar nesses dados para produzir resultados. Em ML, Figura 2.3(B), é fornecido ao computador um conjunto de dados e os resultados correspondentes. O computador aprende e cria um algoritmo que descreve a relação entre os dados e os resultados. Esse algoritmo pode ser usado para efetuar previsões sobre novos conjuntos de dados [14].

Características

De forma a padronizar os termos utilizados neste estado de arte, evitando a redundância ou uso incorreto destes, os mesmos encontram-se explicados abaixo com base no livro *Interpretable ML* escrito por *Christoph Molnar* [15]:

- Algoritmo: define-se como um conjunto de regras que uma máquina segue para atingir um determinado objetivo.
- Algoritmo de ML: é o programa utilizado para efetuar o treino de um modelo de ML a partir de um determinado *dataset*.
- Modelo ML: é o programa aprendido que mapeia entradas para previsões.
- *Dataset*: é um conjunto de dados utilizado para treinar um modelo. Também pode conter dados utilizados na fases de teste e avaliação.
- Tarefa: é a combinação de um conjunto de dados com recursos e um destino. Dependendo do tipo de destino, a tarefa pode ser, por exemplo, classificação, regressão, análise de sobrevivência, agrupamento ou deteção de valores discrepantes.

Tipos de aprendizagem

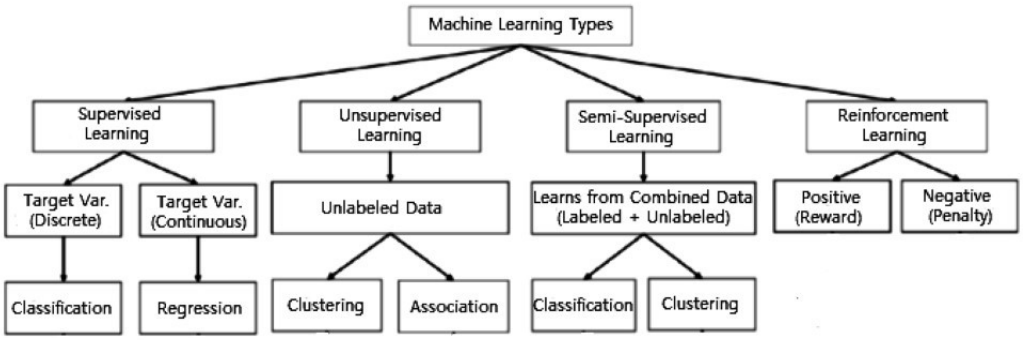


Figura 2.4: Diferentes métodos de aprendizagem presentes em ML [16].

Em ML existem quatro métodos principais de aprendizagem, dependendo do problema em questão o método de aprendizagem que melhor se aplica pode mudar, como se verifica na Figura 2.4. Para melhor perceber cada um destes métodos estes serão explicados abaixo:

- **Aprendizagem supervisionada:** é definida pelo uso de conjunto de dados rotulados para treinar algoritmos. O objetivo é aprender regras gerais que mapeiam entradas para saídas, para ser possível prever a saída para novos dados não estudados. Um problema de aprendizagem supervisionada pode ser de duas categorias. A primeira é a classificação quando a solução é categórica, o que acontece normalmente na classificação de imagens. Por exemplo, um modelo que analisa uma foto e retorna se é um cão ou um gato. A segunda categoria é a regressão utilizada quando existe uma relação linear entre as variáveis de entrada e saída, um exemplo desta categoria é prever a temperatura tendo como base dados meteorológicos [17].

- **Aprendizagem não supervisionada:** utiliza algoritmos para analisar e agrupar conjuntos de dados não rotulados. Estes algoritmos descobrem padrões ocultos ou agrupamentos de dados sem a necessidade de intervenção humana. Esta capacidade torna este tipo de aprendizagem ideal para analisar dados [16]. Este tipo de aprendizagem é aplicado na segmentação de clientes onde, por exemplo, se pretende saber quais as faixas etárias que mais visitam uma loja [18]. A sua capacidade de descobrir semelhanças e diferenças nas informações torna-o a solução ideal para análise exploratória de dados, estratégias de venda cruzada, segmentação de clientes e reconhecimento de imagem.

- **Aprendizagem semi-supervisionada:** é uma mistura dos métodos mencionados anteriormente, atua em dados rotulados e não rotulados. É particularmente útil quando é difícil extrair características relevantes de dados e quando há um grande volume de dados. Algumas áreas de aplicação em que a aprendizagem semi-supervisionada é aplicada incluem a tradução automática e a deteção de fraudes [19].

- **Aprendizagem com reforço:** este tipo de aprendizagem é baseado em recompensas ou penalidades, *feedback* , e o seu objetivo final é utilizar as informações obtidas a partir da interação com o ambiente para apresentar opções que aumentem a recompensa ou minimizem o risco [16].

Problemas em algoritmos de ML

Alguns dos problemas que podem ser enfrentados no desenvolvimento de algoritmos de ML para além de uma escolha errada de dados são o *overfitting* e *underfitting*.

O *overfitting* acontece quando o modelo que está a ser treinado começa a capturar ruídos e variações aleatórias nos dados de treino o que faz com que o modelo não aprenda padrões e relações mais gerais entre os dados. Isto resulta num modelo demasiado específico para os dados de treino, que não apresenta um bom desempenho quando utilizado para fazer previsões ou classificações de novos dados. Uma forma de identificar *overfitting* é quando o modelo apresenta excelentes resultados nos dados de treino e maus resultados nos dados de validação e de teste [20, 21].

Já *underfitting* é exatamente o oposto e ocorre quando o modelo não consegue captar adequadamente os padrões e relações entre os dados de treino. Isto geralmente acontece quando o modelo construido é demasiado simples. É possível detetar este problema quando o modelo possui um mau desempenho tanto nos dados de treino como de validação e de teste [20, 22].

Fases de programação num algoritmo de ML

A elaboração e programação de um algoritmo de ML é compreendida por diversas fases observadas na Figura 2.5, estas podem ser agrupadas em 3 grupos descritos abaixo [23]:

- **Pré-processamento (*Data Management*):** Esta fase é responsável pela aquisição de dados e preparação dos mesmos para as próximas duas fases, assim sendo no final desta fase deve dispor-se de um *dataset* para ser utilizado no treino e um *dataset* para ser utilizado na verificação.

- **Treino (*Model learning*):** Nesta fase o especialista seleciona o modelo a ser utilizado com base no problema, no *dataset* e na sua experiência. Em seguida constrói uma função de perda para medir o erro durante o treino e

os parâmetros utilizados no mesmo. Quando o modelo atinge o desempenho pretendido pode passar-se para a próxima fase.

- **Avaliação (*Model Verification*):** O objetivo principal desta fase é garantir que o modelo tem um bom desempenho em novos dados. Normalmente são construidos gráficos conhecidos por matrizes de confusão, através destes é possível obter dados como a precisão do modelo.

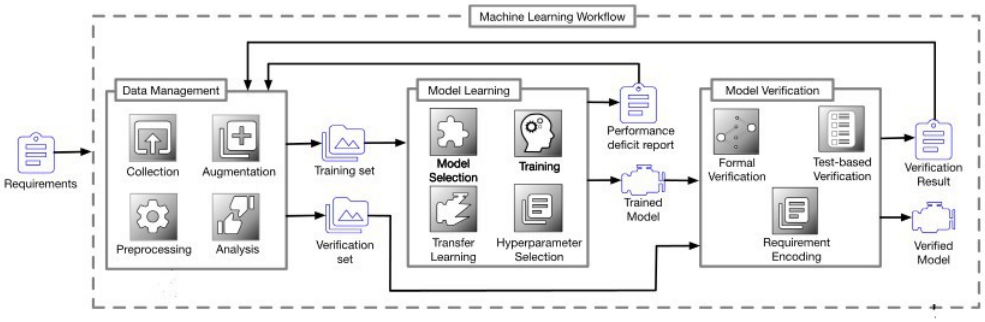


Figura 2.5: *ML Workflow*[23].

Data Augmentation

Data Augmentation é uma técnica que permite aumentar artificialmente o conjunto de dados disponíveis para a fase de treino, para isso são criadas cópias dos dados originais que sofrem modificações. Esta técnica costuma ser utilizada quando um *dataset* inicial é muito pequeno, para evitar *overfitting* ou quando se pretende reduzir o custo/tempo do processo de criação e organização do *dataset*. Algumas das técnicas aplicadas em imagens podem ser vistas na lista abaixo [24, 25]:

- Adicionar ruído
- Recortar
- Inverter
- Rotação
- Contraste
- Luminosidade
- Saturação
- Scaling

2.2 Artificial neural networks

As *Artificial neural networks* (ANN) são modelos computacionais inspirados no cérebro humano que são capazes de realizar ML bem como o reconhecimento de padrões, assim sendo, ANN são um *subset* de ML. Estas consistem numa série de algoritmos que tentam reconhecer correlações inerentes num conjunto de dados [26] [27].

Na Figura 2.6 são visíveis 4 camadas identificadas por: L_1, L_2, L_3, L_4, normalmente designadas como *layers*. A camada L_1 é chamada de camada de entrada e é responsável por receber informação. As camadas L_2 e L_3 são chamadas de camadas ocultas. Quando uma ANN tem mais de duas camadas ocultas passa a ser chamada de DPL. A última camada identificada por L_4 é responsável por produzir a saída final da rede, que pode ser uma classificação, uma previsão numérica ou uma sequência de valores.

Estas camadas consistem num conjunto de nós, cada um destes possui a sua própria pequena esfera de conhecimento, incluindo o que viu e quaisquer regras com as quais foi originalmente programado ou desenvolvido. Para além da característica mencionada anteriormente, estas camadas encontram-se também altamente conectadas, o que significa que cada nó na camada n será conectado a muitos nós na camada n-1 e na camada n+1, cada uma destas ligações possui um peso. Inicialmente este peso é aleatório e as respostas produzidas provavelmente não fazem sentido sendo necessário um processo de treino para que a rede se torne mais precisa[28].

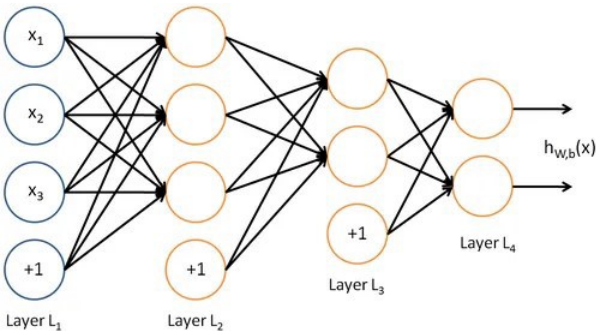


Figura 2.6: Arquitetura de uma ANN [28].

2.2.1 Tipos de Redes Neuronais

Convolutional neural networks

As *Convolutional neural networks* (CNN) são atualmente um dos tipos de ANN mais utilizadas e têm obtido excelentes resultados ao longo da última década numa variedade de campos relacionados com o processamento de imagem e reconhecimento de fala. O aspeto mais benéfico das CNNs é a capacidade de reduzir o número de parâmetros nas ANN. Este passo permitiu que ambos os *researchers* e os *developers* recorressem a modelos de IA para resolver tarefas complexas, o que não era possível com ANNs clássicas [29] [30].

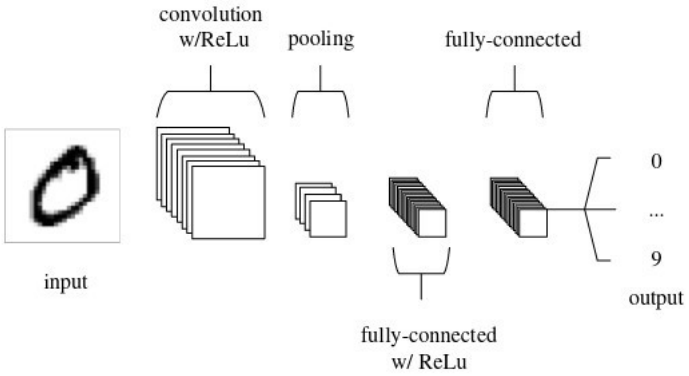


Figura 2.7: Arquitetura de uma CNN com cinco camadas

As CNN são compostas por cinco tipos de camadas: entrada, convolução, agrupamento, *fully-connected* e resposta [31]. Na Figura 2.7 é visível uma arquitetura simplificada utilizada para a classificação de MNIST(*Modified National Institute of Standards and Technology database*) [1] a funcionalidade da mesma pode ser descrita da seguinte forma:

A camada de entrada terá os pixeis da imagem. De seguida, a convolução é um componente fundamental que realiza a extração de características, que normalmente consiste numa combinação de operações lineares e não lineares, ou seja, operação de convolução e função de ativação. Conforme visível na Figura A.1 presente nos anexos, para esta extração de recursos é utilizada uma pequena matriz de números (kernel). Um produto elemento a elemento, entre cada elemento do kernel e os dados de entrada, é calculado e somado para obter o valor de saída na posição correspondente. Este procedimento é repetido aplicando vários kernels para formar um número arbitrário de mapas de recursos, que representam diferentes características. Dois hiperparâmetros principais que definem a operação de convolução são o tamanho e o número de kernels. O primeiro é tipicamente 3 * 3, mas por vezes recorre-se a 5 * 5 ou 7 * 7. Depois desta operação de convolução existe a função de ativação. Esta permite que a CNN capture informações mais complexas e não lineares dos dados de entrada. É, ainda, responsável por transformar a saída de cada camada da CNN num intervalo limitado de valores, importante para garantir a estabilidade numérica e a convergência do algoritmo de treino. Existem vários tipos de funções de ativação (Figura 2.9) comuns em CNN, como a função ReLU(Unidade Linear Retificada), a função sigmoide e a função tangente hiperbólica. A função ReLu é a mais utilizada e consiste em retornar o valor de entrada se ele for positivo e zero caso contrário. Isto permite tornar a rede mais eficiente e rápida em termos de tempo de treino [33].

[1] É uma base de dados com dígitos escritos a mão composta por mais de 60000 exemplos. É um *subset* da base de dados NIST que contem todos os dígitos com um tamanho padrão e centrados. [32]

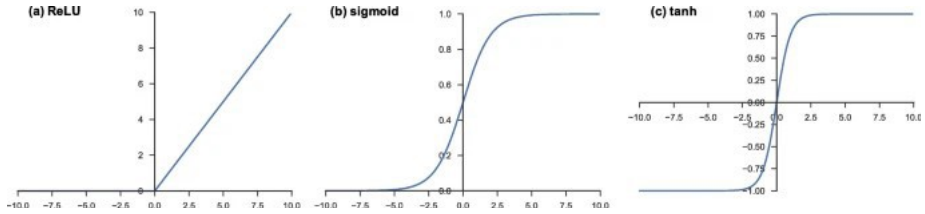


Figura 2.8: Tipos de funções de ativação [33].

A terceira camada é o agrupamento, nesta camada é efetuada uma redução do mapa de características, a fim de introduzir uma invariância mesmo quando a diferença de valores é pouca. Na prática, costumam ser utilizados 2 tipos de agrupamento que podem ser visíveis na (Figura 2.8). O *Max pooling* calcula o valor máximo dos fragmentos de um mapa de características e utiliza-o para criar um mapa de características com amostragem reduzida (agrupada). Outra operação digna de nota é o *Average pooling* que calcula o valor médio dos fragmentos de um mapa de características e utiliza-o para criar um mapa de características com amostragem reduzida. É normalmente utilizado após uma camada convolucional. Extrai características de forma mais suave que o *Max Pooling*, enquanto o mesmo extrai características mais notáveis como arestas. Ambos os métodos adicionam uma pequena quantidade de invariância de translação - o que significa que a translação da imagem por uma pequena quantidade não afeta significativamente os valores da maioria dos resultados agrupados [34].

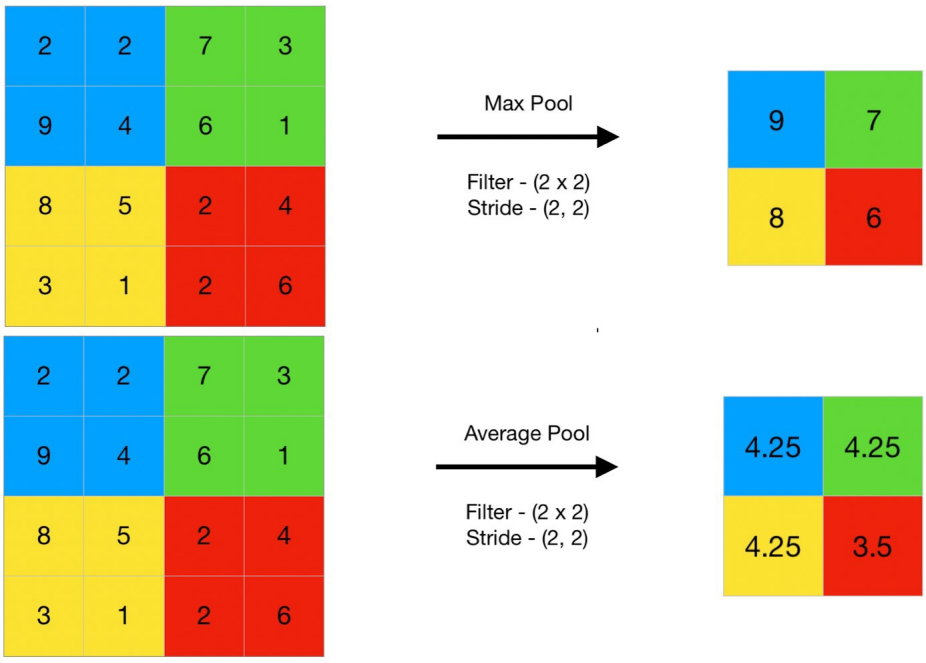


Figura 2.9: Exemplos gráficos da aplicação de *Max Pooling* e *Average Pooling* [34].

Recurrent Neural Networks

Uma *Recurrent Neural Networks* (RNN) é um tipo de ANN adaptada para trabalhar com dados de séries temporais ou dados que envolvam sequências, exemplos disso são aplicações como a Siri, *Google Translate* ou de deteção facial [35]. Existem diversos tipos de RNN, desde modelos que só produzem uma saída até modelos que podem produzir múltiplas. Um dos principais problemas com este tipo de ANN é a dificuldade para treiná-la e o custo computacional envolvido [36].

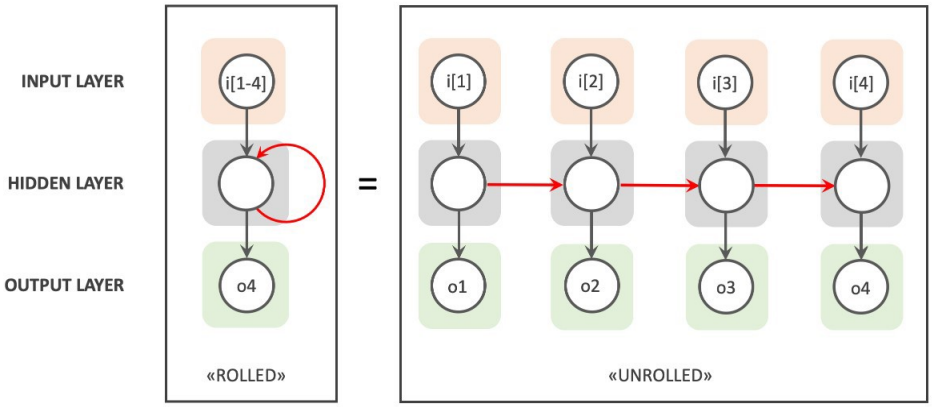


Figura 2.10: Esquema ilustrativo de uma RNN [37]

Um exemplo de uma RNN pode ser visualizado na Figura 2.10. No lado esquerdo da figura, pode ver-se uma RNN simples com nós de entrada, ocultos e de saída. Esta recebe uma entrada sequencial com 4 elementos (i[1-4]), percorre-os e produz um valor de o4. Em cada iteração, a Camada Oculta herda a memória de trabalho das iterações anteriores, conforme indicado pela seta vermelha.

No lado direito, observa-se uma representação "desenrolada"da mesma RNN, que apresenta os *loops* da RNN como uma cadeia de ANNs *Feed Forward* idênticas. Estas ANNs são idênticas, partilhando a mesma estrutura, pesos e funções de ativação. Mais uma vez, as setas vermelhas mostram como a memória de trabalho de cada iteração é passada para a próxima.[37].

Siamese Networks

É um tipo de arquitetura de ANN frequentemente utilizado em tarefas de reconhecimento facial, similaridade de texto e verificação de assinatura. Consiste na utilização de duas ANN idênticas que partilham os mesmos pesos. Ou seja, vão partilhar os mesmos valores utilizados para influenciar o resultado de saída para uma determinada entrada de dados, permitindo assim a ambas as redes aprender métodos de extração de características e padrões idênticos. Após ambas as redes obterem um conjunto de características, as mesmas são comparadas, mediante uma função de ativação sigmoide, a isto chama-se medir a distância entre as mesmas, de

maneira a determinar a sua similaridade. Essa distância é posteriormente utilizada para identificar se os dados de entrada são idênticos ou não [38, 39].

Normalmente para treinar este tipo de redes são utilizados pares de entradas onde a saída desejada já é previamente conhecida. Durante o treino a rede aprende a minimizar a distância entre entradas semelhantes e a maximizar a distância entre entradas diferentes, este procedimento é descrito na Figura 2.11 [38, 39].

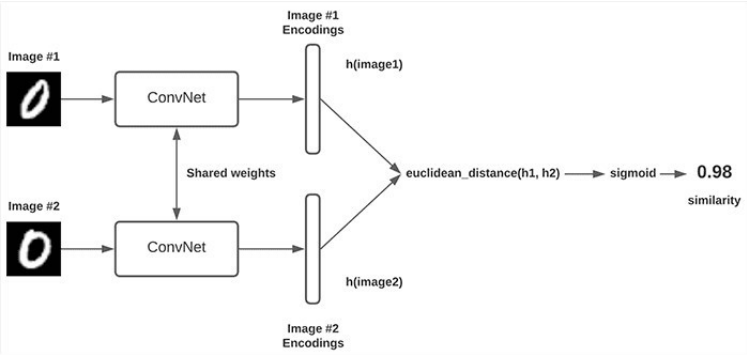


Figura 2.11: Implementação de uma *siamese network* para calcular a similaridade de duas imagens [40].

2.2.2 Análise do Modelo Treinado

Existem diversos métodos para avaliar a performance e desempenho do modelo treinado. Em todos os métodos explicados a seguir é considerado que o modelo produz uma classificação binária, o que significa que se aplicam os seguintes termos [41] [42]:

- ***True Positives* (TP)**: Quando o modelo retorna positivo e o pretendido era positivo.
- ***False Positives* (FP)**: Quando o modelo retorna positivo e o pretendido era negativo.
- ***True Negatives* (TN)**: Quando o modelo retorna negativo e o pretendido era negativo.
- ***False Negatives* (FN)**: Quando o modelo retorna negativo e o pretendido era positivo.

Matriz de confusão

Utilizando os valores presentes nos termos acima indicados é criada a percentagem que cada termo tem na totalidade e criada uma imagem com o formato indicado na equação 2.1 [43].

$$\text{Matriz de Confusão} = \begin{cases} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{cases} \tag{2.1}$$

Exatidão

Métrica que permite medir a proporção de previsões corretas relativamente ao total de previsões efetuadas pelo modelo 2.2. Permite assim concluir a capacidade que o modelo tem de classificar corretamente uma amostra [43].

$$\text{Exatidão} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.2}$$

Precisão

A precisão permite medir a proporção de TP em relação a todos os exemplos classificados como positivos pelo modelo. Isto permite obter a qualidade que as previsões positivas do modelo têm 2.3 [43].

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{2.3}$$

Recall

Também conhecido como sensibilidade, permite avaliar a capacidade que o modelo possui para identificar corretamente TP. Para efetuar essa avaliação é utilizada a equação 2.4. Esta métrica é especialmente relevante quando o foco do modelo está na deteção correta de positivos [43].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.4}$$

Especificidade

Ao contrário da métrica mencionada anteriormente, a Especificidade mede a proporção de TN em relação a todos os negativos presentes no conjunto de dados 2.5. Isto permite avaliar a capacidade que o modelo possui para identificar corretamente TN, é especialmente útil quando o foco do modelo é a deteção correta de negativos [41].

$$\text{Especificidade} = \frac{TN}{TN + FP} \tag{2.5}$$

F1-Score

É uma métrica que combina o *Recall* com a Precisão. O resultado da equação 2.6 resulta num valor entre zero e um, onde um indica a melhor performance possível, o que significa que o modelo é bom a identificar TP e a evitar FN e FP [44].

$$\text{F1-score} = \frac{2 \cdot \text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}. \tag{2.6}$$

Curva ROC

A curva *Receiver Operating Characteristic* (ROC) visualizada na Figura 2.12 é construída utilizando as equações 2.7 e 2.8. Quanto maior for a área representada pela parte inferior da curva (AUC) melhor é o modelo treinado. Para o modelo ser considerado bom ou aceitável a curva tem de estar acima da diagonal representada a tracejado [45].

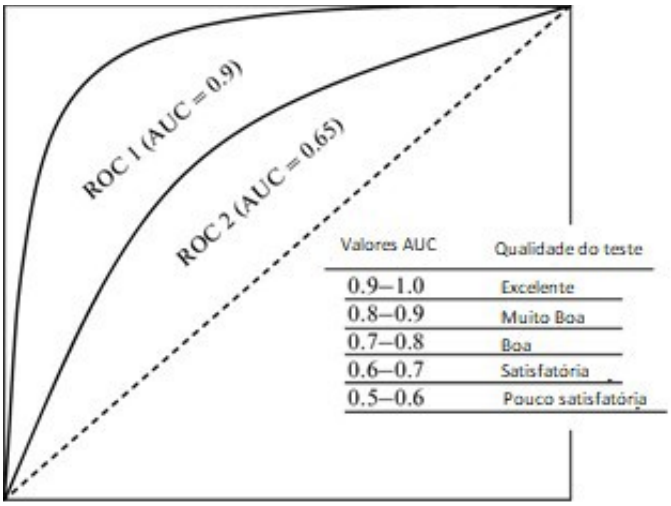


Figura 2.12: Exemplo de curvas ROC adaptado de[46]

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.7}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.8}$$

2.3 Reconhecimento de fala

Ser capaz de falar para um computador e o mesmo reconhecer o que está a ser dito providenciaria uma forma confortável e natural das pessoas utilizarem o mesmo. É aí que nasce o conceito de reconhecimento de fala que sucintamente consiste na capacidade de um programa transformar discurso humano em texto [47]. Outro fator importante a ter em conta quando se fala deste tema é não confundir reconhecimento de fala (*speech recognition*) com reconhecimento de voz (*speaker recognition*), uma vez que, o reconhecimento de voz é a habilidade de identificar quem é o utilizador que está a falar e não o que está a ser dito [48].

2.3.1 Evolução

O reconhecimento de fala já é desenvolvido há 72 anos, e cada vez têm sido feitos mais avanços para melhorar o nível de precisão dos *softwares* utilizados, de maneira a suportar mais idiomas e dialetos. Na figura abaixo 2.13 é possível visualizar os avanços da gigante tecnológica *Google* no que diz respeito à precisão do reconhecimento da fala ao longo dos anos (a vermelho), comparativamente com uma precisão de 95% (linha a azul)[49].

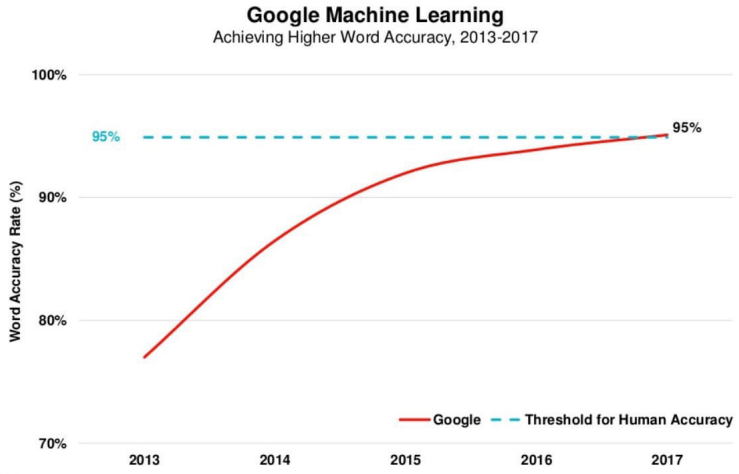


Figura 2.13: Evolução da precisão do reconhecimento de fala da *Google* ao longo dos anos [49].

O primeiro sistema de reconhecimento de fala foi lançado em 1952 pela *Bell Laboratories* e chamava-se *Audrey* 2.14, era um sistema que reconhecia números e não palavras. Apenas 10 anos depois surge outra evolução na área. Na tabela temporal 2.1 é possível visualizar os avanços efetuados ao longo das décadas. Estes avanços permitiram alcançar os sistemas de reconhecimento de fala que existem atualmente aplicados em diversas áreas.

Figura 2.14: Sistema Audrey [50].

Tabela 2.1 Linha cronológica das invenções na área do reconhecimento de fala [49].

1962	A IBM introduz o sistema Shoebox que conseguia reconhecer 16 palavras.
1970	Na década de 1970 são efetuados avanços no reconhecimento de fala, na maioria devido ao departamento de defesa dos Estados Unidos da América que criam o sistema Harpy, este sistema era capaz de reconhecer 1000 palavras (equivalente ao vocabulário de uma criança de 3 anos)
1980	Na década de 1980 é inventado o método estatístico *Hidden Markov Model*.
1990	Na década de 1990, devido à chegada dos computadores pessoais, surge o primeiro *software* de reconhecimento de fala comercial, chamava-se *Dragon Dictate*, e ainda hoje é comercializada uma versão bastante mais evoluída chamada *Dragon NaturallySpeaking*.
2000	Em 2001 o reconhecimento de fala atinge 80% de precisão. No resto da década não existem grandes evoluções, até que em 2009 a *Google* lança o *Google Voice Search*, uma aplicação que chegou a milhões de utilizadores, e permitiu a mesma recolher data de bilhões de pesquisas efetuadas pelos utilizadores.
2010	Em 2011 *Apple* lança a Siri, após a mesma segue-se a *Amazon* com a Alexa, entre outras. Atualmente as empresas competem para atingir a maior taxa de precisão.
Futuro	Com custos cada vez mais reduzidos, sistemas com mais capacidade de *hardware* , avanços nos campos de IA e um crescimento na recolha de dados, é bastante possível que o reconhecimento de fala se torne a próxima interface dominante.

2.3.2 Componentes

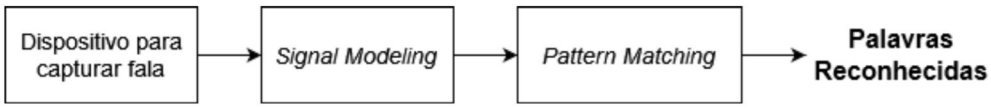


Figura 2.15: Componentes de um sistema de reconhecimento de fala comum [48].

Conforme demonstrado na Figura 2.15 o primeiro componente de um sistema de reconhecimento de fala é o dispositivo que capta a fala do utilizador, este dispositivo geralmente é um microfone. Um microfone capta som (sinal analógico) e converte-o num sinal elétrico. Isto acontece porque a pressão das ondas sonoras fazem um diafragma vibrar, como o diafragma está conectado a uma bobine e a mesma está no meio de um íman estas vibrações causam uma diferença de potencial e consequentemente uma força eletromotriz que percorre o circuito. Assim sendo, o sinal elétrico gerado coincide com as vibrações causadas pelo som em causa. O processo explicado

acima ocorre nos microfones dinâmicos, mas existem outros tipos de microfones nos quais o processo é diferente.

O segundo processo consiste no pré-processamento do sinal e análise do mesmo. Neste processo são aplicadas técnicas para remover barulho de fundo e ajustar o nível do sinal se necessário para o tornar mais claro e adequado para a análise. Esta fase é importante para garantir que o reconhecimento seja mais preciso e confiável. O último processo identificado como *pattern matching* consiste em identificar as características da fala, como padrões de frequência e duração das palavras. Existem duas abordagens principais:

- **Modelos de IA**: estes modelos utilizam redes neuronais como RNN e CNN para efetuar o reconhecimento de fala, a utilização dos mesmos pode ser comprovada em artigos como o *Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions* [51];
- **Modelos estatísticos**: estes modelos utilizam estatística para efetuar o reconhecimento de fala entre os modelos de estatística mais conhecido surge o *Hidden Markov Model* (HMM) [52].

2.3.3 Avaliação de performance

Atualmente, existem diversos sistemas de reconhecimento de fala e todos com uma taxa de precisão superior a 90%, isto dificulta a seleção do sistema para construir o projeto. Para escolher o mais indicado foram utilizados os seguintes métodos.

O primeiro método utilizado é o *Word Error Rate* (WER), é um método bastante simples que se encontra explicado na equação 2.9. É o método mais utilizado para avaliar os sistemas de reconhecimento de voz. Apesar de ser o mais utilizado, apresenta algumas falhas. A primeira é o facto de não facultar uma percentagem real uma vez que não possui limite superior. Não diz o quanto um sistema é bom, apenas permite comparar sistemas, ou seja, indica se um sistema é melhor que outro. Outra falha deste método de avaliação é o facto de não ser simétrico, uma vez que penaliza mais as palavras inseridas do que as palavras eliminadas [53].

$$WER = (I + D + S)/N \tag{2.9}$$

Onde:

- I - palavras inseridas
- D - palavras eliminadas
- S - palavras substituídas
- N - total de palavras

O segundo método é o *Word Accuracy* (WAcc) que vem resolver o problema de não existir limite superior no método referido anteriormente, é feito apenas uma conta de subtração conforme visível na equação 2.10.

$$Waac = 100 - WER \tag{2.10}$$

Outro dos métodos utilizado para avaliar o sistema de reconhecimento de fala é a distância de *Levenshtein*, este método mede a diferença entre duas strings, esta medida é o número total de letras que é necessário alterar, apagar e adicionar para as duas strings ficarem iguais [54]. Para perceber melhor como este processo acontece e entender o algoritmo do mesmo basta analisar as tabelas abaixo. Primeiro é inicializada a tabela 2.2 (Cima) com 2 palavras diferentes, depois preenche-se a tabela, por exemplo, quando a letra C e J se encontram é colocado o número 1 porque seria necessário fazer uma alteração para as letras coincidirem, já na linha a seguir quando a letra "U"da palavra "Custas"e a letra "U"da palavra "Justa"se encontram é na mesma mantido o 1 porque não foi preciso efetuar mais nenhuma alteração. Após preencher a tabela total deve-se obter uma tabela igual a 2.2 (Baixo) e o número no canto inferior direito será a distância de *Levenshtein*.

Tabela 2.2: Primeiro passo do algoritmo da distância de *Levenshtein*.

	C	**U**	**S**	**T**	**A**	**S**
	0	1	2	3	4	5
J	**1**	2				
U	2	**1**				
S	3	2				
T	4	3				
A	5	4				

	C	**U**	**S**	**T**	**A**	**S**
	0	1	2	3	4	5
J	**1**	2	3	4	5	6
U	2	**1**	2	3	4	5
S	3	2	**1**	2	3	4
T	4	3	2	**1**	0	1
A	5	4	3	2	**1**	**2**

Os dois últimos métodos são, o tempo de resposta e as características próprias, enquanto as características próprias podem variar conforme a preferência de cada utilizador e/ou requisitos de projeto, o tempo de resposta quanto menor melhor. Mas este valor pode depender de fatores externos como a velocidade de ligação a internet. Na Tabela 2.3 são apresentados diversos sistemas e as vantagens e desvantagens de cada um.

Tabela 2.3: Comparação de diferentes sistemas de fala.

	Open-Source	Offline	Técnica	Dialetos	Scale
Google Cloud Speech	Não	Não	N/A	120	N/A
Vosk API	Sim	Sim	Deep Learning	20+	Sim
CMU Sphinx	Sim	Sim	Estatística	20+	Sim
Microsoft Azure Speech	Não	Não	N/A	60+	N/A
Wit.ai	Não	Não	N/A	20+	Não

2.4 Reconhecimento Facial

O desenvolvimento da tecnologia de reconhecimento facial automático data dos anos 60, quando o primeiro sistema semi-automatizado foi criado utilizando um método de medição de características faciais em fotografias e comparando-as com um ponto de referência comum. O campo foi fundado por Woody Bledsoe, Helen Chan Wolf e Charles Bisson, que utilizaram um computador para reconhecer rostos humanos em 1964 – 1965. Nos anos 70, 21 características únicas como a espessura dos lábios e a cor do cabelo foram utilizadas, mas as medições foram feitas manualmente. Em 1988, a aplicação do método de análise de componentes principais melhorou a precisão do reconhecimento. Têm sido desenvolvidos ao logo dos anos diversos algoritmos de reconhecimento facial, apesar de ainda não existir nenhum sistema que reconheça faces com a mesma eficiência que o ser humano reconhece, ou seja, a capacidade de reconhecer faces em qualquer ambiente, ângulo ou expressão, existem já sistemas, que em ambientes controlados, superam o desempenho humano, sendo até mesmo capazes de diferenciar gémeos [55]. A vantagem do reconhecimento facial sobre outros métodos biométricos é que se trata de um reconhecimento não intrusivo, o utilizador não necessita de interagir diretamente com o sistema [56]. Hoje em dia, esta tecnologia é amplamente utilizada e ativamente investigada, com esforços contínuos para enfrentar os desafios do mundo real.

2.4.1 Aplicações do Reconhecimento Facial

O reconhecimento facial tem sido cada vez mais aplicado em diversas áreas [57] principalmente na área da segurança. Um exemplo disso é demonstrado no *paper Facial recognition application for border control* [58]. Neste caso o reconhecimento

facial é utilizado para validar se o portador dos documentos corresponde à pessoa indicada nos documentos, isto permite um maior controlo nas fronteiras. Para efetuar a avaliação deste modelo foi utilizada a base de dados LFW.

Não só nas questões de segurança o reconhecimento facial é empregue, outra área onde tem crescido a sua utilização é na saúde [57].Uma das aplicações é a *Face2Gene* que permite detetar doenças raras em crianças [59].

2.4.2 Facial Recognition Datasets

- **Flickr-Faces-HQ Dataset:** *Dataset* composto por 70000 faces humanas, que se encontram categorizadas segundo a idade, etnia e imagem de fundo. Foi criado retirando imagens do *website Flickr*. E foram seguidamente utilizadas duas ferramentas, dlib e Amazon Mechanical Turk, a primeira responsável por recortar as imagens para que todas tivessem uma resolução 1024×1024, e a segunda responsável por remover estátuas e pinturas que tivessem presentes no fundo. Este *dataset* não pode ser utilizado em tecnologias de reconhecimento facial uma vez que muitas pessoas presentes no mesmo não deram autorização para a utilização das suas imagens [60].

- **Tufts Face Dataset:** *Dataset* que contém 7 modalidades de imagem: visível, infravermelho próximo, térmica, esboço computorizado, LYTRO, vídeo gravado e imagens 3D. Contem mais de 10000 imagens, nas quais estão incluídas 74 mulheres e 38 homens de mais de 15 países com uma faixa etária entre 4 e 70 anos. Este *dataset* costuma ser utilizado para comparar algoritmos de reconhecimento facial.

- **Labeled Faces in the Wild (LFW):** É um *dataset* concebido para estudar o problema do reconhecimento facial sem restrições. Contém mais de 13.000 imagens de rostos recolhidos na Web. Cada rosto foi etiquetado com o nome da pessoa fotografada. 1680 das pessoas fotografadas têm duas ou mais fotos distintas no conjunto de dados. Existem agora quatro conjuntos diferentes de imagens LFW (incluindo o original). Entre estes, o conjunto LFW-a e as imagens *deep funneled* produzem resultados superiores para a maioria dos algoritmos de verificação facial [61].

- **UTKFace Dataset:** Este *dataset* é composto por mais de 20 000 imagens de faces compreendidas entre 0 e os 116 anos, cada uma das imagens possui anotações de idade, género e etnia. As imagens apresentam grandes variações em termos de pose, expressão facial, iluminação, oclusão e resolução [62].

2.4.3 Modelos de Reconhecimento de Imagem e Facial

Neste subcapítulo são abordados alguns modelos construidos anteriormente e já revistos cientificamente. O primeiro caso a ser analisado é o *Siamese Neural Networks for One-shot Image Recognition* [39] publicado em 2015.

O dataset utilizado nesta publicação foi Omniglot Dataset que consiste em 50 alfabetos diferentes. Cada alfabeto possui entre catorze e cinquenta e cinco caracteres desenhados por vinte pessoas diferentes (cada carácter foi armazenado numa imagem de 105×105 pixeis). Este dataset foi posteriormente dividido em dois: 40 alfabetos são utilizados para treino e os restantes para avaliação do modelo.

Tabela 2.4: Camadas da Rede Neuronal utilizada no paper *Siamese Neural Networks for One-shot Image Recognition.*

Layer
Conv2D (Conv1)
MaxPooling
Conv2D (Conv2)
MaxPooling
Conv2D (Conv3)
MaxPooling
Conv2D (Conv4)
Flatten
Dense (Dense1)

A rede siamesa construida consiste em duas CNN compostas pelas camadas visíveis na Tabela 2.4. Após execução das mesmas são obtidos dois vetores de características, posteriormente comparados aplicando a norma L1 e medindo a distância (conhecida por distância de Manhatan) 2.11. Ou seja, se o resultado forem dois vetores $\mathbf{V1} = [1, 2, 3]$ e $\mathbf{V2} = [1, -3, -3]$, a distância entre eles é $|1-1|+|2-(-3)|+|3-(-3)|$ o que resulta em 11 [63].

$$\text{Distância L1} = \sum_{i=1}^{n} |v_{1,i} - v_{2,i}| \tag{2.11}$$

Outros componentes utilizados neste paper são:

- Regularização L2: técnica utilizada para evitar *overffiting*, também pode ser chamada regularização de Ridge;
- Otimizador *Stochastic Gradient Descent*: Um algoritmo utilizado para ajustar os parâmetros de um modelo para minimizar a sua função de perda. Costuma ser utilizado em problemas de classificação binária, e se o treino exigir muitos parâmetros ou se a rede neuronal for muito grande, uma vez que é um otimizador rápido e evita saturação de memória GPU [64, 65].

O segundo modelo abordado é fornecido na documentação oficial da biblioteca Keras e intitula-se por *Image similarity estimation using a Siamese Network with a triplet loss* [66]. Neste exemplo o dataset utilizado foi o *Totally Looks Like*, o mesmo é utilizado para construir *triplets* que serão os inputs na rede siamesa. Um *triplet* consiste em três imagens: uma positiva, uma negativa e uma âncora que é uma imagem similar a positiva.

A rede siamesa receberá os dados de entrada e produzirá a distância entre a âncora e a representação incorporada positiva, assim como a distância entre a âncora e a representação incorporada negativa. Para calcular esta distância (conhecida como distancia *euclidean* 2.12) é utilizada a norma L2. Ou seja, se o resultado forem dois vetores $\mathbf{V1} = [1, 2, 3]$ e $\mathbf{V2} = [1, -3, -3]$, a distância entre eles é $\sqrt{(1-1)^2 + (2-(-3))^2 + (3-(-3))^2}$ o que resulta em ≈ 7.81 [67].

$$\text{Distância L2} = \sqrt{\sum_{i=1}^{n}(V1_i - V2_i)^2} \tag{2.12}$$

Outros componentes utilizados neste paper são:

- Triplet Loss: Esta função permite treinar modelos de similaridade, assim sendo costuma ser empregue para tarefas de reconhecimento facial. O principal objetivo desta função é aprender a aproximar pontos de dados semelhantes e afastar pontos de dados dissimilares [68].

- Otimizador Adam: Costuma ser o otimizador usado para pequenos conjuntos de dados e em redes de *deep learning*. É uma escolha mais segura que o otimizador mencionado anteriormente, uma vez que não é necessário despender tanto tempo para ajustar os parâmetros do mesmo [69, 64].

O terceiro e último paper estudado foi Face Recognition Using Popular Deep Net Architectures: A Brief Comparative Study que apresenta diversos modelos pré-treinados e as suas precisões quando treinados no *dataset* LFW [70]. Os modelos pré-treinado são bastante interessantes, pois são modelos DPL treinados em grandes conjuntos de dados para realizar uma tarefa específica e podem ser usados como estão definidos ou personalizados para atender os requisitos específicos pretendidos [71]. Vários estudos demonstram que modelos de linguagem grandes pré-treinados têm um desempenho tão bom quanto modelos especificamente treinados para tarefas personalizadas [72].

O propósito do estudo referido foi determinar qual dos métodos CNN atualmente testados é o mais eficaz no que diz respeito à precisão e outras métricas [70]. Entre os CNN apresentados encontram-se:

- O modelo XCeption que consiste em setenta e uma camadas. A arquitetura Xception é um conjunto de camadas de convolução separáveis com conexões

residuais. Isto torna a arquitetura muito fácil de definir e modificar, sendo apenas necessárias 30 a 40 linhas de código usando uma biblioteca de alto nível [73].

- O modelo VGG16 é uma CNN com dezasseis camadas de profundidade. Este modelo foi pré-treinado recorrendo à base de dados ImageNet com mais de um milhão de imagens. A rede pré-treinada pode classificar imagens em 1.000 categorias de objetos[74].

2.5 Dispensador de Comprimidos

Como referido anteriormente, a idade média da população tem aumentado, criando assim uma grande comunidade de idosos nas sociedades modernas. Esse fenómeno é resultado de diversos aspetos, como a melhoria geral do estilo de vida das pessoas, a adoção de sistemas de assistência digital e o progresso crescente nos campos da ciência, pesquisa, medicina e tecnologia. [75, 76].

2.5.1 Casos de estudo

A Universidade de Salento [75] desenvolveu um dispensador de comprimidos com base na arquitetura demonstrada na Figura 2.16. Este sistema era constituído por um Raspberry Pi Model B, responsável por estabelecer comunicação com a aplicação e com o Arduino Nano. Este último responsável por dispensar o comprimido utilizando um servomotor e um motor passo a passo. Este sistema apresentava diversos compartimentos, cada um deles para um tipo diferente de comprimido.A hora de toma de cada um seria adicionada utilizando a aplicação de telemóvel. Cada vez que um comprimido era dispensado, um sinal sonoro era reproduzido e um led piscava. Este sistema não possuía nenhum sistema de segurança que avaliasse se o comprimido foi tomado ou não. No entanto, é mencionada a possível utilização de um sensor de contacto para essa deteção. É feito ainda um estudo de custos que estima que naquela data (2020) produzir este sistema custaria cerca de 200 € [75].

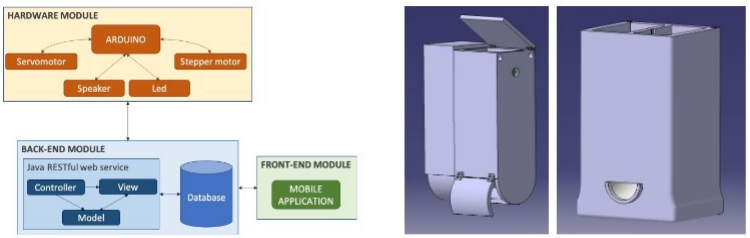


Figura 2.16: Arquitetura e Modelo 3D do sistema.

Nurmiza Binti Othman e Ong Pek Ek [76] propuseram um dispensador de comprimidos inteligente com alarme utilizando as notificações do telemóvel. Para construir o sistema visível na Figura 2.17 recorreram a um Arduino Mega 2560 que comunicava com o utilizador através de botões de pressão. Este Arduino controlava todo o sistema, desde o LCD ao sistema de dispensar comprimidos, que consistia num motor de vibração que fazia comprimidos cair e depois um sensor IR responsável por avaliar quantos comprimidos tinham caído. Para introduzir a componente dos alarmes via notificação é utilizado um Raspberry Pi B+. Neste caso não foi realizada uma orçamentação do custo de produção.

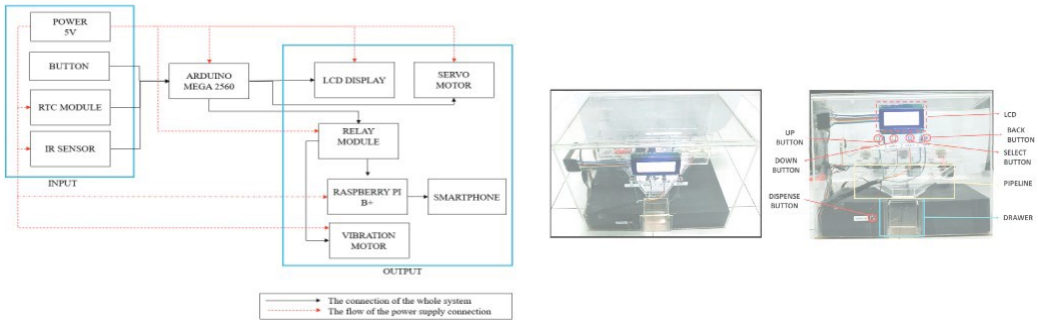


Figura 2.17: Arquitetura e fotos reais do sistema

Outro modelo estudado foi o *Programable IoT Pills Dispenser*. Conforme visível na Figura 2.18 para construir este modelo foi criada uma *Printed Circuit Board* (PCB) com um microprocessador capaz de controlar todos os sensores e periféricos, a mesma fornecia ainda conexão à Internet. Este sistema era constituído por 16 cilindros onde seriam colocadas dosagens de comprimidos, sendo possível programar qual o cilindro que devia ser utilizado para depositar a medicação e a que horas, existindo 9 intervalos de tempo diferentes para cada dia. Para dispensar os comprimidos foi utilizado um solenoide de empurrar e um servomotor, para segurança existia também um micro interruptor que detetava a toma dos comprimidos. Para projetos futuros pretendiam, ainda, desenvolver uma *network* e notificar, por exemplo, hospitais da toma de medicação [77].

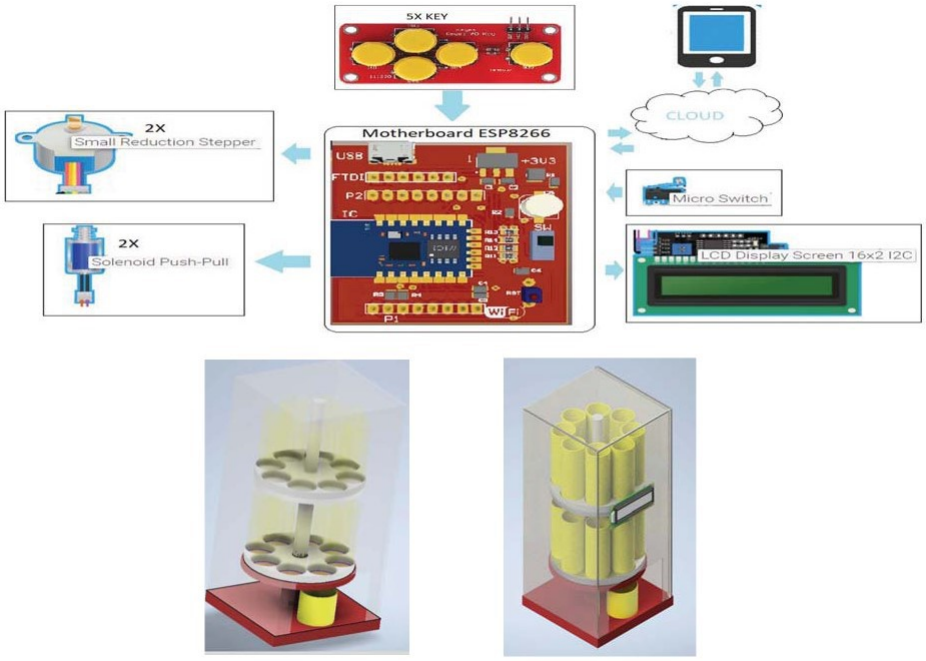


Figura 2.18: Arquitetura e Modelo 3D do sistema

2.5.2 Soluções Comerciais

Atualmente já existem diversas soluções comerciais para compará-las e analisá-las, a Tabela A.2 presente nos Anexos foi criada utilizando as fichas de especificação técnica disponíveis e *reviews* de utilizadores.

Capítulo 3

Protótipo

Neste capítulo vão ser discutidas as arquiteturas de *hardware* e *software* do dispensador, para os vários componentes do mesmo. O projeto contempla três componentes e consequentemente três fases de desenvolvimento visíveis na Figura 3.1. O primeiro módulo será o de Interação, este módulo permite ao utilizador comunicar com o sistema utilizando comandos de voz pré-definidos. O segundo módulo é o de Segurança e Reconhecimento, a sua funcionalidade é garantir e validar a identificação do utilizador, para isso é utilizado um modelo de IA. O último módulo é o que compreende as funcionalidades genéricas de um sistema de medicação e auxilia os restantes módulos, possuindo funcionalidades como base de dados para armazenar informação necessária até um sistema de leds para controlo do que está a acontecer em determinados momentos da execução de tarefas. Esta subdivisão permite também efetuar o desenvolvimento e possíveis melhorias futuras de uma forma mais facilitada.

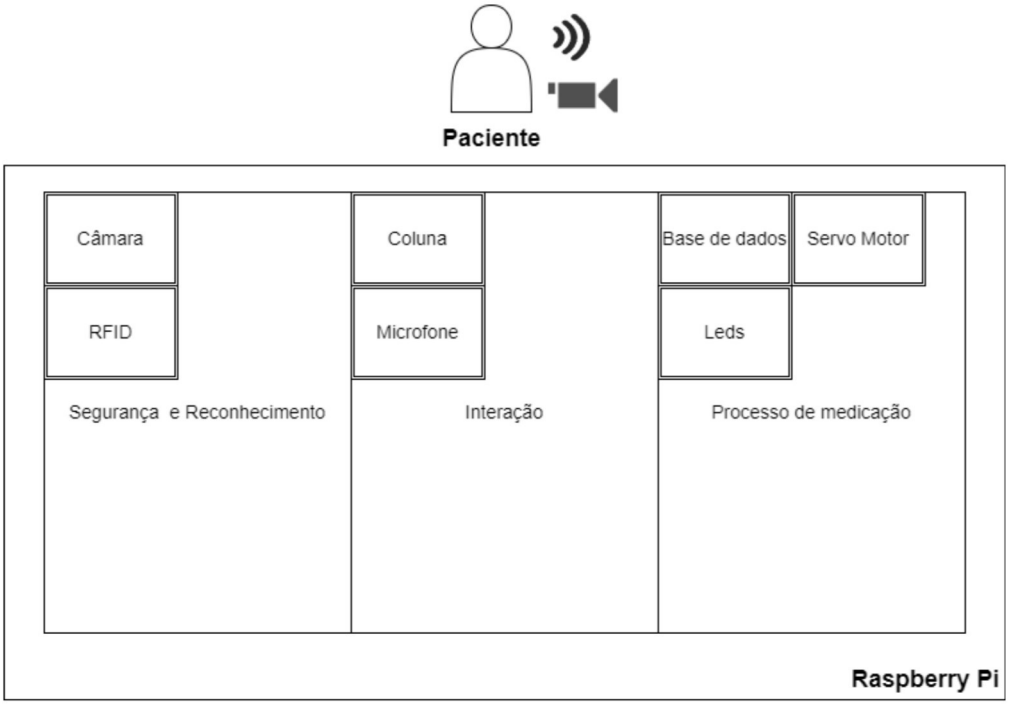


Figura 3.1: Arquitetura geral

3.1 Análise de Requisitos e UML

Após a reunião de toda informação anteriormente apresentada prossegui-se com a fase de desenvolvimento do protótipo. De encontro aos objetivos apresentados e com os cenários de aplicação idealizados em mente, pretendeu-se desenvolver um dispensador com as seguintes funcionalidades:

1. Validar que a pessoa que requer a medicação é a pessoa a que a mesma se destina.
2. Interagir com utilizadores de forma simples. Deverá ser desenvolvido um método de interação que não exija formação ou treino para a mesma;
3. Funcionar sem acesso à *Internet*, sendo, deste modo, de mais fácil distribuição;
4. Possuir funcionalidades extra que permitam ao utilizador recorrer ao sistema no seu dia a dia.
5. Dispensar comprimidos consoante o planeamento.

E com as seguintes funcionalidades extra:

1. Apresentar um modo de administrador que possibilite inserir um planeamento de medicação ou alterar o presente.

2. Alertar o utilizador para a tomar a medicação.

3. Apresentar a funcionalidade de interromper a recolha de informação caso o utilizador o deseje. Um mecanismo físico que bloqueie sistemas como câmeras e microfone.

De forma a armazenar informação relativa ao utilizador (paciente) como o número de identificação do seu cartão, a medicação que o mesmo necessita de tomar e a localização das fotos do utilizador que permitem validar o mesmo, o seguinte modelo de dados 3.2 foi desenvolvido.

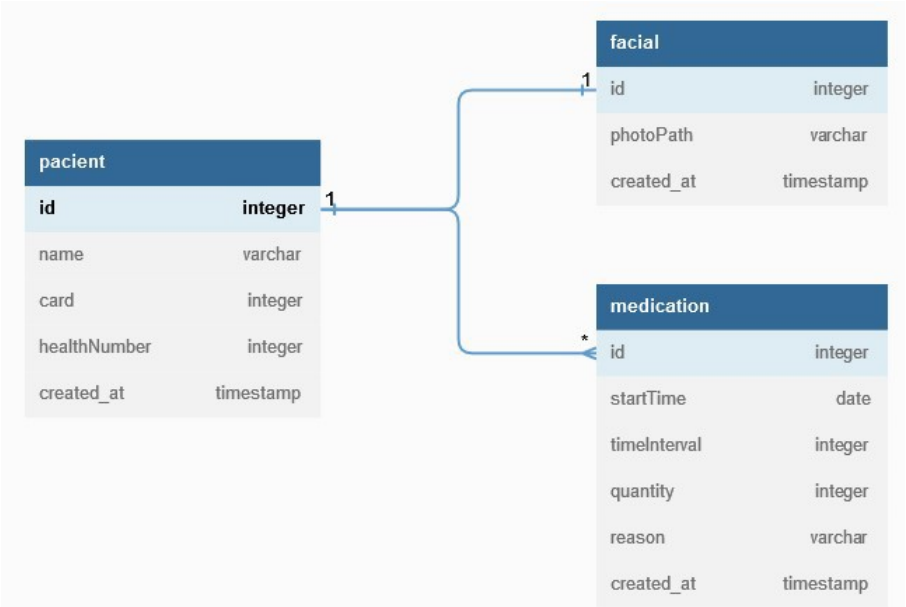


Figura 3.2: Modelo de dados

Na Figura 3.3 é visível o diagrama *Unified Modeling Language* (UML) dos requisitos pretendidos para o projeto. Existem 2 atores, o paciente e o médico. O médico é responsável por alterar prescrições de medicação e aceder ao registo de toma, funciona como uma espécie de administrador da funcionalidade principal do projeto. Já o paciente possui diversas ações uma vez que será o utilizador do sistema. A funcionalidade de alterar a prescrição, apesar de comum a ambos, requer autorização do médico para ser efetuada.

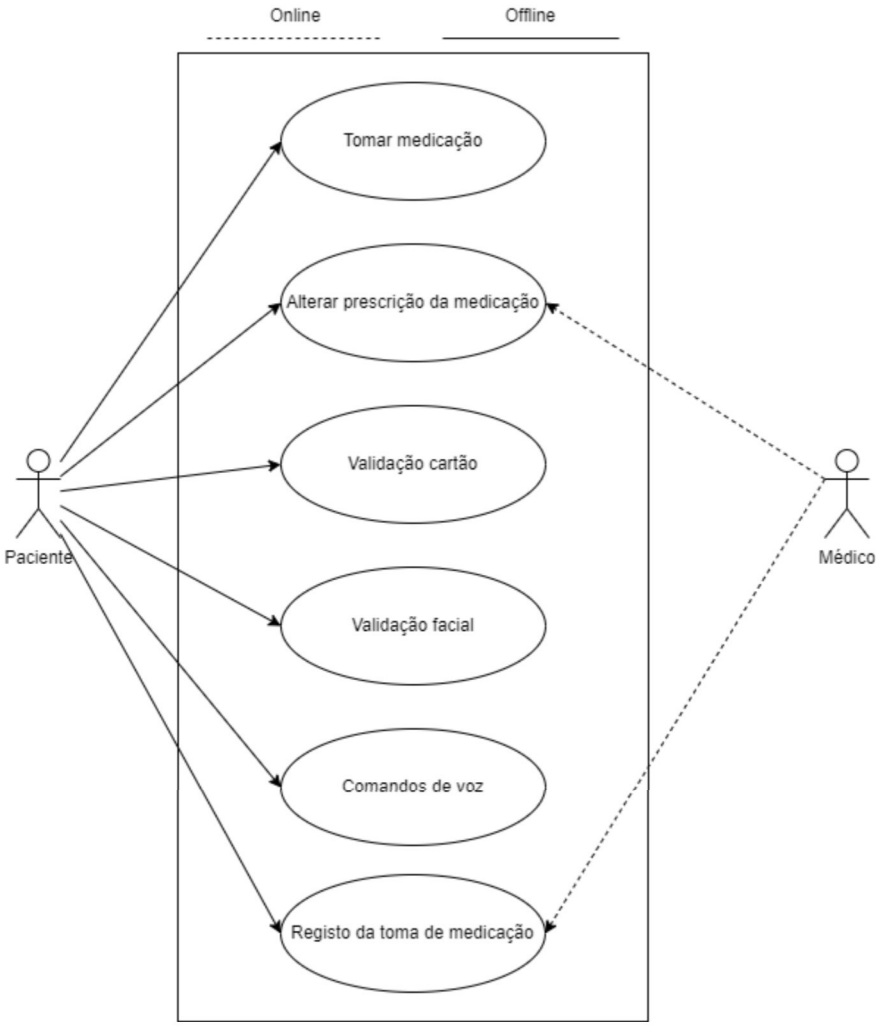


Figura 3.3: Diagrama UML dos requisitos apresentados para o projeto.

3.2 Módulo de Interação

A interface para interação com o utilizador foi desenvolvida em computador recorrendo à linguagem python, uma vez que o mesmo já incorpora componentes como microfone, colunas e câmara. Este módulo é posteriormente utilizado num RaspberryPi.

Esta interface deverá permitir ao utilizador interagir com o sistema de forma facilitada, uma vez que o mesmo poderá possuir dificuldades por se tratar de um idoso. Também deverá possuir um conjunto de ações extra com o intuito de entreter e facilitar certas ações recorrentes como saber as horas e a data.

3.2.1 Software

A fim de desenvolver o *software* para este módulo foi escolhida a linguagem de programação Python devido aos seguintes motivos [78]:

- Linguagem bastante utilizada em projetos de IA;
- Ecossistema com diversas *frameworks* e livrarias,
- Uma linguagem interpretada em vez de compilada, o que permite executar o código noutro sistema diferente do de desenvolvimento desde que o mesmo possua Python;
- Uma grande e ativa comunidade *online*.

Para conseguir interagir com o utilizador foi escolhido o método mais fácil e que exige menos conhecimento e interação física por parte deste, a fala. Para conseguir que o módulo possuísse estas funcionalidades as seguintes livrarias foram utilizadas:

- pyttsx3 - Esta biblioteca permite a conversão de texto para som, o que possibilita que o sistema interaja com o utilizador. Funciona *offline* e é compatível com Python.
- vosk - Como mencionado anteriormente, em 2.3 ,possibilita o reconhecimento de fala em mais de 20 idiomas. Esta biblioteca utiliza modelos *deep learning* e existem já modelos treinados que possuem pequena dimensão (50Mb). Funciona *offline*, é compatível com Python e corre em dispositivos *lightweight* como o RaspberryPi.

Os seguintes comandos serão disponibilizados ao utilizador para interagir com o sistema (Sarah):

- Hey Sarah : Ativa a Sarah (assistente pessoal) desenvolvida no módulo python.
- The time: A assistente pessoal, pelas colunas, transmitirá a hora atual.
- Calculate: A assistente pessoal calcula equações simples de adição, subtração, multiplicação e divisão.
- Sleep Sarah e Goodbye Sarah: Desativa a assistente pessoal.

Descrição

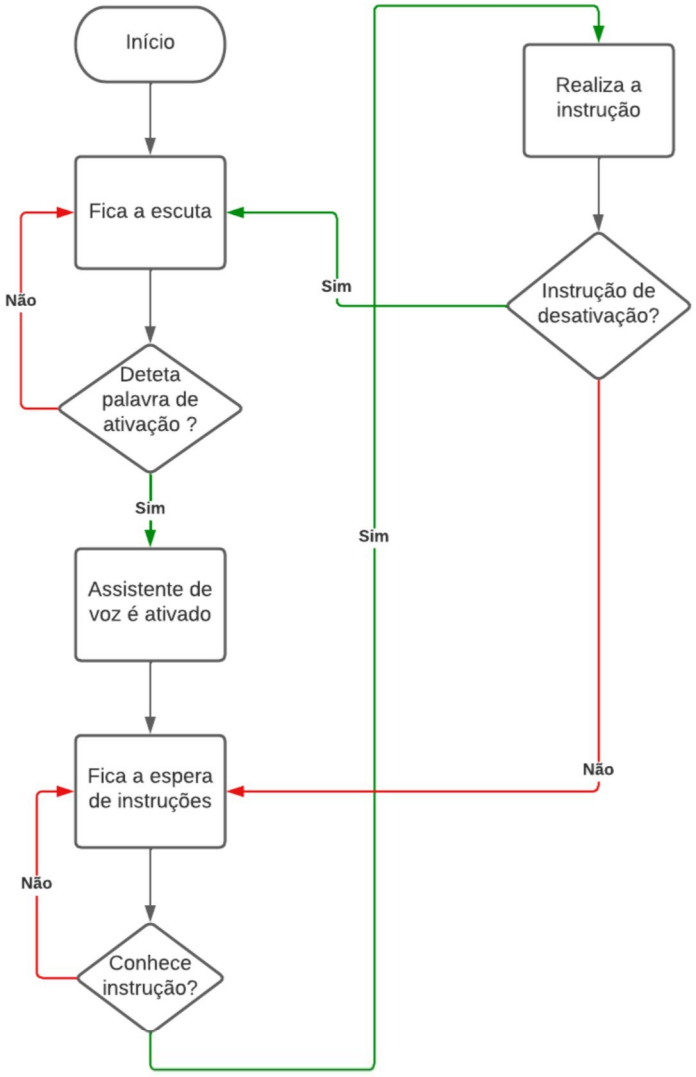


Figura 3.4: Fluxograma geral do módulo de interação e entretenimento

3.2.2 Hardware

Para a realização deste módulo são necessários componentes extra para adicionar à placa de desenvolvimento, uma vez que a placa escolhida não possui microfone nem colunas. Para a escolha do microfone foi tido em conta o preço, a disponibilidade e as características do mesmo. Uma das características requeridas era o facto de funcionar por USB uma vez que o RaspberryPi possui 4 portas livres. Assim sendo o microfone escolhido foi o *Estiq Super Mini USB 2.0* disponível por aproximadamente seis euros na Amazon com as seguintes características:

- **Drivers:** Não é necessária a instalação de drivers
- **Sensibilidade:** -67 dBV/pBar.-47 dBV/Pascal+/-4dB
- **Frequência:** 100-16kHz

A escolha da coluna não é tida em conta nesta dissertação.

3.2.3 Implementação

Para elaborar o módulo de interação baseado nos diagramas visíveis no Apêndice desta dissertação (A.2 e A.3), e com as tecnologias mencionadas acima, utilizou-se para versão de controlo, o Git, aliado à plataforma GitHub. Este método permitiu ter acesso a todas as mudanças efetuadas tanto no RaspberryPi como no computador e controlar o processo de desenvolvimento de forma mais coerente. Após a criação do repositório necessário e da preparação do mesmo, o primeiro passo foi elaborar uma arquitetura de ficheiros para criar uma classe (voice.py) que pudesse ser chamada por outros códigos. Como se trata de um assistente pessoal, faz sentido também ser um ato externo e simplificado alterar a voz e nome do mesmo. Para isso basta alterar a chamada do construtor a classe *Assistant*(Listagem 3.1).

```
1     engine = pyttsx3.init()
2     engine.setProperty('rate', 150) # setting up new voice rate
3     engine.setProperty('volume', 0.7) # setting up volume level
          between 0 and 1
4     engine.setProperty('voice', 'default') #changing index,
          changes voices.
5     yuda = voice.Assistant("sarah", False, tts_engine=engine,
          tts_configs=None)
```

Listagem 3.1: Criação do objeto text-to-speech

A seguir foi pensada uma forma de adicionar mais *features* à classe voice, sem ser necessário alterar o código/processo todo. Para as adicionar foi, então, criada uma pasta com o nome *features* 3.5, onde está contido o código em ficheiros individuais das ações possíveis de executar.

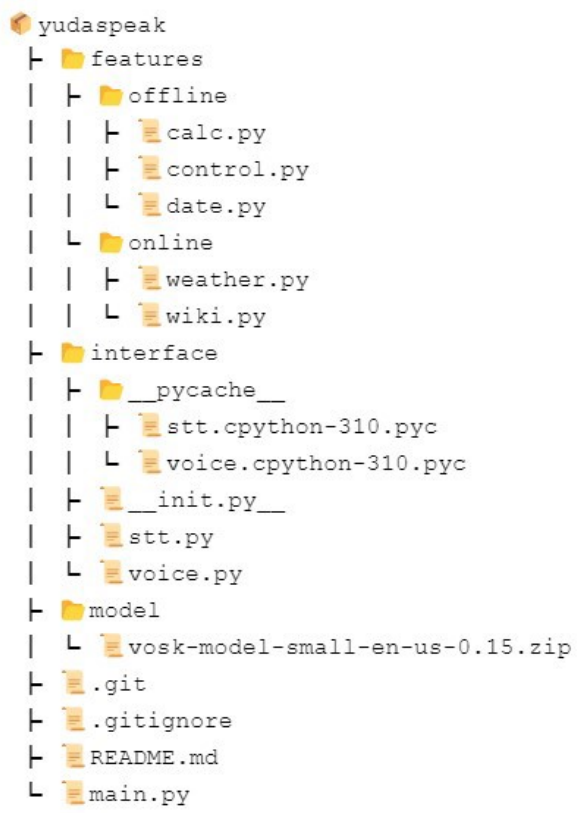


Figura 3.5: Organização do Módulo de Entretenimento e Interação

Como o módulo desenvolvido funcionará apenas *offline* mas numa implementação futura poderá aceder à *internet*, foi tido em conta a natureza das *features* uma vez que algumas requerem acesso à mesma. Assim sendo, foi construido um mecanismo para identificar se o módulo tem ou não acesso à *internet* e disponibilizar assim apenas as *features* que consegue efetuar. No desenvolvimento da assistente de voz (Listagem 3.1) a variável booleana representa o acesso a redes externas.

Resultados

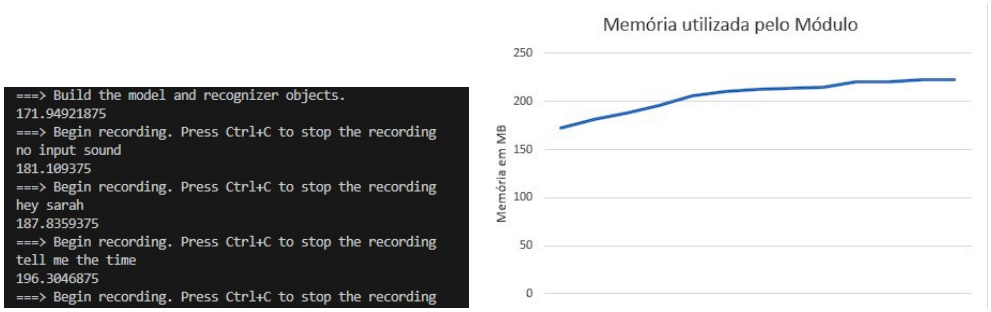


Figura 3.6: Execução do módulo e utilização da memória.

Todos os objetivos pretendidos com a elaboração deste módulo foram concluídos com sucesso. Na Figura 3.6 é visível a execução do módulo e o consumo de memória do mesmo. É ainda visível na Tabela 3.1 os testes efetuados ao modelo. Para efetuar estes testes foram gravadas três faixas de áudio em formato .wav utilizando uma pessoa que tem como língua materna inglês de forma a não influenciar os resultados. Estas gravações foram passadas como *input* para o modelo de forma a simular um utilizador a falar em tempo real. Analisando os resultados obtidos é possível concluir que o modelo satisfaz os requisitos necessários, ou seja, tem um nível de compreensão e de falha dentro do suposto.

Texto .wav	Texto Transcrito	WER	WAcc(%)
please sarah tell me the date	Sem diferenças	0	100
please sarah calculate two plus two	Sem diferenças	0	100
within the scope of this mobility project, students had the opportunity to experience new teaching methodologies brought by teachers from other cultures good practices were shared among professionals in laboratories and various services new areas of academic and scientific cooperation were explored, and new projects were designed enriching the school academically and culturally.	*methodologies* substituida por *methodology is*	≈ 0.0263	≈ 99.97

Tabela 3.1: Performance do Modelo de Fala

3.3 Módulo de Segurança e Reconhecimento

O desenvolvimento deste módulo foi o que requereu mais tempo e esforço, uma vez que se trata do módulo que compreende praticamente todo o Estado de Arte e pesquisa efetuada anteriormente. O objetivo deste é elaborar mecanismos que permitem fornecer reconhecimento e segurança com o intuito de confirmar que o paciente que requer a medicação é o suposto. Para isso foi desenvolvido um modelo de deep learning que pode ser utilizado em reconhecimento facial e posteriormente um mecanismo de rfid para adicionar segurança adicional.

Esta interface tal como a anteriormente mencionada foi desenvolvida em computador, de referir que o computador possui uma gráfica Nvidia RTX 3050 versão *laptop* e um processador AMD Ryzen 5 5600H. Esta interface, como referido, deve

permitir que o sistema valide que o utilizador que toma a medicação é o paciente ao qual a mesma se destina.

3.3.1 Software

Para desenvolver o *software* para este módulo recorreu-se à linguagem de programação python pelos motivos mencionados anteriormente.

De forma a validar o paciente foi utilizado um algoritmo que permite efetuar a leitura de cartões e outro algoritmo com um modelo de DPL que permite a partir da utilização de uma câmara efetuar reconhecimento facial.

Para desenvolver estes dois algoritmos foram utilizadas algumas livrarias, de referir que o modelo foi treinado para aprofundar não só o conhecimento sobre DPL como também efetuar correções necessárias. As livrarias utilizadas foram:

- tensorflow - Para além de ser uma livraria é considerada uma plataforma open-source para ML por fornecer ferramentas, livrarias e recursos que facilitam o desenvolvimento de modelos. Existem outras livrarias que podem ser utilizadas, uma comparação entre elas pode ser visível na Tabela A.1.
- mfrc522 - É uma livraria que implementa funções que permitem ler e escrever para cartões que utilizam diferentes tipos de *Radio Frequency Identification*, quando utilizado um leitor RC522 e SPI.

Descrição

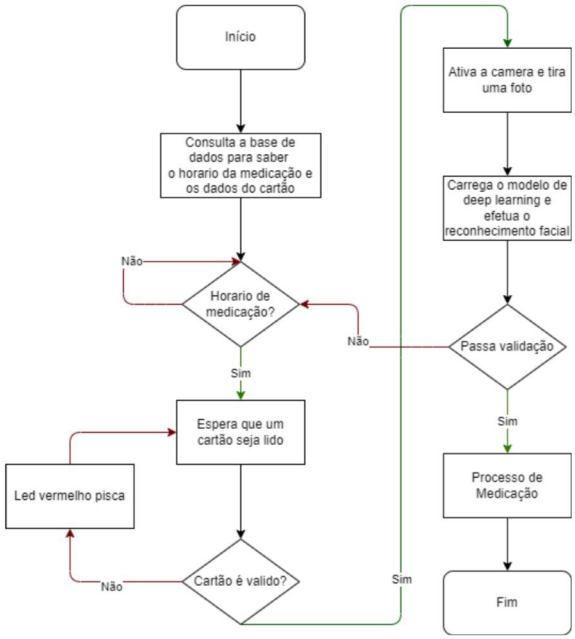


Figura 3.7: Fluxograma geral do módulo de segurança e reconhecimento

3.3.2 Hardware

Para a realização deste módulo será necessário adicionar um componente que permita a placa de desenvolvimento efetuar a captura de imagem. A melhor opção será adicionar a câmara oficial da Raspberry a *Pi Camera Module 2* que possui as seguintes características:

- **Resolução:** 8 Megapixels
- **Sensor:** Sony IMX219
- **Focus:** Ajustável

3.3.3 Implementação Reconhecimento Facial

Para desenvolver um modelo de reconhecimento facial que permitisse ter uma precisão elevada, superior a 80%, foram efetuados diversos treinos com base na pesquisa apresentada no Estado de Arte 2.4.3. Foram aplicadas várias alterações nos modelos para conseguir atingir uma precisão considerável, foram, ainda, efetuadas avaliações dos modelos após serem treinados aplicando os métodos explicados em 2.2.2.

O primeiro passo consistiu na construção do *dataset*. Para todos os treinos efetuados o *dataset* utilizado foi o LFW mencionado no capitulo 2.2.2. O primeiro passo efetuado no *dataset* foi recolher as faces detetadas de todas as imagens no formato desejado, utilizando o algoritmo haar cascade. De seguida o *dataset* é subdividido em três partes:

- *Dataset* treino: Consiste em 80% da totalidade dos dados sendo utilizado durante o treino dos modelos.
- *Dataset* validação: Consiste em 10% da totalidade dos dados, sendo utilizado durante o treino dos modelos para avaliar a precisão do mesmo.
- *Dataset* teste: Consiste em 10% da totalidade dos dados sendo utilizado após o treino para avaliar o modelo.

Após esta divisão é efetuada a criação de *triplets* por se tratar de uma rede siamesa é necessário um modelo de dados que apresenta um objeto que consiste em duas imagens com a mesma face (Âncora e Positivo) e outra imagem com uma face diferente (Negativo), um exemplo desta organização é visível na Figura 3.8.

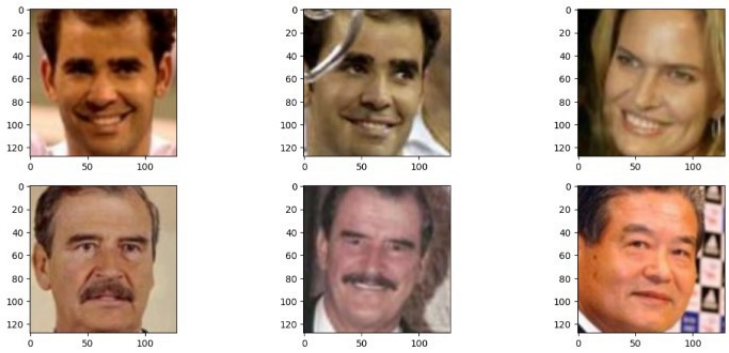


Figura 3.8: Exemplo de dois *triplets*.

O primeiro modelo apresentado na Tabela A.3 consiste num conjunto de camadas de convolução que permitem a extração de características significativas das faces interpoladas com camadas de *max pooling* que ajudam a reduzir a dimensão dos dados mantendo as características mais importantes dos mesmos. No final é ainda utilizada uma camada *flatten* para transformar as representações multidimensionais num vetor unidimensional e outra camada *Dense* utilizada para conectar todos os neurónios da camada anterior.

Com a análise nas Figuras 3.9 é possível concluir que o modelo convergiu por volta da época 80, na fase de treino foi utilizado um *batch size* de 16 e demorou 260 minutos para concluir o treino de 200 épocas. Na Tabela 3.2 é ainda visível as métricas obtidas para este primeiro modelo.

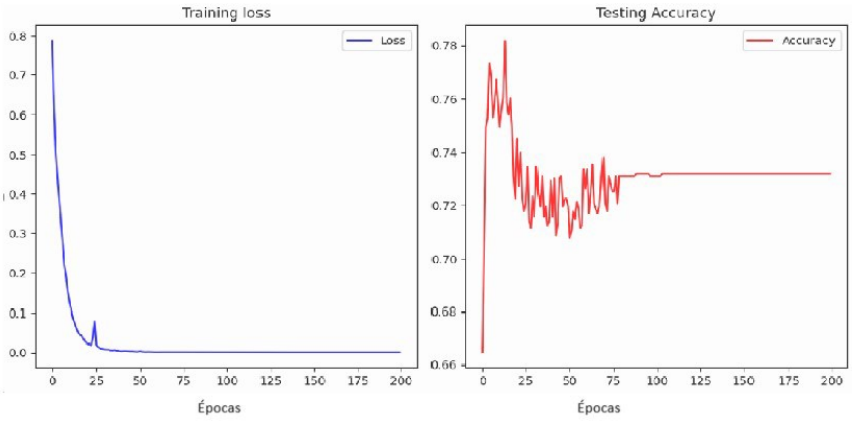


Figura 3.9: Percas e precisão no treino no Modelo A.3

Métricas	**Accuracy**	**F1-Score**	**Recall**
Valor	≈ 0.67	≈ 0.71	≈ 0.76

Tabela 3.2: Performance do Modelo A.3

Uma vez que a precisão atingida no modelo anterior não satisfez os requisitos (precisão superior a 80%) e o poder computacional disponível para efetuar mais treinos é reduzido, foram estudados modelos pré-treinados como os indicados em 2.4.3. No caso, o primeiro modelo pré-treinado utilizado foi o VGG16 visível na Tabela A.4. O treino foi realizado com um *batch size* de 8 e com 100 épocas demorou 406 minutos e permitiu obter os resultados visíveis na Figura 3.10.

Com a análise das figuras é possível concluir que o modelo convergiu por volta das 98 épocas, o que indica que não é necessário aumentar o número de épocas de treino. Na Tabela 3.3 é ainda visível as métricas obtidas para este segundo modelo.

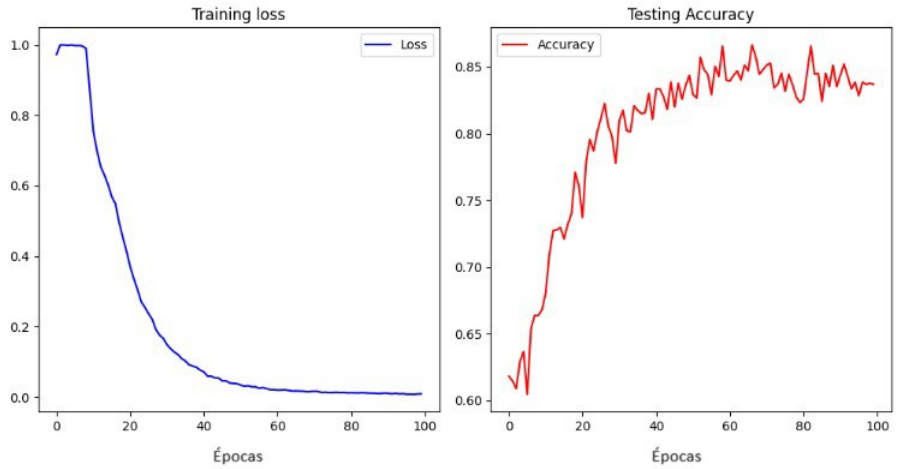


Figura 3.10: Percas e precisão no treino no Modelo A.4

Métricas	**Accuracy**	**F1-Score**	**Recall**
Valor	≈ 0.78	≈ 0.78	≈ 0.78

Tabela 3.3: Performance do Modelo A.4

Uma vez mais, como a precisão atingida no modelo anterior não satisfez os requisitos (precisão superior a 80%) foi utilizado outro modelo pretreinado o Xception visível na Tabela A.5, que consiste numa arquitetura que possui um total de 71 camadas, entre elas 36 de convolução. O treino foi realizado com um *batch size* de 16 e com 100 épocas demorou 226 minutos e permitiu obter os resultados visíveis na Figura 3.11.

Com a análise das figuras é possível concluir que o modelo convergiu por volta das 90 épocas, o que indica que não é necessário aumentar o número de épocas de treino. Na Tabela 3.4 é ainda visível as métricas obtidas para este segundo modelo.

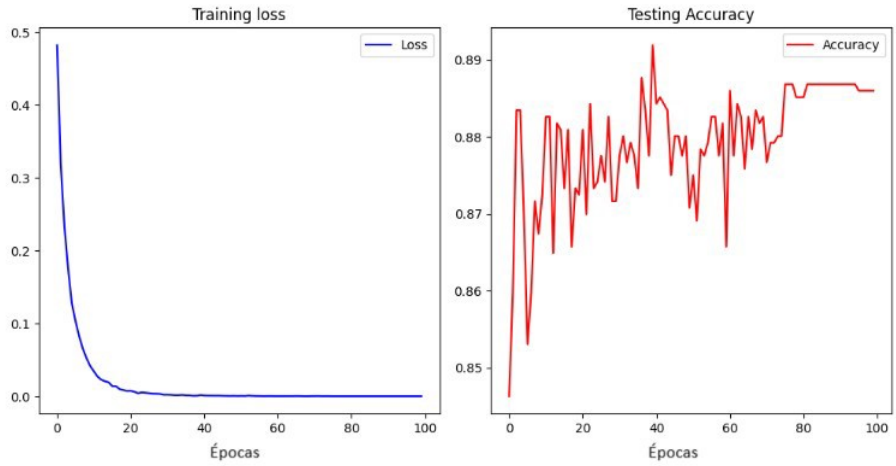


Figura 3.11: Percas e precisão no treino no Modelo A.5

Métricas	**Accuracy**	**F1-Score**	**Recall**
Valor	≈ 0.83	≈ 0.84	≈ 0.85

Tabela 3.4: Performance do Modelo A.5

Uma vez que a precisão atingida no modelo superou os requisitos, foi selecionado para implementar no protótipo.

Assim, para além do treino dos modelos foi necessário construir a lógica de utilização dos mesmos e a integração da sua utilização com o resto do projeto.

Para isso foram construidas as seguintes funções: capture_and_extract_faces, classify_images e compare. A primeira função a ser utilizada é a capture_and_extract_faces, esta função utiliza a PiCam para captar uma foto do paciente e extrair a sua respetiva face. Após esta extração é utilizado o método ***compare*** que acede a base de dados para saber a localização das fotos armazenadas do paciente durante o *setup* e compara as mesmas com a foto captada anteriormente, é também nesta função que a classify_images 3.2 é utilizada.

Este último método é responsável por carregar o modelo e comparar duas fotos, para as comparar é calculada a distância L2 e é fornecido um *treshold*. Para ajudar a escolher este threshold durante a fase de treino foi criado um gráfico que forneceu a média da distância entre a Âncora e o Positivo e a Ancora e o Negativo,Figura 3.12, é possível visualizar que a média da distância quando as imagens são iguais é cerca de 0.8 e quando não são diferentes é cerca de 1.6. Assim, definiu-se que para L2 menores que 1, as imagens são similares. No final, se 80% ou mais da totalidade das fotos comparadas derem similares, o paciente é considerado autenticado A.1.

```
def classify_images(face_list1, face_list2, threshold=1):
    try:
        # Load the saved encoder
        encoder = load_model('./model/encoder.h5', compile=False)

        # Getting the encodings for the passed faces
        tensor1 = encoder.predict(face_list1)
        tensor2 = encoder.predict(face_list2)

        distance = np.sum(np.square(tensor1-tensor2), axis=-1)
        prediction = np.where(distance <= threshold, 0, 1)
        logger.info(f"Prediction value: {prediction}")

        return prediction
    except Exception as e:
        logger.error(f"An error occured: {e}")
        return None
```

Listagem 3.2: Função que calcula a distancia entre duas fotos utilizando L2

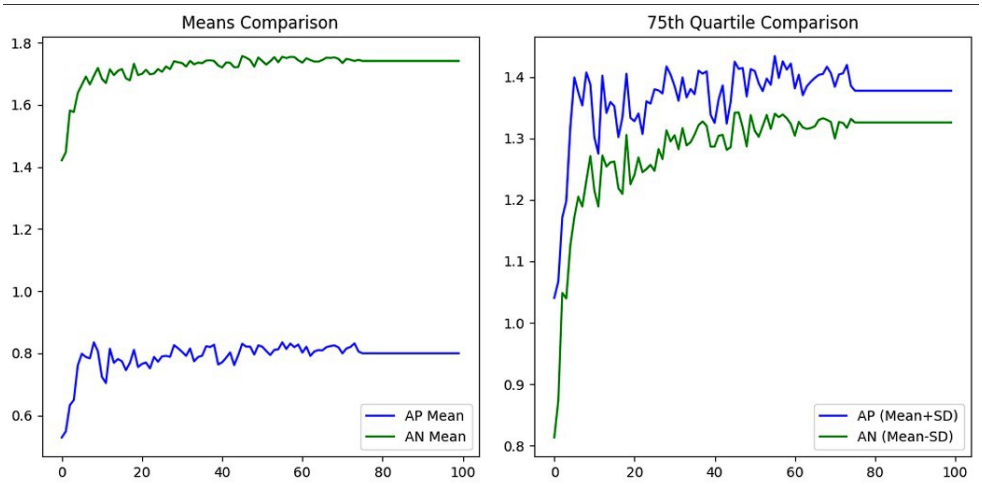


Figura 3.12: Média das distâncias durante o treino do modelo A.5

Resultados

Obteve-se, então, um modelo em formato *h5* com cerca de 80MB que permite efetuar a tarefa pretendida. Com este modelo como visualizado no Código 3.2 foi possível saber se duas imagens são similares ou não. Na Figura 3.13 é ainda visível a aplicação real do modelo a comparar duas fotografias.

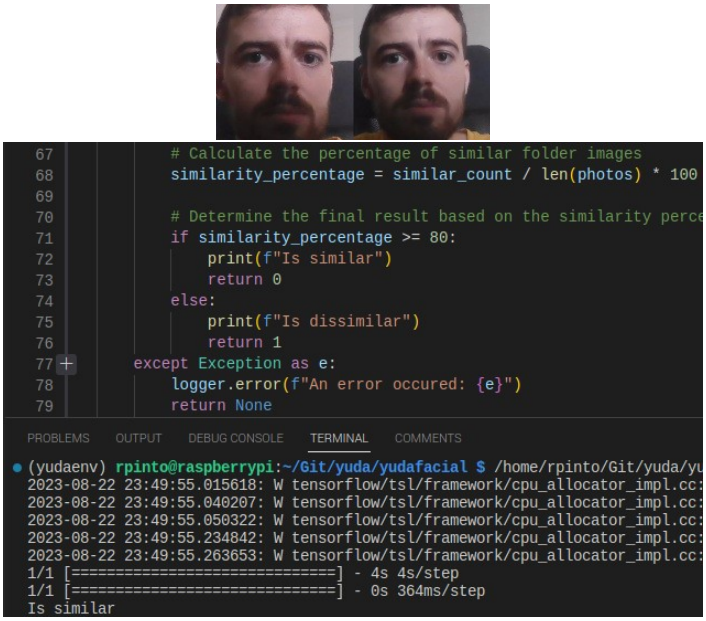


Figura 3.13: Exemplo real do modelo A.5 a comparar duas fotos.

3.3.4 Implementação Sistema RFID

Após atingida a precisão de 80% no model de reconhecimento facial, desenvolveu-se este módulo de modo a obter uma maior segurança e fornecer funcionalidades extra, por exemplo, um cartão de administrador que permite aceder ao sistema sem efetuar reconhecimento facial.

Este submódulo é constituído por quatro funções: rfid_active, rfid_read, get_medication_time e check_user_card. A função principal é a rfid_activate, esta função começa por criar uma conexão à base de dados e obter a data para a qual a medicação foi requerida e o intervalo de toma da mesma, para obter esta informação é utilizado o método get_medication_time.

De seguida, é efetuada a lógica do tempo que consiste em verificar se a data atual é superior à data de requisição da medicação e se a data atual se encontra no intervalo correto de toma (existe um intervalo de 15 minutos de tolerância). Se ambos os requisitos forem cumpridos o paciente pode passar o cartão no leitor rfid aqui os métodos rfid_read e check_user_card são utilizados para verificar se o cartão contem os dados corretos referentes ao paciente. Estes dados estão guardados na base de dados na Tabela *pacient* 3.3. Se a validação do cartão for efetuada com sucesso é ativada a validação do reconhecimento facial, se ambas forem validadas um servo motor é ativado, como representação da dispensação dos medicamentos.

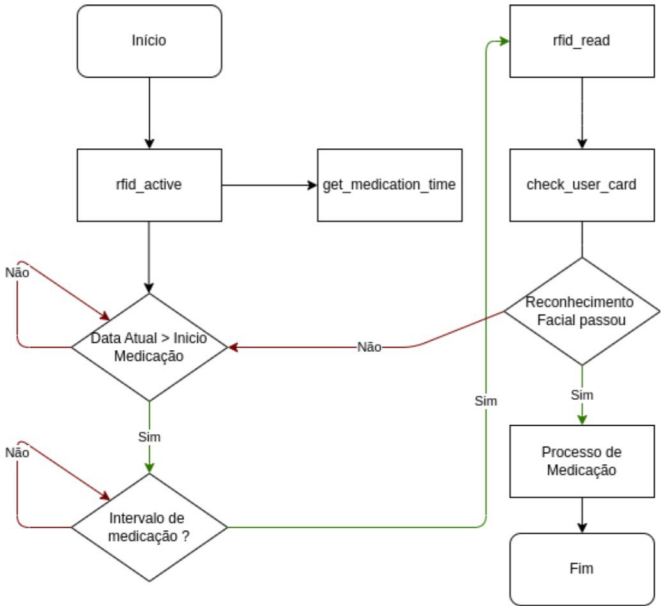


Figura 3.14: Diagrama do código de Segurança

O processo seguinte consistiu na ligação física do módulo mfrc522 (leitor rfid) com o RaspberryPi 3.20 e a inserção de dados necessários na tabela da base de dados de maneira a testar o sistema.

Resultados

Este módulo foi implementado com sucesso, a informação acerca do paciente incluindo o número de cartão do mesmo é armazenada e é possível posteriormente aceder à mesma e comparar com as informações do cartão. Foi, ainda, criada uma função que permite editar a informação contida no cartão, esta funcionalidade pode ser útil futuramente. Na Figura 3.15 é visível o funcionamento do modelo numa aplicação real.

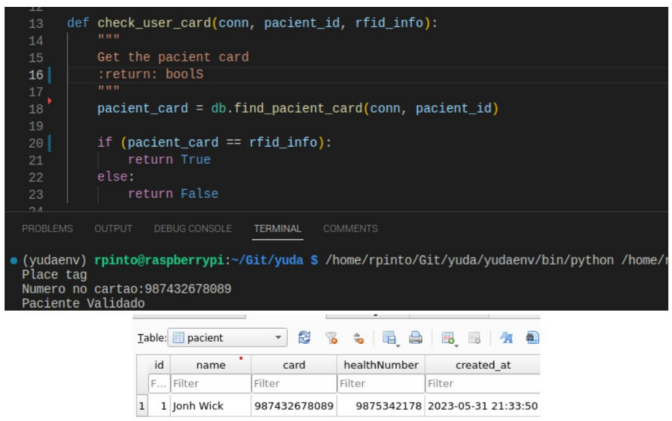


Figura 3.15: Exemplo real do sistema RFID a funcionar.

3.4 Módulo do Processo de Medicação

Esta interface deverá permitir efetuar o seguinte conjunto de tarefas:

- Inserir e consultar base de dados;
- Ativar o servo motor responsável por dispensar comprimidos;
- Acionar leds com o intuito de informar o utilizador se certos processos correram como planeado ou não;

3.4.1 Software

Para conformidade com os restantes módulos a linguagem utilizada foi Python. Para efetuar as ações mencionadas acima recorreu-se à construção de módulos distintos, estes módulos vão ser divididos em três:

- módulo responsável por efetuar leituras e registos na base de dados;
- módulo responsável pelo processo de dispensar comprimidos (servomotor);
- módulo responsável por outras ações que permitem ao utilizador saber o estado de alguns processos leds.

Esta subdivisão permite isolamento e futura adição de funcionalidades extra como, por exemplo, um ecrã lcd. Para facilitar o processo de desenvolvimento e diminuir o tempo requerido as seguintes livrarias foram utilizadas:

- RPi.GPIO - provisiona controlo sobre o GPIO de um RaspberryPi de uma forma mais facilitada, possui alguns problemas como o facto de não possuir suporte para protocolos de comunicação como SPI e I2C.
- sqlite3 - sqlite é uma livraria em C que permite a criação de uma base de dados leve, deixa de ser assim necessário um servidor separado. Outra vantagem é opção de migração para bases de dados maiores como o PostgreSQL ou Oracle.

Descrição

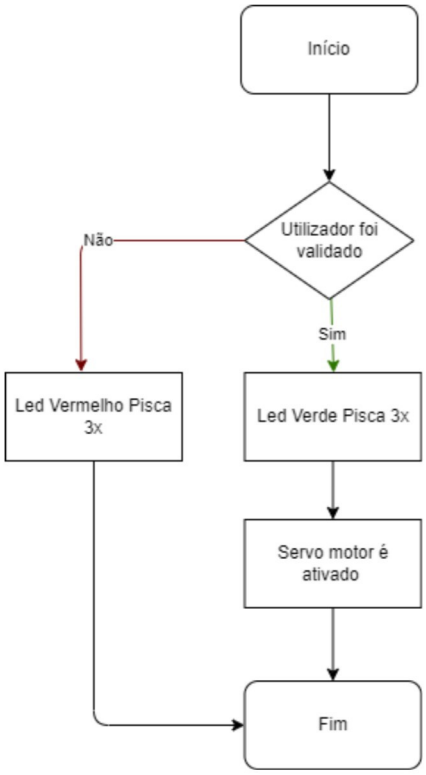


Figura 3.16: Fluxograma geral do processo de medicação

3.4.2 Hardware

Para a realização deste módulo será necessário adicionar os componentes listados abaixo;

- 3 leds de cores distintas(verde,vermelho,azul);
- Servomotor;

3.4.3 Implementação

A implementação deste módulo foi a mais simples, uma vez que apenas possui métodos que permitem acionar os leds e aceder a base de dados. Os seguintes métodos foram criados para controlo do sistema de leds:

- **leds_init**: Configura GPIO.
- **life_lef_blink**: Faz piscar o led azul a cada um segundo.
- **red_blink**: Faz o led vermelho piscar três vezes com um intervalo de um segundo.

- **green_blink**: Faz o led verde piscar três vezes com um intervalo de um segundo.

Relativamente ao acesso à base de dados os seguintes métodos foram criados:

- **create_connection**: Cria uma ligação à base de dados SQLite.
- **find_medication_time(pacient_id)**: Procura a hora de início e o intervalo de tempo da medicação para um paciente específico.
- **find_pacient_card(pacient_id)**: Procura o número do cartão do paciente na tabela 'paciente' para um paciente específico.
- **find_pacient_photo_path(pacient_id)**: Procura para a pasta que contem as fotos do paciente na tabela 'facial', com base no 'pacient_id'.

Resultados

Este módulo foi parcialmente desenvolvido com sucesso. Sendo que foi desenvolvido o acesso à base de dados 3.17, de forma a obter os medicamentos e os respetivos horários de toma, e a lógica de funcionamento da dispensação, ou seja, a ativação do sistema aquando do horário da toma, os alertas visuais, através dos leds, que permitem identificar se o sistema está em funcionamento, se o utilizador é válido, e por fim, a ativação da dispensação simulada pelo servomotor. Sendo um passo posterior aliar um modelo mecânico preciso para a dispensação dos medicamentos, através de sensuares e atuadores, e trabalhar na lógica da dispensação de múltiplos medicamentos no mesmo intervalo de tempo.

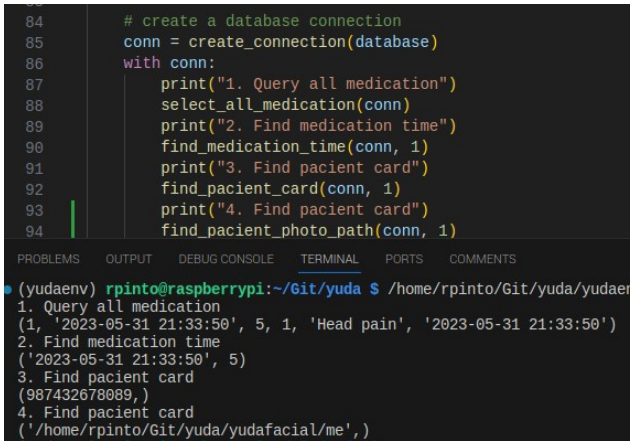


Figura 3.17: Acesso à base de dados

3.5 Resultado

Para concluir o projeto foi necessário harmonizar a funcionalidade de todos os componentes simultaneamente, para tal foi utilizado um sistema de threads 3.18 que é ativado quando um botão (ON/OFF) é pressionado. Após a atuação do botão são lançadas três threads que ativam o assistente pessoal controlado por voz (módulo de Interação),o sistema rfid (módulo de Segurança e Reconhecimento) e um led (*Life led*), cuja funcionalidade é piscar para saber que o sistema de medicação está em funcionamento.

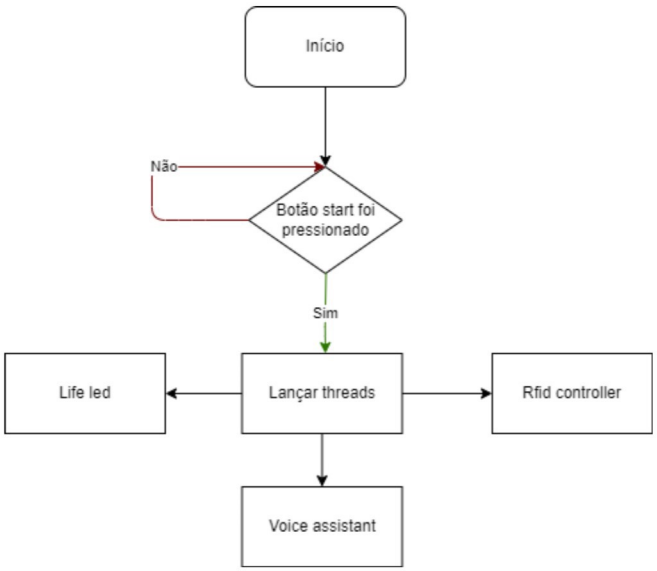


Figura 3.18: Diagrama da função main

Para ser possível a funcionalidade do botão, o ficheiro **.config** do RaspberryPi foi alterado, de forma a que sempre que o mesmo é ligado a seguinte função main 3.3 começa a ser executada em segundo plano. Esta função cria uma conexão à base de dados e ao sistema de fala, o que faz com que o modelo de fala seja carregado apenas uma vez durante toda a execução do código, por esta razão é efetuado, também, o *load* do modelo de reconhecimento facial.

```
1  if __name__ == '__main__':
2      yuda, conn = init()
3      # Set up Button GPIO
4      GPIO.setup(BUTTON_GPIO, GPIO.IN, pull_up_down=GPIO.PUD_UP)
5      GPIO.add_event_detect(BUTTON_GPIO, GPIO.FALLING,
6                            callback=button_released_callback,
                                 bouncetime=200)
7      signal.signal(signal.SIGINT, signal_handler)
8
9      while True:
10         print("RUNNING:", RUNNING)
```

```
11          print("ACTIVE_THREADS:", len(ACTIVE_THREADS))
12          for thread in ACTIVE_THREADS:
13              print("Thread name: " + thread.name +
14                    " is alive: " + str(thread.is_alive()))
15          print("The cpu usage is: " + str(psutil.cpu_percent()))
16          print("The memory usage is: " + str(psutil.virtual_memory
                ()[2]))
17          time.sleep(5)
```

Listagem 3.3: Função main

Na Figura 3.19 é possível visualizar o modelo final em ação e a autenticação de um paciente com sucesso.

Após clicar no botão ON/OFF o sistema ativa, então, os 3 sistemas referidos, ficando o LED life a piscar enquanto o sistema aguarda por comandos de voz ou pela aproximação do cartão RFID se for o horário da toma de medicação .

O utilizador recorre ao comando **"Hey Sarah"** para ativar a assistente pessoal. Note-se que **User:**, corresponde ao que o sistema interpretou do que o Utilizador disse, **Yuda:** corresponde à resposta que o sistema forneceu. Pode, ainda, verificar-se que houve um comando que foi mal identificado pelo sistema em vez de **Goodbye Sarah** o sistema interpretou **to by Sarah**.

O utilizador recorreu, ainda, ao comando **"tell me the time"** para obter a informação das horas, uma das funcionalidades presentes no dispensador.

Observa-se, também, após a compreensão do comando **"Goodbye Sarah"**, a autenticação do paciente, para tal o utilizador aproxima o cartão RFID do leitor e a sua cara da câmara.

Nesta fase pode ler-se no código: **Cartão OK**, que corresponde à autenticação do cartão, *path*, que demonstra o acesso à base de dados para obter o local onde se encontram as fotos armazenadas do utilizador. E, por último, as duas barras no fim da figura que correspondem ao modelo de IA treinado em funcionamento, enquanto este averigua se o utilizador captado é o paciente ao qual se destina a medicação. Uma vez que as fotos eram idênticas, ou seja, o paciente era o correto, surgiu a indicação **Utilizador OK**.

Sendo posteriormente ativa a parte de dispensação do módulo de Processo de Medicação, o led verde pisca 3x e servomotor é ativado, como indicado no código.

Conclui-se, assim, que os sistemas estão a funcionar correta e simultaneamente.

```
RUNNING: False
ACTIVE_THREADS: 0
The cpu usage is: 19.9
The memory usage is: 81.3
Starting button pressed
Listening...
Coloque o cartao no leitor
hey sarah
User:  hey sarah
Yuda: Hello, I'm sarah
Listening...
Listening...
tell me the time
User:  tell me the time
Yuda:  15:20
Listening...
Listening...
to by sarah
User:  to by sarah
Yuda:  I don't know how to do that
Listening...
goodbye sarah
User:  goodbye sarah
Yuda:  Goodbye!
Listening...
Cartao OK
Listening...
/home/rpinto/Git/yuda/yudafacial/me
Listening...
1/1 [==============================] - 4s 4s/step
1/1 [==============================] - 0s 390ms/step
Utilizador OK
Listening...
Servo a rodar 90 graus
```

Figura 3.19: Todos os módulos em funcionamento

Na Figura 3.20 são visíveis todas as ligações físicas de *hardware* efetuadas entre os diversos componentes em formato esquemático e em formato real.

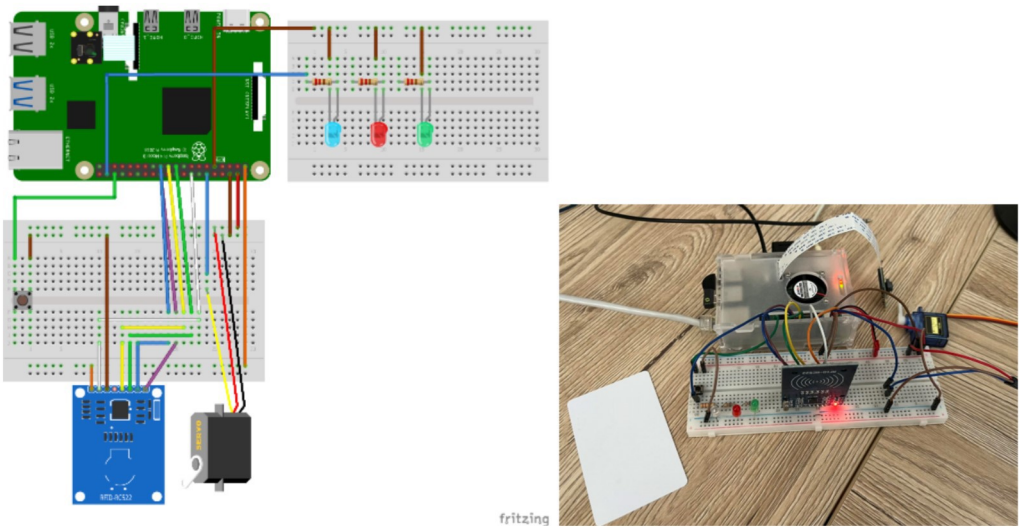


Figura 3.20: Esquema das ligações efetuadas

Capítulo 4

Conclusões

A presente dissertação teve como principal objetivo o desenvolvimento de um dispensador de medicamentos. O protótipo destina-se a auxiliar os utilizadores a cumprirem as suas receitas médicas, de forma a minimizar a incorreta adesão à medicação, que pode ter graves consequências para a saúde. Atualmente existem no mercado alguns produtos que ajudam a gerir a medicação, mas após um estudo foram encontradas algumas limitações, mais concretamente a falta de verificação/validação do paciente em questão aquando da toma da medicação.

Assim, procurou-se resolver esta limitação com a criação de uma solução aberta e versátil, capaz de reconhecer o paciente através da utilização de um cartão de identificação e via ferramentas de reconhecimento facial. Além disso, o protótipo apresenta a capacidade de interação com o utilizador através de comandos de voz.

No horário da toma o utilizador tem de passar o seu cartão pessoal e posicionar-se de maneira a que o dispositivo possa captar uma foto da sua cara com o intuito de validar que é o utilizador a que a medicação se destina.

Para tal, foi realizado o estudo das diversas ferramentas de reconhecimento facial, de forma a escolher a ferramenta mais adequada e treinar a mesma.

Assim, foi treinado um modelo de reconhecimento facial utilizando Siamese Network, obteve-se, desta forma, um modelo cuja precisão ronda os 83%. A partir deste é possível validar o paciente aquando da toma, através da comparação de uma foto do momento com um conjunto de imagens previamente inseridas do mesmo.

Destaca-se, ainda, a possibilidade do utilizador interagir com o dispensador via

comandos de voz, o que facilita a utilização deste para um paciente com as capacidades cognitivas mais debilitadas, como os idosos. Como mencionado anteriormente, para desenvolver este sistema foram utilizadas livrarias no python previamente desenvolvidas, a partir destas foi possível programar o protótipo de maneira a responder a diferentes comandos com ações específicas.

As diversas funcionalidades tiveram de ser pensadas de raiz, envolvendo um código extenso e relativamente complexo para permitir alguma flexibilidade e capacidade de adaptação e expansão.

As funcionalidades mencionadas foram, então, implementadas, resultando num protótipo inovador face aos produtos existentes no mercado.

De referir que a interação do médico ou cuidador com o sistema e registo da toma de medicação no mesmo não foram desenvolvidas, esta decisão foi feita com base na prioridade das ferramentas anteriormente mencionadas e gestão do tempo destinado ao projeto.

De notar que não foi possível implementar o protótipo em 3D, sendo apenas desenvolvidas as funcionalidades a implementar no mesmo. No entanto, conforme referido, não fazia parte do contexto do projeto o desenvolvimento da estrutura física do dispensador.

4.1 Trabalho Futuro

Para além de finalizar o previamente proposto, foram idealizadas possíveis melhorias ao trabalho efetuado e identificados pontos promissores a desenvolver futuramente. Entre os quais: adicionar um sistema de notificações e um sistema de verificação da toma da medicação. Isto tornaria mais seguro e eficiente o dispensador, pois poderia, por exemplo, notificar algum familiar caso a medicação não tivesse sido tomada.Uma questão a aprofundar seria o estudo da dispensação dos comprimidos, procurando uma forma de dispensar os comprimidos com elevada precisão, isto exigiria a construção de um protótipo 3D, o recurso a mais componentes para além do servomotor e o desenvolvimento de uma placa de circuitos impressos.

Outras melhorias que poderiam ser efetuadas, seriam, a partir da recolha de informação de cenários reais, construir um *dataset* com mais informação e treinar novamente os modelos de forma a melhorá-los. Além disso, poderia ser construida uma lógica de base de dados mais complexa, que suportasse um maior número de entradas, recorrendo, por exemplo, a um sistema *cloud* como o Azure disponibilizado pela Windows, neste caso o modelo teria de aceder a *internet* esporadicamente. Por fim, podia ainda ser efetuado um estudo das licenças utilizadas e do custo de produção de forma a minimizar o mesmo e avaliar a possível comercialização do dispensador.

Referências

[1] "Envelhecimento - nações unidas - onu portugal." [Citado na página 1]

[2] "Censos 2021 - população." [Citado na página 1]

[3] I. Castilho, E. Rocha, S. Magalhães, Z. Vaz, and A. Costa, "Polifarmácia e utilização de medicação potencialmente inapropriada no idoso com idade igual ou superior a 75 anos: O caso de uma unidade de saúde familiar," *Acta Médica Portuguesa*, vol. 33, pp. 622–632, 2020. [Citado na página 1]

[4] "World population prospects - population division - united nations." [Citado nas páginas ix e 2]

[5] C. Mosca and P. Correia, "O medicamento no doente idoso," *Acta Farmacêutica Portuguesa*, vol. 1, pp. 75–81, 6 2012. [Citado nas páginas 2 e 4]

[6] "Medicação: erros podem comprometer tratamentos." [Citado nas páginas ix, 2 e 3]

[7] L. Boquete, J. M. Rodriguez-Ascariz, I. Artacho, J. Cantos-Frontela, and N. Peixoto, "Dynamically programmable electronic pill dispenser system," *Journal of medical systems*, vol. 34, pp. 357–366, 2010. [Citado na página 3]

[8] A. L. B. Abrahão, M. J. R. M. Vairinhos, and V. A. da Silva Branco, "Proposta de uma metodologia para o design de dispensadores de medicamentos, baseados em media tangíveis, para seniores de baixa literacia/proposal for a methodology for the design of medication dispensers, based on tangible media, for seniors with low literacy," [Citado na página 3]

[9] B. Z. T. Martins, *Desenvolvimento de um Protótipo de Dispensador Automático de Medicamentos*. PhD thesis, Instituto Politecnico do Porto (Portugal), 2015. [Citado na página 3]

[10] D. M. G. Cancela, "O processo de envelhecimento," *Trabalho realizado no Estágio de Complemento ao Diploma de Licenciatura em Psicologia pela Universidade Lusíada do Porto*, vol. 3, no. 1, 2007. [Citado na página 4]

[11] R. S and B. J, "Artificial intelligence. fears of an ai pioneer.," *Science (New York, N.Y.)*, vol. 349, pp. 252–252, 7 2015. [Citado na página 7]

[12] M. Campbell, A. J. Hoane, and F. H. Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, pp. 57–83, 1 2002. [Citado nas páginas ix e 8]

[13] K. Dwivedi, M. Sharkey, R. Condliffe, J. Uthoff, S. Alabed, P. Metherall, H. Lu, J. Wild, E. Hoffman, A. Swift, and D. Kiely, "Pulmonary hypertension in association with lung disease: Quantitative ct and artificial intelligence to the rescue? state-of-the-art review," *Diagnostics*, vol. 11, p. 679, 04 2021. [Citado nas páginas ix e 8]

[14] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science & Technology*, vol. 9, pp. 14–14, 1 2020. [Citado nas páginas ix e 9]

[15] "2.3 terminology | interpretable machine learning." [Citado na página 9]

[16] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, p. 160, 2021. [Citado nas páginas ix, 10 e 11]

[17] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, "An introduction to machine learning," *CLINICAL PHARMACOLOGY & THERAPEUTICS | VOLUME*, vol. 107, 2020. [Citado na página 10]

[18] "Supervised vs. unsupervised learning in machine learning | springboard." [Citado na página 10]

[19] "Supervised vs. unsupervised learning: What's the difference? | ibm." [Citado na página 11]

[20] "Issues in machine learning - javatpoint." [Citado na página 11]

[21] W. Follow, "Overrtting vs. underrtting: A complete example," 2018. [Citado na página 11]

[22] "What is underfitting? | ibm." [Citado na página 11]

[23] R. Ashmore, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Computing Surveys*, vol. 0, 2021. [Citado nas páginas ix, 11 e 12]

[24] "Data augmentation | tensorflow core." [Citado na página 12]

[25] "A complete guide to data augmentation | datacamp." [Citado na página 12]

[26] P. Picton, "What is a neural network?," *Introduction to Neural Networks*, pp. 1–12, 1994. [Citado na página 12]

[27] "What is a neural network? - artificial neural network explained - aws." [Citado na página 12]

[28] "First neural network for beginners explained (with code) | by arthur arnx | towards data science." [Citado nas páginas ix e 13]

[29] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 6999–7019, 12 2022. [Citado na página 13]

[30] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-January, pp. 1–6, 3 2018. [Citado na página 13]

[31] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015. [Citado na página 14]

[32] "Mnist handwritten digit database, yann lecun, corinna cortes and chris burges." [Citado na página 14]

[33] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018. [Citado nas páginas ix, x, 14, 15 e 64]

[34] "Cnn | introduction to pooling layer - geeksforgeeks." [Citado nas páginas ix e 15]

[35] "What are recurrent neural networks? | ibm." [Citado na página 16]

[36] A.-N. Sharkawy, "Principle of neural network and its main types: Review," *Journal of Advances in Applied & Computational Mathematics*, vol. 7, pp. 8–19, 8 2020. [Citado na página 16]

[37] "Explaining recurrent neural networks - bouvet norge." [Citado nas páginas ix e 16]

[38] M. Heidari and K. Fouladi-Ghaleh, "Using siamese networks with transfer learning for face recognition on small-samples datasets," in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pp. 1–4, IEEE, 2020. [Citado na página 17]

[39] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015. [Citado nas páginas 17 e 26]

[40] "Siamese networks with keras, tensorflow, and deep learning - pyimagesearch." [Citado nas páginas ix e 17]

[41] "Various ways to evaluate a machine learning model's performance | by kartik nighania | towards data science." [Citado nas páginas 17 e 18]

[42] "Various ways to evaluate a machine learning model's performance | by kartik nighania | towards data science." [Citado na página 17]

[43] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, pp. 427–437, 2009. [Citado nas páginas 17 e 18]

[44] "Sklearn documentation." [Citado na página 18]

[45] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," *Proceedings of the 23rd international conference on Machine learningt*, 2006. [Citado na página 19]

[46] O. P. Trifonova, P. G. Lokhov, and A. I. Archakov, "[metabolic profiling of human blood].," *Biomedit{combining double inverted breve}sinskai{combining double inverted breve}a khimii{combining double inverted breve}a*, vol. 60, pp. 281–294, 2014. [Citado nas páginas ix e 19]

[47] I. C. Education, "Speech recognition." Online, May 2022. [Citado na página 19]

[48] R. D. Peacocke and D. H. Graf, "An introduction to speech and speaker recognition," in *Readings in Human–Computer Interaction* (R. M. BAECKER, J. GRUDIN, W. A. BUXTON, and S. GREENBERG, eds.), Interactive Technologies, pp. 546–553, Morgan Kaufmann, 1995. [Citado nas páginas ix, 19 e 21]

[49] SONIX, "A short history of speech recognition." Online, May 2022. [Citado nas páginas ix, xi, 20 e 21]

[50] Letrário, "A brief history of asr: Automatic speech recognition." Online, May 2022. [Citado nas páginas ix e 20]

[51] A. Y. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," *CoRR*, vol. abs/1904.02619, 2019. [Citado na página 22]

[52] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and Trends R in Signal Processing*, vol. 1, pp. 195–304, 2007. [Citado na página 22]

[53] R. Errattahi, A. El Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," *Procedia Computer Science*, vol. 128, pp. 32–37, 2018. 1st International Conference on Natural Language and Speech Processing. [Citado na página 22]

[54] R. Haldar and D. Mukhopadhyay, "Levenshtein distance technique in dictionary lookup methods: An improved approach," 2011. [Citado na página 23]

[55] W. Zhao and R. Chellappa, *Face Processing: Advanced Modeling and Methods*. Electronics & Electrical, Elsevier / Academic Press, 2006. [Citado na página 24]

[56] L. F. Z. Braga, "Sistemas de reconhecimento facial," Master's thesis, Escola de Engenharia de São Carlos, da Universidade de São Paulo, 2013. [Citado na página 24]

[57] "The real world impact of facial detection and recognition - university of york." [Citado nas páginas 24 e 25]

[58] L. R. Carlos-Roca, I. H. Torres, and C. F. Tena, "Facial recognition application for border control," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, 10 2018. [Citado na página 24]

[59] "How ai is using facial detection to spot rare diseases in children." [Citado na página 25]

[60] "Nvlabs/ffhq-dataset: Flickr-faces-hq dataset (ffhq)." [Citado na página 25]

[61] "Lfw face database : Main." [Citado na página 25]

[62] "Utkface | large scale face dataset." [Citado na página 25]

[63] J. M. Phillips, "Data mining; spring 2013 instructor," [Citado na página 26]

[64] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to SGD," *CoRR*, vol. abs/1712.07628, 2017. [Citado nas páginas 26 e 27]

[65] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017. [Citado na página 26]

[66] "Image similarity estimation using a siamese network with a triplet loss." [Citado na página 27]

[67] "Euclidean distance - an overview | sciencedirect topics." [Citado na página 27]

[68] F. Schroff and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," 2015. [Citado na página 27]

[69] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. [Citado na página 27]

[70] T. Gwyn, K. Roy, M. Atay, and S. Furnell, "Face recognition using popular deep net architectures: A brief comparative study," 2021. [Citado na página 27]

[71] A. Lee, "What is a pretrained ai model? | nvidia blog," 2022. [Citado na página 27]

[72] Y. Arslan, K. Allix, L. Veiber, C. Lothritz, T. F. Bissyandé, J. Klein, and A. Goujon, "A comparison of pre-trained language models for multi-class text classification in the financial domain," *The Web Conference 2021 - Companion of the World Wide Web Conference, WWW 2021*, pp. 260–268, 4 2021. [Citado na página 27]

[73] P. V. Lal, U. Srilakshmi, and D. Venkateswarlu, "Face recognition using deep learning xception cnn method," *Journal of Theoretical and Applied Information Technology*, vol. 31, 2022. [Citado na página 28]

[74] "Vgg-16 convolutional neural network - matlab vgg16." [Citado na página 28]

[75] S. Casciaro, L. Massa, I. Sergi, and L. Patrono, "A smart pill dispenser to support elderly people in medication adherence," *2020 5th International Conference on Smart and Sustainable Technologies, SpliTech 2020*, 9 2020. [Citado na página 28]

[76] N. B. Othman and O. P. Ek, "Pill dispenser with alarm via smart phone notification," *2016 IEEE 5th Global Conference on Consumer Electronics, GCCE 2016*, 12 2016. [Citado nas páginas 28 e 29]

[77] M. V. Moise, P. M. Svasta, and A. G. Mazare, "Programable iot pills dispenser," *Proceedings of the International Spring Seminar on Electronics Technology*, vol. 2020-May, 5 2020. [Citado na página 29]

[78] "Why python is a good programming language | learnpython.com." [Citado na página 35]

[79] J. Gonçalves, "Previsão inteligente das alterações metabólicas no cancro retal com base em modelos de machine e deep learning," 2021. [Citado nas páginas xi e 66]

Anexos

Anexo A

A.1 Capítulo 2

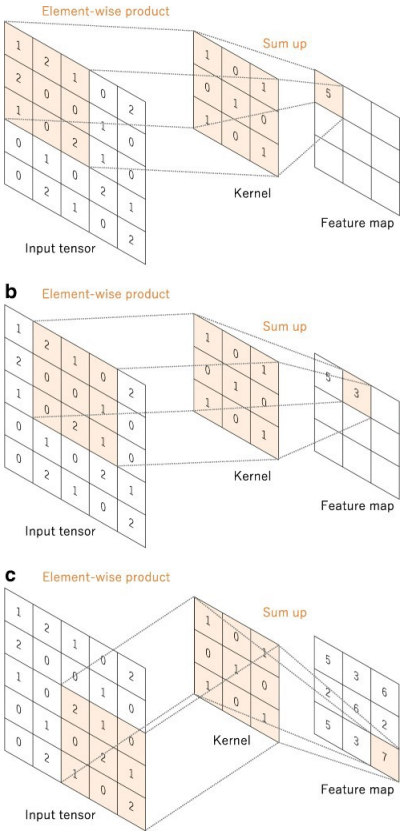


Figura A.1: Exemplo de uma operação de convolução [33].

A.2 Capítulo 3

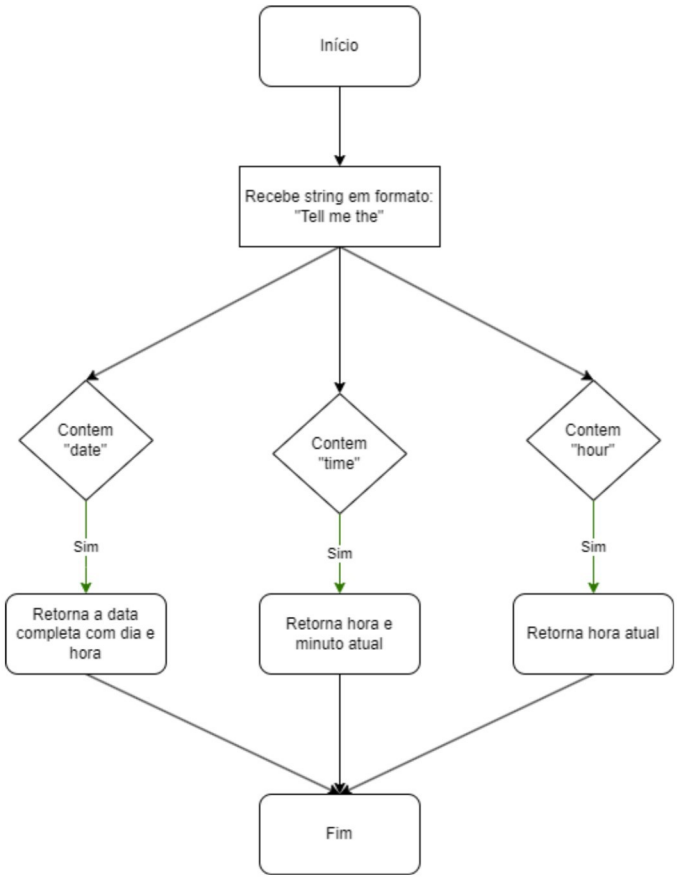


Figura A.2: Diagrama do código em date.py

Tabela A.1: Comparação entre diferentes livrarias [79].

	Scikit Learn	TensorFlow	PyTorch	Keras
Core	Python	C++	Python	Python
Custo	Free Open Source	Free Open Source	Free Open Source	Free Open Source
Plataforma	Linux, Windows, MacOs. Não suporta GPU.	Linux, Windows, MacOs. Nvidia GPUs e CUDA rec.	Linux, Windows, MacOs. CPUs e Nvidia GPUs.	Linux, Windows, MacOs. Backend:CNTK, Tensorflow.
Pros	Bom para inicantes. Permite explorar modelos de ML	Grande comunidade. Visualização superior de gráficos.	Inúmeros modelos pretreinados. Bom para projetos pequenos.	Ideal para aprender.
Cons	Apenas serve para pequenos projetos e *datasets*	Difícil de aprender. Requer muito codigo.	Sem apoio comercial para *cross-platform*	Ambiente pouco configurável.

Tabela A.2: Comparação entre os diferentes dispensadores de medicação disponíveis.

	Nrº Compartimentos	Aplicação Mobile	Funcionalidades	Outros	Preço(€)
Pria	Compartimentos: 28 Capacidade: 10	Sim	Sistema de fala Sistema de reconhecimento facial Monitorização		Inicial: 95 Mensal: 24
Hero	Compartimentos: 10 Capacidade: 90	Sim	Monitorização		Inicial: 93 Mensal: 45
LiveFine	Compartimentos: 28 Capaciade: 6	Não	Alerta o utilizador		65

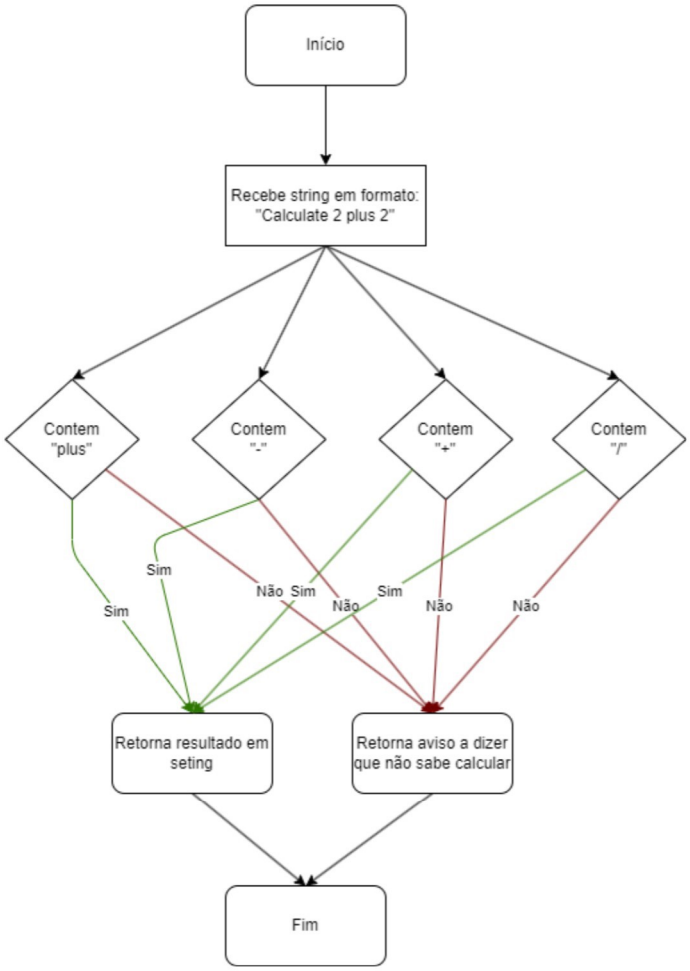


Figura A.3: Diagrama do código em calc.py

Tabela A.3: Rede Neuronal CNN do primeiro modelo (sem XCeption).

Layer	***Output Shape***	**Nr Parametros**
Conv2D	(None, 119, 119, 64)	19,264
MaxPooling	(None, 59, 59, 64)	0
Conv2D	(None, 58, 58, 128)	32,896
MaxPooling	(None, 29, 29, 128)	0
Conv2D	(None, 28, 28, 64)	32,832
MaxPooling	(None, 14, 14, 64)	0
Conv2D	(None, 11, 11, 256)	262,400
Flatten	(None, 30,976)	0
Dense	(None, 64)	1,982,528
BatchNormalization	(None, 64)	256
Dense	(None, 32)	2,080
Lambda	(None, 32)	0
Total params:	**2,334,944**	
Trainable params:	**2,334,816**	
Non-trainable params:	**128**	

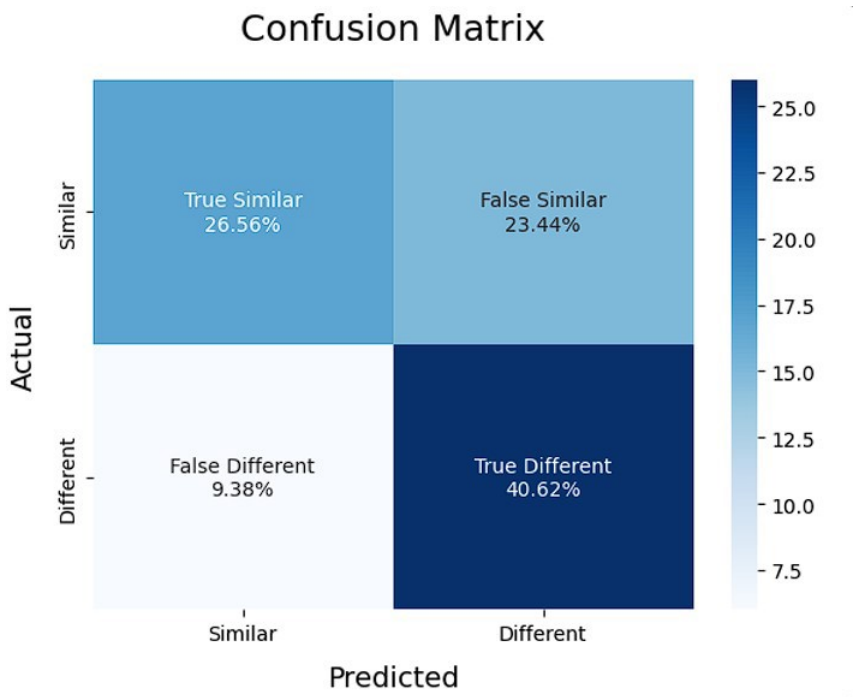


Figura A.4: Matriz de confusão primeiro modelo

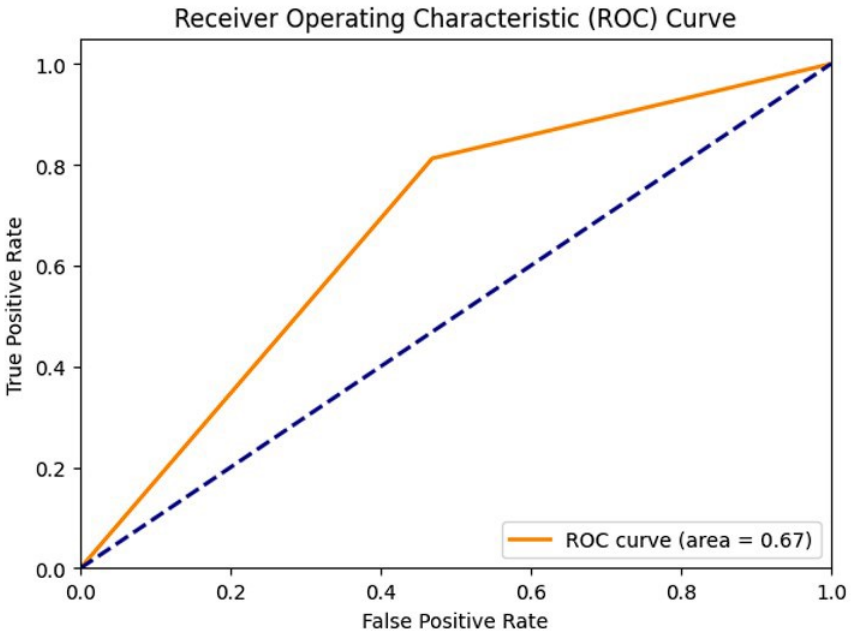


Figura A.5: Curva ROC primeiro modelo

Tabela A.4: Segundo modelo (com VGG16).

Layer	***Output Shape***	**Nr Parametros**
VGG16 (Functional)	(None, 512)	14,714,688
Flatten	(None, 512)	0
Dense	(None, 64)	32,832
BatchNormalization	(None, 64)	256
Dense	(None, 32)	2,080
Lambda	(None, 32)	0
Total params:	**14,749,856**	
Trainable params:	**14,749,728**	
Non-trainable params:	**128**	

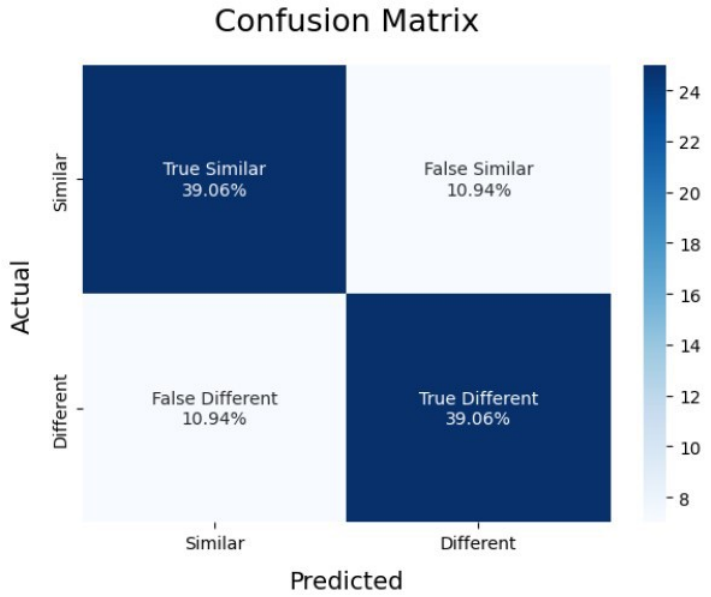


Figura A.6: Matriz de confusão do segundo modelo

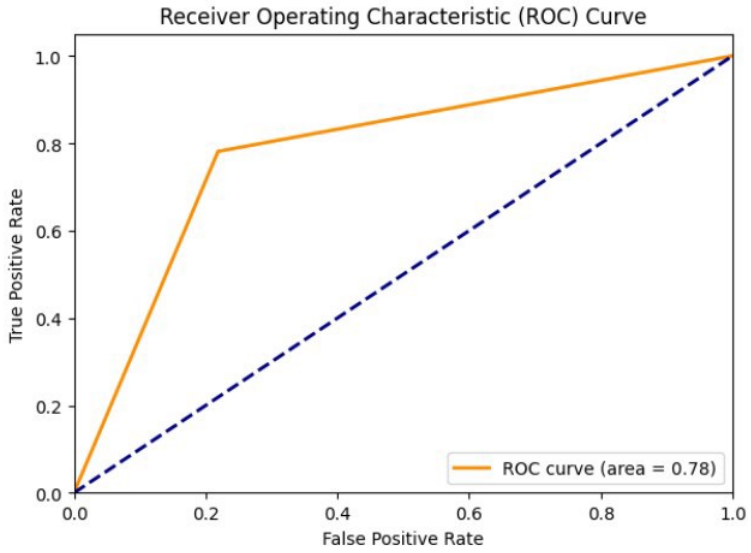


Figura A.7: Curva ROC do segundo modelo

Tabela A.5: Terceiro modelo (com XCeption).

Layer	***Output Shape***	**Nr Parametros**
Xception (Functional)	(None, 2048)	20,861,480
Flatten	(None, 2048)	0
Dense	(None, 64)	131,136
BatchNormalization	(None, 64)	256
Dense	(None, 32)	2,080
Lambda	(None, 32)	0
Total params:	**20,994,952**	
Trainable params:	**8,535,704**	
Non-trainable params:	**12,459,248**	

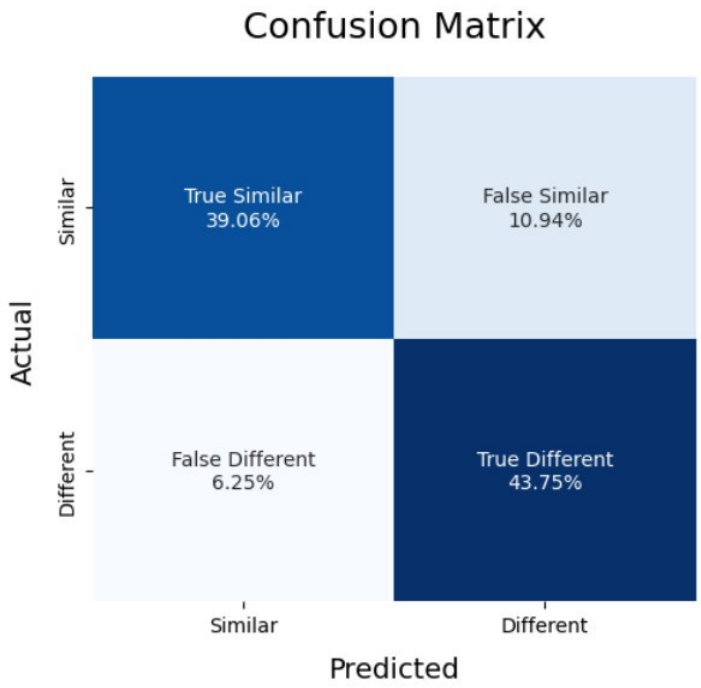


Figura A.8: Matriz de confusão do terceiro modelo

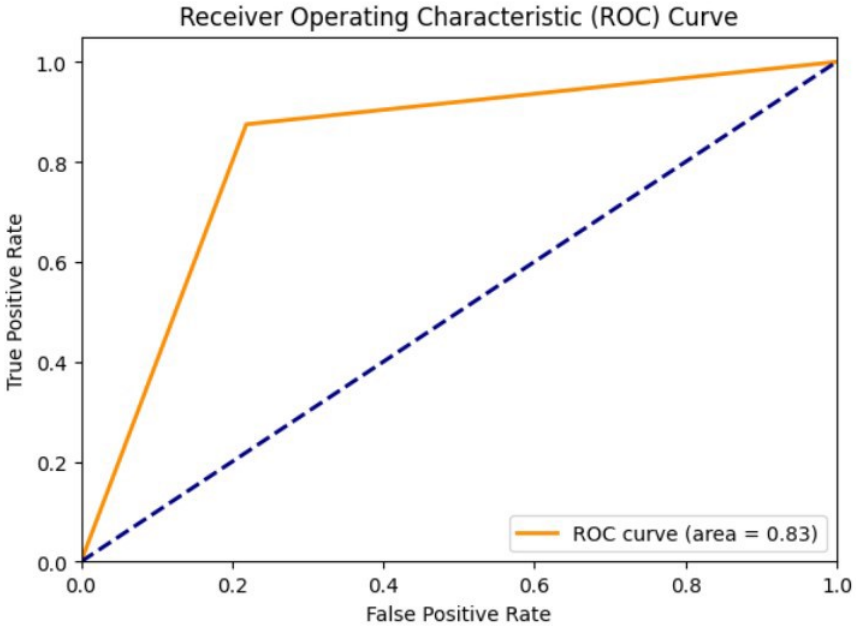


Figura A.9: Curva ROC do terceiro modelo

```
def compare(tmp_path, photos_path):
    try:
        #Get photo taken now
        tmp_photo = read_image(tmp_path)
        photos = os.listdir(photos_path)

        # Initialize a counter for similar predictions
        similar_count = 0

        # Iterate over the images in the folder
        for photo in photos:
            # Construct the full path to the image in the folder
            photo_path = os.path.join(photos_path, photo)

            # Read and preprocess the image from the folder
            photo_process = read_image(photo_path)

            # Compare the input photo with the image from the
                folder
            prediction = classify_images([tmp_photo], [
                photo_process])

            # Determine the result based on the prediction
            if prediction == 0:
                similar_count += 1

        # Calculate the percentage of similar folder images
        similarity_percentage = similar_count / len(photos) * 100

        # Determine the final result based on the similarity
            percentage for folder images
        if similarity_percentage >= 80:
            logger.facial.info(f"Is similar")
            return 0
        else:
            logger.facial.info(f"Is not similar")
            return 1
    except Exception as e:
        print("An error occured: ", e)
        logger.facial.error(f"An error occured: {e}")
        return None
```

Listagem A.1: Função que compara duas fotos