Universidade do Minho
Escola de Engenharia

Fábio Ferreira Cunha

Sistema de Aprendizagem para Redes de Sensores de Tato baseado em Visão por Computador e Supervised Learning

Fevereiro de 2022

Sistema de Aprendizagem para Redes de Sensores de Tato baseado em Visão por Computador e Supervized Learning

Fábio Ferreira Cunha

UMinho | 2022

Universidade do Minho
Escola de Engenharia

Fábio Ferreira Cunha

Sistema de Aprendizagem para Redes de Sensores de Tato baseado em Visão por Computador e Supervised Learning

Fevereiro de 2022

Universidade do Minho
Escola de Engenharia

Fábio Ferreira Cunha

Sistema de Aprendizagem para Redes de Sensores de Tato baseado em Visão por Computador e Supervised Learning

Dissertação de Mestrado
Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor António Fernando Macedo Ribeiro

Fevereiro de 2022

AGRADECIMENTOS

Esta dissertação é o fim de uma etapa que se iniciou há 5 anos. Ao longo deste trajeto, foram vários os apoios e incentivos concedidos, os quais foram extremamente importantes para o meu percurso. Sendo assim, não posso deixar de agradecer a essas pessoas.

Começo por demonstrar o meu agradecimento ao meu supervisor, Professor Doutor Fernando Ribeiro, pela orientação e apoio.

Aos meus colegas do Laboratório de Automação e Robótica, agradeço por todos os conhecimentos compartilhados. Em especial, ao Tiago Ribeiro, que esteve sempre disponível para me ajudar.

Realço a ajuda do artista plástico, Pedro Ildo, que me auxiliou nas simulações bem como na construção das mangas. Muito obrigado pela disponibilidade.

Aos meus amigos e família, por terem sido um pilar fundamental para a minha evolução, que sempre acreditaram em mim e deram todo o amor e apoio necessário.

Por fim, um obrigado a todos os meus colegas de turma, com quem tive a oportunidade de vivenciar bons momentos.

DECLARAÇÃO DE INTEGRIDADE

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

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

RESUMO

A presente dissertação resultou da elaboração de um projeto individual, em ambiente laboratorial, no âmbito do 5º ano do Mestrado Integrado de Engenharia Eletrónica Industrial e Computadores. O objetivo primordial consistiu no desenvolvimento de um sistema de deteção de tato em tempo real, tendo como propósito a sua acoplação ao projeto CHARMIE, que corresponde a um robô colaborativo para ambientes domésticos e médicos.

Este sistema consiste numa manga com um conjunto de sensores embebidos e foi estruturado com base numa outra manga, denominada de calibração, que utiliza um algoritmo de Visão por Computador para adquirir os dados relevantes. Das duas mangas, a de calibração foi avaliada com base em dados de ambientes de simulação e reais, apresentando um ótimo desempenho em ambos. Para a manga de teste, inicialmente, foram desenvolvidos protótipos, com o propósito de analisar qual o sensor mais adequado. Posteriormente, foi construída a manga de teste.

Por fim, há a constituição de três modelos finais: dois deles utilizam a manga de teste com uma estrutura rígida, sendo que a principal distinção entre os dois reside no tipo de rede aplicada. A primeira utiliza uma ANN simples, enquanto que, a segunda reutiliza a CNN desenvolvida para a manga de calibração, através da implementação do *Transfer Learning*. O terceiro modelo emprega a manga maleável com a mesma ANN do primeiro modelo. Estes três métodos distintos são implementados, de forma a avaliar qual a arquitetura que apresenta melhores resultados.

Todos estes modelos utilizam Inteligência Artificial, mais concretamente *Deep Learning*, proporcionando uma capacidade de implementação de sistemas que aprendem com os dados fornecidos, sem haver a necessidade de programação específica. Para ambas as implementações, são utilizados algoritmos de *Supervised Learning*, que utilizam um conjunto de dados referenciados para identificar padrões e classificá-los. Para cada modelo estruturado, foram selecionados os hiperparâmetros, tais como taxa de aprendizagem, *epochs*, *batch size*, de forma detalhada, analisando e comparando os diferentes resultados.

PALAVRAS-CHAVE

Deep Learning, Deteção do Toque, *Supervised Learning*, *Transfer Learning*, Visão por Computador

Abstract

This dissertation started from the development of an individual project, in laboratory environment, in the 5th year of the Integrated Master of Industrial Electronics and Computer Engineering. The main goal was the development of a real-time touch detection system, with the purpose of coupling it to the CHARMIE project, which consists of a collaborative robot for domestic and medical environments.

This system consists of an artificial skin with a set of embedded sensors and was structured based on a sleeve, called calibration, which uses a Computer Vision algorithm to acquire the relevant data. Of the two sleeves, the calibration sleeve was evaluated against data from both simulation and real time environments, performing well in both. For the test sleeve, prototypes were initially developed in order to determine the most suitable sensor. Subsequently, the test sleeve was built.

Finally, three final models were built: two of them use the test sleeve with a rigid structure, and the main distinction between both is the type of network applied. The first uses a simple ANN and the second reuses the CNN developed for the calibration sleeve by implementing Transfer Learning. The third model employs the malleable sleeve with the same ANN as the first model. These three distinct methods are implemented in order to evaluate which architecture performs better.

All of these models use Artificial Intelligence, more precisely, Deep Learning, providing the ability to implement systems that learn from the data given, without the need for specific and rigorous programming, reducing its complexity. For both implementations, Supervised Learning algorithms are used, which use a referenced dataset to identify patterns and classify them. For each structured model the hyperparameters were selected in detail, analysing and comparing the different results.

Keywords

Deep Learning, Touch Detection, Supervised Learning, Transfer Learning, Computer Vision

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

2D – 2 Dimensions (2 Dimensões)

3D – 3 Dimensions (3 Dimensões)

ANN – Artificial Neural Network (Rede Neuronal Artificial)

CHARMIE – Collaborative Home Assistant Robot by Minho Industrial Electronics (Robô Assistente de Casa Colaborativo do Minho Indústria Eletrónica)

CNN – Convolutional Neural Network (Rede Neuronal Convolucional)

CPU – Central Processing Unit (Unidade Central de Processamento)

DDN – Depth Detection Network (Rede de Deteção de Profundidade)

FN – Falso Negativo

FP – Falso Positivo

GPU – Graphics Processing Unit (Unidade de Processamento Gráfico)

HDD – Hard Disk Drive (Unidade de Disco Rígido)

KFCV – K-Fold Cross Validation

ML – Machine Learning (Aprendizagem Máquina)

MNIST – Mixed National Institute of Standards and Technology (Instituto Nacional de Normas e Tecnologia Mixto)

RAM – Random Access Memory (Memória de Acesso Aleatório)

RGB – Red Green Blue (Vermelho Verde Azul)

SDN – Section Detection Network (Rede de Deteção da Secção)

VN – Verdadeiro Negativo

VP – Verdadeiro Positivo

1. INTRODUÇÃO

O presente capítulo visa apresentar o enquadramento do projeto, de modo a identificar o problema que serve de base para esta dissertação. Segue-se a motivação e objetivos que devem ser atingidos para solucionar o problema referido. Por fim, é apresentada a organização do presente documento.

1.1 Enquadramento e Motivação

Atualmente, a indústria utiliza robôs que substituem trabalhadores em tarefas não ergonómicas [1]. Estes robôs são escolhidos porque apresentam caraterísticas benéficas, nomeadamente a precisão, velocidade de execução, elevação de cargas pesadas, qualidade na repetibilidade e um preço inferior ao de um trabalhador humano [2]. Contudo, algumas operações necessitam de intervenção humana, sendo que os robôs são programados para executar uma tarefa específica e não têm a capacidade de se adaptarem às condições reais. Além disso, apresentam limitações no seu movimento, visto que possuem menos graus de liberdade do que o ser humano. A colaboração entre o Ser Humano e a máquina possibilita ultrapassar todas as desvantagens mencionadas anteriormente.

Esta colaboração é a nova tendência no âmbito da robótica industrial e de serviços, apresentando uma enorme importância para a indústria 4.0. No entanto, como o trabalhador opera no mesmo espaço de trabalho do robô é necessário assegurar uma interação segura de modo a não colocar em perigo os intervenientes.

Os fabricantes escolhem diferentes métodos para garantir a segurança do seu produto. Os mais usuais são os elementos ativos e passivos. Os ativos são utilizados para definir os protocolos de segurança, enquanto que os passivos são implementados quando ocorre uma colisão de forma a minimizar os danos [1].

No futuro prevê-se capacitar os robôs colaborativos para cuidarem da vida diária dos humanos, como por exemplo auxiliar os médicos na recuperação dos pacientes [3]. Ao conceber estes robôs, a interação segura é essencial porque por muito preciso que o movimento do robô possa ser não se pode garantir a inexistência de colisões. O contato físico é inevitável e deste modo é importante que a estrutura do robô não seja rígida, diminuindo o perigo inerente.

Normalmente, os robôs apresentam um conjunto de sensores para a deteção do contato/colisão, tais como sensores de distância, sensores de força e sensores de visão. Estes sensores podem prevenir e

detetar eventuais colisões. Contudo, todos estes métodos podem apresentar erros de sistema ou ruído no sinal que influenciam na segurança dos intervenientes. Assim, a implementação do toque é crucial para que sejam capazes de experienciar o meio ambiente tal como os Seres Humanos.

Posto isto, este tema fundamenta-se na criação de um sistema de deteção tátil, indicando o local onde o robô está a ser tocado bem como a profundidade da deformação. Esta implementação capacita o robô a poder sentir o contato em relação ao meio ambiente e, consequentemente, a evitar colisões, possivelmente perigosas, que poderiam não ser facilmente detetadas. A solução proposta baseia-se numa manga de silicone com sensores embebidos. Esta manga tem como vantagens a sua maleabilidade, que permite que seja fácil a sua colocação e possibilita criar deformações capazes de serem detetadas pelos sensores. Ao mesmo tempo existe a facilidade de criação de moldes utilizando impressão 3D. Para além disso, o facto de a manga ser de silicone torna ainda a estrutura do robô menos rígida.

Este modelo tem como finalidade inserir-se no projeto CHARMIE[4], que consiste num robô antropomórfico de serviços genéricos para ambientes domésticos e médicos.

1.2 Objetivos

O principal propósito desta solução é a construção de um ambiente de colaboração seguro entre os Humanos e os robôs. Numa fase inicial são criadas duas mangas diferentes para este processo. A primeira manga de silicone (manga de calibração) é montada num suporte com duas câmaras, uma por baixo e outra por cima. O interior da manga possui um conjunto de pontos brancos para que seja desenvolvido um algoritmo, utilizando Visão por Computador, para detetar a região do toque. A segunda manga (manga de teste) é em silicone com os sensores para ser aplicada nos robôs.

A segunda etapa do projeto consiste em aplicar as mesmas deformações às duas mangas, onde, utilizando um modelo de *Supervised Learning*, a informação da manga de teste será usada como entrada do sistema que é necessário classificar e a informação da manga de calibração será usada no processo que permitirá treinar a rede neuronal, tendo como principal objetivo a deteção automática do local onde o sistema está a ser pressionado bem como a profundidade do contato. Assim, não é necessário desenvolver o modelo matemático dos sensores, permitindo reduzir a complexidade do modelo.

1.3 Organização do documento

O documento é estruturado em seis capítulos. Inicia-se pelo primeiro capítulo, denominado de Introdução, onde é apresentado o enquadramento do projeto, a motivação e os objetivos. O segundo capítulo retrata a Revisão da Literatura que analisa as diferentes áreas temáticas abordadas. Além disso, contém o Estado da Arte, que referencia o estado atual de conhecimento científico acerca do tema abordado. O capítulo 3 consiste nos Métodos e Metodologias teóricas e descreve todos os processos e fundamentos matemáticos utilizados. O capítulo 4 designado por Modelo Prático introduz todos os recursos usados na prática, desde a conceção das mangas em ambiente de simulação até às respetivas estruturações e validações em tempo real. O capítulo 5 consiste no Modelo Final onde são utilizados os modelos reais otimizados e é desenvolvido o método que permite interligar o conjunto de mangas. Por último, o capítulo 6 descreve as Conclusões que menciona os resultados obtidos bem como trabalhos futuros.

Ao longo desta dissertação são utilizados termos cuja origem não advém da Língua Portuguesa, devido à falta de vocabulário e expressões que ao serem traduzidos exponham um mesmo significado. Ademais, são conceitos usualmente empregues pela comunidade científica e, por isso, não devem ser traduzidos. Isto permite que estes conceitos, termos e expressões sejam facilmente entendidos e identificados por todos, independentemente do idioma.

2. REVISÃO DA LITERATURA

O presente capítulo foca-se na Revisão da Literatura. Este é um tópico importante, na medida em que possibilita o desenvolvimento de uma boa base teórica que confere mais conhecimentos e informações relevantes para a parte prática.

Esta secção divide-se em duas etapas. A primeira centra-se na revisão e estruturação de todos os aspetos teóricos necessários para a implementação do modelo proposto. Inicia-se por uma revisão acerca da Inteligência Artificial, uma vez que se trata de um tópico que irá ser desenvolvido neste modelo. Por ser uma área científica extensa e bastante estudada, serão apenas abordados os principais temas que estão englobados, sendo eles: *o Machine Learning e o Deep Learning*. Além disso, é feita uma revisão aprofundada acerca do tema *Deep Learning* para Visão por Computador, visto que é o método utilizado para desenvolver as redes neuronais.

A segunda parte é dedicada à análise e estudo das implementações atuais, desenvolvidas pela comunidade científica, para a resolução do problema apresentado.

2.1 Inteligência Artificial, *Machine Learning* e *Deep Learning*

Nesta secção, é apresentada, de forma sucinta, a relação entre a Inteligência Artificial, *Machine Learning* e *Deep Learning.* Esta associação é estruturada com base nos conceitos essenciais para o seu desenvolvimento.

2.1.1 Inteligência Artificial

A Inteligência Artificial surgiu na década de 1950 e consiste no estudo e técnicas que permitem automatizar tarefas intelectuais normalmente realizadas por humanos [5]. É um campo que abrange áreas como *Machine Learning* e *Deep Learning*, que se encontram evidenciadas na Figura 1. Este domínio tem vindo a ser integrado em várias aplicações ao longo dos últimos anos. Exemplos concretos de empresas que integram estas ferramentas são a Apple, Facebook, Microsoft entre outros [6].

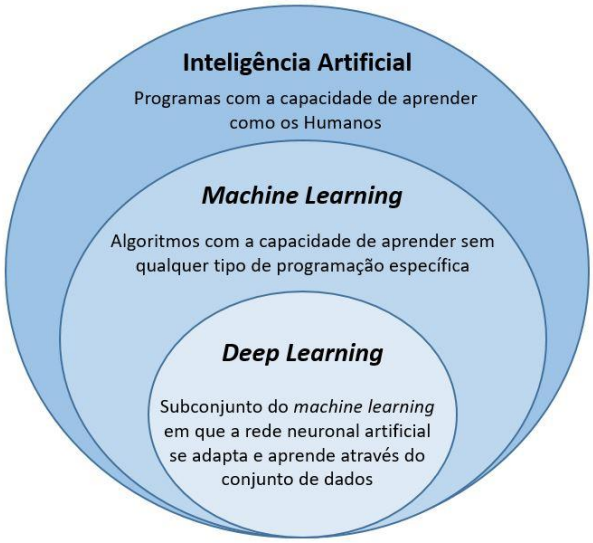


Figura 1 – Inteligência Artificial, *machine learning e deep learning*.

2.1.2 *Machine Learning*

Machine Learning é um ramo específico da Inteligência Artificial, onde são concebidos algoritmos com o intuito de fazer previsões e decisões sem qualquer programação específica para realizar a tarefa [7], através da experiência obtida pelos dados que são utilizados para treinar os algoritmos.

Todo o processo de *Machine Learning* consiste em duas fases [8]: a fase de treino e a de previsão. Na primeira fase, o algoritmo obtém todos os indicadores essenciais provenientes dos dados de treino. Estes dados são introduzidos para construir um modelo matemático que posteriormente é utilizado para fazer previsões e decisões com a entrada de novos dados que são desconhecidos pelo modelo. Quando o algoritmo de aprendizagem atinge um certo patamar na métrica de avaliação, termina o processo de treino. Por fim, decorre a fase de previsão que consiste na estimativa de novas saídas do sistema, tendo como base o modelo estruturado inicialmente.

A Figura 2 representa o fluxo de trabalho do modelo *Machine Learning* (ML).

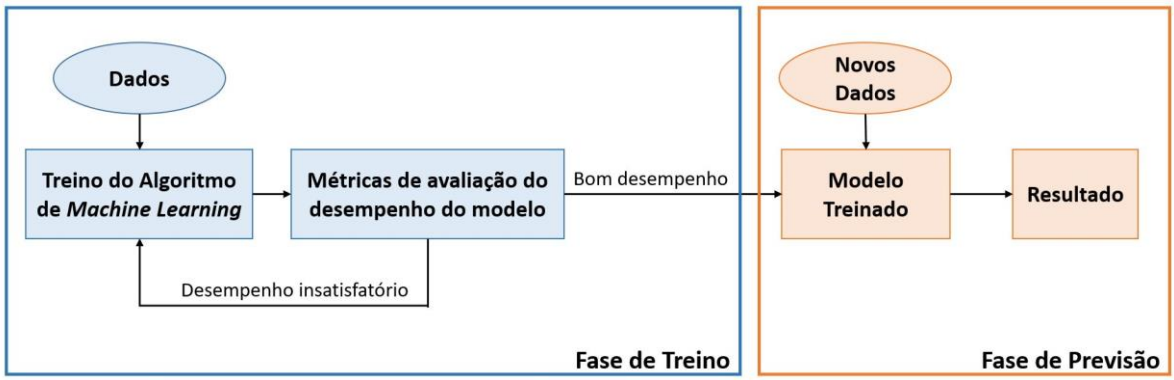


Figura 2 – Fluxo do modelo ML.

Este método cresce quando novos dados são fornecidos para a máquina aprender e é normalmente dividido em três ramos:

- *Supervised Learning*;
- *Reinforcement Learning*;
- *Unsupervised Learning*.

2.1.2.1 *Supervised Learning*

Supervised Learning [9] é um modelo que é desenvolvido para aprender através de dados cujas entradas estão emparelhadas com as devidas saídas. O principal propósito deste modelo é estabelecer uma relação entre a entrada e a saída, sendo os dados de entrada desconhecidos.

No decorrer da fase de treino o algoritmo vai obter padrões nos dados que relacionam com as saídas. Durante este procedimento os parâmetros do modelo são redefinidos, de modo a reduzir a diferença entre o valor previsto e o expectável. Esta diferença deve ser o mais próximo de zero. Após o treino, o algoritmo de aprendizagem recebe novos dados que não têm qualquer relação entre a entrada e a saída, designados de conjunto de teste. O modelo determinará qual a respetiva saída, utilizando o algoritmo desenvolvido na fase de treino.

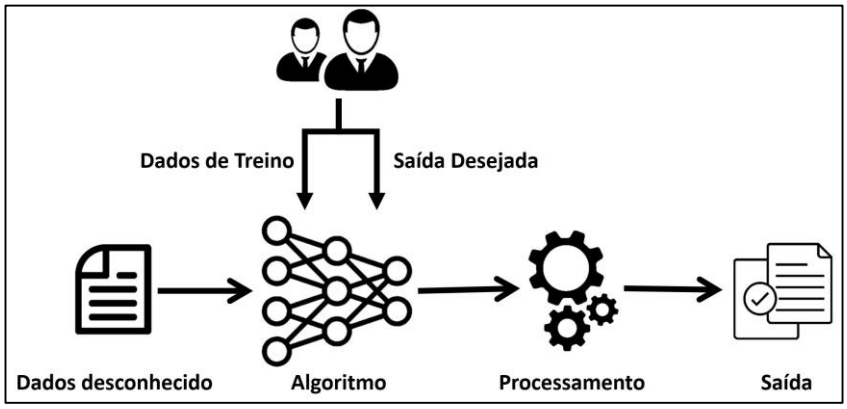


Figura 3 – Fluxo do modelo *Supervised Learning*.

Nesta aprendizagem existem duas principais subcategorias: a classificação e a regressão [10]. A primeira é utilizada para atribuir a uma entrada um número discreto de possíveis saídas, isto é, quando a saída é definida por um conjunto finito de classes. Uma implementação deste método é o algoritmo desenvolvido para classificar dígitos escritos manualmente, através da biblioteca do MNIST [11]. Esta biblioteca é usada para treinar uma rede de modo a classificar os dígitos de 0 a 9, onde este algoritmo classifica as entradas para um número finito de saída.

Por outro lado, a regressão consiste num processo estatístico preditivo em que o modelo tenta encontrar uma relação entre as variáveis dependentes e independentes. O principal objetivo é desenvolver um algoritmo que permita prever um número contínuo como determinar a nota de um teste com base no tempo de estudo, entre outros.

2.1.2.2 *Unsupervised Learning*

O ramo *Unsupervised Learning* é útil para identificar padrões ocultos em dados de entrada que são desconhecidos. Este modelo tem a capacidade de aprender e organizar as informações obtidas sem utilizar um método de comparação, uma vez que não há saídas pré-definidas e, sendo assim, não há a possibilidade de haver uma comparação entre a saída expectável e a obtida e, desta forma, não há uma medida de erro para avaliar o potencial da solução [12]. A falta de direção do algoritmo pode ser vantajosa na medida em que permite procurar padrões que não foram considerados anteriormente [13].

Este método apresenta duas principais maneiras de estudar a estrutura de dados: o *clustering* e *dimensionality reduction*. A principal diferença entre ambos é que o armazenamento em *cluster* geralmente é utilizado para revelar a estrutura de dados, enquanto que, a *dimensionality reduction* permite reduzir o número de caraterísticas de um conjunto de dados e costuma ser motivada por questões computacionais [14].

2.1.2.3 *Reinforcement Learning*

Reinforcement Learning é uma área de aprendizagem que permite mapear situações de modo a maximizar um sinal numérico de recompensa [15]. O algoritmo não é previamente informado acerca de quais ações deve tomar, tendo de descobrir quais ações geram maior recompensa. Contudo, em casos mais complexos as ações podem não afetar diretamente a recompensa, mas sim nos próximos eventos, influenciando toda a gratificação que vem a seguir. Estas duas caraterísticas, tentativa-erro e recompensa atrasada, são os dois atributos distintos mais importantes nesta implementação.

Este método distingue-se do *Supervised Learning* por não precisar de associar as entradas com as saídas antecipadamente, e por não necessitar que as ações abaixo do ideal sejam redefinidas. Em vez disso foca-se em encontrar um equilíbrio entre o conhecimento atual e o desconhecido [16].

2.1.3 *Deep Learning*

O *Deep Learning* é um subconjunto do *Machine Learning* que visa descobrir vários níveis de representações de dados utilizando arquiteturas hierárquicas [17]. As estruturas hierárquicas com um

conjunto de camadas sucessivas interligadas formam uma rede neuronal que são a base da aprendizagem.

Uma representação é um método diferente de visualização, representação ou codificação de um dado [18].

2.1.4 Redes Neuronais Artificiais

Uma Rede Neuronal Artificial (ANN) é um modelo matemático baseado na estrutura e funcionamento das redes neuronais biológicas. Tem como finalidade o processamento de informação, analisando os dados de entrada e fornecendo uma saída. O principal elemento de todas as redes neuronais artificiais são os conjuntos de nós que estão conectados entre si, de modo a haver transferência de informação. Estes nós são denominados de neurónios artificiais, por haver uma analogia com os biológicos.

Assim sendo, estas redes consistem num conjunto de neurónios denominados de entrada (*input*), várias camadas ocultas (*hidden layers*) e, por fim uma camada final designada de saída (*output*) [19].

A Figura 4 apresenta uma arquitetura típica do modelo referido.

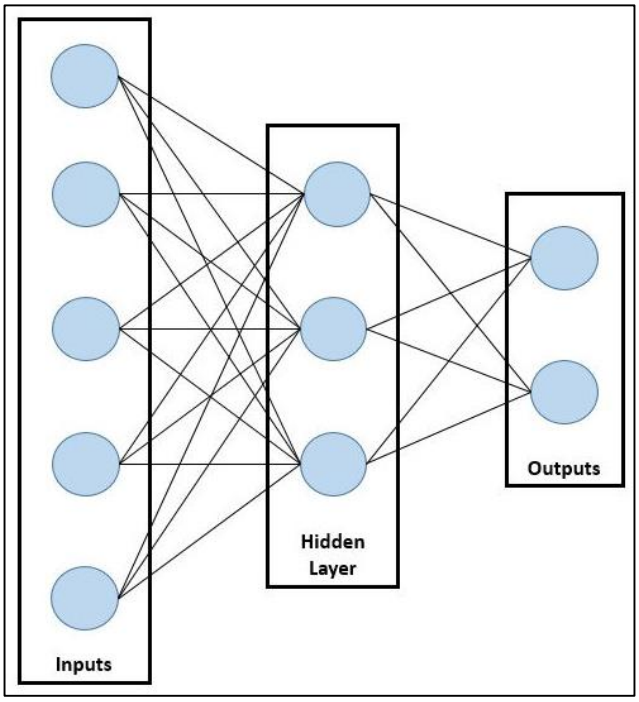


Figura 4 – Arquitetura de uma rede neuronal artificial.

Os neurónios biológicos recebem sinais através de sinapses localizadas nas suas membranas, podendo ter vários sinais de entrada. Quando os sinais recebidos são suficientemente fortes, o neurónio é ativado e emite um sinal. Este sinal pode ser enviado para outra sinapse podendo ativar outros neurónios.

A complexidade de um neurónio biológico é bastante grande. No entanto, quando se modela um neurónio artificial, este é desenvolvido de forma simplificada [20]. Cada neurónio artificial comunica através de uma ligação, tendo como princípio funcional as sinapses que ocorrem nos neurónios reais, que representam o fluxo de informação. Nestas ligações são atribuídos pesos (*weights*). Estes pesos são parâmetros reguláveis e permitem quantificar a influência que as ligações apresentam.

Inicialmente, os sinais recebidos no neurónio são multiplicados pelo respetivo peso. De seguida, é efetuada a soma de todas as multiplicações, de modo a obter a soma de todos dos sinais de entrada. Além disso, é somado um valor numérico, denominado de *bias* ou polarização, que ajuda a controlar o valor no qual a função de ativação será ativada. Após o cálculo anterior, é utilizada uma função matemática, designada por função de ativação.

O fluxo de informação de um neurónio é retratado na Figura 5.

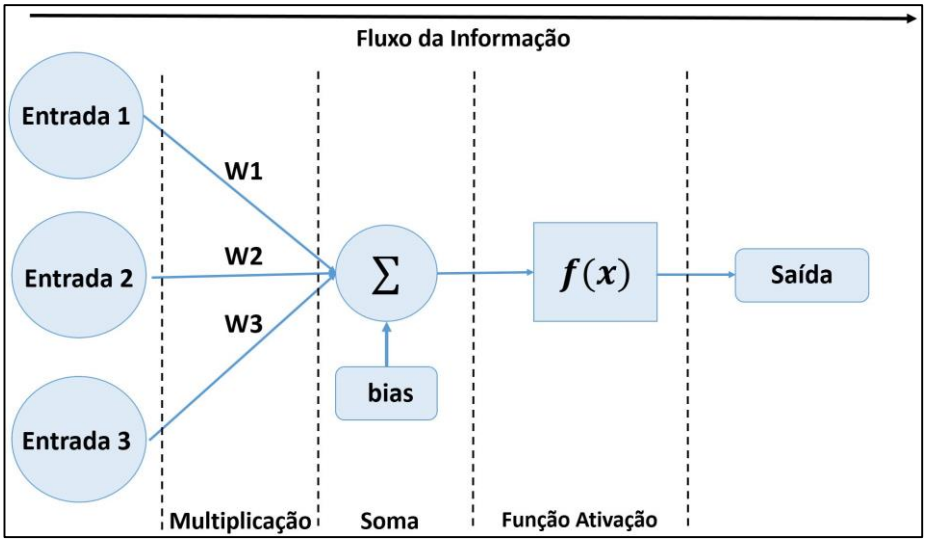


Figura 5 – Princípio de funcionamento de um neurónio artificial.

A forma como os neurónios artificiais se encontram interligados é designado por topologia ou arquitetura. Esta interligação pode ser feita de inúmeras maneiras, resultando em várias topologias possíveis, que são divididas em duas classes básicas: *feed-forward* e *feedback* [21]. A topologia *feed-forward* resulta na organização da rede, onde a informação flui apenas numa direção, da entrada para saída. Deste modo, a informação obtida na saída do neurónio não o influencia. Por outro lado, na topologia *feedback*, como o próprio nome indica, há um retorno de informação, sendo o fluxo de informação bidirecional. Esta rede é influenciada pela informação obtida na saída do neurónio.

A Figura 6 apresenta estas duas topologias.

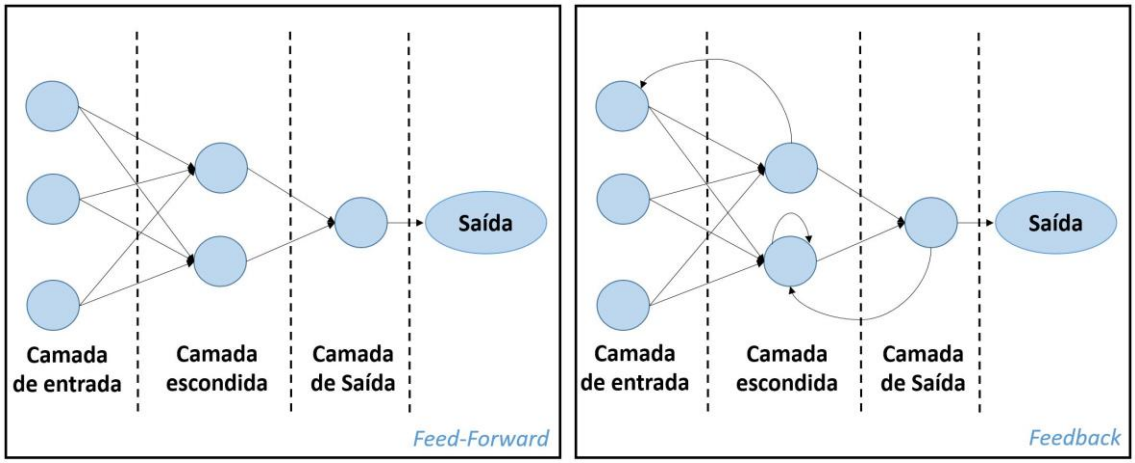


Figura 6 – Topologias básicas de redes neuronais artificiais.

Numa Rede Neuronal Artificial, a função de ativação é bastante importante, na medida em que ajuda na aprendizagem e auxilia no mapeamento não linear entre as entradas e as respetivas saídas. Esta função faz com que seja possível transformar um sinal de entrada num sinal de saída, que posteriormente poderá ser um sinal de entrada na camada seguinte. A sua implementação possibilita filtrar o sinal proveniente da soma das entradas, delimitando o valor numérico de saída de um neurónio.

Se nas redes neuronais não for utilizada a função de ativação, o modelo não tem a capacidade de reconhecer mapeamentos mais complexos, uma vez que a rede neuronal atua como um modelo de regressão linear, apresentando um desempenho limitado, na maior parte das vezes [22]. Assim sendo, a precisão da previsão de uma rede neuronal é definida pela função de ativação utilizada.

As funções não lineares são as mais usuais, uma vez que introduzem uma componente não linear, permitindo uma maior aprendizagem em relação aos métodos lineares. As principais funções de ativação não lineares [23] encontram-se na Tabela 1.

Função de Ativação	**Modelo Matemático**
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Hyperbolic Tangent (TanH)	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{j_i}}$
Rectified Linear Activation (ReLu)	$f(x_i) = max(0,x) = \begin{cases} 0 & se \quad x_i \leq 0 \\ x_i & se \quad x_i > 0 \end{cases}$

Tabela 1 – Principais funções de ativação.

Para se obter um modelo com melhor desempenho é necessário considerar o número de camadas escondidas na rede (*hidden layers*), métodos de treino, aprimoramento de parâmetros de valor aleatório (*weights* e *bias*), escolha da função de ativação, entre outros.

A função ativação é um dos parâmetros mais importantes e é escolhida com base na análise e estudo das principais funções de ativação no modelo desenvolvido. Outro aspeto importante é a configuração dos pesos, uma vez que dependendo dos pesos das conexões o cálculo da saída do neurónio é diferente. Deste modo, ajustando-os, pode-se obter a saída pretendida.

A rede neuronal utiliza um processo de treino ou aprendizagem. Este processo permite adaptar os parâmetros reguláveis (*weights* e *bias*) de uma rede neuronal [24], através de um método contínuo de simulação da rede, baseado na medição do desempenho através de uma função matemática designada por *loss function*.

2.1.5 Redes Neuronais Convolucionais

As Redes Neuronais Convolucionais exibem semelhanças em relação às ANNs comuns. Ambas são constituídas por vários nós, designados de neurónios, que apresentam melhorias ao longo da aprendizagem. A única diferença notável entre as CNNs e as ANNs é que as CNNs são habitualmente utilizadas na área de reconhecimento de padrões, na medida em que permitem codificar caraterísticas importantes das imagens na arquitetura desenvolvida. Este método torna a rede mais adequada para tarefas onde haja o processamento de imagem e reduz significativamente os parâmetros necessários para a criação do modelo.

A principal limitação das ANNs consiste no facto de os parâmetros utilizados no modelo aumentarem com o tamanho da imagem. Isto pode provocar um efeito de *overfitting*, que resulta na incapacidade de a rede aprender de forma eficiente. Sendo assim, quanto menos parâmetros forem utilizados para treinar a rede, menor é a probabilidade de ocorrer *overfitting*. Deste modo, a redução dos parâmetros possibilita melhorar o desempenho preditivo do modelo [25].

Outra desvantagem das ANNs, no problema de classificação de imagens, baseia-se na transformação da imagem fornecida em 2D para um vetor 1D, antes de iniciar o treino. Esta conversão provoca a perda das caraterísticas espaciais da imagem, sendo que esta particularidade é importante, na medida em que permite retratar a disposição dos pixéis [26].

Como a estrutura a ser implementada utiliza o tratamento de imagens, as CNNs são a arquitetura mais adequada e, desta forma, segue-se uma descrição acerca da sua organização bem como as camadas constituintes.

2.1.5.1 Arquitetura das CNN

As CNNs focam-se principalmente no processamento de imagens. Este modelo opera com dados em três dimensões, a altura (*heigh*), a largura (*width*) e a profundidade (*depth*). Esta última caraterística representa o sistema de cores RGB, onde um conjunto de pixéis correspondente a uma imagem colorida apresenta um modelo de cores aditivas em que o vermelho (*Red*), o verde (*Green*) e o azul (*Blue*) são combinados de forma a reproduzir um largo espectro cromático [27]. Assim sendo, a profundidade não se refere ao número total de camadas, mas sim ao sistema RGB.

Ao contrário das ANNs tradicionais, os neurónios da camada atual das CNNs ligam-se a uma região específica da camada anterior, como se observa na Figura 7. Esta região é denominada por *local receptive field* [25] ou campo recetivo local.

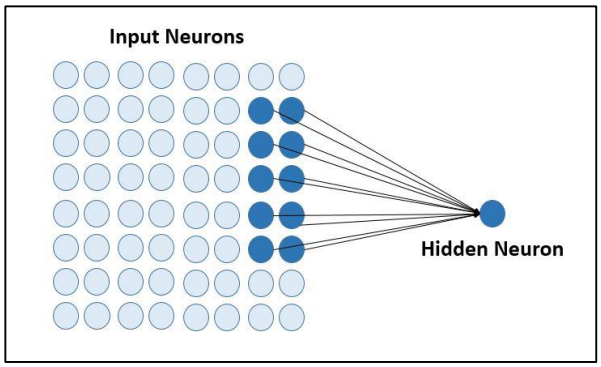


Figura 7 – *Local Receptive Field*.

A Figura 8 ilustra uma arquitetura simplificada para a classificação de imagens, que apresenta todas as camadas utilizadas. Esta configuração é baseada no exemplo presente em [25].

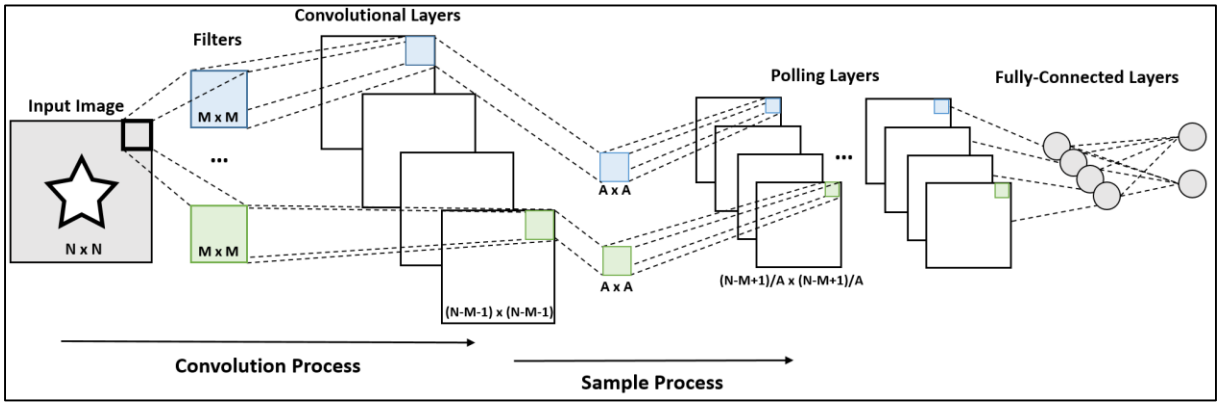


Figura 8 – Arquitetura da CNN em 2D.

As CNNs são compostas por três categorias de camadas. Estas camadas são designadas por *convolution layer*, *pooling layer* e, por fim, *fully-connected layer.*

Do exemplo, a *input image* apresenta o valor dos pixéis da imagem aplicada na entrada do modelo. De seguida, segue-se as *convolutional layers*, que determinam a saída dos neurónios que estão conectados a uma zona específica da imagem. De imediato, são exibidas as *pooling layers.* Estas camadas permitem efetuar uma amostragem decrescente, designada por *downsampling*, ao longo da dimensão espacial da *convolutinal layer,* possibilitando reduzir a quantidade de dados utilizados. Por fim, as *fully connected layers*, ou também conhecidas por *dense layers,* interligam todos os neurónios das camadas anteriores aos das camadas seguintes, permitindo obter todas as caraterísticas de aprendizagem adquiridas anteriormente. Esta camada tenta reproduzir um conjunto de classes para que, posteriormente, sejam usadas para classificar os dados.

Através destes métodos, as CNNs têm a capacidade de transformar a entrada original em várias camadas que, por intermédio de processos como o *convolution* e o *downsampling*, possibilitam otimizar o modelo e prevenir o *overfitting*. Contudo, a descrição desta arquitetura foi retratada de uma forma bastante simplificada, e como se pretende otimizar ao máximo o modelo, os conhecimentos relevantes e detalhados acerca do funcionamento de cada uma das camadas são especificados nos três subcapítulos seguintes.

2.1.5.2 *Convolutional Layer*

A *convolutional layer* consiste numa estrutura com vários filtros, ou também denominados de *kernels*, de dimensão fixa que possibilitam aplicar operações complexas às imagens presentes na entrada [28].

Cada filtro apresenta o mesmo peso (*weight*) e valor de polarização (*bias*) durante o seu deslocamento ao longo da imagem, sendo representando em toda a imagem. Estes filtros são normalmente pequenos em termos de dimensão, mas estendem-se por toda a profundidade da imagem, como demonstra a Figura 9.

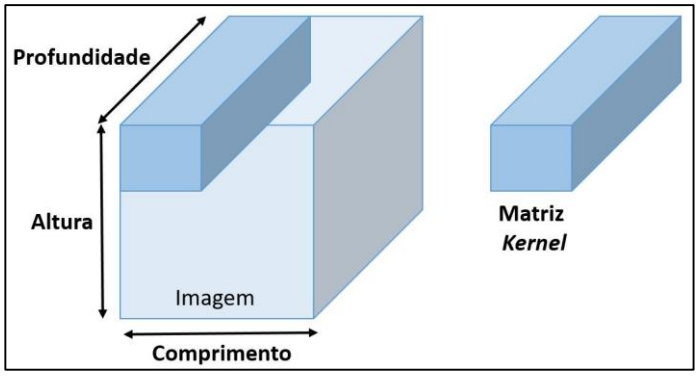


Figura 9 – *Convolution Process* em 3D.

À medida que o filtro desliza pelos pixéis da camada, é produzido um mapa de ativação ou de caraterísticas para cada posição do filtro. Estes filtros, ao longo do processo de treino, são estruturados de forma a encontrar determinadas caraterísticas presentes na entrada. Quando termina a etapa de aprendizagem, os filtros adquirem informações específicas acerca das caraterísticas analisadas, sendo ativados quando localizam esses mesmos atributos na entrada do modelo. Desta forma, as *convolutional layers* são capazes de reduzir substancialmente a complexidade do modelo, através da otimização das suas saídas [25].

Para além da utilização dos filtros, as CNNs apresentam diversas opções que permitem reduzir os parâmetros, sendo que um deles é o número de etapas que o filtro se move ao longo da camada de entrada, designado por *stride* [29].

A Figura 10 mostra um exemplo, onde a imagem de 9x9 com um *stride* de 1 passa na sua saída à dimensão de 7x7. Com o aumento da transição do filtro para 2, caraterizada na Figura 11, a saída passa a ser 4x4. No entanto, como os pixéis circundantes estão correlacionados, é necessário ter em atenção a transição selecionada, sendo que, um grande passo pode provocar a uma perda de informação.

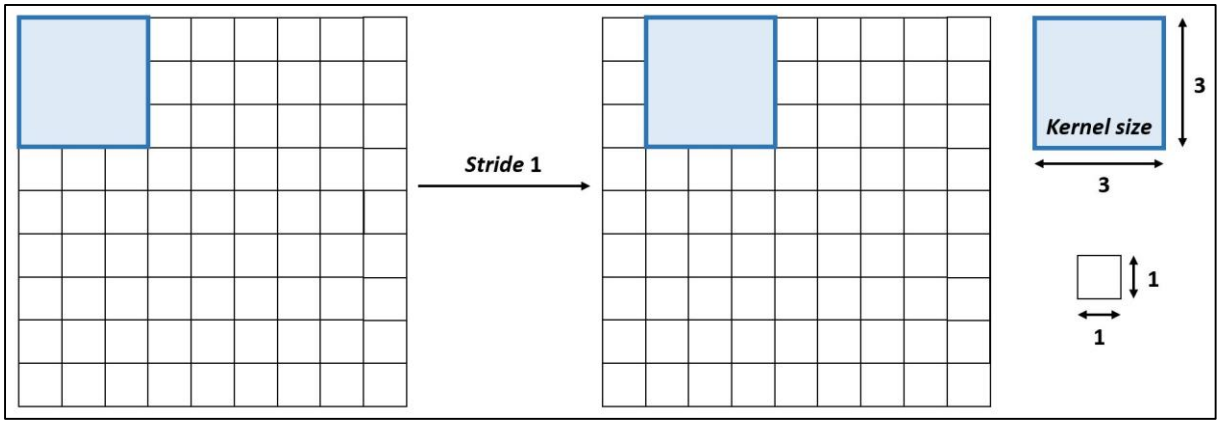


Figura 10 – *Convolution process* com *stride* 1.

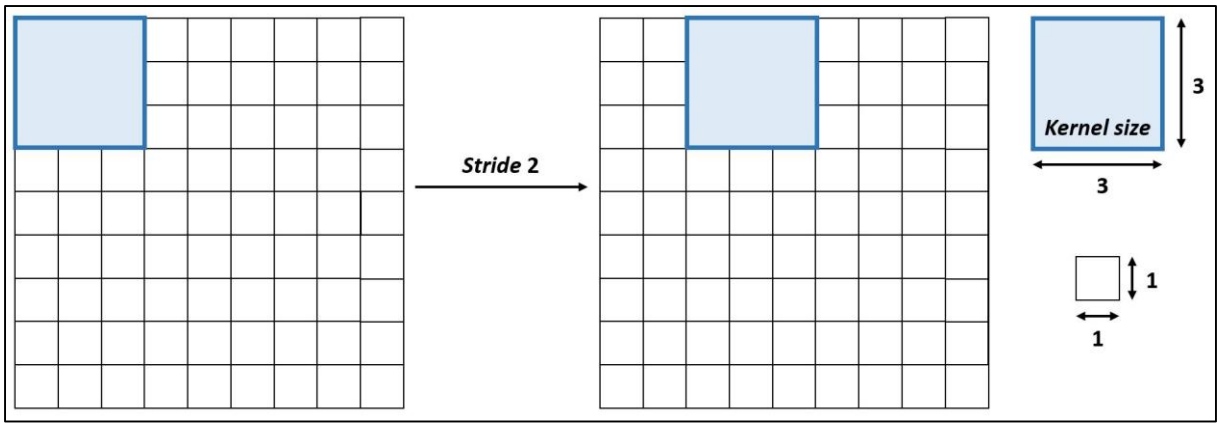


Figura 11 – *Convolution process* com *stride* 2.

Outro método é o *zero padding* [25] que constitui num processo simples de preenchimento da fronteira da camada de entrada, sendo eficaz para obter um maior controlo quanto à dimensionalidade da saída das *convolutional layers*. Este método consiste em preencher a borda final da imagem original com pixéis de valor zero, como é demonstrado na Figura 12.

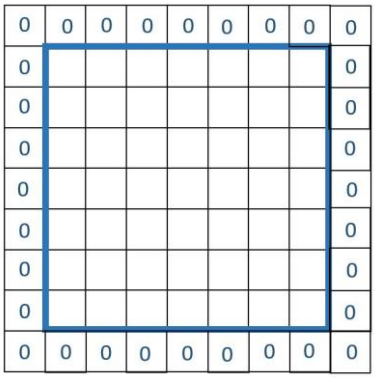


Figura 12 – *Zero padding*.

2.1.5.3 *Pooling Layer*

As *pooling layers* têm como objetivo alterar a representação do mapa de ativações que retratam as caraterísticas presentes na imagem, preservando a informação relevante, enquanto que, os detalhes insignificantes são descartados [30]. A utilização destas camadas nas CNNs destina-se a melhorar a desvantagem das *convolutional layers*, uma vez que estas camadas apresentam como limitação o facto de desenvolverem um mapa de ativação numa dada posição da imagem. Assim sendo, caso haja movimentação na localização da imagem, o modelo obtém um conjunto de mapas de ativação diferente. Estas alterações podem ser causadas por técnicas como o recorte, rotação ou deslocamento da imagem. Sendo assim, esta camada funciona em cima de cada mapa de ativação e redimensiona o seu tamanho.

Os métodos mais comuns de *pooling* são o *max pooling* e o *average pooling*, onde o primeiro obtém o valor máximo da posição atual do filtro e o segundo efetua uma média dos pixéis englobados no filtro. Estes métodos encontram-se apresentados na Figura 13 e Figura 14.

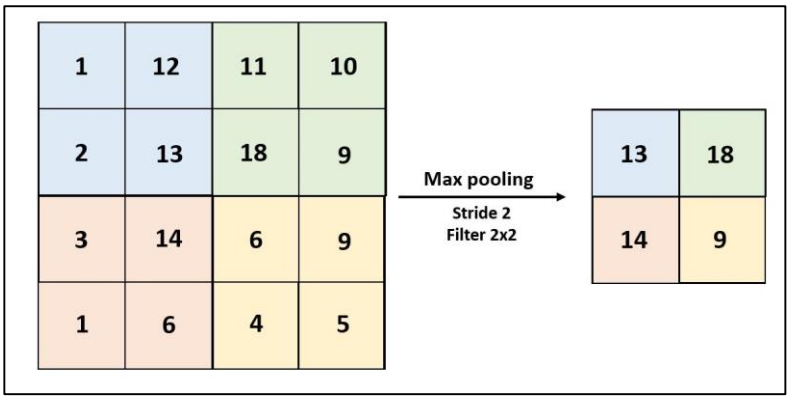


Figura 13 – *Max pooling*.

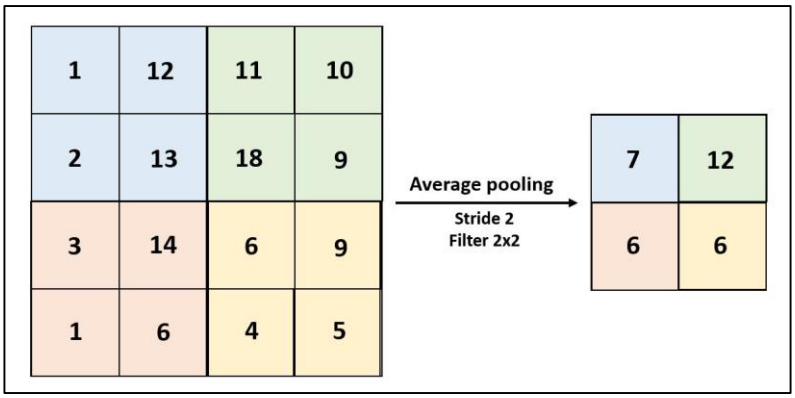


Figura 14 – *Average pooling*.

Habitualmente, as CNNS utilizam a *max pooling* com *kernels* de dimensão 2x2 aplicados com um *stride* de 2 [25].

2.1.5.4 *Fully-Connected Layer*

A *fully-connected layer* [29] é uma camada presente na arquitetura das CNN que apresenta uma disposição dos neurónios semelhante à ANN comum, onde cada nó está diretamente ligado à camada anterior e à seguinte. Sendo assim, as camadas anteriores permitem mapear os dados brutos num conjunto de caraterísticas que se encontram ocultas, enquanto que esta tem como finalidade mapear as caraterísticas adquiridas para a camada de saída [31].

2.1.6 Métricas para avaliação do desempenho do modelo

Num problema comum de classificação de dados, a métrica de avaliação é utilizada para duas fases: a de aprendizagem e a de testes. Na fase de treino a medida é usada para melhorar o algoritmo de classificação, permitindo selecionar a solução que produz as melhores previsões. Na fase de teste, esta grandeza é implementada como um avaliador, na medida em que mede a eficácia da classificação quando é testada com os dados desconhecidos [32].

Existem várias técnicas de avaliação, tais como: a *confusion matrix*, *precision*, *recall*, *accuracy*, *Mean Absolute Error* (MAE) e *Mean Squared Error* (MSE). A *confusion matrix* reúne todas as classificações reais e as previstas relativas a um problema de classificação. Esta matriz encontra-se representada na Tabela 2.

	Classe Positiva Atual	**Classe Negativa Atual**
Classe Prevista Positiva	VP	FN
Classe Prevista Negativa	FP	VN

Tabela 2 – *Confusion Matrix* .

Da tabela, é possível observar algumas nomenclaturas tais como:

- VP (Verdadeiro Positivo) que representa o número de instâncias positivas classificadas corretamente;
- VN (Verdadeiro Negativo) que retrata o número de instâncias negativas classificadas corretamente;
- FP (Falso Positivo) menciona o número de instâncias positivas que são classificadas incorretamente;
- FN (Falso Negativo) expõe o número de instâncias negativas que são classificadas incorretamente.

A partir desta matriz, podem ser geradas várias normas para avaliar o desempenho com diferentes focos de avaliação, uma vez que, apesar da *confusion matrix* fornecer informações relevantes, ocasionalmente pode não ser o suficiente.

Sendo assim, a precisão (*precision*) apresentada na equação 1, é a métrica de avaliação que pode ser usada para a classificação binária bem como a classificação de várias classes. O resultados obtido é baseado na percentagem de previsões corretas sobre o total de instâncias positivas [32].

$$Precision\ (p) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1}$$

Por outro lado, a *recall* é uma medida que possibilita selecionar instâncias positivas que são classificadas corretamente [32]. Esta métrica é definida pela equação 2.

$$Recall\ (r) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

A exatidão (*accuracy*) [32] mede a relação das precisões corretas sobre o número total de instâncias. Este método encontra-se representado na equação 3.

$$Accuracy\ (acc) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{3}$$

A *Mean Absolute Error* (MAE) [33] permite medir a magnitude da média da diferença entre o valor previsto e o obtido, como é possível observar na equação 4. Esta equação fornece a distância que a previsão se encontra da saída atual. Contudo, este processo não permite obter a direção do erro.

$$MAE = \sum_{\mathrm{k}} \frac{1}{2} |\mathrm{out_k} - \widehat{\mathrm{out_k}}| \tag{4}$$

A *Mean Squared Error* (MSE) [34] é uma métrica semelhante a MAE, tendo como diferença o facto da MSE apresentar um erro com expoente quadrático, que permite atribuir mais importância ao erro. A equação 5 demonstra a métrica retratada anteriormente.

$$MSE = \sum_{k} \frac{1}{2}(\mathrm{out}_k - \widehat{\mathrm{out}_k})^2 \tag{5}$$

2.2 Estado da Arte

O estado da arte menciona o maior grau de desenvolvimento de um dispositivo, método ou campo científico alcançado num dado período. A secção seguinte retrata os projetos estruturados que se encontram relacionados com a proposta presente nesta dissertação.

2.2.1 Trabalho Relacionado

Esta secção é destinada à análise das soluções atuais, desenvolvidas pela comunidade científica, para implementar um sistema de deteção de tato.

2.2.1.1 Sistema de deteção tátil

A deteção e localização do toque, além dos diferentes tipos de forças a que um corpo está exposto, é um requisito essencial para permitir uma interação segura robô-humano. De modo a reproduzir a interação física do Ser Humano com o meio ambiente nos robôs, existem várias pesquisas que descrevem o desenvolvimento de sistemas de deteção de toque, que podem ter várias finalidades.

Na última década, os sensores táteis eram focados essencialmente em dedos e mãos robóticas [35]. No entanto, foi estendido para todo o corpo. Várias tecnologias foram utilizadas para melhorar a capacidade de deteção de toque dos robôs, tendo como princípio operacional a resistência elétrica [36], a capacitância [37], a indução eletromagnética [38], a temperatura [39], entre outros. Estes projetos apresentam uma precisão elevada e são considerados eficazes na deteção do toque físico. Contudo, todos os dispositivos apresentados anteriormente, apresentaram dificuldades na sua produção em larga escala, uma vez que há a necessidade de acrescentar vários sensores e mais elementos eletrónicos de aquisição de dados. Como a dimensão é um fator importante, a integração destes elementos torna-se um processo bastante desafiador. Para além disto, há perigos a considerar, tais como o derrame de substâncias, curto-circuito e complicações na instalação elétrica. Sendo assim, este modelo é considerado frágil e, como apresenta bastantes contrapartidas, não é muito usual em aplicações práticas.

Outra aplicação apresentada trata-se do HEX-O-SKIN [40], que é um pequeno circuito elétrico impresso, com a forma de um hexágono, equipado com múltiplos sensores discretos de proximidade, aceleração e temperatura. Este modelo apresenta uma arquitetura de controlo para unir os dados obtidos dos diferentes módulos, possibilitando a reprodução de certas sensações humanas no robô. Contudo, consiste num protótipo em que a sua reprodução e aquisição de dados é bastante dispendiosa.

Outro método presente em [41], trata-se da conceção e fabrico de hardware de um braço e mão para interação humano-robô. O braço robótico apresenta dois módulos de sensores de força insufláveis que amortecem o impacto e fornecem um *feedback*. Apesar de se tratar de um método eficaz para a deteção do toque, apresenta uma limitação. Este modelo não permite localizar a região de contacto, uma vez que os sensores utilizados apenas fornecem a magnitude da força.

2.2.1.2 Sistema de deteção tátil baseado em Visão por Computador

A tecnologia com base em Visão por Computador é capaz de ajudar a superar todos os problemas técnicos referidos. Este método utiliza câmaras de alta resolução e de pequenas dimensões, proporcionando obter informações visuais, bem como o processamento de dados de imagens. Este método combina baixo custo, facilidade de fabricação e diminuição das ligações necessárias. Os sensores baseados em visão, normalmente, observam o comportamento de determinadas quantidades de luz para deduzir as forças aplicadas à superfície do sensor. Em [42], o objetivo foi desenvolver um sensor tátil simples e com sensibilidade, baseado na reflexão da imagem. Neste protótipo, é utilizado várias luzes e é investigada a distribuição da iluminação e as alterações devido a algumas deformações da superfície do sensor.

Outra abordagem, apresentada nas investigações [43] e [44], consiste num sensor tátil, que, posteriormente, pode ser integrado num braço de um robô. O sensor desenvolvido tem uma forma semelhante ao braço humano e é feito de silicone de modo a ser flexível. Apresenta, no seu interior, um conjunto de pontos, designados por marcadores, distribuídos regularmente. Este modelo permite estimar as deslocações detalhadas de todos os marcadores e interpretar todas as informações obtidas dos vários locais de contacto. No entanto, como contrapartida, este método utiliza duas câmaras só para a área do braço robótico. Assim sendo, caso seja aplicado em todas as partes do robô, surge a necessidade de integrar múltiplas câmaras. Esta inserção, tem como inconveniente, a possibilidade de ter de alargar o sensor, uma vez que as câmaras têm de ser inseridas no seu interior. Além disso, a utilização de várias câmaras aumentaria o custo do protótipo, podendo se tornar inviável.

3. MÉTODOS E METODOLOGIAS

Neste capítulo são apresentados os métodos utilizados para o desenvolvimento do modelo pretendido. Cada subcapítulo representa uma das mangas, onde é apresentada uma breve descrição do seu propósito, sendo esta uma etapa importante na medida em que permite analisar e conceber a arquitetura de cada uma das redes.

3.1 Manga de Calibração

A manga de calibração consiste num objeto que apresenta um conjunto de marcadores, que permite mapear a manga. É imperativo que se inicie pelo seu processo de desenvolvimento, uma vez que este recurso tem como objetivo servir de base para treinar a manga de teste.

Este subcapítulo é composto por duas etapas que descrevem o desenvolvimento deste modelo. Começa pela configuração, onde é descrito todos os detalhes da manga e, de seguida, apresenta o fluxo de trabalho, onde foram estruturados sequencialmente todos os processos para conceber o modelo.

3.1.1 Configuração da manga

A manga de calibração tem uma forma semelhante ao braço humano e é feita de silicone. Apresenta no seu interior um conjunto de pontos, designados por marcadores, distribuídos regularmente. Estes marcadores servem para rastrear o movimento da superfície, com o intuito de obter informações acerca do toque. Como é pretendido rastrear um movimento 3D são utilizadas duas câmaras. As câmaras são instaladas nas duas extremidades e não apresentam fios no interior da manga. São também instaladas luzes à sua volta de forma a iluminar a parte interna da manga. Para eliminar o possível ruído provocado pelo efeito da luz a manga é definida a preto e os marcadores a branco.

O modelo apresentado na Figura 15 baseia-se no movimento dos marcadores. A estruturação deste modelo é baseada nos sensores desenvolvidos nas pesquisas anteriores [43] e [44].

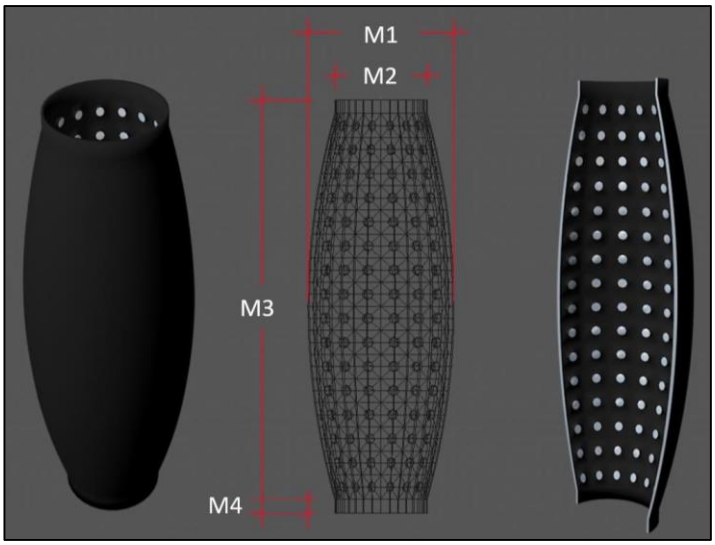


Figura 15 – Protótipo inicial da manga de calibração.

Publicação da imagem permitida pelo autor Pedro Ildo.

Da figura, observa-se quatro medidas, sendo que M1 carateriza o diâmetro do centro da manga, M2 representa o diâmetro das extremidades, M3 define o comprimento total e, por último, M4 descreve a distância entre a extremidade e a primeira linha de marcadores. Estas dimensões encontram-se descritas na Tabela 3.

Medida	**Valor Nominal (mm)**
M1	67
M2	106
M3	340
M4	20

Tabela 3 – Medidas caraterísticas da manga.

A fim de garantir que os marcadores são detetados pelas câmaras, a sua disposição foi definida para apresentar 6 milímetros de diâmetro, uma elevação da parede interna de 3 milímetros e estão distribuídos em 16 linhas, sendo que cada uma contém 16 marcadores. Como a área da manga é dividida em 64 secções cada uma engloba 4 marcadores.

A extremidade da manga apresenta 20 milímetros de comprimento e tem como finalidade permitir que seja possível fixar a estrutura num suporte.

3.1.2 Fluxo de trabalho do modelo

O modelo concebido é estruturado em cinco fases principais, que são realizadas de forma progressiva, como demonstra a Figura 16.

Figura 16 – Etapas da conceção do modelo.

A primeira etapa consiste na aquisição dos dados, que retrata a extração das imagens e das informações. De seguida, são processados os dados de modo a serem mais legíveis para constituírem um *dataset.* Os *datasets* representam um conjunto de dados, os quais se encontram classificados e permitem treinar e testar a rede. A quarta etapa estrutura o processo de treino, onde é utilizado um conjunto específico de dados para ensinar o modelo a correlacionar a entrada com as saídas desejadas. Por fim, é elaborada a avaliação do modelo, no qual são efetuados testes de forma a verificar se executa uma previsão correta.

3.1.2.1 Aquisição dos dados

Como se trata de um algoritmo de *Supervised Learning*, é necessário fornecer à rede um conjunto de dados com as respetivas identificações, sendo essencial adquirir dados que possuam informações acerca da zona de contato da manga de calibração bem como a profundidade do toque.

Na Tabela 4 encontram-se listados os dados que necessitam de ser recolhidos.

Informação Relevante	**Especificação**
Profundidade de deformação	Indica a profundidade da deformação em milímetros
Número da Secção	Indica o número da secção onde ocorreu o toque

Tabela 4 – Informações relevantes para o processo de aquisição de dados.

3.1.2.2 Processamento de dados

Segundo *Pal & Sudeep*, os dados sem qualquer tipo de processamento aplicados a uma CNN podem apresentar um desempenho insatisfatório [45]. Desta forma, surge a necessidade de desenvolver o processamento dos dados, que consiste na transformação dos dados obtidos no seu estado inicial antes de serem aplicados ao algoritmo de aprendizagem. Este processo possibilita melhorar os resultados obtidos bem como diminuir o tempo de treino, uma vez que permite reduzir o número de parâmetros utilizados.

Existem várias etapas que constituem este procedimento. Nesta secção são retratados três métodos: o redimensionamento, a conversão de escala RGB para escala de cinzentos e a normalização.

O redimensionamento carateriza-se pela alteração da resolução da imagem. Em alguns casos, as imagens inseridas apresentam resoluções altas. Contudo, o modelo não necessita deste tipo de resoluções para encontrar padrões. Além disso, a utilização de resoluções altas resulta numa aprendizagem mais lenta, uma vez que é apresentado, na entrada do modelo, um conjunto de pixéis maior. Sendo assim, o redimensionamento torna-se útil na medida em que possibilita diminuir a quantidade de pixéis presentes nas imagens sem eliminar informações relevantes.

A conversão de escala consiste na alteração das propriedades dos pixéis de forma a reduzir os parâmetros utilizados. Todas as imagens possuem três propriedades: a altura, o comprimento e a profundidade. A profundidade de uma imagem consiste no número de canais utilizados, sendo que, no caso das imagens RGB, apresentam 3 canais. As imagens em escala de cinzento, apenas apresentam 1 canal. Este canal detém a média das intensidades dos canais RGB, como carateriza a equação 6.

$$C = \frac{R + G + B}{3} \tag{6}$$

Por fim, apresenta a normalização. Este método permite que as *convolution layers,* existentes no modelo CNN, extraiam com uma maior facilidade as caraterísticas presentes nas imagens, sendo este processo dividido em duas etapas. A primeira permite reajustar os valores numéricos dos pixéis para se encontrarem entre o intervalo [0, 1], dividindo pelo seu valor máximo, que corresponde a 255, como é possível analisar pela equação 7.

$$X' = \frac{data}{\text{max\_value\_pixel}} = \frac{data}{255} \tag{7}$$

A segunda etapa, carateriza a normalização média [45] onde se utiliza a média ao longo das caraterísticas da imagem para organizar o conjunto de dados, permitindo normalizar o brilho de todos os dados fornecidos ao longo da sua dimensão. Esta técnica é evidenciada pela equação 8, onde X' simboliza os dados normalizados, *data* corresponde aos dados originais e μ descreve o vetor médio desenvolvido pelas caraterísticas dos dados.

$$X' = data - \mu \tag{8}$$

Uma forma de implementar a normalização média é calcular a média de cada imagem e subtrair pelos pixéis da própria imagem, permitindo centrar os seus pixéis em torno de 0 [46]. Outro método designado

por subtração média por píxel consiste em, para cada píxel, subtrair a sua média em todas as imagens. Cada píxel é centralizado em torno de 0 e é considerado como sendo uma caraterística diferencial [47].

3.1.2.3 Conjunto de dados

Através da aquisição e do processamento das imagens, é criado um novo conjunto de dados, denominado de *dataset*. Estes dados encontram-se devidamente classificados e preparados para serem aplicados no algoritmo de *Supervised Learning*. Contudo, é necessário dividir o *dataset* em dois subconjuntos. O primeiro é utilizado para treinar o modelo e é designado por conjunto de dados de treino. O segundo, denominado por conjunto de dados de teste, é inserido na entrada da rede, sendo que, apesar de este subconjunto se encontrar classificado, esta informação não é exposta inicialmente, tornando os dados desconhecidos. Posteriormente, o modelo irá efetuar previsões acerca deste subconjunto e o algoritmo irá comparar a informação real com a prevista. Para efetuar esta separação, é preciso assegurar que o conjunto de teste é suficientemente grande para apresentar resultados estatísticos relevantes.

Na Figura 17 é apresentada a divisão do conjunto de dados iniciais em duas categorias: os de treino (*training set*) e os de teste (*test set*). Normalmente, esta divisão utiliza 80% do conjunto de dados para o subconjunto de treino e os 20% para o subconjunto de teste.

Figura 17 – Divisão do *dataset* em subconjuntos.

3.1.2.4 Modelo

Como os objetivos do modelo consistem em classificar a área do toque e obter um valor numérico que permita saber a profundidade do toque, é preciso criar duas redes neuronais personalizadas. Para a deteção da área do toque é utilizada uma rede de aprendizagem de classificação, uma vez que se trata de uma CNN que apresenta um número discreto finito de saídas. Neste caso, a rede tem 65 nós na sua camada final, sendo que 64 correspondem às secções presentes na manga e um dos nós está associado à ausência de toque. Esta rede apenas tem de reconhecer a área do contato onde é aplicada uma força. O nó que representa a área tocada terá a sua saída a "1", enquanto que os restantes apresentarão "0".

A estrutura é denominada por *Section Detection Network* (SDN) e a Figura 18 apresenta um exemplo ilustrativo.

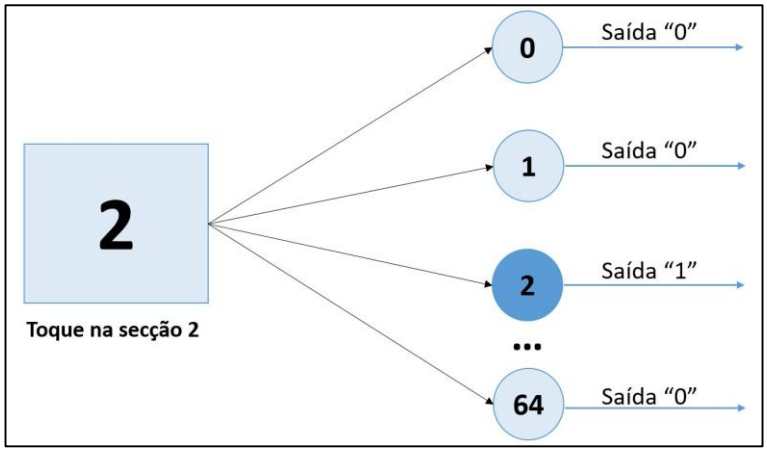


Figura 18 – *Section Detection Network*.

Para a identificação da profundidade do toque é desenvolvida uma rede neuronal designada por *Depth Detection Network* (DDN). Por se tratar de um processo de regressão, esta rede apresenta apenas um nó na camada de saída. Este nó prevê um número contínuo de valores.

A Figura 19 demonstra simplificadamente o funcionamento da rede DDN.

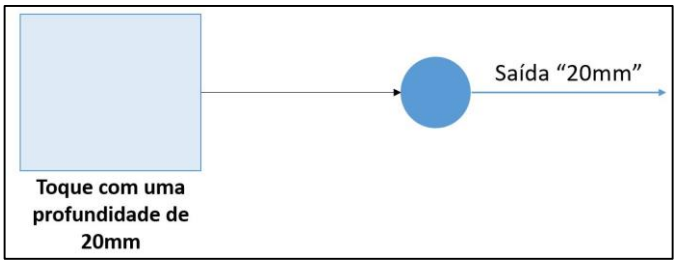


Figura 19 – *Depth Detection Network*.

3.1.2.5 Treino do modelo

As redes neuronais *feed-forward* utilizam normalmente um processo de *backpropagation learning algorithm* para treinar o modelo. Este método permite determinar os melhores valores para os *weights* e *bias* a partir dos exemplos que já se encontram classificados. Contudo, é necessário calcular previamente os erros totais. Este cálculo já foi mencionado anteriormente na Figura 4, onde demonstra o funcionamento de uma rede neuronal. No entanto, este processo foi explicado resumidamente e tornou-se necessário apresentar um conjunto de equações de forma detalhada para uma melhor compreensão. Sendo assim, começa-se pela análise do algoritmo de *feed-forward*.

Para a descrição formal dos nós, foi utilizada uma função denominada de mapeamento Γ, que atribui para cada nó um subconjunto $\Gamma(i)$. Este subconjunto consiste em todos os nós que se encontram numa

determinada camada. O subconjunto $\Gamma^{-1}(i)$ representa todas as ligações das camadas anteriores até um determinado nó i presente no subconjunto $\Gamma(i)$.

A Figura 20 representa uma ANN, onde cada neurónio é ligado a todos os nós da camada seguinte.

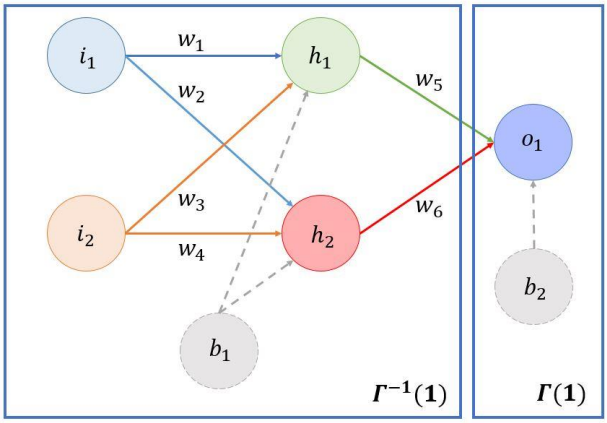


Figura 20 – *Feed foorward neural network* composta por 3 camadas.

A Figura 21 representa a ligação entre o neurónio i e o neurónio j, sendo caraterizada por *weight coeficiente* (w_{ij}) e pelo *bias coefficient* (b_i).

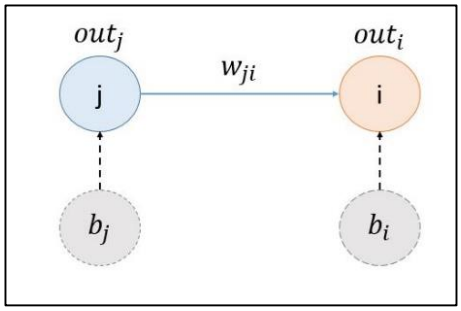


Figura 21 – Ligação entre 2 neurónios.

Para determinar a saída de cada nó é utilizado o sistema de equações 9. Neste caso particular, trata-se do neurónio i.

$$\begin{cases} out_i = f(net_i) \\ net_i = b_i + \sum_{j \in \Gamma_i^{-1}} w_{ji} \times out_j \end{cases} \quad (9)$$

A variável net_i carateriza o potencial do neurónio e out_i simboliza a saída do nó. Para obter esta saída, é essencial utilizar um modelo matemático denominado de função transferência ou ativação ($f(net_i)$). Normalmente, na *input layer* e nas *hidden layers*, a função implementada é a *Rectified Linear Activation* (ReLu), enquanto que na *output layer* é a função *Softmax*. Ambas as equações são descritas na Tabela 1, presente no Capítulo 2.

O processo *forward* permite calcular o erro total, sendo que o processo de adaptação de um modelo de *Supervised Learning* varia os coeficientes de *weights* e *bias*, de forma a minimizar o erro obtido. Para calcular o erro, utiliza-se a métrica de avaliação MSE, que permite adquirir a média da diferença entre o valor desejado e o obtido de todos os neurónios presentes na *output layer*. As variáveis out_k e $\widehat{out_k}$ correspondem, respetivamente, aos vetores das saídas pretendidas e das previstas. A *loss function* calcula o erro de um único treino, encontrando-se definida na equação 10.

$$Loss = MSE = \sum_k \frac{1}{2}(out_k - \widehat{out_k})^2 \tag{10}$$

O cálculo da média do erro total consiste na média da *loss function* para todos os exemplos de treino. Esta equação, descrita em 11, é designada por *cost function* e permite avaliar a performance do modelo a fazer previsões.

$$Cost = \frac{1}{N}\sum_k^N \frac{1}{2}(out_k - \widehat{out_k})^2 \tag{11}$$

Após o cálculo da média do erro total, decorre o processo de *backpropagation*. Este método possibilita conhecer o impacto de uma mudança dos coeficientes reguláveis, *weights* e *bias*, em relação ao erro total, sendo calculado através das derivadas $\frac{\partial E}{\partial w_{ji}}$ e $\frac{\partial E}{\partial b_i}$.

No algoritmo de *backpropagation* é usual utilizar um método designado por *steepest-descent minimisation*, também conhecido por gradiente descendente [48]. Esta estratégia consiste num algoritmo de otimização iterativa que permite encontrar o valor mínimo de uma função. O objetivo é efetuar passos na direção oposta ao gradiente em função do ponto atual, uma vez que o gradiente fornece a direção do maior valor.

A equação 12 representa o algoritmo do gradiente descendente.

$$\theta^{k+1} = \theta^k - \alpha \nabla J(\theta) \tag{12}$$

Da equação, θ^k retrata a posição atual, θ^{k+1} carateriza a posição seguinte, α descreve a taxa de aprendizagem, ∇ indica a direção do maior valor da função e J define o jacobiano, que consiste numa matriz formada pelas derivadas parciais de primeira ordem da função vetorial. Este algoritmo utiliza as derivadas devido ao facto de permitir obter a inclinação do gráfico num ponto específico. A inclinação é referenciada através de uma linha desenhada tangencialmente ao ponto, sendo importante para conhecer qual a direção do movimento de forma a diminuir o erro obtido na próxima iteração.

A Figura 22 representa um exemplo ilustrativo do gradiente descendente, onde se observa o retrato da tangente no ponto azul e no ponto verde. O ponto vermelho retrata o valor mínimo da função, sendo que o modelo tem a informação de que o local mínimo ideal de uma função de ordem superior a 2 apresenta uma inclinação de 0. O gradiente encontra-se inicialmente na posição azul.

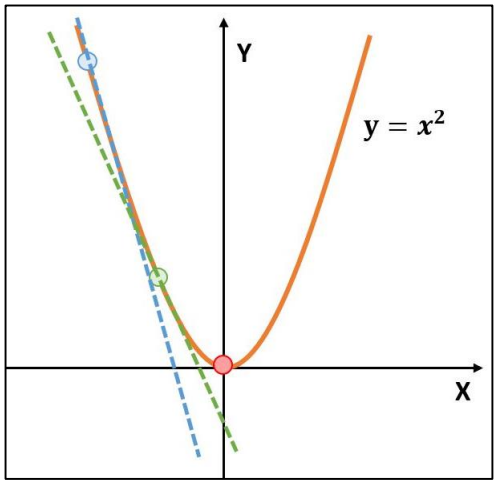


Figura 22 – Representação do gradiente descendente com as tangentes em cada um dos pontos.

Da análise do gráfico é possível observar que a inclinação da tangente no ponto azul é maior do que a do ponto verde. Com isto, o modelo tem conhecimento que necessita de transitar em direção ao ponto verde. Quando o algoritmo transita para a posição verde, encontra-se mais próximo do valor mínimo e movimenta-se lentamente de forma a não ultrapassar a região mínima.

As equações 13 e 14 adaptam o algoritmo do gradiente descendente geral para o modelo desenvolvido anteriormente.

$$w_{ji}^{(k+1)} = w_{ji}^{(k)} - \lambda \left(\frac{\partial E}{\partial w_{ji}} \right)^{k} \tag{13}$$

$$b_{i}^{(k+1)} = b_{i}^{(k)} - \lambda \left(\frac{\partial E}{\partial b_{i}} \right)^{k} \tag{14}$$

Das equações anteriores, λ corresponde ao *learning rate* ($\lambda > 0$) ou taxa de aprendizagem. Este é um parâmetro regulável que controla a resposta do modelo ao erro estimado, cada vez que os *weights* e *bias* são atualizados. Escolher esta taxa é um processo desafiador, uma vez que, caso o valor seja demasiado pequeno é provável resultar num longo processo de treino e o modelo pode mesmo não chegar ao valor mínimo. Caso o parâmetro apresente um valor demasiado alto a aprendizagem é capaz de ser demasiado rápida, podendo resultar num processo de treino instável.

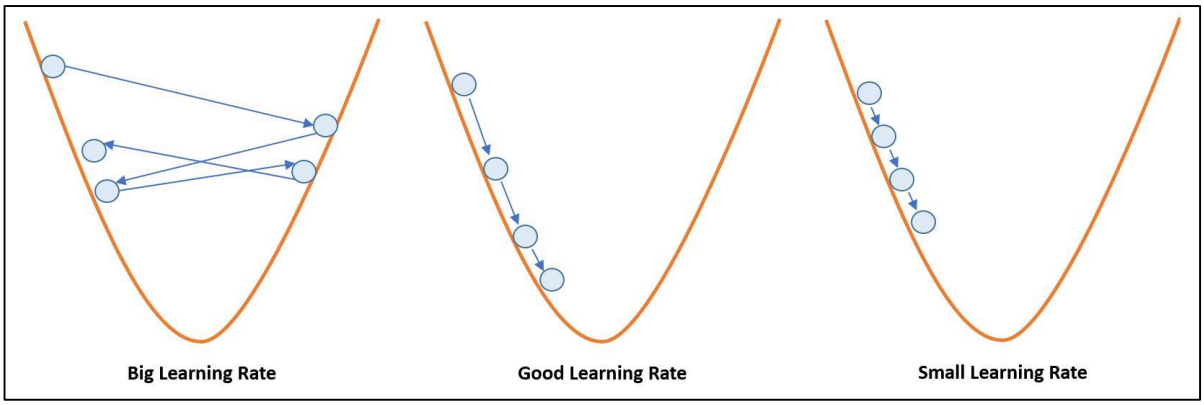


Figura 23 – Representação da transição de estado do gradiente descendente com base no *learning rate*.

Um bom método de certificar que o gradiente descendente está a decorrer de forma correta, é observar o gráfico da *loss function* em função do número de *epochs*. Isto auxilia a avaliar a performance do modelo, uma vez que após cada *epoch* é possível detetar se a taxa de aprendizagem é adequada. A Figura 24 demonstra a diferença entre taxas de aprendizagem adequadas e inadequadas.

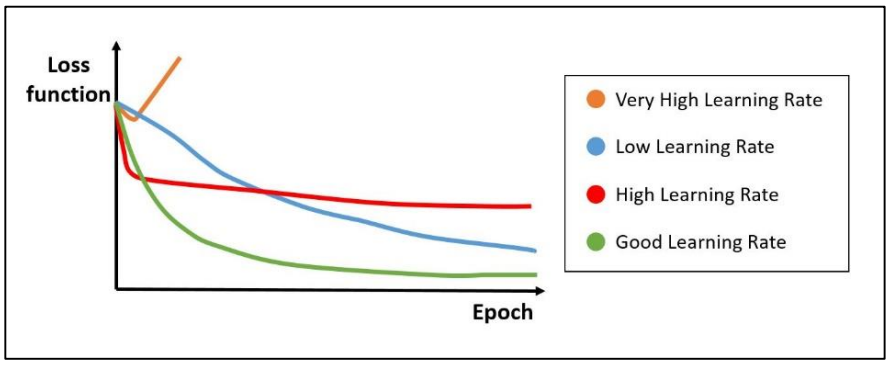


Figura 24 – *Loss function* tendo com diferentes valores de *learning rate*.

Na Figura 24, o exemplo identificado pela cor laranja apresenta uma elevada taxa de aprendizagem e é considerado um modelo com desempenho fraco. Isto porque a *loss function* analisa a diferença entre o valor obtido e o desejado. Como essa diferença aumenta ao longo das *epochs*, significa que o valor desejado e o obtido se encontram cada vez mais distantes.

No exemplo a vermelho o modelo apresenta uma taxa de aprendizagem alta, uma vez que o gradiente atinge um valor numérico que considera ser o mínimo. No entanto o valor obtido não é o menor porque a *loss function* tornou-se constante, mas não convergiu para zero ao longo das *epochs*.

O gráfico azul e verde converge para zero. Contudo, o azul apresenta uma convergência mais lenta. Isto deve-se ao facto de exibir uma taxa de aprendizagem baixa, que efetua passos demasiados pequenos e, consequentemente, demora mais tempo a atingir o seu propósito. No caso do gráfico verde, a taxa de aprendizagem é apropriada, sendo as transições efetuadas mais longas. Quando o valor se encontra

próximo do mínimo, estes passos vão diminuindo a longitude. Sendo assim, o *backpropagation* inicia-se pelo cálculo do erro presente na equação 15, onde $g_k = out_k - \widehat{out_k}$, sendo $k \in output\ layer$.

$$E = \sum_k \frac{1}{2}(out_k - \widehat{out_k})^2 = \sum_k \frac{1}{2}(g_k)^2 \tag{15}$$

Inicialmente, é apresentado um exemplo de forma a demonstrar todas as etapas do cálculo para avaliar o impacto dos pesos (*weights*).

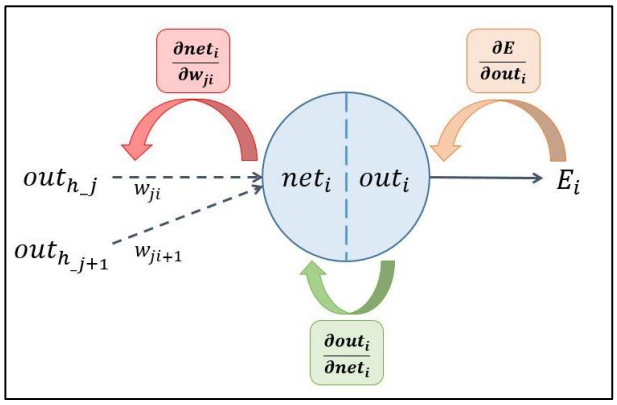


Figura 25 – Impacto dos *weights* no erro total.

A derivada $\frac{\partial E}{\partial w_{ji}}$, que se encontra apresentada na equação 13, carateriza o impacto dos pesos no erro obtido e consiste na multiplicação de todas as etapas referida na Figura 25. Esta derivada é desenvolvida de forma a simplificar o máximo possível.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial out_i} \times \frac{\partial out_i}{\partial net_i} \times \frac{\partial net_i}{\partial w_{ji}}$$

$$\Leftrightarrow \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial out_i} \times \frac{\partial f(net_i)}{\partial net_i} \times \frac{\partial net_i}{\partial w_{ji}}$$

$$\Leftrightarrow \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial out_i} \times f'(net_i) \times \frac{\partial \left(b_i + \sum_{j \in \Gamma_i^{-1}} w_{ji} \times out_j\right)}{\partial w_{ji}}$$

$$\Leftrightarrow \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial out_i} \times f'(net_i) \times out_j \tag{16}$$

Após o avanço no impacto do peso, sucede-se o efeito do *bias* no erro total, sendo que todas as etapas são simbolizadas na Figura 26.

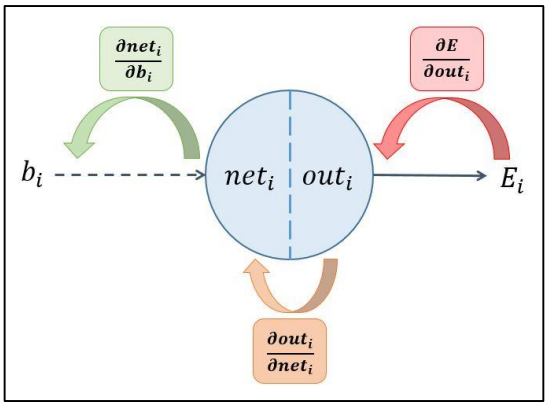


Figura 26 – Efeito do *bias* no erro total.

Sendo assim, realiza-se o desenvolvimento da derivada $\frac{\partial E}{\partial b_i}$, que representa a repercussão do *bias* com base no erro obtido.

$$\frac{\partial E}{\partial b_i} = \frac{\partial E}{\partial out_i} \times \frac{\partial out_i}{\partial net_i} \times \frac{\partial net_i}{\partial b_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial b_i} = \frac{\partial E}{\partial out_i} \times \frac{\partial f(net_i)}{\partial net_i} \times \frac{\partial net_i}{\partial b_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial b_i} = \frac{\partial E}{\partial out_i} \times f'(net_i) \times \frac{\partial (b_i)}{\partial b_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial b_i} = \frac{\partial E}{\partial out_i} \times f'(net_i) \times 1 \quad (17)$$

Através da equação 16 e 17, é possível realizar a seguinte correlação.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial b_i} \times out_j \quad (18)$$

Sendo assim, para o cálculo das derivadas, apenas é suficiente calcular $\frac{\partial E}{\partial b_i}$.

$$\frac{\partial E}{\partial out_i} = g_i \quad i \in output\ layer$$

$$\frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial out_j} \times \frac{\partial out_j}{\partial out_i} \quad i \in hidden\ layer$$

$$\Leftrightarrow \frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial out_j} \times \frac{\partial f(net_j)}{\partial out_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial out_j} \times \frac{\partial f(net_j)}{\partial net_i} \times \frac{\partial net_i}{\partial out_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial out_j} \times f'(net_i) \times \frac{\partial \left(b_i + \sum_{j \in \Gamma_i^{-1}} w_{ji} \times out_j \right)}{\partial out_i}$$

$$\Leftrightarrow \frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial out_j} \times f'(net_i) \times w_{ji} \tag{19}$$

Baseado na equação 17 e 19, chega-se à seguinte equação:

$$\Leftrightarrow \frac{\partial E}{\partial out_i} = \sum_{j \in T_i} \frac{\partial E}{\partial b_i} \times w_{ji} \tag{20}$$

Com base na abordagem desenvolvida, observa-se que é possível calcular, recorrentemente, as derivadas provenientes da *ouput layer* e das *hidden layers*, permitindo reajustar o *weights* e *bias*, de forma a reduzir o erro. Este procedimento propaga-se a partir da *output layer* até à *input layer*.

O processo de treino inicia com valores arbitrários nos *weights* e *bias* e processa-se de forma iterativa, onde cada iteração corresponde a uma *epoch*. Em cada *epoch*, a rede neuronal utiliza o processo *forward* e o *backpropagation*. À medida que o processo iterativo segue, os *weights* convergem gradualmente para o conjunto de valores ideais, onde são necessárias várias *epochs* para terminar o treino da rede. Contudo, é essencial ter algumas precauções acerca do número de *epochs* utilizadas, uma vez que o principal objetivo de um algoritmo de *Machine Learning* é desenvolver uma estrutura generalizada, de forma a aprender conceitos gerais a partir de exemplos introduzidos no modelo. Caso haja um défice ou excesso de aprendizagem, o modelo fica com um mau desempenho. Quando o processo de treino apresenta demasiadas iterações, a rede neuronal memoriza os dados de treino e apresenta resultados perfeitos para os dados utilizados. No entanto, falha para os dados de testes. Esta caraterística é denominada de *overfitting*. Caso não apresente iterações suficientes com os dados de treino, o modelo irá evidenciar uma baixa generalização e uma previsão pouco fiável. Este caso é designado por *underfitting*.

A Figura 27 representa o desempenho da rede neuronal para os conjuntos de dados de treino.

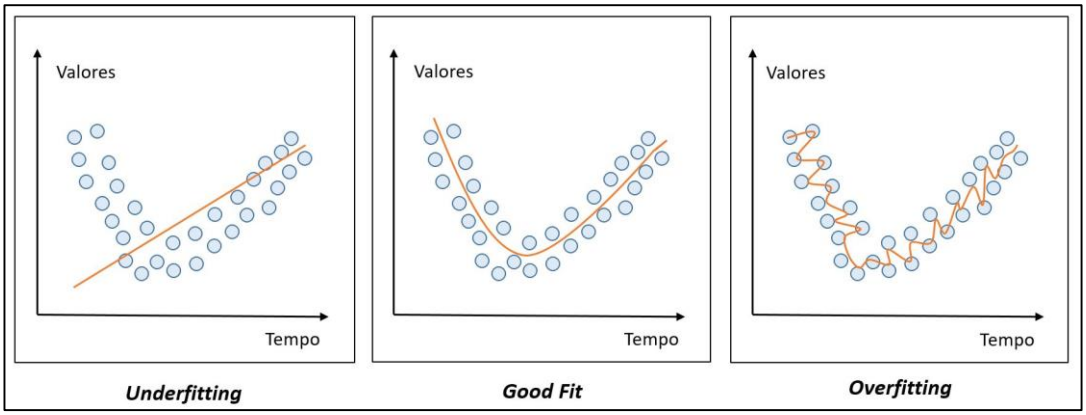


Figura 27 – Desempenho da rede neuronal.

Para uma boa generalização é importante que o modelo apresente um conjunto de dados de treino suficientemente grande para treinar o modelo adequadamente. Este conjunto de dados deve apresentar todos os casos possíveis que se pretende generalizar.

Há dois tipos de generalização [49]: a interpolação e a extrapolação. A interpolação aplica-se a todos os casos que estão próximos de eventos que foram treinados. Em casos que estão fora do âmbito dos dados que foram utilizados para treinar o modelo, utiliza-se a extrapolação. A principal diferença entre estes dois tipos deve-se ao facto de a interpolação ser avaliada como fiável, mas a extrapolação ser considerada pouco confiável. Assim sendo, é possível verificar a importância de ter um número suficiente de dados de treino de forma a evitar a necessidade de extrapolação.

Como o conjunto de dados de treino para o modelo desenvolvido não é suficientemente grandes, torna-se necessário adotar um método que permita utilizar de forma eficiente os dados recolhidos. O método utilizado é o *K-Fold Cross-Validation* (KFCV) [50]. Este procedimento consiste em particionar o conjunto total de dados em k subconjuntos diferentes. Os subconjuntos (k-1) são utilizados para treinar a rede, enquanto que o restante é utilizado para validar o treino. Este processo é efetuado k vezes, alterando de forma sequencial o subconjunto de teste.

A Figura 28 apresenta o processo realizado pelo KFCV.

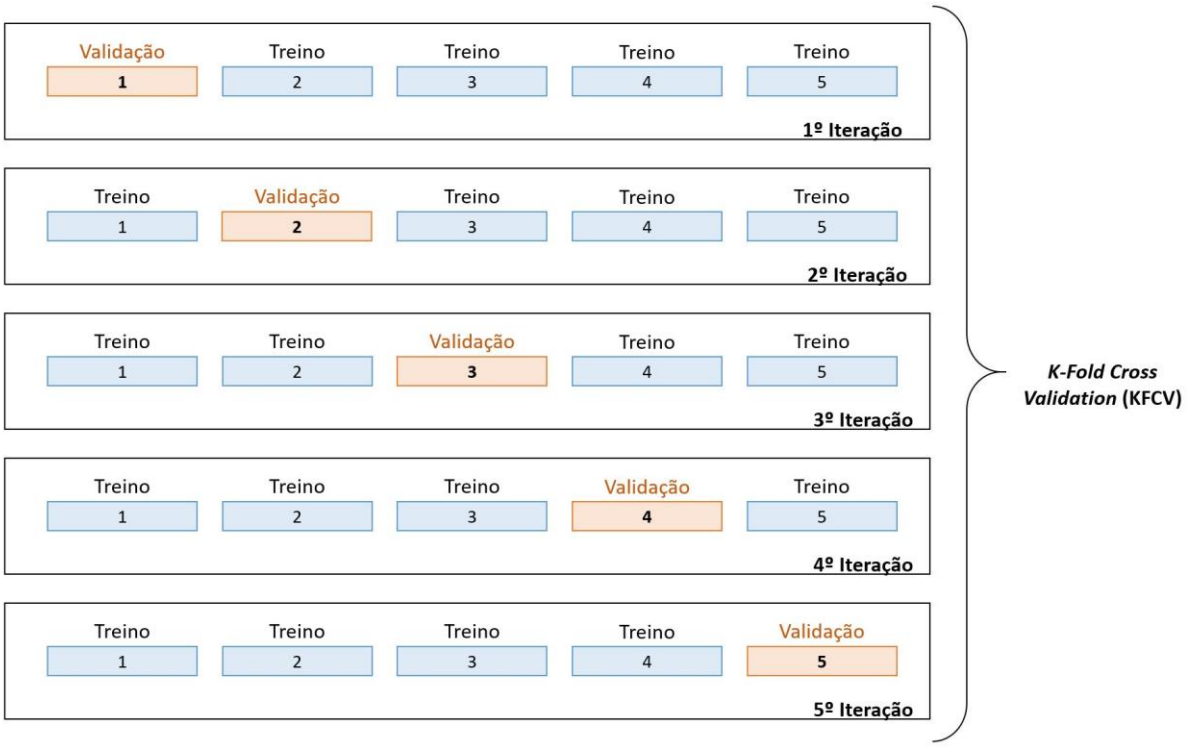


Figura 28 – *K-Fold Cross Validation* com k=5.

Ao fim das k iterações calcula-se a média das *accuracy* obtidas por todos os erros calculados através da equação descrita em 21. Esta ferramenta permite obter uma avaliação métrica mais confiável sobre o desempenho do modelo.

$$Accuracy_{K-Fold} = \frac{1}{K}\sum_{i=1}^{K}(out_i - \widehat{out}_i) \tag{21}$$

No entanto, o KFCV apresenta uma desvantagem. A divisão dos dados em subconjuntos é efetuada de forma aleatória, podendo provocar um desequilíbrio na divisão das classes e, consequentemente, prejudicar a avaliação do modelo.

Sendo assim, é necessário utilizar uma extensão do método anterior, denominado por *Stratified K-Fold Cross Validation*. Esta ferramenta possibilita dividir o conjunto de dados de forma homogénea, sendo que o principal propósito é obter vários subgrupos de dados (*folds*) que representem todas as classes de modo a que a distribuição média de cada classe nos diferentes grupos seja aproximadamente igual [51].

A Figura 29 representa a divisão homogénea de cada classe. O conjunto de dados inicial apresenta 15 classes, onde 10 correspondem à classe 1 e as 5 restantes retratam a classe 2.

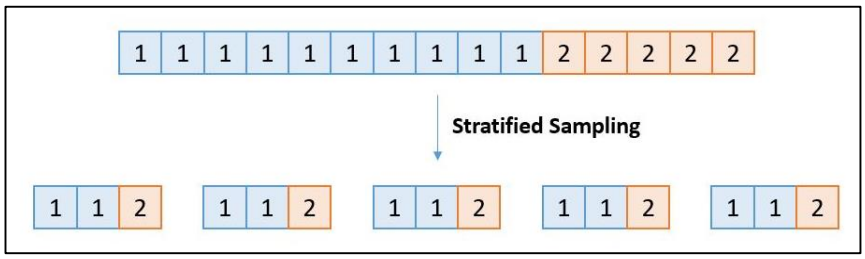


Figura 29 – *Stratified sampling*.

Como se pode constatar, o rácio da classe 2 em relação à classe 1 é de 1/3. O método *K-Fold* não considera este rácio e divide as classes aleatoriamente. Esta ferramenta pode eventualmente estruturar subconjuntos com apenas uma classe, possibilitando apresentar um treino desadequado. No caso do *Stratified K-Fold*, a divisão ocorre tendo em consideração o rácio das classes, sendo que todos os subconjuntos estruturados (*folds*) apresentam a mesma distribuição. Desta forma, os subconjuntos entre si não apresentam desequilíbrios.

3.1.2.6 Teste do modelo

Nesta secção, é avaliada a atuação dos modelos treinados e validados no KFCV com a adição do *Stratified sampling*. O conjunto de dados inicial foi anteriormente dividido em dois subconjuntos: dados de treino e de teste. Os dados de treino permitiram ajustar os parâmetros reguláveis, enquanto que os de testes foram usados para efetuar previsões de dados que são desconhecidos pelo modelo. A partir dos dados

de teste, observa-se os resultados e efetua-se uma comparação qualitativa entre os modelos desenvolvidos de forma a selecionar o que apresenta melhores resultados. A métrica mais comum da avaliação do desempenho com a utilização dos dados de teste é a *accuracy*.

3.2 Manga de Teste

Este subcapítulo apresenta todo os métodos estruturados para serem aplicados na manga de teste. O protótipo a ser desenvolvido é delineado para possuir um conjunto de sensores embebidos num material com propriedades maleáveis, onde é necessário adquirir constantemente os valores numéricos presentes em cada um deles de forma a que o modelo tenha a perceção da interação da manga com o meio ambiente e avalie em tempo real. Sendo assim, surge a necessidade de adicionar um processo de aquisição de dados.

A implementação de um microcontrolador com entradas ADC é ideal para solucionar a imposição anterior, uma vez que é capaz de efetuar uma leitura regular das entradas. Além de adquirir os dados dos sensores, este dispositivo envia-os através do protocolo UART para o computador que é detentor do modelo. Este protocolo consiste num processo de comunicação feito por dois equipamentos, onde é realizado de forma assíncrona [52]. Neste tipo de comunicação é essencial definir alguns parâmetros tais como: a taxa de transmissão, que carateriza a velocidade de transmissão dos dados, o tamanho do pacote de dados, que retrata o número de bits transmitidos e a paridade, que é utilizada para a deteção de erros na transmissão.

A Figura 30 retrata o circuito esquemático, onde são analisados as diferentes interações entre os componentes bem como o tipo de comunicação.

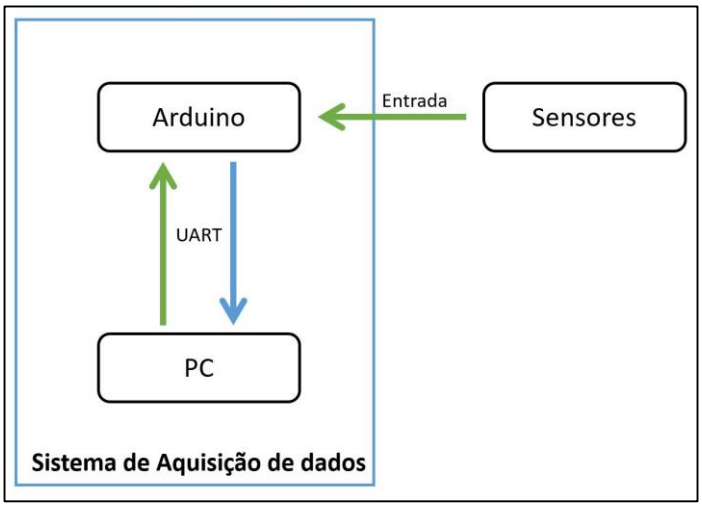


Figura 30 – Circuito esquemático do protótipo da manga de teste.

Na Figura 30 verifica-se a comunicação entre o Arduino e o sensor, onde o sensor apenas transmite os dados obtidos. Na comunicação entre o computador e o Arduino existe uma ligação USB que permite a transmissão/receção entre os dois dispositivos, uma vez que o Arduino transmite um conjunto de dados, mas aguarda a confirmação da receção com sucesso por parte do computador. Após esta sinalização, o Arduino volta a enviar um novo conjunto de dados. Todo este processo é repetido indefinidamente.

A configuração da manga de teste é estruturada com base na manga de calibração, apresentando as mesmas dimensões. A única distinção reside na adição dos sensores que são estrategicamente colocados de forma a possibilitar a deteção em toda a sua superfície. Com esta adição, a estrutura pode sofrer uma alteração na sua espessura. No entanto, todas estas especificações são definidas após a validação do protótipo, que é desenvolvido para avaliar os vários tipos de sensores.

4. MODELO PRÁTICO: DESENVOLVIMENTO

Este capítulo apresenta todos os algoritmos de aprendizagem concebidos. Inicialmente, o modelo de calibração é estruturado com base em imagens fornecidas por um ambiente de simulação. Este ambiente possibilita imitar o funcionamento do sistema que se pretende desenvolver. Logo após o fim deste processo e a conclusão da construção da manga, decorre a utilização dos dados reais nos modelos desenvolvidos anteriormente. Em paralelo aos modelos anteriores, encontra-se a conceção do modelo da manga de teste, que não necessita de ser simulado.

Todas as simulações foram realizadas num computador portátil Apple Macbook Pro, que apresenta uma 2,2 GHz Intel Core i7 como Unidade Central de Processamento (CPU). A unidade de processamento gráfico (GPU) é uma Intel Iris Pro 1536MB e possui uma unidade de disco rígido (HDD) de 255GB bem como uma memória de acesso aleatório (RAM) de 16GB 1600MHz DDR3.

Os modelos de simulação da manga de calibração são estruturados no *Cinema 4D*. Este software consiste numa ferramenta profissional de modelação e simulação 3D, onde o seu conjunto de processos torna os fluxos de trabalho mais eficientes [53].

O desenvolvimento das redes neuronais utiliza o *Pycharm*. Este programa consiste num ambiente de desenvolvimento integrado (IDE) que fornece uma vasta gama de ferramentas para os programadores de *Python* criarem um ambiente prático e produtivo [54]. As principais bibliotecas utilizadas neste IDE são o *Tensorflow* e o *OpenCV*. O *Tensorflow* é uma plataforma de código aberto para *Machine Learning* que apresenta um ecossistema abrangente de recursos, permitindo aos desenvolvedores conceberem redes neuronais artificiais [55]. Esta estrutura apresenta funções que permitem configurar a rede bem como implementar o processo de treino, validação e teste. O *OpenCV* é uma biblioteca que contém os métodos que contribuem para o processamento de imagens, desde a sua leitura, a aquisição das informações relevantes, a execução de um ciclo de vídeo, entre outros [56]. Este mecanismo possibilita estruturar o conjunto de dados que têm como propósito treinar e testar a rede.

4.1 Manga de Calibração

Esta secção inicia-se pela simulação da manga em *Cinema 4D*. Como a manga é bastante pormenorizada, a sua projeção física é um processo longo e foi necessário estruturar um método que permitisse extrair imagens de forma a criar um conjunto de dados que, posteriormente, fossem utilizados para treinar o algoritmo de *Machine Learning*.

De seguida, é apresentada a simulação do modelo, onde são mencionados quatro modelos que possuem diferentes metodologias. Estes modelos foram desenvolvidos devido ao facto de esta implementação se tratar de um processo para o qual há pouca fundamentação, sendo necessário criar novos modelos, realizar testes e analisar os dados obtidos de forma a verificar qual o que apresenta melhor desempenho.

O último subcapítulo retrata a construção da manga, onde são descritos todos os métodos utilizados até atingir a fase final, desde a conceção do molde até ao protótipo final.

4.1.1 Ambiente de simulação

A constituição da manga é desenvolvida no *Cinema 4D*, sendo demonstrado pela Figura 31. Todas as dimensões têm como base o protótipo delineado no Capítulo 3.

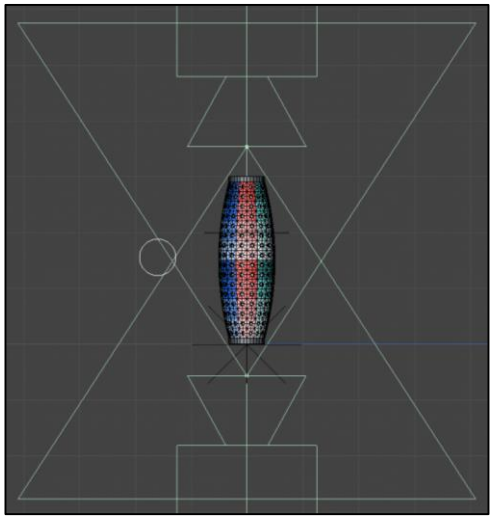


Figura 31 – Manga de calibração desenvolvida em *Cinema 4D*.

Na Figura 31 observa-se a manga e as câmaras nas extremidades. De forma a enumerar as 64 secções, foi desenvolvido um sistema de cores. Este método foi escolhido devido à sua simplicidade e eficácia, onde inicialmente surgiu a ideia de criar uma textura na manga que permitisse projetar a numeração. Apesar de ser um método que seria mais intuitivo visualmente, alteraria as propriedades da manga e, como o objetivo desta ferramenta foi apresentar uma manga semelhante à que seria realizada na prática, concluiu-se que não seria a implementação mais adequada. Sendo assim, tornou-se importante a elaboração de um novo método, que é o caso da sequência de cores. Esta metodologia utiliza oito cores com diferentes tonalidades, sendo que a cada coluna corresponde uma cor e a numeração é estruturada por ordem crescente, de acordo com a tonalidade das cores.

A Tabela 5 apresenta a descrição da cor com a respetiva numeração.

Cor	Sequência Maior Tonalidade → Menor Tonalidade
Vermelho	1 → 8
Azul Escuro	9 → 16
Verde	17 → 24
Amarelo	25 → 32
Laranja	33 → 40
Roxo	41 → 49
Rosa	49 → 56
Azul Claro	57 → 64

Tabela 5 – Mapeamento da manga de calibração em *Cinema 4D*.

É importante referir que a sequência de cores de cada coluna foi repetida, uma vez que cada uma apresenta oito áreas de secção e, como não foi possível adquirir oito tonalidades da mesma cor, houve a necessidade de repetir. A extremidade da manga que apresenta maior tonalidade é onde se inicia a sequência numérica.

A Figura 32 demonstra a sequência de cores, com base na Tabela 5.

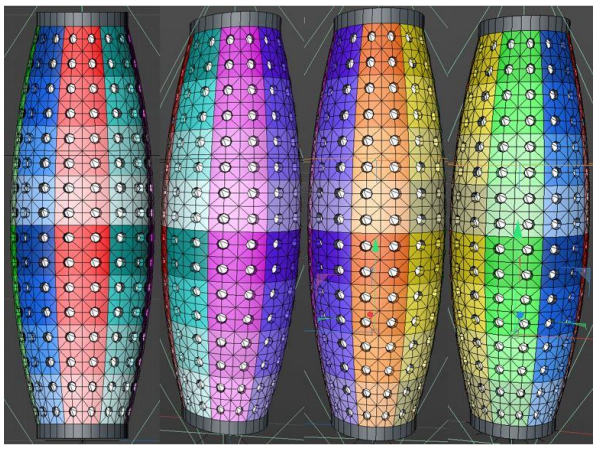


Figura 32 – Sequência de cores para o mapeamento da manga.

Após a classificação das secções, foi utilizada uma esfera com um diâmetro de 1cm que permitiu simular o dedo humano. Esta esfera foi configurada para tocar numa determinada secção, tendo como propósito deformar a manga. Ao surgir a alteração do estado inicial da estrutura, os marcadores brancos presentes no interior da manga deslocaram-se e possibilitaram extrair dados acerca do toque. Durante este

processo, foram extraídas imagens internas provenientes das câmaras e foram armazenadas de forma a constituir um *dataset*.

A Figura 33 representa a simulação do toque na secção 1, onde se visualiza a deformação dos marcadores, observando as imagens da câmara superior e inferior.

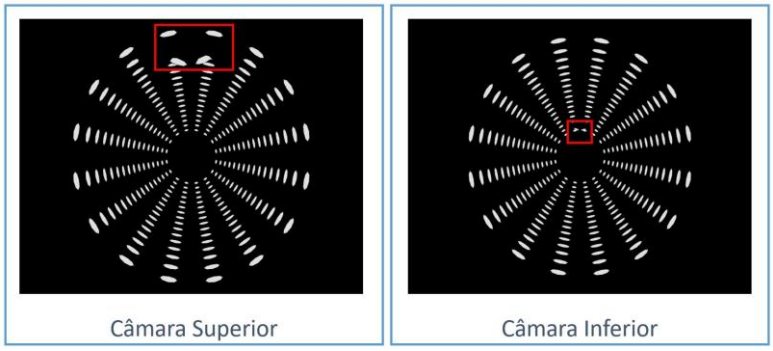


Figura 33 – Toque na secção 1 com base na visualização da câmara superior e inferior.

Por cada secção, foram retiradas oito imagens de cada uma das câmaras. Como o modelo apresenta 65 saídas, foram extraídas 1040 imagens que necessitaram de ser classificadas para treinar o algoritmo de *Machine Learning.* O método utilizado consistiu em armazenar as imagens com nomes específicos, como demonstra a Figura 34.

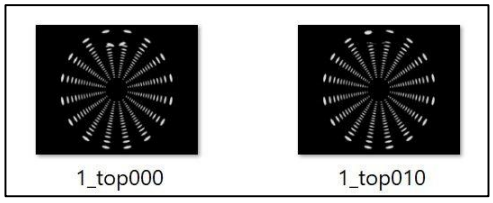


Figura 34 – Método utilizado para salvaguardar informações relevantes da imagem.

Em relação à deteção da profundidade do toque, a extração das imagens com as configurações necessárias para treinar o modelo não foi executada, uma vez que o software não apresenta as funcionalidades necessárias para aplicar as deformações. Isto porque a manga no ambiente virtual não possui uma estrutura rígida idêntica à real e tornou-se inviável aplicar o intervalo de deformações pretendido. Desta forma, o algoritmo de *Supervised Learning* desconhece a profundidade da deformação e não consegue realizar o treino, sendo apenas extraídos os dados que permitiram a deteção da localização do toque nas áreas delineadas.

4.1.2 Simulação do modelo

Neste subcapítulo é demonstrado o desenvolvimento dos vários modelos, tendo como base as imagens extraídas no ambiente de simulação. Como se tratou de um processo de aprendizagem, foi necessário desenvolver melhorias ao longo dos modelos que permitissem alcançar um melhor desempenho.

4.1.2.1 Modelo 1

O primeiro modelo utilizou apenas as imagens da câmara acoplada ao topo da manga, com o intuito de observar a utilidade das duas câmaras. Cada imagem apresenta 720 pixéis de largura, 576 pixéis de altura e três canais para as cores RGB.

Nesta arquitetura as imagens foram redimensionadas para uma resolução de 36x36 e convertidas para a escala de cinzentos. A Figura 35 demonstra a sequência realizada para processamento dos dados utilizados no modelo 1. Como a aquisição de dados foi derivada de apenas uma câmara, o modelo apresentou um *dataset* com 520 imagens.

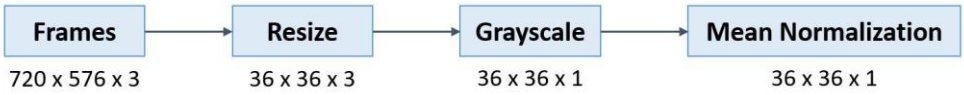


Figura 35 – Processamento dos dados do modelo 1.

Logo após o processo anterior, ocorreu a configuração da rede neuronal. Esta estrutura teve como base o modelo desenvolvido em [44], onde foram utilizadas duas *convolutional layers*, duas *pooling layers*, uma *flatten layer* e apenas uma *fully-connected layer* que apresenta as saídas pretendidas. No Capítulo 3, foram apenas referenciadas as três camadas mais importantes. Contudo, nesta implementação utilizou-se também a *flatten layer*, sendo que esta camada permitiu converter os dados presentes nas *convolutional* e *pooling layers* para um *array* 1D de forma a criar um vetor de caraterísticas que foi ligado à *fully-connected layer*.

A Tabela 6 apresenta os detalhes da rede neuronal desenvolvida.

Camada	**Dimensão**	***Stride***	**Dimensão da Saída**
Convolutional layer	3 x 3	1	34 x 34 x 32
Max pooling layer	2 x 2		17 x 17x 32
Convolutional layer	3 x 3	1	15 x 15 x 64
Max pooling layer	2 x 2		7 x 7 x 64
Flatten			1 x 1 x 3136
Saída			1 x 1 x 65

Tabela 6 – Camadas presentes no modelo 1 com as respetivas dimensões.

Na constituição da rede neuronal, para além das camadas utilizadas e das suas dimensões, foi importante selecionar o algoritmo de otimização que permitiu minimizar o erro bem como as funções de

ativação. O método escolhido foi designado por *Adam* e consiste numa extensão do *stochastic gradient descent* que mantém ao longo do processo de treino o mesmo *learning rate* para todas as atualizações dos *weights* e *bias*. Esta técnica foi apresentada por *D. P. Kingma* e *J. Lei Ba* em [57], onde demonstraram que o *Adam* aplicado ao algoritmo de regressão de reconhecimento de dígitos escritos apresenta uma melhor eficiência, comparativamente com os outros quatro algoritmos analisados. Na decisão das melhores funções de ativação para cada uma das camadas, foi tido em conta a sua popularidade. Sendo assim, na *convolutinal layer* foi aplicada a *ReLU function*, enquanto que na *fully-connected* foi adotada a *Softmax*.

Para além disso, foi importante selecionar os parâmetros reguláveis como o *learning rate*, o *batch size* e as *epochs*. No entanto, estes parâmetros só foram possíveis serem analisados quando o modelo se encontrava na fase de aprendizagem. Desta forma, a seleção do valor mais adequado de cada um deles foi elaborado no processo de treino.

A etapa seguinte simboliza o processo de treino, tendo os dados de entrada as respetivas referências. Este procedimento permitiu treinar o modelo e selecionar os parâmetros de forma a otimizar a rede. Ao longo desta etapa, foram estruturadas três avaliação, sendo que todas elas apresentaram a mesma configuração, tendo apenas como variável o que se propôs a analisar. A primeira analisou o *learning rate* (λ), sendo que a Tabela 7 representa o número do teste com base num determinado valor de λ.

A Figura 36 apresenta os resultados obtidos no processo de treino.

Número do Teste	1	2	3
$\boldsymbol{\lambda}$	0.01	0.001	0.0001

Tabela 7 – Número do teste efetuado e o respetivo *learning rate* para o modelo 1.

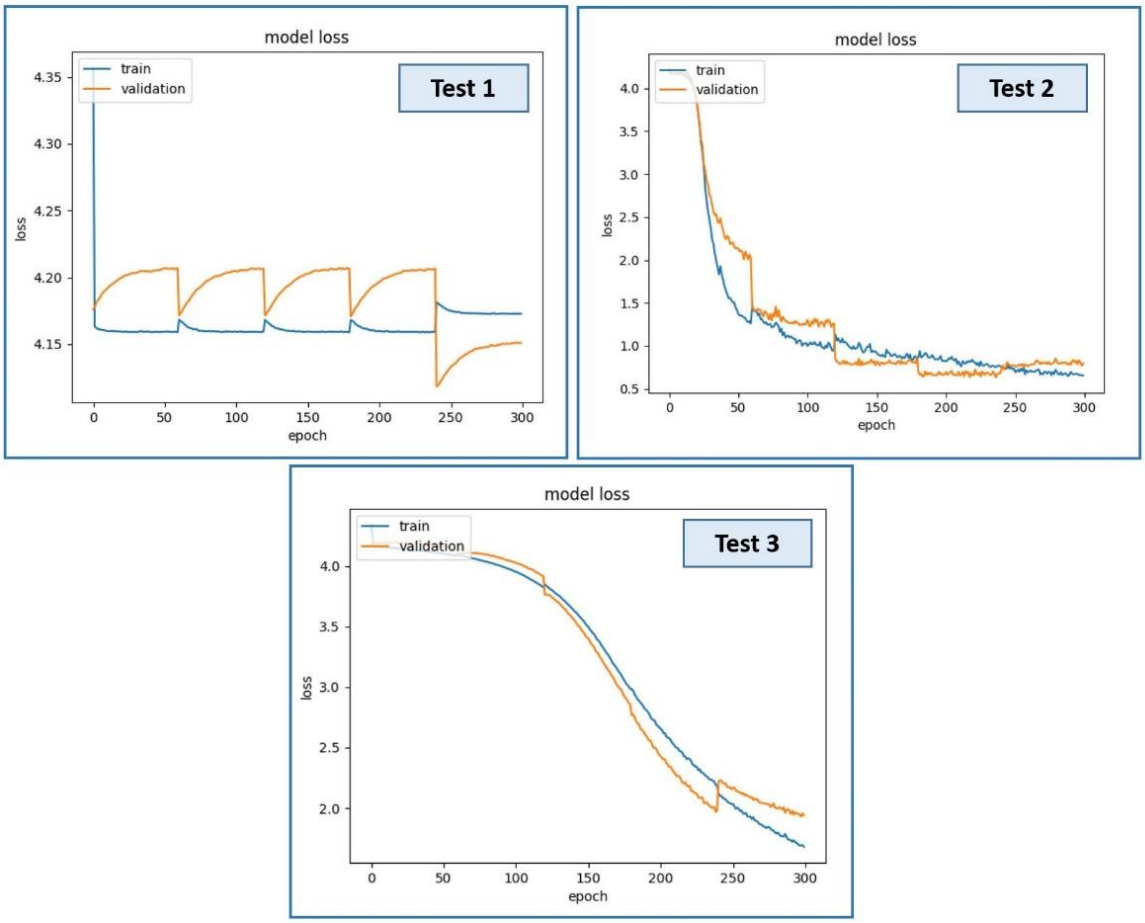


Figura 36 – *Loss function* do modelo 1 no processo de treino em função do *learning rate*.

Na Figura 36 verifica-se que no teste 1 o modelo possui uma aprendizagem instável. Isto deve-se ao facto de o *learning rate* estar demasiado alto, que provoca uma transição de estado do gradiente descente muito brusca, aumentando involuntariamente o erro gerado no processo de treino. Esta alteração resulta num reajuste dos pesos demasiado grande e, consequentemente, provoca oscilações no desempenho do modelo. Em relação ao teste 2, o *learning rate* é o adequado, uma vez que a *loss function* converge exponencialmente para zero. Além disso, não apresenta oscilações significativas, comprovando que o processo de transição ocorre suavemente. O teste 3 demonstra o longo tempo que demora a fase de treino, isto porque o modelo exibe uma curvatura da *loss function* pouco acentuada, concluindo-se que retrata um *learning rate* baixo. Posto isto, é selecionado o *learning rate* de 0.001.

A segunda avaliação consistiu na escolha do número de *epochs* ideal. Para isto, definiu-se um número considerado elevado perante o conjunto de dados. Esta implementação teve como intuito observar quando o gráfico da *loss function* deixa de convergir. A Figura 37 retrata a *loss function* do modelo.

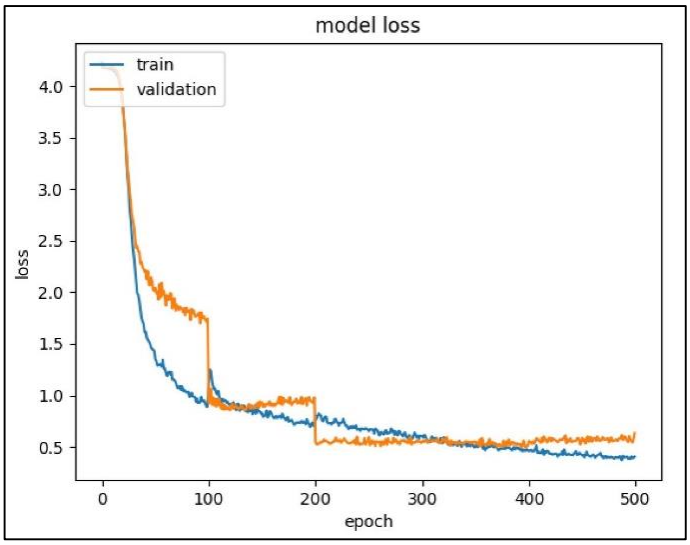


Figura 37 – *Loss function* para uma *epoch* de 100.

Na figura visualiza-se que a linha correspondente aos dados de validação deixa de convergir às 200 *epochs*, mas a azul que simboliza os dados de treino continua a convergir. Isto pode indicar que, a partir dos 200, o modelo apresenta um ligeiro *overfitting*. No entanto, a *loss* dos dados de validação não apresenta um aumento, mantendo-se constante. Posto isto, foi selecionado o número 300, como sendo o mais apropriado. Contudo, como o modelo utiliza o método de *Cross Validation* com k = 5, cada *epoch* definida no modelo representa 5 no gráfico. Sendo assim, a escolha das 300 *epochs*, na verdade correspondem a 60.

Por último, ocorreu a terceira avaliação que carateriza o ajuste do *batch size.* Para isso, iniciou-se pelos valores mais usuais para este tipo de parâmetro, sendo eles o 32, 64 e 128. Como ponto de partida, começa-se com o menor valor e decorre de forma crescente, como demonstra a Figura 38.

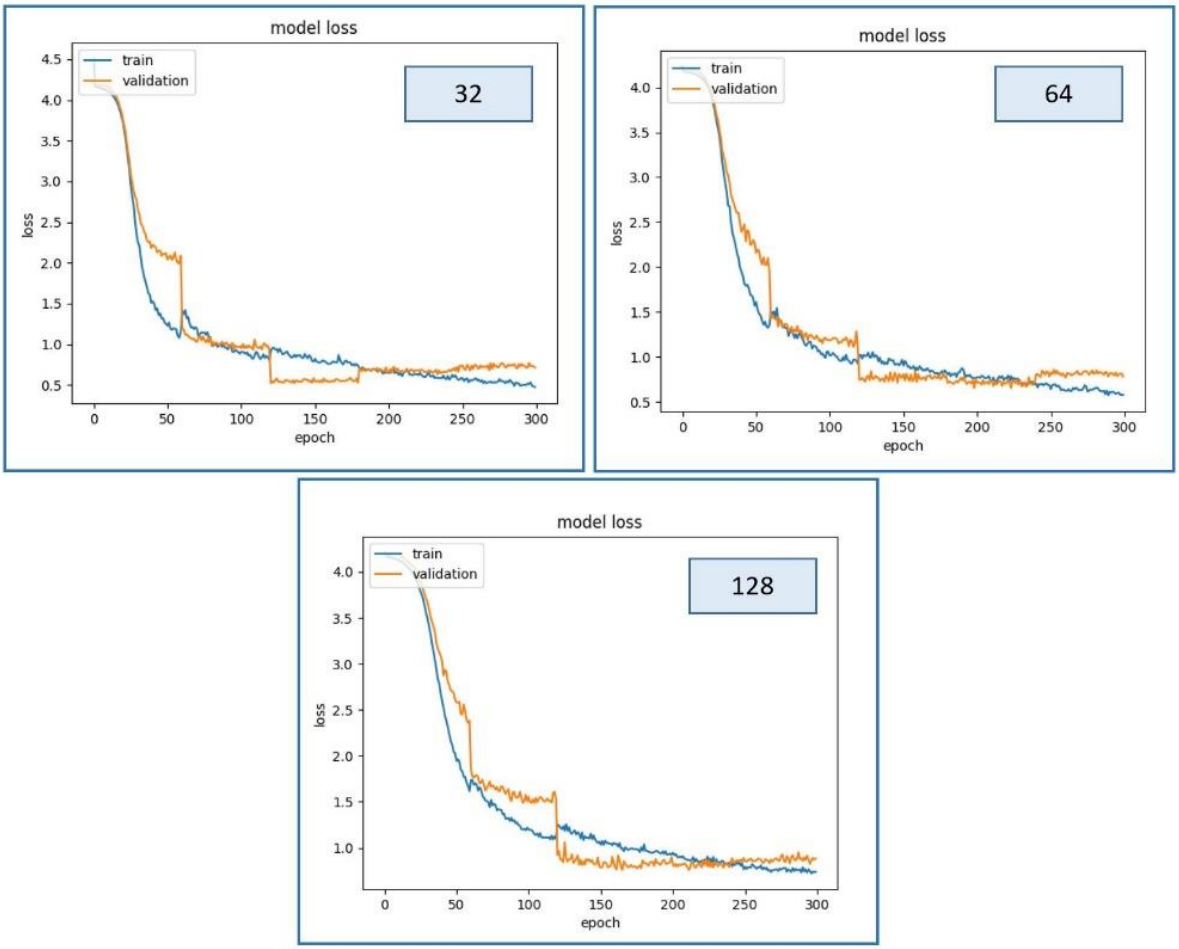


Figura 38 – *Loss function* do modelo 1 com diferentes *batch sizes*.

Apesar de não ser significativo, o modelo apresenta uma maior curvatura quando o *batch size* corresponde a 32 e foi considerado o valor mais adequado. Após esta conclusão, ocorreu a definição dos parâmetros na Tabela 8.

Parâmetro	*Learning rate*	*Epochs*	*Batch size*
Valor	0.001	60	32

Tabela 8 – Parâmetros reguláveis com os valores numéricos selecionados.

Após a configuração dos parâmetros, decorreu o processo de treino e validação do modelo. As métricas de análise do desempenho encontram-se descritas na Figura 39, que apresenta a *accuracy* e a *loss function*.

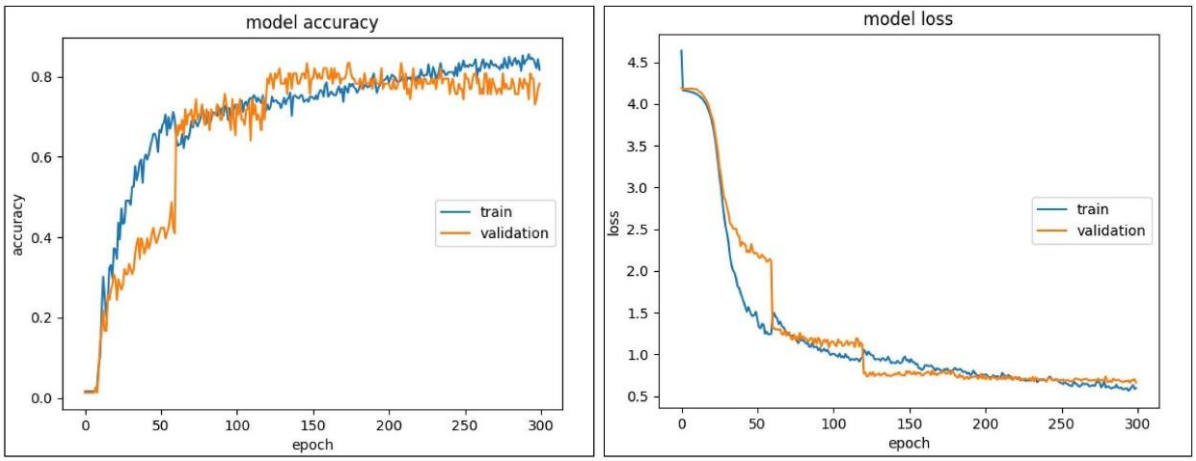


Figura 39 – *Accuracy* e *Loss* do processo de treino do modelo 1.

Analisando os gráficos anteriores, constatou-se uma boa atuação do modelo perante o processo de aprendizagem. Neste processo, apesar de não ser explícito, foi também contabilizado o tempo de treino, sendo considerado uma variável importante na medida em que permite comparar o desempenho dos diferentes modelos em função desta varável. O seu valor foi impresso juntamente com a *accuracy* dos dados de teste, uma vez que esta métrica é obtida no final do processo.

Por último, elaborou-se a avaliação do modelo, onde foram extraídos os resultados provenientes dos dados que não são conhecidos. Nesta implementação, os processos de treino e teste foram efetuados cinco vezes com dados aleatórios de forma a avaliar a execução do modelo com diferentes aprendizagens, sendo que a Tabela 9 apresenta as experiências, tendo como aspeto importante de visualização o tempo de treino bem como a *accuracy* dos dados de teste.

Experiência	**1**	**2**	**3**	**4**	**5**
Tempo de Treino	0:01:35	0:01:22	0:01:36	0:01:39	0:01:24
Accuracy Test	0.63	0.62	0.68	0.71	0.65

Tabela 9 – *Accuracy* dos dados de teste e tempo de treino do modelo 1.

Na Tabela 9 é constatado que o modelo detém um desempenho insatisfatório e tornou-se necessário estruturar novas implementações que possibilitassem melhorar o seu funcionamento.

4.1.2.2 Modelo 2

A performance do modelo 1 foi considerada baixa e, desta forma, foi estruturado o modelo 2. Esta arquitetura implementou uma nova técnica, caraterizada na Figura 40, que consistiu na sobreposição de imagens das diferentes câmaras, tendo ambas o mesmo peso. Este método permitiu adicionar padrões que contribuíram para um aumento do desempenho do modelo.

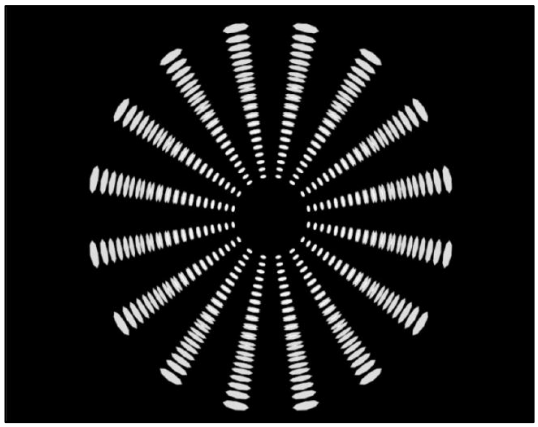


Figura 40 – Sobreposição das imagens provenientes das duas câmaras.

O processamento das imagens foi muito semelhante ao modelo anterior, acrescentando apenas o processo de adição da imagem da câmara inferior. Este processamento é representado na Figura 41, onde são descritas todas as suas etapas bem como a dimensão das imagens em cada uma delas.

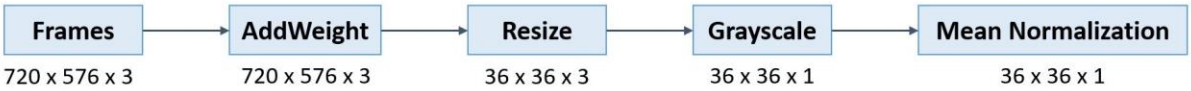


Figura 41 – Processamento dos dados do modelo 2.

Apesar de haver uma alteração na manipulação dos dados, o modelo 2 utilizou o mesmo *dataset* e rede neuronal do modelo 1. A fase de treino e validação desta nova implementação encontra-se exibida na Figura 42 e a Tabela 10 apresenta os resultados obtidos do processo de teste.

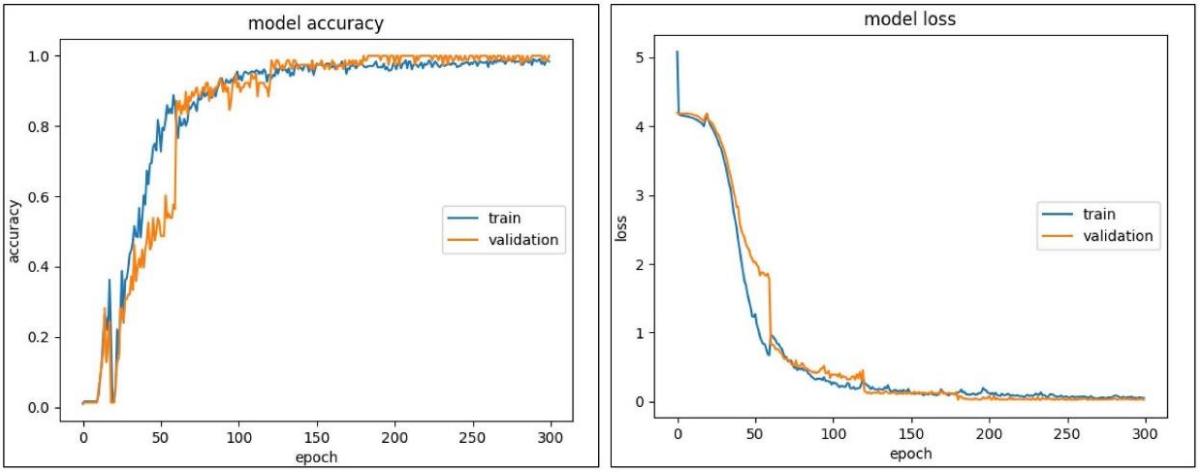


Figura 42 – *Accuracy* e *Loss* do processo de treino do modelo 2.

Experiência	**1**	**2**	**3**	**4**	**5**
Tempo de Treino	0:01:37	0:01:38	0:01:40	0:01:39	0:01:36
Accuracy Test	0.88	0.89	0.91	0.87	0.92

Tabela 10 – *Accuracy* dos dados de teste e tempo de treino do modelo 2.

Da análise das métricas de desempenho do modelo nos diferentes processos, verificou-se que ambos dispõem de uma boa performance. No entanto, considerou-se viável melhorar o modelo porque constatou-se uma limitação na utilização das imagens na escala de cinzento. Certas imagens que resultaram da sobreposição e que representam classes distintas possuem muitas semelhanças, como demonstra a Figura 43. Esta similaridade pode afetar o funcionamento do modelo.

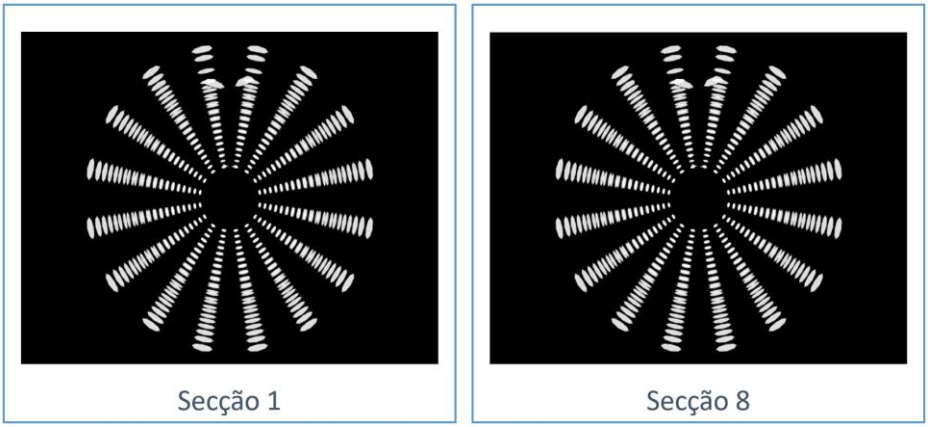


Figura 43 – Toque nas classes 1 e 8 em escala de cinzentos observado pelas duas câmaras.

Na Figura 43 é representado o toque na secção 1 e na secção 8 que resulta da sobreposição das imagens em escala de cinzento, onde se observa que a classe 1 e 8 são muito semelhantes. Esta caraterística pode tornar-se problemática para a correta classificação da área de secção, uma vez que dificulta o modelo a encontrar um padrão e pode originar uma rede com um desempenho mediano. Sendo assim, manifestou-se a necessidade de manipular as imagens de modo a conseguir ser notória a diferença entre classes.

4.1.2.3 Modelo 3

O modelo 3 é caraterizado pelo desenvolvimento de um novo método de processamento das imagens. Este mecanismo consiste na manipulação do canal RGB, sendo que a cada canal corresponde uma câmara. Como só se utiliza duas imagens por processamento, o canal R foi colocado a zero.

A Figura 44 representa a implementação especificada previamente, onde as imagens da câmara superior foram inseridas no canal Blue (Azul), enquanto que as imagens inferiores foram introduzidas no canal Green (Verde).

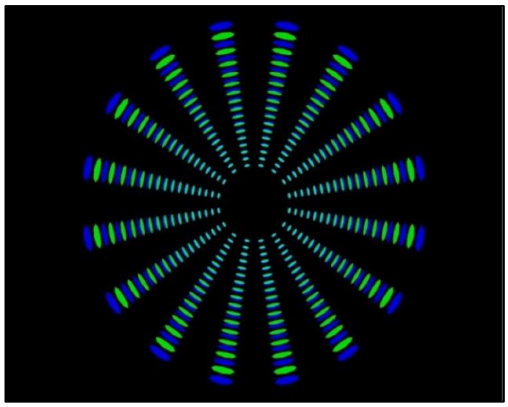


Figura 44 – Representação das imagens nos canais RGB.

Comparativamente com o modelo 2, foi retirado o método que sobrepõe as imagens em escalas de cinzentos e aplicou-se a técnica que manipula as imagens através dos canais RGB. As restantes etapas mantiveram-se, tendo apenas como alteração a sua dimensão. Isto porque o canal RGB utiliza uma profundidade de 3, enquanto que o de escala de cinzento usa somente 1. Este processo de transformação das imagens encontra-se caraterizado na Figura 45.

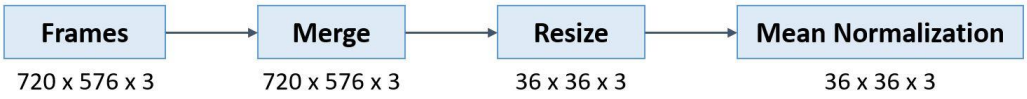


Figura 45 – Processamento dos dados no modelo 3.

Esta nova implementação possibilita distinguir melhor as diferentes classes, podendo resultar num modelo preditivo com um desempenho superior ao modelo 2. De forma a comprovar a diferença entre os dois métodos, foram utilizadas as mesmas imagens ilustradas no modelo 2, mas com o método mais recente.

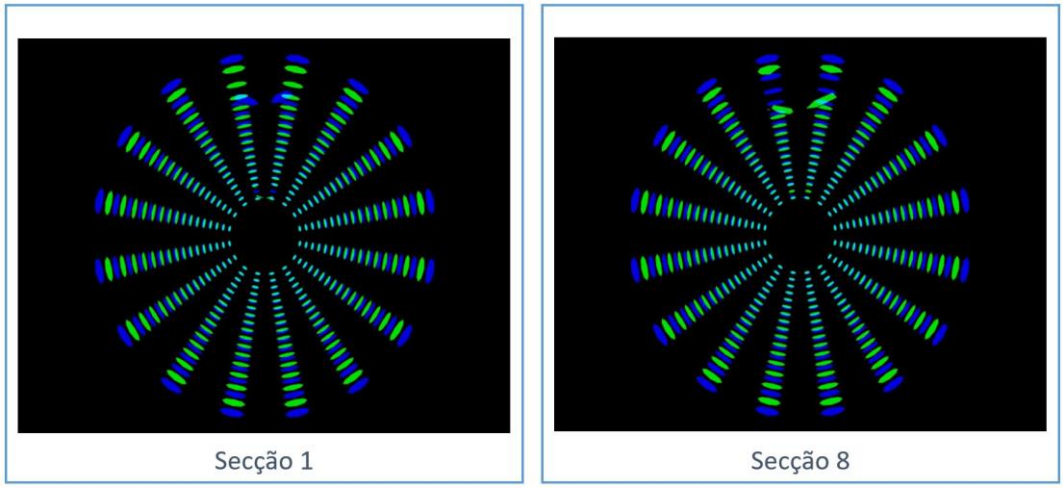


Figura 46 – Toque nas classes 1 e 8 em RGB observado pelas câmaras.

Na Figura 46 verifica-se que há uma maior distinção com o método atual e avançou-se para o processo de treino e teste de forma a verificar a eficiência do modelo. No entanto, é importante mencionar que as

imagens utilizadas nos exemplos anteriores apresentam uma resolução de 720x480 de forma a ser mais percetível observar o contraste, sendo que na entrada do modelo 3 encontram-se com uma resolução de 36x36.

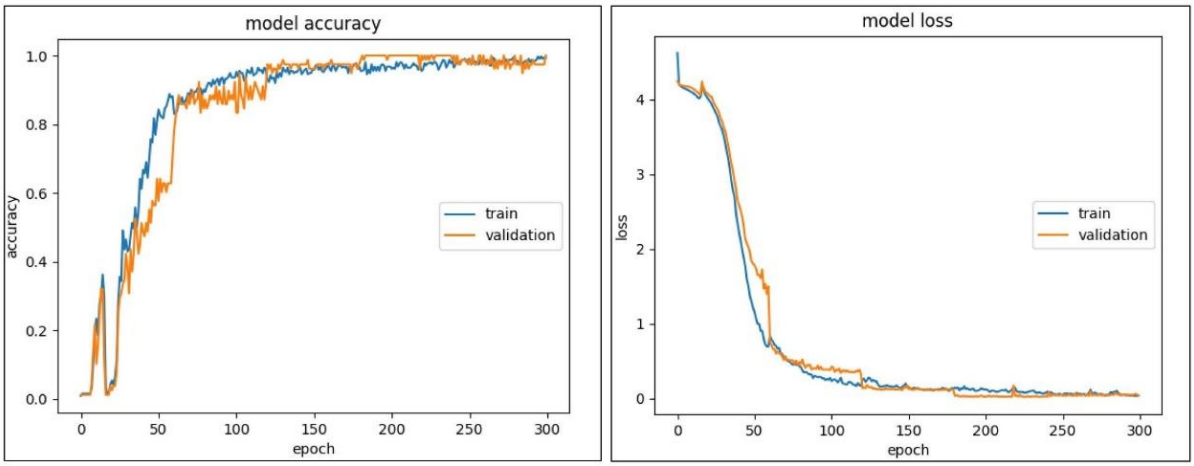


Figura 47 – *Accuracy* e *Loss* do processo de treino do modelo 3.

Experiência	**1**	**2**	**3**	**4**	**5**
Tempo de Treino	0:01:38	0:01:40	0:01:34	0:01:43	0:01:29
Accuracy Test	0.9	0.91	0.92	0.9	0.89

Tabela 11 – *Accuracy* dos dados de teste e tempo de treino do modelo 3.

No entanto, através da comparação dos resultados do modelo 2 e 3, observa-se que não houve uma alteração significativa do desempenho do modelo. Apesar de ser visualmente evidente uma grande diferença, analiticamente não houve uma mudança relevante. Esta adversidade pode ser devido ao facto de a resolução utilizada ser baixa, como demonstra a Figura 48.

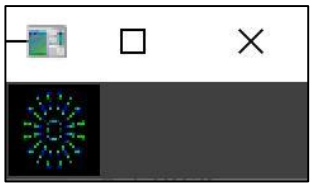


Figura 48 – Representação das imagens nos canais RGB com uma resolução de 36x36.

Sendo assim, o próximo passo consistiu em aumentar a resolução das imagens processadas e, consequentemente, aumentar o tamanho das camadas. Como houve a modificação da dimensão da rede, surgiu a necessidade de criar um novo modelo.

4.1.2.4 Modelo 4

Este modelo consiste no último protótipo desenvolvido. Foi estruturado para aumentar a resolução das imagens processadas e comparar qual a implementação que apresentou melhores resultados, sendo

que as resoluções das imagens do modelo 3 e 4 foram redimensionadas para 108x108. Como houve a alteração das dimensões das imagens, reconstituiu-se a rede neuronal desenvolvida anteriormente, onde as camadas utilizadas foram exatamente as mesmas. A única alteração residiu no seu tamanho.

A Tabela 12 demonstra a constituição da nova rede neuronal.

Camada	**Tamanho**	***Stride***	**Saída da Camada**
Convolutional layer	3 x 3	1	106 x 106 x 32
Max pooling layer	2 x 2		53 x 53 x 32
Convolutional layer	3 x 3	1	51 x 51 x 64
Max pooling layer	2 x 2		25 x 25 x 64
Flatten			1 x 1 x 40000
Saída	SDN		1 x 1 x 65

Tabela 12 – Constituição da rede neuronal do modelo 4.

Do modelo, analisa-se que existiu um aumento da camada de entrada que, consequentemente, aumentou a dimensão de todas as outras camadas, tendo como exceção a camada de saída. Essa camada manteve-se constante, uma vez que o modelo continuou a ter as mesmas saídas das arquiteturas anteriores. Com a implementação deste novo modelo, foram efetuados todos os testes necessários para comparar o método atual com o modelo 2 e 3, sendo que se iniciou pela sobreposição das imagens em escala de cinzentos com a nova resolução. A Figura 49 representa o desempenho do processo de treino, enquanto que a Tabela 13 demonstra o tempo de treino e a métrica de avaliação da fase de teste.

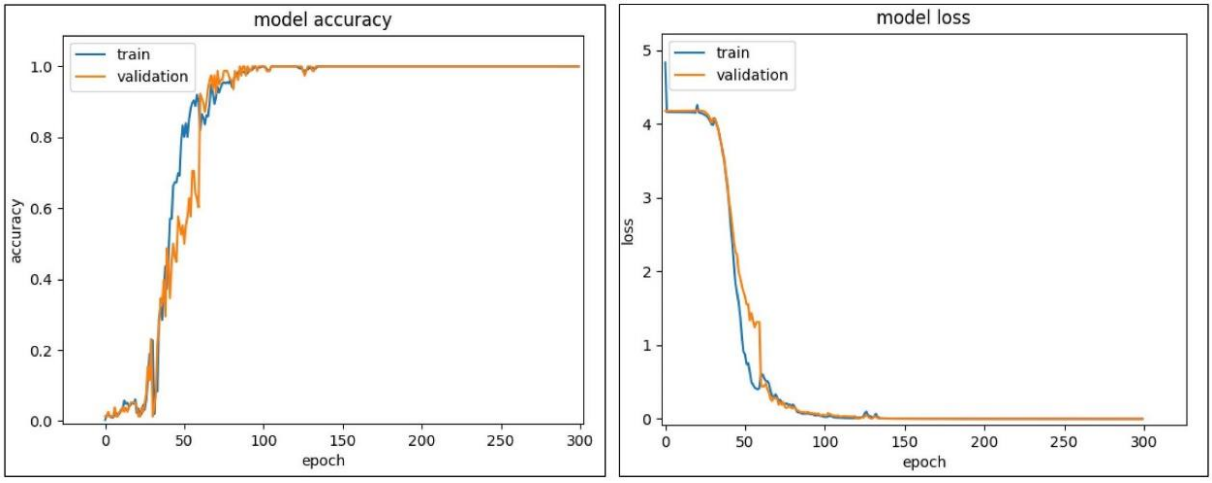


Figura 49 – *Accuracy* e *Loss* do processo de treino do modelo 4 em escala de cinzentos.

Experiência	1	2	3	4	5
Tempo de Treino	0:12:56	0:12:40	0:13:03	0:11:55	0:13:01
Accuracy Test	0.985	0.938	0.962	0.954	0.962

Tabela 13 – *Accuracy* dos dados de teste e tempo de treino do modelo 4 em escala de cinzentos.

Os resultados obtidos das métricas de avaliação para o processo de treino, validação e teste permitiram concluir que o modelo preditivo é excecional.

Para a análise do método RGB, foi utilizado o modelo retratado anteriormente. Contudo, a rede necessitou de sofrer uma pequena alteração na sua dimensão, uma vez que foi preciso aumentar a profundidade das camadas devido ao facto de este processo utilizar três canais.

A Figura 50 e Figura 51 demonstram o desempenho do modelo no processo de treino, enquanto que a Tabela 14 exibe o seu comportamento com os dados de teste.

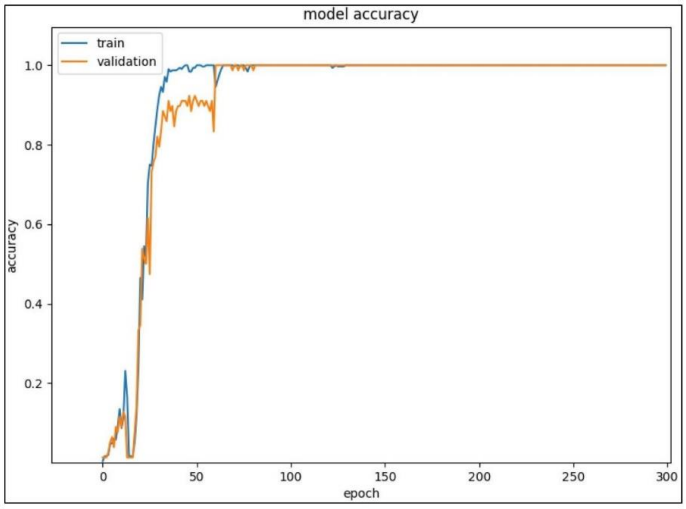


Figura 50 – *Accuracy* do processo de treino do modelo 4 em RGB.

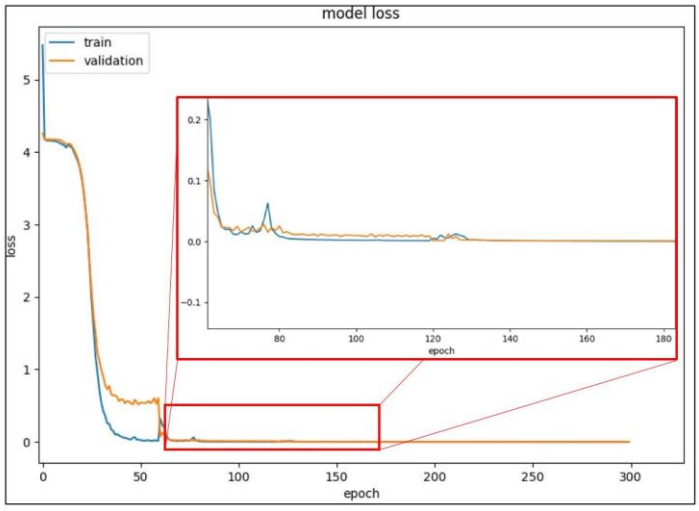


Figura 51 – *Loss* do processo de treino do modelo 4 em RGB.

Experiência	1	2	3	4	5
Tempo de Treino	0:13:20	0:15:13	0:13:41	0:14:06	0:14:53
Accuracy Test	0.938	0.90	0.946	0.90	0.931

Tabela 14 – *Accuracy* dos dados de teste e tempo de treino do modelo 4 em RGB.

Analisando ambos os métodos estruturados neste modelo, concluiu-se que os dois apresentaram bons resultados, tendo uma precisão nos dados de teste próxima de 100%. No entanto, verificou-se que o modelo em escala de cinzentos deteve uma *accuracy* ligeiramente maior, mesmo com uma rede neuronal de dimensões inferiores ao método RGB. Além disso, o aumento da resolução das imagens na camada inicial provocou uma melhoria no processo de previsão do modelo, comparativamente com o modelo 2.

4.1.3 Construção da manga

Nesta secção são apresentadas todas as etapas para a construção da manga de calibração. Este projeto iniciou-se pelo planeamento do modelo que permitiu delinear todas as tarefas. De seguida, foi fabricado o molde com a utilização de uma impressora 3D e por fim foram interligadas todas as peças e produziu-se a manga.

A Figura 52 apresenta um esboço dos moldes desenvolvidos, onde todas as medidas da manga foram previamente definidas no Capítulo 3, tendo apenas ficado por definir a espessura. Nesta figura é também evidenciado um suporte que tem como propósito segurar a manga.

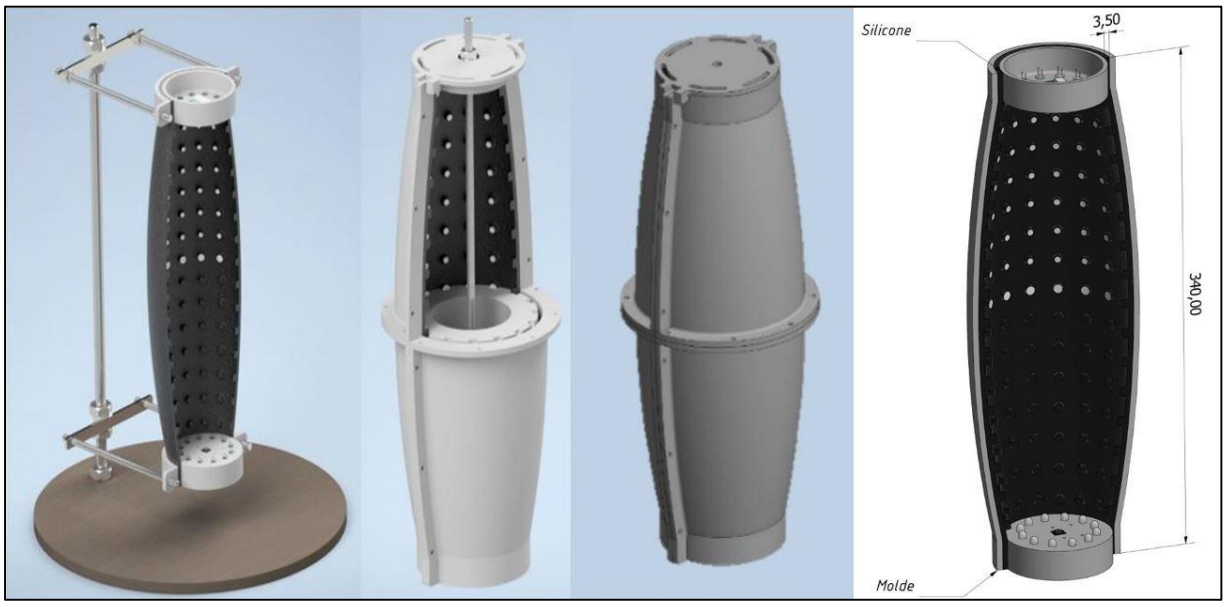


Figura 52 – Vista geral dos moldes e dos suporte em ambiente virtual 3D.

Publicação da imagem permitida pelo autor Pedro Ildo.

Na Figura 52 observam-se dois moldes: o interno e o externo. O molde interno apresenta a definição dos marcadores embebidos para que, ao ser inserido o silicone, haja a sua formação. O molde externo encontra-se distanciado 3.5mm do interior de forma a constituir a espessura da manga. Ambos os moldes se encontram divididos em subpeças devido ao facto de toda a sua estrutura ter sido impressa numa impressora 3D. Como esta ferramenta tem um eixo com um limite inferior ao tamanho total da peça, tornou-se necessário dividir o molde.

A Figura 53 carateriza todas as peças utilizadas para este processo, sendo que algumas não foram utilizadas para construir a manga, como é o caso das T1. Contudo, foram úteis posteriormente.

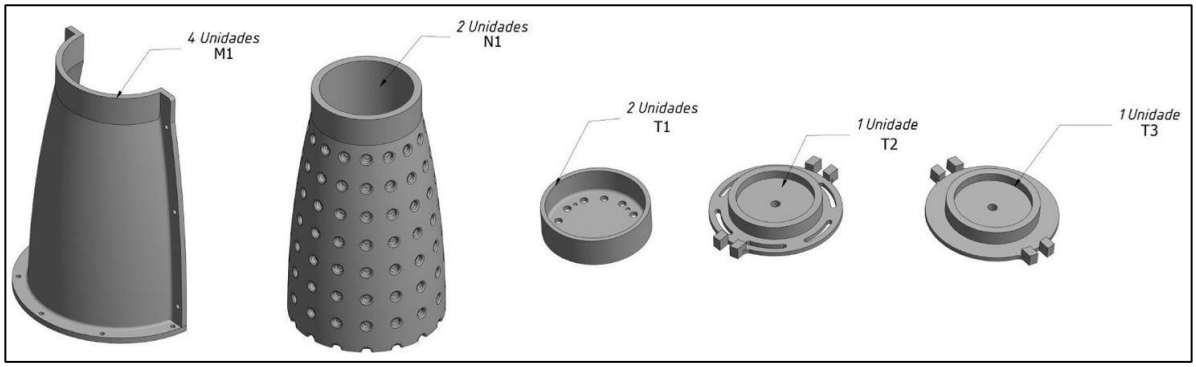


Figura 53 – Descrição geral das peças que constituem o molde.

Publicação da imagem permitida pelo autor Pedro Ildo.

Da figura, as 4 unidades M1 representam o molde externo. As 2 unidades N1 retratam o molde interno. As unidades T2 e T3 caraterizam as peças que interligam o molde interno e externo, além de isolar as extremidades. A única diferença entre ambas deveu-se à parte superior ter necessitado de uns orifícios que possibilitassem aplicar o silicone no molde. Por fim, as 2 unidades T1 permitem implementar as câmaras nas extremidades bem como os leds para iluminar a parte interna. Além disso, permitem ao suporte agarrar as extremidades.

Todas as peças retratadas na Figura 53 encontram-se evidenciadas com as respetivas dimensões em milímetros nas figuras seguintes.

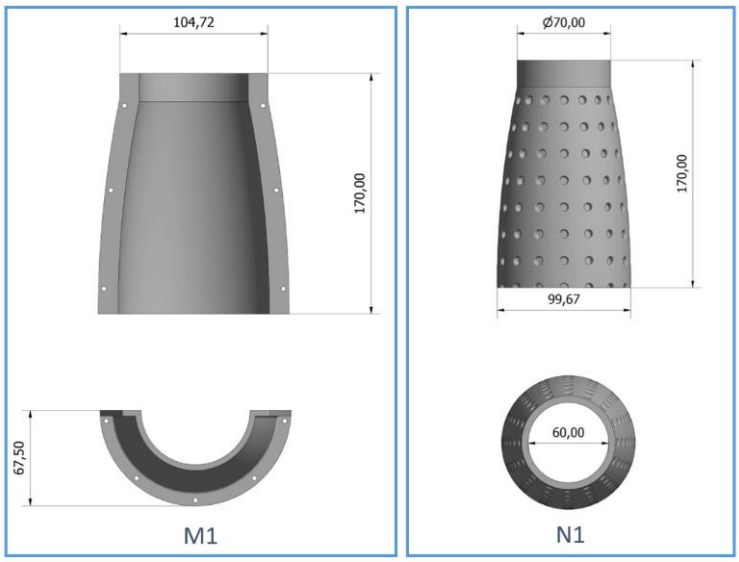


Figura 54 – Descrição da peça M1 e N1.

Publicação da imagem permitida pelo autor Pedro Ildo.

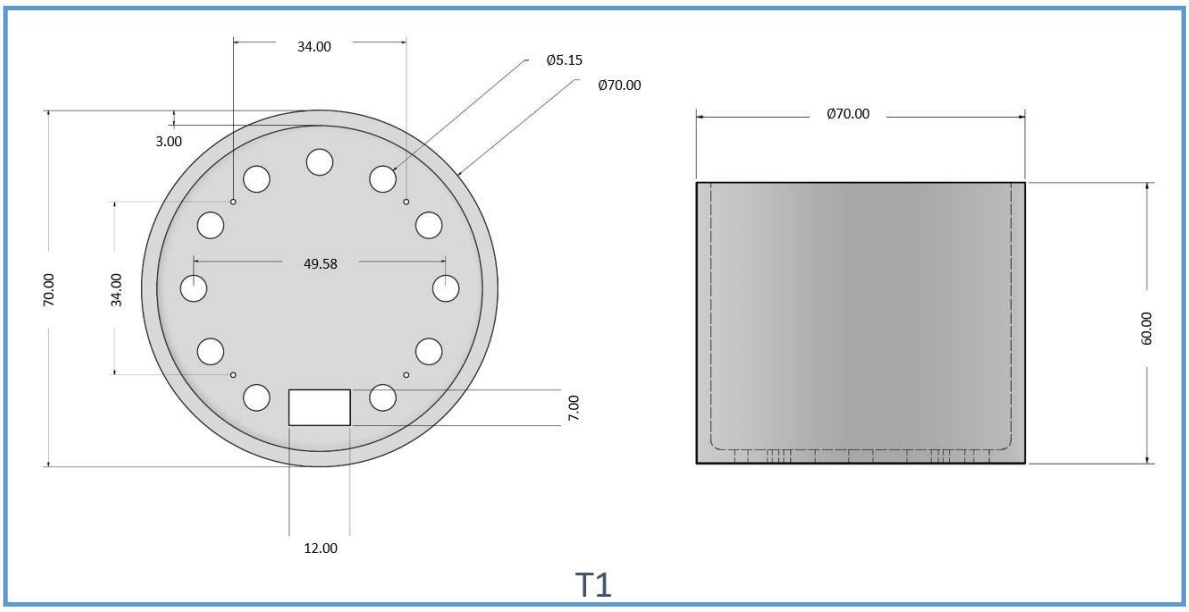


Figura 55 – Descrição da peça T1.

Publicação da imagem permitida pelo autor Pedro Ildo.

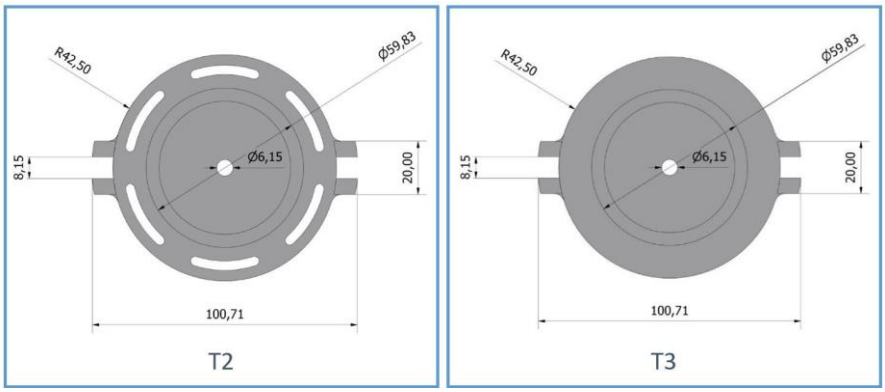


Figura 56 – Descrição da peça T2 e T3.

Publicação da imagem permitida pelo autor Pedro Ildo.

Após a configuração das dimensões das peças, imprimiram-se os moldes que são representados na Figura 57.

Figura 57 – Representação dos moldes reais de 2 unidades M1, 1 unidade N1 e 1 unidade das peças T.

Quando terminou o processo de fabrico, todos os objetos impressos com a exceção do T1 foram conectados, concebendo o molde apresentado na Figura 58.

Figura 58 – Vista externa e interna dos moldes.

A próxima etapa consistiu no seu preenchimento com o silicone do tipo RTV-2 HB Flex 0020. Este material apresentou uma pigmentação preta de forma a colocar o interior da manga a negro para que só sejam observados pelas câmaras os marcadores brancos. Contudo, os marcadores foram desenvolvidos a negro, uma vez que não foi possível estruturar uma implementação que permitisse pigmentar os marcadores a branco e o fundo a preto. Sendo assim, surgiu a necessidade de adaptar este processo de construção. Inicialmente, foi delineado pintar o silicone com um marcador ou uma tinta branca. No entanto, as propriedades constituintes do silicone não permitiram que a tinta aderisse e, consequentemente, ao mover a manga, o material não permanecesse fixo. Posteriormente, chegou-se à conclusão que o silicone só poderia ser manipulado com a adição de um outro silicone. Posto isto, foi utilizado um papel branco recortado com as dimensões dos marcadores e procedeu-se à sua fixação

com o auxílio de um silicone transparente do tipo Olivé C-22. Esta substância é incolor, prevenindo uma possível alteração da aparência do modelo.

A Figura 59 demonstra a parte interna de calibração com a implementação caraterizada anteriormente, onde também apresenta o suporte desenvolvido. Além disso, retrata a fase da divisão das 64 secções na região exterior, onde foram delineados todos os limites de cada uma delas. As fronteiras de cada secção foram definidas através de uma folha de papel branco, utilizando o silicone incolor para fixar.

Figura 59 – Parte interna e externa da manga de calibração em ambiente real.

4.1.4 Estrutura em tempo real

Neste subcapítulo encontram-se evidenciados todos os recursos realizados de forma a viabilizar a implementação em ambiente real. Isto porque, anteriormente, o modelo obtinha imagens provenientes de um ambiente de simulação, onde não havia qualquer tipo de influência do meio. No entanto, em tempo real existe uma infinidade de possíveis interferências.

4.1.4.1 Conjunto de leds

A peça T1 foi delineada com o propósito de isolar a parte interna da manga de influências externas. Contudo, esta estrutura fez com que não houvesse iluminação natural suficiente para que as câmaras capturassem uma imagem clara do interior, tornando-se importante adicionar um conjunto de leds que permitissem assegurar que as câmaras visualizassem os diferentes marcadores.

A Figura 60 carateriza as conexões e componentes que constituem um circuito led.

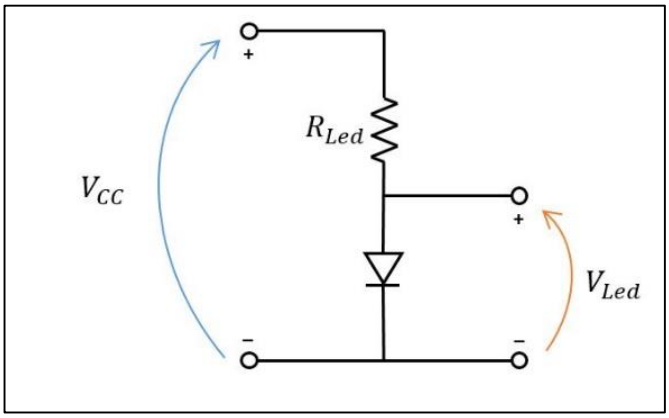


Figura 60 – Circuito esquemático do led.

O dimensionamento da resistência em série com o led foi fundamentado pela equação 22, onde V_{CC} define a tensão de entrada do circuito, V_{Led} e I_{Led} indicam, respetivamente, a tensão e a corrente no led.

$$R_{Led} = \frac{V_{CC} - V_{Led}}{I_{Led}} \tag{22}$$

Para esta determinação foi tida em conta a corrente máxima de 20mA suportada pelo led e, como se trata de um led branco, apresenta uma tensão máxima de 3V. A tensão de entrada foi definida como sendo 5V. Desta forma, a equação 23 apresenta o cálculo de R_{Led} .

$$R_{Led} = \frac{5 - 3}{0.02} = 100\Omega \tag{23}$$

A equação anterior foi desenvolvida com o intuito de obter a resistência mínima necessária para que o led não se danifique. No entanto, como se pretendeu ter um circuito duradouro, aplicou-se uma resistência de 220Ω, que promoveu diminuir a corrente e prolongar o seu tempo de vida.

O circuito esquemático do conjunto de leds é apresentado na Figura 61, utilizando a ferramenta *TinkerCad*.

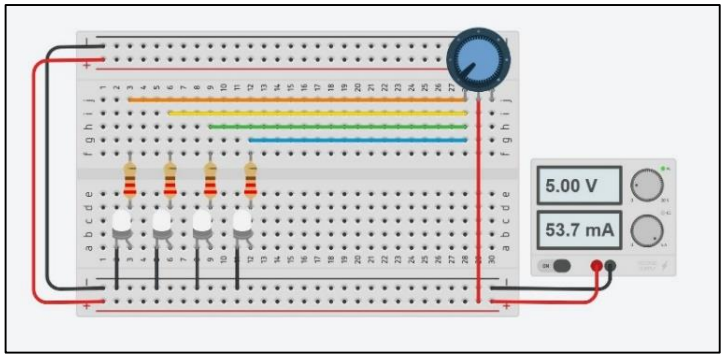


Figura 61 – Circuito elétrico do conjunto de leds em *TinkerCad*.

Na Figura 61 observa-se que circuito é alimentado por uma fonte de tensão de 5V. Logo após a fonte, encontra-se um potenciómetro de 10 kΩ que permite alterar a luminosidade dos leds. Por último, apresenta os leds com as respetivas resistências de 220Ω para limitar a corrente.

Depois do planeamento do circuito, decorreu o seu desenvolvimento prático, como demonstra a Figura 62 que possibilita visualizar o circuito elétrico do conjunto de leds bem como o seu funcionamento quando se encontra aplicado no suporte T1. Além disso, também se observa a câmara inserida no interior da peça.

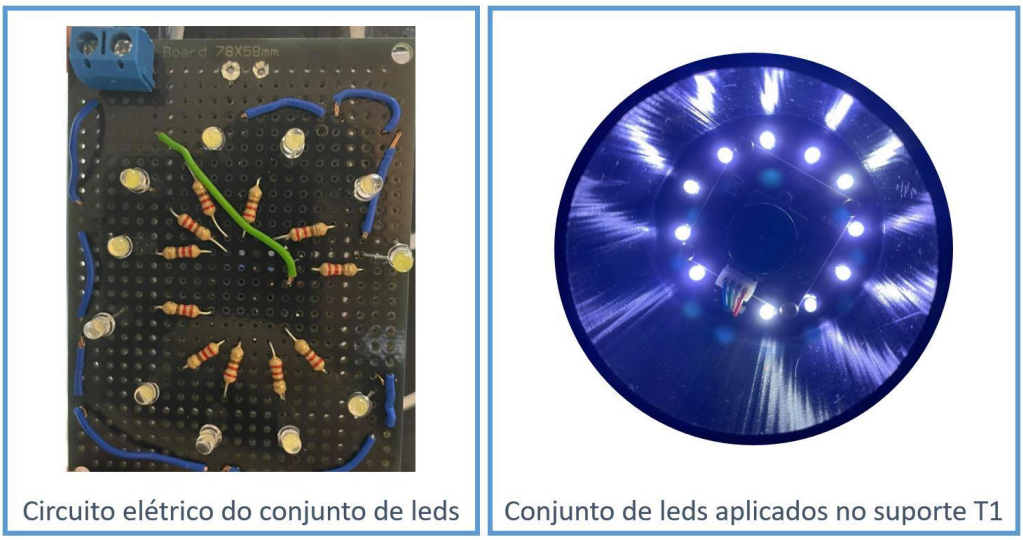


Figura 62 – Circuito elétrico e funcionamento do conjunto de leds.

No circuito esquemático é utilizada uma fonte de tensão de 5V que permite alimentar os componentes. Contudo, no modelo prático utilizou-se uma bateria de chumbo de 12V juntamente com um *step-down*, uma vez que se reutilizou material de outras implementações. Este *step-down* converteu a tensão de entrada 12V, proveniente da bateria, para uma tensão de saída de 5V.

4.1.4.2 Extração das imagens

Após a configuração dos processos que permitiram estruturar o modelo, ocorreu a aquisição das imagens que se encontra caraterizada pela Figura 63. O método aplicado utiliza um contador e um temporizador. O temporizador é utilizado para extrair as imagens de 500 em 500 milissegundos, com o intuito de dar tempo para alterar a posição do toque. O contador permite extrair um número específico de imagens, que neste caso concreto se trata de 51.

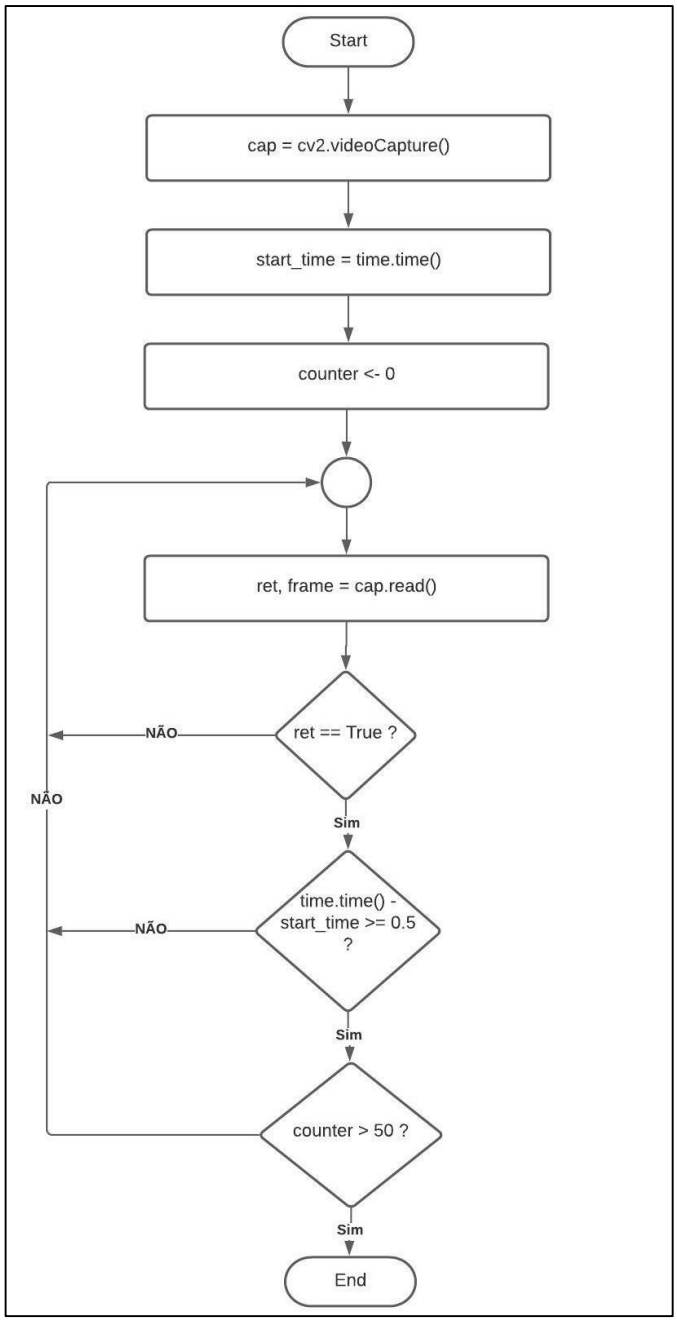


Figura 63 – Fluxograma do processo de aquisição das imagens em tempo real.

As imagens extraídas são armazenadas com uma certa designação. Esta designação apresenta a secção onde ocorreu o toque, qual a câmara que captou e, por último o número que retrata a contagem por cada secção. A Figura 64 representa a nomenclatura utilizada para guardar os dados, onde a atribuição do nome permite conhecer que o modelo se encontra perante uma imagem extraída da câmara superior na classe 2 e foi a vigésima imagem capturada.

Classe	Posição da câmara	Contador
2	_top_	020

Figura 64 – Nomenclatura utilizada para armazenar as imagens.

Com a extração das imagens de ambas as câmaras, obteve-se um conjunto de dados que apresenta um total de 6630 imagens. No entanto, analisando o conteúdo de cada uma delas, constatou-se que a imagem extraída no início de cada secção exibia pouca iluminação. Um dos motivos pode ser devido ao longo processo de ativação do seu foco, fazendo com que não capte uma imagem nítida no início. Sendo assim, as primeiras imagens de cada secção foram eliminadas do conjunto de dados, reduzindo a 6500 imagens, que com a sobreposição passaram a 3250. No processamento de dados também se verificaram algumas diferenças entre o ambiente real e o de simulação, como exibe a Figura 65.

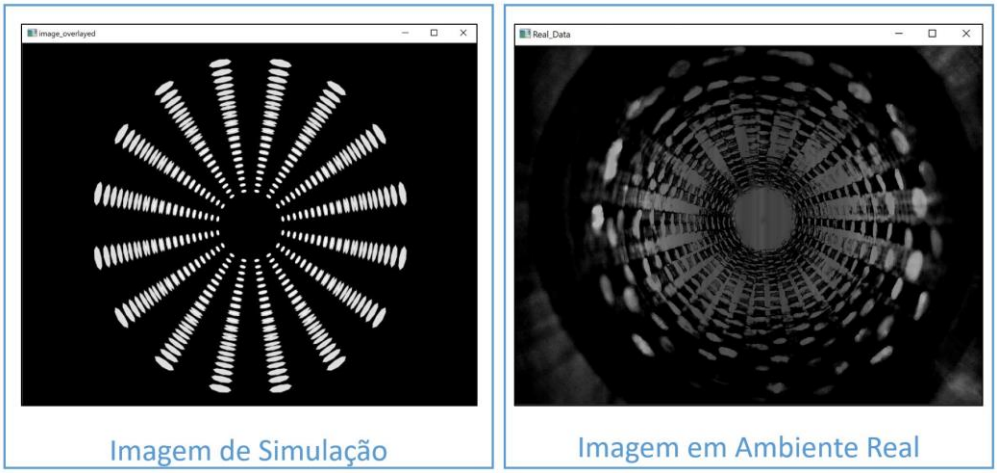


Figura 65 – Comparação entre as imagens de simulação e as reais.

Na Figura 65 observa-se que a nitidez da imagem pertencente aos dados de simulação é completamente diferente da imagens real. Isto deve-se ao eventual ruído produzido pelos leds que interfere no processamento dos dados. No entanto, as resoluções das imagens foram consideradas como sendo suficientes para o modelo adquirir padrões que servem para treinar o algoritmo de *Machine Learning*.

4.1.4.3 Análise dos modelos

O processo de análise dos modelos consistiu em avaliar as redes neuronais desenvolvidas anteriormente com a utilização do conjunto de dados reais. Esta verificação possibilitou observar o seu comportamento de forma a selecionar o que exibiu melhores resultados. Este procedimento iniciou-se pelo modelo 2, uma vez que o modelo 1 com os dados de simulação não obteve o desempenho pretendido. Todos os parâmetros designados no modelo de simulação 2 apresentados na Tabela 8 foram os mesmos utilizados para treinar e testar o modelo com os dados reais.

A Figura 66 representa as métricas utilizadas para avaliar o desempenho do processo de treino.

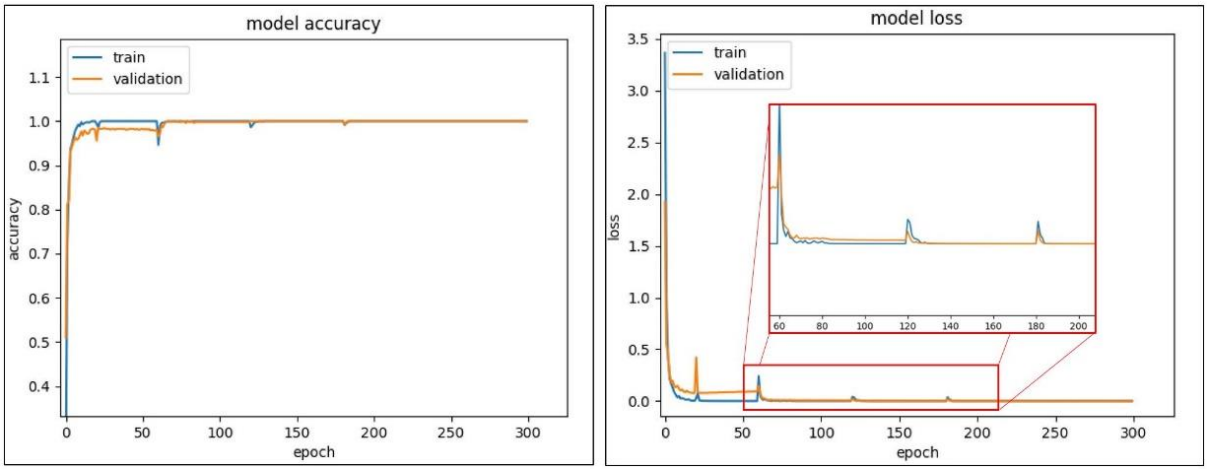


Figura 66 – *Accuracy* e *Loss* do processo de treino do modelo 2 com imagens reais.

A visualização da etapa de treino permite verificar que o modelo apresenta um desempenho excelente. No entanto, a partir da *epoch* 120, a *loss function* estabiliza e deixa de ser necessário continuar o processo de treino. Desta forma, selecionou-se o valor 150 como sendo o número de *epochs* mais apropriado, que na verdade correspondeu a 30. Isto porque o modelo treina e valida cinco vezes com apenas uma *epoch* devido à implementação do *K-Fold Cross Validation* com k=5.

A Figura 67 carateriza o processo de treino com os parâmetros retratados na Tabela 15.

Parâmetros	*Learning rate*	*Epochs*	*Batch size*
Valor	0.001	30	32

Tabela 15 – Parâmetros reguláveis atualizados para o modelo 2 com as imagens reais.

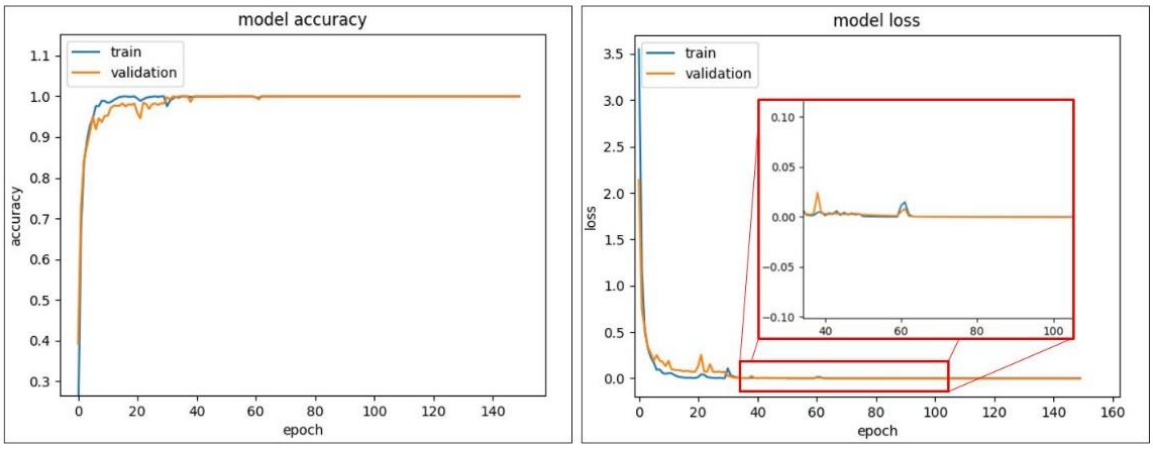


Figura 67 – *Accuracy* e *Loss* do processo de treino do modelo 2 com os novos parâmetros.

As métricas aplicadas para o modelo 2 permitiram concluir que o modelo possui uma boa performance no processo de treino. Para a avaliação da previsão do modelo, foram efetuados cinco testes que possibilitaram analisar o tempo de treino e a *accuracy* dos dados de teste.

Experiência	1	2	3	4	5
Tempo de Treino	0:05:53	0:05:27	0:05:02	0:05:37	0:04:53
Accuracy Test	0.99	0.98	0.99	0.99	0.99

Tabela 16 – *Accuracy* dos dados de teste e tempo de treino do modelo 2 com as imagens reais.

A informação descrita na Tabela 16 expõe um comportamento ideal de previsão do modelo com os dados que são desconhecidos. Esta análise permitiu constatar que este modelo manifesta todos os requisitos necessários para a implementação pretendida. No entanto, ainda faltava averiguar a funcionalidade dos restantes.

Para a análise do modelo 3, foram tidos em conta todos os parâmetros reguláveis do modelo anterior bem como a sua estrutura, de forma a realizar uma análise comparativa. Sendo assim, é retratada a implementação do método RGB com as imagens reais na Figura 68 e, posteriormente, é adicionada a Figura 69 que descreve a atuação do treino. Por fim, é averiguada a execução da previsão do modelo que se encontra exposta na Tabela 17.

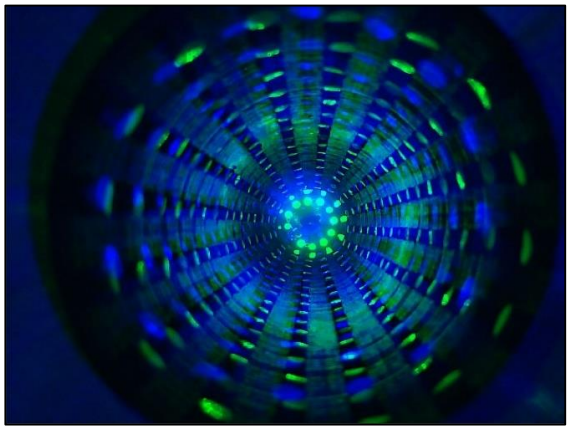


Figura 68 – Método RGB aplicado em ambiente real.

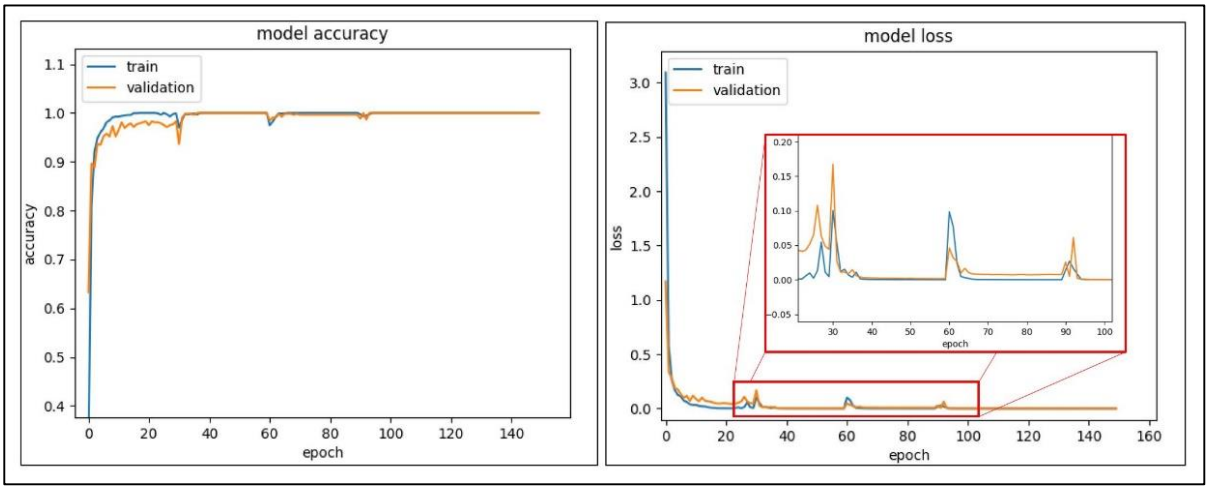


Figura 69 – *Accuracy* e *Loss* do processo de treino do modelo 3 com as imagens reais.

Experiência	1	2	3	4	5
Tempo de Treino	0:05:20	0:04:59	0:05:03	0:05:05	0:05:35
Accuracy Test	0.99	0.99	0.99	0.99	0.99

Tabela 17 – *Accuracy* dos dados de teste e tempo de treino do modelo 3 com as imagens reais.

Da Figura 69 e Tabela 17 observa-se que o modelo possui uma excelente atuação, apresentando um comportamento similar ao modelo anterior. A única diferença residiu no facto do modelo 3 necessitar de uma maior dimensão em relação ao modelo 2. Desta forma, o modelo 2 foi classificado como sendo superior, tendo como fundamento a redução das dimensões da rede neuronal.

Por último, avaliou-se o modelo 4 com as imagens reais e os parâmetros que se encontram na Tabela 15, onde a Figura 70 e Figura 71 demonstram, respetivamente, as métricas aplicada para a avaliação do treino e teste.

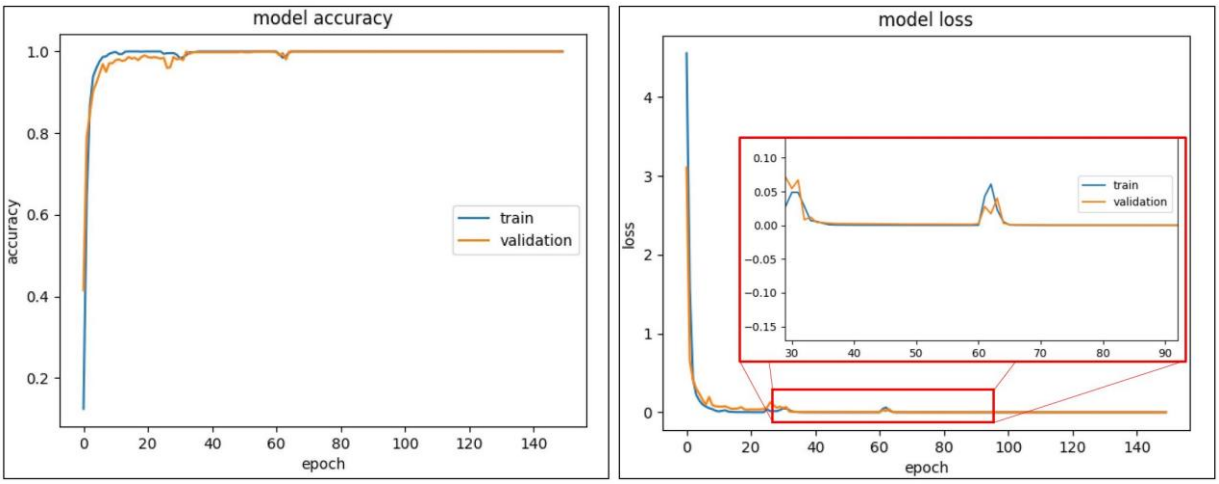


Figura 70 – *Accuracy* e *Loss* do processo de treino do modelo 4 com imagens reais.

```
Training Time:  0:34:01
Accuracy_Test:  0.9953846153846154
```

Figura 71 – *Accuracy* dos dados de teste e tempo de treino do modelo 4 com as imagens reais.

Nas figuras anteriores verifica-se que o modelo salienta um ótimo desempenho. Contudo, a partir da *epoch* 50 o seu comportamento mantém-se constante. Esta análise permitiu deduzir que o método de aprendizagem atingiu o limite máximo e o processo de treino, após a *epoch* 50, foi irrelevante. Assim sendo, o número de *epochs* aplicadas foi reduzido a 50, que tecnicamente diz respeito a 10 devido ao *K-Fold Cross Validation* com k = 5.

A Figura 72 permite examinar o processo de treino do modelo 4 com esta nova alteração.

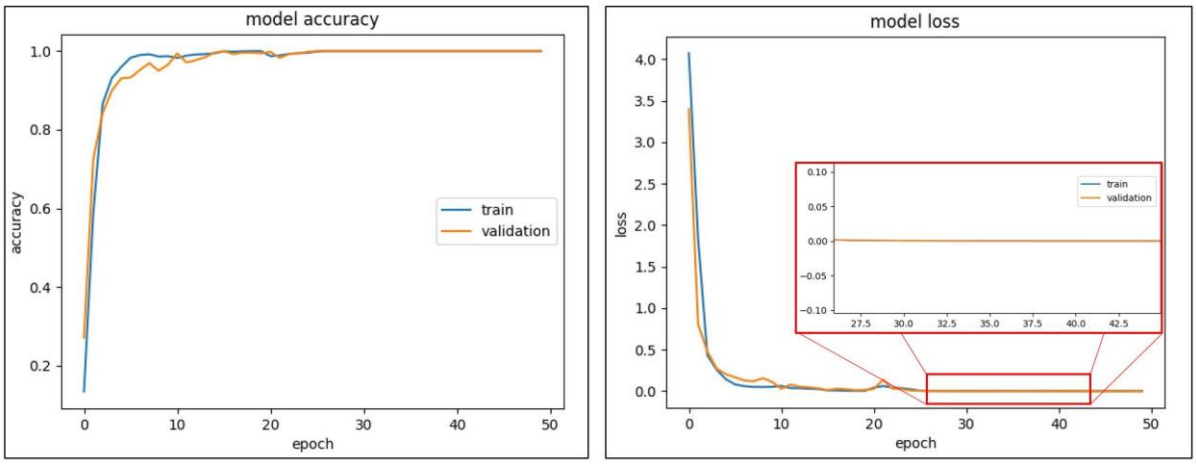


Figura 72 – *Accuracy* e *Loss* do processo de treino do modelo 4 com as imagens reais e epochs =10.

O processo de aprendizagem, com a alteração do número de *epochs*, manteve-se inalterado e, desta forma, o próximo passo foi aplicar a etapa de teste cinco vezes, que se encontra caraterizada na Tabela 18.

Experiência	**1**	**2**	**3**	**4**	**5**
Tempo de Treino	0:12:22	0:13:02	0:11:10	0:11:44	0:11:23
Accuracy Test	0.998	0.997	0.997	0.997	0.994

Tabela 18 – *Accuracy* dos dados de teste e tempo de treino do modelo 4 com as imagens reais.

Com análise do processo de treino e teste confirmou-se que a previsão do modelo se encontra muito semelhante ao anterior, tendo um ligeiro aumento. Relativamente ao tempo de treino foi possível reduzir aproximadamente 66%. Esta alteração promoveu a redução do tempo de treino, sem prejudicar o desempenho do modelo.

Após a examinação de todos os modelos desenvolvidos, concluiu-se que o modelo 4, com as imagens reais em escala de cinzentos e 10 *epochs,* apresentou os melhores resultados.

4.1.4.4 Avaliação do modelo em tempo real

Nesta secção foi retratado o desenvolvimento do algoritmo cuja funcionalidade resumiu-se na avaliação do modelo em tempo real. Este processo foi importante, uma vez que o principal propósito da manga de calibração é a previsão correta da deteção do toque em tempo real, de forma a utilizar os resultados obtidos para treinar a manga de teste. Caso este modelo apresentasse um desempenho insatisfatório, o processo de aprendizagem da manga de teste seria ineficiente.

Assim sendo, a Figura 73 exibe o fluxograma que expõe todas as etapas estruturadas para esta análise.

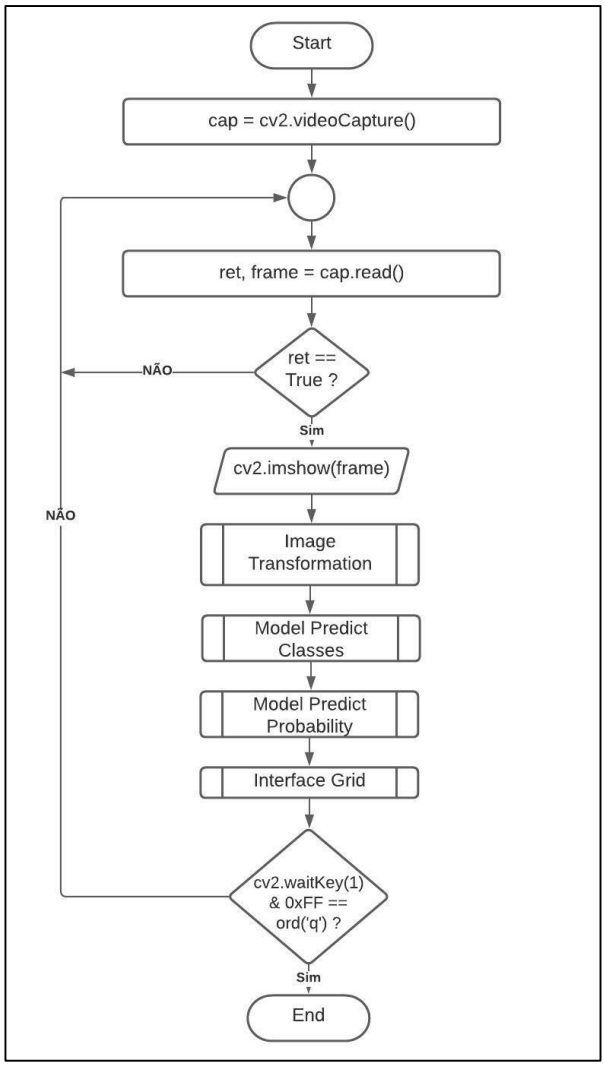


Figura 73 – Fluxograma para o processo de previsão da manga de calibração em tempo real.

Este algoritmo começa pela inicialização das câmaras, onde surge a etapa de leitura das imagens. Este procedimento permite obter um vetor de matrizes de imagens capturadas denominado de *frame* e possibilita receber uma variável booleana designada por *ret*, que proporciona conhecer se o vetor se encontra disponível. Caso a variável *ret* retornar verdadeira, o algoritmo segue os próximos passos. Caso contrário, volta novamente ao processo de aquisição das *frames*. Após o bloco de decisão, prossegue-se à visualização das *frames* de ambas as câmaras e é efetuado o processamento das imagens. De seguida, surge o processo de previsão das classes e da respetiva probabilidade. Por fim, decorre a visualização de uma interface gráfica desenvolvida com o intuito de facilitar a observação da deteção da secção. Esta interface consiste numa tabela onde é representada as 64 secções, onde a classe prevista só varia de cor na tabela quando a probabilidade é superior a 80%. Além disso, imprime a probabilidade obtida bem como a classe prevista. O grau de tonalidade da área que retrata a previsão da classe varia consoante a probabilidade resultante, sendo que a maior tonalidade carateriza o valor numérico máximo.

A Figura 74 apresenta a interface gráfica desenvolvida.

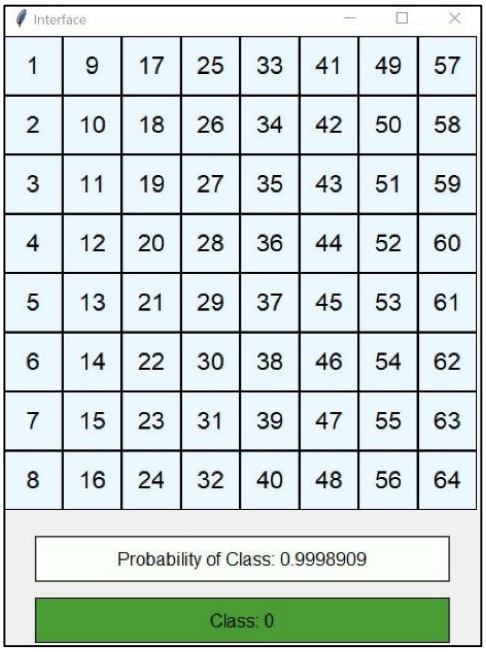


Figura 74 – Interface gráfica da manga de calibração.

Após a descrição de todos os processos necessários, procedeu-se à respetiva avaliação do modelo 4 em escala de cinzentos que se encontra retratada através de um vídeo concebido, cujo link é: https://youtu.be/Wr0C8LR_1AM . Este vídeo possibilita visualizar o toque presente na manga e observa-se as *frames* pertencentes às câmaras em tempo real bem como a interface gráfica que exibe a saída da rede neuronal. Apesar de o vídeo demonstrar o seu funcionamento, as próximas figuras comprovam o correto comportamento do modelo.

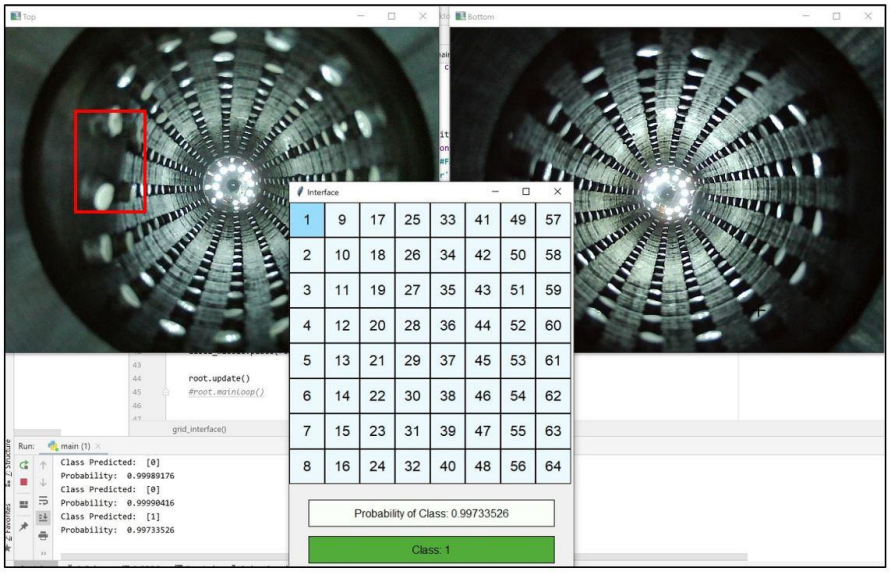


Figura 75 – Classificação em tempo real da classe 1 na manga de calibração.

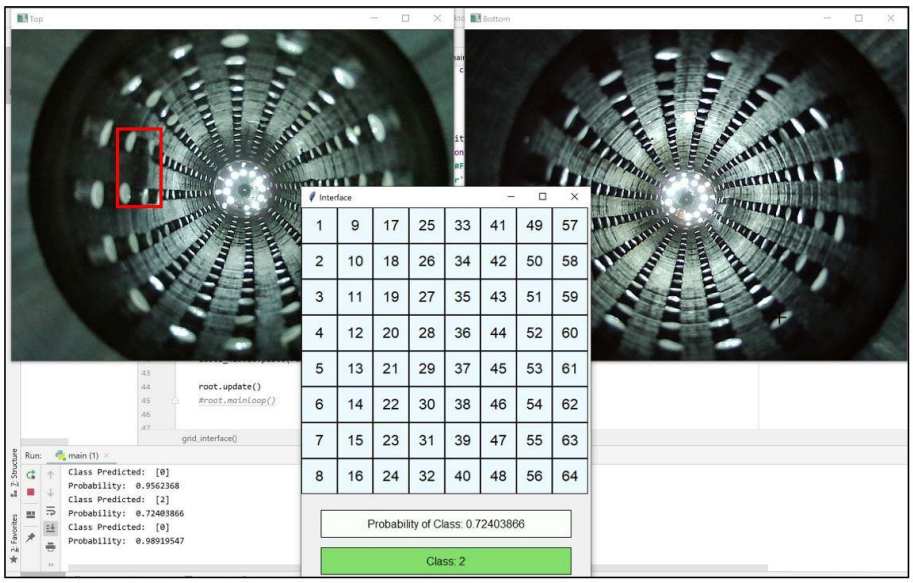


Figura 76 – Classificação em tempo real da classe 2 na manga de calibração.

Em ambas as figuras, verifica-se que a sua classificação corresponde à realidade. Contudo, na Figura 76, o toque é tão subtil que o modelo tem dificuldades em prever o local, apresentando uma probabilidade muito inferior à da Figura 75. Devido a esta discrepância, a interface que representa a classe 2 apresenta uma menor tonalidade, comparativamente com a da classe 1. Além disso, a classe não é evidenciada na tabela devido ao facto de a implementação deter uma limitação na probabilidade. Isto é, quando a probabilidade é inferior a 80%, o algoritmo não sinaliza a classe na tabela. Caso a probabilidade seja superior, a classe é indicada, como demonstra na Figura 75.

4.2 Manga de Teste

A manga de teste é o protótipo final que permitiu capacitar os robôs colaborativos a detetar colisões. No entanto, até chegar ao modelo idealizado, foi necessário seguir um conjunto de processos. A primeira baseou-se no desenvolvimento de protótipos com diferentes tipos de sensores, permitindo analisar o seu desempenho e, desta forma, selecionar o mais apropriado. Todos estes protótipos foram desenvolvidos em silicone e os sensores foram embebidos. A utilização deste material deveu-se ao facto de se tratar de uma substância que apresenta propriedades vantajosas para esta implementação, como já foi previamente referido no Capítulo 2. A segunda etapa consistiu na validação do modelo, onde foram estruturados alguns testes com o intuito de concluir se esta implementação seria viável. Por fim, avaliou-se o modelo estruturado em tempo real, sendo considerado o método mais eficiente para analisar o funcionamento da rede neuronal.

4.2.1 Protótipo da manga

A manga de teste apresenta embebido um conjunto de sensores que permitem extrair valores numéricos. Estes dados possibilitam mapear a deteção do toque e a profundidade de deformação. Contudo, como se trata de um método novo, houve a necessidade de fazer um conjunto de testes que comprovassem a sua eficácia. Sendo assim, desenvolveu-se protótipos com vários tipos de sensores, escolhendo o que possui melhores resultados.

De forma a classificar os modelos, foram elaborados alguns testes que tiveram como finalidade perceber qual o raio máximo de deformação de cada sensor, se era possível desenvolver uma equação que relacione a distância pressionada com o valor obtido no sensor e, por fim, se havia alguma correlação entre a distância e a força com que foi pressionado.

Como este método utiliza redes neuronais, deixa de ser útil a delineação de equações matemáticas que permitem caraterizar os sensores, sendo este um dos principais objetivos desta dissertação. No entanto, foram na mesma desenvolvidas estas avaliações de forma a observar analiticamente o comportamento dos sensores, sendo que os subcapítulos seguintes descrevem o sensor utilizado bem como os resultados obtidos.

4.2.1.1 Manga com sensores resistivos

Inicialmente, foi desenvolvida a manga com os sensores resistivos. Este equipamento varia a resistência quando há uma pressão exercida, sendo que, à medida que surge um aumento da força aplicada, o valor da resistência diminui.

A Figura 77 demonstra dois sensores embebidos na manga de silicone que apresentam uma área de deteção circular com um diâmetro de 12,7 milímetros.

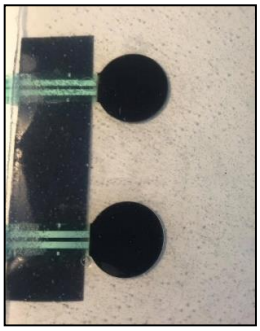


Figura 77 – Manga de teste com sensores resistivos.

Para analisar a variação da resistência, foi implementado um divisor de tensão que se encontra retratado na Figura 78. Este processo consistiu em colocar duas resistências em série num circuito elétrico,

permitindo obter uma tensão na resistência variável, denominada por tensão de saída. Além disso, possibilitou limitar a tensão de saída entre 0 e o valor de tensão de entrada.

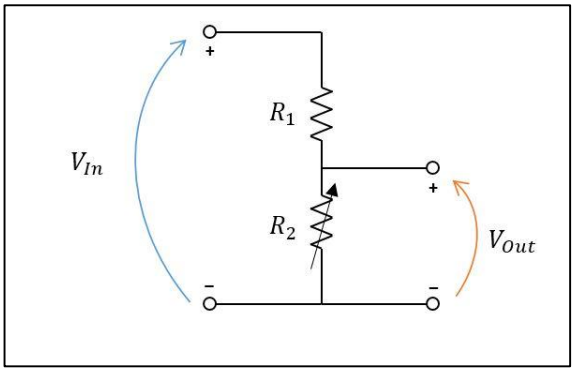


Figura 78 – Divisor de tensão.

Na Figura 78 V_{In} corresponde à tensão de entrada, R_1 carateriza a resistência fixa, R_2 representa o sensor que tem o princípio de funcionamento de uma resistência variável e, por fim, V_{Out} simboliza a tensão obtida nos terminais do sensor.

A tensão de saída está relacionada com a tensão de entrada pela seguinte equação.

$$V_{out} = \frac{R_2}{R_1 + R_2} \times V_{In} \tag{24}$$

Como o sensor resistivo R_2, segundo o *datasheet*, apresenta um intervalo de variação de 250Ω a 6kΩ, tendo um intervalo de força aplicada compreendido entre 100g e 10Kg, foi necessário aplicar uma R1 elevada de forma a que o circuito apresentasse uma maior sensibilidade. Para efetuar a leitura, foi utilizada a entrada ADC do Arduino Uno, medindo a queda de tensão no sensor.

A Figura 79 apresenta o circuito esquemático que foi implementado para testar este protótipo, onde foi utilizado o ambiente de simulação *TinkerCad*. Este ambiente consiste numa ferramenta online de software da *Autodesk* que permite criar circuitos em 3D.

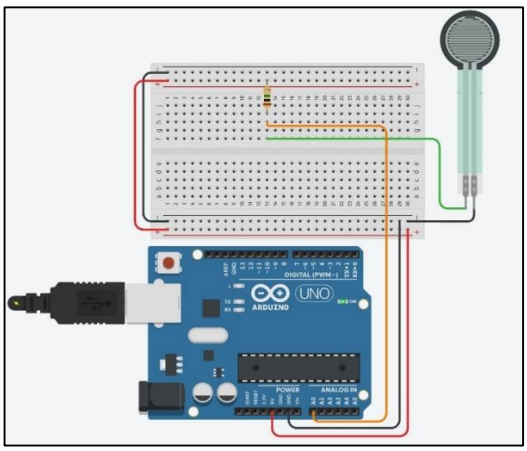


Figura 79 – Circuito esquemático da manga de teste com 1 sensor resistivo.

Após a configuração do circuito, sucedeu-se a realização dos testes, onde foram aplicadas 3 resistências diferentes em R_1. Estas resistências apresentaram um valor nominal de 2MΩ, 1MΩ, 100kΩ. Através da sua análise, concluiu-se que, com o aumento da resistência, o circuito apresentava uma maior sensibilidade. Contudo, para valores acima de 1MΩ, o circuito chegava a um patamar em que os resultados variavam significativamente, ao ponto de não ser possível verificar uma ambiguidade entre a força exercida e o valor numérico obtido no sensor. Sendo assim, foi escolhida a resistência de 1MΩ por ser considerada a mais adequada.

De seguida, efetuou-se outro teste que consistiu em adquirir o raio máximo de deteção. Este ensaio foi importante na medida em que permitiu averiguar a inviabilidade desta implementação. Isto porque, ao observar os resultados obtidos, verificou-se que o sensor apenas detetava quando o toque era efetuado por cima da sua área circular, sendo que, as restantes secções pertencentes à manga, não identificavam o toque. Como o objetivo primordial era a sua deteção em toda a sua superfície da manga, este método deixou de ser útil e passou a ser necessário a estruturação de um novo protótipo que não apresentasse estas contrapartidas.

4.2.1.2 Manga com sensores piezoelétricos

Os sensores piezoelétricos são utilizados para detetar toques ou vibrações, convertendo essa grandeza física num sinal elétrico. Esta classe, teoricamente, serve perfeitamente para o que é pretendido. Sendo assim, começou-se pelo desenvolvimento do circuito esquemático dos sensores piezoelétricos, como demonstra a Figura 80. Esta imagem carateriza apenas um sensor piezoelétrico, mas o modelo prático apresenta quatro sensores. No entanto, neste esquema é referenciado apenas um devido ao facto de o circuito ser igual para os restantes.

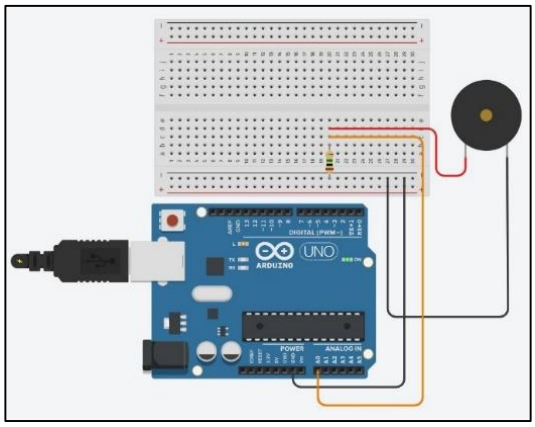


Figura 80 – Circuito esquemático da manga de teste com sensor piezoelétrico.

Na Figura 80 observa-se que o sensor se encontra ligado a uma entrada ADC. Como os sensores piezoelétricos apresentam polarização, foi necessário ter em atenção a forma como são ligados. Sendo

assim, o fio vermelho corresponde ao pino com maior tensão, enquanto que o preto carateriza o de menor tensão. Desta forma, o vermelho foi interligado à entrada ADC e o preto à terra. Por fim, encontra-se, em paralelo com o sensor, uma resistência de 1MΩ de modo a limitar a corrente e a tensão produzida pelo sensor piezoelétrico para salvaguardar a entrada analógica.

De seguida, avançou-se para a etapa de construção da manga, onde foi obtido o produto final que se encontra apresentado na Figura 81.

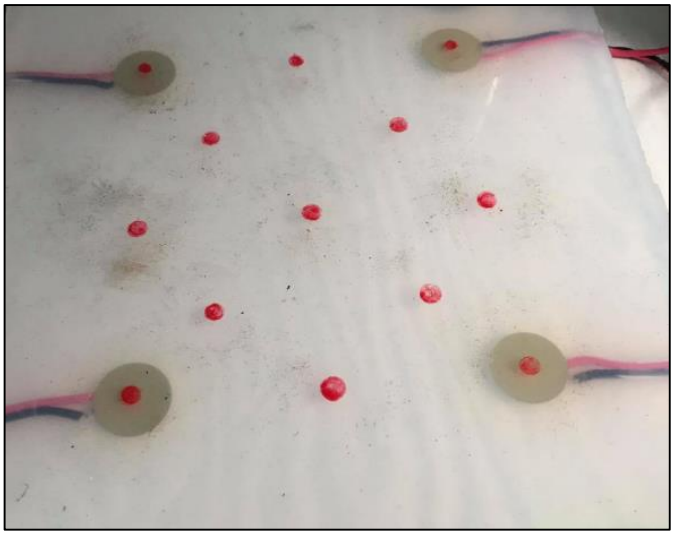


Figura 81 – Manga de teste com 4 sensores piezoelétricos e um conjunto de marcações.

Na Figura 81 visualiza-se quatro sensores piezoelétricos, cuja referência é (7BB-20-6L0), que apresentam um diâmetro de 2cm. A manga desenvolvida consiste num quadrado com 20cm de lado e uma espessura de 1cm, onde observa-se um cujo de pontos a vermelho na manga. Estes 13 marcadores foram implementados, tendo como referência a sua distância a cada um dos sensores, para desenvolver testes que permitissem obter a equação geral dos sensores. Estas equações não foram úteis para a implementação final. No entanto, foram estruturadas para observar o comportamento dos sensores.

A Tabela 19 representa o teste desenvolvido, onde foi obtido o valor da tensão de cada um dos sensores quando surge o toque em cada marcador.

	V	cm	V	cm	V	cm	V	cm
Posição	**Sensor_1**	**d_1**	**Sensor_2**	**d_2**	**Sensor_3**	**d_3**	**Sensor_4**	**d_4**
1	2,67	0	0,22	11	0,36	15	0,12	15
2	0,51	6	0,83	5	0,29	12	0,1	11
3	0,09	11	2,45	0	0,13	10	0,11	10
4	0,99	3,5	0,21	8,5	0,15	8,5	0,1	11,5
5	0,2	9	0,63	3	0,17	12	0,11	8,5
6	0,7	5	0,14	12	0,62	5	0,19	12
7	0,28	7,5	0,48	7	0,61	8	0,18	7,5
8	0,16	12,5	0,39	5	0,23	12,5	0,58	5
9	0,27	8,5	0,07	11,5	0,76	4	0,5	8,5
10	0,23	12	0,15	8,5	0,22	9,5	1,14	3
11	0,09	10	0,07	15	1,62	0	0,44	11
12	0,13	11,5	0,01	11,5	0,77	6	0,68	5
13	0,08	15	0,08	10	0,03	11	1	0

Tabela 19 – Valor da tensão de cada um dos sensores e a respetiva distância a cada um dos marcadores.

O toque aplicado em cada um dos marcadores foi realizado com dedo, tendo em atenção a força exercida, uma vez que se pretende exercer a mesma intensidade em todos os marcadores. Apesar de este teste não ser muito preciso, foi o suficiente para observar a funcionalidade dos sensores.

A Figura 82 retrata a distância de cada um dos sensores em função da respetiva distância.

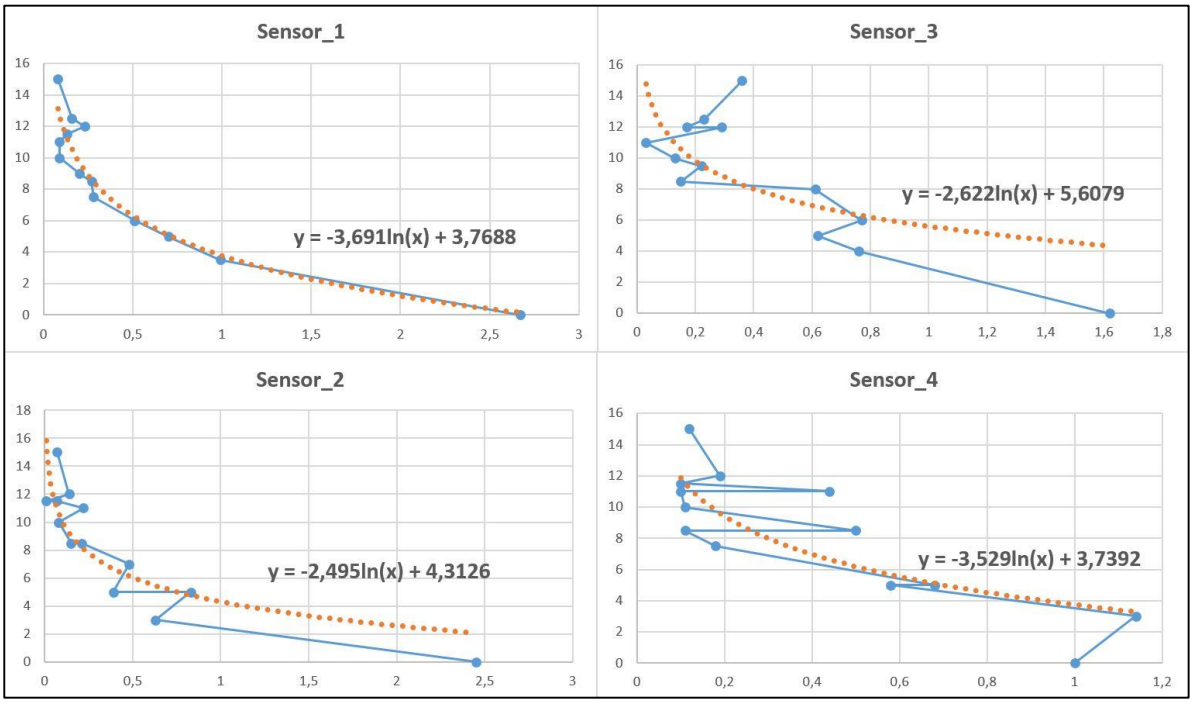


Figura 82 – Gráficos da distância em função da tensão de cada um dos sensores.

Nos gráficos apresentados na Figura 82 visualiza-se que o sensor 1 detém uma linha de tendência muito semelhante à do gráfico obtido. No entanto, os outros 3 exibem uma discrepância acentuada. Estas distinções são bastante interessantes, uma vez que todos os sensores pertencem à mesma família e, teoricamente, a sua funcionalidade deveria ser muito similar. Uma das possíveis explicações para esta irregularidade baseia-se no facto de os sensores serem muito sensíveis e o ruído interferir bastante na precisão da medição. Outro motivo pode ser o facto de os sensores se encontrarem embebidos no silicone.

Apesar de os resultados obtidos não serem os expectáveis, este teste permitiu avaliar a utilidade dos sensores, visto que apresentaram uma área de deteção em toda a manga. Sendo assim, concluiu-se que foi complexo delinear o comportamento analítico dos sensores piezoelétricos. Contudo, com a utilização do modelo de *Supervised Learning*, esta adversidade deixaria de ser relevante. Isto porque o modelo vai treinar e modelar as relações entre as entrada e saídas de dados que não são lineares nem complexas.

4.2.2 Validação do modelo

Após a seleção dos sensores, decorreu a fase de validação do modelo. Como o objetivo primordial era a deteção da área do toque, dividiu-se a manga em várias secções.

O protótipo da manga apresenta quatro sensores e foi dividida em cinco secções, como demonstra a Figura 83.

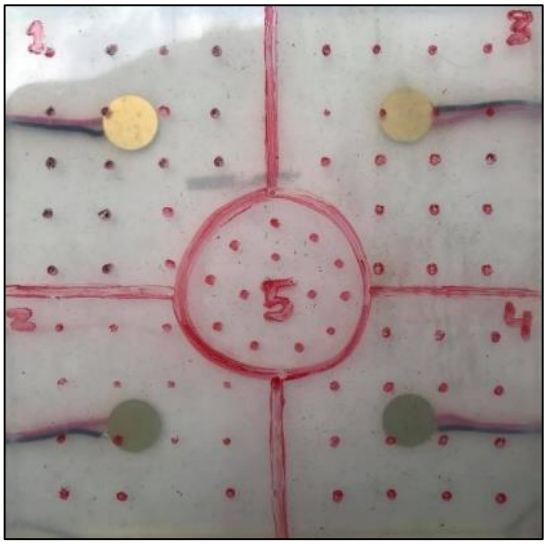


Figura 83 – Manga de teste com os sensores piezoelétricos dividida em 5 secções.

Na Figura 83 observa-se que a área de secção 5 se encontra no centro dos sensores. Esta secção foi estrategicamente definida de forma a verificar se os sensores apresentavam sensibilidade suficiente para detetar naquela região, que foi caraterizada como sendo a mais difícil de detetar. Foi também adicionado um conjunto de pontos para que, posteriormente, sejam efetuadas as devidas avaliações.

Esta fase apresentou três etapas. Começou-se pelo processo de extração dos dados, onde foi desenvolvido um software em Arduino para adquirir os valores de tensão dos sensores. De seguida, desenvolveu-se o modelo, onde foram apresentados todos os processos, desde a leitura dos dados armazenados no Excel até à fase de treino e avaliação. Por fim, elaborou-se a análise do modelo, onde foi testado o modelo com dados que são desconhecidos.

4.2.2.1 Extração dos dados

Este processo foi bastante importante, na medida em que possibilitou treinar o modelo com dados referenciados, desde os valores de tensão presentes nos sensores bem como a área da secção onde ocorreu a deformação, permitindo relacionar as entradas com as respetivas saídas.

A Figura 84 representa o fluxo de trabalho para o processo de aquisição de dados que foi aplicado para treinar o algoritmo de *Machine Learning*.

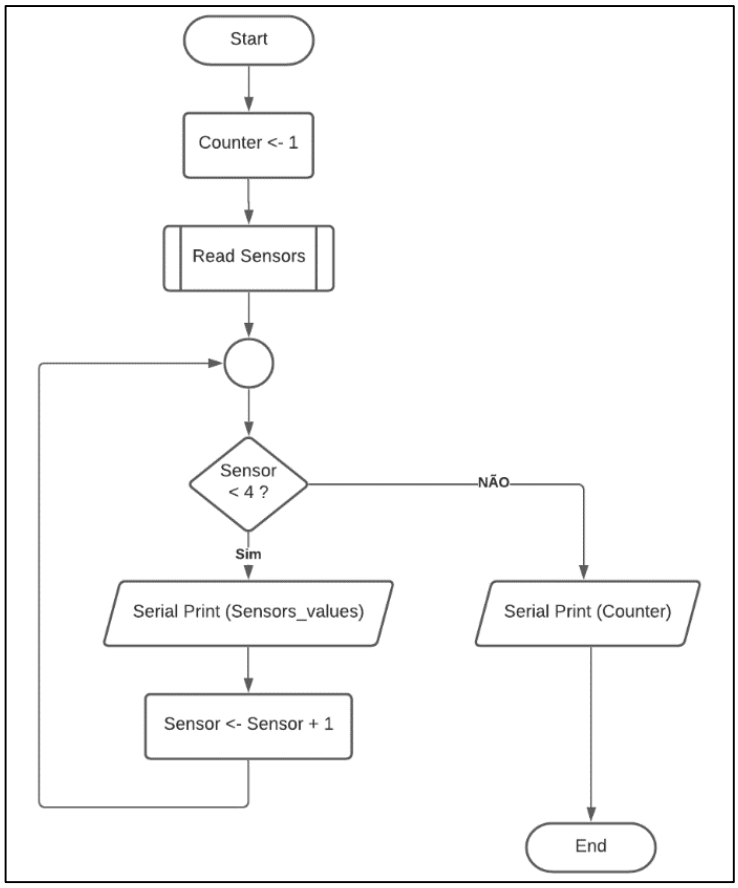


Figura 84 – Fluxograma do processo de aquisição de dados da manga de teste.

Como foram precisas 4 entradas ADC para extrair os valores numéricos dos sensores, foi selecionado o Arduino Uno. Este microcontrolador baseia-se no ATmega328P e apresenta 6 portas ADC de 10 bits [58]. Como as entradas são conversores analógicos-digitais de 10bits, o valor da tensão seria compreendido num intervalo de 0 a 1023. No entanto, como se pretendeu saber o valor analógico, foram convertidos os 4 valores digitais para os respetivos valores analógicos entre 0 a 5V. Esta conversão utiliza a seguinte equação.

$$tensão_{analógica} = \frac{tensão_{digital} \times 5}{1023} \quad (25)$$

Após efetuar o teste, verificou-se que a gama de intervalos obtida era pequena e fazia com que muitas das vezes os sensores apresentassem um valor numérico de 0V. Desta forma, alterou-se o intervalo de valores para uma resolução superior, onde a equação anterior sofreu uma transformação. O valor da tensão digital, juntamente com a gama de 0 a 5V, foi multiplicado por 200, como demonstra a equação 26. Com esta implementação, o modelo passou de um valor mínimo de tensão nos sensores de 4,9mV para 0,98V.

$$tensão_{analógica} = \frac{(tensão_{digital} \times 5) \times 200}{1023} \quad (26)$$

Após a leitura dos sensores, ocorreu o envio dos valores da tensão pela porta série para serem inseridos no modelo. A velocidade de transmissão de dados via porta série foi definida a 9600 bits por segundo, sendo uma das mais usuais. No final deste processo, foi também enviada a contagem de leituras efetuadas.

A Figura 85 apresenta um fluxograma com todas as etapas do processo de leitura dos sensores.

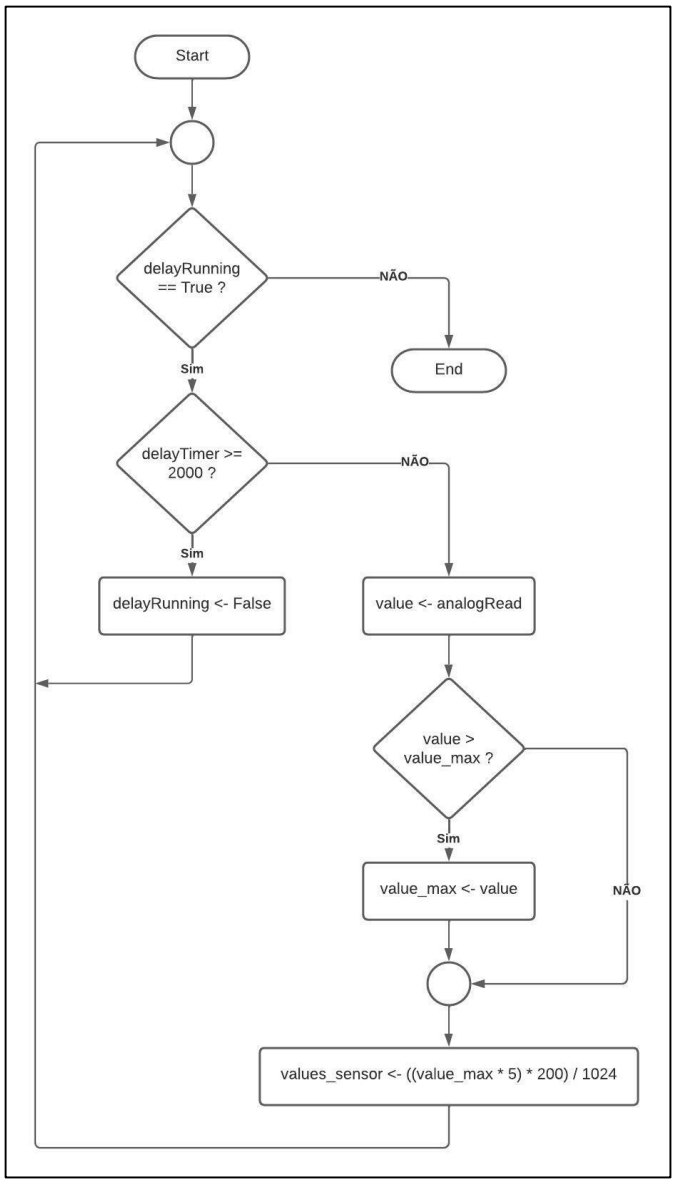


Figura 85 – Fluxograma do processo de leitura dos sensores.

Na Figura 85 observa-se que o processo de leitura dos sensores contém uma referência denominado de *delayRunning*. Enquanto a referência permanecer a verdadeiro, o programa continua a decorrer. Caso a referência se torne falsa, o programa automaticamente termina. Este método teve como propósito ler os

dados fornecidos pelos sensores durante um determinado período de tempo, visto que o propósito principal deste processo foi obter os melhores dados para utilizar no processo de treino. O intervalo de tempo implementado foi de 2000 milissegundos, de forma a que fosse possível aplicar mais do que um toque e extrair os dados com melhores resultados. Sendo assim, o *delayTimer* contabiliza a diferença entre o tempo atual e o tempo em que iniciou esta função. Enquanto que o *delayTimer* for inferior a 2 segundos, o programa extrai os valores de tensão dos sensores, compara entre o valor lido anteriormente e o atual, guarda o maior valor e, por fim, converte o valor digital em analógico.

Após todas estas etapas, as informações existentes nos sensores foram extraídas para o Excel, sendo que foram armazenadas 910 leituras.

4.2.2.2 Desenvolvimento do modelo

Esta etapa consistiu na obtenção e processamento dos dados para serem inseridos na entrada do algoritmo de *Machine Learning*, onde decorreu a criação, treino e avaliação do modelo. Todas estas etapas encontram-se mencionadas no fluxograma demonstrado na Figura 86.

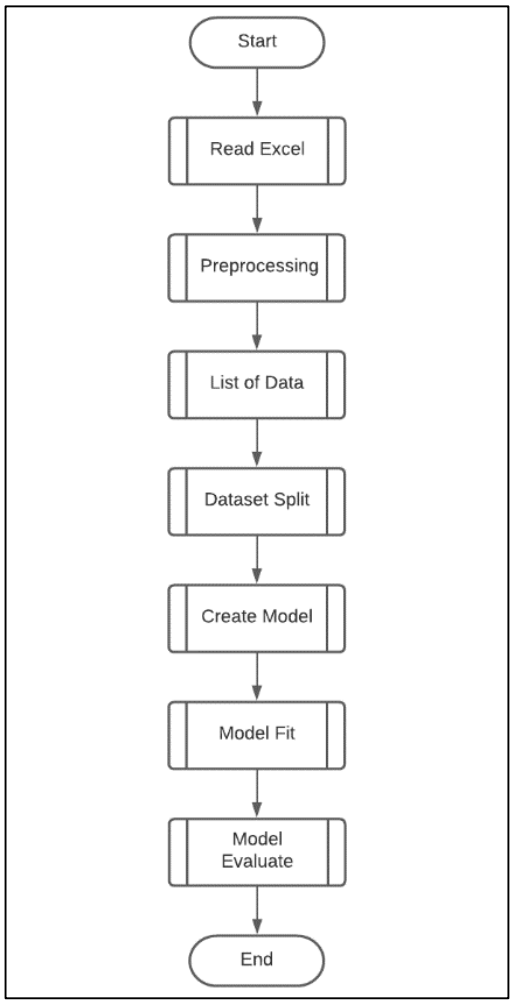


Figura 86 – Fluxograma do desenvolvimento do modelo.

Na Figura 86 visualiza-se que a execução do programa se inicia pela aquisição dos dados retratados no Excel. A segunda fase carateriza o pré-processamento, onde os dados foram normalizados para um valor compreendido entre 0 e 1, de modo a facilitar o processo de treino do modelo. Nesta implementação foi extraído o valor máximo existente no conjunto de dados e, de seguida, foi dividido por todo o conjunto. A etapa seguinte corresponde ao agrupamento de toda a informação anterior. Os valores de tensão foram inseridos numa lista, enquanto que, a área da secção foi colocada numa lista diferente. Tanto os valores como a área da secção foram inseridos na mesma posição nas diferentes listas, permitindo facilitar o seu acesso. Logo após, decorreu a divisão dos dados, onde 80% foram definidos como sendo dados de treino e os restantes 20% representaram os dados de teste. A quinta etapa retrata a criação do modelo, onde foram definidas as camadas a utilizar bem como a sua dimensão. Como se trata de dados sequenciais, as CNNs deixaram de ser a rede neuronal mais adequada, uma vez que são mais apropriadas para o tratamento de dados espaciais, como é o caso das imagens. Assim sendo, a Figura 87 representa o modelo que constitui uma ANN, sendo que, comparativamente com os modelos anteriores, só apresenta *denses layers*.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                320
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 2,598
Trainable params: 2,598
Non-trainable params: 0
```

Figura 87 – Estrutura do modelo da manga de teste.

Na Figura 87 observa-se 3 *dense layers*. As duas primeiras apresentam a função de ativação *ReLU* e a última, que retrata a camada final, contém a função *Softmax*.

A próxima etapa do desenvolvimento do modelo simboliza o processo de treino, onde foram estruturados três testes que permitiram selecionar os melhores parâmetros. A primeira análise consistiu em observar o desempenho do modelo em função do *learning rate*.

A Tabela 20 representa o número do teste com base num determinado valor de λ.

Número do Teste	1	2	3
λ	0.01	0.001	0.0001

Tabela 20 – Análise dos diferentes *learning rates* para o modelo de teste.

A Figura 88 e Figura 89 exibe os resultados obtidos no processo de treino.

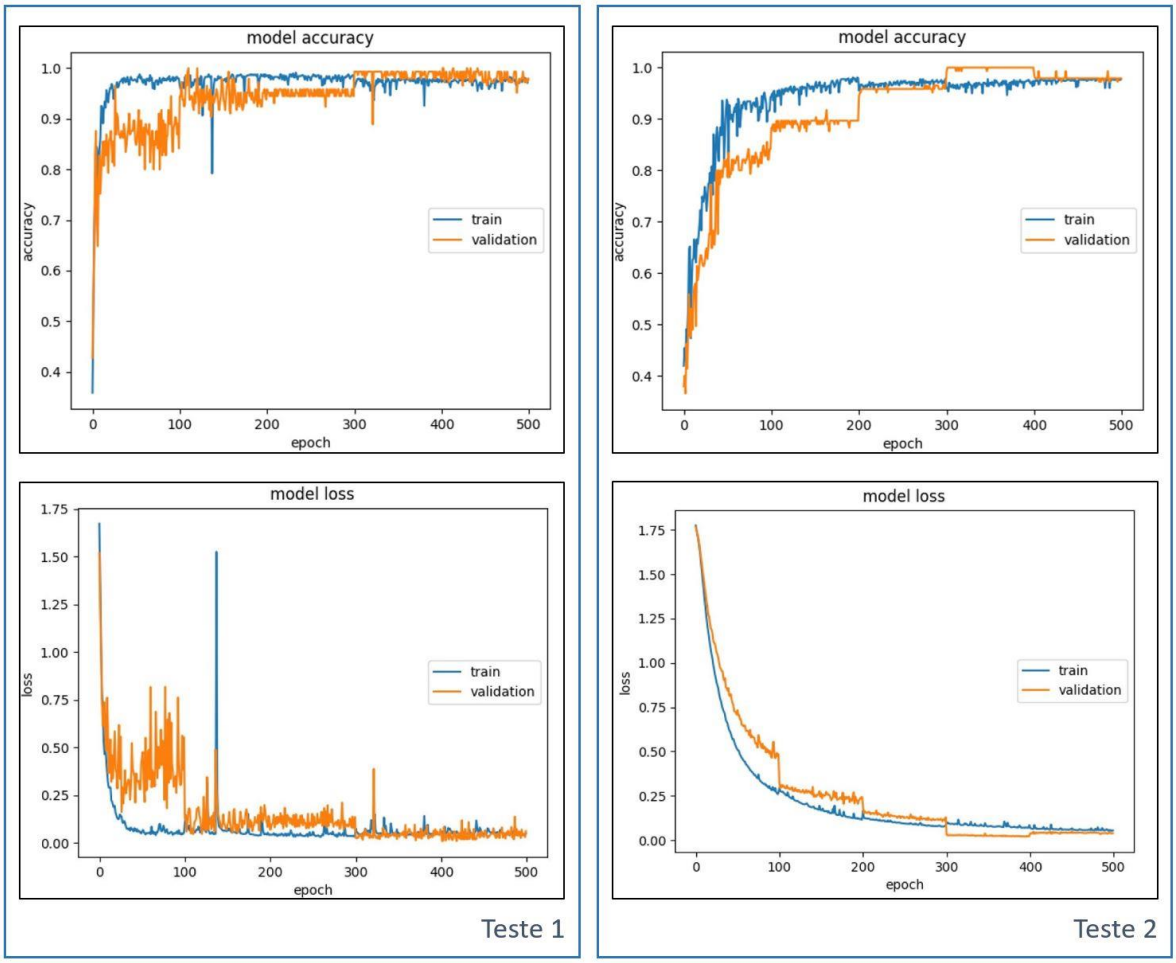


Figura 88 – Análise do desempenho do teste 1 e 2 em função do *learning rate*.

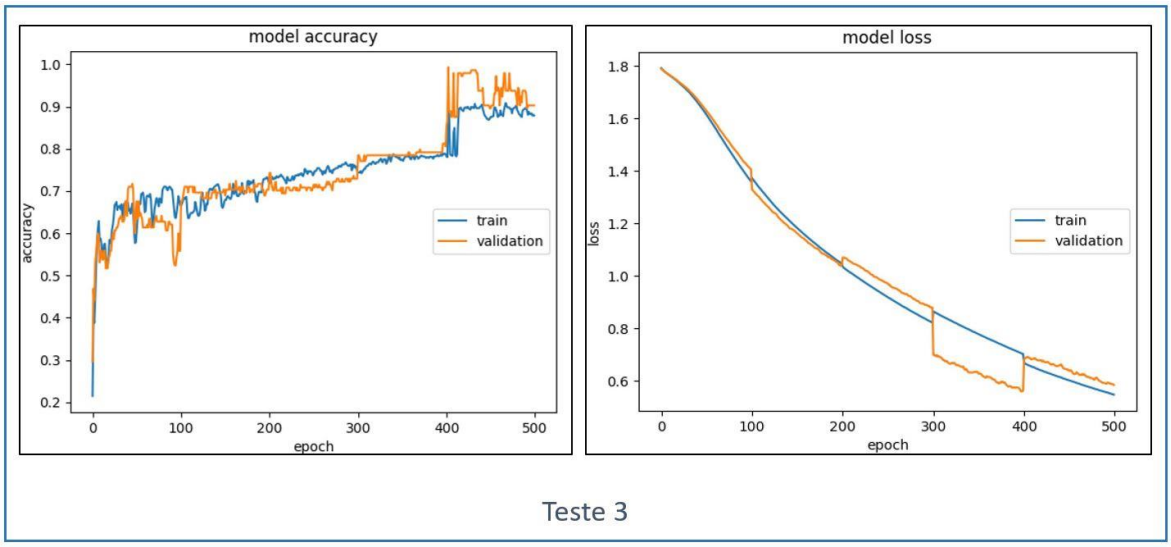


Figura 89 – Análise do desempenho do teste 3 em função do *learning rate*.

Nas figuras anteriores verifica-se que no teste 1 o modelo demonstra uma aprendizagem instável, uma vez que ocorrem várias oscilações ao longo das *epochs*. Esta instabilidade resulta de um *learning rate*

alto que promove transições de estado do gradiente descendente muito elevadas, realizando um reajuste dos *weights* e *bias* muito significativo. Os gráficos retratados no teste 2 caraterizam um *learning rate* adequado. Isto porque a *loss function* converge suavemente para zero, sendo que, nas *epochs* iniciais, apresenta uma maior acentuação na transição. Posteriormente, há uma estabilidade onde não existe variações consideráveis ao erro obtido. O teste 3 exibe uma curvatura pouco acentuada da *loss function* ao longo das *epochs*. Isto quer dizer que o modelo manifesta um *learning rate* baixo, resultando num tempo de treino mais longo. Este acréscimo traduz-se numa perda de tempo e recursos.

Para a escolha do número de *epochs*, foi utilizado o gráfico da *loss function* do teste 2. Nessa avaliação foi aplicado um número de *epochs* igual a 100, que através do *K-Fold* com k=5 traduziram-se em 500.

A Figura 90 retrata a *loss function* do teste 2.

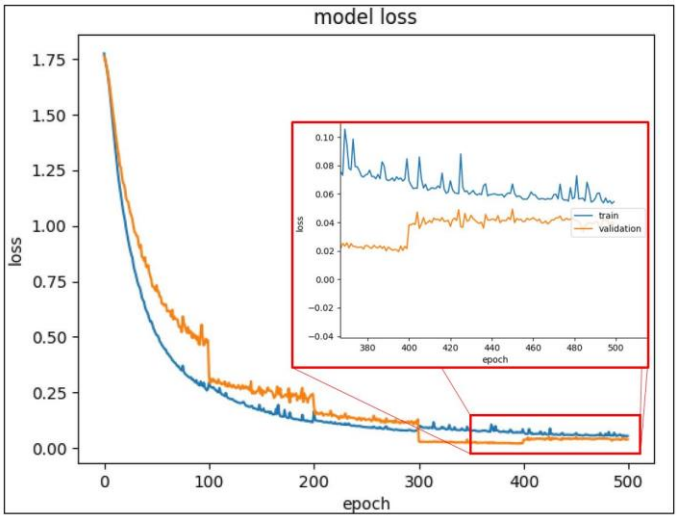


Figura 90 – *Loss function* para 100 *epochs*.

Do gráfico anterior observa-se que a *loss* converge e estabiliza a partir das 300 *epochs*. No entanto, na *epoch* número 400 surge um aumento considerável da *loss* com os dados de validação, enquanto que a dos dados de treino continua a diminuir. Este fenómeno é denominado de *overfitting* e consiste no reajuste intenso dos *weights* e *bias* aos dados de treino, tornando-se ineficaz a prever novos dados. Desta forma, foi selecionado o número 350 do gráfico, que na realidade corresponde a 70 *epochs*, e foi efetuado um teste comparativo que se encontra representado na Figura 91.

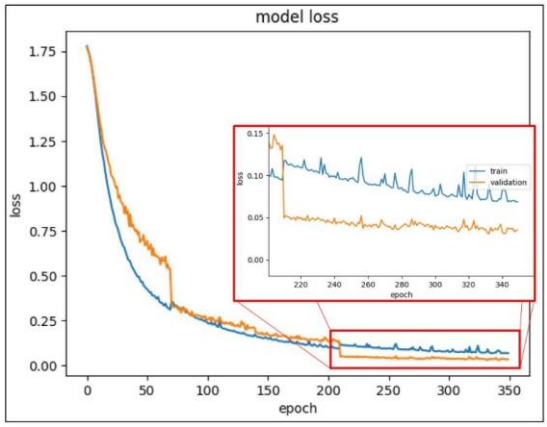


Figura 91 – *Loss function* para 70 *epochs*.

Para o ajuste do *batch size*, foi definida a gama de valores utilizada na simulação, sendo que a Figura 92 evidencia os testes efetuados.

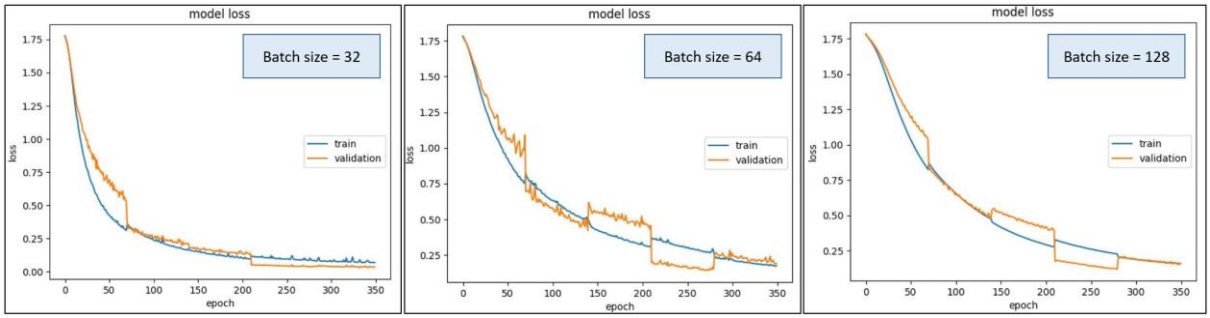


Figura 92 – *Loss function* com diferentes *batch sizes* para manga de teste.

Na Figura 91 certifica-se que o modelo salienta um melhor desempenho no processo de treino quando o *batch size* corresponde a 32. Esta conclusão foi realizada pela análise dos gráficos, tendo como critério de seleção a acentuação da curvatura.

Sendo assim, o modelo final apresenta os seguintes parâmetros reguláveis do modelo final caraterizados na Tabela 21.

Parâmetros	*Learning rate*	*Epochs*	*Batch size*
Valor	0.001	70	32

Tabela 21 – Parâmetros reguláveis atualizados para o modelo da manga de teste com os dados reais.

Por fim, a última etapa do desenvolvimento do modelo consistiu na sua avaliação com base nos dados extraídos pelo *Excel*. Estes dados foram inseridos na entrada do modelo sem qualquer tipo de referenciação. Desta forma, o modelo desconhecia a sua classificação e possibilitou avaliar a sua previsão. Posteriormente, houve a comparação da classificação prevista e a real, utilizando a métrica

designada por *accuracy.* Este indicador encontra-se caraterizado na Tabela 22, juntamente com o tempo de treino.

Experiência	**1**	**2**	**3**	**4**	**5**
Tempo de Treino	0:00:17	0:00:17	0:00:17	0:00:17	0:00:17
Accuracy Test	0.978	0.978	0.973	0.984	0.945

Tabela 22 – *Accuracy* dos dados de teste e tempo de treino do modelo da manga de teste.

Analisando as informações retiradas do modelo, averiguou-se que o modelo apresentou um ótimo desempenho.

4.2.3 Avaliação do protótipo em tempo real

A informação utilizada anteriormente para o processo de treino e teste do modelo foi proveniente de um pequeno conjunto de dados. Estes dados foram extraídos e agrupados com base em referências presentes na manga, sendo que cada referência correspondeu a uma posição específica. Este processo permitiu validar o modelo e, consequentemente, certificar que o sistema funciona corretamente. No entanto, a pequena estrutura de dados pode não ser o suficiente para caraterizar todas as particularidades do modelo. Sendo assim, surgiu a necessidade de desenvolver uma avaliação em tempo real que permitisse analisar se o modelo caraterizava corretamente o toque, utilizando dados que não apresentassem qualquer tipo de controlo. Esta verificação possibilitou comprovar que, tocando aleatoriamente na superfície da manga, o modelo detetou e o classificou acertadamente.

A Figura 93 retrata todos os procedimentos empregues para implementar a análise.

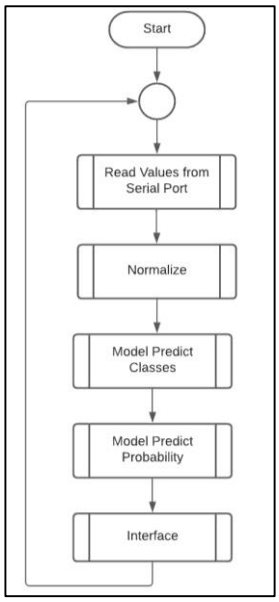


Figura 93 – Fluxograma do processo de avaliação em tempo real da manga de teste.

O teste concebido iniciou-se pela aquisição dos valores de tensão nos sensores que foram enviados pela porta série e, logo após, normalizados e inseridos na entrada do modelo. De seguida, sucedeu-se a etapa em que o modelo previu a classe e a respetiva probabilidade. Por fim, foi exibido um bloco representativo de um processo pré-definido que retrata a interface gráfica. Esta implementação teve como finalidade melhorar a análise do modelo com a representação da manga com as respetivas secções. Nesta manga observou-se a variação da tonalidade de cada secção, tendo como base a probabilidade obtida. Caso a probabilidade fosse superior a 80%, a respetiva secção apresentava uma maior tonalidade, comparativamente com as restantes. Caso contrário, esta previsão não era evidenciada no retrato da manga, mas apenas na parte inferior da janela de visualização. Nessa área foram apresentados dois blocos que caraterizaram a classe prevista bem como a sua probabilidade em tempo real, sem qualquer filtro.

A Figura 94 exibe a interface gráfica bem como alguns exemplos demonstrativos.

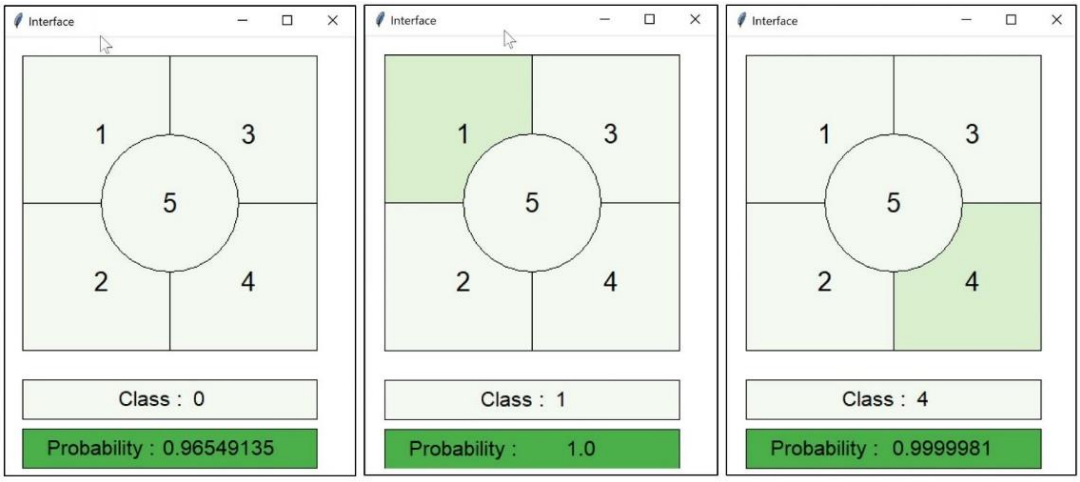


Figura 94 – Exemplos de avaliações em tempo real do protótipo da manga de teste.

O processo posterior ao seu desenvolvimento foi avaliar os resultados obtidos com base nos dados expectáveis. Esta análise foi documentada num vídeo, cujo link é: https://youtu.be/Wr0C8LR_1AM.

Através dos resultados obtidos, comprovou-se que o modelo apresenta um desempenho satisfatório, apesar de na região 5 surgirem algumas dificuldades por parte do algoritmo de prever corretamente. No entanto, como se tratou de uma prova de conceito, o modelo atendeu às expectativas.

4.2.4 Estrutura em tempo real

Após a implementação e validação do protótipo, prosseguiu-se para a estruturação da manga de teste em tempo real, onde foram abordados os tópicos principais que permitiram desenvolver este sistema, sendo eles a sua construção e o circuito esquemático de aquisição de dados.

4.2.4.1 Construção da manga

Esta secção consistiu na conceção da manga de teste, tendo como fundamento a manga de calibração. O principal objetivo foi utilizar o conhecimento adquirido da rede neuronal da manga de calibração para treinar a manga de teste.

A manga de teste foi configurada com as mesmas dimensões presentes na Figura 15 e Tabela 3. A principal diferença residiu nos marcadores internos e na espessura da manga. A manga atual não necessitou dos marcadores internos, uma vez que não utilizou este método para mapear a sua superfície. Além disso, com a adição dos sensores ao silicone, tornou-se necessário aumentar a sua espessura. Esta nova medida foi selecionada, baseando-se no protótipo desenvolvido anteriormente, que deteve uma espessura de 10 milímetros. Desta forma, foram fabricados uns novos moldes que apresentaram as mesmas nomenclaturas. A unidade M1, T2 e T3 sofreram transformações no seu tamanho de forma a proporcionar esta nova espessura. Contudo, ao objeto M1 foi adicionado no seu interior um relevo que permitiu delinear as secções, tornando este processo mais rigoroso. A peça N1 manteve-se com as mesmas dimensões, mas deixou de ter o formato dos marcadores embebidos. Por fim, a unidade T1 foi descontinuada deste sistema, uma vez que este novo processo não utilizou Visão por Computador.

A Figura 95 apresenta a manga de teste com os 16 sensores piezoelétricos que foram colocados na extremidade de cada 4 secções, permitindo detetar em toda a superfície.

Figura 95 – Manga de teste final com os sensores piezoelétricos.

4.2.3.2 Circuito esquemático de aquisição de dados

Este subcapítulo consistiu na constituição do circuito que permitiu extrair os dados da nova manga de teste. Comparativamente com os protótipos desenvolvidos, este circuito foi praticamente igual. A única distinção esteve no microcontrolador aplicado. Isto porque a nova manga apresentou 16 sensores, sendo que o Arduino Uno, utilizado nos processos anteriores, detém seis entradas. Sendo assim, avançou-se para a substituição do Arduino Uno para o Mega, uma vez que possui todas as portas analógicas necessárias para este processo. Apesar de haver esta pequena alteração, o circuito esquemático ilustrado na Figura 30 e o fluxograma da Figura 85 mantiveram-se constantes.

5. MODELO FINAL

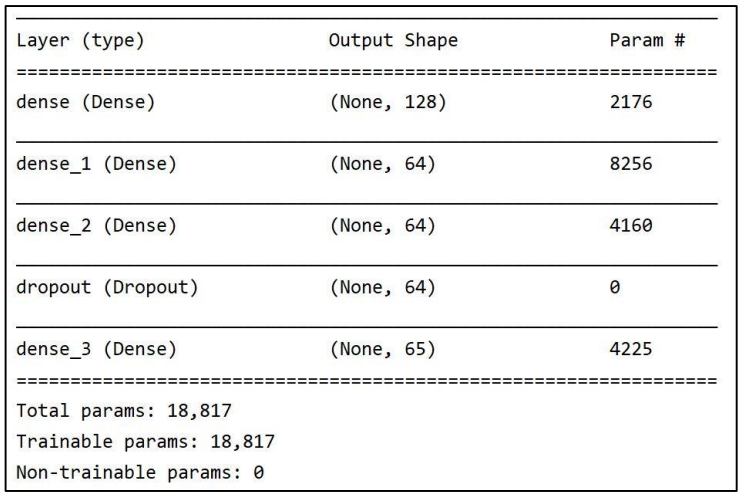
Nesta secção foram abordados os modelos aplicados para a manga de teste, sendo que ao longo deste projeto foram vários os métodos elaborados de forma a viabilizar esta estrutura.

5.1 Manga de Teste com estrutura rígida

Inicialmente, o conceito principal englobava a implementação de uma manga maleável na estrutura do CHARMIE. Contudo, ao longo do processo de desenvolvimento deste modelo, surgiu a ideia de utilizar uma estrutura rígida no interior da manga de teste de forma a simular o seu braço. Esta idealização manifestou-se quando a manga se encontrava no processo final de desenvolvimento, uma vez que, após a aplicação do silicone, foi necessário um tempo de descanso, sendo que a parte interna necessitou de um tempo superior à externa. Sendo assim, foi possível retirar o molde externo, porém o interno manteve-se durante uma semana. A manga não se encontrava 100% concluída, mas estava funcional.

Posto isto, avançou-se para o processo de aquisição de dados e o respetivo desenvolvimento do modelo. Esta arquitetura baseou-se no protótipo estruturado anteriormente para observar a viabilidade dos sensores, onde apresentou somente na sua constituição *dense layers*. O processo de extração de dados também se manteve similar, onde se retirou o valor máximo dos sensores num determinado período de tempo, que neste caso foi de 500 milissegundos. Durante este intervalo de tempo, foi aplicada uma limitação, de forma a retirar o ruído proveniente de quando não surgia o toque. Este processo consistiu na soma de todos os valores numéricos dos sensores, sendo que se fosse inferior a 48, descartava a respetiva leitura. O valor numérico 48 surgiu da visualização do seu comportamento, considerando a correta leitura quando o valor era superior a 3 em cada sensor. Como este modelo apresentou 16 sensores, a sua soma correspondeu a 48.

Para cada secção foram retiradas 50 leituras devidamente classificadas constituindo um conjunto de dados com uma dimensão de 3250. Após esta aquisição, prosseguiu-se para o desenvolvimento do modelo. Para este modelo foram utilizadas quatro *layers*, como retrata a Figura 96.



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	2176
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 64)	4160
dropout (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 65)	4225

Total params: 18,817
Trainable params: 18,817
Non-trainable params: 0

Figura 96 – Arquitetura da rede da manga de teste rígida.

Comparativamente com o protótipo presente na Figura 87, as dimensões das *dense layers* aumentaram. Este aumento deveu-se ao facto de esta implementação constituir mais dados. Além disso, foi adicionada uma *hidden layer* de forma a melhorar o comportamento do modelo e o número de saídas passou a 65. O *dropout* foi aplicado na segunda *hidden layer*, uma vez que sem este método a rede apresentava *overfitting*.

O conjunto de dados foi dividido duas vezes. A primeira formou os conjuntos de dados de teste e validação, sendo esta divisão exatamente igual às anteriores. A segunda caraterizou a formação dos dados de validação que permitiu observar o comportamento do modelo no processo de aprendizagem. Esta divisão também foi igual às restantes.

Para a delineação dos parâmetros como o *learning rate*, *epochs* e *batch size* foram utilizados os mesmos processos que já se encontram documentados previamente. Esta escolha foi baseada no processo de tentativa-erro, onde os valores aplicados foram os mais usuais para este tipo de aplicações.

A Tabela 23 demonstra os valores finais inseridos no modelo.

Learning rate	***Epochs***	***Batch Size***
0,002	70	10

Tabela 23 – Parâmetros reguláveis para a manga de teste de estrutura rígida.

Após a composição da rede neuronal, decorreu o processo de treino e teste, demonstrado nas figuras seguintes.

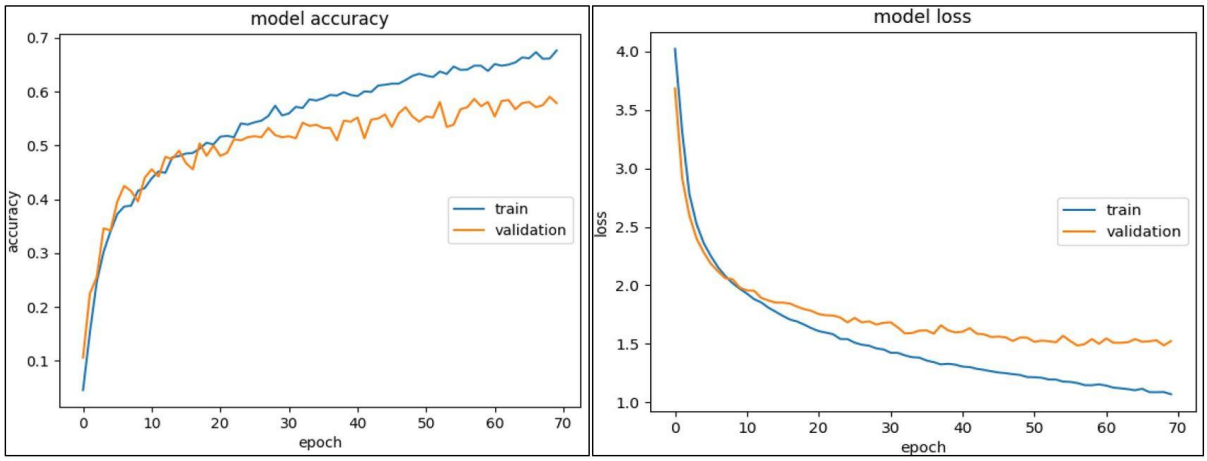


Figura 97 – *Accuracy* e *Loss* do processo de treino para o modelo da manga de teste com estrutura rígida.

```
Training Time:  0:01:30
Accuracy_Test:  0.5769230769230769
```

Figura 98 – *Accuracy* dos dados de teste e tempo de treino do modelo da manga de teste com estrutura rígida.

Na Figura 97 observa-se que o conjunto de dados de validação não consegue convergir para um valor próximo de zero e, desta forma, a rede apresenta um desempenho insatisfatório, como retrata a Figura 98. Analisando o conjunto de dados, verifica-se que os sensores apresentaram valores muito ambíguos, onde várias secções contêm dados muito similares, tornando o modelo incapaz de as diferenciar. Além disso, o facto de existir zonas da manga ainda inacabadas pode ter tido influência, dado que existiram certas regiões que possuíram melhores resultados, como demonstra a Figura 99.

Classe 1-16	Classe 33-48
Accuracy_Test: 0.7470588235294118	Accuracy_Test: 0.5294117647058824

Figura 99 – *Accuracy* dos dados de teste do modelo da manga de teste com estrutura rígida para certas classes.

Apesar de os resultados obtidos poderem revelar que esta implementação poderia exibir melhores resultados, este modelo foi considerado inapropriado para o propósito final porque foi aplicada uma força considerável para que os sensores detetassem. Como a principal finalidade do CHARMIE é evitar colisões perigosas, este método deixou de ser viável.

Ainda que esta estrutura tenha sido considerada inadequada, foi implementado um novo método com o intuito de melhorar os resultados obtidos. Caso a manga maleável apresentasse maus resultados, haveria um novo método que possibilitasse contornar essa situação.

5.2 Manga de Teste com estrutura rígida e *Transfer Learning*

Ao longo deste documento foi possível comprovar que os sensores piezoelétricos apresentaram uma sensibilidade elevada. Estes sensores, apesar de pertencerem à mesma família e serem inseridos no mesmo meio, possuíram resultados distintos. A utilização de valores numéricos provenientes dos sensores para treinar uma rede neuronal foi um método considerado primitivo, uma vez que a rede necessitaria de um conjunto extensivo de dados referenciados. Ao longo do tempo o comportamento dos sensores variou, não havendo uma linearidade. Desta forma, necessitaria de documentar todas estas alterações resultando num conjunto de dados de treino consideravelmente grande. Esta grande escala provocaria um aumento do tempo e recursos computacionais necessários para o processo de aprendizagem. Esta limitação pode ser contornada com a utilização do *Transfer Learning*, onde os conhecimentos adquiridos do processo de treino da manga de calibração são utilizados para uma segunda tarefa, que neste caso consiste no treino da manga de teste. Para a aplicação deste método, é necessário ter alguns fundamentos tais como domínio e tarefa. Neste contexto, o domínio da manga de calibração é constituído pelas imagens provenientes das câmaras, enquanto que o da manga de teste consiste nos valores numéricos obtidos nos sensores. Apesar de terem diferentes domínios, possuem a mesma tarefa que corresponde à classificação do toque.

Este novo modelo propôs potencializar a CNN desenvolvida na manga de calibração para treinar a manga de teste. Para isso, foi essencial efetuar uma transformação dos dados que viabilizou esta implementação. Isto porque a CNN foi configurada para obter na sua entrada uma imagem com uma resolução de 108x108. No entanto, a manga de teste retornou 16 valores numéricos. Esta incompatibilidade de domínios, tornaria impraticável a realização deste processo e os dados dos sensores necessitaram de ser transformados num conjunto de imagens.

É importante referir que esta implementação teve como fundamento o trabalho desenvolvido em [59], que consistiu na transformação de dados de sensores para um domínio visual, utilizando *Deep Learning*. Este modelo tem como aplicação a classificação de uma dada pessoa com base na sua pegada.

5.2.1 Transformação dos dados

Cada valor de tensão do sensor está relacionado com a pressão aplicada na manga. Com a transformação linear dos dados numa imagem de escala de cinzentos, cada conjunto de pixéis representa um sensor e a intensidade do canal retrata a pressão exercida. A intensidade necessita da conversão do valor numérico em cada sensor numa variável compreendida entre 0 a 255, servindo de

indicador da pressão no conjunto de imagens em escala de cinzentos. Esta modificação baseia-se na equação 25, onde os dados analógico-digitais são convertidos num intervalo entre 0 e 5V. No entanto, esta equação utilizou o valor numérico máximo presente em cada toque para delinear o intervalo de valores, tendo como finalidade aumentar a sensibilidade das imagens.

$$Intensidade_i = \frac{255}{\max\_value_i} \times Sensor_i \quad (27)$$

A Figura 100 apresenta a divisão equitativa dos diferentes sensores pela dimensão da imagem, que nesta implementação corresponde a uma resolução de 108x108. Cada sensor detém uma posição específica de forma a que os conjuntos de dados apresentem uma linearidade.

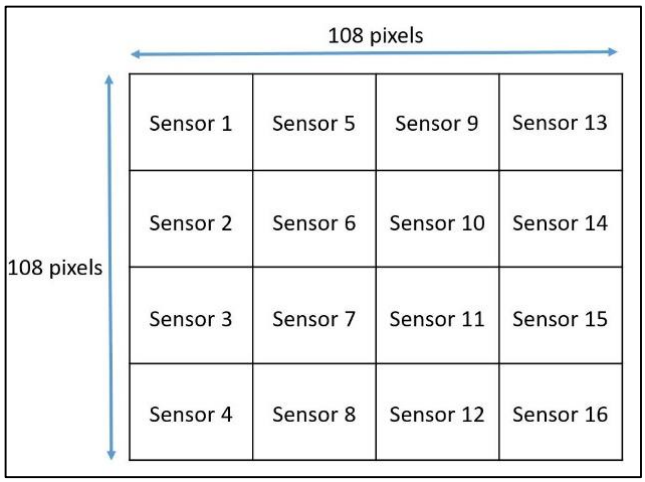


Figura 100 – Matriz transformação dos valores dos sensores em imagens.

A Figura 101 apresenta quatro imagens do conjunto de dados desenvolvido.

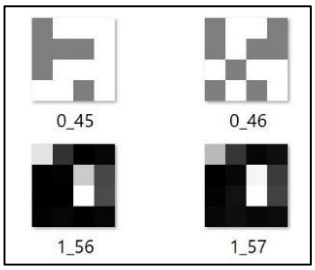


Figura 101 – Conjunto de dados transformados.

5.2.2 *Fine-Tuning*

Após o processo de transformação dos dados, decorreu a modificação do modelo pré-existente através do *fine tuning*. Este processo utilizou os *weights* de uma dada rede neuronal para programar outro processo de aprendizagem semelhante. Nesta situação concreta, consistiu na utilização da CNN estruturada e treinada para a manga de calibração, aplicando os dados da manga de teste na sua entrada. Para esta implementação manteve-se a estrutura da rede neuronal inalterada, uma vez que os

domínios apresentaram as mesmas saídas. Sendo assim, avançou-se para a próxima etapa que consistiu no congelamento de certas camadas. Esta técnica possibilitou congelar as camadas, mantendo os *weights* inalterados durante o processo de treino.

A Figura 102 demonstra a arquitetura da rede utilizada.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 106, 106, 32)	320
max_pooling2d (MaxPooling2D)	(None, 53, 53, 32)	0
conv2d_1 (Conv2D)	(None, 51, 51, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
flatten (Flatten)	(None, 40000)	0
dense (Dense)	(None, 65)	2600065

Total params: 2,618,881
Trainable params: 2,618,881
Non-trainable params: 0

Figura 102 – Arquitetura da CNN desenvolvida no modelo 4.

Na Figura 102 verifica-se que a rede apresenta 3 camadas com parâmetros treináveis, sendo duas delas a *convolutional layer* e uma *dense*. No entanto, esta última deve novamente ser treinada, dado que carateriza as respetivas saídas. O processo seguinte consistiu em congelar as camadas *convolutional* e observar o comportamento do modelo perante esta implementação.

Freeze Layer 1

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 106, 106, 32)	320
max_pooling2d (MaxPooling2D)	(None, 53, 53, 32)	0
conv2d_1 (Conv2D)	(None, 51, 51, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
flatten (Flatten)	(None, 40000)	0
dense (Dense)	(None, 65)	2600065

Total params: 2,618,881
Trainable params: 2,618,561
Non-trainable params: 320

Accuracy_Test: 0.5415384615384615

Freeze Layer 1 e 3

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 106, 106, 32)	320
max_pooling2d (MaxPooling2D)	(None, 53, 53, 32)	0
conv2d_1 (Conv2D)	(None, 51, 51, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
flatten (Flatten)	(None, 40000)	0
dense (Dense)	(None, 65)	2600065

Total params: 2,618,881
Trainable params: 2,600,065
Non-trainable params: 18,816

Accuracy_Test: 0.41384615384615386

Figura 103 – Processo de congelamento de determinadas camadas.

Na Figura 103 averiguou-se que ambas apresentaram um desempenho inferior ao modelo anterior, concluindo que esta nova metodologia não deteve qualquer tipo de vantagens em relação à estrutura anterior.

5.3 Manga de Teste maleável

O terceiro modelo foi fundamentado teoricamente tendo como princípio a união das mangas, sendo que esta implementação permitiria diminuir a complexidade do modelo da manga de teste. Esta estrutura baseava-se na aplicação das mesmas deformações a ambas, onde a classificação em tempo real da manga de calibração serviria para referenciar os valores numéricos dos sensores presentes na manga de teste. Esta referenciação permitiria classificar esses valores de forma a criar o conjunto de dados que, posteriormente, seriam utilizados para treinar e validar o novo algoritmo de *Machine Learning*. Contudo, após a estruturação das mangas, chegou-se à conclusão que a correlação mencionada não era possível, visto que as mangas apresentaram maleabilidades distintas, resultando numa deformação pouco similar. Isto deveu-se ao facto de ter sido necessário aumentar a espessura da manga de teste para colocar os sensores embebidos. A outra limitação observada consistiu no facto de os sensores piezoelétricos apenas detetarem quando surge uma deformação ou vibração, sendo que, ao aplicar uma deformação fixa, os sensores retornam uma tensão de 0V. Sendo assim, ao aplicar uma deformação constante às duas mangas, a manga de calibração classificaria corretamente a secção onde houve o toque, mas o conjunto de dados obtidos pela manga de teste seriam classificados incorretamente, visto que apresentariam um saída de 0V que, na realidade, este valor corresponde à ausência de toque. Desta forma, este modelo utilizou apenas a manga de teste e foi sustentado por quatro etapas: a aquisição dos dados relevantes, a estruturação do modelo, a análise do comportamento da rede com dados de treino/teste e, por último, a avaliação em tempo real.

5.3.1 Aquisição dos dados

A aquisição de dados foi fundamentada por três fases, como retrata a Figura 104.

Figura 104 – Processo de aquisição de dados para a manga de teste maleável.

Este processo inicializou pela aplicação da deformação da manga de teste. De seguida, obteve-se os valores dos sensores da manga de teste e a classificação da região do toque. Esta segunda etapa necessitou da realização de um conjunto de processos, dado que, apesar da utilização do valor máximo dos sensores a cada *delayTimer* de 200 milissegundos ter ajudado para a estabilidade do modelo, não

foi o suficiente para contribuir para um bom desempenho. Sendo assim, foi estruturada a média, onde os valores máximos de cada sensor durante os 200 milissegundos foram armazenados e, após três leituras, foi efetuado o devido cálculo. Nesta implementação surgiu também a conveniência de adicionar um *threshold*, visto que foi preciso fazer uma distinção entre duas categorias: contato ou a ausência de contato. Esta imposição foi importante, uma vez que se pretendeu constituir a média de três leituras do mesmo conjunto, não havendo uma combinação entre diferentes grupos. Isto porque, caso não houvesse esta restrição, tornaria os dados menos precisos. Por fim, constituiu-se a estrutura que permitiu interligar as informações anteriores, onde a média das três leituras foram referenciadas com a respetiva classificação. Por cada secção foram adicionadas 20 medições, resultando num conjunto de dados de 1300 leituras.

5.3.2 Estruturação do modelo

Este modelo foi baseado na estrutura rígida, apresentando a mesma arquitetura e dimensões retratadas na Figura 96. A única distinção residiu nos parâmetros aplicados, sendo estes demonstrados na Tabela 24.

Learning rate	***Epochs***	***Batch Size***
0,002	130	32

Tabela 24 – Parâmetros reguláveis do modelo da manga de teste maleável.

5.3.3 Análise de resultados

Após o desenvolvimento do modelo, prosseguiu-se para a análise do sistema na fase de treino e teste. Estas etapas utilizaram o conjunto de dados adquiridos anteriormente, onde ocorreu a sua divisão em subconjuntos de forma semelhante aos restantes métodos retratados.

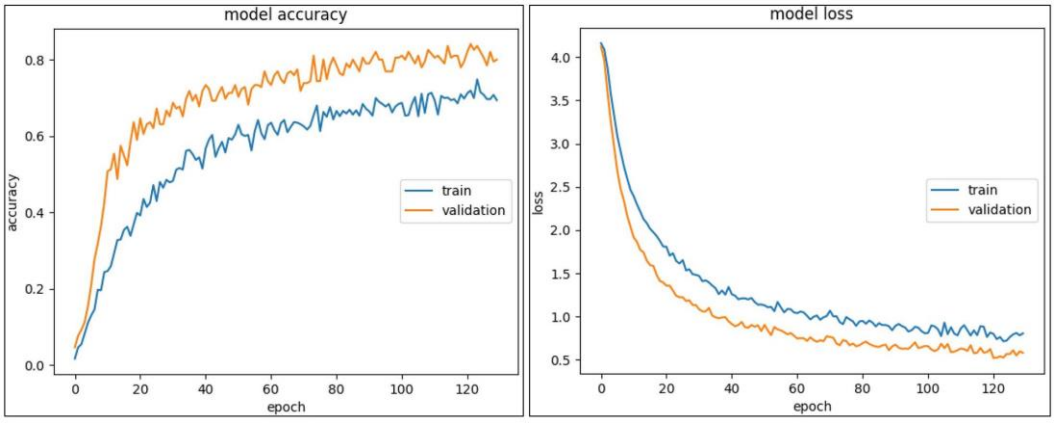


Figura 105 – *Accuracy* e *Loss* do processo de treino para o modelo da manga de teste maleável.

```
Training Time:  0:01:54
Accuracy_Test:  0.7961538461538461
```

Figura 106 – *Accuracy* dos dados de teste e tempo de treino do modelo da manga de teste maleável.

As figuras anteriores permitiram concluir que a rede neuronal possui um bom desempenho e, desta forma, dirigiu-se para a fase de avaliação em tempo real.

5.3.4 Avaliação em tempo real

A avaliação consiste em aplicar à rede neuronal os dados fornecidos em tempo real de forma a observar a classificação do modelo. Esta análise foi fundamentada com a visualização de uma interface gráfica desenvolvida com o intuito de facilitar a observação da deteção da secção, sendo documentada num vídeo exposto no link: https://youtu.be/Wr0C8LR_1AM. No entanto, as figuras seguintes demonstram a correta classificação do modelo perante um conjunto de dados específico.

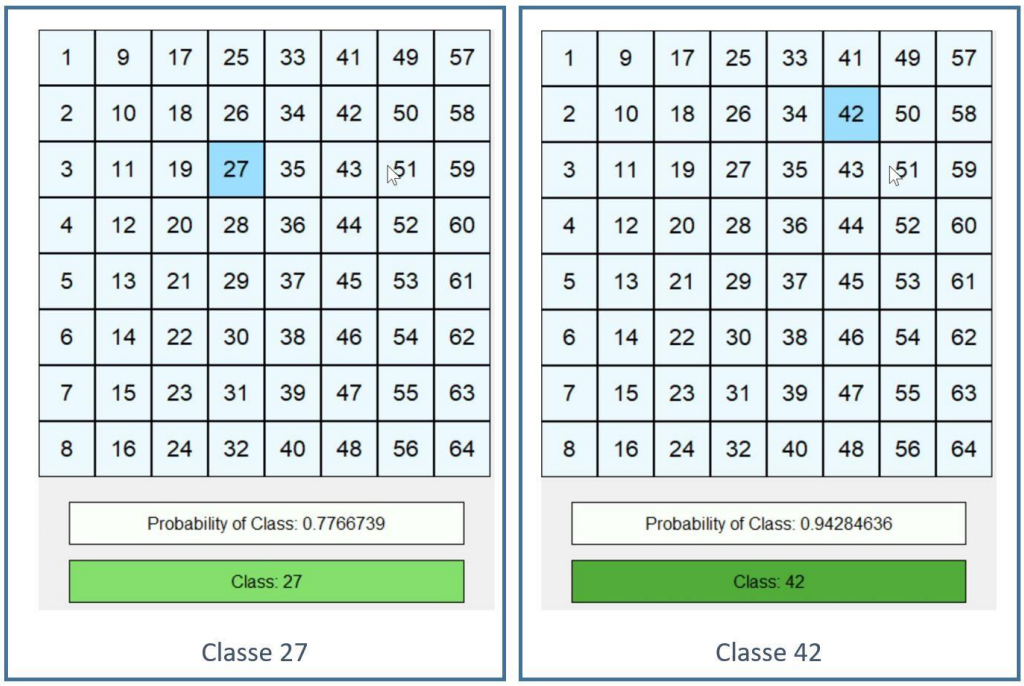


Figura 107 – Exemplos de avaliações em tempo real da manga de teste maleável.

Da análise da documentação deste sistema, avalia-se o modelo com um desempenho mediano, uma vez que representa por várias vezes a classificação da classe na região vizinha. Desta forma, concluiu-se que este método apresenta uma deteção do toque em toda a superfície da manga que constitui as secções. No entanto, a classificação do toque não é tão assertiva. Verificou-se que esta estrutura apresenta um enorme potencial, uma vez que deteta com pequenas deformações e não é necessário aplicar uma força muito intensa.

6. CONCLUSÕES

O projeto desenvolvido propôs a construção de um sistema de deteção de tato em tempo real, de forma a criar um ambiente de colaboração seguro entre os Humanos e os robôs. Este sistema pretendeu correlacionar baixo custo com eficiência, uma vez que existem inúmeras soluções idênticas na comunidade científica. No entanto, utilizam uma grande variedade de sensores, tornando o modelo dispendioso.

Para a conceção do sistema de deteção foram elaboradas duas mangas: a de calibração e a de teste, onde o modelo da manga de calibração em tempo real serviria para classificar os dados dos sensores provenientes da manga de teste. Sendo assim, iniciou-se pela configuração da manga de calibração, que foi inserida no ambiente de simulação *Cinema 4D*. Neste ambiente foram extraídos todos os dados relevantes para constituir o modelo de *Supervised Learning*, enquanto decorria o seu processo de fabricação. A partir da simulação, foram estruturados 4 modelos, sendo que o primeiro utilizou apenas as imagens de uma das câmaras, o segundo usou a sobreposição das imagens em escala de cinzento, o terceiro aplicou um método RGB para diferenciar as imagens de cada uma das câmaras, e por último, o quarto modelo aumentou a resolução das imagens. Isto porque os modelos anteriores utilizavam uma imagem com dimensão 36x36 e este modelo pretendeu verificar a influência da resolução das imagens, utilizando uma de 108x108. De todos os modelos, constatou-se que o primeiro não possuiu caraterísticas suficientes nas imagens para que o modelo pudesse distinguir as diferentes classes. Os modelos 2 e 3 permitiram comparar os diferentes métodos de processamento das imagens, verificando-se que, tanto o método em escala de cinzento como o RGB, detiveram uma *accuracy* média de 90%. No entanto, o método de escala de cinzentos possui uma rede neuronal com uma menor dimensão. Por fim, o modelo 4 exibiu a importância da resolução das imagens para evitar perder caraterísticas relevantes. Este modelo com as imagens em escala de cinzento apresentou uma *accuracy* média de 96%, sendo considerado o mais adequado. Posteriormente, com a finalização da sua construção, foi elaborada a análise em tempo real que possibilitou observar o real desempenho do modelo. Este desempenho foi classificado como bom.

A manga de teste necessitou de um conjunto de sensores que permitissem detetar o contacto em toda a região. De todos os sensores existentes no mercado, foram abordados três: o primeiro tratou-se do sensor capacitivo, que foi de imediato avaliado como inadequado pelo facto de poderem ocorrer mudanças na capacitância quando o sensor se encontra perante um material condutivo. O segundo consistiu no sensor resistivo, onde a variação da resistência é diretamente proporcional à força exercida.

Contudo, este sensor apenas detetou quando ocorreu um toque em cima da sua circunferência, provocando a ausência da deteção em grande parte da superfície da manga. O terceiro sensor analisado foi o piezoelétrico, onde a variação da deformação do material provoca uma determinada tensão nos seus terminais. Com a análise de todos os sensores, concluiu-se que apesar de apresentarem uma grande sensibilidade, os sensores piezoelétricos são os mais apropriados, visto que detetaram em toda a superfície. Desta forma, foi constituída a manga de teste com dimensões semelhantes à manga de calibração e com os sensores piezoelétricos embebidos.

Após a composição das duas mangas, decorreu o seu processo de correlação. Contudo, concluiu-se que a relação mencionada não era possível, visto que as mangas apresentavam maleabilidade distintas, resultando em deformações com pouca similaridade. Isto deveu-se ao facto de ter sido necessário aumentar a espessura da manga de teste para colocar os sensores embebidos. Uma outra limitação residiu no facto de os sensores piezoelétricos apenas detetarem quando surge uma deformação ou vibração. Caso ocorra uma deformação fixa, a manga de calibração classifica corretamente a secção onde existiu o contacto, mas o conjunto de dados provenientes dos sensores da manga de teste seriam classificados incorretamente, visto que apresentariam uma tensão de 0V, quando, na realidade este valor representa a ausência de toque.

Como deixou de ser praticável incorporar a correlação entre as duas mangas, prosseguiu-se para a composição do modelo da manga de teste individual, onde foram criados três modelos: dois utilizam a manga com uma estrutura rígida no seu interior e o último apresenta a estrutura maleável. O primeiro modelo utiliza uma ANN e foi constituído, uma vez que a manga necessitou de manter o molde interno para solidificar o seu interior. Esta conceção permitiu ter um método de comparação entre a manga rígida e a maleável. Através da análise das métricas de avaliação, concluiu-se que esta estrutura não apresenta resultados satisfatórios, detendo uma *accuracy* de 58%. Além disso, este método necessita de uma força de deteção considerável e, como o objetivo é evitar colisões perigosas, este modelo pode não ter a capacidade de detetar perante uma força que já é crítica.

Apesar da manga rígida ter sido classificada de inadequada, foi desenvolvido um segundo modelo, tendo como fundamento o *Transfer Learning.* Este mecanismo consistiu em utilizar a CNN estruturada para a manga de calibração, onde o conhecimento adquirido da CNN serviria de base para instruir a manga de teste. Contudo, este processo foi considerado ineficiente, reproduzindo um desempenho inferior ao modelo inicial, tendo uma *accuracy* máxima de 54%.

O terceiro modelo emprega a ANN desenvolvida no modelo 1 da estrutura rígida, mas com a superfície maleável. Esta implementação resultou num modelo preditivo com uma *accuracy* de 80%. Em tempo real, o modelo apresentou um desempenho mediano, representando por várias vezes as secções vizinhas.

Posto isto, este projeto desenvolveu dois sistemas de deteção individuais. A manga de calibração detém o melhor desempenho, mas necessita que a sua parte interna seja oca, tornando a sua acoplação muito limitada. A manga de teste maleável possui um desempenho satisfatório e apresenta um enorme potencial.

6.1 Trabalho Futuro

O trabalho desenvolvido ao longo deste documento tem como finalidade servir de referência para futuras implementações em algoritmos de *Supervised Learning*, mais concretamente para sistema de deteção de tato. Aplicando todos os conhecimentos teóricos retratados bem como os modelos projetados, é obtida uma ótima estrutura para iniciar um novo projeto.

Quanto ao modelo final caraterizado, surge a necessidade de introduzir novos processos, de forma a melhorar a capacidade preditiva do modelo. As melhorias planeadas para um trabalho futuro são:

- Aumento do conjunto de dados, tendo em consideração a aplicação do contato em múltiplas áreas ao mesmo tempo;
- Implementação do modelo de regressão, que possibilita ter o conhecimento da profundidade da deformação;
- Implementação de mais sensores piezoelétricos, de forma a melhorar a sensibilidade da manga.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Vysocky and P. Novak, "Human - Robot collaboration in industry," *MM Sci. J.*, vol. 2016-June, no. OCTOBER, pp. 903–906, 2016, doi: 10.17973/MMSJ.2016_06_201611.

[2] S. Grahn, B. Langbeck, K. Johansen, and B. Backman, "Potential Advantages Using Large Anthropomorphic Robots in Human-robot Collaborative, Hand Guided Assembly," *Procedia CIRP*, vol. 44, pp. 281–286, 2016, doi: 10.1016/j.procir.2016.02.036.

[3] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida, "Developing a mobile robot for transport applications in the hospital domain," *Rob. Auton. Syst.*, vol. 58, no. 7, pp. 889–899, 2010, doi: 10.1016/j.robot.2010.03.010.

[4] F. Gonçalves, T. Ribeiro, I. Garcia, F. A. Ribeiro, C. Monteiro, and G. Lopes, "Development of an anthropomorphic mobile manipulator with human, machine and environment interaction," *FME Trans.*, vol. 47, no. 4, pp. 790–801, 2019, doi: 10.5937/fmet1904790F.

[5] Plant Robert, "An introduction to artificial intelligence," *32nd Aerosp. Sci. Meet. Exhib.*, 2011.

[6] B. Marr, *Artificial Intelligence in Practice*. 2019.

[7] T. P. Trappenberg, *Fundamentals of Machine Learning*. 2019.

[8] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 8–17, 2015, doi: 10.1016/j.csbj.2014.11.005.

[9] E. G. Learned-miller, "Introduction to Supervised Learning," pp. 92–103, 2010, doi: 10.1142/9789814287319_0007.

[10] Amr E. Mohamed, "Comparative Study of Machine Learning Techniques for Supervised Classification of Biomedical Data," *Acta Electrotech. Inform.*, vol. 14, no. 3, pp. 5–10, 2014, doi: 10.15546/aeei-2014-0021.

[11] Lecun Yann, Cortes Corinna, and Burges Christopher, "THE MNIST DATABASE of Handwritten Digits," *The Courant Institute of Mathematical Sciences*, 1998. http://yann.lecun.com/exdb/mnist/ (accessed Dec. 06, 2020).

[12] S. R. and A. Annamma, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification," *Int. J. Adv. Res. Artif. Intell.*, vol. 2, 2013, doi: 10.1088/1751-8113/44/8/085201.

[13] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 16, no. 1, pp. 186–197, 2008, doi: 10.1109/TASL.2007.909282.

[14] F. Cady, "Unsupervised Learning," *Data Sci. Handb.*, pp. 135–151, 2017, doi: 10.1002/9781119092919.ch10.

[15] A. G. Sutton, Richard S.; Barto, *Reinforcement Learning, second edition: An Introduction*. .

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 19, pp. 237–285, 1996, doi: 10.1613/jair.301.

[17] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016, doi: 10.1016/j.neucom.2015.09.116.

[18] F. Chollet, *Deep Learning with Phyton*. 2018.

[19] J. Bell, “Chapter 5 - Artificial Neural Networks,” *Mach. Learn. Hands-On Dev. Tech. Prof.*, pp. 91–116, 2014.

[20] C. Gershenson, “Artificial Neural Networks for Beginners,” 2003.

[21] K. Suzuki, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. .

[22] S. Sharma, S. Sharma, and A. Athaiya, “Activation Functions in Neural Networks,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 12, pp. 310–316, 2020, doi: 10.33564/ijeast.2020.v04i12.054.

[23] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv*, pp. 1–20, 2018.

[24] A. D. K. A.D.Dongare, R.R.Kharde, “Introduction to Artificial Neural Network,” *Int. J. Eng. Innov. Technol.*, vol. 2, no. 1, 2012.

[25] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” pp. 1–11, 2015.

[26] M. Görner, “Learn TensorFlow and deep learning, without a Ph.D.,” 2017. https://cloud.google.com/blog/big-data/2017/01/learn-tensorflow-and-deep-learning-without-a-phd (accessed Dec. 21, 2020).

[27] G. H. M. Queiroz J. E. R., “Introdução ao Processamento Digital de Imagens,” 2006.

[28] M. Sarıgül, B. M. Ozyildirim, and M. Avci, “Differential convolutional neural network,” *Neural Networks*, vol. 116, pp. 279–287, 2019, doi: 10.1016/j.neunet.2019.04.025.

[29] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.

[30] D. Yu, H. Wang, P. Chen, and Z. Wei, “Mixed pooling for convolutional neural networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8818, pp. 364–375, 2014, doi: 10.1007/978-3-319-11740-9_34.

[31] J. Yang and J. Li, “Application of deep convolution neural network,” *2016 13th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. ICCWAMTIP 2017*, vol. 2018-Febru, pp. 229–232, 2017, doi: 10.1109/ICCWAMTIP.2017.8301485.

[32] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[33] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C. H. Lee, “On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression,” *IEEE Signal Process. Lett.*, vol. 27, pp. 1485–1489, 2020, doi: 10.1109/LSP.2020.3016837.

[34] Z. Wang and A. C. Bovik, “Mean squared error: Lot it or leave it? A new look at signal fidelity measures,” *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, 2009, doi: 10.1109/MSP.2008.930649.

[35] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, “Tactile sensing-from humans to humanoids,” *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 1–20, 2010, doi: 10.1109/TRO.2009.2033627.

[36] H. Zhang and E. So, “Hybrid resistive tactile sensing,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 32, no. 1, pp. 57–65, 2002, doi: 10.1109/3477.979960.

[37] S. Miyazaki and A. Ishida, “Capacitive transducer for continuous measurement of vertical foot

force," *Med. Biol. Eng. Comput.*, vol. 22, no. 4, pp. 309–316, 1984, doi: 10.1007/BF02442098.

[38] S. Wattanasarn, K. Noda, K. Matsumoto, and I. Shimoyama, "3D flexible tactile sensor using electromagnetic induction coils," *Proc. IEEE Int. Conf. Micro Electro Mech. Syst.*, pp. 488–491, 2012, doi: 10.1109/MEMSYS.2012.6170230.

[39] K. Kanao *et al.*, "Highly selective flexible tactile strain and temperature sensors against substrate bending for an artificial skin," *RSC Adv.*, vol. 5, no. 38, pp. 30170–30174, 2015, doi: 10.1039/c5ra03110a.

[40] P. Mittendorfer and G. Cheng, "Humanoid multimodal tactile-sensing modules," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 401–410, 2011, doi: 10.1109/TRO.2011.2106330.

[41] A. Alspach, J. Kim, and K. Yamane, "Design and fabrication of a soft robotic hand and arm system," *2018 IEEE Int. Conf. Soft Robot. RoboSoft 2018*, pp. 369–375, 2018, doi: 10.1109/ROBOSOFT.2018.8404947.

[42] M. Koike, S. Saga, T. Okatani, and K. Deguchi, "Sensing method of total-internal-reflection-based tactile sensor," *2011 IEEE World Haptics Conf. WHC 2011*, pp. 615–619, 2011, doi: 10.1109/WHC.2011.5945556.

[43] L. Van Duong, R. Asahina, J. Wang, and V. A. Ho, "Development of a vision-based soft tactile muscularis," *RoboSoft 2019 - 2019 IEEE Int. Conf. Soft Robot.*, pp. 343–348, 2019, doi: 10.1109/ROBOSOFT.2019.8722814.

[44] S. Yoshigi, J. Wang, S. Nakayama, and V. A. Ho, "Deep Learning-based Whole-Arm Soft Tactile Sensation," *2020 3rd IEEE Int. Conf. Soft Robot. RoboSoft 2020*, pp. 132–137, 2020, doi: 10.1109/RoboSoft48309.2020.9116018.

[45] K. S. Sudeep and K. K. Pal, "Preprocessing for image classification by convolutional neural networks," *2016 IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. RTEICT 2016 - Proc.*, pp. 1778–1781, 2017, doi: 10.1109/RTEICT.2016.7808140.

[46] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2146–2153, 2009, doi: 10.1109/ICCV.2009.5459469.

[47] C. Y. Lee, P. Gallagher, and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 863–875, 2018, doi: 10.1109/TPAMI.2017.2703082.

[48] D. Svozil, V. Kvasnička, and J. Pospíchal, "Introduction to multi-layer feed-forward neural networks," *Chemom. Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997, doi: 10.1016/S0169-7439(97)00061-0.

[49] J. P. Howard, "Interpolation and Extrapolation," *Comput. Methods Numer. Anal. with R*, no. February, pp. 95–132, 2018, doi: 10.1201/9781315120195-4.

[50] J. D. Rodríguez, A. Pérez, and J. A. Lozano, "Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 569–575, 2010, doi: 10.1109/TPAMI.2009.187.

[51] E. G. Adagbasa, S. A. Adelabu, and T. W. Okello, "Application of deep learning with stratified K-fold for vegetation species discrimation in a protected mountainous region using Sentinel-2 image," *Geocarto Int.*, 2019, doi: 10.1080/10106049.2019.1704070.

[52] Y. Y. Fang and X. J. Chen, "Design and simulation of UART serial communication module based

on VHDL," *2011 3rd Int. Work. Intell. Syst. Appl. ISA 2011 - Proc.*, 2011, doi: 10.1109/ISA.2011.5873448.

[53] "Cinema 4D." https://www.maxon.net/en/cinema-4d (accessed Oct. 17, 2021).

[54] "Pycharm." https://www.jetbrains.com/help/pycharm/quick-start-guide.html (accessed Oct. 17, 2021).

[55] "Tensorflow." https://www.tensorflow.org/ (accessed Oct. 17, 2021).

[56] "OpenCV." https://opencv.org/ (accessed Oct. 17, 2021).

[57] D. P. Kingma and J. Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *Iclr*, pp. 1–15, 2015.

[58] "Arduino Uno Rev3." https://store.arduino.cc/products/arduino-uno-rev3?selectedStore=eu (accessed Oct. 10, 2021).

[59] M. S. Singh, V. Pondenkandath, B. Zhou, P. Lukowicz, and M. Liwickit, "Transforming sensor data to the image domain for deep learning - An application to footstep detection," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017-May, no. May, pp. 2665–2672, 2017, doi: 10.1109/IJCNN.2017.7966182.