INSTITUTO POLITÉCNICO DE LISBOA

Instituto Superior de Engenharia de Lisboa
Escola Superior de Tecnologia da Saúde de Lisboa

Identificação de Padrões Geométricos e Numéricos Anómalos em Eletrocardiogramas

Beatriz de Brito Godinho

Trabalho Final de Mestrado para obtenção do grau de
Mestre em **Engenharia Biomédica**

Orientador

Doutor José Alberto de Sousa Rodrigues	ISEL

(Versão definitiva)

Novembro de 2023

Instituto Politécnico de Lisboa
Instituto Superior de Engenharia de Lisboa

Escola Superior de Tecnologia da Saúde de Lisboa

Identificação de Padrões Geométricos e Numéricos Anómalos em Eletrocardiogramas

Beatriz de Brito Godinho

Orientador

Doutor José Alberto de Sousa Rodrigues	ISEL

Júri

Presidente:	
Especialista José Pedro Fulgêncio de Matos	ISEL
Vogais:	
Doutora Maria Amélia Ramos Loja	ESTeSL
Especialista Luís Manuel Gil Martins Brizida	HFF
Doutor José Alberto de Sousa Rodrigues	ISEL

Mestrado em **Engenharia Biomédica**

Novembro de 2023

Agradecimentos

A realização e conclusão desta etapa que é o trabalho final de mestrado só foi possível com a ajuda, apoio e orientação de várias pessoas, que de forma direta ou indireta fizeram parte desta fase e que acreditaram em mim. Assim, quero estender a todos o meu sincero agradecimento.

Em primeiro lugar, ao meu orientador, o Professor Doutor José Alberto Rodrigues por todo o apoio, motivação, paciência, disponibilidade e, a cima de tudo, por todas as horas dedicadas a ajudar-me para conseguir concluir este desafio, muito obrigada.

Aos meus pais, irmão e família, pelo apoio incondicional, paciência, motivação e por acreditarem em mim nesta e em todas as fases da minha vida, é graças aos vossos ensinamentos que me tornei na pessoa que sou hoje.

Ao meu namorado, um especial agradecimento por acreditar sempre em mim, pelo seu apoio, motivação quando esta começava a ser cada vez menos, e pela força que me deu em todos os momentos para conseguir ultrapassar os obstáculos que foram surgindo, sem esquecer de celebrar todas as vitórias, por mais pequenas que fossem.

A todos os meus amigos e colegas, mas em especial à Ana Beatriz Serafim, agradeço a amizade, apoio, entreajuda e, principalmente, as palavras de encorajamento neste percurso, mais concretamente quando a motivação começava a falhar, agradeço por me terem acompanhado ao longo destes dois anos.

À SUCH, aos meus colegas de trabalho, aos enfermeiros e técnicos, destacando a minha colega Carolina Ferreira, agradeço a compreensão e apoio dado, as conversas e brincadeiras que me ajudaram a distrair quando mais precisava mas, principalmente, agradeço a oportunidade que me deram ao ajudarem-me sempre que foi necessário para que eu pudesse continuar o meu crescimento pessoal e profissional.

Não esquecendo todos os docentes que de uma maneira ou outra contribuíram para o meu percurso, nestes 2 anos desafiantes, gostaria de destacar um especial agradecimento ao Professor Doutor Pedro Ferreira, pelo tempo disponibilizado não só referente às unidades curriculares lecionadas, mas também pelas orientações durante o restante percurso académico.

Resumo

As arritmias cardíacas são uma condição referente à atividade elétrica anormal no coração e, consoante o nível patológico, poderá ter implicações graves para o indivíduo podendo, em última instância, constituir uma ameaça.

A aprendizagem profunda tem sido uma técnica importante no auxílio à classificação de eletrocardiogramas, permitindo que a deteção e o diagnóstico sejam alcançados com uma maior precisão, mesmo para as diferentes anomalias elétricas do coração. Utilizando a onduleta discreta é possível extrair as características pretendidas do sinal de eletrocardiograma para, em seguida, treinar um método de aprendizagem profunda. A onduleta discreta é uma técnica capaz de decompor o sinal em diferentes frequências, detetando informação relativa ao tempo e à frequência do sinal. Com a combinação destas duas técnicas, é possível extrair características importantes do sinal de eletrocardiograma, em diferentes escalas, que permitem a identificação precisa de doenças cardiovasculares, mais concretamente de diferentes arritmias.

A combinação da aprendizagem profunda e da onduleta discreta tem um grande potencial para expandir e avançar o campo de análise de eletrocardiogramas e permitir um diagnóstico mais eficiente e preciso de doenças cardiovasculares. Este trabalho apresenta uma visão geral dos métodos de classificação de sinais de eletrocardiogramas utilizando um método de aprendizagem profunda com a contribuição da onduleta discreta.

Palavras-Chave: Arritmias, Aprendizagem Profunda, Onduleta Discreta.

Abstract

Cardiac arrhythmias are a condition referring to abnormal electrical activity in the heart and, depending on the pathological level, can have serious implications for the individual and ultimately can be posing as a threat.

Deep learning has been an important technique in helping to classify electrocardiograms, allowing detection and diagnosis to be achieved with greater precision, even for the different electrical anomalies of the heart. Using the discrete wavelet, it is possible to extract the desired characteristics from the electrocardiogram signal and then train a deep learning method. The discrete wavelet is a technique capable of breaking down the signal into different frequencies, detecting information relating to the time and frequency of the signal. By combining these two techniques, it is possible to extract important characteristics from the electrocardiogram signal, at different scales, which allow for the precise identification of cardiovascular diseases, specifically different arrhythmias.

The combination of deep learning and the discrete wavelet has great potential to expand and advance the field of electrocardiogram analysis and enable more efficient and accurate diagnosis of cardiovascular diseases. This paper presents an overview of methods for classifying electrocardiogram signals using a deep learning method with the contribution of the discrete wavelet.

Keywords: Arrhythmias, Deep Learning, Discrete Wavelet.

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas e Siglas

Sigla	Expansão	Tradução
AI	*Artificial Intelligence*	Inteligência Artificial
AV	Auriculoventricular	
Bpm	Batimentos Por Minuto	
CNN	*Convolutional Neural Network*	Rede Neuronal Convolucional
CWT	*Continuous Wavelet Transform*	Transformada de onduleta contínua
DBN	*Deep Belief Networks*	Rede de Crenças Profundas
DL	*Deep Learning*	Aprendizagem Profunda
DWT	*Discrete Wavelet Transform*	Transformada de onduleta discreta
ECG	Eletrocardiograma	
FFT	*Fast Fourier Transform*	Transformada de Fourier Rápida
FT	*Fourier Transform*	Transformada de Fourier
ML	*Machine Learning*	Aprendizagem Automática
mm	Milímetro	
mm/s	Milímetro por Segundo	
ms	Milissegundos	
mV	MiliVolt	
OMS	Organização Mundial de Saúde	
RBM	*Restricted Boltzmann Machine*	Modelo de Boltzmann Restrito
s	Segundo	
SA	Sinusal	
SAE	*Stacked Auto-Encoders*	Codificadores Automáticos Empilhados
WT	*Wavelet Transform*	Transformada de onduleta

1. Introdução

1.1 Motivação e Objetivos

Segundo a Organização Mundial de Saúde (OMS) as doenças cardiovasculares são a principal causa de morte a nível mundial e, de acordo com a Direção Geral da Saúde, uma em cada três mortes em Portugal é por doenças cardiovasculares, correspondendo a aproximadamente trinta e cinco mil mortes por ano. [9] [11]

O eletrocardiograma (ECG) corresponde à aquisição, de forma não invasiva, dos eventos elétricos que ocorrem durante o ciclo cardíaco, em que cada um dos eventos tem uma forma de onda associada. O ECG e os estudos eletrofisiológicos, em conjunto, fornecem as ferramentas essenciais para a deteção e diagnóstico de anomalias elétricas do coração e, em seguida, monitorização da eficácia do tratamento. Uma vez que o ECG é o melhor método de avaliação da componente elétrica cardíaca, é essencial para a deteção de algumas anomalias cardíacas, tais como arritmias. [30]

A deteção e diagnóstico de algumas doenças cardiovasculares depende da extração e classificação dos batimentos cardíacos, o que requer a leitura e análise dos diferentes constituintes do ciclo cardíaco e, como o sinal não é regular na sua periodicidade, a utilização de *wavelets* permite-nos obter uma representação mais precisa das diferentes características do sinal em diferentes escalas de tempo. Com a *wavelet transform* é possível fazer o pré processamento do sinal para remover ruídos e possíveis artefactos, utilizando filtros para extrair um sinal de ECG dentro da frequência desejada, com a possibilidade de obter diferentes níveis de detalhe. O sinal obtido após a aplicação dos filtros é decomposto de forma a extrair as diversas características específicas do ECG que se pretende analisar. [1] [31]

O *deep learning* (DL), que corresponde a um subconjunto da *artificial intelligence* (AI) e do *machine learning* (ML), é um método importante para a classificação de sinais de ECG, visto que a análise manual destes sinais poderá ser demorada e estar sujeita a erros humanos. Desta forma, o DL é uma ferramenta essencial para otimizar os processos de classificação dos sinais de ECG, auxiliar na previsão de eventos e orientar na tomada de decisões. [30]

O objetivo geral deste estudo consiste em auxiliar a classificação automática de um sinal de ECG antes de uma consulta médica. Este estudo será direcionado para o indivíduo comum, que não possui conhecimentos na área, permitindo que de forma segura consiga perceber se existe algum problema cardíaco que seja necessário ser observado e, desta forma, saber se é aconselhado proceder ao agendamento de uma consulta com um médico cardiologista. De modo a realizar esta análise, numa primeira instância será necessário proceder à extração de parâmetros necessários à, posterior, avaliação do ECG para que, recorrendo a um modelo de arquitetura de DL, possa ser classificado numa das duas categorias estabelecidas, ou seja, classificando o sinal como sendo ”normal”, sem uma patologia cardíaca associada, ou como ”patológico”, tendo algum tipo de patologia cardíaca associada sem fazer distinção desta.

1.2 Organização do Documento

Este trabalho encontra-se organizado em 7 capítulos, seguidos das referências bibliográficas e de anexos. Estes podem ser divididos em duas grandes partes que separam os fundamentos teóricos da parte prática, do testes realizados e resultados obtidos.

No presente capítulo é feita uma breve introdução ao tema do trabalho e é definido o objetivo do mesmo.

No segundo capítulo é realizada uma revisão teórica da eletrofisiologia cardíaca, que constituí a base deste trabalho. Numa primeira fase é explicada a constituição do músculo cardíaco humano e o processo do seu sistema de condução. Em seguida, é explicado o eletrocardiograma, o seu funcionamento e o processo para a identificação do sistema de eixos cardíaco. É também descrito o ciclo cardíaco, onde são identificadas as diferentes ondas, segmentos e intervalos que permitem, por fim, proceder à caracterização que é esperada de um sinal de eletrocardiograma de um indivíduo saudável, sem patologia cardíaca associada.

No terceiro capítulo, após o estudo da teoria da eletrofisiologia cardíaca, procede-se à revisão da literatura, onde é feita uma revisão abrangente e crítica da pesquisa existente relacionada com a classificação de ECG com recurso a modelos de *deep Learning*.

No quarto capítulo é feita uma breve explicação quanto às *wavelets* e o papel que tiveram no desenvolver do trabalho.

No quinto capítulo é feita a contextualização sobre a *deep learning*, onde é feita a distinção entre *machine learning* e *deep learning* e entre os diferentes métodos de *machine learning*. É ainda explicado os conceitos de *overfitting* e *underfitting*.

No sexto capítulo é exposta a metodologia utilizada para o trabalho, que se encontra divido em duas etapas, a extração dos dados necessários e o processamento dos mesmos, sendo realizada a avaliação e classificação dos sinais de ECG.

Por fim, no sétimo capítulo são concluídos os resultados obtidos e apresentadas as perspetivas para trabalhos futuros, para que possa existir uma melhoria de alguns aspetos resultantes deste trabalho.

2. Eletrofisiologia Cardíaca

Neste capitulo são apresentados os conceitos fisiológicos referentes aos fenómenos elétricos associados à atividade cardíaca. Estes conceitos são fundamentais para que seja possível a correta interpretação da morfologia do ECG.

Os ECGs de superfície e os estudos eletrofisiológicos, em conjunto, fornecem as ferramentas essenciais na eletrofisiologia cardíaca, ou seja, na deteção e diagnóstico das anomalias elétricas do coração, mesmo as alterações mais subtis.[29]

2.1 O Músculo Cardíaco

O coração humano está dividido em quatro cavidades, dois átrios e dois ventrículos, sendo as suas paredes constituídas por três camadas. A espessa camada intermédia do coração corresponde ao músculo cardíaco, também conhecido como miocárdio, as camadas interna e externa são mais finas, em comparação ao miocárdio, e correspondem, respetivamente, ao endocárdio e epicárdio, como é possível observar na Figura 2.1. As artérias coronárias são responsáveis por fornecer o sangue ao músculo cardíaco, enquanto as veias cardíacas realizam a drenagem do sangue. [14][28]

Num coração saudável, o miocárdio apresenta uma contração rápida e involuntária e, em seguida, a capacidade de relaxamento de forma coordenada sem ceder à fadiga [28]. As células que constituem o miocárdio apresentam quatro propriedades fundamentais: excitabilidade, a capacidade de responder a um dado estímulo; contratilidade, surge após o estímulo e é representada pela contração muscular; automatismo, capacidade de originar o impulso que determina a contração, um exemplo é o nódulo sinusal (SA); condutividade, propagação do impulso por todo o músculo cardíaco [14].

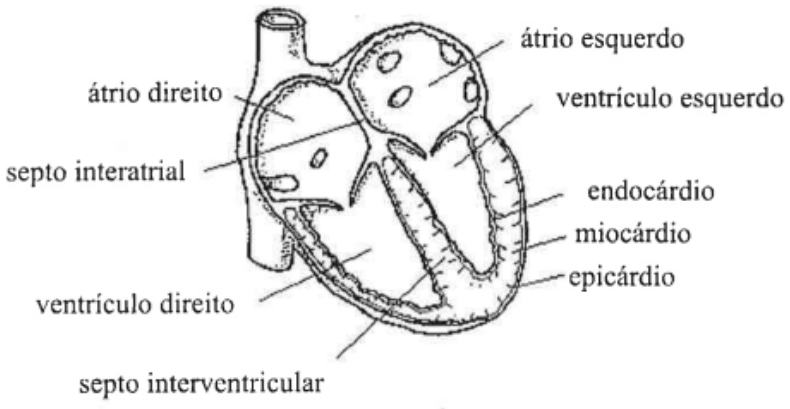

Figura 2.1: Anatomia do coração humano. [14]

O nódulo SA está localizado na junção do átrio direito e da veia cava superior e é responsável por gerar o impulso cardíaco. Os feixes inter-nodais anterior, médio e posterior são responsáveis pela propagação do estímulo nos átrios e nódulo auriculoventricular (AV). O nódulo AV está localizado no septo inter-atrial baixo, sendo composto pela junção de fibras cujo o objetivo é de bloquear e atrasar a

condução do estímulo proveniente dos átrios antes de alcançarem o ventrículo. O feixe de His encontra-se após a bifurcação para os ramos direito e esquerdo. As fibras de Purkinje encontram-se no ápice do coração e são responsáveis pela coordenação dos ventrículos. [29]

2.2 Sistema de Condução Cardíaco

O sistema de condução intrínseca do coração é composto por várias sub-populações de células especializadas que geram espontaneamente atividade elétrica (células *pacemaker*) e células que conduzem esta despolarização pelo coração de forma rápida e altamente coordenada. Este sistema de células controla o momento da transferência de atividade entre as câmaras atriais e ventriculares, permitindo um desempenho hemodinâmico otimizado [29].

Os miócitos são especializados em gerar o impulso cardíaco, no nódulo SA, e a conduzi-lo do átrio até ao nódulo AV sendo denominados de sistema de condução, Figura 2.2. Geralmente os impulsos elétricos deslocam-se pelo nódulo AV a taxas mais lentas que o restante sistema de condução, permitindo que ocorra a passagem do sangue dos átrios para os ventrículos antes da ativação ventricular. Caso o nódulo SA falhe a gerar o impulso elétrico cardíaco, o nódulo AV irá desencadear um ritmo de escape. [29]

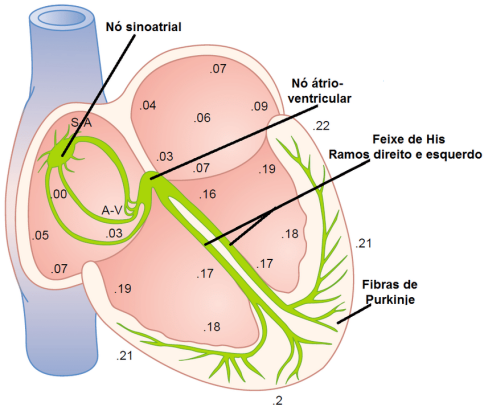

Figura 2.2: Sistema de condução elétrica normal do coração [5].

O automatismo das células *pacemaker* determina a taxa e a regularidade da atividade do nódulo SA. A atividade dos nódulos SA e AV podem ser influenciadas por fatores extrínsecos, como o tónus neural, eletrólitos e medicamentos. As contrações coordenadas dos átrios e ventrículos são reguladas pela transmissão de impulsos elétricos que atravessam uma rede intrínseca destas células musculares cardíacas modificadas. [29]

As células nodais são responsáveis por gerar despolarizações de um ritmo cardíaco normal que pode ser descrito como intrínseco ou automático. Apesar dos ritmos atriais iniciarem-se, normalmente, no nódulo SA existe a possibilidade de variações, podendo a despolarização atrial ser iniciada noutros tecidos quando se verificam taxas atriais mais altas [29]. O nódulo SA é responsável por comandar o ritmo cardíaco, uma vez que a sua rampa de despolarização é mais acentuada, com um maior declive em comparação ao nódulo AV. No entanto, o nódulo AV apresenta três características importantes para o correto funcionamento do coração, tais como a sincronização, quando atrasa o estímulo proveniente do nódulo SA antes de o encaminhar das aurículas para os ventrículos; a filtragem, arritmias ao nível das aurículas são extremamente frequentes e este nódulo tem a capacidade de filtrar algumas frequências mais rápidas; geração do estímulo auxiliar, caso exista uma alteração na condução elétrica e ocorra um bloqueio, o nódulo AV tem a capacidade de gerar um ritmo de escape com frequências cardíacas mais

elevadas para manter o sistema a funcionar.

Após a excitação do nódulo SA a despolarização propaga-se pelos átrios através de células nodais, podendo esta despolarização ser direta para as células miocardias adjacentes, enquanto as vias de miofibrilhas permitem que a excitação atravesse o átrio direito para o átrio esquerdo e, em seguida, para o nódulo AV. Posteriormente à excitação ventricular, os impulsos são conduzidos até ao feixe de His onde a onda de despolarização cardíaca normal propaga-se para os ramos esquerdo e direito, onde alcançam as regiões apicais dos ventrículos esquerdo e direito, respetivamente. Por fim, o sinal percorre as fibras de Purkinje ocorrendo a dispersão da despolarização do miocárdio ventricular. As fibras de Purkinje proporcionam uma ativação rápida e coordenam o padrão de despolarização nas várias regiões do miocárdio ventricular. [29]

A sequência do sistema de condução elétrica do coração inicia-se no nódulo SA, passa para o nódulo AV, feixe de His e, por fim, para as fibras de Purkinje, quanto mais a baixo no sistema for gerado o impulso cardíaco, maior será a probabilidade de o ritmo de escape falhar, levando a uma paragem cardíaca.

2.3 Eletrocardiograma

O ECG é uma representação dos eventos elétricos que ocorrem durante o ciclo cardíaco, onde cada evento tem uma forma de onda associada. A análise do ECG fornece uma visão do estado eletrofisiológico do coração, sendo o registo das despolarizações atrial e ventricular e a repolarização realizado com auxílio de um papel milimetrado. [29]

A distância entre cada linha, tanto vertical como horizontal, é de 1 milímetro (mm) formando um quadrado, sendo que a cada cinco quadrados existe uma linha mais pronunciada. O eixo horizontal mede o tempo, enquanto o eixo vertical mede, consoante a calibração do ECG, a amplitude dos eventos cardíacos. A velocidade de impressão mais utilizada é de 25 mm/s (milímetros por segundo) e a calibração habitualmente utilizada nos equipamentos é de 10 mm, ou seja 10 quadrados menores, que correspondem a 1 miliVolt (mV), mas esta calibração poderá ser alterada consoante as necessidades, outro exemplo de uma calibração utilizada é de 5 mm que correspondem a 0.5 mV. À velocidade de 25 mm/s cada quadrado na vertical equivale a 0,1 mV, enquanto na horizontal um quadrado menor corresponde a 0,04 segundos (s) e um maior a 0,2 s. Uma vez que cinco quadrados maiores correspondem a 1 s, sabe-se que 1 minuto corresponderá a 300 quadrados maiores. Com base nestes dados para um ritmo cardíaco normal, com uma velocidade de impressão de 25 mm/s, o ritmo cardíaco pode ser calculado através da contagem dos quadrados maiores entre duas ondas R consecutivas e, em seguida, dividindo o número obtido por 300, se tiver sido contado o número de quadrados menores a divisão será realizada por 1500. [22][27]

A obtenção do ECG é feita através de elétrodos colocados na superfície do corpo de forma a medir a diferença de potencial elétrico existente durante a propagação do impulso. Por convenção, quando o impulso elétrico se aproxima diretamente do elétrodo positivo, dá origem a uma deflexão ascendente (”positiva”) em relação à linha base isoelétrica, no entanto se o impulso estiver a afastar-se do elétrodo positivo originará uma deflexão descendente (”negativa”) em relação à linha base. Se a onda de despolarização se aproximar perpendicularmente do elétrodo, é produzida uma deflexão isodifásica (deflexão positiva com dimensão igual à deflexão negativa) como está ilustrado na figura 2.3. [22][29]

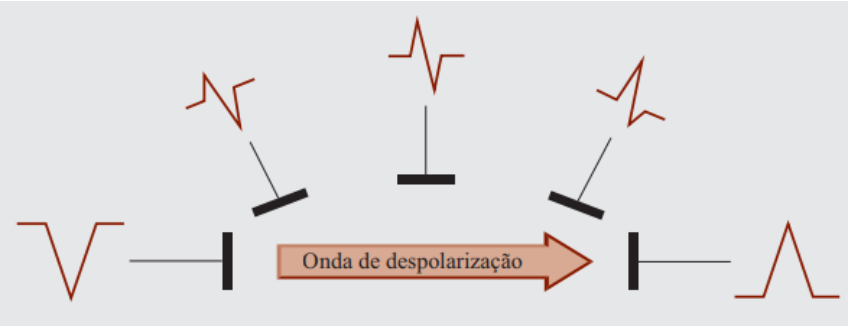

Figura 2.3: Morfologia do traçado de ECG consoante a localização do elétrodo [27].

2.3.1 Sistema de eixos cardíaco

Uma derivação é a combinação de 2 elétrodos que captam e registam uma diferença de potencial, sendo que cada um dos elétrodos cria um ponto de referência. As derivações podem ser divididas em dois grupos: as derivações dos membros, permitem a análise da atividade elétrica sobre o plano frontal que podem ser subdivididas em derivações bipolares e unipolares, e as derivações precordiais, permitem a avaliação da atividade elétrica sobre o plano transversal. [14][29]

As derivações bipolares são três e representam os lados de um triângulo, o "triângulo de *Eithoven*", que correspondem aos elétrodos I, II e III. ilustrado na Figura 2.4. Estas derivações medem a diferença de potencial elétrico entre dois pontos no corpo, com polos opostos, um negativo e um positivo. Com a união destas derivações bipolares ao centro é possível alcançar uma diferença de potencial próxima de zero, em relação à terra. Através deste ponto de união e de um novo elétrodo é possível definir três derivações unipolares, também denominadas de aumentadas (a), que correspondem aos elétrodos aVR, aVL e aVF. A função destas derivações é comparar um ponto no corpo com a referência virtual localizada no centro do peito, ou seja, estas derivações medem a diferença de potencial entre um polo e a terra. O vetor gerado de cada uma destas derivações irá apontar para a área de maior positividade, estando indicada na Tabela 2.1 a localização destas derivações. [27][29]

Tabela 2.1: Configurações da colocação dos elétrodos para visualização no plano frontal.

Derivação	Membro Positivo	Membro Negativo
I	Braço Esquerdo	Braço Direito
II	Perna Esquerda	Braço Direito
III	Perna Esquerda	Braço Esquerdo
aVR	Braço Direito	Braço e Perna Esquerda
aVL	Braço Esquerdo	Braço Direito e Perna Esquerda
aVF	Perna Esquerda	Braço e Perna Direita

As derivações precordiais são utilizadas numa configuração unipolar, sendo estas seis (V1, V2, V3, V4, V5 e V6). Estas derivações analisam a parte anterior do tórax, tendo pontos anatómicos específicos para o seu posicionamento e com a análise a ser realizada segundo o plano transversal. A localização destas derivações encontra-se indicada na Tabela 2.2 e está ilustrada na Figura 2.4. [27][29]

Tabela 2.2: Configurações da colocação dos elétrodos para visualização no plano transversal.

Derivação	Localização
V1	Quarto Espaço Intercostal, na margem direita do externo
V2	Quarto Espaço Intercostal, margem esquerda do externo
V3	Entre os elétrodos V2 e V4
V4	Quinto Espaço Intercostal, na linha mediano-clavicular
V5	Lateral a V4, na linha axilar anterior
V6	Lateral a V4 e V5, na linha média-axilar

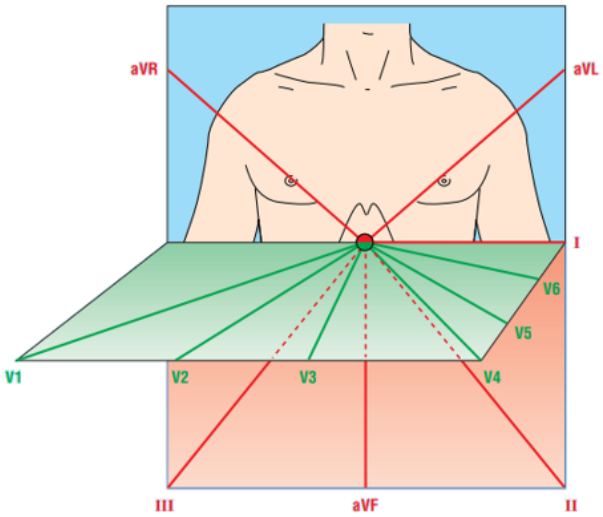

Figura 2.4: Configuração dos elétrodos para visualização do coração segundo o plano frontal (representado a vermelho) e segundo o plano transversal (representado a verde). [22]

A disposição das derivações dá origem a algumas relações anatómicas. Para a visualização da parede inferior do coração pode-se recorrer às derivações II, III e aVF, para a visualização da parede anterior recorre-se às derivações V1 a V4, para a parede lateral são as derivações I, aVL, V5 e V6 e para observação da parede inferior do ventrículo esquerdo através do átrio direito (derivações V1 e aVR). [22]

A identificação do eixo elétrico médio cardíaco é um passo importante para a interpretação do ECG, auxiliando no diagnóstico diferencial [16]. O eixo cardíaco corresponde à direção média da onda de despolarização ventricular, segundo o plano frontal, tendo em consideração o ponto de referência zero para as medições. [22]

Cada derivação gera um vetor que representa a magnitude, polaridade e direção do potencial de ação, uma vez que o eixo cardíaco é constituído pela soma de todos os vetores individuais gerados e que cada potencial de ação é produzido por um miócito, é possível determinar o eixo de cada onda e o intervalo do ciclo cardíaco. [16]

Pode ser determinado o eixo cardíaco das onda P ou T, dos segmentos TP ou ST ou do complexo QRS. No entanto, uma vez que o complexo QRS representa a despolarização ventricular e é facilmente detetável, devido à sua dimensão, é mais comum recorrer a este para a determinação do eixo. [17]

O ponto de referência zero tem o mesmo ponto de vista que a derivação I, em relação ao coração. Assim, a um eixo situado acima desta linha é atribuído um valor negativo, enquanto um eixo situado

abaixo da linha terá um valor positivo, como se pode observar na Figura 2.5, que diz respeito a um diagrama hexaxial representativo da visão de cada derivação em relação ao coração, segundo o plano frontal. O intervalo normal do eixo cardíaco situa-se entre -30º e 90º, caso o eixo seja inferior a -30º é considerado um desvio do eixo para a esquerda, no entanto se o eixo for superior a 90º é considerado um desvio do eixo para a direita e entre os -90º e os 180º corresponde ao eixo indeterminado. [17][22]

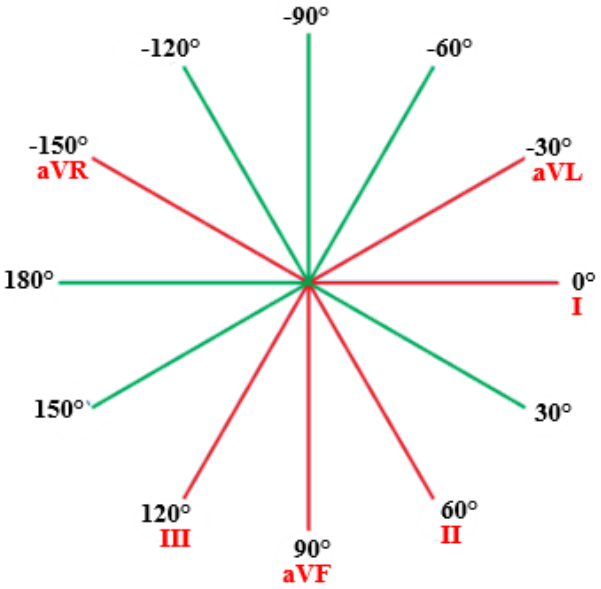

Figura 2.5: Projeção de seis derivações no plano frontal, com os respetivos ângulos associados. [22]

Existem diversos métodos para calcular o eixo cardíaco, sendo o método mais simples através da análise das derivações I, II e III, tendo em consideração as indicações da Tabela 2.3.

Tabela 2.3: Cálculo do eixo cardíaco

	Eixo normal	Desvio do eixo à direita	Desvio do eixo à esquerda
Derivação I	Positivo	Negativo	Positivo
Derivação II	Positivo	Positivo ou Negativo	Negativo
Derivação III	Positivo ou Negativo	Positivo	Negativo

Apesar de ser mais simples analisar apenas três derivações, procedendo à análise das seis derivações obtém-se uma estimativa mais precisa do eixo cardíaco. A direção da propagação do impulso ocorre em direção às derivações com deflexão positiva, afastando-se das derivações com deflexão negativa, e aproxima-se perfazendo um ângulo de 90º da derivação com um complexo QRS isodifásico [22]. Tendo em consideração estes detalhes e o diagrama hexaxial da Figura 2.5, o eixo é determinado da seguinte forma:

1. Identificar a derivação que esteja mais próxima de ser equifásica.

   (ex: derivação II que equivale a 60º no eixo)

2. Acrescentar 90º ao eixo à direita e à esquerda da derivação identificada.

   (ex: à esquerda corresponderá a 150º e à direita a -30º)

3. Examinar os complexos QRS das derivações adjacentes à derivação equifásica

   (ex: derivações I e III).

4. Identificar qual das derivações adjacentes é positiva.

5. O eixo ficará 90º para o lado da derivação adjacente positiva

   (ex: se a derivação para o lado esquerdo for positiva, então o eixo estará a 90º da derivação equifásica para a esquerda, ficando o eixo a 150º segundo este exemplo. Se a derivação positiva for para o lado direito, então o eixo estará a 90º em relação à derivação equifásica para a direita, ficando o eixo a -30º segundo o exemplo).

Desvios para a esquerda poderão ser considerados normais na população mais idosa ou obesa e, em alguns casos, durante a gravidez. Mulheres com idade superior a 40 anos tendem a ter um desvio para a esquerda normal inferior aos homens da mesma idade. É importante ressalvar que apenas são considerados desvios do eixo esquerdo normais se forem até aos -30º. [17]

O desvio do eixo direito pode ser considerado normal na população mais jovem, mais saudável e em forma, e nos recém-nascidos até, aproximadamente, aos 6 meses de idade. Nos recém-nascidos este desvio é normal devido à anatomia dos ventrículos, sendo que o ventrículo direito é maior que o ventrículo esquerdo. [17]

O desvio do eixo poderá ser considerado anormal por diversos fatores, tais como:

- Congénito: Indivíduos que nascem com defeitos dos septos atrias e ventriculares e com dextrocardia, anomalia rara em que a ápice do coração encontra-se apontado para o lado direito, podem apresentar um desvio do eixo para a direita. [17]

- Posição do coração: Determinadas condições físicas podem ter influência na posição do coração. Este pode inclinar para cima e para a esquerda em indivíduos obesos ou grávidas, deslocando o eixo para a esquerda, alonga para baixo em indivíduos em forma física ou com doença grave das vias aéreas ou pneumotórax do lado esquerdo, desviando o eixo para a direita. [17]

- Hipertrofia ventricular: O eixo elétrico desvia-se em direção ao ventrículo afetado, podendo corresponder tanto a um desvio do eixo para a esquerda como para a direita. Em caso de se tratar de uma doença cardíaca generalizada, será bastante provável que o eixo elétrico seja indeterminado. [17]

- Vias de condução alteradas: Uma via acessória do lado esquerdo provoca um desvio do eixo para a direita, enquanto se for do lado direito o desvio provocado será para esquerda. A direção da despolarização poderá ser alterada num individuo com problemas cardiovasculares, uma vez que as áreas necróticas não têm capacidade de despolarização. [17]

- Perturbações nos eletrólitos: Exemplo destas perturbações é a hipercalemia, corresponde aos valores de potássio elevados, pode deslocar o eixo para a esquerda. [17]

2.4 O Ciclo Cardíaco

Um sinal ECG básico durante um ciclo cardíaco de um paciente saudável, sem nenhuma patologia associada, é composto por uma onda P, complexo QRS e onda T. A seguir à onda T pode ser observada, em

alguns casos, uma onda U que está associada à pós-despolarização dos ventrículos [29]. São denominadas de deflexões as regiões que não apresentam potencial nulo. A distância entre duas deflexões é um segmento, enquanto a distância que envolve deflexões e segmentos é um intervalo. A tensão de base do sinal de ECG é chamada de linha isoelétrica. Na Figura 2.6 está ilustrado um ciclo cardíaco com os seus diferentes componentes [14].

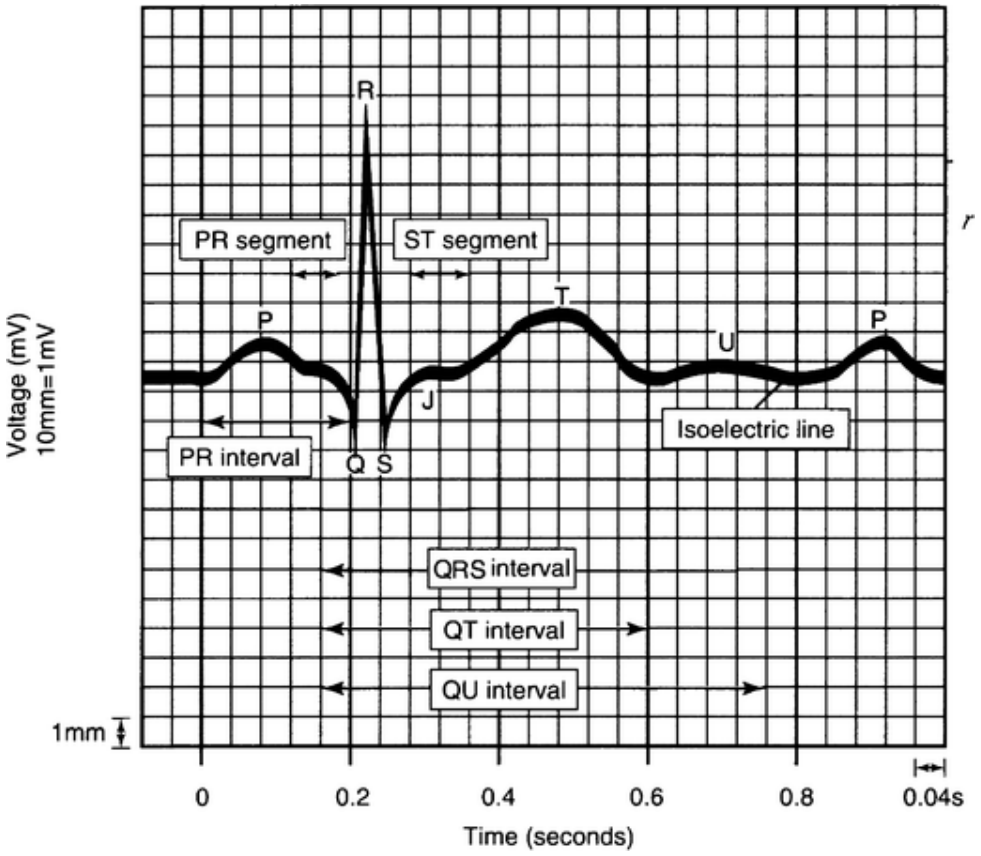

Figura 2.6: O sinal de ECG e os seus componentes [29].

Onda P: Deflexão resultante da despolarização atrial. Durante o ritmo sinusal normal, esta onda é direta (positiva) nos elétrodos I, II e III e invertida (negativa) em aVR. Esta onda, normalmente, tem uma amplitude entre os 0,25 e os 0,30 mV e uma duração inferior a 110 milissegundos (ms). [27][29]

Segmento PR: Parte do sinal compreendida entre a onda P e a onda R, indicado pela linha isoelétrica. A duração deste segmento é de, aproximadamente, 80 ms. Nesta fase o impulso está a propagar-se através do nódulo AV. [14]

Intervalo PR: É medido desde o começo da onda P até ao começo do complexo QRS, que corresponde ao tempo de condução através do nódulo AV. A duração deste intervalo varia com a idade, sendo prolongado em indivíduos com a idade mais avançada. A duração normal deste intervalo em adultos está entre os 120 e 200 ms. [27][29]

Complexo QRS: Este complexo segue-se ao intervalo PR e é produzido pela despolarização ventricular, representa o estímulo consoante se vai propagando pelos ventrículos [14]. É constituído pelas ondas Q, R e S que correspondem a duas deflexões negativas e uma deflexão positiva. A duração total deste complexo num indivíduo adulto é inferior a 110 ms. A amplitude deste complexo é variável, dependendo das condições cardíacas e extra cardíacas que o indivíduo apresenta. [27][29]

Segmento ST: Este segmento diz respeito ao sinal entre o final do complexo QRS, denominado de

ponto J, até ao início da onda T [14], sendo indicado pela linha isoelétrica. Caso se verifique uma elevação do segmento, superior a 1 mm, poderá ser indicativo de uma possível lesão miocárdica, uma pericardite ou poderá corresponder à presença de um aneurisma ventricular. [27][29]

Onda T: Corresponde à deflexão após a repolarização ventricular. Geralmente a amplitude da onda é inferior à amplitude do complexo QRS, estando entre os 0.1 e 0.5 mV. [14][29]

Intervalo QT: Este intervalo é medido desde o começo da onda Q, do complexo QRS, até ao fim da onda T, inclusive. O valor deste intervalo é influenciado pelo género e idade do indivíduo, sendo mais curto no sexo masculino conforme o indivíduo atinge a maturidade. A duração deste intervalo é, geralmente, entre 300 e 460 ms. [27][29]

Onda U: Corresponde ao último componente do ciclo cardíaco e encontra-se após a conclusão da onda T, geralmente, trata-se de uma onda de baixa frequência. Uma vez que é inversamente proporcional à frequência cardíaca, apresenta uma maior amplitude em atletas e uma menor amplitude em crianças e indivíduos bradicárdicos. [27]

2.5 Eletrofisiologia Cardíaca Clínica

As células *pacemaker* de um adulto, em condições fisiológicas normais, apresentam um ritmo sinusal, o ritmo fisiológico cardíaco, entre 60 e 100 batimentos por minuto (bpm), sendo que quando o indivíduo se encontra em repouso a taxa sinusal nodal pode diminuir para cerca de 75 bpm ou menos. No entanto, indivíduos com uma maior resistência atlética poderão ter o seu ritmo cardíaco normal a 40 bpm, ou seja, pode ser considerada uma bradicardia sinusal. [27][29]

Consoante a idade e o ritmo cardíaco de um indivíduo, em condições fisiológicas normais, obtém-se um intervalo PR diferente, sendo mais curto em crianças e alongado em idosos. O valor normal de duração deste intervalo num adulto encontra-se entre 120 e 200 ms, enquanto numa criança o valor mínimo não será inferior a 90 ms. Se a duração deste intervalo PR estiver fora destes parâmetros poderá corresponder a um indivíduo com patologia. [27][29]

A duração do complexo QRS também é influenciado pela idade do indivíduo, sendo maior quanto maior for a idade. A duração normal do complexo QRS, em adultos, encontra-se entre 50 e 110 ms. No entanto, se este complexo QRS apresentar uma duração superior a 140 ms é indicativo de uma patologia associada. [27][29]

A onda P também é influenciada pela idade do indivíduo, desta forma a duração normal para uma criança é entre 60 e 90 ms, enquanto para um adulto está entre os 80 e 110 ms. Pode ser esperada algum tipo de patologia cardíaca se uma criança apresentar uma onda P com uma duração superior a 90 ms, se tiver menos de 10 anos, superior a 100 ms, se estiver entre os 10 e os 15 anos, e no caso de um adulto será quando a duração da onda P ultrapassar os 110 ms. Estas características encontram-se na Tabela 2.4 [27][29]

Tabela 2.4: Identificação das características para um individuo normal e um com patologia.

Normal	Patologia
Ritmo cardíaco: 60 - 100 bpm	Ritmo cardíaco: $<$ 60 bpm ou $>$ 100 bpm
Onda P: 80 - 110 ms	Onda P: $>$ 110 ms
Intervalo PR: 120 - 200 ms	Intervalo PR: $<$ 120 ms ou $>$ 200 ms
Complexo QRS: 50 - 110 ms	Complexo QRS: $>$ 140 ms

Uma arritmia corresponde a perturbações que alteram o ritmo cardíaco normal, sendo caracterizadas por batimentos rápidos, lentos e/ou irregulares. Estas podem ser divididas por diferentes critérios, no entanto pela frequência cardíaca pode-se estar perante uma bradicardia ou uma taquicardia. A bradicardia é caracterizada por ritmos lentos, geralmente inferiores a 60 bpm, enquanto a taquicardia apresenta ritmos excessivamente rápidos, sendo superiores a 100 bpm em repouso. É fundamental salientar que a arritmia sinusal não é patológica, no entanto é um fator importante para o acompanhamento profissional nos mais jovens. [12][27][29]

Geralmente uma bradicardia tem como causa associada o nódulo SA, ou seja, poderá estar-se perante uma disfunção do nódulo SA ou no nódulo AV, mais concretamente um bloqueio AV. Estas condições estão relacionadas diretamente com um problema intrínseco nas células de *pacemaker*, no sistema de condução ou por fatores externos. As taquicardias, normalmente, têm como causas associadas um início anormal do impulso, uma condução anormal deste ou, até mesmo, devido a uma combinação de ambos. [29]

Para além das células do nódulo SA existem outras células do sistema de condução com a capacidade de desenvolver auto-ritmicidade, sendo estas as do nódulo AV e do sistema His-Purkinje. No entanto, os ritmos cardíacos derivados destas células estão num intervalo inferior ao normal, entre 25 e 55 bpm. Estes ritmos geralmente são identificados como ritmos de escape ventricular e são importantes para a sobrevivência do indivíduo uma vez que mantêm algum grau de débito cardíaco em situações em que os nódulos SA e/ou AV estão a funcionar de forma inadequada. [29]

Apesar de as arritmias puderem ocorrer em indivíduos saudáveis, estão mais associadas a doenças cardíacas estruturais, como é exemplo as cardiomiopatias, valvulopatias e problemas genéticos. As consequências eletrofisiológicas e hemodinâmicas de uma arritmia são principalmente determinadas pelo ritmo cardíaco, a sua duração e/ou pelas câmaras de origem, isto é, se a origem é atrial ou ventricular. [29]

3. Estado de Arte

O eletrocardiograma é uma técnica não invasiva, de custos reduzidos e facilmente reprodutível, tendo a capacidade de medir eventos eletrofisiológicos complexos nos tecidos cardíacos. O débito cardíaco por minuto é o principal evento cardiovascular que é imprescindível para manter o fluxo sanguíneo em todo o corpo. Além do volume sanguíneo e da capacidade e força de contração do coração, é fundamental que este consiga manter um ciclo regular de relaxamento e contração, esta regularidade pode ser monitorizada com o ECG [2]. A quantidade de informação que o ECG consegue extrair é fundamental para auxiliar no diagnóstico e orientar decisões clínicas referentes a possíveis tratamentos adequados [13].

A leitura e interpretação do sinal de ECG é realizada de forma manual, numa primeira linha pelos técnicos que efetuam o exame e, posteriormente, pelos médicos cardiologistas em mais detalhe, sendo estes os responsáveis pela elaboração do relatório. Devido à carga de trabalho sentida por estes técnicos e à pressão exercida para aumentar a produção de resultados, é inevitável que haja ocorrência de alguns erros humanos, podendo padrões anormais mais subtis passarem despercebidos. Otimizar a precisão de diagnóstico e a produtividade dos técnicos são fundamentais, assim o desenvolvimento de um modelo de DL é essencial. [30]

O ECG é uma ferramenta essencial para o diagnóstico de doenças cardiovasculares e, com aplicação de técnicas de DL, verifica-se um aumento no nível de precisão e eficácia na classificação dos sinais [13]. A utilização de bases de dados de ECG classificados, como é o caso da *"MIT-BIH Arrhythmia"* [23], tem sido essencial para treinar modelos de DL eficazes. No entanto, o pré-processamento dos dados, incluindo o suavização, remoção do ruído, entre outros, é crucial para que o desempenho dos modelos de DL desenvolvidos seja elevado.

Apesar de haver uma grande evolução nas soluções e estudos publicados na área da AI, artigos sobre a classificação automática de um sinal de ECG, recorrendo a *wavelets* para a extração dos parâmetros e um método de DL para a classificação do sinal, são escassos.

Gothwal et all (2010) realizaram um estudo onde recorreram à *Fast Fourier Transform* (FFT), Transformada de Fourier Rápida, para deteção dos picos do sinal antes de proceder à aplicação de redes neuronais para a identificação das patologias, fazendo a distinção entre as patologias cardiovasculares. Para a realização deste estudo os sinais utilizados foram da base de dados *"MIT-BIH Arrhythmia"*. Neste estudo recorreram à identificação das deflexões Q, R e S para conseguirem identificar o complexo QRS, sendo que devido à característica comum do pico R, este foi identificado em primeiro lugar e foi através deste que os pontos Q e S foram identificados. A FFT foi aplicada ao sinal filtrado, após a remoção das baixas frequências e, em seguida, no sinal resultado foi aplicada a FFT inversa, desta forma o sinal ficou pronto para a identificação dos picos R. A rede neuronal foi treinada recorrendo a 20 conjuntos de dados, onde foram identificados os parâmetros do complexo QRS, desde a largura máxima, mínima e média do complexo até à identificação da frequência cardíaca, depois foram utilizados mais 20 conjuntos de dados para os testes. O método desenvolvido conseguiu alcançar um nível de precisão alto, de 98.48%, superior aos outros métodos utilizados para comparação. [13]

A deteção de arritmias, que se encontrem numa fase inicial, através da utilização de um dispositivo telemóvel foi estudado por *Patel et all* (2012), onde o público alvo são as populações rurais, com difícil acesso a cuidados de saúde. O objetivo deste estudo foi desenvolver um novo sistema para a aquisição e processamento do ECG, com posterior classificação, através de um dispositivo conectado a um *smartphone* por *Bluetooth* que transmite o sinal adquirido para o médico em tempo real. À

semelhança do estudo anterior, a base de dados utilizada para o desenvolvimento do algoritmo foi a *"MIT-BIH Arrhythmia"*. Os parâmetros seriam selecionados e filtrados em tempo real, tendo sido verificada uma dificuldade acrescida na implementação de um algoritmo capaz de detetar os complexos QRS e os picos P no *smartphone*, visto que a capacidade dos CPUs é inferior a um computador. O algoritmo desenvolvido capaz de funcionar num *smartphone* procede à filtragem do sinal, diferenciação e deteção da média temporal para deteção do complexo QRS. O algoritmo desenvolvido mostrou-se eficiente e com uma precisão na classificação de 97.3%, no entanto foi verificado que existem algumas exceções na classificação resultantes da ampla variedade de patologias cardíacas. [25]

É fundamental enfatizar que a interpretação dos modelos de DL, nesta área da cardiologia, continuam a ser um grande desafio, visto que é necessário compreender as previsões que o modelo faz e que apenas desta forma podem ser tomadas decisões clínicas. Isto é, o desenvolvimento destes modelos e a implementação apenas servem de auxílio para os profissionais, detetando com antecedência possíveis patologias.

A utilização de DL na leitura e interpretação dos ECG ainda está numa fase pouco desenvolvida, com grande potencial de crescimento. Os *wearables* são uma das aplicações mais úteis para a área, isto deve-se ao simples facto de que estes dispositivos têm a capacidade de registar um evento que ocorra para que seja possível, posteriormente, proceder à análise do mesmo. Por vezes os testes de holter não têm a capacidade de captar alguns eventos cardíacos, uma vez que estes testes têm uma duração de apenas 24h, o evento poderá não ocorrer neste período de tempo não sendo detetada a patologia, é nesta fase que o *wearable* passa a ser uma mais valia para o profissional de saúde consultar os registos.

Em suma, a classificação de sinais de ECG recorrendo a métodos de DL têm vindo cada vez mais a ter avanços significativos, originando modelos altamente precisos e com grande potencial de deteção precoce de patologias, levando a uma tratamento atempado de algumas patologias cardíacas. No entanto a questão da interpretação dos sinais continuam a ser áreas em que a pesquisa realizada até à data não é o suficiente, é necessário continuar a investigar e desenvolver novas e melhores técnicas para a prevenção de arritmias cardíacas.

4. *Wavelets*

A deteção e diagnóstico de doenças cardiovasculares depende da extração e classificação dos batimentos cardíacos, o que requer a leitura e análise dos diferentes constituintes do ciclo cardíaco e, como o sinal não é regular na sua periodicidade, recorre-se às *wavelets* para obter uma representação mais precisa das diferentes características do sinal em diferentes escalas de tempo. [1] [31]

4.1 *Wavelet Transform*

A *Wavelet Transform* (WT) é uma técnica de processamento de imagens e sinais que recorre a funções matemáticas, as *wavelets*, para decompor os dados em componentes de diferentes frequências e escalas e, posteriormente, proceder à analise de cada componente com uma resolução proporcional à sua escala. As *wavelets* possuem a capacidade de regularizar o sinal (extraindo o "ruído") permitindo a posterior extração de informações espectrais e de localização temporal. Existem vários tipos de *wavelets* que são utilizadas em diferentes aplicações, consoante as características do sinal, ou inclusive da imagem, a analisar. [8] [20]

A WT tem um amplo leque de funções importantes para o pré processamento do sinal, tais como [6]:

- Compressão de imagens: redução do tamanho de imagens mantendo a qualidade mínima necessária para permanecer aceitável a sua leitura.
- Filtragem de sinais: filtrar o ruído, removendo possíveis artefactos, e extrair as características pretendidas de cada um dos sinais, bastante utilizado nos sinais de ECG.
- Análise de séries temporais: identificação de padrões em diferentes escalas temporais.
- Deteção de bordas em imagens: deteção de limites e características em imagens digitais.
- Processamento de sinais de áudio: análise do áudio para possibilitar a separação de componentes de baixo e alta frequência.

A WT pode ser divida em dois grupos principais, consoante a dilatação e a translação que as *wavelets* sofrem para a análise dos sinais ou imagens, podendo ser dividida em *Continuous Wavelet Transform* (CWT) e em *Discrete Wavelet Transform* (DWT). [8]

4.1.1 *Continuous Wavelet Transform*

A WT da função f é definida pela transformada integral da Equação 4.1 em que a Equação 4.2 representa a "função mãe", ou "*wavelet* mãe". Esta *wavelet* mãe é escolhida consoante a aplicação pretendida e a sua forma será responsável por determinar as propriedades da WT. Para além da *wavelet* mãe a WT envolve uma função de escala que é utilizada para realizar a análise nas diferentes escalas de resolução. [6]

$$W_f^{\psi}(a,b) = \int_{-\infty}^{\infty} f(t)\psi_{a,b}(t)dt \qquad a > 0, \tag{4.1}$$

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4.2}$$

A dilatação (parâmetro a) e a translação (parâmetros b) correspondem, respetivamente, à escala e deslocamento da *wavelet* mãe e da função escala, e permitem que seja realizada a análise do sinal original em várias escalas temporais e de frequência. A dilatação é responsável por aumentar ou diminuir a escala da função, enquanto a translação desloca a função ao longo do eixo temporal. A combinação destas operações permite a obtenção de coeficientes de *wavelets* em várias escalas e posições. [6] [8]

Se o parâmetro $a > 1$ a *wavelet* mãe sofre uma dilatação, ou seja, é esticada no eixo tempo ou espacial, o que permitirá a análise de componentes de frequências mais baixas e estruturas com uma escala superior. Por sua vez, se o parâmetro for $0 < a < 1$ então ocorre uma contração da *wavelet* mãe, ou seja, a função será comprimida no eixo do tempo ou espacial, permitindo a análise de componentes de frequência mais alta e detalhes finos do sinal. Quando o parâmetro $b > 0$ ocorre uma translação para a direita, ou seja, a *wavelet* mãe será deslocada para a direita no gráfico segundo o eixo tempo ou espacial. Se o parâmetro $b < 0$ irá ocorrer uma translação da *wavelet* mãe para a esquerda do gráfico. [6]

Os coeficientes de *wavelets* resultam da aplicação da WT a um sinal e representam a contribuição das funções após a dilatação e translação da *wavelet* mãe e da função escala. Os coeficientes de *wavelets* podem ser divididos em dois conjuntos, os coeficientes de aproximação e os de detalhe. Os coeficientes de aproximação dizem respeito às informações de baixa frequência ou às tendências do sinal e têm a capacidade de fazer o ajuste da *wavelet* à curva do sinal, se o valor absoluto obtido for elevado verifica-se um bom ajuste, no entanto se o valor for perto de zero, trata-se de um mau ajuste ao sinal. Os coeficientes de detalhe representam as informações de alta frequência ou os detalhes finos do sinal. Um exemplo prático seria imaginar os coeficientes de aproximação como se fosse uma árvore no seu todo, enquanto os coeficientes de detalhe é como se a árvore estivesse mais próxima e estivéssemos apenas a observar as suas folhas. [7]

A WT origina uma representação multirresolução do sinal, ou seja, as informações extraídas são armazenadas em diferentes níveis de detalhe que se assemelham a uma pirâmide. Na base da pirâmide de resolução estão representadas as informações de baixa frequência e, consoante se vai subindo na pirâmide, maior será o nível de detalhe armazenado. [19]

O objetivo desta representação multirresolução do sinal é, no final, conseguir reconstruir o sinal original através dos coeficientes de *wavelets* 4.3. Desta forma, é necessário fazer a inversão da WT através da soma dos coeficientes de aproximação e de detalhe e aplicando operações de dilatação e translação inversas para cada escala. [19]

$$f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{0}^{\infty} \frac{1}{a^2} W_f^{\psi}(a,b) \psi_{a,b}(t) \, da \, db, \tag{4.3}$$

com

$$C = \int_{0}^{\infty} \frac{|\hat{\psi}(\omega)|^2)}{|\omega|)}$$

a condição de admissibilidade para a *wavelet* mãe ψ, sendo $\hat{\psi}$ a correspondente *fourier transform* de ψ.

Alguns exemplos de *wavelets* contínuas são: [3][6]

- *wavelet* Morlet: Consiste numa onda plana modulada por uma função Gaussiana. Utilizada na análise de sinais do domínio do tempo-frequência, um exemplo é a análise de séries temporais.
- *wavelet* de Maar: Também conhecida por chapéu mexicano devido à sua forma. Esta pode ser ajustada para diferentes escalas de frequência, permitindo a análise de sinais no domínio tempo-frequência.
- *wavelet* Daubechies: Conhecida como família de Daubechies, é utilizada em compressão de dados, remoção de ruído de sinais e na análise de imagens.

- *wavelet* Meyer: Utilizada para o processamento de imagens, para a restauração ou deteção dos limites desta.

Na CWT a transformada integral da Equação 4.1 assume valores contínuos para os parâmetros a e b, ou seja, analisa o sinal com uma ampla variedade de escalas de resolução, o que possibilita a extração de uma representação, contínua das frequências em relação ao tempo ou espaço, completa do sinal. [6]

Geralmente recorre-se à utilização da CWT para uma análise detalhada em escalas contínuas, ou seja, para a deteção de características em imagens, análise de eventos em séries temporais, entre outros. [8]

4.1.2 *Discrete Wavelet Transform*

Na DWT o sinal é divido em vários segmentos discretos, de escala j e localização k, sendo a decomposição deste sinal realizado de forma hierárquica em coeficientes de detalhe e aproximação em diferentes níveis de resolução. Desta forma, é obtida uma representação discreta em relação ao tempo ou espaço. A DWT é definida pela Equação 4.4, onde a Equação 4.5 corresponde às funções da *wavelet* e d_k^j são os coeficientes da *wavelet*. [19] [20]

$$d_k^j = 2^{-\frac{j}{2}} \sum_n f(t)\psi_k^j(t) \quad dt, \tag{4.4}$$

$$\psi_k^j(t) = 2^{-\frac{j}{2}}\psi(2^{-j}t - k) \tag{4.5}$$

$$a_{j_0,k} = 2^{-\frac{j}{2}} \sum_n f(t)\phi_{j_0,k}(t) \quad dt, \tag{4.6}$$

A transformação inversa é uma operação mais delicada, no caso discreto, uma vez que a amostragem pode levar a perda de informação. Por isso, não se pode simplesmente aplicar a fórmula da inversão.

O sinal discreto $f \approx y_n$, com dimensão n, pode ser decomposto como

$$y_n(t) = \sum_{k=1}^{\frac{n}{2^j}} a_{j_0,k}\Phi_{j_0,k}(t) + \sum_{j=j_0}^{J-1}\sum_{k=1}^{\frac{n}{2^j}} d_k^j\psi_k^j(t) \tag{4.7}$$

A DWT é bastante utilizada para a compressão de dados, remoção de ruído, deteção dos limites de imagens, entre outras tarefas de processamento de sinais ou imagens. [19]

Alguns exemplos de *wavelets* discretas são: [6]

- Haar: Uma vez que é das mais simples, é bastante utilizada para auxiliar a compreender os princípios da DWT.

- Daubechies: Estas *wavelets* na DWT são utilizadas para o processamento de sinais e imagens, que inclui a compressão de imagens.

- Symlet: É uma variante das *wavelets* de Daubechies. Uma vez que têm uma boa suavização conseguem preservar informações importantes de sinais, sendo geralmente utilizadas para a remoção de ruído de sinais e imagens.

Na Figura 4.1 está ilustrado o processo de decomposição do sinal, sendo primeiro decomposto numa aproximação e depois num detalhe, por sua vez essa aproximação é decomposta numa nova aproximação e num detalhe até atingir o nível estabelecido (por exemplo, uma *wavelet* sym5 equivale a um nível 5).

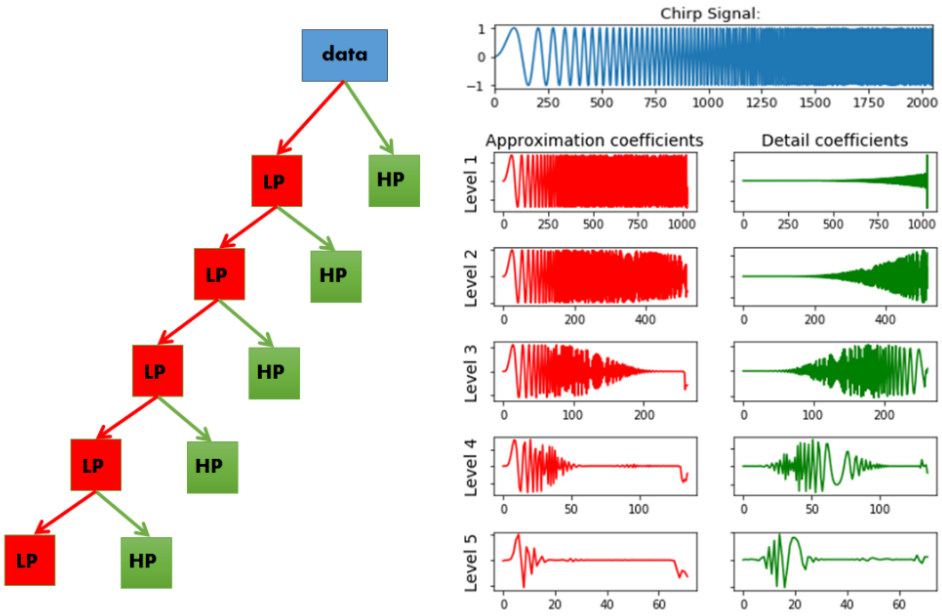

Figura 4.1: Exemplo dos coeficientes de aproximação e de detalhe, com os filtros passa-alto (HP) e passa-baixo (LP) aplicados ao sinal em cada nível para uma aplicação de DWT. [10]

4.2 *Fourier Transform*

A *Fourier Transform* (FT), pode ser expressa como uma soma de ondas senoidais, ou seja, de senos e cossenos. [6] É uma ferramenta bastante útil na análise de sinais, desde que estes sejam estacionários no domínio do tempo, para que a representação das suas componentes seja precisa, uma vez que a FT localiza corretamente as componentes das ondas de frequência, mas não do tempo. [3]

A FT perde informações temporais detalhadas, uma vez que fornece apenas aproximação global das frequências presentes no sinal, ou seja, não fornece detalhes da localização das variações do sinal. Desta forma, se o sinal não for estacionário e localizado esta transformada não será adequada ao sinal, não tendo a capacidade de, por exemplo, detetar eventos transitórios ou anómalos num sinal de ECG, tais como arritmias. [3]

Para conseguir ultrapassar esta desvantagem foi criada a transformada de fourier de curta duração, ou seja, a função de janela extrai dados num pequeno intervalo e, em seguida, calcula a FT. Esta é uma solução para sinais que são globalmente estacionários, mas localmente não são estacionários. A função janela desloca-se ao longo do eixo do tempo para conseguir calcular a FT de diferentes segmentos do sinal. No entanto, se a largura da janela for estreita a localização do sinal poderá ser melhor no domínio do tempo e pior no domínio espetral. [3]

As *wavelets* são comparadas com frequência às FT, No entanto, as *wavelets* têm a vantagem de serem capazes de extrair informações locais e momentâneas de um sinal, enquanto a FT apenas consegue obter as informações globais de frequência.

5. Métodos de *Deep Learning*

A *Artificial Intelligence* corresponde à utilização de um modelo informático com capacidade de tomar decisões onde o seu desempenho melhora com a experiência que vai adquirindo. A AI pode ser classificada em diferentes subconjuntos, o *Machine Learning*, e este tem o *Deep Learning*, como subconjunto, como está ilustrado na Figura 5.1. [30]

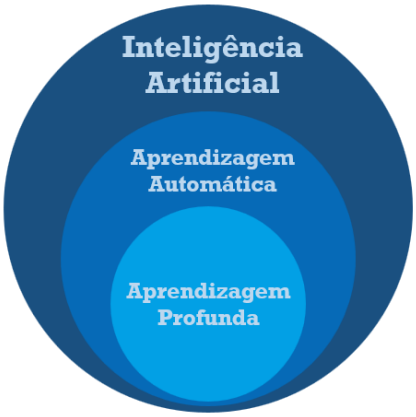

Figura 5.1: *Artificial Intelligence* e os seus subconjuntos.

5.1 *Machine Learning*

Os algoritmos de ML são considerados bem-sucedidos quando os processos de tomada de decisão são generalizados através de conjuntos de dados conhecidos. Estes algoritmos de ML podem ser divididos em aprendizagem supervisionada, não supervisionada, semi-supervisionada e por reforço, consoante o conjuntos de dados fornecidos e do objetivo final. [24]

Existem alguns conceitos que são necessários para a compreensão dos tipos de aprendizagem. Visualizando os dados de entrada como uma tabela temos assim que: [24] [26]

- Entidade: corresponde a cada uma das linhas ou amostras, no caso desta tese, a cada indivíduo que faz parte do estudo.
- Atributo: corresponde às colunas e, neste caso, diz respeito a cada característica associada a um tipo de dados, ou seja, duração da onda P, entre outros.
- Classe: corresponde à última coluna da tabela, no caso da aprendizagem supervisionada, onde é identificada a categoria que a amostra pertence.

Os dados podem ser divididos em três conjuntos: [24][30]

- Treino: utilizados para treinar o sistema de aprendizagem que compara os dados de entrada e os de saída. Normalmente entre 60-80% do conjunto de dados, consoante a disponibilidade dos dados e qualidade dos resultados.
- Validação: utilizados para determinar a eficiência do modelo e, se necessário, consequentemente redefinir os parâmetros. Normalmente entre 10-20% do conjunto de dados.

- Teste: utilizados para avaliar o modelo. É importante que os dados deste conjunto sejam independentes dos utilizados nos outros dois. Normalmente entre 10-20% do conjunto de dados.

5.1.1 Aprendizagem supervisionada

A aprendizagem supervisionada é a forma mais comum de ML, podendo ser utilizada na DL ou não. [18] Na aprendizagem supervisionada os dados de entrada estão classificados e são disponibilizados pelo utilizador, sendo utilizados para treino, onde o algoritmo procura uma maneira de produzir os dados de saída pretendidos. [24]

Existem duas formas para desenvolver o modelo de aprendizagem, os dados podem ser divididos em dados de treino e de teste ou podem ser divididos em dados de treino, validação e teste. Uma vez que, recorrendo à segunda hipótese, é possível isolar os dados de teste do modelo de aprendizagem, esta será a mais indicada. [24] [26]

Durante o treino é apresentado, neste caso, um sinal ao modelo que irá gerar um resultado na forma de um vetor de pontuações para cada classe. O intuito é que a classe pretendida tenha a pontuação mais elevada, em comparação a outras classes, no entanto a probabilidade de este objetivo ser alcançado antes da conclusão do treino é reduzida. A função que mede o erro entre as pontuações de saída e o padrão de pontuações desejado é calculada e, consoante os resultados, o modelo ajusta os parâmetros de forma a reduzir este erro, ou seja, a reduzir a ”função de custo”. Estes parâmetros ajustáveis são designados de pesos e correspondem a números reais que definem a função de entrada-saída do modelo. [18]

Após a etapa de treino procede-se à medição do desempenho do modelo, para tal recorre-se ao conjunto de teste que é constituído por dados novos. Esta etapa permite avaliar a capacidade do modelo de generalização, ou seja, a capacidade de gerar respostas ponderadas com dados novos. [18]

O objetivo é construir um modelo, recorrendo a dados de treino, que tenha a capacidade de prever de forma precisa sobre dados novos, que o modelo não tenha visto, com as mesmas características do conjunto de treino usado. O ideal é construir um modelo capaz de generalizar, para dados que não conhecidos, com maior precisão possível. Se o modelo for demasiado complexo para as informações extraídas é denominado de *overfitting*. Se o modelo for simples demais para os dados é denominado de *underfitting*. [24]

O *overfitting* ocorre quando o modelo é ajustado muito próximo das características do conjunto de treino, o modelo obtido é bastante preciso para este conjunto, no entanto não tem a capacidade de generalizar para dados novos. Desta forma, o modelo aprende aspetos desadequados sobre os dados causando uma falsa impressão de bom desempenho. O *underfitting* é o oposto, quando o modelo é demasiado simples poderá ser mais complicado extrair as características necessárias e determinar a variabilidade dos dados, neste caso, o modelo poderá não ter um bom desempenho ainda na fase de treino. [24]

5.1.2 Aprendizagem não supervisionada

Na aprendizagem não supervisionada, apenas os dados de entrada são conhecidos sendo que estes não têm uma classe associada, ou seja, o algoritmo não terá conhecimento dos dados de saída. Este método procura uma função que seja capaz de descrever a estrutura das camadas escondidas e, uma vez que os dados não estão classificados, não é possível proceder ao cálculo da exatidão da estrutura encontrada. [24] [26]

Apesar deste tipo de aprendizagem se assemelhar ao modo de funcionamento do cérebro humano, geralmente é um método mais complicado de compreender e de avaliar os resultados obtidos. [24]

5.1.3 Aprendizagem semi-supervisionada

A aprendizagem semi-supervisionada é um meio termo entre a aprendizagem supervisionada e a não supervisionada, isto é, apenas alguns dos dados de entrada é que têm uma classe associada. O número de dados que estão classificados é inferior aos não classificados, sendo a tarefa do modelo organizar os dados e prever a sua classe. [26]

Este método de aprendizagem, geralmente, recorre numa fase inicial a dados classificados para treinar o modelo e, em seguida, procede à utilização de dados não classificados para complementar o treino. Desta forma consegue-se atingir um nível de precisão melhorado. [26]

5.1.4 Aprendizagem por reforço

Na aprendizagem por reforço os dados de entrada são fornecidos para que o modelo avalie e tome uma sequência de decisões. O modelo evolui por tentativa e erro, sendo recompensado ou penalizado consoante as decisões tomadas, onde o objetivo é tentar maximizar a recompensa total. Os dados de saída neste modelo não são classificados sendo da responsabilidade do modelo descobrir, com base em diretrizes, como maximizar a recompensa. Numa primeira fase os testes serão aleatórios, no entanto vão evoluindo até serem alcançadas táticas mais sofisticadas. [26]

5.2 *Deep Learning*

A aprendizagem de representações consiste num conjunto de métodos que permitem que o modelo receba os dados brutos e que, de forma automática, interprete as representações necessárias para proceder à deteção e classificação. Os métodos de DL são métodos de aprendizagem de representações, compostos por múltiplas camadas de processamento, ou níveis de representação, que permitem que os modelos aprendam representações de dados com vários níveis de abstração. [18] [26]

Geralmente o DL é utilizada para identificar padrões em grandes conjuntos de dados e fazer previsões, tendo a capacidade de aprender a realizar tarefas de classificação, recorrendo a imagens, textos, sinais ou sons. O DL é um subconjunto de ML que tem a capacidade de resolver problemas mais complexos, conseguindo adquirir graus de exatidão superiores aos obtidos com os métodos de ML devido à sua arquitetura subjacente. [21] [26] [30]

Os modelos de DL são compostos por vários modelos lineares simples que estão organizados em série, sem linearidades intermédias para promover uma representação complexa da informação. Cada uma das séries é denominada de "camada", sendo que o número destas camadas contribuem para o nome deste modelo, "profunda". Estes modelos simples transformam a representação a um determinado nível (começando com os dados de entrada não processados) numa representação num nível superior, gradualmente mais abstrato consoante o aumento dos níveis. Com a integração de um número adequado de transformações, torna-se possível a aprendizagem de conceitos complexos. [18] [30]

Na Figura 5.2 está ilustrado uma arquitetura de DL para a extração, com maior eficácia, de características relevantes dos sinais de ECG. A rede neuronal ilustrada foi definida com três camadas, uma camada de entrada (no lado esquerdo, identificada de L1 a L4, com a cor bordô), uma camada oculta (ao centro, identificada de N1 a N5, com a cor vermelha) e uma camada de saída (no lado direito, identificada por y0 e y1, com a cor salmão).

A seleção do número de camadas ocultas, ou intermédias, e do número de neurónios para a rede neuronal é importante para se conseguir uma arquitetura mais competente. O número de neurónios à entrada corresponde ao número de parâmetros disponibilizados e à saída corresponde às opções de classificação. Na arquitetura de exemplo da Figura 5.2 a camada de entrada é constituída por quatro

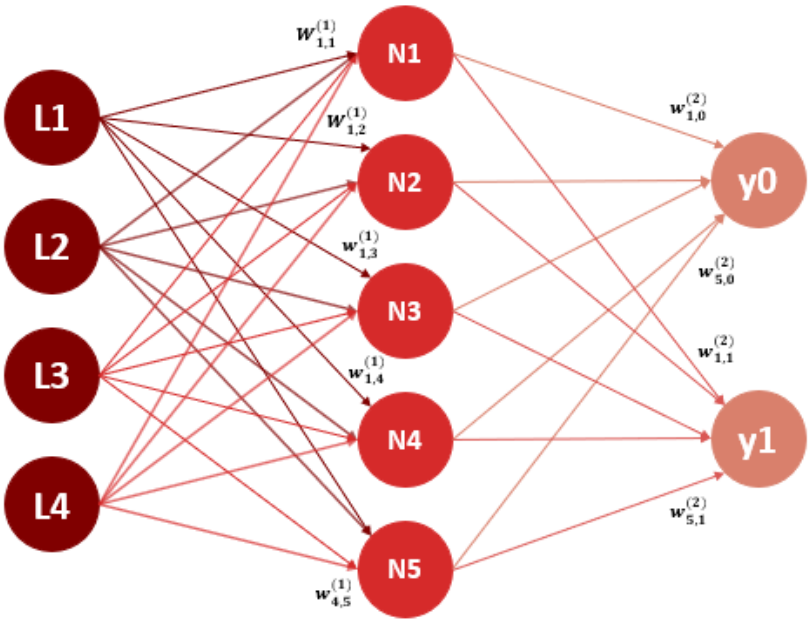

Figura 5.2: Arquitetura de um modelo de Deep Learning. [11]

neurónios e a camada de saída por dois neurónios. A decisão do número de camadas ocultas e do número de neurónios destas, geralmente, é por tentativa e erro, no entanto podem ser tidos em consideração alguns aspetos.

Para a seleção do número de camadas ocultas é necessário determinar a linearidade dos dados. Se os dados fossem lineares não seria utilizado um modelo de DL, este apenas é utilizado para dados não lineares logo, nesta situação não seriam necessárias camadas ocultas. Se os dados forem menos complexos, com pequenas dimensões ou características , então apenas será necessária uma ou duas camadas ocultas. Caso os dados sejam mais complexos, de grandes dimensões ou com muitas características, então para que o modelo seja otimizado podem ser utilizadas entre três a cinco camadas ocultas. Consoante se aumenta o número de camadas ocultas, maior será a complexidade do modelo o que fará com que o *overfitting* seja uma possibilidade. [30]

A seleção do número de neurónios nas camadas ocultas é decidido com base nas dimensões na camada de entrada e de saída, isto é, o número de neurónios desta camada deveria ser $\frac{2}{3}$ da dimensão da camada de entrada mais a dimensão da camada de saída e número de neurónios deve de ir diminuindo nas camadas seguintes, consoante se aproxima da camada de saída. No exemplo anterior o número de neurónios ideal seriam 4.67, ou seja, os 5 neurónios utilizados. Se o número de neurónios for reduzido poderá resultar em *underfitting*, no entanto se for selecionado um número de neurónios demasiado alto poderá dar origem a *overfitting*. [15]

É importante ter em consideração que estas indicações podem e devem ser moldadas de acordo com o caso de utilização, em que o número de camadas ocultas e de neurónios selecionados podem ser superior ao ideal e que estes últimos podem aumentar nas camadas seguintes.

Tendo em consideração a arquitetura da Figura 5.2, vamos analisa-la do ponto de vista matemático. Em notação vetorial, designamos por ρ^k a função de ativação na camada k, por X os parâmetros de entrada, N é o resultados neuronais na camada intermédia. W^k corresponde aos pesos dos neurónios na camada k, Y_{prev} corresponde aos valores de saída, as previsões, Y_{real} os valores esperados e B^k é o vetor *bias* (ou tendência).

$$N = \begin{bmatrix} N_1 \\ \vdots \\ N_5 \end{bmatrix} \quad L = \begin{bmatrix} L_1 \\ \vdots \\ L_4 \end{bmatrix} \quad Y_{\text{prev}} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \quad Y_{\text{real}} = \begin{bmatrix} \bar{y}_0 \\ \bar{y}_1 \end{bmatrix} \quad B^{(1)} = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_5^{(1)} \end{bmatrix} \quad B^{(2)} = \begin{bmatrix} b_1^{(2)} \\ b_5^{(2)} \end{bmatrix}$$

O pretendido é quantificar o valor de cada neurónio seguinte. Para tal, começando com os valores de entrada (L), é necessário proceder ao cálculo dos pesos, previamente indicando qual a importância de cada parâmetro, com as tendências em função dos resultados, para que o modelo possa ”aprender”. O modelo quando é guardado apenas armazena os resultados dos pesos e da tendência.

Assim, na forma vetorial temos as equações

$$N = \rho^1(w^{(1)}L - B^{(1)}) \tag{5.1}$$

$$Y_{\text{prev}} = \rho^2(w^{(2)}N - B^{(2)}) \tag{5.2}$$

que correspondem a

$$\begin{bmatrix} N_1 \\ \vdots \\ N_5 \end{bmatrix} = \rho^1 \left(\begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} & w_{1,4}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{5,1}^{(1)} & w_{5,2}^{(1)} & w_{5,3}^{(1)} & w_{5,4}^{(1)} \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_4 \end{bmatrix} - \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_5^{(1)} \end{bmatrix} \right) \tag{5.3}$$

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \rho^2 \left(\begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} & w_{1,3}^{(2)} & w_{1,4}^{(2)} & w_{1,5}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} & w_{2,3}^{(2)} & w_{2,4}^{(2)} & w_{2,5}^{(2)} \end{bmatrix} \begin{bmatrix} N_1 \\ \vdots \\ N_5 \end{bmatrix} - \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} \right) \tag{5.4}$$

O objetivo é determinar os pesos W^k e *bias* B^k tais que

$$Y_{\text{prev}} \text{ seja o mais próximo possível de } Y_{\text{real}}$$

Para isso define-se uma função de custo, dependente dos pesos e das *bias*, que se assume ter todas as propriedades suficientes para a existência de um minimizante. Existem várias formas de definir a função a minimizar, sendo a forma clássica a baseada na norma euclidiana 5.5:

$$\text{Custo} = \frac{1}{2}\sum_{i=0}^{1}(y_i - \bar{y}_i)^2 \tag{5.5}$$

No caso das previsões binárias, como em estudo (o resultado previsto será 0, ou 1) é frequente usar a função *Log Loss* baseada na média do simétrico do logaritmo da máxima verosimilhança

$$\text{Log Loss} = -\frac{1}{2}\sum_{i=1}^{2}(\bar{y}_i \cdot \log(y_i) + (1 - \bar{y}_i) \cdot \log(1 - y_i)) \tag{5.6}$$

No *TensorFlow* esta função é identificada por `binary_crossentropy`

Assim definida, a função *Log Loss* depende dos pesos e das *bias* de todas as camadas, os quais se podem determinar por um método de gradiente descendente (GD), gradiente descendente estocástico (SGD) ou ainda pelo poderoso método *Adaptive Moment Estimation* (Adam). [4]

Os principais algoritmos utilizados em DL, que permitem criar diferentes arquiteturas de redes neuronais profundas, são: [18] [26]

- CNN (*Convolutional Neural Network*): Rede neuronal artificial em que a ligação entre os neurónios da rede é inspirada no córtex visual, sendo do tipo "*Feed-Forward*". Este algoritmo tira partido das propriedades dos sinais naturais recorrendo a ligações locais, pesos partilhados, agrupamento e utilização de várias camadas. Esta arquitetura está organizada como uma série de fases em que as primeiras fases são constituídas por dois tipos de camadas, camadas convolucionais (deteção de conjunções locais de características da camada anterior) e camadas de agrupamento (junção de características semelhantes numa só).

- RBM (*Restricted Boltzmann Machine*): Rede neuronal artificial que permite a aprendizagem através de um conjunto de entradas, recorrendo a métodos aleatórios.

- DBN (*Deep Belief Networks*): Rede neuronal profunda, constituída por várias camadas escondidas, conectadas entre si, mas não entre os seus constituintes.

- SAE (*Stacked Auto-Encoders*): Utilizam matrizes DBN para codificar e descodificar os dados de entrada. Aos dados de entrada é adicionado ruído aleatório e, posteriormente, obriga-se o sistema a obter saídas que correspondam às entradas sem o ruído. Com a adição do ruído aleatória a dimensão dos dados de treino aumenta e, consequentemente, o *overfitting* reduz.

6. Resultados

O presente trabalho compreende duas etapas, a primeira é o processo de recolha de dados, que consiste na extração dos parâmetros necessários à classificação do sinal. A segunda etapa consiste na análise e processamento dos dados extraídos, recorrendo a um método de DL. Todas as etapas foram implementadas em *Python*.

Para este trabalho recorreu-se à base de dados de acesso livre *MIT-BIH Arrhythmia Database* [23]. Esta base de dados contém 48 excertos de meia hora de registos de ECG, obtidos de 47 indivíduos estudados pelo Laboratório de Arritmia, do Hospital Beth Israel de Boston, entre 1975 e 1979. Apesar de alguns parâmetros extraídos estarem fora dos valores indicados na Tabela 2.4, foi considerada válida a informação constante da base de dados do MIT. Desde o momento em que uma patologia foi assinalada na base de dados, o ECG foi considerado, para a segunda etapa, como sendo patológico, sendo realizada uma classificação binária (em patológico ou saudável) independentemente do grau de patologia associada. Para a realização da segunda etapa, foram objeto de estudo 44 sinais dos 48 que compõe a base de dados original.

A análise de sinais de ECG's têm associada uma elevada complexidade sendo bastante complicada a sua classificação, inclusive a nível médico, pois esta depende do indivíduo, de patologias associadas, do seu estilo de vida, idade, entre diversos outros fatores.

6.1 Recolha de dados

Para esta etapa foi desenvolvido um procedimento em *Python* para leitura e parametrização dos sinais recolhidos, cujo código se apresenta no capítulo A, Apêndice 1.

Com o objetivo de potencializar o algoritmo, começamos ativar as bibliotecas pretendidas, que correspondem às linhas de código 10 a 15, Figura 6.1. Posteriormente foi necessário desenvolver uma função, linhas de código 3 a 8 onde o objetivo é identificar o maior valor que receciona, neste caso irá determinar qual a maior diferença verificada de distância entre picos R.

```
3   def max_2(n):
4       a = np.zeros(np.size(n, 0)-1)
5
6       a = [ int( n[k+1]-n[k] )  for  k in range(np.size(n, 0)-1)]
7
8       return np.amax(a)
9
10  import wfdb
11  from wfdb import processing
12  import matplotlib.pyplot as plt
13  import numpy as np
14  from scipy.signal import find_peaks, peak_widths
15  from skimage.restoration import (denoise_wavelet, estimate_sigma)
```

Figura 6.1: Bibliotecas utilizadas no algoritmo.

Em seguida, na linha 17 foi definida a *wavelet* utilizada em todos os sinais, a *sym5*, no entanto foi necessário fazer testes prévios com outras *wavelets*, de modo a assegurar que esta seria a opção mais adequada aos sinais estudados. Esta etapa é muito importante no contexto global de análise, uma vez que se pretende ”suavizar”a curva do gráfico do sinal, de forma a eliminar o ruído. É importante salientar que um dos sinais, que posteriormente foi removido, necessitou que a *wavelet* fosse alterada para uma *db5*. Na Figura 6.2, é realizada a leitura do sinal pretendido, linha 21, e a abertura do ficheiro *”NormPat.dat”* com a opção de adicionar os parâmetros à lista (identificado com a letra *a* = *append*, as outras duas opções seriam de escrita, substituindo continuamente o texto no ficheiro, com a letra *w* = *write*, e a de leitura, com a letra *r* = *read*). Na linha 25 é definida a dimensão da amostra, baseado na informação constante na base de dados do MIT. Nas linhas 26, 28 e 30 guarda-se as informações do ficheiro, onde a variável ”ecg”corresponde ao sinal que será tratado e é determinada a frequência do sinal.

```
22  file = '212'
23  f = open("NormPat.dat", "a")
24
25  amostra = 3000
26  record = wfdb.rdrecord(file, sampto=amostra)
27
28  ecg = record.p_signal[:, 0]
29
30  frequency = record.fs*1.
```

Figura 6.2: Identificação de variáveis base.

Na Figura 6.3, na linha 32, obtém-se o gráfico com o registo da base de dados que é definido como controlo, linhas 34 a 36, nos passos seguintes.

```
32  ann = wfdb.rdann(file, 'atr', sampto=amostra)
33
34  wfdb.plot_wfdb(record=record, annotation=ann, plot_sym=True,
35                 time_units='seconds', title=file,
36                 figsize=(10,4), ecg_grids='all')
```

Figura 6.3: Abertura do ECG com identificação de pontos R.

Um exemplo do por defeito é gráfico obtido é a Figura 6.4, onde estão identificados os picos R do sinal. Salienta-se que o gráfico superior está representado em ”MLIImV”refere-se a ”milímetros por milivolt”e o inferior está em ”milivolt”um milivolt é igual a um milésimo de volt. Não sendo essencial, esta etapa destina-se a controlo.

Em seguida procede-se à conversão da abcissa do sinal para segundos, que consiste na divisão do tamanho do sinal pela frequência, linha 38 da Figura 6.5. Na linha 40 é aplicada a *wavelet* selecionada ao sinal em estudo, sendo necessário definir o número de níveis adequados para otimizar a sua suavização e remoção de ruído, sem comprometer a extração de dados. Nas linhas 42, 44 e 46 são encontrados os picos R, calculado a distância máxima entre picos R e calculado o número de bpm, respetivamente. Para o cálculo dos bpm recorre-se ao número de picos R calculado, converte-se para segundos e depois multiplica-se por 60 para que seja dado em minutos.

Na Figura 6.6 são construídos três vetores para os picos P, Q e S, individualmente, nas linhas 48 a 50, e em seguida são definidas duas vizinhanças ”v_P1”e ”v_P2”para que se consiga determinar à volta do pico R as ondas pretendidas. A onda P estará englobada num determinado intervalo, linha 59, entre a posição que corresponde ao pico R menos a vizinhança ”v_P1”definida e o próprio pico R. Sabe-se que o

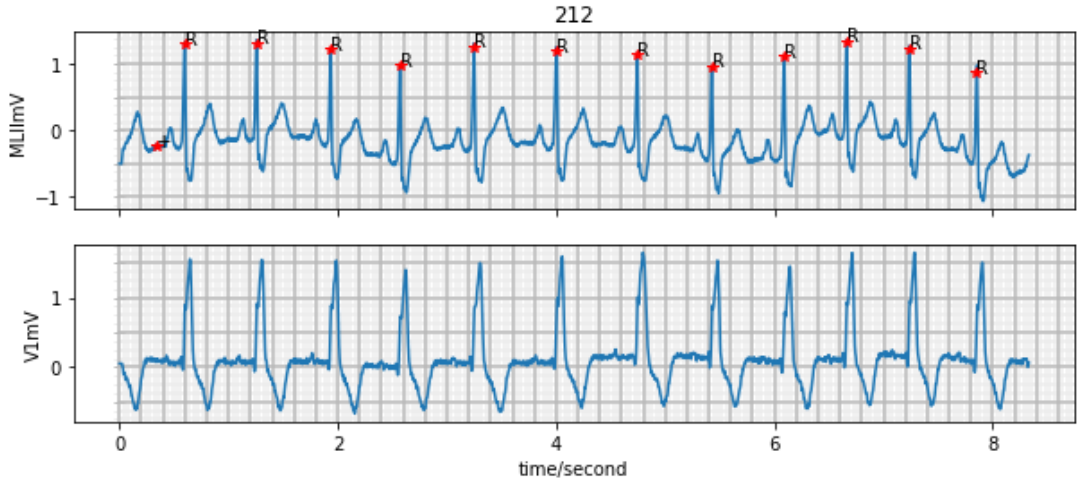

Figura 6.4: Exemplo do gráfico de referência de um dos sinais.

```
38  time_data = np.arange(ecg.size) / frequency
39
40  y= denoise_wavelet(ecg, sigma=2, wavelet=waveletname, wavelet_levels=3)
41
42  pico_R, _ = find_peaks(y, height= np.mean(y), distance=200 )
43
44  RR_max = 1000.*max_2(pico_R)/frequency
45
46  BPM = pico_R.shape[0]*60/(amostra/frequency)
```

Figura 6.5: Cálculo de algumas variáveis e aplicação da *wavelet* ao sinal.

ponto Q corresponde ao final da onda P, logo é necessário recuar um pouco no pico R para que possa ser determinado o mínimo dessa zona, que corresponde ao ponto Q, linha 61. Através do ponto Q sabe-se que atrás encontra-se o ponto P, sendo este calculado nas linhas 63 e 64. Em seguida é feita a soma de todas as distâncias PR determinadas e na linha 67 para determinar o ponto S foi necessário avançar no pico R, para que seja possível proceder ao cálculo da distância do complexo QRS, linha 69.

É importante ressalvar que o cálculo dos picos P e dos complexos QRS apenas é realizado àqueles que se encontram no interior do sinal, isto é, o primeiro e último pico R para os cálculos destes parâmetros é desprezado, para que se possa evitar possíveis erros.

Posteriormente, na Figura 6.7, é determinada a média do intervalo PR e do complexo QRS, sendo feita a sua conversão para segundos ao dividir pela frequência. Na linha 74 é determinada a largura da onda P a uma altura relativa de, por exemplo, 40% para todas as ondas, ou seja, se esta altura subir ou descer para uma onda, o mesmo acontecerá para as outras ondas, logo, uma altura relativa demasiado baixa poderá implicar que a largura da onda compreende uma porção do sinal maior que a onda em si, demasiado alta fica uma largura reduzida que se traduz numa duração reduzida, o que não se aproxima do valor real previsto. Na linha 75 é realizada a conversão para ms.

Entre as linhas 77 e a 95 é feita a impressão dos gráficos, na mesma imagem, num vetor com 4 linhas e 1 coluna, na posição 0 é o sinal original, posição 1 está o sinal reconstruido, ou seja sem ruído e com os picos R identificados, a posição 2 diz respeito ao sinal reconstruido com os picos P identificados, enquanto a posição 3 identifica um sinal com os pontos correspondentes ao complexo QRS assinalados.

As linhas 98 a 101 são para extrair os bpm calculados e fornecidos pela própria base de dados, para

```
pico_P = np.zeros(np.size(pico_R)-2,dtype=int)
Q      = np.zeros(np.size(pico_R)-2,dtype=int)
S      = np.zeros(np.size(pico_R)-2,dtype=int)

v_P1 = 30
v_P2 = 60

QRS = 0
PR  = 0

for  i in range(1,np.size(pico_R)-1):
    p_w       = np.argmin(y[pico_R[i]-v_P1 : pico_R[i]])
#
    Q[i-1]      = pico_R[i]- v_P1 + p_w
#
    pico_P[i-1] = pico_R[i]  - v_P2 \
                + np.argmax( y[pico_R[i]-v_P2 : pico_R[i]- v_P1 + p_w] )
    PR = PR + pico_R[i] -pico_P[i-1]
#
    s_w       = np.argmin(y[pico_R[i] : pico_R[i]+v_P1])
    S[i-1]      = pico_R[i] + s_w
    QRS = QRS + S[i-1] - Q[i-1]
```

Figura 6.6: Determinação das ondas que constituem o complexo QRS.

```
PR = 1000.*(PR/np.size(pico_P))/frequency
QRS = 1000.*(QRS/np.size(Q))/frequency

cpico_P = peak_widths(y, pico_P, rel_height=.4)
OP_max = 1000*np.mean(cpico_P[0])/frequency
```

Figura 6.7: Cálculo do intervalo PR, complexo QRS e largura da onda P.

que sirvam de ponto de comparação e auxiliem a determinar a eficácia do algoritmo desenvolvido.

```
rr = wfdb.processing.ann2rr(file, 'atr', pn_dir=None, start_time=None,
                            stop_time=None, format=None, as_array=True)

hr = wfdb.processing.calc_mean_hr(rr, fs=1, min_rr=None, max_rr=None, rr_units='samples')
```

Figura 6.8: Extração do bpm padrão fornecido pela base de dados.

Como foi mencionado anteriormente, foi necessário remover da base de dados criada alguns sinais que, apesar de ser claro que existe um patologia associada, não foi possível ajustar o algoritmo de modo a conseguir realizar uma leitura e extração de dados fidedigna, um exemplo de um desses sinais é a Figura 6.9.

Neste sinal, Figura 6.9 é possível observar de forma imediata uma discrepância na distância entre picos R que terá influência no ajuste do algoritmo. A identificação de todos os picos R do sinal, assinalados a amarelo na figura, implica que pontos aleatórios entre picos também sejam identificados, uma vez que este primeiro passo está comprometido e não é possível concretizar com precisão, todos os passos seguintes também estarão incorretos, isto é, a identificação das ondas P será superior ao real e os pontos do complexo QRS não corresponderão. No entanto, havia a possibilidade de aumentar a distância entre picos R, na fase de deteção, o que faria com que os pontos aleatórios detetados e que não correspondem a picos R desaparecem-se, porém o pico central na correnteza de três picos R também seria eliminado, ou seja, identificando um número inferior ao real de picos R e, consequentemente, um número inferior ao real de picos P.

Figura 6.9: Exemplo de um sinal removido dos testes.

Por sua vez, um exemplo de um bom sinal onde são facilmente identificáveis os seus constituintes e, desta forma, consegue-se confirmar que todos os parâmetros pretendidos estão a ser bem identificados, é o sinal das Figuras 6.10, 6.11,

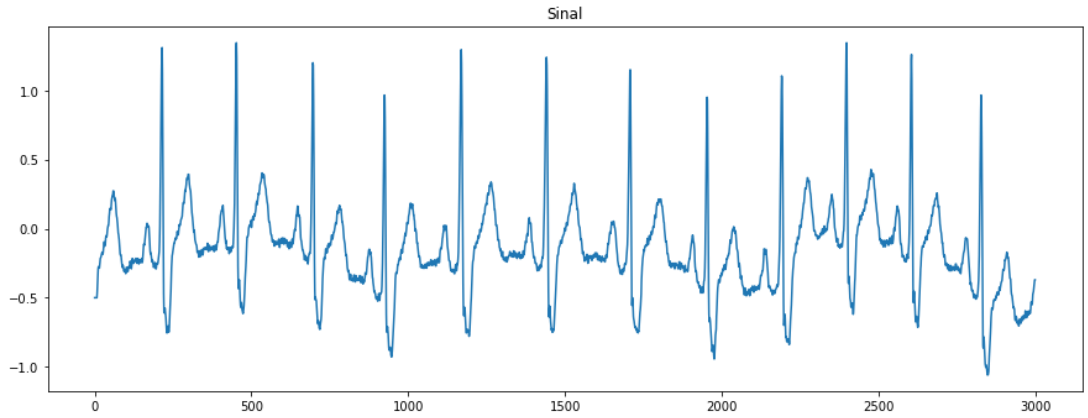

Figura 6.10: Exemplo de um sinal original bom.

Após a conclusão do desenvolvimento do algoritmo foi necessário proceder à realização de vários testes, ajustando os parâmetros necessários consoante cada um dos sinais em estudo.

Na Tabela 6.1 são identificados dois conjuntos de informações, as colunas 2 à 6 têm indicadas os parâmetros de análise, ajustados manualmente para cada um dos sinais, neste conjunto de dados são identificados o número de níveis da *wavelet*, a distância necessária para a identificação dos picos R, as vizinhanças dos picos R e a altura do pico P, que se traduz no cálculo da duração deste pico. O segundo conjunto, das colunas 7 à 11 com a tonalidade cinza, corresponde aos parâmetros extraídos, onde são obtidos os bpm calculados pelo o algoritmo, os bpm que a própria base de dados disponibiliza, a coluna inteira deste parâmetro está identificada por uma tonalidade cinza claro para fazer a distinção, a duração da onda P, o intervalo PR e a duração do complexo QRS, estas últimas três variáveis são dadas em ms. As linhas que estão identificadas, na primeira coluna, com uma tonalidade amarela, correspondem aos sinais considerados normais pela base de dados do MIT. Desta forma, os sinais ditos normais perfazem um total de 16 sinais, enquanto os sinais patológicos perfazem um total de 28.

Os resultados da Tabela 6.1 são apenas uma pequena amostra, sendo que a tabela é apresentada na integra no capítulo B. Apêndice 2.

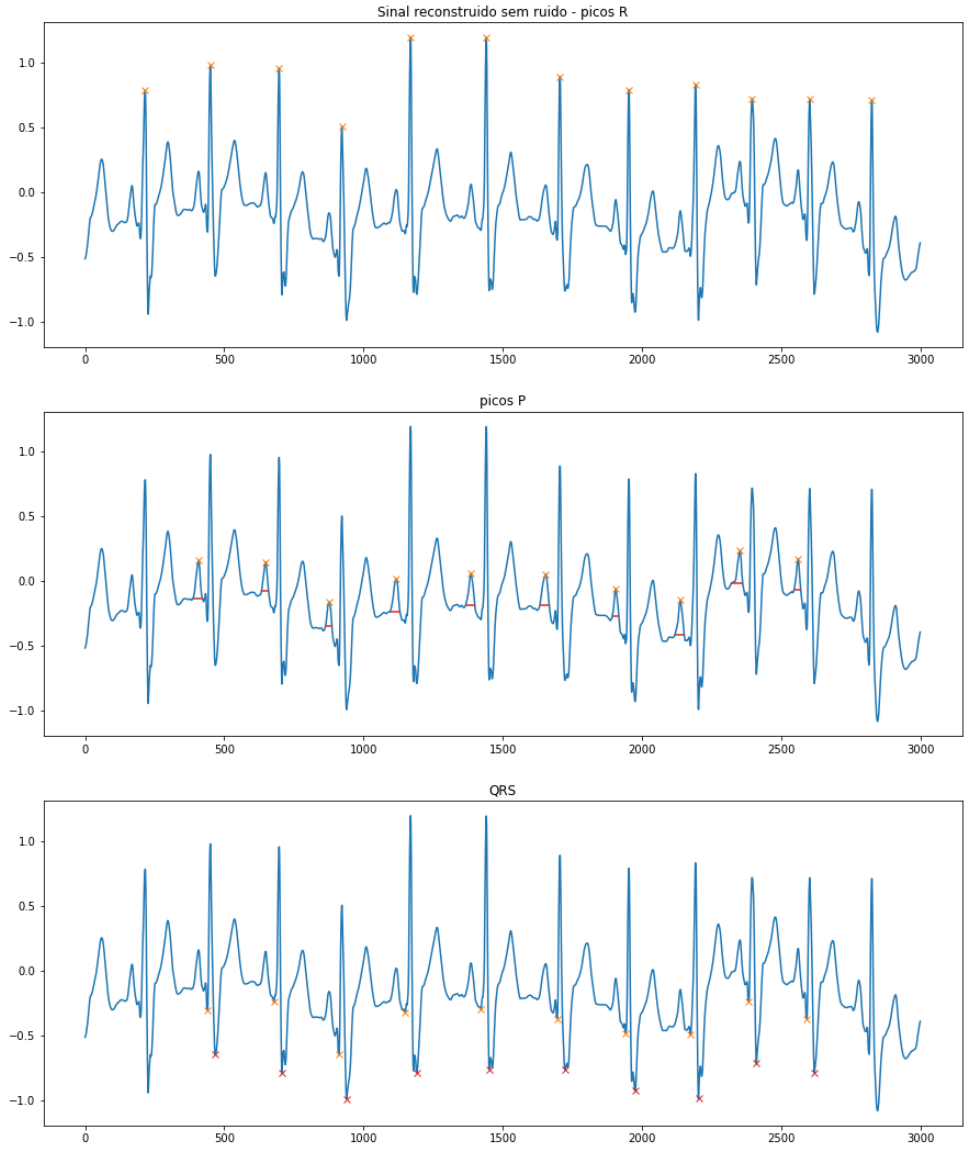

Figura 6.11: Exemplo de um sinal bem identificado. Gráfico superior - identificação de picos R; Gráfico intermédio - identificação dos picos P; Gráfico inferior - identificação do complexo QRS.

Com os parâmetros extraídos, nesta fase, e identificados na Tabela 6.1, irá avançar-se para a segunda etapa deste trabalho, o processamento dos dados extraídos.

Tabela 6.1: Porção da tabela de resultados com indicação dos parâmetros de análise e os extraídos.

ID	Niv	pico_R	v_P1	v_P2	cpico_P	bpm	bpmPa	OP	PR	QRS
1	4	180	30	70	0.3	79.2	75.6	106.7	167.0	109.0
2	4	150	30	60	0.43	72.0	62.3	85.1	142.0	128.2
3	3	180	50	200	0.45	72.0	72.9	145.7	494.4	129.2
4	3	250	23	70	0.75	64.8	69.5	92.6	167.5	75.8
5	2	180	50	170	0.45	79.2	76.8	197.3	385.8	154.9
6	4	180	30	90	0.45	79.2	89.5	94.3	156.8	136.4
7	3	300	30	60	0.65	57.6	69.7	78.8	114.8	75.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
18	4	220	30	100	0.6	57.6	62.4	106.6	109.3	138.0
19	4	220	30	70	0.42	93.6	82.4	1134.0	141.7	126.0
20	4	220	35	80	0.8	50.4	50.5	100.7	188.9	126.1
21	4	280	35	100	0.3	50.4	54.3	117.6	112.8	138.9
22	4	180	40	100	0.9	86.4	67.8	75.3	144.7	130.0
23	4	180	40	100	0.6	57.6	71.3	95.7	121.8	146.8
24	4	120	30	120	0.5	108.0	103.3	69.4	160.9	141.9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
38	3	200	30	100	0.55	79.2	87.5	62.2	119.8	63.0
39	4	200	40	110	0.4	79.2	87.8	104.7	147.5	133.6
40	3	200	40	110	0.8	72.0	71.2	84.3	249.7	76.0
41	3	180	30	90	0.7	86.4	82.0	74.3	157.8	98.3
42	3	250	30	90	0.6	57.6	66.8	95.8	155.1	73.2
43	2	150	40	60	0.6	100.8	104.8	52.1	134.0	108.3
44	3	200	30	60	0.4	93.6	91.9	73.2	132.1	83.08

6.2 Processamento de dados

Para esta etapa recorrer-se-á, mais uma vez, à linguagem de programação *Pyhton* para o desenvolvimento do algoritmo para que se proceda à análise e processamento dos dados, recorrendo a um método de ML. O algoritmo é apresentado na integra no capítulo C. Apêndice 3. Em primeiro lugar é necessário carregar as bibliotecas necessárias para potencializar o algoritmo, das linhas 3 à 16, Figura 6.12.

Seguidamente, procede-se à leitura da base de dados extraída na etapa anterior, ilustrada na Tabela 6.1, indicando que a separação das colunas é feita pela vírgula ",", linha 18 da Figura 6.13. Depois os dados são separados em duas variáveis, os dados de *input* que correspondem às primeiras quatro colunas, na tabela anterior são as colunas com o cinza mais escuro, e os dados de *output* que corresponde à classificação do sinal.

Em seguida, na linha 23, procede-se à conversão do sinal para que depois, na linha 24 se possa proceder à definição dos dados de treino e dos dados de teste, tendo sido utilizados 30% dos dados para teste.

```
3   import os
4   os.environ['KMP_DUPLICATE_LIB_OK']='True'
5   import keras
6   from keras.utils import np_utils
7   from sklearn.model_selection import train_test_split
8   import numpy as np
9   from keras.models import Sequential
10  from keras.models import load_model
11  from keras.layers import Dense, Activation
12  from keras.optimizers import SGD
13  from numpy import loadtxt
14  import matplotlib.pyplot as plt
15  from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
16  from sklearn import metrics
```

Figura 6.12: Bibliotecas utilizadas no algoritmo.

```
18  dataset = loadtxt('./treino.dat', delimiter=',')
19
20  x = dataset[:,0:4]
21  y = dataset[:,4]
```

Figura 6.13: Leitura da base de dados e definição das variáveis de *input* e *output*.

```
23  y_convertido = np_utils.to_categorical(y)
24  x_treino, x_teste, y_treino, y_teste = train_test_split(x, y_convertido, test_size = 0.30)
```

Figura 6.14: Definição dos dados de treino e de teste.

Para que seja possível desenvolver o método de DL é necessário definir o número de camadas, sabendo que a primeira é a de entrada e a última de saída, apenas fica em falta definir o número de camadas ocultas. Este modelo é composto por três camadas, ou seja, apenas tem uma camada oculta que é definida na linha 27. A camada de entrada recorre ao modelo sequencial. Na camada oculta é fundamental definir o número de neurónios utilizados e a função de ativação, estes são ajustados de forma a otimizar o resultado obtido. A camada de saída é constituída por apenas dois neurónios, correspondentes à classificação binária implementada, sendo necessário definir a função de ativação. Como é possível observar na Figura 6.15, numa das simulações realizadas, que está a servir de exemplo, a camada oculta, linha 27, foi definida com 5 neurónios e com a função de ativação *"sigmoid"*, enquanto a camada de saída, linha 28, foi definida com 2 neurónios e com a função de ativação igual à entrada *"sigmoid"*, estes parâmetros foram alterados de modo a otimizar os resultados.

```
26  modelo = Sequential()
27  modelo.add(Dense(5, input_dim=x.shape[1], kernel_initializer='normal', activation='sigmoid'))
28  modelo.add(Dense(2, kernel_initializer='normal',  activation='sigmoid'))
```

Figura 6.15: Definição das camadas do modelo desenvolvido.

Em seguida, de modo a otimizar/minimizar a função de custo obtida é necessário recorrer a um otimizador, neste caso foi utilizado o método de *"adam"*, linha 30 da Figura 6.16. As linhas seguintes, 32 e 33, consistem no treino do modelo onde é definido o número de iterações, neste caso 1000 iterações, uma vez que este é um processo de otimização requer que sejam estabelecidos o número de iterações para cada etapa.

```
30  modelo.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
31
32  resultado = modelo.fit(x_treino, y_treino, epochs=1000, batch_size=150,
33                         validation_data=(x_teste, y_teste))
```

Figura 6.16: Definição do modelo de treino e do otimizador.

Na linha 35 são feitas as previsões da classificação e, em seguida, é aplicado o modelo aos dados de teste para determinar a percentagem de erro nas previsões. Idealmente, com os sinais de teste disponibilizados o modelo aprendeu, com um erro de 0 ou próximo de 0, a reconhecer um sinal normal ou um sinal patológico. A avaliação da rede neuronal é feita nas linhas 40 e 43, sendo o valor determinado impresso nas linhas 41 e 44.

```
35  previsoes = modelo.predict(x_teste)
36
37  np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})
38  print(previsoes)
39
40  _, acc_treino = modelo.evaluate(x_treino, y_treino)
41  print('Precisão treino: %.2f ' % (acc_treino*100))
42
43  _, acc_teste = modelo.evaluate(x_teste, y_teste)
44  print('Precisão teste: %.2f ' % (acc_teste*100))
```

Figura 6.17: Aplicação do modelo desenvolvido aos dados de teste.

Por fim, são impressos os gráficos de precisão, de custo e a matriz de resultados obtidos, sendo o modelo de teste criado e guardado para futuras aplicações, por exemplo, utilizar o modelo desenvolvido para classificar conjuntos de dados novos e não classificados.

Desta forma, utilizando um método de DL de 3 camadas, com uma camada oculta com 5 neurónios, foi implementado um modelo keras usando 1000 iterações e uma dimensão de amostra de 150 para a determinação dos pesos. A precisão do modelo determina a percentagem de valores previstos que correspondem aos valores reais, se os dois valores forem iguais o modelo é considerado preciso.

		Classificação Real	
		Saudável	**Patológico**
Classificação Prevista	**Saudável**	N1	N2
	Patológico	N3	N4

Figura 6.18: Ilustração de uma matriz de resultados em valores relativos.

A matriz de resultados, também denominada de matriz de confusão, foi adicionada posteriormente ao código para auxiliar na visualização dos resultados obtidos, permitindo a identificação da quantidade de previsões realizadas de forma correta, quer para sinais patológicos como saudáveis, e na quantidade de previsões realizadas incorretamente, na Figura 6.18 está ilustrado como é desenvolvida uma matriz de

resultados. A diagonal sombreada a verde corresponde às previsões corretas, enquanto a diagonal laranja diz respeito às previsões incorretas, ou seja, um valor que seja identificado na célula N3 corresponderá a uma classificação prevista como sendo patológica, quando na realidade trata-se de uma classificação saudável.

A Tabela 6.2 identifica uma amostra dos testes realizados para determinar o conjunto de parâmetros que otimizaria o desenvolvimento do modelo, minimizando a função de custo. A tabela com o conjunto de resultados é apresentada na integra no capítulo D. Apêndice 4.

Na Tabela 6.2, mais uma vez, são identificados dois conjuntos de informações, das colunas 2 a 6 são definidos os parâmetros de teste, enquanto as colunas 7 e 8 correspondem aos valores obtidos para o conjunto de treino e o conjunto de teste. De acordo com os valores obtidos e com os gráficos de cada teste, o modelo que conseguiu otimizar a função de custo é o número 26, identificado a verde na tabela. Este é constituído por um conjunto de dados de teste de 30% da amostra, o ativador utilizado na camada oculta é o *"sigmoid"*, com 5 neurónios, que é o mesmo ativador utilizado na camada de saída, enquanto o otimizador utilizado é o *"adam"*.

Tabela 6.2: Porção da tabela dos resultados obtidos, com indicação dos parâmetros testados e os resultados de de precisão de treino e teste.

ID	Test_size	Ativador1	Ativador2	Otimizador	Neurónios	Prec. Treino	Prec. Teste
1	0.3	sigmoid	sigmoid	adam	6	86	61
2	0.3	sigmoid	sigmoid	adam	5	80	69.23
3	0.3	sigmoid	sigmoid	adam	4	80	69.23
4	0.2	sigmoid	sigmoid	adam	6	70.59	66.67
5	0.2	sigmoid	sigmoid	adam	5	79.41	77.78
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
22	0.3	softmax	selu	adam	5	63.33	61.54
23	0.3	softmax	elu	adam	5	63.3	61.54
24	0.3	softmax	relu	adam	5	63.33	61.54
25	0.3	sigmoid	sigmoid	adam	4	83.33	76.92
26	0.3	sigmoid	sigmoid	adam	5	70	84.62
27	0.3	sigmoid	sigmoid	adam	6	66.67	84.62
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
45	0.3	sigmoid	tanh	adam	4	60	53.85
46	0.3	sigmoid	tanh	sgd	4	66.67	53.85
47	0.3	tanh	relu	adam	4	63.33	53.85
50	0.3	sigmoid	softmax	adam	5	57.14	68.18

A precisão do modelo determina a percentagem de valores previstos que correspondem aos valores reais, se os dois valores forem iguais o modelo é considerado preciso. A Figura 6.19 corresponde à extração do gráfico representativo da função de precisão, a Figura 6.20 corresponde à representação da função de custo e a Figura 6.21 corresponde à matriz de resultados do modelo. Na figura correspondente à função de precisão observa-se que a precisão de validação do modelo foi superior à precisão do treino do modelo com 70% e 84.62%, respetivamente, estes valores estão identificados a verde na Tabela 6.2.

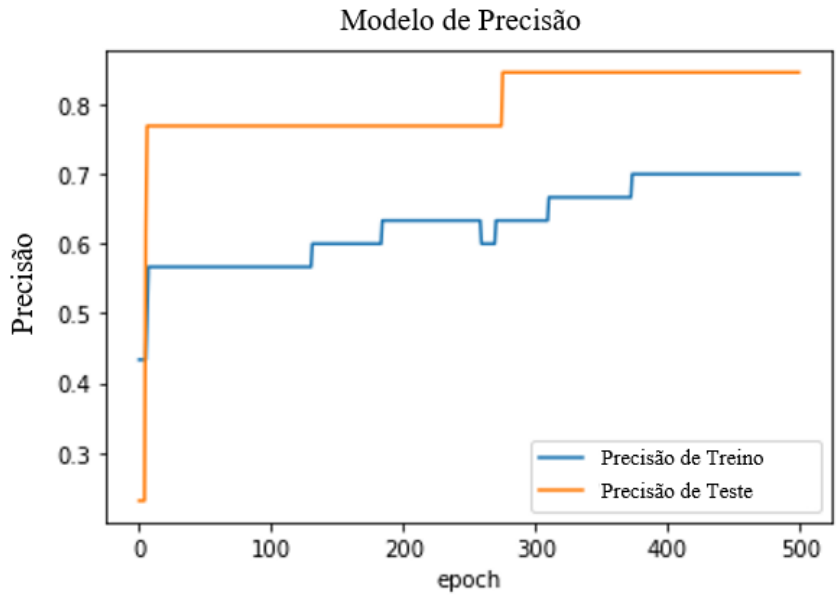

Figura 6.19: Função de precisão do modelo número 26.

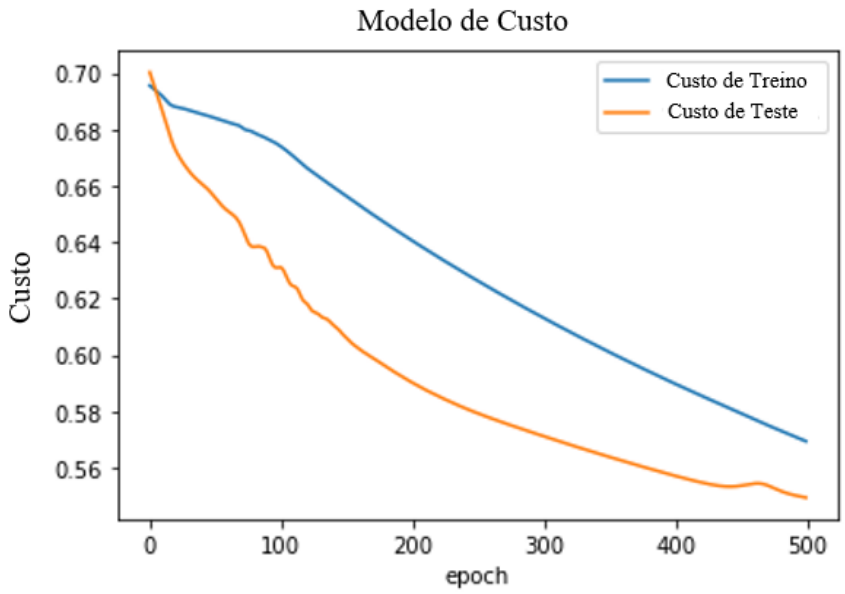

Figura 6.20: Função de custo do modelo número 26.

Uma vez que os resultados obtidos não corresponderam às previsões realizados no início do projeto, recorrendo à Tabela 2.4 procedeu-se à simulação paramétrica para os sinais de ECG, dividindo estes resultados em saudáveis e patológicos, com as respetivas classificações. Para tal, foi necessário desenvolver um novo algoritmo em *Pyhton* para criar uma base de dados aleatória, baseada em parâmetros reais, que servirá para treino do modelo. O algoritmo é apresentado na integra no capítulo E. Apêndice 5.

Inicialmente, à semelhança dos algoritmos anteriores, foi necessário carregar as bibliotecas necessárias para a randomização dos valores, seguindo-se a abertura do ficheiro *"treino.dat"* com a opção de adicionar dados.

		Previsto	
		Patológico	**Saudável**
Real	**Patológico**	1	0.17
	Saudável	0	0.83

Figura 6.21: Matriz de resultados do modelo número 26.

Em seguida, foram definidos os limites das características que correspondiam a um sinal saudável, linhas 7 e 8, e os limites que correspondiam a um sinal patológico, linhas 10 e 11.

```
7   a = [60, 80, 120, 50]
8   b = [100, 110, 200, 110]
9
10  pa = [30, 110, 80, 140]
11  pb = [100, 200, 200, 200]
```

Figura 6.22: Definição dos limites das características dos sinais.

A base de dados desenvolvida é constituída por uma amostra elevada, correspondendo a 60.000 sinais saudáveis e a 60.000 sinais patológicos, perfazendo uma amostra com uma dimensão de 120.000 sinais de treino.

Seguidamente, com recurso a um ciclo *"for"* foi criada a amostra aleatória de sinais saudáveis, indicando apenas os parâmetros pretendidos, linhas 20 a 23, e escrevendo ao mesmo tempo no ficheiro os valores obtidos, linhas 24 e 25, Figura 6.23.

```
19  for _ in range(N): # saudaveis
20      s0 = a[0] + (random() * (b[0] - a[0]))
21      s1 = a[1] + (random() * (b[1] - a[1]))
22      s2 = a[2] + (random() * (b[2] - a[2]))
23      s3 = a[3] + (random() * (b[3] - a[3]))
24      f.write('\n%.1f,%.1f,%.1f,%.1f,%.0f' %
25              (s0,s1,s2,s3,t1))
```

Figura 6.23: Obtenção da amostra aleatória de sinais saudáveis.

Por fim, antes da conclusão do algoritmo, recorre-se a outro ciclo *"for"* para criar de forma aleatória a amostra correspondente aos sinais patológicos com os parâmetros pretendidos, linhas 28 a 35, no entanto, neste caso foi necessário excluir do intervalo de valores pretendidos, os valores que correspondiam a sinais saudáveis, tendo sido feito um cálculo para que, por exemplo, se um valor gerado *s* na posição 0

fosse superior ao valor definido no vetor *a* nessa mesma posição, será realizada uma soma para que este valor passe a estar fora do intervalo dos sinais saudáveis. Este ligeiro ajuste foi necessário aplicar em dois parâmetros que correspondiam às linhas 30 e 34, como se pode observar na Figura 6.24.

```
27  for _ in range(N): #  patologicos
28      s0 = pa[0] + (random() * (pb[0] - pa[0]))
29      if s0 > a[0]:
30          s0 = s0 + 41
31      s1 = pa[1] + (random() * (pb[1] - pa[1]))
32      s2 = pa[2] + (random() * (pb[2] - pa[2]))
33      if s2 > a[2]:
34          s2 = s2 + 81
35      s3 = pa[3] + (random() * (pb[3] - pa[3]))
36      f.write('\n%.1f,%.1f,%.1f,%.1f,%.0f' %
37              (s0,s1,s2,s3,t2))
```

Figura 6.24: Obtenção da amostra aleatória de sinais patológicos.

Com a obtenção e utilização desta base de dados aleatória, é necessário ajustar o algoritmo anterior para que seja realizada uma leitura de treino e uma leitura de teste. A leitura de treino recorre aos dados aleatórios obtidos, do ficheiro "treino.dat", enquanto a leitura de teste recorre aos dados completos da base do MIT, do ficheiro "Valores.csv". O algoritmo, após estas alterações, pode ser consultado na integra no capítulo F. Apêndice 6.

A seguinte alteração que foi necessário fazer passou pela separação dos dados, no algoritmo anterior ilustrado na Figura 6.14 linha 24, os dados de treino e de teste eram separados recorrendo a uma função de *"split"*, definindo a dimensão da amostra de treino, neste algoritmo foi realizada a leitura e definido, nas linhas 24 e 25, os dados de treino, nas linhas 27 e 28 os dados de teste e nas linhas 30 e 31 foi realizada a conversão dos valores do *y* para as duas amostras, Figura 6.25.

```
24  x_tre = data_treino[:,0:4]
25  y_tre = data_treino[:,4]
26
27  x_tes = data_teste[:,0:4]
28  y_tes = data_teste[:,4]
29
30  y_trec = np_utils.to_categorical(y_tre)
31  y_tesc = np_utils.to_categorical(y_tes)
```

Figura 6.25: Separação dos dados para treino e teste.

Na Tabela 6.3 encontram-se identificados das colunas 2 a 6 são definidos os parâmetros selecionados, enquanto as colunas 7 e 8 correspondem aos valores de precisão obtidos para o conjunto de treino e o conjunto de teste, respetivamente. De acordo com os valores obtidos e com os gráficos de cada teste, o modelo que conseguiu otimizar a função de custo é o número 54, destacado a verde na tabela. Para alcançar este resultado fora realizadas 50 iterações, selecionando o *"relu"* como ativador da camada oculta, com 4 neurónios, e a *"tanh"* como ativador utilizado na camada de saída, o método de *"adam"* permanece o melhor otimizador para estes conjuntos de dados.

A Figura 6.26 corresponde ao gráfico da função de precisão obtido, a Figura 6.27 corresponde à representação da função de custo e a Figura 6.28 corresponde à matriz de resultados do modelo identificado como sendo o melhor. A precisão de treino do modelo corresponde a 59.83% e a precisão de teste corresponde a 60.47%, valores que estão sublinhados a verde 6.3.

ID	Neurónios	Epochs	Ativador1	Ativador2	Otimizador	Prec. Treino	Prec. Teste
51	5	20	relu	relu	adam	50.0	37.21
52	4	50	relu	relu	adam	69.64	53.49
53	4	50	tanh	relu	adam	83.62	41.87
54	4	50	relu	tanh	adam	59.83	60.47
55	4	30	relu	tanh	adam	69.55	41.86

Tabela 6.3: Resultados obtidos recorrendo a duas bases de dados, uma de treino e outra de teste.

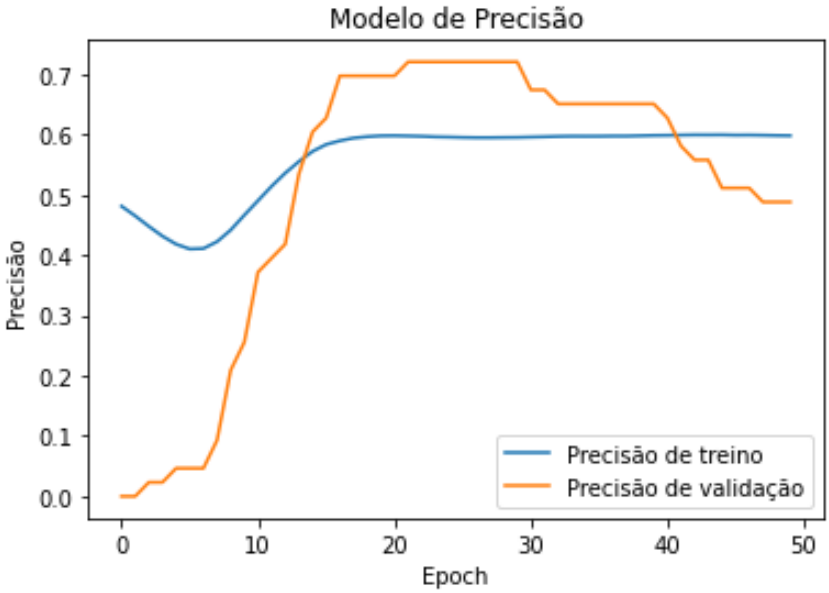

Figura 6.26: Função de precisão do modelo número 54.

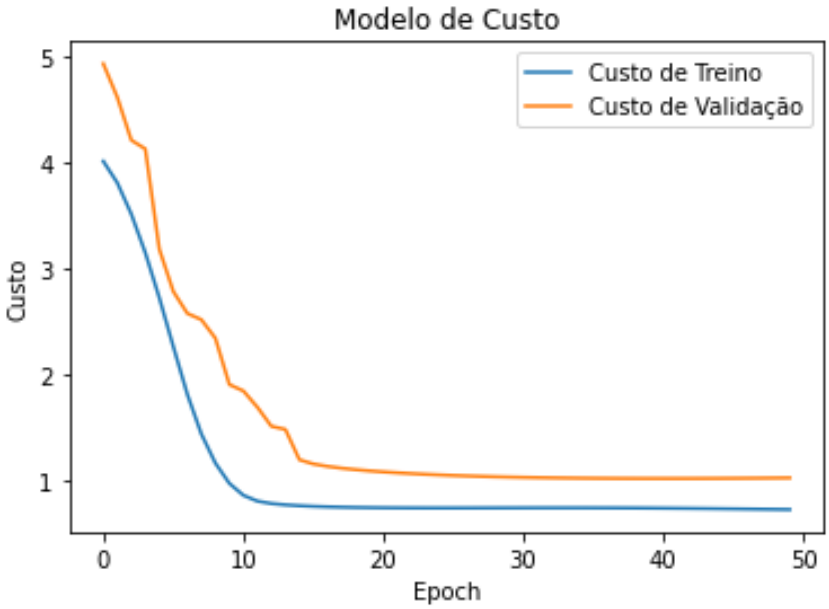

Figura 6.27: Função de custo do modelo número 54.

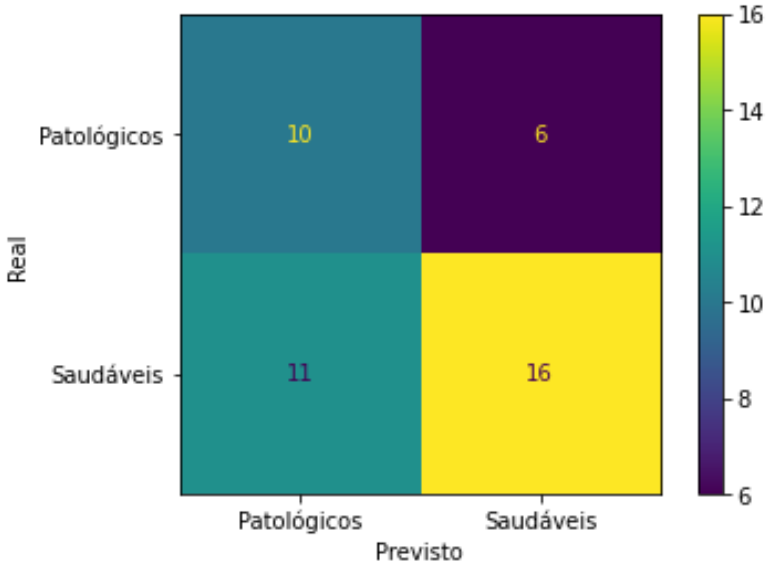

Figura 6.28: Matriz de resultados do modelo número 54.

Um teste importante para compreender e determinar a eficácia do modelo desenvolvido passou por recorrer apenas à base de dados aleatória, criada com base em parâmetros reais, para proceder ao treino e ao teste do modelo. As Figuras 6.29, 6.30 e 6.31 correspondem, respetivamente, à representação da função de precisão, à função de custo e à matriz de resultados do modelo tendo-se observado uma precisão de 100%. Este resultado ilustra bem algumas das dificuldades e limitações sentidas com a base de dados do MIT.

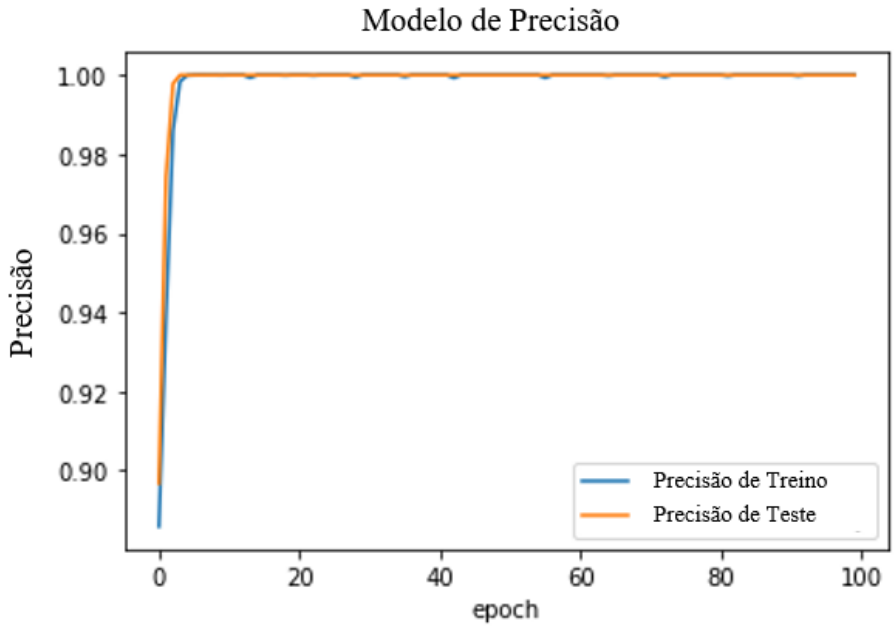

Figura 6.29: Função de precisão do modelo obtido com a base de dados aleatória.

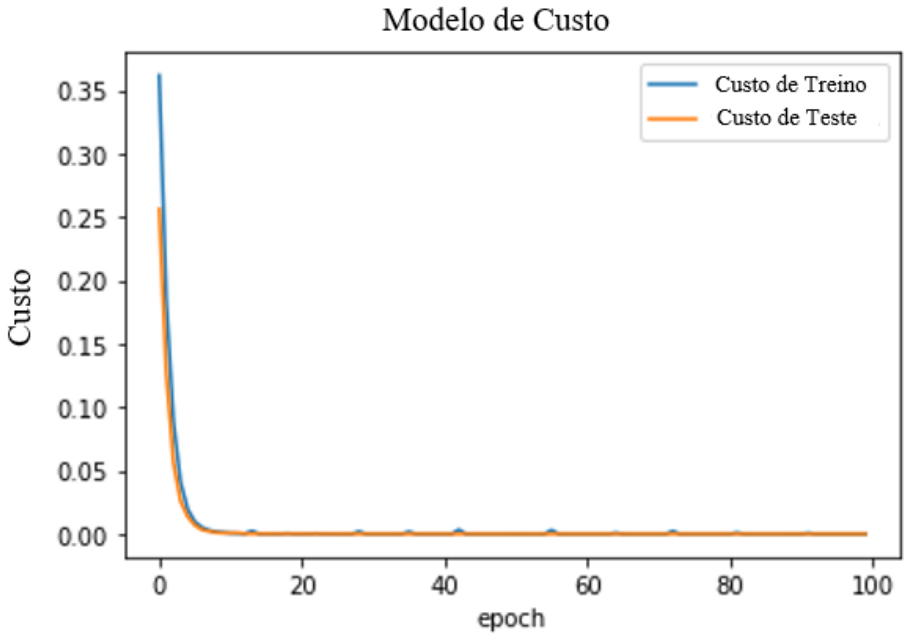

Figura 6.30: Função de custo do modelo obtido com a base de dados aleatória.

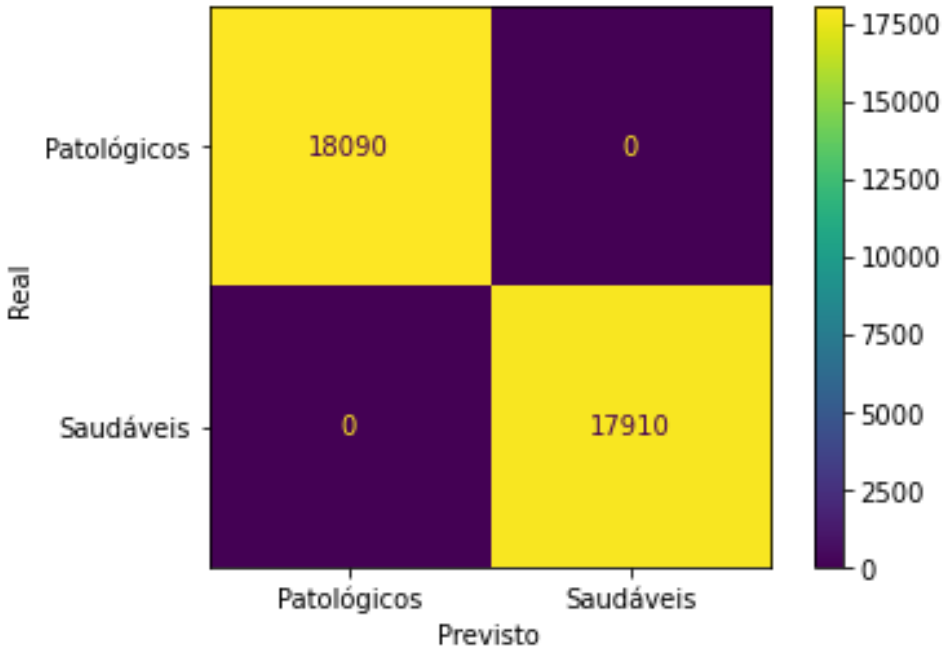

Figura 6.31: Matriz de resultado do modelo obtido com a base de dados aleatória.

O último teste realizado passou pela leitura do modelo obtido, anteriormente, com a base de dados aleatória onde se alcançou uma precisão de 100% para proceder à análise e classificação da amostra completa da base de dados do MIT. Para tal foi necessário desenvolver um novo algoritmo onde se procedeu à leitura da amostra, linha 9, à identificação das variáveis de entrada e de saída, linhas 11 e 12, à leitura do modelo salvo para a realização do teste, linha 14 e, por fim, procede-se à avaliação da rede neuronal, tendo sido determinada a precisão de teste em 60.47%.

```python
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
from keras.utils import np_utils
import numpy as np
from keras.models import load_model
from numpy import loadtxt

dataset = loadtxt('./Valores.csv', delimiter=',')

x = dataset[:,0:4]
y = dataset[:,4]

modelo = load_model("teste60.h5")

y_convertido = np_utils.to_categorical(y)

predicoes = modelo.predict(x)

np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})
print(predicoes)

_, acc_teste = modelo.evaluate(x, y_convertido)
print('Precisão teste: %.2f ' % (acc_teste*100))
```

Figura 6.32: Código *Pyhton* completo para teste e classificação da amostra.

Verificou-se que, mesmo reduzindo a dimensão da base de dados aleatória utilizada para treino, dos 60.000 saudáveis e 60.000 patológicos para metade, perfazendo um total de 60.000 dados, o resultado da precisão de teste após a avaliação da rede neuronal permanece o mesmo, ou seja, a precisão de teste é de 60.47%. Apesar deste resultado da precisão de teste ter-se mantido inalterado, é possível observar uma alteração nas funções de precisão e de custo, Figuras 6.33 e 6.34, respetivamente. Recorrendo à matriz de resultados, Figura 6.35, é possível perceber que houve um sinal que foi previsto incorretamente como sendo saudável, mas na verdade corresponde a um sinal patológico.

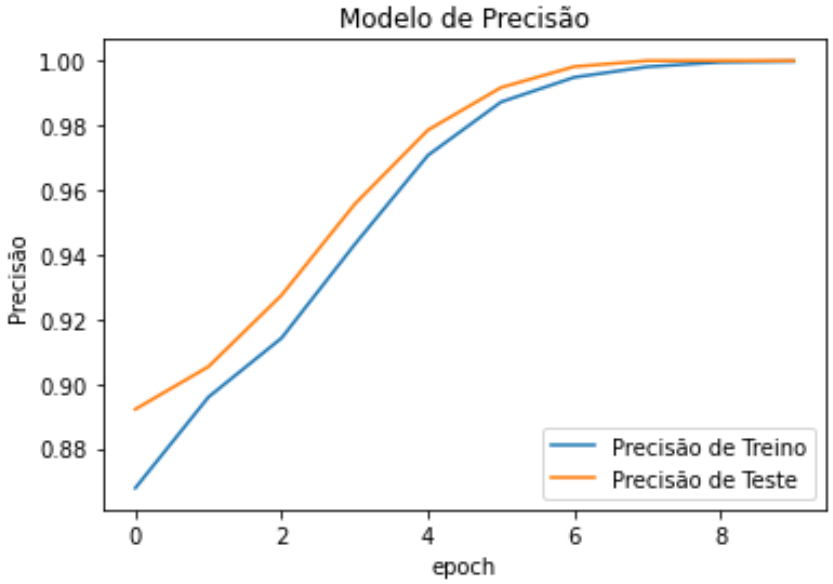

Figura 6.33: Função de precisão do modelo obtido com uma base de dados menor.

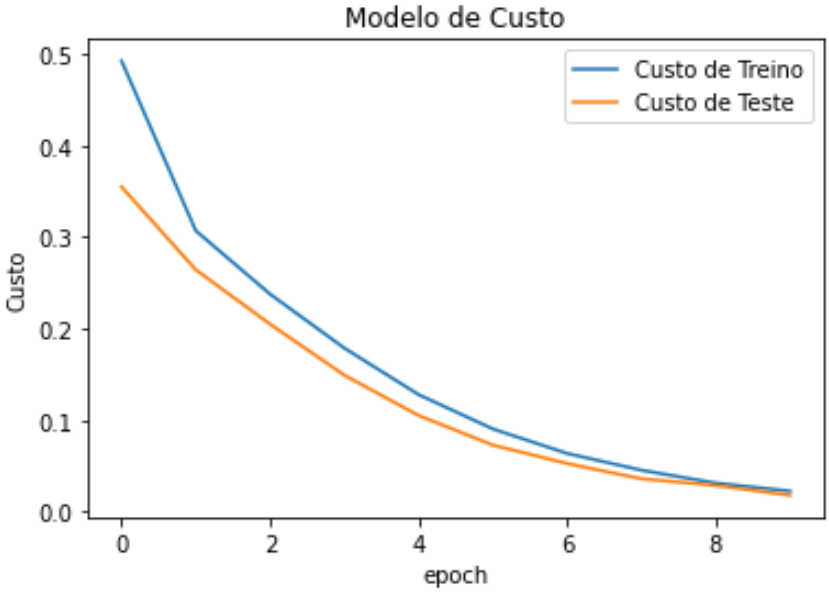

Figura 6.34: Função de custo do modelo obtido com uma base de dados menor.

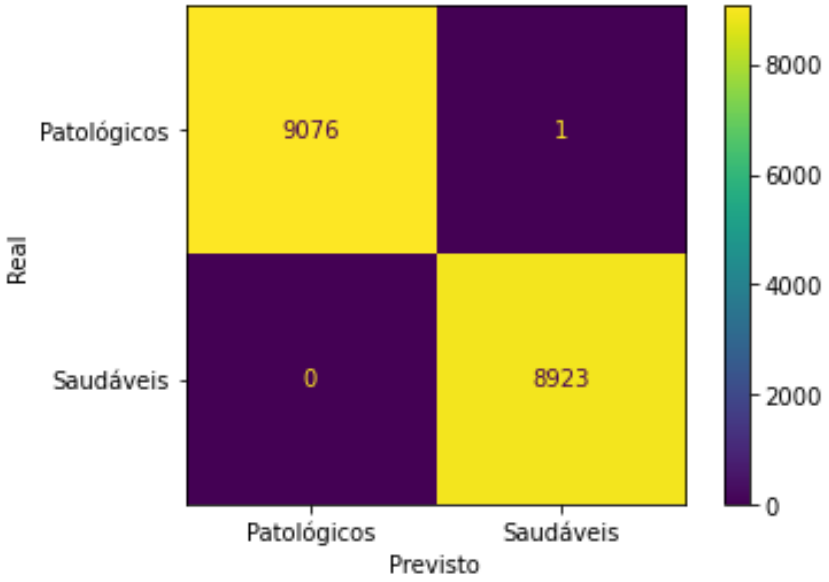

Figura 6.35: Matriz de resultado do modelo obtido com uma base de dados menor.

7. Conclusão

O ECG é o melhor método de avaliação da função cardíaca, sendo essencial para a deteção de diversas anomalias cardíacas, onde um modelo de DL poderá ser um forte aliado na identificação e classificação das anomalias. Assim, o objetivo deste trabalho foi desenvolver um modelo de DL que auxilie o individuo comum, sem conhecimentos na área, a perceber se há a possibilidade de ter algum problema cardíaco que necessite de ser observado, aconselhando-o a agendar uma consulta com um médico cardiologista. Este auxilio é prestado sob a forma de classificação binária de um sinal de ECG como sendo ”normal”ou ”patológico”, sem distinguir o tipo de patologia associada.

Uma vez que o processo está divido em duas fases, a primeira é a identificação e extração dos parâmetros pretendidos, baseada na sua caraterização geométrica, de cada um dos sinais e o segundo no processamento desses mesmos dados para proceder à classificação do sinal. Desta forma, a precisão dos resultados depende da precisão das duas etapas, basta uma das etapas estar comprometida que os resultados obtidos não serão favoráveis.

Neste trabalho a utilização de *wavelets* é fundamental para suavização do sinal e reduzir o ruído associado, permitindo a identificação dos parâmetros. A escolha da *wavelet* pode determinar o intervalo do valor dos parâmetros, sendo esta fase extremamente importante na construção da base de dados para, posteriormente, conseguir obter o conjunto de dados de treino para a rede neuronal artificial, com um nível de eficácia de classificação de patologias o mais alta possível.

Apesar de alguns parâmetros extraídos na primeira etapa estarem fora dos valores indicados no segundo capítulo, na literatura, a informação disponibilizada na base de dados do MIT foi considerada válida e, posteriormente, foi verificado que o modelo de DL desenvolvido é preciso na identificação e classificação de patologias cardíacas, incluindo alguns sinais mais complicados. Visto que o padrão do sinal de ECG varia de indivíduo para indivíduo, dependendo de alguns fatores mencionados previamente, o nível de precisão alcançado com o modelo transmite confiança no processo desenvolvido.

Uma das dificuldades sentidas foi a base de dados, o ideal seria ter uma base de dados com um número de amostras superior para melhorar a precisão do modelo desenvolvido. Outra dificuldade sentida foi devido aos 4 sinais da base de dados que foram removidos, estes eram incompatíveis com o algoritmo desenvolvido, não sendo possível proceder à sua classificação com um nível de confiança alto.

As perspetivas futuras para este trabalho passam pela ampliação da base de dados, aumentando a amostra aumenta-se o nível de confiança e eficiência do modelo, o aperfeiçoamento do algoritmo de recolha dos parâmetros, de modo a ser possível a identificação e extração de parâmetros dos 4 sinais removidos, e o ajuste do modelo de DL de modo a que seja capaz de classificar os sinais com patologia e fazer a distinção do tipo de patologia presente no sinal, no entanto, é importante salientar que estas alterações exigem uma carga horária despendida superior.

Referências Bibliográficas

[1] R. Alharbey, S. Alsubhi, K. Daqrouq, and A. Alkhateeb. The continuous wavelet transform using for natural ecg signal arrhythmias detection by statistical parameters. *Alexandria Engineering Journal*, 61(12), 2022.

[2] D. E. Becker. Fundamentals of electrocardiography interpretation. *Anesthesia progress*, 53(2), 2006.

[3] N. Bhatnagar. *Introduction to wavelet transforms*. CRC Press, 2020.

[4] O. Calin. *Deep Learning Architectures: A Mathematical Approach*. Springer Publishing Company, Incorporated, 1st edition, 2020.

[5] R. Campos. *Modelagem eletromecânica do coração com autômato celular e sistemas massa-mola*. PhD thesis, Universidade Federal de Juiz de Fora, 2016.

[6] J. E. Castilho, M. O. Domingues, A. Pagamisse, and O. Mendes. Introdução ao mundo das wavelets. *Sociedade Brasileira de matemática Aplicada e Computacional*, 2012.

[7] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE transactions on information theory*, 36(5):961–1005, 1990.

[8] I. Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[9] S. B. dos Santos. A doença cardiovascular é 80% prevenível ou tratável, desde que seja identificada precocemente. https://visao.pt/visaosaude/2022-09-15-a-doenca-cardiovascular-e-80-prevenivel-ou-tratavel-desde-que-seja-identificada-precocemente/, 2022.

[10] M. Fundamentals. A guide for using the wavelet transform in machine learning. https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/, 2018.

[11] B. Godinho and J. A. Rodrigues. Deep learning ecg classification with discrete wavelet contributions. *International Forum of Engineers & Practitioners*, 2023.

[12] A. Goldberger, Z. Goldberger, A. Shvilkin, et al. Bradycardias and tachycardias: review and differential diagnosis. *Goldberger's clinical electrocardiography. 9th ed. Amsterdam: Elsevier*, 2018.

[13] H. Gothwal, S. Kedawat, R. Kumar, et al. Cardiac arrhythmias detection in an ecg beat signal using fast fourier transform and artificial neural network. *Journal of Biomedical Science and Engineering*, 4(04), 2011.

[14] H. Hamaguchi. *Estudo de sinais de ECG utilizando métodos matemáticos para análise de sistemas dinâmicos não lineares*. PhD thesis, Universidade de São Paulo, 2006.

[15] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786), 2006.

[16] A. H. Kashou, H. Basit, and L. Chhabra. *Electrical Right and Left Axis Deviation*. StatPearls Publishing, 2017.

[17] L. Kuhn and L. Rose. Ecg interpretation part 1: Understanding mean electrical axis. *Journal of Emergency Nursing*, 34(6), 2008.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[19] P.-Y. Lin. An introduction to wavelet transform. *Graduate Institute of Communication Engineering National Taiwan University, Taipei, Taiwan, ROC*, 2007.

[20] S. Mallat et al. A wavelet tour of signal processing: the sparce way. *AP Professional, Third Edition, London*, 2009.

[21] MatLab. What is deep learning: 3 things you need to know. https://www.mathworks.com/discovery/deep-learning.html.

[22] S. Meek and F. Morris. Introduction. i—leads, rate, rhythm and cardiac axis. *Bmj*, 324(7334), 2002.

[23] G. Moody and R. Mark. Mit-bih arrhythmia database, 2005.

[24] A. C. Müller and S. Guido. *Introduction to machine learning with Python: a guide for data scientists*. "O'Reilly Media, Inc.", 2016.

[25] A. M. Patel, P. K. Gakare, and A. Cheeran. Real time ecg feature extraction and arrhythmia detection on a mobile platform. *Int. J. Comput. Appl*, 44(23), 2012.

[26] J. M. N. G. Pires. Aprendizagem profunda: estudo e aplicações. Master's thesis, Universidade de Évora, 2017.

[27] H. J. L. Reis, H. P. Guimarães, A. D. Zazula, R. G. Vasque, and R. D. Lopes. *ECG manual prático de eletrocardiograma*. Atheneu, 2013.

[28] A. Saxton, M. A. Tariq, and B. Bordoni. Anatomy, thorax, cardiac muscle. In *StatPearls [Internet]*. StatPearls Publishing, 2021.

[29] D. C. Sigg, P. A. Iaizzo, Y.-F. Xiao, and B. He. *Cardiac electrophysiology methods and models*. Springer Science & Business Media, 2010.

[30] S. Somani, A. J. Russak, F. Richter, S. Zhao, A. Vaid, F. Chaudhry, J. K. De Freitas, N. Naik, R. Miotto, G. N. Nadkarni, et al. Deep learning and the electrocardiogram: review of the current state-of-the-art. *EP Europace*, 23(8), 2021.

[31] M. M. Suhail and T. A. Razak. Cardiac disease detection from ecg signal using discrete wavelet transform with machine learning method. *Diabetes Research and Clinical Practice*, 187, 2022.

A. Apêndice 1

```
3   def max_2(n):
4       a = np.zeros(np.size(n, 0)-1)
5
6       a = [ int( n[k+1]-n[k] )  for  k in range(np.size(n, 0)-1)]
7
8       return np.amax(a)
9
10  import wfdb
11  from wfdb import processing
12  import matplotlib.pyplot as plt
13  import numpy as np
14  from scipy.signal import find_peaks, peak_widths
15  from skimage.restoration import (denoise_wavelet, estimate_sigma)
16
17  waveletname = 'sym5'
18
19  from IPython import get_ipython
20  get_ipython().magic('reset -sf')
21
22  file = '212'
23  f = open("NormPat.dat", "a")
24
25  amostra = 3000
26  record = wfdb.rdrecord(file, sampto=amostra)
27
28  ecg = record.p_signal[:, 0]
29
30  frequency = record.fs*1.
31
32  ann = wfdb.rdann(file, 'atr', sampto=amostra)
33
34  wfdb.plot_wfdb(record=record, annotation=ann, plot_sym=True,
35                 time_units='seconds', title=file,
36                 figsize=(10,4), ecg_grids='all')
37
38  time_data = np.arange(ecg.size) / frequency
39
40  y= denoise_wavelet(ecg, sigma=2, wavelet=waveletname, wavelet_levels=3)
```

Figura A.1: Código *Pyhton* completo para etapa de recolha de dados (1/3).

```
41
42  pico_R, _ = find_peaks(y, height= np.mean(y), distance=200 )
43
44  RR_max = 1000.*max_2(pico_R)/frequency
45
46  BPM = pico_R.shape[0]*60/(amostra/frequency)
47
48  pico_P = np.zeros(np.size(pico_R)-2,dtype=int)
49  Q      = np.zeros(np.size(pico_R)-2,dtype=int)
50  S      = np.zeros(np.size(pico_R)-2,dtype=int)
51
52  v_P1 = 30
53  v_P2 = 60
54
55  QRS = 0
56  PR  = 0
57
58  for  i in range(1,np.size(pico_R)-1):
59      p_w       = np.argmin(y[pico_R[i]-v_P1 : pico_R[i]])
60  #
61      Q[i-1]      = pico_R[i]- v_P1 + p_w
62  #
63      pico_P[i-1] = pico_R[i]  - v_P2 \
64                  + np.argmax( y[pico_R[i]-v_P2 : pico_R[i]- v_P1 + p_w] )
65      PR = PR + pico_R[i] -pico_P[i-1]
66  #
67      s_w       = np.argmin(y[pico_R[i] : pico_R[i]+v_P1])
68      S[i-1]      = pico_R[i] + s_w
69      QRS = QRS + S[i-1] - Q[i-1]
70
71  PR = 1000.*(PR/np.size(pico_P))/frequency
72  QRS = 1000.*(QRS/np.size(Q))/frequency
73
74  cpico_P = peak_widths(y, pico_P, rel_height=.4)
75  OP_max = 1000*np.mean(cpico_P[0])/frequency
76
77  fig, ax = plt.subplots(nrows=4, ncols=1, figsize=(15,25))
78  ax[0].set_title("Sinal")
79  ax[0].plot(ecg)
```

Figura A.2: Código *Pyhton* completo para etapa de recolha de dados (2/3).

```
80
81  ax[1].set_title("Sinal reconstruido sem ruido - picos R")
82  ax[1].plot(y)
83  ax[1].plot(pico_R, y[pico_R], "x")
84
85  ax[2].set_title("picos P")
86  ax[2].plot(y)
87  ax[2].plot(pico_P, y[pico_P], "x")
88  ax[2].hlines(*cpico_P[1:], color="C3")
89
90  ax[3].set_title("QRS")
91  ax[3].plot(y)
92  ax[3].plot(Q,y[Q], "x")
93  ax[3].plot(S,y[S], "x", color="green")
94
95  plt.show()
96
97
98  rr = wfdb.processing.ann2rr(file, 'atr', pn_dir=None, start_time=None,
99                              stop_time=None, format=None, as_array=True)
100
101 hr = wfdb.processing.calc_mean_hr(rr, fs=1, min_rr=None, max_rr=None, rr_units='samples')
102
103
104 print('\n ECG %s, RR :  %.1f , frequência calculada (bpm)): %.1f,\
105       frequência padrão: %.1f, OP (ms): %.2f , PR : %.1f , QRS (ms): %.2f\n' % \
106       (file, RR_max, BPM, hr*frequency , OP_max, PR, QRS ))
107
108 f.write('\n%s  %.1f  %.1f, %.1f, %.1f, %.1f' %
109         (file, RR_max, BPM , OP_max,PR,  QRS ))
110
111 f.close()
```

Figura A.3: Código *Pyhton* completo para etapa de recolha de dados (3/3).

B. Apêndice 2

Tabela B.1: Tabela de resultados obtidos na etapa de recolha de dados.

ID	Niv	pico_R	v_P1	v_P2	cpico_P	bpm	bpmPa	OP	PR	QRS
1	4	180	30	70	0.3	79.2	75.6	106.7	167.0	109.0
2	4	150	30	60	0.43	72.0	62.3	85.1	142.0	128.2
3	3	180	50	200	0.45	72.0	72.9	145.7	494.4	129.2
4	3	250	23	70	0.75	64.8	69.5	92.6	167.5	75.8
5	2	180	50	170	0.45	79.2	76.8	197.3	385.8	154.9
6	4	180	30	90	0.45	79.2	89.5	94.3	156.8	136.4
7	3	300	30	60	0.65	57.6	69.7	78.8	114.8	75.0
8	2	230	40	120	0.8	64.8	71.1	65.6	175.8	161.9
10	3	180	30	60	0.95	93.6	84.2	47.5	91.2	122.7
11	4	180	35	100	0.60	72.0	70.9	82.1	239.6	168.1
12	3	180	30	60	0.4	86.4	84.8	78.9	147.8	79.7
13	3	180	30	100	0.6	57.6	59.7	107.2	131.5	72.2
14	3	300	22	70	0.75	57.6	65.2	88.7	175.9	68.5
15	3	180	30	60	0.8	86.4	80.5	92.0	132.5	83.1
16	2	250	30	60	0.80	50.4	51.1	107.2	151.1	95.6
17	4	220	30	80	0.33	72.0	76.5	104.3	181.2	112.9
18	4	220	30	100	0.6	57.6	62.4	106.6	109.3	138.0
19	4	220	30	70	0.42	93.6	82.4	1134.0	141.7	126.0
20	4	220	35	80	0.8	50.4	50.5	100.7	188.9	126.1
21	4	280	35	100	0.3	50.4	54.3	117.6	112.8	138.9
22	4	180	40	100	0.9	86.4	67.8	75.3	144.7	130.0
23	4	180	40	100	0.6	57.6	71.3	95.7	121.8	146.8
24	4	120	30	120	0.5	108.0	103.3	69.4	160.9	141.9
25	3	180	30	60	0.7	86.4	88.8	70.8	140.6	73.9
26	4	600	30	60	0.3	36.0	79.3	62.7	96.3	126.9
27	3	120	70	130	0.3	100.8	101.0	46.2	220.1	256.3
28	3	180	30	100	0.75	93.6	101.4	118.1	170.2	60.9
29	4	130	40	140	0.3	100.8	89.2	63.2	221.1	161.6
30	3	180	30	60	0.85	86.4	91.8	85.2	133.3	83.9
31	3	150	30	80	0.85	108.0	109.5	66.8	144.7	73.3
32	3	150	90	140	0.8	79.2	76.3	128.8	192.9	321.3

Tabela B.2: Continuação da tabela de resultados obtidos na etapa de recolha de dados.

ID	Niv	pico_R	v_P1	v_P2	cpico_P	bpm	bpmPa	OP	PR	QRS
33	2	150	30	90	0.8	115.2	113.0	69.01	174.6	61.1
34	2	200	30	200	0.45	72.0	75.8	148.0	499.0	146.2
35	4	210	40	90	0.25	79.2	76.9	47.6	111.4	138.0
36	3	280	30	60	0.9	72.0	68.8	94.8	102.8	65.3
37	3	160	70	110	0.5	79.2	81.8	69.2	156.8	184.3
38	3	200	30	100	0.55	79.2	87.5	62.2	119.8	63.0
39	4	200	40	110	0.4	79.2	87.8	104.7	147.5	133.6
40	3	200	40	110	0.8	72.0	71.2	84.3	249.7	76.0
41	3	180	30	90	0.7	86.4	82.0	74.3	157.8	98.3
42	3	250	30	90	0.6	57.6	66.8	95.8	155.1	73.2
43	2	150	40	60	0.6	100.8	104.8	52.1	134.0	108.3
44	3	200	30	60	0.4	93.6	91.9	73.2	132.1	83.08

C. Apêndice 3

```
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
import keras
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
import numpy as np
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense, Activation
from keras.optimizers import SGD
from numpy import loadtxt
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import metrics

dataset = loadtxt('./treino.dat', delimiter=',')

x = dataset[:,0:4]
y = dataset[:,4]

y_convertido = np_utils.to_categorical(y)
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y_convertido, test_size = 0.30)

modelo = Sequential()
modelo.add(Dense(5, input_dim=x.shape[1], kernel_initializer='normal', activation='sigmoid'))
modelo.add(Dense(2, kernel_initializer='normal',  activation='sigmoid'))

modelo.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

resultado = modelo.fit(x_treino, y_treino, epochs=1000, batch_size=150,
                       validation_data=(x_teste, y_teste))

previsoes = modelo.predict(x_teste)

np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})
print(previsoes)

_, acc_treino = modelo.evaluate(x_treino, y_treino)
print('Precisão treino: %.2f ' % (acc_treino*100))
```

Figura C.1: Código *Pyhton* completo para etapa do processamento de dados (1/2).

```
43  _, acc_teste = modelo.evaluate(x_teste, y_teste)
44  print('Precisão teste: %.2f ' % (acc_teste*100))
45
46  print(resultado.history.keys())
47  plt.plot(resultado.history['accuracy'])
48  plt.plot(resultado.history['val_accuracy'])
49  plt.title('Accuracy model')
50  plt.ylabel('accuracy')
51  plt.xlabel('epoch')
52  plt.legend(["Training accuracy", "Validation accuracy"], loc='lower right')
53  plt.show()
54
55  plt.plot(resultado.history['loss'])
56  plt.plot(resultado.history['val_loss'])
57  plt.title('Modelo loss')
58  plt.ylabel('loss')
59  plt.xlabel('epoch')
60  plt.legend(["Training loss", "validation loss"], loc='upper right')
61  plt.show()
62
63
64  y_prediction = modelo.predict(x_teste)
65  y_prediction = np.argmax(y_prediction, axis= 1 )
66
67  y_test = np.argmax(y_teste, axis = 1)
68
69  cm = confusion_matrix(y_test, y_prediction )
70  print('Matriz de confusão')
71  print(cm)
72  disp = ConfusionMatrixDisplay(confusion_matrix=cm,
73                                display_labels=['Patológicos','Saudáveis'])
74
75  disp.plot()
76  disp.ax_.set(xlabel='Previsto', ylabel='Real')
77  plt.show()
78
79  modelo.save("teste50.h5")
80  print("modelo salvo")
```

Figura C.2: Código *Pyhton* completo para etapa do processamento de dados (2/2).

D. Apêndice 4

Tabela D.1: Tabela dos resultados obtidos na etapa de processamento de dados, com indicação dos parâmetros testados, precisão de treino e precisão de teste.

ID	Test_size	Ativador1	Ativador2	Otimizador	Neurónios	Prec. Treino	Prec. Teste
1	0.3	sigmoid	sigmoid	adam	6	86	61
2	0.3	sigmoid	sigmoid	adam	5	80	69.23
3	0.3	sigmoid	sigmoid	adam	4	80	69.23
4	0.2	sigmoid	sigmoid	adam	6	70.59	66.67
5	0.2	sigmoid	sigmoid	adam	5	79.41	77.78
6	0.2	sigmoid	sigmoid	adam	4	70.59	66.67
7	0.2	tanh	exponetial	adam	6	79.41	77.78
8	0.2	tanh	exponetial	adam	5	58.82	77.78
9	0.2	tanh	exponetial	adam	4	64.71	88.89
10	0.3	tanh	sigmoid	adam	5	80.0	69.23
11	0.3	tanh	softmax	adam	5	63.3	61.54
12	0.3	tanh	softplus	adam	5	76.67	69.23
13	0.3	tanh	softsign	adam	5	56.67	76.92
14	0.3	tanh	tanh	adam	5	66.67	92.31
15	0.3	tanh	selu	adam	5	60	76.92
16	0.3	tanh	elu	adam	5	60	69.23
17	0.3	tanh	relu	adam	5	63.33	61.54
18	0.3	softmax	sigmoid	adam	5	57.67	76.92
19	0.3	softmax	softmax	adam	5	66.67	76.92
20	0.3	softmax	softplus	adam	5	73.33	69.23
21	0.3	softmax	softsign	adam	5	57.67	76.92
22	0.3	softmax	selu	adam	5	63.33	61.54
23	0.3	softmax	elu	adam	5	63.3	61.54
24	0.3	softmax	relu	adam	5	63.33	61.54
25	0.3	sigmoid	sigmoid	adam	4	83.33	76.92
26	0.3	sigmoid	sigmoid	adam	5	70	84.62
27	0.3	sigmoid	sigmoid	adam	6	66.67	84.62
28	0.3	relu	relu	adam	4	56.67	84.62
29	0.3	relu	relu	adam	5	56.67	76.92
30	0.3	relu	relu	adam	6	66.67	69.23

Tabela D.2: Continuação da tabela dos resultados obtidos na etapa de processamento de dados, com indicação dos parâmetros testados, precisão de treino e precisão de teste.

ID	Test_size	Ativador1	Ativador2	Otimizador	Neurónios	Prec. Treino	Prec. Teste
31	0.3	tanh	tanh	adam	4	66.67	61.54
32	0.3	tanh	tanh	adam	5	60	69.23
33	0.3	tanh	tanh	adam	6	73.33	76.92
34	0.3	sigmoid	sigmoid	sgd	4	63.33	84.62
35	0.3	sigmoid	sigmoid	sgd	5	63.33	46.15
36	0.3	sigmoid	sigmoid	sgd	6	73.33	30.77
37	0.3	relu	relu	sgd	4	56.67	76.92
38	0.3	relu	relu	sgd	5	43.33	53.85
39	0.3	relu	relu	sgd	6	40	46.15
40	0.3	tanh	tanh	sgd	4	53.33	46.15
41	0.3	tanh	tanh	sgd	5	66.67	61.64
42	0.3	tanh	tanh	sgd	6	66.67	53.87
43	0.3	tanh	sigmoid	adam	4	66.67	76.92
44	0.3	tanh	sigmoid	sgd	4	70	69.23
45	0.3	sigmoid	tanh	adam	4	60	53.85
46	0.3	sigmoid	tanh	sgd	4	66.67	53.85
47	0.3	tanh	relu	adam	4	63.33	53.85
50	0.3	sigmoid	softmax	adam	5	57.14	68.18

E. Apêndice 5

```
from random import seed
from random import random

f = open("treino.dat", "a")

a = [60, 80, 120, 50]
b = [100, 110, 200, 110]

pa = [30, 110, 80, 140]
pb = [100, 200, 200, 200]

t1 = 0 # saudavel
t2 = 1 # patologico

N = 60000
seed(1)

for _ in range(N): # saudaveis
   s0 = a[0] + (random() * (b[0] - a[0]))
   s1 = a[1] + (random() * (b[1] - a[1]))
   s2 = a[2] + (random() * (b[2] - a[2]))
   s3 = a[3] + (random() * (b[3] - a[3]))
   f.write('\n%.1f,%.1f,%.1f,%.1f,%.0f' %
           (s0,s1,s2,s3,t1))

for _ in range(N): #  patologicos
   s0 = pa[0] + (random() * (pb[0] - pa[0]))
   if s0 > a[0]:
       s0 = s0 + 41
   s1 = pa[1] + (random() * (pb[1] - pa[1]))
   s2 = pa[2] + (random() * (pb[2] - pa[2]))
   if s2 > a[2]:
       s2 = s2 + 81
   s3 = pa[3] + (random() * (pb[3] - pa[3]))
   f.write('\n%.1f,%.1f,%.1f,%.1f,%.0f' %
           (s0,s1,s2,s3,t2))

f.close()
```

Figura E.1: Código *Pyhton* completo para o desenvolvimento da base de dados aleatória.

F. Apêndice 6

```
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
import keras
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
import numpy as np
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense, Activation
from keras.optimizers import SGD
from numpy import loadtxt
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import metrics

data_treino = loadtxt('./treino.dat', delimiter=',')

data_teste = loadtxt('./Valores.csv', delimiter=',')



x_tre = data_treino[:,0:4]
y_tre = data_treino[:,4]

x_tes = data_teste[:,0:4]
y_tes = data_teste[:,4]

y_trec = np_utils.to_categorical(y_tre)
y_tesc = np_utils.to_categorical(y_tes)

#x_treino, x_teste, y_treino, y_teste = train_test_split(x, y_convertido, test_size = 0.30)

modelo = Sequential()
modelo.add(Dense(4, input_dim=x_tre.shape[1], kernel_initializer='normal', activation='tanh'))
modelo.add(Dense(2, kernel_initializer='normal',  activation='tanh'))

modelo.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figura F.1: Código *Pyhton* completo para utilização da base de dados aleatória para treino e da base de dados do MIT para o teste (1/3).

```
resultado = modelo.fit(x_tre, y_trec, epochs=30, batch_size=120000,
                       validation_data=(x_tes, y_tes))

previsoes = modelo.predict(x_tes)

np.set_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})
print(previsoes)

_, acc_treino = modelo.evaluate(x_tre, y_trec)
print('Precisão treino: %.2f ' % (acc_treino*100))

_, acc_teste = modelo.evaluate(x_tes, y_tesc)
print('Precisão teste: %.2f ' % (acc_teste*100))

print(resultado.history.keys())
plt.plot(resultado.history['accuracy'])
plt.plot(resultado.history['val_accuracy'])
plt.title('Modelo de Precisão')
plt.ylabel('Precisão')
plt.xlabel('Epoch')
plt.legend(["Precisão de treino", "Precisão de validação"], loc='lower right')
plt.show()

plt.plot(resultado.history['loss'])
plt.plot(resultado.history['val_loss'])
plt.title('Modelo de Custo')
plt.ylabel('Custo')
plt.xlabel('Epoch')
plt.legend(["Custo de Treino", "Custo de Validação"], loc='upper right')
plt.show()


#Predict
y_prediction = modelo.predict(x_tes)
y_prediction = np.argmax(y_prediction, axis= 1 )
```

Figura F.2: Código *Pyhton* completo para utilização da base de dados aleatória para treino e da base de dados do MIT para o teste (2/3).

```
76
77  y_test = np.argmax(y_tesc, axis = 1)
78
79  #Create confusion matrix and normalizes it over predicted (columns)
80
81  cm = confusion_matrix(y_test, y_prediction )
82  #                     ,normalize='pred')
83
84  print('Matriz de confusão')
85  print(cm)
86
87  disp = ConfusionMatrixDisplay(confusion_matrix=cm,
88                                display_labels=['Patológicos','Saudáveis'])
89
90
91  disp.plot()
92  disp.ax_.set(xlabel='Previsto', ylabel='Real')
93  plt.show()
94
95
96  modelo.save("teste55.h5")
97  print("modelo salvo")
```

Figura F.3: Código *Pyhton* completo para utilização da base de dados aleatória para treino e da base de dados do MIT para o teste (3/3).