INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Deteção de arritmias cardíacas em eletrocardiogramas usando deep learning

GABRIEL MOREIRA DA ROCHA
Outubro de 2018

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Deteção de arritmias cardíacas em eletrocardiogramas usando *deep learning*

Gabriel Moreira da Rocha

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Elsa Ferreira Gomes

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro 2018

À minha família e amigos

Resumo

As arritmias cardíacas são perturbações do ritmo cardíaco comuns e podem causar sérios riscos na vida das pessoas, sendo hoje em dia umas das principais causas de morte na população em geral, essencialmente nos países desenvolvidos e em desenvolvimento. Muitas destas mortes poderiam ser evitadas se fosse realizada a deteção e a monitorização prévia destas arritmias a partir do Eletrocardiograma (ECG). O ECG é um exame fundamental no diagnóstico de doenças cardiovasculares e em várias patologias clínicas, registando a informação relativa ao funcionamento do coração através da sua atividade elétrica em cada batimento cardíaco.

Através da análise dos dados obtidos por ECG, pretende-se a identificação dos períodos com maior probabilidade de ocorrência de arritmia, possibilitando assim uma maior eficácia na deteção e previsão dos sistemas baseados em ECG.

Desta forma, os pacientes poderão melhorar bastante a sua qualidade de vida, garantindo uma maior rapidez na intervenção médica. Ademais, esta abordagem permitirá evitar os efeitos colaterais das arritmias e possivelmente reduzir a administração da medicação.

O objetivo deste trabalho passa pelo desenvolvimento de uma metodologia capaz de classificar sinais resultantes do ECG, para deteção de arritmias cardíacas.

São apresentadas nesta dissertação diversas técnicas utilizadas para o processamento e classificação dos sinais ECG, pretendendo-se que sejam aplicadas neste trabalho algumas destas técnicas.

Palavras-chave: *Deep Learning*, Neural *Networks*, Eletrocardiograma, Arritmia Cardíaca

Abstract

Heart arrhythmia, a group of conditions in which the heartbeat is irregular, is known nowadays as one of the main causes of death, targeting specially the population in developed and developing countries. Most of these fatalities could've been avoided by the previous detection and monitorization of this condition, through an Electrocardiography (ECG). The ECG is a fundamental exam on the diagnosis of heart conditions and several clinical pathologies, through the heart's electrical activity, it registers information related to its behavior.

Considering the data obtained from the ECG, it is intended the identification of periods in which there is more probability of arrhythmia occurrence, allowing a more efficient detection and prediction on ECG based systems.

Taking this into account, it is foreseen an improvement on patient's quality of life, assuring a quick medical intervention, avoiding the collateral effects of this condition, and possibly decreasing the dependence on medication.

The main purpose of this work is centered on the development of a methodology, capable of identifying ECG signals, for the detection of heart arrhythmia.

On this thesis, there will be presented several techniques for the processing and classification of ECG signals, some of which will be applied on this work.

Keywords: Deep Learning, Neural Networks, Electrocardiogram, Heart Arrhythmia

Agradecimentos

As minhas primeiras palavras de agradecimento vão para os meus pais, por todo o apoio e dedicação que sempre tiveram para comigo, pela excelente educação que me deram e por me terem proporcionado a oportunidade de estudar no ensino superior. Estendo este agradecimento também aos meus irmãos e à minha namorada, por todo o apoio e incentivo ao longo do meu percurso académico.

Quero também deixar um agradecimento especial para a minha orientadora, a Professora Elsa Ferreira Gomes, que me proporcionou a oportunidade de realizar este trabalho, que me acompanhou ao longo da realização do mesmo e que se mostrou sempre disponível para ajudar.

Por fim, quero deixar um grande agradecimento também aos meus amigos, colegas de curso e a todos os que de alguma forma me ajudaram, me apoiaram e confiaram em mim neste meu percurso académico.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

AF	Fibrilação Atrial
AHP	Processo de análise hierárquica
ANN	Rede Neuronal Artificial
ANOVA	Análise de Variância
AVG	Média
AUC	Área sob a Curva
CNN	Rede Neuronal Convolucional
CRNN	Rede Neuronal Recorrente Convolucional
DFA	*Detrended Fluctuation Analysis*
DNN	*Deep Neural Network*
DT	Árvore de Decisão
DWT	Transformada de *Wavelet* Diádica
ECG	Eletrocardiograma
EM	Maximização da Expectativa
FFT	Transformação Rápida de *Fourier*
FN	Falso Negativo
FP	Falso Positivo
IC	Índice de Consistência
ICA	Análise de Componente Independente
IR	Índice Aleatório
kNN	k-vizinhos mais próximos
LLE	*Largest Lyapunov Exponent*
LSTM	*Long Short-Term Memory*

MD-PSO	*Multi-Dimensional Particle Swarm Optimization*
ME	Mistura de Especialistas
MLP	Perceptrão Multi-camada
NCD	Novo modelo de Desenvolvimento de Conceito
NN	Rede Neuronal
OPF	*Optimum-Path Forest*
PCA	Análise de Componente Principal
ReLU	Unidade Linear Retificada
RC	Razão de Consistência
ROC	Curva de Operação do Recetor
SOM	Mapas Auto-Organizados
SVM	Máquina de Vetores de Suporte
Tanh	Tangente Hiperbólica
TFN	Taxa de Falsos Negativos
TFP	Taxa de Falsos Positivos
TVN	Taxa de Verdadeiros Negativos
TVP	Taxa de Verdadeiros Positivos
VM	Máquina Virtual
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
WT	Transformada de *Wavelet*

1 Introdução

Neste primeiro capítulo é apresentado, de uma forma resumida, o trabalho desenvolvido neste projeto. Inicialmente é feito um enquadramento do tema relacionado com o projeto, tal como a descrição do problema que motiva o seu desenvolvimento, e ainda os objetivos a atingir com o mesmo. Segue-se uma descrição dos resultados esperados, uma pequena secção onde será feita, de uma forma breve, a análise de valor do projeto e ainda a abordagem tomada para o desenvolvimento do mesmo. Por fim, é feita uma apresentação sucinta dos capítulos que fazem parte deste documento.

1.1 Contexto

A área da saúde é das mais importantes na sociedade, motivando uma investigação constante, de modo a que se alcancem novos tipos de tratamentos ou se aperfeiçoem os existentes.

Cada vez mais a tecnologia tem sido aplicada na área da saúde, refletindo-se desta forma em melhorias constantes na mesma. Uma das áreas da saúde que mais tem beneficiado com isto é a área da medicina diagnóstica, que tem alcançado melhorias frequentes tanto do ponto de vista dos pacientes como dos profissionais da área. Estas melhorias referem-se a uma maior simplificação, maior rapidez e segurança na elaboração dos exames, procedimentos e geração dos resultados (iClinic, 2014).

Para além disso, houve um grande aperfeiçoamento da precisão dos diagnósticos, reduzindo as possibilidades de erros e proporcionando de igual modo uma melhor seleção do tratamento a adotar numa determinada patologia.

O ponto principal deste trabalho assenta na análise de dados obtidos por eletrocardiogramas (ECG). Estes exames contribuem para o estudo das doenças cardíacas, na medida em que registam a informação do funcionamento cardíaco através da atividade elétrica de cada batimento.

Uma arritmia é uma perturbação do ritmo cardíaco. Geralmente não é possível perceber os batimentos cardíacos do nosso coração, no entanto, é possível sentir pequenas irregularidades de ritmo. Por norma, estes casos não são sinal de doença cardíaca, no entanto se a frequência das mesmas aumentar ou se aparecem acompanhadas de outros sintomas (tonturas, desmaios, cansaço, dor no peito, falta de ar), podem aí sim ser manifestações de arritmias mais graves (CUF, 2015) (OMS, 2017).

O Eletrocardiograma permite registar e medir o ritmo do nosso coração. Se for realizado durante o momento em que surge a arritmia, este pode ser suficiente para fazer o diagnóstico, no entanto, em inúmeras situações, as arritmias podem ocorrer de forma esporádica e episódica. Nestes casos, o ECG pode revelar-se insuficiente para fazer o diagnóstico, revelando-se, portanto, necessária a realização de uma série de exames, tais como: o Holter (ECG durante 24 horas), o ecocardiograma, a prova de esforço, análises laboratoriais, entre outros. (CUF2, 2018).

Neste trabalho, estes conceitos serão abordados diversas vezes, dada a sua importância para o tema em questão.

1.2 Problema

Segundo a Organização Mundial de saúde, as doenças cardíacas são hoje em dia uma das principais causas de morte nos países desenvolvidos. Diversos instrumentos e métodos são utilizados diariamente para analisar o comportamento do coração (Coração Saudável, 2018).

A Fibrilação Atrial (AF) é a arritmia cardíaca mais comum, ocorrendo em 1 a 2% da população em geral e está associada a acidente vascular cerebral, hospitalização, insuficiência cardíaca e doença arterial coronária. Estima-se que mais de 12 milhões de europeus e norte-americanos sofram de AF, e sua prevalência triplique nos próximos 30 a 50 anos (PhysioNet, 2017).

Por norma, os especialistas (geralmente cardiologistas) interpretam o ECG com bastante precisão, no entanto o erro humano é muito comum e variável. Para além disto, a análise efetuada pelo cardiologista pode ser bastante pessoal, podendo existir uma grande variação do diagnóstico entre diferentes especialistas. Desta forma, a deteção automática de arritmias cardíacas visa auxiliar o especialista num diagnóstico mais célere e preciso (Junior S., 2008).

Desta forma, através da análise dos dados obtidos por eletrocardiogramas, pretende-se detetar arritmias em humanos.

1.3 Objetivos

O objetivo deste trabalho passa pelo desenvolvimento de uma metodologia capaz de classificar sinais resultantes do ECG, para deteção de arritmias cardíacas em humanos, usando técnicas de *data mining*, em particular *deep learning*.

Existem diversas técnicas que são utilizadas para o processamento dos sinais ECG e para os classificar.

Pretende-se que seja realizado também um estudo acerca de técnicas utilizadas previamente, em trabalhos desenvolvidos anteriormente em dados de sinais cardíacos para deteção de patologia cardíaca, e perceber se existe a possibilidade de aplicar algumas das mesmas neste trabalho. Desta forma, será possível efetuar uma comparação entre os resultados obtidos e publicados com os resultados provenientes deste trabalho.

1.4 Resultados esperados

Na secção 1.3, foi já mencionado o objetivo deste projeto: o desenvolvimento de uma metodologia que classificasse sinais resultantes do ECG, para detetar arritmias cardíacas. Foi também referido que para tal, serão aplicadas técnicas com redes neurais em processamento e classificação de sinais por categorias.

Espera-se que o desenvolvimento deste projeto resulte num sistema eficaz e eficiente na identificação de ocorrências de arritmias cardíacas. Para proceder à avaliação destes resultados serão usadas algumas medidas de avaliação (métricas) tais como a taxa de acerto (*accuracy*), a sensibilidade, especificidade e AUC (*Area Under the Curve*), que serão descritos posteriormente nesta dissertação.

O *dataset* utilizado neste projeto, é o disponibilizado pelo concurso (*The PhysioNet in Cardiology Challenge 2017*) e como resultado, é esperado que obtenha melhores resultados quando comparados com os publicados pelos participantes do concurso, que usaram o mesmo conjunto de dados.

1.5 Análise de Valor

Uma vez que a arritmia cardíaca é uma doença que afeta cerca de a 1 a 2% da população em geral e é expectável que a sua prevalência aumente nos próximos anos, existe obviamente uma maior preocupação por parte da comunidade científica em estudar e identificar as origens da mesma (PhysioNet, 2017).

Posto isto, tendo este trabalho como propósito a contribuição para o desenvolvimento de uma metodologia de deteção de arritmias cardíacas a partir de eletrocardiogramas, é viável afirmar que o mesmo tem valor de negócio. Este sistema poderá ser usado pelos técnicos de saúde, possibilitando desta forma o apoio na decisão sobre diagnósticos de pacientes, e proporcionando assim uma melhoria na qualidade de vida dos mesmos.

Este tópico será abordado com maior detalhe na secção 2.2.

1.6 Abordagem

Para este trabalho, será implementada uma aplicação de *software* para deteção de arritmias cardíacas, através de sinais obtidos por ECG, usando técnicas de *Data Mining*.

Com este intuito, foi efetuado um estudo sobre alguns trabalhos desenvolvidos anteriormente em problemas similares, verificando-se que em grande parte deles é usada uma metodologia de deteção de padrões para a extração de atributos juntamente com um algoritmo de classificação, como por exemplo o *Random Forest* ou SVM. Desta forma, esta metodologia foi colocada na lista de possíveis abordagens a seguir no desenvolvimento do trabalho.

Para além desta metodologia, com este estudo, foi também considerada como possível abordagem a seguir, a aplicação de *deep learning*, mais concretamente Redes Neurais Convolucionais, que possui desde logo as camadas de extração de atributos integradas. Para a implementação das CNN a escolha seria o *TensorFlow*[1] como *framework* e a linguagem de programação Python[2]. Esta segunda abordagem acabou por ser a selecionada, como metodologia a seguir. A escolha desta abordagem em muito se deve ao facto de o *deep learning*, neste caso a CNN, ter desde logo o "componente" de extração de atributos integrado, contribuindo imenso para a redução do esforço humano nessa tarefa. Desta forma, um maior esforço pode ser aplicado em tarefas mais significativas, tal como a fase de pré-processamento do conjunto de dados, permitindo ainda, desta forma, a apresentação de um protótipo do projeto de forma mais célere. Para além disso, um médico especialista efetua o diagnóstico de um ECG através da visão sobre os sinais, ou seja, sobre uma imagem. Posto isto, conclui-se que a aplicação de CNN sobre as imagens ECG é o mais semelhante ao processo de diagnóstico do médico.

A escolha do *TensorFlow* e do Python é fundamentada na secção de estado da arte das tecnologias, secção 3.3.

1.7 Organização do Documento

Nesta secção serão apresentados brevemente os capítulos que compõem esta dissertação.

O primeiro capítulo é a introdução, onde é apresentado um breve enquadramento do projeto, a descrição e valorização do problema, evidenciando os objetivos traçados para o projeto, assim como os resultados esperados com o desenvolvimento do mesmo. É igualmente apresentada de forma sucinta, a análise de valor deste desenvolvimento, bem como a abordagem que se pretende seguir. Por fim, é exposta a estrutura do documento.

[1] https://www.tensorflow.org/
[2] https://www.python.org/

De seguida, é apresentado o capítulo de contexto, onde são descritos os conceitos essenciais à compreensão do problema e é indicado o contexto no qual este está inserido. É também retratada a análise de valor do projeto, de forma mais detalhada.

No terceiro capítulo serão apresentados os módulos de estado da arte. Inicialmente é apresentado um resumo do estudo efetuado ao *Machine Learning*, com maior ênfase sobre as metodologias usadas em problemas de classificação. De seguida, encontra-se o estado da arte em soluções existentes, onde será apresentado o estudo efetuado sobre soluções existentes para problemas similares e ainda soluções para o mesmo problema, usando o mesmo *dataset* (participantes do concurso). Por fim, o estado da arte em tecnologias, onde serão apresentadas e descritas as tecnologias e ferramentas que irão ser utilizadas neste desenvolvimento, tendo em conta o estudo comparativo efetuado.

O quarto capítulo apresenta o *design* da solução proposta, onde será demonstrada a abordagem seguida para alcançar a solução, a descrição detalhada da implementação do modelo assim como a apresentação dos dados (*dataset*) que serão utilizados na solução.

No quinto capítulo, o de Avaliação, são descritas as medidas e metodologias de avaliação utilizadas, assim como os resultados obtidos com a solução.

O sexto capítulo, refere-se ao capítulo das conclusões onde são apresentadas as considerações finais relativas ao trabalho realizado. São apresentados os objetivos atingidos, assim como as limitações da solução proposta e possíveis melhorias para o futuro. Por último, é feita uma apreciação final do autor.

Por fim, encontra-se a bibliografia, onde estão expostas todas as fontes que servem de apoio a esta dissertação.

2 Contexto

Neste capítulo é feita uma contextualização do tema do projeto, tendo em conta o problema a resolver. É realizado o enquadramento do projeto relativamente à área de negócio em que se encontra inserido, apresentando os principais conceitos, processos e intervenientes, importantes para a compreensão do trabalho desenvolvido.

Para além do enquadramento do problema, este capítulo é ainda composto pela análise de valor do projeto, que será descrita de forma detalhada.

2.1 Enquadramento do Problema

Este trabalho, surge a partir do desafio de um concurso lançado pela *PhysioNet* (*The PhysioNet in Cardiology Challenge 2017*). O desafio consistia no desenvolvimento de uma metodologia capaz de classificar, a partir de uma gravação curta de ECG (entre 30 a 60 segundos), se a gravação mostra um ritmo sinusal normal, AF, um ritmo alternativo (outro ritmo), ou se o sinal é ruidoso demais para ser classificado, detetando desta forma possíveis arritmias cardíacas em humanos.

A arritmia é o nome genérico de diversas perturbações que alteram a frequência e/ou o ritmo dos batimentos cardíacos, podendo ser caracterizada por ritmos excessivamente rápidos (taquicardia), lentos (bradicardia) ou apenas irregulares (CUF2, 2018).

Na Figura 1 estão representados dois registos de sinais cardíacos, um com o ritmo normal, e outro com a existência de arritmia cardíaca (fibrilação atrial). No registo referente à arritmia, é possível observar um ritmo irregular, desorganizado, o que mostra que os batimentos do coração são irregulares, batendo muitas vezes rápido demais, outras vezes com pouca força.

Figura 1 - Ritmo Normal vs Fibrilação Atrial (Jama Network, 2010)

Esta doença ocorre em cerca de 1 a 2% da população em geral, sendo que afeta essencialmente pessoas com mais de 60 anos de idade, dada a presença de doenças cardíacas e de outros problemas de saúde que se podem associar às arritmias. A Fibrilação Atrial é a arritmia mais frequente e afeta cerca de 1% das pessoas com menos de 55 anos (CUF2, 2018). Elas podem ocorrer por diversas razões, podendo mesmo levar à morte, constituindo, por isso, um caso de emergência médica. A maior parte delas é, no entanto, inofensiva.

O ECG é o exame que permite registar e medir o ritmo do coração, sendo que se este for realizado durante o momento em que surge a arritmia, pode chegar para fazer o diagnóstico. Em certas situações, as arritmias podem ocorrer de forma esporádica e episódica, levando a que, para fazer o diagnóstico e realmente determinar se os sintomas sentidos pelo paciente estão relacionados com arritmias, o médico cardiologista possa ter que pedir outros exames, tais como: o *Holter* (ECG durante 24 horas), o ecocardiograma, a prova de esforço, análises laboratoriais, entre outros (CUF3, 2018).

O ritmo do coração, em termos de batimentos por minuto, pode ser facilmente calculado contando os picos R da onda de ECG durante um minuto de gravação. Na Figura 2 está exemplificada uma única forma de onda de ECG, onde, a onda P, o complexo QRS e a onda T representam a contração/despolarização dos átrios, a contração/despolarização dos ventrículos e a repolarização dos ventrículos, respetivamente.

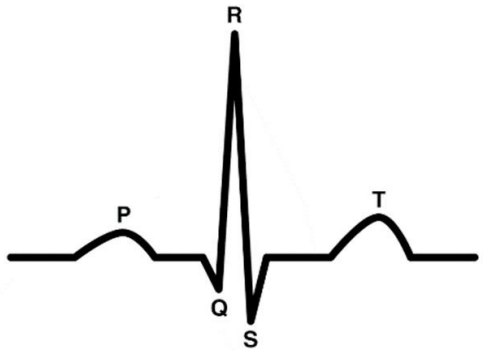

Figura 2 - Forma de onda de ECG normal/saudável (Isin et al., 2017)

Por norma, os cardiologistas interpretam o ECG com bastante precisão, no entanto o erro humano é plausível. Para além disso, a análise efetuada pelo especialista pode ser bastante pessoal, sendo possível a existência de uma grande variação nos diagnósticos entre vários especialistas. Desta forma, a deteção automática de arritmias cardíacas visa auxiliar o especialista num diagnóstico mais rápido e preciso (Junior S., 2008).

2.2 Análise de Valor

Nesta secção será apresentada de forma detalhada, a análise de valor do projeto, tendo em conta alguns pontos, tais como, a identificação dos 5 elementos chave do modelo "*the new concept development model*" (NCD), a definição dos conceitos "valor", "valor para o cliente" e "valor percetível" relativamente ao tema do projeto, a descrição da proposta de valor do produto e por fim a utilização do modelo de negócio Canvas para descrever a ideia de negócio.

2.2.1 Novo modelo de desenvolvimento de conceito (NCD)

De acordo com *Peter Koen* (Koen et al., 2002), o novo modelo de desenvolvimento de conceito NCD, define cinco pontos chave: a identificação e análise da oportunidade, a criação e seleção da ideia e a definição do conceito.

Serão descritos de seguida, cada um destes pontos chave relativamente ao tema do projeto.

2.2.1.1 Identificação da Oportunidade

As arritmias cardíacas são hoje em dia uma das principais causas de morte em países desenvolvidos e em desenvolvimento, sendo que muitos destas mortes poderiam ser evitadas através de uma monitorização atempada destas perturbações, a partir do ECG. Esta monitorização e a deteção automática de arritmias cardíacas podem auxiliar os especialistas, no que toca a elaborar os diagnósticos de forma mais rápida e precisa.

Com isto, é possível observar aqui duas formas de identificação de oportunidades, a identificação de necessidades e observação de deficiências (aperfeiçoamento de atividades existentes).

Este ponto chave pode ser analisado através de alguns métodos/técnicas, tais como (Koen et al., 2002):

- Exploração tecnológica;
- Técnica de definição de um *Roadmap* tecnológico;
- Análise de inteligência competitiva;
- Pesquisas de mercado;
- Planeamento de cenários.

2.2.1.2 Análise da Oportunidade

A arritmia cardíaca tem cura, mas deve ser tratada o mais depressa possível quando detetada, de modo a que o tratamento seja instituído o mais rapidamente possível, evitando complicações que a doença pode trazer como ataque cardíaco, acidente vascular cerebral ou até mesmo morte. Desta forma, é possível identificar aqui uma lacuna onde é possível melhorar este processo de deteção de arritmias, de modo a melhorar a qualidade de vida dos pacientes (contribuindo para uma redução de possíveis complicações), assim como apoiar no trabalho dos técnicos de saúde, ajudando-os na realização dos diagnósticos.

Recentemente, a inteligência artificial e o *deep learning* prometeram algoritmos que não só podem tomar decisões de saúde baseados em dados e livres do erro humano como também podem processar conjuntos de dados muito mais vastos do que qualquer ser-humano conseguiria. Sistemas de *deep learning* já se encontram no mercado, oferecendo assistência na interpretação de exames de imagem cardíacos (Brown, 2017).

Neste ponto, podem ser usadas várias das técnicas usadas na identificação das oportunidades, tais como o *Roadmapping*, análise de tendências tecnológicas, análise de inteligência competitiva, planeamento de cenários, entre outras. Na identificação das oportunidades, estas técnicas são usadas para determinar se existe uma oportunidade, já aqui, na análise de oportunidade, são gastos mais recursos, fornecendo mais detalhes relativamente à adequação da oportunidade selecionada (Koen et al., 2002).

2.2.1.3 Criação da ideia

Atualmente, existem diversos estudos relativos aos sistemas de deteção e previsão de arritmias cardíacas, que recorrem a outras metodologias para o processamento e análise dos dados, usando o reconhecimento de padrões.

A maioria das técnicas convencionais de reconhecimento destes padrões foram já aplicadas com sucesso às tarefas de deteção de arritmia do ECG anteriormente. No entanto, os desempenhos recentes obtidos por métodos usando *deep learning*, essencialmente em concursos de reconhecimento de padrões (Krizhevsky et al., 2012), incentivou uma pesquisa mais profunda e implementação destas técnicas no campo da imagem médica e processamento de sinal.

Para a análise deste ponto, podem ser usadas as seguintes técnicas (Koen et al., 2002):

- Identificação de novas soluções tecnológicas;
- Abordagens etnográficas;
- Uma cultura organizacional que encoraja os funcionários a gastar o tempo livre testando e validando as suas próprias ideias, assim como outras.
- Uma variedade de incentivos para estimular ideias;
- Rotação de emprego frequente para o incentivo à partilha de conhecimento;
- Inclusão de pessoas com distintos estilos cognitivos nas equipas de enriquecimento de ideias.

2.2.1.4 Seleção da ideia

Neste ponto chave, depois de efetuado o estudo para a geração e enriquecimento de ideias, a seleção da ideia que pode alcançar um maior valor, passa pelo desenvolvimento de uma solução, usando uma metodologia muito recente neste tipo de projetos, nomeadamente pelo uso do *deep learning*. As tecnologias selecionadas para o desenvolvimento do mesmo, estão descritas posteriormente neste documento.

A análise deste ponto chave pode ser feita pelos seguintes métodos (Koen et al., 2002):

- Metodologias de portefólios, baseados em múltiplos fatores, como probabilidade de sucesso técnico, probabilidade de sucesso comercial, ajuste estratégico, etc;
- Processos de seleção de ideias com *feedback* imediato para os criadores das ideias;
- Uso da teoria de opções para avaliar projetos.

2.2.1.5 Definição do Conceito

O conceito do produto a ser desenvolvido, corresponde a uma aplicação de *software*, que tem como finalidade a deteção de arritmias cardíacas, e futuramente, se possível, a sua previsão em tempo real, através da classificação de sinais resultantes de ECG. Este desenvolvimento será efetuado recorrendo ao *deep learning*, aplicando técnicas com redes neurais convolucionais (CNN) para o processamento e classificação dos dados. Com este desenvolvimento, será possível melhorar o processo de monitorização e a deteção de arritmias cardíacas por parte dos técnicos de saúde, assim como melhorar a qualidade de vida dos pacientes, contribuindo para uma redução de possíveis complicações da doença.

Através das seguintes técnicas é possível analisar este ponto chave (Koen et al., 2002):

- Definição cuidadosa das metas e resultados do projeto;
- Definição de critérios que descrevam o que pode ser considerado um projeto atrativo, tendo em conta a nível financeiro, o tamanho e o crescimento do mercado, etc.
- Compreender e determinar o limite de capacidade de desempenho da tecnologia;
- Participação por parte do cliente em testes reais ao produto (de preferência, esta participação deve começar antes mesmo da conclusão do produto).

2.2.2 Valor

A criação de valor é a chave para qualquer negócio, e qualquer atividade de negócio se trata por trocas de bens ou serviços tangíveis e/ou intangíveis e ter o seu valor aceite e recompensado por clientes (Nicola et al., 2012).

2.2.2.1 Valor para o cliente

Este conceito, "valor para o cliente" pode ser definido como uma perceção pessoal decorrente da associação entre um cliente e a oferta de uma organização e poderá ocorrer devido à

redução de sacrifício, presença de benefício, uma combinação sensata entre o sacrifício e benefício ou a uma conjugação de todos ou parte destes elementos (Woodall, 2003).

2.2.2.2 Valor Percetível

Diferentes clientes têm uma perceção de valor distinta para os mesmos produtos/serviços. Além disso, as organizações envolvidas no processo de compra podem ter perceções diferentes da entrega de valor dos clientes (Ulaga e Eggert, 2006).

O valor para o cliente pode ser interpretado com uma perspetiva longitudinal de valor com os benefícios e sacrifícios, englobando quatro valores temporais:

- Pré-compra: caracteriza-se pelos desejos e ideias preconcebidas do consumidor em relação ao valor;
- Transação: trata-se do valor para o cliente que é definido no momento da transação, aquisição ou troca;
- Pós-compra: período em que o consumidor avalia o seu grau de satisfação, que é resultado da experiência de consumo;
- Após utilização: reproduz a experiência de utilização por parte do consumidor e o período em que faz a venda ou troca.

A previsão de arritmias cardíacas pode vir a ter um papel muito importante na vida dos doentes que sofrem desta patologia, contribuindo desta forma na melhoria da qualidade de vida dos mesmos.

Com um sistema de alertas ao paciente nos momentos em que uma arritmia cardíaca poderia estar prestes a acontecer, este poderia preparar-se da melhor forma para enfrentá-la, seja tomando medicação indicada pelo médico, ou qualquer outra abordagem. Desta forma poderá ser possível reduzir a administração de medicação e obviamente reduzir ou mesmo evitar as consequências destas arritmias.

Na Tabela 1 é apresentada esta perspetiva longitudinal para o cliente relativamente ao uso deste sistema em cada momento temporal:

Tabela 1 - Perspetiva longitudinal de valor

	Benefícios	**Sacrifícios**
Pré-Compra	Qualidade de vida; Maior segurança; Maior rapidez na intervenção médica;	Preço.
Transação		Custo de aquisição;

	Benefícios	Sacrifícios
Pós-compra		Custos de entrega/instalação; Tempo de aprendizagem de uso do sistema; Custos de manutenção, reparação;
Após utilização	Deteção/Previsão de arritmias cardíacas; Qualidade de vida; Redução da medicação; Redução de consequências; Menor risco.	

2.2.3 Proposta de Valor

Uma proposta de valor define a forma como as organizações trabalham concentrando as suas atividades no melhor atendimento possível aos seus clientes enquanto o fazem de forma lucrativa (Barnes et al., 2009).

Com a descrição deste conceito, é de seguida apresentada a proposta de valor deste produto.

Desenvolvimento de um algoritmo de deteção de arritmias cardíacas, com uma elevada taxa de acerto, de forma a garantir a fiabilidade dos resultados da deteção. Este sistema, proporciona aos utilizadores, essencialmente pacientes com problemas cardíacos, um sistema de monitorização em tempo real, que garante uma maior rapidez na intervenção médica, reduzir as possíveis consequências das arritmias, reduzir a medicação associada e melhorar claramente a qualidade de vida.

Existem sistemas semelhantes no mercado, sendo que este sistema pretende destacar-se pela sua metodologia de desenvolvimento, usada no processamento dos sinais ECG. Com esta metodologia, a deteção de arritmias aumenta significativamente a taxa de acerto, para além da grande rapidez de resposta.

2.2.4 Modelo de negócio Canvas

Para que uma ideia se transforme num negócio de sucesso é preciso entregar valor ao cliente de forma a gerar o resultado esperado. Para que tal aconteça, é necessário que toda a estrutura por trás do negócio esteja a direcionada para isso. Resumidamente, deve-se identificar claramente o público-alvo, ou seja, o cliente, o problema que está a tentar resolver, a sua estratégia de resolução e ainda como este negócio se irá manter estruturado, gerando lucro.

A ideia de negócio que poderá ser associada ao tema desta dissertação é a comercialização de um produto que passa pela integração de um componente de *hardware* com um *software*. A

proposta passa pela composição de um componente físico capaz de obter os dados através de um eletrocardiograma efetuado por um paciente. Para além deste componente físico captar estes dados, têm de efetuar de seguida uma série de operações, tais como analisar, interpretar e classificar os sinais obtidos. Estas operações fazem parte do software que será desenvolvido, recorrendo a metodologias referidas nesta dissertação.

Relativamente ao modelo de negócio, esta comercialização seria efetuada ao utilizador final, ou seja, os pacientes com arritmias cardíacas ou atletas de alta competição, através de lojas de vendas de equipamentos médicos, sugeridas pelos profissionais de saúde.

Na Tabela 2, é apresentado o modelo de negócio Canvas desenvolvido, propondo desta forma um possível modelo de negócio, usando o tema desta dissertação.

Tabela 2 - Modelo de Canvas

<table>
<tr><th>Key Partners</th><th>Key Activites</th><th>Value Propositions</th><th>Customer Relationships</th><th>Customer Segments</th></tr>
<tr><td rowspan="4">Hospitais;
Centros de saúde;
Clínicas;
Empresas no setor da saúde;
Profissionais de saúde.</td><td>Desenvolvimento do software;
Comunicação com hardware;
Suporte ao cliente – Atualizações do software.</td><td rowspan="3">Software de deteção e previsão de arritmias cardíacas;
Melhorias na qualidade de vida dos pacientes;
Elevada eficácia na deteção e previsão de arritmias;
Rápida resposta e intervenção;
Menor risco.</td><td>Suporte técnico através de profissionais de saúde;
Campanhas de marketing;
Preço base variável, de acordo com a disponibilidade financeira do paciente.</td><td rowspan="3">Pacientes com problemas cardíacos;
Atletas.</td></tr>
<tr><td>Key Resources</td><td>Channels</td></tr>
<tr><td>Linguagens de Programação;
Frameworks;
Know-How.</td><td>Farmácias;
Lojas de vendas de equipamentos médicos;
Venda ao paciente através de profissionais de saúde.</td></tr>
<tr><td>Cost Structure</td><td></td><td>Revenue Streams</td><td></td><td></td></tr>
<tr><td>Desenvolvimento e manutenção da aplicação;
Custo do Hardware;
Investigação;
Licenças;
Publicidade.</td><td></td><td>Venda do produto;
Reparação/manutenção.</td><td></td><td></td></tr>
</table>

2.2.5 Método Análise Hierárquica

Existem diversos métodos analíticos de tratamento da informação que podem ser usados para calcular o valor criado.

Neste conjunto, encontram-se os métodos multicritério em tomadas de decisão, onde se enquadra o Método de Análise Hierárquica (AHP - *Analytic Hierarchy Process*), que será aplicado neste trabalho.

Para o desenvolvimento deste trabalho, existem várias tomadas de decisões que têm de ser efetuadas. Uma das mais importantes trata-se da escolha do algoritmo de classificação, que será utilizado no processo de classificação dos sinais ECG. Assim sendo, este é o problema onde será aplicado o método de análise hierárquica.

Na Figura 3 é exibida a árvore hierárquica de decisão, onde o problema está decomposto numa hierarquia, composta pelo objetivo, no primeiro nível hierárquico, três critérios associados ao problema de decisão, no segundo nível hierárquico e três alternativas disponíveis no terceiro nível hierárquico.

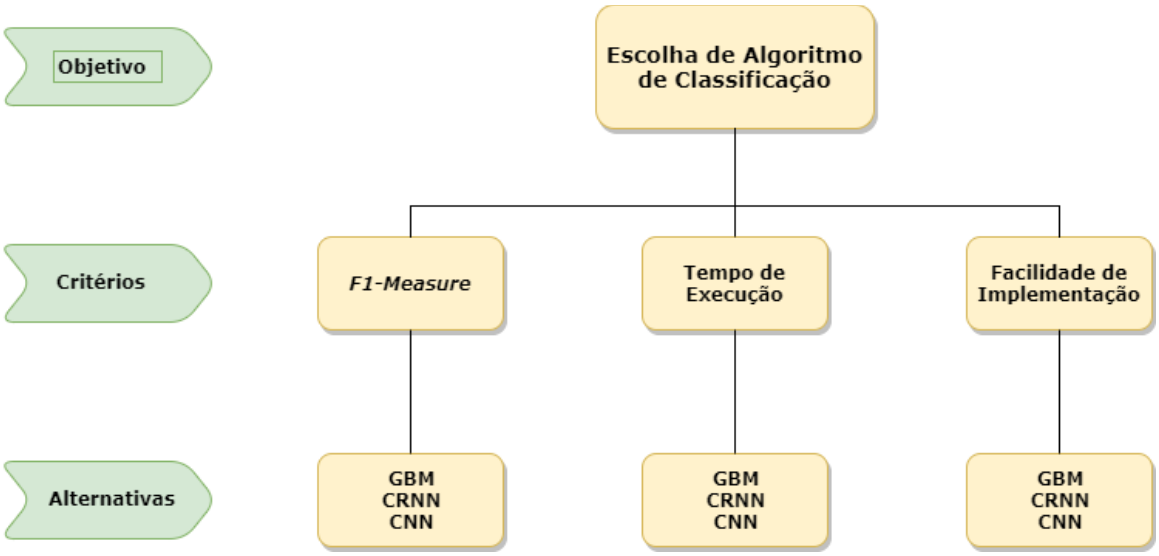

Figura 3 - Árvore hierárquica de decisão

Foram analisados diversos artigos referentes a abordagens utilizadas por algumas equipas no concurso lançado pela *PhysioNet*, de onde o problema desta dissertação surgiu.

Tendo em conta que os resultados obtidos pelos algoritmos de classificação dependem muito do conjunto de dados utilizado, a definição das alternativas e critérios apenas seria exequível através de resoluções de outros autores para o mesmo problema a que esta dissertação se refere, utilizando o mesmo conjunto de dados.

Assim sendo, nesta análise, foram definidas as seguintes alternativas, como algoritmos de classificação:

- **GBM** (*Gradient Boosting Machine*) - Utilizado por Dawid Smolen (Smolen, 2017);
- **CRNN** (*Convolutional Recurrent Neural Network*) - Utilizado pela equipa de Martin Zihlmann (Zihlmann et al., 2017);
- **CNN** (*Convolutional Neural Network*) - Utilizado pela equipa de Sandeep Chandra Bollepalli (Chandra et al., 2017).

Como critérios de decisão, foram definidos os seguintes:

- ***F1-Measure*** - Medida de avaliação utilizada para calcular a eficácia dos algoritmos de classificação das equipas, usando o mesmo conjunto de dados, de forma a definir a classificação das mesmas no concurso;
- **Tempo de execução** – Percentagem de quota de tempo de computação utilizada tendo em conta a quota alocada para cada equipa;
- **Facilidade de implementação** – Grau de dificuldade de implementação necessária para a utilização do algoritmo de forma a obter os resultados obtidos pelas equipas, tendo em conta o trabalho descrito nos artigos.

Tendo em conta os três critérios da estrutura hierárquica foi desenvolvida a seguinte matriz de comparação paritária.

Tabela 3 - Matriz de comparação par a par dos critérios

Critérios	**F1-*Measure***	**Tempo de execução**	**Facilidade de implementação**
F1-*Measure*	1	5	2
Tempo de execução	1/5	1	1/2
Facilidade de implementação	1/2	2	1
Totais Coluna	1 2/3	8	3 1/2

Os valores utilizados para a comparação pertencem à escala fundamental, definida por Saaty (Saaty, 1991).

O critério de decisão *F1-Measure* é fortemente mais importante que o critério Tempo de execução e ligeiramente mais importante que a Facilidade de implementação.

O critério Facilidade de implementação é ligeiramente mais importante que o critério Tempo de execução.

De seguida é exibida a matriz de comparação dos critérios normalizada, ou seja, com os critérios igualados à mesma unidade. Também é calculada a média aritmética dos valores de cada linha da matriz normalizada, de forma a identificar a ordem de importância de cada critério, ou seja, obtendo o vetor de prioridades (pesos dos critérios).

Tabela 4 - Matriz de comparação dos critérios normalizada e pesos estimados

Critérios	**F1-*Measure***	**Tempo de execução**	**Facilidade de implementação**	**Pesos**
F1-Measure	0,5882	0,6250	0,5714	0,5949
Tempo de execução	0,1176	0,1250	0,1429	0,1285
Facilidade de implementação	0,2941	0,2500	0,2857	0,2766
				1,0000

A definição desta matriz permite avaliar as prioridades relativas para os três algoritmos de classificação de acordo com os critérios. O critério que se destaca e que tem mais importância para a escolha do algoritmo é a medida de avaliação *F1-Measure* (com 0,5949), seguido da facilidade de implementação (com 0,2766) e por último, tempo de execução (com 0,1285).

Posteriormente é efetuada a avaliação da consistência das prioridades relativas, ou seja, é calculada a Razão de Consistência (RC), de forma a verificar a consistência dos valores da matriz.

Para calcular a RC obtém-se em primeiro lugar o valor de λ_{max} que representa o maior valor próprio da matriz normalizada, obtido a partir da seguinte equação: Ax = λ_{max} x, onde x é o vetor das prioridades.

Portanto, é efetuada a multiplicação entre a matriz normalizada com o vetor das prioridades, dando origem ao vetor soma dos pesos obtidos como resultado.

Tabela 5 - Cálculo do vetor soma dos pesos obtidos

0,5882	0,6250	0,5714		0.5949		1.79060
0,1176	0,1250	0,1429	x	0.1285	=	0.38578
0,2941	0,2500	0,2857		0.2766		0.83105

Assim sendo, dividindo os elementos do vetor soma dos pesos obtidos no passo anterior pelo correspondente valor do vetor das prioridades, e calculando de seguida a média destes valores, obtém-se o valor próprio (λ_{max}).

λ_{max} = média{1.79060/0.5949, 0.38578/0.1285, 0.83105/0.2766} = 3.005538594

Com isto, já é possível agora obter o Índice de Consistência (IC), através da seguinte fórmula:

$$IC = \frac{(\lambda \max - n)}{(n-1)}$$

Com os valores referentes ao caso em concreto, IC = (3.005538594-3) / (3-1) = 0.002769.

Para finalizar esta verificação da consistência, deve ser então calculada a RC, que é obtida pela seguinte fórmula:

$$RC = \frac{IC}{IR}$$

A medida IR (Índice Aleatório) pode ser obtida pela seguinte tabela:

Tabela 6 - Valores de IR para matrizes quadradas de ordem n

1	2	3	4	5	6	7	8	9	10
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Posto isto, é usado o valor 0.58, de ordem 3 neste cálculo do RC.

RC = IC/0.58 = 0.002769/0.58 = 0.00477465

Como este valor de RC é inferior a 0.1, pode-se concluir que os valores das prioridades relativas utilizado estão consistentes.

De seguida são construídas as matrizes de comparação paritárias para cada critério, considerando cada uma das alternativas selecionadas, efetuando também a normalização das mesmas e obtendo os vetores de prioridades (Pesos).

É importante referir que os valores atribuídos às alternativas nos diferentes critérios são originados do estudo efetuado aos artigos de onde as alternativas foram utilizadas.

Tabela 7 - Matriz de comparação paritária para o critério F1-*Measure*

F1-*Measure*	**GBM**	**CRNN**	**CNN**
GBM	1	1	3
CRNN	1	1	3
CNN	1/3	1/3	1
Total	2.3333	2.3333	7.0000

Tabela 8 - Matriz de comparação paritária para o critério F1-*Measure* normalizada e pesos estimados

F1-*Measure*	**GBM**	**CRNN**	**CNN**	**Pesos**
GBM	3/7	3/7	3/7	0,4286
CRNN	3/7	3/7	3/7	0,4286
CNN	1/7	1/7	1/7	0.1429
				1

Tabela 9 - Matriz de comparação paritária para o critério Tempo de Execução

Tempo de Execução	GBM	CRNN	CNN
GBM	1	1/2	1/6
CRNN	2	1	1/3
CNN	6	3	1
Total	9.0000	4.5000	1.5000

Tabela 10 - Matriz de comparação paritária para o critério Tempo de Execução normalizada e pesos estimados

Tempo de Execução	GBM	CRNN	CNN	Pesos
GBM	1/9	1/9	1/9	0,1111
CRNN	0.2222	0.2222	0.2222	0,2222
CNN	2/3	2/3	2/3	0.6667
				1

Tabela 11 - Matriz de comparação paritária para o critério Facilidade de Implementação

Facilidade de Implementação	GBM	CRNN	CNN
GBM	1	1/4	1/8
CRNN	4	1	1/4
CNN	8	4	1
Total	13.0000	5.2500	1.3750

Tabela 12 - Matriz de comparação paritária para o critério Facilidade de Implementação normalizada e pesos estimados

Facilidade de Implementação	GBM	CRNN	CNN	Pesos
GBM	0	0	0	0,0718
CRNN	1/3	1/5	1/5	0,2267
CNN	5/8	3/4	5/7	0.7015
				1

Após a obtenção dos valores dos vetores de prioridades, é necessário aferir a prioridade composta para as alternativas, por conseguinte, é construída uma matriz com os valores dos vetores de prioridades das alternativas e multiplica-se esta matriz com o vetor das prioridades dos critérios, obtido no início do método.

Tabela 13 - Escolha da alternativa

	F1-*Measure*	**Tempo de Execução**	**Facilidade de Implementação**				
GBM	0,4286	0,1111	0,0718		0.5949		0,28911
CRNN	0,4286	0,2222	0,2267	x	0.1285	=	0.34623
CNN	0,1429	0,6667	0,7015		0.2766		**0,36472**

Assim sendo, a terceira alternativa "CNN" aparece como a mais indicada para ser utilizada como algoritmo de classificação (com 0.36472), em função dos critérios definidos e das suas respetivas importâncias.

3 Estado da arte

Neste capítulo é apresentado o estado da arte relacionado com o contexto do projeto. Inicia-se por um estudo geral acerca de *machine learning*, alguns dos algoritmos mais utilizados atualmente, assim como as principais fases e técnicas utilizadas, sobretudo nos problemas de classificação.

De seguida é efetuado um estudo sobre algumas abordagens já existentes, relativas ao mesmo problema, e por fim, um estudo sobre as tecnologias atuais no desenvolvimento orientado ao *deep learning*.

3.1 *Machine Learning*

Resumidamente, os sistemas de *Machine Learning* são sistemas que aprendem automaticamente programas a partir de dados. O uso destes sistemas tem vindo a espalhar-se rapidamente na última década. À medida que são disponibilizados mais dados, problemas mais ambiciosos podem ser abordados. Estes sistemas são já usados em diversos contextos como sistemas de recomendação, colocação de anúncios, pesquisas da web, filtros de spam, entre outros (Domingos, 2012).

Essencialmente, os algoritmos de *Machine Learning* podem ser divididos nas categorias de aprendizagem supervisionada e aprendizagem não supervisionada.

Neste trabalho, tratando-se de um problema de classificação de sinais provenientes de ECG, serão usados algoritmos de classificação, que fazem parte da aprendizagem supervisionada. Assim sendo, será dada uma maior ênfase sobre esta categoria de algoritmos neste capítulo.

3.1.1 Aprendizagem Supervisionada

Os algoritmos de aprendizagem supervisionada são algoritmos que usam um conjunto de dados que contêm características (atributos), em que cada exemplo está associado a uma categoria (classes). O modelo é construído através da definição das classes e dos exemplos de cada classe.

Com base no conhecimento prévio aprendido do conjunto de dados de treino, espera-se que a máquina, avalie e preveja as saídas (classes/categorias) para dados de entrada que não conhece. A ideia é que a máquina "aprenda" a partir da experiência de treino, pelos valores rotulados do conjunto de dados, sendo capaz depois de identificar/classificar exemplos não rotulados com alta precisão (Goodfellow et al., 2016).

Segundo *Jason Brownlee* (Brownlee J., 2016), os problemas de aprendizagem com supervisão podem ser agrupados em dois tipos de algoritmos:

- Classificação: algoritmo que classifica o resultado de uma previsão, ou seja, quando a variável de saída é uma categoria/etiqueta;
- Regressão: algoritmo que tenta prever um determinado resultado com base nas variáveis anteriores.

Alguns exemplos de algoritmos supervisionados de *Machine Learning* são:

- *Linear Regression* para problemas de regressão;
- *Random Forest* para problemas de classificação e regressão;
- *Support Vector Machines* (SVM) para problemas de classificação;
- *k-Nearest Neighbors* (kNN) para problemas de classificação e regressão;
- *Artificial Neural Networks* (ANN) para problemas de classificação e regressão.

3.1.2 Aprendizagem Não Supervisionada

Os algoritmos de aprendizagem não supervisionada são algoritmos que usam um conjunto de dados que contêm muitas características, e em seguida, aprendem as propriedades úteis da estrutura desse conjunto de dados. O objetivo da aprendizagem não supervisionada é modelar a estrutura ou a distribuição subjacente aos dados, a fim de aprender mais sobre os mesmos. Ao contrário da aprendizagem supervisionada, estes algoritmos não possuem informação sobre a classe associada a cada exemplo, sendo a aprendizagem efetuada através da descoberta de similaridades nos dados (agrupamentos de dados com características semelhantes) (Goodfellow et al., 2016).

De acordo com *Jason Brownlee* (Brownlee J., 2016), os problemas de aprendizagem sem supervisão podem ser agrupados nos seguintes tipos de algoritmos:

- *Clustering*: um problema de *clustering* é onde se quer descobrir os agrupamentos inerentes aos dados, como por exemplo o agrupamento de clientes pelo comportamento de compra;
- Associação: é um algoritmo que tenta aprender através de regras de associação, em grandes porções dos dados, como por exemplo, pessoas que compram o produto X também tendem a comprar o produto Y.

Alguns exemplos de algoritmos sem supervisão de *Machine Learning* são:

- *k-means* para problemas de *clustering*;
- *Apriori* para problemas de aprendizagem de regras de associação.

3.1.3 Pré-Processamento

O processo de deteção de arritmias cardíacas passa por várias fases, sendo que a primeira delas é o pré-processamento dos dados.

A fase de pré-processamento visa essencialmente "conhecer" os dados de forma a prepará-los para a fase seguinte.

Esta fase é muito importante nos casos em que os dados, os sinais ECG no caso deste projeto, contêm ruído, dificultando a sua análise. Este ruído pode ter como origem, por exemplo, a rede elétrica ou mesmo a má colocação dos elétrodos no paciente. Desta forma, é necessário aplicar um filtro aos sinais, de modo a reduzir o ruído que interfere de forma negativa nos resultados obtidos.

Muitas aplicações exigem um pré-processamento complexo, pois o conjunto de dados original pode não ser completamente confiável, completo e consistente à partida, provocando uma dificuldade acrescida para muitas arquiteturas de *deep learning* trabalharem sobre o mesmo.

A formatação de imagens para ter a mesma escala é o único tipo de pré-processamento que é estritamente necessário, no caso das CNN (para processamento e análise de imagens). Existem ainda arquiteturas que exigem imagens de um tamanho padrão, levando a que seja necessário recortar ou dimensionar as imagens, de forma a que se ajustem a esse tamanho (Goodfellow et al., 2016).

O pré-processamento de dados é composto pelo seguinte conjunto de etapas (Han et al., 2012):

3.1.3.1 Limpeza dos dados

A limpeza dos dados refere-se ao processo que aumenta a qualidade dos dados de entrada, removendo dados ruidosos, completando dados incompletos e corrigindo inconsistências nos mesmos.

Na eventualidade desta etapa não ser aplicada, torna-se complicado considerar que os dados são fiáveis, o que consequentemente leva a uma desconfiança nos resultados de qualquer processo de aprendizagem de um algoritmo (Han et al., 2012).

3.1.3.2 Integração dos dados

A integração dos dados trata-se do procedimento para reunir dados provenientes de diversas fontes num armazenamento de dados coerente, como uma base de dados, por exemplo. No entanto, como os dados provém de múltiplas fontes, a probabilidade de haver inconsistências e redundâncias nos mesmos é bastante alta. Por esta razão, geralmente as etapas limpeza dos dados e integração dos dados são executadas como uma só etapa de pré-processamento na preparação dos dados para um armazenamento (Han et al., 2012).

3.1.3.3 Redução dos dados

Esta etapa é responsável por reduzir o tamanho dos dados, por exemplo, agregando, eliminando recursos redundantes/irrelevantes ou aplicando a técnica de *clustering*. A aplicação de técnicas de redução de dados permite que os dados de entrada tenham menos volume,

mantendo sempre a sua integridade e produzindo os mesmos resultados analíticos. Algumas dessas técnicas incluem:

- Redução de dimensionalidade - Remove atributos irrelevantes do conjunto de dados, de forma a encontrar o conjunto mínimo de atributos, embora o resultado final deva ser idêntico ou melhor que o resultado original com todos os atributos;
- Redução de numerosidade - Os dados são substituídos por representações alternativas menores.

3.1.3.4 Transformação dos dados

Nesta etapa, os dados são transformados ou consolidados de modo que o processo resultante seja mais eficiente e os padrões encontrados sejam mais fáceis de compreender. Estas transformações ou consolidações de dados resultam de operações de agregação, generalização, normalização e discretização dos dados (Han et al., 2012).

É importante mencionar que grande parte dos erros são corrigidos durante esta etapa de transformação de dados, nomeadamente erros que têm como base erro humano, como por exemplo os erros originados por um processamento de dados incorreto.

As técnicas de *Dataset Augmentation* e de *Data Balance* fazem parte desta etapa de pré-processamento.

A técnica de *Dataset Augmentation* pode ser vista como uma forma de pré-processar os dados de treino, podendo ser uma forma de reduzir o erro de generalização da maioria dos modelos. Sucintamente, trata-se de uma técnica com o objetivo de criar novos "dados" com diferentes orientações, através dos dados existentes. Escalas, translações, rotações, inversões ou mesmo corte das imagens em locais ligeiramente diferentes são alguns exemplos de aumento de dados (Goodfellow et al., 2016). Importante referir que esta técnica está associada ao aumento do conjunto de dados para o treino do modelo, podendo ser bastante útil no caso de haver uma escassa quantidade de dados.

As técnicas de *Data Balance* têm como objetivo, garantir que todas as classes de saída de um determinado algoritmo de aprendizagem estejam balanceadas, ou seja, no conjunto de dados não deverá haver uma diferença significativa no número de exemplos de diferentes classes. Algumas estratégias que podem ser seguidas para isto, podem passar pelo redimensionamento do conjunto de dados, ignorando exemplos das classes que possuem um tamanho maior de exemplos, ou usar custos diferentes para classificar cada uma das classes.

A Figura 4 resume as quatro etapas de pré-processamento de dados descritas anteriormente.

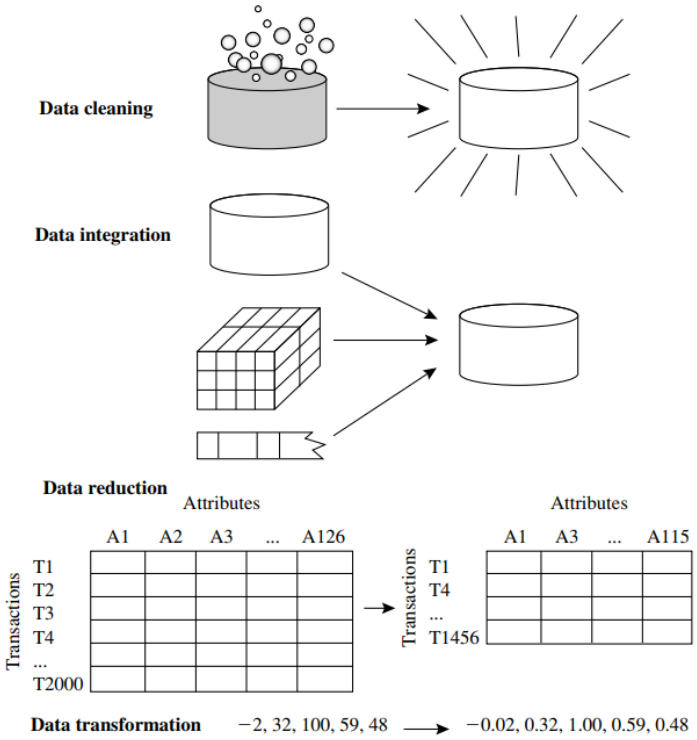

Figura 4 - Etapas do Pré-Processamento de Dados (Han et al., 2012)

Segundo Dorian Pyle (Pyle D, 1999), o pré-processamento é a etapa que requer um maior esforço envolvido ao longo de todo o processo, estimando ainda que cerca de 80% do tempo despendido no processo seja utilizado nesta etapa.

3.1.4 Extração de Atributos

A segunda etapa no processo é a extração de atributos, aplicada após o pré-processamento dos dados, tratando-se de uma etapa onde serão identificados os atributos presentes no conjunto de dados.

Mais concretamente, a extração de atributos é extrair e converter a informação relativa aos dados de entrada num conjunto de características chamado de vetor de características, reduzindo o padrão de representação de dados. O conjunto de características extrairá as informações relevantes dos dados de entrada de forma a executar depois a tarefa de classificação (Tamil et al., 2008).

Existem bastantes metodologias de extração de atributos, sendo que existem algumas que são frequentemente utilizadas em análise de sinal, segundo o estudo efetuado neste trabalho, tais como o *Fast Fourier Transform* (FFT) e o *Wavelet Transform* (WT).

O *Fast Fourier Transform* é uma ferramenta computacional que facilita a análise do sinal, através de ferramentas matemáticas, como análise de espectro de potência e simulação de filtro por meio de computadores digitais (Cochran et al., 1967).

O *Wavelet Transform* surgiu nos últimos anos como uma poderosa ferramenta de análise de tempo-frequência e codificação de sinais, usada para a interrogação de sinais não-estacionários complexos, sendo este um problema dos sinais ECG. Esta ferramenta é capaz de representar sinais em distintas resoluções, dilatando e compactando as suas funções de base. A principal vantagem do WT é que possui um tamanho de janela variável, sendo ampla em baixas frequências e estreita em altas frequências, levando assim a uma resolução tempo-frequência bastante boa em todas as gamas de frequência (Tamil et al., 2008).

No contexto do projeto, a maioria dos dados do ECG são dados de séries temporais com duração de cerca de 30 ou 60s, com tempo de amostragem de cerca de 0,003s.

Existem muitos algoritmos de *machine learning* com algumas opções de extração de atributos que lidam com dados de séries temporais. Mas neste trabalho, serão usadas as redes neurais convolucionais, que possuem desde logo extratores de atributos, nas primeiras camadas de neurónios. A extração automática de atributos é uma das grandes vantagens que o *deep learning* tem sobre os algoritmos tradicionais de *machine learning*. Com esta extração de atributos automática, a obrigatoriedade de conhecimento especializado por parte de quem desenvolve deixa de existir. A criação dos atributos de forma manual deixa também de fazer sentido, contribuindo desta forma para uma redução de tempo na construção de um protótipo de um algoritmo de classificação de ECG (Pyakillya et al., 2017).

3.1.5 Algoritmos de Classificação

Nesta secção são apresentados alguns dos principais algoritmos de aprendizagem supervisionada, mais concretamente algoritmos de classificação.

3.1.5.1 k-Nearest Neighbors (kNN)

Este algoritmo, consiste num conjunto de técnicas que podem ser usadas para classificação ou regressão. É um algoritmo de aprendizagem não paramétrico, ou seja, não faz qualquer pressuposto na distribuição dos dados subjacentes, sendo a estrutura do modelo determinada a partir dos mesmos (Goodfellow et al., 2016).

Assim sendo, pode-se assumir que este algoritmo é recomendável para um estudo de classificação quando não há conhecimento suficiente sobre os dados.

Na classificação usando este algoritmo, o resultado é uma associação de classe. Um objeto é classificado pela maioria dos votos dos seus vizinhos, sendo o objeto atribuído à classe mais comum entre os seus vizinhos mais próximos (Tweedale et al., 2012).

Segundo Eduardo Luz (Luz et al., 2016), os algoritmos com uma abordagem preguiçosa, como é o caso deste, não são muito usados para o problema da classificação de arritmias, uma vez que a sua eficiência está intimamente ligada ao conhecimento prévio para realizar a classificação de cada amostra, que é representada pelo conjunto de treino completo, levando assim a um alto

custo computacional durante a fase de teste. Este custo pode invalidar o seu uso para diagnóstico em tempo real.

3.1.5.2 Support Vector Machines

O algoritmo *Support vector machines* (SVM) é um algoritmo de classificação muito poderoso, desenvolvido por Cortes e Vapnik (Cortes et al., 1995), sendo uma das abordagens mais influentes para a aprendizagem supervisionada. É um algoritmo muito popular no *machine learning* para reconhecimento de padrões, especialmente para classificação binária, ou seja, classificação para duas classes.

É semelhante ao algoritmo *"logistic regression*", também usado para tarefas de classificação, dado que é conduzido por uma função linear supervisionada $w^Tx + b$. No entanto, ao contrário do algoritmo *"logistic regression*", o algoritmo SVM não fornece probabilidades, gerando apenas uma identidade de classe. Basicamente, quando $w^Tx + b$ é positivo, o SVM prevê que a classe positiva está presente, da mesma forma, quando é negativa, este prevê que a classe negativa está presente (Goodfellow et al., 2016).

A máquina de aprendizagem recebe um conjunto de dados de entrada de treino, pertencentes a duas classes, com categorias associadas. Os dados de entrada estão na forma de vetores de atributos e o SVM encontra o híper-plano separando os dados de entrada e deixando a melhor margem de separação entre eles, tal como pode ver na Figura 5. Se os dados não forem linearmente separáveis, os pontos de dados são projetados num espaço geralmente de maior dimensão, onde os pontos de dados se tornam efetivamente linearmente separáveis (Kampouraki et al., 2009).

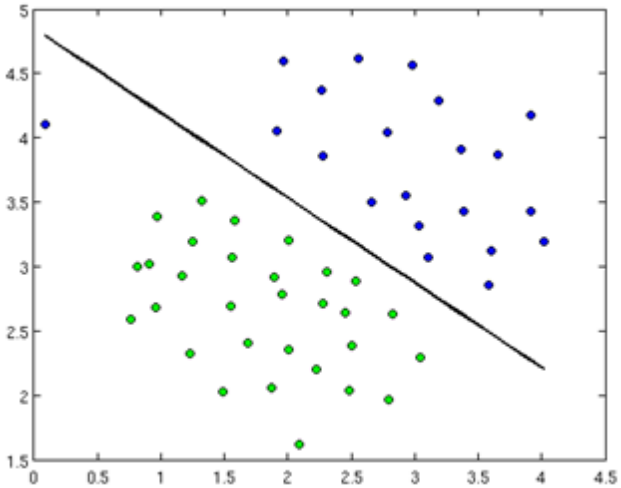

Figura 5 - Support Vector Machines[3]

[3] Image source: http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex7/ex7.html

Apesar da extrema precisão dos algoritmos de classificação SVM, é importante mencionar que estes acabam por se revelar uma abordagem muito demorada quando aplicada a um processamento de grandes conjuntos de dados (Samui et al., 2017).

3.1.5.3 Random Forest

Segundo *Leo Breiman* e *Adele Cutler* (Random Forests, 2016), o algoritmo de *Random Forest*, consiste na criação de múltiplas árvores de decisão a partir da entrada de um conjunto de dados, que são usados de seguida para classificar um novo exemplo. Basicamente, cada árvore é baseada num determinado número de exemplos do conjunto de dados de treino, selecionados aleatoriamente. Cada nó de cada árvore é construído a partir de um subconjunto aleatório dos atributos. Ao receber um exemplo de teste cada árvore irá decidir (votar) sobre qual a classe a que pertence. A classe mais votada será a classe prevista pelo modelo.

Este algoritmo é muito utilizado não só para tarefas de classificação, mas também para tarefas de regressão, usando o *Random Forest regressor*. Nos problemas de classificação, as árvores tendem a votar na classe mais popular, já nos problemas de regressão, as suas respostas são ponderadas de forma a poder obter uma estimativa.

O *Random Forest* é extremamente eficiente, mesmo com grandes conjuntos de dados, alcançando resultados muito bons, num curto tempo de processamento. Em diversas experiências, consegue melhor taxa de acerto do que qualquer outro algoritmo, não existindo sobreajustamento, independentemente do número de árvores geradas (Breiman, 2001).

Na Figura 6, pode observar-se como funciona o algoritmo *Random Forest*, com um exemplo com três árvores de decisão, e uma classificação obtida de cada uma delas. A previsão final é baseada na votação por maioria, o que no caso da figura será "Classe B".

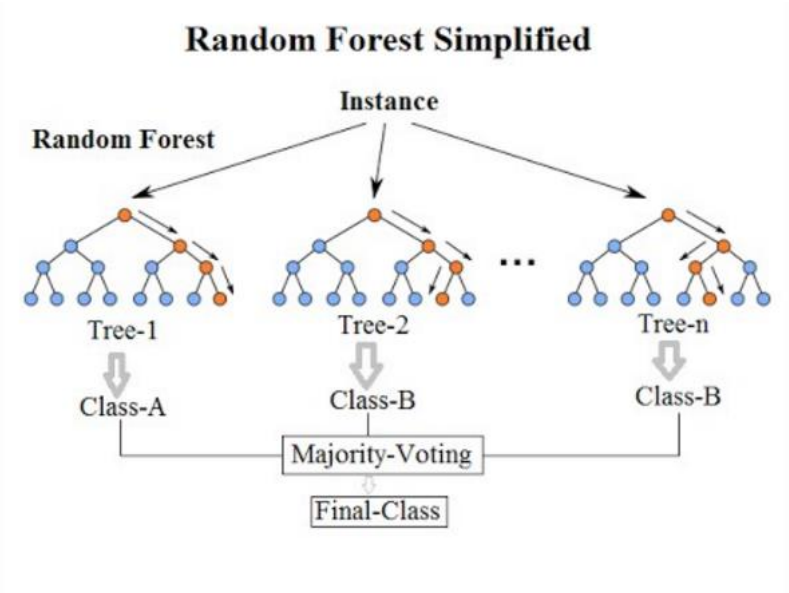

Figura 6 - Algoritmo *Random Forest* (Mishra et al., 2018)

Este algoritmo tem como principais características:

- Grande taxa de acerto entre os diversos algoritmos;
- Enorme eficiência na execução em grandes bases de dados;
- Fornecimento de estimativas de quais variáveis são importantes na classificação;
- Possui métodos para equilibrar erros em *datasets* não balanceados de população de classes;
- Possui um método eficaz para estimar os dados em falta e mantém a taxa de acerto mesmo quando está em falta uma grande parte dos dados.

3.1.5.4 Artificial Neural Networks

As *Artificial Neural Networks* (ANN), geralmente chamadas de redes neurais, são sistemas computacionais distribuídos, constituídos por vários neurónios artificiais (inspirado no funcionamento do cérebro humano) que estão correlacionados de acordo com a arquitetura de rede. O objetivo da rede neuronal é converter os dados de entrada em dados de saída significativos, sendo que o modo de ensino pode ser com ou sem supervisão (Saravanan et al., 2014).

Na criação de uma rede neuronal, é fundamental determinar corretamente quais os dados de entrada, quantas camadas podem ser usadas e ainda qual o tipo de função de ativação que será implementado. Desta forma, uma ANN é caracterizada pela sua arquitetura, o seu algoritmo de processamento e o seu algoritmo de aprendizagem. A arquitetura especifica a quantidade de neurónios assim como a forma como estes estão conectados. O algoritmo de processamento especifica como a rede calcula as saídas para um determinado conjunto de entradas, através de um conjunto de pesos. Por fim, o algoritmo de aprendizagem especifica como a rede adapta os seus pesos tendo como base os conjuntos de dados de entrada de treino (Song et al., 1996).

A principal vantagem das ANN é a sua adaptabilidade, uma vez que estas são capazes de aprender com os exemplos que lhe são apresentados, muitas vezes encontrando relações muito pormenorizadas entre os dados, que são perdidas até mesmo por especialistas. Esta característica é bastante útil para os problemas em que existem muitos parâmetros potenciais que podem afetar a solução. Trazem, no entanto, algumas desvantagens, tais como a inexistência de uma forma definitiva de escolha da arquitetura ideal e de encontrar a melhor solução, para além de depender sempre da precisão do conjunto de treino (Song et al., 1996).

Na Figura 7 são apresentadas as diferentes camadas numa típica ANN.

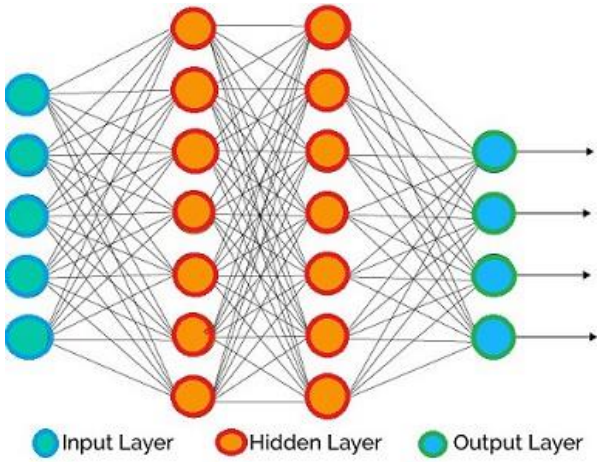

Figura 7 - Camadas de uma *Artificial Neural Network* (Xenon Stack, 2017)

A camada de entrada (*Input Layer*) contém as unidades (neurónios) que recebem a entrada de dados do exterior sobre o qual a rede irá aprender e reconhecer sobre o processo, enquanto a a camada de saída (*Output Layer*) contém os neurónios que respondem às informações sobre como aprendeu qualquer tarefa. Entre estas camadas, estão os neurónios da camada oculta (*Hidden Layer)*, que é responsável por transformar a entrada em algo que a unidade de saída possa usar de alguma forma (Xenon Stack, 2017).

3.1.6 Deep Learning

O *deep learning* trata-se de uma abordagem de Inteligência Artificial, mais especificamente, um tipo de *machine learning*, que permite que os sistemas de computador melhorem com a experiência e os dados. Este tipo de aprendizagem alcança um grande poder e flexibilidade pela forma como aprende a representar, baseado em hierarquias de conceitos, em que cada conceito é definido em relação a conceitos mais simples, e representações mais abstratas são definidas relativamente às menos abstratas (Goodfellow et al., 2016).

Esta abordagem é indicada para treinar ANN's, possuindo pelo menos duas camadas ocultas, como se pode ver na Figura 8. Esta é aliás, a principal diferença de uma *Deep Neural Network* (DNN) de uma ANN simples, a quantidade de camadas ocultas.

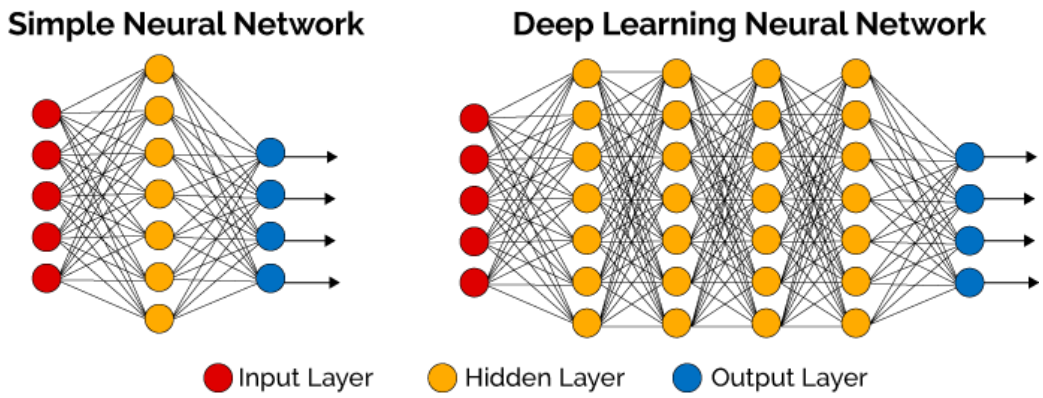

Figura 8 - Rede Neuronal Simples vs Rede Neuronal *Deep Learning* (Towards Data Science, 2018)

Esta abordagem é conhecida por resultados bastante confiáveis na resolução de problemas relacionados com processamento de imagens, mas também é indicada para problemas complexos com um grande número de características como entrada (Goodfellow et al., 2016).

Segundo Ignatov (Ignatov, 2015), ao lidar com dados tão complexos como sinais de ECG, parece razoável usar *machine learning*, tanto para a extração de atributos como para a classificação. Esta abordagem é bastante utilizada em muitos problemas de análise de dados relacionados com o problema considerado, tais como: reconhecimento de fala, classificação de imagem, processamento de sinais de EEG. Atualmente, um progresso considerável na resolução dessas tarefas foi feito por modelos baseados em arquitetura de *deep learning*.

O *deep learning* está subdividido em quatro arquiteturas fundamentais, sendo que neste documento será dado um maior detalhe sobre a CNN, visto ser a arquitetura adotada para o desenvolvimento do projeto. Um maior detalhe sobre estas arquiteturas pode ser encontrado em (Gibson et al., 2017).

3.1.6.1 Unsupervised Pretrained Networks (UPNs)

Neste grupo, existem três arquiteturas específicas (Gibson et al., 2017):

- *Autoencoders*: *Auto-Encoder* é usado para restaurar a entrada, devendo ter desta forma uma camada de saída capaz de restaurar a entrada. Isto implica um cuidado redobrado na seleção da função de ativação.

- *Deep Belief Networks (DBNs)*: Composição de redes não supervisionadas, onde a camada oculta de cada sub-rede serve como camada visível para a próxima camada. Basicamente, é composta de unidades ocultas que envolvem conexões entre camadas e não entre unidades em cada camada. Esta arquitetura é utilizada em muitas aplicações e serviços da vida real, como descoberta de drogas, eletroencefalografia, etc.

- *Generative Adversarial Networks (GANs)*: As GANs são um exemplo de uma rede que usa aprendizagem não supervisionada para treinar dois modelos em paralelo. Uma

característica importante dos modelos gerativos em geral é que estes possuem uma contagem de parâmetros significativamente menor que o normal, em relação à quantidade de dados em que a rede está a ser treinada.

3.1.6.2 Recurrent Neural Networks

As redes neurais recorrentes fazem parte da família de redes neurais *feed-forward* diferindo das restantes redes deste género na capacidade de enviar informação ao longo do tempo, tal como é ilustrado na Figura 9.

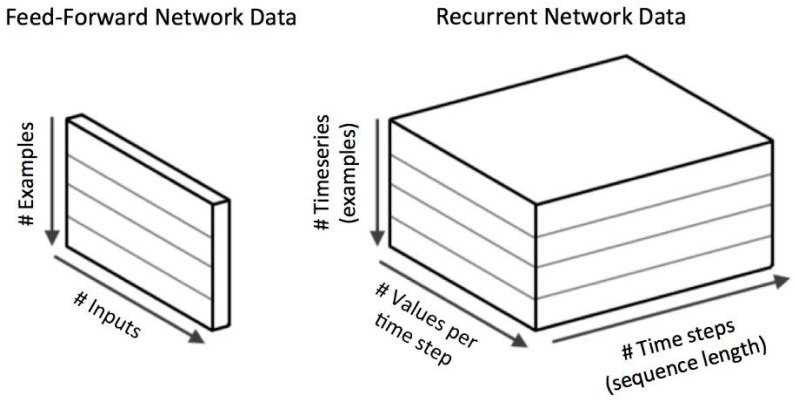

Figura 9 - Rede neuronal *feed-forward* vs Rede neuronal recorrente (Gibson et al., 2017)

Segundo Adam Gibson e Josh Patterson (Gibson et al., 2017), em tempos, estas redes foram difíceis de treinar, no entanto, com os avanços na pesquisa (otimização, arquiteturas, paralelismo, etc) estas tornaram-se mais acessíveis.

Estes tipos de redes foram concebidos para reconhecer sequências, por exemplo, um sinal de fala ou um texto.

3.1.6.3 Recursive Neural Networks

A arquitetura de uma rede neuronal recursiva consiste numa matriz de peso partilhado e uma estrutura de árvore binária que permite que a rede recursiva aprenda sequências variadas de palavras ou partes de uma imagem.

As redes neurais recursivas, tal como as redes neurais recorrentes, podem lidar com entradas de comprimento variável. A principal diferença é que as redes neurais recorrentes têm a capacidade de modelar as estruturas hierárquicas no conjunto de dados de treino.

Uma rede neuronal recursiva é muito parecida com uma rede hierárquica na qual não há qualquer aspeto de tempo para a sequência de entrada, no entanto a entrada deve ser processada hierarquicamente em forma de árvore (Gibson et al., 2017).

3.1.6.4 Convolutional Neural Networks (CNNs)

A rede neuronal convolucional é uma classe de *deep learning* com redes neurais artificiais *feed-forward* que é aplicada à análise de imagens. O objetivo de uma CNN é aprender características de ordem superior nos dados por intermédio de convoluções.

Este tipo de redes é muito adequado para o reconhecimento de objetos em imagens, sendo consistentemente superior em competições de classificação de imagem. Podem identificar rostos, pessoas, placas de rua, e muitos outros aspetos de dados visuais. As CNN's sobrepõem-se até à análise de texto por intermédio do reconhecimento ótico de caracteres, sendo também úteis no processamento de linguagem natural. A análise de som é também um ponto forte das CNN's (Gibson et al., 2017).

A eficácia das CNN's no reconhecimento de imagem é uma das principais razões pelas quais o mundo reconhece a capacidade do *deep learning*. A Figura 10 mostra como as CNN's são boas em construir posições e extrair atributos a partir de dados de uma imagem.

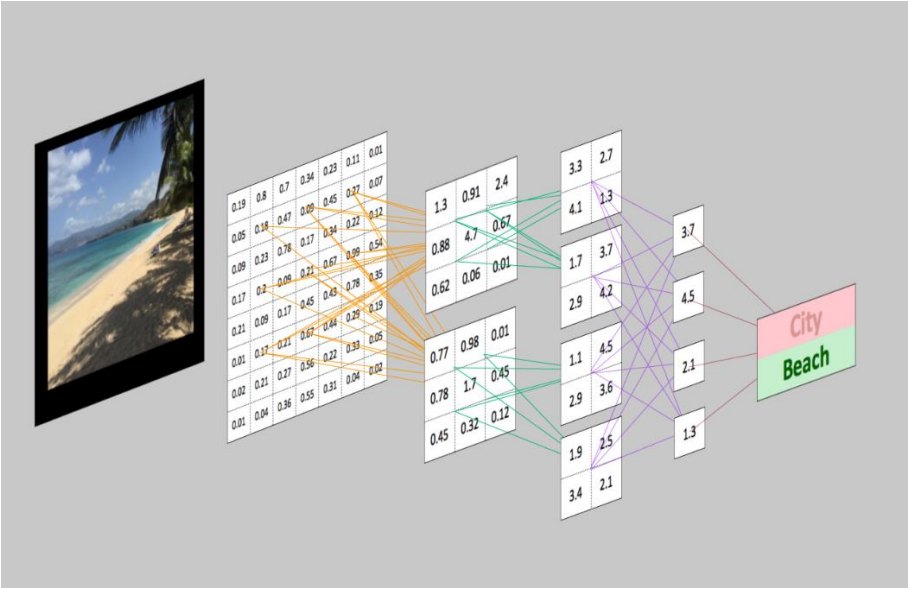

Figura 10 - Extração de Características numa CNN (Gibson et al., 2017)

As CNN's transformam os dados da camada de entrada através de todas as camadas conectadas num conjunto de pontuações de classe fornecidas pela camada de saída. Existem muitas variações da arquitetura CNN, mas estas são baseadas no padrão de camadas, tal como está demonstrado na Figura 11.

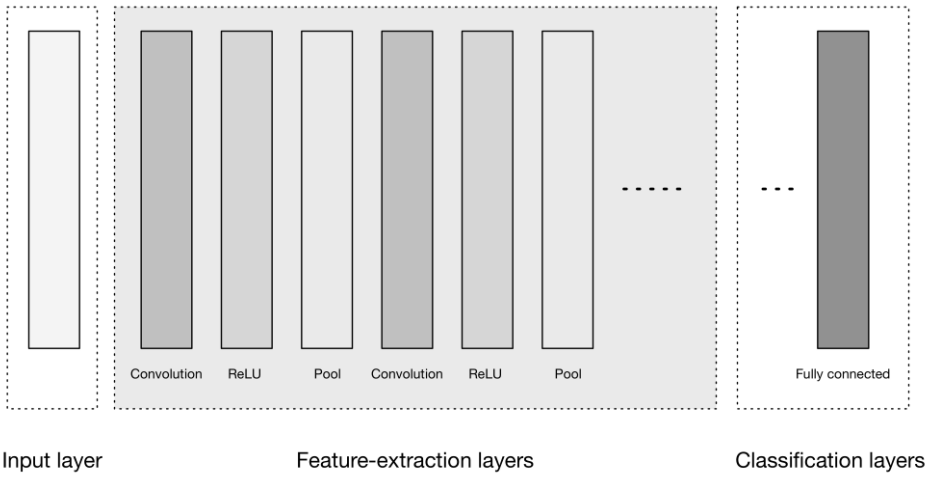

Figura 11 - Arquitetura genérica de uma CNN (Gibson et al., 2017)

Na Figura 11, é possível observar 3 grupos principais:

1. Camada de entrada

As camadas de entrada são responsáveis por carregar os dados de entrada das imagens para processamento na rede. Estes dados de entrada especificam a largura, a altura e o número de canais (profundidade). Normalmente, o número de canais é três, para os valores RGB de cada pixel.

2. Camadas de extração de características

As camadas de extração de características têm o seguinte padrão de sequência repetitivo: Camada de convolução, camada da função de ativação e camada de *pooling.* Estas camadas são as responsáveis por encontrar as diversas características das imagens, construindo progressivamente características de ordem superior.

Uma convolução é um tipo especializado de operação linear, sendo definida como uma operação matemática que descreve uma regra para combinar dois conjuntos de informação. Na Figura 12, é ilustrada a operação de convolução. O *kernel* é deslizado pelos dados de entrada para produzir os dados de saída (características). Em cada etapa, o *kernel* é multiplicado pelos valores dos dados de entrada dentro dos seus limites, criando uma única entrada no mapa de saída (atributos ou características).

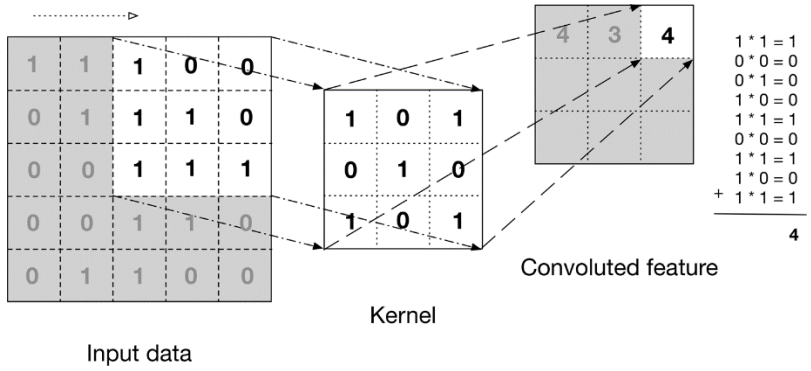

Figura 12 - Convolução (Gibson et al., 2017)

3. Camadas de classificação

Por fim, as camadas de classificação, nas quais se tem uma ou mais camadas totalmente conectadas para obter os atributos de ordem mais alta e produzir probabilidades ou pontuações de classe. Estas camadas estão totalmente conectadas a todos os neurónios da camada anterior, como o próprio nome indica.

3.1.7 Funções de Ativação

As funções de ativação são as responsáveis pelo processamento de cada neurónio nas redes neurais, nas camadas ocultas, determinando como estes serão transferidos entre as camadas da rede, tendo em consideração a sua entrada.

O design de unidades ocultas, como é o caso das funções de ativação, é uma área que está extremamente ativa na investigação, não possuindo ainda muitos princípios teóricos definitivos. Por norma, não é possível prever antecipadamente que função de ativação funcionará melhor num determinado projeto. O processo de design consiste em tentativa e erro, intuindo que a função de ativação pode funcionar bem e, em seguida, treinando uma rede com esse tipo de função, avaliando o seu desempenho num conjunto de validação (Goodfellow et al., 2016).

Nesta secção são apresentadas algumas das principais funções de ativação, que foram consideradas no desenvolvimento deste projeto.

3.1.7.1 Sigmoid

A função logística Sigmoid é uma função não-linear, que pode ser calculada pela fórmula matemática $\sigma\,(x) = \frac{1}{1+\exp(-x)}$, surgindo frequentemente a trabalhar com distribuições de probabilidade, especialmente distribuições de probabilidade usadas em modelos de *deep learning*. Consiste na transformação de um valor real num valor entre 0 e 1, em que, um número negativo relativamente grande é transformado em 0, enquanto que um número positivo é transformado em 1. A função sigmoid satura quando a sua entrada é muito positiva ou muito

negativa, o que significa que a função se torna insensível a pequenas alterações na sua entrada (Goodfellow et al., 2016).

Na Figura 13 está representada esta função, sendo possível observar a não-linearidade da mesma.

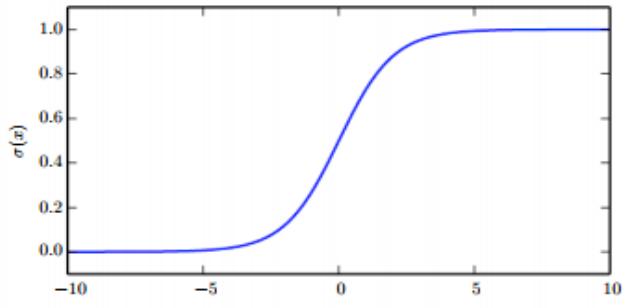

Figura 13 - Função de Ativação Sigmoid (Goodfellow et al., 2016)

3.1.7.2 Hyperbolic Tangent (Tanh)

A função de ativação Tanh é também uma função sigmoidal, representada através da fórmula matemática $\tanh(x) = 2\sigma(2x) - 1$, onde σ representa a função de ativação logística sigmoid (mencionado na secção 3.1.7.1), consistindo na transformação de um valor real num intervalo de -1 e 1.

Segundo *Goodfellow*, antes da introdução de unidades lineares retificadas, a maioria das redes neurais utilizava as funções de ativação sigmoid ou tanh. *Goodfellow* refere ainda que, quando uma função de ativação sigmoidal deve ser usada, a função tangente hiperbólica, geralmente funciona melhor do que a função sigmoid (Goodfellow et al 2016).

Na Figura 14, está representada esta função, comparativamente à função Sigmoid.

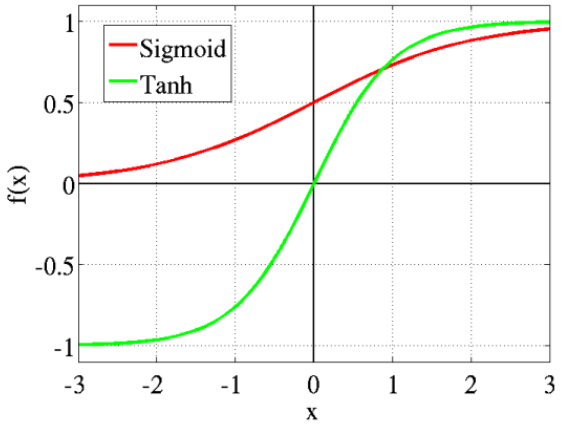

Figura 14 - Função de Ativação Tanh vs Sigmoid (Glorot et al., 2011)

3.1.7.3 Rectified Linear Unit (ReLU)

A função de ativação linear retificada (Relu), têm como intervalo [0, ∞], e define-se pela fórmula matemática $f(x) = \max(0, x)$, como se pode ver na Figura 15.

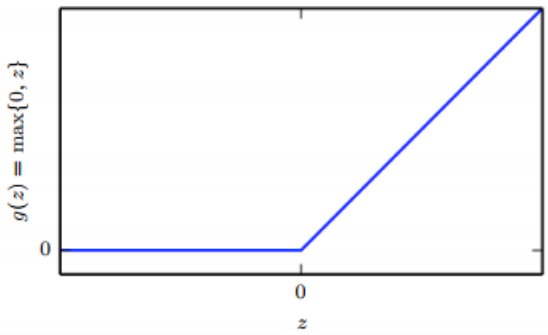

Figura 15 - Função de Ativação ReLU (Goodfellow et al., 2016)

Esta função de ativação é a função de ativação padrão (mais utilizada) em grande parte das redes neurais *feedforward*.

As unidades lineares retificadas são de otimização simples, uma vez que são muito semelhantes às unidades lineares, sendo que a única diferença entre elas é que uma unidade linear retificada produz zero em metade do seu domínio. Uma desvantagem das unidades lineares retificadas é que elas não podem aprender através de métodos baseados em gradientes em exemplos para os quais a sua ativação é 0. (Goodfellow et al., 2016)

3.1.7.4 Softplus

A função Softplus é uma versão suave da função de ativação linear retificada, como se pode verificar na Figura 16, sendo definida pela expressão matemática $\log(1 + \exp(x))$. (Goodfellow et al., 2016)

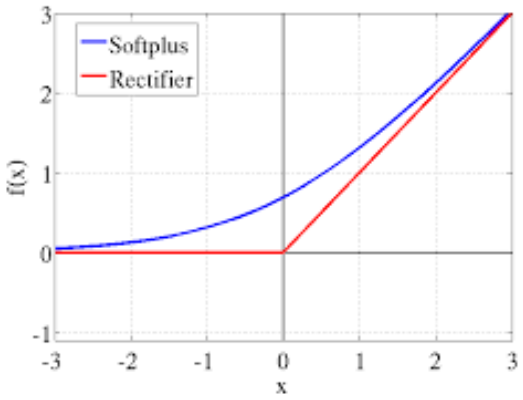

Figura 16 - Função de Ativação Softplus vs ReLU (Glorot et al., 2011)

Segundo *Xavier Glorot*, *Antoine Bordes* e *Yoshua Bengio* (Glorot et al., 2011), o uso do Softplus é fortemente desencorajado, uma vez que encontraram resultados bastante melhores no ReLU, numa comparação que efetuaram entre ambos.

3.1.8 Funções de Otimização

As funções de otimização, também conhecidas como função de treino, são usadas para melhorar a atualização de pesos e *biases* de neurónios na rede neuronal, de forma a reduzir o erro, ou seja, refere-se à tarefa de minimizar ou maximizar alguma função *f*(*x*), alterando *x*. A maioria dos algoritmos de *Deep Learning* envolve algum tipo de otimização.

Denominam-se funções de otimização, pois ajudam a minimizar a taxa de perda pelo processo de treino da rede. O uso da função de otimização correta também aprimora o processo de aprendizagem, ajudando a melhorar os modelos tornando-os mais precisos (Goodfellow et al., 2016).

Nesta secção são apresentadas algumas das principais funções de otimização, que foram consideradas no desenvolvimento deste projeto.

3.1.8.1 Gradient descent

Segundo *Goodfellow* (Goodfellow et al., 2016), o algoritmo de otimização *Gradient descent* permanece ainda como o algoritmo de otimização mais dominante para modelos de *Deep Learning*, hoje em dia.

Este algoritmo consiste na procura de um ponto estacionário, que representa a perda mínima possível de ser aplicada no processo de aprendizagem, tal como se pode observar na Figura 17. O objetivo é maximizar ou minimizar uma função, denominada de função objetivo. Quando está em processo de minimização, também se pode chamar de função de custo, função de perda ou função de erro.

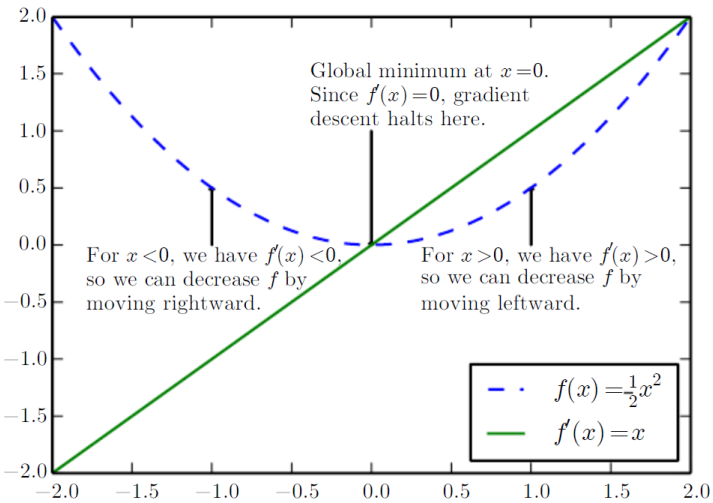

Figura 17 - Função de Otimização *Gradient descent* (Goodfellow et al., 2016).

3.1.8.2 Adam

Adam é um algoritmo de otimização que pode ser usado em vez do algoritmo Gradient descent, para calcular e atualizar os pesos da rede de forma iterativa com base nos dados de treino. O seu nome deriva da frase "*adaptative moments*".

Este algoritmo tem como principais vantagens, o facto de ser de simples configuração, onde os parâmetros de configuração padrão se saem bem na maioria dos problemas, é computacionalmente eficiente, tendo ainda muito poucos requisitos de memória. É adequado ainda a problemas com uma grande quantidade de dados e/ou parâmetros (Machine Learning Mistery, 2017).

Goodfellow (Goodfellow et al., 2016), refere que o algoritmo Adam é geralmente considerado como sendo bastante robusto para a escolha de híper-parâmetros, embora o parâmetro relativo ao *learning rate* às vezes precise ser diferente do padrão sugerido.

3.1.9 Medidas de Avaliação

Nesta secção são apresentadas algumas das medidas de avaliação mais comuns, especialmente em algoritmos de classificação na área dos ECG.

Para calcular as fórmulas referentes às medidas de avaliação é necessário a quantificação dos seguintes termos, tendo em conta o que o algoritmo classificador previu e o que realmente são as classificações corretas (Hossin et al., 2015).

- Verdadeiro Positivo (VP) - representa a quantidade de casos referentes à classe positiva que foram classificados corretamente. Relativamente ao tema do projeto, corresponde à quantidade de arritmias detetadas corretamente;
- Falso Positivo (FP) - representa a quantidade de casos referentes à classe negativa, mas que o algoritmo classificou como sendo da classe positiva, ou seja, corresponde à quantidade de arritmias que não foram detetadas pelo mesmo;
- Verdadeiro Negativo (VN) - representa a quantidade de casos referentes à classe negativa que foram classificados corretamente. Assim sendo, corresponde à quantidade de casos que foram detetados como não sendo arritmias corretamente;
- Falso Negativo (FN) - representa a quantidade de casos referentes à classe positiva, mas que o algoritmo classificou como sendo da classe negativa, ou seja, corresponde à quantidade de arritmias que foram detetadas pelo algoritmo incorretamente.

3.1.9.1 Matriz de Confusão

A matriz de confusão é utilizada em problemas de classificação binária (problema com duas classes possíveis de classificar), para encontrar a exatidão e precisão do modelo.

Esta é apresentada sob a forma de uma tabela com duas entradas (Tabela 14), uma constituída pelas classes desejadas e a outra pelas classes previstas pelo modelo. Já as células são preenchidas pelos valores que correspondem ao cruzamento das entradas.

A matriz de confusão é definida da seguinte forma:

Tabela 14 - Matriz de Confusão (Hossin et al., 2015)

	Positivos previstos	**Negativos previstos**
Positivos originais	VP	FN
Negativos originais	FP	VN

A matriz de Confusão em si não é uma medida de desempenho, no entanto grande parte das métricas de desempenho são baseadas na matriz de confusão e nos valores que esta contém (Hossin et al., 2015).

3.1.9.2 Sensibilidade

Sensibilidade, também conhecido por *Recall* ou Taxa de Verdadeiros Positivos (TVP), corresponde à taxa de acerto na classe positiva (Hossin et al., 2015). Esta medida de avaliação, relativamente ao tema do trabalho, corresponde à proporção de pacientes que realmente possuem arritmias cardíacas, tal como o algoritmo teve como previsão.

A fórmula aplicada para esta medida é a seguinte:

$$Sensibilidade = \frac{\mathrm{VP}}{VP + FN}$$

3.1.9.3 Especificidade

A medida de avaliação Especificidade, ou taxa de verdadeiros Negativos (TVN), corresponde à taxa de acerto na classe negativa. Esta medida quantifica a proporção de casos negativos que foram corretamente classificados, ou seja, indica-nos a proporção de pacientes que não possuem arritmias cardíacas, tal como o algoritmo previu (Hossin et al., 2015).

Assim sendo, pode-se determinar que a especificidade corresponde ao oposto da sensibilidade.

Para calcular esta medida aplica-se a seguinte fórmula:

$$Especificidade = \frac{\mathrm{VN}}{VN + FP}$$

3.1.9.4 Taxa de Falsos Positivos

A Taxa de Falsos Positivos (TFP), calcula a proporção de exemplos negativos que não foram classificados corretamente como positivos (Hossin et al., 2015). É definida com a seguinte fórmula:

$$TFP = \frac{\mathrm{FP}}{FP + VN} = 1 - Especificidade$$

3.1.9.5 Taxa de Falsos Negativos

A Taxa de Falsos Negativos (TFN), calcula a proporção de casos positivos que não foram classificados corretamente como negativos (Hossin et al., 2015). É definida com a seguinte fórmula:

$$TFP = \frac{\text{FN}}{VP + FN} = 1 - Sensibilidade$$

3.1.9.6 Taxa de acerto

Esta medida, do inglês *accuracy*, nos problemas de classificação, corresponde à percentagem de instâncias que o algoritmo previu corretamente, ou seja, a taxa de exemplos positivos e negativos corretamente classificados (Hossin et al., 2015).

Esta métrica é calculada com a seguinte fórmula:

$$Taxa\ de\ Acerto = \frac{\text{VP} + \text{VN}}{VN + FP + VP + FN}$$

3.1.9.7 Precisão

A precisão é uma medida de avaliação que nos indica qual a proporção de pacientes que foram diagnosticados como tendo arritmias, e que na verdade tiveram. Os pacientes que foram diagnosticados como tendo arritmias são os VP e FP, já os que sofrem realmente desse problema são os VP (Hossin et al., 2015).

Para avaliar o algoritmo com esta métrica, é utilizada a seguinte formula:

$$Precisão = \frac{\text{VP}}{VP + FP}$$

3.1.9.8 Area Under the Curve (AUC)

A curva ROC (*Receiver Operating Curve*) representa a sensibilidade (taxa de verdadeiros positivos) em função da taxa de falsos positivos. A área sob a curva ROC varia de 0 a 1, sendo que 1 representa um algoritmo classificador perfeito e 0 um algoritmo classificador que está sempre errado. Uma área ROC de 0,5 representa um algoritmo classificador que é aproximadamente aleatório. A taxa de acerto é muitas vezes preterida por esta área ROC quando são utilizados conjuntos de dados muito desequilibrados, uma vez que captura de forma mais eficaz o *tradeoff* entre verdadeiros positivos e verdadeiros negativos (Raeder T., 2008).

3.1.9.9 F-Measure

Esta medida de avaliação calcula a eficácia de um algoritmo classificador, combinando as medidas de avaliação precisão e sensibilidade. É possível ainda definir uma medida *F-Measure* que atribua pesos arbitrários para os valores de precisão ou sensibilidade. Esta medida

apresentada é conhecida como *F1-Measure*, pois dá igual importância a estas 2 métricas (Raeder T., 2008).

A fórmula para *F1-Measure* é a seguinte:

$$F = \frac{2 * \text{Precisão} * \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}}$$

3.2 Estado da Arte em soluções existentes

Neste subcapítulo é apresentado o estudo efetuado acerca de algumas abordagens já existentes para a deteção e previsão de arritmias cardíacas, referentes à análise deste trabalho.

Na Tabela 15 estão ilustradas, de forma resumida, algumas das abordagens estudadas. Mais concretamente, serão apresentados os pontos-chave de cada um dos estudos, desde o conjunto de dados utilizado para o caso em concreto, as metodologias utilizadas, como algoritmos de classificação, de avaliação, entre outros, e ainda as medidas de avaliação utilizadas pelos autores para avaliar as suas soluções.

Tabela 15 - Resumo das soluções existentes

Referências	***Dataset***	**Metodologias**	**Medidas de Avaliação**
(Isin et al., 2017)	6 registos da Base de dados ECG MIT-BIH (3 registos (normal, arritmia cardíaca e batimentos estimulados) de treino e 3 registos de teste).	Algoritmo *Pan-Tompkins* para deteção QRS; Métodos *transferred deep learning* para extração de atributos; Algoritmo *scaled conjugate gradient back-propagation* para classificação.	Taxa de reconhecimento; Taxa de acerto.
(Özbay et al., 2001)	Registos de 10 tipos de arritmias selecionados da Base de dados ECG MIT-BIH para treino; Registos de 17 pacientes para testes.	Algoritmo *scaled conjugate gradient back-propagation* para classificação.	Taxa de acerto.

Referências	*Dataset*	Metodologias	Medidas de Avaliação
(Kaya et al., 2015)	43 registos da Base de dados ECG MIT-BIH.	Métodos Principal *Component Analysis* (PCA), *Independent Component Analysis* (ICA), e *Self Organizing Maps* (SOM) para redução de atributos; Algoritmos *Neural Networks* (NN), *k-nearest neighbour* (k-NN), *Decision Trees* (DT) e SVM para classificação; Método de validação cruzada de 10-*folds*;	Taxa de acerto; Especificidade; Sensibilidade.
(Inan et al., 2006)	40 registos da Base de dados ECG MIT-BIH (18 para treino e 22 para teste).	Método *Wavelet Transform* para extração de atributos; Algoritmo NN para classificação.	Taxa de acerto; Sensibilidade.
(Ince et al, 2008)	40 registos da Base de Dados MIT-BIH.	Método PCA aplicado ao *Dyadic Wavelet Transform* (DWT) na extração de atributos; Algoritmo *Feed-Forward* ANN usando a técnica *Multi-Dimensional Particle Swarm Optimization* (MD-PSO) para a classificação.	Taxa de acerto; Sensibilidade; Precisão.
(Kelwade et al., 2015)	554 registos da Base de dados ECG MIT-BIH (388 de treino e 166 de teste).	Algoritmo *Pan-Tompkins* para deteção QRS; Métodos lineares (*Time domain analysis e Frequency domain analysis*) e métodos não-lineares (*Poincare plot geometry*, *Spectral entropy*, *Largest Lyapunov Exponent* (LLE) e *Detrended fluctuation analysis* (DFA)) para extração de atributos; Algoritmo *MultiLayer Perceptron* (MLP) *Neural Network* para classificação; Algoritmo *back-propagation* usado para comparar os resultados dos testes com os expectáveis.	Matriz de Confusão; Taxa de acerto.
(Zihlmann et al., 2017)	8528 registos ECG.	Pré-processamento dos dados aplicando uma transformação logarítmica; Usadas técnicas de *Data Augmentation* (*dropout bursts* e *random resampling*) e de *Ensembling*; Algoritmos CNN e *Convolutional Recurrent Neural Network* (CRNN) para classificação; Método de validação cruzada de 5-*folds*.	Taxa de acerto; F1-Measure.

Referências	*Dataset*	Metodologias	Medidas de Avaliação
(Ubeyli, 2008)	450 registos para treino e 450 para teste da Base de dados *Physiobank*	*Discrete Wavelet Transforms* para extração de atributos; Arquitetura *Mixture of experts* (ME) *Neural Network* recorrendo aos algoritmos *Expectation Maximization* (EM) para classificação.	Matriz de Confusão; Taxa de acerto; Sensibilidade; Especificidade; AUC.
(Luz E. et al., 2012)	44 registos da Base de dados ECG MIT-BIH (22 para treino e 22 para teste).	Usadas seis abordagens de extração de atributos, oriundas de trabalhos anteriores de classificação de arritmias de outros autores; Algoritmos *Optimum-Path Forest* (OPF), *Bayesian and Multilayer* ANN, SVM e MLP para classificação.	Taxa de acerto; Sensibilidade; Especificidade.
(Gomes et al., 2014)	312 auscultações do Real Hospital Português (Recife, Brasil)	Algoritmo *Multiresolution Motif Discovery (in Time Series)* para descobrir padrões; Algoritmo *Random Forest* para classificação; Método de validação cruzada de *10-folds.*	Taxa de Acerto.

Através deste resumo de algumas soluções existentes na Tabela 15, é possível notar que existem muitas possibilidades de abordagens, uma vez que todas elas são diferentes. Existem, no entanto, alguns pontos que são muito idênticos em grande parte destas abordagens, tal como a utilização de conjuntos de dados fornecidos pela base de dados ECG MIT-BIH por exemplo.

Segundo Eduardo Luz (Luz et al., 2016), esta base de dados é a mais representativa para o estudo deste problema, relativo às arritmias, sendo dessa forma usada para a maioria das pesquisas publicadas. Revela ainda que, esta base de dados foi a primeira a estar disponível para esse objetivo e tem sido constantemente refinado ao longo dos anos.

Relativamente aos algoritmos de classificação usados, existe também uma grande diversificação, no entanto, denota-se uma predominância sobre o SVM e as redes neurais, como a CNN por exemplo.

Quanto aos algoritmos, nota ainda para o uso do algoritmo classificador *Optimum-Path Forest* (OPF) no trabalho de Eduardo Luz (Luz E. et al., 2012), que segundo o próprio, foi utilizado pela primeira vez numa tarefa de classificação de sinais ECG, tendo sido comparado com outros três algoritmos de classificação muito utilizados (SVM, *Bayesian* e MLP). Este algoritmo obteu um resultado muito robusto e muito similar aos restantes, melhor até sobre alguns deles, em algumas métricas. Esta técnica revelou-se assim, muito promissora para a análise de sinal de ECG.

Relativamente às medidas de avaliação utilizadas pelos autores nos seus trabalhos, é de realçar o facto de em quase todos ter sido utilizada a taxa de acerto, reforçando a ideia de que esta é a medida mais importante para avaliar os resultados obtidos por um sistema de classificação. A sensibilidade e a especificidade foram também medidas muito utilizadas, mostrando que são também muito importantes neste contexto, uma vez que o número de batimentos cardíacos para cada classe poderia representar a maior parte da taxa de acerto total, enquanto estas medidas dependem diretamente do número de casos para cada classe.

Importante referir também que foram usadas bastantes metodologias, técnicas e ferramentas nas etapas de pré-processamento e de extração de atributos, destacando os métodos no domínio de *wavelet*, que foram bastante utilizados na extração de atributos.

Muitas destas metodologias e medidas de avaliação utilizadas, são apresentadas e detalhadas nas secções 3.1.3, 3.1.4, 3.1.5 e 3.1.6.

Na Tabela 16 é apresentado um resumo de algumas das abordagens relativamente às metodologias utilizadas pelas equipas no concurso lançado pela *PhysioNet* (*The PhysioNet in Cardiology Challenge 2017*). Para além das metodologias que utilizaram, também os seus resultados no concurso serão apresentados, através da medida de avaliação F1-*Measure*, que foi a medida utilizada no concurso para definir a classificação.

Tabela 16 - Metodologias utilizadas pelas equipas no concurso

Referências	Metodologias utilizadas	F1-*Measure*
(Xiong et al., 2017)	Desenvolveram 3 abordagens como algoritmos de classificação dos ECG: CNN de 16 camadas como principal e RNN e *Spectogram Learning* como alternativas para comparações; CNN: No treino, os dados foram inseridos na rede em lotes, tendo sido realizado o *batch-normalization* e aplicado o ReLU para acelerar o treino. Usaram o *Dropout* para reduzir o *overfitting* da CNN nos dados de treino antes da camada de convolução. Usaram também a função *softmax* para representar valores das saídas para cada classe como uma probabilidade. Usaram o *Adam Optimizer* para otimizar os parâmetros de rede. Utilizaram validação cruzada com 5 *folds*.	RNN - 0.72% Spect. Learn. - 0.78% CNN – 0.82%
(Smolen, 2017)	Desenvolveram um sistema que é composto por 2 modelos: RNN e *Gradient Boosting Machine* (GBM); Na extração de atributos foram projetados 36 atributos, que podem ser divididos em 5 categorias: *statistical features*, QRS *morphology features*, RR-*interval features*, *noise features* e *frequency-based features*; Utilizaram validação cruzada com 10 *folds*.	0.81%

Referências	Metodologias utilizadas	F1-*Measure*
(Zabihi et al., 2017)	Combinam atributos de vários domínios: tempo, frequência, tempo-frequência, espaço de fase e meta-nível. Utilizam uma abordagem de seleção de atributos baseada num algoritmo *Random Forest*. Usaram ainda outro algoritmo *Random Forest* como algoritmo de classificação. Utilizaram validação cruzada com 10 *folds*.	0.826%
(Zihlmann et al., 2017)	Desenvolveram 2 arquiteturas de redes neurais profundas para classificação: CNN (24 camadas) e CRNN (24 camadas CNN e 3 camadas LSTM (*Long Short-Term Memory*)). Ambas as arquiteturas consistem em quatro partes: 1 - Pré-Processamento dos dados aplicando uma transformação logarítmica; 2 - Pilha de camadas convolucionais para extração de atributos; 3 - Na CNN, agregação de atributos ao longo do tempo pela média, já na CRNN é através de um bloco LSTM; 4 - Aplicam um classificador linear com *softmax* para calcular as probabilidades de classe; Utilizaram o *Adam Optimizer* para otimizar os parâmetros de rede, e usaram ainda técnicas de *Data Augmentation* (*dropout bursts* e *random resampling*) e de *Ensembling*; Utilizaram validação cruzada com 5-*folds*.	CRNN: 0.82%
(Warrick et al., 2017)	Desenvolveram um sistema que combina uma CNN e 3 camadas de LSTM, com técnicas de *pooling*, *dropout* e de normalização, para melhorar a taxa de acerto. Aplicam a CNN para extrair atributos locais e discriminativos, seguidamente de três camadas de LSTM para codificar os padrões sequenciais. Adicionada depois uma camada densa para processar os dados de saída do terceiro LSTM. No fim, a função *softmax* é adotada para prever a classe. Utilizaram validação cruzada com 10-*folds*.	0.80%

Com este resumo das metodologias de algumas das equipas deste concurso, é possível perceber que todos optam por diferentes abordagens para o mesmo problema, apesar de usarem o mesmo conjunto de dados. Denota-se, ainda assim, uma ligeira superioridade no uso de algoritmos de classificação baseados em redes neurais artificiais, com uma maior incidência sobre as CNN.

A importância deste estudo é muito elevada, uma vez que poderá também servir para estabelecer comparações com a solução que será implementada neste trabalho, visto que o conjunto de dados ECG que será usado é o mesmo.

3.3 Estado da Arte em tecnologia

Neste subcapítulo é apresentado o estado da arte relativamente às tecnologias emergentes no desenvolvimento orientado ao *deep learning*. Mais concretamente será feito um estudo comparativo entre algumas tecnologias e respetivas *frameworks* para o desenvolvimento de aplicações nesta área, que permitem dar a conhecer as principais escolhas feitas para a implementação deste trabalho.

Com base no trabalho de *Kovalev* (Kovalev et al., 2016) é elaborado um estudo comparativo entre diferentes *frameworks* orientadas ao *deep learning*, relativas essencialmente às linguagens de programação Python e Java, que foram as consideradas para o desenvolvimento deste projeto.

3.3.1 Java vs Python

O **Java** é uma linguagem de programação desenvolvida por James Gosling, juntamente com a sua equipa, na década de 1990 (Java History, 2018).

Os programas Java são executados por outro programa chamado Java VM. Em vez de executar diretamente no sistema operativo nativo, o programa é interpretado pelo Java VM para o sistema operativo nativo. Isto significa que qualquer máquina que possua o Java VM instalado, poderá executar aplicações Java independentemente da máquina em que foram desenvolvidas (Java Essentials, 2018).

O **Python** trata-se de uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, funcional e de tipagem dinâmica e forte.

A filosofia do Python passa por dar uma maior ênfase à importância do esforço do programador sobre o esforço computacional. Dá uma maior prioridade à legibilidade do código em relação à velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos da sua biblioteca padrão e por módulos e *frameworks* desenvolvidos por terceiros (Python, 2018).

Segundo os estudos efetuados pela comunidade de programação Tiobe (Tiobe, 2018), é possível constatar que o Java é a linguagem de programação mais popular do mundo, com uma larga vantagem, apesar de que têm vindo a perder esta grande vantagem sobre as restantes ao longo do tempo. O Python, ao contrário do Java, tem vindo a ganhar força nos últimos anos, tendo até subido uma posição no ranking (subiu para a quarta posição por troca com C#) no último ano.

Na Figura 18 e Figura 19 é possível observar este ranking das linguagens de programação atualmente, e o histórico das principais linguagens de programação nos últimos anos, respetivamente.

Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2		C	11.857%	+3.41%
3	3		C++	5.726%	+0.30%
4	5	↑	Python	5.168%	+1.12%
5	4	↓	C#	4.453%	-0.45%
6	8	↑	Visual Basic .NET	4.072%	+1.25%
7	6	↓	PHP	3.420%	+0.35%
8	7	↓	JavaScript	3.165%	+0.29%
9	9		Delphi/Object Pascal	2.589%	+0.11%
10	11	↑	Ruby	2.534%	+0.38%
11	-	⇈	SQL	2.356%	+2.36%
12	16	⇈	Visual Basic	2.177%	+0.30%
13	15	↑	R	2.086%	+0.16%
14	18	⇈	PL/SQL	1.877%	+0.33%
15	13	↓	Assembly language	1.833%	-0.27%
16	12	⇊	Swift	1.794%	-0.33%
17	10	⇊	Perl	1.759%	-0.41%
18	14	⇊	Go	1.417%	-0.69%
19	17	↓	MATLAB	1.228%	-0.49%
20	19	↓	Objective-C	1.130%	-0.41%

Figura 18 - Ranking de Popularidade das Linguagens de Programação (Tiobe 2018)

Programming Language	2018	2013	2008	2003	1998	1993	1988
Java	1	2	1	1	18	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	5
Python	4	7	6	12	26	16	-
C#	5	5	7	9	-	-	-
Visual Basic .NET	6	13	-	-	-	-	-
JavaScript	7	9	8	7	21	-	-
PHP	8	6	4	5	-	-	-
Perl	9	8	5	4	3	11	-
Ruby	10	10	9	19	-	-	-
Objective-C	18	3	45	47	-	-	-
Ada	28	15	17	14	7	7	2
Lisp	31	12	14	13	6	4	3
Pascal	136	14	19	97	11	3	13

Figura 19 - Histórico do Ranking de Popularidade das Linguagens de Programação (Tiobe 2018)

3.3.2 Frameworks Deep Learning

Nesta secção é apresentado o estudo comparativo elaborado por Kovalev (Kovalev et al., 2016) sobre algumas das *frameworks* orientadas ao *deep learning* mais populares, entre elas o Theano, Torch, Caffe, TensorFlow e Deeplearning4J.

Para efetuar este estudo comparativo entre as *frameworks*, foram efetuados diversos testes utilizando redes neurais com variação de profundidade e largura, sendo que os principais pontos de comparação foram a velocidade de convergência das redes neurais profundas na fase de treino, assim como a precisão da classificação das redes treinadas no estágio de previsão. Mais especificamente, as métricas quantitativas que foram tidas em conta nestes testes foram:

- Tempo de convergência;
- Tempo de previsão (classificação);
- Taxa de acerto da classificação;
- Tamanho do código fonte, que é necessário para a descrição do algoritmo de teste, medido como número de linhas de código-fonte.

Estes testes de redes neurais foram efetuados usando duas funções de ativação diferentes:

- *Hyperbolic tangent* (Tanh);
- *Rectified linear unit function* (ReLU).

Importante referir ainda que, *Kovalev* definiu o número de épocas a 10 em todas as experiências (*Epoch*, como pode ver nas Figura 20, Figura 21 e Figura 22).

O número de épocas é um híper-parâmetro que define o número de vezes que o algoritmo trabalhará sobre todo o conjunto de dados de treino, ou seja, uma época consiste num ciclo completo de treino no conjunto de treino.

Nas Figura 20, Figura 21 e Figura 22 são exibidos os resultados de comparação das cinco *frameworks* pelo tempo de treino das redes neurais, o tempo de previsão e a taxa de acerto da classificação.

Em cada uma destas figuras estão os resultados obtidos, usando as duas funções de ativação mencionadas, em cada gráfico.

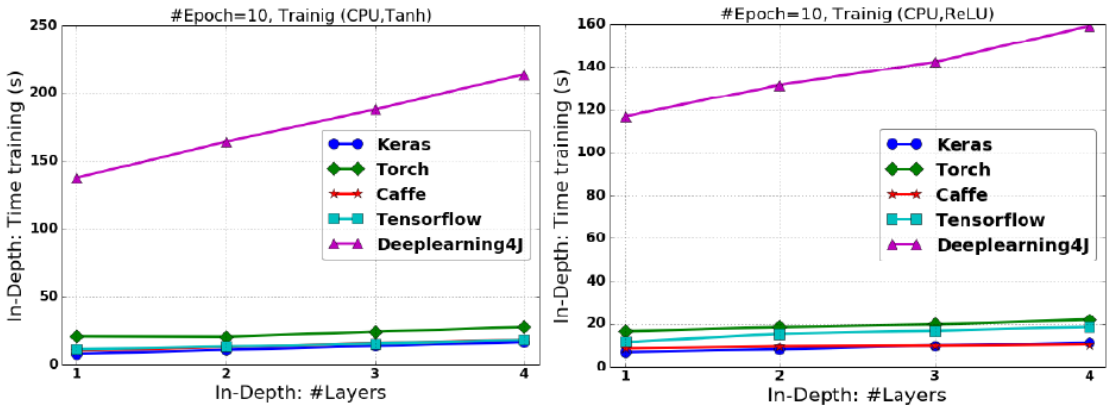

Figura 20 - Tempo de treino das redes neurais "em profundidade" usando as funções Tanh e ReLU respetivamente (Kovalev et al., 2016)

Comparando os resultados do tempo de treino apresentados na Figura 20, é possível inferir imediatamente que os padrões de mudanças de tempo de treino representadas pelas formas das curvas são muito semelhantes. No entanto, os valores absolutos do tempo de treino usando a função de ativação ReLU são muito menores (ou seja, melhores) comparando o caso da função de Tanh.

É ainda possível observar que todas as *frameworks*, exceto a Deeplearning4J, são muito semelhantes relativamente ao tempo de treino das redes neurais. A *framework* DeepLearning4j apresenta um tempo de treino bem superior que as restantes *frameworks*.

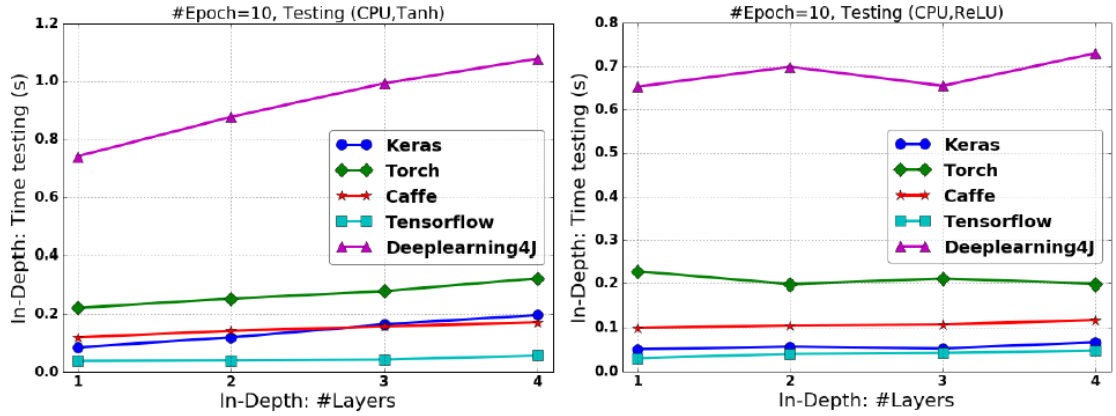

Figura 21 - Tempo de previsão das redes neurais "em profundidade" usando as funções Tanh e ReLU respetivamente (Kovalev et al., 2016)

Em contraste com os resultados obtidos relativamente ao tempo de treino das redes, apesar de alguma oscilação no tempo de previsão observado no uso da função ReLU, em geral, são muito semelhantes aos da função Tanh.

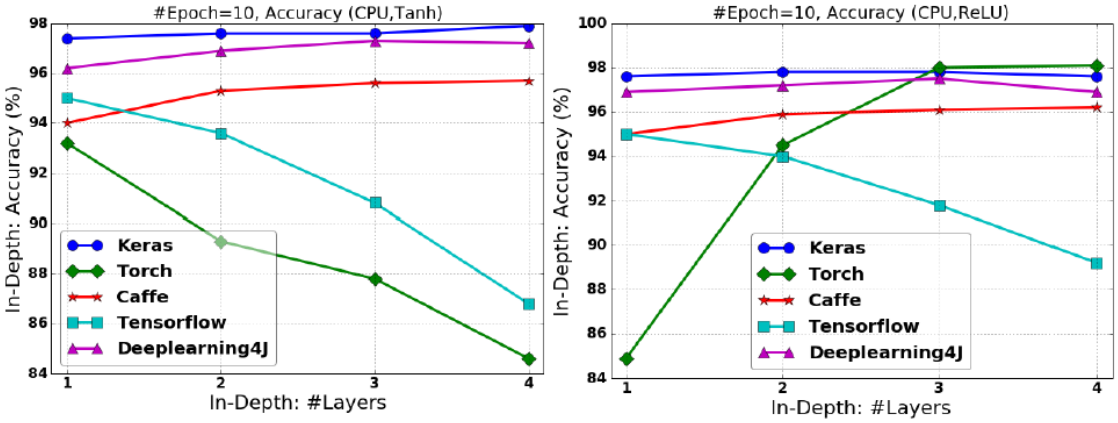

Figura 22 - Taxa de acerto das redes neurais "em profundidade" usando as funções Tanh e ReLU respetivamente (Kovalev et al., 2016)

Relativamente ao ponto de comparação da taxa de acerto fornecida pelas redes com as funções de Tanh e ReLU, apresentado na Figura 22, é possível constatar que a *framework* Torch mudou o seu comportamento nas diferentes funções, sendo que a sua taxa de acerto de classificação

cresce usando a função ReLU ao contrário da função Tanh. As outras *frameworks* não alteram significativamente.

Foram ainda efetuadas mais algumas experiências com redes neurais "em largura" usando a função Relu, uma vez que esta permitiu obter melhores resultados. Nestas experiências foi possível notar que a *framework* Deeplearning4j teve um tempo de treino muito mais elevado que as restantes *frameworks*, tal como se tinha já sucedido nas redes neurais "em profundidade". O mesmo se passou com o tempo de previsão, tendo sido muito idênticos os resultados obtidos com os dois tipos de redes. Quanto à taxa de acerto da classificação, foi possível observar que todas estas *frameworks* obtiveram valores bem elevados (cerca de 97%) para todos os tamanhos de camada interna examinados, sendo que no caso da *framework* Torch mostrou um crescimento claro de taxa de acerto com o crescimento do tamanho da camada.

A complexidade das *frameworks* foi medida em linhas de código fonte necessárias para implementar o algoritmo correspondente.

É apresentada de seguida uma ilustração (Figura 23) onde apresenta o número de linhas de código usadas por cada algoritmo de cada *framework*, sendo possível observar desde logo, que as *frameworks* Theano e Tensorflow, da linguagem Python foram as que utilizaram menos linhas de código.

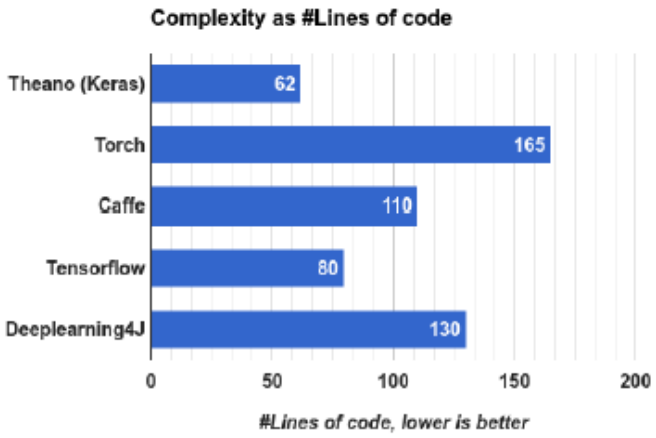

Figura 23 - Complexidade das *frameworks* em linhas de código (Kovalev et al., 2016)

Considerando todos os resultados obtidos nos diversos testes realizados, para cada uma das *frameworks*, Kovalev elaborou um ranking com as seguintes classificações (Kovalev et al., 2016):

Tabela 17 - Ranking de *frameworks*, segundo Kovalev (Kovalev et al., 2016)

Ranking	***Framework***	**Linguagem de Programação**
1	Theano	Python
2	Tensorflow	Python
3	Caffe	Python
4	Torch	Lua
5	DeepLearning4J	Java

Este ranking vai de encontro às expetativas, uma vez que as *frameworks* que apresentaram melhores resultados aos testes efetuados pertencem à linguagem de programação Python, que possui uma maior comunidade de desenvolvimento na área de *deep learning*. As duas *frameworks* que mais se destacaram foram a Theano e o Tensorflow, sendo que ambas conseguem ter resultados muito similares no desempenho do processamento de dados nos diferentes casos de teste realizados no estudo referido (Kovalev et al., 2016).

Desta forma, com a contribuição deste estudo comparativo, ficou decidida a utilização da linguagem de programação Python e da *framework* Tensorflow para o desenvolvimento deste projeto. O facto de a *framework* Tensorflow ser bem mais recente que todas as outras *frameworks* e de ter sido desenvolvida por uma grande empresa no mundo tecnológico, a Google, contribuiu bastante para esta decisão, para além dos resultados obtidos pela *framework* (Tensorflow, 2018).

4 Design

Neste capítulo será apresentado o *design* da solução, começando por uma análise ao conjunto de dados que será utilizado no desenvolvimento da mesma. De seguida será apresentada a abordagem considerada para solucionar o problema apresentado. Ainda neste capítulo, é possível visualizar a implementação da solução proposta que contém os detalhes sobre cada um dos procedimentos efetuados com o objetivo de resolver o problema.

4.1 *Dataset*

Para o desenvolvimento deste projeto, será utilizado um *dataset* disponibilizado num concurso (PhysioNet, 2017). Estes dados foram gravados usando o dispositivo AliveCor e doados a este desafio. As gravações foram realizadas por pessoas que adquiriram o dispositivo ECG monocanal a esta entidade, AliveCor (Clifford, 2017).

O *dataset* é composto por um total de 12.186 gravações ECG, dos quais 8.528 seriam utilizados para efeitos de treino e 3.658 para teste. No entanto, as gravações correspondentes ao conjunto de testes não estão disponíveis ao público, tendo como propósito a pontuação dada aos participantes, durante o período do desafio. Desta forma, serão utilizadas neste projeto apenas as gravações fornecidas para o treino dos modelos de classificação. Estas gravações possuem um intervalo de 9 a 61 segundos de comprimento de tempo, aproximadamente, tendo sido realizadas a partir de 2 elétrodos, um em cada mão (pelo dispositivo AliveCor) a 300Hz (Clifford, 2017).

Na Tabela 18 são exibidos os detalhes do *dataset* de treino, sendo que estas gravações foram classificadas em quatro categorias (ritmo normal, AF, outro ritmo e gravações com ruído). É exposta a quantidade de registos existentes por categoria, aliada a algumas métricas relativas ao comprimento de tempo dos mesmos, tais como a média e a mediana do comprimento de

tempo de uma gravação, o desvio padrão e ainda os comprimentos máximos e mínimos existentes em cada uma das categorias. É importante referir ainda que a classificação destes dados ECG foi efetuada manualmente por uma equipa de especialistas, tendo sido realizada uma série de revisões aos mesmos. Não é, no entanto, garantida uma taxa de exatidão de 100% da classificação dos mesmos (Clifford, 2017).

Tabela 18 - Detalhes do *Dataset* de treino (PhysioNet, 2017)

Categoria	**Registos**	**Comprimentos de Tempo**				
		Média	**Desvio Padrão**	**Máximo**	**Mediana**	**Mínimo**
Normal	5076	31.9	10.0	61.0	30	9.0
AF	758	31.6	12.5	60.0	30	10.0
Outro	2415	34.1	11.8	60.9	30	9.1
Com ruído	279	27.1	9.0	60	30	10.2
Total	8528	32.5	10.9	61	30	9.0

Os dados relativos às gravações ECG estão armazenados em ficheiros .mat, estando estes acompanhados por um ficheiro .hea, que contém a informação sobre as características do sinal. Na Figura 24 são apresentados exemplos das formas de onda do sinal ECG, com duração de 20 segundos, para as quatro categorias.

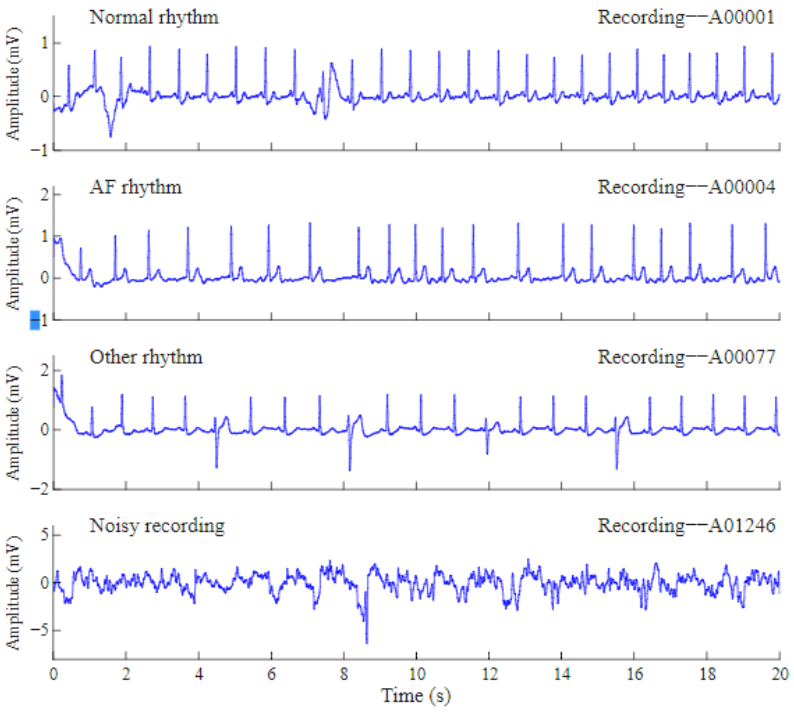

Figura 24 - Exemplos de formas de onda ECG (PhysioNet, 2017)

4.2 Abordagem

Para a resolução do problema apresentado, são expostas duas abordagens de metodologias possíveis, para a análise e classificação de sinais ECG.

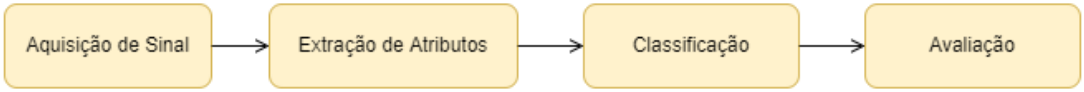

Figura 25 - Metodologias de Desenvolvimento - Abordagem 1

Figura 26 - Metodologias de Desenvolvimento - Abordagem 2

Na primeira abordagem, representada na Figura 25, é exibida a metodologia mais habitual, composta pelas seguintes etapas:

- **Aquisição de Sinal** - o sinal é adquirido, sendo que neste ponto poderá vir a ser necessário efetuar algum pré-processamento dos dados ainda, como aplicação de filtros para reduzir possíveis ruídos, por exemplo;
- **Extração de Atributos** - extração de atributos do *dataset* e consequente criação de um novo conjunto de dados, composto pelos atributos extraídos;
- **Classificação** - utilizar algoritmos de classificação, tais como *Random Forests* ou SVM (secção 3.1.5);
- **Avaliação** – obter os resultados da classificação dos sinais de ECG, e avaliar esses resultados, segundo algumas medidas de avaliação, tais como as que estão enumeradas na secção 3.1.9.

Na segunda abordagem, representada na Figura 26, é apresentada uma metodologia *deep learning*, usando redes neurais convolucionais (ou CNN).

Tal como na abordagem 1, esta possui a etapa de aquisição de sinal, onde poderá ser necessário efetuar algum pré-processamento dos dados também e ainda a etapa relativa à avaliação, onde se obtêm os resultados da classificação dos sinais, avaliando-os de seguida.

A camada de deteção de atributos da CNN aprende implicitamente através de dados de treino, evitando desta forma a extração de atributos e classificação de forma explícita. Ou seja, usando CNN é possível mapear diretamente o sinal de entrada para obter os resultados (Liu T, 2015).

Como já foi referido anteriormente, a segunda abordagem representa a metodologia que será seguida para o desenvolvimento da resolução do problema, sendo utilizado o *TensorFlow* para a criação das CNN.

4.2.1 Processo de Desenvolvimento

Na Figura 27 é apresentado o diagrama de atividades que representa de uma forma geral o processo de desenvolvimento da solução.

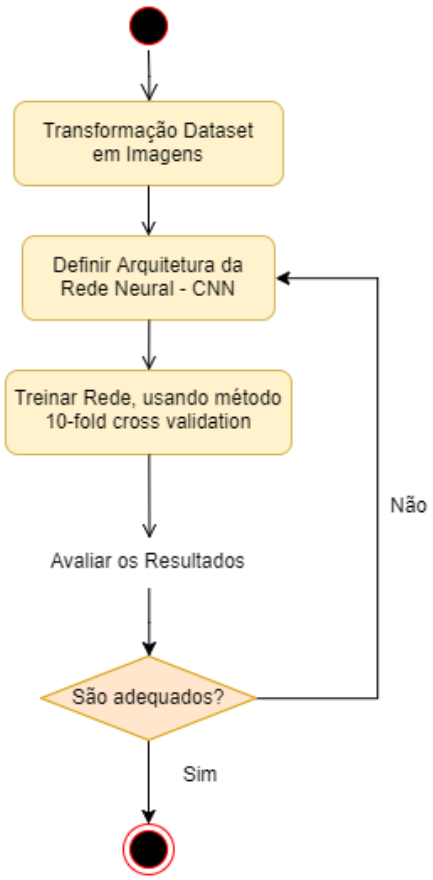

Figura 27 - Processo de Desenvolvimento da Solução

O processo de desenvolvimento inicia-se com a transformação do *dataset*, que está disponibilizado em formato .mat, para imagens, uma vez que as ferramentas de trabalho propostas não trabalham diretamente com esse tipo de formatação.

Posteriormente, é definida a arquitetura e os parâmetros necessários da rede neuronal CNN, para tratar os dados ECG, convertidos anteriormente para imagens, utilizando o *Tensorflow* e Python.

Com a rede neuronal construída, são realizados agora testes na mesma, com as imagens obtidas anteriormente. Estes testes serão efetuados através da aplicação do método de validação cruzada de 10 *folds* ao *dataset* de imagens. Este método consiste em dividir o conjunto de dados de entrada em 10 subconjuntos do mesmo tamanho (9 conjuntos para treino e um conjunto para teste) e construir 10 submodelos. Sequencialmente, um subconjunto será avaliado usando o algoritmo de classificação treinado nos outros 9 subconjuntos, sendo que o conjunto de teste usado será diferente em cada iteração, não estando também presente nos dados de treino

dessa iteração. Desta forma, cada subconjunto do conjunto de dados completo será previsto uma vez. A taxa de acerto média destas 10 iterações será calculada como o resultado da classificação. O mesmo se aplica nas restantes medidas de classificação. A técnica de validação cruzada pode ajudar a evitar o problema do *over-fitting*. (Japkowicz et al., 2011).

Por fim, é necessário efetuar uma análise e avaliação dos resultados obtidos por este modelo de classificação.

4.3 Implementação

Neste subcapítulo serão apresentados e detalhados os procedimentos efetuados, de forma sequencial, para a implementação do *design* sugerido anteriormente.

Durante a fase de implementação, diferentes abordagens foram seguidas, tanto no pré-processamento dos dados, como na definição da arquitetura do modelo, sempre com o objetivo de melhorar os resultados. Na descrição técnica da implementação serão apresentadas estas abordagens, de um modo geral, sendo que será dado um maior detalhe sobre o modelo final implementado.

4.3.1 Pré-Processamento

Conforme referido na secção 4.1 deste documento, os *datasets* utilizados para a realização deste projeto foram disponibilizados num concurso (PhysioNet, 2017). Como foi mencionado também, apenas estavam disponíveis 8.528 registos para o desenvolvimento deste projeto, uma vez que os restantes 3.658 registos eram privados do concurso, para efeitos de validação de resultados.

Estes registos estão armazenados em ficheiros .mat, existindo ainda um ficheiro "REFERENCE-v3.csv", fornecido também pelo concurso, juntamente com os registos, onde está o mapeamento dos registos com as respetivas classes associadas.

A CNN bidimensional requer imagens como dados de entrada. Portanto, na primeira etapa, a de pré-processamento, foi efetuada a transformação dos conjuntos de dados (sinais de ECG) em imagens. Foram seguidas duas abordagens, cujo resultado final (as imagens geradas), se revelou bastante diferente entre ambas. Ambas as soluções foram implementadas usando a linguagem Python, recorrendo também a bibliotecas externas, que serão apresentadas na descrição das abordagens.

4.3.1.1 Transformação dos Datasets em Imagens - Abordagem 1

Na primeira abordagem, foram utilizadas as bibliotecas externas numpy [4] e *scipy* [5] para manipulação dos dados (sinais ECG) e gravação dos mesmos em imagens, respetivamente. A execução desta solução incluiu todos os registos (8528 registos), transformando-os em imagens do tipo .bmp. Tal como foi mencionado na secção 4.1 deste documento, os registos não são todos do mesmo tamanho, possuindo um intervalo de 9 a 61 segundos de comprimento de tempo, aproximadamente. Por cada segmento de 1 segundo, estão contemplados cerca de 300 sinais, perfazendo assim um intervalo de 2.700 a 18.300 sinais entre os diversos registos. Nesta transformação, cada sinal origina um pixel na imagem, que se diferencia pela cor consoante o seu valor (usando *bytescale*[6]). Desta forma, a execução desta solução deu origem à geração de imagens com intervalos de resolução desde os 2700x1 até aos 18.300x1, em escala de cinza.

Na Figura 28, é apresentado um exemplo de uma imagem, correspondente ao registo A00001, sendo um registo categorizado com a classe "Ritmo Normal". Esta imagem possuí um tamanho 9000x1, que indica que este registo corresponde a uma gravação de 30 segundos.

Figura 28 - Abordagem 1: Registo A00001 convertido em imagem A00001.bmp

Olhando para esta imagem, é possível verificar que é uma tarefa árdua, ou mesmo impossível para o ser humano, a identificação da categoria que pertence este registo ECG apenas observando a imagem, ao contrário do que acontece com as imagens na Figura 24, que são bem mais legíveis.

No início do processo de treino, ainda fazendo parte do pré-processamento, o algoritmo abre as imagens .bmp, lê as mesmas e extrai uma parte (96x96 bytes), criando novas imagens de 96x96 de resolução, com a parte extraída. Estas novas imagens são depois adicionadas a uma lista de imagens que posteriormente servirá para alimentar a rede neuronal, que está detalhada na secção 4.3.2.1. Esta resolução de 96x96 foi definida de modo a permitir que sejam extraídas partes iguais em todas as imagens, incluindo as mais reduzidas (2700x1).

4.3.1.2 Transformação dos Datasets em Imagens - Abordagem 2

A segunda abordagem surgiu da necessidade de melhorar os resultados do modelo que usava as imagens concebidas na Transformação dos Datasets em Imagens - Abordagem 1. Uma vez que se tratavam de imagens com resoluções variáveis e com grande discrepância entre a altura e largura, apenas uma parte de cada imagem era usada pela rede neuronal. Desta forma, o risco de usar uma parte da imagem desinteressante, tendo em conta a sua categoria, era grande. O principal objetivo seria, portanto, procurar uma forma alternativa de representar os sinais numa imagem, e de preferência que estas fossem usadas na sua totalidade pela rede.

[4] http://www.numpy.org/
[5] https://www.scipy.org/
[6] https://docs.scipy.org/doc/scipy/reference/generated/scipy.misc.bytescale.html#scipy.misc.bytescale

Após um estudo sobre o assunto, e implementação de alguns testes, chegou-se a uma alternativa, bastante viável. Foram utilizadas as bibliotecas externas Matplotlib[7] para a criação de gráficos a partir dos *arrays* de valores (sinais) e o OpenCV[8] para criar os ficheiros de imagem do tipo .png, com os gráficos. Como, em sinais de ECG, as cores não têm qualquer importância, foi efetuada a conversão para imagens na escala de cinza, tal como tinha sido feito também na primeira abordagem.

A resolução destas imagens podia neste caso ser configurada, tendo sido definida a resolução 512x512. Desta forma, foram obtidas 8528 imagens bidimensionais, com a mesma resolução, correspondentes aos registos ECG disponíveis.

Na Figura 29, é apresentada a imagem correspondente ao mesmo exemplo (A00001) da Figura 28.

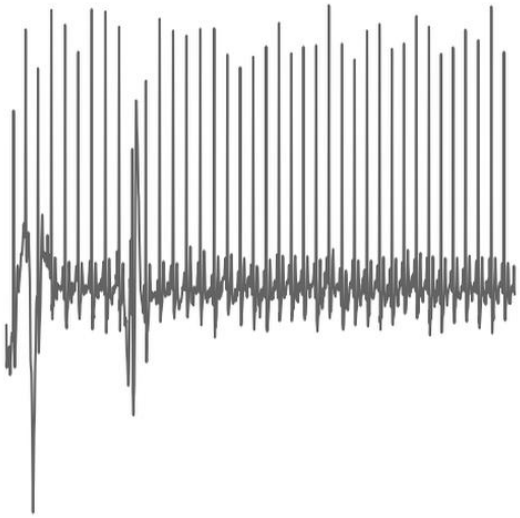

Figura 29 - Abordagem 2: Registo A00001 convertido em imagem A00001.png

Ao contrário do que acontece com as imagens geradas na primeira abordagem, as imagens resultantes desta segunda abordagem são muito mais legíveis ao ser humano, permitindo até, em alguns casos identificar a categoria associada ao registo ECG, através da análise da imagem.

Segundo (Jun et al., 2018), ao transformar os sinais de ECG unidimensionais em imagens de ECG bidimensionais, a filtragem de ruído e a extração de atributos (ou características) deixam de ser necessárias, o que é muito importante, pois algumas das batidas de ECG são por vezes ignoradas nessas etapas. Para além disso, os dados de treino podem ser ampliados, aumentando as imagens de ECG, o que resulta numa maior taxa de acerto na classificação. O aumento de dados é difícil de ser aplicado em sinais unidimensionais, pois a distorção do sinal unidimensional de ECG pode degradar o desempenho do classificador.

[7] https://matplotlib.org/
[8] https://opencv.org/

No princípio do processo de treino, ainda fazendo parte do pré-processamento, o algoritmo percorre sobre as imagens geradas anteriormente (512x512), e efetua o redimensionamento das mesmas para um tamanho mais reduzido, com resolução de 128x128. Estas novas imagens são adicionadas a uma lista de imagens que ficam guardadas em memória, sendo posteriormente submetidas na rede neuronal, que as usará para efetuar o seu processo de aprendizagem. Esta resolução 128x128 foi definida de acordo com a configuração implementada na rede neuronal, que está detalhada na secção 4.3.2.2.

4.3.1.3 Balanceamento dos dados

A distribuição do conjunto de dados, que compreende um total de 8528 gravações ECG, pelas quatro categorias (ritmo normal, AF, outro ritmo e dados ruidosos) é largamente desequilibrada (desbalanceada), tal como se pode verificar na secção 4.1, e mais detalhadamente na Tabela 18. Grande parte dos registos pertencem à classe do ritmo normal (cerca de 60%), 9% são registos de pacientes com arritmias (AF), 28% registos com outros ritmos, e os restantes 3% são registos com ruído.

Na secção 3.1.3.4 são apresentadas algumas técnicas para transformar e consolidar os dados do *dataset*, sendo que algumas delas seriam bastante úteis no caso de os conjuntos de dados não serem balanceados, como é o caso do *dataset* utilizado neste trabalho.

Uma das técnicas apresentadas trata-se da técnica de *Dataset Augmentation*, que seria útil para aumentar os registos das classes que possuem menos exemplos, como é o caso dos registos com ritmo ruidoso e com AF. No entanto, não foi possível a implementação desta técnica, uma vez que, por limitações de *hardware*, não seria possível executar um treino da rede com todos os registos disponíveis no modelo implementado. O aumento de registos das classes com menos exemplos revelava-se desta forma desnecessário, pois os mesmos acabariam por não ser utilizados na sua totalidade de qualquer forma.

Em contrapartida, foi usada a estratégia de redimensionamento do conjunto de dados, também ela descrita na secção 3.1.3.4, sendo uma estratégia enquadrada nas técnicas de balanceamento de dados.

Foi definido um parâmetro "*records_per_class*" no modelo, que deve ser passado na execução do mesmo, definindo a quantidade de registos de cada classe que devem ser tidos em conta no treino da rede, e ignorando os restantes. Este parâmetro é usado na fase de obter as imagens que farão parte da entrada da CNN, não permitindo que haja um número de registos por classe superior ao mesmo. Por exemplo, ao definir o parâmetro com o valor 100, a execução do modelo será efetuada com no máximo 400 registos de treino, que correspondem aos 100 exemplos de cada uma das quatro classes. Esta medida foi usada para determinar a possibilidade de se distinguirem as classes num cenário de equilíbrio. Embora se trate de um problema diferente do original, permite aferir até que ponto uma falta de resultados se deve ao desbalanceamento das classes.

Desta forma, é possível garantir que os dados estão 100% balanceados, com a mesma quantidade de registos de cada classe.

4.3.1.4 Seleção de imagens por segmentos

Como foi referido na secção 4.3.1.1, os registos (do *dataset*) não são todos do mesmo tamanho, havendo registos com um intervalo de 9 a 61 segundos de comprimento de tempo, aproximadamente. No entanto, as imagens foram todas geradas com o mesmo tamanho, 512x512 de resolução. Uma imagem que corresponde a um registo de 60 segundos tem o dobro de informação (sinais) que uma imagem que corresponde a um registo de 30 segundos, no entanto, ambas as imagens terão a mesma resolução. É possível antever que a imagem relativa ao registo de 60 segundos ficará mais distorcida que a do registo de 30 segundos.

Na Figura 30, são exibidas três imagens, que correspondem a três registos com diferentes comprimentos de tempo (10, 30 e 60 segundos).

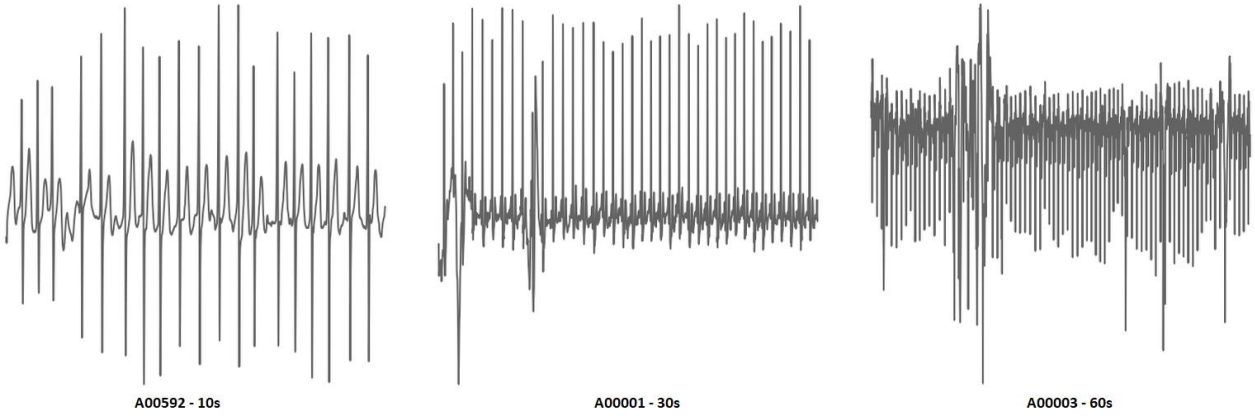

Figura 30 - Três registos com diferentes intervalos de tempo

Como é possível observar, a primeira imagem (correspondente ao registo A00592) está muito mais nítida que a segunda (registo A00001), que por sua vez está mais nítida que a terceira imagem (registo A00003). Isto deve-se aos comprimentos de tempo que os registos possuem, e consequentemente a quantidade de sinais que cada registo possui.

Foi então efetuado um estudo sobre os registos existentes, de forma a obter estatísticas sobre os comprimentos de tempo dos mesmos, tal como se pode observar na Tabela 19.

Tabela 19 - Estatísticas dos registos (Comprimentos de Tempo)

Segmentos (Comprimentos de Tempo)	Quantidade de Registos	Percentagem
10s ~ 19s	616	7.22%
19s ~ 29s	351	4.12%
30s	5977	70.09%
31s ~ 39s	450	5.27%
40s ~ 49s	161	1.89%
50s ~ 59s	125	1.47%
60s	809	9.49%
> 60s	39	0.45%
Total	8528	100%

Com estas métricas, foi possível retirar informação relevante para resolver o problema do fornecimento da rede com imagens referentes a registos com diferentes comprimentos de tempo.

Foi definido um método de seleção de imagens por segmentos, ou seja, durante a fase de seleção das imagens que farão parte da entrada da CNN, apenas poderão entrar imagens do mesmo segmento (mesmos comprimentos de tempo).

Olhando para a quantidade de registos de cada segmento, na Tabela 19, é possível verificar que grande parte dos registos (cerca de 70%) correspondem a gravações de 30 segundos. Este segmento de registos, demonstra ser um ótimo conjunto de dados para o treino da rede.

4.3.1.5 Divisão de imagens por segmentos

Com a utilização da técnica de seleção de imagens por segmentos (ver secção 4.3.1.4), e após algumas experiências efetuadas nos modelos utilizando essa técnica de pré-processamento, surgiu a ideia de implementar um método que dividisse as imagens em pequenas janelas de tempo (tendo sido definido 10s como janela de tempo).

A implementação desta ação de pré-processamento, pode oferecer grandes vantagens ao processo de deteção de arritmias, tais como:

- **Equidade nos dados**: o processo de aprendizagem de um modelo é alimentado apenas com imagens com a mesma janela de tempo;
- **Qualidade nos dados**: o processo de aprendizagem de um modelo é alimentado com imagens mais nítidas, podendo levar a uma maior capacidade de extração de atributos por parte do algoritmo;

- ***Data-Augmentation***: Esta técnica de divisão de imagens por segmentos, faz também parte da técnica de *data-augmentation* (referida na secção 4.3.1.3). Usando esta técnica, um registo de 30 segundos dá origem a 3 imagens (de 10 segundos).

Na Figura 31, é apresentado um exemplo desta divisão de imagem por segmentos. Aparece a imagem original (que corresponde ao registo A00001) com o comprimento de tempo original, de 30 segundos. Em baixo, aparecem 3 imagens, que são o resultado da aplicação desta técnica, que correspondem aos segmentos de 10 segundos desse mesmo registo.

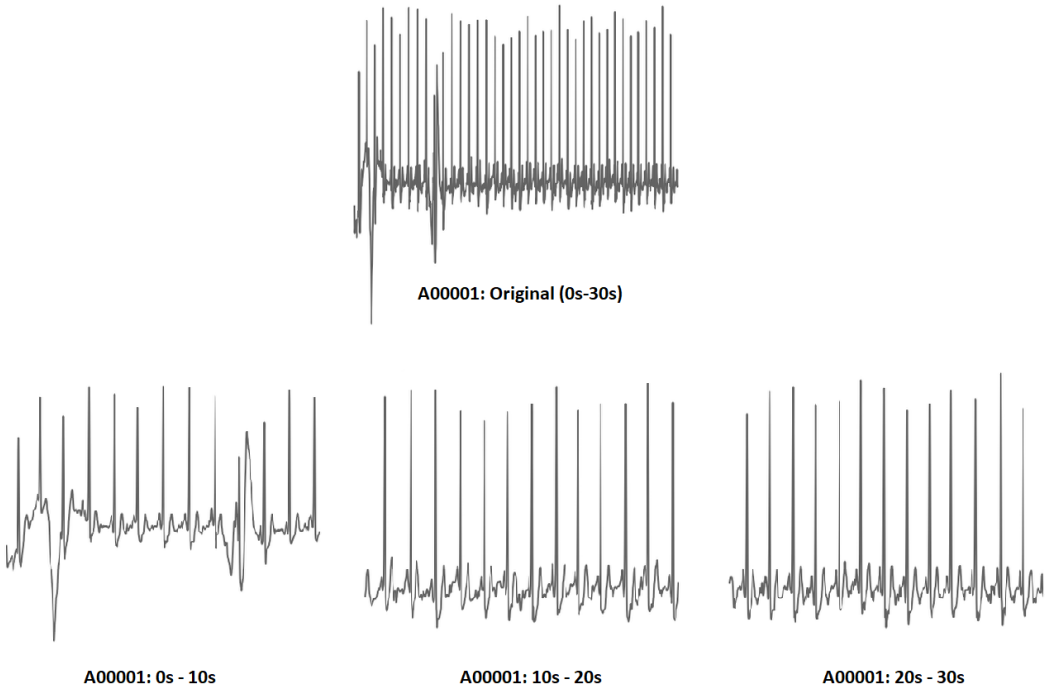

Figura 31 – Divisão de Imagem em segmentos de 10 segundos

A aplicação desta técnica de pré-processamento não traz apenas vantagens, havendo também possíveis desvantagens. Uma delas é a possibilidade de perda de informação, que pode ser importante para a classificação de um registo numa determinada classe. Por exemplo, um registo ECG de 30 segundos pode estar classificado com a classe “Fibrilação Atrial”, uma vez que o ritmo cardíaco está irregular dos 10 aos 30 segundos, no entanto, os primeiros 10 segundos o ritmo aparenta estar normal. Ao processo de aprendizagem de modelo, esta imagem, referente aos primeiros 10 segundos do registo, será indicada como sendo um exemplo com “Fibrilação Atrial”.

4.3.2 Arquitetura do Modelo

Na secção 4.2 foram apresentadas as abordagens iniciais de metodologias que foram estudadas para o desenvolvimento deste projeto, tendo sido selecionada a segunda abordagem, que usava a metodologia *deep learning*, através de CNN.

Tal como na fase de pré-processamento, também na implementação do modelo foram seguidas duas abordagens, ambas utilizando a mesma metodologia (*deep learning*, usando CNN), sendo utilizado o *TensorFlow* como *framework* de implementação em auxílio da linguagem Python.

Em cada uma destas abordagens, foram implementados dois modelos, com a mesma arquitetura. O primeiro modelo é adaptado ao problema descrito neste documento, para classificação multiclasse (4 classes). O segundo modelo está adaptado a um problema de classificação binária, ou seja, um problema de duas classes apenas (com arritmia (AF) e sem arritmia (ritmo normal, outro ritmo e ritmo com ruído em conjunto)). A decisão da criação desta alternativa em cada abordagem, surgiu para simplificar o processo de aprendizagem do modelo (passando de 4 para 2 classes), e desta forma conseguir melhorar os resultados de deteção de arritmias cardíacas, sendo que esse é o problema que este trabalho pretende ajudar a resolver.

Importante referir, que não foram usadas arquiteturas já conhecidas, como por exemplo a *AlexNet*, tal como fez Alex Krizhevsky num dos seus projetos (Krizhevsky et al., 2012). As arquiteturas dos modelos foram desenhadas e implementadas de raiz, tendo em conta o conjunto de dados existente, assim como os recursos de *hardware* disponíveis para a execução do treino do modelo.

Nesta secção serão descritos os procedimentos efetuados, assim como as especificações técnicas das arquiteturas dos modelos implementados, nas diferentes abordagens.

4.3.2.1 Abordagem 1 – Modelos para 2 e 4 classes

A primeira abordagem na implementação da rede neuronal, teve em consideração o conjunto de imagens geradas na abordagem 1 (.bmp) do pré-processamento, descrita na secção 4.3.1.1.

Neste ponto, quando o processo de treino do modelo é iniciado, já possui a lista de imagens geradas (96x96) a partir das imagens originais (que possuem tamanhos desde os 2700x1 até aos 18.300x1)

A Figura 32 mostra a arquitetura geral do modelo CNN implementado nesta primeira abordagem.

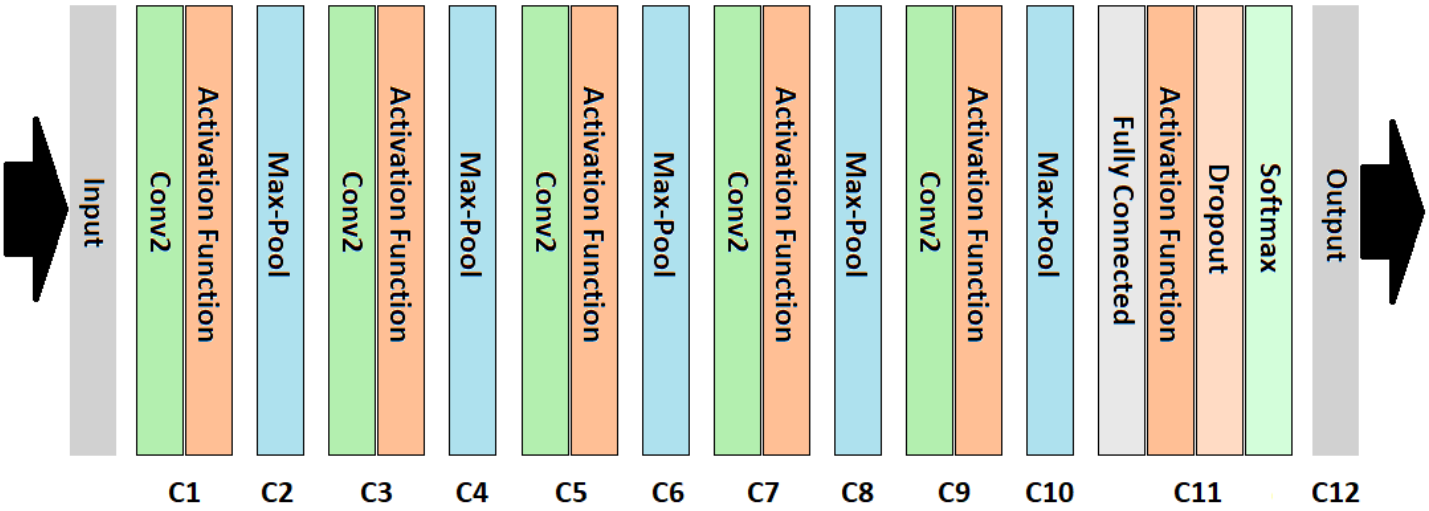

Figura 32 - Arquitetura da rede neuronal - Abordagem 1

A arquitetura da rede neuronal é composta por 12 camadas: 5 camadas convolucionais (C1, C3, C5, C7 e C9), onde é sempre executada a função de ativação após a convolução. Após cada camada convolucional, segue-se uma camada de *pooling* (*max-pooling*), perfazendo um total de 5 camadas também (C2, C4, C6, C8 e C10).

A penúltima camada (C11) é totalmente conectada, sendo responsável por efetuar a classificação das características obtidas após as camadas convolucionais. Esta classificação é seguida do *dropout*, que ajuda a evitar o *over-fitting* dos dados de treino, sendo depois submetida ao algoritmo *softmax,* que é responsável por produzir a distribuição sobre as várias classes. É importante referir ainda que a rede neuronal utiliza uma função de otimização (função de treino), de forma a atualizar os parâmetros da rede neuronal e assim minimizar a taxa de perda.

Como é possível observar na Figura 32, não está definida qual a função de ativação que está a ser usada no modelo. Isto deve-se ao facto de que a mesma não está definida de forma estática no modelo, sendo parametrizável na fase de execução do mesmo. Ou seja, no momento de iniciar a execução do treino do modelo, a função de ativação que se pretende utilizar é passada como argumento, sendo que o modelo está preparado para receber uma das quatro funções descritas na secção 3.1.7 (*tanh*, *relu*, *sigmoid* ou *softplus*).

O mesmo acontece para a função de otimização, que é um argumento também submetido na execução do modelo. O modelo está preparado apenas também para as funções de otimização descritas na secção 3.1.8 (*Gradient Descent* ou *Adam*).

A Tabela 20 descreve a arquitetura detalhada do modelo CNN, que está representado na Figura 32.

Tabela 20 - Arquitetura detalhada da rede neuronal – Abordagem 1

Camada	Tipo	Tamanho *Kernel*	*Stride*	*Kernel*	Tamanho *Input*
C1	*Conv2D*	5 x 5	1	32	96 x 96 x 1
C2	*Max-Pool*	2 x 2	2		96 x 96 x 32
C3	*Conv2D*	5 x 5	1	64	48 x 48 x 32
C4	*Max-Pool*	2 x 2	2		48 x 48 x 64
C5	*Conv2D*	5 x 5	1	128	24 x 24 x 128
C6	*Max-Pool*	2 x 2	2		24 x 24 x 128
C7	*Conv2D*	5 x 5	1	256	12 x 12 x 256
C8	*Max-Pool*	2 x 2	2		12 x 12 x 256
C9	*Conv2D*	5 x 5	1	512	6 x 6 x 512
C10	*Max-Pool*	2 x 2	2		6 x 6 x 512
C11	*Full-Connected*			1024	3 x 3 x 1024
C12	*Output*			4 / 2*	1024

* 4 para o modelo de 4 classes, e 2 para o modelo binário.

Resumidamente, as camadas convolucionais utilizam filtros de 5x5 (Tamanho *Kernel*) para calcular as características (*Kernel* representa a quantidade de características), com um *stride* de 1 pixel, já nas camadas de *max-pooling*, o filtro utilizado na operação de *pooling* tem tamanho 2x2, com um *stride* de 2 pixéis.

4.3.2.2 Abordagem 2 – Modelos para 2 e 4 classes

Tal como foi mencionado na secção 4.3.1.2, a implementação da segunda abordagem surgiu da necessidade de melhorar os resultados apresentados pela primeira abordagem implementada. Para isso, foram geradas as novas imagens (com resolução 512x512), efetuado o seu pré-processamento, tal como está explicado nessa secção e ainda implementada uma nova rede neuronal, que será descrita nesta secção.

Neste ponto, quando o processo de treino do modelo é iniciado, já possui a lista de imagens redimensionadas (112x112) a partir das imagens originais (512x512).

A Figura 33 mostra a arquitetura geral do modelo CNN implementado nesta primeira abordagem.

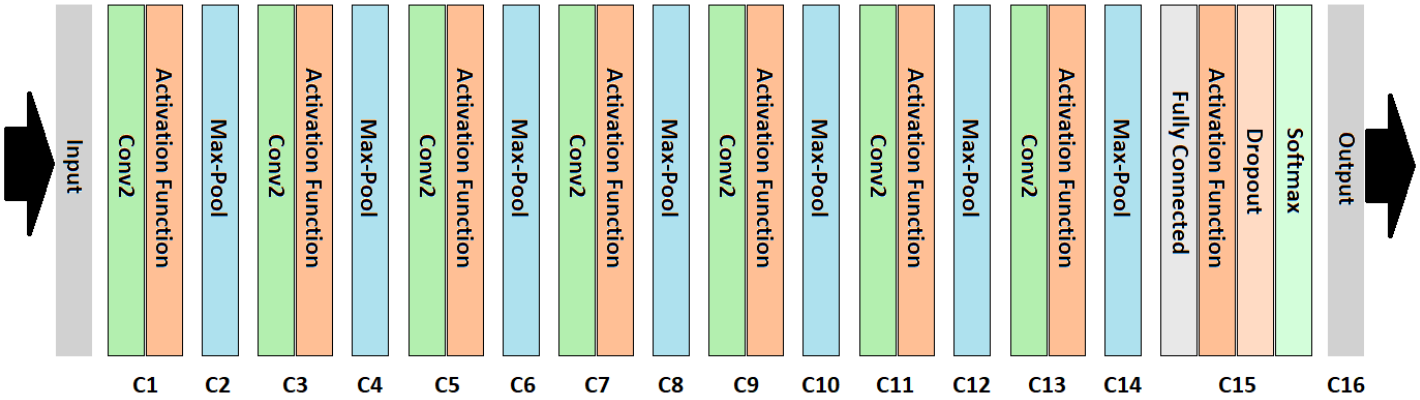

Figura 33 - Arquitetura da rede neuronal - Abordagem 2

A arquitetura desta rede neuronal é composta por 16 camadas: 7 camadas convolucionais (C1, C3, C5, C7, C9, C11 e C13), onde é também sempre executada a função de ativação após a convolução. Depois de cada camada convolucional vêm uma camada de *max-pooling*, perfazendo um total de 7 camadas também (C2, C4, C6, C8, C10, C12 e C14).

As camadas convolucionais utilizam filtros de 5x5 para calcular as características, com um *stride* de 1 pixel. Nas camadas de *max-pooling*, o filtro utilizado na operação de *pooling* tem tamanho 2x2, com um *stride* de 2 pixéis. Estes detalhes podem ser vistos na Tabela 21.

A penúltima camada (C15) é totalmente conectada, sendo responsável por efetuar a classificação das características obtidas após as camadas convolucionais. Esta classificação é seguida do *dropout*, sendo depois submetida ao algoritmo *softmax,* responsável por produzir a distribuição sobre as várias classes. A rede neuronal utiliza ainda uma função de otimização, de forma a atualizar os parâmetros da rede neuronal e assim minimizar a taxa de perda.

Tal como na primeira abordagem, não estão definidas as funções de ativação e de otimização de forma estática no modelo, sendo estas submetidas como parâmetros no início da execução da aprendizagem do modelo. O modelo está preparado para receber uma das quatro funções de ativação descritas na secção 3.1.7 (*tanh*, *relu*, *sigmoid* ou *softplus*) assim como uma das funções de otimização descritas na secção 3.1.8 (*Gradient Descent* ou *Adam*).

A Tabela 21 descreve a arquitetura detalhada do modelo CNN, que está representado na Figura 33.

Tabela 21 - Arquitetura detalhada da rede neuronal – Abordagem 2

Camada	Tipo	Tamanho *Kernel*	*Stride*	*Kernel*	Tamanho *Input*
C1	*Conv2D*	5 x 5	1	32	128 x 128 x 1
C2	*Max-Pool*	2 x 2	2		128 x 128 x 32
C3	*Conv2D*	5 x 5	1	64	64 x 64 x 32
C4	*Max-Pool*	2 x 2	2		64 x 64 x 64
C5	*Conv2D*	5 x 5	1	128	32 x 32 x 128
C6	*Max-Pool*	2 x 2	2		32 x 32 x 128
C7	*Conv2D*	5 x 5	1	256	16 x 16 x 256
C8	*Max-Pool*	2 x 2	2		16 x 16 x 256
C9	*Conv2D*	5 x 5	1	512	8 x 8 x 512
C10	*Max-Pool*	2 x 2	2		8 x 8 x 512
C11	*Conv2D*	5 x 5	1		4 X 4 X 1024
C12	*Max-Pool*	2 x 2	2		4 X 4 X 1024
C13	*Conv2D*	5 x 5	1		2 X 2 X 2048
C14	*Max-Pool*	2 x 2	2		2 X 2 X 2048
C15	*Full-Connected*			4096	1 x 1 x 4096
C16	*Output*			4 / 2*	4096

* 4 para o modelo de 4 classes, e 2 para o modelo binário.

Como é possível constatar, as configurações das redes neurais referentes às duas abordagens não são completamente díspares, apresentando dissemelhanças essencialmente a quantidade de camadas, a entrada de dados (tamanho das imagens de entrada) e a quantidade de características que a rede irá extrair em cada imagem (*Kernel*).

4.3.3 Configuração de validação de resultados

Para a obtenção dos resultados dos modelos implementados, foram realizados testes, com base no método de validação cruzada (*cross-validation*) de 10 *folds*. Em cada uma das 10 combinações possíveis, 9 *folds* são usados como dados de treino e 1 como dados de teste. Será considerado como resultado, o valor médio obtido destas 10 combinações, para as diferentes medidas de avaliação do modelo.

A configuração de validação utilizada está representada na Figura 34.

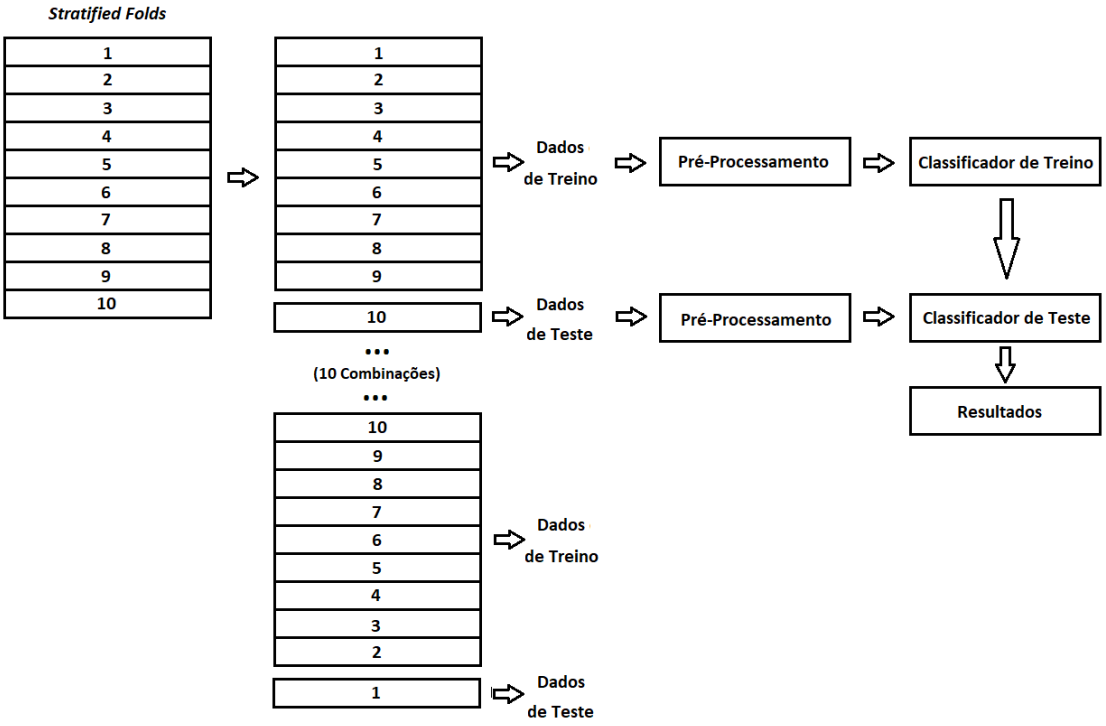

Figura 34 - Esboço da configuração de validação de resultados

Para a avaliação dos resultados obtidos em cada um dos modelos de classificação, são usadas algumas das medidas descritas na secção 3.1.9.

Para além da taxa de acerto, serão usadas como medidas de avaliação o AUC, *F-Measure*, sendo ainda apresentada a matriz confusão nos resultados dos modelos de 2 classes (classificação binária). A utilização da medida *F-Measure* servirá essencialmente para efeitos de comparação com os resultados dos participantes do concurso (PhysioNet, 2017), visto que foi esta a medida de avaliação utilizada no concurso, para definir as pontuações dos modelos. A medida AUC será utilizada essencialmente porque é uma medida muito eficaz.

No capítulo 5, é possível observar os resultados obtidos para cada um dos modelos implementados, tendo em conta as diversas funções de ativação e de otimização. De igual forma, é apresentada ainda uma comparação dos resultados obtidos com os resultados dos participantes do concurso.

5 Experiências e avaliação

Neste capítulo são identificadas as medidas utilizadas na avaliação das experiências efetuadas nos vários modelos de classificação implementados, tal como as hipóteses e as metodologias de avaliação utilizadas nas mesmas. Ademais são apresentados os resultados de algumas destas experiências, através de comparações entre as várias abordagens seguidas, de forma a ser possível selecionar os modelos que apresentam melhores resultados, oferecendo um maior grau de confiança para a resolução do problema proposto. Para culminar, são efetuadas comparações dos melhores resultados obtidos com os resultados de outras soluções, implementadas pelos participantes do concurso (PhysioNet, 2017).

5.1 Medidas de avaliação

A representação de resultados para problemas com duas classes tem na matriz confusão, uma medida de avaliação simples de aplicação. Por norma, esta apresenta os valores dos elementos correta e incorretamente classificados, tanto na classe positiva como na negativa. Esta representação é utilizada neste trabalho, na aplicação dos algoritmos de classificação de duas classes, correspondendo a classe positiva à classe com problema (AF) e a classe negativa à classe sem problema (restantes classes – ritmo normal, outro ritmo e com ruído).

Após a análise dos trabalhos de outros autores de trabalhos nesta área, é possível verificar que a medida de avaliação mais utilizada é a taxa de acerto (*accuracy*), seguida da sensibilidade e especificidade. Como tal, também esta solução será avaliada usando a taxa de acerto em todos os modelos implementados.

Serão ainda utilizadas as medidas *F-Measure* e *Area Under the ROC Curve* (AUC), servindo a primeira essencialmente para efetuar comparações com os resultados dos participantes do concurso (PhysioNet, 2017), visto esta ter sido a medida de avaliação utilizada no concurso.

5.2 Hipóteses e Metodologias de avaliação

A hipótese nula afirma que os desempenhos dos modelos são idênticos, ou seja, as médias (ou medianas, dependendo do tipo de teste paramétrico ou não paramétrico) de uma dada medida de avaliação são iguais. A hipótese nula atua então contra a hipótese alternativa, na qual o modelo de classificação apresentado é melhor do que os outros modelos.

A decisão sobre o melhor modelo de classificação apresentado será efetuada através da comparação entre as diferentes abordagens implementadas usando o algoritmo *Convolutional Neural Networks*, com diferentes configurações/parâmetros. Estas comparações serão feitas sobre as diferentes medidas de avaliação mencionadas na secção anterior.

Relativamente às experiências a efetuar, o conjunto de dados será dividido em 10 subconjuntos do mesmo tamanho, sendo que um subconjunto será usado para ser avaliado pelos algoritmos de classificação, previamente treinados pelos outros nove subconjuntos. Basicamente, será usado um método de validação cruzada com a utilização de 10 subconjuntos (*10-fold cross-validation*) para o treino e teste dos algoritmos de classificação.

5.3 Avaliação dos resultados

Nesta secção serão apresentados os resultados de algumas experiências efetuadas nos diversos modelos de classificação implementados, para algumas das medidas de avaliação mencionadas na secção 5.1. No total, foram implementados 4 modelos, referentes à combinação das duas abordagens de arquiteturas da rede neuronal com os dois tipos de classificação (binária e multiclasse).

Os resultados são apresentados sob a forma de tabelas, sendo que em cada linha da tabela estarão os resultados alusivos a uma determinada experiência. Esta tabela está subdividida entre 3 a 4 colunas, apresentando-se na primeira a quantidade de exemplos (assim como a quantidade referente a cada classe) utilizados na experiência. Na segunda coluna está a terminologia das medidas de avaliação utilizadas. A terceira e quarta coluna (quando existe), referem-se às funções de otimização utilizadas na experiência. As colunas correspondentes às funções de otimização estão subdivididas em 4 colunas, referentes a cada uma das funções de ativação usadas nas experiências. Desta forma, é possível apresentar os resultados obtidos pelos modelos, nas diferentes métricas definidas, com as várias combinações entre as funções de ativação e otimização. As medidas de avaliação apresentadas nas diversas tabelas são as seguintes:

- **Taxa de Acerto:** percentagem de exemplos que o modelo previu corretamente, ou seja, a taxa de exemplos positivos e negativos corretamente classificados;
- **AUC:** Média da taxa de verdadeiros positivos em função da taxa de falsos positivos, em todas as classes, ou seja, é calculado o AUC de cada classe como classe positiva, e todas as outras como classe negativa, sequencialmente, sendo efetuada a média até À obtenção do resultado final de AUC;

- **AVG *F-Measure*:** Média dos resultados da medida *F-Measure* para todas as classes;
- ***F-Measure* AF:** *F-Measure* da classe positiva, ou seja, o resultado desta medida na classe Fibrilação Atrial (arritmia). É destacada esta medida para esta classe, uma vez que o problema que esta solução pretende resolver, passa pela deteção de arritmias, portanto, o sucesso da classificação de arritmias é desta forma o ponto-chave da solução.

5.3.1 1ª abordagem - Modelo de classificação multiclasse

Na Tabela 22, são apresentados os resultados obtidos pelo modelo referente à 1ª abordagem de arquitetura da rede neuronal para a classificação multiclasse (4 classes), tendo sido este o 1º modelo implementado e a apresentar resultados aceitáveis nas experiências. Foram efetuadas diversas experiências com diferentes quantidades de dados, sempre balanceados pelas 4 classes, usando a estratégia descrita na secção 4.3.1.3. Foi definido um padrão com diferentes quantidades de exemplos a usar por classe (200, 400, 800 e 1200 exemplos) nas experiências, por forma a ser possível efetuar comparações entre os resultados deste modelo com os demais, implementados posteriormente.

Estas experiências foram realizadas utilizando a combinação de todas as funções de ativação mencionadas na secção 353.1.7 com as funções de otimização descritas na secção 3.1.8. É importante referir ainda que estas foram realizadas com diferentes parâmetros e configurações da rede, de forma a obter os melhores resultados possíveis. Um destes parâmetros foi o número de épocas, ou seja, o número de vezes que o modelo trabalha sobre o conjunto de dados de treino, tendo o modelo sido testado com 1, 5 e 10 épocas. Os resultados do modelo com 10 épocas foram os que obtiveram melhor resultado, razão pela qual são os que estão reproduzidos na Tabela 22Tabela 24. No Anexo I são apresentados os resultados das experiências deste modelo com 1 e 5 épocas.

Tabela 22 - Resultados da 1ª abordagem para classificação multiclasse - 10 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	**0.895**	0.735	0.385	0.695	0.440	0.250	0.280	0.310
	AUC	**0.930**	0.823	0.590	0.797	0.627	0.500	0.520	0.540
	AVG *F-Measure*	**0.892**	0.715	0.317	0.674	0.421	0.100	0.214	0.204
	F-Measure AF	**0.867**	0.714	0.356	0.741	0.377	0.400	0.197	0.134
400 (100 cada classe)	**Taxa de Acerto**	**0.848**	0.670	0.358	0.768	0.298	0.255	0.258	0.320
	AUC	**0.898**	0.780	0.572	0.845	0.532	0.503	0.505	0.547
	AVG *F-Measure*	**0.844**	0.661	0.296	0.761	0.268	0.110	0.185	0.224
	F-Measure AF	**0.858**	0.631	0.402	0.788	0.310	0.402	0.275	0.319
800 (200 cada classe)	**Taxa de Acerto**	0.709	0.653	0.304	**0.733**	0.348	0.250	0.273	0.314
	AUC	0.806	0.768	0.536	**0.822**	0.565	0.500	0.515	0.543
	AVG *F-Measure*	0.696	0.650	0.233	**0.726**	0.290	0.100	0.166	0.200
	F-Measure AF	0.702	0.666	0.309	**0.767**	0.319	0.400	0.217	0.297
1200 (300 cada classe)	**Taxa de Acerto**	0.438	**0.675**	0.265	0.600	0.329	0.237	0.265	0.284
	AUC	0.626	**0.782**	0.507	0.732	0.557	0.500	0.507	0.529
	AVG *F-Measure*	0.399	**0.668**	0.161	0.590	0.260	0.097	0.136	0.171
	F-Measure AF	0.501	**0.705**	0.399	0.652	0.391	0.007	0.412	0.245

Na Tabela 22, apresentam-se os resultados obtidos seguindo a primeira abordagem descrita. Pode observar-se que os melhores resultados foram conseguidos de forma partilhada entre várias funções de ativação (*Tanh*, *Softplus* e *Relu*), com a função de otimização (treino) *Adam Optimizer*, sendo que estas variações ocorreram com o aumento de exemplos do conjunto de dados.

Nas experiências com 200 e 400 exemplos (50 e 100 de cada classe, respetivamente), o modelo obteve melhores resultados, e bastante satisfatórios (acima de 0.85% em todas as medidas), com a utilização da função de ativação tangente hiperbólica (*Tanh)*, juntamente com o *Adam Optimizer*. Nas experiências seguintes, com conjuntos de dados mais alargados (800 e 1200 registos), os resultados foram piorando com o aumento de dados. Em contrapartida, as funções *Softplus* e *Relu* revelaram uma ligeira melhoria de resultados com o aumento de dados, apresentando desta forma os resultados mais satisfatórios nas experiências com 800 e 1200 exemplos, respetivamente.

Por sua vez, as experiências efetuadas com a função de otimização *Gradient Descent Optimizer*, apresentam resultados menos satisfatórios. É presumível que o parâmetro relativo ao *learning rate* não esteja com o valor mais apropriado, uma vez que esta função de treino mantém o mesmo valor deste parâmetro em todas as atualizações dos pesos, ao contrário da função *Adam Optimizer*, que o vai adaptando.

É possível observar ainda que, em geral, os resultados da medida *F-Measure* AF apresentados são melhores que os resultados da média desta medida nas 4 classes (AVG *F-Measure*). Observa-se assim, que há um maior sucesso na classificação de arritmias em comparação à média de classificação das várias classes, sendo um bom ponto de partida para uma boa solução para o problema.

Na Figura 35 pode visualizar-se um gráfico que apresenta a comparação da variação da média da *F-Measure* das 4 classes, nas diversas funções de ativação (usando a função de otimização *Adam Optimizer*), com o aumento da quantidade de exemplos utilizados nas experiências a este modelo.

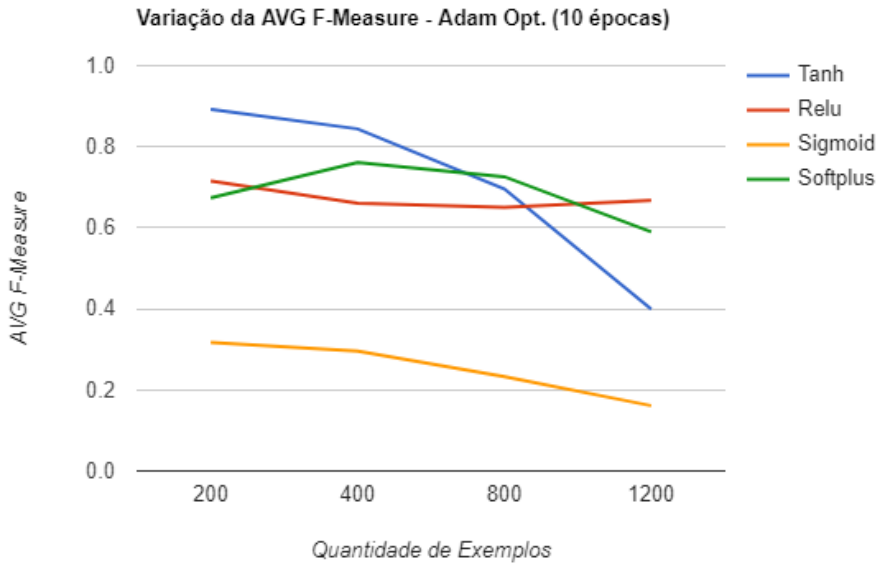

Figura 35 - Variação da média da *F-Measure* com aumento do *dataset* - 1ª Abordagem para classificação multiclasse

Analisando a Figura 35, é de fácil perceção que os resultados (AVG *F-Measure*) pioram, com o aumento da quantidade de exemplos usados para treino do modelo (usando qualquer uma das funções de ativação). Esta degradação de resultados pode acontecer pelo facto de ser usada sempre a mesma ordem de exemplos no treino da rede, sendo que os exemplos mais simples de classificar são os primeiros a serem usados. Desta forma, ao aumentar o conjunto de dados, podem surgir exemplos mais difíceis de classificar, levando à degradação dos resultados.

A função *Relu*, entre as funções de ativação é a que consegue manter um maior equilíbrio nos resultados das várias experiências, tendo inclusive melhorado os resultados entre as experiências com 800 e 1200 exemplos, tendo sido entre estas duas experiências onde houve um maior declínio de resultados nas restantes funções de ativação.

5.3.2 1ª abordagem - Modelo de classificação binária

Na Tabela 23, são apresentados os resultados obtidos pelo modelo referente à 1ª abordagem de arquitetura da rede neuronal para a classificação binária (com e sem arritmia), tendo sido este o 1º modelo de classificação binária implementado, com resultados muitos satisfatórios nas experiências efetuadas. Foram efetuadas diversas experiências com diferentes quantidades de exemplos, sempre balanceando as 2 classes, usando a estratégia descrita na secção 4.3.1.3. Foi também definido um padrão com diferentes quantidades de exemplos a usar por classe (100, 200, 400 e 600 exemplos) nas experiências, por forma a ser possível efetuar comparações entre os resultados deste modelo com os restantes modelos de classificação binária, implementados posteriormente.

Estas experiências foram de igual modo efetuadas utilizando a combinação de todas as funções de ativação descritas neste documento (ver secção 353.1.7) com as funções de otimização (ver secção 3.1.8). Tal como nas experiências efetuadas ao modelo multiclasse, também neste caso o modelo foi testado com 1, 5 e 10 épocas, sendo que também aqui os resultados com 10 épocas foram os que obtiveram melhor resultado, como seria de esperar. No Anexo II são apresentados os resultados das experiências deste modelo com 1 e 5 épocas.

Tabela 23 - Resultados da 1ª abordagem para classificação binária - 10 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	**0.970**	0.860	0.540	0.840	0.780	0.500	0.550	0.500
	AUC	**0.970**	0.860	0.540	0.840	0.780	0.500	0.550	0.500
	AVG *F-Measure*	**0.970**	0.855	0.455	0.828	0.776	0.333	0.452	0.333
	F-Measure AF	**0.973**	0.840	0.519	0.800	0.765	0.667	0.660	0.667
200 (100 cada classe)	**Taxa de Acerto**	**0.955**	0.840	0.580	0.815	0.585	0.500	0.525	0.510
	AUC	**0.955**	0.840	0.580	0.815	0.585	0.500	0.525	0.510
	AVG *F-Measure*	**0.953**	0.837	0.500	0.796	0.568	0.333	0.467	0.354
	F-Measure AF	**0.962**	0.835	0.504	0.792	0.616	0.667	0.631	0.671
400 (200 cada classe)	**Taxa de Acerto**	**0.883**	0.828	0.505	0.823	0.518	0.500	0.498	0.500
	AUC	**0.883**	0.828	0.505	0.823	0.518	0.500	0.498	0.500
	AVG *F-Measure*	**0.881**	0.821	0.349	0.816	0.466	0.333	0.352	0.333
	F-Measure AF	**0.885**	0.844	0.536	0.838	0.630	0.667	0.659	0.667
600 (300 cada classe)	**Taxa de Acerto**	0.822	**0.852**	0.502	0.832	0.567	0.500	0.498	0.500
	AUC	0.822	**0.852**	0.502	0.832	0.567	0.500	0.498	0.500
	AVG *F-Measure*	0.818	**0.851**	0.357	0.818	0.522	0.342	0.338	0.333
	F-Measure AF	0.833	0.851	0.475	**0.857**	0.665	0.664	0.664	0.667
1380 (690 cada classe)	**Taxa de Acerto**	0.559	0.796	0.500	**0.846**	0.505	0.500	0.500	0.500
	AUC	0.559	0.796	0.500	**0.846**	0.505	0.500	0.500	0.500
	AVG *F-Measure*	0.522	0.781	0.333	**0.839**	0.348	0.333	0.333	0.333
	F-Measure AF	0.569	0.827	0.467	**0.867**	0.668	0.667	0.667	0.667

Na Tabela 23, apresentam-se os resultados obtidos seguindo a primeira abordagem de classificação binária descrita. Do mesmo modo como no modelo de classificação multiclasse desta abordagem de arquitetura de rede neuronal, os melhores resultados nas diversas medidas de avaliação, para este modelo de classificação binário, foram obtidos usando a combinação das funções de ativação *Tanh*, *Relu* e *Softplus*, com a função de otimização *Adam Optimizer*, sendo que estas variações ocorreram com o aumento de exemplos do conjunto de dados.

Nas experiências com 100, 200 e 400 exemplos (50, 100 e 200 de cada classe, respetivamente), o modelo obteve melhores resultados, e muito bons (acima de 0.88% em todas as medidas), com a utilização da função de ativação tangente hiperbólica (*Tanh)*, juntamente com o *Adam Optimizer*. Nas experiências com 100 e 200 exemplos, os resultados ultrapassaram os 0.95% de taxa de acerto, bem como nas restantes medidas utilizadas. Na experiência seguinte, com 600 exemplos (300 de cada classe), os resultados foram muito semelhantes na utilização das 3 funções de ativação (entre os 0.82 e 0.85% em todas as medidas, aproximadamente), mas com uma ligeira vantagem na utilização das funções *Relu* e *Softplus*. Desta forma, seria difícil

escolher uma função de ativação, para destacar o melhor modelo de classificação binário desta abordagem. Para isso, foi efetuada mais uma experiência, com uma maior quantidade de exemplos utilizados (1380 exemplos, 690 de cada classe). Analisando os resultados desta experiência, é possível observar que o *Softplus* e o *Relu* são as funções de ativação que melhor se encaixam neste modelo de classificação, apresentando resultados muito positivos, com especial destaque o *Softplus*, que conseguiu os melhores resultados.

A utilização da função de otimização *Gradient Descent Optimizer* com a função tangente hiperbólica como função de ativação neste modelo, deu origem a resultados bem aceitáveis, quando o conjunto de dados era relativamente mais baixo (com 100 exemplos, teve taxa de acerto de 0.78%). Nas restantes experiências utilizando esta função de otimização, os resultados foram todos eles pouco satisfatórios, à semelhança do que havia acontecido na abordagem multiclasse.

Assim como no modelo de classificação multiclasse, também aqui é possível observar que, em geral, os resultados da medida *F-Measure* AF são melhores que os resultados da média desta medida (AVG *F-Measure*) nas 2 classes, indicando assim que há um maior sucesso na classificação da classe "com arritmia" do que na classificação da classe "sem arritmia", sendo um bom indicador de sucesso deste modelo.

Na Figura 36 é possível visualizar um gráfico que apresenta a comparação da variação da média da *F-Measure* das 2 classes, nas diversas funções de ativação (usando a função de otimização *Adam Optimizer*), com o aumento da quantidade de exemplos utilizados nas experiências a este modelo.

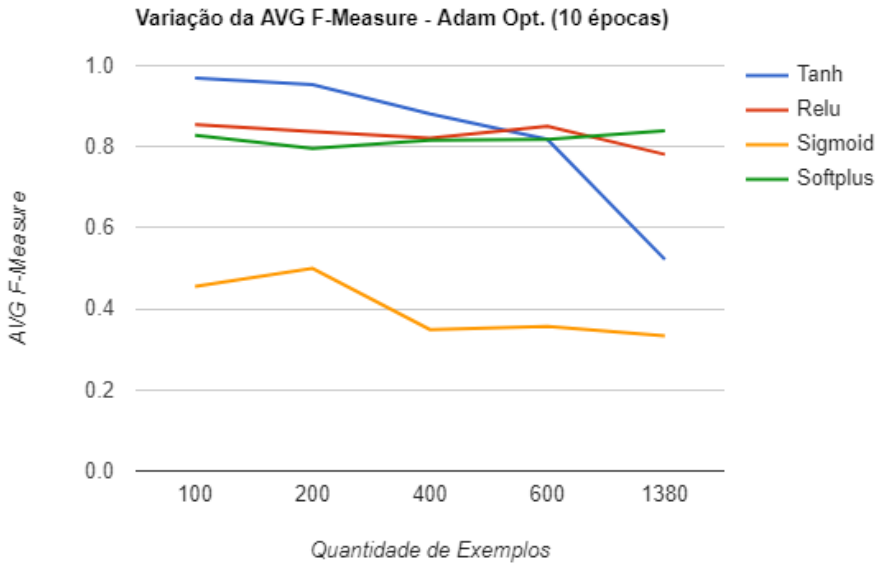

Figura 36 - Variação da média da *F-Measure* com aumento do *dataset* - 1ª Abordagem para classificação binária

Analisando o gráfico da Figura 36, é possível observar uma clara vantagem nos resultados obtidos pelo modelo usando a função de ativação *Tanh*, em comparação com as restantes, quando o conjunto de dados das experiências é de 100 e 200 exemplos. Com o aumento deste tamanho do *dataset*, é possível verificar que com o uso desta função, os resultados do modelo pioram abruptamente, chegando mesmo a valores muito baixos, perto dos resultados obtidos com a função *Sigmoid*.

As funções *Relu* e *Softplus*, conseguem manter um grande equilíbrio nos resultados das várias experiências com diferentes quantidades de dados, entre os 0.80% e os 0.86%, aproximadamente. Destaca-se aqui o uso da função *Softplus*, uma vez que, à exceção entre as experiências com 100 e 200 exemplos, existe uma melhoria constante nos resultados com o aumento de dados de treino e teste.

5.3.3 2ª abordagem - Modelo de classificação multiclasse

Na Tabela 24, são exibidos os resultados obtidos pelo modelo referente à 2ª abordagem de arquitetura da rede neuronal para a classificação multiclasse (4 classes). Foram efetuadas várias experiências com as diferentes quantidades de dados padrão (200, 400, 800 e 1200), sempre balanceados pelas 4 classes, usando a estratégia descrita na secção 4.3.1.3. A fase de seleção das imagens por segmentos, da secção 4.3.1.4, não foi aplicada nas experiências descritas nesta tabela.

Estas experiências foram realizadas utilizando a combinação de todas as funções de ativação mencionadas na secção 3.1.7 com as funções de otimização descritas na secção 3.1.8. É importante referir ainda que estas foram realizadas com diferentes parâmetros e configurações da rede, de forma a obter os melhores resultados possíveis. Um destes parâmetros foi o número de épocas, ou seja, o número de vezes que o modelo trabalha sobre o conjunto de dados de treino, tendo o modelo sido testado com 5 e 10 épocas. Os resultados do modelo com 10 épocas foram os que obtiveram melhor registo, razão pela qual são os que estão reproduzidos na Tabela 24. No Anexo III são apresentados os resultados das experiências deste modelo com 5 épocas.

Tabela 24 - Resultados da 2ª abordagem para classificação multiclasse - 10 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	0.535	0.805	0.310	**0.820**	0.250	0.250	0.275	0.250
	AUC	0.690	0.870	0.540	**0.880**	0.500	0.500	0.517	0.500
	AVG *F-Measure*	0.509	0.795	0.184	**0.810**	0.186	0.100	0.155	0.100
	F-Measure AF	0.517	**0.828**	0.282	0.760	0.172	0.400	0.163	0.400
400 (100 cada classe)	**Taxa de Acerto**	0.408	0.845	0.305	**0.855**	0.243	0.250	0.250	0.250
	AUC	0.605	0.897	0.537	**0.903**	0.495	0.500	0.500	0.500
	AVG *F-Measure*	0.346	0.838	0.189	**0.844**	0.103	0.100	0.100	0.100
	F-Measure AF	0.451	0.861	0.376	**0.904**	0.056	0.400	0.400	0.400
800 (200 cada classe)	**Taxa de Acerto**	0.401	**0.846**	0.250	0.824	0.255	0.250	0.250	0.250
	AUC	0.601	**0.898**	0.500	0.883	0.503	0.500	0.500	0.500
	AVG *F-Measure*	0.330	**0.842**	0.100	0.814	0.111	0.100	0.100	0.100
	F-Measure AF	0.420	**0.831**	0.280	0.817	0.014	0.400	0.000	0.400
1200 (300 cada classe)	**Taxa de Acerto**	0.354	**0.758**	0.255	0.641	0.250	0.255	0.250	0.250
	AUC	0.571	**0.839**	0.500	0.761	0.500	0.503	0.500	0.500
	AVG *F-Measure*	0.268	**0.737**	0.101	0.630	0.100	0.100	0.100	0.100
	F-Measure AF	0.444	**0.784**	0.243	0.677	0.000	0.400	0.000	0.400

Tal como é evidente na Tabela 24, os melhores resultados nas diversas medidas de avaliação, para o modelo em questão, foram obtidos usando a combinação das funções de ativação *Relu* e *Softplus*, com a função de otimização *Adam Optimizer*. Os resultados destas combinações são muito semelhantes nas diferentes medidas de avaliação, sendo que com a função *Softplus*, o modelo alcançou melhores resultados com 200 e 400 exemplos (imagens) de entrada, enquanto que com 800 e 1200 exemplos, o *Relu* apresentou melhores resultados.

As funções sigmoidais (*Tanh* e *Sigmoid*), em conjunto com o *Adam Optimizer* obtiveram resultados menos satisfatórios, nas diversas experiências em geral. No entanto, nas experiências com menos exemplos de entrada, a função de ativação *Tanh*, conseguiu obter resultados aceitáveis, com especial destaque na experiência com 200 registos, em que obteve 0.69% de AUC.

As experiências efetuadas com a função de otimização *Gradient Descent Optimizer*, apresentam resultados pouco satisfatórios, tal como aconteceu com as experiências realizadas aos modelos da 1ª abordagem (secções 735.3.1 e 5.3.2). É provável também aqui, que o parâmetro relativo ao *learning rate* não esteja com o valor mais apropriado, quando é usada esta função de otimização, uma vez que esta função mantém o mesmo valor deste parâmetro em todas as atualizações dos pesos, ao contrário da função *Adam Optimizer*, que o vai adaptando.

Na Figura 37 ilustra um gráfico que apresenta a comparação da variação da média da *F-Measure* das 4 classes, nas diversas funções de ativação (usando a função de otimização *Adam Optimizer*), com o aumento da quantidade de exemplos utilizados nas experiências a este modelo.

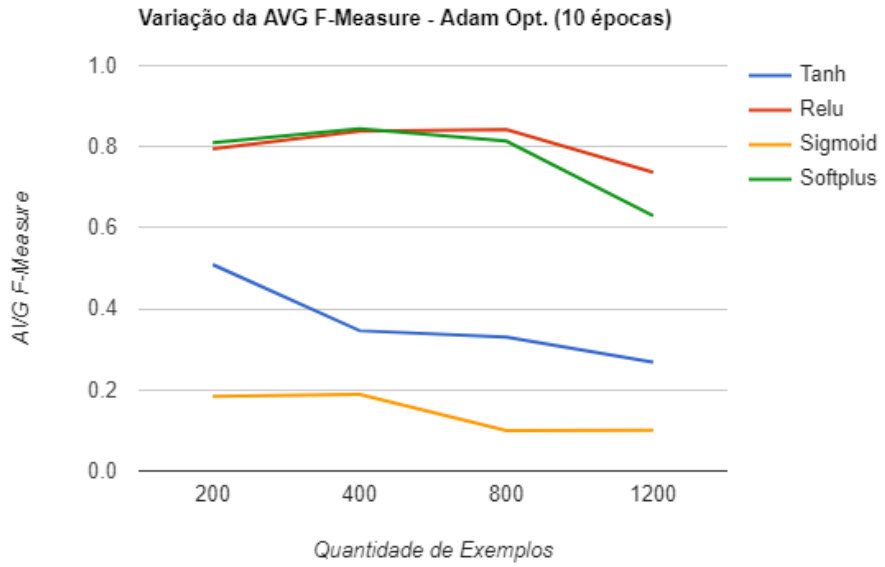

Figura 37 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação multiclasse

Com o aumento da quantidade de exemplos de entrada no processo de aprendizagem do modelo, é possível verificar que os resultados, com as diversas funções de ativação se foram degradando, como se pode ver na Figura 37. Esta degradação de resultados pode acontecer pelo facto de ser usada sempre a mesma ordem de exemplos no treino da rede, sendo que os exemplos mais simples de classificar são os primeiros a serem usados. Desta forma, ao aumentar o conjunto de dados, podem surgir exemplos mais difíceis de classificar, levando à degradação dos resultados.

A função de ativação *Relu*, entre todas é a que consegue manter um maior equilíbrio nos resultados, tendo apenas decaído entre as experiências com 800 e 1200 exemplos, mas mesmo assim manteve um ótimo registo nos resultados obtidos. A utilização das funções *Tanh* e *Sigmoid* no modelo, além de apresentar resultados bem inferiores em comparação às funções *Relu* e *Softplus*, estes ainda se vão degradando mais com o aumento de dados nas experiências.

5.3.4 2ª abordagem - Modelo de classificação binária

Na Tabela 25, são apresentados os resultados obtidos pelo modelo referente à 2ª abordagem de arquitetura da rede neuronal para a classificação binária (com e sem arritmia). Foram efetuadas diversas experiências com os diferentes tamanhos padrão de *dataset* (100, 200, 400

e 600 exemplos), de forma balanceados, usando a estratégia descrita na secção 4.3.1.3, e sem aplicar ainda fase de seleção das imagens por segmentos, da secção 4.3.1.4.

Estas experiências foram realizadas usando a mesma combinação de funções de ativação com as funções de otimização utilizadas nas experiências efetuadas ao modelo multiclasse. Tal como nessas experiências, também neste caso o modelo foi testado com 5 e 10 épocas, sendo que também aqui os resultados com 10 épocas foram os que obtiveram melhor registo. No Anexo IV são apresentados os resultados das experiências deste modelo com 5 épocas.

Tabela 25 - Resultados da 2ª abordagem para classificação binária - 10 épocas

Exemplos	**Métricas**	***Adam Optimizer***				***Gradient Descent Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada)	**Taxa de Acerto**	0.740	0.890	0.500	**0.910**	0.400	0.500	0.500	0.500
	AUC	0.740	0.890	0.500	**0.910**	0.400	0.500	0.500	0.500
	AVG *F-Measure*	0.731	0.870	0.333	**0.902**	0.322	0.333	0.367	0.333
	F-Measure AF	0.719	0.898	0.400	**0.917**	0.528	0.667	0.634	0.667
200 (100 cada)	**Taxa de Acerto**	0.785	0.865	0.500	**0.895**	0.515	0.500	0.500	0.500
	AUC	0.785	0.865	0.500	**0.895**	0.515	0.500	0.500	0.500
	AVG *F-Measure*	0.766	0.859	0.333	**0.892**	0.397	0.333	0.367	0.333
	F-Measure AF	0.828	0.878	0.467	**0.882**	0.660	0.667	0.634	0.667
400 (200 cada)	**Taxa de Acerto**	0.540	0.878	0.500	**0.905**	0.513	0.500	0.500	0.500
	AUC	0.540	0.878	0.500	**0.905**	0.513	0.500	0.500	0.500
	AVG *F-Measure*	0.492	0.875	0.333	**0.892**	0.381	0.333	0.333	0.333
	F-Measure AF	0.623	**0.882**	0.533	0.865	0.657	0.667	0.667	0.667
600 (300 cada)	**Taxa de Acerto**	0.553	0.872	0.500	**0.907**	0.510	0.500	0.500	0.500
	AUC	0.553	0.872	0.500	**0.907**	0.510	0.500	0.500	0.500
	AVG *F-Measure*	0.524	0.859	0.333	**0.898**	0.371	0.333	0.333	0.333
	F-Measure AF	0.625	0.894	0.667	**0.925**	0.635	0.667	0.667	0.667

Tal como no modelo de classificação multiclasse desta abordagem de arquitetura de rede neuronal, os melhores resultados nas diferentes métricas, para o modelo de classificação binário, foram obtidos usando a combinação das funções de ativação *Relu* e *Softplus*, com a função de treino *Adam Optimizer*. Os resultados destas combinações são muito idênticos nas diferentes medidas de avaliação. Desta vez, o modelo alcançou resultados ligeiramente melhores usando a função *Softplus*, em comparação com a função *Relu*, em todas as experiências efetuadas (com diferentes tamanhos de *dataset)*.

A combinação da função de ativação Tanh, com a função de treino *Adam Optimizer*, obteve também resultados bons nas experiências com 100 e 200 exemplos de entrada no processo de aprendizagem (acima dos 0.73% nas 3 medidas de avaliação). No entanto, com o aumento de exemplos, esta teve um declínio nos resultados obtidos.

As experiências efetuadas com a função de ativação *Sigmoid*, assim como as experiências usando a função de otimização *Gradient Descent Optimizer*, apresentam resultados pouco satisfatórios, tal como já acontecia no modelo multiclasse (Tabela 24).

Na Figura 38 é apresentado um gráfico que ilustra a variação da média da *F-Measure* das 2 classes, das experiências efetuadas a este modelo, usando as várias funções de ativação (usando a função de treino *Adam Optimizer*), com o aumento do tamanho do *dataset*.

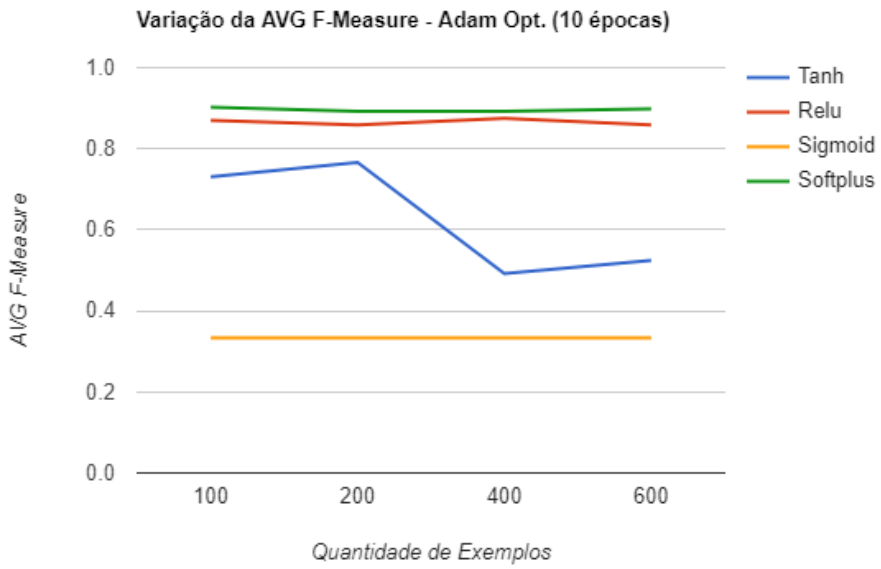

Figura 38 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação binária

Na Figura 38 é possível verificar que, com o aumento do tamanho do *dataset* no processo de aprendizagem do modelo de classificação binária, usando as funções de ativação *Relu* e *Softplus*, os resultados se mantêm muito constantes e bastante satisfatórios (0.90% com o *Softplus* e 0.87 com o *Relu*, aproximadamente). A função de ativação *Sigmoid* apresenta uma média de *F-Measure* constante nos 0.333%, tendo ainda como resultado da medida de avaliação AUC, o valor constante de 0.50%, demonstrando ser aproximadamente aleatório. Por último, com a função *Tanh*, é visível uma melhoria na média da *F-Measure* entre a experiência com 100 exemplos para a experiência com 200 exemplos. No entanto, com o aumento do *dataset*, os resultados sofrem uma degradação enorme, chegando a valores a rondar os 0.50% de média de *F-Measure*.

5.3.5 2ª abordagem - Modelo de classificação multiclasse (exemplos de 30 segundos)

As experiências efetuadas ao modelo de classificação multiclasse, apresentadas na secção 5.3.3, foram realizadas com segmentos de ECG (imagens) aleatórios do *dataset*. Assim sendo, a rede neuronal teve um processo de aprendizagem com segmentos de ECG de diferentes comprimentos de tempo, podendo ir dos 9 aos 61 segundos, aproximadamente. Desta forma, é muito provável a existência de imagens um pouco distorcidas neste processo, tal como foi referido na secção 4.3.1.1. Posto isto, usando a técnica de seleção das imagens por segmentos, da fase de pré-processamento dos dados, seria necessário verificar a influência do comprimento de tempo variável dos segmentos ECG, no desempenho do modelo.

Como apresentado na Tabela 19, o número de segmentos de comprimentos diferentes varia muito, sendo que, cerca de 70% dos exemplos do *dataset* possuem um comprimento de tempo de 30 segundos. Este é um facto que deve ser tido em consideração na utilização desta técnica, de seleção de imagens por segmentos. Foram, portanto, usados exemplos do *dataset* com comprimentos de tempo de 30 segundos. Desta forma, viabiliza uma possível comparação dos resultados destas experiências com os resultados das experiências em que os segmentos ECG tinham diferentes comprimentos de tempo, e assim avaliar a influência desta ação de pré-processamento.

Na Tabela 26 estão expostos os resultados obtidos por este modelo de classificação multiclasse, utilizando esta técnica. Os tamanhos dos conjuntos de dados utilizados nestas experiências, foram as mesmas que as quantidades utilizadas nas experiências sem esta seleção de imagens por segmentos (200, 400, 800 e 1200).

Estas experiências foram realizadas com as diversas funções de ativação utilizadas anteriormente, em conjunto com a função de treino *Adam Optimizer*, com 5 e 10 épocas. Os resultados apresentados na Tabela 26 correspondem às experiências com 10 épocas, sendo que os resultados das experiências com 5 épocas são apresentados no Anexo V. A função de treino *Gradient Descent Optimizer*, não foi nestas experiências utilizada, visto apresentar resultados pouco satisfatórios nas experiências anteriores.

Tabela 26 - Resultados da 2ª abordagem para classificação multiclasse (exemplos de 30 segundos) - 10 épocas

Exemplos	Métricas	*Adam Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	0.595	0.895	0.270	**0.905**
	AUC	0.730	0.930	0.513	**0.937**
	AVG *F-Measure*	0.581	0.884	0.139	**0.895**
	F-Measure AF	0.550	**0.931**	0.192	0.907
400 (100 cada classe)	**Taxa de Acerto**	0.480	**0.888**	0.378	0.875
	AUC	0.653	**0.925**	0.585	0.917
	AVG *F-Measure*	0.437	**0.878**	0.269	0.864
	F-Measure AF	0.496	0.891	0.412	**0.906**
800 (200 cada classe)	**Taxa de Acerto**	0.343	**0.879**	0.271	0.811
	AUC	0.553	**0.915**	0.497	0.867
	AVG *F-Measure*	0.286	**0.871**	0.112	0.792
	F-Measure AF	0.458	**0.889**	0.123	0.837
1200 (300 cada classe)	**Taxa de Acerto**	0.336	**0.818**	0.293	0.755
	AUC	0.529	**0.865**	0.500	0.821
	AVG *F-Measure*	0.212	**0.791**	0.113	0.711
	F-Measure AF	0.449	**0.854**	0.192	0.797

Algumas observações podem ser feitas a partir dos resultados apresentados na Tabela 26.

Tal como no modelo com o conjunto de dados não segmentado (secção 5.3.3), os melhores resultados foram obtidos usando a combinação das funções de ativação *Relu* e *Softplus*, com a função de otimização *Adam Optimizer*. Os resultados destas combinações são muito semelhantes nas diferentes medidas de avaliação, sendo que com a função *Softplus*, o modelo alcançou melhores resultados com 200 exemplos de entrada, enquanto que com nas restantes experiências (400, 800 e 1200 exemplos), o *Relu* apresentou melhores resultados.

As funções sigmoidais (*Tanh* e *Sigmoid*), em conjunto com o *Adam Optimizer* obtiveram também nestas experiências, resultados menos satisfatórios.

A comparação destes resultados, usando a técnica de seleção de imagens por segmentos, com os resultados sem a utilização desta técnica, é efetuada na secção 0, sendo dessa forma possível avaliar a influência desta ação de pré-processamento.

Na Figura 39 é apresentado um gráfico que ilustra a variação da média da *F-Measure* das 4 classes, das experiências efetuadas a este modelo, usando as várias funções de ativação (e usando a função de treino *Adam Optimizer*), com o aumento do tamanho do *dataset*.

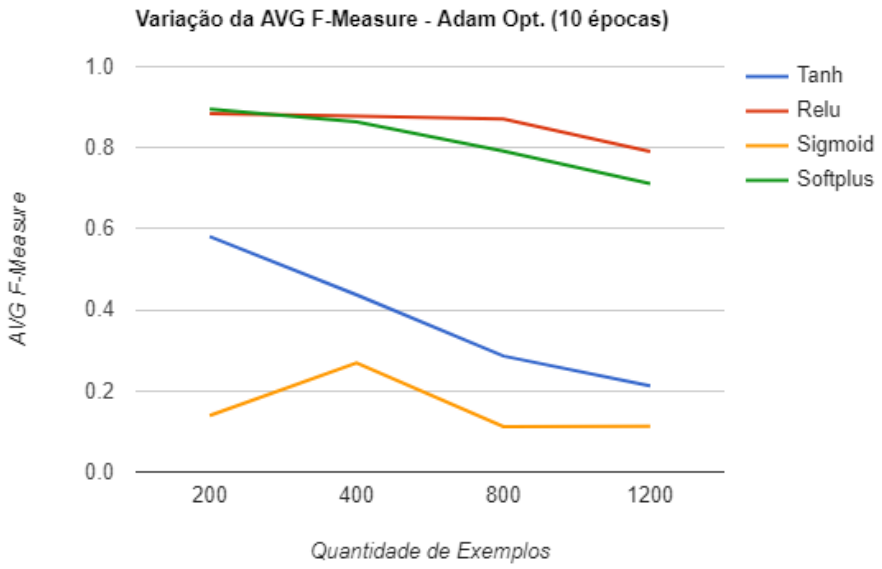

Figura 39 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação multiclasse (exemplos de 30 segundos)

Analisando a Figura 39, é possível observar que a utilização das funções de ativação *Relu* e *Softplus* no modelo, originam resultados bem superiores em comparação às restantes funções. É também visível que, ao longo das experiências, com o aumento das quantidades de exemplos de entrada, os resultados se vão deteriorando. Mais uma vez, a função *Relu*, entre as funções de ativação é a que consegue manter um maior equilíbrio nos resultados das várias experiências (com diferentes quantidades de dados), para além de apresentar os melhores resultados, comparativamente às restantes funções, com o mesmo conjunto de dados.

5.3.6 2ª abordagem - Modelo de classificação binária (exemplos de 30 segundos)

Na Tabela 27, pode-se observar os resultados obtidos pelo modelo referente à 2ª abordagem de arquitetura da rede neuronal para a classificação, usando a técnica de seleção de imagens por segmentos, tal como se sucedeu com o modelo de classificação multiclasse (ver secção 5.3.5). Também esta abordagem de classificação binária tinha sido já testada (ver secção 5.3.4), sendo que a seleção dos exemplos do *dataset* foi aleatória, não utilizando a técnica de seleção de imagens por segmentos. Foram então efetuadas experiências com os diferentes tamanhos de *dataset* (100, 200, 400 e 600 exemplos), balanceados pelas 2 classes, tal como aconteceu nas outras experiências, mas com a utilização desta técnica de segmentação.

Estas experiências foram efetuadas com as várias funções de ativação, juntamente à função de treino *Adam Optimizer*, com 5 e 10 épocas. Os resultados apresentados na Tabela 27 correspondem às experiências com 10 épocas, sendo que os resultados das experiências com 5 épocas podem ser vistos no Anexo VI. A função de treino *Gradient Descent Optimizer*, não foi

utilizada nestas experiências, visto ter apresentado resultados pouco satisfatórios nas experiências anteriores.

Tabela 27 - Resultados da 2ª abordagem para classificação binária (exemplos de 30 segundos) - 10 épocas

Exemplos	**Métricas**	***Adam Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	0.740	**0.860**	0.500	0.**860**
	AUC	0.740	**0.860**	0.500	0.**860**
	AVG *F-Measure*	0.720	**0.845**	0.333	0.840
	F-Measure AF	0.729	**0.890**	0.400	0.863
200 (100 cada classe)	**Taxa de Acerto**	0.740	0.890	0.500	**0.905**
	AUC	0.740	0.890	0.500	**0.905**
	AVG *F-Measure*	0.726	0.877	0.333	**0.897**
	F-Measure AF	0.754	0.901	0.467	**0.918**
400 (200 cada classe)	**Taxa de Acerto**	0.610	0.903	0.500	**0.913**
	AUC	0.610	0.903	0.500	**0.913**
	AVG *F-Measure*	0.598	0.898	0.333	**0.905**
	F-Measure AF	0.656	0.916	0.533	**0.929**
600 (300 cada classe)	**Taxa de Acerto**	0.588	0.897	0.500	**0.912**
	AUC	0.588	0.897	0.500	**0.912**
	AVG *F-Measure*	0.566	0.889	0.333	**0.901**
	F-Measure AF	0.645	0.916	0.533	**0.929**

Observando a Tabela 27, é possível identificar que com o uso das funções *Relu* e *Softplus* no modelo de classificação binário, os resultados são muito superiores comparativamente ao uso das funções sigmoidais (*Tanh* e *Sigmoid*), para todas as medidas de avaliação. Os resultados destas combinações são muito semelhantes nas diferentes medidas de avaliação, sendo que com a função *Softplus* os resultados conseguem ser um pouco superiores, nas diversas experiências com diferentes quantidades de dados. Importa realçar ainda que, os resultados das experiências com mais de 100 exemplos, usando a função *Softplus*, tiveram resultados sempre acima dos 0.90%, em todas as medidas de avaliação utilizadas.

A comparação destes resultados, usando a técnica de seleção de imagens por segmentos, com os resultados sem a utilização desta técnica (da secção 5.3.4), é efetuada na secção 0, sendo dessa forma possível avaliar a influência desta ação de pré-processamento.

Na Figura 40Figura 39 é apresentado um gráfico que ilustra a variação da média da *F-Measure* das 2 classes, das experiências efetuadas a este modelo, usando as várias funções de ativação (e usando a função de treino *Adam Optimizer*), com o aumento do tamanho do *dataset*.

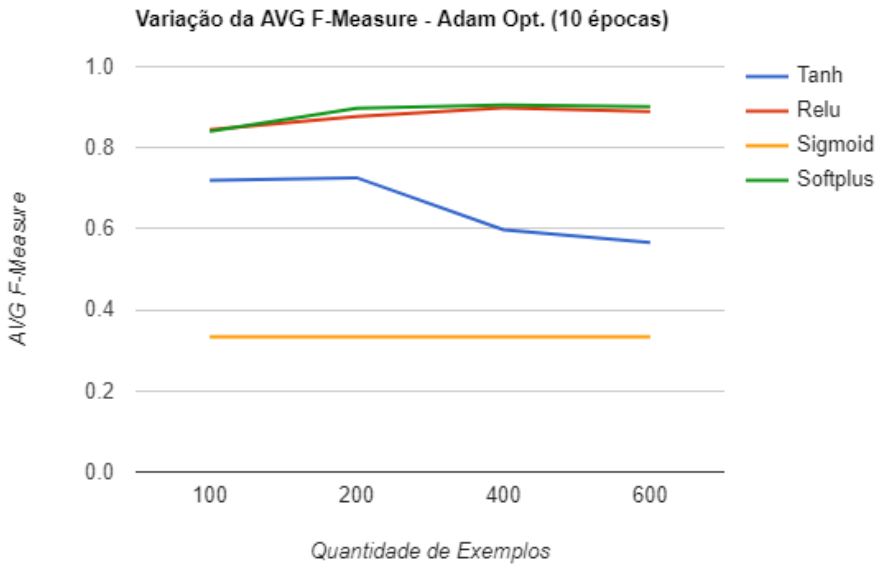

Figura 40 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação binária (exemplos de 30 segundos)

Na Figura 40, pode-se observar que a utilização das funções de ativação *Relu* e *Softplus* no modelo, originam resultados bastante superiores em comparação às restantes funções, tal como tinha já referido anteriormente. Usando as funções *Relu* e *Softplus*, para além de os resultados serem muito satisfatórios, é ainda visível que, ao longo das experiências, com o aumento das quantidades de exemplos de entrada, os resultados vão sempre melhorando. Ao contrário do que acontecia nas experiências efetuadas ao modelo de classificação multiclasse (Figura 39).

A combinação da função de ativação Tanh, com a função de treino *Adam Optimizer*, obteve também resultados bons nas experiências com 100 e 200 exemplos de treino (acima dos 0.72% nas várias medidas). No entanto, com o aumento da quantidade de exemplos, esta apresentou um declínio nos resultados.

5.3.7 2ª abordagem - Modelo de classificação multiclasse (segmentos de 10 segundos)

As experiências efetuadas ao modelo de classificação multiclasse, apresentadas na secção 5.3.3, foram realizadas com registos de ECG (imagens) aleatórios do *dataset* (com diferentes comprimentos de tempo). Já as experiências apresentadas na secção 5.3.5, foram efetuadas com segmentos de ECG selecionados (apenas exemplos de 30 segundos), usando a técnica de seleção de imagens por segmentos (descrita na secção 4.3.1.4). Faltava aplicar agora a técnica de divisão de imagens por segmentos (secção 4.3.1.5), da fase de pré-processamento, para

assim poder comparar com estas experiências anteriores, de forma a ser possível avaliar a influência desta ação de pré-processamento.

Portanto, foram efetuadas experiências ao modelo, alimentando o processo de aprendizagem, apenas com exemplos do *dataset* com comprimentos de tempo de 10 segundos, resultantes da aplicação desta técnica.

Na Tabela 28 estão expostos os resultados por este modelo para a classificação multiclasse, utilizando esta técnica. Os tamanhos dos conjuntos de dados utilizados nestas experiências, foram as mesmas que as utilizadas nas experiências sem a aplicação desta técnica (200, 400, 800 e 1200 exemplos).

Estas experiências foram realizadas com as diversas funções de ativação utilizadas até então, em conjunto com a função de treino *Adam Optimizer*, com 10 épocas.

Tabela 28 - Resultados da 2ª abordagem para classificação multiclasse (segmentos de 10 segundos) - 10 épocas

Exemplos	**Métricas**	***Adam Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	0.630	**0.895**	0.250	0.865
	AUC	0.753	**0.930**	0.500	0.910
	AVG *F-Measure*	0.604	**0.886**	0.100	0.849
	F-Measure AF	0.719	**0.899**	0.280	0.825
.400 (100 cada classe)	**Taxa de Acerto**	0.485	0.883	0.338	**0.888**
	AUC	0.657	0.922	0.558	**0.925**
	AVG *F-Measure*	0.438	0.875	0.234	**0.883**
	F-Measure AF	0.535	0.870	0.291	**0.883**
800 (200 cada classe)	**Taxa de Acerto**	0.409	**0.854**	0.250	0.838
	AUC	0.606	**0.903**	0.500	0.892
	AVG *F-Measure*	0.344	**0.842**	0.100	0.826
	F-Measure AF	0.415	**0.860**	0.280	0.859
1200 (300 cada classe)	**Taxa de Acerto**	0.411	0.683	0.271	**0.799**
	AUC	0.607	0.789	0.497	**0.866**
	AVG *F-Measure*	0.307	0.657	0.112	**0.775**
	F-Measure AF	0.458	0.676	0.123	**0.811**

Observando a Tabela 28, identifica-se claramente que com o uso das funções *Relu* e *Softplus* neste modelo, os resultados são muito superiores comparativamente ao restante das funções (*Tanh* e *Sigmoid*), para todas as medidas de avaliação. Os resultados usando estas duas funções (*Relu* e *Softplus*) são idênticos nas diferentes medidas de avaliação, em todas experiências.

A comparação destes resultados, usando a técnica de divisão de imagens por segmentos de 10 segundos, com os resultados das experiências das restantes abordagens, é efetuada na secção 0, sendo dessa forma possível avaliar a influência desta ação de pré-processamento.

Na Figura 41Figura 39 é apresentado um gráfico que ilustra a variação da média da *F-Measure* das 4 classes, das experiências efetuadas a este modelo, usando as várias funções de ativação (e usando a função de treino *Adam Optimizer*), com o aumento do tamanho do *dataset*.

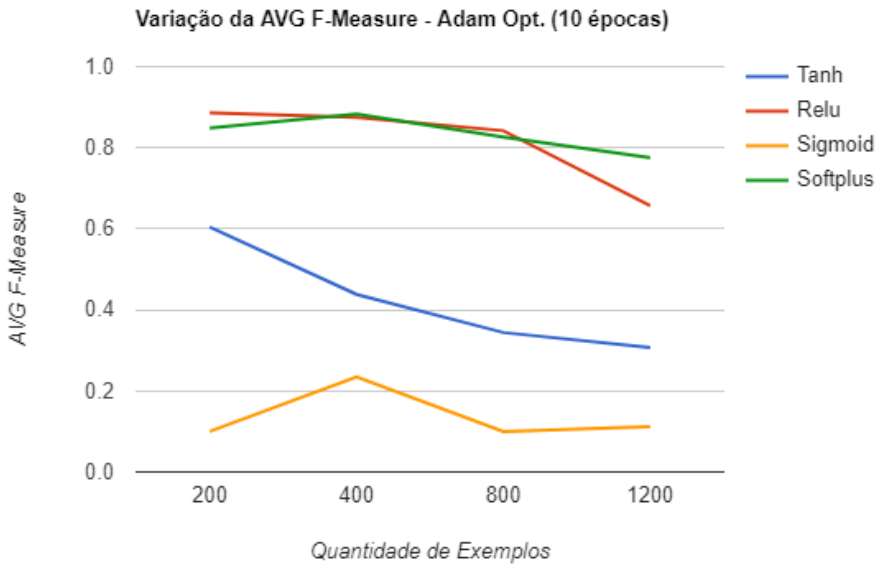

Figura 41 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação multiclasse (segmentos de 10 segundos)

Analisando a Figura 41, é possível observar que a utilização das funções de ativação *Relu* e *Softplus* no modelo, mais uma vez, originam resultados bem superiores em comparação às restantes funções. É também visível que, ao longo das experiências, com o aumento das quantidades de exemplos de treino, os resultados se vão deteriorando. A função *Softplus*, entre as funções de ativação é a que consegue manter um maior equilíbrio nos resultados das várias experiências (com diferentes quantidades de dados), para além de apresentar os melhores resultados, comparativamente às restantes funções, com o mesmo conjunto de dados.

5.3.8 2ª abordagem - Modelo de classificação binária (segmentos de 10 segundos)

Na Tabela 29Tabela 27, pode-se analisar os resultados obtidos pelo modelo referente à 2ª abordagem de arquitetura da rede neuronal para a classificação, usando a técnica de divisão de imagens por segmentos, tal como se sucedeu com o modelo de classificação multiclasse (ver secção 5.3.7). Também este modelo de classificação binária tinha sido já testado (ver secção 5.3.4 e secção 5.3.6), com os exemplos do *dataset* aleatórios e com a aplicação da técnica de seleção de imagens por segmentos (exemplos de 30 segundos apenas). Faltava aplicar agora a

técnica de divisão de imagens por segmentos (secção 4.3.1.5), para assim poder comparar com estas experiências anteriores, de forma a ser possível avaliar a influência desta ação de pré-processamento. Foram então efetuadas experiências com os diferentes tamanhos de *dataset* (100, 200, 400 e 600 exemplos), balanceados pelas 2 classes, tal como aconteceu nas outras experiências, mas com a utilização desta técnica de segmentação.

Estas experiências foram efetuadas com as várias funções de ativação utilizadas até então, e a função de treino *Adam Optimizer*, com 10 épocas de treino.

Tabela 29 - Resultados da 2ª abordagem para classificação binária (segmentos de 10 segundos) - 10 épocas

Exemplos	**Métricas**	***Adam Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	0.830	0.880	0.500	**0.910**
	AUC	0.830	0.880	0.500	**0.910**
	AVG *F-Measure*	0.820	0.859	0.333	**0.909**
	F-Measure AF	0.806	0.891	0.533	**0.918**
200 (100 cada classe)	**Taxa de Acerto**	0.835	**0.900**	0.500	**0.900**
	AUC	0.835	**0.900**	0.500	**0.900**
	AVG *F-Measure*	0.830	**0.888**	0.333	0.882
	F-Measure AF	0.811	**0.918**	0.533	0.908
400 (200 cada classe)	**Taxa de Acerto**	0.730	0.885	0.500	**0.938**
	AUC	0.730	0.885	0.500	**0.938**
	AVG *F-Measure*	0.803	0.880	0.333	**0.937**
	F-Measure AF	0.742	0.903	0.467	**0.940**
600 (300 cada classe)	**Taxa de Acerto**	0.658	**0.915**	0.500	0.908
	AUC	0.658	**0.915**	0.500	0.908
	AVG *F-Measure*	0.633	**0.911**	0.333	0.901
	F-Measure AF	0.654	**0.929**	0.533	0.924
1200 (600 cada classe)	**Taxa de Acerto**	0.500	**0.863**	0.500	0.850
	AUC	0.500	**0.863**	0.500	0.850
	AVG *F-Measure*	0.633	**0.843**	0.333	0.824
	F-Measure AF	0.654	**0.894**	0. 667	0.835

Na Tabela 29 é possível identificar que o modelo de classificação binários, com o uso das funções *Relu* e *Softplus*, os resultados são muito superiores comparativamente ao uso das funções sigmoidais (*Tanh* e *Sigmoid*), para todas as medidas de avaliação. Os resultados destas melhores combinações são idênticos nas diferentes medidas de avaliação, em todas as experiências. Importa realçar ainda que, os resultados das experiências usando estas funções (*Relu* e *Softplus*), tiveram sempre resultados próximos dos 0.90%, em todas as medidas de avaliação utilizadas.

A comparação destes resultados, usando a técnica de divisão de imagens por segmentos de 10 segundos, com os resultados das experiências das restantes abordagens, é efetuada na secção 0, sendo dessa forma possível avaliar a influência desta ação de pré-processamento.

Na Figura 42Figura 39 é apresentado um gráfico que ilustra a variação da média da *F-Measure* das 2 classes, das experiências efetuadas a este modelo, usando as várias funções de ativação (e usando a função de treino *Adam Optimizer*), com o aumento do tamanho do *dataset*.

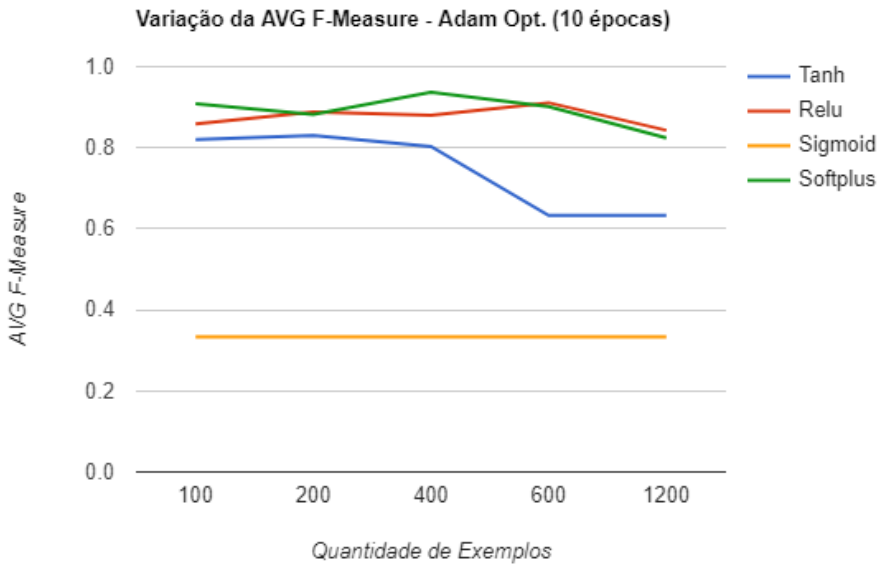

Figura 42 - Variação da média da *F-Measure* com aumento do *dataset* - 2ª Abordagem para classificação binária (segmentos de 10 segundos)

Na Figura 42, pode-se observar que a utilização das funções de ativação *Relu* e *Softplus* neste modelo binário, dá origem a resultados bastante superiores em comparação às restantes funções, tal como tinha já referido anteriormente. Usando as funções *Relu* e *Softplus*, para além de os resultados serem muito satisfatórios, é ainda visível que, ao longo das experiências, com o aumento das quantidades de exemplos de entrada, os resultados vão melhorando várias vezes. Ao contrário do que acontecia nas experiências efetuadas ao modelo de classificação multiclasse (Figura 41).

A combinação da função de ativação Tanh, com a função de treino *Adam Optimizer*, obteve também resultados muito bons nas experiências com 100, 200 e 400 exemplos de treino (acima dos 0.80% nas várias medidas). No entanto, com o aumento da quantidade de exemplos, esta apresentou uma degradação nos resultados.

5.3.9 Discussão de Resultados

Na Figura 43 e Figura 44 é possível visualizar dois gráficos que apresentam um resumo dos melhores resultados obtidos por cada um dos modelos de classificação multiclasse testados, para as medidas de avaliação AVG *F-Measure* e *F-Measure* AF, respetivamente.

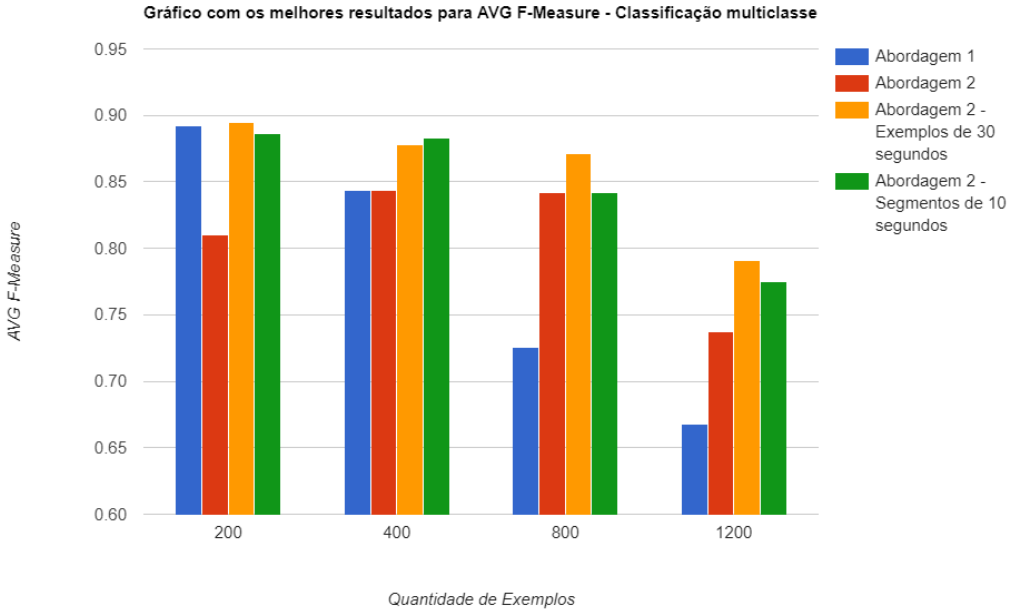

Figura 43 - Melhores resultados para média da *F-Measure* - Classificação Multiclasse

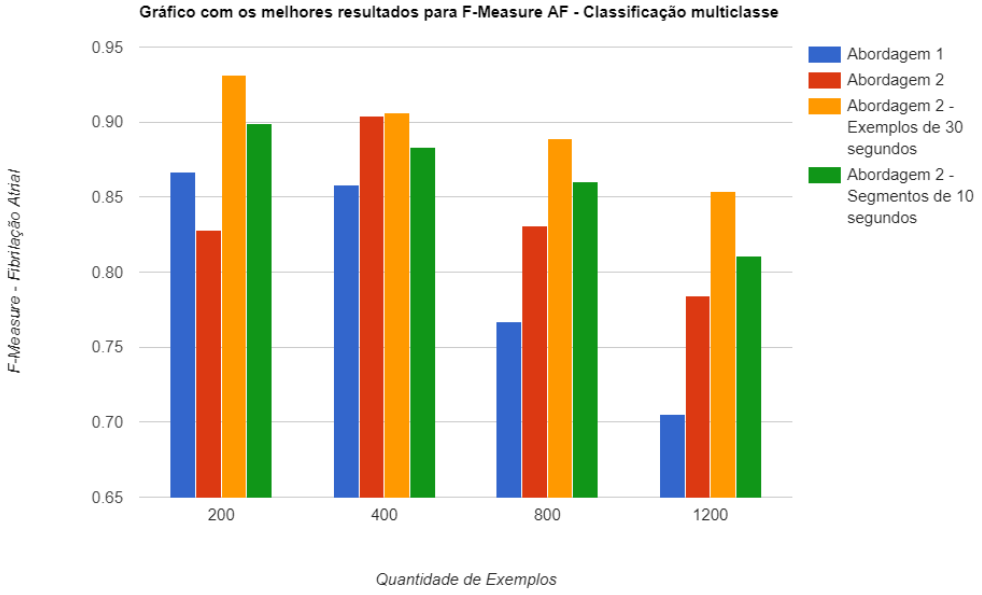

Figura 44 - Melhores resultados para *F-Measure* AF - Classificação Multiclasse

Analisando a Figura 43 e Figura 44, observa-se facilmente que, no geral, a 2ª abordagem de arquitetura de rede neuronal apresenta resultados bem superiores relativamente à 1ª abordagem. Em nosso entender, esta superioridade justifica-se, pela arquitetura de rede mais complexa, com mais camadas de processamento (ver secção 4.3.2.2), e pelo conjunto de imagens mais legíveis (ver secção 4.3.1.2) com que esta rede é treinada/testada. Com a aplicação das técnicas de pré-processamento, seleção de imagens por segmentos de 30 segundos ou divisão de imagens por segmentos de 10 segundos (ver secções 4.3.1.4 e 4.3.1.5, respetivamente) nestes modelos da 2ª abordagem, é possível observar ainda uma grande melhoria nos resultados.

É possível visualizar no gráfico, esta 2ª abordagem, apenas com a aplicação da técnica de balanceamento de dados (a vermelho), a 2ª abordagem com aplicação da técnica de seleção de imagens por segmentos de 30 segundos (a laranja) e finalmente, a 2ª abordagem com a aplicação da técnica de divisão de imagens em segmentos de 10 segundos (a verde).

A 2ª abordagem com a seleção de imagens por segmentos de 30 segundos revela-se ainda assim a melhor abordagem a seguir, alcançando resultados bem superiores, comparativamente às restantes abordagens. Com o aumento de quantidades de exemplos nas experiências, os resultados desta abordagem distanciam-se (para melhor) cada vez mais das restantes abordagens. Este distanciamento não é, no entanto, surpreendente, visto que, para um comprimento de segmento fixo, o desempenho dos modelos tende a aprimorar-se quantos mais exemplos são fornecidos para treino.

Se compararmos os resultados da média da medida *F-Measure* das 4 classes (Figura 43) com os resultados da *F-Measure* da classe AF (Figura 44), é possível observar que os valores da *F-Measure* desta classe são bem superiores aos valores da média. Estes dados indicam que os modelos apresentam um maior sucesso na classificação desta classe, ou seja, na deteção de arritmias cardíacas, comparativamente às restantes classes.

Na Figura 45 e Figura 46 pode-se observar dois gráficos que apresentam um resumo dos melhores resultados obtidos por cada um dos modelos de classificação binários testados, para as medidas de avaliação AVG *F-Measure* e *F-Measure* AF, respetivamente.

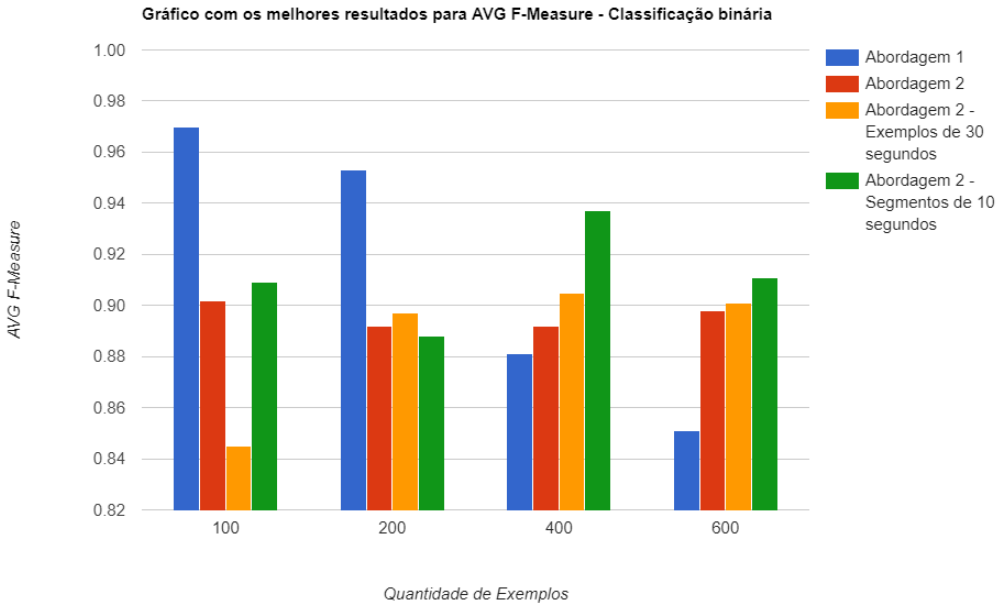

Figura 45 - Melhores resultados para média da *F-Measure* - Classificação Binária

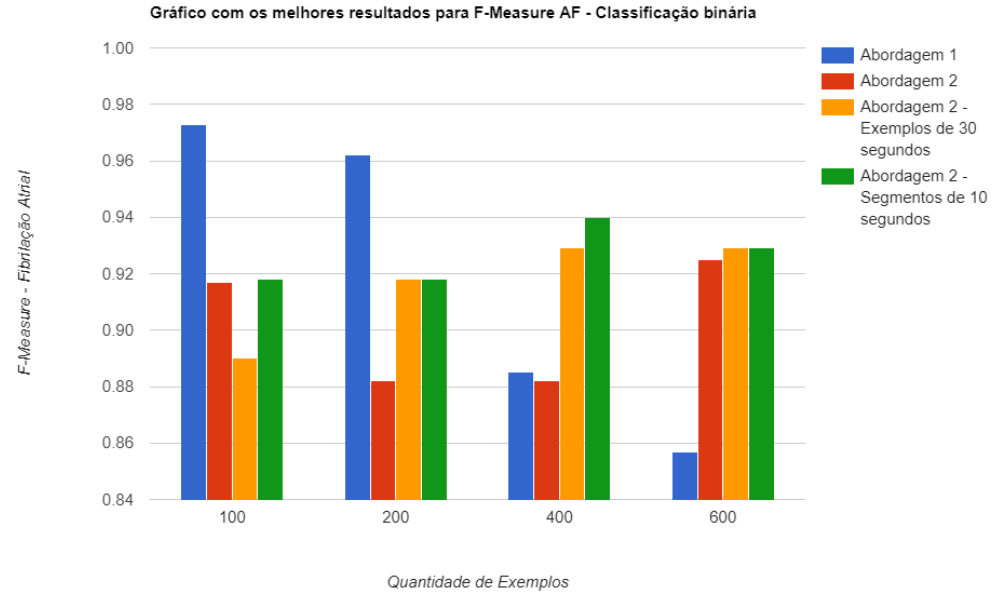

Figura 46 - Melhores resultados para *F-Measure* AF - Classificação Binária

A Figura 45 e Figura 46, mostram que, no caso da classificação binária, a 1ª abordagem apresenta resultados um pouco superiores, comparativamente às restantes abordagens, nas experiências com 100 e 200 exemplos. Esta obtém excelentes resultados nestas experiências, com a média de *F-Measure* acima dos 0.95%, e *F-Measure* da classe "com arritmia" acima dos 96%. Nas experiências com conjuntos de dados maiores (acima de 200 exemplos) a 2ª abordagem revela-se superior no sucesso da classificação dos registos ECG. A aplicação das

técnicas de pré-processamento, seleção de imagens por segmentos ou divisão de imagens por segmentos (ver secções 4.3.1.4 e 4.3.1.5, respetivamente) nestes modelos da 2ª abordagem, apresenta também melhorias nos resultados, tal como acontecia na classificação multiclasse. Neste caso, a divisão de imagens por segmentos de 10 segundos revela-se ainda assim a melhor abordagem a seguir, alcançando resultados um pouco melhores, comparativamente às restantes.

Importante ainda referir que esta alternativa de modelos de classificação binária, apesar de alterar um pouco o problema descrito nesta dissertação, surgiu para tentar melhorar o processo de deteção de arritmias, visto ser esse o principal objetivo deste sistema. Se compararmos os gráficos das Figura 43 com a Figura 45 e da Figura 44 com a Figura 46, é possível notar realmente uma melhoria acentuada nos resultados.

5.4 Comparação dos resultados com outras soluções

A Tabela 30 apresenta os resultados de algumas soluções desenvolvidas por outros autores (participantes do concurso lançado pela PhysioNet (PhysioNet, 2017)) para resolver este problema, utilizando o mesmo conjunto de dados. As abordagens seguidas por estes autores podem ser vistas na Tabela 16.

Os resultados são apresentados através da medida de avaliação F-*Measure*, visto que esta foi a medida utilizada no concurso para classificar as soluções e assim definir o *ranking* dos resultados dos participantes.

Importa referir que os resultados apresentados por estes autores, resultam de um teste com 3.658 registos, privados ao concurso (não disponíveis ao público), que estavam destinados para testes (ver secção 4.1).

Para além disso, neste trabalho, foram aplicadas técnicas de balanceamentos de dados, como a seleção de apenas alguns registos de cada classe, ignorando os restantes, de modo a resolver o grande desbalanceamento das classes. Ao comparar os resultados obtidos tem de ter-se em conta esta alteração ao problema inicial.

Portanto, devido a estas duas condicionantes, estes resultados dos participantes devem ser vistos apenas como indicadores, não podendo ser diretamente comparados com a solução desenvolvida no âmbito desta dissertação.

Tabela 30 - Resultados de outras soluções

***Ranking* no concurso**	**Autores**	**Algoritmo de Classificação**	**Resultado (*F-Measure*)**
1	(Zabihi et al., 2017)	*Random Forest*	0.826%
5	(Xiong et al., 2017)	CNN	0.82%
5	(Zihlmann et al., 2017)	CRNN	0.82%
9	(Smolen, 2017)	RNN e GBM	0.81%
18	(Warrick et al., 2017)	CNN e LSTM	0.80%

Não tendo resultados da solução desenvolvida no âmbito desta dissertação que possam ser diretamente comparáveis com os resultados das soluções apresentadas na Tabela 30, são então expostos, na Tabela 31, os melhores resultados (AVG *F-Measure*) das experiências efetuadas.

Tabela 31 – Melhores resultados (AVG *F-Measure*) das experiências

Modelo de Classificação	**Quantidade de Exemplos**	**Resultado (*F-Measure*)**
2ª abordagem (Seleção de imagens de 30 segundos)	200	0.895%
2ª abordagem (Divisão de imagens em segmentos de 10 seg.)	200	0.884%
2ª abordagem (Divisão de imagens em segmentos de 10 seg.)	400	0.883%
2ª abordagem (Seleção de imagens de 30 segundos)	400	0.878%
2ª abordagem (Seleção de imagens de 30 segundos)	800	0.871%
2ª abordagem (Divisão de imagens em segmentos de 10 seg.)	800	0.842%
2ª abordagem (Seleção de imagens de 30 segundos)	1200	0.791%
2ª abordagem (Divisão de imagens em segmentos de 10 seg.)	1200	0.775%

Como é possível observar, os melhores resultados referentes à média da *F-Measure* das 4 classes, obtidos pelos modelos da solução desta dissertação, não se diferenciam muito dos resultados das soluções apresentados pelos outros autores, situados na Tabela 30. Poderá ser um bom indicador de qualidade por parte da solução desenvolvida.

Foram discutidos e comparados os resultados de diferentes abordagens seguidas neste trabalho. Foram também apresentados alguns resultados publicados para o mesmo *dataset*. anterior. Em algumas das abordagens propostas apresentadas obtiveram melhores resultados, como pode ser observado na Tabela 31. Nesta tabela, para a medida de avaliação *F-Measure*, obtêm-se valores como 0.895 e 0.884 contra 0.826 do melhor resultado obtido noutros trabalhos (ver Tabela 30).

Para suportar a afirmação de que este modelo de classificação é melhor quando comparado com outros modelos, devem ser utilizados testes estatísticos sobre os resultados obtidos para modelos do mesmo *dataset* e para o mesmo problema.

Para comparar os vários modelos em simultâneo, como seria esperado neste trabalho, Demsar (Demsar, 2006) recomenda o teste de Friedman. Este teste, não-paramétrico, muito semelhante ao teste ANOVA (também recomendado pelo autor), que classifica os modelos para cada conjunto de dados separadamente, listando-os depois numa tabela classificativa consoante os seus resultados de desempenho. Assim, neste caso, seria de usar o teste de Friedman (teste não paramétrico) para comparação dos modelos (seguido do teste de Nemenyi, um teste post-hoc).

6 Conclusão

O trabalho apresentado nesta dissertação tem por principal objetivo contribuir para o desenvolvimento de uma metodologia capaz de classificar sinais resultantes do ECG, para deteção de arritmias cardíacas em humanos. Desta forma, este sistema poderá ser usado pelos técnicos de saúde, oferecendo apoio na decisão sobre os diagnósticos de pacientes, proporcionando assim uma melhoria na qualidade de vida destes.

Neste documento, foi apresentada uma metodologia para o tratamento dos dados obtidos a partir de eletrocardiogramas, utilizando técnicas de pré-processamento, e de seguida aplicar e avaliar as diferentes abordagens de modelos de classificação sugeridos (usando *deep learning*, mais concretamente Redes Neurais Convolucionais).

Os resultados obtidos pelos modelos de classificação mostram que a 2ª abordagem implementada (ver secção 4.3.2.2) obtém os resultados mais consistentes ao longo das diferentes experiências. Esta superioridade por parte da 2ª abordagem é alcançada tanto no modelo multiclasse (referente ao problema original, de 4 classes) como no modelo binário (alternativa para melhorar resultados de deteção de arritmias, sendo este o principal objetivo). A aplicação das técnicas de seleção de imagens por segmentos ou de divisão de imagens por segmentos (ver secções 4.3.1.4 e 4.3.1.5, respetivamente) nestes modelos da 2ª abordagem, demonstram ainda uma melhoria substancial nos resultados. Foram efetuadas diversas análises aos resultados obtidos, nomeadamente com as medidas taxa de acerto, AUC e *F-Measure*. Os melhores resultados obtidos, para estas diferentes abordagens, podem ser visualizados na secção 5.3.9.

Por último, considerando todo o trabalho desenvolvido e os resultados obtidos, pensamos que esta metodologia poderá servir como suporte útil para o desenvolvimento de sistemas de deteção e previsão de arritmias cardíacas em ECG.

6.1 Trabalho Futuro

Numa perspetiva de continuação deste trabalho, são de considerar os seguintes pontos:

- Fazer uma melhor avaliação dos diferentes parâmetros e configurações dos modelos, avaliando as suas possíveis melhorias. Avaliar, nomeadamente, a variação do número de épocas (superior a 10) e a distribuição dos exemplos de treino;
- Testar mais técnicas de pré-processamento dos dados, referidas na literatura. Como por exemplo, a divisão dos registos ECG em vários segmentos de imagens, em que cada segmento corresponde a um pico R da onda de ECG;
- Melhorar o *hardware* das máquinas utilizadas para as experiências, de modo a ser possível efetuar testes mais complexos, abrangendo um maior conjunto de dados nos modelos implementados;
- Otimizar os modelos implementados, de modo a reduzir o tempo de execução de um treino da rede neuronal;
- Alargar esta análise a outros tipos de batimentos cardíacos, de forma a conseguir mais classes de análise;
- Realizar mais experiências, com maiores e diferentes conjuntos de dados, de modo a consolidar os resultados obtidos.

Referências

(Barnes et al., 2009) Barnes, C., Blake, H., Pinder, D. (2009). *Creating and Delivering Your Value Proposition: Managing Customer Experience for Profit*. Kogan Page Publishers.

(Breiman, 2001) Breiman, L., (2001). *Random Forests. Machine Learning*. (pp. 5-32).

(Brown, 2017) Gizmodom, (2017). *Um Algoritmo consegue diagnosticar uma doença cardíaca melhor que um ser humano?*. [online] Disponível em: http://gizmodo.uol.com.br/algoritmo-doenca-cardiaca/ [visto a 17 de Janeiro 2018].

(Brownlee J., 2016) Brownlee J. (2016). *Supervised and Unsupervised Machine Learning Algorithms*. [online] Disponível em: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ [visto a 02 de fevereiro 2018].

(Chandra et al., 2017) B. S. Chandra, C. S. Sastry, S. Jana and S. Patidar, (2017). *Atrial Fibrillation Detection Using Convolutional Neural Networks.*

(Clifford, 2017) Gari Clifford, Chengyu Liu, Benjamin Moody, Li-wei H. Lehman, Ikaro Silva, Qiao Li, Alistair Johnson, Roger G. Mark. (2017). AF Classification from a Short Single Lead ECG Recording: the PhysioNet Computing in Cardiology Challenge 2017. Computing in Cardiology (Rennes: IEEE), Vol 44.

(Cochran et al., 1967) W.T. Cochran, J.W. Cooley, D.L. Favin, H.D. Helms, R.A. Kaenel, W.W. Lang, G.C. Maling, D.E. Nelson, C.M. Rader, P.D. Welch. (1967). *What is the fast Fourier transform?*, IEEE Explore em 1664 – 1674, Outubro 1967.

(Coração Saudável, 2018) Coração Saudável, (2018). *Dados estatísticos.* [online] Disponível em: http://coracaosaudavel.web.ua.pt/index.php/dados-estatisticos [visto a 14 de Janeiro 2018].

(Cortes et al., 1995) Cortes, C. & Vapnik, V. (1995). *Support-vector network*. *Machine Learning*, 20, 1–25.

(CUF, 2015) CUF, (2015). *Exercício físico & doenças cardiovasculares*. [online] Disponível em: https://www.saudecuf.pt/mais-saude/artigo/exercicio-fisico-doencas-cardiovasculares [visto a 13 de Janeiro 2018].

(CUF2, 2018) CUF, (2018). *Arritmias*. [online] Disponível em: https://www.saudecuf.pt/mais-saude/doencas-a-z/arritmias [visto a 14 de Janeiro 2018].

(CUF3, 2018) CUF, (2018). *Arritmias Cardiacas*. [online] Disponível em: https://www.saudecuf.pt/unidades/alvalade/areas-clinicas/cardiologia/sintomas-doencas-e-tratamentos/arritmias-cardiacas [visto a 14 de Janeiro 2018].

(Demsar, 2006) Janez Demsar. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7, pp.1-30.

(Domingos, 2012) Domingos P. (2012). *A Few Useful Things to Know about Machine Learning*. Communications of the ACM, Vol. 55 No. 10, Páginas 78-87.

(Gibson et al., 2017) Adam Gibson, Josh Patterson. (2017). *Deep Learning.* [pdf]. Disponível em: http://opencarts.org/sachlaptrinh/pdf/27976.pdf

(Glorot et al., 2011) Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. Em *Proceeding of the Conference on Artificial Intelligence and Statistics*.

(Gomes et al., 2014) E. F. Gomes, A. M. Jorge, and P. J. Azevedo. Classifying heart sounds using SAX motifs, random forests and text mining techniques. Em *18th International Database Engineering & Applications Symposium*, IDEAS 2014 (2014) July 7-9,Porto, Portugal, 334–337.

(Goodfellow et al., 2016) Ian Goodfellow, Yoshua Bengio, Aaron Courville. (2016). *Deep Learning*, [pdf] MIT Press. Disponível em: http://www.deeplearningbook.org.

(Han et al., 2012) Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* [pdf]. (pp. 83–124).

(Hossin et al., 2015) M Hossin and MN Sulaiman. (2015). Review on evaluation metrics for data classification evaluations. Em *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

(iClinic, 2014) IClinic, (2014). *O Poder da tecnologia aplicada à Saúde*. [online] Disponível em: http://blog.iclinic.com.br/o-poder-da-tecnologia-aplicada-a-saude [visto a 13 de Janeiro 2018].

(Ignatov, 2015) Ignatov Andrey Dmitrievich. (2015). *Deep Learning in information analysis of electrocardiogram signals for disease diagnostics* (Tese de Bacharelato). Moscow Institute of Physics and Technology, Moscovo, Rússia.

(Inan et al., 2006) O. T. Inan, L. Giovangrandi, G. T. A. Kovacs. (2016). *Robust neural network-based classification of premature ventricular contractions using wavelet transform and timing interval features*, IEEE Trans. Biomed. Eng., vol. 53, no. 12, pt. 1, pp. 2507–15.

(Isin et al., 2017) Ali Isin, Selen Ozdalili. (2017). *Cardiac arrhythmia detection using deep learning.* Procedia Computer Science 120, 268-275.

(Jama Network, 2010) Jama Network, (2010), *Atrial Fibrillation*. [online] Disponível em: https://jamanetwork.com/journals/jama/fullarticle/185278 [visto a 23 de Setembro 2018].

(Japkowicz et al., 2011) Japkowicz, N., & Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511921803

(Java Essentials, 2018) Oracle, (2018), *Java Essentials*. [online] Disponível em: http://www.oracle.com/technetwork/java/compile-136656.html [visto a 11 de Fevereiro 2018].

(Java History, 2018) Oracle, (2018). *The History of Java Technology*. [online] Disponível em: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html [visto a 11 de Fevereiro 2018].

(Jun et al., 2018) Tae Joon Jun, Hoang Minh Nguyen, Daeyoun Kang, Dohyeun Kim, Daeyoung Kim e Young-Hak Kim. (2018). ECG arrhythmia classification using a 2-D convolutional neural network.

(Junior S., 2008) Júnior S. (2008). *Detecção e Classificação de arritmias cardíacas utilizando redes neurais artificiais auto-organizáveis*. Mestrado. Universidade Católica do Paraná.

(Kampouraki et al., 2009) A. Kampouraki, G. Manis, and C. Nikou. (2009). Heartbeat time series classification with support vector machines. *Information Technology in Biomedicine*, IEEE Transactions em, 13(4):512–518.

(Kaya et al., 2015) Yasin Kaya, Hüseyin Pehlivan. (2015). Classification of Premature Ventricular Contraction in ECG. Em *International Journal of Advanced Computer Science and Applications(IJACSA)*, Volume 6 Issue 7, 2015.

(Kelwade et al., 2015) J. P. Kelwade, S. S. Salankar. (2015). Prediction of Cardiac Arrhythmia using Artificial Neural Network. *International Journal of Computer Applications,* Volume 115, 20.

(Koen et al., 2002) Peter A. Koen, Greg M. Ajamian, Scott Boyce, Allen Clamen, Eden Fisher, Stavros Fountoulakis, Albert Johnson, Pushpinder Puri, Rebecca Seibert. (2012). *Fuzzy Front End: Effective Methods, Tools and Techniques*.

(Kovalev et al., 2016) Vassili Kovalev, Alexander Kalinovsky, Sergey Kovalev. (2016). *Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?*.

(Krizhevsky et al., 2012) Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems, 1097–1105.

(Liu T, 2015) Tianyi Liu, Shuangsang Fang, Yuehui Zhao, Peng Wang, Jun Zhang. (2015). *Implementation of Training Convolutional Neural Networks*.

(Luz et al., 2016) Eduardo José da S.Luz, William Robson Schwartz, Guillermo Cámara-Chávez, David Menotti. (2016). *ECG-based heartbeat classification for arrhythmia detection: A survey*. Computer Methods and Programs in Biomedicine. Volume 127, Páginas 144-164.

(Luz E. et al., 2012) E.J.d.S. Luz, T.M. Nunes, V.H.C. De Albuquerque, J.P. Papa, D. Menotti. (2012). *ECG arrhythmia classification based on optimum-path forest.*

(Machine Learning Mistery, 2017) Machine Learning Mistery, (2017). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. [online] Disponível em: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ [visto a 23 de Setembro 2018]

(Mishra et al., 2018) Arvind K. Mishra, Silao V. Ramteke, Phalguni Sen e Amit Kumar Verma. (2018). Geotechnical and Geological Engineering. Páginas 36: 1647.

(Nicola et al., 2012) Susana Nicola, Eduarda Pinto Ferreira, J. J. Pinto Ferreira. (2012). A Novel framework for modeling value for the customer, an essay on negotiation. *International Journal of Information Technology & Decision Making*, 11:03, 661-703.

(OMS, 2017) OMS, (2017). *World Heart Day 2017*. [online] Disponível em: http://www.who.int/cardiovascular_diseases/world-heart-day-2017/en/ [visto a 13 de Janeiro 2018].

(Özbay et al., 2001) Özbay, Yüksel & Karlik, Bekir. (2001). *A Recognition of ECG Arrhythmias Using Artificial Neural Networks*. 5.

(PhysioNet, 2017) Physionet, (2017). *AF Classification from a short lead ECG recording: the PhysioNet/Computing in Cardiology Challenge 2017*. [online] Disponível em: https://www.physionet.org/challenge/2017/ [visto a 14 de Janeiro 2018].

(Pyakillya et al., 2017) B Pyakillya, N Kazachenko and N Mikhailovsky. (2017). *Deep Learning for ECG Classification*. IOP Conf. Series: Journal of Physics: Conf. Series 913

(Pyle D, 1999) D. Pyle. (1999). *Data Preparation for Data Mining*. California: Morgan Kaufmann Publishers.

(Python, 2018) Python Software Foundation, (2018), *Python*. [online] Disponível em: https://www.python.org/ [visto a 11 de Fevereiro 2018].

(Raeder T., 2008) Troy Raeder. (2008). *Model Monitor User's Guide version 1.0*. Department of Computer Science and Engineering, University of Notre Dame.

(Random Forests, 2016) Leo Breiman, Adele Cutler. (2016). *Random Forests*. [online] Disponível em: https://www.stat.berkeley.edu/~breiman/RandomForests/ [visto a 02 de fevereiro 2018].

(Saaty, 1991) Saaty, T. (1991). *Método de análise hierárquica*. São Paulo: McGraw-Hill.

(Samui et al., 2017) Pijush Samui, Sanjiban Sekhar Roy e Valentina E. Balas. (2017). *Handbook of Neural Computation*. (1ª ed.).

(Saravanan et al., 2014) Saravanan K and S. Sasithra. (2014). Review on Classification based on Artificial Neural Networks*. International Journal of Ambient Systems and Applications* (IJASA) Vol.2, No.4.

(Smolen, 2017) Dawid Smoleń. (2017). *Atrial Fibrillation Detection Using Boosting and Stacking Ensemble.*

(Song et al., 1996) Song, Y.H., Johns, A., Aggarwal, R. (2016). *Computational Intelligence Applications to Power Systems*, Science Press & Kluwer Academic Publishers, Beijing.

(Tamil et al., 2008) Emran Bin Mohd Tamil, N. H. Kamarudin, R. Salleh, A. M. Tamil. (2008). *A review on feature extraction & classification techniques for biosignal processing (Part I: Electrocardiogram)*.

(Tensorflow, 2018) Tensorflow, (2018). *Tensorflow*. [online] Disponível em: https://www.tensorflow.org/ [visto a 11 de Fevereiro 2018]

(Tiobe, 2018) Tiobe, (2018), *Tiobe*. [online] Disponível em: https://www.tiobe.com/tiobe-index/ [visto a 11 de Fevereiro 2017]

(Towards Data Science, 2018) Towards Data Science, (2018). *MNIST vs MNIST — how I was able to speed up my Deep Learning*. [online] Disponível em: https://towardsdatascience.com/mnist-vs-mnist-how-i-was-able-to-speed-up-my-deep-learning-11c0787e6935 [visto a 26 de Setembro 2018]

(Tweedale et al., 2012) Jeffrey Tweedale, Lakhmi C. Jain. (2012). Advanced Techniques for Knowledge Engineering and Innovative Applications*.* em *16th International Conference*, KES 2012, San Sebastian, Spain, Setembro 10-12.

(Ubeyli, 2008) E. D. Ubeyli. (2008). *Implementing wavelet transform/mixture of experts network for analysis of electrocardiogram beats*.

(Ulaga e Eggert, 2006) Ulaga W, Eggert A. (2006) Value-based differentiation in business relationships: Gaining and sustaining key supplier status. *Journal of Marketing*: January 2006, Vol. 70, No. 1, pp. 119-136.

(Warrick et al., 2017) Philip Warrick, Masun Nabhan Homsi. (2017). *Cardiac Arrhythmia Detection from ECG Combining Convolutional and Long Short-Term Memory Networks*.

(Woodall, 2003) Woodall T. (2003). *Conceptualising 'Value for the Customer' : An Attributional, Structural and Dispositional Analysis*.

(Xenon Stack, 2017) Jagreet Kaur Gill. (2017). *Overview of Artificial Neural Networks and its Applications*. [online] Disponível em: https://www.xenonstack.com/blog/data-science/overview-of-artificial-neural-networks-and-its-applications [visto a 05 Fevereiro 2018].

(Xiong et al., 2017) Zhaohan Xiong, Martin K. Stiles, Jichao Zhao. (2017). *Robust ECG Signal Classification for Detection of Atrial Fibrillation Using a Novel Neural Network.*

(Zabihi et al., 2017) Morteza Zabihi, Ali Bahrami Rad, Aggelos K. Katsaggelos, Serkan Kiranyaz, Susanna Narkilahti, Moncef Gabbouj. (2017). *Detection of Atrial Fibrillation in ECG Hand-held Devices Using a Random Forest Classifier*.

(Zihlmann et al., 2017) Martin Zihlmann, Dmytro Perekrestenko, Michael Tschannen, (2017). *Convolutional Recurrent Neural Networks for Electrocardiogram Classification.*

Anexos

Anexo I

Resultados da 1ª abordagem para classificação multiclasse - 1 época

Exemplos	**Métricas**	***Adam Optimizer***				***Gradient Descent Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	**0.635**	0.320	0.255	0.320	0.275	0.250	0.250	0.255
	AUC	**0.757**	0.547	0.503	0.547	0.517	0.500	0.500	0.503
	AVG *F-Measure*	**0.621**	0.249	0.113	0.238	0.239	0.100	0.100	0.109
	F-Measure AF	**0.674**	0.331	0.169	0.316	0.359	0.320	0.000	0.000
400 (100 cada classe)	**Taxa de Acerto**	**0.480**	0.343	0.248	0.313	0.275	0.248	0.250	0.258
	AUC	**0.653**	0.562	0.498	0.542	0.517	0.498	0.500	0.505
	AVG *F-Measure*	**0.445**	0.288	0.099	0.229	0.247	0.100	0.100	0.110
	F-Measure AF	**0.490**	0.341	0.120	0.299	0.246	0.000	0.400	0.356
800 (200 cada classe)	**Taxa de Acerto**	**0.381**	0.341	0.260	0.333	0.253	0.250	0.246	0.250
	AUC	**0.588**	0.561	0.507	0.555	0.502	0.500	0.498	0.500
	AVG *F-Measure*	**0.325**	0.263	0.119	0.275	0.197	0.100	0.103	0.100
	F-Measure AF	**0.454**	0.389	0.356	0.430	0.245	0.400	0.008	0.400
1200 (300 cada classe)	**Taxa de Acerto**	**0.315**	0.289	0.254	0.282	0.274	0.290	0.248	0.237
	AUC	**0.542**	0.525	0.499	0.521	0.522	0.525	0.496	0.500
	AVG *F-Measure*	**0.230**	0.199	0.104	0.184	0.198	0.200	0.140	0.096
	F-Measure AF	0.369	**0.391**	0.201	0.340	0.258	0.292	0.285	0.000

Resultados da 1ª abordagem para classificação multiclasse - 5 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	**0.840**	0.570	0.295	0.565	0.360	0.250	0.240	0.275
	AUC	**0.893**	0.713	0.530	0.710	0.573	0.500	0.493	0.517
	AVG *F-Measure*	**0.833**	0.528	0.198	0.526	0.352	0.100	0.165	0.146
	F-Measure AF	**0.791**	0.504	0.351	0.579	0.418	0.120	0.033	0.183
400 (100 cada classe)	**Taxa de Acerto**	**0.790**	0.530	0.295	0.578	0.338	0.250	0.240	0.250
	AUC	**0.860**	0.687	0.530	0.718	0.558	0.500	0.493	0.500
	AVG *F-Measure*	**0.790**	0.515	0.200	0.556	0.298	0.100	0.122	0.100
	F-Measure AF	**0.788**	0.567	0.367	0.612	0.291	0.400	0.053	0.400
800 (200 cada classe)	**Taxa de Acerto**	0.493	**0.611**	0.283	0.610	0.313	0.263	0.269	0.251
	AUC	0.662	**0.741**	0.522	0.740	0.542	0.508	0.513	0.501
	AVG *F-Measure*	0.464	**0.598**	0.175	0.594	0.223	0.137	0.185	0.103
	F-Measure AF	0.528	0.616	0.402	**0.650**	0.304	0.210	0.141	0.000
1200 (300 cada classe)	**Taxa de Acerto**	0.313	**0.503**	0.256	0.500	0.311	0.232	0.254	0.268
	AUC	0.540	**0.668**	0.501	0.666	0.547	0.495	0.500	0.518
	AVG *F-Measure*	0.245	**0.485**	0.117	0.476	0.235	0.099	0.159	0.145
	F-Measure AF	0.421	**0.551**	0.315	0.539	0.329	0.050	0.311	0.144

Anexo II

Resultados da 1ª abordagem para classificação binária - 1 época

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	**0.820**	0.550	0.500	0.520	0.610	0.500	0.500	0.500
	AUC	**0.820**	0.550	0.500	0.520	0.610	0.500	0.500	0.500
	AVG *F-Measure*	**0.802**	0.470	0.333	0.383	0.580	0.333	0.333	0.333
	F-Measure AF	**0.774**	0.469	0.333	0.368	0.631	0.667	0.667	0.667
200 (100 cada classe)	**Taxa de Acerto**	**0.735**	0.510	0.530	0.545	0.495	0.500	0.500	0.500
	AUC	**0.735**	0.510	0.530	0.545	0.495	0.500	0.500	0.500
	AVG *F-Measure*	**0.684**	0.393	0.385	0.436	0.457	0.333	0.333	0.333
	F-Measure AF	**0.693**	0.432	0.425	0.495	0.447	0.067	0.000	0.667
400 (200 cada classe)	**Taxa de Acerto**	**0.655**	0.555	0.515	0.545	0.508	0.500	0.500	0.500
	AUC	**0.655**	0.555	0.515	0.545	0.508	0.500	0.500	0.500
	AVG *F-Measure*	**0.619**	0.478	0.366	0.456	0.474	0.341	0.333	0.333
	F-Measure AF	0.665	0.571	0.328	0.467	0.603	0.664	0.000	**0.667**
600 (300 cada classe)	**Taxa de Acerto**	**0.643**	0.525	0.500	0.538	0.522	0.500	0.522	0.500
	AUC	**0.643**	0.525	0.500	0.538	0.522	0.500	0.522	0.500
	AVG *F-Measure*	**0.606**	0.463	0.333	0.463	0.447	0.333	0.409	0.333
	F-Measure AF	**0.677**	0.606	0.600	0.606	0.642	0.000	0.293	0.667
1380 (690 cada classe)	**Taxa de Acerto**	0.506	0.509	0.504	**0.513**	0.500	0.500	0.501	0.500
	AUC	0.506	0.509	0.504	**0.513**	0.500	0.500	0.501	0.500
	AVG *F-Measure*	0.354	0.360	0.353	**0.364**	0.340	0.333	0.344	0.333
	F-Measure AF	0.630	0.669	0.515	**0.672**	0.665	0.667	0.479	0.667

Resultados da 1ª abordagem para classificação binária - 5 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	**0.920**	0.770	0.600	0.710	0.550	0.500	0.500	0.500
	AUC	**0.920**	0.770	0.600	0.710	0.550	0.500	0.500	0.500
	AVG *F-Measure*	**0.915**	0.765	0.508	0.684	0.520	0.333	0.345	0.333
	F-Measure AF	**0.918**	0.748	0.599	0.708	0.475	0.667	0.029	0.667
200 (100 cada classe)	**Taxa de Acerto**	**0.930**	0.695	0.555	0.675	0.540	0.525	0.505	0.500
	AUC	**0.930**	0.695	0.555	0.675	0.540	0.525	0.505	0.500
	AVG *F-Measure*	**0.927**	0.681	0.454	0.652	0.510	0.437	0.436	0.333
	F-Measure AF	**0.941**	0.651	0.483	0.619	0.591	0.576	0.347	0.667
400 (200 cada classe)	**Taxa de Acerto**	**0.800**	0.725	0.525	0.720	0.503	0.500	0.498	0.508
	AUC	**0.800**	0.725	0.525	0.720	0.503	0.500	0.498	0.508
	AVG *F-Measure*	**0.798**	0.703	0.420	0.697	0.466	0.333	0.354	0.349
	F-Measure AF	**0.793**	0.766	0.411	0.762	0.594	0.667	0.657	0.670
600 (300 cada classe)	**Taxa de Acerto**	**0.827**	0.752	0.522	0.732	0.492	0.500	0.502	0.500
	AUC	**0.827**	0.752	0.522	0.732	0.492	0.500	0.502	0.500
	AVG *F-Measure*	**0.825**	0.747	0.383	0.730	0.355	0.336	0.373	0.333
	F-Measure AF	**0.839**	0.778	0.508	0.740	0.651	0.466	0.652	0.667
1380 (690 cada classe)	**Taxa de Acerto**	0.510	0.616	0.501	**0.685**	0.499	0.501	0.500	0.500
	AUC	0.510	0.616	0.501	**0.685**	0.499	0.501	0.500	0.500
	AVG *F-Measure*	0.355	0.559	0.340	**0.633**	0.339	0.345	0.333	0.333
	F-Measure AF	0.625	0.706	0.415	**0.758**	0.664	0.665	0.667	0.667

Anexo III

Resultados da 2ª abordagem para classificação multiclasse - 5 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	0.475	**0.730**	0.250	0.670	0.255	0.250	0.250	0.250
	AUC	0.650	**0.820**	0.500	0.780	0.503	0.500	0.500	0.500
	AVG *F-Measure*	0.441	**0.699**	0.100	0.632	0.171	0.100	0.100	0.100
	F-Measure AF	0.543	0.722	0.240	**0.732**	0.113	0.400	0.400	0.400
400 (100 cada classe)	**Taxa de Acerto**	0.350	0.728	0.250	**0.800**	0.238	0.250	0.260	0.250
	AUC	0.567	0.818	0.500	**0.867**	0.492	0.500	0.507	0.500
	AVG *F-Measure*	0.291	0.700	0.100	**0.780**	0.110	0.100	0.116	0.100
	F-Measure AF	0.367	0.782	0.160	**0.818**	0.037	0.400	0.107	0.400
800 (200 cada classe)	**Taxa de Acerto**	0.401	0.613	0.250	**0.734**	0.250	0.250	0.255	0.250
	AUC	0.601	0.742	0.500	**0.823**	0.500	0.500	0.503	0.500
	AVG *F-Measure*	0.329	0.564	0.100	**0.708**	0.100	0.100	0.110	0.100
	F-Measure AF	0.417	0.647	0.160	**0.750**	0.000	0.400	0.122	0.400
1200 (300 cada classe)	**Taxa de Acerto**	0.259	0.529	0.255	**0.650**	0.237	0.250	0.250	0.250
	AUC	0.503	0.686	0.500	**0.767**	0.500	0.500	0.500	0.500
	AVG *F-Measure*	0.115	0.483	0.101	**0.618**	0.096	0.100	0.100	0.100
	F-Measure AF	0.411	0.549	0.325	**0.692**	0.000	0.400	0.400	0.400

Anexo IV

Resultados da 2ª abordagem para classificação binária - 5 épocas

Exemplos	Métricas	*Adam Optimizer*				*Gradient Descent Optimizer*			
		Tanh	***Relu***	***Sigmoid***	***Softplus***	***Tanh***	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	**0.830**	0.760	0.500	0.730	0.520	0.500	0.500	0.500
	AUC	**0.830**	0.760	0.500	0.730	0.520	0.500	0.500	0.500
	AVG *F-Measure*	**0.823**	0.715	0.333	0.679	0.460	0.333	0.333	0.333
	F-Measure AF	**0.847**	0.723	0.533	0.714	0.499	0.667	0.667	0.667
400 (100 cada classe)	**Taxa de Acerto**	0.635	0.760	0.500	**0.805**	0.525	0.500	0.500	0.500
	AUC	0.635	0.760	0.500	**0.805**	0.525	0.500	0.500	0.500
	AVG *F-Measure*	0.607	0.729	0.333	**0.784**	0.400	0.333	0.348	0.333
	F-Measure AF	0.696	0.774	0.667	**0.807**	0.635	0.667	0.658	0.667
800 (200 cada classe)	**Taxa de Acerto**	0.553	**0.835**	0.500	0.763	0.500	0.500	0.500	0.500
	AUC	0.553	**0.835**	0.500	0.763	0.500	0.500	0.500	0.500
	AVG *F-Measure*	0.524	**0.832**	0.333	0.720	0.338	0.333	0.333	0.333
	F-Measure AF	0.601	**0.850**	0.600	0.804	0.666	0.667	0.667	0.667
1200 (300 cada classe)	**Taxa de Acerto**	0.553	0.720	0.500	**0.780**	0.500	0.500	0.500	0.500
	AUC	0.553	0.720	0.500	**0.780**	0.500	0.500	0.500	0.500
	AVG *F-Measure*	0.531	0.675	0.333	**0.748**	0.338	0.333	0.333	0.333
	F-Measure AF	0.600	0.783	0.533	**0.823**	0.665	0.667	0.667	0.667

Anexo V

Resultados da 2ª abordagem para classificação multiclasse (exemplos de 30 segundos) - 5 épocas

Exemplos	**Métricas**	***Adam Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
200 (50 cada classe)	**Taxa de Acerto**	0.460	0.710	0.270	**0.745**
	AUC	0.640	0.807	0.513	**0.830**
	AVG *F-Measure*	0.407	0.676	0.127	**0.714**
	F-Measure AF	0.363	0.715	0.211	**0.726**
400 (100 cada classe)	**Taxa de Acerto**	0.395	**0.775**	0.250	0.765
	AUC	0.597	**0.850**	0.500	0.843
	AVG *F-Measure*	0.332	**0.751**	0.100	0.739
	F-Measure AF	0.443	**0.806**	0.080	0.797
800 (200 cada classe)	**Taxa de Acerto**	0.342	0.725	0.277	**0.766**
	AUC	0.551	0.809	0.500	**0.837**
	AVG *F-Measure*	0.279	0.703	0.108	**0.741**
	F-Measure AF	0.450	0.776	0.087	**0.804**
1200 (300 cada classe)	**Taxa de Acerto**	0.309	0.652	0.293	**0.675**
	AUC	0.509	0.750	0.500	**0.766**
	AVG *F-Measure*	0.172	0.590	0.113	**0.628**
	F-Measure AF	0.470	0.727	0.000	**0.735**

Anexo VI

Resultados da 2ª abordagem para classificação binária (exemplos de 30 segundos) - 5 épocas

Exemplos	**Métricas**	***Adam Optimizer***			
		Tanh	***Relu***	***Sigmoid***	***Softplus***
100 (50 cada classe)	**Taxa de Acerto**	0.820	**0.850**	0.500	0.810
	AUC	0.820	**0.850**	0.500	0.810
	AVG *F-Measure*	0.808	**0.825**	0.333	0.787
	F-Measure AF	**0.820**	0.802	0.400	0.777
200 (100 cada classe)	**Taxa de Acerto**	0.685	0.825	0.500	**0.830**
	AUC	0.685	0.825	0.500	**0.830**
	AVG *F-Measure*	0.664	**0.806**	0.333	0.795
	F-Measure AF	0.726	0.825	0.467	**0.869**
400 (200 cada classe)	**Taxa de Acerto**	0.573	0.810	0.500	**0.838**
	AUC	0.573	0.810	0.500	**0.838**
	AVG *F-Measure*	0.554	0.791	0.333	**0.813**
	F-Measure AF	0.638	0.839	0.400	**0.851**
600 (300 cada classe)	**Taxa de Acerto**	0.557	**0.830**	0.500	0.802
	AUC	0.557	**0.830**	0.500	0.802
	AVG *F-Measure*	0.506	**0.800**	0.333	0.778
	F-Measure AF	0.648	**0.872**	0.533	0.837