

APRENDIZADO FEDERADO APLICADO À DETECÇÃO DE
ARRITMIAS EM ELETROCARDIOGRAMAS

Lucas Costa Favaro

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Rodrigo de Souza Couto

Rio de Janeiro

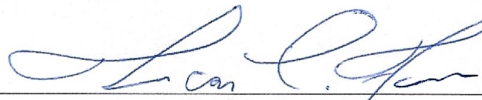
Agosto de 2022

APRENDIZADO FEDERADO APLICADO À DETECÇÃO DE
ARRITMIAS EM ELETROCARDIOGRAMAS

Lucas Costa Favaro

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO
DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA PO-
LITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:



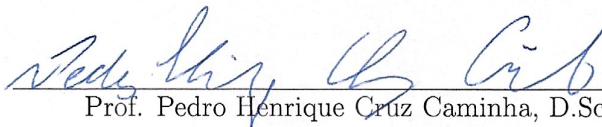
Lucas Costa Favaro

Orientador:



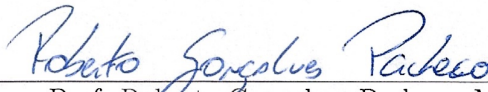
Prof. Rodrigo de Souza Couto, D.Sc.

Examinador:



Prof. Pedro Henrique Cruz Caminha, D.Sc.

Examinador:



Prof. Roberto Gonçalves Pacheco, M.Sc.

Rio de Janeiro
Agosto de 2022

Declaração de Autoria e de Direitos

Eu, *Lucas Costa Favaro* CPF 169.281.627-67, autor da monografia *Aprendizado Federado Aplicado à Detecção de Arritmias em Eletrocardiogramas*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.



Lucas Costa Favaro

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica - Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

AGRADECIMENTO

Em primeiro lugar, agradeço aos meus pais Wilson e Gilvanice por todo suporte e apoio, e à minha família por todo apoio mútuo.

Aos amigos que fiz na UFRJ, Matheus, Pedro e Paulo, por todos os “almocinhos” e por compartilharem essa jornada comigo.

Agradeço aos meus amigos do ensino médio Andrey, Ariel, Beatriz, Bruna, Clara, João Farney, João Marcio, João Moreira e Joyce por sempre estarem ao meu lado, e aos meus professores do ensino médio por fornecerem a base da minha formação, em especial ao “coach” Guilherme.

Agradeço também a todos os professores da UFRJ que participaram da minha formação, em especial ao meu orientador Rodrigo pela disponibilidade e paciência.

RESUMO

O aprendizado federado é uma abordagem que permite que modelos baseados em aprendizado de máquina sejam desenvolvidos sem a necessidade da centralização do *dataset*, garantindo privacidade e segurança aos clientes que cedem esses dados. Essa abordagem, por garantir que os dados continuam privados, pode ser útil em aplicações que envolvem dados médicos, como eletrocardiogramas utilizados na identificação de arritmias. Este trabalho analisa a viabilidade do aprendizado federado em um modelo distribuído de detecção de arritmias baseado em redes neurais convolucionais (CNNs - *Convolutional Neural Networks*). O modelo utilizado foi baseado nas implementações da literatura, com realização de testes nas versões centralizada e distribuída. O conjunto de dados MIT-BIH foi utilizado para treinamento e teste do modelo. Para a análise, este trabalho propõe uma nova divisão do conjunto de dados, baseada em uma divisão apresentada na literatura. Tal divisão visa respeitar as recomendações da Associação para o Avanço da Instrumentação Médica (AAMI) e fornecer mais dados para a fase de treinamento. O aprendizado federado foi incorporado ao modelo utilizando o *framework* Flower e utilizou-se o FederatedAveraging para agregação dos parâmetros dos clientes nos casos distribuídos. O modelo distribuído alcançou um *recall* de 83,14% e precisão de 58,47% no caso ideal. No caso em que os dados apresentam distribuição não independente e não identicamente distribuída (não-IID), o modelo alcançou um *recall* de 85,1% e precisão de 53,03%. Em comparação, o modelo centralizado obteve *recall* de 82,01% e precisão de 57,34%. Esses resultados indicam que o aprendizado federado é uma opção viável para o desenvolvimento de um modelo de detecção de arritmias em eletrocardiogramas.

Palavras-Chave: aprendizado federado, redes neurais, redes neurais convolucionais, arritmia, eletrocardiograma, FederatedAveraging.

ABSTRACT

Federated learning is an approach that allows machine learning models to be developed without the dependency on centralized data, ensuring privacy and security for the clients that own the data. This approach can be very useful in applications based on medical data such as electrocardiograms used to identify arrhythmias because it ensures privacy. The present work analyses the feasibility of utilizing federated learning in a distributed model of arrhythmia detection based on Convolutional Neural Networks (CNNs). The model was based on implementations present in the literature and experiments were performed with both the centralized and distributed versions. The MIT-BIH dataset was used for training and testing those versions. This work proposes a new division of the dataset, based on another division present in the literature. Such division is intended to follow the recommendations from the Association for the Advancement of Medical Instrumentation (AAMI) and provide more data to the training dataset. Federated Learning was embedded in the model using the Flower framework. FederatedAveraging was utilized as the algorithm to aggregate the parameters on the distributed setting. The distributed model achieved a recall of 83.14% and a precision of 58.47% in the ideal scenario. Considering the non-independent and non-identically distributed (non-IID) scenario, the model achieved a recall of 85.1% and precision of 53.03%. For comparison, the centralized model achieved a recall of 82.01% and a precision of 57.34%. These results indicate that Federated Learning is a feasible option for the development of a distributed model of arrhythmia detection.

Key-words: federated learning, neural networks, convolutional neural networks, arrhythmia, electrocardiogram, FederatedAveraging.

SIGLAS

UFRJ - Universidade Federal do Rio de Janeiro

FL - Federated Learning

IoT - Internet of Things

IID - Independente e Identicamente Distribuídos

AAMI - Associação para Avanço na Instrumentação Médica

MIT - Massachusetts Institute of Technology

BIH - Boston's Beth Israel Hospital

AHA - American Heart Association

SVM - Support Vector Machine

MLP - Multilayer Perceptron

CNN - Convolutional Neural Network

FedAvg - Federated Averaging

GD - Gradiente Descendente

SGD - Stochastic Gradient Descent

ECG - Eletrocardiograma

ReLU - Rectified Linear Unit

Sumário

1	Introdução	1
1.1	Objetivos e Delimitação	3
1.2	Organização do Texto	4
2	Redes Neurais e Aprendizado Federado	5
2.1	Redes Neurais	5
2.1.1	Processo de Aprendizado	7
2.1.2	Gradiente Descendente Estocástico - SGD	9
2.1.3	Regularização	11
2.2	Redes Neurais Convolucionais	15
2.3	Aprendizado Federado	19
3	Trabalhos Relacionados	23
3.1	Bases de dados e Classificadores Tradicionais	24
3.2	Modelos baseados em CNN	25
3.3	Redes com Aprendizado Federado	26
4	Metodologia	29
4.1	Bibliotecas e Frameworks	29
4.2	Conjunto de Dados	29
4.2.1	Recomendações da AAMI	30
4.2.2	Classificação	31
4.2.3	Geração e tratamento dos dados	32
4.2.4	Divisões dos Conjuntos de Treino e Teste	33
4.3	Arquitetura da Rede e Definição de Parâmetros	35
4.4	Modelos Distribuídos	36

4.4.1	Dados IID	36
4.4.2	Dados Não-IID	37
4.5	Critérios de Avaliação	38
5	Experimentos e Resultados	39
5.1	Modelo Centralizado	39
5.2	Aprendizado Federado	40
5.2.1	Caso Ideal	41
5.2.2	Caso Não-IID	43
6	Conclusão	48
	Bibliografia	50

Lista de Figuras

2.1	Representação de um neurônio.	6
2.2	Representação simplificada de um neurônio.	6
2.3	Exemplo de Rede Neural com camada oculta, representada na cor azul. . .	7
2.4	Fluxo de propagação da informação. Em azul mostra-se a fase de <i>forward propagation</i> . Em vermelho mostra-se a fase de <i>backward propagation</i> . . .	8
2.5	Efeito do aumento na variável θ quando a inclinação é positiva	10
2.6	Exemplo de overfitting.	13
2.7	Exemplo de camada com <i>dropout</i> de dois neurônios.	14
2.8	Exemplo de operação de convolução em 1 dimensão.	16
2.9	Exemplo de funcionamento de uma camada convolucional em uma rede neural.	17
2.10	Operação de <i>pooling</i>	18
2.11	Arquitetura da rede proposta.	18
2.12	Modelo centralizado (à esquerda) e modelo federado (à direita).	20
4.1	Fluxograma simplificado das etapas seguidas no trabalho.	30
4.2	Representação do complexo QRS.	31
4.3	Divisão dos batimentos entre as classes.	33
5.1	Recall obtido nos experimentos do modelo centralizado, para diferentes tamanhos de <i>batch</i>	40
5.2	Recall obtido nos experimentos do cenário IID, para diferentes tamanhos de <i>batch</i>	41
5.3	Precisão obtida nos experimentos do cenário IID, para diferentes tamanhos de <i>batch</i>	43

5.4	Resultado da função perda nos experimentos do cenário IID, para diferentes tamanhos de <i>batch</i>	44
5.5	Recall obtido nos experimentos do cenário não-IID, para diferentes tamanhos de <i>batch</i>	45
5.6	Precisão obtida nos experimentos do cenário não-IID, para diferentes tamanhos de <i>batch</i>	46
5.7	Resultado da função perda nos experimentos do cenário não-IID, para diferentes tamanhos de <i>batch</i>	47
5.8	Experimento com <i>oversampling</i> reduzido do caso não-IID, para <i>batch</i> de tamanho 512 e uma época local.	47

Lista de Tabelas

3.1	Trabalhos Relacionados envolvendo Aprendizado Federado	28
4.1	Mapeamento Classe AAMI x Tipos MIT-BIH.	32
4.2	Distribuição das Gravações entre os conjuntos DS1 e DS2	34
4.3	Distribuição das Classes entre os conjuntos DS1 e DS2	34
4.4	Distribuição das Gravações entre os conjuntos DS3 e DS4	34
4.5	Distribuição das Classes entre os conjuntos DS3 e DS4	35
4.6	Distribuição de amostras apresentando arritmias entre os clientes . .	37
4.7	Distribuição de amostras apresentando arritmias entre os clientes . .	37
5.1	Resultado dos experimentos para o modelo centralizado.	40

Capítulo 1

Introdução

Nas últimas décadas, com o aumento da capacidade computacional e a maior facilidade de acesso a máquinas mais potentes, técnicas de aprendizado profundo se tornaram cada vez mais populares [1]. Sistemas baseados em aprendizado de máquina se tornaram parte do cotidiano de diversas indústrias e passaram a ser tema central do discurso público, com governos ao redor do mundo investindo cada vez mais em programas para explorar a tecnologia [2]. De forma geral, para que um modelo baseado em IA, em especial aprendizado profundo, consiga ter o desempenho esperado é necessário que haja uma grande quantidade de dados com qualidade disponíveis.

Com a proliferação de dispositivos móveis e de dispositivos de Internet das Coisas, uma quantidade sem precedentes de dados é gerada todos os dias ao redor do mundo [3]. Esses dados podem ser utilizados por empresas e pesquisadores para a criação de modelos de aprendizado de máquina. Contudo, muitas vezes os dados podem ter uma natureza sensível ou privada, como dados médicos ou dados pessoais como chamadas de voz. Isso acaba levando a uma resistência dos clientes em ceder seus dados para o treinamento e aprimoramento de tais modelos.

A centralização de dados é uma abordagem na qual os dados dos clientes são enviados para um servidor central. O servidor utiliza os dados para o treinamento de um modelo baseado em aprendizado de máquina. Esses modelos centralizados tem acesso aos dados de todos os clientes, o que pode ser um problema no caso de dados privados e sensíveis. Tendo isto em vista, diversas técnicas de aprendizado distribuído começaram a ser desenvolvidas de forma a mitigar os problemas causados

pela centralização dos dados.

Modelos distribuídos são modelos nos quais os dados dos clientes não são enviados diretamente para o servidor central. Em geral, cada cliente realiza computações localmente e apenas o resultado dessas computações é enviado ao servidor. Em um artigo de 2016, Ramage *et al.* propuseram o Aprendizado Federado [4]. Essa abordagem mantém os dados distribuídos nos dispositivos dos clientes e utiliza um modelo compartilhado que agrega informações em um servidor centralizado. No aprendizado federado, cada cliente recebe do servidor a versão atual dos parâmetros do modelo global e os aprimora localmente utilizando seus dados para treinamento. Em seguida, os clientes enviam os parâmetros dos modelos locais obtidos para o servidor, que os agrega e gera um modelo global compartilhado, aumentando a privacidade e a segurança dos modelos baseados em aprendizado profundo.

A detecção de arritmias cardíacas é uma área que pode se beneficiar do aprendizado federado [5, 6]. Uma arritmia é uma alteração na frequência cardíaca que pode indicar doenças como infarto do miocárdio, taquicardia ventricular e fibrilação atrial. Essas alterações geralmente são constatadas por profissionais altamente especializados por meio da análise de eletrocardiogramas. Algumas características presentes nos eletrocardiogramas, como a amplitude e duração de um batimento, deformações no formato da onda e o intervalo entre os batimentos, indicam a presença de arritmias. Essas arritmias podem ser identificadas por meio de redes neurais [7, 8].

Como dados médicos são privados, há o desafio de garantir a segurança e a privacidade dos dados fornecidos pelos clientes, e, ao mesmo tempo, fornecer dados suficientes e com qualidade para o treinamento de um modelo. Trabalhos recentes sugerem o aprendizado federado como alternativa viável para a detecção de arritmias no cenário em que os dados são mantidos privados [5, 6].

Com a adoção do aprendizado federado, os dados ficam distribuídos entre os clientes. Nos casos práticos, essa distribuição é diferente para cada cliente. Nesse cenário, os dados podem não ser independentes e identicamente distribuídos (não-IID) entre os clientes. Em outras palavras, cada subconjunto de dados pode ter uma distribuição de probabilidades diferente dos demais e alguma dependência em relação a algum outro subconjunto. O cenário onde os dados são não-IID representa

um desafio para a utilização do aprendizado federado. Nesse caso, pode ocorrer uma perda de desempenho caso o algoritmo de agregação dos parâmetros não leve em consideração a diferença na distribuição dos dados entre os clientes [6, 9]. Esse contexto é importante uma vez que a incidência de arritmias em um conjunto de dados contendo diferentes populações tende a ser não-IID.

Um modelo de detecção de arritmias baseado em aprendizado federado pode ser uma ferramenta importante para auxiliar no diagnóstico de doenças cardíacas e acelerar a identificação de pacientes que necessitam de atendimento especializado. Devido as vantagens do aprendizado federado, dados de diferentes hospitais poderiam ser utilizados para o aprimoramento de tal modelo sem que haja o compartilhamento de dados médicos dos pacientes.

Este projeto busca analisar a viabilidade do aprendizado federado aplicado a um modelo de detecção de arritmias em eletrocardiogramas. Para tal, analisam-se as principais vantagens e desafios do uso de tal abordagem, explorando diferentes cenários de distribuição de dados entre os clientes.

1.1 Objetivos e Delimitação

O objetivo geral deste projeto é analisar a viabilidade de um modelo computacional baseado em aprendizado federado para identificação de arritmias em eletrocardiogramas. Dessa forma, tem-se como objetivos específicos: (1) analisar e explorar as vantagens e desafios do aprendizado federado; (2) analisar o desempenho do modelo distribuído em comparação ao modelo centralizado; (3) analisar o desempenho do modelo distribuído no cenário em que os dados não são independentes e identicamente distribuídos entre os clientes (não-IID), apontando possíveis melhorias no modelo.

O modelo para detecção de arritmias utilizado neste trabalho foi desenvolvido com base nos modelos de redes neurais convolucionais (CNNs - *Convolutional Neural Networks*) propostos por Li *et al.* [7], Hannun *et al.* [8] e Gao *et al.* [10]. Para tal, este trabalho se baseia nas recomendações da Associação para Avanço na Instrumentação Médica (AAMI) para o desenvolvimento de modelos computacionais para classificação de arritmias [11].

A base de dados de Arritmias MIT-BIH (MIT-BIH *Arrhythmia Database*) [12] foi utilizada para treinamento e avaliação do modelo, sendo proposta uma nova divisão do conjunto de dados, de forma a evitar que os dados de um mesmo paciente sejam utilizados para treinamento e teste ao mesmo tempo. Essa divisão foi concebida baseada na divisão apresentada por Chazal *et al.* [13], que mantém as distribuições de arritmias iguais entre os conjuntos e aloca os dados de cada paciente apenas para treinamento ou para teste.

O modelo distribuído foi implementado utilizando o algoritmo FederatedAveraging por meio do *framework* Flower, de forma a simplificar a implantação do aprendizado federado [14]. Destaca-se que o modelo proposto neste trabalho considera condições ideais de rede, nas quais todos os clientes sempre estão disponíveis ao servidor e não ocorrem falhas na comunicação.

Para o caso ideal, onde os dados são independentes e identicamente distribuídos (IID), o modelo baseado em aprendizado federado apresentou *recall* de 83.14% e precisão de 58.47%. No cenário não-IID, o *recall* foi de 85.1% e a precisão de 53.03%. O modelo centralizado, utilizado para comparação, apresentou *recall* de 82.01% e precisão de 57.34%.

1.2 Organização do Texto

O Capítulo 2 expõe de maneira resumida os conceitos teóricos necessários para o desenvolvimento e entendimento do projeto. O Capítulo 3 apresenta os principais trabalhos relacionados, de modelos centralizados a modelos distribuídos que utilizam aprendizado federado. Esse capítulo destaca as principais contribuições de cada trabalho, analisando fatores comuns entre eles e destacando as principais diferenças. A metodologia utilizada neste trabalho é apresentada detalhadamente no Capítulo 4, expondo as etapas seguidas desde a preparação dos dados até os experimentos envolvendo modelos baseados em aprendizado federado. O Capítulo 5 expõe os resultados obtidos nos experimentos, discutindo seus significados e importância. Por fim, o Capítulo 6 traz as considerações finais e indica possíveis trabalhos futuros.

Capítulo 2

Redes Neurais e Aprendizado Federado

Neste capítulo são introduzidos conceitos básicos utilizados no desenvolvimento deste trabalho.

2.1 Redes Neurais

De maneira geral, redes neurais são estruturas de processamento formadas por unidades interconectadas, chamadas de neurônios, capazes de aprender com o ambiente. Tais estruturas são baseadas nas redes neurais biológicas, o sistema nervoso dos animais, e por este motivo também são conhecidas como redes neurais artificiais [15]. Sobre essa analogia com as estruturas biológicas, Haykin [16] destaca que o conhecimento é armazenado como o peso das conexões entre os neurônios, Além disso, esse conhecimento é obtido do ambiente por meio de um processo de aprendizado, que se dá por meio de um algoritmo.

A Figura 2.1 mostra a representação gráfica de um neurônio. A saída de um neurônio i qualquer pode ser representada matematicamente por meio da equação

$$y_i = \varphi(u_i) = \varphi\left(\sum_{j=1}^n w_{ij}x_j\right). \quad (2.1)$$

A função $\varphi(\cdot)$ é chamada de função de ativação e é responsável por limitar a amplitude da saída e definir seu valor. Os parâmetros w_{ij} são os pesos das conexões

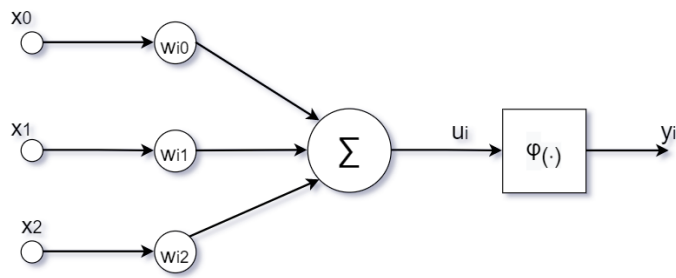


Figura 2.1: Representação de um neurônio.

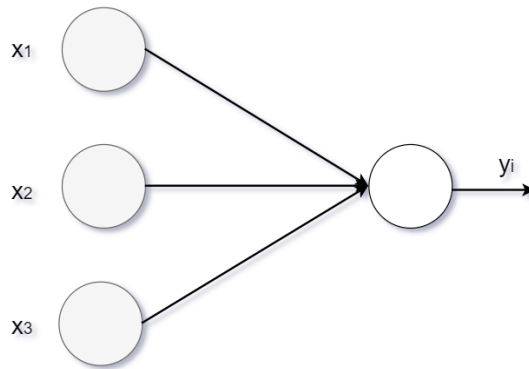


Figura 2.2: Representação simplificada de um neurônio.

que multiplicam as entradas x_j , que podem ser a saída de outros neurônios ou uma entrada vinda do ambiente (isto, é dados fornecidos à rede neural) [16].

A Figura 2.2 traz uma representação simplificada de uma rede neural. Na figura, os pesos estão implícitos nas conexões, e o somatório e a função de ativação implícitas na saída.

Redes neurais são formadas por camadas conectadas. Uma camada é um conjunto de neurônios semelhantes. A camada que recebe informação do ambiente é chamada de camada de entrada e a camada que retorna o resultado da rede é chamada de camada de saída. As camadas que não têm contato com a entrada ou com a saída são chamadas de camadas ocultas. As redes de múltiplas camadas, e sem *feedback* (ou seja, quando a entrada de um neurônio está ligada a sua saída), são chamadas de MLP (*Multilayer Perceptron*). Esse nome é uma referência ao

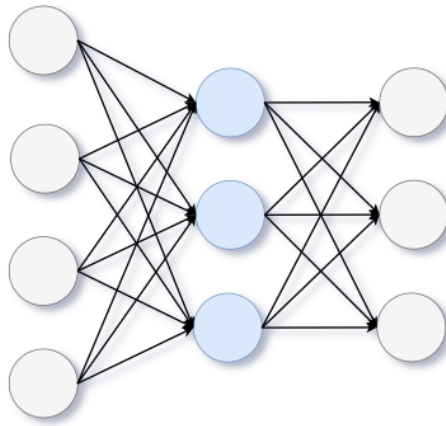


Figura 2.3: Exemplo de Rede Neural com camada oculta, representada na cor azul.

Perceptron [16], um dos primeiros modelos de rede neural. A Figura 2.3 traz um exemplo de rede neural com uma camada oculta.

2.1.1 Processo de Aprendizado

Como mencionado anteriormente, as redes neurais são capazes de aprender com o ambiente por meio de um processo de aprendizado e armazenam o conhecimento obtido nos pesos das conexões entre neurônios [16]. Esse aprendizado pode ser supervisionado, quando são fornecidos dados rotulados para que a rede aprenda com exemplos, ou não-supervisionado, quando a rede aprende padrões a partir de dados não-rotulados [15].

Durante o processo de aprendizado, uma tarefa é executada e avaliada conforme uma métrica pré-definida. Conforme a experiência, o contato com o ambiente aumenta e a tarefa tende a ser executada melhor em relação à métrica [17]. Assim, o processo de aprendizado numa rede neural pode ser pensado como um problema de otimização. Para uma tarefa qualquer, define-se uma função custo $J(\theta)$, onde θ são os parâmetros da rede. Essa função deve ser minimizada para que a tarefa seja executada com a melhor performance possível, dada uma métrica qualquer para avaliação do desempenho da rede. Tal métrica pode ser, por exemplo, o *recall* ou a precisão, conforme explorado no Capítulo 4. Porém, no processo de aprendizado,

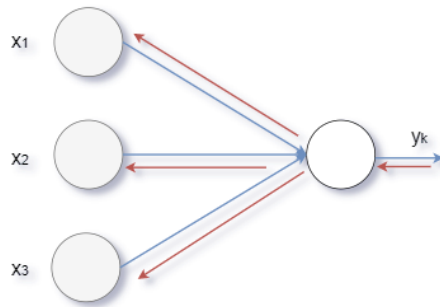


Figura 2.4: Fluxo de propagação da informação. Em azul mostra-se a fase de *forward propagation*. Em vermelho mostra-se a fase de *backward propagation*.

como destacado por Goodfellow *et al.* [17], uma função custo $J(\theta)$ é minimizada para que a métrica pela qual se avalia o desempenho da rede seja otimizada. Por exemplo, em problemas de classificação, para melhorar a acurácia pode-se tentar minimizar uma função custo que a princípio não está diretamente relacionada com a acurácia. Desta forma, a otimização é indireta, diferentemente dos problemas de otimização clássicos.

O processo de aprendizado pode ser dividido em duas etapas. Na primeira etapa, chamada de *forward propagation* [16], as entradas x são propagadas, camada por camada, da primeira até a última camada, gerando uma saída y . Essa primeira fase resulta em um custo $J(\theta)$. Para que a rede aprenda, esse custo tem que ser propagado na direção contrária de maneira a escolher os parâmetros da rede neural de forma a minimizar $J(\theta)$. Para tal, são realizadas modificações nos valores da conexões entre os neurônios. Essa segunda etapa é chamada de *backward propagation* [16].

Um ciclo completo do processo de aprendizado, com todos os dados disponíveis para treinamento, é chamado de época. Os dados de uma época podem ser divididos em *batches*, que são subconjuntos dos dados passados pela rede em cada iteração (*forward e backward propagation*) de uma época. Ou seja, a cada iteração utiliza-se no treinamento um subconjunto dos dados. Uma época é o conjunto de todas as iterações.

Grande parte das redes neurais utiliza um algoritmo de aprendizado (otimizador) baseado no cálculo do gradiente. Dessa forma, é necessário computar o

gradiente na fase de *backward propagation* a partir do custo $J(\theta)$. Esse cálculo geralmente é feito por meio do algoritmo de *backpropagation* [17]. Esse algoritmo utiliza a regra da cadeia do Cálculo para calcular as derivadas e encontrar o gradiente de maneira eficiente. O desenvolvimento do algoritmo de *backpropagation* é considerado um marco no estudo de redes neurais artificiais [16].

Em posse do gradiente, é possível utilizar um método de otimização para modificar os valores dos pesos das conexões da rede. Dentre os mais utilizados, destaca-se o método do Gradiente Descendente Estocástico, o SGD (*Stochastic Descent Gradient*), descrito a seguir.

2.1.2 Gradiente Descendente Estocástico - SGD

O SGD (*stochastic gradient descent*) é uma variação do Gradiente Descendente (GD). Esses métodos utilizam o cálculo de derivadas para realizar pequenas alterações na entrada (θ) de uma função (J), resultando em melhorias na sua saída $J(\theta)$ [17].

As derivadas representam o declive (inclinação) da reta tangente a uma curva qualquer. Dessa forma, se o declive for positivo, um aumento na variável θ resulta em um aumento da saída da função $J(\theta)$. E, de forma oposta, se o declive for negativo, um aumento em θ resulta numa diminuição de $J(\theta)$. A Figura 2.5 mostra o efeito de um aumento na variável θ quando o declive, dado pela derivada $J'(\theta)$, é positivo.

No processo de aprendizado de máquina, a função $J(\theta)$ é a função custo que se deseja minimizar. Assim, deve-se fazer pequenos ajustes em θ na tentativa de encontrar o mínimo da função, ponto no qual o declive é zero. Em outras palavras, deve-se mover no sentido contrário da derivada pois, quando a derivada é positiva, uma variação negativa na variável de interesse resulta em uma diminuição da função custo. Quando a derivada é negativa, uma variação positiva resulta na diminuição do custo. Dessa forma, o ajuste pode ser feito invertendo o sinal da derivada para que o movimento seja na direção contrário do declive e multiplicando-a por um parâmetro que define o tamanho do movimento nessa direção:

$$\Delta\theta = -\alpha \frac{dJ(\theta)}{d\theta}. \quad (2.2)$$

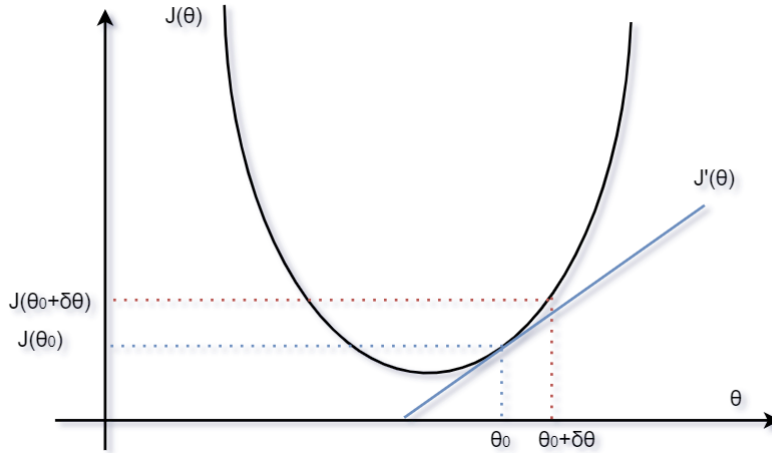


Figura 2.5: Efeito do aumento na variável θ quando a inclinação é positiva

Na Equação 2.2, a constante α é chamada de *learning rate* e representa o tamanho do ajuste realizado em cada etapa do processo de minimização da função. Essa equação se refere ao caso em que θ é uma variável unidimensional. Porém, em casos práticos, J é uma função multivariável e θ é o vetor dos parâmetros $\theta_0, \theta_1, \theta_2, \dots, \theta_n$ da rede neural. Neste caso, é necessário calcular as derivadas parciais de J para cada parâmetro da rede. Utilizando o gradiente (o vetor das derivadas parciais de J), é possível escrever a variação nos parâmetros da rede a cada iteração na forma vetorial como:

$$\Delta\boldsymbol{\theta} = -\alpha\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) \quad (2.3)$$

Onde:

$$\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \left[\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0}, \dots, \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_n} \right]^T \quad (2.4)$$

O método do Gradiente Descendente consiste em realizar iterações da Equação 2.3 até que se atinja a convergência. Contudo, muitas vezes o custo computacional do cálculo do Gradiente é muito alto, uma vez que a função custo pode depender de um número muito grande de entradas, o que resulta num tempo grande para cada iteração da Equação 2.3 [17]. Para contornar esse problema utiliza-se o método do Gradiente Estocástico Descendente (SGD), uma extensão do Gradiente Descendente.

No SGD, ao invés de se calcular o gradiente, é calculada uma estimativa do gradiente. Nessa estimativa, um *minibatch* é criado, contendo apenas um pequeno número de exemplos (subconjunto) das entradas amostradas aleatoriamente. Geralmente essa amostra contém uma a algumas centenas de exemplos, e a função custo é calculada sobre este *minibatch*. Como o número de exemplos é pequeno, o custo computacional é muito menor. Dado que uma estimativa é utilizada, um número maior de iterações é realizado para que o SGD convirja. Porém, apesar do número maior de iterações, o SGD tende a ser mais computacionalmente eficiente do que o GD, dado que uma iteração do GD é extremamente mais custosa do que no SGD. Isso compensa o maior número de passos.

Em geral, o SGD não utiliza valores computados anteriormente para o cálculo atual. Isso faz com que o progresso em direção ao mínimo da função custo seja mais lento, devido a oscilações na direção. Para acelerar o SGD na direção relevante, é possível utilizar o momento (*momentum*) [18]. O momento é uma adaptação na forma de se ajustar os pesos da rede que adiciona um fator de memória ao cálculo dos novos pesos. Utilizando o momento, o ajuste em direção ao mínimo da função considera também uma “média” dos últimos ajustes, além do movimento no sentido oposto ao gradiente. O novo ajuste é dado por:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \alpha \nabla_{\theta} J(\boldsymbol{\theta}). \quad (2.5)$$

Na Equação 2.5, \mathbf{v}_t é o ajuste atual, \mathbf{v}_{t-1} é o ajuste anterior e γ é o parâmetro de momento. O ajuste apresentado na Equação 2.3 é somado ao ajuste anterior para compor o ajuste atual. Considerando esse novo ajuste, a equação para a atualização dos parâmetros da rede passa a ser:

$$\Delta \boldsymbol{\theta} = -\mathbf{v}_t \quad (2.6)$$

2.1.3 Regularização

Em geral, após o processo de aprendizado (isto é, o treinamento), é esperado que a rede neural realize corretamente a tarefa para qual foi construída quando novos dados são fornecidos. Esta capacidade é chamada de *generalização* [16]. Para que se possa avaliar a capacidade de generalização de uma rede, os dados disponíveis

para o aprendizado são divididos em um conjunto de treinamento e um conjunto de testes. Durante o treinamento, o processo de aprendizado ocorre com os dados do respectivo conjunto e a rede é avaliada sobre os dados que foram utilizados para o aprendizado. O conjunto de testes é formado por dados que a rede neural não teve acesso durante o processo de aprendizado. A avaliação do desempenho da rede sobre esse conjunto indica o quão boa é sua capacidade de generalizar [16]. Quando o erro na fase de testes é muito maior do que o erro na fase de treinamento isso indica que a rede não teve bom desempenho para dados que não foram vistos durante o treinamento. Nesse caso, diz-se que ocorreu um *overfitting* [16].

O termo *overfitting*, ou sobreajuste, surge ao se pensar no processo de aprendizado como um problema de ajuste de curvas (*curve fitting*). A rede acaba memorizando o mapeamento entrada-saída dos dados utilizados no treinamento, se tornando especializada nos dados apresentados durante essa fase [16]. Nesse caso, a rede perde a capacidade de generalizar e não consegue realizar corretamente o mapeamento entrada-saída para dados que são diferentes daqueles vistos durante o aprendizado. Esse fenômeno pode ocorrer, por exemplo, quando a rede aprende uma característica que está presente apenas no conjunto de treinamento, devido a ruído ou outros fatores [16]. A Figura 2.6 representa um exemplo no qual ocorre *overfitting*. Os dados de entrada são representados pelos “X”, a curva pontilhada representa a curva que melhor se ajusta aos dados e a curva em vermelho representa um caso de *overfitting*.

Existem diversas técnicas que para redes neurais com o objetivo de evitar o *overfitting* e diminuir o erro de generalização. A aplicação dessas técnicas é chamada de regularização. Conforme a definição de Goodfellow *et al.*, regularização é “qualquer modificação feita a um algoritmo de aprendizado, de forma a diminuir o erro de generalização sem diminuir o erro de treinamento” [17]. A forma mais comum de regularização consiste em limitar a capacidade dos modelos adicionando uma penalidade à função custo $J(\boldsymbol{\theta})$:

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \alpha\Omega(\boldsymbol{\theta}), \quad (2.7)$$

onde α é um hiper-parâmetro que indica o quão forte é a influência da penalidade na função regularizada $\hat{J}(\boldsymbol{\theta})$. Em geral, redes muito complexas possuem parâmetros

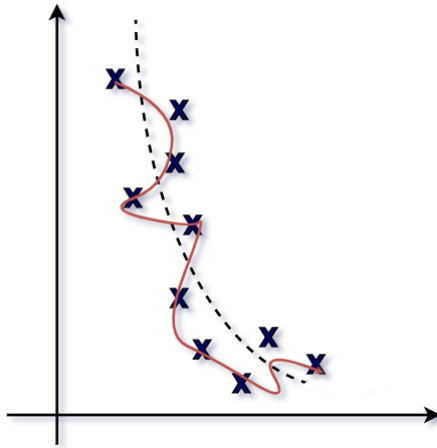


Figura 2.6: Exemplo de overfitting.

de valor elevado e tendem a ser excessivamente especializadas em relação aos dados de treinamento. Ao se introduzir uma penalidade à rede neural, pesos (parâmetros) muito grandes são desencorajados, fazendo com que a rede se torne mais simples e não se especialize nos dados de treinamento. A penalidade $\Omega(\boldsymbol{\theta})$ geralmente está relacionada a uma norma. Pode-se utilizar a norma L1 [19], de forma que a penalidade seja dada por:

$$\Omega(\boldsymbol{\theta}) = \sum_{i \in \mathcal{P}} w_i, \quad (2.8)$$

onde w_i representa os pesos das conexões da rede e \mathcal{P} é o conjunto dos parâmetros.

Outra forma de regularização, chamada de *dropout*, consiste em remover temporariamente alguns neurônios de uma rede neural, juntamente com suas conexões de entrada e saída [20]. Na maioria das redes modernas, essa remoção pode ser feita multiplicando a saída por zero [17]. Na prática, o *dropout* é aplicado em neurônios de algumas camadas de uma rede neural. A escolha de quais neurônios remover é realizada de maneira aleatória com probabilidade p . O valor p pode ser escolhido por meio de um conjunto de validação. De maneira mais simples, p pode ser configurado em 0.5, que é obtém valores satisfatórios para a maioria das tarefas e arquiteturas de rede [20]. A Figura 2.7 apresenta um exemplo de camada com *dropout* de dois neurônios.

Além das formas de regularização mencionadas, a diminuição do erro de gene-

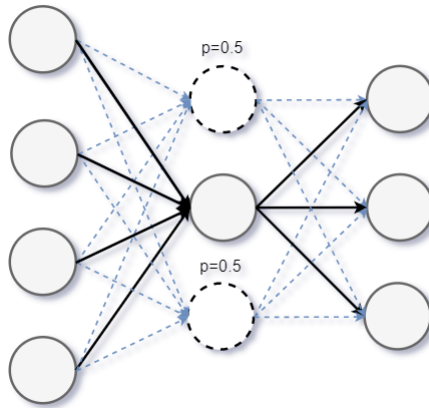


Figura 2.7: Exemplo de camada com *dropout* de dois neurônios.

realização pode ser alcançada por uma técnica chamada *batch normalization*. Nessa técnica, a ideia de normalizar as entradas de uma rede neural para melhora de performance é estendida para as camadas ocultas, utilizando *minibatches* para aumentar a eficiência computacional [21, 22]. A normalização das entradas é uma técnica muito utilizada para melhorar a performance e estabilidade de redes neurais. Essa técnica faz com que a variação das entradas da rede seja menor, diminui a influência de valores atípicos e faz com que diferentes entradas possam estar na mesma escala. Porém, mesmo com as entradas da rede normalizadas, alguns parâmetros podem crescer e gerar saídas de valor elevado nas camadas ocultas. Essas saídas elevadas alimentam outras camadas da rede e podem gerar instabilidades devido à introdução de valores atípicos nas camadas subsequentes. Para mitigar esse problema, a *batch normalization* é realizada pela introdução de uma camada na rede neural, cuja saída é dada por:

$$y(\mathbf{x}) = \gamma \frac{\mathbf{x} - \mu_B}{\sigma_B} + \beta, \quad (2.9)$$

onde μ_B e σ_B são respectivamente a média e o desvio padrão do *minibatch*, γ é um parâmetro de escala e β um parâmetro de deslocamento. Esses parâmetros são otimizados durante o processo de treinamento. Assim, uma camada de *batch normalization* pode ser introduzida após algumas das camadas ocultas da rede neural com o intuito de aumentar sua estabilidade e eficiência.

2.2 Redes Neurais Convolucionais

Redes neurais convolucionais, ou CNNs (do inglês *Convolutional Neural Network*), são redes neurais com múltiplas camadas nas quais, em pelo menos uma camada, é utilizada a operação de convolução ao invés da multiplicação matricial [17]. Essas redes são muito utilizadas em problemas de classificação de padrões e no processamento de dados que possuem estrutura em forma de grade, como dados de séries temporais (caso unidimensional) e imagens (caso bidimensional) [23]. A convolução é uma operação matemática que, quando aplicada a duas funções de variáveis reais, resulta numa terceira função que é a superposição ponderada do deslocamento de uma função sobre a outra [24]. Em termos matemáticos, para uma variável discreta, é possível definir a operação de convolução como

$$x(n) * w(n) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(n - \tau). \quad (2.10)$$

Na Equação 2.10, levando em consideração a aplicação em redes neurais, a função $x(n)$ é a entrada da camada, a função $w(n)$ é chamada de *kernel*, a função que contém os pesos a serem aplicados, o símbolo “*” representa o operador de convolução, n é o instante do tempo e τ é o instante da medição. Nota-se na Equação 2.10 que o *kernel* é invertido em relação à entrada. Assim, conforme τ aumenta o índice $n - \tau$ diminui. Isso faz com que a operação seja comutativa, ou seja, a ordem das funções não altera o resultado. Essa propriedade, contudo, geralmente não é relevante para as aplicações com aprendizado de máquina e muitas bibliotecas implementam a operação de convolução sem inverter o *kernel* [17]. A Figura 2.8 ilustra a operação de convolução para o caso unidimensional. O resultado da operação é gerado pelo produto escalar do *kernel* de 3 elementos com cada subconjunto de 3 elementos consecutivos da entrada.

As CNN possuem características que tornam seu uso vantajoso em um grande conjunto de problemas envolvendo aprendizado de máquina, em especial na classificação de padrões e problemas envolvendo séries temporais ou imagens [16]. As redes convolucionais possuem *conexões esparsas*. Ou seja, nas CNNs cada unidade de saída interage apenas com um subconjunto das unidades de entrada, diferente das redes MLP tradicionais, nas quais, em geral, todas as unidades de saída

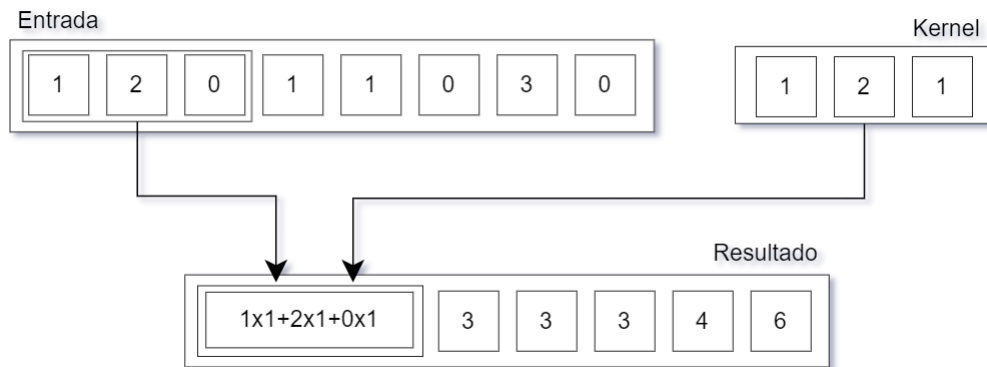


Figura 2.8: Exemplo de operação de convolução em 1 dimensão.

interagem com todas as unidades de entrada [17]. A Figura 2.9 apresenta a operação de convolução onde a entrada é uma imagem representada na forma matricial. A imagem possui tamanho 3×3 e seus elementos representam a intensidade dos *pixels*. O *kernel* possui tamanho 2×2 e a região em destaque na entrada mostra como se dá o produto com os elementos do *kernel*. O elemento em destaque na saída é a soma dos termos resultantes do produto e os demais elementos na saída são obtidos fazendo o *kernel* deslizar pelo restante da imagem. É possível notar pela Figura 2.9 que apenas alguns dos elementos da entrada tem influência em um elemento da saída (os elementos que participam do produto que resulta naquela saída). Isso faz com que seja possível detectar características locais da entrada, uma vez que apenas um conjunto de pontos de dados da entrada contribui para cada ponto de dado da saída [16]. Além disso, o número de parâmetros necessários para computar a saída é menor do que no caso completamente conectado, o que torna o processamento mais rápido e diminui a quantidade de memória necessária para armazenamento de parâmetros.

Outra característica importante das camadas convolucionais é o compartilhamento de parâmetros, um parâmetro é utilizado mais de uma vez no cálculo das saídas de uma camada. Como resultado, a rede só precisa aprender um conjunto de parâmetros ao invés de aprender conjuntos diferentes para cada região [16]. Na Figura 2.9, é possível notar que os parâmetros utilizados para gerar cada saída são os mesmos: os elementos do *kernel*. Esse compartilhamento resulta na equi-

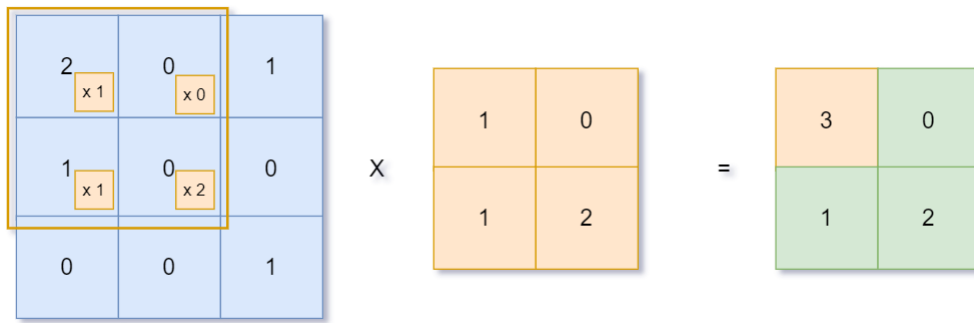


Figura 2.9: Exemplo de funcionamento de uma camada convolucional em uma rede neural.

variância à translação: a saída é transladada na mesma proporção quando ocorre uma translação na entrada. Em séries temporais isso significa que uma espécie de linha do tempo das propriedades de interesse é criada. Caso um evento na entrada seja movido para um momento futuro, a mesma representação da propriedade de interesse aparecerá na saída no mesmo momento futuro.

As camadas convolucionais geralmente são seguidas por uma camada de *pooling*, além de uma camada de ativação não-linear anterior a esta camada. Essa camada realiza uma operação de forma a substituir a saída da rede em uma certa localização por um valor que resuma as estatísticas da região. Entre as operações mais utilizadas nesse tipo de camada estão a média e o cálculo do valor máximo (*max-pooling*) das saídas dos neurônios de uma camada. O *pooling* resulta na invariância a pequenas translações na entrada das redes (implicando na remoção da equivariância resultante das camadas convolucionais para pequenas translações) [17]. Isso significa que pequenas mudanças na entrada não afetam a saída da camada de *pooling*. Esta propriedade é útil quando é mais importante identificar a presença de uma característica do que localizá-la exatamente [17]. A Figura 2.10 traz um exemplo de uma camada de *max-pooling* de tamanho 2, na qual os cálculos são realizados de 2 em 2 neurônios.

A rede neural proposta neste trabalho, apresentada na Figura 2.11, possui duas camadas convolucionais com *kernel* de tamanho 21, além de camadas de ativação do tipo *leaky RELU*, camadas de *dropout*, camadas de *pooling* de tamanho

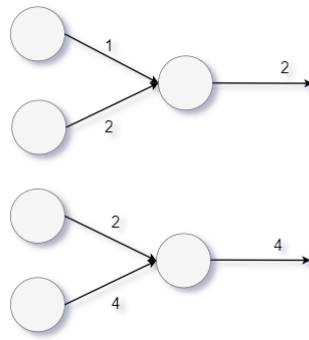


Figura 2.10: Operação de pooling.

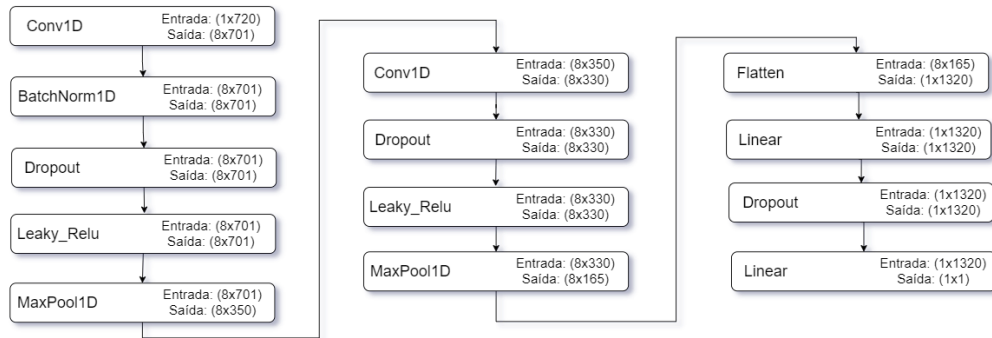


Figura 2.11: Arquitetura da rede proposta.

dois, camada de *batch normalization* e mais duas camadas totalmente conectadas. A camada de *dropout* e a camada de *batch normalization* são utilizadas como forma de regularização. Também são utilizadas a regularização L1 e o parâmetro de *weight decay*, uma forma de regularização detalhada no Capítulo 4. As camadas de ativação do tipo *leaky RELU* são camadas cuja função de ativação $f(x)$ é dada por $f(x) = x$ para $x > 0$ e $f(x) = \alpha x$ para $x < 0$, onde $0 < \alpha < 1$. Essa rede neural explora as características das CNNs e redes neurais discutidas neste Capítulo e as utiliza para a realização da tarefa de detecção de arritmias cardíacas.

2.3 Aprendizado Federado

Os dispositivos eletrônicos presentes na sociedade contemporânea produzem (e consomem) uma quantidade massiva de dados, que podem ser utilizados para o treinamento de modelos computacionais baseados em aprendizado de máquina [25]. Em contrapartida, esses dados são, geralmente, sensíveis por serem privados e seu armazenamento em uma mesma localização (isto é, um servidor central utilizado no aprendizado de máquina tradicional), traz riscos e responsabilidades em relação aos dados. Além disso, a grande quantidade de dados implica em dificuldades para a transmissão de dados dos dispositivos até o servidor [4]. O Aprendizado Federado, FL (*Federated Learning*), é uma variação do aprendizado de máquina. O objetivo é treinar um modelo centralizado enquanto os dados continuam distribuídos entre clientes distintos, solucionando o problema da privacidade e do compartilhamento de dados [25], uma vez que o servidor não tem acesso aos dados dos clientes. Assim, os dispositivos locais computam atualizações nos parâmetros do modelo, utilizando seus próprios dados, e compartilham essas atualizações com o servidor. O servidor agrega essas atualizações e computa os parâmetros globais, que são enviados aos dispositivos para que atualizem seus modelos locais [4].

A Figura 2.12, mostra as diferenças entre o modelo centralizado (à esquerda) e o modelo federado (à direita). No centralizado, os dados são enviados pelos dispositivos ao servidor, que treina um modelo global e o envia para cada dispositivo. No federado, o servidor apenas agrega as atualizações enviadas pelos dispositivos e envia os parâmetros globais resultantes dessas agregações para cada dispositivo.

Com a popularidade e eficiência do SGD, é natural que os primeiros algoritmos que foram desenvolvidos tenham sido baseados na técnica do Gradiente Descendente Estocástico. Dentre os algoritmos mais famosos de aprendizado federado destaca-se o *Federated Averaging*, que é uma extensão da versão federada do SGD (*FederatedSGD*) [26].

No *FederatedSGD*, uma fração C de clientes é selecionada a cada rodada de comunicação e cada cliente k selecionado computa o gradiente, $\nabla_{\theta} J(\theta)$, da função perda $J(\theta)$ [26]. Cada cliente envia esse gradiente computado ao servidor central, que os agrega por meio da equação:

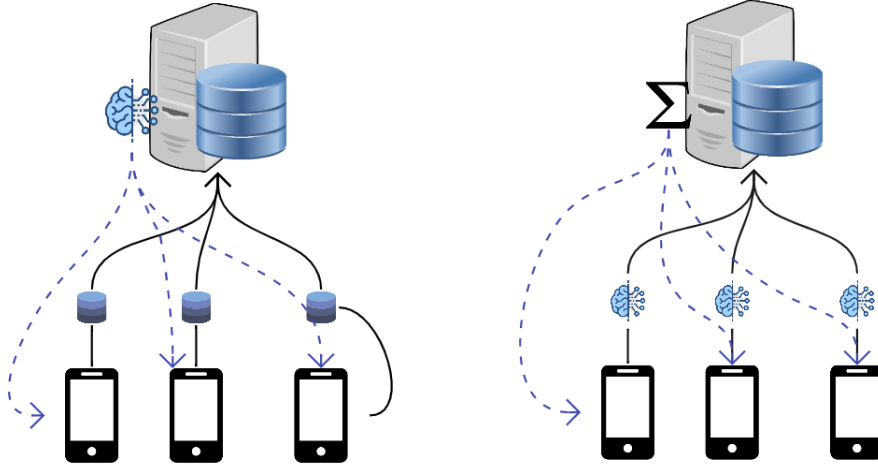


Figura 2.12: Modelo centralizado (à esquerda) e modelo federado (à direita).

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \sum_{k=1}^K \frac{n_k}{n} \nabla J_k(\boldsymbol{\theta}_t), \quad (2.11)$$

onde n é o tamanho total do conjunto de dados dos clientes somados, K é o número total de clientes, n_k é o tamanho da partição de dados do cliente k , α é o *learning rate* e $\boldsymbol{\theta}$ são os parâmetros da rede.

Uma forma equivalente de atualização dos parâmetros é obtida com os clientes realizando um passo do SGD (ou GD) e com o servidor agregando os resultados por meio de uma média ponderada [26], fazendo

$$\boldsymbol{\theta}_{t+1} = \sum_{k=1}^K \frac{n_k}{n} \boldsymbol{\theta}_{t+1}^k, \quad (2.12)$$

onde $\boldsymbol{\theta}^k$ são os parâmetros locais do cliente k e a atualização de $\boldsymbol{\theta}^k$ é dada por:

$$\boldsymbol{\theta}_{t+1}^k = \boldsymbol{\theta}_t - \alpha \nabla J_k(\boldsymbol{\theta}_t). \quad (2.13)$$

Com o algoritmo de atualização escrito dessa forma, é possível realizar mais de um passo do treinamento localmente antes de enviar os pesos para o servidor [26]. Esse algoritmo é chamado de *Federated Averaging* (FedAvg) e tem três principais parâmetros: a fração de clientes, selecionados a cada rodada, C ; o número de épocas locais E ; e o tamanho B do *minibatch*, usado para a atualização dos parâmetros dos

clientes. O Algoritmo 1 apresenta de forma simplificada o *FederatedAveraging* do lado do servidor, enquanto o Algoritmo 2 apresenta o *FederatedAveraging* do lado do cliente. A variável K representa o número total de clientes.

Algoritmo 1: *FedAvg* - SERVIDOR

Entrada: C, K

Saída: Pesos enviados para os clientes.

```

1 início
2   inicializa  $\theta_0$ 
3   para cada rodada de comunicação  $t=1,2,\dots$  faça
4      $m \leftarrow \text{Max}(C \cdot K, 1)$ 
5      $S_k \leftarrow$  conjunto aleatórios de  $m$  clientes
6     para  $k \in S_k$  faça
7        $\theta_{t+1}^k \leftarrow$  Recebe pesos  $\theta_t$  do cliente  $k$ 
8     fim
9      $\theta_{t+1} = \sum_{k=1}^K \frac{n_k}{n} \theta_{t+1}^k$ 
10  fim
11  envia pesos  $\theta_{t+1}$  aos clientes
12 fim
13 retorna  $\theta_{t+1}$ 

```

Algoritmo 2: *FedAvg* - CLIENTE

Entrada: θ, B, E **Saída:** Pesos enviados para o servidor.

```
1 início
2   recebe  $\theta$  do servidor
3    $\beta \leftarrow$  (Divisão do subconjunto de dados do cliente  $k$  em batches de
   tamanho  $B$  )
4   para  $i \in [1, E]$  faça
5     para  $b \in \beta$  faça
6        $\theta \leftarrow \theta - \alpha \nabla J(\theta)$ 
7     fim
8   fim
9   envia pesos  $\theta$  para o servidor
10 fim
11 retorna  $\theta$ 
```

Capítulo 3

Trabalhos Relacionados

A criação de grandes bases de dados como a MIT-BIH [12] e AHA [27], datadas da década de 80, representam um marco nos trabalhos de detecção de arritmias. A partir do momento que foram criadas, essas bases se tornaram o padrão para os testes de desempenhos de detectores e classificadores de arritmias. Assim, a maioria dos trabalhos publicados é baseada nesses conjuntos de dados.

Nos trabalhos relacionados, diversos tipos de classificadores foram propostos, de modelos baseados em *support-vector machines* (SVM) a modelos baseados em redes neurais convolucionais. Conforme novos algoritmos e novas tecnologias surgem, novos trabalhos são propostos incorporando a novidade. Dessa maneira, com o desenvolvimento e popularização do aprendizado federado, trabalhos empregando essa técnica na detecção de arritmias em eletrocardiogramas começaram a ser publicados.

Este capítulo expõe diversos trabalhos relacionados à detecção de arritmias, com foco naqueles que utilizam redes neurais convolucionais e aprendizado federado, destacando pontos relevantes para o desenvolvimento do projeto. Os trabalhos foram divididos em três seções de acordo com suas características. Aqueles envolvendo classificadores tradicionais e análise das bases de dados são expostos na seção 3.1. A seção 3.2 apresenta trabalhos envolvendo modelos centralizados baseados em CNNs. Por fim, a seção 3.3 apresenta os modelos que utilizam aprendizado federado.

3.1 Bases de dados e Classificadores Tradicionais

Moody e Mark [12], idealizadores da MIT-BIH, dão um panorama geral da metodologia utilizada na construção da base e refletem sobre o impacto do conjunto de dados nos problemas de detecção de arritmias. Os autores observam que, antes da criação de bases públicas e representativas, mais especificamente a MIT-BIH e a AHA, os desenvolvedores utilizavam seus próprios dados para gerar estatísticas sobre a performance dos detectores. Essas estatísticas tinham pouco valor, já que cada detector utilizava um conjunto de dados diferente [12].

Com a criação e divulgação das primeiras bases, diversos trabalhos passaram a ser desenvolvidos e a performance dos modelos puderam passar a ser melhor avaliadas. Porém, como observado por Luz *et al.* [28], muitos desses trabalhos utilizam os dados de um mesmo paciente no treinamento e no teste ao mesmo tempo. Isso não condiz com situações reais, uma vez que os dados de novos pacientes nunca teriam sido vistos por um detector. Muitos dos trabalhos feitos dessa forma apresentam acurácia acima de 95%. O classificador proposto por Ye *et al.* [29], por exemplo, baseado em um modelo de SVM e técnicas de análise de componentes principais, apresentou acurácia de 99,1% nos dados, sem separação de pacientes para treino e teste.

Luz *et al.* [28] analisam diversos trabalhos e destacam o artigo de Chazal *et al.* [13]. Esse trabalho propõe uma divisão no conjunto de dados de maneira a separar pacientes para treino e teste, resultando em um cenário mais realista. Além da divisão do conjunto de dados, Chazal *et al.* [13] propõem ainda uma classificação baseada em análise discriminante linear com acurácia em torno de 83%. Além disso, Luz *et al.* [28] analisam o desempenho de diversos classificadores utilizando técnicas de extração de *features* propostas em outros artigos, como os já citados Ye *et al.* [29] e Chazal *et al.* [13]. Os autores apontam que os modelos com melhor acurácia, SVM e MLP, apresentam um viés que favorece a classe dominante (isto é, ausência de arritmia). O modelo Bayesiano, por sua vez, obteve melhor desempenho em relação ao balanceamento das classes, apesar da menor acurácia em comparação a outros classificadores.

A questão do viés, também explorada por Chazal *et al.* [13], mostra que apenas a acurácia não é suficiente para análise do desempenho de modelos, uma vez

que o desbalanceamento do conjunto de dados deve ser levado em conta. Assim, outros parâmetros como o *recall* e a precisão devem ser considerados na análise [13, 28, 29].

3.2 Modelos baseados em CNN

Além dos métodos já citados, muitos dos trabalhos publicados sobre classificação ou detecção de arritmias, principalmente na última década, utilizam modelos baseados em Redes Neurais Convolucionais na tarefa de classificação.

Li *et al.* [7] propõem uma rede neural com 7 camadas ao todo, sendo duas camadas convolucionais de uma dimensão. Os autores utilizam a MIT-BIH e os dados são pré-processados utilizando métodos baseados em wavelets, e o classificador proposto atingiu uma acurácia de 97,5% no conjunto de dados. Os autores dividiram os batimentos aleatoriamente nos conjuntos de treino e teste, sem impor a restrição de não utilizar dados de um mesmo paciente para treino e teste simultaneamente. Como observado por Luz [28], isso pode estar está relacionado com a acurácia mais alta.

Já Hannun *et al.* [8] propõem uma rede de 34 camadas, sendo três camadas convolucionais. Os autores coletaram 64.121 gravações de eletrocardiogramas de 29163 pacientes. Um comitê de cardiologistas realizou anotações em cada gravação, classificando os batimentos em 12 classes. Para regularização, dado que as classes são desbalanceadas, foram introduzidas camadas de *dropout* e *batch normalization*. Hanum *et al.* [8] dividiram o conjunto de dados de maneira a não ocorrer interseção de pacientes entre os conjuntos de treino e teste. E, para avaliação, outros cardiologistas classificaram independentemente os batimentos, de forma a servir de comparação para os resultados do classificador. O modelo proposto obteve acurácia de 80,9% e superou a performance dos cardiologistas independentes (75,1%).

Kurniawan *et al.* [30] propõem uma CNN no espectro da frequência, diferentemente dos outros trabalhos que são contidos no domínio do tempo. Nessa abordagem, a entrada das camadas convolucionais é o espectro de Fourier da imagem de um electrocardiograma. A transformação para o espectro da frequência foi feita por meio da Transformada Rápida de Fourier e a rede proposta contém 8 camadas

convolucionais em duas dimensões [30]. O modelo apresentou acurácia de 98,6% nos dados do MIT-BIH. Porém, assim como no modelo de Li *et al.* [7], os dados foram divididos aleatoriamente nos conjuntos de treino e teste, sem se preocupar com a interseção de pacientes nos dois conjuntos.

3.3 Redes com Aprendizado Federado

Devido às vantagens do aprendizado federado aplicado à dados médicos (principalmente em relação a privacidade e segurança) alguns trabalhos buscaram incorporá-lo na tarefa de classificação de arritmias, após a introdução da técnica em 2016 [4].

Sakib *et al.* [5] propõem um modelo de aprendizado federado, no qual dispositivos nos nós de borda recebem os dados individuais dos clientes, realizam treinamento com os dados locais e compartilham os parâmetros com o servidor central e os outros nós de borda, atualizando os pesos de maneira assíncrona. Os autores seguem as recomendações da AAMI [11] para o desenvolvimento de modelos computacionais para detecção de arritmias, que incluem a não-interseção de pacientes nos conjuntos de treino e teste e a utilização da base MIT-BIH para avaliação. No trabalho em questão, não é explorado o caso de dados não-IID entre os clientes.

Gao *et al.* [10] realizam uma avaliação do aprendizado federado aplicado à dispositivos de Internet das Coisas (IoT). Para tal, utiliza-se um modelo baseado em CNN de uma dimensão e o problema da detecção de arritmias como estudo de caso. No artigo, são avaliados os casos IID e não-IID. Entretanto, os dados são divididos nos conjuntos de teste e treino de maneira aleatória. Os autores também realizam testes com dispositivos de IoT para avaliar a sobrecarga causada pelo aprendizado federado. Para esses testes, Gao *et al.* [10] escreveram um manual de como instalar e utilizar a biblioteca PyTorch [31] em dispositivos “Raspberry Pi”, disponibilizado no artigo [10].

Zhang *et al.* [6] propõem uma estratégia de compartilhamento de uma fração de dados de modo a otimizar o aprendizado federado em situações de dados não-IID. Na estratégia proposta, uma fração dos dados dos clientes são compartilhados com o servidor, de maneira a formar uma distribuição de dados global. O servidor

realiza rodadas de treinamento e envia os pesos atualizados para os clientes. A partir dos seus dados locais, os clientes repetem o mesmo processo e enviam os pesos atualizados para o servidor. Essa estratégia, segundo os autores, não fere a privacidade do aprendizado federado, pois o servidor é considerado uma entidade confiável [6]. Essa abordagem garante que um cliente não terá acesso aos dados de outro cliente. Porém, como uma fração dos dados dos clientes é compartilhada com o servidor central, acaba-se perdendo a privacidade em relação ao servidor, uma das principais vantagens da utilização do aprendizado federado.

Ma *et al.* [9] propõem uma técnica de aprendizado federado combinado ao alinhamento de *features*. Um modelo utilizando aprendizado federado é treinado utilizando os parâmetros enviados por múltiplos clientes. Em seguida, um modelo local é treinado por cada cliente utilizando seus dados privados. Esse modelo local faz uso de um módulo de alinhamento de *features*, no qual CNNs são utilizadas para extrair as *features* do modelo global e do modelo local. O modelo final de cada cliente é gerado de forma a minimizar a distância entre as *features* globais e locais. De maneira geral, a técnica proposta tem como objetivo diminuir o impacto das diferenças entre os conjuntos de dados locais e globais. Além disso, a técnica permite que o modelo local tenha mais adaptabilidade e alcance maior generalidade no treinamento.

Raza *et al.* [32], utilizam *transfer learning* e Inteligência Artificial Explicável para construir um modelo que garanta privacidade e segurança dos dados no contexto de IoT. O modelo é formado por dois módulos: o módulo de aprendizado federado e o módulo de IA Explicável. CNNs são utilizadas no módulo federado para classificar os dados de electrocardiogramas. Esse módulo utiliza *transfer learning* como forma de otimização do processo de aprendizado. Em seguida, o módulo de IA Explicável é utilizado para facilitar a interpretação dos resultados do classificador e agilizar a tomada de decisões de médicos. Os autores utilizam dispositivos “Raspberry Pi” para os testes com a base de dados do MIT e ressaltam o impacto do desbalanceamento dos dados no problema de classificação [32].

Este trabalho se diferencia dos demais por analisar a viabilidade de um modelo de detecção de arritmias baseado em aprendizado federado respeitando as recomendações da AAMI, utilizando uma base de dados pública e considerando o caso

não-IID. Além disso, é proposta uma nova divisão da MIT-BIH de forma a evitar a interseção de pacientes nos conjuntos de teste e treinamento. A Tabela 3.1 mostra a diferença deste trabalho em relação aos trabalhos da literatura envolvendo aprendizado federado no problema de detecção de arritmias. A tabela traz informações sobre a realização de experimentos com o caso não-IID, a preocupação com interseção de pacientes nos conjuntos de teste e treinamento e a utilização de bases de dados públicas.

Tabela 3.1: Trabalhos Relacionados envolvendo Aprendizado Federado

Artigo	Base de Dados Pública	Conjuntos de treino e teste sem interseção	Experimento com caso não-IID
Sakib <i>et al.</i> [5]	Sim	Sim	Não
Gao <i>et al.</i> [10]	Sim	Não	Sim
Zhang <i>et al.</i> [6]	Não informado	Não	Sim
Ma <i>et al.</i> [9]	Não	Não informado	Sim
Raza <i>et al.</i> [32]	Sim	Não	Não
Este trabalho.	Sim	Sim	Sim

Capítulo 4

Metodologia

Este capítulo apresenta a metodologia adotada neste trabalho. A Figura 4.1 traz o fluxograma simplificado das etapas seguidas durante a execução do projeto, que são detalhadas no decorrer do capítulo.

4.1 Bibliotecas e Frameworks

Todos os experimentos realizados utilizam a linguagem de programação Python [33], com auxílio das bibliotecas PyTorch [31], Scikit-learn [34], Pandas [35] e Numpy [36]. Além disso, os testes e experimentos envolvendo aprendizado federado foram realizados por meio do *framework Flower*, que permite simplificar a implementação do aprendizado federado [14].

4.2 Conjunto de Dados

Este trabalho utiliza a base de dados de arritmias do MIT, a “MIT-BIH Arrhythmia Database” [12], desenvolvida com dados de eletrocardiogramas de 47 pacientes, coletados entre 1975 e 1979. O conjunto de dados possui 48 gravações de aproximadamente 30 minutos, amostradas a 360 Hz.

Das 48 gravações presentes na base, 23 foram geradas pelos pesquisadores do MIT escolhendo-se aleatoriamente amostras de pacientes coletadas no *Boston’s Beth Israel Hospital*, para fornecer formas de onda que aparecem de maneira rotineira em exames clínicos. Esse é um subconjunto representativo das formas de onda mais

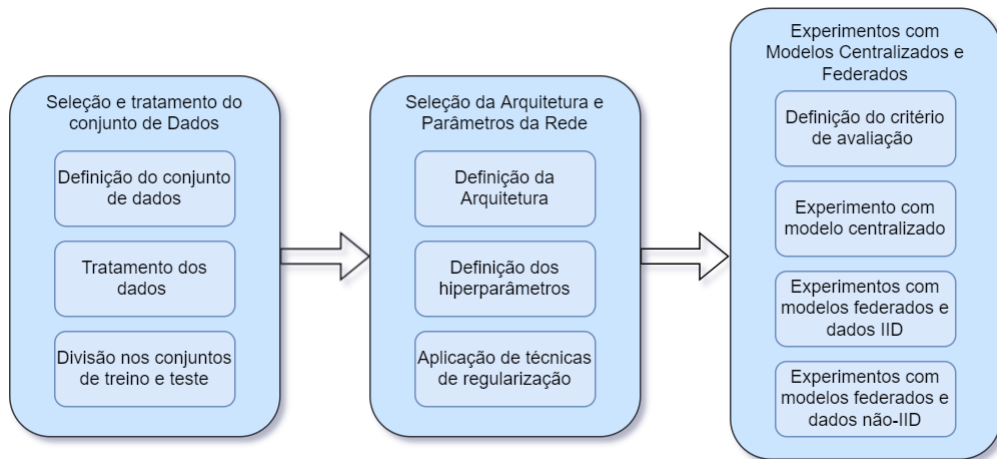


Figura 4.1: Fluxograma simplificado das etapas seguidas no trabalho.

comuns. Essas 23 gravações são fornecidas nos arquivos numerados de 100 a 124. As outras 25 gravações foram geradas escolhendo-se pacientes do hospital cujas amostras contém fenômenos clinicamente raros mas importantes e são fornecidas nos arquivos numerados de 200 a 234. Os nomes dos arquivos disponibilizados não são numerados sequencialmente. [12].

Os batimentos em cada gravação foram encontrados por meio da detecção dos picos-R do complexo QRS [37]. O complexo QRS, ilustrado na Figura 4.2, é uma combinação de 3 deflexões presentes em um eletrocardiograma comum que permitem identificar graficamente anormalidades do eletrocardiograma. A duração do complexo QRS, o intervalo entre dois complexos consecutivos, a amplitude dos picos e deformações que antecedem o QRS podem ser indicativos da presença de arritmias. O pico-R é o ponto com máxima amplitude do complexo QRS. Cada pico foi classificado quanto ao tipo de batimento por meio de anotações realizadas por dois cardiologistas.

4.2.1 Recomendações da AAMI

A AAMI, desenvolveu o padrão “ANSI/AAMI/ISO EC57, 1998- R2008” [11] para experimentos envolvendo a classificação de arritmias por modelos computacionais. Esse padrão visa fornecer um “terreno comum” para os diferentes trabalhos

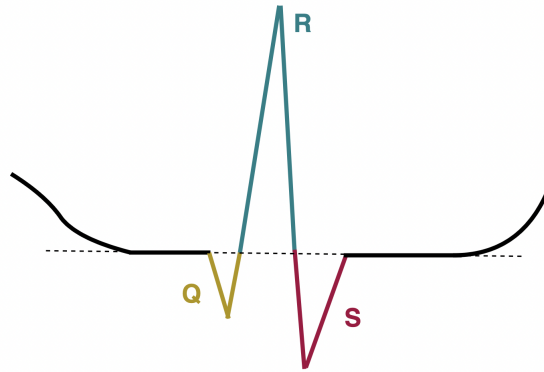


Figura 4.2: Representação do complexo QRS.

na área e minimizar erros e vieses. Nesse padrão, é recomendado o uso da base de dados públicas. No caso do MIT-BIH, é recomendada a exclusão de 4 gravações que possuem batimentos compassados. A AAMI indica que o *recall*, a precisão e a acurácia sejam utilizados como critérios para avaliação dos resultados.

Além disso, a AAMI recomenda que sejam criados conjuntos de dados de treino e teste de maneira que as amostras de um mesmo paciente não sejam utilizadas simultaneamente para treino e teste [11]. Como notado por Luz *et. al* [28], quando a recomendação não é seguida, os classificadores facilmente atingem resultados excelentes. Porém esses resultados não refletem a realidade, uma vez que os dados de um novo paciente não teriam sido utilizados no treinamento do modelo. Dessa forma, apenas 44 das 48 gravações do MIT-BIH são utilizadas neste trabalho e o conjunto de dados foi dividido de maneira a não utilizar amostras do mesmo paciente simultaneamente para treino e teste, como detalhado mais à frente. Além disso, *recall* e precisão foram utilizados para avaliação dos resultados.

4.2.2 Classificação

As amostras foram divididas em duas classes. A classe “0” representando os batimentos considerados normais e a classe “1” representando as arritmias, de acordo com as classes definidas pela AAMI.

A AAMI define cinco classes: SVEB (batimento ectópico supraventricular), VEB (batimento ectópico ventricular), F (batimento de fusão), Q (batimento des-

conhecido), N (batimentos que não se enquadram nas outras classes, ou seja, sem arritmias) [11]. O MIT utiliza diferentes símbolos nas anotações dos eletrocardiogramas para representar o tipo de um batimento. A Tabela 4.1 apresenta os símbolos utilizados nas anotações do MIT-BIH e sua equivalência com as Classes definidas pela AAMI de acordo com a definição de cada tipo de batimento [11, 12]. Neste trabalho, a classe “0” representa a classe N da AAMI e a classe “1” representa todas as outras classes, dado que o objetivo é apenas detectar as arritmias sem discriminá-las pelo tipo.

Tabela 4.1: Mapeamento Classe AAMI x Tipos MIT-BIH.

Classe AAMI	Tipos MIT-BIH
N	N, L, R, e, j, ‘.’
SVEB	A, a, J, s
VEB	V, E
F	F
Q	P, f, U.

É importante observar que o conjunto de dados possui classes extremamente desbalanceadas, uma vez que cerca de 90% dos batimentos pertencem à classe “0”, como pode ser observado na Figura 4.3.

4.2.3 Geração e tratamento dos dados

Para cada gravação, os batimentos foram representados por meio de uma janela de 360 amostras (isto é, um segundo) à esquerda e 360 amostras à direita da localização de cada pico-R, totalizando 721 amostras (isto é, dois segundos) por batimento.

Os sinais de batimentos passaram por um filtro passa-baixas *lowpass_biquad*, disponível na biblioteca PyTorch [31], para remover ruídos indesejados acima de 60 Hz. Cada sinal foi normalizado utilizando a normalização min-max da Equação 4.1, dado que as redes neurais apresentam melhor desempenho com entradas de menor amplitude.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}. \quad (4.1)$$

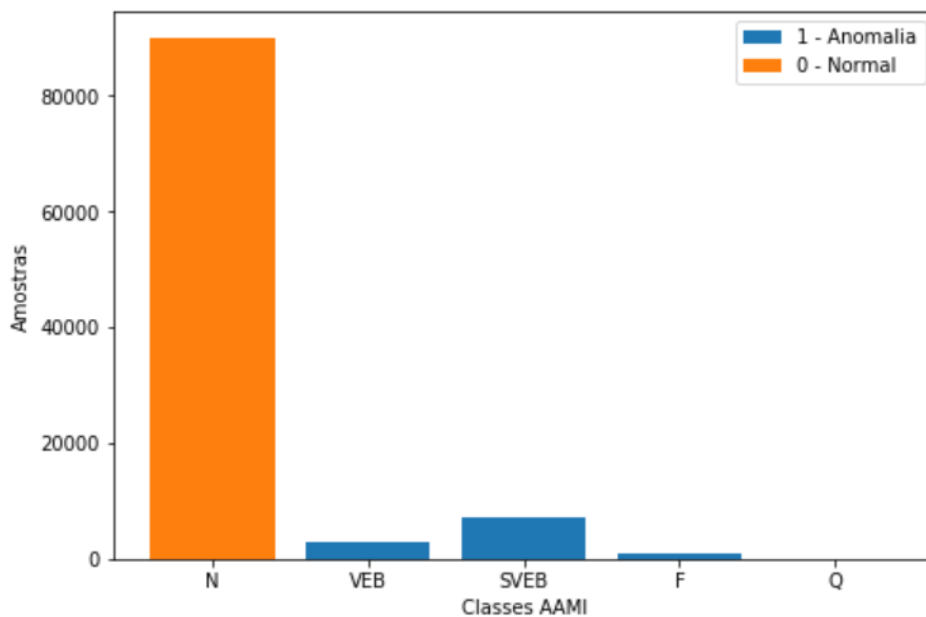


Figura 4.3: Divisão dos batimentos entre as classes.

4.2.4 Divisões dos Conjuntos de Treino e Teste

Conforme mencionado anteriormente, a AAMI recomenda que os conjuntos de treino sejam divididos de forma que os dados de um mesmo paciente não sejam utilizados para treino e teste simultaneamente [11].

Chazal *et al.* [13] propõem a divisão do conjunto de dados nos conjuntos DS1 e DS2. A Tabela 4.2 apresenta a divisão dos arquivos do MIT-BIH entre os dois conjuntos. Essa divisão, além de utilizar dados de um paciente apenas para treino ou teste (isto é, nunca os dois simultaneamente), leva em consideração a distribuição das classes, buscando manter um equilíbrio entre os conjuntos DS1 e DS2. O número de amostras por Classe AAMI em cada um dos conjuntos é exposto na Tabela 4.3.

A divisão proposta por Chazal *et. al* [13] tem 54% dos dados no conjunto DS1, utilizado para treino, e 46% dos dados no conjunto DS2, utilizado para teste. Visando aumentar a quantidade de dados disponíveis para treino, a divisão do conjunto de dados nos conjuntos DS3 e DS4 é proposta neste trabalho. A Tabela 4.4 apresenta os conjuntos DS3 e DS4 e os arquivos presentes em cada um deles. Essa divisão proposta busca substituir DS1 e DS2 de maneira que 75% do conjunto de dados seja utilizado para treinamento e o restante para teste. Assim como na divisão

Tabela 4.2: Distribuição das Gravações entre os conjuntos DS1 e DS2

DS1					DS2				
101	114	122	207	223	100	117	210	221	233
106	115	124	208	230	103	121	212	222	234
108	116	201	209	220	105	123	213	228	232
109	118	203	215	205	111	200	214	231	219
112	119				113	202			

Tabela 4.3: Distribuição das Classes entre os conjuntos DS1 e DS2

	N	SVEB	VEB	F	Q
DS1	45844	943	3788	415	8
DS2	44238	1836	3221	388	7

de Chazal *et al.* [13], as gravações são dispostas nos conjuntos DS3 e DS4 de forma a manter a distribuição das classes próxima da distribuição do conjunto original.

Tabela 4.4: Distribuição das Gravações entre os conjuntos DS3 e DS4

DS3						DS4			
100	103	101	106	108	109	105	114	116	124
112	115	118	111	113	117	200	203	209	213
121	123	119	122	201	205	214	222	231	234
207	208	215	220	223	230				
202	210	212	232	233	228				
219	221								

A Tabela 4.5 apresenta a divisão de amostras por classe AAMI nos conjuntos DS3 e DS4. Neste trabalho a classe “0” equivale à classe AAMI “N” e a classe “1” equivale ao conjunto das demais classes. Como os dados são extremamente desbalanceados entre as classes “0” e “1” foi aplicada a técnica de *oversampling* aleatório nas amostras de classe 1 do conjunto de treino (DS3), de forma a contornar o problema do desbalanceamento de classes. O *oversampling* aleatório consiste

em duplicar aleatoriamente amostras da classe minoritária, que é classe com menor número de amostras do conjunto de dados. Esse *oversampling* foi realizado considerando as classes AAMI, de forma que a transformação das classes AAMI nas classes “0” e “1” foi realizada após este processo. Dessa forma a proporção dos diferentes tipos de arritmia na classe “1” foi mantida mesmo após o *oversampling*.

Tabela 4.5: Distribuição das Classes entre os conjuntos DS3 e DS4

	N	SVEB	VEB	F	Q
DS3	63786	2033	5014	427	4
DS4	20990	608	1586	292	11

4.3 Arquitetura da Rede e Definição de Parâmetros

Como descrito no Capítulo 3, as Redes Neurais Convolucionais são bastante utilizadas na literatura em trabalhos envolvendo classificação de arritmias. A rede neural apresentada na Seção 2.2 é utilizada neste trabalho para detecção de arritmias.

A função perda utilizada foi a entropia cruzada ombinada a uma camada sigmoide, e o otimizador utilizado foi o SGD com um momento de 0,9 [38]. O *learning rate* escolhido após alguns testes foi 0,01.

As camadas de *Batch Normalization* e *Dropout*, como descrito no Capítulo 2, são utilizadas como formas de regularização. Além dessas camadas, foi implementada a regularização L1 [19] e foi utilizado o parâmetro *weight decay* do *PyTorch*. O *weight decay* é um parâmetro passado ao otimizador (SGD) que atribui uma penalidade à atualização dos pesos seguindo as equações

$$\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} + \alpha \mathbf{G}^{(i)}, \quad (4.2)$$

$$\mathbf{G}^{(i)} = \frac{d\mathbf{L}}{d\mathbf{W}^{(i)}} + \beta \mathbf{W}^i, \quad (4.3)$$

onde \mathbf{W} são os pesos, \mathbf{G} é o gradiente armazenado, \mathbf{L} é o gradiente calculado sem regularização e β e α são constantes, sendo β o parâmetro *weight decay* passado

para a função e α o *learning rate*. O β utilizado foi de 0,001.

O modelo centralizado serve como base para comparações dos resultados dos casos distribuídos. O treinamento local foi realizado por 100 épocas e com *batches* de tamanho 16, 128 e 512.

4.4 Modelos Distribuídos

O algoritmo de aprendizado federado utilizado no experimento (*FedAvg*) é implementado e disponibilizado pela biblioteca *Flower* [14].

Os experimentos com modelos distribuídos foram divididos em dois casos. No caso ideal (dados independentes identicamente distribuídos - IID), todos os clientes possuem a mesma distribuição de dados. O outro é o caso no qual os clientes possuem distribuições de dados diferentes entre si (não-IID). Nos dois cenários, os testes foram implementados utilizando cinco clientes independentes que se comunicam com um servidor central considerando condições ideais de rede (isto é, todos os participantes disponíveis o tempo todo sem interrupções na comunicação). Os clientes realizam o treinamento por um número definido de épocas locais e enviam os parâmetros obtidos para o servidor. Cada iteração desse processo é chamada de rodada de comunicação. Em todos os casos, foram realizadas 100 rodadas de comunicação entre clientes e servidor.

4.4.1 Dados IID

Para o caso IID, os dados do conjunto DS3 foram distribuídos igualmente entre os clientes. Os batimentos foram divididos aleatoriamente entre eles, de maneira que todos possuíssem a mesma distribuição, sem diferenciação entre pacientes. A proporção de arritmias em cada cliente é apresentada na Tabela 4.6.

Neste cenário, considerado ideal, o algoritmo *FedAvg* foi utilizado para agregação dos parâmetros dos clientes. Os experimentos foram realizados com 1, 5 e 10 épocas locais. E em cada um dos três cenários de épocas locais, foram utilizados *batches* de tamanho 16, 128 e 512.

Tabela 4.6: Distribuição de amostras apresentando arritmias entre os clientes

Cliente	Porcentagem de Arritmias
0	10,32%
1	10,58%
2	10,41%
3	10,65%
4	10,49%

4.4.2 Dados Não-IID

Para o caso não-IID, os batimentos do conjunto DS3 foram distribuídos entre os clientes por pacientes, simulando um caso mais realista, no qual cada cliente pode ter uma distribuição diferente dos dados. A distribuição de arritmias por cliente no caso não-IID é apresentada na Tabela 4.7 e representa um caso no qual todos os clientes possuem distribuição bastante diferente da distribuição global dos dados que apresenta 10,53% de arritmias. Os dados foram distribuídos entre os clientes de forma que alguns clientes possuíssem apenas gravações com fenômenos clínicos raros e outros possuíssem apenas formas de onda consideradas rotineiras.

Tabela 4.7: Distribuição de amostras apresentando arritmias entre os clientes

Cliente	Porcentagem de Arritmias
0	21,53%
1	7,18%
2	4,59%
3	14,64%
4	0,66%

Assim como no cenário IID, os experimentos foram realizados com 1, 5 e 10 épocas locais, utilizando *batches* de tamanho 16, 128 e 512.

4.5 Critérios de Avaliação

Para o problema de detecção de arritmias em eletrocardiogramas, a sensibilidade (*recall*) é o parâmetro considerado de maior importância, uma vez que mede a capacidade do experimento de identificar o número total de pacientes com arritmias numa população [39]. O *recall* é definido como

$$S = \frac{VP}{VP + FN}, \quad (4.4)$$

onde VP são os verdadeiros positivos (arritmias corretamente identificadas) e FN são os falsos negativos (arritmias identificadas como batimentos normais).

Além do *recall*, a precisão foi utilizada para avaliar a quantidade de amostras classificadas como positivas que realmente representam uma arritmia, indicando a proporção de classificações realizadas de maneira correta [39].

A precisão é definida como

$$P = \frac{VP}{VP + FP}, \quad (4.5)$$

onde FP são os falsos positivos (batimentos normais classificados como arritmias).

Assim, o *recall* foi utilizado em conjunto com a precisão para avaliação do desempenho dos experimentos e comparação de resultados.

Capítulo 5

Experimentos e Resultados

Neste capítulo são apresentados os resultados dos experimentos realizados seguindo a metodologia apresentada no Capítulo 4. São apresentados os resultados obtidos utilizando o modelo centralizado para fins de comparação. Em seguida, são apresentados os resultados do modelo distribuído considerando o caso onde os dados são IID e o caso onde os dados são não-IID.

Os experimentos foram realizados utilizando um computador com processador i5-9600K, GPU Nvidia RTX3080, 64 GB de RAM e sistema operacional Ubuntu 22.04 LTS.

5.1 Modelo Centralizado

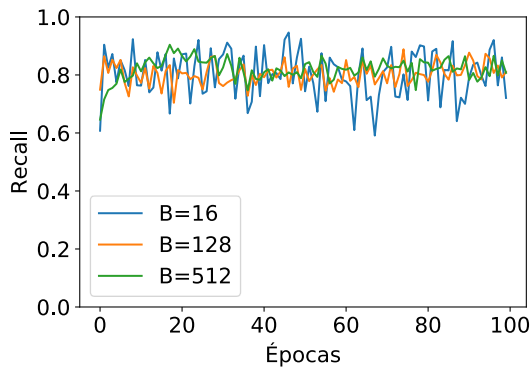
Os resultados para o modelo centralizado são apresentados na Tabela 5.1. A Tabela mostra os valores de *recall* e precisão obtidos para cada tamanho de *batch*. É possível notar que o desempenho do modelo melhora conforme se aumenta o tamanho do *batch*. Esse efeito é observado em outros trabalhos envolvendo CNNs [40]. Porém, destaca-se que um aumento no tamanho do *batch* resulta em maior tempo de treinamento. Dessa forma, é necessário que seja utilizado o *batch* mais adequado à infraestrutura computacional disponível.

A Figura 5.1a apresenta o *recall* obtido ao longo das épocas. Esse *recall* é avaliado em cada época no conjunto de testes. Cada curva representa os resultados obtidos para um tamanho de *batch* diferente. Os tamanhos dos *batches* são apresentados na figura como “B”. Observando a figura, nota-se que um *batch* menor resulta

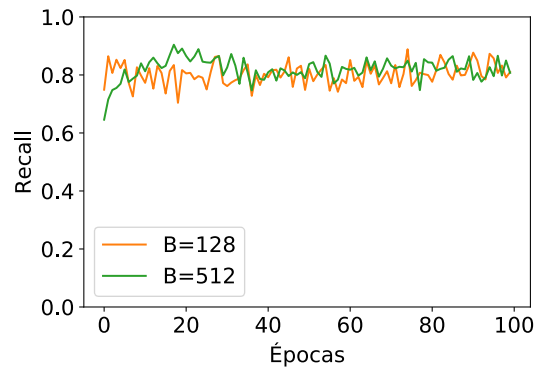
em maior ruído no recall obtido. Esse comportamento é esperado no SGD, visto que, para um *batch* de menor tamanho, há maior ruído no cálculo e são necessários mais passos para se minimizar a função custo. A Figura 5.1b apresenta as curvas de recall excluindo o *batch* de menor tamanho. É possível notar que, mesmo para os maiores *batches*, o resultado obtido é ruidoso e o modelo apresenta dificuldade para atingir a convergência, o que pode ser explicado pelo grande desbalanceamento entre as classes do problema. As curvas do caso centralizado apresentadas na Figura 5.1 são utilizadas como referência para a comparação dos resultados do modelo distribuído.

Tabela 5.1: Resultado dos experimentos para o modelo centralizado.

Batch	Recall	Precisão
16	80,37%	43,37%
128	80,29%	58,03%
512	82,01%	57,34%



(a) Todos os *batches*.



(b) Sem o *batch* de tamanho 16.

Figura 5.1: Recall obtido nos experimentos do modelo centralizado, para diferentes tamanhos de *batch*.

5.2 Aprendizado Federado

Os experimentos com o modelo distribuído, utilizando aprendizado federado, foram realizados considerando dois cenários: o caso ideal com dados IID e o cenário mais realista, com dados não-IID. Todos os experimentos foram realizados com 5

clientes e 100 rodadas de comunicação utilizando o algoritmo *FedAvg* para agregação dos parâmetros no servidor.

5.2.1 Caso Ideal

A Figura 5.2 apresenta os resultados do *recall* obtidos com uma, cinco e dez épocas locais por cliente. Além disso, o resultado de *recall* com *batch* de tamanho 512 da Tabela 5.1 é plotado para comparação. Uma rodada de comunicação corresponde a um ciclo completo de treinamento pelo número definido de épocas locais e o compartilhamento com o servidor dos parâmetros obtidos nesse ciclo. Pela Figura 5.2a, nota-se que o resultado obtido por meio do aprendizado federado para o caso com 1 época é menos ruidoso que o resultado do modelo centralizado apresentado na Figura 5.1.

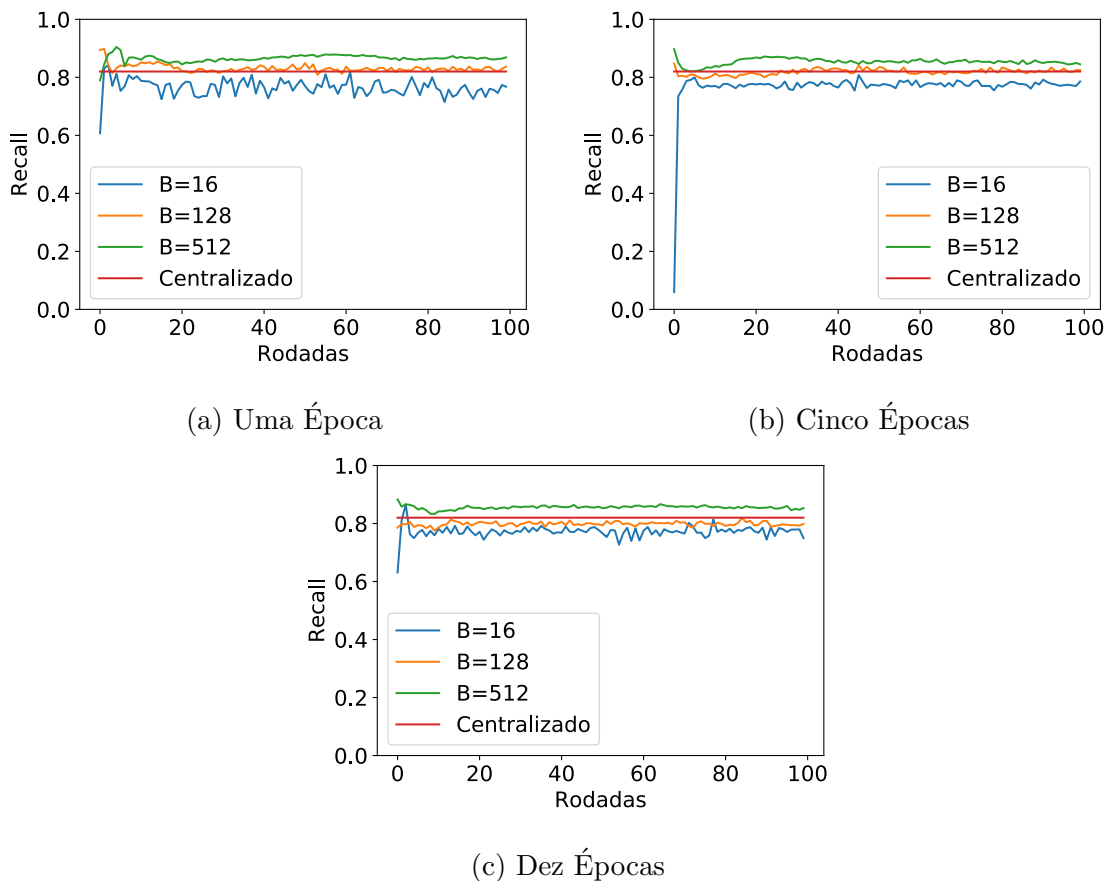


Figura 5.2: Recall obtido nos experimentos do cenário IID, para diferentes tamanhos de *batch*

Os resultados obtidos na Figura 5.2 com os *batches* maiores são melhores do

que o obtido no modelo centralizado. Isso indica que o aprendizado federado pode melhorar o desempenho do modelo, uma vez que os pesos obtidos pelo treinamento de cada cliente são agregados na tentativa de fornecer a melhor solução para o problema.

A Figura 5.3 expõe a precisão obtida nos experimentos para o caso IID, utilizando o valor da precisão da Tabela 5.1 como referência. Comparando essa figura com a Figura 5.2 é possível notar que há um *tradeoff* entre *recall* e precisão. Esse *tradeoff* fica mais evidenciado nos maiores *batches* com um maior número de épocas. Comparando as Figuras 5.2c e 5.3c, por exemplo, é possível notar que as curvas com maior *recall* apresentam menor precisão e vice-versa. Para o *batch* de maior tamanho, é notável que há uma queda na precisão conforme se aumenta o número de épocas. Isso pode ser um indício de que o aumento no tamanho do *batch* pode resultar em uma diminuição no desempenho da rede. Uma das explicações para esse efeito é a redução no tamanho do conjunto de dados dos clientes em relação ao modelo centralizado. Além disso, um *batch* maior pode necessitar de ajustes no *learning rate* para ter o desempenho esperado. Esse efeito pode indicar que existe um ponto de equilíbrio entre *recall* e precisão que resulta no melhor desempenho do modelo. Isso pode ser obtido ao se ajustar o número de épocas e o tamanho do *batch*, não sendo simplesmente uma questão de aumentar ambos.

O resultado da função perda é exposto na Figura 5.4, na qual é possível notar que os maiores *batches* apresentam menor perda e que o *batch* de tamanho 128 tem o melhor desempenho conforme se aumenta o número de épocas. O melhor desempenho do *batch* de tamanho 128 pode ser um indício de que ele está mais próximo do tamanho com maior equilíbrio em relação ao *tradeoff recall-precisão*. Além disso, é notável que um tamanho de *batch* menor causa um resultado mais ruidoso, como já observado no modelo centralizado.

Esses resultados próximos aos resultados obtidos utilizando o modelo centralizado indicam que num cenário onde os dados são IID o aprendizado federado é uma alternativa viável. Porém, devido ao *tradeoff* entre o *recall* e a precisão, pode ser necessário que seja definido um compromisso entre essas duas métricas.

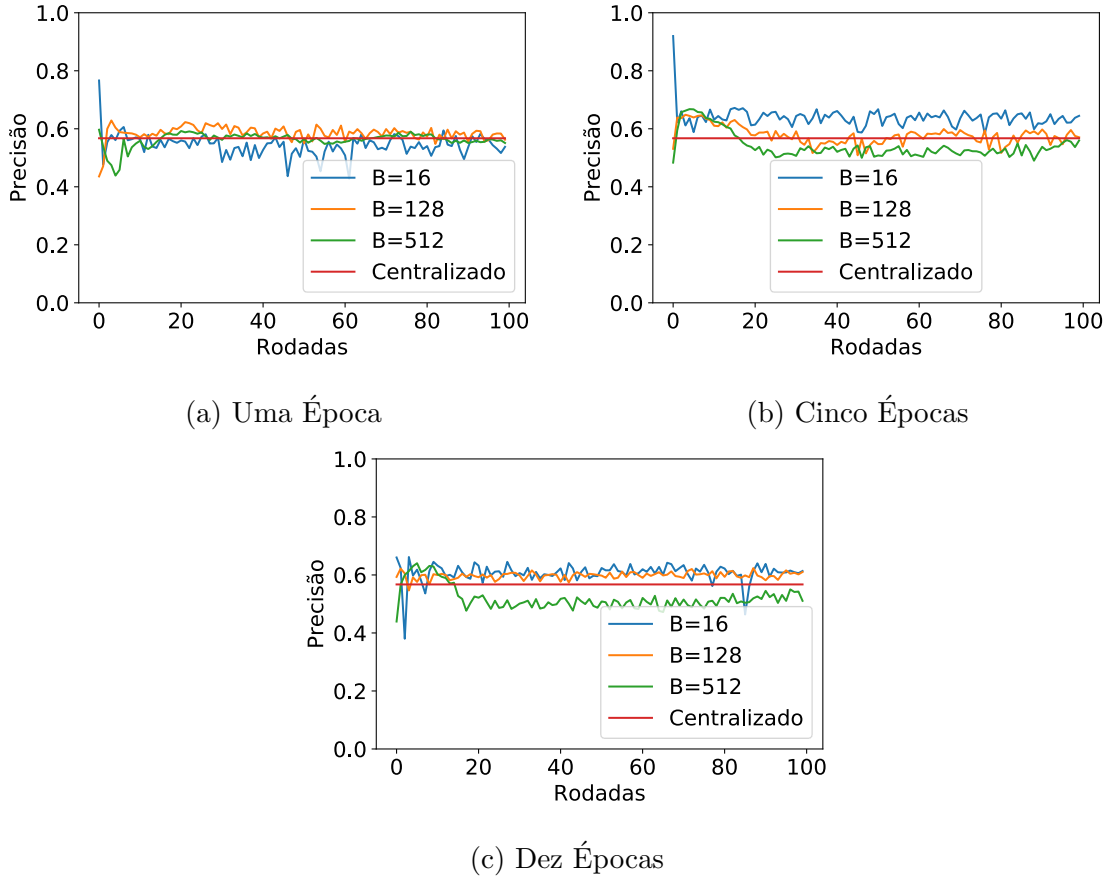


Figura 5.3: Precisão obtida nos experimentos do cenário IID, para diferentes tamanhos de *batch*.

5.2.2 Caso Não-IID

Para o cenário no qual os dados são não-IID, os resultados de recall, precisão e perda são expostos nas Figuras 5.5, 5.6 e 5.7, respectivamente. Os efeitos observados no cenário ideal ocorrem com maior intensidade neste cenário: o ruído é maior com um tamanho de *batch* menor e há um *tradeoff* entre recall e precisão. Dessa forma, observa-se que, apesar dos melhores resultados para o *recall* em comparação com o cenário IID, a precisão cai drasticamente ao se comparar com os cenários anteriores. A rede classifica então mais amostras como “anormais” com uma grande quantidade de falsos positivos. Esse comportamento pode ser explicado pelo fato de que a distribuição não-idêntica entre os clientes faz com que os parâmetros não convirjam para os valores ótimos. Isso ocorre uma vez que o algoritmo de agregação, *FedAvg*, não leva em consideração o quão distante a distribuição de um cliente está da distribuição global dos dados. Além disso, o *oversampling* da classe minoritária

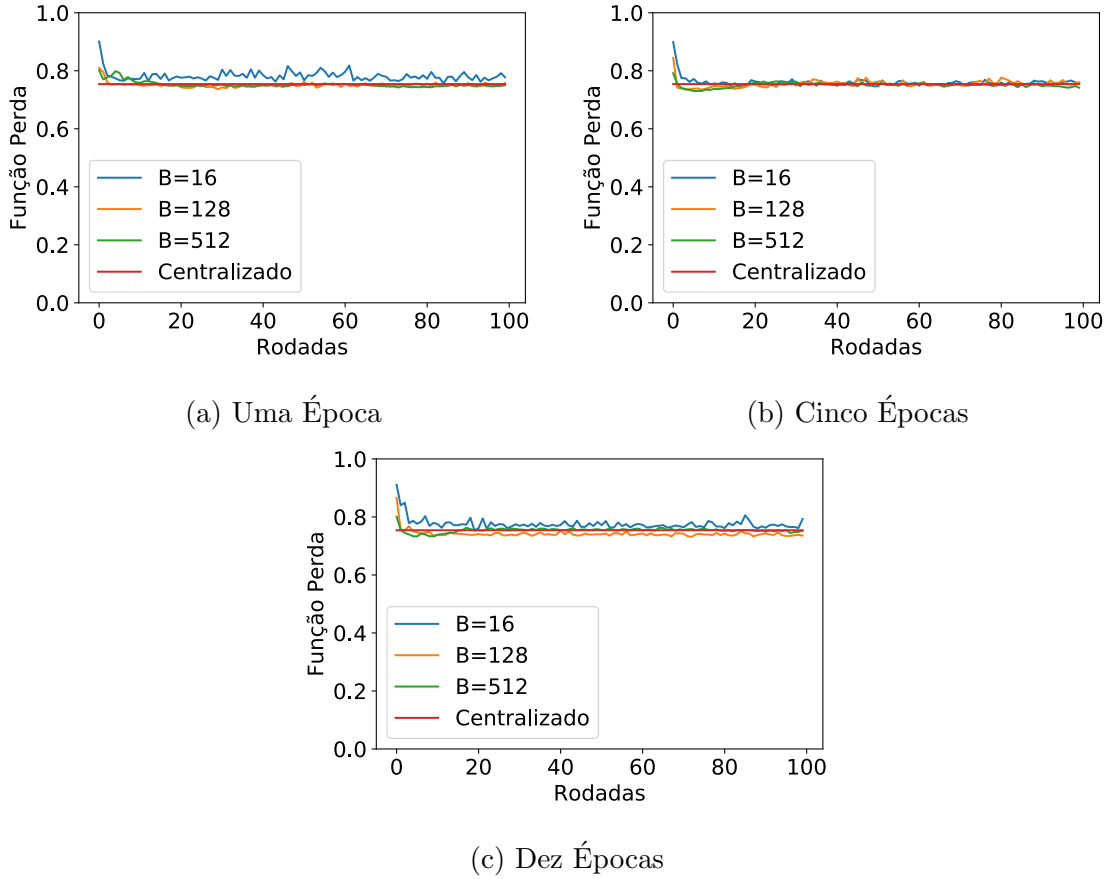


Figura 5.4: Resultado da função perda nos experimentos do cenário IID, para diferentes tamanhos de *batch*.

tem um efeito negativo em relação ao caso centralizado e ao cenário federado ideal. A classe minoritária é aquela com menos amostras. No caso deste trabalho a classe minoritária é a classe “1”, correspondente às arritmias. Como alguns clientes não possuem amostras de arritmias suficientes em comparação com os dados globais, o *oversampling* nesses clientes resulta em um modelo local que sofre de *overfitting* e que prejudica o modelo global.

A Figura 5.8 traz os resultados de recall e precisão ao se reduzir a taxa de *oversampling* para o experimento com *batch* de tamanho 512 e uma época local, comparando ao caso em que a taxa de *oversampling* é mantida a mesma que nos casos anteriores, como no experimento da Figura 5.5. Esse comportamento indica que, para o cenário onde a distribuição dos dados é diferente entre os clientes, podem ser necessárias adaptações no modelo local em cada cliente de forma a garantir o melhor resultado do modelo global. Assim, em um cenário no qual se conhecem

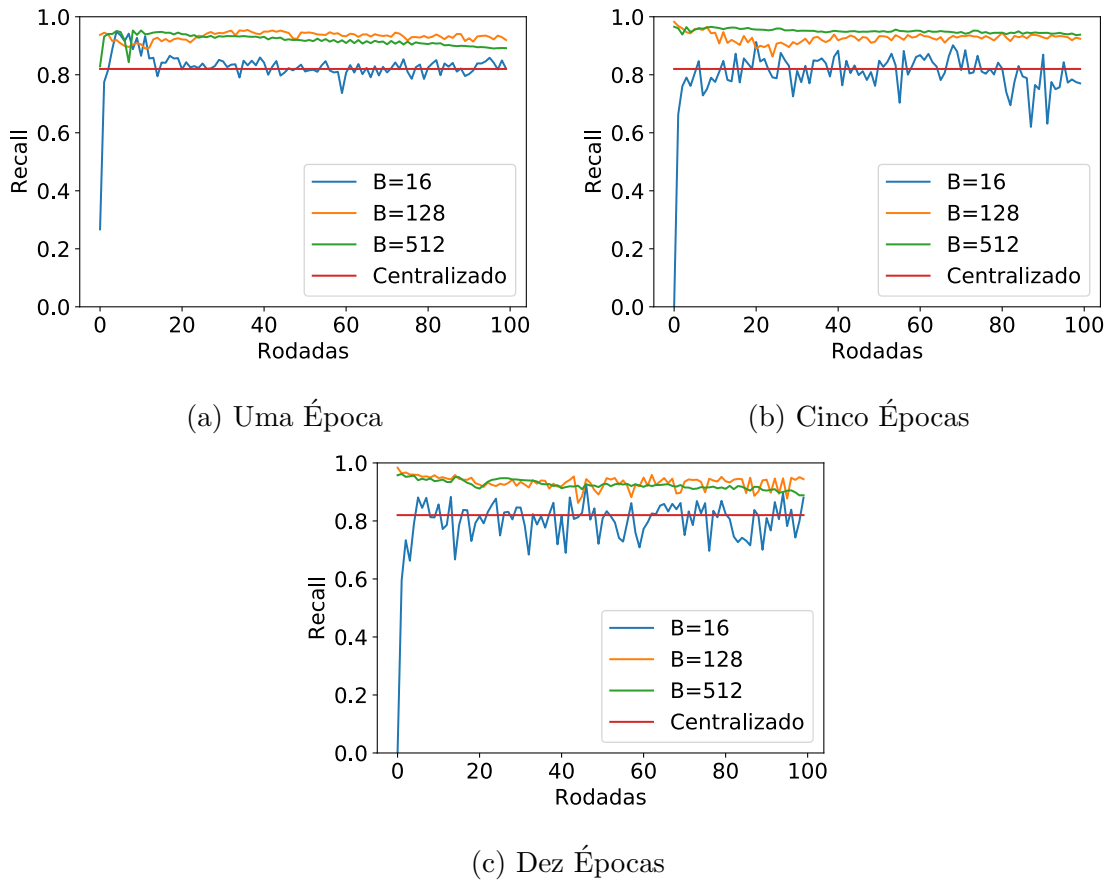
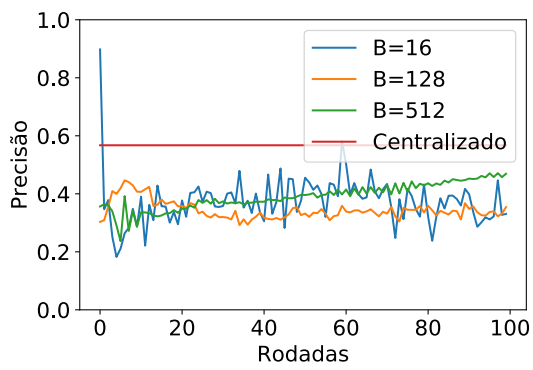
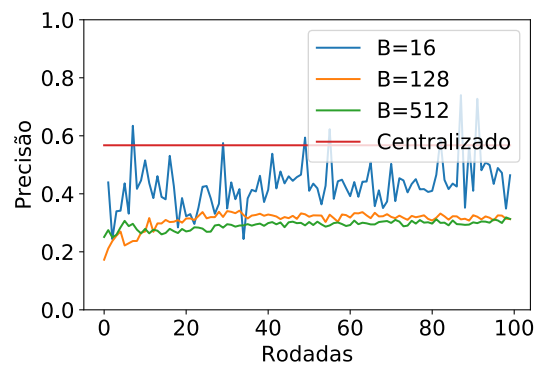


Figura 5.5: Recall obtido nos experimentos do cenário não-IID, para diferentes tamanhos de *batch*

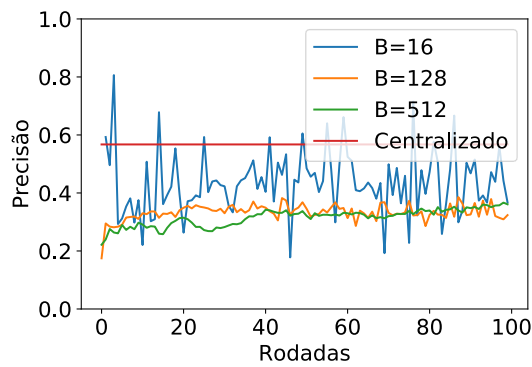
as estatísticas globais do problema, como o problema da incidência de arritmias em uma população, essas pequenas alterações no modelo local são factíveis, pois se tem conhecimento *a priori* das estatísticas dos dados globais (mesmo sem acesso direto aos dados em si). Além disso, alguns algoritmos mais recentes de aprendizado federado, como a versão federada dos otimizadores adaptativos [41], dão maior importância, a cada rodada de comunicação, para os clientes que apresentaram maior contribuição para a melhora do resultado do modelo global. Dessa forma, esses algoritmos podem representar ganhos em relação ao *FedAvg* neste cenário.



(a) Uma Época

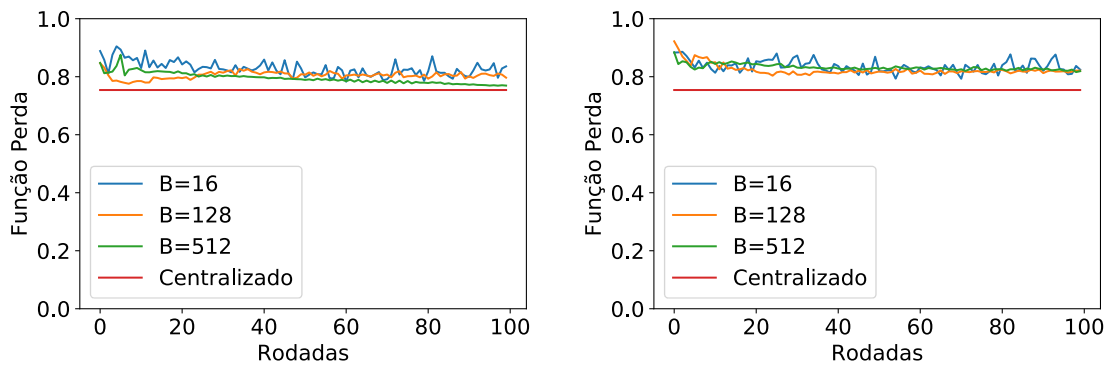


(b) Cinco Épocas



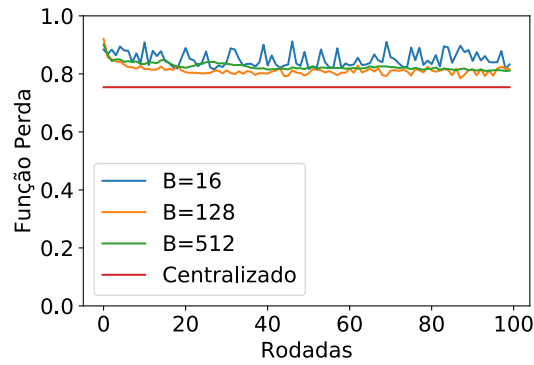
(c) Dez Épocas

Figura 5.6: Precisão obtida nos experimentos do cenário não-IID, para diferentes tamanhos de *batch*.



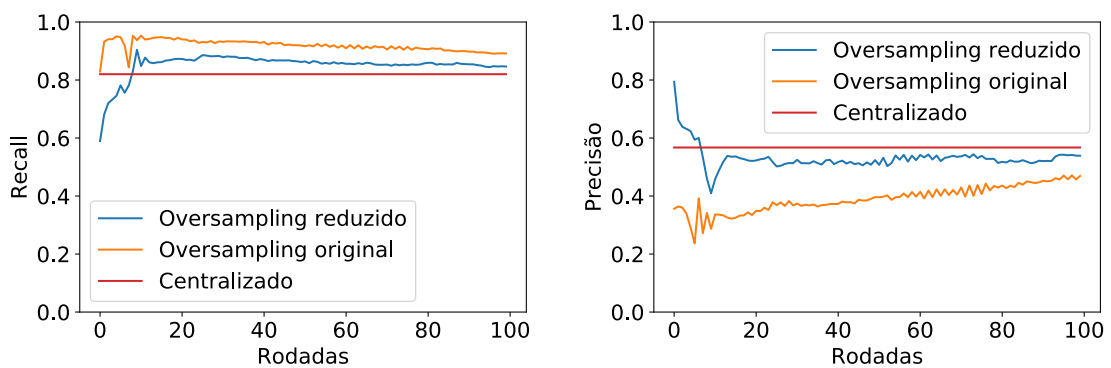
(a) Uma Época

(b) Cinco Épocas



(c) Dez Épocas

Figura 5.7: Resultado da função perda nos experimentos do cenário não-IID, para diferentes tamanhos de *batch*.



(a) Recall

(b) Precisão

Figura 5.8: Experimento com *oversampling* reduzido do caso não-IID, para *batch* de tamanho 512 e uma época local.

Capítulo 6

Conclusão

Neste trabalho foi analisada a viabilidade de um modelo distribuído de detecção de arritmias utilizando aprendizado federado. Os experimentos foram realizados com os dados do MIT-BIH [12] de modo a considerar o caso ideal onde os dados são IID e o caso mais realista com dados não-IID. O modelo centralizado foi utilizado como referência para comparações.

Um modelo de detecção de arritmias pode ser importante para auxiliar médicos no diagnóstico de problemas cardíacos e acelerar a identificação de pacientes que necessitam de maior atenção. Tal modelo poderia se beneficiar do aprendizado federado para utilizar dados de diferentes hospitais e aumentar seu desempenho sem violar a privacidade dos dados dos pacientes.

Com os resultados apresentados neste trabalho, é possível concluir que é viável o desenvolvimento de um modelo computacional baseado em aprendizado federado para a identificação de arritmias em eletrocardiogramas. No cenário ideal, o modelo distribuído apresentou resultados compatíveis e até melhores que o modelo centralizado. Isso indica que, em alguns casos o aprendizado federado pode atuar aumentando o desempenho da rede.

O cenário não-IID representa um desafio no desenvolvimento de modelos de detecção de arritmias baseados em aprendizado federado. Isso ocorre pois clientes com dados distribuídos de formas muito diferentes entre si acabam resultando em um modelo que não se ajusta bem à distribuição global dos dados. Porém, mesmo neste cenário, pequenas alterações nos modelos locais, como a redução do *oversampling*, geram resultados comparáveis àqueles obtidos no cenário ideal e no modelo

centralizado. Além disso, tais problemas podem ser resolvidos utilizando outros algoritmos de agregação, como a versão federada de otimizadores adaptativos [41].

É importante notar que este trabalho considerou condições ideais da rede de comunicação, focando apenas o desempenho dos modelos. Assim, problemas de comunicação devem ser considerados em uma implementação do aprendizado federado, pois podem trazer desafios em relação ao desempenho e custo de provisionamento da rede. Ainda assim, as vantagens em relação à privacidade e segurança e as evidências apontadas neste e em outros trabalhos indicam que o aprendizado federado é uma importante técnica, que pode ser utilizada para a identificação de arritmias em eletrocardiogramas. A análise do desempenho de um modelo em condições reais de rede pode ser uma direção para trabalhos futuros.

Como mencionado anteriormente, um dos desafios do aprendizado federado reside na utilização de algoritmos de agregação que possibilitem um desempenho melhor em cenários de dados não-IID. Dessa forma, a análise do desempenho e o desenvolvimento de outros algoritmos que funcionem melhor no cenário não-IID se mostra uma interessante direção para trabalhos futuros. Além disso, o desenvolvimento de estratégias que possibilitem ajustes nos modelos locais dos clientes, de maneira independente, é outra possível direção para prover um melhor modelo global.

Referências Bibliográficas

- [1] LECUN, Y., BENGIO, Y., HINTON, G., “Deep learning”, *Nature*, v. 521, 2015.
- [2] LEE, K.-F., *Inteligência Artificial*. Rio de Janeiro, Editora Globo S.A., 2019.
- [3] KAVRE, M., GADEKAR, A., GADHADE, Y., “Internet of Things (IoT): A Survey”. In: *2019 IEEE Pune Section International Conference (PuneCon)*, pp. 1–6, 2019.
- [4] KONEČNÝ, J., MCMAHAN, H. B., RAMAGE, D., *et al.*, “Federated Optimization: Distributed Machine Learning for On-Device Intelligence”, *CoRR*, v. abs/1610.02527, 2016.
- [5] SAKIB, S., FOUUDA, M. M., MD FADLULLAH, Z., *et al.*, “Asynchronous Federated Learning-based ECG Analysis for Arrhythmia Detection”. In: *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 277–282, 2021.
- [6] ZHANG, M., WANG, Y., LUO, T., “Federated Learning for Arrhythmia Detection of Non-IID ECG”. In: *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pp. 1176–1180, 2020.
- [7] LI, D., ZHANG, J., ZHANG, Q., *et al.*, “Classification of ECG signals based on 1D convolution neural network”. In: *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6, 2017.
- [8] AY, H., P, R., M, H., *et al.*, “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network”, *Nat Med*, v. 25, pp. 65–69, 2019.

- [9] MA, L., TANG, R., LUO, J., *et al.*, “Personalized Federated Learning for ECG Classification Based on Feature Alignment”, *Security and Communication Networks*, v. 2021, 2021.
- [10] GAO, Y., KIM, M., ABUADBBA, S., *et al.*, “End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things”. In: *2020 International Symposium on Reliable Distributed Systems (SRDS)*, pp. 91–100, 2020.
- [11] AAMI, A., “Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms”, *American National Standards Institute, Inc. (ANSI), ANSI/AAMI/ISO*, v. EC57, 1998-(R)2008, 2008.
- [12] MOODY, G., MARK, R., “The impact of the MIT-BIH Arrhythmia Database”, *IEEE Engineering in Medicine and Biology Magazine*, v. 20, n. 3, pp. 45–50, 2001.
- [13] CHAZAL, P. D., O’DWYER, M., REILLY, R., “Automatic classification of heartbeats using ECG morphology and heartbeat interval features”, *IEEE Transactions on Biomedical Engineering*, v. 51, n. 7, pp. 1196–1206, 2004.
- [14] BEUTEL, D. J., TOPAL, T., MATHUR, A., *et al.*, “Flower: A Friendly Federated Learning Research Framework”, *CoRR*, v. abs/2007.14390, 2020.
- [15] GURNEY, K., *An Introduction to Neural Networks*. USA, Taylor and Francis, Inc., 1997.
- [16] HAYKIN, S. S., *Neural networks and learning machines*. 3 ed. Upper Saddle River, NJ, Pearson Education, 2009.
- [17] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learning*. MIT Press, 2016.
- [18] RUDER, S., “An overview of gradient descent optimization algorithms”, *CoRR*, v. abs/1609.04747, 2016.
- [19] TIBSHIRANI, R., “Regression Shrinkage and Selection via the Lasso”, *Journal of the Royal Statistical Society. Series B (Methodological)*, v. 58, n. 1, pp. 267–288, 1996.

- [20] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., *et al.*, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, *Journal of Machine Learning Research*, v. 15, n. 56, pp. 1929–1958, 2014.
- [21] BJORCK, J., GOMES, C., SELMAN, B., *et al.*, “Understanding Batch Normalization”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, p. 7705–7716, Red Hook, NY, USA, 2018.
- [22] LUO, P., WANG, X., SHAO, W., *et al.*, “Towards Understanding Regularization in Batch Normalization”, *CoRR*, v. abs/1809.00846, 2018.
- [23] YAMASHITA, R., NISHIO, M., DO, R. K. G., *et al.*, “Convolutional neural networks: an overview and application in radiology”, *Insights into Imaging*, v. 9, 8 2018.
- [24] HAYKIN, S., VEEN, B. V., *Signals and Systems*. 2 ed. USA, John Wiley & Sons, Inc., 2002.
- [25] KONEČNÝ, J., MCMAHAN, H. B., YU, F. X., *et al.*, “Federated Learning: Strategies for Improving Communication Efficiency”, *CoRR*, v. abs/1610.05492, 2016.
- [26] MCMAHAN, H. B., MOORE, E., RAMAGE, D., *et al.*, “Federated Learning of Deep Networks using Model Averaging”, *CoRR*, v. abs/1602.05629, 2016.
- [27] GOLDBERGER A., A., L., GLASS, L., *et al.*, “PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals”, *Circulation [Online]*, v. 101, n. 23, pp. e215–e220, 2000.
- [28] LUZ, E. J. D. S., NUNES, T. M., de Albuquerque, V. H. C., *et al.*, “ECG arrhythmia classification based on optimum-path forest”, *Expert Systems with Applications*, v. 40, n. 9, pp. 3561–3573, 2013.
- [29] YE, C., COIMBRA, M., KUMAR, B., “Arrhythmia Detection and Classification using Morphological and Dynamic Features of ECG Signals”, *Conference proceedings : ... Annual International Conference of the IEEE Engineering in*

- Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, v. 2010, pp. 1918–21, 08 2010.
- [30] KURNIAWAN, A., ANANDA, PRADANGGAPASTI, F. N., *et al.*, “Arrhythmia Classification on Electrocardiogram Signal Using Convolution Neural Network Based on Frequency Spectrum”. In: *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pp. 29–33, 2020.
- [31] PASZKE, A., GROSS, S., MASSA, F., *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035, 2019.
- [32] RAZA, A., TRAN, K. P., KOEHL, L., *et al.*, “Designing ECG monitoring healthcare system with federated transfer learning and explainable AI”, *Knowledge-Based Systems*, v. 236, pp. 107763, 2022.
- [33] VAN ROSSUM, G., DRAKE, F. L., *Python 3 Reference Manual*. Scotts Valley, CA, CreateSpace, 2009.
- [34] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., *et al.*, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.
- [35] MCKINNEY, W., OTHERS, “Data structures for statistical computing in python”. In: *Proceedings of the 9th Python in Science Conference*, v. 445, pp. 51–56, Austin, TX, 2010.
- [36] HARRIS, C. R., MILLMAN, K. J., WALT, S. J. V. D., *et al.*, “Array programming with NumPy”, *Nature*, v. 585, n. 7825, pp. 357–362, Sep. 2020.
- [37] PEREZ-RIERA, A. R., ABREU, L. C. D., BARBOSA-BARROS, R., *et al.*, “R-Peak Time: An Electrocardiographic Parameter with Multiple Clinical Applications”, *Annals of noninvasive electrocardiology : the official journal of the International Society for Holter and Noninvasive Electrocardiology, Inc*, v. 21, n. 1, pp. 9–10, 2016.

- [38] SUTSKEVER, I., MARTENS, J., DAHL, G., *et al.*, “On the importance of initialization and momentum in deep learning”. In: Dasgupta, S., McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, v. 28, *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013.
- [39] POWERS, D., “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation”, *Mach. Learn. Technol.*, v. 2, 01 2008.
- [40] RADIUK, P., “Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets”, *Information Technology and Management Science*, v. 20, pp. 20–24, 12 2017.
- [41] REDDI, S. J., CHARLES, Z., ZAHEER, M., *et al.*, “Adaptive Federated Optimization”, *CoRR*, v. abs/2003.00295, 2020.