Relatório Final - Iniciação Científica

Classificação de arritmias cardíacas utilizando *data augmentation*

Bolsista: Eduardo Paes Leme Jaqueira
Orientador: Magno Teófilo Madeira da Silva
Coorientador: Renato Candido

Processo Nº: 127301/2023-2

Universidade de São Paulo (USP)
Escola Politécnica
Departamento de Engenharia de Sistemas Eletrônicos (PSI)
Av. Prof. Luciano Gualberto, 158, trav. 3
São Paulo - SP
05508-010

São Paulo

Setembro de 2024

Sumário

Resumo

Este relatório final tem como principal propósito apresentar os resultados obtidos para o processo de *data augmentation* para bancos de dados de arritmias cardíacas com o objetivo melhorar a acurácia da rede classificadora dessas arritmias. Com esse objetivo são exploradas duas arquiteturas de redes gerativas e comparados seus resutaldos. Primeiramente são apresentados as motivações e os objetivos dessa pesquisa. Posteriormente são apresentados os resultados alcançados com as duas redes consideradas para uma das classes do problema e suas eficácias comparadas. Tendo em vista a rede com melhor desempenho na geração do sinal proposto anteriormente, a mesma arquitetura é então utilizada para a geração de sinais de uma classe mais complexa e menos representativa. Para a geração de sinais dessa classe é realizado um estudo da classe e são determinados sinais definidos como *templates* para serem utilizados como métrica de comparação. Após o treinamento da rede, os dados gerados são adicionados ao banco de dados desbalanceado e a rede classificadora é então retreinada. Os resultados obtidos pelo processo de *data augmentation* são comparados com os resultados obtidos previamente. Ao final, é apresentada uma conclusão sobre o projeto, comparando os resultados obtidos e com os objetivos propostos para essa pesquisa. O relatório termina com um apêndice contendo os fundamentos teóricos de redes neurais.

1 Introdução

A classificação de arritmias cardíacas depende de uma análise precisa da atividade elétrica do coração, registrada no sinal de eletrocardiograma (ECG) [1]. O sinal de ECG contém informações clínicas que permitem que essa tarefa seja realizada por especialistas qualificados, o que pode ser demorado e é inviável quando se necessita de resultados imediatos. Diante disso, soluções baseadas em técnicas de aprendizado de máquina para a classificação automática de arritmias cardíacas têm sido amplamente propostas na literatura [1–11]. As altas taxas de erro dessas soluções fazem com que elas ainda estejam longe de serem empregadas na prática. Por isso, essa área de pesquisa ainda desperta interesse na comunidade científica. A título de referência, na EUSIPCO (*European Signal Processing Conference*) 2019, as seções técnicas intituladas "ECG & Cardio" e "ECG Processing" foram dedicadas a sinais de ECG e o artigo [9] foi apresentado na seção "Neural Networks and Biomedicine". Além disso, desde o início da década de 2000, a PhysioNet/Cinc tem criado desafios voltados para diagnóstico automatizado de doenças cardiovasculares, incluindo arritmias cardíacas. O mais recente foi proposto em 2022 com o objetivo de desenvolver soluções automáticas para detectar anormalidades cardíacas a partir de gravações de fonocardiograma [12].

Muitas das soluções existentes na literatura não seguem as recomendações da *Association for the Advancement of Medical Instrumentation* (AAMI) [13] e utilizam dados de ECG dos mesmos pacientes tanto no conjunto de teste, quanto no de treinamento. Essas soluções até atingem baixas taxas de erro, mas não são clinicamente realistas. Isso ocorre porque na prática o sistema de classificação deve ser utilizado em pacientes, cujos dados não foram usados no treinamento.

Dentre os classificadores que seguem uma divisão realista dos pacientes, os trabalhos descritos resumidamente a seguir são considerados como estado da arte na literatura. Em [2], desenvolveu-se uma análise de discriminante linear (*linear discriminant analysis* – LDA) para classificar os batimentos em um problema de cinco classes, a partir de uma determinada extração de características. Em [3], características temporais, morfológicas e estatísticas do sinal foram utilizadas junto a um algoritmo de *sequential forward floating* search para encontrar combinações de características ótimas, com classificadores baseados em LDA e redes perceptron multicamada (*multilayer perceptron* – MLP). Em [4], foi proposta uma LDA com pesos e extração de características de intervalos RR e características morfológicas usando transformada de wavelet. Em [5], o ECG foi representado em três dimensões por meio do temporal vectorcardiogram (TVCG). Essa representação foi usada em redes complexas para extração de características, que, por sua vez, foram consideradas como entrada de um classificador baseado em máquina de vetores de suporte (*support vector machine* – SVM) . Por fim, [8] aprimorou o modelo de [5] com o uso de *particle swarm optimization* para seleção de características.

Na iniciação científica (IC) da aluna Natália Nagata, realizada na EPSUP entre março de 2020 e agosto de 2021 com bolsa FAPESP (Processo número 2019/26911-6), foram propostas soluções baseadas em redes MLP, redes neurais recorrentes (*recurrent neural network* – RNN), LDA e combinações desses métodos para classificação automática de arritmias cardíacas. A fim de se obter resultados clinicamente realistas para o diagnóstico, dados de um mesmo paciente não foram usados simultaneamente nas fases de treinamento e de teste. Foram utilizados sinais de ECG do banco de dados MIT-BIH *Arrhythmia Database* (MITDB) [14, 15], levando em conta o agrupamento das anotações em quatro classes [13]: batimentos do nó sinoatrial (N), supraventriculares ectópicos (S), ventriculares ectópicos (V) e fusão de batimentos normais e ventriculares ectópicos (F). Dentre as soluções propostas, a combinação da rede MLP com a LDA apresentou um desempenho melhor em relação aos modelos individuais e alcançou métricas de classificação superiores às da literatura para as classes S e V [11]. Para lidar com o desbalanceamento do número de dados entre as classes, os dados da classe N foram subamostrados e as funções custo foram ponderadas de forma proporcional ao inverso da quantidade de dados de cada classe. Os resultados dessa IC são animadores, pois se aproximam dos resultados da literatura, sendo inclusive superiores para as classes S e V.

Apesar de terem sido consideradas técnicas usuais para lidar com classes desbalanceadas, acredita-se que métricas de classificação superiores só podem ser alcançadas com um banco de dados balanceado. Os resultados dessa IC foram publicados em [11,16–19]. Apesar do MITDB ser um dos bancos de dados mais utilizados na literatura, ele apresenta um grande desbalanceamento entre as classes, como pode ser visto na Figura 1. Verifica-se nessa figura que 92% dos batimentos do MITDB pertencem à classe N e os 8% restantes são divididos dentre as outras três classes de arritmia consideradas.

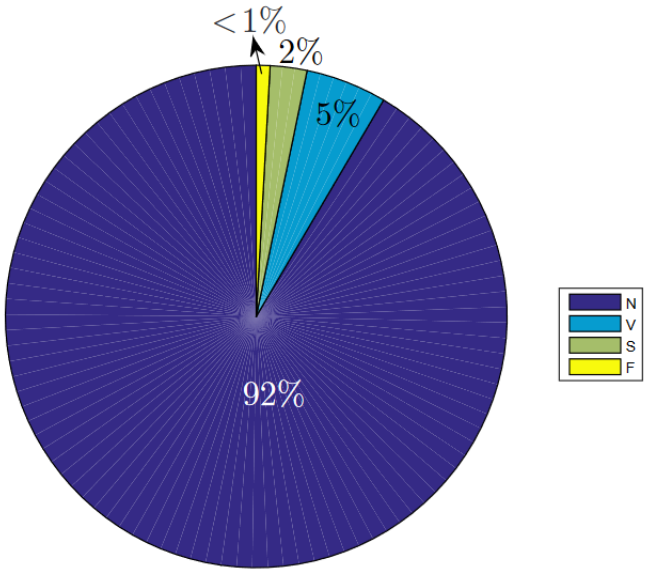


Figura 1: Distribuição das cinco classes de batimentos do MITDB.

Para lidar com dados desbalanceados, existem várias técnicas na literatura, descritas resumidamente a seguir. A subamostragem consiste na exclusão de exemplos da classe sobre-representada enquanto a sobreamostragem consiste na adição de exemplos sintéticos por meio da amostragem aleatória dos exemplos da classe minoritária. Em geral, evita-se usar a subamostragem já que com essa técnica, dados são desprezados [32]. A sobreamostragem é mais utilizada, sendo que a técnica SMOTE (*Synthetic Minority Over-sampling Technique*), proposta em [22], merece destaque. Outra técnica frequentemente utilizada na literatura consiste na inclusão de pesos na função custo, calculados na proporção inversa do número de dados de cada classe, como feito em [11,23]. Uma outra solução que tem ganhado destaque nos últimos anos treina um modelo gerativo para gerar exemplos da classe sub-representada. Considerando um conjunto de dados com distribuição p_{data}, o modelo gerativo usa esses dados no treinamento para aprender a representar uma estimativa dessa distribuição. Essas estimativas, por sua vez, têm distribuição p_{model}. Em alguns casos, o modelo estima p_{model} explicitamente. Em outros, o modelo só é capaz de gerar amostras da distribuição p_{model}. Há ainda modelos que são capazes de realizar as duas tarefas. Neste contexto, a rede gerativa adversária (*Generative Adversarial Network* – GAN) [24, 25] e o autocodificador variacional (*Variational Autoencoder* – VAE) [26, 27] têm sido usados recentemente para gerar amostras da distribuição p_{model}, balanceando bancos de dados a partir da inclusão de dados sintéticos. Essa técnica é conhecida em aprendizado de máquina como *data augmentation*.

Na classificação de arritmias cardíacas, a técnica de *data augmentation* com modelos gerativos também tem sido usada para gerar dados sintéticos de sinais de ECG com o objetivo de balancear os bancos de dados e com isso alcançar melhores métricas de classificação [28–31]. Em [31], por exemplo, uma GAN foi utilizada para gerar sinais realistas de ECG de uma classe minoritária do MITDB. Com o banco de dados mais balanceado, conseguiu-se melhorar o desempenho de classificação. É neste contexto que se insere este projeto de IC. Inicialmente, pretende-se abordar conceitos de aprendizado de máquina, com foco na combinação de uma rede MLP com uma LDA, de modo a reproduzir os resultados de [11]. Em seguida, pretende-se usar GAN e VAE para se chegar a uma versão mais balanceada do MITDB com o objetivo de melhorar os resultados previamente atingidos. A seguir estão os objetivos do plano de pesquisa inicial.

1.1 Plano de trabalho e cronograma

Os objetivos principais estabelecidos nesse projeto de pesquisa são:

1. Pré-processamento e extração de características do sinal de ECG. O objetivo é entender e implementar um algoritmo para detecção do complexo QRS do sinal de ECG como o algoritmo de Pan-Tompkins [48] e estudar as características (entradas) utilizadas em [2,11] para classificação de arritmias cardíacas;

2. Estudo de técnicas de aprendizagem de máquina. Apesar do aluno ter feito uma disciplina de graduação

que aborda redes neurais, é importante que ele se aprofunde e estude técnicas adicionais como as redes neurais convolucionais, que eventualmente podem ser usadas nos modelos gerativos. Para isso, a ideia é que ele faça alguns módulos do curso *Introduction to Deep Learning* minstrado por Andrew Ng disponível no *Coursera*;

3. Estudo e implementação das redes MLP, LDA e da combinação desses modelos como feito em [11] para detecção e classificação de arritmias cardíacas. Pretende-se implementar as soluções em Python utilizando o PyTorch. A ideia aqui é que o aluno consiga reproduzir os resultados de [11];
4. Estudo e implementação de diferentes versões da GAN para geração de sinais de ECG sintéticos;
5. Estudo e implementação do VAE para geração de sinais de ECG sintéticos;
6. Uso da GAN e VAE para balancear o banco de dados na busca de melhores resultados na classificação;
7. Análise e comparação dos resultados;
8. Redação do relatório final.

Com esses objetivos em mente, primeiramente será apresentado a seguir o funcionamento simplificado de um sinal de eletrocardiograma (ECG), antes de apresentar as abordagens exploradas para o desenvolvimento dos sinais sintéticos.

1.2 Eletrocardiograma

Um eletrocardiograma (ECG) é um procedimento médico não invasivo que registra a atividade elétrica do coração ao longo do tempo [64,65]. É uma ferramenta diagnóstica fundamental para avaliar a função cardíaca e identificar possíveis anormalidades no ritmo e na condução elétrica do coração [64]. Durante um ECG, eletrodos são colocados na pele do paciente, geralmente no peito, braços e pernas. Esses eletrodos detectam as correntes elétricas geradas pelo músculo cardíaco durante cada batimento cardíaco. Essas correntes são amplificadas e registradas como padrões de ondas no papel ou em um monitor.

Os principais componentes de um traçado de ECG incluem [66]:

1. Onda P: Representa a despolarização atrial, indicando a contração dos átrios;
2. Complexo QRS: Indica a despolarização dos ventrículos, sinalizando a contração ventricular;
3. Onda T: Reflete a repolarização ventricular, indicando o relaxamento dos ventrículos.

Além disso, há outros elementos e características que podem ser observados em um ECG, como intervalos de tempo específicos e segmentos que indicam a propagação do impulso elétrico pelo coração [66].

O ECG é utilizado para diagnosticar uma variedade de condições cardíacas, incluindo arritmias, infarto do miocárdio, distúrbios de condução e hipertrofia ventricular. Em específico para essa pesquisa, a interpretação do complexo QRS, que envolve a análise de sua morfologia, duração e amplitude, permite identificar os outros elementos do ECG, como as ondas P e T. Alterações no complexo QRS podem indicar várias condições cardíacas, incluindo distúrbios de condução, como em específico a arritmia cardíaca.

Um batimento cardíaco normal registrado no sinal de ECG exibe a forma de onda destacada na Figura 2, onde seus intervalos e segmentos mais relevantes estão delineados [67]. Esse sinal foi produzido pelo programa ecgsyn.m [68] e retirado de [11].

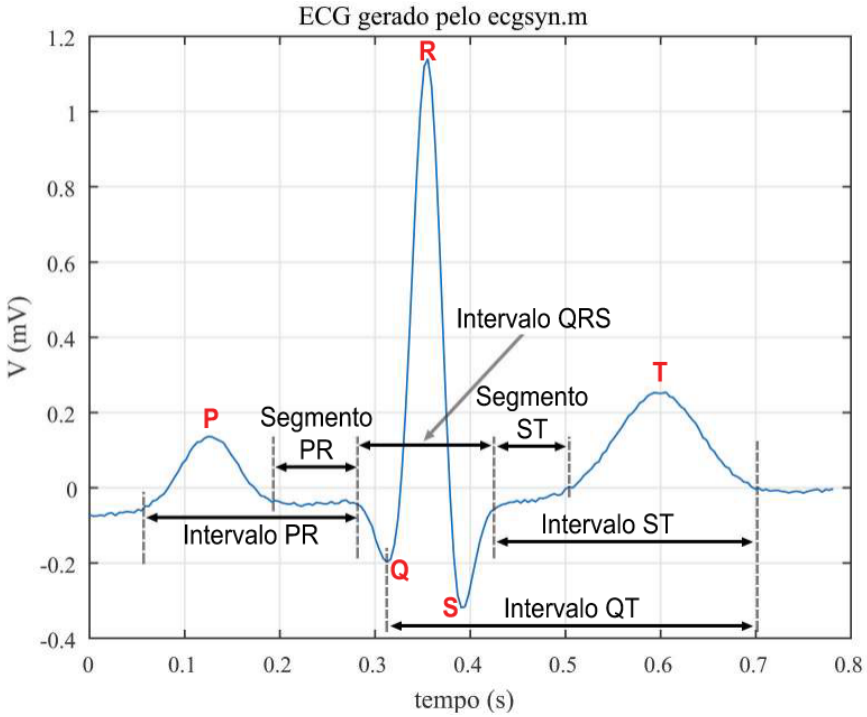


Figura 2: Sinal exemplo de ECG com delimitação de seus principais componentes.

Em resumo, o complexo QRS é uma parte crucial da análise do ECG e fornece informações valiosas sobre a função elétrica dos ventrículos cardíacos.

Logo, estabelecidos os fundamentos teóricos para a construção do projeto, serão apresentado a seguir os resultados obtidos na geração de sinais sintéticos do ECG utilizando a rede do tipo GAN.

2 Rede GAN para geração de sinal de ECG

Primeiramente, para o estudo da classificação dos sinais de ECG, de acordo com as classes determinadas (N, S, V, F) [13], foi realizada uma análise e reprodução do trabalho desenvolvido anteriormente em [11]. Nessa pesquisa, os sinais foram divididos nas respectivas classes, com a análise de 3 batimentos em conjunto, fazendo uso de uma rede MLP com LDA (*linear discriminant analysis*) [33, 34] utilizando a linguagem de programação *Python* e a biblioteca para aprendizado de máquina, *Tensorflow*. Na reprodução dos resultados, foi utilizada a mesma linguagem, porém adaptando os códigos para a bilbioteca *PyTorch*.

2.1 Classificador MLP-LDA

Nesse estudo, os dados de entrada foram selecionados de forma a tentar realizar um balanceamento das classes, além de ser utilizada a função Entropia Cruzada Categórica, fazendo o uso de pesos para diminuir o efeito do desbalanceamento pré-existente no dataset em favor da classe N [11]. A estrutura da rede MLP utilizada é dada por [960-32(ReLU)-16(ReLU)-4(Softmax)].

Para o treinamento dessa rede foi utilizado um passo de adaptação $\eta = 0{,}001$, 250 épocas e o Adam como otimizador. A matriz de confusão é apresentada na Figura 3. Nesse caso, os índices apresentados na horizontal, na parte inferior, representam a classe real dos sinais analisados, enquanto os índices na vertical, no canto esquerdo, indicam a classificação atribuída a cada batimento pela rede.

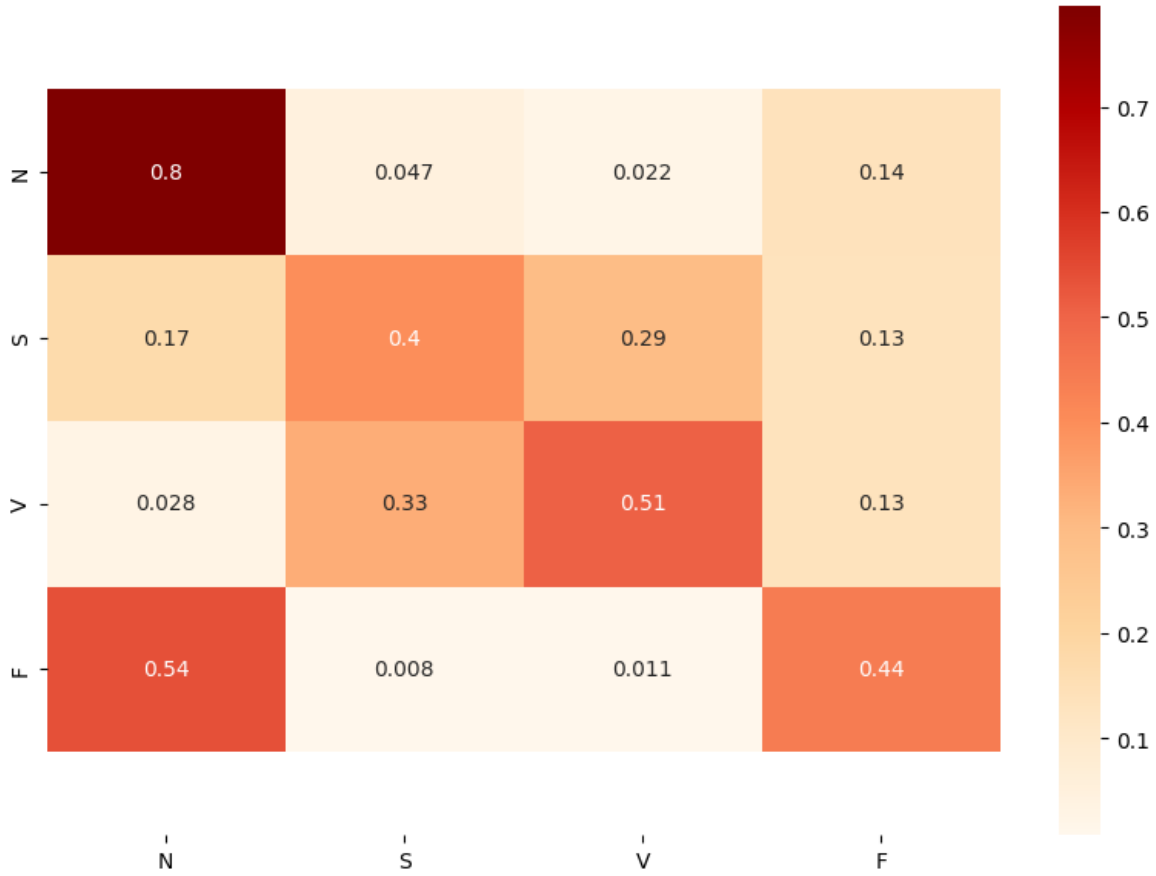


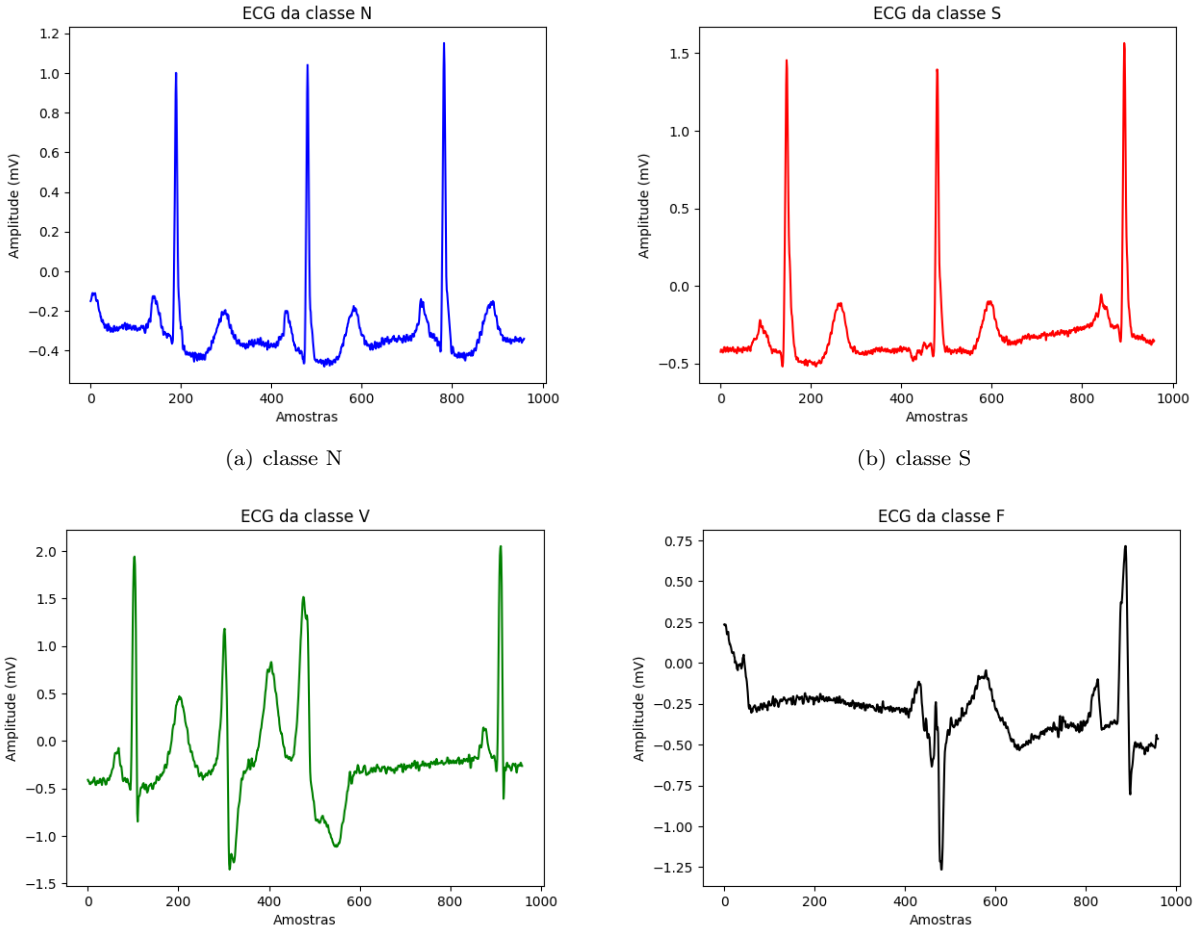
Figura 3: Tabela de relação entre classe real e classificação atribuída pela rede MLP-LDA.

Assim, é possível perceber como a rede apresenta maior facilidade para classificar batimentos da classe N, apresentando um acerto de 80% para essa classe e um acerto menor para as outras classes, entre 40% e 50% aproximadamente. Isso ocorre devido ao desbalanceamento das classes, onde se insere esta pesquisa para geração de sinais sintéticos para as outras classes.

Para a geração de sinais de ECG é proposto o uso de uma rede do tipo GAN como no caso da geração do exemplo do sinal senoidal descrito no Apêndice A. Entretanto, a construção de sinais de ECG é consideravelmente mais trabalhosa que a de sinais senoidais apresentada, tendo em vista a variedade encontrada nos sinais desse tipo. Desse modo, primeiramente, trabalhou-se no pré-processamento dos dados, de modo a obter um banco de dados mais homogêneo para facilitar o treinamento da rede como apresentado a seguir. Utilizou-se a linguagem de programação *Python* e a biblioteca *PyTorch* para a construção das redes pretendidas e tratamento de dados, assim como a biblioteca *NumPy*.

2.2 Organização do *Dataset*

Fazendo uso da base do banco de dados MITDB para arritmias cardíacas, separou-se os batimentos do banco de dados de acordo com sua classe. A Figura 4, apresenta esses batimentos.



(a) classe N

(b) classe S

(c) classe V

(d) classe F

Figura 4: Sinais de ECG separados em diferentes classificações.

Esse sinais foram então normalizados, dividindo-os pela máxima amplitude do módulo cada sinal, de forma a estarem contidos entre -1 e 1.

Como primeira abordagem, foi explorada a geração de sinais do tipo N, devido a sua regularidade e quantidade no banco de dados real. Assim, os sinais N do banco de dados, representados anteriormente por 12178 vetores de tamanho 960, foram divididos em 3, fornecendo então 36534 vetores de tamanho 320. A seguir será apresentada a metodologia aplicada para geração dos sinais e avaliação dos modelos.

2.3 Metodologia

Para gerar os sinais de ECG pretendidos, propôs-se a utilização de uma rede do tipo GAN, onde o treinamento adversário aumentaria a qualidade dos sinais gerados pelo gerador da rede com o passar da épocas no treinamento [50]. Assim, o gerador foi então alimentado com um vetor espaço latente de tamanho $(100, 1)$ e forneceu uma saída de tamanho $(320, 1)$. As saídas criadas pelo gerador foram então apresentadas à rede do discriminador em conjunto com os sinais reais do banco de dados. Nessa rede, o discriminador então fornece uma saída, baseada na função custo escolhida, determinando se cada sinal é real ou sintético.

Com o intuito de determinar a eficácia das redes desenvolvidas para a geração dos sinais sintéticos, utilizaram-se as métricas propostas nos Métodos 2, 3 e 4 de [35]. Esses métodos utilizam uma função de distância entre os sinais de batimento gerados e um sinal real definido como padrão (*template*), para calcular diferentes aspectos que serão comparados. Para o estudo nesse projeto, que também foi utilizado no artigo [35], foi escolhida a função de distância DTW. Em [35], são definidas 5 métricas para avaliação das curvas geradas, (s_1, s_2, s_3, s_4, s_5), porém somente as métricas de índice $2, 3$ e 4 são relevantes para a pesquisa aqui exposta. Desse modo, definem-se 3 métricas (s_2, s_3, s_4), para a avaliação da rede treinada, que serão apresentadas a seguir.

2.3.1 Métricas de avaliação dos sinais gerados

A métrica denominada como s_2 determina uma média das distâncias entre todos os batimentos gerados e o *template* escolhido, sendo assim dada por

$$s_2 = \frac{1}{N_G} \sum_{i=1}^{N_G} \text{DTW}(\mathbf{v}_i, \mathbf{t}), \tag{1}$$

onde N_G indica o número de batimentos gerados pela rede, $\mathbf{v}_i$ os vetores dos batimentos gerados pela rede e $\mathbf{t}$ o vetor do *template* selecionado.

A métrica s_3 baseia-se na escolha do "melhor batimento", ou seja, ela indica qual a menor distância dentre os batimentos gerados em relação ao *template*. Desse modo,

$$\mathbf{v}_{\text{melhor}}^{\text{DTW}} = \text{argmin DTW}(\mathbf{v}_i, \mathbf{t}), \tag{2}$$

$$s_3 = \text{DTW}(\mathbf{v}_{\text{melhor}}^{\text{DTW}}, \mathbf{t}). \tag{3}$$

Por fim, a métrica s_4 é definida como uma combinação das métricas s_2 e s_3 [35]. Nesse caso, é definido um limiar para avaliar todas as curvas como "aceitáveis" ou não, ou seja,

$$\eta^{\text{DTW}} = \frac{s_2 + s_3}{2}. \tag{4}$$

Portanto todos os batimentos que, comparativamente com o *template*, apresentam uma distância menor do que esse limiar, são contabilizados para formar uma taxa de produtividade (*productivity rate*) s_4, dada por

$$s_4 = \frac{N_{\text{best}}^{\text{DTW}}}{N_G}, \tag{5}$$

onde $N_{\text{best}}^{\text{DTW}}$ indica o número de batimentos gerados abaixo do limiar η^{DTW}.

A curva de *template* selecionada para a comparação utilizando as métricas apresentadas anteriormente foi fornecida em [35]. Essa curva foi selecionada por um especialista como representativa da classe N e pode ser observada na Figura 5 a seguir.

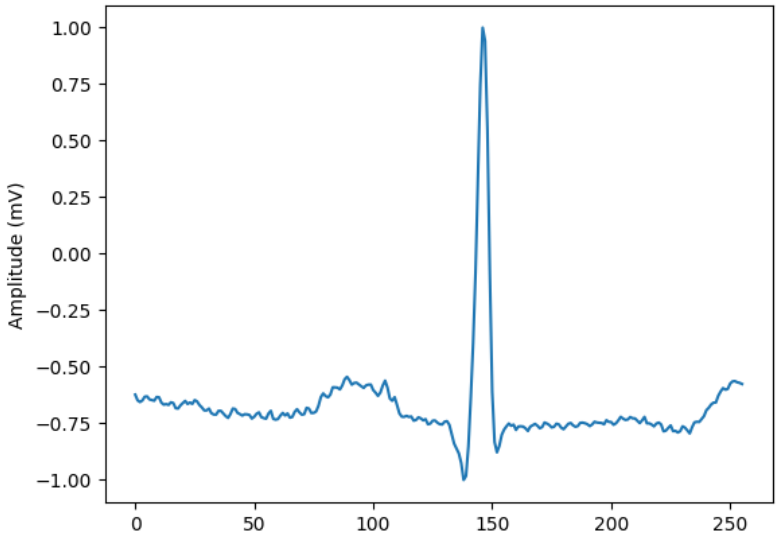


Figura 5: *Template* de batimento de ECG para classe N.

Como pode ser observado no gráfico apresentado acima, esse batimento trata de um vetor de tamanho $(256, 1)$, diferente do tamanho $(320, 1)$ gerado pela rede. Portanto, para realizar a comparação nas métricas utilizando o DTW, as curvas sintetizadas são reamostradas utilizando o método de Fourier com a função *resample* da biblioteca *scipy*. As especificações sobre esse método podem ser encontradas em [69].

A seguir são apresentadas as estruturas utilizadas na rede GAN, tanto para o gerador quanto para o discriminador, sendo definidos também os outros parâmetros de treinamento da rede.

2.4 Configurações da Rede

Para a geração dos sinais de ECG da classe N, foram abordadas as estruturas de rede MLP e RNN, essa última em especial com o uso do bloco BiLSTM. As estruturas de rede utilizadas são descritas nas Tabelas 1, 2 e 3.

Tabela 1: Configurações do Gerador e Discriminador na rede MLP-MLP.

Gerador	[100-100(ReLU)-64(ReLU)-32(ReLU)-32(ReLU)-64(ReLU)-128(ReLU)-320(Linear)]
Discriminador	[320-320(ReLU)-256(ReLU)-128(ReLU)-64(ReLU)-32(ReLU)-16(ReLU)-1(Sigmoidal)]

Tabela 2: Configurações do Gerador e Discriminador na rede BiLSTM-MLP.

Gerador	[100-BiLSTM(100, 50)-BiLSTM(100,50)-320(TanH)]
Discriminador	[320-320(ReLU)-256(ReLU)-128(ReLU)-64(ReLU)-32(ReLU)-16(ReLU)-1(Sigmoidal)]

Tabela 3: Configurações do Gerador e Discriminador na rede WGAN-BiLSTM-MLP.

Gerador	[100-BiLSTM(100, 50)-BiLSTM(100,50)-320(TanH)]
Discriminador	[320-320(ReLU)-256(ReLU)-128(ReLU)-64(ReLU)-32(ReLU)-16(ReLU)-1(Linear)]

Para todos os testes foi utilizado um passo de adaptação $\eta = 0{,}00005$ e um treinamento de 500 épocas com tamanho de minibatch $nb = 44$, além de inicialização de Xavier [44] para os pesos das camadas. Em ambos os casos foi utilizado um treinamento múltiplo do discriminador para cada treinamento do gerador, com razão de 5 para 1. Além disso, para as configurações apresentadas, foram realizados testes utlizando tanto a função de custo de entropia cruzada, quanto a função custo de Wasserstein.

2.5 Testes e Resultados

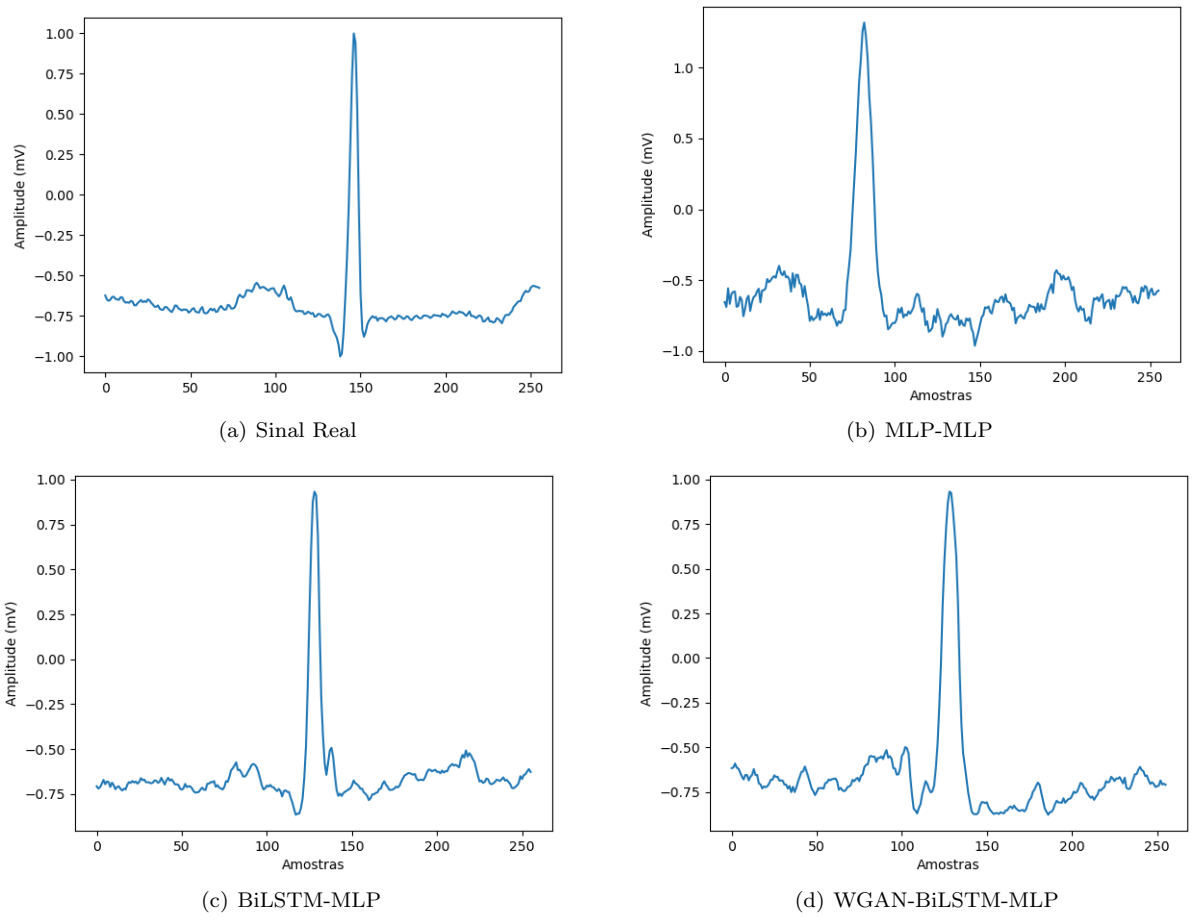
Utilizando as configurações de rede apresentadas na subseção anterior, e fazendo uso da função custo de entropia cruzada para as redes MLP-MLP e BiLSTM-MLP, foram realizados testes de geração de sinais com essas redes. Para tal, foram gerados 2200 vetores de espaço latente, que então foram submetidos às redes propostas, gerando os resultados apresentados na Tabela 4, considerando as métricas de avaliação propostas.

Tabela 4: Métricas obtidas das redes MLP-MLP,BiLSTM-MLP e WGAN-BiLTSM-MLP.

	MLP-MLP	BiLSTM-MLP	WGAN-BiLSTM-MLP
s_2	8,932	5,508	5,462
s_3	1,198	0,740	0,813
s_4	0,161	0,253	0,265

A partir dos resultados apresentados, observa-se que as redes BiLSTM-MLP e WGAN-BiLSTM-MLP, apresentam resultados superiores aos da rede MLP-MLP. Essas redes apresentam uma média menor de distância para o sinal de *template* (s_2) e fornecem um sinal mais coerente com o *template* (s_3) e uma maior taxa de produtividade (s_4).

São apresentados na Figura 6 os casos representantes das melhores curvas geradas por cada rede em relação à métrica s_3.



(a) Sinal Real

(b) MLP-MLP

(c) BiLSTM-MLP

(d) WGAN-BiLSTM-MLP

Figura 6: Sinais de ECG com menor DTW gerados por diferentes arquiteturas comparados com sinal real.

Nesse caso, como apresentado na Tabela 4, o sinal que possui menor similaridade com o sinal real é o da Figura 6 (b). Apesar de apresentar seu pico mais a esquerda do gráfico que esperado, a métrica do DTW é capaz de identificar essa translação do sinal, identificando-o como o melhor entre os sinais gerados por aquela rede. Ademais, utilizando o bloco BiLSTM, é possível perceber também um sinal menos ruidoso, o que pode ser explicado pela relação temporal entre os sinais passados e futuros proporcionada pelo bloco, gerando modificações menos abruptas entre os sinais próximos.

Analisando novamente a Tabela 4, é possível também observar que os resultados obtidos por ambas as redes BiLSTM, com e sem o uso da técnica de Wasserstein [58], são consideravelmente próximos, com a rede WGAN apresentando uma pequena vantagem quanto a taxa de produtividade (s_4).

Com os resultados obtidos com a rede GAN consolidados, partiu-se então para uma abordagem utilizando a rede VAE. Os resultados obtidos com esse tipo de arquitetura serão apresentados nas seções subsequentes.

3 Adaptação dos resultados parciais para a rede VAE

Apesar da classe N ser a classe mais representativa do banco de dados utilizado, e portanto ser a classe que não necessita do balanceamento de dados proposto, a classe N foi escolhida para ser a primeira a ser gerada nos experimentos utilizando a rede VAE. Essa decisão se deve justamente pela sobre-representatividade da classe, possibilitando portanto uma maior facilidade no processo de geração dos sinais sintéticos desse conjunto.

Passando pela geração dos sinais de classe N, foram gerados então os sinais de classe S. A classe S, apresenta alta similaridade a olho nu com a classe N e portanto, a princípio, utilizou-se uma arquitetura de rede neural semelhante para a geração dos sinais representativos das duas classes.

Com esses objetivos em mente, foi então utilizada a rede VAE para realizar a geração desses sinais sintéticos, normalizados no intervalo $[-1, 1]$. Após o treinamento da rede, passou-se então para um processo de geração dos sinais de forma contínua, respeitando-se a utilização apenas dos sinais gerados considerados

aceitáveis a partir de parâmetros estabelecidos dinamicamente. Em posse dos sinais sintéticos gerados, expandiu-se o banco de dados, a fim de retreinar a rede e averiguar a taxa de acerto em sinais de eletrocardiograma de teste reais.

A seguir será apresentado o processo definido para a geração dos sinais de classe N.

3.1 Geração de sinais da classe N com rede VAE

Para a geração dos sinais da classe N, foi-se utilizada uma rede VAE com arquitetura do tipo MLP tanto no Codificador (C) quanto no Decodificador (D). No caso do Codificador foi proposta uma estrutura de configuração [320-320(LeakyReLU)-512(LeakyReLU)-BN-10], enquanto para o Decodificador foi utilizada uma estrutura do tipo [10-512(LeakyReLU)-512(LeakyReLU)-BN-320(LeakyReLU)- 320], onde BN indica uma camada de normalização de batch [44]. Para as funções de *LeakyReLU* [44] utilizadas foi estabelecido um parâmetro de inclinação negativa $m = 0.2$. A Figura 7 apresenta um esquema da arquitetura proposta.

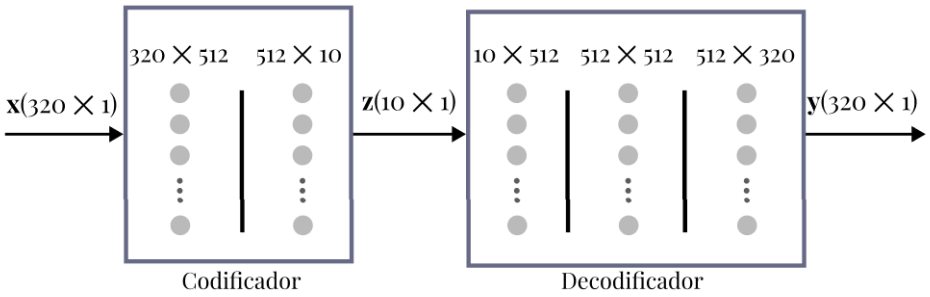


Figura 7: Esquema de arquitetura da rede VAE.

Para a transmissão do sinal entre as etapas do Codificador e Decodificador foi utilizado um processo de reparametrização [55]. A função custo escolhida foi uma combinação da função de divergência de Kullback-Leibler (KLB) [55] e do erro quadrático médio (MSE) [39], definida por

$$\Theta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\mu}, \boldsymbol{\varepsilon}) = \text{MSE}(\boldsymbol{x}, \boldsymbol{y}) - 0.7\text{KLB}(\boldsymbol{\mu}, \boldsymbol{\varepsilon}). \tag{6}$$

A rede foi treinada por 600 épocas, utilizando tamanho de minibatch $nb = 22$, passo de adaptação $lr = 10^{-4}$, além do otimizador Adam.

Para a delimitação do limiar de aceitação dos sinais gerados foi utilizado como limite o valor de distância DTW entre a amostra e o *template* da classe N inferior a $2s_3$. Os resultados obtidos, conforme os parâmetros estabelecidos, foram de $s_2 = 5{,}861$, $s_3 = 0{,}876$ e $s_4 = 0{,}082$. O sinal representativo do valor de DTW de s_3 pode ser observado na Figura 8.

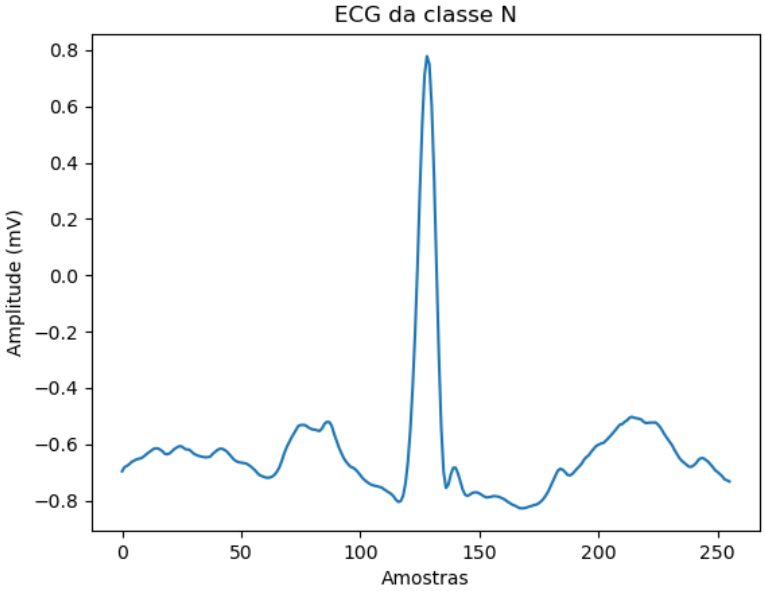


Figura 8: Sinal sintético da classe N gerado por VAE com menor distância DTW do *template* representativo.

Esses sinais, quando submetidos a rede MLP pré-treinada, atingiram taxa de acerto de 100% como pertencentes a classe N. Assim, a rede VAE se mostrou capaz de gerar sinais convincentes para a classe a qual se propôs. Entretanto o resultado de 100% apenas se mostrou possível devido a alta quantidade de dados referentes a classe N disponíveis para treinamento da rede. A seguir serão comparados os resultados obtidos para a geração dos sinais da classe N utilizando a arquitetura do tipo GAN e a arquitetura do tipo VAE.

3.2 Comparação dos sinais gerados pela rede GAN e pela rede VAE

Tendo em vista os resultados alcançados com os diferentes tipos de rede neural, GAN e VAE, podemos observar que, apesar de ambas as redes serem capazes de gerar os sinais desejados, existem diferenças, tanto na qualidade dos sinais sintéticos da classe N gerados, quando no custo computacional para realizar esses procedimentos. De modo geral, podemos comparar as métricas s_2, s_3 e s_4 de ambos os testes. Os resultados são comparados a seguir na Tabela 5.

Tabela 5: Comparação das métricas obtidas com rede GAN e VAE para geração de ECGs sintéticos (N).

	s_2	s_3	s_4
GAN	0,051	0,740	0,253
VAE	0,059	0,876	0,082

Comparando os resultados, podemos observar uma alta similaridade nos resultados obtidos, principalmente considerando as métricas s_2 e s_3.

A princípio, os resultados obtidos com a rede GAN são aparentemente superiores, pois são capazes de gerar sinais com maior índice de similaridade (distância DTW) e com uma maior taxa de aproveitamento dos sinais gerados, 25% em comparação à 8%. Entretanto, com o objetivo de complementar o banco de dados com sinais sintéticos de forma a balancear as classes com sinais representativos de cada classe, a adição de sinais com um grau de similaridade excessiva pode apresentar baixa utilidade, pois seria uma expansão de baixa eficácia. Assim, o nível ótimo para as métricas s_2 e s_3 não é determinado.

Além disso, apesar da rede GAN apresentar uma taxa de aproveitamento consideravelmente maior quando comparada com a rede VAE apresentada, o custo computacional da rede GAN é consideravelmente maior. Comparando ambos os treinamentos, obtemos um tempo de treinamento para a rede GAN cerca de cinco vezes maior que o tempo de treinamento obtido com a rede VAE. Assim, apesar de, num processo de geração de sinais sintéticos, uma maior parcela dos sinais ser aproveitado utilizando a rede GAN, o custo para utilizar a rede VAE

é consideravelmente reduzido, possibilitando uma maior adaptabilidade da rede. Desse modo, prosseguiu-se com o estudo para a geração dos sinais sintéticos utilizando a rede VAE.

3.3 Geração de sinais da classe S com rede VAE

A partir de uma análise prévia dos sinais presentes na classe S do banco de dados, encontrou-se elevada variedade nos formatos das curvas apresentadas. Desse modo, ao se utilizar uma única curva como *template*, a classe S estaria sub-representada pois, pelo método adotado de decisão dos sinais autênticos, a distância de DTW apresentaria maior distribuição de valores, mesmo para sinais que seriam, de outro modo, representativos da classe. Desse modo, foi decidida outra abordagem para a seleção do *template* da classe S.

3.3.1 *Template* da classe S

Primeiramente, para avaliar como os sinais da classe S se distribuem em termos da distância de DTW, selecionou-se aleatoriamente um sinal do banco de dados, pertencente a classe S. Esse sinal foi então medido em relação a todos os outros sinais representativos da classe no banco de dados. A distribuição dos valores de distância de DTW pode ser observado no histograma apresentado na Figura 9.

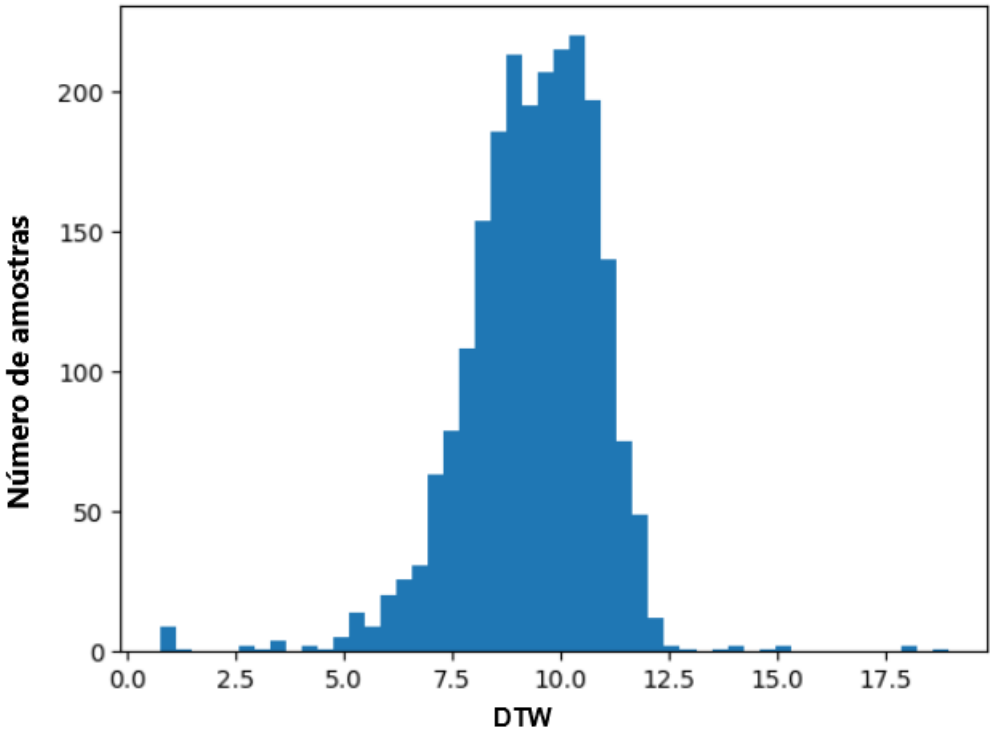


Figura 9: Histograma de distribuição dos valores de distância de DTW na classe S.

Como é possível observar pelo gráfico da Figura 9, os valores de DTW encontram-se, majoritariamente entre 7.5 e 10, ou seja, consideravelmente diferentes do sinal aleatório selecionado.

Desse modo, observando-se que para esse sinal os valores de DTW variam entre 1 e 19, selecionou-se seis valores de DTW, igualmente espaçados nesse intervalo, e os sinais representados por esse DTW foram selecionados como *templates*. Os seis *templates* selecionados podem ser observados na Figura 10 abaixo.

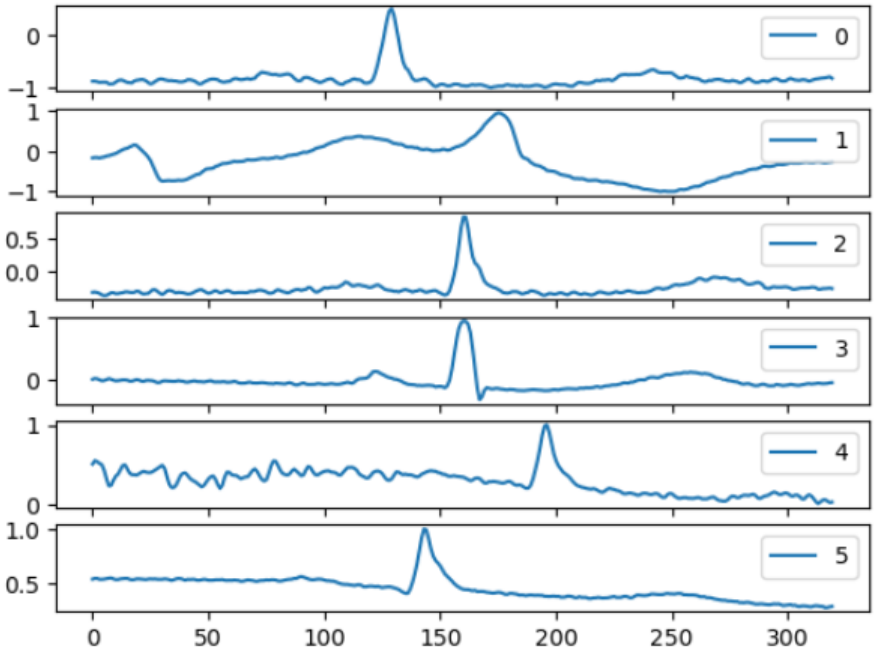


Figura 10: *Templates selecionados representativos da classe S.*

Ao se utilizar os seis modelos apresentados como *templates* para representar a classe S, ao se submeter cada sinal do banco de dados da classe S a comparação com os templates, considerando-se apenas o menor resultado de cada série de comparações, obteve-se um valor de distância de DTW mínima de 0,096 e máxima de 5,680. Portanto, os novos *templates* representam de forma consideravelmente mais completa a classe S. Ademais, caso fossem selecionados mais *templates* poderia se obter uma representação mais realçada da classe S, porém o custo computacional para realizar os cálculos de comparação durante a elaboração das métricas de geração dos sinais seria mais elevado, assim utilizaram-se apenas seis sinais como *templates* para representação da classe de maneira satisfatória. A seguir será apresentado o processo de geração dos sinais da classe S.

3.4 Arquitetura e Geração de sinais

Para a arquitetura da rede VAE utilizada na geração de sinais S, devido a similaridade da estrutura dos sinais S com os sinais N, optou-se por utilizar a mesma arquitetura apresentada para a classe N e a mesma função custo. Foram utilizados, para o treinamento da rede 800 épocas, tamanho de minibatch $nb = 25$, passo de adaptação $lr = 5 \times 10^{-5}$ e otimizador Adam.

Como os sinais gerados para a classe S apresentam maior variabilidade de maneira natural, optou-se por utilizar um limiar proporcionalmente maior ao utilizado na classe N, assim foram aceitos sinais que apresentassem distância de DTW inferior a $3s_3$ quando comparandos com os *templates* da classe S. Nesse caso, comparando-se o sinal com cada *template* de S, a distância de DTW é definida como o menor valor dentre as comparações.

Para esse modelo, os resultados obtidos para os parâmetros foram $s_2 = 5{,}758$, $s_3 = 0{,}759$ e $s_4 = 0{,}120$. O sinal representativo do valor de DTW de s_3, ou seja, seu apresenta a maior similaridade com um *template* do conjunto de *templates*, pode ser observado na Figura 11.

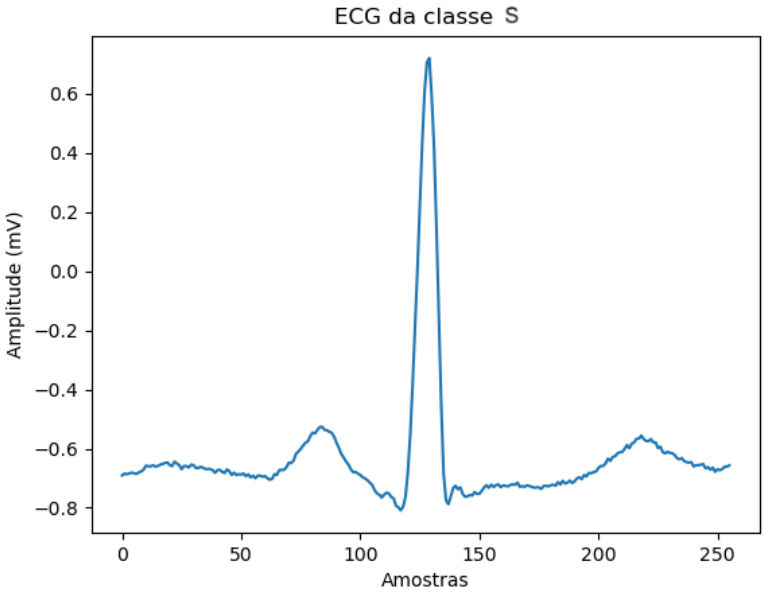
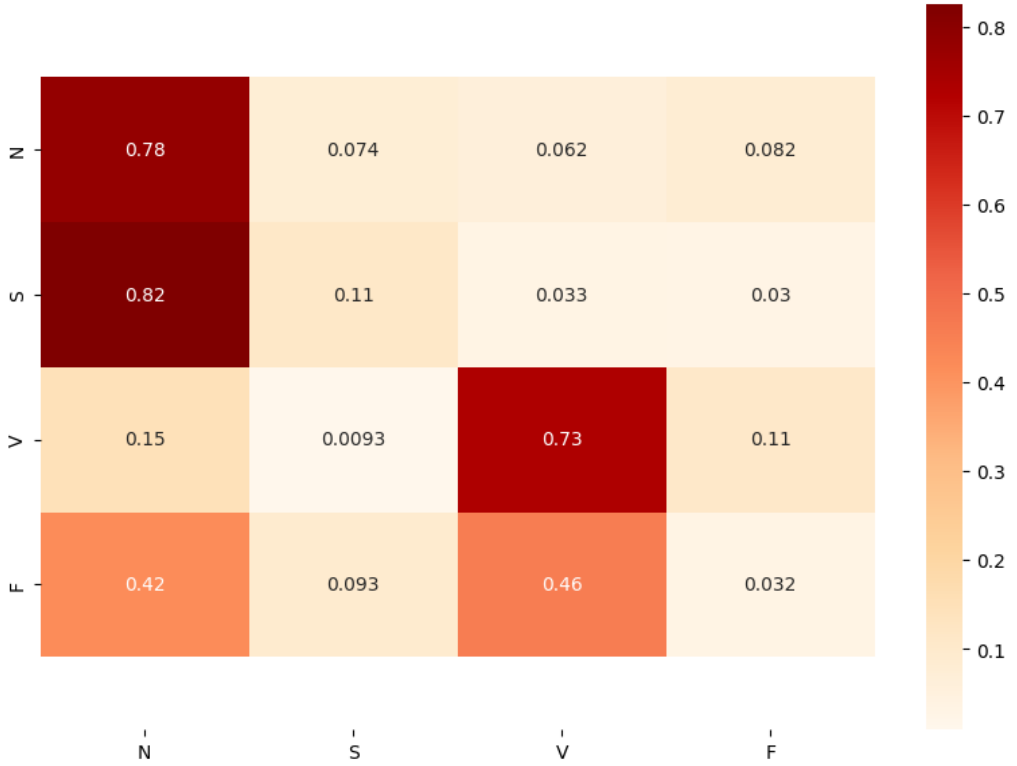
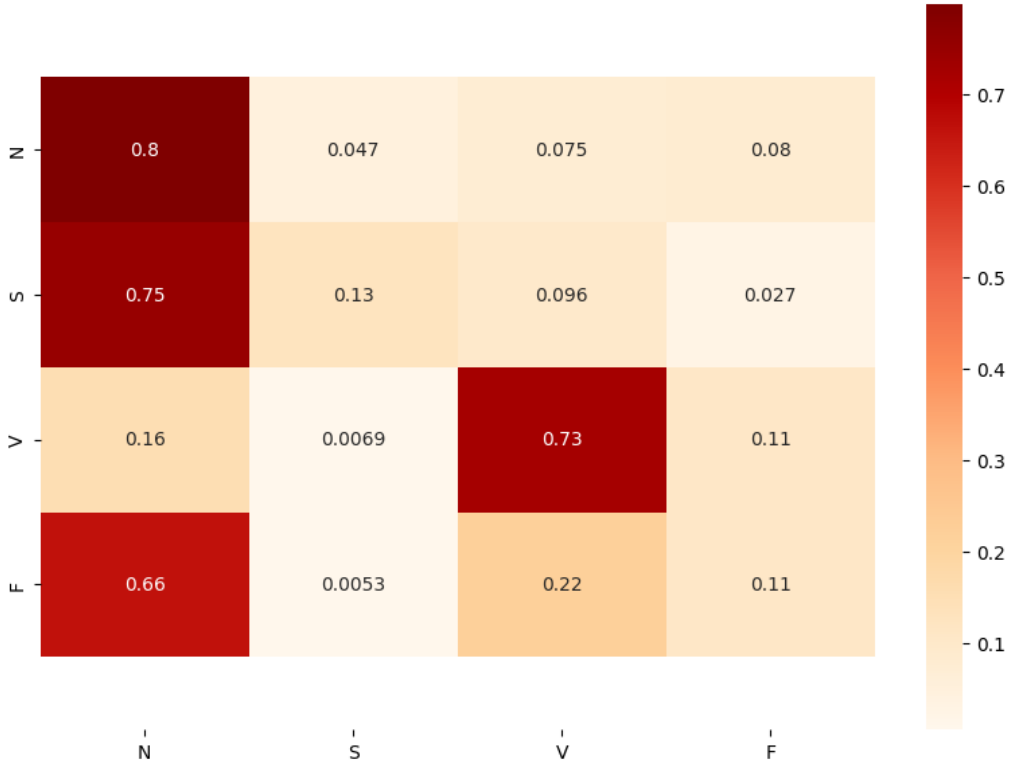


Figura 11: Sinal sintético da classe S gerado por VAE com menor distância DTW dos *templates* representativos.

Os sinais gerados, de tamanho 320, foram combinados com sinais N também com tamanho 320 para formar sequências NSN. Para realizar a ligação dos sinais NS e SN foi feita uma média da amplitude dos 10 últimos pontos de cada sinal. Essa amplitude média foi determinada como um "ponto central" da ligação e então foram substituidos os 5 últimos valores de cada sinal conectado, tendo em vista uma gradação entre esse ponto central médio e o ponto final de cada sinal, permitindo assim a conexão desses sinais de maneira não abrupta. Esse sinais foram então complementados ao banco de dados para treinamento da rede MLP classificadora [16]. Os resultados da rede antes e depois do processo de *data augmentation* podem ser observados nas matrizes de confusão apresentadas na Figura 12 a seguir.



(a) MLP



(b) MLP *augmented*

Figura 12: Matrizes de confusão das redes MLP com banco de dados sem e com *data augmentation*.

Como podemos observar pela comparação das matrizes, ao se utilizar o processo de *data augmentation*, os resultados obtidos para a classe S são superiores aos encontrados anteriormente, aumentando de 11% para 13%. Além disso, é possível observar que o enriquecimento do banco de dados referentes a classe S, diminui também o erro de classificação de outras classes com S. Mais notavelmente, podemos observar que, anteriormente ao

processo de *data augmentation*, havia um erro de classificação de sinais F como pertencentes a classe S de 9,3%, sendo então reduzido a 0,5%. Ademais, diminuiu-se o erro de classificação dos sinais S como pertencentes a classe N, com a qual compartilha diversas características. A seguir serão apresentadas as conclusões do projeto de pesquisa.

4 Conclusões

O projeto de pesquisa teve como princípio o melhoramento do banco de dados MITDB, pelo processo de *data augmentation*, a fim de melhorar o desempenho da rede neural classificadora MLP-LDA desenvolvida em [16]. Para alcançar esse objetivo, foi estudado o sinal de eletrocardiograma, levando em consideração os aspectos fundamentais que permitem sua classificação, assim como as especificidades da rede classificadora proposta anteriormente [16]. O projeto teve como guia o estudo realizado em artigos existentes na literatura [33–35], levando-se em consideração também as necessidades apresentadas pela rede classificadora.

A primeira estrutura proposta foi a estrutura da rede GAN. Diferentes arquiteturas foram colocadas sob teste, tanto para a rede do Gerador quanto para a rede do Discriminador. Com a rede GAN, foram propostas combinações utilizando a rede WGAN e também a estrutura recorrente para redes neurais, BiLSTM. Os resultados obtidos com esses tipos de rede, para a geração de sinais da classe N, foram apresentados e, para os modelos propostos, a rede GAN, combinada com a estrutura BiLSTM, apresentou o melhor desempenho.

Em seguida, foi explorada a geração de sinais da mesma classe N, utilizando porém a estrutura VAE. Os resultados da rede VAE e da rede GAN foram comparados tendo em vista as métricas s_2, s_3 e s_4 apresentadas [35]. Ao final da comparação, ambas as redes apresentaram resultados relevantes para a geração de sinais da classe N.Entretanto a rede VAE apresentou resultados melhores considerando a relação entre custo computacional e desempenho. Desse modo, prosseguiu-se com essa estrutura para a geração de sinais referentes a classe S.

Para a avaliação dos sinais gerados, pela rede, para a classe S foi realizado primeiramente um estudo do banco de dados dessa classe. Algumas métricas foram traçadas e um conjunto de seis sinais de eletrocardiograma desse banco de dados foi selecionado como conjunto de *templates*. Em posse desses sinais, foram gerados os sinais sintéticos para a classe S, utlizando a mesma arquitetura VAE utilizada para a geração de sinais da classe N. Os resultados obtidos, assim como as métricas resultantes, foram apresentados. Posteriormente, os dados gerados utilizando a rede VAE treinada foram adicionados ao bando de dados de treinamento da rede MLP, e foram comparados os resultados do teste, com e sem *data augmentation*, através de duas matrizes de confusão. Por fim, pode se observar que os resultados obtidos após o processo de *data augmentation*, realizado com sinais sintéticos da classe S, apresentou melhoras na classificação da rede, não só para sinais da própria classe S, mas também para sinais da classe F.

Tendo em vista o plano de pesquisa proposto inicialmente, foram alcançados os objetivos propostos referentes ao estudo do sinal de eletrocardiograma, o estudo de redes neurais e modelos gerativos, além da implementação da rede neural MLP proposta [16]. Foi estudado e testado o uso de redes GAN para a geração de sinais sintéticos, assim como o uso de redes VAE para o mesmo próposito. Os resultados obtidos com a rede VAE foram satisfatórios para a geração de sinais da classe desbalanceada S e os resultados da rede neural MLP treinada com data augmentation foram comparado com os resultados obtidos anteriormente ao balanceamento. Devido ao tempo extenso utilizado no processo de adaptação das redes para geração dos sinais sintéticos propostos, não foi possível realizar a geração de sinais sintéticos para as outras classes minoritárias V e F.

Portanto, os resultados obtidos com a geração dos sinais sintéticos, em complementação ao projeto de classificação de arritmias cardíacas, para balanceamento do banco de dados foi alcançado. Foram explorados os resultados com ambas as estrutura GAN e VAE e comparados os resultados obtidos antes e após o retreinamento da rede. Com base nos resultados alcançados durante a pesquisa foi elaborado e publicado um artigo no Simpósio Brasileiro de Telecomunicações e Processamento de sinais (SBrT - 2024), com o título Classificação de sinais de ECG sintéticos, apresentado no apêndice B do relatório.

A Fundamentação Teórica

Primeiramente, nesta seção, serão apresentados alguns conceitos gerais relacionados a redes neurais. As redes MLP e RNN são apresentadas como possíveis arquiteturas que serão utilizadas na construção da rede gerativa pretendida. Posteriormente, um exemplo de geração de sinais periódicos será apresentado utilizando a rede do tipo GAN, que será a rede em foco nesse projeto de pesquisa.

A.1 Rede Perceptron Multicamada

Nessa subseção, são apresentados os fundamentos da rede MLP, iniciando com a descrição de sua unidade fundamental, o "perceptron".

A.1.1 Perceptron de Rosenblatt

O modelo do neurônio para aplicação computacional, denominado "perceptron", foi originalmente proposto pelo psicólogo americano Frank Rosenblatt [36] numa tentativa de simplificar a implementação de um "processo de decisão", baseado no funcionamento de um neurônio biológico. Nesse neurônio artificial, o sinal recebido faz o papel do estímulo do neurônio biológico. Ponde ando-se esse sinal com pesos e somando-se o resultado a um *bias* obtém-se o sinal de entrada da função não linear, chamada de função de ativação. A saída dessa função faz o papel do estímulo transmitido pelo neurônio.

Matematicamente, considerando M pesos ω_k, $k = 1,2,...M$ e o *bias* b_1, define-se o vetor

$$\boldsymbol{\omega} = [b_1 \ \omega_1 \ \omega_2 \ \cdots \ \omega_M \ ]^T . \tag{7}$$

em que $(\cdot)^T$ representa a transposição. Definindo o vetor de entrada como

$$\mathbf{x} = [\, 1 \ x_1 \ x_2 \ \cdots \ x_M]^T , \tag{8}$$

e a função de ativação não linear por $\varphi(\cdot)$, obtém-se a saída do neurônio, dada por

$$y = \varphi(\mathbf{x}^T \boldsymbol{\omega}). \tag{9}$$

Um esquema ilustrativo desse processo pode ser observado na Figura 13. Cabe observar que como o *bias* b_1 foi incorporado ao vetor de pesos, a primeira posição do vetor de entrada deve ser igual a 1.

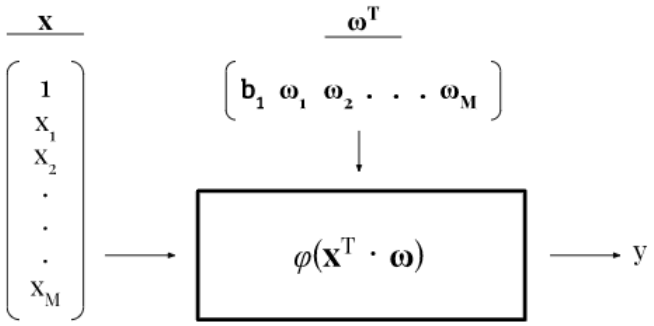


Figura 13: Esquema de funcionamento de um perceptron.

A função de ativação tem o objetivo de definir o domínio da saída do neurônio, restringindo essa saída e, idealmente, mantendo-a dentro do escopo do problema que se deseja abordar. No caso do perceptron de Rosenblatt, a função de ativação utilizada é a função sinal, o que faz com que a curva de separação seja linear. Entretanto, um único neurônio não é capaz de resolver problemas mais complexos e por isso foram organizados em camadas dando origem à rede perceptron multicamada [37], como será explicado posteriormente. As funções de ativação mais utilizadas em redes neurais são descritas a seguir.

A.1.2 Funções de Ativação

As funções de ativação são componentes essenciais em redes neurais artificiais. Essas funções fazem parte de cada neurônio da rede para introduzir não-linearidade nas saídas [43]. A função de ativação determina se um neurônio deve ser ativado ou não, com base em uma combinação linear das entradas que recebe.

Existem diversas funções de ativação utilizadas em redes neurais [44]. Algumas das mais comuns são: *Rectified Linear Unit*, conhecida como ReLU, a função Sigmoidal e a Tangente Hiperbólica (TanH), além da função Linear simples. Suas equações são dadas, respectivamente, por:

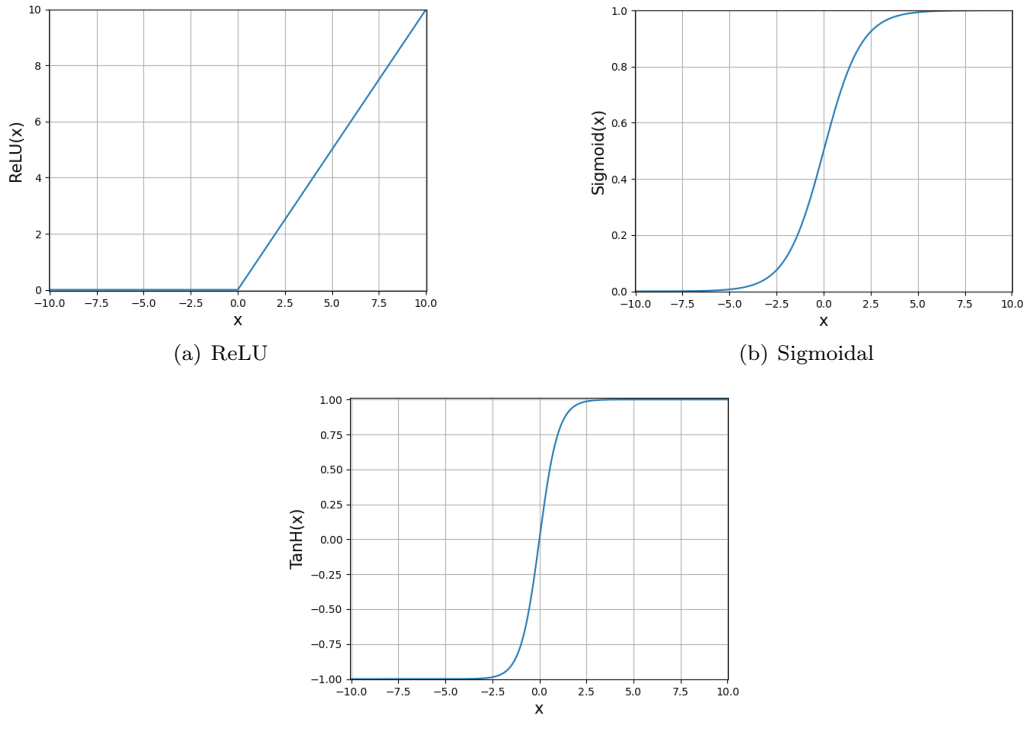
$$\text{ReLU}(x) = max(0,x), \tag{10}$$

$$\text{Sigmoidal}(x) = \frac{1}{1+e^{-x}}, \tag{11}$$

$$\text{TanH}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{12}$$

Os gráficos dessas funções de ativação podem ser observados na Figura 14.

Diferente da função sinal, utilizada no perceptron de Rosenblatt, as funções sigmoidal e TanH possuem derivada em todos os pontos. Por isso, ambas são utilizadas em redes neurais com frequência, uma vez que o algoritmo de atualização dos pesos depende da derivada dessa função. Já a função ReLU não possui derivada definida em $x = 0$. No entanto, ela tem sido frequentemente utilizada já que ajuda o algoritmo de atualização dos pesos a evitar mínimos locais. Cabe observar que existem outras funções de ativação, como a softmax, utilizada em problemas de classificação multiclasse e diferentes variantes da ReLU. Vamos abordar essas funções apenas se forem utilizadas neste trabalho.



(a) ReLU

(b) Sigmoidal

(c) TanH

Figura 14: Gráficos de funções de ativação.

A.1.3 Rede Perceptron Multicamada

Para problemas mais complexos, o uso de um único neurônio não permite atingir bons resultados. Nesses problemas, se faz necessária a criação de uma rede que associa uma gama de neurônios artificiais em série e em paralelo, incorporando combinações mais complexas entre saídas dos neurônios e pesos associados. Essa rede é

chamada de Perceptron Multicamada (*Multilayer Perceptron* – MLP).

Numa rede MLP, os neurônios da camada j são associados aos neurônios das camadas $j-1$ e $j+1$, sem associação dos neurônios da mesma camada. Essas camadas podem ser divididas em 3 tipos: camada de entrada, camadas ocultas e camada de saída. Destas, apenas as camadas de entrada e saída podem ser acessadas, como ilustrado na Figura 15. Para denotar as configurações de camadas e números de neurônios, em uma rede MLP, será utilizada a notação $[N_0 - N_1(\phi_1) - N_2(\phi_2) - \cdots - N_j(\phi_j) - N_L(\phi_L)]$, onde N_0 representa o número de entradas, N_j o número de neurônios na camada j e N_L o número de neurônios na camada de saída, além de (ϕ_i) representar a função de ativação referente a cada camada i. Na Figura 15 é exemplificada uma rede MLP com configuração de camadas [2-3(ReLU)-3(ReLU)-1(Sigmoidal)]. Por simplicidade, assume-se que a não identificação da função de ativação, ϕ_i, para cada camada específica, indica que na camada é utilizada apenas uma saída linear simples.

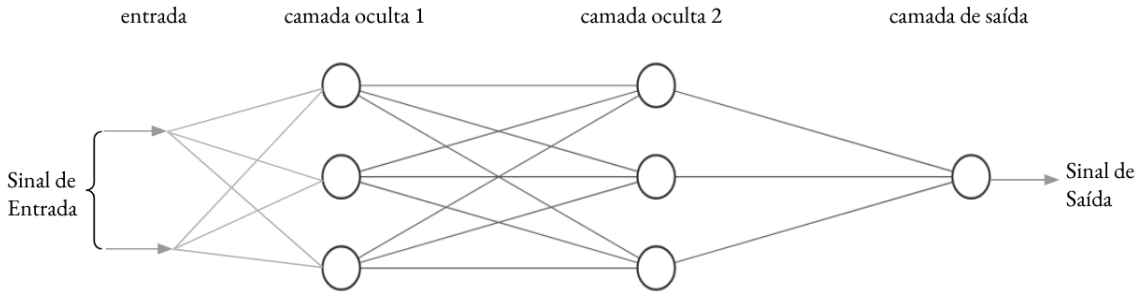


Figura 15: Ilustração de uma rede MLP com configuração [2-3(ReLU)-3(ReLU)-1(Sigmoidal)].

Para realizar uma determinada tarefa, os pesos dos neurônios da rede precisam ser adaptados. Nessa adaptação, utiliza-se uma parte dos dados disponíveis. Essa etapa é chamada de treinamento. Uma outra parte dos dados precisa ser reservada para verificar se o treinamento foi adequado e se a rede consegue efetuar a tarefa com dados diferentes dos usados no treinamento. Nessa etapa, chamada de teste, os pesos não são mais atualizados. O teste proporciona as métricas de desempenho da rede e indica se a rede precisa de ajustes. A seguir, aborda-se o algoritmo de ajuste dos pesos no treinamento.

A.1.4 Algoritmo *Backpropagation*

Uma ferramenta essencial no treinamento de redes neurais é o uso de épocas de treinamento [38]. Considere, por exemplo, que estejam disponíveis nt dados de treinamento. Muitas vezes a convergência do algoritmo de atualização dos pesos não é atingida com a utilização desses nt dados uma única vez. Além disso, há aplicações em que é muito difícil gerar mais dados para o treinamento. Diante disso, os nt dados disponíveis são utilizados mais de uma vez no treinamento. Chama-se *época* cada vez que esses nt dados são utilizados. Para inserir diversidade, esses dados devem ser embaralhados a cada época para que a sequência dos dados não influencie no treinamento. A cada vez que o algoritmo de aprendizado utiliza todos os nt dados de entrada, uma época é contada, e a próxima época se inicia, continuamente pelo número ne de épocas definido *a priori*.

Para realizar o cálculo do algoritmo, convém definir os vetores $\boldsymbol{v}^{(j)}$ e $\boldsymbol{y}^{(j)}$, que representam, respectivamente, o vetor de saída dos N_j neurônios da camada j antes da intervenção da função de ativação e o vetor de saída dos N_j neurônios da camada j após a intervenção da função de ativação, com $j = 1,2,\ldots,L$, onde L indica o número de camadas da rede. Desse modo, os vetores $\boldsymbol{v}^{(j)}$ e $\boldsymbol{y}^{(j)}$ são da forma

$$\boldsymbol{v}^{(j)} = \begin{bmatrix} v_1^{(j)} \\ v_2^{(j)} \\ \vdots \\ v_{N_j}^{(j)} \end{bmatrix}, \boldsymbol{y}^{(j)} = \begin{bmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \vdots \\ y_{N_j}^{(j)} \end{bmatrix}.$$

Além dos vetores já descritos, é preciso definir a matriz de pesos $\boldsymbol{W}^{(j)}$. Essa matriz engloba não só os pesos sinápticos, mas também os *biases* associados, combinando os vetores de pesos definidos em (7) de cada

neurônio da camada j, ou seja,

$$\boldsymbol{W}^{(j)} = \begin{bmatrix} \boldsymbol{\omega}_1^{(j)} \\ \boldsymbol{\omega}_2^{(j)} \\ \vdots \\ \boldsymbol{\omega}_{N_j}^{(j)} \end{bmatrix}.$$

Por fim, para associar os vetores de saída $\boldsymbol{y}^{(j)}$ da camada j, como vetores de entrada para a camada $j+1$, um vetor intermediário é criado. Esse vetor $\boldsymbol{x}^{(j)}$ é necessário pois ele insere o valor "1" na primeira posição do vetor, para permitir a utilização do *bias* de acordo com a formulação utilizada. Esse vetor é então definido por

$$\boldsymbol{x}^{(j)} = \begin{bmatrix} 1 \\ \boldsymbol{y}^{(j-1)} \end{bmatrix},$$

com o vetor de entrada da rede dado como

$$\boldsymbol{x}^{(0)} = \begin{bmatrix} 1 \\ x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{N_0}^{(0)} \end{bmatrix}.$$

O algoritmo *Backpropagation* consiste em duas etapas na fase de treinamento. Primeiramente é realizado o chamado *feedforward*, onde o sinal de entrada é fornecido à rede e transmitido pelas camadas intermediárias até atingir a camada de saída, para gerar o sinal de saída. O sinal de saída de cada camada j é passado para a camada $j+1$ como sinal de entrada, sendo então submetido ao processo descrito pela Figura 13, em cada neurônio, na camada seguinte.

Para as saídas de cada camada, a fórmula

$$\boldsymbol{v}^{(j)}(n) = \boldsymbol{W}^{(j)}(n-1)\boldsymbol{x}^{(j)}(n), \tag{13}$$

explicita o vetor de saída $\boldsymbol{v}^{(j)}$, da camada j na iteração n do algoritmo, sendo a mesma notação utilizada para a matriz de pesos $\boldsymbol{W}^{(j)}$ de cada camada e para os vetores de saídas $\boldsymbol{y}^{(j)}$, com cada vetor $\boldsymbol{y}^{(j)}(n)$ dado por

$$\boldsymbol{y}^{(j)}(n) = \varphi(\boldsymbol{v}^{(j)}(n)). \tag{14}$$

A Figura 16 ilustra o processo descrito.

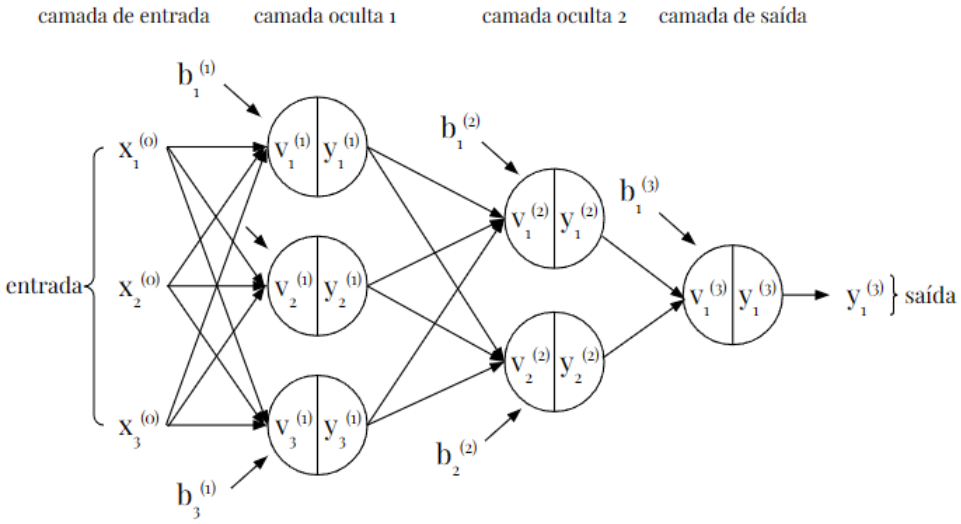


Figura 16: Representação do processo de *feedforward* de uma rede MLP com configuração [3-3(ReLU)-2(ReLU)-1(ReLU)].

Em posse agora do sinal de saída, a rede realiza a atualização dos pesos sinápticos e dos *biases* dos neurônios de cada camada, contidos nas matrizes $\boldsymbol{W}^{(j)}$, por meio da minimização de uma função custo. Um exemplo de função custo é a entropia cruzada binária (*Binary Cross-Entropy* – BCE) [41]. Nessa função, calcula-se a dissimilaridade entre duas distribuições, dada pela equação

$$\Theta(\mathbf{y},\mathbf{p}) = -\frac{1}{N_L}\sum_{i=1}^{N_L}(y_i \log(p_i) + (1-y_i)\log(1-p_i)). \tag{15}$$

onde N_L indica o número de neurônios na última camada. Além disso, na equação acima, y_i representa o sinal esperado em cada neurônio i da camada de saída e p_i representa a saída obtida em cada neurônio i da camada de saída da rede.

Para realizar a atualização dos pesos, um processo de retropropagação é implementado. Nesse processo, primeiramente calcula-se o gradiente local na camada de saída, seguido pelo gradiente das camadas intermediárias. Então, calcula-se a derivada da função custo, em relação aos pesos da camada por

$$\nabla\boldsymbol{\Theta}^{(j)}(n) = \frac{\partial\Theta(n)}{\partial\boldsymbol{W}^{(j)}(n-1)}. \tag{16}$$

Por fim, é realizado o ajuste dos vetores de pesos de cada camada, levando em conta o gradiente calculado, pela equação

$$\boldsymbol{W}^{(j)}(n) = \boldsymbol{W}^{(j)}(n-1) - \eta\nabla\boldsymbol{\Theta}^{(j)}(n), \tag{17}$$

onde η representa o passo de adaptação.

O algoritmo se repete pelo número de iterações pré-determinado. Além de fazer uso da divisão em épocas, é vantajoso também determinar o padrão de vezes em que ocorre o ajuste de pesos. Em uma primeira observação, é possível suspeitar que o caso estocástico, em que se atualizam os pesos a cada dado de treinamento, seja ideal. Entretanto, raramente essa hipótese se verifica. Os métodos mais comuns utilizam os chamados *batch* ou *mini-batch*. Nesses casos, o vetor gradiente é estimado por um certo intervalo, e então é atualizado usando essa estimativa, ao invés de usar cada dado de treino. No caso do *batch* o número de dados utilizados na estimativa é igual ao número de dados do conjunto de treino, ou seja, há apenas uma iteração de atualização dos pesos por época. Enquanto no caso do *mini-batch* um valor menor nb é determinado, tendo então $\frac{N_t}{nb}$ iterações por época. Então, a eficiência da rede é verificada na fase de teste, determinando se os parâmetros escolhidos são adequados ou não, e se a rede é capaz de solucionar o problema de maneira satisfatória.

A arquitetura da rede MLP apresenta diversas vantagens e flexibilidade para a resolução de diversos problemas. Contudo, para o processamento de sinais temporais, esse tipo de arquitetura pode ser pouco eficiente, sendo necessária então uma arquitetura mais robusta. Com essa questão em mente, descreve-se a seguir uma rede neural recorrente específica.

A.2 Redes Neurais Recorrentes

Diferentemente da rede MLP, onde a saída, é dada por

$$\mathbf{y} = \varphi(\mathbf{x}^T\boldsymbol{\omega}), \tag{18}$$

na rede RNN é utilizado um vetor de estados que introduz memória no sistema. Dessa forma, o cálculo da saída do bloco passa a ter dependência dos vetores de entrada passados, além do vetor de entrada do instante atual. Considerando um vetor de entrada $\mathbf{x}$, o bloco A da Figura 17 utiliza a saída passada através do bloco de atraso representado por z^{-1}.

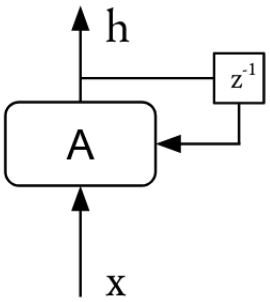


Figura 17: Representação de bloco fundamental da rede RNN.

Para avaliar o funcionamento desse tipo de estrutura, é realizado um processo denominado "desdobramento"(*unfolding*), de modo a permitir a interpretação do funcionamento desse bloco considerando as entradas temporais fornecidas. Desse modo, para o vetor de entrada $\mathbf{x}^{(t)}$, referente ao passo de tempo t, é aplicado um vetor de pesos $\boldsymbol{\omega}_x$. A saída do passo de tempo $t-1$, $\mathbf{h}^{(t-1)}$ é ponderado pelo vetor de pesos $\boldsymbol{\omega}_h$ para realizar a conexão temporal no bloco seguinte levando a

$$\mathbf{h}^{(t)} = \begin{cases} 0 & t = 0. \\ \varphi(\mathbf{x}^{(t)}\boldsymbol{\omega}_x + \mathbf{h}^{(t-1)}\boldsymbol{\omega}_h) & t \neq 0. \end{cases} \tag{19}$$

Um esquema do funcionamento desse tipo de rede, utilizando duas camadas ocultas, após o *unfolding* pode ser observado na Figura 18.

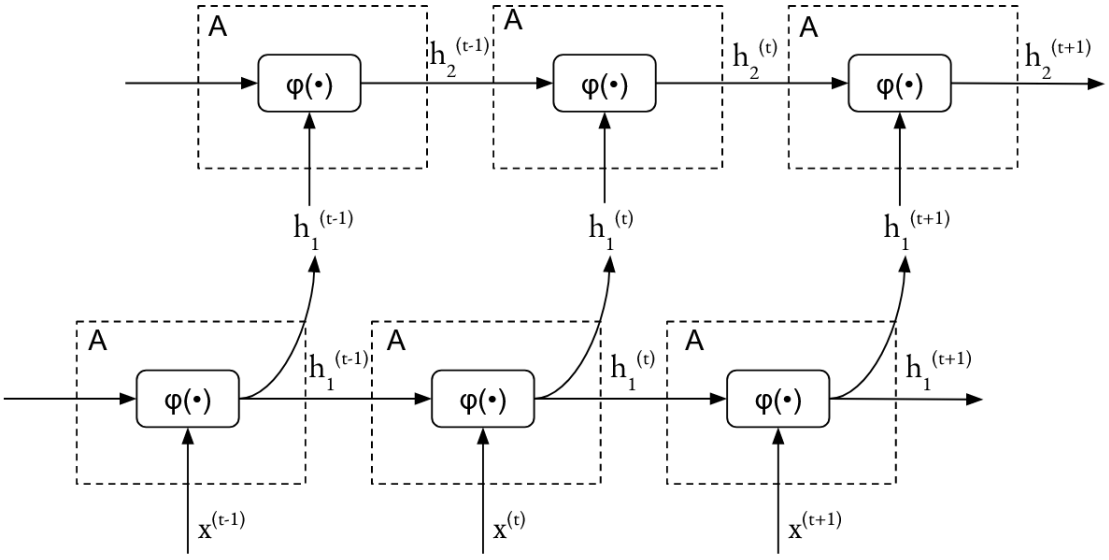


Figura 18: Esquema de funcionamento de rede RNN após o *unfolding*.

Analisando o esquema apresentado na Figura 18, assim como a Equação (19), observa-se que, no caso da rede RNN, para cada saída temporal, referente ao passo de tempo t, utiliza-se a saída anterior, referente ao passo de tempo $t-1$. Assim, a rede utiliza as entradas, necessariamente, em uma maneira ordenada.

Tendo esse funcionamento em vista, para realizar o treinamento das redes RNN, são comumente utilizados dois diferentes algoritmos para o treinamento desse tipo de rede, o algoritmo BPTT (*Backpropagation Through Time*) ou o algoritmo RTRL (*Real Time Recurrent Learning*). O algoritmo que será utilizado no projeto, e apresentado a seguir, é o algoritmo BPTT. Esse algoritmo funciona como uma extensão do algoritmo de retropropagação [44], alinhando o tempo como uma série ordenada para realização dos cálculos de erro e gradiente.

A.2.1 O algoritmo BPTT

Para apresentar o funcionamento do algoritmo de aprendizado em uma rede RNN [44], toma-se o modelo

apresentado na Figura 19 como o esquema de uma rede RNN genérica.

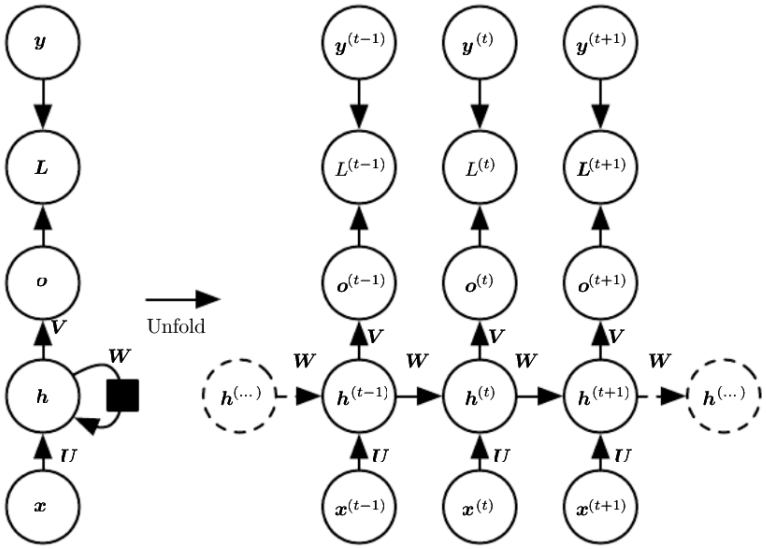


Figura 19: Grafo de rede RNN antes e após *unfolding*.
Fonte: [44]

Assumindo a função de ativação como tangente hiperbólica e saídas discretas, aplicando a função softmax na camada de saída, podemos então descrever o processo de *feedforward*, com o estado inicial $\mathbf{h}^{(0)}$ e considerando $t = 1, \cdots, \tau$, como:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}, \tag{20}$$

$$\mathbf{h}^{(t)} = \mathrm{TanH}(\mathbf{a}^{(t)}), \tag{21}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}, \tag{22}$$

$$\hat{\mathbf{y}}^{(t)} = \phi(\mathbf{o}^{(t)}), \tag{23}$$

onde ϕ representa uma função de ativação não especificada, $\mathbf{b}$ e $\mathbf{c}$ representam vetores de *bias* e $\mathbf{U}$, $\mathbf{V}$ e $\mathbf{W}$ representam matrizes de peso entre, entrada e camada oculta, camada oculta e saída e entre camadas ocultas, respectivamente.

Além disso, nessa rede, L representa a função custo, responsável por realizar a comparação entre as saídas obtidas pela rede e o sinal real desejado, para posteriormente auxiliar no cálculo dos gradientes. Logo, para uma sequência $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}\}$ de entradas e $\{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(\tau)}\}$ de saídas, com um mesmo comprimento τ, a função custo pode ser calculada pelo somatório dado:

$$L(\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}\}, \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(\tau)}\}) = \sum_t L^{(t)}. \tag{24}$$

Nesse modelo de rede, para calcular o gradiente, é preciso que seja realizada primeiramente a propagação da entrada, da esquerda para a direita de acordo com a ilustração após o *unfolding* da Figura 19, seguida pela retropropagação, da direita para a esquerda no mesmo grafo. Desse modo, a recursão é iniciada nos nós imediatamente anteriores ao custo final, e o gradiente $\nabla_{\mathbf{o}^{(t)}} L$ de cada saída $\mathbf{o}^{(t)}$, referente ao passo de tempo t é então calculado para cada neurônio i da camada oculta por

$$(\nabla_{\mathbf{o}^{(t)}} L)_i = \frac{\partial L}{\partial \mathbf{o}_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial \mathbf{o}_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}, \tag{25}$$

onde para o passo de tempo final $t = \tau$, o gradiente de $\mathbf{h}^{(\tau)}$ é dado por

$$\nabla_{\mathbf{h}^{(\tau)}} L = \mathbf{V}^\top \nabla_{\mathbf{o}^{(\tau)}} L. \tag{26}$$

Assim, é possível então iterar de $t = \tau - 1$ a $t = 1$, para realizar a retropropagação dos gradientes no tempo. Levando-se em consideração os valores de $\mathbf{o}^{(t)}$ e $\mathbf{h}^{(t+1)}$, o gradiente de $\mathbf{h}^{(t)}$ pode ser calculado da forma

$$\nabla_{\mathbf{h}^{(t)}} L = \left(\frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}}\right)^{\top} (\nabla_{\mathbf{h}^{(t+1)}} L) + \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}}\right)^{\top} (\nabla_{\mathbf{o}^{(t)}} L), \tag{27}$$

em que $\left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}}\right)^{\top}$ é $\mathbf{V}^{\top}$. Desse modo, os gradientes dos parâmetros $\mathbf{b}$, $\mathbf{c}$, $\mathbf{V}$, $\mathbf{W}$ e $\mathbf{U}$ podem ser obtidos de maneira análoga e podem ser encontrados em [44].

Assim, com os princípios de funcionamento da rede RNN estabelecidos, a seguir será abordado a construção e uso do bloco LSTM [42] em especial.

A.2.2 Bloco LSTM

O bloco LSTM foi introduzido, em princípio, com o objetivo de solucionar o problema de "desvanecimento de gradiente"(*vanishing gradient*) pela mudança no controle dos pesos inerentes ao bloco recorrente das redes RNN, fazendo com que esses pesos sejam controlados por outra unidade oculta da rede. Além disso, o bloco LSTM pode modificar sua escala de tempo com base nas entradas fornecidas uma vez que as constantes de tempo do sistemas são obtidas nas saídas do próprio modelo [44], provando-se útil para diversas aplicações [45, 46].

Similarmente à estrutura RNN apresentada na Figura 18, o LSTM é construído como um bloco fundamental que recebe uma entrada $\mathbf{x}^{(t)}$, para o passo de tempo t, e fornece uma saída $\mathbf{h}^{(t)}$ na camada oculta. Porém, dentro da célula do bloco LSTM são realizadas algumas operações adicionais para obter as interações desejadas. A estrutura do bloco pode ser observada, com *unfolding*, na Figura 20 abaixo.

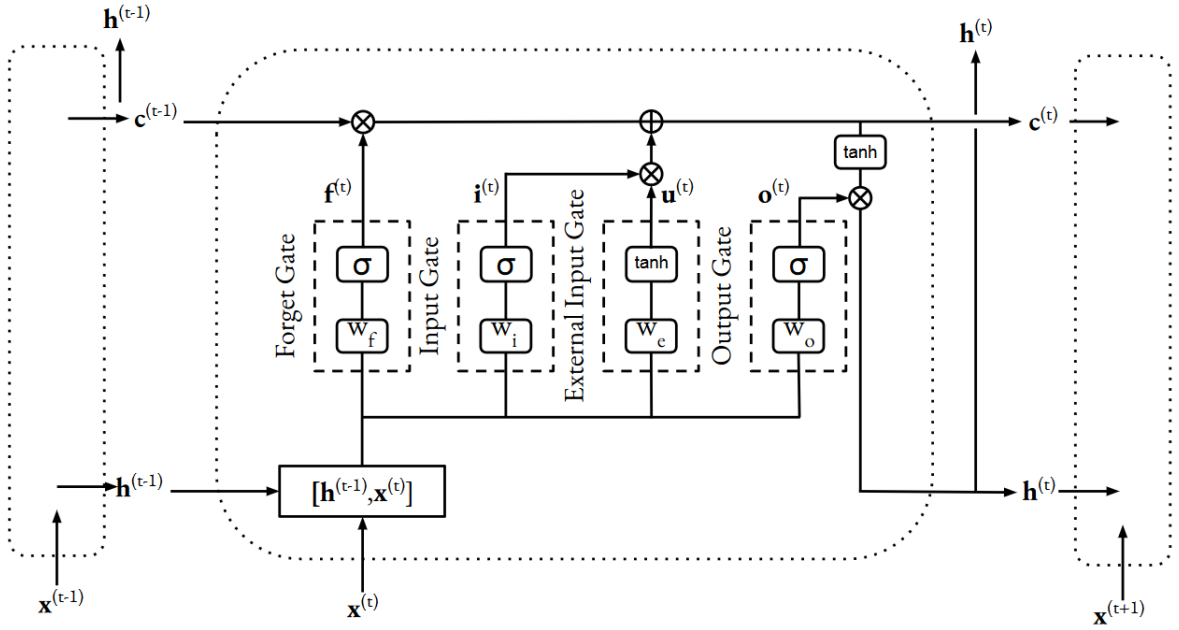


Figura 20: Esquema de funcionamento do bloco LSTM após *unfolding*.

No diagrama, o bloco σ representa a função sigmoidal, enquanto o símbolo $\bigotimes$ representa a multiplicação elemento a elemento e o símbolo $\bigoplus$ a soma elemento a elemento.

Nesse bloco, além do uso da saída $\mathbf{h}^{(t-1)}$ para obter $\mathbf{h}^{(t)}$, é também utilizada uma saída auxiliar representada por $\mathbf{c}^{(t-1)}$. Essa saída é controlada dinâmicamente pelo *Forget Gate*, representado na porta $f_i^{(t)}$, com i representando o número do neurônio da camada oculta. A saída dessa porta determina um valor de peso entre 0 e 1 por

$$f_i^{(t)} = \sigma(\mathbf{b}_i^f + \sum_j \mathbf{U}_{i,j}^f \mathbf{x}_j^{(t)} + \sum_j \mathbf{W}_{i,j}^f \mathbf{h}_j^{(t-1)}), \tag{28}$$

com $\mathbf{b}^f$, $\mathbf{U}^f$, $\mathbf{W}^f$ como biases, pesos e pesos para os *forget gates*, respectivamente.

O *External Input Gate*, $g_i^{(t)}$, é computado de maneira similar ao *Forget Gate*, utilizando diferentes parâmetros, por

$$g_i^{(t)} = \sigma(\mathbf{b}_i^g + \sum_j \mathbf{U}_{i,j}^g \mathbf{x}_j^{(t)} + \sum_j \mathbf{W}_{i,j}^g \mathbf{h}_j^{(t-1)}). \tag{29}$$

Enquanto a saída auxiliar de memória $\mathbf{c}_i^{(t)}$ é calculada por

$$c_i^{(t)} = f_i^{(t)} c_i^{(t-1)} + g_i^{(t)} \sigma(\mathbf{b}_i + \sum_j \mathbf{U}_{i,j} \mathbf{x}_j^{(t)} + \sum_j \mathbf{W}_{i,j} \mathbf{h}_j^{(t-1)}) \tag{30}$$

$$= f_i^{(t)} c_i^{(t-1)} + g_i^{(t)} u_i^{(t)}. \tag{31}$$

Para calcular a saída oculta, $\mathbf{h}_i^{(t)}$, do bloco, é necessário ainda calcular a saída do *Output Gate*, similarmente aos outros *gates*, dado por

$$o_i^{(t)} = \sigma(\mathbf{b}_i^o + \sum_j \mathbf{U}_{i,j}^o \mathbf{x}_j^{(t)} + \sum_j \mathbf{W}_{i,j}^o \mathbf{h}_j^{(t-1)}). \tag{32}$$

Assim, a saída auxiliar é submetida a função tangente hiperbólica e multiplicada pela saída do *Output Gate*, gerando a saída oculta como

$$\mathbf{h}_i^{(t)} = \tanh(c_i^{(t)}) p_i^{(t)}. \tag{33}$$

Ainda com o objetivo de melhorar a performance da rede composta por blocos LSTM, implementa-se também a chamada lstm-bidirecional (*bidirectional-lstm* – BiLSTM) [47].

Nesse tipo de bloco, o funcionamento de cada bloco individual mantém-se igual ao de um bloco LSTM comum, descrito anteriormente, porém ao invés de obter-se a saída da camada extraindo informações apenas do passo de tempo atual e de passos de tempo anteriores, utiliza-se também a informação referente aos passos de tempo futuros [47, 48]. A Figura 21 representa a estrutura de uma rede com uma camada de blocos BiLSTM.

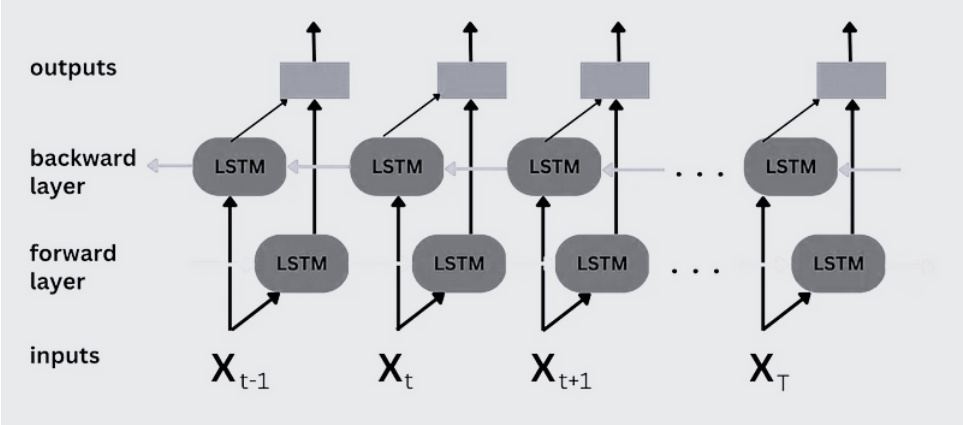


Figura 21: Esquema de funcionamento de camada com bloco BiLSTM.
Fonte: [49]

Nessa arquitetura, as informações de entrada do estado "atual" e de estados "passados" são utilizadas para gerar uma primeira saída no chamado *forward layer*. Posteriormente, as saídas obtidas em cada bloco LSTM da primeira camada são combinadas com as entradas utilizadas na primeira camada, porém referentes ao passo de tempo anterior ao das saídas utilizadas. Desse modo, nessa segunda camada, denominada de *backward layer*, as informações do "futuro" são combinadas com as informações do "presente", gerando a saída final das camadas compostas de blocos LSTM.

Para a representação adotada nesse relatório, será utilizada a nomenclatura LSTM(a, b) para o bloco LSTM original, ou BiLSTM(a, b) para o bloco BiLSTM, sendo "a" representativo da dimensão do vetor de entrada e "b" do número de neurônios na camada do bloco LSTM ou BiLSTM.

Desse modo, estabelecidos os conceitos para as arquiteturas de redes que serão utilizadas, a seguir será apresentada a estrutura da rede de tipo GAN.

A.3 Rede Adversária Gerativa

Redes Adversárias Gerativas (*Generative Adversarial Network* – GAN) [50], são tipos de redes aplicadas em diversas áreas na atualidade [51–53]. Essas redes são constituídas por dois modelos treinados simultaneamente. O primeiro modelo, identificado por G é o gerador, responsável por aprender a distribuição dos dados fornecidos. O outro modelo, identificado por D, se trata do discriminador, o qual é responsável por classificar a entrada fornecida como real ou gerada por G [50]. A seguir, será apresentado o procedimento de treinamento dessa rede e posteriormente técnicas comumente utilizadas para se obter um bom treinamento.

A.3.1 Treinamento Adversarial

O treinamento da GAN é definido por um jogo de soma-zero [54], no qual o ganho de um dos modelos implica na perda do outro. Nesse caso, a fim de gerar dados mais convincentes, o objetivo do treinamento do gerador é maximixar a probabilidade de erro do discriminador. Para treinar esses modelos em conjunto, um procedimento é proposto em [50, 55].

O procedimento de treinamento consiste em, primeiramente, fornecer à rede um vetor $\mathbf{z}$ pertencente a um espaço latente, com distribuição p(z). Esse vetor é então fornecido ao gerador e combinado com os pesos $\mathbf{w_G}$, gerando o sinal $\mathbf{x_G} = G(\mathbf{z}, \mathbf{w_G})$.

O discriminador funciona como um classificador binário, que busca identificar se os dados fornecidos foram gerados por G. Para tanto, utiliza os pesos $\mathbf{w}_D$ e uma função de ativação sigmoidal na saída e é treinado com um conjunto de dados composto por dados sintéticos criados pelo gerador $\mathbf{x}_G$ e dados reais $\mathbf{x}_R$. A Figura 22 representa o procedimento descrito.

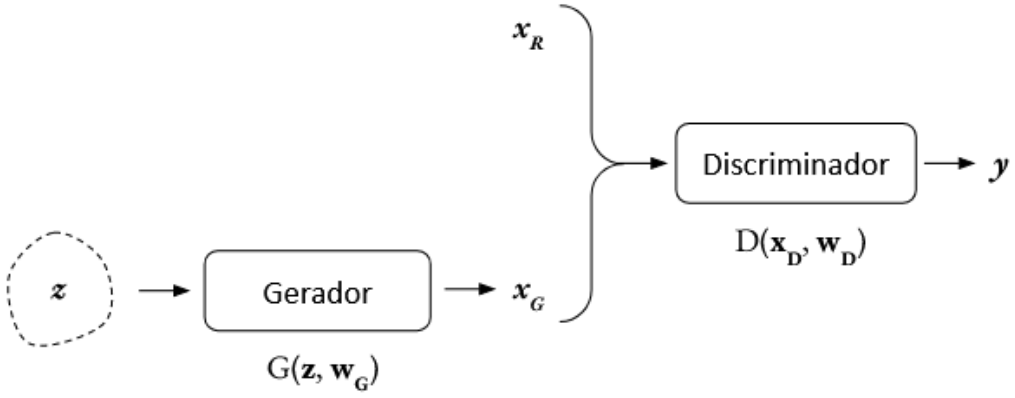


Figura 22: Esquema de funcionamento da GAN.

Para diferenciar os dados reais dos dados gerados, durante a etapa de treinamento, são utilizados rótulos para cada tipo de sinal sendo

$$l = \begin{cases} 1, & \text{para dados reais,} \\ 0, & \text{para dados sintéticos.} \end{cases}$$

Desse modo, idealmente, a resposta do discriminador deveria ser $\frac{1}{2}$ para todos os dados fornecidos, representando a impossibilidade da rede de diferenciar os dados reais dos dados sintéticos. O discriminador, D, é então treinado para maximizar a probabilidade de acertar o rótulo de cada dado fornecido, enquanto o gerador, G, é treinado para minimizar a função $\log(1 - D(G(\mathbf{z})))$. Desse modo, a função V, correspondente ao treinamento de ambos os modelos, de acordo com o jogo de minimax [54] é dada por:

$$\min_G \max_D V(D,G) = E_{x_D \sim p_{dados}(x_D)}[\log(D(x_D))] + E_{z \sim p(z)}[\log(1 - D(G(z)))], \tag{34}$$

onde E representa a esperança matemática calculada conforme a distribuição indicada em subescrito.

Apesar de apresentar diversas vantagens na geração de dados sintéticos, as redes do tipo GAN, comumente apresentam dificuldades no treinamento [56, 57]. Desse modo, foram desenvolvidas técnicas para aperfeiçoar o treinamento dessas redes. Uma dessas técnicas, relevantes para o projeto proposto, será apresentada a seguir.

A.3.2 Wasserstein GAN

A rede Wasserstein GAN (WGAN) [58] foi desenvolvivida com o objetivo de estabilizar o treinamento da rede GAN ao combater problemas comuns encontrados durante o treinamento como convergência de gradiente e *mode collapse* [58–60]. Esse objetivo foi alcançado através da criação de uma nova função custo, *Wasserstein Loss*, e pelo aumento da estabilidade do processo de otimização da rede.

Primeiramente, para a *Wassertein Loss* (WLOSS), é preciso que se mude o valor dos rótulos, l, utilizados. Nesse caso, utiliza-se $l = 1$ para os sinais reais e $l = -1$ para os sinais sintéticos e substitui-se a função sigmoidal na saída do discriminador por uma saída linear. Dadas essas mudanças o discriminador da WGAN é comumente nomeado como "crítico"e a função custo é dada por

$$\text{WLOSS}(y, p) = -\frac{1}{n}\sum_{i=1}^{n}(y_i p_i). \tag{35}$$

Similarmente a função BCE, a WLOSS faz uma correção entre o sinal obtido $\mathbf{p}$ e o sinal desejado $\mathbf{y}$. Desse modo, a minimização da função custo do crítico, na WGAN, pode ser dada por

$$\min_D V(D) = -(E_{x\sim p_{dados}(x_D)}[D(x_D)] - E_{z\sim p(z)}[D(G(z))]), \tag{36}$$

e a minimização da função custo do gerador por

$$min_G V(G) = -(E_{z\sim p(z)}[D(G(z))]). \tag{37}$$

Devido a substituição da função custo, é preciso também adicionar uma restrição aos pesos, para que não ocorra uma explosão de gradiente [58]. Para tanto, utiliza-se um método denominado *weight clipping* [59], restrigindo os pesos do crítico a cada iteração do minibatch.

Usualmente, durante o treinamento da rede WGAN, o crítico é treinado múltiplas vezes para cada treinamento do gerador, sendo comumente utilizada uma razão de cinco para um, além de utilizar-se um menor passo de adaptação e optar pelo otimizador RMSProp em detrimento do Adam [59].

Para exemplificar os conceitos propostos nessa seção, será apresentado a seguir uma aplicação de rede GAN para geração de sinais periódicos.

A.4 Rede GAN para geração de sinais senoidais

Tendo em vista a qualidade da rede de tipo GAN para aprender as distribuições dos dados fornecidos, propõe-se a geração de pontos representativos de um sinal sintético a partir de sinais aleatórios retirados de um espaço latente. O objetivo desse experimento é observar se a rede GAN é capaz de simular um resultado semelhante ao apresentado na Figura 23, partindo de um sinal de entrada aleatório.

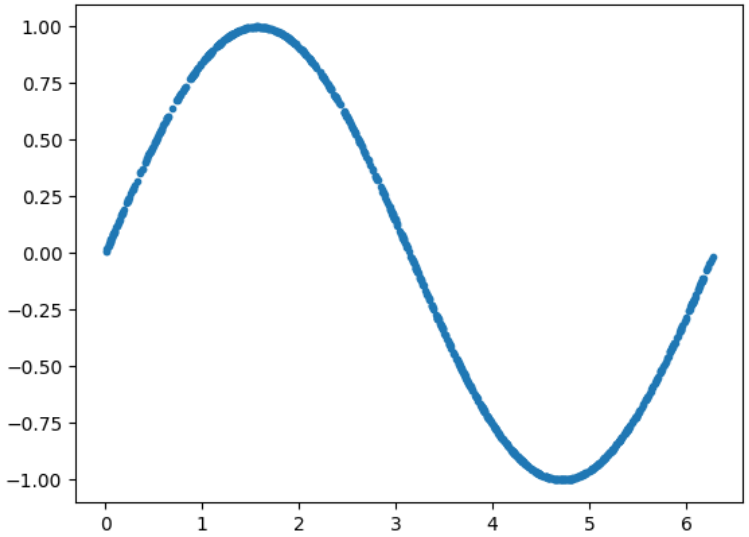


Figura 23: Sinal senoidal real.

No exemplo apresentado, uma série de 1024 pares ordenados (x_1,x_2), pertencentes a distribuição

$$x_1 = 2\pi i \ , i \in [0, 1[\ , \tag{38}$$
$$x_2 = \text{sen}(x_1), \tag{39}$$

determina a curva apresentada.

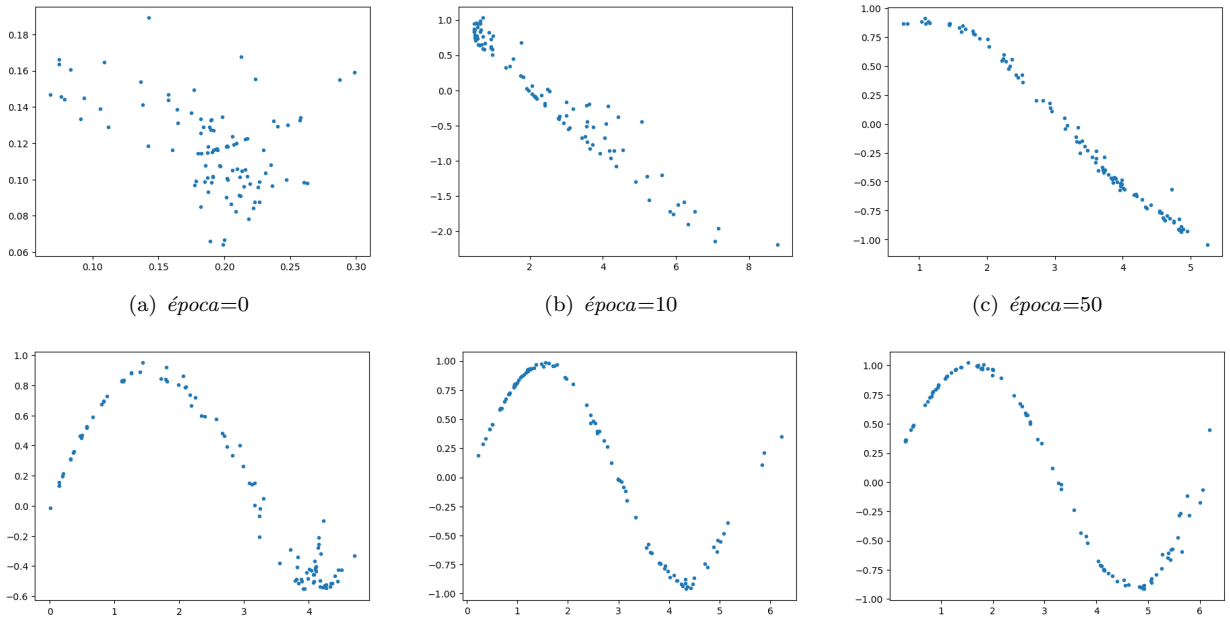
Para as configurações da rede GAN, tanto para o discriminador quanto para o gerador, foram utilizadas redes MLP, nas configurações apresentadas na Tabela 6.

Tabela 6: Configurações do Gerador e Discriminador.

Gerador	Discriminador
[2-2(ReLU)-16(ReLU)-32(ReLU)-2(Linear)]	[2-2(ReLU)-256(ReLU)-128(ReLU)-64(ReLU)-1(Sigmoidal)]

Desse modo, para o gerador foi utilizada uma rede com 2 camadas ocultas e uma camada de saída com duas saídas, representando o par ordenado desejado. Similarmente para o discriminador, foram utilizadas três camadas ocultas e uma camada de saída com uma saída. O neurônio do discriminador deve decidir se o sinal de entrada é real ou sintético. Além disso, foi aplicada a técnica de *dropout* [44] no discriminador.

Para o treinamento foram consideradas 300 épocas, com um passo de adaptação $\eta = 0.001$ e algoritmo de otimização Adam. Os resultados para diferentes épocas podem ser observados na Figura 24.



(a) *época*=0 (b) *época*=10 (c) *época*=50

(d) *época*=100 (e) *época*=200 (f) *época*=300

Figura 24: Curvas de sinais senoidais sintéticos gerados ao longo de 300 épocas.

Sendo o gráfico relativo a esse treinamento apresentado na Figura 25 como o erro de treinamento em função do número de épocas.

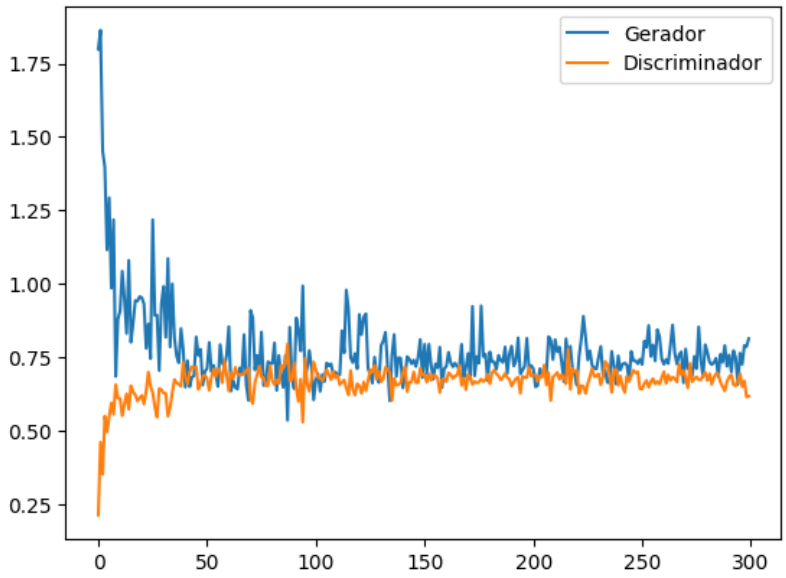
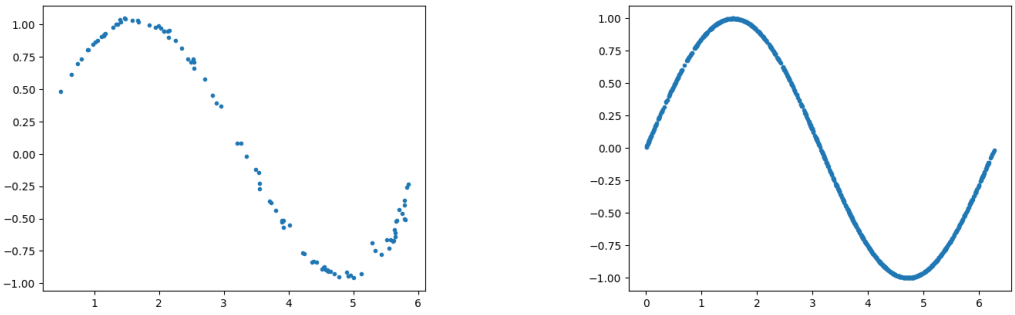


Figura 25: Gráfico de treinamento do gerador e discriminador ao longo de 300 épocas.

Como esperado, com o passar das épocas as curvas de custo do gerador e do discriminante convergem até a estabilização do treinamento. É possível observar, comparando a curva real e a curva sintética gerada pela rede totalmente treinada apresentadas na Figura 26, que a rede GAN é capaz de gerar o sinal sintético senoidal desejado de maneira satisfatória.



(a) Pontos gerados pela GAN

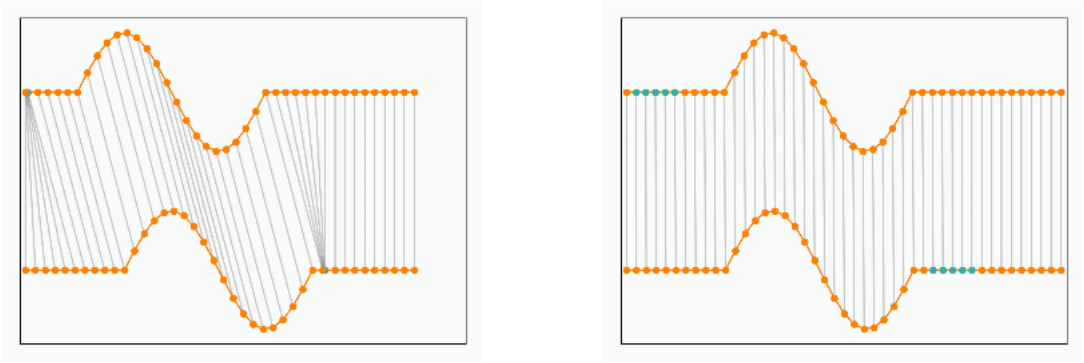
(b) Sinal senoidal real

Figura 26: Comparação entre sinal senoidal sintético e real.

Desse modo, a rede GAN proposta foi capaz de convincentemente pontos semelhantes aos pontos reais, utilizados para treinamento, com a mesma distribuição. É válido ressaltar que os resultados obtidos, apesar de satisfatórios, não representam perfeitamente o sinal senoidal, podendo a rede ser ainda ajustada mais profundamente a fim de obter resultados ainda mais similares ao reais. Esses ajustes, entretanto, não serão feitos nesse projeto. A seguir será apresentada a métrica *dynamic time warping* (DTW), que será utilizada para comparação dos sinais reais e sintéticos.

A.5 *Dynamic Time Warping*

O DTW é uma métrica utilizada para medir a similaridade entre duas séries temporais [62]. Para tal, o DTW usa um algoritmo para fazer um alinhamento temporal entre as duas séries temporais, ou seja, fazer um alinhamento dos elementos de cada série com a outra, de modo a minimizar a distância euclidiana entre elas [63]. A Figura 27 ilustra o processo descrito.



(a) (b)

Figura 27: Alinhamento temporal de duas séries.
Fonte: [63]

Desse modo, considerando duas curvas $\mathbf{x}$ e $\mathbf{x}'$, temos que a expressão do DTW para elas é formulada por

$$\mathrm{DTW}(\mathbf{x}, \mathbf{x}') = \min_{\pi \in \mathbf{A}(\mathbf{x}, \mathbf{x}')} \left(\sum_{(i,j) \in \pi} d(x_i, x'_j)^q \right)^{\frac{1}{q}}, \tag{40}$$

onde π identifica uma trajetória de alinhamento de tamanho K. Essa trajetória consiste de um sequência de K pares $((i_0, j_0), \cdots, (i_{K-1}), j_{K-1}))$. Além disso, $\mathbf{A}(\mathbf{x}, \mathbf{x}')$ é o conjunto de todas as trajetórias possíveis [63]. Assim, para determinar a trajetória ideal, deve-se garantir que ambas as séries temporais comecem no mesmo ponto, ou seja,

$$\pi_0 = (0,0) \tag{41}$$

$$\pi_{K-1} = (n-1, m-1), \tag{42}$$

e a sequência deve ser monotonicamente crescente tanto em i quanto em j, além de todos os índices aparecerem pelo menos uma vez [62,63]. Mais informações sobre essa métrica podem ser encontrada em [63]. A seguir será explorado o funcionamento de uma rede do tipo Autoencoder.

A.6 *Autoencoder*

O modelo de rede neural Autoenconder [1] trata de um modelo especialmente eficaz no processo de geração de dados sintéticos [1]. Esse modelo propõe, assim como na rede GAN, uma estrutura que consiste em duas subestruturas neurais, Encoder e Decoder, combinadas.

As subsestruturas de Encoder e Decoder, são estruturas específicas em que, dada uma entrada $\boldsymbol{x}$, a rede Encoder transforma esse sinal em uma representação intermediária $\boldsymbol{z(x)}$, enquanto a rede Decoder interpreta esse sinal $\boldsymbol{z(x)}$ em uma saída $\boldsymbol{y}$. Essa saída tenta emular o sinal de entrada $\boldsymbol{x}$, estando sujeita, porém, aos parâmetros aprendidos pelos modelos da rede. As especificidades desse modelo serão apresentados a seguir.

A.6.1 Estrutura Neural

A rede Autoencoder consiste, tipicamente, em uma estrutura que possui um sinal de entrada com mesmo tamanho do sinal de saída [1], em uma tentativa de mapear o próprio sinal de entrada na saída de mesma dimensão. Para impedir uma solução trivial, onde a rede simplesmente repassa o sinal inalterado para a saída, é implementada uma camada intermediária reduzida, forçando a adaptação de parâmetros. A Figura 28 ilustra uma estrutura simplificada desse tipo de rede utilizando camadas MLP em uma configuração [5-5-3-5].

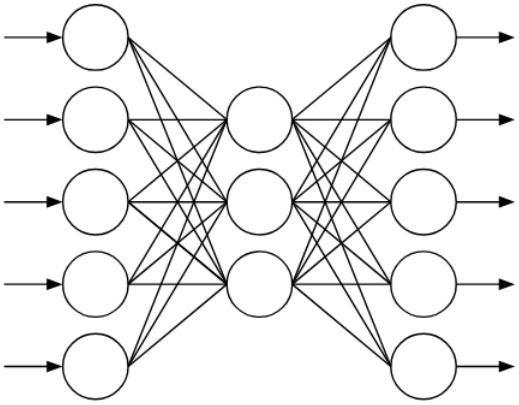


Figura 28: Modelo Autoencoder com estrutura [5-5-3-5].

Como, para esse tipo de estrutura, tipicamente, se utiliza um número de neurônios na camda oculta menor do que o número de neurônios da camada de entrada e saída, a reconstrução do sinal de entrada de maneira perfeita se torna impossível, de modo geral [1]. Assim, os parâmetros treináveis de cada camada da rede neural adicionam determinado grau de distorção ao sinal obtido. Como a estrutura da rede se baseia numa rede MLP, o algoritmo de treinamento utilizado é similar ao algoritmo *backpropagation* [44] apresentado no apêndice A.1.4.

Para adaptar a arquitetura com o mapeamento desejado do sinal de entrada, utiliza-se uma função específica no treinamento desse tipo de rede. Essa função será apresentada a seguir.

A.6.2 Divergência de Kullback Leribler

A Divergência de Kullback-Leibler (KL) [70] é uma parte fundamental no treinamento de uma rede VAE, sendo usada para regularizar a distribuição latente aprendida pela rede. Essa função mede a diferença entre duas distribuições de probabilidade. No contexto do VAE, ela é usada para garantir que a distribuição latente $q_\varphi(\mathbf{z}|\mathbf{x})q_\varphi \cdot (\mathbf{z}|\mathbf{x})$ esteja próxima da distribuição alvo $p(\mathbf{z})$.

Essa função é composta principalmente por duas parte. A primeira parte é um Termo de Reconstrução (E), responsável por medir o quão bem o VAE consegue reconstruir a entrada $\mathbf{x}$ a partir da amostra latente $\mathbf{z}$. Este termo é semelhante à função de custo usada em autoencoders tradicionais. Além disso, é utilizado um Termo de Regularização (Divergência KL) [70]: A Divergência KL é usada para regularizar a distribuição latente e assegurar que ela não se desvie muito da distribuição normal $p(\mathbf{z})$. Desse modo, a função pode ser expressa por

$$\Theta = E(\mathbf{x}, \mathbf{y}) - \alpha KL(\mu, \varepsilon), \tag{43}$$

onde α indica um peso definido para o Termo de Regularização [70].

Além disso, para permitir a adaptação do sinal de saída do Encoder, de modo a fazer a manutenção das características do sinal original, na transmissão para o Decoder, é utilizado um método de reparametrização. Esse método será apresentado a seguir.

A.6.3 Método de reparametrização

O método de reparametrização [71] permite que o modelo faça a amostragem da variável latente $\mathbf{z}$ de forma diferenciável, para que seja possível aplicar *backpropagation* [44] e otimizar a rede neural. A ideia do método de reparametrização é transformar o processo de amostragem de uma distribuição complexa (por exemplo, uma distribuição normal em um processo diferenciável, separando a parte determinística da estocástica.

No VAE, o espaço latente $\mathbf{z}$ é normalmente assumido como uma distribuição normal $N(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$, onde $\boldsymbol{\mu}$ e $\boldsymbol{\sigma}$ são os parâmetros que o modelo aprende. Em vez de amostrar diretamente de $N(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$, introduzimos uma variável auxiliar $\boldsymbol{\epsilon}$ que segue uma distribuição normal padrão $N(0,1)$. A variável latente $\mathbf{z}$ é então expressa como uma transformação da variável $\boldsymbol{\epsilon}$ dada por,

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}. \tag{44}$$

Aqui, $\boldsymbol{\epsilon}$ é uma variável estocástica (fixa) e a equação acima é diferenciável em relação a $\boldsymbol{\mu}$ e $\boldsymbol{\sigma}$, que são os parâmetros aprendidos pela rede neural.

O método de reparametrização do VAE resolve o problema da não diferenciabilidade [71] ao transformar a amostragem estocástica da variável latente $\mathbf{z}$ em uma função diferenciável em relação aos parâmetros $\boldsymbol{\mu}$ e $\boldsymbol{\sigma}$. Isso permite que o processo de amostragem seja integrado ao treinamento via backpropagation, essencial para a otimização do modelo.

B Artigo SBrT - 2024

XLII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS - SBrT 2024, 01–04 DE OUTUBRO DE 2024, BELÉM, PA

Classificação de sinais de ECG sintéticos

Eduardo P. L. Jaqueira, Renato Candido e Magno T. M. Silva

***Resumo*— Neste artigo, sinais de eletrocardiograma sintéticos foram gerados utilizando dois modelos gerativos: um baseado na rede adversária gerativa e outro no autocodificador variacional. Os sinais sintéticos foram classificados por uma rede perceptron multicamada treinada com sinais reais. A taxa de acerto de classificação dos sinais sintéticos foi superior a 80%, o que indica que esses sinais podem ser usados para melhorar as métricas de classificação de arritmias cardíacas.**

***Palavras-Chave*— Aprendizado de máquina, rede adversária gerativa, autocodificador variacional, eletrocardiograma, aumento de dados.**

***Abstract*— In this paper, synthetic electrocardiogram signals were generated using two generative models: one based on the generative adversarial network and other on the variational autoencoder. The synthetic signals were classified by a multilayer perceptron network trained with real signals. The correct classification rate of synthetic signals was superior to 80%, which indicates that these signals can be used to improve cardiac arrhythmia classification metrics.**

***Keywords*— Machine learning, generative adversarial network, variational autoencoder, electrocardiogram, data augmentation.**

I. Introdução

Técnicas de aprendizado de máquina [1] têm sido amplamente estudadas para a classificação de arritmias cardíacas por meio do sinal de eletrocardiograma (ECG) [2], [3]. Entretanto, altas taxas de erro dessas técnicas fazem com que elas ainda estejam longe de serem empregadas na prática. Em [3], foram propostas soluções baseadas em redes pecerptron multicamada (*multilayer perceptron* – MLP), redes neurais recorrentes, análise de discriminante linear e combinações desses modelos para classificação automática de arritmias cardíacas. Foram utilizados sinais de ECG do banco de dados MIT-BIH *Arrhythmia Database* (MITDB) [4], levando-se em conta quatro classes: batimentos do nó sinoatrial (N), supraventriculares ectópicos (S), ventriculares ectópicos (V) e fusão de batimentos normais e ventriculares ectópicos (F). Dentre as soluções propostas, a combinação da MLP com a LDA apresentou o melhor desempenho e alcançou métricas de classificação superiores às da literatura para as classes S e V.

Apesar de terem sido consideradas técnicas usuais para lidar com classes desbalanceadas em [3], acredita-se que métricas de classificação superiores só podem ser alcançadas com um banco de dados balanceado. Na classificação de arritmias cardíacas, a técnica de *data augmentation* com modelos gerativos também tem sido usada para gerar sinais de ECG sintéticos com o objetivo de balancear os bancos de dados [5], [6]. Neste trabalho, são utilizados uma rede adversária gerativa (*gerative adversarial network* – GAN) [5]–[7] e um autocodificador variacional (*variational autoencoder* – VAE) [6], [8] para gerar sinais de ECG sintéticos da classe N. 92% dos batimentos do MITDB são dessa classe. Apesar de ser mais importante gerar batimentos das classes menos representativas na busca do balanceamento do banco de dados, decidiu-se iniciar com a classe N neste estudo preliminar já que há mais dados para o treinamento dos modelos gerativos.

O artigo está organizado da seguinte forma. Na Seção II, são descritos os modelos gerativos propostos. Na Seção III, são mostrados os resultados de simulação e a Seção IV fecha o artigo com as principais conclusões e trabalhos futuros.

Eduardo P. L. Jaqueira, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, emails: eduardo.jaqueira@usp.br; renatocan@lps.usp.br; magno.silva@usp.br. Este trabalho foi financiado pela CAPES (código de financiamento 001), pelo CNPq (127301/2023-2, 303826/2022-3 e 404081/2023-1) e pela FAPESP (2021/02063-6).

II. Modelos propostos

A GAN conta com duas redes neurais: o gerador e o discriminador [7]. Um vetor de espaço latente $\mathbf{x}$ de tamanho 100 é gerado a partir de uma distribuição normal de média zero e variância unitária. O vetor $\mathbf{x}$ é então fornecido ao gerador, produzindo um sinal sintético $\mathbf{x}_G$ de tamanho 320. Em seguida, $\mathbf{x}_G$ é transmitido ao discriminador, onde é classificado como "real" ou "sintético". Como o objetivo da pesquisa é gerar sinais próximos dos reais, idealmente o discriminador não deve ser capaz de diferenciar sinais reais do banco de dados de sinais sintéticos gerados pelo gerador treinado. No gerador da GAN, utilizou-se uma rede neural recorrente bidirecional composta por blocos BiLSTM (*bidirectional long short-term memory*) [10]. Especificamente, foram utilizadas duas camadas BiLSTM com vetor de estados de tamanho 100 e tamanho do *mini-batch* igual a 22, sendo então a saída dessa rede submetida a uma camada linear e à função tangente hiperbólica. No discriminador, foi utilizada uma rede MLP composta de 6 camadas ocultas, diminuindo gradualmente o número de neurônios de cada camada $(320, 2^8, 2^7, \cdots, 2^4)$, e uma camada de saída com 1 neurônio. O esquema da arquitetura descrita para a rede GAN pode ser observado na Figura 1. Consideraram-se ainda a entropia cruzada binária como função custo, ReLU e sigmoidal como funções de ativação nas camadas ocultas e de saída, respectivamente, otimizador Adam, *minibatch* de 22, passo de adaptação de 5×10^{-5}, *dropout* e treinamento por 400 épocas.

No codificador e decodificador do VAE [8], foram utilizadas redes MLP com duas e três camadas, respectivamente. Na saída do codificador, foi feita uma reparametrização do sinal [9]. Em ambas as redes, consideraram-se uma combinação do erro quadrático médio e da divergência de Kullback-Leibler [1], [9] como função custo, LeakyReLU como função de ativação, normalização de *batch*, 600 épocas, *minibatch* de 22 e passo de adaptação de 10^{-4}. No treinamento do VAE, um sinal de ECG real $\mathbf{x}$ de tamanho 320 é fornecido ao codificador

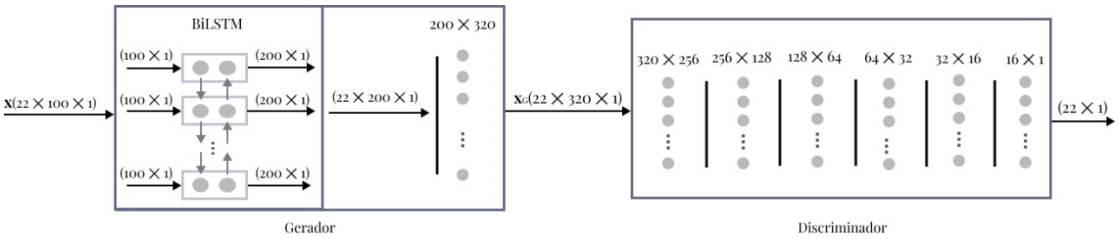


Fig. 1: Arquitetura esquemática da GAN com blocos BiLSTM no gerador e rede MLP no discriminador.

com 320 neurônios na camada de entrada. Esse sinal é então expandido para a camada seguinte contendo 512 neurônios e em seguida comprimido, produzindo três vetores de tamanho 10: $\mathbf{z}$, $\boldsymbol{\mu}$ e $\boldsymbol{\varphi}$. O vetor $\mathbf{z}$ é fornecido ao decodificador que procura "recuperar" o sinal anterior, expandindo esse vetor ao longo das camadas. Primeiramente para uma camada de 512 neurônios e chegando a uma camada de 320 neurônios, obtendo a saída $\mathbf{y}$. O esquema da arquitetura descrita para essa rede pode ser observada na Figura 2.

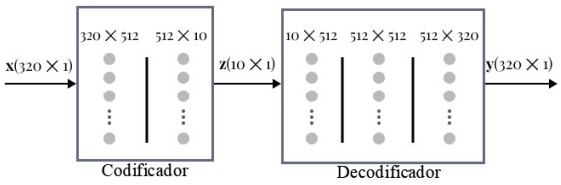


Fig. 2: Arquitetura esquemática do VAE.

III. Simulações e Resultados

Os modelos gerativos foram treinados com batimentos da classe N do MITDB. Como entrada, foram consideradas amostras de batimentos individuais centralizados em seus picos, normalizadas para o intervalo $[-1, 1]$ e organizadas em vetores de tamanho 320. Após o treinamento, os sinais sintéticos gerados foram comparados com um modelo representativo da classe N (*template*) [6], utilizando *dynamic time warping* (DTW) [11]. Calculou-se então a menor distância entre os batimentos gerados e o *template*, denotada por s_3. A partir dessa distância, definiu-se um limiar $(2,5 \times s_3)$. Foram considerados como aceitáveis apenas os sinais com valores de s_3 inferiores ao limiar, o que levou a taxas de aproveitamento de $26,5\%$ e $8,8\%$ dos sinais gerados para a GAN e para o VAE, respectivamente.

Os batimentos aceitos foram unidos de modo a gerar sinais com três batimentos em vetores de tamanho 960. Esse sinais foram então classificados com a rede MLP treinada com sinais de ECG reais de [3]. Os resultados da classificação estão na Tabela I. A taxa de acerto na classificação dos batimentos gerados pela GAN foi de 85% enquanto para o VAE essa taxa ficou em torno de 81%. Houve erros de classificação divididos entre as classes S e V. Cabe observar que a acurácia geral da rede MLP de [3] é de 77,2%, sendo que para a classe N, ela apresentou sensibilidade de 82,4%, precisão de 88,7% e F_1-*score* de 85,4%. Comparando essas métricas com as taxas de acerto obtidas, conclui-se que os sinais sintéticos gerados são adequados.

TABELA I: Percentual da classificação - ECGs sintéticos (N).

	N	S	V	F
GAN	85,0%	6,3%	8,7%	0,0%
VAE	80,6%	7,2%	12,2%	0,0%

IV. Conclusão

A geração de sinais de ECG sintéticos é uma solução promissora para o balanceamento de bancos de dados e obtenção de melhores métricas de classificação de arritmias cardíacas. O estudo preliminar realizado neste artigo indica que, para as configurações propostas, a GAN apresenta vantagem quanto à "qualidade" dos sinais sintéticos gerados em relação ao VAE. Entretanto, o tempo de treinamento e o custo computacional do VAE é cerca de cinco vezes menor que o da GAN. Por isso, vale a pena buscar configurações do VAE que levem a resultados semelhantes aos da GAN em temos de aproveitamento dos sinais gerados e taxa de acerto de classificação. Dando continuidade à pesquisa, pretende-se utilizar os mesmos métodos aqui apresentados para gerar sinais de ECG sintéticos das classes menos representativas. Em seguida, pretende-se retreinar a rede MLP de [3] com um banco de dados sintético balanceado e validar com dados reais.

Referências

[1] C. Bishop, *Deep Learning: Foundations and Concepts*, Springer, 2023.
[2] S. K. Berkaya *et al.*, "A survey on ECG analysis," *Biomedical Signal Processing and Control*, vol. 43, pp. 216–235, 2018.
[3] N. Nagata, R. Candido, M. T. M. Silva, "Combinaçõoes de redes neurais e discriminantes lineares para classificação de arritmias cardíacas," in Anais do SBrT, Fortaleza, CE, 2021.
[4] G. B. Moody, R. G. Mark, "The impact of the MIT-BIH arrhythmia database," IEEE Eng. Med. Biol. Mag., vol. 20, pp. 45–50, 2001.
[5] A. M. Delaney, E. Brophy, T. E. Ward, "Synthesis of realistic ECG using generative adversarial networks," disponível em `https://arxiv.org/abs/1909.09150`, arXiv, 2019.
[6] E. Adib, F. Afghah, J. J. Prevost, "Synthetic ECG signal generation using generative neural networks," dispnível em `https://arxiv.org/abs/2112.03268`, arXiv, 2021.
[7] I. Goodfellow *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, pp. 139–144, 2020.
[8] D. P. Kingma, M. Welling, *An Introduction to Variational Autoencoders*, Now Publishers, Norwell, MA, 2019.
[9] L. Dinh, *Reparametrization in deep learning*, Tese de Doutorado, Univ. Montréal, 2018.
[10] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", *Physica D: Nonlinear Phenomena*, vol. 404, 2020.
[11] P. Senin, *Dynamic time warping algorithm review*, Tecnhical report, University of Hawaii, 2008.

Referências

[1] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ECG analysis," Biomedical Signal Processing and Control, vol. 43, pp. 216–235, 2018.

[2] P. de Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," IEEE Transactions on Biomedical Engineering, vol. 51, no. 7, pp. 1196–1206, 2004.

[3] T. Mar, S. Zaunseder, J. P. Mart´ınez, M. Llamedo, and R. Poll, "Optimization of ECG classification by means of feature selection," IEEE Transactions on Biomedical Engineering, vol. 58, no. 8, pp. 2168–2177, 2011.

[4] C. Lin and C. Yang, "Heartbeat classification using normalized RR intervals and morphological features," Mathematical Problems in Engineering, vol. 2014, 2014.

[5] G. Garcia, G. Moreira, E. Luz, and D. Menotti, "Improving automatic cardiac arrhythmia classification: Joining temporal-VCG, complex networks and SVM classifier," in 2016 International Joint Conference on Neural Networks (IJCNN). IEEE, 2016, pp. 3896–3900.

[6] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network," Information Sciences, vol. 405, pp. 81–90, Sep. 2017.

[7] U. R. Acharya, H. Fujita, O. S. Lih, M. Adam, J. H. Tan, and C. K. Chua, "Automated detection of coronary artery disease using different durations of ECG segments with convolutional neural network," Knowledge-Based Systems, vol. 132, pp. 62–71, Sep. 2017.

[8] G. Garcia, G. Moreira, D. Menotti, and E. Luz, "Inter-patient ECG heartbeat classification with temporal VCG optimized by PSO," Scientific Reports, vol. 7, no. 1, pp. 1–11, 2017.

[9] R. Banerjee, A. Ghose, and S. Khandelwal, "A novel recurrent neural network architecture for classification of atrial fibrillation using single-lead ECG," in Proc. of 27th European Signal Processing Conference (EUSIPCO), Sep. 2019, pp. 1–5.

[10] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," Nature Medicine, vol. 25, pp. 65–69, Jan. 2019.

[11] N Nagata, R. Candido, and M. T. M. Silva, "Combinaçõoes de redes neurais e discriminantes lineares para classificação de arritmias cardíacas," in Anais do Simp´osio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'21), Fortaleza, CE, 2021.

[12] M. A. Reyna et al., "Heart murmur detection from phonocardiogram recordings:The george b. moody physionet challenge 2022," medRxiv preprint, pp. 1–16, 2022, https://www.medrxiv.org/content/10.1101/2022.08.11.22278688v1.

[13] Association for the Advancement of Medical Instrumentation et al., "ANSI/AAMI EC57:1998/(R)2008 - Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," American National Standards Institute, Arlington, VA, USA, 2008, Association for the Advancement of Medical Instrumentation (AAMI), ANSI/AAMI/ISO EC57,1998-(R)2008, 2008.

[14] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, *"PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," Circulation, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full* PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[15] G. B. Moody and R. G. Mark, *"The impact of the MIT-BIH arrhythmia database," IEEE Engineering in Medicine and Biology Magazine, vol. 20, no. 3, pp. 45–50,* 2001.

[16] N. Nagata, R. Candido, and M. T. M. Silva, *"Classification of arrhythmias using multilayer perceptron neural networks," in 28th USP International Symposium of Undergraduate Research,* 2020.

[17] N Nagata, R. Candido, and M. T. M. Silva, "O efeito da divisão de dados na classificação de arritmias usando redes neurais convolucionais.," in Anais do Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'21), Fortaleza, CE, 2021.

[18] N Nagata, R. Candido, and M. T. M. Silva, "Classificaação de arritmias cardíacas com redes neurais perceptron multicamada," in 73a Reunião Anual da Sociedade Brasileira para o Progresso da Ciência (SBPC'21), Juiz de Fora, MG, 2021.

[19] N. Nagata, R. Candido, and M. T. M. Silva, "Classificaação de arritmias cardíacas utilizando redes neurais convolucionais," in 29o Simpósio Internacioanal de Iniciação Científica da USP, 2021.

[20] F. M. Nolle, F. K. Badura, J. M. Catlett, R. W. Bowser, and M. H. Sketch, *"CREI-GARD, a new concept in computerized arrhythmia monitoring systems," Computers in Cardiology, vol.* 13, pp. 515–518, 1986.

[21] D. Dua and C. Graff, *"UCI machine learning repository,"* 2017, Disponível em: http://archive.ics.uci.edu/m.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *"SMOTE: synthetic minority oversampling technique," Journal of artificial intelligence research, vol.* 16, pp. 321–357, 2002.

[23] Y. Ho and S. Wookey, *"The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," IEEE Access, vol.* 8, pp. 4806–4813, 2020.

[24] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *"Generative adversarial networks* ,"https://arxiv.org/abs/1406.2661, 2014.

[25] I. Goodfellow et al., *"Generative adversarial networks," Communications of the ACM, vol.* 63, no. 11, pp. 139–144, Nov. 2020.

[26] Diederik P. Kingma and Max Welling, *"Auto-encoding variational bayes,"* https://arxiv.org/abs/1312.6114, 2013.

[27] Diederik P. Kingma and Max Welling, *An Introduction to Variational Autoencoders, Now Publishers*, Norwell, MA, 2019.

[28] Anne Marie Delaney, Eoin Brophy, and Tomas E. Ward, *"Synthesis of realistic ECG using generative adversarial networks,*" arXiv, https://arxiv.org/abs/1909.09150, 2019.

[29] Rohan Banerjee and Avik Ghose, *"Synthesis of realistic ecg waveforms using a composite generative adversarial network for classification of atrial fibrillation," in Proc. of 29th European Signal Processing Conference* (EUSIPCO), 2021, pp. 1145–1149.

[30] Fan Cao, Aamani Budhota, Hao Chen, and Kuldeep Singh Rajput, *"Feature matching based ecg generative network for arrhythmia event augmentation," in Proc. of 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society* (EMBC), 2020, pp. 296–299.

[31] Edmond Adib, Fatemeh Afghah, and John J. Prevost, *"Arrhythmia classification using CGAN-augmented ECG signals," in Proc. of IEEE International Conference on Bioinformatics and Biomedicine* (BIBM), 2022, pp. 1865–1872.

[32] Nunes, Sara Monteiro Morgado Dias. Subamostragem em séries temporais. Diss. 2001.

[33] E. J. S. Luz et al., *"ECG-based heartbeat classification for arrhythmia detection: A survey," Computer methods and programs in biomedicine, vol.* 127, pp. 144–164, 2016.

[34] G. Garcia et al., *Inter-patient ECG heartbeat classification with temporal VCG opti- mized by PSO," Scientific Reports, vol.* 7, no. 1, pp. 1–11, 2017.

[35] Adib, Edmond, Fatemeh Afghah, and John J. Prevost. "Synthetic ecg signal generation using generative neural networks."arXiv preprint arXiv:2112.03268 (2021).

[36] Frank Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological Review*, 65(6):386, 1958.

[37] C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.

[38] S. Haykin, "Neural Networks and Learning Machines", 3/E. Pearson Education, 2009.

[39] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[40] Medsker, Larry R., and L. C. Jain. "Recurrent neural networks."Design and Applications 5.64-67 (2001): 2.

[41] Ramos, Daniel, et al. "Deconstructing cross-entropy for probabilistic binary classifiers."Entropy 20.3 (2018): 208.

[42] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory."Neural computation 9.8 (1997): 1735-1780.

[43] A. Y. Ng, K. Katanforoosh, and Y. B. Mourri, "Neural networks and deep learning," deeplearning.ai, disponível em: https://www.coursera.org/learn/neural-networks-deep-learning. Acesso em: 21 jan. 2020.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[45] Wöllmer, Martin, et al. "Bidirectional LSTM networks for context-sensitive keyword detection in a cognitive virtual agent framework."Cognitive Computation 2 (2010): 180-190.

[46] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks."Advances in neural information processing systems 27 (2014).

[47] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks,"in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.

[48] S. Siami-Namini, N. Tavakoli and A. S. Namin, "The Performance of LSTM and BiLSTM in Forecasting Time Series,"2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019.

[49] https://medium.com/@souro400.nath/why-is-bilstm-better-than-lstm-a7eb0090c1e4

[50] Goodfellow, Ian, et al. "Generative adversarial nets."Advances in neural information processing systems 27 (2014).

[51] Wang, Xuan, et al. "A Review of GAN-Based Super-Resolution Reconstruction for Optical Remote Sensing Images."Remote Sensing 15.20 (2023): 5062.

[52] Chen, Genlang, et al. "EmotionalGAN: Generating ECG to enhance emotion state classification."Proceedings of the 2019 International conference on artificial intelligence and computer science. 2019.

[53] Silva, Julio Cezar Soares, and Adiel Teixeira de Almeida Filho. "Using GAN-generated market simulations to guide genetic algorithms in index tracking optimization."Applied Soft Computing 145 (2023): 110587.

[54] Von Neumann, John. "On the Theory of Games of Strategy."Mathematische Annalen 100 (1928): 295-320.

[55] Bishop, Christopher M., and Hugh Bishop. "Deep learning: foundations and concepts."(2024).

[56] Zhang, Han, et al. “Self-attention generative adversarial networks.”International conference on machine learning. PMLR, 2019.

[57] Zhong, Jiachen, Xuanqing Liu, and Cho-Jui Hsieh. “Improving the speed and quality of gan by adversarial training.”arXiv preprint arXiv:2008.03364 (2020).

[58] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks.”International conference on machine learning. PMLR, 2017.

[59] Foster, David. Generative deep learning. “ O’Reilly Media, Inc.”, 2022.

[60] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” Advances in Neural Information Processing Systems, pp. 2234–2242, 2016.

[61] Kashima, Hisashi, et al. “K-means clustering of proportional data using L1 distance.”2008 19th International Conference on Pattern Recognition. IEEE, 2008.

[62] Senin, Pavel. “Dynamic time warping algorithm review.”Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA 855.1-23 (2008): 40.

[63] https://rtavenar.github.io/blog/dtw.html.

[64] A. Noordergraaf, Circulatory system dynamics, vol. 1, Elsevier, 2012.

[65] R. M. Rangayyan, Biomedical signal analysis, vol. 33, John Wiley & Sons, 2015.

[66] Garcia, Gabriel, et al. ”Inter-patient ECG heartbeat classification with temporal VCG optimized by PSO.”Scientific reports 7.1 (2017): 10543.

[67] D. B. Geselowitz, “*On the theory of the electrocardiogram*,” Proceedings of the IEEE, vol. 77, no. 6, pp. 857–876, 1989.

[68] P. E. McSharry et al., “*ECGSYN - A realistic ECG waveform generator (version 1.0.0)*,” 2003. Disponível em https://physionet.org/content/ecgsyn/1.0.0/.

[69] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.resample.html

[70] Kullback, Solomon. ”Kullback-leibler divergence.”(1951).

[71] DINH, Laurent. Reparametrization in deep learning. 2018.