UNIVERSIDADE DO ESTADO DO PARÁ – UEPA

CENTRO DE CIÊNCIAS NATURAIS E TECNOLOGIA – CCNT

CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE

HIAN MOTLEY STAFFORD CORREA BARROSO

Classificação de arritmias cardíacas com redes neurais artificiais: um estudo comparativo entre PyTorch e Tensorflow

Castanhal – PA

2024

HIAN MOTLEY STAFFORD CORREA BARROSO

Classificação de arritmias cardíacas com redes neurais artificiais: um estudo comparativo entre PyTorch e Tensorflow

Trabalho de Conclusão de Curso apresentado como requisito final para obtenção do grau de Bacharel em Engenharia de Software pela Universidade do Estado do Pará (UEPA), sob a orientação do Prof. Me. Eng. Ítalo Flexa Di Paolo.

Banca Examinadora:

Prof. Me. Ítalo Flexa Di Paolo – Orientador
Universidade do Estado do Pará - UEPA

Profa. Dra. Marta de Oliveira Barreiros
Universidade do Estado do Pará - UEPA

Profa. Dra. Adriana Rosa Garcez Castro
Universidade Federal do Pará - UFPA

Apresentado em: 29 / 08 / 2024
Nota: 9,5

Agradecimentos

Gostaria de expressar minha profunda gratidão ao meu orientador Ítalo, que desempenhou um papel crucial em cada etapa deste artigo. Seu apoio e orientação foram fundamentais para o meu progresso acadêmico, fornecendo a confiança e a motivação necessárias para alcançar meus objetivos nesta fase decisiva da graduação. Agradeço sinceramente por todo o apoio. Também sou grato aos meus amigos Ícaro e Rafael pela motivação, apoio, conhecimento e descontração que me ofereceram durante a graduação. Aos amigos de fora que acreditaram na ideia desde o início e me incentivaram a persistir. E à professora Marta que foi de fundamental importância para o meu crescimento acadêmico, mesmo no final do curso, numa área que pretendo seguir. Finalmente, agradeço à minha mãe e ao meu irmão pelo apoio constante, mesmo à distância, e por sempre me encorajarem a seguir em frente.

Classificação de arritmias cardíacas com redes neurais artificiais: um estudo comparativo entre PyTorch e Tensorflow

Hian Motley Stafford Correa Barroso[1]; Ítalo Flexa Di Paolo[2]

[1]Centro de Ciências Naturais e Tecnologia – Universidade do Estado do Pará
CEP: 68745-000 – Castanhal – PA – Brasil

[1]Centro de Ciências Naturais e Tecnologia – Universidade do Estado do Pará
CEP: 67125-118– Ananindeua – PA – Brasil

hian.mscbarroso@aluno.uepa.br; itflexa@uepa.br

Abstract. *This study presents and compares the PyTorch and TensorFlow frameworks for automatic classification of cardiac arrhythmias using the PTB electrocardiogram diagnostic dataset. Accuracy, precision, recall and f1-Score indicators will be evaluated, in addition to the computational training performance. MLP, ResNet18 and AlexNet models were implemented in both frameworks, in addition to comparison using 1D and 2D signals. PyTorch outperformed TensorFlow in terms of computational performance and classification performance indicators. This paper also highlights PyTorch's ability to use better-performing hardware. The results achieved were also compared with contemporary literature, reaching the state of the art, with 99.55% accuracy, 99.49% precision, 99.38% recall and 99.44% f1-Score.*

Resumo. *Este estudo apresenta e compara os frameworks PyTorch e TensorFlow para classificação automática de arritmias cardíacas usando o conjunto de dados de diagnóstico de eletrocardiograma PTB. Serão avaliados indicadores de acurácia, precisão, recall e f1-Score, além da performance computacional de treinamento. Foram implementados modelos MLP, ResNet18 e AlexNet em ambos os frameworks, além de comparação utilizando sinal 1D e 2D. O PyTorch superou o TensorFlow em termos de desempenho computacional e indicadores de desempenho de classificação. Este artigo ainda destaca a capacidade do PyTorch de utilizar hardware com melhor desempenho. Os resultados alcançados ainda foram comparados com a literatura contemporânea, alcançando o estado da arte, com 99,55% de acurácia, 99,49% de precisão, 99,38% de recall e 99,44% de f1-Score.*

1. Introdução

O eletrocardiograma (ECG) é um registro que mostra o que o músculo cardíaco faz durante um batimento cardíaco. Ele fornece informações úteis aos cardiologistas sobre como o coração funciona e o ritmo dele. As gravações de eletrocardiograma permitem que cardiologistas qualificados identifiquem uma variedade de anormalidades cardíacas. No entanto, um eletrocardiograma de 24 horas, capaz de registrar mais de 100.000 batimentos cardíacos, é frequentemente muito extenso para que um cardiologista possa fazer uma avaliação completa. Em termos práticos, uma inspeção visual pode levar horas para diagnosticar algumas doenças cardíacas, num processo manual e tedioso, passível de perda de informações importantes. Nesse contexto, sistemas automatizados podem apoiar cardiologistas para analisar com melhor precisão as inúmeras informações de eletrocardiograma coletadas por dispositivos de monitoramento (SUN et al, 2012).

Abordagens baseadas em aprendizado supervisionado, que buscavam adaptar métodos supervisionados comuns, dominaram as primeiras tentativas de abordar esse

problema. Usando métodos como análise discriminante linear, árvores de decisão ou abordagens baseadas em conjuntos aproximados para a classificação de ECGs, alguns estudos extraíram características relacionadas à morfologia do ECG, e a partir destes destaques nos sinais, técnicas baseadas em Redes Neurais Artificiais (RNA) podem ser usadas para classificação automática de ECGs (SUN et al, 2012).

Nos últimos anos, foram desenvolvidas estratégias de Aprendizado Semissupervisionado (SSL) para classificação de ECGs para resolver esse problema. Essas abordagens geralmente escolhem um subconjunto representativo de batimentos cardíacos do grupo de treinamento, recebem também a rotulagem manual dos cardiologistas e, em seguida, ensinam classificadores a ler esses batimentos rotulados. Essas estratégias ajudam na classificação de batimentos cardíacos ao usar o que os especialistas sabem sobre rótulos de batimentos, mas a quantidade e a qualidade dos batimentos escolhidos ainda são importantes (SUN et al, 2012).

As Redes Neurais Profundas (DNNs) aplicadas em ECG, especialmente as capacidades das Redes Neurais Convolucionais (CNNs) que usam convolução 1D ou 2D, vêm ganhando espaço em pesquisas recentes. Modelos de aprendizado profundo usam um mecanismo de aprendizado de ponta a ponta que usa dados como entrada e previsão de classe como saída. Eles podem automaticamente aprender características invariáveis e hierárquicas diretamente dos dados. O sinal ECG é 1D, e pode ser representado por 2D a partir dos processos de transformação, gerando uma representação matricial ou de imagem, que posteriormente sirvam de entrada para modelos de aprendizado profundo. Redes Neurais Recorrentes (RNN) e CNNs são modelos de aprendizado profundo comumente usados para classificar ECG 1D. As CNNs também podem ser usadas para classificar eletrocardiogramas transformados em 2D. É demonstrado experimentalmente que a representação 2D do eletrocardiograma oferece uma classificação de batimentos cardíacos mais precisa do que a representação 1D (AHMAD, 2021).

Neste estudo, são aplicadas técnicas de aprendizado profundo para a classificação automática de batimentos cardíacos. Em particular, examinamos os benefícios do uso de representações 1D e 2D dos sinais de eletrocardiograma, fazendo um estudo comparativo de ferramentas computacionais mais adequadas para esta finalidade, tendo como referência o ambiente em Python.

O objetivo geral deste trabalho é o de analisar técnicas e ferramentas de redes neurais artificiais para classificar arritmias cardíacas de forma automática. Para tanto, serão necessários os seguintes objetivos específicos:

- Selecionar uma base de dados pública de arritmias cardíacas;
- Definir um escopo de técnicas e ferramentas para classificação automática;
- Analisar os resultados obtidos entre as técnicas e ferramentas e em relação à literatura contemporânea.

Este artigo está organizado em 4 seções, sendo a primeira esta Introdução. A segunda seção apresenta a metodologia, destacando a base de dados de estudo, as ferramentas e técnicas usadas para o desenvolvimento deste trabalho. A terceira seção apresenta resultados experimentais e discussões, comparando os resultados alcançados com a literatura contemporânea e a quarta e última seção as conclusões obtidas.

2. Metodologia

A base de dados *PTB Diagnostic ECG Database*, ou simplesmente PTB, foi utilizada para a análise de arritmias cardíacas. Essa base de dados contém registros de eletrocardiogramas (ECG) de 290 pacientes. Neste estudo, ela foi dividida, de forma aleatória, em conjuntos de treinamento e teste em uma proporção de 80/20, ou seja, 80% para treino e 20% para teste.

Para estabelecer uma linha de base, foi implementada uma arquitetura de Perceptron. Para introduzir não-linearidades no modelo, uma rede Perceptron Multicamadas (MLP) ocultas foi implementada usando a função de ativação ReLU. Ao ajustar hiperparâmetros, como o número de camadas e o número de neurônios por camada, a arquitetura final da MLP será otimizada pela técnica de validação cruzada.

A linguagem de programação e ambiente principal para implementar e avaliar os modelos é baseado em Python, enquanto Pandas e NumPy serão usados para manipular dados e arrays. Foi utilizado *Deep Learning* e *Machine Learning* com TensorFlow 2.16.2, Keras, PyTorch 2.3.1 e SciKit-learn, e visualização de dados com Matplotlib e Seaborn. A análise de CNN aplicada a sinais de ECG em 2D será realizada usando as arquiteturas AlexNet e ResNet. O desenvolvimento dos modelos será baseado no Python 3.10.9.

Além da rede MLP, que é utilizada para classificar dados no formado 1D, foram utilizadas duas CNNs para dados 2D: ResNet-18 e AlexNet. A transformação de dados 1D em 2D foi realizada usando gráficos de recorrência.

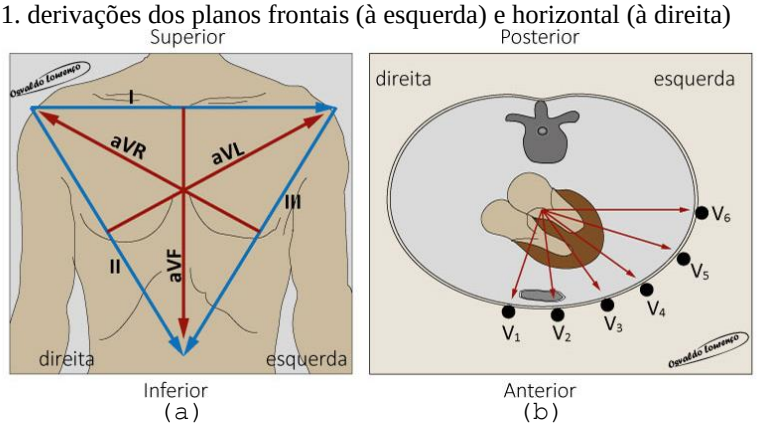
Métricas quantitativas e indicadores de desempenho de *hardware* são utilizados para avaliar os modelos. Mais detalhes metodológicos são apresentados a seguir.

2.1 Base de dados

A Associação para o Avanço da Instrumentalização Médica (AAMI), é uma organização, a qual foi fundada em 1967. Contando com a participação de mais de 10.000 profissionais que visam o desenvolvimento, gestão de utilização de uma saúde segura e eficaz. Essa Associação é a fonte principal de padrões, sendo reconhecidos nacional e internacionalmente para definir um consenso para a indústria de dispositivos médicos, assim como informações, suporte e orientações para os profissionais de tecnologia de saúde, sendo a maior líder global em seu desenvolvimento. A norma EC57 (ANSI/AAMI EC57…, 2020), define uma forma para realizar os testes e reportar os resultados obtidos com o objetivo de torná-los reprodutíveis e comparáveis (LUZ, 2012).

A base de dados PTB apresentada por Bousseljot (1995) e Goldberger et al. (2000) contém informações pré-processadas com 549 registros de 290 pessoas. Suas idades variam de 17 a 87 anos (idade média de 57,2 anos). Destes, 209 homens (idade média de 55,5 anos) e 81 mulheres (idade média de 61,6 anos) constituem a população. Para uma mulher e quatorze homens, as idades não foram informadas. Existem pessoas numeradas 124, 132, 134 ou 161, e cada uma é representada por uma a cinco gravações. As 12 derivações convencionais (I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6), conforme pode-se observar na Figura 1.

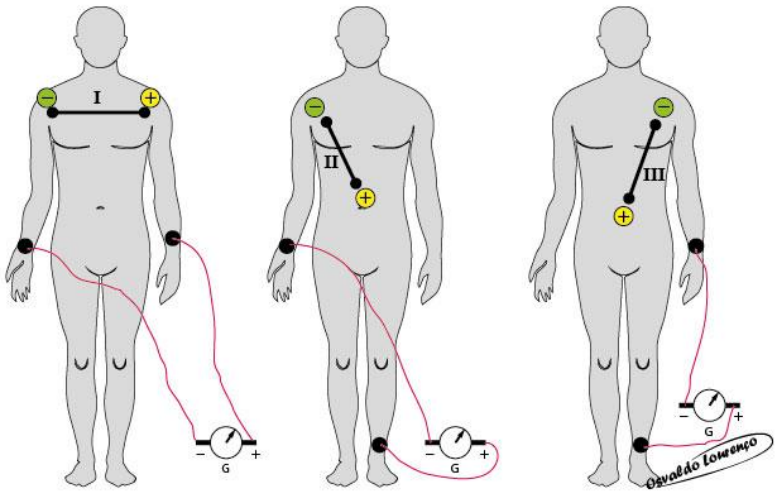
Figura 1. derivações dos planos frontais (à esquerda) e horizontal (à direita)



Fonte: ANGOMED (2014).

Neste estudo, foram utilizados o conjunto de dados PTB pré-processado por Kashuee, Fazeli e Sarrafzadeh (2018). Esta base consiste em registros de ECG de 290 indivíduos: 148 foram diagnosticados com infarto do miocárdio (IM), 52 eram controles saudáveis e o indivíduo restante foi diagnosticado com seis doenças diferentes. Cada registro inclui sinais de ECG de 12 diferenças, apresentados na frequência de 1.000 Hz. Apenas a derivação II, conforme observado na Figura 1.a, do ECG foi usada do estudo referenciado, levando em conta as categorias de IM e controles de altura para as análises. A derivação II tem o elétrodo negativo, na qual está conectado no braço direito e na perna esquerda fica conectado o elétrodo positivo.

Figura 2. Derivações Bipolares.



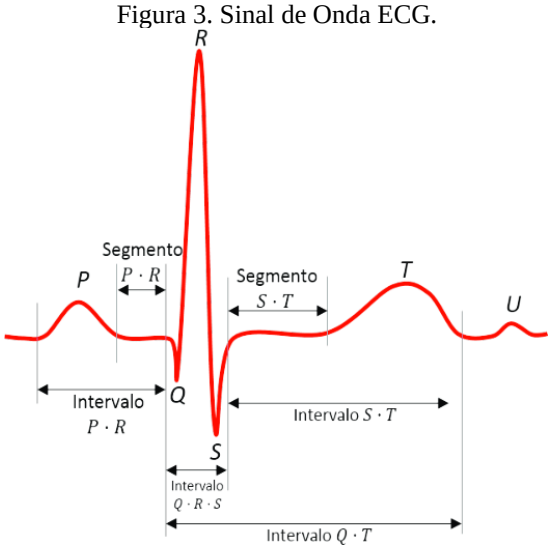
Fonte: ANGOMED (2014).

A derivação II, conforme a Figura 2, é um tipo de variação da MLII (*Modified Lead II*), uma derivação bipolar que se encontra paralelamente à derivação DII padrão. Quanto à sua posição anatômica, um eletrodo é colocado na crista ileal esquerda, enquanto o outro é posicionado na fossa infraclavicular, no meio do músculo deltóide, 2 cm abaixo da borda da clavícula direita (WOLF, 2004).

Na sua etapa de pré-processamento, foi proposto um método eficiente de pré-processamento e extração de batimentos cardíacos para preparar sinais de ECG para análise. Primeiro, dividimos o sinal de ECG contínuo em janelas de 10 segundos e escolhemos uma dessas janelas. A seguir, foi normalizado valores de amplitude para o

intervalo entre zero e um. Para obter os picos R do ECG, primeiro determinou-se as localizações máximas com base nas passagens por zero da primeira derivada e depois foi aplicado um limite de 0,9 aos valores normalizados dessas localizações máximas. O intervalo nominal de intervalos dessa janela (T) é determinado através da mediana dos intervalos R-R. Para obter um comprimento fixo e predeterminado, foi selecionado uma parte do sinal com comprimento igual a 1,2T para cada pico R e foi preenchido cada parte com zeros (KACHUEE, FAZELI, SARRAFZADEH, 2018), conforme mostrado na Figura 3.

Figura 3. Sinal de Onda ECG.



Fonte: AKULA (2019).

Também na Figura 3, o traçado de ECG é um sinal com um padrão recorrente que se decompõe em três ondas distintas: P, QRS e T, associadas a cada ansiedade. Medir as amplitudes e durações dessas ondas é necessário para reconhecer anomalias do ECG. A primeira onda de ECG, chamada onda P, é causada pela despolarização do átrio; a segunda onda, denominada onda QRS e composta por múltiplas deflexões, é causada pela despolarização dos ventrículos; e a última onda, chamada onda T, é causada pela repolarização dos ventrículos. Segmentos (segmento ST) e intervalos (intervalo PR e intervalo QT) são os componentes que não possuem essas ondas.

Os valores abaixo ou acima dos intervalos normais podem indicar várias anomalias e doenças cardíacas. As amplitudes, durações e morfologia dos diferentes componentes do ECG fornecem informações sobre o coração. Os dois domínios básicos dos sinais de ECG são o domínio do tempo e o domínio da frequência. É obter informações sobre as frequências envolvidas na possível formação do sinal ao longo do tempo através da representação temporal da atividade elétrica cardíaca por meio da representação no domínio do tempo. Por outro lado, as informações sobre os componentes de frequência envolvidos no sinal são fornecidas pela representação no domínio da frequência. Os especialistas em processamento de sinais são obrigados a dados contidos em ambos os domínios, portanto, eles se alternam entre si usando técnicas de transformação apropriadas (AKULA, 2019).

A Tabela 1 apresenta o quantitativo de dados por classe da base PTB utilizada neste estudo, sendo que cada registro contém um batimento rotulado com 187 posições cada, na frequência de 125 Hz, em duas classes (Anormal ‘A’ e Normal ‘N’), totalizando 14.552 registros, bem como a divisão 80/20 entre dados de treino e teste.

Tabela 1 – Quantitativo de dados da base de estudo.

	A	N	Total
Treino	8.404	3.237	11.641
Teste	2.102	809	2.911
Total	10.506	4.046	14.552

Fonte: ECG… (2018).

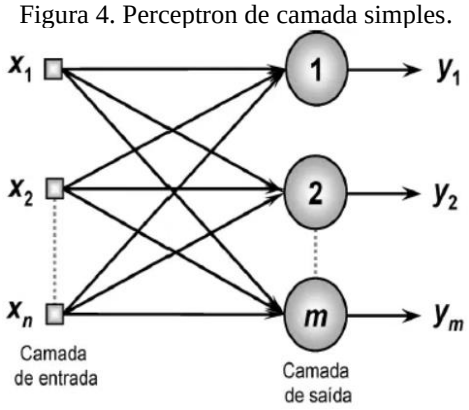
2.2 Rede Neurais Perceptron

Uma rede neural é principalmente um sistema destinado a reproduzir a maneira como o cérebro executa uma tarefa específica. Isso geralmente é feito usando componentes eletrônicos ou é replicado em um computador digital. As redes neurais usam unidades de processamento chamados "neurônios" (células computacionais simples) para trabalhar bem (HAYKIN, 2001).

As RNAs são frequentemente usadas para resolver problemas complexos em que o comportamento das variáveis não é totalmente conhecido. A capacidade de aprender por meio de exemplos e generalizar a informação aprendida cria um modelo não-linear, que torna sua aplicação bastante eficiente na análise espacial (SPÖRL et al., 2011).

A forma mais básica de rede neural artificial, o Perceptron, idealizado por Rosenblatt em 1958, é baseada na retina e é projetada para a percepção eletrônica de sinais, como a identificação de padrões geométricos. Sua simplicidade reside na estrutura de uma camada neural com apenas um neurônio. Apesar de ser uma rede simples, atraiu muitos pesquisadores interessados em IA e a comunidade científica se interessou por ela. O Perceptron é uma rede *feedforward* de uma única camada que não realimenta (VINICIUS, 2017).

Figura 4. Perceptron de camada simples.



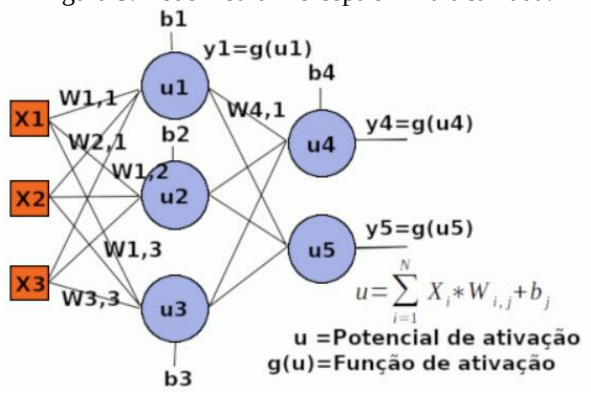
Fonte: VINICIUS (2017).

Sua principal aplicação é na classificação de padrões. São limitados os recursos do Perceptron de camada única à classificação de padrões linearmente separáveis. Na prática, o problema em questão não permite uma separação linear precisa. Portanto, o uso de um Perceptron multicamadas é necessário (AMBRÓSIO, 2002).

Os modelos neurais artificiais mais conhecidos são os do tipo MLP. Uma rede MLP tem três camadas. Eles são a camada de entrada, a camada intermediária ou escondida e a camada de saída (NIED, 2007). As entradas são prolongadas da camada de entrada para a camada de saída através de uma ou mais camadas ocultas na arquitetura da RNA de múltiplas camadas, de acordo com Machado e Fonseca Junior (2013).

Em outras palavras, um vetor de entrada é transmitido para a saída multiplicando-se pelos pesos de cada camada. Em seguida, uma função de ativação é aplicada (o modelo de cada neurônio da rede contém uma função de ativação não-linear, sendo a não-linearidade variável em qualquer ponto). Em seguida, este valor é transmitido para a camada seguinte até atingir a camada de saída (NIED, 2007). A Figura 5 representa visualmente o funcionamento de uma MLP.

Figura 5. Rede Neural Perceptron Multicamada.



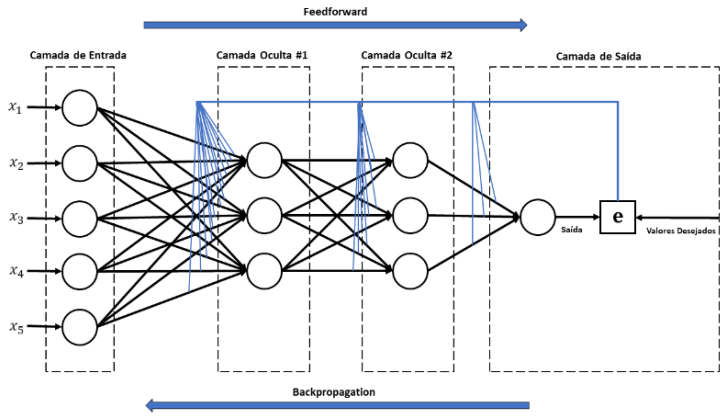
Fonte: HERBERT (2022).

Na Figura 5, X representam as entradas, W representa os pesos, u representa os potenciais de ativação, b representa os *biases*, g(u) representa as funções de ativação e y representa as saídas (HERBERT, 2022).

2.3 Cálculo de erro

O desenvolvimento do algoritmo de retropropagação por David Rumelhart em 1986 permitiu o treinamento de RNAs com várias camadas. Em geral, o processo é concluído em duas etapas. Os pesos sinápticos da rede recebem valores aleatórios e se movem em uma única direção através das camadas internas na primeira etapa, conhecida como *feedforward*, até que o sinal seja reproduzido na camada de saída. O aprendizado da rede funciona efetivamente segunda etapa. A saída desejada/esperada e a saída fornecida pela camada de saída da rede são usadas para encontrar a diferença entre os dois valores. A rede calcula o erro e propaga a correção para as camadas internas até a entrada, ajustando os pesos sinápticos (Figura 7). Isso é feito caso o resultado não esteja dentro de um padrão aceitável (FARIAS, 2021).

Figura 7. Arquitetura MLP e sua Retro-propagação de Erro.



Fonte: FARIAS (2021).

Em tarefas de classificação binária, como em *autoencoders*, a função de perda *Binary Cross Entropy* (BCELoss) é comumente usada para medir o erro entre a saída prevista por um modelo e o valor real (Pytorch, 2023). A equação 1 que mostra a função de perda BCELoss para um exemplo n é definida como:

$$L_n = -[y_n \,.\log(x_n) + (1 - y_n)\,.\,\log(1 - x_n)\,] \quad (1)$$

Onde y_n é o rotulo verdadeiro (0 ou 1) e x_n é a probabilidade prevista pelo modelo. Ao trabalhar com os lotes de dados (*batch size N*), existem duas maneiras de diminuir a perda total: pela média das perdas individuais do lote (o parâmetro de redução é chamado de "*mean*") ou pela soma das perdas (o parâmetro de redução é chamado de "*sum*").

Mas em alguns casos, como quando x_n é igual a 0 ou 1, a função logarítmica da BCELoss pode se tornar indefinida, resultando em $\log(0) = -\infty$. A condição em questão produz termos infinitos na função de perda. Isso é um problema porque a inclusão de um valor infinito no cálculo da perda implica a existência de um gradiente infinito durante a retropropagação. Isso dificultaria o processo de ajuste de pesos não linear, especialmente para modelos de regressão linear.

Para evitar esses problemas, a função BCELoss do PyTorch faz uma coisa útil: a saída da função logarítmica é limitada (ou pressionada) para um valor mínimo de -100. Assim, é possível evitar que a equação da perda produza termos infinitos. Isso garante que a função de perda permaneça finita e que o cálculo dos gradientes durante a retropropagação seja linear e estável. Isso é necessário para treinar bem modelos que usam essa função de perda, como em uma rede MLP PyTorch e Tensorflow.

A função de perda conhecida como entropia cruzada é usada para calcular o erro entre as previsões de uma rede neural e os alvos verdadeiros. É particularmente útil para problemas de classificação com várias classes. Os *logits* de entrada de uma tarefa que usa classes C são os valores não normalizados de cada classe; esses valores não precisam ser positivos ou somar 1. A função de perda espera que o alvo contenha probabilidades de cada classe, ou índices de classes no intervalo [0, C) (Pytorch, 2023). A equação para a entropia cruzada combinada com os índices de classe é:

$$l_n = -w_{y_n}\, log\, \frac{\exp\,(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})}\,.\,1\{y_n \neq ignore\_index\} \quad (2)$$

Nessa equação 2, x_{n,p_n} representa o *logit* que foi prevista para o modelo y_n, enquanto $\sum_{c=1}^{C} \exp\,(x_{n,c})$ é a soma dos exponenciais dos *logits* que foram previstos para todas as classes *C.* A probabilidade prevista para a classe verdadeira y_n é calculada pela razão entre esses dois termos. O termo $\log \frac{\exp\,(x_{n,p_n})}{\sum_{c=1}^{C} \exp\,(x_{n,c})}$ aplica o logaritmo dessa probabilidade (Pytorch, 2023).

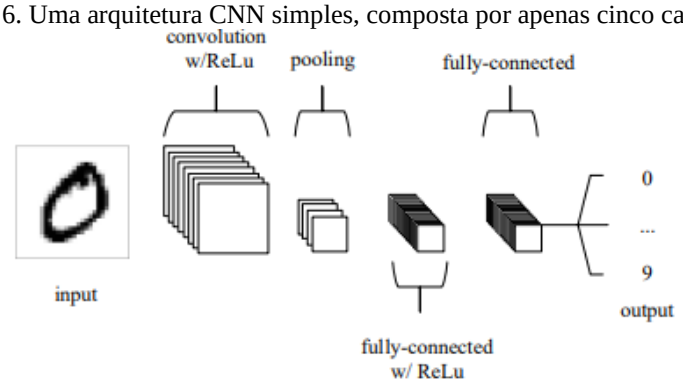
Além disso, w_{y_n} é um peso atribuído à classe verdadeira. Isso permite que várias classes tenham maior ou menor importância, o que é especialmente útil quando as classes não estão equilibradas. Caso seja necessário, rótulos específicos são ignorados durante o cálculo da perda por meio da função indicadora $1\{y_n \neq ignore\_index\}$. Por exemplo, isso é útil para tarefas em que os rótulos específicos não devem afetar o treinamento do modelo (Pytorch, 2023).

2.4 Rede Neurais Convolucionais

As *Convolutional Neural Networks* (CNNs) foram desenvolvidas para processar imagens. Essa arquitetura foi projetada para lidar com esse tipo de dados. Os neurônios das camadas de uma CNN diferem em sua organização em três dimensões (altura, largura e profundidade). A profundidade é uma dimensão terceira do volume de ativação, não o número total de camadas. Os neurônios de uma camada em uma CNN se conectam a uma pequena porção da camada anterior. Por exemplo, uma entrada com dimensões 64x64x3 (altura, largura e profundidade) dá como resultado uma camada de saída com dimensões 1x1xn (onde n é o número de classes possíveis). Isso condensa a dimensão total em um volume menor de pontuações de classe distribuídos pela dimensão de profundidade (O'SHEA, 2015).

As CNNs têm três tipos de camadas: camadas convolucionais, camadas de *pooling* (amostragem) e camadas totalmente conectadas. A arquitetura de uma CNN surge quando essas camadas são combinadas.

Figura 6. Uma arquitetura CNN simples, composta por apenas cinco camadas.



Fonte: O'SHEA (2015).

Os valores dos *pixels* da imagem são armazenados na camada de entrada, que é comparável a outras variedades de redes neurais artificiais (ANN). A camada convolucional determina a saída dos neurônios conectados a regiões locais de entrada por meio do cálculo do produto escalar entre seus pesos e a região conectada ao volume de entrada. A unidade linear retificada (ReLU) adiciona uma função de ativação elementar, como a sigmoide, à saída da ativação criada pela camada anterior. A camada de *pooling* faz amostragem ao longo da dimensionalidade espacial da entrada, reduzindo o número de parâmetros dentro dessa ativação. As camadas totalmente conectadas fazem funções semelhantes às das ANNs comuns, tentando produzir escores de classe a partir das ativações para classificação. Para melhorar o desempenho entre essas camadas, é recomendado o uso do ReLU (O'SHEA, 2015).

2.5 AlexNet

O AlexNet é uma das aplicações de redes neurais convolucionais profundas pioneiras que realmente transformaram o campo do aprendizado de máquina. Este modelo inventivo ganhou o primeiro lugar no desafio ImageNet LSVRC-2012 em 2012 com uma precisão impressionante de 84,7%. Isso superou significativamente o segundo colocado, que teve uma precisão de 73,8% (SINGH, 2023).

O ImageNet é uma coleção de mais de 15 milhões de imagens de alta resolução organizadas em cerca de 22.000 categorias. Usando a ferramenta de *Crowdsourcing*

Amazon Mechanical Turk, os indivíduos rotularam as imagens que foram coletadas da internet. Um subconjunto do ImageNet foi usado no desafio anual *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), que faz parte do *Pascal Visual Object Challenge*, a partir de 2010 (KRIZHEVSKY, 2017).

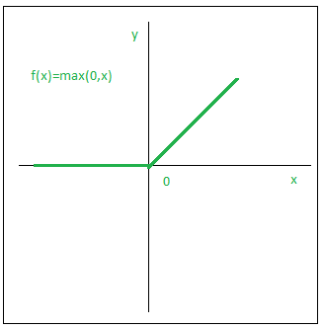
Cada uma das 1.000 categorias do subconjunto ILSVRC contém aproximadamente 1.000 imagens; isso resulta em aproximadamente 1,2 milhões de imagens de treinamento, 50.000 imagens de validação e 150.000 imagens de teste. Embora o conjunto completo do ImageNet tenha muitas categorias, o ILSVRC se concentra em um número menor de categorias, o que facilita a competição (KRIZHEVSKY, 2017).

O ImageNet desempenhou um papel importante no desenvolvimento de modelos de aprendizado de máquina, particularmente redes neurais convolucionais profundas (CNNs), como o AlexNet. Por exemplo, a arquitetura do AlexNet foi treinada neste conjunto de dados e conseguiu vencer o ILSVRC-2012, demonstrando seu poder de modelos com grande capacidade de aprendizado (KRIZHEVSKY, 2017).

A arquitetura do AlexNet é composta por cinco camadas convolucionais (CONV) e três camadas totalmente conectadas (FC). A decisão de AlexNet de usar a função de ativação ReLU (Rectified Linear Unit) é uma decisão estratégica que teve um impacto significativo em seu desempenho excepcional (SINGH, 2023).

A ReLU (figura 8) é conhecida por acelerar o processo de treinamento ao resolver o problema do gradiente desvanecido, que é comum em funções de ativação como *Tanh* e *Sigmoid*. A fórmula para esta função de ativação simples, mas eficaz é $f(x)=max(0, x)$. Onde $f(x)$ representa a saída da função e ativação; x representa a entrada para a função de ativação, podendo ser qualquer valor real; e $max(0, x)$, que é a operação matemática que retorna x se x for maior que 0, caso contrário, retorna 0. Isso permite treinamento mais rápido e eficiente (NAIR et al, 2010).

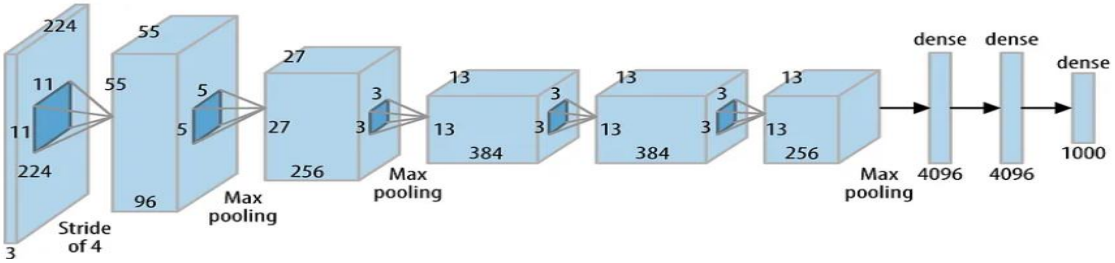
Figura 8. Função ReLu.



Fonte: JOSHI (2021).

Além disso, para melhorar a estabilidade da rede em geral, AlexNet usa a Normalização de Resposta Local (LRN) para equilibrar a ativação dos neurônios. A rede recebe imagens com dimensões de 224 x 224 x 3 (largura, altura e profundidade) e cria um vetor de probabilidade 1000 x 1 com a probabilidade de que cada imagem pertença a uma das 1000 classes, como mostrado na Figura 9 (SINGH, 2023).

Figura 9. Arquitetura de uma rede AlexNet.



Fonte: SINGH (2023).

2.6 ResNet

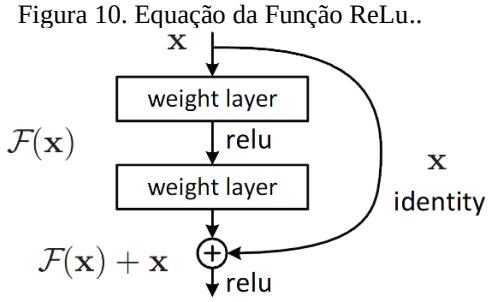
Uma rede neural convolucional da Microsoft chamada ResNet ganhou o concurso ILSRVC de 2015 no conjunto de dados ImageNet. A eficácia dessa rede superou a dos seres humanos.

O problema da degradação pode surgir quando redes mais profundas começam a convergir. A precisão da rede diminui rapidamente à medida que a profundidade aumenta. Além disso, o *overfitting* não causa essa degradação, e adicionar mais camadas ao modelo profundo aumentaria o erro de treinamento (SILVA, 2018).

A ResNet é composta por blocos residuais. Cada bloco recebe uma entrada x, que é processada por uma série de operações convolucionais antes da ReLU e outra convolução. O resultado dessa operação, chamado f(x), é então adicionado à entrada inicial x. O mapeamento residual que resulta dessa soma permite que o modelo aprenda a função residual f(x) em vez da função direta. Além disso, as camadas do ResNet replicam aquelas do modelo mais baixo aprendido. Isso sugere que um modelo mais avançado não deve ter um erro de treinamento maior do que sua versão menos avançada (HE et al., 2016).

O resultado do processamento H(x) altera significativamente o espaço de saída em relação ao espaço de entrada x, de acordo com a equação (H(x) = f(x) + x). Por outro lado, a função f(x) funciona na ResNet apenas como uma regularização, o que leva a uma mudança no espaço de entrada (figura 10). Assim, como resultado dessa modificação no espaço de entrada, a otimização de mapas residuais é mais simples com a ResNet (HE et al., 2016).

Figura 10. Equação da Função ReLu..



Fonte: HE et al. (2015).

2.7 Transformação 1D em 2D

A descrição matemática aqui apresentada é baseada em Casdagli (1997). Um gráfico de recorrência é uma imagem obtida de uma série temporal, representando as

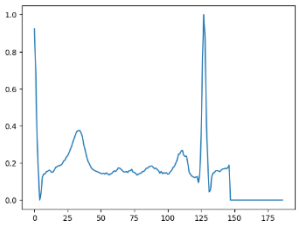
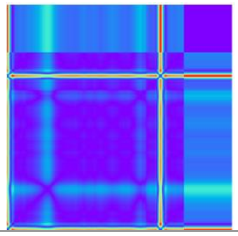
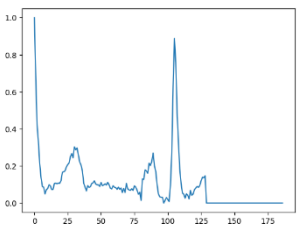
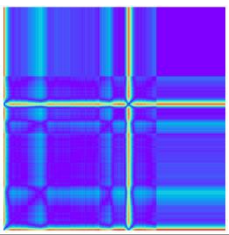
distâncias euclidianas pareadas para cada valor (e mais geralmente para cada trajetória) na série temporal. Ou seja, mostra quando um estado do sistema ($\overrightarrow{x_i}$) visita a mesma região de espaço de fase ($\overrightarrow{x_j}$), ou seja, quando $\overrightarrow{x_i} \approx \overrightarrow{x_j}$, onde i e j indexam tempos distintos dentro da mesma série temporal.

Em outras palavras, a recorrência acontece quando um estado $\overrightarrow{x_i}$ do sistema passa em uma vizinhança do estado $\overrightarrow{x_j}$, onde o estado de $\overrightarrow{x_i}$ está em um tempo $t = i.\Delta t$ e $\overrightarrow{x_j}$ em um tempo $t = j.\Delta t$, sendo Δt a taxa de amostragem. Matematicamente o Gráfico de Recorrência é a representação da matriz quadrada N x N dada por:

$$R_{ij}(\varepsilon) = \Theta(\varepsilon - \|\overrightarrow{x_i} - \overrightarrow{x_j}\|) \quad (3)$$

onde $\overrightarrow{x_i}$ e $\overrightarrow{x_j}$, com $i,j = 1, \ldots, N$, são vetores no espaço de fase, representando os estados do sistema, e ε é o limiar (threshold), ou seja, é o limite de tolerância dentro do qual constam-se as coordenadas no espaço de fase como pontos de recorrência, ou ainda, o raio em torno de um ponto da trajetória dentro do qual pontos vizinhos são considerados recorrentes. A função Θ é a de Heaviside, que transforma efetivamente pares de coordenadas em 1 (pontos recorrentes), se a distância entre esses pontos da trajetória for inferior ao limiar ε, e em 0 (pontos não recorrentes) caso contrário; e $\|.\|$ é a norma, sendo mais usadas a norma mínima, a máxima e a euclidiana. A Figura 11 apresenta os gráficos de recorrência para um registro de cada classe, sendo $\varepsilon = 0.1$ e mapa de cores *rainbow*. Assim, cada registro foi transformado numa imagem, para servir de entrada para a CNN.

Figura 11 – Gráficos de recorrência para um registro de cada classe.

	Série Temporal	**Gráfico de Recorrência**
Classe "N" (Normal)		
Classe "A" (Anormal)		

Fonte: Próprio autor (2024).

2.8 Indicadores de desempenho

A norma ANSI/AAMI EC57:2012 R2020 (ANSI/AAMI, 2020) também recomenda métricas de avaliação dos modelos de classificação automática para comparar os resultados de diferentes estudos, sendo elas a acurácia (Acc), precisão ou acertos positivos (Prec), sensibilidade (Recall) e especificidade (Spe). Adicionalmente, também é utilizada a média harmônica entre precisão e sensibilidade (f1-score). Além destes, também são utilizados indicadores de performance computacional para avaliação de

desempenho de uso de recursos de hardware, os quais são o tempo médio por época de processamento e o consumo de memória para treinamento.

A acurácia representa a taxa de acerto global de um modelo de classificação, que é a proporção de suposições corretas em relação ao tamanho total do conjunto de dados. Em outras palavras, ela avalia a capacidade do modelo de prever corretamente tanto os casos positivos quanto os negativos, fornecendo um quadro do desempenho do modelo (SILVA, Renato et al., 2012).

A proporção de padrões da classe positiva que são identificados corretamente é conhecida como sensitividade, também conhecida como recall. Essa métrica mede a capacidade do classificador de identificar corretamente casos positivos. Isso mostra o quão bem o modelo funciona para encontrar a classe positiva (SILVA, Renato et al., 2012).

Por outro lado, a especificidade calcula a quantidade de padrões da classe negativa que foram corretamente identificados. Ela avalia a eficiência do classificador na identificação de casos negativos, refletindo sua eficácia na diferenciação da classe negativa (SILVA, Renato et al., 2012).

A precisão, também conhecida como precisão, é a porcentagem de padrões que foram classificados como pertencentes à classe positiva que realmente pertencem à classe. Ao identificar a classe positiva e reduzir o número de falsos positivos, essa métrica é crucial para avaliar a confiabilidade do modelo (SILVA, Renato et al., 2012).

3. Resultados e Discussões

3.1 Ambiente computacional

Os testes foram realizados utilizando um processador AMD Ryzen 5 5600G, com vídeo integrado Radeon Graphics, 16 GB de RAM, 128GB de HD SSD. Para as versões das ferramentas, foram utilizadas para testes Python na sua versão 3.10.9, Pytorch 2.3.1+cpu, Tensorflow 2.16.2.

3.2 Modelos de redes neurais

A arquitetura de a AlexNet utilizada neste estudo é descrita na Tabela 2, que incluem detalhes sobre cada camada, bem como tamanhos de saída, tamanhos de *kernel*, preenchimento (*padding*) e comprimento. Observa-se que a camada de classificação foi adaptada para 2 classes de saída, diminuindo-se a quantidade de neurônios, pois sua proposta original foi projetada para classificar 1.000 classes.

Tabela 2 – Parâmetros estruturais da rede AlexNet.

Layer Name	Output Size	Kernel Size	Padding	Strid
Input 3x224x224				
Conv2D	3x64	11x11	2x2	4x4
ReLu				
MaxPool2D		3	0	2
Conv2D	64x192	5x5	2x2	1x1
ReLu				
MaxPool2D		3	0	2
Conv2D	192x384	3x3	1x1	1x1

ReLu				
Conv2D	184x256	3x3	1x1	1x1
ReLu				
Conv2D	256x256	3x3	1x1	1x1
ReLu				
MaxPool2D		3	0	2
Avgpool:				
AdaptiveAvgPool2d	6x6			
Classifier:				
Dropout(0.5)				
Linear	1024			
ReLu				
Dropout(0.5)				
Linear	2			

Fonte: Adaptado de Krizhevsky (2014).

As camadas convolucionais (Conv2D) são essenciais para a aplicação de filtros (*kernels*) à imagem de entrada para extrair características cruciais. Uma função de ativação ReLU segue essas camadas convolucionais e adiciona não-linearidade ao modelo, permitindo que ele aprenda padrões mais complexos. A função ReLU ajuda o modelo a capturar e representar relações não-lineares nos dados.

As características extraídas pelas camadas convolucionais são dimensionadas com o uso de camadas de *pooling* (*MaxPool2D*). Isso reduz os custos computacionais e controla o overfitting. O *Max Pooling* escolhe os valores máximos dentro de uma janela específica de tamanho (3 por 3), mantendo as características mais importantes enquanto reduz a resolução espacial da entrada. Esse processo de agregação é fundamental para aumentar a eficiência do modelo ao mesmo tempo em que mantém a integridade dos dados extraídos.

A saída *AdaptiveAvgPool2d* é ajustada para um tamanho fixo (6x6), independentemente do tamanho da entrada. Essa característica é particularmente útil na preparação da saída das camadas convolucionais para uso na camada de classificação. Essas camadas garantem que a entrada da camada de classificação tenha um formato consistente, facilitando o processamento posterior, ajustando a saída para um tamanho predeterminado.

Na arquitetura, o classificador inclui camadas totalmente conectadas (lineares) com *dropout* intercaladas para evitar *overfitting*. Uma saída de 1.024 unidades está presente na primeira camada linear. Uma função de ativação ReLU e uma camada de saída com probabilidade de 0,5 estão presentes na segunda camada linear. Isso é repetido, e a saída da segunda camada linear é de 2 unidades. Uma maneira eficaz de evitar que o modelo se ajuste demais aos dados de treinamento é usar o *dropout* entre as camadas lineares. Isso melhora a capacidade de generalização do modelo.

Outra CNN utilizada neste estudo é a ResNet18, apresentada na Tabela 3, proposta por He et al. (2015). Nesta rede, a primeira camada convolucional (Conv2D) aplica um filtro de 7x7 com *padding* de 3x3 e *stride* de 2x2, resultando em uma saída de 3x64. Isso facilita a captura de características de baixo nível na imagem. Após isso, várias camadas convolucionais adicionais são organizadas em blocos, também conhecidos como *layers*. Cada bloco tem duas camadas Conv2D com filtros de 3 por 3, *padding* de 1 por 1 e *stride* de 1 por 1. Para introduzir não-linearidade, as funções de normalização de *batch* (bn1, bn2) e ReLU são usadas.

Tabela 3 – Parâmetros estruturais da rede ResNet18.

Layer Name	**Output Size**	**Kernel Size**	**Padding**	**Strid**
Input 3x224x224				
Conv2d	3x64	7x7	3x3	2x2
bn1	64			
ReLu				
MaxPool2D		3	1	2
Layer 1:				
Conv2D	64x64	3x3	1x1	1x1
bn1	64			
ReLu				
Conv2D	64x64	3x3	1x1	1x1
bn2	64			
Layer 2:				
Conv2D	64x64	3x3	1x1	1x1
bn1	64			
ReLu				
Conv2D	64x64	3x3	1x1	1x1
bn2	64			
Layer 3:				
Conv2D	64x128	3x3	1x1	2x2
bn1	128			
ReLu				
Conv2D	128x128	3x3	1x1	1x1
bn2	128			
Layer 4:				
Conv2D	128x256	3x3	1x1	2x2
bn1	256			
ReLu				
Conv2D	256x256	3x3	1x1	1x1
bn2	256			
Layer 5:				
Conv2D	256x512	3x3	1x1	2x2
bn1	512			
ReLu				
Conv2D	512x512	3x3	1x1	1x1
bn2	512			
Avgpool				
AdaptiveAvgPool2d	1x1			
Linear	2			

Fonte: Adaptado de HE et al. (2015).

A dimensão das características extraídas pelas camadas convolucionais é reduzida pelas camadas de *pooling*. O MaxPool2D, a primeira camada de *pooling*, tem um *kernel* de 3 com *padding* de 1 e *stride* de 2, o que ajuda a reduzir a resolução espacial das características, controlar o *overfitting* e diminuir o custo computacional.

Os blocos de camadas são organizados em cinco “camadas” separadas, cada uma das quais aumenta a profundidade da rede e a complexidade das características extraídas. As saídas das camadas Conv2D no nível 1 são de 64x64, enquanto as saídas do nível 3 aumentam para 64x128 com um stride de 2x2, indicando uma redução mais agressiva da dimensão espacial. Os filtros convolucionais aumentam em número à medida que a rede se aprofunda (128x256 no *Layer* 4 e 256x512 no *Layer* 5), permitindo a captura de características mais complexas e de alto nível.

A camada de *pooling* adaptativo (AdaptiveAvgPool2d) ajusta a saída para um tamanho predeterminado de 1x1, independentemente do tamanho da entrada. A padronização da entrada da camada de classificação garante que o tamanho da entrada da camada totalmente conectada seja sempre consistente.

Nessa arquitetura, a camada linear final é composta por uma única camada densa totalmente conectada, o que reduz a saída para 2 unidades, correspondendo ao número de classes a serem classificadas, sendo também uma adaptação ao modelo original, projetado para 1.000 classes. O classificador é simples e direto para a tarefa de classificação final porque não menciona *dropouts* ou outras camadas intermediárias.

A inclusão de camadas de normalização em série (bn1, bn2) após as camadas convolucionais, que não são mencionadas na Tabela 2, é uma característica distintiva desta arquitetura. A normalização de *batch* acelera o treinamento e aumenta a estabilidade do modelo.

A configuração de uma rede neural *Multi-Layer Perceptron* (MLP) implementada com PyTorch é descrita na Tabela 4. A rede tem duas camadas totalmente conectadas, também conhecidas “fc”, cada uma das quais desempenha uma função específica na transformação e aprendizado de dados de entrada. O primeiro conjunto de neurônios é conectado à entrada pela camada fc1, que recebe cada registro de série temporal contendo 187 pontos amostrais que representam um sinal ECG. Em seguida uma camada de 100 neurônios numa camada escondida. Este valor foi encontrado empiricamente, apresentando os melhores indicadores para esta arquitetura. Em seguida camada é fc2, que conecta esta camada escondida à saída com um neurônio para uma classificação binária.

Tabela 4 – Parâmetros estruturais da rede MLP.

Layer name	**Input/Output Size**
Linear (fc1)	187/100
Linear (fc2)	100/1
ReLu	1/1
Sigmoid	1/1

Fonte: Próprio autor (2024).

Uma camada linear, ou totalmente conectada, é a primeira camada da rede e possui uma saída de 100 neurônios. Os pesos e vieses que foram ajustados durante o processo de treinamento são usados nesta camada para aplicar uma transformação linear aos dados de entrada. Essa camada serve principalmente para projetar os dados de entrada em um espaço de maior dimensão, o que permite a captura de relações complexas entre as características dos dados.

A função de ativação ReLU (*Rectified Linear Unit*) é usada pela rede após a aplicação da primeira camada linear. A ReLU é uma função não-linear que zera todos os valores negativos da saída da camada anterior, mas mantém os valores positivos inalterados. Como resultado desta ativação, a rede ganha não-linearidade, o que lhe permite aprender padrões de dados mais complexos e abrangentes. Além disso, a ReLU ajuda a reduzir o problema do desaparecimento do gradiente, que é comum em redes profundas.

A saída de 100 neurônios da camada anterior é recebida pela segunda camada linear da rede e transformada em uma nova saída de 1 neurônio. Esta camada linear usa um novo conjunto de vieses e pesos para aplicar uma transformação linear, semelhante à

primeira camada linear. Esta camada tem como objetivo melhorar ainda mais a representação dos dados, diminuindo a dimensionalidade enquanto mantém as características mais pertinentes.

A função de ativação *Sigmoid* finalmente transmite a saída da segunda camada linear. A função *Sigmoid* é particularmente útil em tarefas de classificação binária, onde a saída pode ser interpretada como uma probabilidade. Ela mapeia a saída para um intervalo de 0 a 1. Essa ativação facilita a tomada de decisões finais do modelo, transformando a soma ponderada das entradas em valores probabilísticos.

3.3 Indicadores de desempenho

Neste estudo, as redes neurais apresentadas não utilizados pesos pré-treinados, sendo tanto as camadas convolucionais quanto as fortemente conectadas trabalhadas com pesos zerados e atualizados durante o aprendizado. Diversos treinamentos foram conduzidos e os resultados aqui apresentados foram os melhores observados.

O tempo médio por época para vários modelos treinados nos *frameworks* TensorFlow e PyTorch está na Tabela 5. Esta duração é uma medida significativa da eficiência computacional no treinamento de modelos de aprendizado de máquina e mostra o desempenho dos *frameworks* em termos de velocidade.

Tabela 5 – Tempo Médio por Época.

Modelo	Tempo médio por época (minutos)
MLP Tensorflow	0,09
MLP Pytorch	3,40
ResNet18 Tensorflow	20,26
ResNet18 Pytorch	24,32
AlexNet Tensorflow	7,23
AlexNet Pytorch	15,19

Fonte: Próprio autor (2024).

O tempo por época médio do *Multilayer Perceptron* (MLP) no TensorFlow é de apenas 0,09 segundos, enquanto o de PyTorch é significativamente maior, com 3,40 minutos. Essa discrepância pode ser atribuída às várias implementações e otimizações de cada *framework*, particularmente quando se trata de modelos mais simples como o MLP. Possivelmente devido às técnicas de otimização mais avançadas para tarefas básicas, o TensorFlow funciona muito melhor.

O TensorFlow apresenta 20,26 minutos por época média para o modelo ResNet18, enquanto o PyTorch apresenta 24,32 minutos. Os intervalos de tempo mais longos mostram a complexidade intrínseca do modelo ResNet18. No entanto, o TensorFlow mostra uma pequena vantagem em termos de eficiência, indicando que seu gerenciamento de recursos e otimizações internas são um pouco melhores para este tipo de arquitetura de rede neural.

O tempo médio por época do TensorFlow é de 7,23 minutos, enquanto o de PyTorch é de 15,19 minutos, ao levar em consideração o modelo AlexNet. O TensorFlow é, mais uma vez, mais rápido para este modelo específico, o que contribui para esta diferença significativa. A implementação mais eficaz e o melhor uso dos recursos de hardware disponíveis podem resultar na superioridade da velocidade de treinamento do TensorFlow sobre o AlexNet.

Detalhes dos parâmetros de configuração utilizados para treinar os modelos nos *frameworks* TensorFlow e PyTorch pode ser encontrada na Tabela 6. A compreensão das condições em que os modelos foram treinados e a realização de comparações entre várias configurações depende desses parâmetros. É importante observar que este trabalho buscou encontrar os melhores indicadores com estas ferramentas, mas não foi possível utilizar as mesmas configurações em todas as arquiteturas em cada *framework*. Em Tensorflow, foi observado um alto consumo de memória, sendo impossível utilizar um tamanho de lote de 128, que foi o melhor valor encontrado para os resultados em Pytorch. Nesse sentido, optou-se por reduzir o lote em Tensorflow para o máximo possível de processar com o *hardware* disponível, chegando-se ao valor de 32, que obviamente impacta nos indicadores, mas se apresenta como uma desvantagem a ser ponderada na busca de melhores resultados para este problema.

Tabela 6 – Parâmetros de Configuração.

Modelos	Optimizer	Image Resolution	Learning Rate	Batch Size	Epochs	Drop Period	Drop Learning Rate
MLP Tensorflow	adam	1x187	1e-3	32	300	11	0.5
MLP Pytorch	adam	1x187	1e-3	32	300	11	0.5
ResNet18 Tensorflow	adam	3x224x224	1e-4	32	100	11	0.5
ResNet18 Pytorch	adam	3x224x224	1e-4	128	100	11	0.5
AlexNet Tensorflow	adam	3x224x224	1e-4	32	100	11	0.5
AlexNet Pytorch	adam	3x224x224	1e-4	128	100	11	0.5

Fonte: Próprio autor (2024).

O otimizador Adam apresentou melhores resultados em todos os modelos implementados, com uma taxa de aprendizado inicial de 1e-3 para os modelos em MLP e 1e-4 para CNNs, tamanho de *batch* de 32 e 128 apresentaram melhores resultados de acordo com os modelos apresentados, e 300 épocas de treinamento foram colocadas como condição final de treinamento. Um detalhe importante é que para as CNN em Tensorflow, não foi possível trabalhar com 128 de tamanho de batch, pois o consumo de memória nesta condição era superior aos 16 GB de RAM disponível, evidenciando maior consumo de recursos computacionais para este *framework*.

A imagem tinha uma resolução de 224x224x3 e 100 épocas de treinamento. A principal diferença entre os dois é o tamanho do lote: TensorFlow usa 32 e PyTorch 128. Se o modelo ou o hardware não forem otimizados para lidar com batches grandes, um batch maior no PyTorch pode levar a um uso mais eficiente das GPUs. No entanto, isso também pode aumentar o tempo por época. Este ajuste pode afetar a velocidade e a eficiência do treinamento.

Em ambos os *frameworks*, o AlexNet foi treinado usando o otimizador Adam, uma taxa de aprendizado de 1e-4, uma resolução de imagem de 224x224x3 e 100 épocas de treinamento, de forma semelhante ao ResNet18. O tamanho do lote também é diferente: 32 para TensorFlow e 128 para PyTorch. Esta configuração mostra uma abordagem contínua para otimização do treinamento para modelos complexos, que pode se concentrar na eficiência do uso do hardware.

Os parâmetros de configuração, principalmente o tamanho do *batch*, podem ter um impacto significativo nas diferenças de tempo médio por época entre TensorFlow e PyTorch. Como visto no PyTorch para ResNet18 e AlexNet, um tamanho de *batch* maior pode aumentar a utilização de RAM, o que resulta em treinamento mais rápido por iteração de *batch*, mas seu tamanho deve ser testado em diferentes configurações para buscar que apresente melhor resultado.

Ao analisar os resultados da Tabela 7, observa-se que ambos os modelos MLP apresentaram bom desempenho, mas eles ficaram atrás dos modelos ResNet18 e AlexNet em termos de precisão, acurácia, recall e F1-Score. Apesar de ser um bom resultado, o modelo MLP treinado com TensorFlow teve a menor acurácia de 97,11% entre os modelos testados. As métricas de precisão (96,50%), recall (96,29%) e f1-Score (96,39%) mostram que o modelo é relativamente equilibrado, embora não seja tão alto quanto os modelos convolucionais.

Tabela 7 - Indicadores de desempenho da classificação em cada modelo e framework.

Modelos	Acurácia (%)	Precisão (%)	Recall (%)	F1-Score (%)
MLP Tensorflow	97,11	96,50	96,29	96,39
MLP Pytorch	97,76	97,13	97,31	97,22
ResNet18 Tensorflow	99,10	98,92	98,84	98,88
ResNet18 Pytorch	99,48	99,48	99,22	99,35
AlexNet Tensorflow	98,72	98,84	97,97	98,40
AlexNet Pytorch	99,55	99,49	99,38	99,44

Fonte: Próprio autor (2024).

Além disso, o MLP treinado com PyTorch demonstrou uma ligeira melhora na acurácia, de 97,76%, bem como uma ligeira melhora nas métricas de precisão (97,13%), recall (97,31%) e f1-Score (97,22%).

Os modelos ResNet18 tiveram bons desempenhos em ambas as bibliotecas. O modelo ResNet18 treinado em Tensorflow teve uma precisão de 99,10%. Além disso, todas as métricas de precisão, recall e F1-Score estão muito próximas de suas pontuações máximas. Portanto, o desempenho dele foi excelente em termos de generalização e erro de minimização. O modelo treinado com PyTorch teve um desempenho um pouco melhor que o Tensorflow, com uma precisão de 99,48%. O modelo treinado com o PyTorch teve o melhor desempenho entre todos os modelos analisados e mostra o quão bom são as técnicas de aprendizado profundo do modulo implementado na biblioteca PyTorch para a arquitetura ResNet18.

O modelo AlexNet treinado com Tensorflow atingiu uma taxa de precisão de 98,72%, e suas métricas de precisão, recall e f1-Score também foram muito boas, demonstrando a robustez do modelo. O modelo treinado com PyTorch alcançou uma precisão de 99,55%, superando o Tensorflow. As métricas relacionadas também são notáveis, com um F1-Score de 99.44%, indicando o melhor desempenho nas tarefas de classificação, incluindo no consumo de recursos computacionais.

Os resultados da Tabela 8 sobre o modelo MLP treinado com o Tensorflow mostraram uma acurácia de 98,14%, precisão de 97,86%, recall de 98,14% e F1-Score de 98,00% para a Classe A. Esses valores mostram que as métricas estão bem equilibradas e com um bom desempenho. Além disso, os resultados da Classe N mostraram uma taxa de precisão de 95,14%, uma taxa de recall de 94,43% e uma taxa de F1-Score de 94,78%. Embora tenha um bom desempenho, é menos preciso do que a Classe A, mostrando uma pequena diferença no comportamento entre as classes.

Tabela 8 - Indicadores de desempenho por classe.

	Classe A				Classe N			
Modelos	Acurácia (%)	Precisão (%)	Recall (%)	F1-Score (%)	Acurácia (%)	Precisão (%)	Recall (%)	F1-Score (%)
MLP Tensorflow	98.14	97.86	98.14	98.00	94.43	95.14	94.43	94.78
MLP Pytorch	98.33	98.56	98.33	98.45	96.29	95.70	96.29	95.99
ResNet18 Tensorflow	99,42	.99,33	99,42	99,38	98,26	98,51	98,26	98,39
ResNet18 Pytorch	98.64	99.50	98.64	99.06	99.80	99.47	99.80	99.64
AlexNet Tensorflow	99.66	98.58	99.66	99.12	96.29	99.10	96.29	97.68
AlexNet Pytorch	99.01	99.37	99.01	99.19	99.76	99.61	99.76	99.69

Fonte: Próprio autor (2024).

O desempenho dos *frameworks* TensorFlow e PyTorch é notável no modelo ResNet18. Para a classe A, os resultados de acurácia de 99,42%, precisão de 99,33%, recall de 99,42% e pontuação F1-Score de 99,38% são indicativos de uma excelente capacidade de generalização e minimização de erros no TensorFlow. Os resultados para a classe N foram um pouco menores. Eles tiveram uma precisão de 98,51%, uma acurácia de 98,26%, um recall de 98,26% e uma F1-Score de 98,39%, mas ainda tiveram um desempenho sólido e consistente. Por outro lado, a classe A do PyTorch mostra uma precisão de 99,50%, uma acurácia de 98,64% e um recall de 98,9%.

Os *frameworks* TensorFlow e PyTorch têm características distintas para o modelo AlexNet. Em termos práticos, os modelos em Tensorflow consomem muito mais memória que os modelos em PyTorch, exigindo adequação nos parâmetros de configuração para busca de melhores desempenhos, enquanto o modelo em PyTorch permite melhor flexibilidade na edição de parâmetros por consumir menos recursos de *hardware*.

A classe “A” mostra um desempenho notável do modelo no TensorFlow com uma taxa de precisão de 99,58%, recall de 99,66% e F1-Score de 99,12%. Os resultados na classe N foram um pouco menores. Eles tiveram uma F1-Score de 97,68%, uma precisão de 99,10%, uma acurácia de 96,29% e um recall de 96,29%. Esses resultados mostram boas métricas, mas um desempenho um pouco menor no *recall*. Por outro lado, os dados mostram que a classe A do PyTorch tem uma precisão de 99,37%, uma acurácia de 99,01%, um recall de 99,01% e uma pontuação F1-Score de 99,19%. Isso indica que a classe “A” tem um desempenho robusto e equilibrado. Já para a classe N, o PyTorch mostra uma precisão de 99,61%, uma acurácia excepcional de 99,76% e um recall de 99,76% e F1-Score de 99,69%.

3.4 Comparação com a literatura

O desempenho dos modelos melhorou significativamente ao longo dos anos, conforme demonstrado na Tabela 9 pela análise dos resultados de classificação dos batimentos cardíacos usando o banco de dados PTB. Observa-se que os estudos mais recentes têm apresentado melhorias significativas em comparação com os estudos mais antigos, que, por exemplo, Dicker et al. (2019) alcançaram uma acurácia de 83,82%. Por exemplo, Ahmad et al. (2021) descobriu uma precisão de 99,2%. Isso sugere que os modelos estão cada vez mais capazes de identificar padrões complexos em dados.

Tabela 9 - Comparação dos resultados da classificação dos batimentos cardíacos da base PTB com métodos anteriores.

Modelo Anteriores	Acurácia (%)	Precisão (%)	Recall (%)
Dicker et al. (2019)	83,82	82	95
Acharya et al. (2017)	95,22	95,49	94,19
Kojuri et al. (2015)	95,6	97,9	93,3
Kashuee et al. (2018)	95,9	95,2	95,1
Liu et al. (2018)	96	97,37	95,4
Sharma et al. (2015)	96	99	93
Chen et al. (2018)	96,18	97,32	93,67
Cao et al. (2020)	96,65	-	-
Ahamed et al. (2020)	97,66	-	-
Ahmad et al. (2021)	99,2	98,0	98,0
Este trabalho	**99,55**	**99,49**	**99,38**

Fonte: Próprio autor (2024).

O modelo criado por este trabalho apresentou a melhor taxa de precisão entre todos os estudos analisados, atingindo uma taxa de precisão de 99,55% em comparação com pesquisas anteriores. Ainda que essa diferença pareça pequena, representa um avanço significativo na precisão do modelo, especialmente em um campo onde até mesmo uma melhoria mínima pode ter um grande impacto na prática clínica.

Os indicadores de precisão e recall do estudo deste trabalho são mais precisos do que os dos modelos anteriores. A precisão de 99,49% indica uma alta eficiência na identificação de classes positivas com poucos falsos positivos. Além disso, o recall de 99,38% mostra que o modelo foi capaz de identificar a maioria dos casos positivos e reduzir o número de falsos negativos. Em contraste, pesquisas como as de Ahmad et al. (2021), que alcançaram precisão e recall de 98%, mostram que o modelo proposto por este trabalho é mais confiável na classificação dos batimentos cardíacos.

Os avanços no estudo deste trabalho podem ser atribuídos a melhorias metodológicas, como desenvolvimentos na arquitetura do modelo, métodos de pré-processamento mais eficientes e uso de conjuntos de dados mais representativos. Esse progresso mostra o estado da arte na classificação de batimentos cardíacos, enfatizando o potencial do modelo que está sendo desenvolvido para uso clínico.

Assim, esta contribuição se mostra significativa ao não apenas melhorar métricas de desempenho, mas também em alcançar o estado da arte na classificação para estes dados, como pode-se observar na literatura contemporânea comparada, podendo influenciar futuras pesquisas e práticas na área.

Ao discutir as comparações, é importante destacar que o objetivo foi obter os melhores indicadores possíveis para os modelos utilizando os dois *frameworks* analisados, e não apenas realizar testes em condições idênticas. Nesse contexto, os melhores resultados obtidos com o TensorFlow só foram possíveis ao utilizar lotes menores de dados, devido ao alto consumo de memória associado a esse *framework*. Essa abordagem foi necessária para maximizar o desempenho do TensorFlow, o que permitiu alcançar os resultados apresentados. Isso ressalta a importância de ajustar as condições de treinamento para cada *framework*, visando extrair o máximo potencial de cada um.

Analisar os resultados usando métodos e ferramentas e comparando-os à literatura atual. Esse objetivo foi alcançado. Os resultados mostraram que o modelo do PyTorch era mais preciso do que o TensorFlow e também tinha mais tempo de treinamento. A análise dos resultados e a comparação com a literatura mostram que essas descobertas estão alinhadas com as tendências gerais relatadas: PyTorch geralmente se destaca. Mas uma

análise mais aprofundada e uma comparação com uma coleção maior de estudos podem fornecer uma compreensão mais abrangente.

4. Conclusões

O objetivo principal deste estudo foi analisar ferramentas e métodos de redes neurais artificiais para classificar automaticamente arritmias cardíacas. Foi estabelecido um conjunto de três metas específicas para atingir esse objetivo.

O objetivo de escolher uma base de dados aberta sobre arritmias cardíaca foi alcançado. A base de dados PTB Diagnostic ECG foi a base de dados pública escolhida. Uma grande variedade de sinais cardíacos foi fornecida por esta base de dados, o que foi necessário para o treinamento e validação dos modelos de classificação.

Definir um conjunto abrangente de recursos e métodos para a classificação automática também foi alcançado. Os *frameworks* PyTorch e TensorFlow foram usados para desenvolver e implementar uma variedade de técnicas de redes neurais. O modelo escolhido foi treinado e avaliado em ambas as ferramentas. Ele tinha camadas lineares e funções de ativação ReLU e *Sigmoid*. Essas ferramentas foram comparadas para avaliar seu desempenho em termos de precisão, tempo de treinamento e perda.

O PyTorch se destacou na capacidade de generalizar modelos, alcançando métricas um pouco melhores em termos de precisão, acurácia, recall e f1-Score, principalmente nas arquiteturas ResNet18 e AlexNet. Essa vantagem se traduz em uma menor taxa de erros durante a classificação e uma maior capacidade de prever com precisão as classes positivas e negativas.

A comparação revelou diferenças na eficiência computacional além das métricas de desempenho. O PyTorch fez melhor uso dos recursos de *hardware*, o que pode resultar em tempos de treinamento mais rápidos e melhor aproveitamento de RAM, enquanto o TensorFlow, embora poderoso, pode exigir ajustes mais complicados para otimização em suas tarefas.

Na análise também se pode destacar que o PyTorch oferece maior flexibilidade no desenvolvimento e teste de modelos, devido à sua interface mais intuitiva e ao suporte dinâmico de gráficos computacionais. Isso pode ser especialmente útil para pesquisadores que desejam maior controle sobre o fluxo de dados e o comportamento dos modelos durante o treinamento.

Um dos pontos fortes foi a base de dados de eletrocardiograma *PTB Diagnostic ECG* fornecer uma ampla gama de sinais cardíacos, fornecendo uma base sólida para o treinamento e avaliação dos modelos, e por estar disponível publicamente, estes modelos podem ser comparados com outros trabalhos.

Outro ponto forte foi a análise comparativa de PyTorch e TensorFlow, que mostrou as diferenças de desempenho entre esses *frameworks* comuns. Essa comparação permitiu uma avaliação aprofundada das capacidades e eficiências de cada ferramenta. Isso ajudou na escolha consciente de padrões para pesquisas futuras e aplicações práticas em *deep learning*. Além disso, os indicadores de desempenho da classificação apresentaram melhores desempenhos em relação à literatura contemporânea comparada.

As descobertas feitas sobre os resultados de PyTorch e TensorFlow confirmam a abordagem utilizada no trabalho. A consistência da literatura com os resultados reforça a

credibilidade das descobertas e aumenta o acervo de informações sobre a classificação automática de arritmias cardíacas usando redes neurais artificiais.

Um dos pontos fracos do trabalho foi a limitação do *hardware* usado no treinamento. Tempos de treinamento significativamente mais longos foram causados pela falta de um computador com configurações mais avançadas, como uma GPU dedicada, ao contrário de Ahmad et al. (2021) que utilizou uma NVIDIA GTX-1070 GPU. Como resultado dessa restrição de *hardware*, não foi possível realizar um número maior de épocas de treinamento, o que poderia ter permitido resultados ainda melhores. Seria possível explorar mais a fundo o potencial dos modelos criados e obter melhores performances com recursos computacionais mais fortes.

Vários caminhos podem ser explorados em trabalhos futuros para aprofundar e ampliar os resultados deste estudo. Para começar, experimentos podem ser realizados para avaliar a generalização dos modelos desenvolvidos. Isso permite identificar padrões consistentes e possíveis discrepâncias comparando os resultados obtidos com várias bases de dados, como os conjuntos de dados do PhysioNet Challenge ou o MIT-BIH Arrhythmia Database. Investigar como as arquiteturas de redes neurais mais complexas, como redes neurais convolucionais (CNNs) e redes neurais recorrentes (RNNs), e aplicar técnicas de aprendizado profundo mais avançadas, podem melhorar a precisão e a robustez dos modelos. No entanto, explorar estas propostas exige melhores recursos de *hardware*.

Além disso, a eficiência dos modelos de classificação pode ser melhorada examinando os efeitos de várias abordagens de pré-processamento dos sinais de eletrocardiograma, como filtragem de ruído, normalização e segmentação dos batimentos cardíacos, bem como estudando técnicas de engenharia de características para extrair características pertinentes dos sinais de eletrocardiograma. Investigações sobre a combinação de redes neurais com outras técnicas de aprendizado de máquina, como máquinas de vetor de suporte (SVM) e k-vizinhos mais próximos (KNN), para o desenvolvimento de modelos híbridos que capturem diferentes aspectos dos dados de eletrocardiograma (ECG), e a investigação sobre o uso de técnicas de conjuntos, onde vários modelos são combinados para aumentar a robustez e a precisão da classificação.

Por fim, estudar métodos de atualização contínua dos modelos para incorporar novos dados e melhorar continuamente o desempenho, bem como os métodos de monitoramento e manutenção dos modelos em produção para garantir que eles permaneçam precisos e relevantes ao longo do tempo são necessários para garantir a longevidade e a eficácia das soluções propostas. Em estudos futuros, será possível validar e reforçar as descobertas deste estudo, bem como avançar significativamente na classificação automática de arritmias cardíacas, melhorando a saúde e o bem-estar dos pacientes.

Referências

ACHARYA, U. R.; FUJITA, H.; OH, S. L.; HAGIWARA, Y.; TAN, J. H.; ADAM, M. Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. Inf. Sci., vol. 415, pp. 190–198, Nov. 2017.

AHMAD, Z. et al. M. ECG Heartbeat Classification Using Multimodal Fusion. IEEE Access, v. 9, p. 100615-100626, 2021. DOI: 10.1109/ACCESS.2021.3097614. Disponível em:

https://www.researchgate.net/publication/353395631_ECG_Heartbeat_Classification_Using_Multimodal_Fusion. Acesso em: 18 jul. 2024.

AHAMED, M. A.; HASAN, K. A.; MONOWAR, K. F.; MASHNOOR, N.; HOSSAIN, M. A. ECG heartbeat classification using ensemble of efficient machine learning approaches on imbalanced datasets. In: *Proceedings of the 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, nov. 2020, p. 140-145.

AKULA, R.; MOHAMED, H. Automation algorithm to detect and quantify Electrocardiogram waves and intervals. 1st International Workshop on Industrial Applications of Internet of Things (IAIoT-2019), April 29 - May 2, 2019, Leuven, Belgium. Disponível em: https://www.researchgate.net/publication/333255053_Automation_algorithm_to_detect_and_quantify_Electrocardiogram_waves_and_intervals. Acesso em: 21 jul. 2024.

AMBRÓSIO, P. E. Redes neurais artificiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmonares. 2002. Ribeirão Preto – SP. Dissertação (Mestrado) – Faculdade de Filosofia, Universidade de São Paulo. Disponível em: https://www.teses.usp.br/teses/disponiveis/59/59135/tde-26102002-155559/publico/Dissertacao.pdf. Acesso em: 26 jul. 2024.

ANSI/AAMI EC57:2012 (R2020). Testing And Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms. AAMI, 2020. Disponível em: https://webstore.ansi.org/Standards/AAMI/ANSIAAMIEC572012R2020. Acesso em: 15 jul. 2024.

ANGOMED. Sistema de Derivações Eletrocardiográficas. 2014. Disponível em: https://angomed.com/sistema-de-derivacoes-eletrocardiograficas. Acesso em: 24 jul. 2024.

BOUSSELJOT, R.; KREISELER, D.; SCHNABEL, A. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Biomedizinische Technik, Band 40, Ergänzungsband 1, 1995, p. 317.

CAO, Y.; WEI, T.; LIN, N.; ZHANG, D.; RODRIGUES, J. J. P. C. Multi-channel lightweight convolutional neural network for remote myocardial infarction monitoring. In Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW), Apr. 2020, pp. 1–6.

CASDAGLI, M. C. Recurrence plots revisited. Physica D: Nonlinear Phenomena, Vol. 108, n. 1–2, 1997. Disponível em: https://doi-org.ez3.periodicos.capes.gov.br/10.1016/S0167-2789(97)82003-9. Acesso em 10 ago. 2024.

CHEN, Y.; CHEN, H.; HE, Z.; YANG, C.; CAO, Y. Multi-channel lightweight convolution neural network for anterior myocardial infarction detection. In Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Oct. 2018, pp. 572–578.

DIKER, A.; CÖMERT, Z.; AVCI, E.; TOĞAÇAR, M.; ERGEN, B. A novel application based on spectrogram and convolutional neural network for ECG classification. In Proc. 1st Int. Informat. Softw. Eng. Conf. (UBMYK), nov. 2019, pp. 1–6.

ECG Heartbeat Categorization Dataset. 2018. [Online]. Disponível em: https://www.kaggle.com/shayanfazeli/heartbeat. Acesso em: 10 ago. 2024.

ENTROPIA cruzada para machine learning. Aprendiz Artificial, 12 jan. 2023. Disponível em https://www.aprendizartificial.com/entropia-cruzada-para-machine-learning/. Acesso em: 10 ago. 2024.

FARIAS, Thiago S.; ROSSI, Rafael G. Estudo comparativo de arquiteturas de redes neurais em análise de sentimentos. 2021. Disponível em: https://www.researchgate.net/publication/353347502_Estudo_Comparativo_de_Arquiteturas_de_Redes_Neurais_em_Analise_de_Sentimentos. Acesso em: 19 ago. 2024.

GOLDBERGER, A. L. et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation, v. 101, n. 23, p. e215-e220, 2000. Disponível em: https://physionet.org/content/ptbdb/1.0.0. Acesso em: 28 ago. 2024.

HAYKIN, S. Redes Neurais- Princípios e Práticas. BOOKMAN, São Paulo, 2ª ed. 2001. 900 p. Acesso em: 25 jul. 2024.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. 2015. Disponível em: https://doi.org/10.48550/arXiv.1512.03385. Acesso em: 29 jul. 2024.

HERBERT, Í. Introdução ao perceptron MLP. 28 de março de 2022. Disponível em: https://www.italoinfo.com.br/ia/mlp_intro/index.php. Acesso em: 28/07/2024.

JOSHI, Vineet. Funções de Ativação. Tradução de Acervo Lima. 2021. Disponível em: https://acervolima.com/funcoes-de-ativacao/. Acesso em: 29 jul. 2024.

KACHUEE, M.; FAZELI, S.; SARRAFZADEH, M. ECG Heartbeat Classification: A Deep Transferable Representation. University of California, Los Angeles (UCLA), Los Angeles, USA, 2018. Disponível em: https://arxiv.org/pdf/1805.00794. Acesso em: 28 jun. 2024.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, v. 60, n. 6, p. 84-90, 2017. Disponível em: https://doi.org/10.1145/3065386. Acesso em: 17 ago. 2024.

KRIZHEVSKY, Alex. One weird trick for parallelizing convolutional neural networks. Google Inc., 29 abr. 2014. Disponível em: https://arxiv.org/pdf/1404.5997 . Acesso em: 18 ago. 2024.

KOJURI, J.; BOOSTANI, R.; DEHGHANI, P.; NOWROOZIPOUR, F.; SAKI, N. Prediction of acute myocardial infarction with artificial neural networks in patients with nondiagnostic electrocardiogram. J. Cardiovascular Disease Res., vol. 6, no. 2, pp. 51–59, May 2015.

LIU, W.; ZHANG, M.; ZHANG, Y.; LIAO, Y.; HUANG, Q.; CHANG, S.; WANG, H.; HE, J. Real-time multilead convolutional neural network for myocardial infarction detection. IEEE J. Biomed. Health Inform., vol. 22, no. 5, pp. 1434–1444, Sep. 2018.

LUZ, E. J. S. Classificação automática de arritmias: um novo método usando classificação hierárquica. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Ouro Preto, Ouro Preto, 2012. Disponível em:

https://www.repositorio.ufop.br/items/4ed58a08-ec05-44a1-af95-49e43b0f7505. Acesso em: 17 jul. 2024.

MACHADO, W. C.; FONSECA JÚNIOR, E. S. Redes Neurais Artificiais aplicadas na previsão do VTEC no Brasil. Boletim de Ciências Geodésicas, v. 19, n. 2, p. 227-246, 2013. Disponível em: https://revistas.ufpr.br/bcg/article/download/32401/205660. Acesso em: 26 jul. 2024.

NAIR, V.; HINTON, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010. Disponível em: https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf. Acesso em: 29 jul. 2024.

NIED, A. Treinamento de redes neurais artificiais baseado em sistemas de estrutura variável com taxa de aprendizado adaptativa. 2007. Belo Horizonte, MG. Tese (Doutorado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais. Disponível em: https://ppgee.ufmg.br/documentos/Defesas/669/Tese_Ademir_Nied.pdf. Acesso em: 26 jul. 2024.

O'SHEA, KEIRON, e RYAN NASH. "An Introduction to Convolutional Neural Networks.". arXiv. 2015. Disponível em: https://arxiv.org/pdf/1511.08458. Acesso em: 28 jul. 2024.

PyTorch**.** CrossEntropyLoss — PyTorch 2.0 Documentation. Disponível em: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. Acesso em: 16 ago. 2024.

PyTorch. BCELoss — PyTorch 2.0 Documentation 2.4. Disponível em: https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html. Acesso em: 19 ago. 2024.

SILVA, Rodrigo Emerson Valentim da. Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens. 2018. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Universidade Federal do Ceará, Campus Quixadá, Quixadá, 2018. Disponível em: https://repositorio.ufc.br/bitstream/riufc/39475/1/2018_tcc_revsilva.pdf. Acesso em: 29 jul. 2024.

SILVA, Renato M.; ALMEIDA, Tiago A.; YAMAKAMI, Akebo. Análise de desempenho de redes neurais artificiais para classificação automática de web spam. *Revista Brasileira de Computação Aplicada*, Passo Fundo, v. 4, n. 2, p. 42-57, out. 2012. ISSN 2176-6649.

SHARMA, L. N.; TRIPATHY, R. K.; DANDAPAT, S. Multiscale energy and eigenspace approach to detection and localization of myocardial infarction. IEEE Transactions on Biomedical Engineering, v. 62, n. 7, p. 1827-1837, jul. 2015.

SINGH, Karan. AlexNet, VGGNet, ResNet, and Inception. Medium, 21 jun. 2023. Disponível em: https://medium.com/@karansingh/alexnet-vggnet-resnet-and-inception . Acesso em: 28 jul. 2024.

SUN, Li; LU, Yanping; YANG, Kaitao; LI, Shaozi. ECG Analysis Using Multiple Instance Learning for Myocardial Infarction Detection. *IEEE Transactions on Biomedical Engineering*, v. 59, n. 12, p. 3348-3356, dez. 2012. DOI: 10.1109/TBME.2012.2213597. Disponível em:

https://www.researchgate.net/publication/230755455_ECG_Analysis_Using_Multiple_Instance_Learning_for_Myocardial_Infarction_Detection. Acesso em: 07/08/2024.

SPÖRL, C.; CASTRO, E. G.; LUCHIARI, A. Aplicação de Redes Neurais Artificiais na construção de modelos de fragilidade ambiental. Revista do Departamento de Geografia, v. 21, n. 1, p. 113-135, 2011. Acesso em: 25 jul. 2024.

VINICIUS, Anderson. Redes Neurais Artificiais. Medium, 23 nov. 2017. Disponível em: https://medium.com/@avinicius.adorno/redes-neurais-artificiais-418a34ea1a39. Acesso em: 26 jul. 2024.

WOLF, Alexandre Stürmer. Análise automática de sinais eletrocardiográficos por redes neurais artificiais. 2004. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Centro Técnico Científico, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2004. Disponível em: https://www2.dbd.puc-rio.br/pergamum/tesesabertas/0210429_04_cap_02.pdf . Acesso em 19 jul. 2024.