ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

JULIO ALEXANDER SIEG

ANÁLISE DA FUSÃO DE SENSORES PARA IDENTIFICAÇÃO DE ATIVIDADES DIÁRIAS

Porto Alegre
2024

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANÁLISE DA FUSÃO DE SENSORES PARA IDENTIFICAÇÃO DE ATIVIDADES DIÁRIAS

JULIO ALEXANDER SIEG

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. César Augusto Missio Marcon

Porto Alegre
2024

DEDICATÓRIA

Dedico este trabalho a todos que estiveram presentes e me deram apoio nos momentos mais difíceis, em que duvidei da minha capacidade de concluir essa dissertação.

Em primeiro lugar sempre a Deus, que me deu saúde e sabedoria para seguir em frente todos os dias.

Também dedico aos meus pais, que sempre fizeram tudo ao seu alcance para me proporcionar as condições necessárias para uma boa educação.

Dedico a minha noiva Suelem, que esteve presente em cada dia dos últimos anos, me motivando e dado forças em cada momento em que pensei em desistir.

Por último, mas não menos importante, dedico a cada um dos que me relembraram da importância desse trabalho, da importância para minha vida, dos motivos pelos quais eu iniciei a caminhada e da sua admiração pela minha conclusão da mesma.

A todos vocês (sem citar nomes para não correr o risco de esquecer de ninguém) o meu muito obrigado!

“Se não houver frutos, valeu a beleza das flores; se não houver flores, valeu a sombra das folhas; se não houver folhas, valeu a intenção da semente.”
(Henfil)

AGRADECIMENTOS

Agradeço ao professor e orientador César Marcon pela ajuda, revisões e conselhos, essenciais para o desenvolvimento deste trabalho.

Agradeço aos meus familiares e amigos por todo o apoio durante esse período do mestrado, propiciando todas as condições para o bom desenvolvimento do mesmo.

Agradeço também aos colegas do grupo de pesquisa do LifeSenior, por cada uma das reuniões ou discussões sobre os trabalhos de pesquisa, que sempre trouxeram ricas contribuições para o desenvolvimento desta pesquisa.

Agradeço também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), o Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTIC), a Toth LifeCare e a Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) pelo apoio financeiro, estrutural e tecnológico disponibilizados.

ANÁLISE DA FUSÃO DE SENSORES PARA IDENTIFICAÇÃO DE ATIVIDADES DIÁRIAS

RESUMO

Detecção de atividades diárias e quedas são temas vitais para a prevenção de doenças e tratamento da saúde de pessoas idosas. A obtenção de informação sobre estes temas pode ser alcançada por tecnologias baseadas em imagem ou sensores de diversas naturezas. O uso de câmeras permite obter dados acurados de atividades humanas, mas é limitado em cobertura e, principalmente, interfere significativamente a privacidade do idoso. O uso de sensores de diversas naturezas permite trazer informações relevantes do idoso, mantendo privacidade; adicionalmente, o emprego de sensores vestíveis, do inglês *wearable*, permite o monitoramento constante do idoso com maior área de cobertura que a área provida por uma câmera.

Sensores vestíveis podem ser uma solução para monitorar atividades diárias de idosos, sem a necessidade de cuidadores ou alterações no ambiente. Além disso, o reconhecimento de atividades diárias, é um tema de pesquisa, que pode trazer benefícios para diferentes áreas, incluindo a área de cuidados pessoais e prevenção ou detecção de quedas.

Esse trabalho busca analisar a fusão de sensores, comumente encontrados em dispositivos vestíveis, para entender o quanto a união de determinados sensores pode ajudar a identificar de uma maneira mais assertiva atividades diárias, principalmente aquelas que possam ajudar na identificação de quedas.

Palavras-Chave: saúde, fusão de sensores, detecção de atividades diárias, detecção de quedas, sensoriamento, dispositivo *wearable*.

ANALYSIS OF THE FUSION OF SENSORS FOR IDENTIFICATION OF DAILY ACTIVITIES

ABSTRACT

The detection of daily activities and falls is vital for the prevention of diseases and the treatment of the health of older people. Information on these topics can be obtained by image-based technologies or sensors of various natures. The use of cameras allows for obtaining accurate data on human activities, but it is limited in coverage and, mainly, significantly interferes with the privacy of the elderly. The use of sensors of different natures allows for bringing relevant information from the elderly and maintaining privacy; additionally, wearable sensors allow constant monitoring of the elderly with a larger coverage area than the area provided by a camera.

Wearable sensors can be a solution to monitor the daily activities of the elderly without the need for caregivers or changes in the environment. In addition, the recognition of daily activities is a research topic that can benefit different areas, including personal care and the prevention or detection of falls.

This work seeks to analyze the fusion of sensors commonly found in wearable devices to understand how the union of specific sensors can help to more assertively identify daily activities, especially those that can help identify falls.

Keywords: health, sensor fusion, daily activity recognition, fall detection, sensoring, wearable devices.

LISTA DE FIGURAS

Figura 2.1 – Conceito de funcionamento de um acelerômetro de 2 eixos (Fonte: [68]) 23
Figura 2.2 – Dados da atividade “Ficar parado” (Fonte: [54]) 23
Figura 2.3 – Dados da atividade “Caminhada” (Fonte: [54]) 24
Figura 2.4 – Estrutura de um giroscópio de um eixo (Fonte: [68]) 25
Figura 2.5 – Dados barométricos para atividades com elevador, escada rolante e escadas (Fonte: [42]) 26
Figura 2.6 – Tipos de algoritmos de aprendizado de máquina (Fonte: [61]) 27
Figura 3.1 – Fluxograma da operação do sistema (Fonte: Adaptado de [60]) 30
Figura 3.2 – Tabela comparativa de modelos e resultados (Fonte: [43]) 31
Figura 3.3 – Visão geral dos resultados (Fonte: [38]) 33
Figura 3.4 – Fluxograma do método proposto (Fonte: [63]) 35
Figura 3.5 – Framework de reconhecimento de atividades diárias proposto (Fonte: [72]) 36
Figura 4.1 – Proposta inicial do sistema de monitoramento de saúde LifeSenior (Fonte: [45]) 41
Figura 4.2 – Versão comercial do sistema de monitoramento de saúde LifeSenior (Fonte: [10]) 42
Figura 4.3 – Dispositivo vestível do projeto LifeSenior (Fonte: [45]) 43
Figura 4.4 – Interfaces do aplicativo LifeSenior Mobile (Fonte: [69]) 44
Figura 4.5 – Interfaces do aplicativo LifeSenior Cloud (Fonte: [69]) 45
Figura 4.6 – Arquitetura do ambiente de simulação do projeto LifeSenior (Fonte: Projeto LifeSenior) 46
Figura 4.7 – Simulador *wearable* (Fonte: Projeto LifeSenior) 47
Figura 4.8 – Simulador concentrador (Fonte: Projeto LifeSenior) 47
Figura 4.9 – Aplicação do trabalho no projeto LifeSenior (Fonte: o Autor) 48
Figura 5.1 – Fluxo aplicado na metodologia (Fonte: o Autor) 49
Figura 5.2 – Relação de quantidade de amostras x atividade do *dataset* PAMAP2 (Fonte: o Autor) 55
Figura 5.3 – Exemplo de matriz antes da aplicação da função *StandardScaler* (Fonte: o Autor) 56
Figura 5.4 – Exemplo de matriz após da aplicação da função *StandardScaler* (Fonte: o Autor) 56

Figura 5.5 – Exemplo gráfico de criação de amostras utilizando SMOTE (Fonte: [13]) 58
Figura 5.6 – Exemplificação de *oversampling* (Fonte: [13]) 59
Figura 5.7 – Exemplificação de *undersampling* (Fonte: [13]) 59
Figura 5.8 – Exemplo de imagem RGB (Fonte: [21]) 61
Figura 5.9 – Amostra antes da classificação utilizando KNN (Fonte: [22]) 61
Figura 5.10 – Amostra classificada utilizando KNN com k = 3 (Fonte: [22]) 62
Figura 5.11 – Exemplo de uma reta ótima separando dois grupos (Fonte: [18]) . . . 62
Figura 5.12 – Exemplo de transformação não-linear (Fonte: [18]) 63
Figura 5.13 – Bloco de memória LSTM (Fonte: [52]) 64
Figura 5.14 – Representação visual do funcionamento de uma BiLSTM (Fonte: [67]) 66
Figura 5.15 – Definição das camadas iniciais da rede neural proposta (Fonte: o Autor) 67
Figura 5.16 – Exemplo de janela deslizante (Fonte: o Autor) 68
Figura 6.1 – Comparativo do número de amostras das classes do *dataset* antes e depois do uso da técnica SMOTE (Fonte: o Autor) 73
Figura 6.2 – *Validation Loss* versus *Training Loss* em caso de *underfitting* (Fonte: [8]) 77
Figura 6.3 – *Validation Loss* versus *Training Loss* em caso de *overfitting* (Fonte: [8]) 78
Figura 6.4 – *Validation Loss* versus *Training Loss* sem *underfitting* e/ou *overfitting* (Fonte: [8]) 79
Figura 6.5 – Teste com 20 épocas e janela deslizante de tamanho 5 (Fonte: o Autor) 79
Figura 6.6 – Teste com 20 épocas e janela deslizante de tamanho 30 (Fonte: o Autor) 80
Figura 6.7 – Teste com 10 épocas e janela deslizante de tamanho 20 (Fonte: o Autor) 80
Figura 6.8 – Teste com 10 épocas e janela deslizante de tamanho 30 (Fonte: o Autor) 81
Figura 6.9 – Teste com 10 épocas e janela deslizante de tamanho 100 (Fonte: o Autor) 81
Figura 6.10 – Teste com 100 épocas e janela deslizante de tamanho 30 (Fonte: o Autor) 82
Figura 6.11 – *Training loss versus Test loss* - Sensor de acelerômetro no peito (Fonte: o Autor) 82

Figura 6.12 – *Training loss versus Test loss* - Sensor de acelerômetro no pulso (Fonte: o Autor) 84
Figura 6.13 – *Training loss versus Test loss* - Acelerômetro + HR no pulso (Fonte: o Autor) 85
Figura 6.14 – *Training loss versus Test loss* - Acelerômetro + HR + Temperatura no pulso (Fonte: o Autor) 87
Figura 6.15 – *Training loss versus Test loss* - Giroscópio + Magnetômetro no pulso (Fonte: o Autor) 88
Figura 6.16 – *Training loss versus Test loss* - Acelerômetro + Giroscópio + Magnetômetro no pulso (Fonte: o Autor) 89
Figura 6.17 – *Training loss versus Test loss* - Acelerômetro + Giroscópio + Magnetômetro + HR + Temp no pulso (Fonte: o Autor) 91
Figura A.1 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (A) (Fonte: o Autor). 103
Figura A.2 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (B) (Fonte: o Autor). 104
Figura A.3 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (C) (Fonte: o Autor). 104
Figura A.4 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (D) (Fonte: o Autor). 104

LISTA DE TABELAS

Tabela 3.1 – Métodos de divisão dos dados (Fonte: [34]) 34

Tabela 3.2 – Resultados experimentais (Fonte: [34]) 34

Tabela 3.3 – Resultados obtidos com diferentes algoritmos de Machine Learning (ML) (Fonte: [33]) 38

Tabela 3.4 – Tabela comparativa entre trabalhos relacionados (Fonte: o Autor) . . 39

Tabela 5.1 – Resumo de características dos *datasets* pesquisados (Fonte: o Autor) 53

Tabela 6.1 – Ambientes de execução *versus* Tempo de execução (Fonte: o Autor) 69

Tabela 6.2 – Ambiente de execução *versus* Tempo de execução utilizando GPU (Fonte: o Autor) 70

Tabela 6.3 – Taxas de acertos e erros com o *dataset* desbalanceado (Fonte: o Autor) 72

Tabela 6.4 – Taxas de acertos e erros após o uso de SMOTE (Fonte: o Autor) 73

Tabela 6.5 – Taxas de acertos e erros após o uso de pesos em amostras e em classes (Fonte: o Autor) 75

Tabela 6.6 – Taxas de acerto das estratégias discutidas (Fonte: o Autor) 77

Tabela 6.7 – Taxas de acerto - Sensor de acelerômetro posicionado no peito (Fonte: o Autor) 83

Tabela 6.8 – Taxas de acerto - Sensor de acelerômetro no pulso (Fonte: o Autor) 85

Tabela 6.9 – Taxas de acerto - Acelerômetro + HR no pulso (Fonte: o Autor) 86

Tabela 6.10 – Taxas de acerto - Acelerômetro + HR + Temperatura no pulso (Fonte: o Autor) 87

Tabela 6.11 – Taxas de acerto - Giroscópio + Magnetômetro no pulso (Fonte: o Autor) 89

Tabela 6.12 – Taxas de acerto - Acelerômetro + Giroscópio + Magnetômetro no pulso (Fonte: o Autor) 90

Tabela 6.13 – Taxas de acerto - Acelerômetro + Giroscópio + Magnetômetro + HR + Temp no pulso (Fonte: o Autor) 91

Tabela 6.14 – Taxas de acerto sumarizadas em relação a fusão de sensores (Fonte: o Autor) 93

LISTA DE ALGORITMOS

Algoritmo 6.1 – Utilização da biblioteca SMOTE (Fonte: o Autor) 72
Algoritmo 6.2 – Utilização da função *compute_sample_weight* (Fonte: o Autor) . . 74
Algoritmo 6.3 – Utilização da função *compute_class_weight* (Fonte: o Autor) 75
Algoritmo B.1 – Carregamento das bibliotecas e definição do caminho do *dataset* (Fonte: o Autor) 105
Algoritmo B.2 – Definição de dicionário de dados (Fonte: o Autor) 106
Algoritmo B.3 – Função para definição de janelas deslizantes (Fonte: o Autor) . . . 107
Algoritmo B.4 – Função de randomização dos dados e agrupamento de dados em um só *dataframe* (Fonte: o Autor) 107
Algoritmo B.5 – Definição das colunas do *dataframe* (Fonte: o Autor) 108
Algoritmo B.6 – Pré-processamento do *dataframe* (Fonte: o Autor) 109
Algoritmo B.7 – Definição da janela deslizante, divisão dos dados e definição dos parâmetros da rede (Fonte: o Autor) 110
Algoritmo B.8 – Treinamento do modelo e obtenção de históricos de perda (Fonte: o Autor) 111
Algoritmo B.9 – Visualização e avaliação dos resultados (Fonte: o Autor) 112

LISTA DE SIGLAS

ACC – Acelerômetro

ADL – Activity of Daily Living

ANN – Artificial Neural Networks

API – Application Programming Interface

AWS – Amazon Web Services

BB-FUS – Better-than-the-Best Fusion

BILSTM – Bidirectional LSTM

BLE – Bluetooth Low Energy

BR/EDR – Bluetooth Basic Rate/Enhanced Data Rate

CMYK – Cyan, Magenta, Yellow, and Key

CNN – Convolutional Neural Network

CPU – Central Processing Unit

DBF – Deep Belief Network

DT – Decision Tree

EC – Ensemble Classifier

ECG – Eletrocardiograma

EDA – Atividade Eletrodérmica

EMG – Eletromiografia

G – Gauss

GPU – Graphics Processing Unit

HAR – Reconhecimento de Atividades Humanas

HDL – Aprendizado Profundo Hierárquico

HAR – Heart Rate

HSV – Hue Saturation Value

IA – Inteligência Artificial

IOT – Internet of Things

IMU – Unidade de Medida Inercial

ISM – Industrial, Científica e Médica

JSON – Javascript Object Notation

KNN – K-nearest Neighbors

KPCA – Análise de Componentes Principais com Kernel

LSTM – Long Short-Term Memory

M2M – Machine to Machine

MEMS – Micro-ElectroMechanical Systems

ML – Machine Learning

NAN – Not a Number

PCP – Gráfico de Coordenadas Paralelas

PPG – Fotopletismografia

RF – Random Forest

RGB – Red, Green, Blue

RSS – Sinal de Força de Recepção

SEMG – Eletromiografia de Superfície

SMOTE – Synthetic Minority Over-sampling Technique

SVM – Support Vector Machine

T – Tesla

TCP – Transmission Control Protocol

WI-FI – Wireless Fidelity

WSN – Rede de Sensores Sem Fio

SUMÁRIO

1 INTRODUÇÃO 17

1.1 MOTIVAÇÃO 19

1.2 OBJETIVOS 20

1.3 OBJETIVOS ESPECÍFICOS 20

1.4 ORGANIZAÇÃO DO DOCUMENTO 20

2 FUNDAMENTAÇÃO TEÓRICA 22

2.1 SENSORES DE MOVIMENTO 22

2.2 APRENDIZADO DE MÁQUINA 26

3 TRABALHOS RELACIONADOS 29

3.1 FALL DETECTION SYSTEM USING WEARABLE SENSORS WITH AUTOMATED NOTIFICATION 29

3.2 EXPLORING HUMAN ACTIVITY RECOGNITION USING FEATURE LEVEL FUSION OF INERTIAL AND ELECTROMYOGRAPHY DATA 30

3.3 DEEP LEARNING NETWORKS FOR EATING AND DRINKING RECOGNITION BASED ON SMARTWATCH SENSORS 31

3.4 ENHANCING HUMAN ACTIVITY RECOGNITION THROUGH SENSOR FUSION AND HYBRID DEEP LEARNING MODEL 32

3.5 A FRAMEWORK FOR DAILY LIVING ACTIVITY RECOGNITION USING FUSION OF SMARTPHONE INERTIAL SENSORS DATA 32

3.6 HIERARCHICAL DEEP LEARNING MODEL WITH INERTIAL AND PHYSIOLOGICAL SENSORS FUSION FOR WEARABLE-BASED HUMAN ACTIVITY RECOGNITION . . . 33

3.7 FUSION OF SMARTPHONE SENSOR DATA FOR CLASSIFICATION OF DAILY USER ACTIVITIES 34

3.8 WEARABLE MULTI-SENSOR DATA FUSION APPROACH FOR HUMAN ACTIVITY RECOGNITION USING MACHINE LEARNING ALGORITHMS 36

3.9 SENSOR FUSION FOR RECOGNITION OF ACTIVITIES OF DAILY LIVING 37

3.10 A ROBUST HUMAN ACTIVITY RECOGNITION SYSTEM USING SMARTPHONE SENSORS AND DEEP LEARNING 37

3.11 DATA FUSION AND MULTIPLE CLASSIFIER SYSTEMS FOR HUMAN ACTIVITY DETECTION AND HEALTH MONITORING: REVIEW AND OPEN RESEARCH DIRECTIONS 38

3.12 COMPARAÇÃO ENTRE TRABALHOS RELACIONADOS 39

4 CONTEXTUALIZAÇÃO DO TRABALHO 41
4.1 ARQUITETURA DO SISTEMA LIFESENIOR 41
4.1.1 WEARABLE LIFES 42
4.1.2 LIFESENIOR MOBILE 44
4.1.3 LIFESENIOR CLOUD 44
4.2 LIFESENIORPROFILE 45
4.3 SIMULADOR LIFESENIOR 45
4.4 APLICAÇÃO DA PESQUISA REALIZADA NO PROJETO LIFESENIOR 47

5 ETAPAS DO TRABALHO 49
5.1 MOTIVAÇÃO NA ESCOLHA DE SENSORES 50
5.1.1 ACELERÔMETRO TRIAXIAL 50
5.1.2 SENSOR DE BATIMENTOS CARDÍACOS 51
5.1.3 GIROSCÓPIO 51
5.1.4 MAGNETÔMETRO 52
5.2 ESCOLHA DO *DATASET* 52
5.3 PRÉ-PROCESSAMENTO DOS DADOS 54
5.4 APLICAÇÃO DE TÉCNICAS PARA *DATASETS* DESBALANCEADOS 57
5.4.1 SMOTE - SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE 57
5.4.2 PENALIZAÇÃO 58
5.4.3 REAMOSTRAGEM 58
5.5 ESCOLHA DO MODELO DE ML 60
5.5.1 CONVOLUTIONAL NEURAL NETWORK (CNN) 60
5.5.2 K-NEAREST NEIGHBORS (KNN) 61
5.5.3 SUPPORT VECTOR MACHINE (SVM) 62
5.5.4 LONG SHORT-TERM MEMORY (LSTM) 63
5.5.5 BIDIRECTIONAL LONG SHORT-TERM MEMORY (BILSTM) 65
5.5.6 DEFINIÇÃO DO MODELO PARA ESTE TRABALHO 65
5.6 DEFINIÇÃO DOS PARÂMETROS E TREINAMENTO DO MODELO 66
5.7 DEFINIÇÃO DE JANELAS DESLIZANTES 68

6 RESULTADOS EXPERIMENTAIS 69
6.1 DEFINIÇÃO DO AMBIENTE DE EXECUÇÃO 69
6.2 DATASET DESBALANCEADO 70
6.2.1 APLICAÇÃO DA FUNÇÃO SMOTE 71

6.2.2 APLICAÇÃO DE PESOS EM AMOSTRAS E CLASSES 74
6.2.3 COMPARAÇÃO ENTRE ESTRATÉGIAS 76
6.3 DEFINIÇÃO DO NÚMERO DE ÉPOCAS E TAMANHO DA JANELA DESLIZANTE 76
6.4 FUSÃO DE SENSORES *VERSUS* IDENTIFICAÇÃO DE ATIVIDADES DIÁRIAS 82
6.5 COMPARAÇÃO ENTRE TAXAS DE ACERTO DE SENSORES FUNDIDOS 90

7 CONCLUSÃO E TRABALHOS FUTUROS 94

REFERÊNCIAS BIBLIOGRÁFICAS 96

APÊNDICE A – Dataset **103**

APÊNDICE B – Código-fonte **105**

1. INTRODUÇÃO

O envelhecimento populacional é um fenômeno global, e o Brasil não é exceção. Até 2020, a população de idosos (pessoas com mais de 60 anos) representava um aumento significativo, alcançando cerca de 18.8%[1] em uma população total de 211.8 milhões [28]. Este crescimento traz desafios cruciais para os sistemas de saúde, uma vez que os idosos estão mais suscetíveis a doenças crônicas [55] e incidentes como quedas. A necessidade de soluções inovadoras em cuidados de saúde, como a telemedicina e o monitoramento contínuo do estado de saúde, torna-se cada vez mais premente, especialmente para aqueles que vivem de forma independente.

A prevenção de quedas, um dos riscos mais comuns enfrentados pelos idosos em ambientes domésticos, é um ponto crítico. Estudos indicam que pessoas com mais de 65 anos estão particularmente em risco de quedas, muitas vezes necessitando de assistência imediata após o evento [35]. Enquanto várias soluções existem, como sistemas de alerta baseados em braceletes conectados a telefones fixos [1], estas tendem a ser caras e limitadas em cobertura. Além disso, a implementação de sistemas baseados em câmeras invadem a privacidade do idoso.

A ascensão dos dispositivos inteligentes, como *smartphones*, *smartwatches* e pulseiras *fitness*, abre novas possibilidades para o monitoramento eficaz dos idosos [36]. Conforme o portal Statistics, as vendas de *smartwatches* em todo o mundo tem crescido significativamente (e.g., entre 2014 a 2017 dobraram de ano para ano) [40]. Esses dispositivos vestíveis, equipados com uma variedade de sensores, oferecem uma solução menos invasiva e mais adaptável para o monitoramento contínuo e em tempo real. O grupo LifeSenior[2], o qual Este Autor está inserido, visa utilizar um dispositivo vestível do tipo relógio, incorporando múltiplos sensores, para rastrear uma gama de atividades diárias e potenciais situações de emergência.

O foco principal desta dissertação é explorar como a combinação de diferentes sensores em dispositivos vestíveis pode aprimorar a identificação de atividades diárias e a prevenção de quedas em idosos. Investigaremos a eficácia de diferentes sensores, quando utilizados em conjunto, na melhoria da assertividade na detecção de padrões de atividade. Baseando-se no modelo de relógio do LifeSenior, este estudo aplica tecnologias avançadas para determinar a importância relativa de cada sensor na identificação de atividades específicas dos idosos.

A capacidade de detectar e analisar atividades diárias é crucial para a saúde e bem-estar dos idosos. Um sistema eficaz de monitoramento pode fornecer informações

[1] Os texto terá os números reais no formato inglês para manter a coerência com as figuras e tabelas importadas de outras referências.

[2] LifeSenior é um projeto em cooperação academia-indústria para o monitoramento remoto de idosos. O Capítulo 4.1 apresenta um detalhamento deste projeto.

valiosas sobre o comportamento e a evolução da saúde do idoso, permitindo intervenções proativas para prevenir incidentes e gerenciar condições de saúde. O presente estudo visa contribuir para este campo, oferecendo *insights* sobre a utilização eficaz da tecnologia vestível no monitoramento de idosos, focando na fusão de sensores e sua aplicabilidade prática. Embora o trabalho não aborde diretamente a utilização dessas informações para a previsão de eventos, como quedas, a detecção de atividades diárias serve como um importante alicerce para futuras investigações neste campo.

Esse trabalho busca fornecer uma contribuição significativa ao estudo da saúde dos idosos, especificamente na aplicação de tecnologias vestíveis para a detecção de atividades diárias. Espera-se que os resultados deste estudo possam ser aplicados para melhorar a qualidade de vida dos idosos, proporcionando uma base para o desenvolvimento de sistemas de monitoramento mais eficientes e menos intrusivos.

1.1 Motivação

A detecção de atividades diárias é uma informação importante para a tomada de decisões voltadas à saúde; mormente, para idosos, sujeitos à maior incidência de comorbidades. Um sistema que tenha o mapeamento das atividades do idoso ao longo do tempo pode entender o comportamento deste e ver sua evolução. Por exemplo, o sistema pode avaliar a quantidade de movimentação diária e detectar que ele está em um período de baixa atividade; esta informação pode ser útil para predizer um problema neurológico ou uma situação de depressão, que pode ser tratada com antecedência, mitigando problemas de saúde.

Este trabalho foca apenas na avaliação de determinados sensores em conjunto, buscando identificar quais destes podem ser mais importantes para a **detecção** de algumas atividades diárias, deixando o uso desta informação para **previsibilidade** como um trabalho futuro. Adicionalmente, dentro do grupo LifeSenior, a informação de atividade diária é usada para auxiliar na detecção de quedas; inclusive, como trabalho futuro, a detecção de atividades diárias poderá ser usada na prevenção de quedas do idoso.

O sistema LifeSenior incluiu um dispositivo vestível composto por sensores que realizam o monitoramento de sinais vitais do portador, bem como, de eventos relativos ao dispositivo, permitindo estimar algumas informações sobre o portador do dispositivo. Dentro do conjunto de sensores de eventos relativos ao dispositivo estão o giroscópio, acelerômetro, magnetômetro e barômetro. Os dados destes sensores podem ser integrados de forma a coletar estados do portador, tais como (i) “dormindo”, (ii) “subindo escadas”, (iii) “caminhando”, (iv) “escovando os dentes” e (v) “caindo”.

Adicionalmente, um sistema de informação, baseado, por exemplo, em aprendizado de máquina, pode associar os estados do portador do dispositivo vestível, aumentando a assertividade de uma detecção. Um caso onde está associação é possível está na correlação de uma queda com o estado de “descendo uma escada”; o algoritmo de detecção de quedas pode inferir a probabilidade e gravidade de uma queda ao descer uma escada, integrar os dados sensoriais do dispositivo e aumentar a assertividade na decisão a ser tomada.

1.2 Objetivos

O objetivo principal desta dissertação é investigar como a integração de diversos sensores em dispositivos vestíveis pode aprimorar a identificação de atividades diárias. Este estudo visa explorar a sinergia entre tecnologias de sensoriamento e técnicas avançadas de análise de dados, para desenvolver um sistema de monitoramento mais eficiente e menos intrusivo para a população idosa.

1.3 Objetivos específicos

- Pesquisar e analisar sistemas que empregam dispositivos vestíveis para identificar atividades diárias, bem como algoritmos implementados e sensores utilizados;
- Pesquisar e analisar *datasets* e os sensores utilizados, buscando empregar estes mesmos sensores na análise de quais são mais úteis, quando utilizados em conjunto, para determinar com precisão as atividades diárias. Preferencialmente, os sensores utilizados para análise devem ser os mesmos presentes no dispositivo vestível estudado pelo grupo LifeSenior;
- Examinar como diferentes configurações de sensores, tais como acelerômetros, giroscópios e magnetômetros, podem ser combinadas para otimizar a coleta de dados sobre atividades diárias dos idosos. Este exame busca determinar quais combinações oferecem a maior precisão e relevância para o monitoramento.
- Identificar corretamente atividades diárias, permitindo estudos futuros de melhoria de eficiência energética, já que durante a identificação de algumas atividades diárias alguns sensores podem ser desativados, ou reduzida sua taxa de amostragem.
- Contribuir para o campo da saúde geriátrica, oferecendo *insights* e soluções tecnológicas inovadoras. Este objetivo visa reforçar a importância de abordagens de monitoramento que sejam tanto eficazes quanto respeitosas às necessidades e à privacidade dos idosos.

1.4 Organização do documento

Esta seção delineia a estrutura da presente dissertação, expondo o conteúdo e as contribuições de cada capítulo. O trabalho está estruturado em sete capítulos interligados.

O Capítulo 1, denominado Introdução, introduz o tema abordado neste estudo, estabelecendo os resultados esperados por meio da definição dos objetivos e expondo as limitações da pesquisa para proporcionar uma visão clara do escopo proposto.

O Capítulo 2 aborda a fundamentação teórica, proporcionando uma base conceitual e teórica para a pesquisa. Este capítulo é crucial para situar o leitor no contexto do estudo, evidenciando a compreensão adquirida sobre os conceitos, teorias e estudos relevantes relacionados.

O Capítulo 3 explora os trabalhos relacionados, oferecendo uma análise crítica e abrangente dos estudos anteriores associados ao tema da pesquisa, contextualizando o trabalho no panorama da literatura existente. Este capítulo revela o que já foi investigado, identifica lacunas existentes e esclarece como a presente pesquisa contribuirá para o campo de estudo.

O Capítulo 4 contextualiza o trabalho, estabelecendo relações entre esta dissertação e os projetos já desenvolvidos ou em progresso no âmbito do projeto LifeSenior. Além disso, proporciona uma visão sucinta da arquitetura do sistema LifeSenior e de seus componentes de hardware e software.

O Capítulo 5 descreve as etapas do trabalho, detalhando cada uma desde a escolha do conjunto de dados (*dataset*) e sensores até os parâmetros utilizados na execução do algoritmo.

O Capítulo 6 apresenta os resultados experimentais provenientes dos testes realizados, algoritmos executados e conclusões obtidas. Os resultados são expostos detalhadamente, incluindo a motivação por trás de cada teste realizado.

No Capítulo 7, são apresentadas as conclusões do trabalho, estabelecendo a relação entre os objetivos inicialmente delineados e os resultados alcançados. Além disso, são sugeridas possibilidades para a continuação da pesquisa por meio de trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresenta conceitos relacionados a sensores de movimento utilizados no andamento do trabalho e que estão presentes no projeto LifeSenior, bem como, conceitos básicos de aprendizado de máquina.

2.1 Sensores de movimento

Sensores são dispositivos capazes de detectar e responder a algum estímulo. Diferentes tipos de sensores são utilizados na indústria, nos mais diversos produtos, para obter informações de propriedades físicas, como: calor, pressão, movimento e luz. Ao reconhecer um estímulo, a função do sensor é emitir um sinal que possa ser convertido e interpretado por outros dispositivos. Existem muitos sensores que podem ser utilizados para medir propriedades físicas. Os mais importantes para esse trabalho são os sensores de movimento, capazes de trazer informações referentes à movimentação do dispositivo ao qual estão inseridos medindo aceleração ao longo de eixos de referência [26].

É preferível medir atividades físicas usando acelerometria, pois a aceleração é proporcional à força externa, portanto, pode refletir a intensidade e a frequência do movimento humano. Dados de aceleração podem ser usados para derivar informações de velocidade e deslocamento. Alguns acelerômetros podem responder à gravidade para fornecer detecção de inclinação em relação aos planos de referência quando giram conectados a objetos. Os dados de inclinação resultantes podem ser usados para classificar posturas corporais (orientações). Com essas características, a acelerometria consegue fornecer informações suficientes para medir uma gama de atividades humanas. Os acelerômetros são amplamente aceitos como sensores úteis e práticos para dispositivos vestíveis para medirem e avaliarem atividades em ambientes clínicos/laboratoriais ou em ambientes de vida livre [16].

A Figura 2.1 exemplifica o funcionamento de um acelerômetro de 2 eixos. Uma massa de prova de silício é suspensa e suportada por molas, também de silício, conferindo a esta massa uma mobilidade conhecida. O acelerômetro, por estar fixado ao sistema a ser medido, sofrerá a ação das mesmas forças impostas a este sistema. Uma variação de velocidade provocará, pelo princípio da inércia, uma alteração entre a distância dos eletrodos fixos e dos eletrodos móveis ligados à massa de prova. Em repouso, as capacitâncias C1 e C2 são iguais; sob efeito da aceleração, C1 e C2 variarão dependendo do sentido e intensidade das forças que provoquem essa mudança de estado inercial. A partir dos sinais elétricos provenientes destas variações de capacitância são obtidos os dados de aceleração [68].

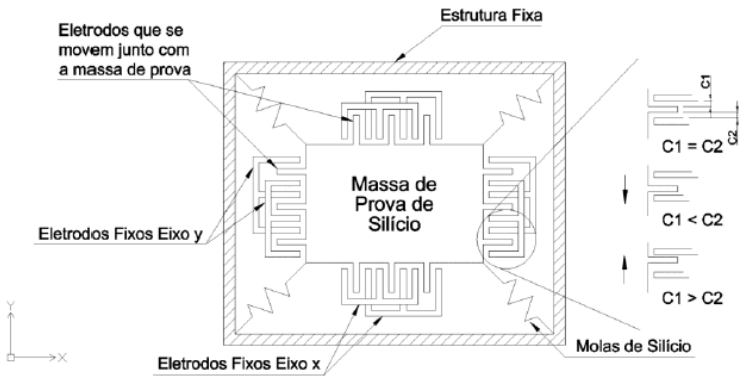


Figura 2.1 – Conceito de funcionamento de um acelerômetro de 2 eixos (Fonte: [68])

Acelerômetros têm sido usados em diversas partes do corpo humano, mas mais frequentemente no quadril ou pulso, sendo o último o mais comum devido à sua alta aceitabilidade para os participantes de estudos. Segundo Hartley et al. [32], todos os 24 participantes de seu estudo reportaram não sentirem desconforto ao utilizar um acelerômetro preso ao corpo. Ainda, 16 participantes reportaram comentários semelhantes ao de terem esquecido que algum dispositivo estava ligado ao seu corpo.

As informações apresentadas na Figura 2.2 e na Figura 2.3 trazem uma comparação de dados obtidos através do sensor de acelerometria apenas. Enquanto a Figura 2.2 traz os dados de acelerometria, em um período de 5 segundos, para a atividade de "Ficar parado", a Figura 2.3 traz os dados para a atividade de "Caminhada", no mesmo intervalo de tempo. A Figura 2.3 mostra que a variação significativa da amostragem do acelerômetro frente a Figura 2.2 permite inferir que o usuário do dispositivo está em um estado diferente - no caso, caminhando ao invés de estar parado.

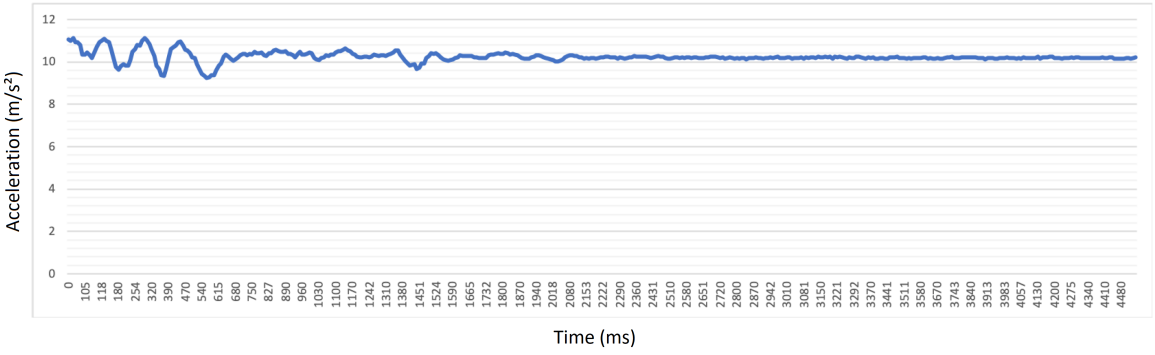


Figura 2.2 – Dados da atividade "Ficar parado" (Fonte: [54])

Alguns *datasets* que foram estudados, sendo eles o ShimFall&ADL [2] e o *dataset* for Activity of Daily Living (ADL) Recognition [12], usam principalmente dados de acelerometria. Porém, outros sensores podem ser utilizados para melhorar a eficácia de algoritmos de detecção de quedas e de detecção de atividades diárias.

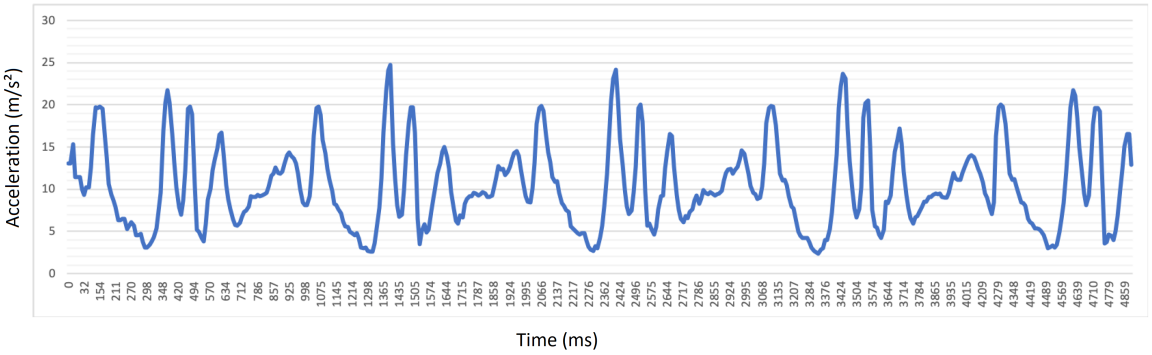


Figura 2.3 – Dados da atividade “Caminhada” (Fonte: [54])

Acelerômetro, giroscópio e magnetômetro são sensores, comumente integrados em produtos discretos como dispositivos vestíveis devido ao seu tamanho pequeno e ao baixo consumo de energia. Esses sensores contêm sistemas microeletromecânicos, em inglês Micro-ElectroMechanical Systems (MEMS), dispositivos miniaturizados que combinam componentes eletrônicos e mecânicos, e que aplicam diferentes estruturas de medição para parâmetros físicos correspondentes, como aceleração, velocidade angular e magnitude do campo magnético. Com base nas medidas, o sistema pode estimar a aceleração e orientação de parte do corpo humano, reconhecer postura e detectar queda [41].

Embora esses sensores de orientação tenham várias limitações quando usados separadamente, a combinação de dados dos sensores acelerômetro-giroscópio-magnetômetro pode melhorar essas deficiências. Na ausência de aceleração, acelerômetros podem ser usados para medir o ângulo do sensor de orientação em relação à gravidade; no entanto, os acelerômetros não podem fornecer informações sobre a rotação em torno do eixo vertical. As informações de orientação dos acelerômetros podem ser complementadas com dados de velocidade angular dos giroscópios. Infelizmente, os giroscópios não são isentos de problemas - um pequeno erro na medição da velocidade angular pode resultar em erros crescentes na orientação estimada (ou seja, erros de integração). Dessa forma, os magnetômetros usam o campo magnético da Terra como referência e podem fornecer informações sobre a rotação em torno do eixo vertical e corrigir erros de integração do giroscópio [37].

A Figura 2.4 apresenta a estrutura de um giroscópio de um eixo. Há dois conjuntos de molas de silício: Kx, suportando a estrutura externa e conferindo mobilidade na direção do eixo x; e Ky permitindo uma constante vibração da parte interna na direção do eixo y, causada por atuadores eletrostáticos. A força Coriolis proveniente de uma rotação em torno do eixo z faz com que tanto a parte central quanto a parte externa da estrutura se movam na direção x. Este movimento da parte externa causa uma variação da capacitância entre eletrodos fixos e móveis conectados a esta parte da estrutura, mesmo princípio de funcionamento do acelerômetro explicado anteriormente. A grande vantagem

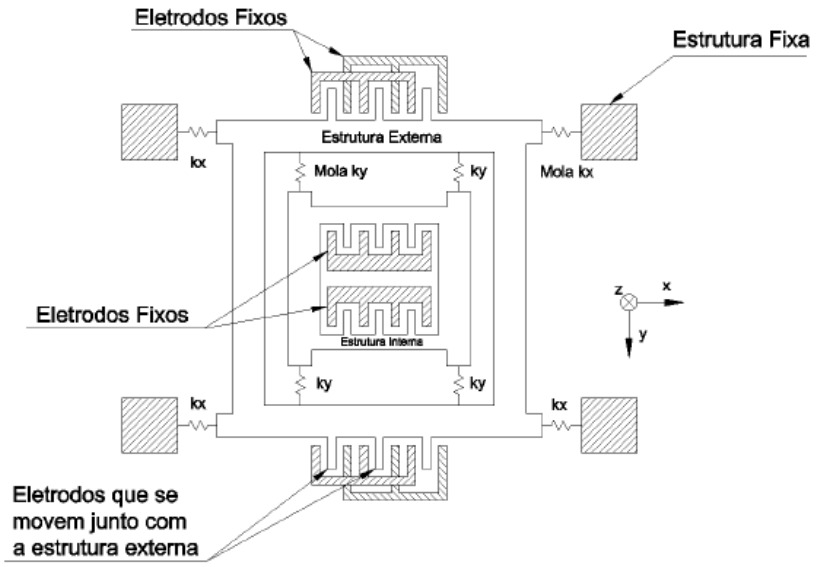


Figura 2.4 – Estrutura de um giroscópio de um eixo (Fonte: [68])

do giroscópio é funcionar perfeitamente em ambientes de gravidade zero, permitindo seu uso em equipamentos desenvolvidos para exploração espacial [68].

Os barômetros estão entre os mais antigos sensores da engenharia. Historicamente, eles têm sido usados principalmente como sensores ambientais para medir a pressão atmosférica para previsões meteorológicas ou como altímetros para aeronaves. Com o advento dos barômetros baseados em MEMS e sua incorporação sistemática em *smartphones* e dispositivos vestíveis, surgiu uma vasta gama de novas aplicações para o uso de barômetros. Por exemplo, é possível usar barômetros em conjunto com outros sensores para rastrear e identificar classes de atividades humanas, medindo mudanças na altitude. A escala das mudanças de altitude varia de uma queda a deslocamentos verticais como “subir ladeiras”, “subir escadas” e “andar de elevador”. Em aplicações recentes, os padrões das séries temporais de sinais barométricos têm se mostrado um bom indicador de atividades subjacentes, como “caminhar” ou “ficar ocioso” [42].

Na Figura 2.5, são apresentados dados barométricos de “descida/subida de elevador” (Figura 2.5a), “descida/subida de escada rolante” (Figura 2.5b) e “descida/subida em escadas convencionais” (Figura 2.5c). A Figura 2.5a divide os dados de um elevador parando em cada andar, repetindo por duas vezes o mesmo processo para descida e subida. Depois, um elevador descendo até o meio do prédio, parando, e continuando logo em seguida, fazendo esse mesmo procedimento na descida e na subida por 6 vezes. Por último, um elevador descendo até o térreo, e voltando ao topo do prédio, sem paradas, repetindo o mesmo procedimento por 10 vezes. A Figura 2.5b, se baseia em descidas e subidas em escadas rolantes, repetindo o mesmo processo por 10 vezes. A Figura 2.5c traz dados de subidas e descidas de escadas em um mesmo prédio, sendo que existe uma interconexão entre dois prédios para ser possível realizar o procedimento em todos os andares. A série temporal da pressão barométrica dos três tipos de atividades, indicou

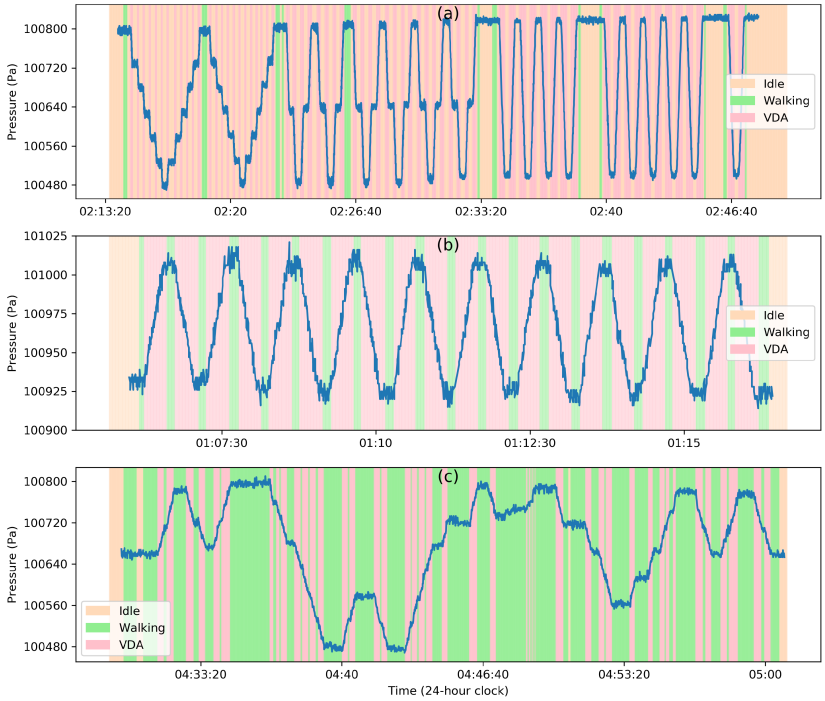


Figura 2.5 – Dados barométricos para atividades com elevador, escada rolante e escadas (Fonte: [42])

que a mudança no padrão de pressão tem flutuações mais altas à medida que nos movemos do passeio de elevador para o passeio de escada rolante e para subir escadas. Essa diferença se deve à velocidade com que o deslocamento vertical ocorreu e, no caso de escadas, foi afetado pelo desenho da escada (tipo, distância entre escadas consecutivas, etc.) e velocidade de caminhada [42].

2.2 Aprendizado de máquina

Os dados coletados pelos sensores podem ser usados para detectar problemas de saúde, por exemplo. No entanto, esses sensores fornecem uma enorme quantidade de dados brutos e heterogêneos, que precisam ser armazenados, processados e integrados, a fim de extrair informações mais relevantes sobre as atividades da vida diária de uma pessoa. Neste contexto, tecnologias de processamento de *big data* combinadas com técnicas de aprendizado de máquina fornecem a infraestrutura necessária para armazenar e processar grandes quantidades de dados de sensores. Além de identificar as atividades de vida diária, as técnicas de aprendizado de máquina também podem ser utilizadas para detectar a rotina do idoso e desvios dessa rotina [17].

Pesquisas em Inteligência Artificial (IA), particularmente, em aprendizado de máquina, cresceram rapidamente nos últimos anos no contexto de análise de dados e computação que normalmente permite que os aplicativos funcionem de maneira inteligente.

O aprendizado de máquina geralmente fornece aos sistemas a capacidade de aprender e aprimorar a partir da experiência, automaticamente, sem ser especificamente programado, e geralmente é referido como uma das tecnologias mais recentes e mais populares da quarta revolução industrial (também chamada de Indústria 4.0). Em geral, eficácia e eficiência de uma solução de aprendizado de máquina dependem da natureza e características dos dados e do desempenho dos algoritmos de aprendizado [61]; sendo estes algoritmos divididos em quatro categorias: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizado por reforço, conforme pode ser visto na Figura 2.6.

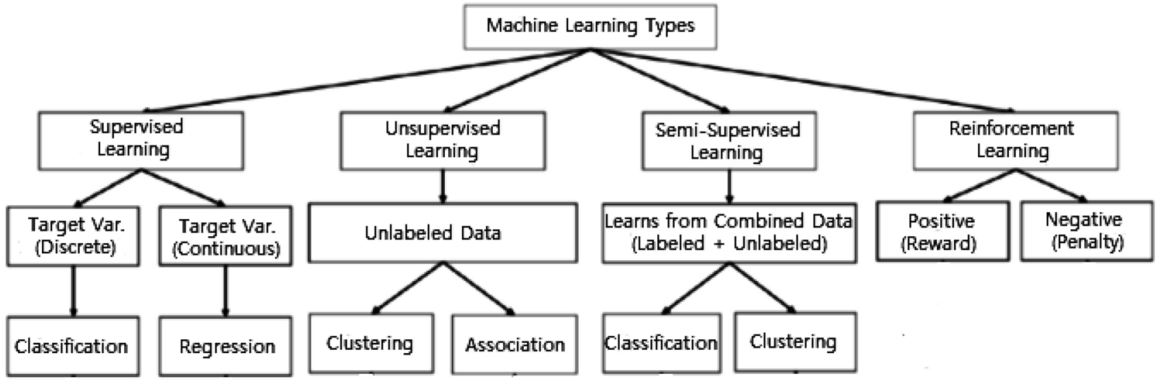


Figura 2.6 – Tipos de algoritmos de aprendizado de máquina (Fonte: [61])

O **aprendizado supervisionado** é tipicamente a tarefa de aprender uma função que mapeia uma entrada para uma saída com base em pares de amostra de entrada-saída. Ele usa dados de treinamento rotulados e uma coleção de exemplos de treinamento para inferir uma função. O aprendizado supervisionado é realizado quando certos objetivos são identificados a serem alcançados a partir de um determinado conjunto de entradas, ou seja, uma abordagem orientada por tarefas [61].

O **aprendizado não supervisionado** analisa conjuntos de dados não rotulados sem a necessidade de interferência humana, ou seja, um processo orientado a dados. Isso é amplamente usado para extrair recursos generativos, identificar tendências e estruturas significativas, agrupamentos em resultados e propósitos exploratórios. As tarefas de aprendizado não supervisionadas mais comuns são agrupamento, estimativa de densidade, aprendizado de recursos, redução de dimensionalidade, encontrar regras de associação e detecção de anomalias [61].

O **aprendizado semi-supervisionado** pode ser definido como uma hibridização dos métodos supervisionados e não supervisionados, pois opera em dados rotulados e não rotulados [61].

O **aprendizado por reforço** é um tipo de algoritmo de aprendizado de máquina que permite que agentes e máquinas de software avaliem automaticamente o comportamento ideal em um contexto ou ambiente específico para melhorar sua eficiência [61].

Existe uma grande quantidade de algoritmos utilizados nas mais diferentes áreas e aplicações do mundo real. Nos trabalhos relacionados, apresentados no Capítulo 3, notou-se o uso de algoritmos clássicos, como K-nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest (RF), dentre outros, e de algoritmos de aprendizado profundo, como Long Short-Term Memory (LSTM) e Convolutional Neural Network (CNN) para detecção de quedas e de atividades diárias. Dessa forma, alguns desses algoritmos terão um aprofundamento maior nesse trabalho.

3. TRABALHOS RELACIONADOS

Para elaboração desta dissertação, alguns trabalhos relacionados foram utilizados a fim de entender quais tecnologias têm sido empregadas para o mesmo fim deste trabalho, a fusão de sensores. Além disso, comparações entre tecnologias, resultados e sensores, ajudam a mostrar o que esta dissertação tem a contribuir em seus resultados e conclusões. Para encontrar trabalhos que embasem essa dissertação, foram utilizadas algumas *keywords*, sendo elas: *sensor fusion*, *daily activity*, adl, har, *health*, *elderly people*. Além disso, foram considerados apenas trabalhos que estivessem entre os anos de 2018 e 2024, afim de que os resultados aqui apresentados estivessem alinhados com o que vem sendo estudado nos últimos anos.

Esse capítulo traz uma descrição sumarizada de cada trabalho relacionado, com as técnicas utilizadas e os resultados encontrados. Ao fim desse capítulo, na Seção 3.12, há uma tabela comparativa entre os trabalhos relacionados e o trabalho realizado e apresentado nessa dissertação.

3.1 Fall detection system using wearable sensors with automated notification

Saod et al. [60] utilizaram dados de 10 participantes, homens e mulheres, com idades entre 30 e 55 anos. Seis atividades diárias foram consideradas, sendo elas: “ficar em pé”, “estar sentado e levantar-se”, “caminhar”, “deitar na cama”, “sentar-se em uma cadeira” e “sentar-se no chão”. Além disso, quatro situações de queda foram consideradas, sendo elas: “cair para frente”, “cair para trás”, “cair para o lado esquerdo” e “cair pro lado direito”. Os dados foram coletados através de um sistema composto de um Arduino, com giroscópio e acelerômetro, resumidamente, preso ao peito do participante. Com os dados coletados, os autores avaliaram os eixos x, y e z e definiram um valor limiar para cada atividade diária, ou seja, qual era o valor mínimo e qual era o valor máximo, para todos os eixos, considerando todos os participantes. Dessa forma, eles conseguiram definir limitadores do que consideram ser uma queda. Se os valores dos eixos x, y e z estiverem dentro dos limitadores, significa, para os autores, que o participante está em um movimento de queda. Dessa forma, um bipe é acionado pelo período de 10 segundos, onde o próprio participante, durante esse tempo, pode marcar a ação como sendo um falso-positivo. Caso isso não ocorra, uma notificação é enviada ao familiar do participante, informando sobre a queda. O modelo proposto pode ser visto na Figura 3.1.

Com essa fusão de sensores, e a estratégia de verificar todos os eixos, definindo limitadores para cada atividade diária, os testes realizados mostraram uma acurácia de 97% em seus resultados.

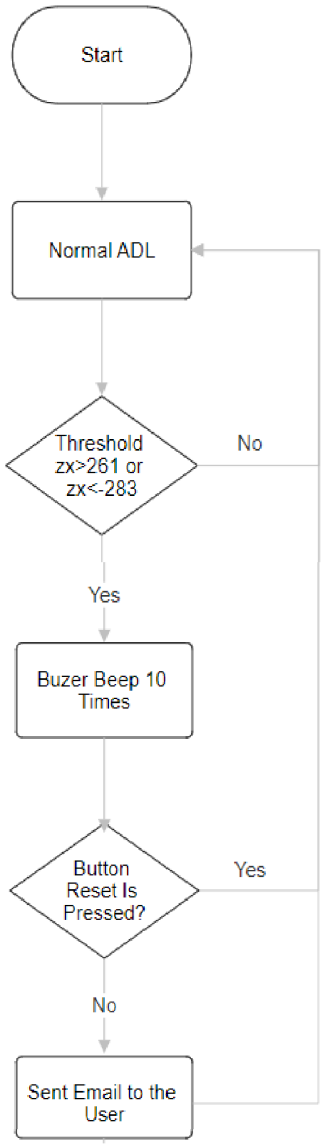


Figura 3.1 – Fluxograma da operação do sistema (Fonte: Adaptado de [60])

3.2 Exploring human activity recognition using feature level fusion of inertial and electromyography data

Yunus et al. [14] realizaram uma exploração sobre como dados inerciais, combinados com eletromiografia de superfície e aprendizado de máquina, podem trazer benefícios para o reconhecimento de atividades humanas. O estudo se baseia em quatro atividades básicas da vida diária: “caminhar”, “ficar em pé”, “descer escadas” e “subir escadas”. Para o estudo, 10 pessoas saudáveis foram consideradas, sendo que as mesmas utilizaram um dispositivo chamado Shimmer3, que tem em sua composição acelerômetro, giroscópio e eletromiografia (EMG).

Os dados obtidos passaram por uma série de etapas de pós-processamento, para então serem aplicados em dois modelos de classificação supervisionados (SVM e KNN). En-

tão, os resultados obtidos apresentaram uma acurácia 3.5% maior utilizando SVM, e 6.3% maior utilizando KNN, considerando a fusão de dados inerciais com dados de eletromiografia de superfície (sEMG).

3.3 Deep learning networks for eating and drinking recognition based on smartwatch sensors

Mekruksavanich et al. [43] comparam resultados de algumas redes neurais profundas em cima de dados coletados de sensores de *smartwatches* e *smartphones*. Inicialmente os autores utilizaram dados do *dataset* WISDM-HARB, acessível publicamente e disponibilizado pela Universidade da Califórnia. Esse *dataset* traz dados de 51 participantes para 18 atividades da vida diária. Como o trabalho foca apenas em atividades relacionadas a alimentação, apenas as atividades de “comer sopa (So)”, “comer batatas fritas (Ch)”, “comer massa (Pa)”, “comer um sanduíche (Sa)” e “beber de um copo (Cup)” foram consideradas. Os dados foram pré-processados através de estratégias de remoção de ruído e normalização.

O estudo investigou seis redes neurais profundas, sendo elas CNN, LSTM, Bidirectional LSTM (BiLSTM), Stacked LSTM, CNN-LSTM, e LSTM-CNN. Alguns estudos foram realizados em cima dos hiperparâmetros, como números de épocas, tamanho de *batch*, taxa de aprendizagem, dentre outros. Os testes foram realizados com todas as redes, considerando janelas de diferentes tamanhos.

Os resultados apresentados na Figura 3.2 mostram que a rede neural Stacked LSTM com uma dimensão de 25 atingiu uma acurácia de mais de 97%.

Models	Accuracy(%)				
	W200 S25	W200 S50	W200 S100	W200 S150	W200 S200
CNN	96.857	90.300	86.650	82.171	77.445
LSTM	93.795	83.900	73.220	67.192	65.885
BiLSTM	96.887	87.500	78.358	70.588	64.753
Stacked LSTM	97.372	84.400	76.052	70.346	65.400
CNN-LSTM	96.716	96.200	77.872	73.681	68.149
LSTM-CNN	95.291	91.000	80.380	78.896	66.613

Figura 3.2 – Tabela comparativa de modelos e resultados (Fonte: [43])

3.4 Enhancing human activity recognition through sensor fusion and hybrid deep learning model

O trabalho de Tarekegn et al. [66] busca analisar a fusão de alguns sensores para obter uma melhor acurácia no reconhecimento de atividades humanas. Para isso, foram utilizados dados de sensores de um cinto inteligente. Os dados foram coletados através de vinte participantes executando atividades da vida diária. As atividades incluem “caminhada”, “descer escadas”, “subir escadas”, “sentar”, “levantar”, e “ficar em pé”. Cada participante utilizou o cinto preso ao peito, com três unidades de medida inercial (IMUs) em posições diferentes do cinto: no meio, na esquerda e na direita. Cada uma das unidades obteve dados de giroscópio e de acelerômetro das três posições.

Após pré-processamento e análise dos dados, é realizada uma fusão de redes neurais para buscar obter melhores resultados no trabalho proposto. Inicialmente, uma rede neural CNN é aplicada para extração de recursos do *dataset*. O resultado dessa extração é uma sequência de *feature maps* (mapas de recursos), que servem como entrada para a arquitetura BiLSTM, já que estas são boas com processamento de dados sequenciais, aprendendo a dependência temporal dos dados.

Como resultado do treinamento e teste desses dados, foi possível notar que ao aumentar a janela de tempo considerada, os resultados melhoram significativamente. Com uma janela de 5s foi possível obter os melhores resultados, chegando a uma acurácia de 96%. Após essa janela de 5s, os resultados permaneceram estagnados ou pioraram. A acurácia para atividades como “ficar em pé”, chegou a 98.4%.

3.5 A framework for daily living activity recognition using fusion of smartphone inertial sensors data

Khan et al. [38] propõem um modelo de reconhecimento de atividades humanas no ambiente cotidiano, utilizando dados de sensores inerciais de *smartphones*. O foco do trabalho é reconhecer atividades humanas fora do ambiente controlado de laboratório, utilizando um conjunto de dados público chamado “ExtraSensory”. Desse conjunto de dados, seis atividades humanas foram selecionadas: “deitar”, “sentar”, “ficar em pé”, “correr”, “andar” e “andar de bicicleta”. Três classificadores de aprendizado de máquina (RF, KNN e Decision Trees (DTs)) são utilizados para a identificação das atividades humanas. O esquema proposto alcança uma precisão de classificação de 89.98%, superando, segundo os autores, esquemas existentes no reconhecimento de atividades humanas fora do laboratório. O artigo destaca a importância do reconhecimento de atividades humanas em ambientes não controlados, sendo útil para segurança pública, assistência médica e

monitoramento de atividades de motoristas, por exemplo. O estudo apresenta uma metodologia que combina dados de acelerômetro e giroscópio de *smartphones* para reconhecer atividades humanas no ambiente externo ao laboratório.

Além disso, o artigo discute estudos relacionados que empregam diferentes abordagens, como CNNs e modelos híbridos CNN-LSTM, para reconhecimento de atividades humanas em ambientes controlados e não-controlados. O trabalho conclui, destacando a eficácia do modelo proposto, que supera outros esquemas de reconhecimento de atividades humanas em ambientes não controlados. A Tabela I apresenta uma visão geral dos resultados experimentais encontrados pelos autores, onde o classificador RF alcança a maior precisão média de 89.98%, superando os classificadores DT e KNN.

TABLE I
AVERAGE PERFORMANCE VALUES OF RF, DT, AND KNN CLASSIFIERS FOR HUMAN ACTIVITY RECOGNITION IN THE WILD

Classifier	Accuracy	Precision	Recall	F-Measure	Kappa
RF	**0.898**	**0.902**	**0.900**	**0.900**	**0.861**
DT	0.846	0.846	0.846	0.846	0.788
KNN	0.871	0.870	0.871	0.871	0.822

Figura 3.3 – Visão geral dos resultados (Fonte: [38])

3.6 Hierarchical deep learning model with inertial and physiological sensors fusion for wearable-based human activity recognition

O artigo de Hwang et al. [34] propõe o desenvolvimento de um sistema avançado de Reconhecimento de Atividades Humanas (HAR) por meio da aplicação de uma abordagem de aprendizado profundo hierárquico (HDL). O modelo integrativo utiliza dados de sensores inerciais, como acelerômetros (ACC), e sensores fisiológicos, incluindo fotopletismografia (PPG) e atividade eletrodérmica (EDA), provenientes de dispositivos vestíveis. A HDL incorpora uma estrutura de árvore hierárquica, combinando CNN e LSTM. Os resultados obtidos indicam melhorias substanciais em termos de precisão e estabilidade em comparação com modelos convencionais, como RF, XGBoost e HHAR-net.

A Tabela 3.1 ilustra os três métodos que os autores buscaram realizar em cima dos dados do *dataset* utilizado. No método "*All-access*", os dados são extraídos em 80% para treino e 20% para testes, de todas as atividades para todos os sujeitos. Já o segundo método proposto, "*Cross-subject*", pega 12 sujeitos para treino e 3 para testes. E por último, o método "*Cross-activity*" exclui duas atividades de cada sujeito para teste, e exclui outras para treino.

	Details
All-access	Extract train (80%) and test (20%) sets from each activity in all subjects. Access all activities and subjects for training.
Cross-subject	12 subjects for training and 3 subjects for testing. Sequentially process (i.e. first test set is subject 1,2,3 and the second one is 4,5,6 while remained subjects are for train set). Average 5 trials (cross-validation).
Cross-activity	Randomly exclude two activities in each subject for testing while others for training. Average 4 trials (cross-validation).

Tabela 3.1 – Métodos de divisão dos dados (Fonte: [34])

Destaca-se a capacidade do sistema em reconhecer uma variedade de movimentos associados às atividades humanas normais em diferentes cenários do mundo real. O conjunto de dados empregado é o PPG-DaLiA, que contém dados de 15 sujeitos, para atividades de “sentar”, “subir e descer escadas”, “jogar futebol de mesa”, “andar de bicicleta”, “dirigir um carro”, “almoçar”, “caminhar” e “trabalhar”. Os dados são coletados através de sensores de eletrocardiograma (ECG), respiração, acelerometria, temperatura e PPG, presentes em dispositivos vestíveis.

A Tabela 3.2 apresenta os resultados encontrados pelos autores em cada um dos métodos, considerando modelos tradicionais de aprendizado de máquina, como RF e XGBoost, e o método proposto pelos autores, chamado HDL.

	All-access			**Cross-subject**			**Cross-activity**		
	A	**P+A**	**P+A+E**	**A**	**P+A**	**P+A+E**	**A**	**P+A**	**P+A+E**
RF	94.13% (78.71%)	94.16% (78.25%)	93.95% (77.84%)	88.91% (56.99%)	88.85% (56.7%)	89.13% (58%)	87.23% (50.65%)	87.1% (49.98%)	87.27% (50.83%)
XGBoost	95.82% (84.6%)	95.72% (84.35%)	95.63% (84.02%)	89.62% (59.86%)	90.07% (61.9%)	90.64% (65.09%)	87.92% (53.41%)	88% (54.36%)	88.6% (57.49%)
HHAR-net	92.56% (73.1%)	92.42% (72.39%)	92.45% (71.85%)	88.05% (53.76%)	88.46% (56.11%)	89.27% (58.95%)	86.75% (48.67%)	86.86% (49%)	87.07% (49.80%)
HDL	95.92% (85.95%)	**97.16%** **(90.63%)**	95.74% (86.47%)	91.05% (68.76%)	**92.23%** **(73.48%)**	92.01% (71.91%)	89.79% (61.99%)	**90.18%** **(64.07%)**	89.84% (62.62%)

Tabela 3.2 – Resultados experimentais (Fonte: [34])

O trabalho contribui para o avanço no reconhecimento de atividades humanas, especialmente em contextos de dispositivos vestíveis, demonstrando a eficácia da fusão de dados de sensores inerciais e fisiológicos por meio de uma abordagem HDL.

3.7 Fusion of smartphone sensor data for classification of daily user activities

Şengül et al. [63] desenvolveram um método híbrido de fusão de dados para estimar três tipos de atividades diárias de usuários (“estar em reunião”, “caminhar” e “dirigir um veículo motorizado”), utilizando dados de acelerômetro e giroscópio adquiridos de um *smartwatch* conectado a um *smartphone*. Este método envolve a fusão de características

usando a matriz de séries temporais e a fusão de dados com o algoritmo Better-than-the-Best Fusion (BB-Fus) modificado. O estudo empregou uma abordagem de reconhecimento de padrões estatísticos para a classificação de atividades. Os dados foram adquiridos dos sensores de *smartphones* e pré-processados usando filtro de Kalman. Foram extraídas características, posteriormente fundidas, usando métodos baseados em valor próprio da matriz e o algoritmo BB-Fus. Um aplicativo de celular foi desenvolvido para a aquisição de dados de acelerômetro e giroscópio, utilizando um *smartwatch* como agente de coleta de dados. Além das atividades já citadas, sub-atividades foram definidas para cada atividade principal garantindo a variabilidade dos dados durante a aquisição. Os dados foram coletados de 20 voluntários utilizando o aplicativo móvel desenvolvido. Cada participante gravou dados dos sensores do *smartphone* por 2 minutos em situações reais, sendo que um total de 354 minutos de dados foram adquiridos.

O método proposto pelos autores pode ser visto na Figura 3.4, ilustrando cada etapa realizada até a classificação das atividades e identificação dos resultados.

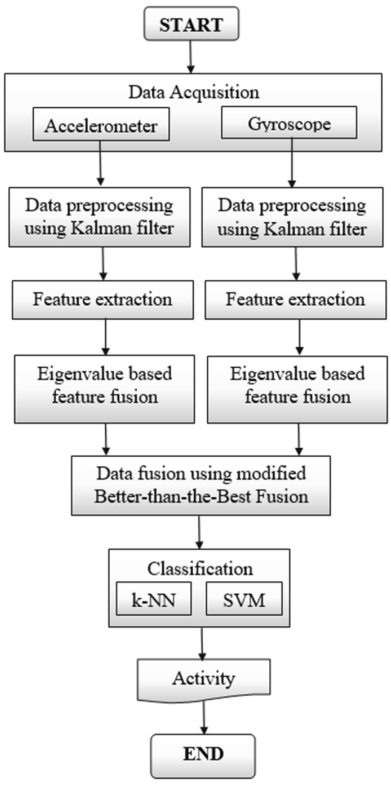


Figura 3.4 – Fluxograma do método proposto (Fonte: [63])

A classificação para todas as atividades foi realizada usando o método proposto, alcançando uma performance de classificação de 98.32% com o classificador SVM e 97.42% com o KNN. As atividades de “reunião-caminhada” e “reunião-transporte” motorizado foram classificadas com taxas de acerto de 98.23% e 99.23% respectivamente quando utilizando dados do acelerômetro. Com dados do giroscópio, atividades de “caminhada-transporte” motorizado foram classificadas com uma taxa de 99.28%.

3.8 Wearable multi-sensor data fusion approach for human activity recognition using machine learning algorithms

Vidya e Sasikumar [72] focam o seu trabalho no reconhecimento de atividades humanas utilizando sensores vestíveis. Eles abordam desafios como dados temporais complexos, identificação de vetores de características discriminativas a partir de dados multimodais e redução de dimensionalidade. O artigo propõe um *framework* de aprendizado de máquina supervisionado para classificação de atividades humanas baseado em análise de tempo-frequência multirresolução do sinal de força de recepção (RSS) entre os sensores vestíveis. O *framework* proposto é apresentado na Figura 3.5; seis atividades foram estudadas, sendo elas: “andar”, “ficar em pé”, “sentar”, “deitar”, “pedalar” e “dobrar”. O estudo utilizou um *dataset* de reconhecimento de atividades disponível no repositório UCI. Esse *dataset* contém sinais de acelerômetros incorporados em sensores vestíveis e uma rede de sensores sem fio (WSN) ambiental. O *dataset* inclui dados de quatro sensores vestíveis e um sensor ambiental.

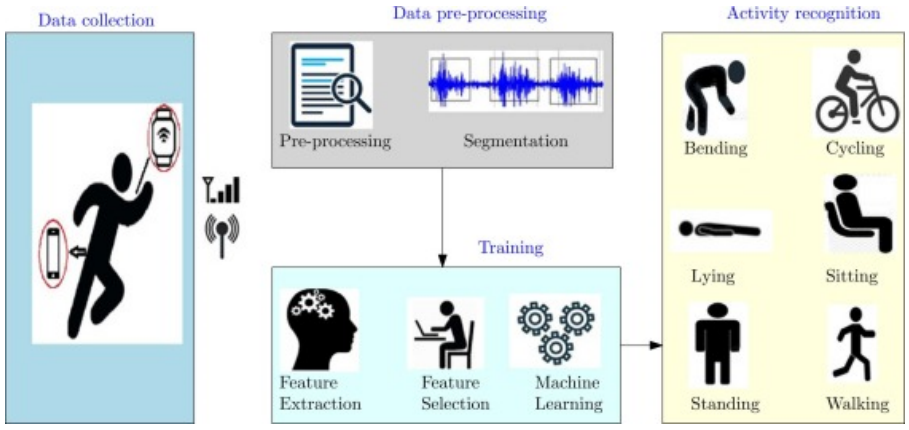


Figura 3.5 – Framework de reconhecimento de atividades diárias proposto (Fonte: [72])

Para os resultados, foram utilizados quatro modelos de classificadores de aprendizado de máquina supervisionados: SVM, KNN, Ensemble Classifier (EC), e DT, sendo que a eficácia do *framework* proposto foi avaliada usando uma matriz de confusão e um gráfico de coordenadas paralelas (PCP). DT teve a menor taxa de classificação incorreta de 6.3%, seguido pelo SVM com 16.7%. O desempenho dos classificadores foi visualizado através do PCP, mostrando que SVM e DT tiveram desempenhos comparáveis com a menor taxa de classificação incorreta entre todas as classes de saída.

3.9 Sensor fusion for recognition of activities of daily living

O trabalho de Jiaxuan et al. [73] propõe o sistema “ADL Recognition System”, que utiliza dados de sensores de um único ponto de contato, como *smartphones*, para processar a fusão de séries temporais de sensores. Este sistema visa perfilar com precisão as ADLs de uma pessoa e descobrir padrões de vida, com um foco particular no cuidado de idosos que vivem de forma independente. As ADLs estudadas no artigo incluem uma ampla gama de tarefas para manter as condições de vida desejadas de uma pessoa, como “comer”, “dormir”, “usar o banheiro”, “tomar banho”, “vestir-se”, “usar o telefone”, “fazer compras”, “preparar refeições”, “fazer tarefas domésticas”, “lavar roupas” e “gerenciar medicações”.

Apesar de não citar um *dataset* específico, o trabalho menciona o desenvolvimento de um aplicativo, o “ADL Recorder App”, que é utilizado para coletar dados de sensores integrados em *smartphones*, como microfone, GPS, detecção de passos, acelerômetro, giroscópio e magnetômetro. Esses dados foram utilizados para o reconhecimento das ADLs. Mais de 30 voluntários participaram da fase inicial de testes do conceito, incluindo pesquisadores, desenvolvedores e interessados no projeto. Os participantes usaram o aplicativo em seus *smartphones* durante dois meses para coletar dados sobre suas atividades diárias.

A conclusão dos autores é de que a fusão de sensores e o processamento de grandes volumes de dados através de técnicas de aprendizado de máquina tornam possível a construção de um sistema automatizado de reconhecimento de ADL. O sistema proposto oferece uma solução abrangente para o reconhecimento de ADLs e a descoberta de padrões ocultos. Os estudos mostraram que o sistema de reconhecimento de ADL foi capaz de observar e classificar com precisão as ADLs em ambientes reais, como campus universitários e residências. As taxas de reconhecimento foram todas acima de 90%.

3.10 A robust human activity recognition system using smartphone sensors and deep learning

Hassan et al. [33] desenvolveram um sistema robusto de reconhecimento de atividades humanas baseado em dados de sensores inerciais de *smartphones*, como acelerômetros e giroscópios. Foram extraídas 561 informações baseadas em métodos de extração de características de sensores inerciais para reconhecimento de atividades humanas, sendo que houve uma redução de dimensão usando uma técnica de Análise de Componentes Principais com Kernel (KPCA).

Para os testes, os autores usaram um *dataset* disponível publicamente na “UCI Machine Learning Repository”, consistindo de 12 atividades físicas, como “caminhar”, “sentar” e “deitar”. O banco de dados incluiu 7767 eventos para treinamento e 3162 eventos para testes, cada um com 561 características básicas. Quanto aos resultados, o sistema proposto, que utiliza DBN (Deep Belief Network) para reconhecimento de atividades, alcançou uma taxa média de reconhecimento de 89.61% e uma precisão geral de 95.85%. Comparado com abordagens tradicionais como SVMs multi-classes, o método mostrou superioridade, com capacidade de distinguir entre atividades transitórias básicas e não transitórias.

A Tabela 3.3 traz os resultados obtidos pelos autores em comparação com testes realizados com SVMs e ANNs (Artificial Neural Networks).

Approach	Total testing samples	Total rightly classified samples	Overall accuracy (%)	Total wrongly classified samples	Overall error (%)
ANN	3162	2816	89.06	346	10.94
SVM	3162	2976	94.12	186	5.88
DBN	3162	3031	95.85	131	4.14

Tabela 3.3 – Resultados obtidos com diferentes algoritmos de Machine Learning (ML) (Fonte: [33])

3.11 Data fusion and multiple classifier systems for human activity detection and health monitoring: review and open research directions

O estudo Nweke et al. [48] aborda a detecção de atividades humanas e monitoramento da saúde usando fusão de dados e sistemas de múltiplos classificadores, focando em dispositivos móveis e vestíveis.

O trabalho oferece uma análise aprofundada de métodos de fusão de dados, modalidades de sensores e algoritmos de aprendizado de máquina, incluindo aprendizado profundo, para detecção eficaz de atividades e monitoramento de saúde.

Além disso, são avaliadas as forças e fraquezas de diferentes modalidades de sensores e métodos de fusão, e discute-se a aplicação dessas tecnologias em monitoramento de saúde, estimativa de gasto energético e atividades diárias. Ainda, é apresentada uma revisão de classificadores base e métodos de design para sistemas de múltiplos classificadores.

3.12 Comparação entre trabalhos relacionados

A Tabela 3.4 agrupa informações cruciais dos trabalhos relacionados a respeito de sensores, metodologia e resultados obtidos, permitindo comparar os demais trabalho com o proposto nesta dissertação.

Tabela 3.4 – Tabela comparativa entre trabalhos relacionados (Fonte: o Autor)

Trabalho	Sensores	Atividades	Sujeitos	Tipo de solução	Modelos de ML
[60]	a, b	A, B, C, D, E, F, G	10	Vestível	N/A
[14]	a, b, c	A, C, H	10	Vestível	7, 8
[43]	a, b	I, J, K, L, M	51	Vestível & Smartphone	1, 2, 3, 4, 5, 6
[66]	a, b	A, B, C, E, H	20	Vestível	1
[38]	a, b, d, e, f, g	A, C, D, E, N, O	60	Vestível & Smartphone	7, 9, 10
[34]	b, h, i	C, E, H, O, P, Q, R, S	15	Vestível	1, 2, 9, 11, 12
[63]	a, b	C, Q, T	20	Vestível & Smartphone	7, 8
[72]	b, j	A, C, D, E, O, U	N/A	Vestível & Smartphone	7, 8, 10, 13
[73]	a, b, d, g	C, E, R, S, V, W, X, Y, Z, Aa, Ab	30	Smartphone	N/A
[33]	a, b, d, g	A, B, C, D, E, G, H	N/A	Smartphone	8, 14, 15
[48]	N/A	N/A	N/A	N/A	N/A

Sensores: a - Giroscópio, b - Acelerômetro, c - EMG, d - Magnetômetro, e - Localização, f - Áudio, g - Sensores comuns de smartphones, h - EDA, i - PPG, j - Sensores de ambiente

Atividades: A - “Ficar em pé”, B - “Levantar-se”, C - “Caminhar”, D - “Deitar na cama”, E - “Sentar em uma cadeira”, F - “Sentar no chão”, G - “Queda”, H - “Descer e subir escadas”, I - “Tomar sopa”, J - “Comer batatas fritas”, K - “Comer massa”, L - “Comer um sanduíche”, M - “Beber de um copo”, N - “Correr”, O - “Pedalar”, P - “Jogar futebol de mesa”, Q - “Dirigir”, R - “Fazer uma refeição”, S - “Trabalhar no computador”, T - “Em uma reunião”, U - “Flexão”, V - “Cozinhar”, W - “Cortar vegetais”, X - “Lavar pratos”, Y - “Usar o banheiro”, Z - “Dar a descarga”, Aa - “Lavar um banheiro”, Ab - “Usar um *smartphone*”

Modelos de *Machine Learning*: 1 - CNN, 2 - LSTM, 3 - BiLSTM, 4 - Stacked LSTM, 5 - CNN-LSTM, 6 - LSTM-CNN, 7 - KNN, 8 - SVM, 9 - RF, 10 - DT, 11 - XGBoost, 12 - HHAR-net, 13 - EC, 14 - ANN, 15 - DBN

O uso de dispositivos vestíveis foi a solução mais comum empregada nos trabalhos relacionados e igualmente no trabalho aqui descrito. Um diferencial principal desta dissertação é a utilização de um *dataset* com um grande número de sensores e de atividades, permitindo explorar com maior profundidade as tecnologias de fusão de sensores. O trabalho proposto optou pela utilização de um conjunto de dados com um número maior de sensores, que facilita os testes de fusão de sensores. Em relação às atividades presentes em cada um dos conjuntos de dados, os trabalhos relacionados trazem atividades diversas, mantendo algumas em comum. Assim como estes trabalhos, o conjunto de dados escolhido aqui possui uma grande quantidade de atividades diferentes. Quanto aos algoritmos utilizados, os trabalhos relacionados trazem modelos, propostas e resultados diferentes. Apenas o trabalho [43] utiliza BiLSTM permitindo comparar seus resultados com os resultados da BiLSTM utilizada no desenvolvimento desta dissertação.

4. CONTEXTUALIZAÇÃO DO TRABALHO

4.1 Arquitetura do sistema LifeSenior

Este trabalho está inserido no contexto do LifeSenior, que teve origem no projeto intitulado “LifeSenior – Monitor contínuo de situações emergenciais”, submetido à “Chamada Pública MCTI/SECIS/FINEP/FNDCT - Viver Sem Limite - 01/2015”, sendo esta submissão fruto de uma cooperação entre a PUCRS e a empresa Toth Lifecare.

A Figura 4.1 descreve abstratamente a primeira arquitetura do sistema de monitoramento de saúde LifeSenior; esta arquitetura inclui um dispositivo vestível (*wearable*), rede e *backend* Sigfox, sistema de nuvem e aplicativo idealizado.

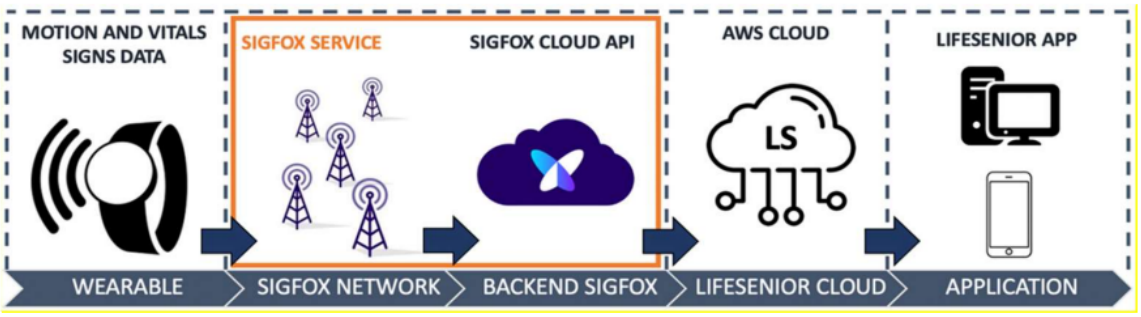


Figura 4.1 – Proposta inicial do sistema de monitoramento de saúde LifeSenior (Fonte: [45])

O Sigfox é uma tecnologia de comunicação sem fio proprietária de baixo consumo e longo alcance que foi desenvolvida para atender a várias redes IoT (Internet of Things) e M2M (Machine to Machine). Foi lançado para permitir a comunicação enquanto consome uma quantidade extremamente baixa de energia. Também é conhecida como rede 0G e foi projetada para a transmissão de mensagens curtas de status de sensores e dados de posição. É uma conexão bidirecional ponta a ponta, comparável a um canal de transporte transparente, que começa com o modem SigFox certificado e termina com o aplicativo baseado na web. Os dados de configuração e as mensagens dos dispositivos terminais e dos sensores são transmitidos ao sistema de TI através deste canal de transporte. No lado da nuvem, uma API (Application Programming Interface) web cuida do gerenciamento de dispositivos terminais, bem como da configuração e integração de dados. A API é baseada em HTTPS e no formato JSON (JavaScript Object Notation). Existe um operador global e, portanto, não é necessário *roaming* [47].

Ao longo do desenvolvimento do LifeSenior, houve uma transição no sistema de comunicação do *wearable* para a nuvem, passando do SigFox para o Bluetooth de Baixa Energia, em inglês Bluetooth Low Energy (BLE). Esta mudança foi impulsionada pela ne-

cessidade de um fluxo de dados mais intenso, em resposta às alterações nas exigências do projeto. Como resultado, a estrutura originalmente apresentada na Figura 4.1 foi modificada para a configuração mostrada na Figura 4.2. Nesta nova arquitetura, foi integrado um sistema de concentração de dados, possivelmente um *smartphone*, que se conecta ao dispositivo vestível via BLE. Esse sistema de concentração, por sua vez, estabelece conexão com a nuvem usando as tecnologias convencionais de telefonia ou Wi-Fi (Wireless Fidelity).

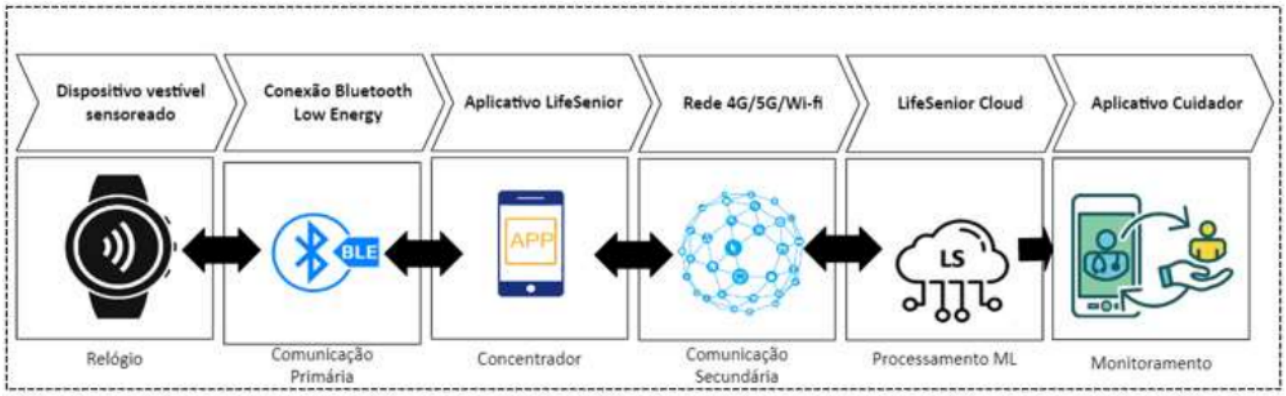


Figura 4.2 – Versão comercial do sistema de monitoramento de saúde LifeSenior (Fonte: [10])

O BLE foi aprimorado para melhorar a taxa de transferência de dados enquanto mantém um consumo extremamente baixo de energia. Esta tecnologia foi desenvolvida para funcionar na faixa de frequência Industrial, Científica e Médica (ISM), que está entre 2.4 e 2.5 GHz. Essa é a mesma faixa de frequência utilizada pelo Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) e pelo Wi-Fi. O BLE se destaca pelo seu consumo de corrente ser, em média, de 1.5 a 2 vezes menor em comparação com outras tecnologias sem fio concorrentes [10].

A troca do Sigfox pelo BLE foi realizada porque o BLE satisfaz a necessidade de baixo consumo de energia, oferece suporte para alta transferência de dados, e está amplamente disponível na maioria dos *smartwatches* e *smartphones* modernos.

4.1.1 Wearable LIFES

O dispositivo vestível LIFES, mostrado nas Figuras 4.1 e 4.2, foi desenvolvido com um design que remete a um relógio convencional. A intenção por trás dessa escolha é coletar uma ampla gama de dados de monitoramento físico em um dispositivo que tenha uma aparência e formato familiar e agradável para o público-alvo.

O LIFES vai além da complexidade de um relógio comum, ele incorpora circuitos especializados que sensoriam sinais vitais. Além disso, o *wearable* possui um sistema

Figura 4.3 – Dispositivo vestível do projeto LifeSenior (Fonte: [45])

que monitora movimentos, projetado para entender a atividade biológica do usuário e identificar circunstâncias perigosas, como uma possível queda.

Dados sobre a movimentação de idosos são constantemente recolhidos através da tecnologia iNEMO da STMicroelectronics [65]. Esta solução integra um acelerômetro de 9 eixos, um giroscópio 3D e um magnetômetro 3D, todos acondicionados em uma única unidade chamada LSM9DS1 [65]. Além disso, o LIFES emprega o sensor ADPD1080 PPG, desenvolvido pela Analog Devices [5], que é responsável pelo processamento periódico dos dados recolhidos do usuário monitorado.

Além das especificações técnicas do LIFES para identificar situações de perigo, é crucial considerar a facilidade de uso do dispositivo. Um ponto de atenção particular é a duração da bateria entre as recargas, um desafio considerável devido às dificuldades enfrentadas pelos idosos no manuseio de componentes eletrônicos. O *wearable* LifeSenior tem como meta um tempo estimado entre cargas de cerca de 30 dias. Isto implica em utilizar componentes com baixo consumo de energia, e bateria com alta capacidade (na versão atual do *wearable*, um lítio recarregável de 100 mAh - bateria de íons) [45].

O dispositivo vestível é equipado com uma tela de *e-paper* para exibição das informações captadas pelos sensores. Esta tecnologia é vantajosa por consumir energia somente durante a atualização da tela, permitindo que informações estáticas permaneçam visíveis sem uso adicional de energia. A versão atual do *wearable* incorpora o *e-paper* modelo ET011TT3, produzido pela E-INK [24]. Este *display* é primordialmente feito de materiais plásticos, e sua capacidade de “desligar” sem perder a visibilidade das informações atende a certos requisitos de comunicação por radiofrequência.

4.1.2 LifeSenior Mobile

A Figura 4.4 mostra interfaces do aplicativo LifeSenior Mobile, que possui modos de login para “Paciente” e “Cuidador”. A Seção “Paciente” inclui funcionalidades apropriadas para o idoso monitorado e deve ser instalada no dispositivo pessoal do usuário, mantendo próximo ao *wearable*. A Seção “Cuidador” exibe dados fisiológicos do idoso e envia notificações, como alertas em caso de queda do idoso. Algumas informações, como a frequência respiratória, cardíaca e movimento, estão disponíveis tanto no modo “Paciente” quanto no “Cuidador”.

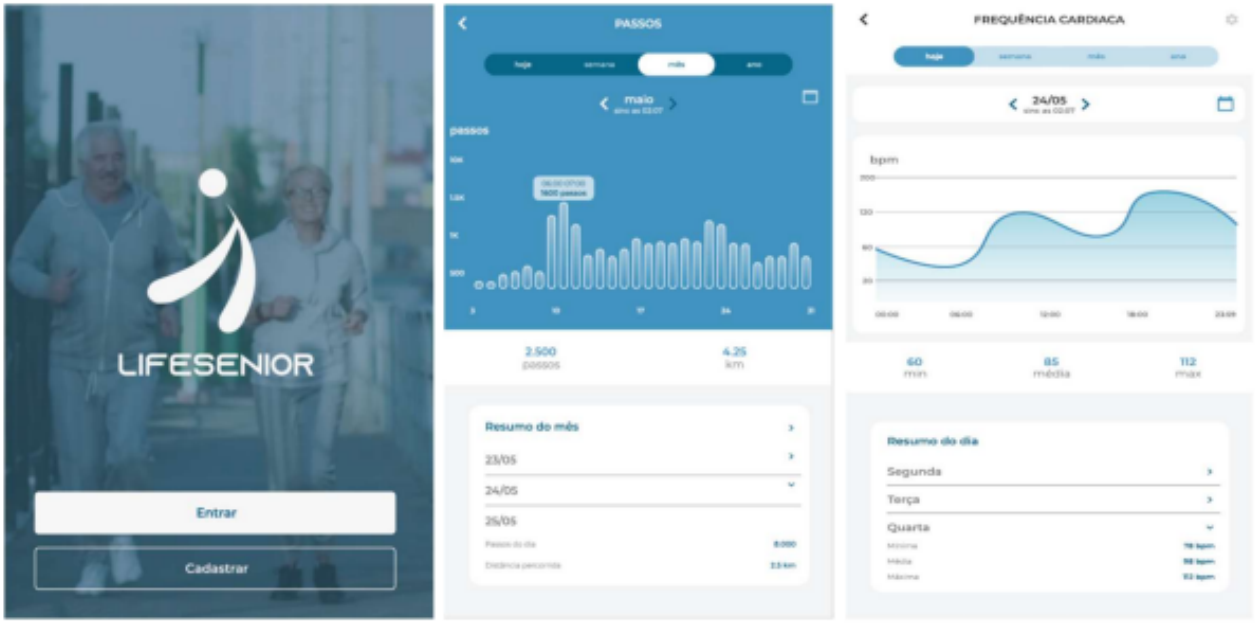


Figura 4.4 – Interfaces do aplicativo LifeSenior Mobile (Fonte: [69])

O aparelho móvel atua como um concentrador, estabelecendo comunicação com o dispositivo vestível por meio da tecnologia BLE. Além disso, o aplicativo LifeSenior Mobile se comunica com o LifeSenior Cloud, permitindo a interação remota com o LIFES.

4.1.3 LifeSenior Cloud

A Figura 4.5 ilustra uma tela do LifeSenior Cloud com o monitoramento parcial, localização e um aviso de detecção de queda de um paciente.

A versão atual do LifeSenior Cloud gerencia as mensagens recebidas de todos os dispositivos vestíveis com um serviço em nuvem fornecido pela Amazon Web Services (AWS) [3]. Qualquer nova mensagem recebida na rede é identificada por um número de dispositivo exclusivo e se correlaciona com a lista de parceiros, redirecionando a mesma

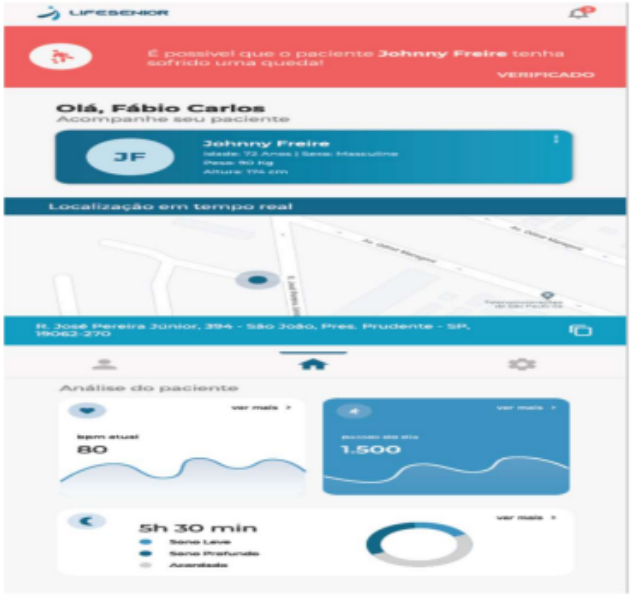


Figura 4.5 – Interfaces do aplicativo LifeSenior Cloud (Fonte: [69])

para a nuvem do cliente. O LifeSenior Cloud armazena todas as mensagens recebidas de cada dispositivo, notificando o aplicativo sempre que necessário.

4.2 LifeSeniorProfile

Como visto, o desenvolvimento deste trabalho colabora com outros trabalhos em desenvolvimento ou concluídos dentro do projeto LifeSenior. Os resultados e conclusões aqui discutidos poderão ser utilizados em futuras pesquisas. Assim como esse trabalho, outro trabalho desenvolvido dentro do projeto, por outros pesquisadores, é intitulado LifeSeniorProfile, um *dataset* de informações multissensoriais, que inclui, além dos tradicionais sensores de movimento, sensores de dados fisiológicos coletados durante atividades diárias e situações de risco. Esse banco de dados foi projetado para analisar mudanças em aspectos vitais causados por atividades diárias e quedas. Estes dados permitem explorar a fusão de aspectos fisiológicos e dados de movimento, e classificar os sensores com base na sua importância na detecção de atividades diárias e quedas, permitindo dar peso a cada elemento detectado [46].

4.3 Simulador LifeSenior

O projeto LifeSenior tem como objetivo final ter um sistema físico funcional, composto pelo dispositivo vestível e demais componentes anteriormente apresentados na Seção 4.1 e detalhados na Figura 4.2. No entanto, para acelerar e validar etapas científicas

do projeto, foi proposta a implementação de um ambiente simulado com os principais componentes que fazem parte do projeto. O ambiente de simulação está ilustrado na Figura 4.6, e permite simular etapas de recepção de eventos provenientes dos sensores do dispositivo vestível (simulador *wearable*), assim como a interpretação desses eventos pelos algoritmos de *machine learning* do concentrador (simulador APP concentrador). Ainda, os comandos provenientes do concentrador podem ser avaliados, assim como os seus efeitos em relação à redução do consumo de energia do *wearable* e a variação dos eventos que são transmitidos do *wearable* até o concentrador.

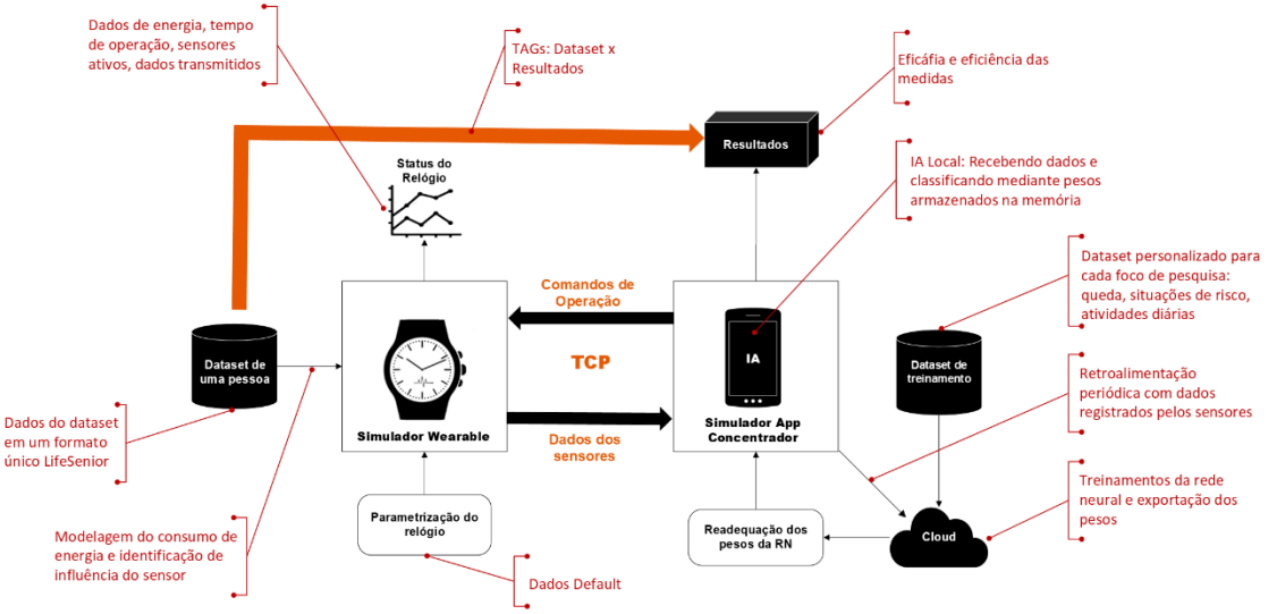


Figura 4.6 – Arquitetura do ambiente de simulação do projeto LifeSenior (Fonte: Projeto LifeSenior)

O simulador *wearable* é alimentado com dados de *datasets* previamente selecionados e padronizados, sendo que sua taxa de amostragem é adequada com os comandos de energia vindos do concentrador. Dessa forma, a taxa será maior em caso de detecção de riscos, e menor para situações normais. Os *datasets* têm informações temporais e o simulador *wearable* pode repassar para o simulador do concentrador os tempos precisos em que os eventos ocorrem, tendo assim a avaliação semelhante ao idoso sendo monitorado. Periodicamente, o concentrador pode se comunicar automaticamente com a LifeSenior Cloud e realizar um novo treinamento com novos dados registrados pelo *wearable*, atualizando os pesos da rede neural.Esse comando também poderá ser feito de forma manual pelo operador do simulador.

O ambiente simulado é composto por duas aplicações desenvolvidas em Python. Uma aplicação implementa o funcionamento do *wearable*: recebe dados dos *datasets* e comunica com o aplicativo concentrador - envio de dados e recepção de comandos. A outra aplicação implementa o concentrador: recebe dados do *wearable* e insere no

algoritmo de *machine learning* que detecta quedas, situações de risco e atividades diárias. Assim que os eventos são detectados, o concentrador envia comandos para o *wearable*.

A comunicação entre o *wearable* e o aplicativo concentrador no ambiente real acontece via BLE, no simulador ela é realizada via protocolo TCP (Transmission Control Protocol). Nos dois módulos é possível acompanhar em tempo real o dado que é enviado/recebido, bem como os comandos. A Figura 4.7 apresenta a primeira versão da tela da aplicação que simula o *wearable*, enquanto que a Figura 4.8 ilustra a primeira versão da tela da aplicação que simula o concentrador.

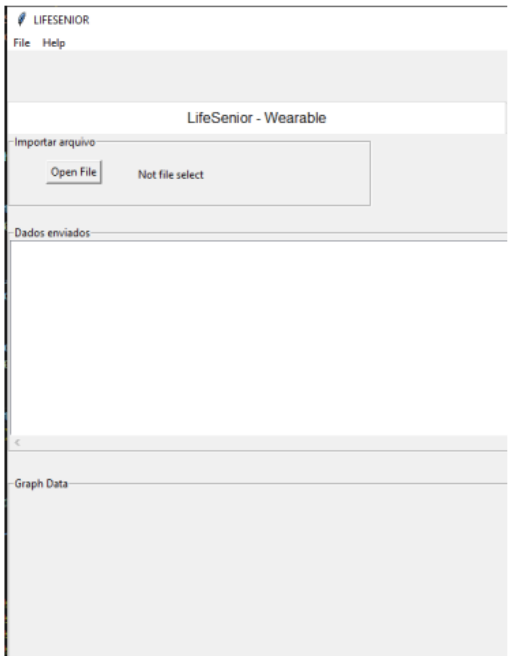


Figura 4.7 – Simulador *wearable* (Fonte: Projeto LifeSenior)

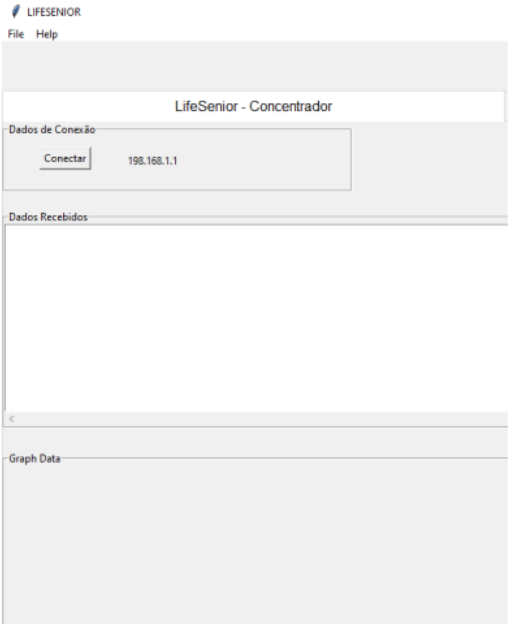


Figura 4.8 – Simulador concentrador (Fonte: Projeto LifeSenior)

4.4 Aplicação da pesquisa realizada no projeto LifeSenior

O projeto LifeSenior, descrito em detalhes no Capítulo 4, utiliza um ambiente simulado para testar rapidamente as soluções que posteriormente serão implementadas fisicamente no relógio e em outros dispositivos. Com base no simulador proposto e apresentado na Seção 4.3, este trabalho implementa a etapa circulada em vermelho na Figura 4.9, que é apresentada no decorrer desse capítulo.

Como é possível verificar na Figura 4.9, existem formas de identificar situações de risco/queda dentro do projeto proposto. A primeira forma, destacada com a cor azul, consiste em avaliar as amostras dos sensores, aplicando técnicas de aprendizado de máquina para identificar situações de risco e/ou queda, chegando em resultados que poderão ser acertos, falsos positivos ou negativos. A segunda forma, marcada com a cor vermelha,

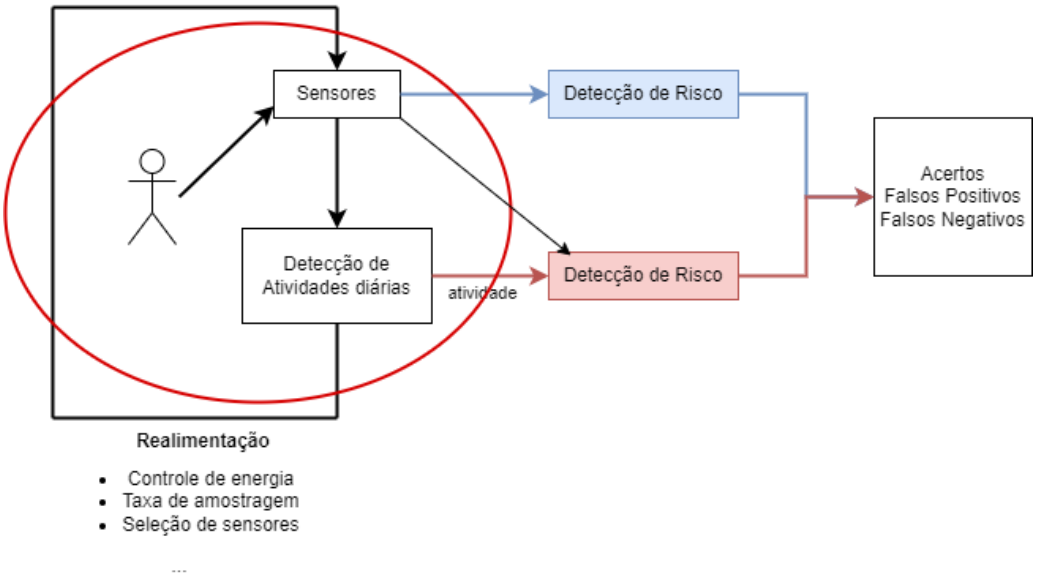


Figura 4.9 – Aplicação do trabalho no projeto LifeSenior (Fonte: o Autor)

parte do princípio de que o tipo de atividade diária afeta na probabilidade de riscos, como uma queda. Dessa forma, utilizamos os mesmos sensores e suas amostras para identificar primeiramente as atividades diárias, e em seguida, a definição de essa atividade ser ou não um risco para o bem-estar do usuário. Na prática, este modo de identificação agrega dois níveis de aprendizado de máquina; o primeiro nível, especializado em detecção da atividade diária, que fornece mais uma entrada para o segundo nível, que é o detecção de risco. Este tipo de abordagem tem maior potencial para aumentar as taxas de acerto de detecção, reduzindo os números de falsos positivos e negativos.

A detecção das atividades diárias possibilita a realimentação de dados no simulador. Em outras palavras, pode ser realizada a seleção dos melhores sensores para identificar determinada atividade, ajudando inclusive na minimização do consumo de energia, visto que uma atividade poderá ter uma taxa de amostragem menor/maior que outra, e os sensores utilizados na identificação de cada atividade serão selecionados seletivamente. Por isso, principalmente, a fusão de sensores para detecção de determinadas atividades diárias se torna uma etapa essencial.

O projeto LifeSenior possui um *dataset* chamado LifeSeniorProfile, que é apresentado no Capítulo 4.2. Esse *dataset* foi criado utilizando os mesmos sensores presentes no relógio utilizado no projeto. Porém, algumas etapas de pré-processamento ainda são necessárias para finalização do *dataset* e, por isso, durante o desenvolvimento do trabalho, optamos por utilizar um *dataset* similar, multissensorial. A fusão de sensores pode ser testada de forma similar, e os resultados podem ser considerados futuramente quando aplicados ao LifeSeniorProfile.

5. ETAPAS DO TRABALHO

Este capítulo descreve as etapas necessárias para alcançar o objetivo geral deste trabalho, incluindo a seleção e refinamento do conjunto de dados e sensores, além da parametrização e aplicação de um modelo de aprendizado de máquina nos dados escolhidos.

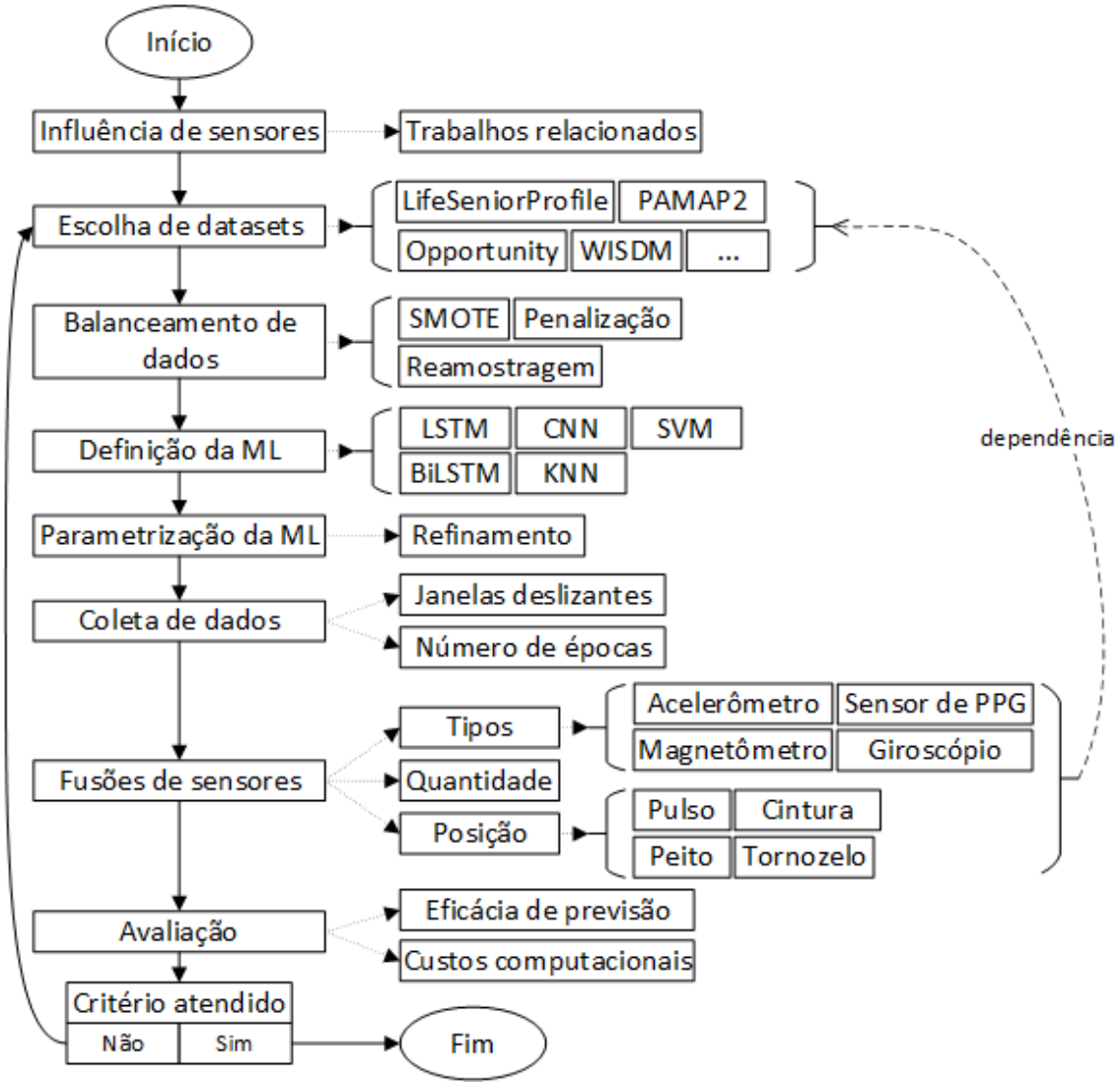


Figura 5.1 – Fluxo aplicado na metodologia (Fonte: o Autor)

A Figura **??** apresenta o fluxo iterativo aplicado na metodologia e na avaliação dos resultados desta dissertação. As setas contínuas representam o fluxo principal; as setas pontilhadas descrevem subatividades; a seta tracejada ilustra a dependência de atividades; e chaves representam conjunto de subatividades (não é necessário que todas foram realizadas).

A primeira etapa é a **influência dos sensores**, que se baseia em trabalhos relacionados e no projeto LifeSenior para definir quais são os sensores estudados e considerados para este trabalho. Após, a etapa de **escolha dos *datasets***, onde as características

principais de cada *dataset* são avaliadas para verificar o que melhor se adéqua aos sensores escolhidos e o seu posicionamento no corpo humano dos sujeitos que compõe o *dataset*. Com os dados em mãos, se torna necessário realizar o **balanceamento dos dados** através do estudo de diferentes abordagens verificando a diferença entre os resultados obtidos na execução de cada uma. A **definição e a parametrização da ML** são as etapas em que modelos de *Machine Learning* são estudados e os parâmetros refinados para definição do modelo que mais se adéqua ao problema que buscamos resolver.

A análise e discussão dos resultados de cada teste está detalhada no Capítulo 6. Com todos os parâmetros definidos, é realizada a definição do número de janelas deslizantes, número de épocas e o melhor ambiente de execução para a **fusão de sensores**, onde são realizados testes com diferentes tipos de sensores e posições no corpo para posterior análise dos resultados obtidos. Por fim, a avaliação dos resultados consiste em verificar a eficácia das previsões, analisando também a quantidade de sensores utilizada e custos computacionais envolvidos. A decisão final de critério atendido ou não se baseia em entender se os resultados são satisfatórios ou se alguma alteração pode ser realizada para buscar resultados melhores; neste caso, parte do fluxo pode ser refeito, como por exemplo, a escolha de um novo *dataset* ou grupo de sensores.

5.1 Motivação na escolha de sensores

Essa seção busca descrever alguns sensores e a sua importância dentro do reconhecimento de atividades diárias.

5.1.1 Acelerômetro triaxial

O acelerômetro triaxial é um dispositivo eletrônico projetado para medir acelerações em três eixos ortogonais: X, Y e Z. Esses dispositivos são fundamentais em várias aplicações industriais, esportivas, de saúde e de pesquisa, pois oferecem uma visão abrangente da dinâmica do movimento em um objeto ou pessoa. *Wearables* e dispositivos médicos utilizam acelerômetros para monitorar atividades físicas, detectar quedas ou avaliar padrões de movimento. Atletas e treinadores utilizam acelerômetros triaxiais para analisar a biomecânica do movimento, melhorando a performance e reduzindo o risco de lesões [70]. Uma das questões que pode ser respondida através do uso de acelerômetros neste trabalho é a importância deles na detecção de atividades que envolvam movimentos como caminhada e corrida, bem como, o impacto da retirada do acelerômetro para identificar os mesmos, quando do uso de outros sensores para realizar a mesma detecção.

5.1.2 Sensor de batimentos cardíacos

Este sensor serve para medir a frequência cardíaca, sendo útil tanto em tratamentos médicos quanto em atividades físicas. Enquanto é realizado um exercício, a musculatura demanda uma necessidade extra de oxigênio, fazendo o coração trabalhar mais. Essa sobrecarga pode ser saudável, desde que o ritmo cardíaco seja mantido dentro de um limite aceitável. O monitor cardíaco ajuda, então, no controle e no conhecimento desses batimentos, demonstrando se o ritmo cardíaco permanece estável ou há algum risco de infarto, por exemplo. Seja durante um tratamento, seja durante uma atividade física, o ritmo do coração é uma informação importante para determinar a saúde do organismo [23]. Por toda essa alteração que uma atividade física pode causar no organismo, nos batimentos do coração, esse trabalho busca estudar também o quanto essa informação ajudará na identificação, não apenas de atividades de corrida e caminhada, mas também atividades mais comuns do dia-a-dia, como por exemplo “dormir”, onde a frequência do batimento cardíaco diminui.

5.1.3 Giroscópio

Um giroscópio é um dispositivo que mede a velocidade angular, ou seja, a velocidade com que um objeto gira. Ele tem aplicações em diversas áreas, desde bússolas em aeronaves até estabilizadores de câmera em *smartphones*. Contudo, os giroscópios tradicionais são frequentemente grandes e pesados, sendo o giroscópio MEMS uma maravilha da engenharia miniaturizada. Em vez de depender de discos rotativos ou rodas, como nos giroscópios tradicionais, os giroscópios MEMS funcionam usando a vibração de microestruturas muito pequenas. Quando o dispositivo que contém o giroscópio gira, essas microestruturas se movem devido à força Coriolis, uma força resultante da rotação. Esta movimentação é então detectada por sensores eletrônicos integrados, que convertem o movimento em um sinal elétrico. Esse sinal pode ser interpretado por um microcontrolador ou outro sistema para determinar a velocidade angular [25]. Normalmente o giroscópio é combinado com outros sensores, como o acelerômetro por exemplo, para identificação de atividades diárias. A avaliação da importância da combinação do giroscópio com o acelerômetro é apenas uma das validações que podem ser realizadas no decorrer do trabalho.

5.1.4 Magnetômetro

Um magnetômetro é um dispositivo projetado para medir a intensidade, direção e, em alguns casos, a variação do campo magnético em uma determinada região. Essa medida é expressa em unidades como Tesla (T) ou Gauss (G), dependendo da escala de medição. O funcionamento de um magnetômetro é intrinsecamente vinculado aos princípios físicos que governam o comportamento dos campos magnéticos. Em termos gerais, um magnetômetro converte a informação magnética em sinais elétricos mensuráveis.

A incorporação de magnetômetros em dispositivos de tecnologia vestível está revolucionando a forma como monitoramos nossa saúde. Em relógios inteligentes e pulseiras, esses sensores podem medir a atividade magnética do corpo, proporcionando informações valiosas sobre padrões de sono, atividade física e até mesmo detectando irregularidades cardíacas. Essa aplicação inovadora tem potencial para melhorar a qualidade de vida e contribuir para a prevenção de problemas de saúde [49].

Quando combinado com acelerômetro e giroscópio, o magnetômetro pode ser um forte aliado na identificação de algumas atividades diárias. Esse trabalho busca qualificar a importância do magnetômetro para a detecção de cada atividade.

5.2 Escolha do *dataset*

Para realizar esse trabalho, buscamos um *dataset* que fornecesse um grande número de amostras com várias classes de atividades diárias correlacionadas aos dados de vários sensores de movimento e fisiológicos. Essas características são fundamentais, visto que esse trabalho foca em avaliar como a fusão de sensores impacta nos resultados obtidos da execução de um modelo de aprendizado de máquina.

A busca do *dataset* se tornou uma tarefa mais difícil do que imaginada, principalmente pelo fato de que muitos dos *datasets* encontrados continham poucos sensores em sua composição, normalmente focando apenas em acelerometria.

A Tabela 5.1 traz informações importantes sobre alguns dos *datasets* encontrados, que foram considerados antes da tomada de decisão pelo *dataset* escolhido. Os dados são organizados nas seguintes colunas: Nome - Identificador do *Dataset*; Dispositivo de coleta - Especifica qual o dispositivo utilizado e/ou qual é a posição do mesmo no corpo; Atividades - Atividades diárias coletadas; e Ano - Data de disponibilização do *dataset*.

Os *datasets* [39, 58, 11] apresentam um cenário que foi muito encontrado durante a busca - o seu principal, ou único, conjunto de sensores está presente em celulares. Além destes, o *dataset* [75] também utilizou um dispositivo semelhante, chamado Moti-

Tabela 5.1 – Resumo de características dos *datasets* pesquisados (Fonte: o Autor)

Dataset	Dispositivo	Sujeitos	Atividades	Ano
WISDM [39]	a	29	A, B, C, D, E, F	2010
Opportunity [59]	f	4	Y	2011
UCI-HAR [58]	a	30	A, C, D, E, F, G	2012
MHEALTH [9]	b, c, d	10	A, B, C, E, F, G, H, I, J, K, L	2014
USC-HAD [75]	e	14	A, B, C, D, E, F, G, H, I, M, N	2012
HHAR [11]	a	9	A, C, D, E, H	2015
Handy [6]	c	30	O, P, Q, R, S, T, U, V X, W	2018
PAMAP2 [57]	b, c, d	9	A, B, E, F, G, H, Z, Aa, Ab, Ac, Ad, Ae, Af, Ag, Ah, Ai	2012

Dispositivos: a - Celular, b - Vestível no Peito, c - Vestível no Pulso, d - Vestível no Tornozelo, e - MotionNode f - Sensores no corpo, em objetos e sensores de ambiente amplamente espalhados

Atividades: A - “Caminhada”, B - “Corrida”, C - “Subir escadas”, D - “Descer escadas”, E - “Sentar”, F - “Ficar em pé”, G - “Deitar-se”, H - “Pedalar”, I - “Pular”, J - “Elevação frontal dos braços”, K - “Cintura inclinada para frente”, L - “Agachar”, M - “Elevador para cima”, N - “Elevador para baixo”, O - “Cortar”, P - “Limpar a mesa”, Q - “Limpar janelas”, R - “Beber água”, S - “Tomar sopa”, T - “Amassar massa”, U - “Usar *tablet*”, V - “Usar *mouse*”, X - “Escrever com caneta”, W - “Escrever com teclado”, Y - “Simulação de atividades em um *studio* contendo cozinha, portas, máquina de café, mesa e cadeira”, Z - “Caminhada nórdica”, Aa - “Assistir TV”, Ab - “Usar um computador”, Ac - “Dirigir”, Ad - “Usar um aspirador de pó”, Ae - “Usar um ferro de passar roupas”, Af - “Dobrar roupas”, Ag - “Limpar a casa”, Ah - “Jogar futebol”, Ai - “Pular corda”

onNode, conectado a um celular, o que leva a um cenário parecido dos anteriores. Esse foi um dos pontos considerados como critério de exclusão, visto que a ideia era encontrar *datasets* onde a coleta tivesse sido feita através de vestíveis de pulso, já que este foi realizado dentro do contexto do projeto LifeSenior, que desenvolveu o *dataset* LifeSeniorProfile descrito na Seção 4.2 e que utiliza vestíveis de pulso em outras pesquisas também.

O conjunto de dados Opportunity [59] foi considerado e até foi utilizado para alguns estudos iniciais. No entanto, optamos não utilizar o mesmo, visto que ele possui uma ampla gama de sensores, incluindo sensores de parâmetros ambientais e de objetos, como também vestíveis posicionados e diversas partes de corpo humano. Além disso, o *dataset* envolve atividades realizadas dentro de casa, em um ambiente que simula um apartamento. Nesse caso, o critério de exclusão foi com base em procurar atividades diárias comuns, realizadas em diversos ambientes, e também atividades físicas realizadas

fora de casa. Adicionalmente, em relação a grande quantidade de sensores presente no conjunto de dados, haveria uma etapa considerável de pré-processamento para remover informações e utilizar apenas aquelas que fossem úteis para o trabalho. No entanto, esse foi um dos *datasets* mais completos encontrados durante a pesquisa, e seu uso em trabalhos futuros deve ser considerado.

Dois *datasets* [9, 6] se aproximaram bastante do cenário ideal para o trabalho, tendo estes o seu conjunto de dados construído com base em dispositivos vestíveis de coleta, presos no pulso e/ou peito do participante, além de ter um conjunto de atividades realizadas internamente e/ou externamente, incluindo algumas atividades físicas ou esportes. Esta dissertação optou utilizar o *dataset* PAMAP2 [57], que supre os requisitos buscados e está disponível publicamente, sendo composto por sensores presos em três partes do corpo do participante, com um grande número de atividades diárias, além de possuir o diferencial de ter o sensor de batimentos cardíacos. O Apêndice A traz mais informações sobre o *dataset*, além de um resumo dos dados e colunas presentes no mesmo.

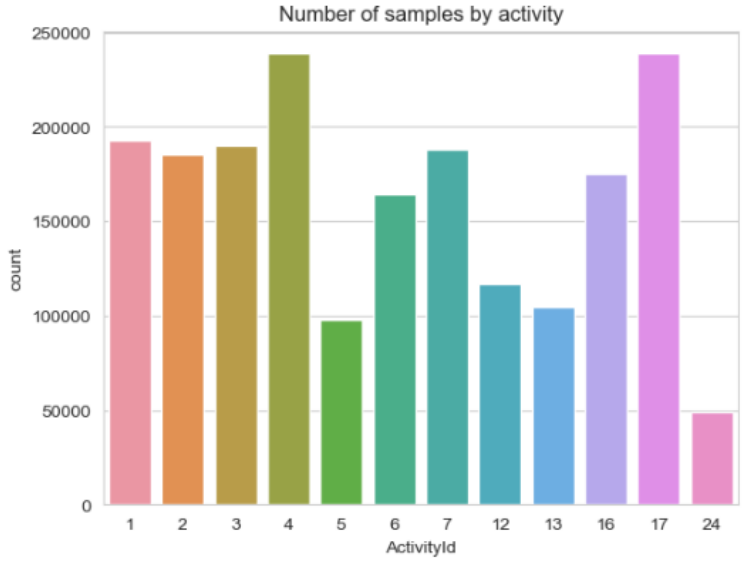
5.3 Pré-processamento dos dados

Na etapa de pré-processamento, buscamos entender como os dados estavam presentes no *dataset* e qual a melhor forma de organizá-los para seu posterior uso. Além disso, houve a necessidade de aplicar algumas técnicas visto que o *dataset* estava desbalanceado, assunto que também será discutido nesta seção.

O conjunto de dados escolhido é constituído por nove arquivos *.dat*, cada um correspondendo a informações dos sensores utilizados em um participante da coleta de dados. A primeira etapa juntou todos os arquivos em um só *dataframe* Python, possibilitando ter todos os dados em um só conjunto utilizado durante a execução de todo o algoritmo, evitando a leitura de vários arquivos.

Com esses dados no *dataframe*, foi realizada uma nomeação de colunas, para facilitar a identificação durante a execução do algoritmo, sem a necessidade de consultar frequentemente a documentação do *dataset*. Posteriormente, alguns dados foram desconsiderados, visto que a documentação traz a informação de que dados de orientação, e dados relacionados à atividade que possui o identificador 0 na coluna que identifica a atividade diária, devem ser considerados inválidos para qualquer análise. Ainda, considerando a diferença de frequência entre os dispositivos utilizados, algumas células do *dataset* trazem dados vazios ou NaN (Not a Number). Para esse caso, foi usada a estratégia de propagar o último valor válido para a próxima amostra, utilizando a função *fillna* da biblioteca *pandas* do Python.

Após o pré-processamento do *dataset*, foi realizada uma análise gráfica dos dados de cada classe. Essa análise, apresentada na Figura 5.2, mostra a grande disparidade entre dados de cada atividade.



Atividades: 1 - “Deitado”, 2 - “Sentado”, 3 - “Em pé”, 4 - “Caminhando”, 5 - “Correndo”, 6 - “Pedalando”, 7 - “Caminhada nórdica”, 12 - “Subindo as escadas”, 13 - “Descendo as escadas”, 16 - “Usando um aspirador de pó”, 17 - “Passando roupas”, 24 - “Pulando corda”

Figura 5.2 – Relação de quantidade de amostras x atividade do *dataset* PAMAP2 (Fonte: o Autor)

Enquanto atividades como “caminhada” (4) e “passar roupa” (17) possuem quase 250.000 amostras cada, a atividade de “pular corda” (24) traz apenas cerca de 50.000 amostras. Qualquer execução de modelo utilizando um *dataset* desbalanceado como esse, pode acabar levando em conta o número superior de dados de uma atividade específica, trazendo um viés nos resultados.

Conforme o Google for Developers [30], um conjunto de dados de classificação com proporções de classes distorcidas é chamado de desbalanceado. As classes que compõem uma grande proporção do conjunto de dados são chamadas de classes predominantes. Aquelas que compõem uma proporção menor são classes minoritárias. Ainda, o modelo de treinamento passará a maior parte do tempo em amostras de classes predominantes e não aprenderá o suficiente com amostras de classes minoritárias.

A Seção 5.4 traz informações sobre técnicas que podem ser utilizadas para *datasets* desbalanceados, além de especificar a técnica escolhida para esse trabalho e o motivo da sua escolha.

Visto que o trabalho busca analisar a fusão de sensores e quais os resultados encontrados com essa fusão, é necessário treinar o modelo utilizando apenas algumas colunas do *dataset*. Como as colunas tiveram nomes previamente definidos, a escolha

destas pode ser feita rapidamente utilizando a função *Dataframe.loc* da biblioteca *pandas*, que permite selecionar as colunas desejadas através de seu nome.

Para reescalar os dados de entrada, foi utilizada a função *StandardScaler* da biblioteca *sklearn*. Essa etapa é importante já que os dados são provenientes de diferentes sensores, e por isso, podem ter escalas desproporcionais que, se utilizadas sem o pré-processamento, podem impactar negativamente nos resultados do treinamento e aplicação do modelo.

A função *StandardScaler* calcula a média das amostras de treinamento (u) e o desvio padrão das amostras de treinamento (s) de cada amostra (x) no conjunto de dados de treinamento [62]. As Figuras 5.3 e 5.4 apresentam, respectivamente, os valores antes e depois da aplicação da função *StandardScaler* em uma matriz de dados.

$$z = (x - u)/s$$

Aplicar a função *StandardScaler* aos dados garante que cada característica contribua proporcionalmente para a análise final, evitando que variáveis com maior magnitude dominem aquelas com menor magnitude [62].

```
[[25, 5000],
 [30, 6000],
 [35, 7000],
 [40, 8000],
 [45, 9000]]
```

Figura 5.3 – Exemplo de matriz antes da aplicação da função *StandardScaler* (Fonte: o Autor)

```
[[-1.41421356, -1.41421356],
 [-0.70710678, -0.70710678],
 [ 0.        ,  0.        ],
 [ 0.70710678,  0.70710678],
 [ 1.41421356,  1.41421356]]
```

Figura 5.4 – Exemplo de matriz após da aplicação da função *StandardScaler* (Fonte: o Autor)

Além disso, utilizando a função *toCategorical* da biblioteca *keras* os valores inteiros que representam diferentes categorias e que estão presentes em uma coluna pré-selecionada do *dataset*, foram convertidos em uma matriz com valores binários e com

a quantidade de colunas igual a quantidade de categorias dos dados. Esse tipo de codificação é comumente chamada de *one-hot* e oferece vantagens como: **Melhoria na performance do modelo** - ao representar as classes de forma binária, evita que o modelo interprete os identificadores como tendo uma ordem ou hierarquia natural entre eles; **Prevenção de viés de ordem** - Sem a conversão para codificação *one-hot*, o modelo poderia inferir erroneamente uma relação de ordem entre as classes (por exemplo, supondo que a classe 2 é de alguma forma "superior" a classe 1); **Facilita a interpretação de resultados** - melhora a classificação, selecionando a classe prevista sem a necessidade de mapear números inteiros de volta para rótulos de classe [53].

5.4 Aplicação de técnicas para *datasets* desbalanceados

Datasets desbalanceados se mostraram mais comuns do que o esperado; é difícil ter *datasets* com a mesma quantidade de amostras por classe, visto que algumas atividades diárias são mais facilmente executadas, ou tendem a ser realizadas em tempos diferentes. Com isso, o trabalho de treinar um modelo sem que haja um viés para as classes majoritárias é um desafio. Algumas técnicas podem ser aplicadas, nesse caso, para pré-processar os dados, criando amostrar sintéticas, adicionado pesos ou até mesmo removendo dados. Essa seção apresenta algumas dessas técnicas que foram testadas e as quais os resultados serão apresentados no Capítulo 6.

5.4.1 SMOTE - Synthetic Minority Over-sampling Technique

SMOTE [15] é uma técnica utilizada em *datasets* desbalanceados que propõe a geração de um certo número de amostras, de modo que a distribuição entre classes minoritárias e *majoritárias* fique equilibrada. Uma das suas vantagens é evitar a perda de dados já que nenhuma amostra precisa ser removida. As amostras geradas são similares às amostras já existentes, com pequenas variações. O seu funcionamento consiste em selecionar aleatoriamente um exemplo da classe minoritária, e encontrar seus k vizinhos mais próximos de mesma classe. Em seguida, um exemplo sintético é criado em um ponto selecionado aleatoriamente nessa linha que conecta os dois exemplos anteriormente encontrados.

A Figura 5.5 mostra um exemplo de como esse processo acontece. O SMOTE é apenas aplicado em dados de treino, para que, ao realizar a execução do modelo em dados de teste, os resultados sejam reais e não baseados em amostras geradas. SMOTE não é uma técnica que tende a se comportar bem para um conjunto de dados muito grande, pois demanda muito processamento.

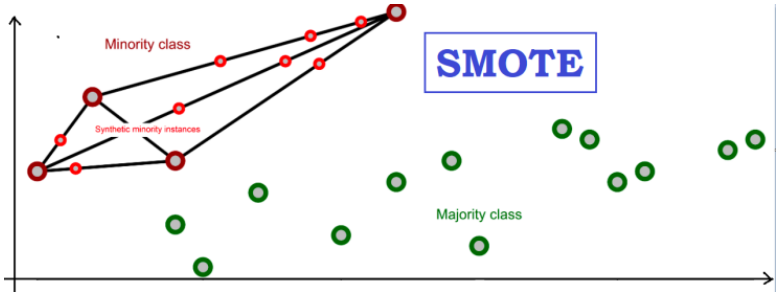


Figura 5.5 – Exemplo gráfico de criação de amostras utilizando SMOTE. A classe Minority, representada pelos quatro círculos vermelhos grandes agrega mais oito instâncias (círculos vermelhos menores), de forma ao número de círculos vermelhos ser próximo dos 13 círculos verdes. - (Fonte: [13])

5.4.2 Penalização

A penalização consiste em definir pesos para classes ou para amostras, durante a fase de treinamento. Na utilização de pesos para classes, por exemplo, o objetivo é penalizar o erro nas classes minoritárias, estabelecendo um valor de peso mais elevado para estas, da mesma forma que será definido um peso menor para as classes majoritárias. Esse peso elevado para as classes minoritárias fará com que o algoritmo foque em reduzir os erros para estas classes. A maioria das bibliotecas de definição de modelo tem um parâmetro para definição do peso de cada classe ou para balancear automaticamente todas as classes [64].

5.4.3 Reamostragem

A técnica de reamostragem consiste em remover ou adicionar dados ao *dataset*. Quando se trata de adicionar dados (*oversampling*), a técnica consiste em adicionar cópias de amostras das classes minoritárias, até que a distribuição dos dados fique igualitária e/ou equilibrada. A duplicação excessiva de dados da mesma classe, pode levar ao *overfitting* [13].

Segundo a Amazon [4], o *Overfitting*, “ajuste excessivo” em tradução literal, é um comportamento indesejável de aprendizado de máquina que ocorre quando o modelo de aprendizado de máquina fornece previsões precisas para dados de treinamento, mas não para novos dados. Quando os cientistas de dados usam modelos de aprendizado de máquina para fazer previsões, eles primeiro treinam o modelo em um conjunto de dados conhecido. Em seguida, com base nessas informações, o modelo tenta prever resultados para novos conjuntos de dados. Um modelo de *overfitting* pode fornecer previsões imprecisas e sem um eficácia para todos os tipos de novos dados.

Undersampling é uma técnica que consiste em remover dados das classes majoritárias, deixando, da mesma forma, a distribuição dos dados mais equilibrada. Como dados serão removidos, o uso dessa técnica pode acelerar o processo de treino/teste, mas pode causar a perda de informações importantes. A execução da técnica pode ser feita de duas formas: **remoção aleatória** - a maneira mais fácil e rápida de equilibrar os dados selecionando aleatoriamente algumas amostras da classe majoritária; **NearMiss** - utiliza a menor distância média dos K-vizinhos mais próximos, ou seja, seleciona os valores baseando-se no método KNN para reduzir a perda de informação. As Figuras 5.6 e 5.7 exemplificam como a aplicação do *oversampling* e do *undersampling*, respectivamente, irão impactar nos dados de cada classe [13].

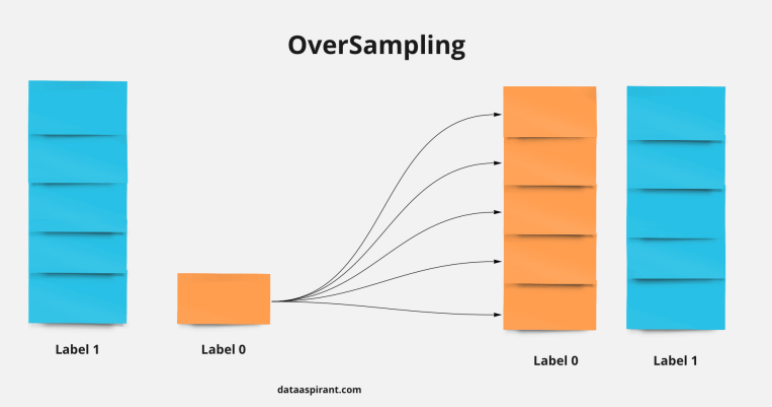


Figura 5.6 – Exemplificação de *oversampling*. Os dados de rótulo “Label 0” são replicados de forma a alcançar o mesmo número de amostras dos dados de rótulo “Label 1”. - (Fonte: [13])

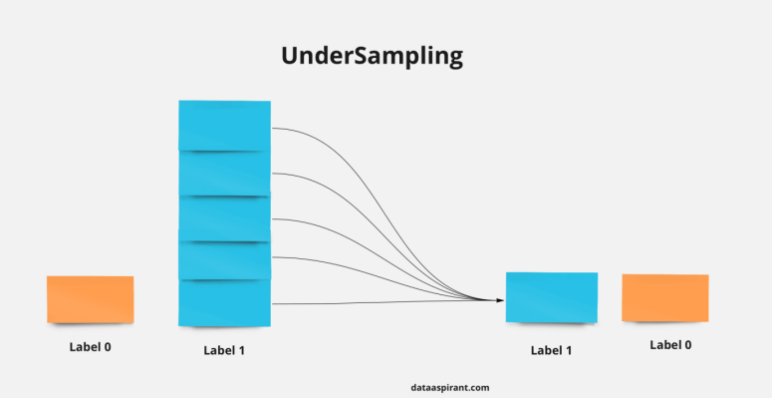


Figura 5.7 – Exemplificação de *undersampling*. Os dados de rótulo “Label 1” são reduzidos de forma a alcançar o mesmo número de amostras dos dados de rótulo “Label 0”. - (Fonte: [13])

5.5 Escolha do modelo de ML

A escolha do modelo de ML mais adequado envolve a combinação de fatores relacionados ao problema específico, aos dados, recursos computacionais e experiência disponíveis. A experimentação e a validação são componentes críticos desse processo.

Durante a pesquisa e desenvolvimento deste trabalho, alguns modelos foram encontrados na lista de trabalhos relacionados. Essa seção descreve brevemente estes modelos.

5.5.1 Convolutional Neural Network (CNN)

Uma CNN é um algoritmo de Aprendizado Profundo que pode captar uma imagem de entrada, atribuir importância (pesos e vieses que podem ser aprendidos) a vários aspectos/objetos da imagem e ser capaz de diferenciar um do outro. O pré-processamento exigido em uma CNN é muito menor em comparação com outros algoritmos de classificação. Enquanto nos métodos primitivos os filtros são feitos à mão, com treinamento suficiente, as CNNs têm a capacidade de aprender esses filtros/características. A arquitetura de uma CNN é análoga àquela do padrão de conectividade de neurônios no cérebro humano e foi inspirada na organização do Visual Cortex. Os neurônios individuais respondem a estímulos apenas em uma região restrita do campo visual conhecida como Campo Receptivo. Uma coleção desses campos se sobrepõe para cobrir toda a área visual [21].

Uma CNN é capaz de capturar com sucesso as dependências espaciais e temporais em uma imagem através da aplicação de filtros relevantes. A arquitetura executa um melhor ajuste ao conjunto de dados da imagem devido à redução no número de parâmetros envolvidos e à capacidade de reutilização dos pesos. Em outras palavras, a rede pode ser treinada para entender melhor a sofisticação da imagem [21].

Na Figura 5.8, há uma imagem RGB (Red, Green, Blue) que foi separada por seus três planos coloridos – Vermelho, Verde e Azul. Existem vários desses espaços de cores nos quais existem imagens – Escala de cinza, RGB, HSV (Hue Saturation Value), CMYK (Cyan, Magenta, Yellow, and Key), etc. A função da CNN é reduzir as imagens para uma forma mais fácil de processar, sem perder recursos que são críticos para obter uma boa previsão [21].

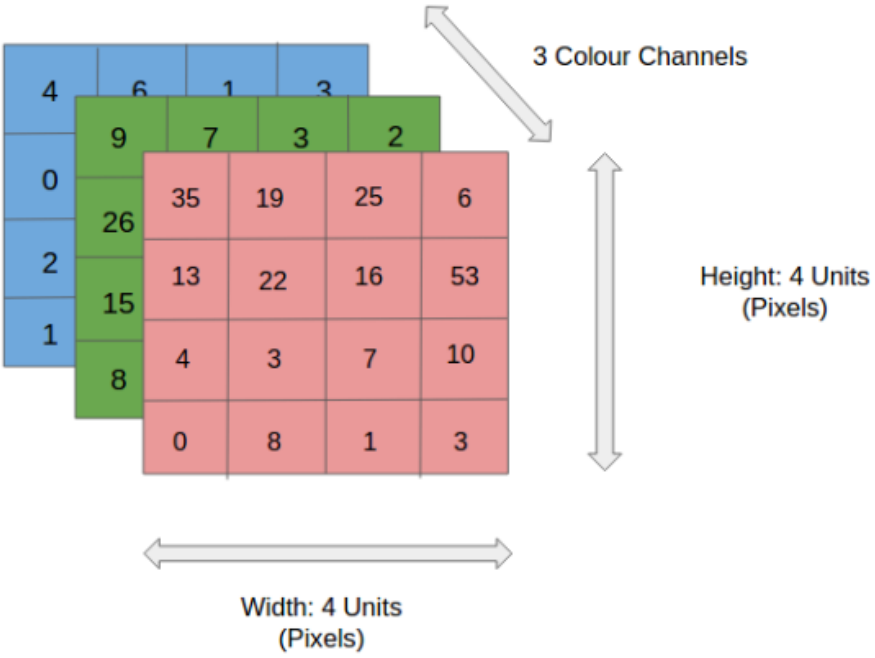


Figura 5.8 – Exemplo de imagem RGB (Fonte: [21])

5.5.2 K-nearest neighbors (KNN)

KNN é um algoritmo muito utilizado em problemas de classificação. Ele classifica cada amostra de um conjunto de dados avaliando sua distância em relação aos vizinhos mais próximos. Se os vizinhos mais próximos forem majoritariamente de uma classe, a amostra em questão será classificada nesta categoria. No exemplo da Figura 5.9, intuitivamente é possível acreditar que a amostra que ainda não está classificada (verde), pertence à classe vermelha. No entanto, o algoritmo não possui “intuição”, e dessa forma, realizará um cálculo matemático para definir a solução [22].

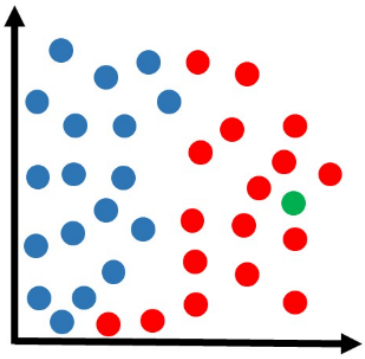


Figura 5.9 – Amostra antes da classificação utilizando KNN (Fonte: [22])

Utilizando o método KNN com k = 3, sendo k o número de vizinhos a ser avaliado, nesse caso, a amostra seria classificada dentro da classe vermelha. A Figura 5.10 traz a definição da solução para esse caso.

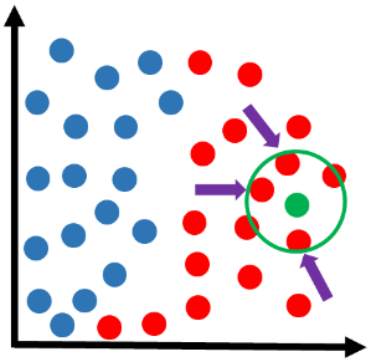


Figura 5.10 – Amostra classificada utilizando KNN com k = 3 (Fonte: [22])

5.5.3 Support Vector Machine (SVM)

SVM é um algoritmo de aprendizado supervisionado muito utilizado para classificar dados em grupos diferentes — sendo também utilizado para regressão. O algoritmo busca uma linha de separação entre duas classes distintas analisando os dois pontos, um de cada grupo, mais próximos da outra classe. Isto é, SVM escolhe a reta — também chamada de hiperplano em maiores dimensões — entre dois grupos que se distancia mais de cada um. A Figura 5.11 apresenta uma linha vermelha ilustrando a separação de duas classes, nesse caso, a reta é a mais distante entre os dois grupos, considerando apenas os pontos de cada grupo mais próximos da reta [18].

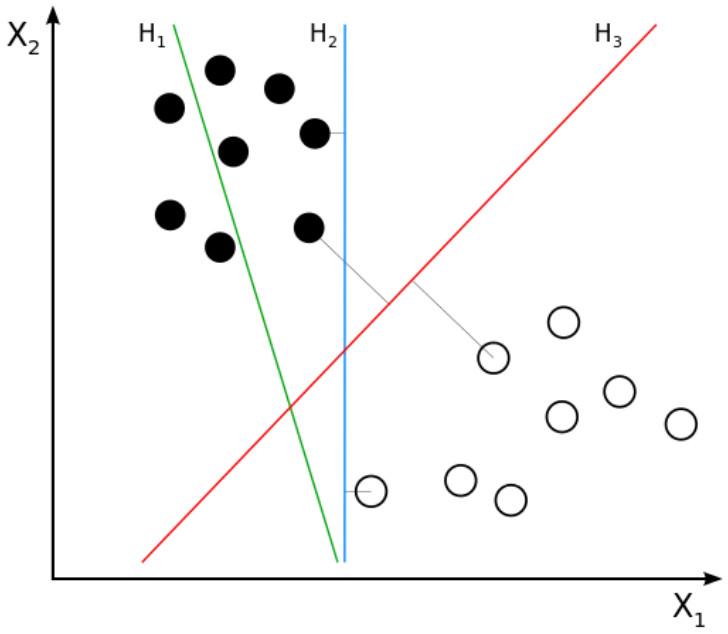


Figura 5.11 – Exemplo de uma reta ótima separando dois grupos (Fonte: [18])

Quando é necessário classificar os dados em mais de duas classes, é possível dividir o problema multiclasse em várias classificações binárias, que podem ser *one vs*

one ou *one vs all* (também chamada de *one vs rest*). No *one vs one*, se existirem três classes (A, B, C) por exemplo, as comparações (A, B), (A, C) e (B, C) serão realizadas, e a classe mais votada será escolhida. Já *one vs all* consiste em comparar cada classe com todo o resto, ou seja, as mesmas três classes A, B e C serão divididas nas seguintes comparações (A, B+C), (B, A+C) e (C, A+B) [18].

Existem grupos que não podem ser separados somente por hiperplanos. Nesse caso, pode ser utilizada uma SVM não-linear para delimitar as duas classes, que traçará uma ou mais linhas retas ou curvas para separar as classes da melhor forma possível. O algoritmo primeiro faz uma transformação não-linear do espaço para depois separar os grupos com um SVM linear. A Figura 5.12 exemplifica esse processo.

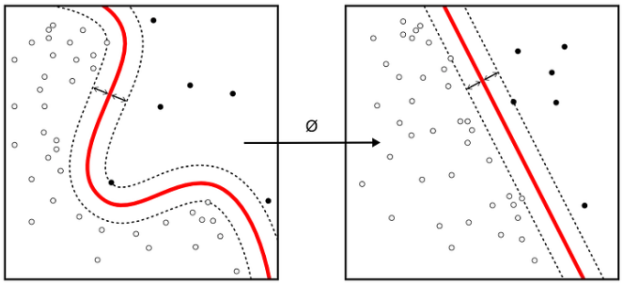


Figura 5.12 – Exemplo de transformação não-linear entre o espaço das entradas (à esquerda) e o espaço das *features* (à direita) - (Fonte: [18])

Esse modelo não requer conhecer a base de dados para conseguir uma predição com boa acurácia. Além disso, o SVM funciona bem em espaços com muitas dimensões. Contudo, o resultado do SVM é dificilmente interpretável (mas possível) e, conforme o tamanho do *dataset* vai aumentando, o tempo necessário para fazer cálculos cresce muito rapidamente [18].

5.5.4 Long Short-Term Memory (LSTM)

LSTM, em português memória de curto e longo prazo, é um tipo especial de rede neural recorrente, capaz de aprender conexões em sequências de informação. Dentre as principais aplicações das redes neurais LSTM estão os processamentos de (i) tarefas de linguagem natural; (ii) áudio; e (iii) sequência de *frames* de vídeo. LSTMs, assim como todas as arquiteturas de redes neurais artificiais, são bioinspiradas; ou seja, funcionam de forma análoga a um processo biológico. A arquitetura de uma LSTM consiste em um conjunto de células de memória conectadas de forma recorrente [52]. A Figura 5.13 apresenta uma célula (ou bloco) de memória de uma LSTM. Esses blocos retêm e manipulam informações por meio de portões (gates) que controlam a passagem das informações entre os blocos [20]. As LSTMs possuem três tipos de portões:

- ***Forget gate* (portão do esquecimento)** - decide as informações que devem ser descartadas; duas entradas: x_t (entrada no momento específico) e h_t-1 (saída de célula anterior) são alimentadas ao *gate* e multiplicadas por matrizes de peso, seguidas pela adição do *bias*. O resultado é passado por uma função de ativação, com saída binária. Caso a saída seja 0, a informação é esquecida. Para 1, a informação é mantida para uso futuro.

- ***Input gate* (portão de entrada)** - adiciona informações ao estado da célula de memória; já o *Input Modulation Gate* pode ser considerado como uma parte do *Input Gate*, sendo que o mesmo é utilizado para modular a informação que será repassada a célula de memória.

- ***Output gate* (portão de saída)** - decide quais informações úteis do bloco de memória devem ser levadas para o próximo bloco [52].

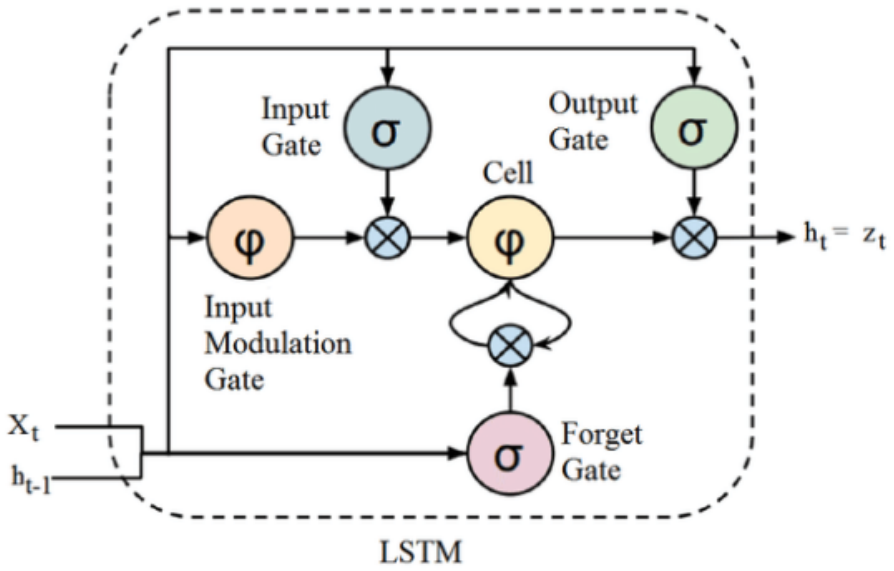


Figura 5.13 – Bloco de memória LSTM (Fonte: [52])

Recentemente, abordagens de aprendizado de máquina profundo ganharam uma quantidade significativa de interesse de pesquisa. Zebin et al. [74] trabalharam em séries temporais com sensores inerciais para reconhecer atividades com rede LSTM recorrente. Ordonez et al. [50] apresentaram uma rede convolucional e uma rede neural recorrente LSTM para reconhecimento de atividades usando dados de sensores vestíveis. Eyobu et al. [27] propuseram uma abordagem com representação de características e aumento de dados para classificação de atividades baseadas em dados vestíveis de uma IMU usando uma rede neural profunda do tipo LSTM.

5.5.5 Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM, em português LSTM bidirecional, é um tipo de rede neural recorrente usada principalmente no processamento de linguagem natural. Ao contrário da rede LSTM padrão, a entrada flui em ambas as direções e consegue utilizar informações de ambos os lados. A BiLSTM é uma ferramenta poderosa para modelar as dependências sequenciais entre palavras e frases em ambas as direções da sequência [7].

A BiLSTM adiciona mais uma camada LSTM, que inverte a direção do fluxo de informações; isso significa que a sequência de entrada flui para trás na nova camada LSTM. Adicionalmente, as saídas de ambas as camadas LSTM são combinadas de várias maneiras, como média, soma, multiplicação ou concatenação [7].

Taylor et al. [67] testaram algoritmos de aprendizado de máquina, dentre eles um baseado em BiLSTM para classificação de atividades diárias de pessoas idosas usando dados de um radar micro-Doppler. A Figura 5.14 exemplifica o funcionamento da rede através de uma representação visual. O exemplo apresenta uma classificação baseada em imagens, sendo que a mesma está sendo transformada em *arrays* de dados. Esses dados são utilizados como entrada para a rede BiLSTM que executa o algoritmo de LSTM de duas maneiras. A primeira maneira é a tradicional em uma rede LSTM: analisar os dados antigos para chegar na previsão futura. A segunda maneira é inversa, ou seja, analisar os dados futuros como base nos resultados antigos; o mesmo processamento, mas de maneira inversa. Todas essas camadas ocultas são inteiramente conectadas em uma camada oculta final, que faz a classificação dos dados.

5.5.6 Definição do modelo para este trabalho

O trabalho realizado utiliza o modelo BiLSTM apresentado na Seção 5.5.5, devido à sua capacidade de capturar informações em ambas as direções. Além disso, LSTMs bidirecionais têm sido usadas para modelar uma ampla variedade de problemas, desde processamento de linguagem natural, reconhecimento de fala, classificação de sons e previsões de séries temporais. Adicionalmente, BiLSTM é uma técnica poderosa para capturar padrões temporais e sendo generalizada para dados não vistos, i.e., novos dados que um modelo de aprendizado de máquina não encontrou durante sua fase de treinamento. Estes dados representam instâncias reais ou hipotéticas que podem ser usadas para testar a capacidade do modelo de generalizar o que aprendeu a partir do conjunto de treinamento para situações novas e desconhecidas. [29].

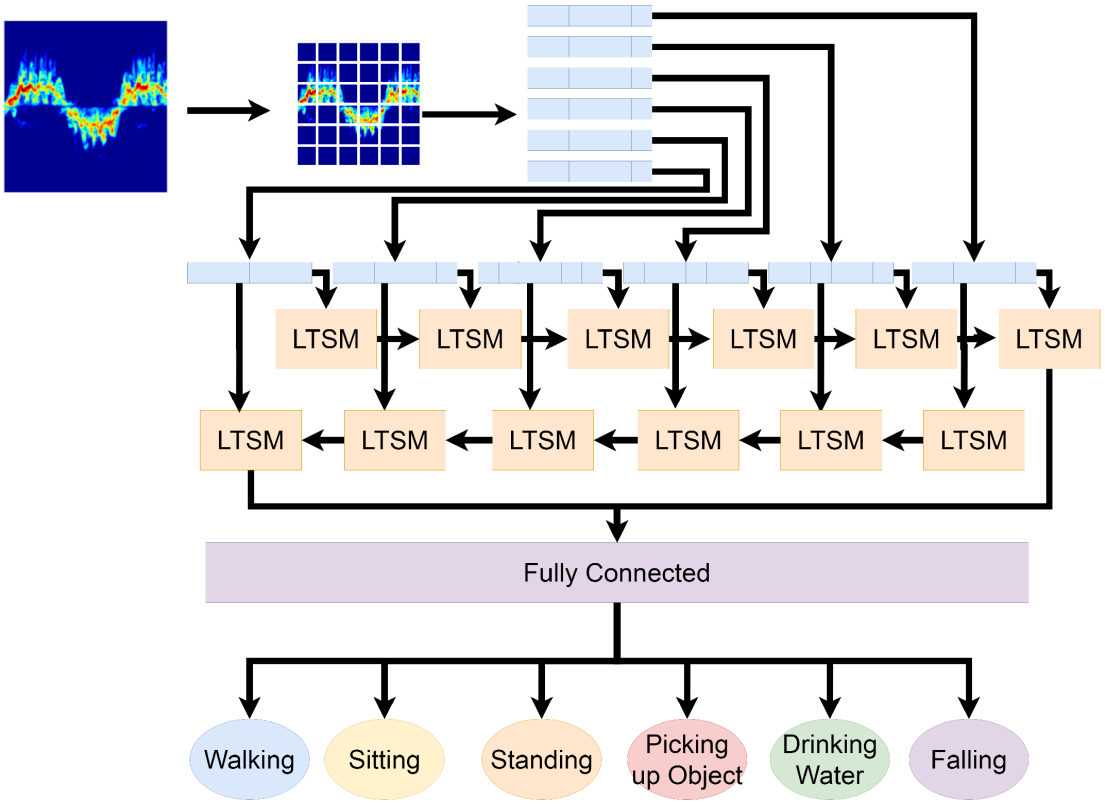


Figura 5.14 – Representação visual do funcionamento de uma BiLSTM, que no exemplo é usada para classificar sete atividades diárias, como “Caminhando” (*Walking*) e “Sentando” (*Sitting*) (Fonte: [67])

Por fim, trabalhos desenvolvidos dentro do projeto LifeSenior [45] já utilizaram LSTM em suas avaliações, por isso, o uso de BiLSTM colabora com esses trabalhos, facilitando comparações e introduzindo o uso da LSTM em ambas as direções.

5.6 Definição dos parâmetros e treinamento do modelo

A definição do modelo requer a divisão de dados em conjuntos de treino e teste. Essa é uma prática realizada para que o modelo treinado com um certo subconjunto de dados, seja testado com outro subconjunto do mesmo *dataset*, evitando o *overfitting*. Considerando isso, os dados do *dataset* PAMAP2 foram separados em dois subconjuntos, chamado de 70/30, onde 70% dos dados são usados para treino e 30% usados para testes e validação.

A implementação do modelo é feita em camadas; Utilizamos uma rede neural BiLSTM para a primeira camada, seguida das camadas *Flatten*, *Dropout* e *Dense*. *Flatten* é usada para transformar a matriz multidimensional resultante das camadas anteriores em um vetor unidimensional, que pode ser útil quando se deseja aplicar camadas densas

[71]. *Dropout* é a camada onde são definidos uma quantidade de neurônios para serem desligados aleatoriamente, evitando *overfitting* na etapa de treinamento [71]. Por fim, *Dense* implementa a rede neural totalmente conectada definindo a dimensão da saída (número de classes) e a função de ativação [71].

Este trabalho utiliza a função de ativação *softmax* que é comumente empregada em redes neurais para tarefas de classificação [56]. Sua utilização garante que a saída da rede represente a probabilidade dos dados pertencerem a uma das classes predefinidas.

Outros parâmetros utilizados para a definição do modelo não são relevantes para esse momento sendo definidos na apresentação de resultados, no Capítulo 6. A Figura 5.15 traz a ilustração de todas as camadas utilizadas na definição do modelo.

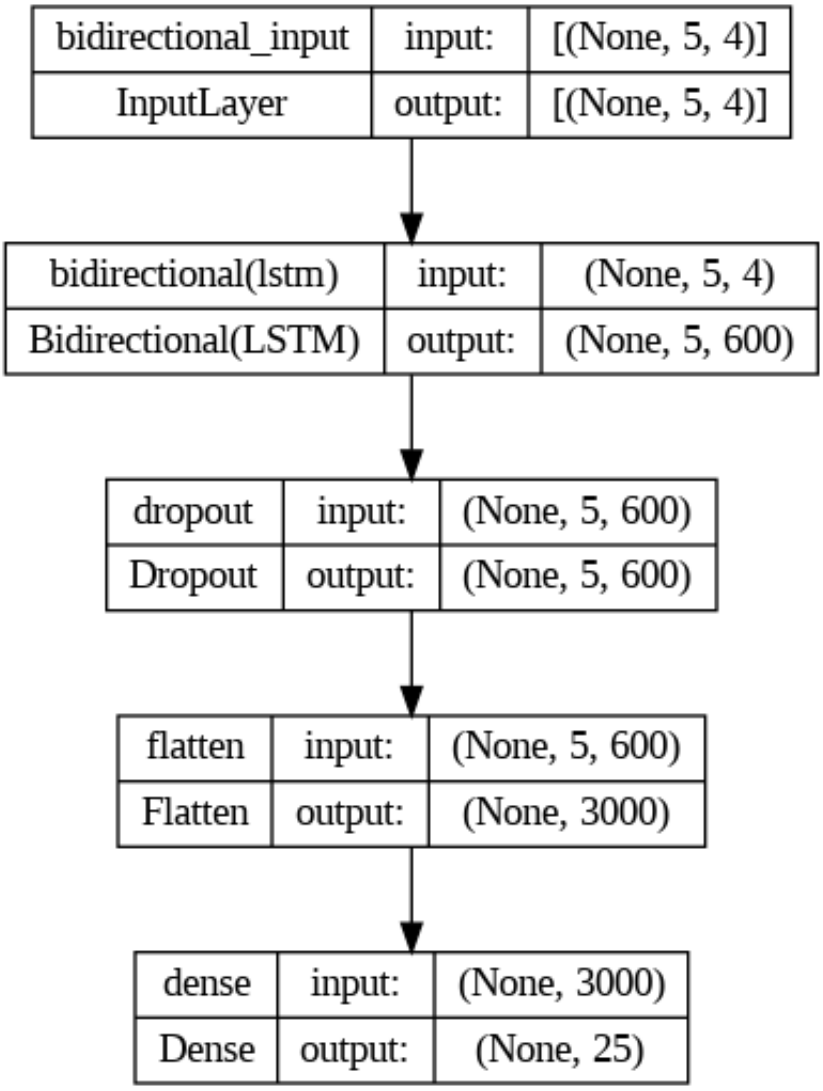


Figura 5.15 – Definição das camadas iniciais da rede neural proposta (Fonte: o Autor)

Ao estimular uma rede neural com uma entrada, os valores dos pesos são aleatórios, ou seja, não garantem corresponder com as características da classe que queremos. Assim, esse processo de definição dos pesos é realizado em épocas. Os pesos são atualizados a cada vez que a rede neural é estimulada. Não há uma definição de um número

padrão de épocas para uma rede neural – cada caso deve ser estudado para estimar o número de épocas mais apropriado [31]. Utilizamos 15 épocas para testes, porém, como será apresentado na discussão de resultados, esse número foi alterado para aumentar a eficácia das previsões.

5.7 Definição de janelas deslizantes

Conforme [19], o conceito de janelas deslizantes envolve dividir os dados em janelas sobrepostas de tamanho fixo e processar cada janela de forma independente, reduzindo a complexidade do algoritmo utilizado. Por exemplo, considerando uma sequência de amostras de dados que representam os valores de uma série temporal, como é o caso desse trabalho, uma janela deslizante de tamanho 3 envolveria a divisão dos dados em janelas sobrepostas de 3 pontos consecutivos e o processamento de cada janela de forma independente. A primeira janela consistiria nos primeiros 3 pontos, a segunda janela consistiria no segundo, terceiro e quarto pontos e assim por diante. Essa técnica é frequentemente usada para extrair características de uma sequência de dados, como para classificação de imagens ou detecção de objetos. Pode ser útil em situações onde o contexto ou relacionamento entre pontos de dados adjacentes é importante, pois a sobreposição entre janelas permite que o contexto seja preservado. A Figura 5.16 exemplifica como uma janela deslizante monta a nova estrutura de dados que será processada. Cada cor diferente na imagem representa um conjunto de amostras selecionado no *dataset* PAMAP2. Na figura, a janela deslizante tem um tamanho 7, já que considera uma janela de 7 amostras para processamento. O Capítulo 6 traz informações sobre os tamanhos de janelas deslizantes escolhidos e os resultados obtidos no trabalho.

Nesse trabalho, as janelas deslizantes são úteis já que, para a formação de uma atividade por completo, apenas uma amostra não é relevante, mas sim uma “janela” de amostras em conjunto. Uma função foi definida no código para facilitar esse processo de quebra do *dataframe* na quantidade de instantes de tempo escolhida.

Time	ActivityID	HeartRate	Hand Temperature	Hand Accel X	Hand Accel Y	Hand Accel Z	Hand Accel2 X	Hand Accel2 Y	Hand Accel2 Z	Hand Gyro X	Hand Gyro Y	Hand Gyro Z	Hand Magn X
15.47	0	141	24.75	-998.278	261.759	246.038	-102.341	269.055	245.977	0.0429727	-0.042494	0.0261664	516.354
15.48	0	NaN	24.75	-975.577	246.441	250.279	-99.628	256.789	258.057	0.0410709	-0.0927293	-0.004726	516.326
15.49	0	NaN	24.75	-103.272	243.953	34.565	-996.105	244.737	291.305	0.0933546	-0.0670724	-0.0174201	521.295
15.5	0	NaN	24.75	-109.992	229.537	371.633	-10.53	24.519	336.664	0.0338018	-0.132737	-0.0253312	51.637
15.51	0	NaN	24.75	-108.863	18.399	379.653	-110.711	225.982	373.003	-0.0259602	-0.156064	-0.0143308	522.538
15.52	0	NaN	24.75	-105.234	0.848548	357.439	-110.001	164.021	383.702	-0.030274	-0.23785	-0.065492	517.602
15.53	0	NaN	24.75	-101.929	0.0481915	346.658	-106.327	0.716133	364.236	0.0743668	-0.288862	-0.101703	511.434
15.54	0	NaN	24.75	-960.758	-0.17879	405.342	-100.508	-0.0129493	367.357	0.247163	-0.285913	-0.135282	511.401
15.55	0	NaN	24.75	-948.164	-0.479489	447.968	-964.395	-0.347827	40.061	0.383824	-0.276547	-0.155755	515.069
15.56	0	NaN	24.75	-966.283	-0.703317	474.714	-96.279	-0.604329	438.416	0.52525	-0.288771	-0.263766	512.666
15.57	0	NaN	24.75	-955.131	-153.785	486.708	-968.923	-108.682	47.477	0.603111	-0.252774	-0.311723	518.872

Figura 5.16 – Exemplo de janela deslizante (Fonte: o Autor)

6. RESULTADOS EXPERIMENTAIS

Esse capítulo discute os parâmetros aplicados para construção da rede neural, e os resultados encontrados, que serão apresentados de maneira incremental, discutindo em cada etapa as melhorias e alterações realizadas.

6.1 Definição do ambiente de execução

Uma das primeiras etapas realizadas durante o desenvolvimento do trabalho, foi a verificação de qual o melhor ambiente de execução dos testes. Essa definição é importante já que há um tempo limitado para realizar a pesquisa, e o tempo de processamento de um modelo altera significativamente baseado na configuração de *hardware* do ambiente de execução, além da quantidade de épocas e quantidade de camadas do modelo. Essa verificação foi realizada com a rede neural apresentada na Figura 5.15, e os resultados obtidos são apresentados na Tabela 6.1. O ambiente de execução foi definido através de três opções: máquina local, Google Colab em sua versão gratuita, e a versão paga (Pro) do Google Colab.

Tabela 6.1 – Ambientes de execução *versus* Tempo de execução (Fonte: o Autor)

Número de Épocas	Ambiente de Execução	Tempo	Acurácia
3	Máquina Local	3723s	87.60%
3	Google Colab	4111s	84.52%
5	Máquina Local	6614s	88.12%
5	Google Colab	8089s	86.23%
10	Máquina Local	11819s	89.64%
10	Google Colab	13663s	88.41%
10	Google Colab Pro	11766s	88.65%

O teste foi realizado através da execução da rede neural em todos os dados do *dataset* escolhido, alterando apenas o número de épocas, que implicou no aumento do tempo de execução quando a alteração foi realizada de maneira incremental. É possível notar que a ferramenta do Google Colab, mesmo quando paga - na sua versão *Pro*, não apresentou um resultado superior à máquina local. Nesse caso, a escolha mais óbvia seria

optar pelo uso da máquina local ou mesmo do Google Colab na versão gratuita por não haverem diferenças significativas nos resultados. No entanto, uma alteração importante dentro da ferramenta do Google Colab foi a alteração do ambiente de execução, de CPU (Central Processing Unit) para GPU (Graphics Processing Unit). Como citado pela NVidia [44], GPUs realizam cálculos técnicos mais rapidamente e com maior eficiência energética do que as CPUs. Isso significa que oferecem um desempenho melhor para treinamento e inferência de IA, bem como ganhos em uma ampla gama de aplicações que usam computação acelerada.

Tabela 6.2 – Ambiente de execução *versus* Tempo de execução utilizando GPU (Fonte: o Autor)

Quantidade de Épocas	Ambiente de Execução	Tempo	Acurácia
3	Google Colab	596s	82.20%
5	Google Colab	1007s	83.72%
10	Google Colab	2065s	85.85%

A Tabela 6.2 apresenta os resultados obtidos utilizando a opção de GPU do Google Colab. Como é possível perceber na tabela, o tempo de execução é reduzido significativamente em comparação com a CPU, além de que a acurácia é similar, com uma pequena redução se comparado com a máquina local. Por esse motivo, optamos por continuar os testes com a versão gratuita do Google Colab e o ambiente de execução baseado em GPU.

6.2 Dataset desbalanceado

Conforme descrito na Seção 5.3, os dados do *dataset* estavam desbalanceados, requerendo uma estratégia para não gerar um viés no aprendizado de máquina. O *dataset* inteiro foi considerado, não apenas uma atividade em específico, assim como feito na definição do ambiente de execução anteriormente apresentada. O número inicial de épocas foi definido em 15 para todos os testes, com uma janela deslizante de tamanho 5. Note que estes valores não são baseados em nenhuma experimentação, servindo apenas para fins de comparação de melhoria de resultados de processamento e de escolha de estratégia. A escolha do número de épocas e do tamanho da janela deslizante para esta dissertação é discutida com maiores detalhes na Seção 6.3. O algoritmo de cada estratégia foi executado no Google Colab utilizando CPU, visto que esses testes foram realizados antes da alteração do ambiente de execução para GPU; porém, para efeitos de compara-

ção, essa alteração não é importante já que todos as estratégias foram executadas com a mesma configuração.

Para efeitos de comparação com as estratégias apresentadas nesta seção, é importante citar as taxas de acerto e erro do modelo apresentado, sem a aplicação de nenhuma estratégia, tendo então um *dataset* desbalanceado. A Tabela 6.3 traz as taxas obtidas para cada classe antes da aplicação da estratégia, utilizando o ambiente de execução de CPU. Não foi possível identificar algum padrão entre os resultados encontrados e a quantidade de amostras de cada atividade, visto que nesse caso a atividade “Correndo” é uma das que possui a menor quantidade de amostras originalmente, mas ainda assim apresenta uma taxa de acertos alta. No entanto, sem comprovação científica, acreditamos que as maiores taxas de acerto podem estar também relacionadas aos dados dos sensores que compõem essas atividades, já que na atividade “Deitado” os valores devem se manter sem variação significativa para acelerômetro e giroscópio por exemplo, sendo um caso bem específico de atividade que tem uma variação de dados quase nula. Por outro lado, uma atividade como “Correndo” apresenta variação significativa no sensor de acelerometria.

Em relação as atividades com menores taxas de acerto também não é possível correlacionar apenas com as menores quantidades de amostras, apesar de as atividades “Descendo as escadas” e “Subindo as escadas”, que possuem as piores taxas de acerto, terem também uma quantidade de amostras baixa em relação as demais. Outro ponto importante em relação a essas duas atividades é a variação similar de dados para ambas, sendo que o que diferencia as duas é apenas o sentido em que a atividade é realizada.

6.2.1 Aplicação da função SMOTE

Inicialmente, a estratégia SMOTE foi utilizada. O Algoritmo 6.1 utiliza a função *fit_resample* da biblioteca *imblearn.SMOTE*, recebendo por parâmetro *x*, que é uma matriz contendo os dados das colunas de amostras de cada sensor do *dataset*, e *y*, que é um vetor unidimensional contendo apenas o identificador da atividade diária a qual os dados de sensores daquela linha se referem.

A Figura 6.1 compara os resultados obtidos antes e depois da aplicação da estratégia nos dados do *dataset*. Cada linha do lado esquerdo da tabela se refere a uma classe e a quantidade de amostras da classe, mostrando o desbalanceamento do *dataset* antes da aplicação do SMOTE. O lado direito, mostra que após a aplicação da função, o *dataset* fica totalmente balanceado, sendo que cada classe tem exatamente o mesmo número de amostras.

A Tabela 6.4 apresenta o resultado da execução do algoritmo após o balanceamento utilizando essa estratégia, onde se destacam 1, 5 e 6 como sendo as classes com

Tabela 6.3 – Taxas de acertos e erros com o *dataset* desbalanceado (Fonte: o Autor)

Atividade	Acertos (%)	Erros (%)
Deitado (1)	97.10%	2.90%
Sentado (2)	90.98%	9.02%
Em pé (3)	87.74%	12.26%
Caminhando (4)	88.41%	11.59%
Correndo (5)	94.04%	5.96%
Pedalando (6)	92.70%	7.30%
Caminhada nórdica (7)	82.79%	17.21%
Subindo as escadas (12)	69.26%	30.74%
Descendo as escadas (13)	68.62%	31.38%
Usando aspirador de pó (16)	82.49%	17.51%
Passando roupas (17)	87.67%	12.33%
Pulando corda (24)	85.89%	14.11%

maior taxa de acerto. Nesse momento, a identificação do numerador de cada classe não é importante, já que a importância está na verificação da taxa de acertos após a estratégia aplicada no *dataset*. Para efeito de comparação, esse algoritmo também foi executado no ambiente de execução CPU, mostrando que, com o mesmo ambiente, resultados com uma diferença significativa podem ser encontrados após a aplicação da estratégia SMOTE.

Comparando a Tabela 6.3, anteriormente apresentada, é possível notar uma melhora na taxa de acertos de quase todas as classes. Convém informar que Orta [51] propõe usar a técnica de SMOTE apenas nos dados de treino, descrevendo que a eficácia é melho-

```
smote = SMOTE()
x, y = smote.fit_resample(x, y)
```

Algoritmo 6.1 – Trecho de código de utilização da função *fit_resample* da biblioteca SMOTE (Fonte: o Autor)

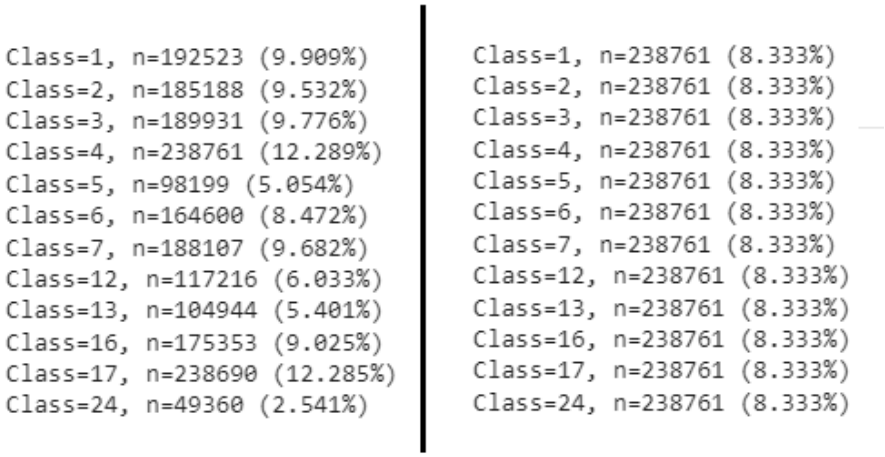


Figura 6.1 – Comparativo do número de amostras das classes do *dataset* antes e depois do uso da técnica SMOTE (Fonte: o Autor)

Tabela 6.4 – Taxas de acertos e erros após o uso de SMOTE (Fonte: o Autor)

Classe	Acertos (%)	Erros (%)
Deitado (1)	**97.32%**	**2.68%**
Sentado (2)	91.32%	8.68%
Em pé (3)	90.62%	9.38%
Caminhando (4)	89.20%	10.80%
Correndo (5)	92.83%	7.17%
Pedalando (6)	**94.91%**	**5.09%**
Caminhada nórdica (7)	84.58%	15.42%
Subindo as escadas (12)	77.98%	22.02%
Descendo as escadas (13)	79.78%	20.22%
Usando aspirador de pó (16)	84.32%	15.68%
Passando roupas (17)	89.82%	10.18%
Pulando corda (24)	**95.40%**	**4.60%**

rada com esta abordagem, pois os dados de treino ficam balanceados, mas os dados de teste seriam validados no algoritmo de maneira original, sem que amostras geradas fos-

sem comparadas com a rede construída. Essa etapa não foi realizada no desenvolvimento do trabalho e, por isso, será citada como trabalho futuro.

6.2.2 Aplicação de pesos em amostras e classes

Outro teste realizado para balanceamento do *dataset* foi a penalização de classes e amostras. Inicialmente, a aplicação de pesos em amostras é realizada utilizando a função *class_weight.compute_sample_weight* da biblioteca *sklearn.utils*. Dois parâmetros são necessários na chamada da função, onde o primeiro *(*class_weight) recebe valores de pesos pré-definidos ou o valor *(*balanced) que se refere a manter o balanceamento idêntico em todas as amostras, sendo que esse foi o valor utilizado na execução deste algoritmo. Além disso, o segundo parâmetro *(*y) recebe um vetor unidimensional, o identificador da atividade diária para cada linha do *dataset*. A chamada utilizada na execução desse algoritmo pode ser conferida no Algoritmo 6.2. O valor resultante da execução dessa função é utilizado como parâmetro de entrada no treinamento do modelo.

```
1 sample_w = class_weight.compute_sample_weight(class_weight='balanced', y=y_train)
2 history = model.fit(X_train,y_train,epochs = 15, validation_split=0.33,
3 sample_weight=sample_w)
```

Algoritmo 6.2 – Trecho de código de utilização da função *compute_sample_weight* da biblioteca *sklearn* (Fonte: o Autor)

De forma bastante similar ao algoritmo anterior, outra estratégia consiste na aplicação de pesos em classes, e não mais em amostras. Nesse caso, a função utilizada é similar à função utilizada para pesos em amostras. A função *class_weight.compute_class_weight* da biblioteca *sklearn.utils* recebe três parâmetros em sua chamada, onde o primeiro *(*class_weight) é igual ao da função *class_weight.compute_sample_weight*, recebendo o valor *balanced*; o segundo (*classes*) recebe o contador da quantidade de classes distintas entre as amostras; e o terceiro parâmetro *(*y) recebe o mesmo vetor unidimensional da chamada da função *class_weight.compute_sample_weight*. Dessa forma, a chamada resultante é apresentada no Algoritmo 6.3 e o resultado também é utilizado como parâmetro de entrada da função de treinamento do modelo.

A execução de ambas as estratégias trouxe resultados que são apresentados na Tabela 6.5. De maneira geral, não há diferenças significativas entre uma estratégia ou outra, os percentuais variam positivamente para uma estratégia ou para outra dependendo da classe. As maiores diferenças entre taxas de acerto foram identificadas nas atividades

```
y_integers = np.argmax(y_train, axis=1)
class_weights = class_weight.compute_class_weight('balanced',
classes = np.unique(y_integers), y = y_integers)
model.fit(X_train,y_train,epochs = 100, validation_split=0.33,
class_weight=class_weights)
```

Algoritmo 6.3 – Trecho de código de utilização da função *compute_class_weight* da biblioteca *sklearn* (Fonte: o Autor)

“Subindo as escadas” e “Descendo as escadas”, sendo que para as quase todas as demais atividades, a diferença foi menor do que 2%.

Tabela 6.5 – Taxas de acertos e erros após o uso de pesos em amostras e em classes (Fonte: o Autor)

Classe	Acertos com Pesos em Amostras (%)	Acertos com Pesos em Classes (%)	Diferença entre taxas de acerto (%)
Deitado (1)	97.45%	97.14%	0.31%
Sentado (2)	92.07%	91.06%	1.01%
Em pé (3)	87.62%	88.18%	0.56%
Caminhando (4)	83.77%	87.46%	3.69%
Correndo (5)	93.05%	92.08%	0.97%
Pedalando (6)	92.52%	93.96%	1.44%
Caminhada nórdica (7)	83.03%	82.44%	0.59%
Subindo as escadas (12)	75.74%	70.50%	5.24%
Descendo as escadas (13)	75.59%	69.00%	6.59%
Usando aspirador de pó (16)	84.47%	81.39%	3.08%
Passando roupas (17)	86.04%	87.35%	1.31%
Pulando corda (24)	88.38%	86.57%	1.81%

6.2.3 Comparação entre estratégias

De maneira resumida e para facilitar a análise, a Tabela 6.6 traz as taxas de acertos para todas as classes considerando as três estratégias apresentadas. Além disso, o valor obtido sem uso de nenhuma estratégia também é informado em uma das colunas. As maiores taxas de acerto de cada classe foram destacadas deixando em vermelho as menores taxas de acerto e em azul as maiores taxas de acerto, mas não se notam grandes diferenças ou algo que facilite a tomada de decisão. Todas as estratégias apresentaram melhorias para algumas classes e piora para outras, no entanto, utilizamos **pesos em classes** visto que obteve os piores resultados e objetivamos explorar melhorias frente a estes resultados no desenvolvimento do trabalho. A estratégia SMOTE, apesar de ter o maior número de taxas de acerto superiores, ainda apresenta dúvidas sobre seu uso, visto que a recomendação é a de aplicação da estratégia apenas em dados de treinamento, e durante o desenvolvimento deste trabalho essa etapa não foi testada. Além disso, em relação a estratégia de aplicação de pesos em amostras ou classes, os resultados são bastante similares. Considerando que o ambiente de execução para essa comparação não foi o final, já com a definição correta de épocas e de janela deslizante utilizada no algoritmo final, todos os resultados tendem a aumentar a eficácia.

6.3 Definição do número de épocas e tamanho da janela deslizante

Tendo o ambiente de execução definido e o *dataset* escolhido e balanceado, a próxima etapa realizada é a definição do número de épocas em que o algoritmo será executado, e a janela deslizante que melhor performa no algoritmo, atingindo a maior acurácia possível e com a menor taxa de erros. Além disso, é importante verificar se nenhuma situação de *underfitting* ou *overfitting* ocorre.

Como já discutido, não existe um “número mágico” para épocas e/ou tamanho da janela deslizante, por esse motivo a estratégia adotada foi a de executar o algoritmo com diferentes valores para épocas e janela deslizante, avaliando o gráfico de percentual de erros em dados de treino (*training loss*) *versus* percentual de erros em dados de teste/validação (*test/validation loss*).

Training Loss é uma métrica usada para avaliar como um modelo de aprendizado profundo se ajusta aos dados de treinamento. *Test/Validation Loss* é uma métrica usada para avaliar o desempenho de um modelo de aprendizagem profunda no conjunto de validação. Na maioria dos projetos de aprendizagem profunda, *training loss* e *test/validation loss* são visualizados em conjunto em um gráfico. O objetivo é diagnosticar o desempenho do modelo e identificar quais aspectos precisam de ajuste [8].

Tabela 6.6 – Taxas de acerto das estratégias discutidas (Fonte: o Autor)

Classe	Original (%)	SMOTE (%)	Pesos em Amostras (%)	Pesos em Classes (%)	Diferença entre melhor e pior (%)
Deitado (1)	97.10%	97.32%	97.45%	97.14%	0.35%
Sentado (2)	90.98%	91.32%	92.07%	91.06%	1.09%
Em pé (3)	87.74%	90.62%	87.62%	88.18%	3.00%
Caminhando (4)	88.41%	89.20%	83.77%	87.46%	5.43%
Correndo (5)	94.04%	92.83%	93.05%	92.08%	1.96%
Pedalando (6)	92.70%	94.91%	92.52%	93.96%	2.39%
Caminhada nórdica (7)	82.79%	84.58%	83.03%	82.44%	2.14%
Subindo escadas (12)	69.26%	77.98%	75.74%	70.50%	8.72%
Descendo escadas (13)	68.62%	79.78%	75.59%	69.00%	11.16%
Usando aspirador de pó (16)	82.49%	84.32%	84.47%	81.39%	3.08%
Passando roupas (17)	87.67%	89.82%	86.04%	87.35%	3.78%
Pulando corda (24)	85.89%	95.40%	88.38%	86.57%	9.51%

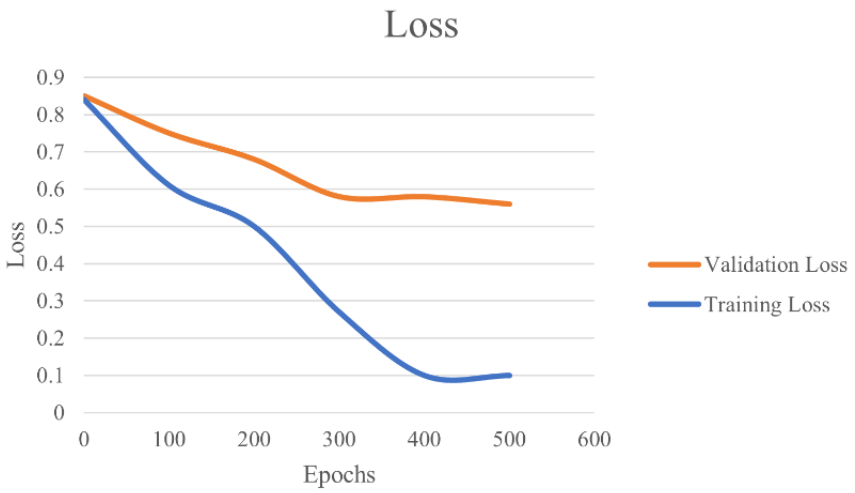


Figura 6.2 – *Validation Loss* versus *Training Loss* em caso de *underfitting* (Fonte: [8])

O *underfitting* pode ser indicado quando o *validation loss* é maior que o *training loss*. Conforme ilustra a Figura 6.2, o *underfitting* ocorre quando o modelo não consegue modelar com precisão os dados de treinamento e, portanto, gera grandes erros. Os resultados indicam que é necessário mais treinamento e, alternativamente, também podemos aumentar os dados de treinamento obtendo mais amostras ou alterar as dimensões da janela deslizante [8].

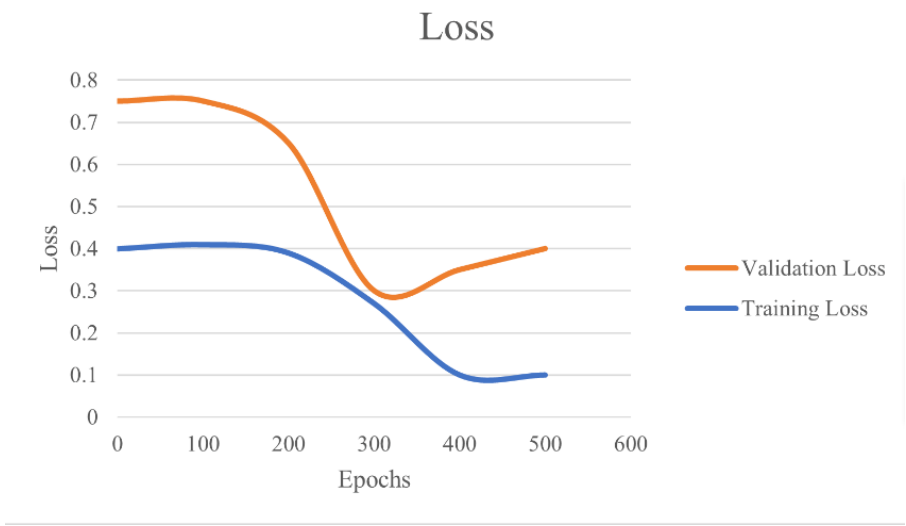


Figura 6.3 – *Validation Loss* versus *Training Loss* em caso de *overfitting* (Fonte: [8])

Por outro lado, a Figura 6.3 traz um exemplo de gráfico que apresenta *overfitting*. Nesse caso, o *validation loss* também é maior que o *training loss*, sendo que o modelo tem um bom desempenho nos dados de treinamento, mas um desempenho ruim nos novos dados do conjunto de validação. Em certo ponto, o *validation loss* chega a diminuir, mas começa a aumentar novamente após algumas épocas. Uma razão comum para esta ocorrência é que o modelo pode ser muito complexo para os dados ou que o modelo foi treinado por um longo período. Uma estratégia que pode ser adotada nesse caso é o *Early Stopping* (parada antecipada), onde o treinamento é interrompido quando a perda é baixa e estável [8].

Por fim, a Figura 6.4 apresenta um caso ideal, onde não ocorre *underfitting* ou *overfitting*, sendo que *training loss* e *validation loss* diminuem uniformemente até estabilizar.

Tendo essa definição, foram realizados testes utilizando o algoritmo já discutido e o ambiente de execução Google Colab com GPU. O primeiro teste foi realizado utilizando 20 épocas e uma janela deslizante de tamanho 5. Esse teste resulta no gráfico apresentado na Figura 6.5. Analisando o gráfico, é possível notar que o *test loss* é maior do que o *training loss*, sendo que não ocorre uma tendência visível de diminuição do *test loss*. Essa figura é bastante similar ao caso anteriormente apresentado de *underfitting*, motivado principalmente pelo tamanho da janela deslizante ser muito pequeno.

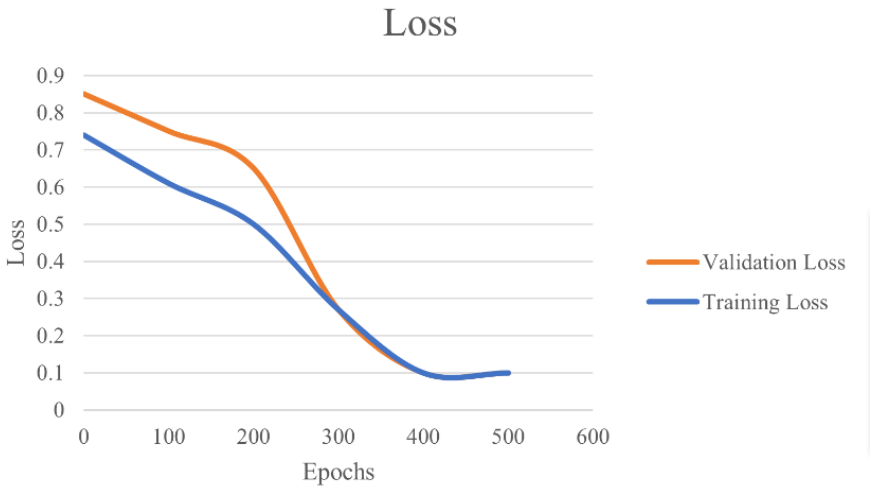


Figura 6.4 – *Validation Loss* versus *Training Loss* sem *underfitting* e/ou *overfittting* (Fonte: [8])

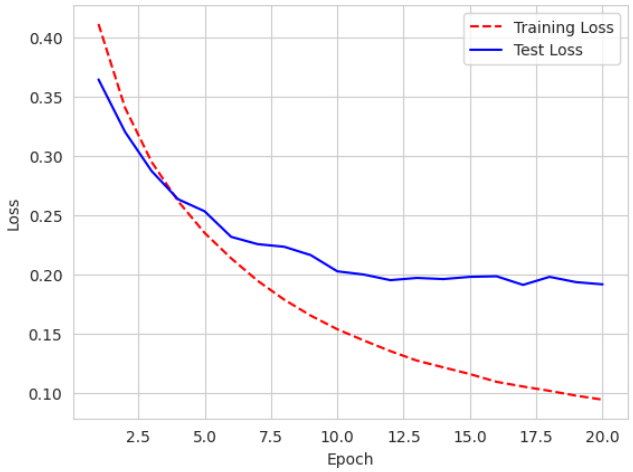


Figura 6.5 – Teste com 20 épocas e janela deslizante de tamanho 5 (Fonte: o Autor)

No teste subsequente, aumentamos o tamanho da janela deslizante para 30. Esse número é significativamente superior ao testado anteriormente e traz mais informações de amostras para cada época, permitindo ao algoritmo aprender mais sobre os dados para tomadas de decisão futuras. A Figura 6.6 ilustra que este teste apresenta uma melhora no gráfico, onde tanto *training loss* quanto *test loss* tendem a diminuir os erros, sendo que as duas linhas ficam próximas. Como já discutido anteriormente, esse é um caso ideal, onde não ocorre *underfitting* e também não ocorre *overfitting*. Durante essa execução, notamos a necessidade de utilizar um plano pago da ferramenta Google Colab, visto que a ferramenta apresentou problemas como falta de memória no ambiente de execução, além de falta de recursos para processamento.

Apesar de já ser um caso ótimo de parâmetros para o modelo, optamos por realizar mais alguns testes para avaliar a eficácia do modelo em relação aos dados. O teste seguinte considera uma janela deslizante um pouco menor, de tamanho 20, e 10 épocas,

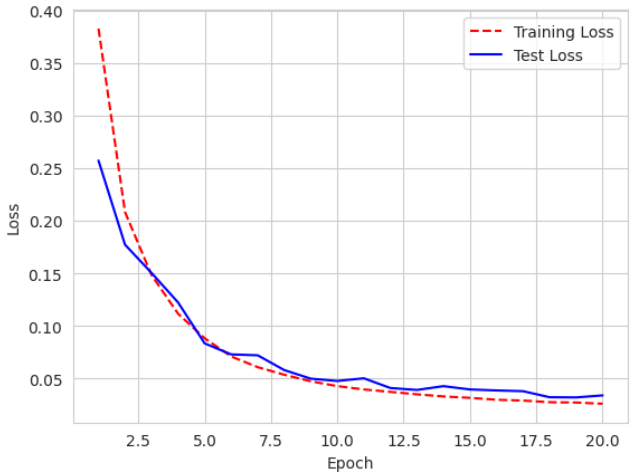


Figura 6.6 – Teste com 20 épocas e janela deslizante de tamanho 30 (Fonte: o Autor)

também menor em relação ao teste anterior. Nesse caso, a Figura 6.7, mostra um caso bastante similar ao anterior, porém, as linhas ainda apresentam uma tendência de queda, sem apresentar uma estabilidade em uma reta.

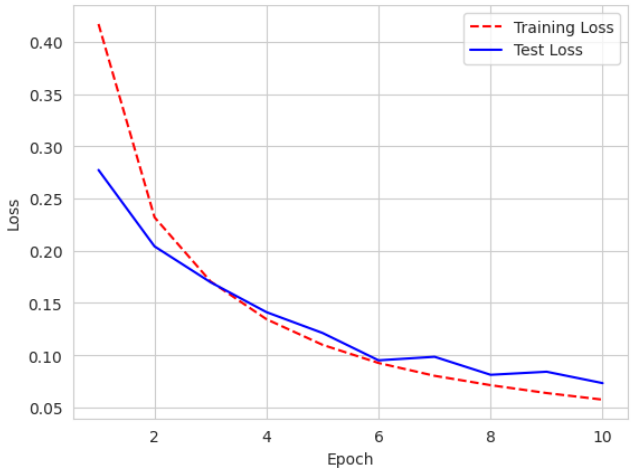


Figura 6.7 – Teste com 10 épocas e janela deslizante de tamanho 20 (Fonte: o Autor)

As linhas próximas e apresentando tendência de queda, sem ainda ter atingido estabilidade, é o que se nota ao voltar ao tamanho anterior de janela deslizante (30), mantendo as 10 épocas, sendo possível notar apenas uma pequena redução no *test loss* em comparação ao caso anterior. A Figura 6.8 apresenta esse caso.

Como as variações foram pequenas nos últimos testes optamos por fazer novos testes com grandes alterações nos parâmetros. A primeira tentativa foi com uma janela deslizante de tamanho 300, sendo que esse teste não pôde ser realizado por falta de recursos de processamento do plano pago do Google Colab. Então, foi utilizado o tamanho 100 para a janela deslizante, mantendo as 10 épocas. A Figura 6.9 mostra a diminuição e estabilidade das duas linhas, mas sem relevância suficiente para justificar a necessidade

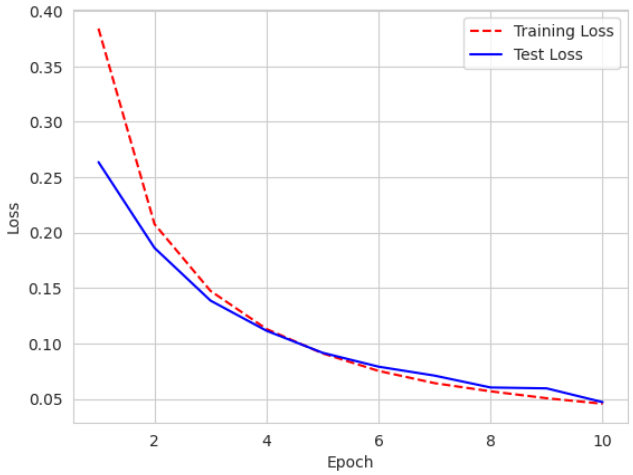


Figura 6.8 – Teste com 10 épocas e janela deslizante de tamanho 30 (Fonte: o Autor)

de janela deslizante com um tamanho tão superior aos testes anteriores, haja visto que o aumento de tamanho na janela deslizante aumentou o tempo de processamento.

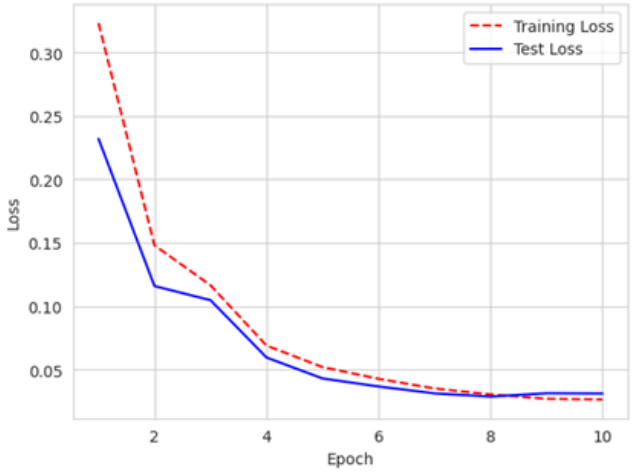


Figura 6.9 – Teste com 10 épocas e janela deslizante de tamanho 100 (Fonte: o Autor)

Por fim, aumentamos o número de épocas de maneira expressiva para 100, mantendo o tamanho de 30 para a janela deslizante, visto que não se notou uma grande diferença no teste realizado entre 30 e 100 para a janela deslizante. A Figura 6.10 mostra que as duas linhas se mantêm com uma variação consistente até pouco mais de 40 épocas. Após esse período, ambas têm uma tendência de subida, e depois apresentam um comportamento aparentemente caótico. Esse caso é semelhante ao apresentado no exemplo de *underfitting*.

Os testes realizados sugerem que o melhor *tradeoff* entre eficácia de resultados e custo computacional está próximo de uma janela deslizante de tamanho 30; ressaltamos que tamanhos menores de janelas tiveram bons resultados, mas aumentar um pouco essa janela resulta em uma eficácia ligeiramente superior, sendo que esse tamanho não causa uma complexidade tão grande que implique em problemas de processamento. Adi-

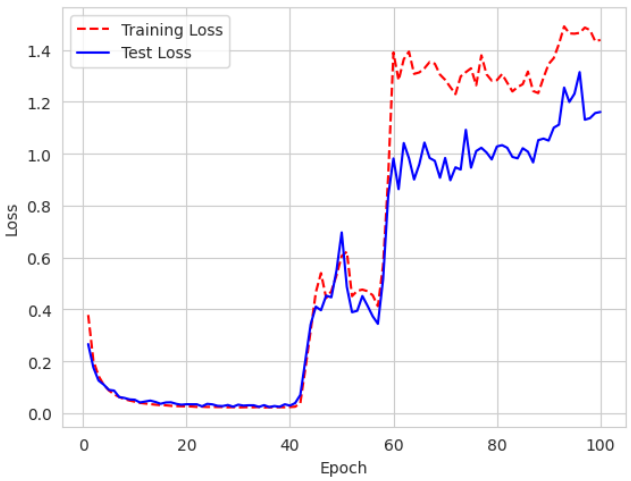


Figura 6.10 – Teste com 100 épocas e janela deslizante de tamanho 30 (Fonte: o Autor)

cionalmente, os testes mostraram que 10 épocas são suficientes para atingir um comportamento próximo ao desejado, no entanto, optamos por utilizar 20 épocas devido à este número apresentar resultados ainda melhores e mais estáveis.

6.4 Fusão de sensores *versus* Identificação de atividades diárias

Tendo o *dataset* balanceado e a definição do número de épocas e tamanho de janela deslizante, alguns testes foram realizados para verificar como a alteração do número de sensores treinados em conjunto impacta no resultado final da identificação de atividades diárias. O Apêndice B traz todo o código utilizado para a execução dos testes.

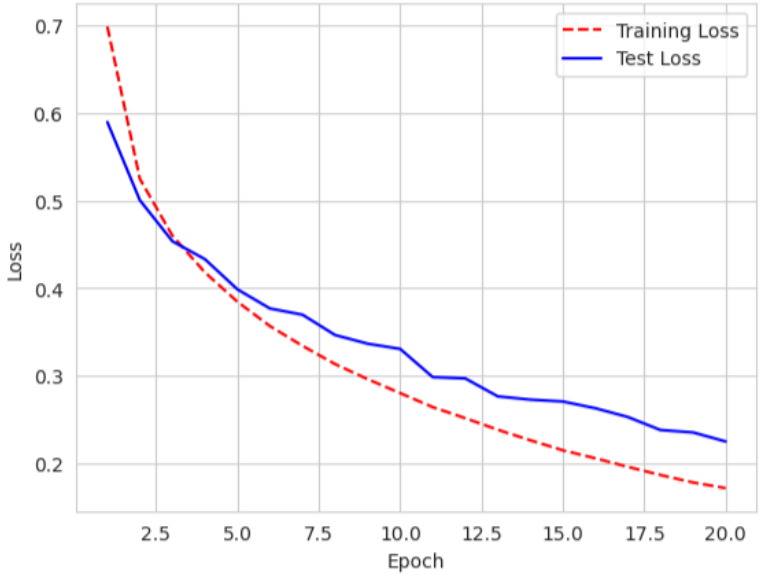


Figura 6.11 – *Training loss versus Test loss* - Sensor de acelerômetro no peito (Fonte: o Autor)

O primeiro teste realizado buscou avaliar apenas os dados de acelerometria do peito do participante, visto que o sensor de acelerometria possui muitos dados e cada atividade diária costuma ter informações diferentes já que os movimentos são diferentes. A análise da Figura 6.11 sugere que a rede pode ser treinada com algumas épocas a mais para atingir a estabilidade, visto que as linhas ainda têm uma tendência de queda.

Tabela 6.7 – Taxas de acerto - Sensor de acelerômetro posicionado no peito (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	96.08%
Sentado (2)	85.89%
Em pé (3)	83.72%
Caminhando (4)	96.17%
Correndo (5)	97.83%
Pedalando (6)	96.94%
Caminhada nórdica (7)	96.64%
Subindo as escadas (12)	94.16%
Descendo as escadas (13)	94.84%
Usando aspirador de pó (16)	90.25%
Passando roupas (17)	86.02%
Pulando corda (24)	98.02%

Mesmo com essa informação, ainda é possível obter uma taxa de acertos alta apenas com os dados de acelerometria do peito, como é possível notar pela Tabela 6.7. A tabela apresenta taxas mais baixas de acerto em atividades em que o usuário tem pouco movimento: “Sentado”, “Em pé” e “Passando roupas” - sendo que esta última atividade (“Passando roupas”) é realizada com movimentos, mas normalmente não na região do peito. Às demais atividades apresentam taxas de acerto acima dos 90%.

Como o projeto LifeSenior utiliza um relógio, realizamos o mesmo teste com o sensor de acelerometria no pulso do usuário. Nesse caso, o comportamento foi bastante parecido com o teste anterior, sendo que a Figura 6.12 apresenta uma tendência

de queda, mostrando que o treinamento com mais épocas poderia aumentar as taxas de acerto.

Da mesma forma que no exemplo anterior, a Tabela 6.8 traz informações sobre as taxas de acerto com o treinamento baseado apenas no sensor de acelerometria do pulso. Nesse caso, devido à posição do sensor, as taxas de acerto são ainda maiores. Acreditamos que isso se deve ao fato de que as atividades que são realizadas em pé ou parado, têm uma taxa maior de movimentação na região do pulso do usuário. Exercícios físicos ou atividades com muita movimentação apresentam uma taxa de acertos ainda superior, próxima dos 100%, como é o caso de: “Correndo”, “Pedalando”, “Pulando corda” e “Caminhada nórdica”.

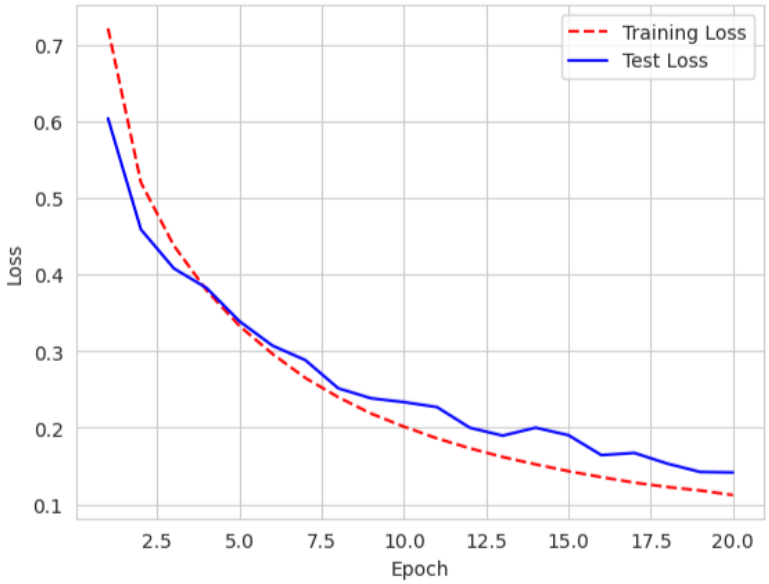


Figura 6.12 – *Training loss versus Test loss* - Sensor de acelerômetro no pulso (Fonte: o Autor)

Ainda considerando o pulso do usuário, outros testes podem ser realizados, tal como adicionar o sensor de batimentos do coração no treinamento. Esse sensor tem uma taxa de amostragem menor do que o sensor de acelerometria, porém, a sua variação com atividades que envolvem muita movimentação e/ou exercícios físicos tende a ser grande. Além disso, em situações de baixa ou quase nenhuma movimentação, a variação de batimentos cardíacos para baixo costuma ser significativa. No teste realizado, o primeiro ponto que pode ser observado na Figura 6.13 é uma melhora em relação ao teste realizado apenas com sensor de acelerometria no pulso. Notamos que as linhas estão mais próximas e, apesar de apresentar ainda uma leve tendência de queda, as linhas já parecem formar uma reta ao final das 20 épocas.

Como é possível notar na Tabela 6.9, há uma melhora na taxa de acertos em todas as classes, sendo que as classes de atividades físicas ou classes que apresentam uma maior movimentação e que elevam batimentos cardíacos apresentam uma variação

Tabela 6.8 – Taxas de acerto - Sensor de acelerômetro no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	96.37%
Sentado (2)	92.83%
Em pé (3)	92.95%
Caminhando (4)	95.95%
Correndo (5)	98.65%
Pedalando (6)	98.06%
Caminhada nórdica (7)	98.27%
Subindo as escadas (12)	92.34%
Descendo as escadas (13)	91.60%
Usando aspirador de pó (16)	95.63%
Passando roupas (17)	94.62%
Pulando corda (24)	97.60%

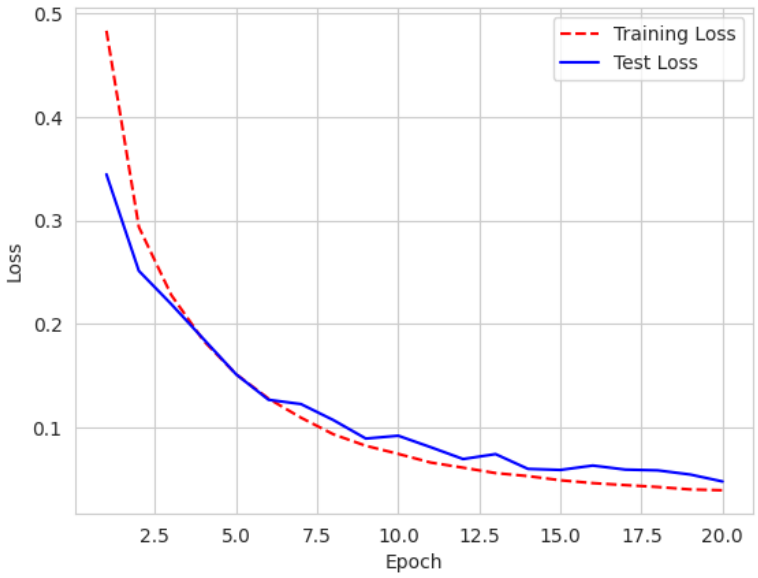


Figura 6.13 – *Training loss versus Test loss* - Acelerômetro + HR no pulso (Fonte: o Autor)

ainda maior. É o caso por exemplo das atividades: “Caminhada”, “Subindo as escadas”

Tabela 6.9 – Taxas de acerto - Acelerômetro + HR no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	99.38%
Sentado (2)	98.64%
Em pé (3)	96.98%
Caminhando (4)	98.84%
Correndo (5)	99.77%
Pedalando (6)	99.74%
Caminhada nórdica (7)	99.36%
Subindo as escadas (12)	97.09%
Descendo as escadas (13)	96.58%
Usando aspirador de pó (16)	98.73%
Passando roupas (17)	98.04%
Pulando corda (24)	99.78%

e “Pulando corda”. A variação também é grande para atividades que são realizadas com o participante parado como é o caso de “Sentado” e “Em pé”, que tiveram aumentos significativos nas taxas de acerto em comparação ao teste realizado apenas com o sensor de acelerometria.

Apesar das taxas de acerto serem altas, um novo teste realizado considera a adição do sensor de temperatura no treinamento. Nesse caso, o teste é realizado com os sensores de acelerometria, batimentos cardíacos e temperatura, todos no pulso do usuário. A Figura 6.14 evidencia que não houve muita alteração em relação ao teste anterior - as épocas são suficientes e a estabilidade em reta é atingida, não havendo situações de *underfitting* ou *overfitting*.

Em relação aos resultados apresentados na Tabela 6.10 não acontecem grandes variações nas taxas de acerto. Algumas classes apresentam uma melhora, mas nada tão significativo, até porque as taxas de acerto já são bem altas.

No entanto, em muitos casos de *datasets* encontrados, o Acelerômetro é um sensor que está presente em uma IMU junto com outros sensores, normalmente Mag-

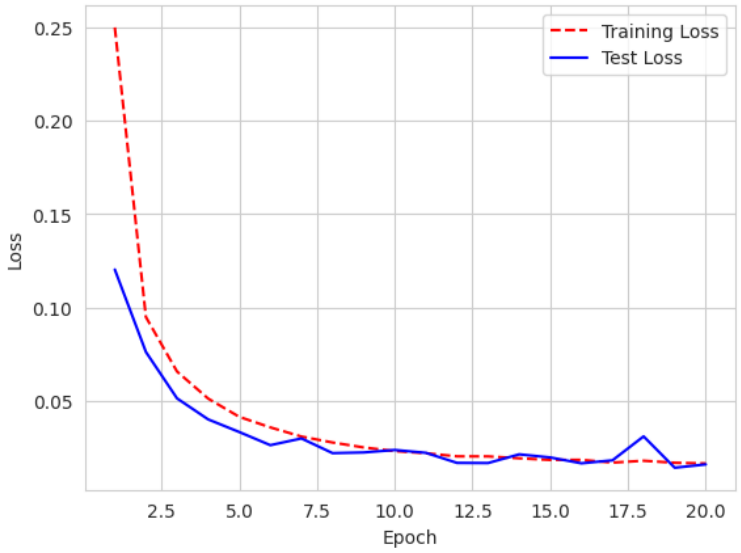


Figura 6.14 – *Training loss versus Test loss* - Acelerômetro + HR + Temperatura no pulso (Fonte: o Autor)

Tabela 6.10 – Taxas de acerto - Acelerômetro + HR + Temperatura no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	99.89%
Sentado (2)	99.36%
Em pé (3)	99.42%
Caminhando (4)	99.94%
Correndo (5)	99.87%
Pedalando (6)	99.84%
Caminhada nórdica (7)	99.93%
Subindo as escadas (12)	99.17%
Descendo as escadas (13)	98.09%
Usando aspirador de pó (16)	99.47%
Passando roupas (17)	99.50%
Pulando corda (24)	99.76%

netômetro e Giroscópio. Nesse sentido, é importante estudar o impacto da retirada do Acelerômetro dos testes, para entender se sua relevância, apresentada nos testes anteriores, impacta o modelo treinado quando da sua não existência. Para tanto, realizamos primeiramente um teste apenas com dados do Giroscópio e do Magnetômetro. Os resultados obtidos e apresentados na Figura 6.15 e na Tabela 6.11 mostram que os dados desses dois sensores apenas apresentam excelentes taxas de acerto para teste, assim como ocorreu com o acelerômetro em teste realizado anteriormente.

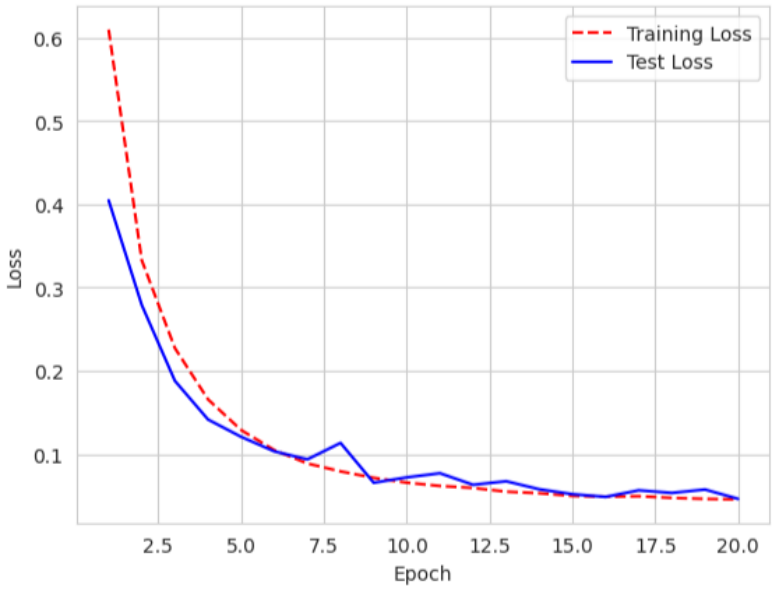


Figura 6.15 – *Training loss versus Test loss* - Giroscópio + Magnetômetro no pulso (Fonte: o Autor)

Quando o Acelerômetro é adicionado de volta a fusão, juntamente com o Magnetômetro e o Giroscópio anteriormente testados, as atividades apresentam taxas de acerto acima de 98%, como é possível notar na Tabela 6.12. Atividades que apresentam uma maior movimentação do braço do sujeito de teste, como “Subindo escadas”, “Descendo escadas”, “Passando roupas” e “Caminhando”, por exemplo, são as atividades que apresentam as maiores taxas de melhora em comparação com o exemplo anterior que não possuía dados de acelerometria. As demais atividades já apresentavam taxas de acerto altas, e isso se mantém com pequenas variações. O gráfico de épocas, apresentado na Figura 6.16, mostra que de forma similar ao exemplo sem dados de acelerometria, as linhas se encontram e apresentam uma tendência de estabilidade em reta com baixas taxas de erros.

Por último, uma nova fusão propõe utilizar todos os sensores disponíveis para o pulso do sujeito de teste. Esse teste se faz necessário para entender a mudança que acontece quando do uso de todos os sensores, e também para verificar como o algoritmo se comporta com uma grande quantidade de informações de uma vez. A Figura 6.17 mostra a taxa de erros muito baixa, similar a testes anteriormente realizados e a Tabela

Tabela 6.11 – Taxas de acerto - Giroscópio + Magnetômetro no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	99.63%
Sentado (2)	99.50%
Em pé (3)	98.78%
Caminhando (4)	97.96%
Correndo (5)	99.66%
Pedalando (6)	99.37%
Caminhada nórdica (7)	99.47%
Subindo as escadas (12)	98.32%
Descendo as escadas (13)	98.41%
Usando aspirador de pó (16)	98.50%
Passando roupas (17)	97.87%
Pulando corda (24)	99.49%

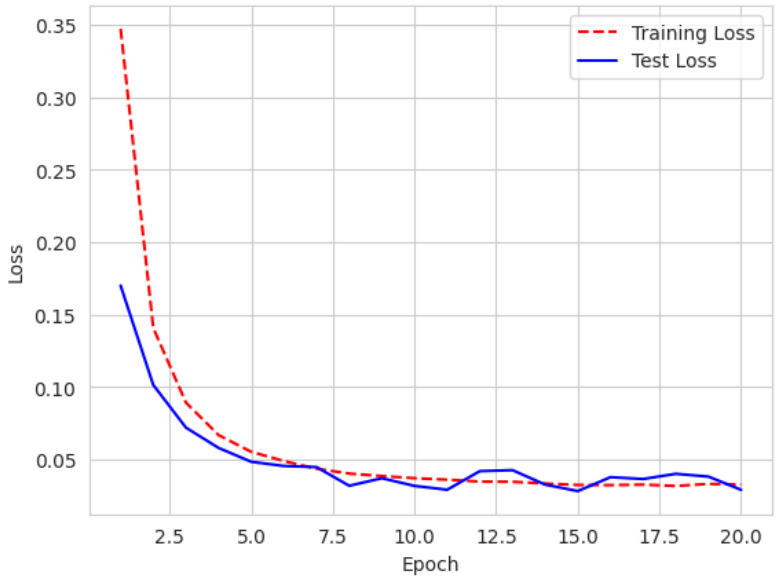


Figura 6.16 – *Training loss versus Test loss* - Acelerômetro + Giroscópio + Magnetômetro no pulso (Fonte: o Autor)

Tabela 6.12 – Taxas de acerto - Acelerômetro + Giroscópio + Magnetômetro no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	99.87%
Sentado (2)	99.43%
Em pé (3)	99.52%
Caminhando (4)	99.61%
Correndo (5)	99.72%
Pedalando (6)	99.66%
Caminhada nórdica (7)	99.71%
Subindo as escadas (12)	98.79%
Descendo as escadas (13)	98.62%
Usando aspirador de pó (16)	99.06%
Passando roupas (17)	98.97%
Pulando corda (24)	99.53%

6.13 mostra as taxas de acerto próximas dos 100%, com pouco espaço para melhorias. No entanto, por ser um teste com uma grande quantidade de informações e pouca diferença em relação a resultados anteriores, acreditamos não ser necessário usar toda essa quantidade de sensores de uma vez já que o tempo e custo de processamento é maior.

Por fim, tentamos realizar um teste considerando todos os sensores de todas as posições do corpo, mas o teste não pôde ser realizado porque, mesmo na ferramenta do Google Colab em sua versão Pro, não houveram recursos suficientes para a execução do algoritmo com tantas informações.

6.5 Comparação entre taxas de acerto de sensores fundidos

A Tabela 6.14 apresenta os resultados encontrados de maneira sumarizada para discussão, sendo que a taxas de acerto mais alta e a taxa de acerto mais baixa para cada

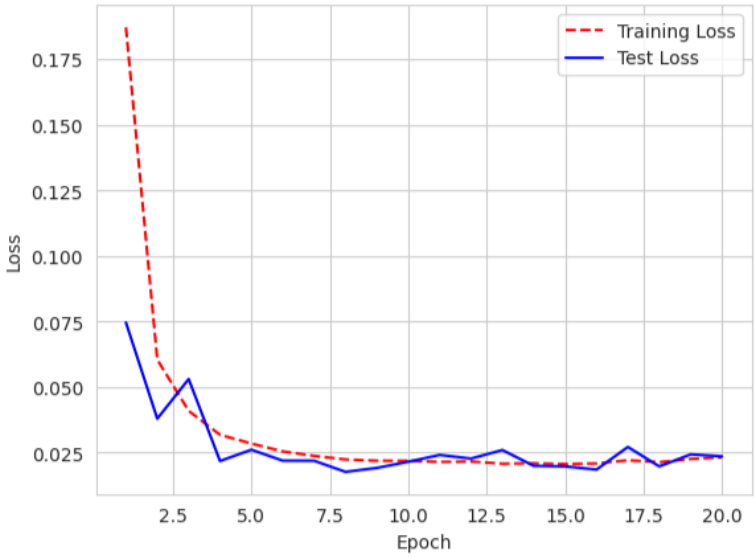


Figura 6.17 – *Training loss versus Test loss* - Acelerômetro + Giroscópio + Magnetômetro + HR + Temp no pulso (Fonte: o Autor)

Tabela 6.13 – Taxas de acerto - Acelerômetro + Giroscópio + Magnetômetro + HR + Temp no pulso (Fonte: o Autor)

Classe	Acertos (%)
Deitado (1)	99.92%
Sentado (2)	99.78%
Em pé (3)	99.57%
Caminhando (4)	99.81%
Correndo (5)	99.94%
Pedalando (6)	99.91%
Caminhada nórdica (7)	99.92%
Subindo as escadas (12)	98.78%
Descendo as escadas (13)	99.25%
Usando aspirador de pó (16)	99.60%
Passando roupas (17)	99.67%
Pulando corda (24)	99.89%

classe, foram destacadas em azul e vermelho, respectivamente. A Fusão E apresenta a maior quantidade de taxas de acerto superiores para cada classe e o fator principal é o número de sensores fundidos neste exemplo.

No entanto, a Fusão B, apresenta taxas de acerto muito altas para a maioria das classes, mesmo tendo a fusão de apenas 3 sensores em sua composição, destacando aqui o Acelerômetro que deu bons resultados em todos os testes em que foi incluído.

Ainda, destacamos a Fusão C, que utiliza apenas dados de Giroscópio e Magnetômetro, e mesmo assim obteve taxas muito altas de acerto, com valores próximos ou acima de 99% para quase todas as classes.

As atividades “Passando roupas”, “Sentado” e “Em pé”, apresentam taxas de acerto baixas se considerarmos o Teste 1, que traz apenas dados de acelerometria no peito. A importância da fusão é justificada quando podemos observar que para a atividade “Em pé”, por exemplo, há um aumento de cerca de 16% na taxa de acerto após a fusão em comparação com o Teste 1. Como todas as fusões foram realizadas com sensores posicionados no pulso do sujeito, podemos realizar a comparação com o Teste 2, que possui apenas dados de acelerometria do pulso, e mesmo assim conseguimos ver aumentos significativos após a fusão, como é o caso da atividade “Descendo as escadas”, onde há um aumento de cerca de 8% na taxa de acerto após a fusão. Todos esses testes e comparações mostram a importância da fusão de dados de sensores para a obtenção de resultados melhores nas taxas de acerto na execução do algoritmo de identificação de atividades diárias.

Tabela 6.14 – Taxas de acerto sumarizadas em relação a fusão de sensores (Fonte: o Autor)

Classe	Teste 1	Teste 2	Fusão A	Fusão B	Fusão C	Fusão D	Fusão E
Deitado (1)	96.08%	96.37%	99.38%	99.89%	99.63%	99.87%	99.92%
Sentado (2)	85.89%	92.83%	98.64%	99.36%	99.50%	99.43%	99.78%
Em pé (3)	83.72%	92.95%	96.98%	99.42%	98.78%	99.52%	99.57%
Caminhando (4)	96.17%	95.95%	98.84%	99.94%	97.96%	99.61%	99.81%
Correndo (5)	97.83%	98.65%	99.77%	99.87%	99.66%	99.72%	99.94%
Pedalando (6)	96.94%	98.06%	99.74%	99.84%	99.37%	99.66%	99.91%
Caminhada nórdica (7)	96.64%	98.27%	99.36%	99.93%	99.47%	99.71%	99.92%
Subindo as escadas (12)	94.16%	92.34%	97.09%	99.17%	98.32%	98.79%	98.78%
Descendo as escadas (13)	94.84%	91.60%	96.58%	98.09%	98.41%	98.62%	99.25%
Usando aspirador de pó (16)	90.25%	95.63%	98.73%	99.47%	98.50%	99.06%	99.60%
Passando roupas (17)	86.02%	94.62%	98.04%	99.50%	97.87%	98.97%	99.67%
Pulando corda (24)	98.02%	97.60%	99.78%	99.76%	99.49%	99.53%	99.89%

Testes iniciais sem fusão:
Teste 1 - Acelerometria no Peito,
Teste 2 - Acelerometria no Pulso

Fusões propostas:
Fusão A - Acelerômetro + HR no Pulso,
Fusão B - Acelerômetro + HR + Temperatura no Pulso,
Fusão C - Giroscópio + Magnetômetro no Pulso,
Fusão D - Acelerômetro + Giroscópio + Magnetômetro no Pulso,
Fusão E - Acelerômetro + Giroscópio + Magnetômetro + HR + Temp no Pulso

7. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho teve como objetivo avaliar como a fusão de sensores no treinamento de uma rede neural impacta nas taxas de acerto obtidas com dados de teste não vistos, além de apresentar ainda os desafios do treinamento da rede neural e balanceamento de um *dataset* público, etapas necessárias para treinar a rede e obter os resultados apresentados.

Além disso, procuramos encontrar um *dataset* que possuísse um grande número de sensores, principalmente posicionados no pulso do sujeito, visto que este trabalho é usado como base para futuros estudos dentro do projeto LifeSenior, projeto que também foi apresentado no decorrer da escrita desta dissertação. Após a escolha do *dataset* foi necessário estudar e testar estratégias de balanceamento dos dados, para obter melhores resultados na execução da rede neural.

Ainda, foram realizados estudos e testes para identificar os melhores parâmetros para treinamento da rede neural, o número de épocas ideal para o algoritmo desenvolvido e para o *dataset* escolhido, bem como, identificar o modelo de aprendizado de máquina que mais se adéqua ao problema de identificar atividades diárias.

No que se refere a arquitetura, optamos pelo uso de uma rede neural BiLSTM, principalmente pela sua capacidade de capturar informações em ambas as direções e por redes desse tipo serem usadas para previsões de séries temporais. Além disso, identificamos que 20 épocas com janelas deslizantes de tamanho 30 foram suficientes para o modelo treinado obter resultados eficazes.

Em relação a estratégia de balanceamento do *dataset*, optamos por utilizar pesos em classes, já que foi a estratégia que apresentou os piores resultados em comparação com as demais estratégias, e o trabalho objetivou explorar o aumento das taxas de acerto.

Na fusão dos dados, foi possível notar que o aumento do número de sensores resulta em taxas de acerto cada vez mais altas, o que é esperado visto que uma maior quantidade de dados pode ajudar a rede a classificar com mais precisão. No entanto, foi possível notar que a escolha correta de um número menor de sensores pode trazer bons resultados. Foi o que a fusão de acelerômetro, sensor de batimentos cardíacos e sensor de temperatura mostraram, ao apresentar taxas de acerto altas. Essa definição é importante no sentido de trabalhos futuros que venham a estudar a utilização dessa fusão mínima de sensores para economia de energia de um dispositivo *wearable*.

Ainda no tópico de trabalhos futuros, novos testes podem ser realizados com outras combinações de sensores para realizar ainda mais descobertas. Além disso, como este trabalho considerou apenas um *dataset*, novos trabalhos podem focar em aprimorar o algoritmo para outros *datasets* públicos, realizando testes com dados e sensores desconhecidos e verificando a acurácia obtida.

Outro estudo é a aplicação de técnicas de balanceamento dos dados do *dataset*. Aqui pode ser citado também o uso correto da biblioteca SMOTE apenas em dados de treinamento, verificando se ocorre uma melhora significativa ao utilizar a biblioteca dessa forma, e a aplicação da técnica de reamostragem de dados, técnica esta que não foi considerada nos testes realizados no decorrer deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abreu, J.; Oliveira, R.; Garcia-Crespo, A.; Rodriguez-Goncalves, R. “Tv interaction as a non-invasive sensor for monitoring elderly well-being at home”. *Sensors*, vol. 21, 2021.

[2] Althobaiti, T.; Katsigiannis, S.; Ramzan, N. “Triaxial accelerometer-based falls and activities of daily life detection using machine learning”. *Sensors*, vol. 20, 2020.

[3] Amazon Web Services. “Computação em nuvem com a aws”. Capturado em: https://aws.amazon.com/pt/what-is-aws/, Jan 2024.

[4] Amazon Web Services. “What is overfitting”. Capturado em: https://aws.amazon.com/pt/what-is/overfitting/, Dez 2023.

[5] Analog Devices, Inc. “ADPD1080 / ADPD1081 Data Sheet”, 2020.

[6] Açıcı, K.; Erdaş, C.; Aşuroğlu, T.; Oğul, H. “Handy: A benchmark dataset for context-awareness via wrist-worn motion sensors”. *Data*, vol. 3, 2018.

[7] Baeldung. “Differences between bidirectional and unidirectional lstm”. Capturado em: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm, Jul 2022.

[8] Baeldung. “Training and validation loss in deep learning”. Capturado em: https://www.baeldung.com/cs/training-validation-loss-deep-learning, Fev 2024.

[9] Banos, O.; Garcia, R.; Saez, A. “Mhealth dataset”. Capturado em: https://archive.ics.uci.edu/dataset/319/mhealth+dataset, Dec 2023.

[10] Bartoski, V. “Análise comparativa de algoritmos de machine learning na detecção de quedas”, Relatório Técnico, Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Pontifícia Universidade Católica do Rio Grande do Sul, 2022, 47p.

[11] Blunck, H.; Bhattacharya, S.; Prentow, T.; Kjrgaard, M.; Dey, A. “Heterogeneity activity recognition dataset”. Capturado em: https://archive.ics.uci.edu/dataset/344/heterogeneity+activity+recognition, Dec 2023.

[12] Bruno, B.; Mastrogiovanni, F.; Sgorbissa, A. “A public domain dataset for adl recognition using wrist-placed accelerometers”. In Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, 2014, pp. 738–743.

[13] Canuma, P. “How to deal with imbalanced classification and regression data”. Capturado em: https://neptune.ai/blog/how-to-deal-with-imbalanced-classification-and-regression-data, Nov 2023.

[14] Celik, Y.; Stuart, S.; Woo, W.; Pearson, L.; Godfrey, A. “Exploring human activity recognition using feature level fusion of inertial and electromyography data”. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2022, pp. 1766–1769.

[15] Chawla, N.; Bowyer, K.; Hall, L.; Kegelmeyer, W. “Smote: synthetic minority over-sampling technique”. *Journal of artificial intelligence research*, vol. 16, 2002.

[16] Che-Chang, Y.; Hsu, Y. “A review of accelerometry-based wearable motion detectors for physical activity monitoring”. *Sensors*, vol. 10, 2010.

[17] Chifu, E.; Chifu, V.; Pop, C.; Vlad, A.; Salomie, I. “Machine learning based technique for detecting daily routine and deviations”. In Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2018, pp. 183–189.

[18] Coutinho, B. “Modelos de predição | svm”. Capturado em: https://medium.com/turing-talks/turing-talks-12-classifica%C3%A7%C3%A3o-por-svm-f4598094a3f1, Jan 2024.

[19] Data Overload. “Sliding window technique — reduce the complexity of your algorithm”. Capturado em: https://medium.com/@data-overload/sliding-window-technique-reduce-the-complexity-of-your-algorithm-5badb2cf432f, Nov 2023.

[20] Deep Learning Book. “Deep learning book”. Capturado em: https://www.deeplearningbook.com.br, Jul 2022.

[21] Deep Learning Book. “Introdução às redes neurais convolucionais”. Capturado em: https://www.deeplearningbook.com.br/introducao-as-redes-neurais-convolucionais, Jan 2024.

[22] Didática Tech. “O que é e como funciona o algoritmo knn”. Capturado em: https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-knn/, Jan 2024.

[23] DIMAVE. “Monitor cardíaco: entenda aqui qual é a sua função”. Capturado em: https://dimave.com.br/qual-a-funcao-do-monitor-cardiaco/, Jan 2024.

[24] E-Ink Display Solutions. “E-ink display solutions”. Capturado em: https://eink.tdglobal.in/, Jan 2024.

[25] Electricity Magnetism. “Giroscópio mems”. Capturado em: https://www.electricity-magnetism.org/pt-br/giroscopio-mems/, Jan 2024.

[26] Engerey. “Tipos e aplicações de sensores na indústria”. Capturado em: http://www.engerey.com.br/blog/tipos-e-aplicacoes-de-sensores-na-industria, Fev 2022.

[27] Eyobu, O.; Han, D. “Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network”. *Sensors*, vol. 18, 2018.

[28] Faculdade de Ciências Médicas da Santa Casa de São Paulo. “Envelhecimento populacional gera mudanças de comportamento no brasil”. Capturado em: https://fcmsantacasasp.edu.br/blog/envelhecimento-populacional-gera-mudancas-de-comportamento-no-brasil/, Jan 2022.

[29] Fine Proxy. “Lstm bidirecional”. Capturado em: https://fineproxy.org/pt/wiki/bidirectional-lstm, Jan 2024.

[30] Google Developers. “Imbalanced data”. Capturado em: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data?hl=en, Nov 2023.

[31] Hardesty, L. “Explained: Neural networks”. Capturado em: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414, Jan 2024.

[32] Hartley, P.; Keevil, V.; Westgate, K.; White, T.; Brage, S.; Romero-Ortuno, R.; Deaton, C. “Using accelerometers to measure physical activity in older patients admitted to hospital”. *Current Gerontology and Geriatrics Research*, vol. 2018, 2018.

[33] Hassan, M.; Uddin, Z.; Mohamed, A.; Almogren, A. “A robust human activity recognition system using smartphone sensors and deep learning”. *Future Generation Computer Systems*, vol. 81, 2018.

[34] Hwang, D.; Chet, P.; Yu, Y.; Wang, Y.; Spachos, P.; Hatzinakos, D.; Plataniotis, K. “Hierarchical deep learning model with inertial and physiological sensors fusion for wearable-based human activity recognition”. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 21–25.

[35] Ivascu, T.; Cincar, K.; Dinis, A.; Negru, V. “Activities of daily living and falls recognition and classification from the wearable sensors data”. In Proceedings of the E-Health and Bioengineering Conference (EHB), 2017, pp. 627–630.

[36] Jurca, R.; Cioara, T.; Anghel, I.; Antal, M.; Pop, C.; Moldovan, D. “Activities of daily living classification using recurrent neural networks”. In Proceedings of the RoEduNet Conference: Networking in Education and Research (RoEduNet), 2018, pp. 1–4.

[37] Kendell, C.; Lemaire, E. “Effect of mobility devices on orientation sensors that contain magnetometers.” *Journal of rehabilitation research and development*, vol. 46, 2008.

[38] Khan, S.; Shah, S.; Noorani, S.; Arsalan, A.; Ehatisham, M.; Raheel, A.; Ahmed, W. “A framework for daily living activity recognition using fusion of smartphone inertial sensors data”. In Proceedings of the International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2023, pp. 1–6.

[39] Kwapisz, J.; Weiss, G.; Moore, S. “Activity recognition using cell phone accelerometers”. *SIGKDD Explor*, vol. 12, 2011.

[40] Laricchia, F. “Smartwatch unit sales worldwide from 2014 to 2017”. Capturado em: https://www.statista.com/statistics/538237/global-smartwatch-unit-sales/, Dez 2022.

[41] Lin, C.-L.; Chiu, W.-C.; Chu, T.-C.; Ho, Y.-H.; Chen, F.-H.; Hsu, C.-C.; Hsieh, P.-H.; Chen, C.-H.; Lin, C.-C.; Sung, P.-S.; Chen, P.-T. “Innovative head-mounted system based on inertial sensors and magnetometer for detecting falling movements”. *Sensors*, vol. 20, 2020.

[42] Manivannan, A.; Chin, W.; Barrat, A.; Bouffanais, R. “On the challenges and potential of using barometric sensors to track human activity”. *Sensors*, vol. 20, 2020.

[43] Mekruksavanich, S.; Jantawong, P.; Hnoohom, N.; Jitpattanakul, A. “Deep learning networks for eating and drinking recognition based on smartwatch sensors”. In Proceedings of the Research, Invention, and Innovation Congress: Innovative Electricals and Electronics (RI2C), 2022, pp. 106–111.

[44] Merritt, R. “Por que as gpus são ideais para ia”. Capturado em: https://blog.nvidia.com.br/2024/01/30/por-que-as-gpus-sao-ideais-para-ia/, Fev 2024.

[45] Much, M.; Marcon, C.; Hessel, F.; Cataldo Neto, A. “Lifesenior – a health monitoring iot system based on deep learning architecture”. In Proceedings of the Human Aspects of IT for the Aged Population. Supporting Everyday Life Activities, 2021, pp. 293–306.

[46] Much, M.; Sieg, J.; Moraes Filho, A.; Bartoski, V.; Schreiber, G.; Marcon, C. “Lifeseniorprofile: A multisensor dataset for elderly real-time activity track”. In Proceedings of the International Conference on Health Informatics (HEALTHINF), 2023, pp. 453–460.

[47] Netmore. “Sigfox explained”. Capturado em: https://netmoregroup.com/sigfox-explained/, Jan 2024.

[48] Nweke, H.; Teh, Y.; Mujtaba, G.; Al-garadi, M. “Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions”. *Information Fusion*, vol. 46, 2019.

[49] Oliveira, D. “O que é e para que serve um magnetômetro”. Capturado em: https://olhardigital.com.br/2024/01/02/reviews/o-que-e-e-para-que-serve-um-magnetometro/, Jan 2024.

[50] Ordóñez, F.; Roggen, D. “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition”. *Sensors*, vol. 16, 2016.

[51] Orta, M. “Splitting training and testing data for imbalanced datasets using smote: A comparison”. Capturado em: https://copyprogramming.com/howto/imbalanced-dataset-train-test-split-before-and-after-smote, Jan 2024.

[52] Passos, B. “Lstm”. Capturado em: https://ateliware.com/blog/long-short-term-memory, Jul 2022.

[53] Piepenbreier, N. “One-hot encoding in machine learning with python”. Capturado em: https://datagy.io/one-hot-encoding/, Fev 2024.

[54] Pires, I.; Marques, G.; Garcia, N.; Flórez-Revuelta, F.; Teixeira, M.; Zdravevski, E.; Spinsante, S.; Coimbra, M. “Pattern recognition techniques for the identification of activities of daily living using a mobile device accelerometer”. *Electronics*, vol. 9, 2020.

[55] Portal do envelhecimento. “Doenças crônicas e o envelhecimento”. Capturado em: http://www.portaldoenvelhecimento.com/acervo/artieop/Geral/artigo250.htm, Jan 2022.

[56] Reis, B. “Redes neurais – funções de ativação”. Capturado em: https://www2.decom.ufop.br/imobilis/redes-neurais-funcoes-de-ativacao/#:~:text=A%20fun%C3%A7%C3%A3o%20de%20ativa%C3%A7%C3%A3o%20softmax,maior%20indica%20a%20classe%20vencedora, Dez 2023.

[57] Reiss, A. “PAMAP2 Physical Activity Monitoring”. Capturado em: https://archive.ics.uci.edu/dataset/231/pamap2+physical+activity+monitoring, Dec 2023.

[58] Reyes-Ortiz, J.; Anguita, D.; Ghio, A.; Oneto, L.; Parra, X. “Human Activity Recognition Using Smartphones Dataset”. Capturado em: https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones, Dec 2023.

[59] Roggen, D.; Calatroni, A.; Nguyen-Dinh, L.-V.; Chavarriaga, R.; Sagha, H. “OPPORTUNITY Activity Recognition Dataset”. Capturado em: https://archive.ics.uci.edu/dataset/226/opportunity+activity+recognition, Dec 2023.

[60] Saod, A.; Mustafa, A.; Soh, Z.; Ramlan, S.; Harron, N. “Fall detection system using wearable sensors with automated notification”. In Proceedings of the IEEE

International Conference on Control System, Computing and Engineering (ICCSCE), 2021, pp. 182–187.

[61] Sarker, I. “Machine learning: Algorithms, real-world applications and research directions”. *SN Computer Science*, vol. 2, 2021.

[62] Scikit Learn‘Official Website. “Standardscaler”. Capturado em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, Fev 2024.

[63] Şengül, G.; Ozcelik, E.; Misra, S.; Damaševičius, R.; Maskeliūnas, R. “Fusion of smartphone sensor data for classification of daily user activities”. *Multimedia Tools and Applications*, vol. 80, 2021.

[64] Singh, K. “How to improve class imbalance using class weights in machine learning”. Capturado em: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/, Fev 2024.

[65] STMicroelectronics. “inemo-inertial modules”. Capturado em: https://www.st.com/en/mems-and-sensors/inemo-inertial-modules.html, Jan 2024.

[66] Tarekegn, A.; Ullah, M.; Cheikh, F.; Sajjad, M. “Enhancing human activity recognition through sensor fusion and hybrid deep learning model”. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW), 2023, pp. 1–5.

[67] Taylor, W.; Dashtipour, K.; Shah, S.; Hussain, A.; Abbasi, Q.; Imran, M. “Radar sensing for activity classification in elderly people exploiting micro-doppler signatures using machine learning”. *Sensors*, vol. 21, 2021.

[68] Torres, H. “Sensores inerciais”. Capturado em: https://www.embarcados.com.br/sensores-inerciais-parte-2/#Acelerometros, Fev 2022.

[69] Toth Lifecare. “Sistema de monitoramento lifesenior”. Capturado em: https://www.tothlifecare.com.br/, Jan 2024.

[70] Viana, G. “Acelerômetro triaxial”. Capturado em: https://pt.scribd.com/document/382549658/Resumo-Sobre-Acelerometro, Jan 2024.

[71] Viceri. “Arquiteturas de redes neurais convolucionais para reconhecimento de imagens”. Capturado em: https://viceri.com.br/insights/arquiteturas-de-redes-neurais-convolucionais-para-reconhecimento-de-imagens/, Dez 2023.

[72] Vidya, B.; Sasikumar, P. “Wearable multi-sensor data fusion approach for human activity recognition using machine learning algorithms”. *Sensors and Actuators A: Physical*, vol. 341, 2022.

[73] Wu, J.; Feng, Y.; Sun, P. “Sensor fusion for recognition of activities of daily living”. *Sensors*, vol. 18, 2018.

[74] Zebin, T.; Sperrin, M.; Peek, N.; Casson, A. “Human activity recognition from inertial sensor time-series using batch normalized deep lstm recurrent networks”. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018, pp. 1–4.

[75] Zhang, M.; Sawchuk, A. “Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors”. In Proceedings of the ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware), 2012, pp. 1036–1043.

APÊNDICE A – DATASET

O *dataset* PAMAP2 [57], utilizado para o desenvolvimento e testes desse trabalho, foi criado através da utilização de um conjunto de sensores composto por três unidades de medição inercial (IMUs) e um monitor de frequência cardíaca. Os IMUs, da marca Colibri da Trivisio, incluem sensores MEMS de 3 eixos (dois acelerômetros, um giroscópio e um magnetômetro), todos amostrados a 100 Hz. Para informações de frequência cardíaca, foi utilizado o monitor HR BM-CS5SR da BM innovations GmbH. Os sensores foram colocados em três posições diferentes do corpo: um IMU e a faixa de frequência cardíaca no peito, um IMU no pulso do braço dominante e outro IMU no tornozelo do lado dominante.

Nove sujeitos participaram da coleta de dados, sendo oito homens e uma mulher, predominantemente funcionários ou estudantes do instituto de pesquisa. A idade média dos participantes era de aproximadamente 27 anos, e o Índice de Massa Corporal (IMC) médio era de cerca de 25 kg/m². Cada sujeito seguiu um protocolo de 12 atividades, que incluíam “deitar”, “sentar”, “ficar em pé”, “caminhar”, “correr”, “pedalar”, “caminhada nórdica”, “passar roupa”, “limpar com aspirador”, “pular corda”, “subir escadas” e “descer escadas”. Além disso, realizaram atividades opcionais como “assistir TV”, “trabalhar no computador”, “dirigir um carro”, “dobrar roupas”, “limpar a casa” e “jogar futebol”. No total, foram coletadas mais de 10 horas de dados das 18 atividades.

As Figuras A.1, A.2, A.3 e A.4 apresentam o conteúdo de cada arquivo do *dataset* com suas respectivas colunas. Cada arquivo .dat tem dados de um dos sujeitos que participou do estudo. Esses arquivos possuem múltiplas linhas e colunas, onde cada linha representa uma amostra, e cada coluna possui dados de um dos sensores coletados. Algumas linhas contêm informações NaN (Not a Number), pois os dados do sensor podem não ter sido coletados naquela amostra.

Time	ActivityID	HeartRate	Hand Temperature	Hand Accel X	Hand Accel Y	Hand Accel Z	Hand Accel2 X	Hand Accel2 Y	Hand Accel2 Z	Hand Gyro X	Hand Gyro Y	Hand Gyro Z	Hand Magn X
15.47	0	141	24.75	-998.278	261.759	246.038	-102.341	269.055	245.977	0.0429727	-0.042494	0.0261664	516.354
15.48	0	NaN	24.75	-975.577	246.441	250.279	-99.628	256.789	258.057	0.0410709	-0.0927293	-0.004726	516.326
15.49	0	NaN	24.75	-103.272	243.953	34.565	-996.105	244.737	291.305	0.0933546	-0.0670724	-0.0174201	521.295
15.5	0	NaN	24.75	-109.992	229.537	371.633	-10.53	24.519	336.664	0.0338018	-0.132737	-0.0253312	51.637
15.51	0	NaN	24.75	-108.863	18.399	379.653	-110.711	225.982	373.003	-0.0259602	-0.156064	-0.0143308	522.538
15.52	0	NaN	24.75	-105.234	0.848548	357.439	-110.001	164.021	383.702	-0.030274	-0.23785	-0.065492	517.602
15.53	0	NaN	24.75	-101.929	0.0481915	346.658	-106.327	0.716133	364.236	0.0743668	-0.288862	-0.101703	511.434
15.54	0	NaN	24.75	-960.758	-0.17879	405.342	-100.508	-0.0129493	367.357	0.247163	-0.285913	-0.135282	511.401
15.55	0	NaN	24.75	-948.164	-0.479489	447.968	-964.395	-0.347827	40.061	0.383824	-0.276547	-0.155755	515.069
15.56	0	NaN	24.75	-966.283	-0.703317	474.714	-96.279	-0.604329	438.416	0.52525	-0.288771	-0.263766	512.666
15.57	0	NaN	24.75	-955.131	-153.785	486.708	-968.923	-108.682	47.477	0.603111	-0.252774	-0.311723	518.872
15.58	0	141	24.75	-952.021	-172.918	471.424	-961.794	-161.587	479.409	0.782889	-0.28316	-0.375533	516.391
15.59	0	NaN	24.75	-9.372	-120.087	452.225	-948.332	-16.924	474.882	108.864	-0.31253	-0.422623	508.789
15.6	0	NaN	24.75	-932.969	-0.177478	444.245	-934.298	-108.936	470.205	128.927	-0.264017	-0.4293	512.659
15.61	0	NaN	24.75	-924.075	0.393375	471.127	-932.092	-0.168368	460.941	131.244	-0.195315	-0.469395	515.067
15.62	0	NaN	24.75	-905.181	0.959998	459.673	-925.517	0.480556	460.792	112.962	-0.136162	-0.511653	509.993
15.63	0	NaN	24.75	-909.415	0.883405	448.089	-923.768	0.79755	4.577	0.838336	-0.042463	-0.595992	511.297

Figura A.1 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (A) (Fonte: o Autor).

Hand Magn Y	Hand Magn Z	Orientation	Orientation	Orientation	Orientation	Chest Temperature	Chest Accel X	Chest Accel Y	Chest Accel Z	Chest Accel2 X	Chest Accel2 Y	Chest Accel2 Z
395.346	-112.478	0.64893	-0.503093	-0.354745	0.447151	31.875	-0.268179	990.328	-336.347	-0.36191	984.957	-320.796
432.573	-115.125	0.648978	-0.503015	-0.354371	0.447465	31.875	-0.30868	986.522	-344.152	-0.42159	994.029	-322.245
417.528	-112.501	0.648709	-0.503316	-0.354256	0.447606	31.875	-0.382124	100.542	-336.603	-0.467637	988.014	-331.302
407.721	-116.422	0.64881	-0.503072	-0.353799	0.448097	31.875	-0.818565	975.937	-263.728	-0.676405	991.042	-303.964
441.657	-116.455	0.649079	-0.502683	-0.353503	0.448377	31.875	-13.383	938.787	-214.211	-103.558	971.431	-252.422
431.864	-119.064	0.649458	-0.501886	-0.352891	0.449203	31.875	-152.773	852.322	-202.657	-159.351	914.151	-21.447
422.721	-122.983	0.649842	-0.500837	-0.351401	0.450983	31.875	-146.573	74.661	-240.922	-173.412	817.566	-199.553
422.765	-119.046	0.649242	-0.501029	-0.350653	0.452214	31.875	-105.676	659.694	-267.004	-148.598	726.995	-230.191
50.763	-125.666	0.648431	-0.501268	-0.348847	0.454504	31.875	-0.758809	655.678	-289.683	-113.249	64.849	-272.948
446.865	-125.625	0.647042	-0.501814	-0.347324	0.45704	31.875	-0.826932	659.649	-262.715	-0.982222	634.856	-260.988
468.353	-128.275	0.645252	-0.502641	-0.345911	0.459727	31.875	-0.727557	655.511	-312.859	-104.191	640.904	-259.423
457.277	-126.951	0.642804	-0.503992	-0.344337	0.462847	31.875	-0.882829	620.752	-44.855	-0.92769	621.371	-341.157
54.822	-124.366	0.639602	-0.506095	-0.342323	0.466468	31.875	-0.882877	662.145	-452.557	-0.948959	616.986	-4.409
484.061	-130.896	0.635723	-0.508917	-0.339662	0.470626	31.875	-113.153	70.027	-39.505	-103.773	650.225	-451.328
582.009	-137.522	0.631603	-0.511766	-0.336804	0.475117	31.875	-14.654	787.135	-372.705	-12.559	713.585	-393.585
60.959	-129.655	0.627741	-0.514172	-0.33536	0.478646	31.875	-163.343	84.704	-431.285	-158.169	780.047	-378.083
596.448	-134.899	0.624219	-0.515705	-0.334782	0.481996	31.875	-175.562	895.818	-462.642	-191.127	842.063	-42.153

Figura A.2 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (B) (Fonte: o Autor).

Chest Gyro X	Chest Gyro Y	Chest Gyro Z	Chest Magn X	Chest Magn Y	Chest Magn Z	Orientation	Orientation	Orientation	Orientation	Ankle Temperature	Ankle Accel X	Ankle Accel Y	Ankle Accel Z
-0.0852794	0.0122102	-0.0184075	160.398	-301.135	32.846	0.512364	-0.741783	-0.360678	-0.239067	311.875	107.309	248.629	-134.938
-0.110942	-0.133588	0.0772646	16.817	-30.271	327.113	0.51306	-0.741768	-0.359856	-0.238857	311.875	117.072	183.273	-110.377
-0.0154851	-0.138495	0.0851918	159.103	-303.669	331.285	0.512982	-0.742123	-0.359267	-0.238809	311.875	106.284	-0.699382	-171.655
-0.00943508	-0.125746	0.0320645	163.053	-306.392	328.508	0.513152	-0.742321	-0.358832	-0.238482	311.875	855.014	-322.831	-195.371
0.0252809	-0.0807326	-0.0999621	164.438	-305.146	32.146	0.51373	-0.741738	-0.359575	-0.237934	311.875	864.873	-352.924	-237.532
-0.0268484	-0.0660476	-0.239958	166.851	-305.245	331.348	0.514514	-0.740604	-0.361188	-0.237327	311.875	100.706	-254.946	-255.976
0.0170739	-0.0576584	-0.382584	166.862	-301.372	327.096	0.514969	-0.740033	-0.362492	-0.236133	311.875	108.713	-118.982	-228.986
0.0188723	-0.0129498	-0.470031	166.894	-306.535	329.945	0.515352	-0.739278	-0.364257	-0.234942	311.875	110.976	-0.0129268	-24.492
0.127812	0.0184759	-0.541057	174.599	-299.075	324.316	0.515772	-0.738537	-0.36633	-0.233119	311.875	111.793	0.66907	-233.683
0.236368	0.182057	-0.618649	175.813	-300.416	329.973	0.515413	-0.737769	-0.369306	-0.231645	311.875	136.931	205.834	-235.826
0.330439	0.152833	-0.622665	175.922	-302.995	325.758	0.514933	-0.737192	-0.372301	-0.229746	311.875	143.192	242.685	-146.657
0.424546	0.194864	-0.623533	17.835	-29.793	329.976	0.513996	-0.736824	-0.375491	-0.227826	311.875	124.347	264.932	0.635152
0.545607	0.301199	-0.636941	176.836	-295.307	341.231	0.511802	-0.736752	-0.379365	-0.22658	311.875	123.851	199.598	164.203
0.662046	0.505903	-0.684705	176.917	-291.432	33.275	0.509458	-0.736482	-0.383846	-0.225189	311.875	12.141	119.387	241.629
0.719739	0.646526	-0.766716	179.368	-286.366	335.558	0.506702	-0.736034	-0.389025	-0.223979	311.875	118.948	0.202244	315.312
0.839249	0.830443	-0.836577	176.636	-283.694	342.574	0.502974	-0.73557	-0.395123	-0.223231	311.875	118.715	0.38847	36.148
0.904257	0.896392	-0.915223	172.616	-280.973	349.581	0.498771	-0.73473	-0.402411	-0.222405	311.875	117.152	121.725	453.421

Figura A.3 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (C) (Fonte: o Autor).

Ankle Accel2 X	Ankle Accel2 Y	Ankle Accel2 Z	Ankle Gyro X	Ankle Gyro Y	Ankle Gyro Z	Ankle Magn X	Ankle Magn Y	Ankle Magn Z	Orientation	Orientation	Orientation	Orientation
106.778	172.467	-126.455	0.714237	-0.235885	-0.341782	-442.263	-102.857	736.341	0.466904	-0.486197	0.608846	-0.418234
109.495	248.129	-101.113	0.238571	-0.279597	-0.536439	-443.354	-107.931	807.278	0.46659	-0.488971	0.607694	-0.417024
115.358	146.827	-0.829364	-0.122434	-0.307176	-0.737734	-444.643	-106.575	8.214	0.466964	-0.492189	0.607102	-0.413669
102.712	-101.313	-132.926	-0.347597	-0.349365	-0.938808	-442.026	-111.868	779.004	0.46873	-0.494896	0.606311	-0.409581
858.616	-317.694	-178.254	-0.469862	-0.457702	-105.988	-434.225	-118.694	779.357	0.469978	-0.497652	0.605975	-0.405287
879.526	-341.922	-205.531	-0.520352	-0.516928	-106.766	-437.792	-128.803	821.899	0.471109	-0.500406	0.605911	-0.400652
100.271	-234.516	-212.585	-0.579226	-0.568197	-111.635	-431.398	-13.171	779.608	0.47315	-0.503431	0.60562	-0.394855
108.401	-0.892578	-180.186	-0.691206	-0.570983	-121.277	-430.078	-133.059	793.853	0.475735	-0.506333	0.60526	-0.38854
110.355	0.166156	-185.171	-0.784424	-0.499024	-125.173	-432.375	-145.823	779.737	0.479068	-0.50861	0.604882	-0.382004
115.753	104.317	-214.497	-0.992361	-0.374624	-118.573	-426.051	-140.976	822.453	0.482994	-0.510167	0.604553	-0.37545
138.469	240.576	-14.652	-120.908	-0.290633	-105.711	-423.227	-150.108	893.574	0.48647	-0.511869	0.604936	-0.367952
138.521	269.464	-0.240203	-142.487	-0.193403	-104.652	-425.809	-153.871	794.245	0.49187	-0.511344	0.60523	-0.360955
126.268	26.358	134.056	-149.637	-0.0816322	-108.899	-424.571	-151.353	808.442	0.497512	-0.510959	0.605204	-0.353739
12.405	191.085	223.705	-141.479	0.0221894	-113.428	-421.985	-156.653	737.679	0.503388	-0.510585	0.604574	-0.346977
120.626	0.989137	3.044	-106.163	0.0717717	-123.529	-416.696	-164.637	752.141	0.508495	-0.511449	0.603031	-0.34089
119.906	0.430462	383.321	-0.577335	0.15028	-138.059	-415.255	-172.434	752.282	0.512778	-0.51386	0.600076	-0.336029
119.035	0.643083	455.948	0.00238025	0.175081	-151.826	-411.244	-185.534	667.476	0.515961	-0.517969	0.595695	-0.332625

Figura A.4 – *Screenshot* parcial de um dos arquivos presentes no *dataset* (D) (Fonte: o Autor).

APÊNDICE B – CÓDIGO-FONTE

Este apêndice apresenta o código desenvolvido em Python, para treinamento e execução da rede neural, considerando uma execução com os sensores do pulso sendo utilizados em conjunto com o sensor de batimentos cardíacos.

```
# Import Libs
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px
import seaborn as sns
from os import listdir
from random import shuffle
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils import class_weight
from collections import Counter
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix

# Dataset Path
path = 'PATH_OF_THE_DATASET'
```

Algoritmo B.1 – Inicialmente são carregadas as bibliotecas e definido o caminho onde se encontram os arquivos do *dataset* (Fonte: o Autor)

```
# Dataset Activities and IDs
activities = {
  'lying': '1',
  'sitting': '2',
  'standing': '3',
  'walking': '4',
  'running': '5',
  'cycling': '6',
  'Nordic walking': '7',
  'watching TV': '9',
  'computer work': '10',
  'car driving': '11',
  'ascending stairs': '12',
  'descending stairs': '13',
  'vacuum cleaning': '16',
  'ironing': '17',
  'folding laundry': '18',
  'house cleaning': '19',
  'playing soccer': '20',
  'rope jumping': '24',
  'other': '0',
}
```

Algoritmo B.2 – Após, um dicionário de dados é criado para especificar o identificador de cada uma das atividades (Fonte: o Autor)

```
# Changes data to the format for LSTM training for sliding window approach
def lstm_data_transform(x_data, y_data, num_steps=5):
    x, y = list(), list()
    # Loop of the entire data set
    for i in range(x_data.shape[0]):
        # compute a new (sliding window) index
        end_ix = i + num_steps
        # if index is larger than the size of the dataset, we stop
        if end_ix >= x_data.shape[0]:
            break
        # Get a sequence of data for x
        seq_x = x_data[i:end_ix]
        # Get only the last element of the sequency for y
        seq_y = y_data[end_ix]
        # Append the list with sequencies
        x.append(seq_x)
        y.append(seq_y)
    # Make final arrays
    x_array = np.array(x)
    y_array = np.array(y)
    return x_array, y_array
```

Algoritmo B.3 – Uma função foi criada e posteriormente ela é chamada para separar os dados do *dataset* no tamanho da janela deslizante escolhida (Fonte: o Autor)

```
# Randomize Files funciton
def randomize_files(file_list):
  shuffle(file_list)

# Get files from path
files = listdir(path);

# Join multiple files in one DF
csv_files = [f for f in files if f.endswith('.dat')]
dfs = []
for csv in csv_files:
    df_loaded = pd.read_csv(path + csv, delimiter=' ', header=None)
    dfs.append(df_loaded)
df = pd.concat(dfs, ignore_index=True)
```

Algoritmo B.4 – Para evitar qualquer tipo de tendência, foi definida uma função para randomizar a ordem dos arquivos. Além disso, todos os dados foram agrupados em um só *dataframe* para facilitar o processamento dos mesmos (Fonte: o Autor)

```
# Define DF columns

df.columns = ["Timestamp", "ActivityId", "HeartRate", "Hand_Temp", "Hand_Accel_X",
"Hand_Accel_Y", "Hand_Accel_Z", "Hand_Accel2_X", "Hand_Accel2_Y", "Hand_Accel2_Z",
"Hand_Gyro_X", "Hand_Gyro_Y", "Hand_Gyro_Z", "Hand_Magn_X", "Hand_Magn_Y",
"Hand_Magn_Z", "Hand_Orie_1", "Hand_Orie_2", "Hand_Orie_3", "Hand_Orie_4",
"Chest_Temp", "Chest_Accel_X", "Chest_Accel_Y", "Chest_Accel_Z", "Chest_Accel2_X",
"Chest_Accel2_Y", "Chest_Accel2_Z", "Chest_Gyro_X", "Chest_Gyro_Y", "Chest_Gyro_Z",
"Chest_Magn_X",
"Chest_Magn_Y", "Chest_Magn_Z", "Chest_Orie_1", "Chest_Orie_2",
"Chest_Orie_3",
"Chest_Orie_4", "Ankle_Temp", "Ankle_Accel_X",
"Ankle_Accel_Y",
"Ankle_Accel_Z", "Ankle_Accel2_X", "Ankle_Accel2_Y", "Ankle_Accel2_Z", "Ankle_Gyro_X",
"Ankle_Gyro_Y",
"Ankle_Gyro_Z", "Ankle_Magn_X", "Ankle_Magn_Y",
"Ankle_Magn_Z", "Ankle_Orie_1",
"Ankle_Orie_2", "Ankle_Orie_3", "Ankle_Orie_4"]
```

Algoritmo B.5 – As colunas do *dataframe* foram nomeadas para facilitar a seleção das mesmas em outras etapas do processamento e execução do código (Fonte: o Autor)

```
# Pre-processment

# Fill dataframe NaN values with forward fill method
df = df.fillna(method='ffill')

## If there's still any null value, fill with zero
df = df.fillna(0)

## Remove activities with 0 - As described on dataset readme
df = df[df.ActivityId != 0]

## Remove orientation coluns, invalid as described by readme
df = df.drop(
    columns=[
      "Hand_Orie_1", "Hand_Orie_2", "Hand_Orie_3", "Hand_Orie_4",
      "Ankle_Orie_1", "Ankle_Orie_2", "Ankle_Orie_3", "Ankle_Orie_4",
      "Chest_Orie_1", "Chest_Orie_2", "Chest_Orie_3", "Chest_Orie_4",
    ]
  )

# Print samples x activity
sns.set_style("whitegrid")
sns.countplot(x = "ActivityId", data = df)
plt.title("Number of samples by activity")
plt.show()

# LSTM construction and processment

## Define columns to use (sensors columns) - Heart rate and hand accelerometer
x = df.loc[:, ['Hand_Accel_X', 'Hand_Accel_Y', 'Hand_Accel_Z', 'HeartRate']];

# Define activity id column
y = df.iloc[: ,[1]].copy()

# Standardize X data as it can be from different scales
scaler_x = StandardScaler()
x = scaler_x.fit_transform(x)

# A numpy array (or) a vector which has integers that represent different categories,
# can be converted into a numpy array (or) a matrix which has binary values
# and has columns equal to the number of categories in the data.
y = tf.keras.utils.to_categorical(y)
```

Algoritmo B.6 – Pré-processamento do *dataframe*, removendo colunas não utilizadas, nulas e/ou inválidas segundo a descrição de uso do *dataset*. Além disso, seleção de colunas utilizadas e transformação dos dados para processamento (Fonte: o Autor)

```
# Define sliding window size
num_steps = 30

# Call data transform function that will create the sliding windows
x, y = lstm_data_transform(x, y, num_steps)

# Create train and test arrays
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33)

# Create model with params
model = tf.keras.Sequential()
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(300, return_sequences=True),
input_shape=(num_steps, 4)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(25, activation='softmax')) #n_classes
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])

#Add class_weights to the classes
y_integers = np.argmax(y_train, axis=1)
class_weights = class_weight.compute_class_weight('balanced',
classes = np.unique(y_integers), y = y_integers)
list_cw = class_weights.tolist()
final_class_weights = list()
for i in range(0,25):
    if i in y_integers:
        final_class_weights.append(list_cw.pop(0))
    else:
        final_class_weights.append(0)
class_weights = {l:c for l,c in zip(range(0,25), final_class_weights)}
```

Algoritmo B.7 – Definição do tamanho da janela deslizante, divisão dos dados em treino e teste, definição dos parâmetros da rede e dos pesos de cada classe (Fonte: o Autor)

```
# Train the model
history = model.fit(X_train,y_train,epochs = 20, validation_split=0.33,
class_weight=class_weights)

# Get training and test loss histories
training_loss = history.history['loss']
test_loss = history.history['val_loss']

# Create count of the number of epochs
epoch_count = range(1, len(training_loss) + 1)
```

Algoritmo B.8 – Treinamento do modelo e obtenção dos dados de perda, além da criação de uma matriz com dados de perda para cada época (Fonte: o Autor)

```
# Visualize loss history
plt.plot(epoch_count, training_loss, 'r--')
plt.plot(epoch_count, test_loss, 'b-')
plt.legend(['Training Loss', 'Test Loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show();

# Make predictions to evaluate the model
preds = model.predict(X_test)

## Create two arrays, one with predictions, other with the real classes
classes_pred=np.argmax(preds,axis=1)
classes_y_test=np.argmax(y_test,axis=1)

hits = []; errors = []; total = []

for i in range(25):
    hits.append(0)
    errors.append(0)
    total.append(0)
for index, item in enumerate(classes_y_test):
    if(classes_y_test[index] == classes_pred[index]):
        hits[item] += 1
    else:
        errors[item] += 1
    total[item] += 1
for index, item in enumerate(total):
    if(total[index] != 0):
        print(str(index) +
        ' - Hits: ' + str((hits[index] / total[index]) * 100) +
        ' - Errors: ' + str((errors[index] / total[index]) * 100))

# Create two arrays
classes_y_test_array = np.array(classes_y_test)
classes_pred_array = np.array(classes_pred)
df = pd.DataFrame({'real': list(classes_y_test_array),
'pred': list(classes_pred_array)}, columns=['real', 'pred'])

## Print dataframe with real and predicted info
df.head()
```

Algoritmo B.9 – Visualização gráfica de épocas e taxas de acertos e erros para cada classe (Fonte: o Autor)

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Pesquisa e Pós-Graduação
Av. Ipiranga, 6681 – Prédio 1 – Térreo
Porto Alegre – RS – Brasil
Fone: (51) 3320-3513
E-mail: propesq@pucrs.br
Site: www.pucrs.br