João Vitor Yukio Bordin Yamashita

Sistema de Controle para Prótese Ativa de Joelho utilizando Deep Learning Embarcado

Brasil

Dezembro de 2024

João Vitor Yukio Bordin Yamashita

Sistema de Controle para Prótese Ativa de Joelho utilizando Deep Learning Embarcado

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação da UNIFEI (área de concentração: Matemática da Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciências e Tecnologia da Computação.

Universidade Federal de Itajubá – UNIFEI

Programa de Pós-Graduação em Ciência e Tecnologia Computação

Orientador: João Paulo R. R. Leite

Coorientador: Jeremias Barbosa Machado

Brasil

Dezembro de 2024

Dedico este trabalho aos meus pais, que sempre me apoiaram incondicionalmente, e aos professores que inspiraram minha paixão pela ciência e tecnologia.

Agradecimentos

Agradeço primeiramente ao meu orientador, Prof. Dr. João Paulo Ribeiro Rodrigues Leite, pelo constante apoio, orientação e incentivo durante todo o desenvolvimento deste trabalho. Sua expertise e dedicação foram fundamentais para a realização desta pesquisa.

Agradeço também ao meu coorientador, Prof. Dr. Jeremias Barbosa Machado, pelas valiosas contribuições e insights que enriqueceram significativamente este estudo.

À Universidade Federal de Itajubá e ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação, pela oportunidade e recursos fornecidos para a realização deste mestrado.

À Luísa, meu amor, que esteve ao meu lado em todos os momentos, tornando esta jornada mais leve e inspiradora com seu apoio incondicional, carinho e compreensão.

Aos meus colegas e amigos, que compartilharam comigo momentos de estudo e reflexão, tornando esta jornada mais leve e enriquecedora.

Finalmente, sou grato à minha família, pelo amor, paciência e incentivo constantes, sem os quais este trabalho não teria sido possível.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Este trabalho aborda o desenvolvimento de modelos de Redes Neurais Convolucionais (CNNs) otimizados para a predição do ângulo do joelho em sistemas embarcados, visando aplicações em próteses de membros inferiores. Utilizando Unidades de Medida Inercial (IMUs) de baixo custo e técnicas de otimização multiobjetivo, foram desenvolvidos modelos especializados para diferentes tipos de marcha (curta, natural e longa), associados a um classificador capaz de identificar o padrão de marcha em tempo real. Os resultados demonstram que os modelos especializados superam significativamente um modelo combinado, alcançando um erro médio quadrático (RMSE) de $2,050°$, uma melhoria de mais de 48% com relação a este último. Além disso, a implementação em um microcontrolador Kendryte K210 validou a viabilidade de implantação em *hardware* acessível, mantendo a eficiência computacional necessária para aplicações em tempo real. Este estudo contribui para o avanço das tecnologias assistivas, indicando que modelos eficientes de aprendizado profundo podem ser efetivamente implementados em sistemas embarcados, melhorando a qualidade de vida de indivíduos com amputações de membros inferiores. Essa abordagem aumenta ainda a acessibilidade financeira de soluções protéticas avançadas, tornando-as mais viáveis para indivíduos em regiões economicamente desfavorecidas.

Palavras-chave: Redes Neurais Convolucionais. *TinyML*. Sistemas Embarcados. Aprendizado de Máquina.

Abstract

This work addresses the development of optimized Convolutional Neural Network (CNN) models for knee angle prediction in embedded systems, targeting applications in lower limb prosthetics. By leveraging low-cost Inertial Measurement Units (IMUs) and multi-objective optimization techniques, specialized models were developed for different gait types (short, natural, and long strides), associated with a classifier capable of real-time gait pattern identification. The results demonstrate that specialized models significantly outperform a combined model, achieving a root mean square error (RMSE) of 2.050°, an improvement of over 48% compared to the latter. Furthermore, implementation on a Kendryte K210 microcontroller validated the feasibility of deployment on accessible hardware while maintaining the computational efficiency required for real-time applications. This study contributes to the advancement of assistive technologies, indicating that efficient deep learning models can be effectively implemented on embedded systems, improving the quality of life for individuals with lower limb amputations. This approach further increases the affordability of advanced prosthetic solutions, making them more accessible to individuals in economically disadvantaged regions.

Keywords: Convolutional Neural Networks. *TinyML*. Embedded Systems. Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MLP	Perceptron Multicamadas
CNN	Rede Neural Convolucional
CNN1D	Rede Neural Convolucional Unidimensional
CNN2D	Rede Neural Convolucional Bidimensional
LSTM	*Long Short-Term Memory*
RNN	Rede Neural Recorrente
RMSE	Erro Quadrático Médio da Raiz
MSE	Erro Quadrático Médio
RSE	Erro Quadrático Relativo
R^2	Coeficiente de Determinação
ESP32	Microcontrolador ESP32
MaixBit	Microcontrolador MaixBit
IMU	Unidade de Medição Inercial
EMG	Eletromiografia

Sumário

1 Introdução

1.1 Contextualização

Amputações de membros inferiores representam um desafio significativo de saúde global, afetando milhões de indivíduos em todo o mundo. Em 2017, cerca de 57,7 milhões de pessoas no mundo apresentaram lesões traumáticas nos membros (MCDONALD et al., 2021). No Brasil, os dados são igualmente preocupantes. Em 2022, foram realizadas 31.190 amputações de membros inferiores no Sistema Único de Saúde (SUS), o que equivale a uma média de 85 procedimentos por dia (Sociedade Brasileira de Angiologia e de Cirurgia Vascular, 2023). De acordo com Aljarrah et al. (2019), as amputações de membros inferiores constituem uma parcela considerável de todas as amputações, sendo as doenças vasculares e o diabetes as causas predominantes. Em muitas regiões, especialmente em países de baixa e média renda, a incidência dessas condições está aumentando (World Health Organization, 2023b), levando a taxas crescentes de amputação e uma necessidade correspondente de soluções protéticas eficazes.

O impacto da perda de membros inferiores vai além do indivíduo, afetando sua mobilidade, capacidade de trabalho e qualidade de vida geral (AMTMANN et al., 2015). Essa situação é agravada em áreas onde o acesso a dispositivos protéticos adequados e serviços de reabilitação é limitado. A disparidade no acesso a esses serviços essenciais representa uma barreira substancial para a participação social e econômica de amputados (World Health Organization, 2023b). Segundo a Organização Mundial da Saúde (World Health Organization, 2023a), tecnologias assistivas são críticas para abordar essas lacunas e melhorar a qualidade de vida, destacando a necessidade urgente de tecnologias protéticas escaláveis e acessíveis.

1.2 Cenário Nacional

As regiões Norte e Nordeste do Brasil enfrentam desafios socioeconômicos significativos que impactam diretamente a qualidade de vida de pessoas com amputações de membros inferiores ((IBGE), 2021). Nessas áreas, além do alto número de amputações (Sociedade Brasileira de Angiologia e de Cirurgia Vascular, 2023), há uma carência de acesso a próteses eficientes devido aos altos custos associados a tecnologias avançadas.

Segundo dados da Sociedade Brasileira de Angiologia e de Cirurgia Vascular (2023), em 2022, o Brasil registrou 31.190 amputações, com as regiões Norte e Nordeste somando juntas 103.083 amputações ao longo dos anos analisados, conforme ilustrado na Figura 1.

Essa realidade evidencia a urgência em desenvolver soluções acessíveis que atendam às necessidades desses indivíduos.

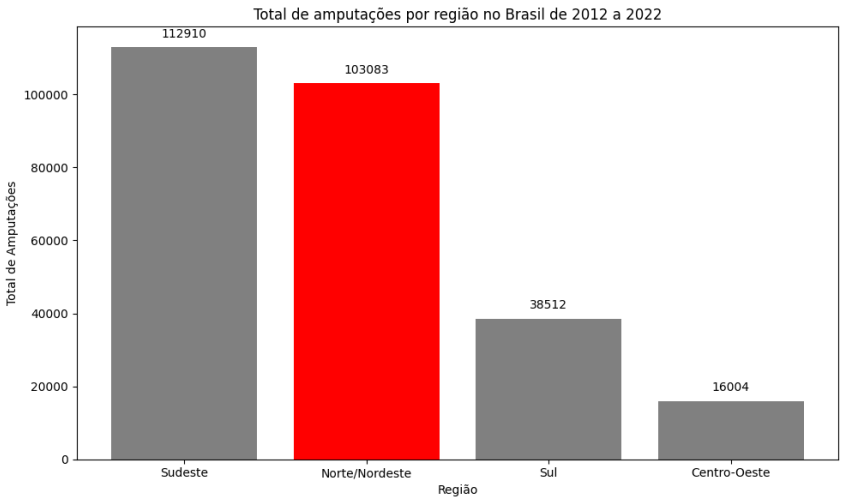


Figura 1 – Total de amputações de membros inferiores por região no Brasil de 2012 a 2022, destacando as regiões Norte e Nordeste em vermelho.

O alto custo das próteses ativas disponíveis no mercado apresenta ainda um obstáculo significativo para a acessibilidade e inclusão de pessoas amputadas. Os dispositivos listados na Tabela 1 (Bionics For Everyone, 2022), com preços que variam de $20.000 a mais de $100.000 USD, ilustram essa realidade. Considerando os dados de renda per capita das regiões Norte e Nordeste do Brasil, que em 2024 foram de R$ 1.107 e R$ 1.023 (Instituto Brasileiro de Geografia e Estatística (IBGE), 2024), respectivamente, o alto custo das próteses microprocessadas torna-se ainda mais evidente. Em uma conversão aproximada, esses dispositivos podem custar entre R$ 100.000 e R$ 500.000, valores que representam uma disparidade significativa em relação ao rendimento médio mensal dessas regiões.

Prótese Ativa de Joelho	**Faixa de Preço (USD)**	**Disponibilidade**
Ottobock Genium X3	> $100.000	Global
Ossur Power Knee	$70.000 a $90.000	Global
Reboocon IntelLeg	$70.000 a $90.000	UE, Rússia e China
Ottobock Genium	$60.000 a $80.000	Global
Ossur Rheo Knee XC	$60.000 a $70.000	Global
Ossur Rheo Knee	$40.000 a $50.000	Global
Ottobock C-Leg	$40.000 a $50.000	Global
Freedom Quattro	$30.000 a $40.000	Global
Freedom Plie Knee	$30.000 a $40.000	Global
Blatchford Orion Knee	$30.000 a $40.000	Global
Nabtesco/Proteor Allux	$30.000 a $40.000	Global
Ottobock Kenevo	$25.000 a $35.000	Global
Teh Lin V-One	$20.000 a $30.000	Global

Tabela 1 – Preços e Disponibilidade de Próteses Ativas de Joelho

Para um indivíduo nessas áreas, a aquisição de uma prótese de R$ 100.000 corresponderia a cerca de 8 anos de rendimento domiciliar per capita na região Norte ou 8,5 anos na região Nordeste, sem considerar outras despesas. O contraste evidencia o quão inacessíveis são essas tecnologias para a maioria da população, especialmente em regiões economicamente vulneráveis. Esse cenário reforça a necessidade urgente de alternativas mais viáveis, que possibilitem o acesso a tecnologias assistivas de qualidade, permitindo a reintegração social e econômica de pessoas amputadas em situação de vulnerabilidade econômica.

1.3 Funcionamento Básico das Próteses Ativas

Próteses ativas modernas combinam sistemas preditivos e de controle para oferecer movimentos mais naturais e responsivos aos usuários. O diagrama apresentado na Figura 2 ilustra a integração entre esses sistemas.

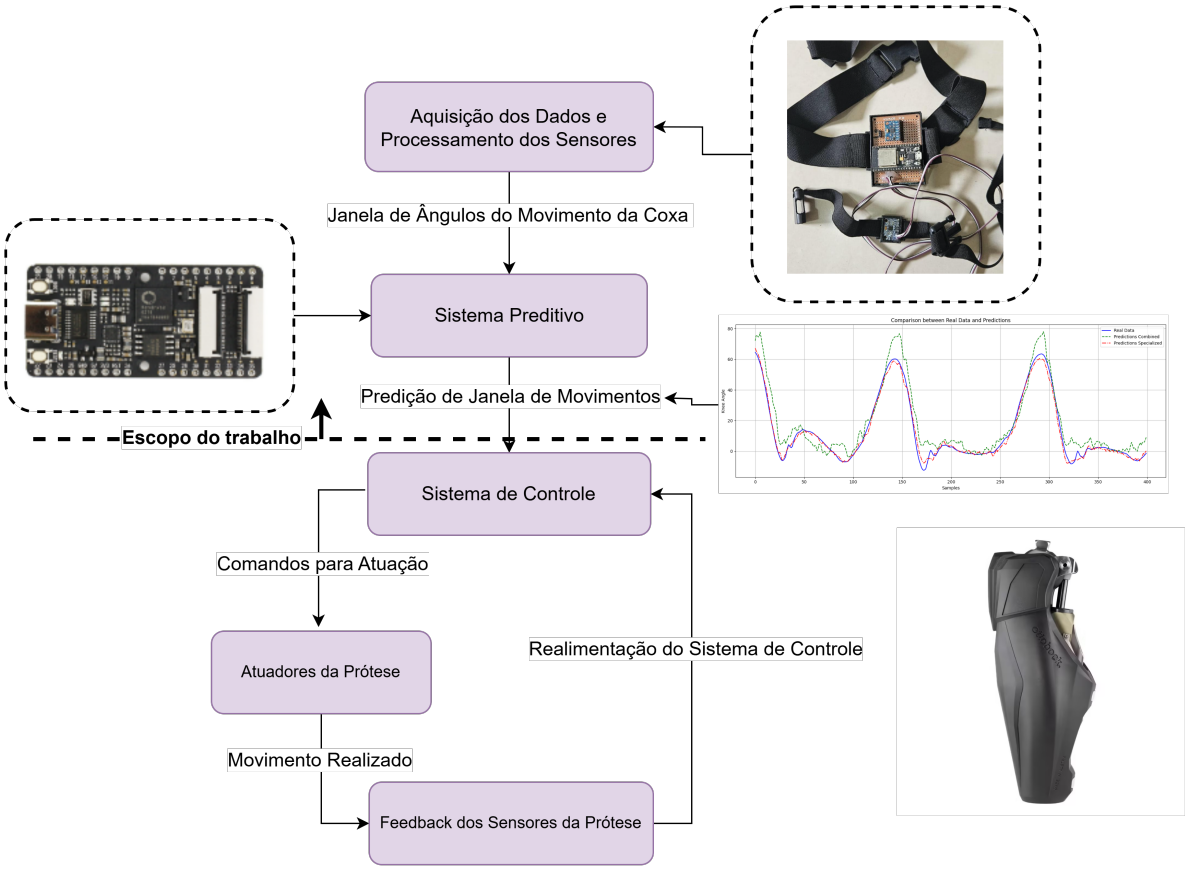


Figura 2 – Diagrama do funcionamento básico das próteses ativas, mostrando a integração entre o sistema preditor e o sistema de controle.

O funcionamento do sistema pode ser descrito pelas seguintes etapas:

1. **Aquisição de Dados:** Os sensores, como EMG, acelerômetros e giroscópios, captam sinais relevantes do usuário.

2. **Predição:** O sistema preditivo processa os sinais e gera uma estimativa da intenção de movimento.

3. **Controle:** O sistema de controle utiliza a predição para gerar comandos precisos aos atuadores.

4. **Atuação:** Os atuadores realizam os movimentos previstos, como flexão ou extensão de articulações.

5. **Feedback:** Sensores na prótese monitoram o desempenho real e ajustam os comandos conforme necessário.

Esse modelo de integração garante que as próteses ativas possam responder de maneira eficiente às intenções dos usuários, melhorando a mobilidade e qualidade de vida.

1.4 Problema de Pesquisa

Próteses ativas, equipadas com juntas motorizadas e sistemas de controle avançados, representam um avanço significativo em relação aos dispositivos protéticos passivos tradicionais. Diferentemente das próteses passivas, que dependem exclusivamente do movimento do membro residual do usuário e das forças gravitacionais, as próteses ativas podem gerar seu próprio movimento por meio de atuadores, proporcionando propulsão e imitando mais de perto a função natural do membro (HERR, 2009). Essa melhoria leva a uma simetria de marcha aprimorada, aumento da velocidade de caminhada e redução do custo metabólico para o usuário (SAWICKI; GORDON; FERRIS, 2005). Além disso, as próteses ativas oferecem melhor adaptabilidade a diferentes terrenos e atividades, aumentando a mobilidade e a qualidade de vida do usuário (SUP; VAROL; GOLDFARB, 2011). A capacidade de controlar ativamente o movimento permite que os usuários realizem tarefas que seriam desafiadoras ou impossíveis com dispositivos passivos, expandindo significativamente suas capacidades funcionais (AU; BERNIKER; HERR, 2008).

A eficácia das próteses ativas é fortemente influenciada pela sua capacidade de prever e responder com precisão aos movimentos pretendidos pelo usuário em tempo real. Sistemas preditivos desempenham um papel crucial na interpretação de sinais de sensores, como eletromiografia (EMG) ou unidades de medição inercial (IMUs), para antecipar o movimento do usuário e ajustar os atuadores das juntas protéticas (HUANG et al., 2011b). A predição precisa garante uma integração melhorada entre o membro residual do usuário e a prótese, resultando em transições mais suaves entre diferentes fases da marcha e atividades (FARINA; ASZMANN, 2014). Além disso, o controle preditivo aprimora a segurança e

a confiabilidade das próteses ativas, ajustando-se de forma preemptiva a mudanças na velocidade de caminhada, direção ou terreno (YOUNG; SIMON; HARGROVE, 2014). Portanto, o desenvolvimento de algoritmos preditivos eficientes e precisos é essencial para maximizar o desempenho e a aceitação do usuário de dispositivos protéticos ativos (ZHANG; EVANGELISTI; LETTIERI, 2013).

Próteses de membros inferiores equipadas com sistemas embarcados representam uma solução promissora para os desafios de acessibilidade e eficiência em tecnologias assistivas (YOUNG; FERRIS, 2017). Ao incorporar sensores de baixo custo e capacidade de processamento em tempo real, é possível aprimorar significativamente a funcionalidade das próteses sem aumentar proporcionalmente seus custos.

Tradicionalmente, modelos sofisticados como RNNs, LSTMs e *Transformers* têm sido utilizados para prever movimentos (ZEROUAL et al., 2020). No entanto, devido à sua complexidade e alta demanda de recursos computacionais, esses modelos não são ideais para implementação em *hardware* de baixo custo (ABADADE et al., 2023). Essa limitação destaca uma oportunidade significativa de pesquisa: explorar o uso de modelos mais simples e eficientes em microcontroladores. Isso é particularmente relevante para o caso de próteses ativas, onde a eficiência e o baixo consumo de recursos são cruciais. Além disso, há uma escassez notável de estudos que discutam técnicas de processamento e tratamento de sinais essenciais para a implementação bem-sucedida dessas aplicações em *hardware* acessível (HAFER et al., 2023).

Este estudo aborda essas lacunas empregando otimização multiobjetivo para selecionar os modelos mais adequados para sistemas embarcados que lidam com dados específicos de marcha. O objetivo é avaliar metodicamente uma variedade de modelos de aprendizado de máquina e aprendizado profundo, integrando um sistema personalizado projetado especificamente para esta aplicação.

1.5 Objetivos

1.5.1 Objetivo Geral

Desenvolver um sistema baseado na implementação de modelos de aprendizado profundo (*deep learning*) otimizados para sistemas embarcados, visando a predição dos ângulos articulares do joelho com boa qualidade a partir do processamento de dados de movimento da coxa capturados por IMUs.

1.5.2 Objetivos Específicos

- Realizar a aquisição de dados de movimento utilizando um microcontrolador ESP32 interligado com sensores MPU6050 IMUs.

- Aplicar técnicas de aumento de dados (*data augmentation*) para aprimorar o conjunto de dados e melhorar a robustez do modelo.

- Empregar otimização multiobjetivo, incorporando a fronteira de Pareto e métricas ponderadas, para selecionar as configurações ótimas do modelo que equilibrem precisão e eficiência computacional.

- Desenvolver modelos especializados para diferentes tipos de marcha—passadas curtas, naturais e longas—utilizando um classificador para selecionar o modelo apropriado em tempo real.

- Implementar e testar os modelos selecionados em uma placa de desenvolvimento Sipeed MaixBit com um microcontrolador Kendryte K210, avaliando seu desempenho em termos de precisão e eficiência.

1.6 Justificativa

A relevância deste estudo reside na possibilidade de aprimorar a funcionalidade e responsividade das próteses de membros inferiores, tornando-as mais acessíveis e eficientes. Ao otimizar modelos de *deep learning* para implementação em sistemas embarcados de baixo custo, como o Kendryte K210, espera-se contribuir para o desenvolvimento de soluções protéticas avançadas e economicamente viáveis.

Além disso, o enfoque em técnicas de processamento de sinais e otimização de modelos atende a uma necessidade identificada na literatura por abordagens que considerem as limitações computacionais dos dispositivos embarcados, sem comprometer a qualidade das previsões. Isso é particularmente importante em contextos de recursos limitados, onde soluções de baixo custo podem ter um impacto significativo na qualidade de vida dos indivíduos amputados.

Este trabalho visa superar as limitações identificadas em estudos anteriores, como Karakish, Fouz e Elsawaf (2022), que constatou que, embora as CNNs oferecessem melhor desempenho, elas eram mais lentas, e Huang et al. (2019a), que dependia de *software* proprietário para implementação e utilizava sensores mais caros, como EMG. Ao aproveitar uma estratégia de aquisição de dados e otimização que maximiza a eficiência sem a necessidade de *hardware* ou *software* caros, este estudo pretende contribuir para o campo das tecnologias assistivas vestíveis, aprimorando a funcionalidade dos dispositivos em aplicações do mundo real.

1.7 Estrutura do Trabalho

De forma geral, este trabalho desenvolve um sistema que começa com a coleta de dados e culmina na implementação em dispositivos embarcados, visando solucionar um problema específico por meio de modelos de aprendizado de máquina. A estrutura do sistema está organizada em várias etapas principais, incluindo a aquisição de dados, a seleção e comparação de arquiteturas de modelos utilizando Otimização Multiobjetivo, a aplicação de modelos especializados e a implementação do sistema em um dispositivo embarcado. A Figura 3 ilustra de maneira simplificada todo o sistema desenvolvido. Essas etapas são detalhadas nas seções subsequentes, proporcionando uma visão abrangente do desenvolvimento e implementação do sistema proposto.

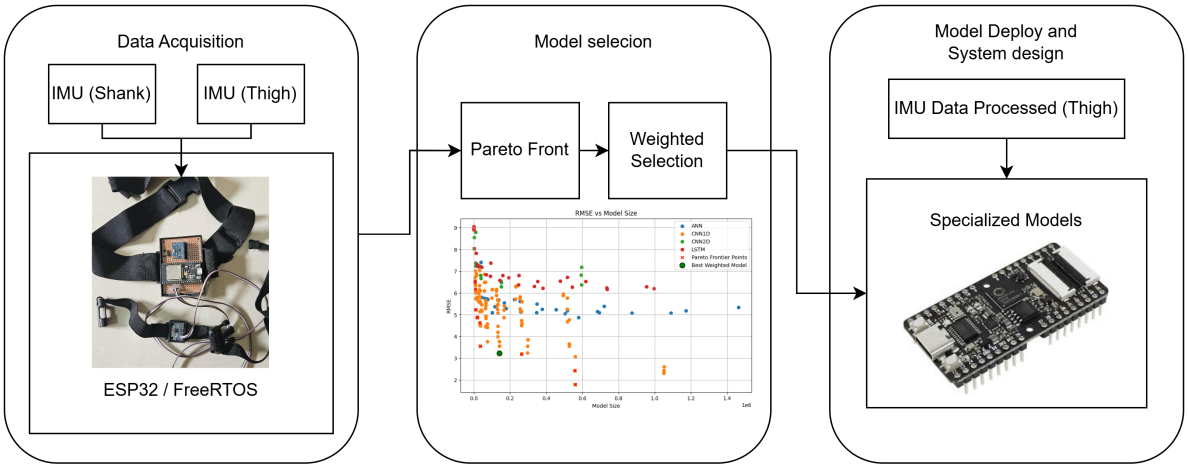


Figura 3 – Visão geral da arquitetura do sistema.

1.8 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma:

- **Capítulo 2 - Revisão Bibliográfica**: Apresenta os trabalhos relacionados, discutindo estudos prévios sobre aquisição de dados de movimento, processamento de sinais e modelagem preditiva em próteses de membros inferiores.
- **Capítulo 3 - Fundamentação Teórica**: Apresenta os conceitos teóricos fundamentais que embasam o trabalho, incluindo teorias, modelos e métodos relevantes para a pesquisa realizada.
- **Capítulo 4 - Materiais e Métodos**: Descreve detalhadamente a metodologia empregada, incluindo a configuração do sistema de aquisição de dados, técnicas de pré-processamento e aumento de dados, arquitetura dos modelos propostos e o processo de otimização multiobjetivo.

- **Capítulo 5 - Resultados e Discussão**: Apresenta os resultados obtidos com os modelos desenvolvidos, comparando o desempenho entre modelos generalistas e especializados, bem como a implementação em sistemas embarcados e sua eficiência.
- **Capítulo 6 - Conclusões e Trabalhos Futuros**: Resume as principais conclusões do estudo, destacando as contribuições realizadas e sugerindo direções para pesquisas futuras que possam expandir e aprimorar os resultados alcançados.

2 Revisão Bibliográfica

Neste capítulo, são explorados os principais trabalhos relacionados à aquisição e processamento de sinais para análise de marcha, bem como os métodos de modelagem preditiva aplicados em próteses de membros inferiores. A revisão visa identificar os avanços recentes e os principais desafios no uso de aprendizado de máquina e *deep learning* em sistemas embarcados para controle de próteses ativas de baixo custo. Primeiramente, discutem-se estudos sobre o uso de Unidades de Medida Inercial (IMUs) para aquisição de dados de movimento, abordando sua precisão e limitações. Em seguida, são analisadas as arquiteturas de modelos preditivos aplicáveis à predição de parâmetros de marcha, destacando os desafios de implementação em *hardware* com recursos limitados. Essa análise evidencia as lacunas na literatura, que este trabalho visa preencher ao propor uma abordagem de *deep learning* otimizada para sistemas embarcados, contribuindo para o desenvolvimento de próteses mais acessíveis e adaptativas.

2.1 Aquisição e Processamento de Sinais na Análise de Marcha

A análise precisa dos parâmetros de movimento humano é fundamental para o desenvolvimento de próteses eficientes. As IMUs têm se mostrado ferramentas valiosas nesse contexto devido à sua portabilidade e capacidade de fornecer dados em tempo real. Estudos como o de Seel, Raisch e Schauer (2014) demonstraram a viabilidade de utilizar IMUs para medições precisas dos ângulos articulares durante a marcha, sem a necessidade de magnetômetros ou suposições sobre um campo magnético homogêneo.

No trabalho de Han, Wong e Murray (2018), foi desenvolvido um algoritmo eficiente de estimação de erro de dois pontos para calcular com precisão os ângulos de *pitch*, *roll* e *yaw* da coxa e perna, utilizando apenas acelerômetros e giroscópios. Essa abordagem elimina a dependência de magnetômetros, reduzindo a complexidade e o custo do sistema.

Pacini Panebianco et al. (2018) realizaram uma revisão abrangente sobre algoritmos de detecção de eventos de marcha utilizando os sensores inerciais. Os autores destacaram que o desempenho desses algoritmos é significativamente influenciado pelo posicionamento dos sensores, pelas variáveis analisadas e pelas estratégias computacionais empregadas. A colocação dos sensores nas regiões da coxa e perna mostrou ser eficaz para a análise temporal da marcha, o que reforça a importância de uma correta disposição dos sensores para a obtenção de dados precisos.

Além disso, Gui, Tang e Mukhopadhyay (2015a) enfatizaram a precisão alcançada ao combinar técnicas de filtragem complementar na fusão de dados de IMUs. Essa combinação

melhora a estimativa de inclinação e orientação de movimentos, essencial para aplicações que requerem alta fidelidade nos dados de movimento, como no controle de próteses.

Kluge et al. (2017) investigaram a validade concorrente e a confiabilidade de um sistema de medição inercial para a avaliação de parâmetros espaço-temporais da marcha, demonstrando que sistemas baseados em IMUs podem fornecer medições precisas comparáveis a sistemas de referência.

Essas contribuições reforçam a eficácia das IMUs na análise de marcha e sustentam a escolha dessas ferramentas neste trabalho.

2.2 Modelagem Preditiva em Próteses de Membros Inferiores

A predição dos ângulos articulares e fases da marcha é muito importante para o desenvolvimento de sistemas de próteses mais responsivos e adaptativos. Diversos estudos têm explorado o uso de redes neurais profundas para essa finalidade.

Zaroug et al. (2021) investigaram a aplicação de diferentes arquiteturas de redes LSTM para prever as trajetórias futuras da cinemática dos membros inferiores durante a marcha. Utilizando dados de cinemática coletados de participantes caminhando em velocidades preferenciais e impostas, eles demonstraram que *autoencoders* baseados em LSTM alcançaram os melhores resultados, com baixos erros quadráticos médios normalizados.

Por outro lado, Huang et al. (2019a) desenvolveram um modelo de rede neural recorrente profunda para prever os ângulos do joelho em tempo real, utilizando uma combinação de sinais eletromiográficos (EMG) e medições inerciais. O modelo foi implementado em um microcontrolador, evidenciando a possibilidade de aplicação em sistemas de próteses motorizadas. Eles alcançaram um horizonte preditivo de 50 ms com um baixo erro de predição de $\pm$2,93 graus.

No estudo de Su e Gutierrez-Farewik (2020), também foi apresentada uma aplicação de uma rede LSTM para a predição da trajetória da marcha e fase da marcha, aspectos vitais para dispositivos robóticos assistivos. Utilizando uma função de perda ponderada, o método melhorou significativamente a precisão da predição das trajetórias dos segmentos dos membros inferiores e das fases da marcha até 200 ms à frente.

Além desses, Huang et al. (2011a) exploraram a identificação contínua de modos de locomoção para pernas protéticas baseadas na fusão neuromuscular-mecânica, mostrando que a combinação de dados EMG e cinemáticos melhora a precisão na identificação dos modos de marcha. Young e Hargrove (2016) propuseram um método de classificação para reconhecimento de intenção independente do usuário em amputados transfemorais utilizando próteses de membros inferiores motorizadas, destacando a importância de modelos generalizáveis.

Karakish, Fouz e Elsawaf (2022) exploraram o uso de deep learning para prever trajetórias da marcha diretamente em um microcontrolador embarcado, oferecendo uma solução de baixo custo para o controle de próteses ativas. Eles compararam o desempenho de Redes Neurais Convolucionais e Perceptrons Multicamadas em um microcontrolador ESP32, concluindo que as CNNs, apesar de mais lentas, apresentaram melhor desempenho com dados de treinamento limitados.

2.3 Desafios e Oportunidades

Embora os modelos baseados em LSTM e outras arquiteturas recorrentes tenham demonstrado bom desempenho na predição de parâmetros da marcha, sua complexidade computacional representa um obstáculo significativo para a implementação em sistemas embarcados de baixo custo e recursos limitados (ABADADE et al., 2023). Esses modelos demandam considerável capacidade de processamento e memória, tornando-se inadequados para dispositivos que exigem operação em tempo real com restrições estritas de energia e *hardware*. Existe, portanto, uma lacuna notável na literatura em relação a soluções que consigam simultaneamente oferecer alto desempenho preditivo sem comprometer a operação em tempo real no contexto de sistemas embarcados. A maioria dos estudos foca em alcançar elevada precisão sem considerar as limitações práticas impostas pelo *hardware* embarcado, resultando em modelos impraticáveis para implementação em dispositivos reais de próteses.

Neste contexto, a utilização de redes convolucionais otimizadas para sistemas embarcados surge como uma alternativa promissora. As CNNs têm a capacidade de extrair características relevantes dos dados de entrada com menor sobrecarga computacional em comparação com modelos recorrentes (GHOLAMI; NAPIER; MENON, 2020). Além disso, técnicas de otimização e compressão de modelos podem ser aplicadas para adaptar as redes neurais às restrições de *hardware* dos microcontroladores (HEIM et al., 2021), permitindo a operação em tempo real sem sacrificar significativamente a precisão.

Han et al. (2021) apresentam uma pesquisa abrangente sobre redes neurais dinâmicas, discutindo como essas redes podem ajustar sua estrutura e comportamento em resposta a diferentes entradas ou condições, o que é altamente relevante para sistemas de próteses que devem se adaptar a diversas condições de marcha em tempo real. As redes neurais dinâmicas oferecem flexibilidade e eficiência computacional, tornando-se adequadas para implementação em sistemas embarcados com recursos limitados.

A implementação de modelos especializados para diferentes tipos de marcha, associada a um sistema de classificação que selecione o modelo apropriado em tempo real, pode melhorar ainda mais o desempenho dos sistemas de próteses. Essa abordagem permite que cada modelo seja ajustado para capturar as nuances específicas de cada padrão de

marcha, aumentando a precisão das predições e a responsividade do sistema, ao mesmo tempo em que mantém a eficiência computacional necessária para operação em tempo real em sistemas embarcados.

2.4 Considerações Finais

A revisão bibliográfica evidencia a evolução das técnicas de aquisição de dados e modelagem preditiva na área de próteses de membros inferiores. Há um movimento crescente em direção ao desenvolvimento de soluções que conciliem alta qualidade de previsão com eficiência computacional, visando a implementação em sistemas embarcados acessíveis.

Este trabalho propõe contribuir para essa tendência, explorando o potencial das CNNs otimizadas para sistemas embarcados e abordagens de modelos especializados. Ao enfrentar os desafios identificados na literatura, espera-se avançar na direção de próteses mais inteligentes, responsivas e acessíveis, impactando positivamente a qualidade de vida de indivíduos amputados.

3 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos teóricos e os modelos de redes neurais relevantes para o desenvolvimento do sistema de controle para próteses ativas de joelho. Esta fundamentação oferece as bases necessárias para o entendimento das arquiteturas de aprendizado de máquina e técnicas de processamento de sinais utilizadas no trabalho. Primeiramente, abordam-se os modelos de redes neurais tradicionais, como Perceptron Multicamadas (MLP) e Redes Neurais Convolucionais (CNNs), e suas variantes específicas para dados unidimensionais e sequenciais. Em seguida, exploram-se as Redes LSTM, amplamente usadas para processamento de dados temporais. Em seguida, são apresentadas as métricas de avaliação e parâmetros de modulação de complexidade, que foram essenciais para a escolha e otimização das arquiteturas propostas. Finalmente, discute-se a importância da aquisição de dados de movimento por meio de sensores inerciais e o uso de técnicas de *data augmentation* para aprimorar a robustez e a generalização dos modelos.

Por que *deep learning*?
Embora abordagens clássicas de aprendizado de máquina (por exemplo, árvores de decisão ou regressão linear) possam ser inicialmente mais simples de interpretar, a conversão desses modelos em C puro (por meio de ferramentas como o `m2cgen`) tende a resultar em códigos excessivamente extensos e de desempenho inferior em microcontroladores de recursos limitados. Por outro lado, os *frameworks* de *deep learning* otimizados para sistemas embarcados, como o `TensorFlow Lite Micro`, viabilizam a execução de redes neurais em dispositivos de baixa potência de forma eficiente. Além disso, diversos *hardwares* modernos - incluindo o Kendryte K210, utilizado neste trabalho - contam com aceleradores específicos para redes neurais, proporcionando inferência mais veloz e consumo energético reduzido. Essa combinação de ferramentas e suporte especializado torna as redes neurais profundas (como MLPs, CNNs e LSTMs) uma escolha particularmente vantajosa para aplicações em *TinyML*, permitindo maior qualidade preditiva e melhor otimização de recursos em um cenário de restrições computacionais.

3.1 Perceptron Multicamadas (MLP)

O Perceptron Multicamadas (MLP) é uma classe de redes neurais artificiais *feedforward*, que consiste em pelo menos três camadas de nós: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Com exceção dos nós de entrada, cada nó é um neurônio que utiliza uma função de ativação não linear. O MLP aproveita uma técnica de aprendizado supervisionado chamada *backpropagation* para treinamento

(RUMELHART; HINTON; WILLIAMS, 1986).

A arquitetura de um MLP é composta por:

- **Camada de Entrada**: Recebe os dados de entrada. O número de neurônios nesta camada é igual ao número de características do conjunto de dados.
- **Camadas Ocultas**: Processam as informações recebidas da camada de entrada. O número de camadas ocultas e de neurônios em cada camada pode variar dependendo da complexidade do problema. Cada neurônio em uma camada está conectado a todos os neurônios da próxima camada.
- **Camada de Saída**: Produz o resultado final da rede. O número de neurônios nesta camada depende do número de classes ou do horizonte de predição (para problemas de classificação ou *forecasting*) ou pode ser um único neurônio (para problemas de regressão). Como mostrado na Figura 4, o MLP para problemas de regressão possui múltiplas camadas ocultas e uma única saída contínua. A Figura 5 ilustra um MLP para problemas de classificação ou de *forecasting* de multiplas saídas, onde a camada de saída possui múltiplos neurônios correspondentes às classes ou horizonte de predição.

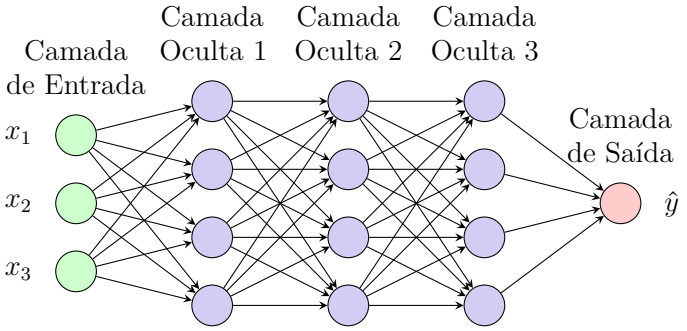


Figura 4 – Arquitetura de um MLP para problemas de regressão com múltiplas camadas ocultas e uma saída contínua.

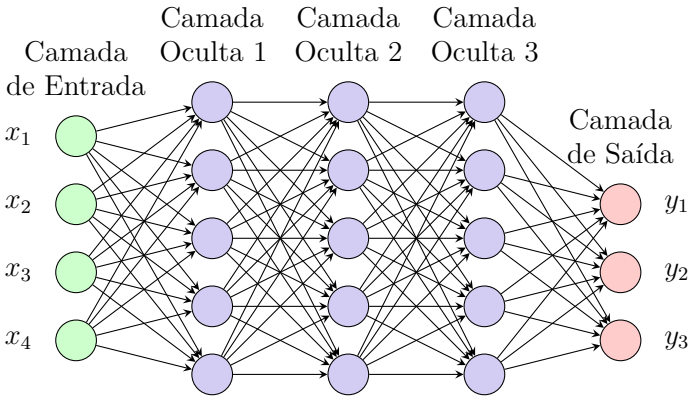


Figura 5 – Arquitetura de um MLP para problemas de classificação com múltiplas camadas ocultas e múltiplas saídas correspondentes às classes.

Os MLPs têm sido amplamente utilizados em diversas áreas, como reconhecimento de padrões, incluindo classificação de imagens, reconhecimento de escrita manual e reconhecimento de voz (SCHMIDHUBER, 2015). Além disso, são aplicados na previsão de séries temporais, permitindo a previsão de valores futuros com base em dados históricos (LIM; ZOHREN; ROBERTS, 2021). Também são empregados em sistemas de controle, como controle preditivo e modelagem de sistemas dinâmicos (HAN; ZHANG; ZHANG, 2019). No processamento de sinais, os MLPs são utilizados para filtragem e análise de sinais em aplicações biomédicas e de engenharia (WANG; CHEN, 2018).

3.2 Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (CNNs) são uma classe especializada de redes neurais projetadas para processar dados que possuem uma topologia em grade, como imagens ou sinais temporais. As CNNs reconhecem efetivamente padrões espaciais e temporais nos dados, tornando-as altamente adequadas para aplicações de visão computacional e análise de séries temporais (LECUN; BENGIO, 1998).

A arquitetura de uma CNN tipicamente consiste em várias camadas, incluindo camadas convolucionais, de *pooling* e totalmente conectadas. As camadas convolucionais aplicam uma operação matemática chamada convolução, que processa os dados de entrada com a ajuda de filtros ou kernels. Esses filtros permitem que a rede aprenda características específicas de baixo e alto nível a partir dos dados.

Durante a operação de convolução, cada filtro em uma camada convolucional aplica uma convolução aos dados de entrada da camada anterior. A operação de convolução em uma CNN bidimensional é representada pela Equação 3.1.

$$a_i^{(l)} = \sigma\left(\sum_{m,n} K_{mn}^{(l)} \cdot x_{(i+m)(j+n)}^{(l-1)} + b_i^{(l)}\right) \tag{3.1}$$

Nesta equação, $a_i^{(l)}$ representa a ativação do i-ésimo neurônio na l-ésima camada, que é computada como uma soma ponderada das entradas da camada $l-1$, modificada pelo filtro $K_{mn}^{(l)}$. Os índices m e n percorrem as dimensões espaciais do filtro, e $(i+m)(j+n)$ indexa a posição correspondente nos dados de entrada. A função σ representa a função de ativação, como a Unidade Linear Retificada ($ReLU$), que introduz não linearidade ao modelo. O termo $b_i^{(l)}$ é o viés que ajusta a saída juntamente com a soma ponderada.

Para ilustrar melhor o funcionamento das CNNs, a Figura 6 apresenta a arquitetura típica de uma Rede Neural Convolucional. Nessa estrutura, é possível visualizar as diferentes camadas que compõem a rede: camadas convolucionais, de *pooling* e totalmente conectadas. As camadas convolucionais aplicam filtros aos dados de entrada para extrair características locais, enquanto as camadas de *pooling* reduzem a dimensionalidade dos dados, mantendo

as informações mais relevantes. As camadas totalmente conectadas interpretam essas características para realizar a tarefa de classificação ou regressão desejada.

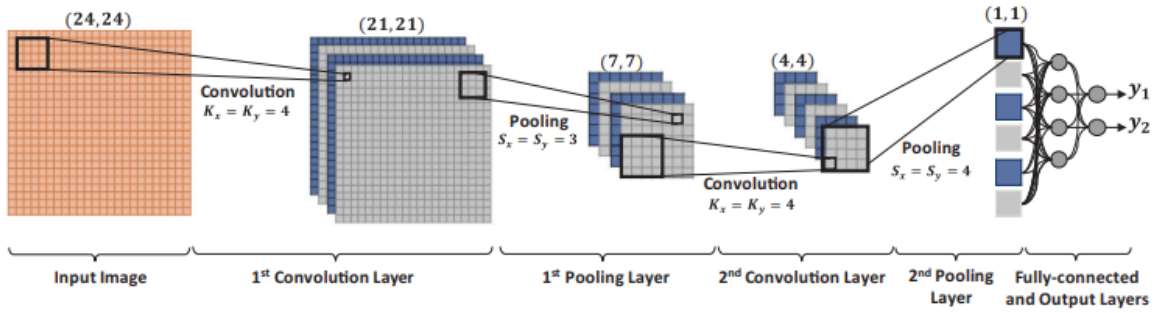


Figura 6 – Arquitetura típica de uma Rede Neural Convolucional (CNN), mostrando as camadas convolucionais, de *pooling* e totalmente conectadas. Fonte: (KIRANYAZ et al., 2021)

3.2.1 Redes Neurais Convolucionais Unidimensionais (CNN1D)

Embora as CNNs tenham sido amplamente utilizadas para processar dados bidimensionais, como imagens, elas também podem ser adaptadas para lidar com dados unidimensionais, como séries temporais e sinais sequenciais. Nesse contexto, destacam-se as CNN1D, que são especialmente projetadas para processar dados estruturados em uma única dimensão (KIRANYAZ et al., 2021).

A principal diferença entre as CNNs 1D e 2D reside na dimensionalidade dos dados de entrada e dos filtros utilizados nas camadas convolucionais. Enquanto as CNNs 2D aplicam filtros bidimensionais para capturar padrões espaciais em imagens, as CNNs 1D utilizam filtros unidimensionais que se deslocam ao longo da sequência de dados, permitindo a extração de características locais em séries temporais.

A convolução em uma CNN1D pode ser representada pela Equação 3.2.

$$a_i^{(l)} = \sigma\left(\sum_{k=1}^{K} w_k^{(l)} \cdot x_{i+k-1}^{(l-1)} + b^{(l)}\right) \tag{3.2}$$

Nesta equação, $w_k^{(l)}$ representa o peso do k-ésimo elemento do filtro na camada l, $x_{i+k-1}^{(l-1)}$ é a entrada correspondente na posição $i+k-1$ da camada anterior, e $b^{(l)}$ é o viés da camada l. A função de ativação σ introduz não linearidade ao modelo, permitindo a aprendizagem de representações complexas nos dados.

Para exemplificar a operação de convolução em uma CNN unidimensional, a Figura 7 demonstra como um filtro 1D é aplicado a um sinal de entrada unidimensional. Nessa ilustração, o filtro se desloca ao longo do sinal de entrada, posição por posição, realizando multiplicações ponto a ponto entre os valores do filtro e os valores correspondentes do sinal.

Em seguida, os produtos são somados para produzir a ativação correspondente àquela posição.

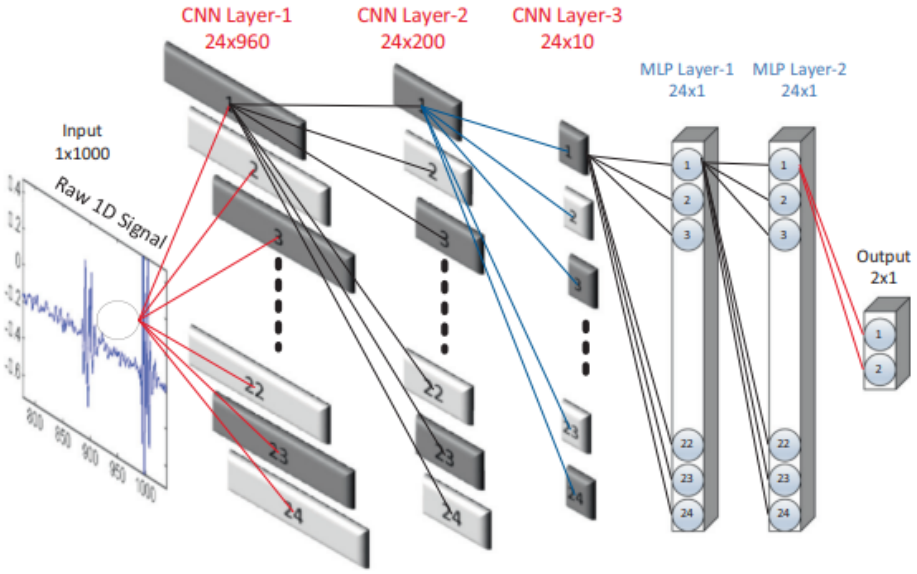


Figura 7 – Operação de convolução em uma CNN unidimensional. O filtro 1D se desloca ao longo do sinal de entrada, multiplicando e somando os valores para gerar as ativações. Fonte: (KIRANYAZ et al., 2021)

As CNN1D têm demonstrado excelente desempenho em diversas aplicações que envolvem dados sequenciais e temporais, como a análise de sinais biomédicos, onde são muito utilizadas na análise de sinais ECG para detecção precoce de arritmias e outras anomalias cardíacas, devido à sua capacidade de aprender diretamente das formas de onda brutas, eliminando a necessidade de extração manual de características e aumentando a precisão e eficiência do diagnóstico (KIRANYAZ; INCE; GABBOUJ, 2016). Além disso, são empregadas no reconhecimento de fala, permitindo a transcrição de áudio em texto e a detecção de comandos de voz em sistemas embarcados (SAINATH et al., 2015). Em engenharia civil e mecânica, as CNN1D são utilizadas para o monitoramento de integridade estrutural, detectando danos e falhas em estruturas e máquinas através da análise de dados de vibração coletados por sensores, o que possibilita intervenções de manutenção preditiva, aumentando a segurança e reduzindo custos operacionais (ABDELJABER et al., 2017). Também são aplicadas na detecção de falhas em motores elétricos, monitorando em tempo real para identificar anomalias e falhas, contribuindo para a melhoria da confiabilidade de sistemas industriais (ZHANG et al., 2017).

Uma vantagem significativa das CNN1D em relação à sua contraparte 2D é a menor complexidade computacional. Devido ao processamento unidimensional, as CNN1D requerem menos parâmetros e operações matemáticas, tornando-as ideais para aplicações em tempo real e em dispositivos com menos recursos computacionais. Além disso, as CNN1D podem ser treinadas efetivamente com conjuntos de dados menores, o que é

particularmente útil em domínios onde a coleta de dados rotulados é desafiadora ou custosa (KIRANYAZ et al., 2021).

3.3 Redes LSTM (*Long Short-Term Memory*)

As redes LSTM são um tipo especializado de redes neurais recorrentes (RNNs) projetadas para resolver o problema de dependências de longo prazo em dados sequenciais. Ao contrário das RNNs tradicionais, que lutam para manter informações ao longo de sequências extensas, as LSTMs são equipadas com uma arquitetura única que inclui células de memória e portas, permitindo que retenham informações por períodos mais longos de forma eficaz (HOCHREITER; SCHMIDHUBER, 1997a).

O núcleo de uma unidade LSTM consiste em um estado de célula, que atua como uma esteira transportadora carregando informações ao longo do processamento da sequência. Este estado de célula é modulado por três portas distintas: a porta de entrada, porta de esquecimento e porta de saída. Essas portas controlam o fluxo de informações para dentro e para fora da célula, bem como a retenção e remoção de informações do estado da célula. As operações de uma LSTM podem ser descritas pelas Equações 3.3 e pela Figura 8 abaixo:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Porta de Esquecimento)} \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Porta de Entrada)} \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Estado de Célula Candidato)} \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t && \text{(Atualização do Estado de Célula)} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Porta de Saída)} \\
h_t &= o_t * \tanh(C_t) && \text{(Saída)}
\end{aligned}
\tag{3.3}
$$

Onde σ denota a função de ativação sigmoide, tanh é a função tangente hiperbólica, W e b são os pesos e vieses aprendidos, h_{t-1} é a saída anterior, x_t é a entrada atual, f_t é a porta de esquecimento, i_t é a porta de entrada, $\tilde{C}_t$ é o estado de célula candidato, C_t é o estado de célula atualizado, o_t é a porta de saída e h_t é a saída atual.

As redes LSTM têm sido utilizadas em diversos domínios que envolvem dados sequenciais e temporais, devido à sua capacidade de modelar dependências de longo prazo. Uma das principais aplicações é o Processamento de Linguagem Natural (PLN), onde as LSTMs são empregadas em tarefas como tradução automática (BAHDANAU; CHO; BENGIO, 2015), geração de linguagem natural (SUTSKEVER; VINYALS; LE, 2014), análise de sentimento (TANG; QIN; LIU, 2015) e modelagem de linguagem (JOZEFOWICZ et al., 2016), permitindo que os modelos captem contextos de longo alcance em sequências de texto. Além disso, no reconhecimento e síntese de fala, as LSTMs têm melhorado

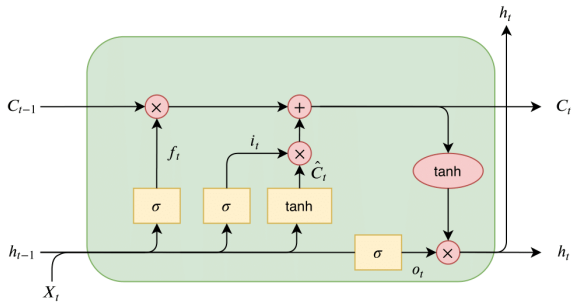


Figura 8 – Ilustração de uma camada LSTM. Fonte: (MAR, 2023)

significativamente a precisão no reconhecimento automático de fala ao lidar com as dependências temporais do áudio (GRAVES; MOHAMED; HINTON, 2013) e são utilizadas em síntese de voz, como nos modelos de conversão texto-fala (FAN et al., 2014).

Outra aplicação importante é a previsão de séries temporais, onde as LSTMs são eficazes na previsão de séries temporais complexas, incluindo previsões financeiras (FISCHER; KRAUSS, 2018), demandas de energia (ZHENG; ZHU; ZHANG, 2017) e monitoramento de tráfego (MA et al., 2015). Além disso, nas análises de dados biomédicos, as LSTMs são aplicadas na detecção de arritmias cardíacas a partir de eletrocardiogramas (XIA et al., 2018), reconhecimento de gestos em sinais de eletromiografia (DU et al., 2017) e na análise de sinais de EEG para detecção de epilepsia (HUSSAIN; PARK, 2019).

3.4 Avaliação do Desempenho dos Modelos

Para medir o desempenho dos sistemas, foram utilizadas as seguintes métricas:

3.4.1 Raiz do Erro Quadrático Médio (RMSE)

O RMSE é uma métrica comumente usada para avaliar a precisão de um modelo, medindo a raiz quadrada da média das diferenças quadradas entre os valores previstos e observados. É definido pela Equação 3.4:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.4}$$

onde y_i representa o valor real, $\hat{y}_i$ o valor previsto e n o número de observações. O RMSE fornece uma estimativa da magnitude dos erros de predição e é sensível a grandes erros.

3.4.2 Erro Quadrático Médio (MSE)

O MSE calcula a média dos quadrados das diferenças entre os valores previstos e observados. É uma medida da qualidade do estimador e é definido pela Equação 3.5:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

O MSE penaliza erros maiores mais do que erros menores (eleva as diferenças ao quadrado), sendo útil para avaliar modelos onde erros significativos são particularmente indesejáveis.

3.4.3 Erro da Raiz Quadrada (RSE)

O Erro da Raiz Quadrada (RSE) mede a diferença absoluta entre os valores previstos e observados em cada ponto, fornecendo uma medida direta da magnitude do erro individual. É definido como:

$$\text{RSE}_i = \sqrt{(y_i - \hat{y}_i)^2} \tag{3.6}$$

O RSE é útil para avaliar o erro ponto a ponto nas predições do modelo, permitindo identificar discrepâncias específicas entre os valores previstos e observados.

3.4.4 Coeficiente de Determinação (R^2)

O coeficiente de determinação, conhecido como R^2, é uma medida estatística que explica o quão bem os resultados observados são replicados pelo modelo, baseado na proporção da variação total dos resultados explicada pelo modelo. É definido como:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{3.7}$$

onde y_i representa os valores reais, $\hat{y}_i$ os valores previstos, $\overline{y}$ a média dos valores reais e n o número de observações. Um R^2 de 1 indica que o modelo prevê perfeitamente os resultados observados, enquanto um R^2 de 0 indica que o modelo não explica nenhuma variabilidade nos resultados observados em relação à média.

3.5 Modulação de Complexidade com α e β

Os parâmetros α e β foram definidos para controlar as arquiteturas das redes neurais, ajustando respectivamente sua largura e profundidade. A modulação de α afeta o número de neurônios ou filtros por camada, influenciando diretamente a capacidade da

rede, o que é crucial para o desempenho e a generalização, especialmente em dispositivos com recursos limitados (HEIM et al., 2021). A modulação de β ajusta o número de camadas ou blocos, impactando a profundidade da rede e sua capacidade de aprender representações hierárquicas.

3.5.1 Modulação de Complexidade com α

O parâmetro α foi definido como um fator criado para padronizar e controlar as arquiteturas testadas, ajustando a largura das redes neurais. Ao controlar o número de neurônios ou filtros em cada camada por meio de α, a capacidade representacional da rede é influenciada diretamente, o que tem implicações significativas no desempenho e na generalização do modelo. Especialmente em dispositivos com recursos limitados, como MCUs, otimizar a largura da rede é essencial para atender às restrições de memória e computação (HEIM et al., 2021).

MLP: Em redes MLP, aumentar α incrementa o número de neurônios em cada camada oculta. Isso permite que a rede capture relações não lineares mais complexas nos dados. No entanto, um número excessivo de neurônios pode levar ao sobreajuste (*overfitting*), onde o modelo se ajusta muito bem aos dados de treinamento, mas tem um desempenho ruim em dados não vistos.

CNN: Nas CNNs, α ajusta o número de filtros em cada camada convolucional. Um maior número de filtros permite que a rede aprenda uma variedade mais ampla de características nas imagens, incluindo texturas, bordas e padrões complexos. No entanto, isso também aumenta o número de parâmetros, o que pode levar a um maior risco de sobreajuste e a um aumento significativo no uso de memória e no tempo de processamento. Em aplicações de *TinyML*, onde os recursos são limitados, é importante equilibrar a largura da rede para otimizar o desempenho sem exceder as restrições de *hardware* (HEIM et al., 2021).

LSTM: Para LSTMs, α determina o número de unidades de memória em cada camada. Mais células de memória permitem que a rede capture dependências temporais mais complexas em séries temporais ou sequências textuais. Contudo, redes LSTM mais largas podem enfrentar desafios como aumento do tempo de treinamento e necessidade de mais dados para evitar sobreajuste.

3.5.2 Modulação de Profundidade com β

O parâmetro β foi definido como um fator criado para padronizar e controlar a profundidade das arquiteturas testadas. O parâmetro β está associado à profundidade da rede, influenciando o número de camadas ou blocos de camadas que a rede possui. Redes mais profundas têm a capacidade de aprender representações hierárquicas dos dados, mas

também apresentam desafios únicos, especialmente em dispositivos com recursos limitados (HEIM et al., 2021).

MLP: Em MLPs, aumentar β adiciona mais camadas ocultas. Isso permite que a rede capture interações de alto nível entre os neurônios, potencialmente melhorando a capacidade de modelar relações complexas nos dados. Entretanto, redes muito profundas podem sofrer com o problema de gradientes desaparecendo, dificultando o treinamento eficaz das camadas mais iniciais.

CNN: Nas CNNs, β representa a adição de mais blocos convolucionais e de *pooling*. No entanto, a profundidade adicional pode levar a dificuldades de treinamento devido ao desaparecimento do gradiente e à saturação da precisão.

LSTM: Para LSTMs, aumentar β implica adicionar mais camadas empilhadas de unidades LSTM. Isso pode melhorar a capacidade da rede de aprender padrões em diferentes escalas temporais. No entanto, a profundidade adicional aumenta a complexidade computacional e o risco de problemas de treinamento.

3.6 Padrões Arquiteturais com Base na Modulação de α e β

A aplicação dos parâmetros α e β para ajustar a largura e a profundidade das redes permite adotar padrões arquiteturais que conciliam complexidade, capacidade de representação e eficiência, os padrões discutidos a seguir podem ser encontrados no Apêndice C. Em MLPs, é comum empregar um formato "em funil", onde as primeiras camadas são mais largas e, gradualmente, o número de neurônios diminui nas camadas posteriores. Esse padrão, discutido em abordagens gerais de *deep learning* (GOODFELLOW; BENGIO; COURVILLE, 2016), ajuda a extrair características gerais nas primeiras camadas e especializá-las nas últimas, ao mesmo tempo em que contribui para evitar o sobreajuste e reduzir os custos computacionais.

No caso de CNNs, um padrão amplamente utilizado envolve aumentar o número de filtros conforme a profundidade da rede cresce, partindo de um número menor nas camadas iniciais e dobrando-os em camadas mais profundas, como abordado em arquiteturas como a VGG (SIMONYAN; ZISSERMAN, 2015). Esse método permite que camadas iniciais capturem padrões simples, enquanto camadas mais profundas representem características complexas, mantendo um equilíbrio entre expressividade e uso de recursos, principalmente em contextos de dispositivos com recursos limitados (HEIM et al., 2021).

Para LSTMs, o controle de α e β pode manter um número fixo de unidades por camada ou ajustar a quantidade de forma sistemática. Embora existam menos padrões "clássicos" para a distribuição de unidades em LSTMs, a ideia de empilhar múltiplas camadas (aumentando β) é bem estabelecida (HOCHREITER; SCHMIDHUBER, 1997b).

Dessa forma, é possível aumentar a capacidade de extrair dependências temporais complexas sem tornar o modelo desnecessariamente complicado ou difícil de treinar.

3.6.1 Uso e ilustração do Controle das arquiteturas

A Figura 9 fornece uma representação visual de como α e β afetam a arquitetura de modelo MLP. Ao ajustar esses parâmetros, é possível explorar um espectro de arquiteturas que equilibram desempenho e eficiência computacional, adaptando o modelo às restrições específicas da aplicação, especialmente em ambientes com recursos limitados.

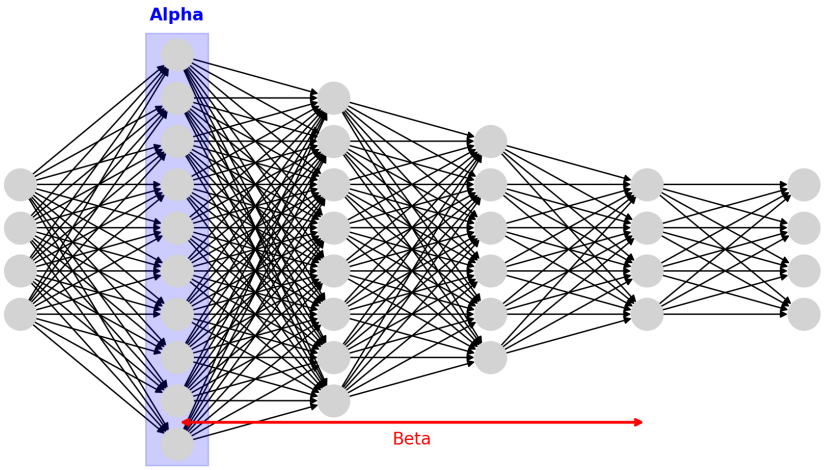


Figura 9 – Representação visual de como α e β controlam o tamanho do modelo.

Variando os valores de α e β, busca-se encontrar um ponto ótimo entre precisão e tamanho do modelo. Inicialmente, o aumento do tamanho do modelo tende a melhorar a precisão; entretanto, chega-se a um ponto de saturação onde a precisão não apresenta ganhos significativos, independentemente de quão grande o modelo se torne (FRANKLE; CARBIN, 2019; KAPLAN et al., 2020). Isso ocorre porque a capacidade adicional do modelo não contribui para uma melhor generalização, mas aumenta o consumo de recursos computacionais (LECUN; BENGIO; HINTON, 2015). Portanto, é essencial identificar esse equilíbrio para otimizar o desempenho do modelo em relação às restrições de *hardware* e requisitos da aplicação. Desta forma, o ajuste de α e β deve levar em consideração as limitações do *hardware* e os requisitos da aplicação. Conforme discutido por Heim et al. (2021), em sistemas com recursos restritos, como dispositivos de *TinyML*, otimizar a arquitetura da rede para atender às restrições de memória e energia é fundamental.

3.7 Seleção dos Modelos

3.7.1 Fronteira de Pareto

A fronteira de Pareto é um conceito utilizado para determinar o conjunto de soluções que oferecem o melhor compromisso entre múltiplos objetivos conflitantes. No contexto da seleção de modelos, a fronteira de Pareto é crucial para equilibrar objetivos como minimizar o erro de predição e a complexidade computacional. A aplicação dessa abordagem permite identificar modelos que oferecem o melhor equilíbrio entre precisão e eficiência, sem que a melhoria em um objetivo resulte em detrimento de outro (DEB, 2001). No contexto deste trabalho, aplicou-se a fronteira de Pareto para identificar os modelos que equilibram da melhor forma a precisão e o tamanho do modelo, controlados pelos parâmetros α e β.

Na prática, a fronteira de Pareto é composta por soluções não dominadas, ou seja, soluções para as quais não existe outra que seja melhor em todos os critérios simultaneamente.

Ao variar o tamanho do modelos, uma gama de modelos foi gerada (discutidos nas seções 3.1, 3.2 e 3.3) com diferentes arquiteturas. Esses parâmetros influenciam diretamente o tamanho das redes, afetando tanto a complexidade computacional quanto a capacidade de generalização do modelo. Plotando o tamanho do modelo contra sua performance, foi obtido um conjunto de soluções onde algumas se destacam por não serem superadas em ambos os critérios simultaneamente. Essas soluções formam a fronteira de Pareto.

A fronteira de Pareto tem sido amplamente aplicada em diversas áreas que envolvem problemas de otimização com múltiplos objetivos conflitantes. Algumas das principais aplicações incluem:

- **Engenharia e Design de Sistemas**: Na engenharia mecânica e aeroespacial, a fronteira de Pareto é utilizada para otimizar projetos de componentes e sistemas, equilibrando fatores como peso, resistência e custo (DEB; JAIN, 2014). Por exemplo, no projeto de asas de aeronaves, é necessário equilibrar a eficiência aerodinâmica com a resistência estrutural e o peso.

- **Aprendizado de Máquina e Seleção de Modelos**: Na seleção de modelos de aprendizado de máquina, a fronteira de Pareto é aplicada para equilibrar a complexidade do modelo e o desempenho preditivo (JIN; SENDHOFF, 2001). Modelos mais complexos podem oferecer melhor desempenho, mas também podem ser propensos a overfitting e requerer mais recursos computacionais.

- **Planejamento de Recursos Energéticos**: Na otimização de sistemas de energia, como redes elétricas inteligentes, a fronteira de Pareto auxilia no balanceamento entre

custo operacional, emissões de carbono e confiabilidade do fornecimento (ZHANG; EVANGELISTI; LETTIERI, 2013).

3.7.2 Métricas Ponderadas

Complementando a análise da fronteira de Pareto, as métricas ponderadas permitem priorizar entre as escolhas ótimas com base na importância relativa de cada critério. Após normalizar as métricas para uma escala comum, foram atribuídos pesos que refletem a relevância estratégica de cada uma. A pontuação final de cada modelo foi calculada multiplicando cada métrica normalizada pelo seu peso correspondente e somando os resultados, conforme ilustrado na Equação (3.8):

$$\text{Pontuação Ponderada} = \sum_{i=1}^{n} w_i \times \text{metrica\_norm}_i \tag{3.8}$$

Onde n é o número de critérios e w_i é o peso do critério e metrica_norm_i é a métrica normalizada correspondente. Essa abordagem permite incorporar preferências específicas e direcionar a busca por soluções que atendam melhor aos objetivos estratégicos do problema.

O uso de métricas ponderadas é comum em situações onde é necessário combinar múltiplos critérios em uma única função objetivo. Algumas das aplicações incluem:

- **Planejamento Urbano e Ambiental**: Na avaliação de impactos ambientais, métricas ponderadas são utilizadas para combinar diferentes indicadores, como qualidade do ar, ruído e uso do solo, refletindo as prioridades da comunidade (SAATY, 2008).

- **Sistemas de Recomendação**: Em plataformas de comércio eletrônico e streaming, métricas ponderadas ajudam a equilibrar a relevância dos itens recomendados com a novidade e diversidade, melhorando a satisfação do usuário (ABDOLLAHPOURI; BURKE; MOBASHER, 2019).

- **Gerenciamento de Riscos**: Em gestão de projetos, os riscos são avaliados considerando a probabilidade de ocorrência e o impacto potencial, utilizando pesos para refletir a importância relativa de diferentes tipos de riscos (HILLSON, 2002).

- **Desenvolvimento de Produtos**: Na engenharia de produtos, métricas ponderadas são aplicadas para equilibrar atributos como custo, desempenho, confiabilidade e tempo de mercado (COOPER, 2001).

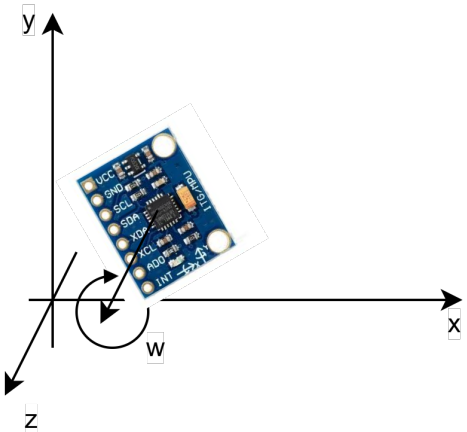
3.8 Aquisição e Tratamento dos Dados

3.8.1 Processamento de Sinais

As Unidades de Medição Inercial (IMUs) são sensores amplamente utilizados para capturar informações sobre o movimento e a orientação de um objeto no espaço tridimensional. Uma IMU típica combina diferentes sensores, como acelerômetros, giroscópios e, em alguns casos, magnetômetros, para fornecer dados precisos sobre aceleração linear, velocidade angular e orientação magnética. Os acelerômetros medem as forças de aceleração em três eixos (X, Y e Z), permitindo a detecção de movimentos lineares e a influência da gravidade. Os giroscópios registram as taxas de rotação ao redor desses mesmos eixos, proporcionando informações sobre a orientação angular. A integração desses sensores permite que as IMUs monitorem de forma contínua a posição, velocidade e orientação de dispositivos.

O uso isolado dos dados do acelerômetro ou do giroscópio apresenta limitações significativas: o acelerômetro é sensível a ruídos de alta frequência e acelerações lineares, resultando em medições instáveis (SABATINI, 2011; SEEL; RAISCH; SCHAUER, 2014); já o giroscópio, ao fornecer taxas de rotação que requerem integração ao longo do tempo, acumula erros que levam a uma deriva significativa (MADGWICK; HARRISON; VAIDYANATHAN, 2011; ZIHAJEHZADEH; PARK, 2016).

A estimativa do ângulo baseada apenas no giroscópio é obtida pela integração da velocidade angular, conforme ilustrado na equação 3.9 onde $\omega(t)$ é a velocidade angular medida pelo giroscópio. A figura 10 ilustra a velocidade angular no eixo z. No entanto, devido a imprecisões no sensor e ruídos de medição, a integração contínua tende a introduzir erros cumulativos.



$$\theta_{\text{gyro}}(t) = \theta_{\text{gyro}}(t_0) + \int_{t_0}^{t} \omega(t')\, dt' \quad (3.9)$$

Figura 10 – MPU6050 inclinado em relação aos eixos de referência, capaz de medir a velocidade angular a partir de seus eixos internos.

Por outro lado, o acelerômetro fornece uma estimativa do ângulo com base na componente gravitacional:

$$\theta_{\text{acc}} = \arctan\left(\frac{a_y}{a_z}\right), \tag{3.10}$$

onde a_y e a_z são as leituras do acelerômetro nos eixos y e z, respectivamente. Contudo, o acelerômetro é sensível a acelerações lineares e ruídos de alta frequência.

A Figura 11 ilustra um módulo de aceleração inclinado, mostrando como a componente gravitacional se decompõe nos eixos x e y. A partir dessa decomposição, é possível determinar o ângulo de inclinação do sensor em relação ao plano horizontal.

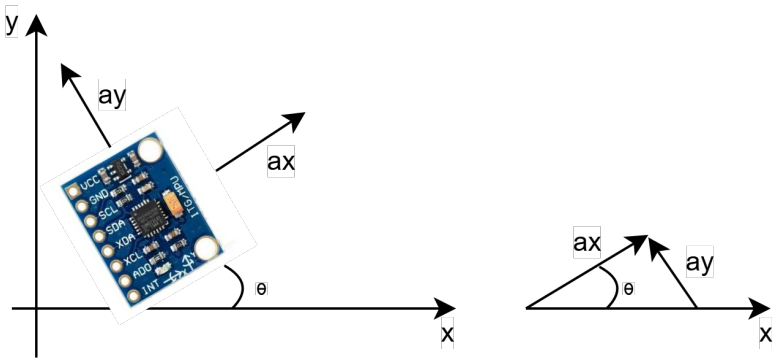


Figura 11 – Representação do ângulo estimado pelo acelerômetro. A figura ilustra um módulo de aceleração inclinado, permitindo a decomposição da componente gravitacional nos eixos x (ax) e y (ay). A partir dessa decomposição, é possível determinar o ângulo de inclinação θ do sensor em relação ao plano horizontal.

A Figura 12 apresenta os dados coletados de um acelerômetro (Acc), giroscópio (Gyr) e o ângulo combinado (Cmb) ao longo do tempo. O gráfico ilustra a natureza complementar desses sensores: o giroscópio (linha verde) fornece informações rápidas sobre mudanças angulares, mas apresenta deriva ao longo do tempo, enquanto o acelerômetro (linha vermelha) exibe ruídos significativos em mudanças abruptas. A combinação de ambos os sinais (linha azul) resulta em uma estimativa mais precisa da orientação.

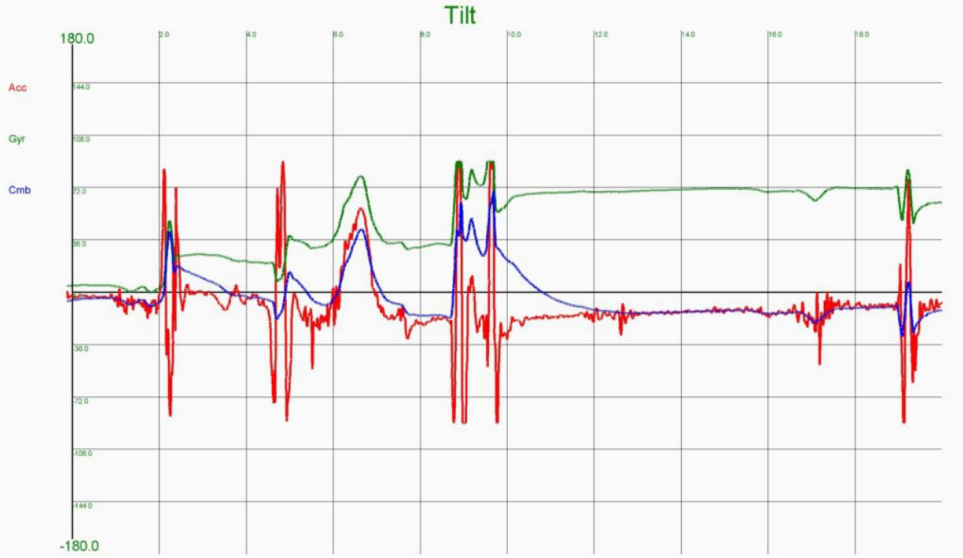


Figura 12 – Dados de sensores de uma IMU: aceleração (Acc), velocidade angular (Gyr) e combinação resultante (Cmb). (Xplicity, 2023)

Para superar essas limitações, empregou-se um filtro complementar que combina os dados do acelerômetro e do giroscópio (GUI; TANG; MUKHOPADHYAY, 2015b). Essa técnica aproveita as características complementares de cada sensor: o acelerômetro fornece informações confiáveis de baixa frequência sobre a orientação, enquanto o giroscópio oferece dados precisos de alta frequência sobre mudanças na orientação.

A implementação básica do filtro complementar pode ser expressa pela equação:

$$\theta_{\text{filtro}}(t) = \alpha \left[\theta_{\text{filtro}}(t - \Delta t) + \omega(t)\Delta t\right] + (1 - \alpha)\theta_{\text{accel}}(t), \tag{3.11}$$

onde $\theta_{\text{filtro}}(t)$ é o ângulo estimado pelo filtro no tempo t, $\theta_{\text{filtro}}(t - \Delta t)$ é o ângulo estimado no instante anterior, $\omega(t)$ é a velocidade angular do giroscópio, Δt é o intervalo de tempo entre as amostras, $\theta_{\text{acc}}(t)$ é o ângulo estimado pelo acelerômetro, e α é um coeficiente de filtragem que determina a contribuição de cada sensor (tipicamente $0{,}98 \leq \alpha \leq 0{,}99$ (ADAMS, 2024)).

Essa equação representa a combinação de um filtro passa-altas aplicado aos dados do giroscópio e um filtro passa-baixas aplicado aos dados do acelerômetro, ilustrado na Figura 13. O termo α controla a frequência de corte do filtro, permitindo ajustar a resposta do sistema para melhor desempenho em diferentes condições.

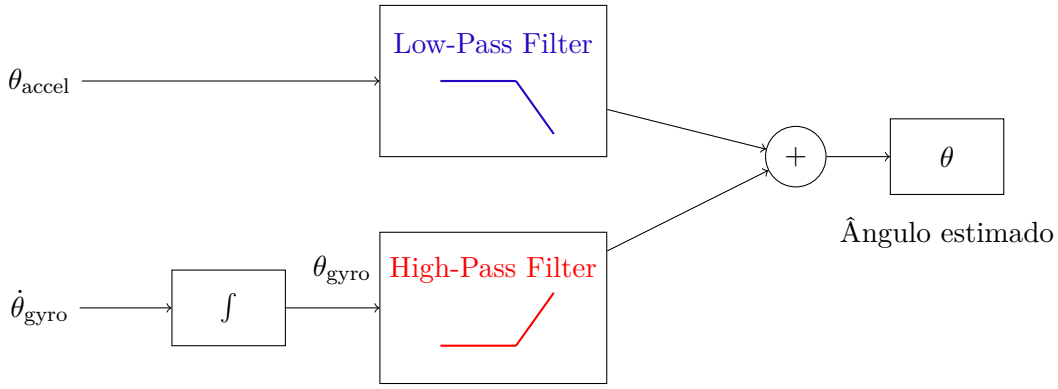


Figura 13 – Funcionamento do filtro complementar para estimativa de ângulo usando dados de acelerômetro e giroscópio provindos de uma IMU

A combinação desses dois conjuntos de dados através do filtro complementar resulta em estimativas de ângulo mais precisas e estáveis, preservando a fidelidade dos dados de movimento enquanto reduz efetivamente o ruído. Essa abordagem é fundamental para lidar com os desafios únicos dos dados baseados em IMUs em aplicações de movimento humano, como na análise de marcha (SEEL; RAISCH; SCHAUER, 2014; ZIHAJEHZADEH; PARK, 2016).

3.8.2 *Data Augmentation*

Para aprimorar a robustez e a capacidade de generalização dos modelos de aprendizado de máquina (TSINGANOS et al., 2020), foram implementadas diversas técnicas de aumento de dados (*data augmentation*) inspiradas em Tran e Choi (2020). Essas técnicas incluíram adição de ruído gaussiano, escalonamento, Deformação Temporal Arbitrária (*Arbitrary Time Deformation* - ATD) e Perturbação Estocástica de Magnitude (*Stochastic Magnitude Perturbation* - SMP).

3.8.2.1 Adição de Ruído Gaussiano

A adição de ruído gaussiano é uma técnica simples, porém eficaz, que envolve a inserção de ruído aleatório com distribuição normal (*Gaussian*) aos dados originais. Esse método simula variações naturais e ruídos que podem ocorrer durante a coleta de dados em ambientes reais, ajudando o modelo a aprender a ignorar flutuações irrelevantes e a focar nas características essenciais do sinal. O ruído gaussiano é definido pela média e pelo desvio padrão, parâmetros que controlam a intensidade e a dispersão do ruído adicionado. A implementação dessa técnica pode ser expressa matematicamente como:

$$X_{\text{ruído}} = X + \mathcal{N}(0, \sigma^2) \tag{3.12}$$

onde X representa os dados originais e $\mathcal{N}(0, \sigma^2)$ é o ruído gaussiano com média 0 e variância σ^2.

3.8.2.2 Escalonamento

O escalonamento (*scaling*) envolve a alteração da amplitude dos dados de entrada, ajustando-os para uma faixa específica. Essa técnica é útil para simular diferentes intensidades de movimento ou variações na sensibilidade dos sensores. O escalonamento pode ser realizado multiplicando os dados por um fator de escala constante ou por um fator aleatório dentro de um intervalo pré-definido. Matemáticamente, o escalonamento é representado pela Equação 3.13:

$$X_{\text{escalonado}} = \alpha \cdot X \tag{3.13}$$

onde α é o fator de escala aplicado aos dados originais X.

3.8.2.3 Deformação Temporal Arbitrária (ATD)

A Deformação Temporal Arbitrária (*Arbitrary Time Deformation* - ATD) é uma técnica que altera a velocidade do sinal sem modificar sua amplitude. Essa metodologia é particularmente útil para simular variações nos diferentes ritmos de marcha, como passadas mais rápidas ou mais lentas, sem alterar a força ou a intensidade do movimento. A ATD é realizada aplicando uma transformação no eixo temporal dos dados, permitindo que o modelo aprenda a reconhecer padrões de movimento independentemente da velocidade. Matematicamente, a ATD pode ser expressa como:

$$X_{\text{ATD}}(t) = X\left(\frac{t}{\gamma}\right) \tag{3.14}$$

onde γ é o fator de deformação temporal que controla a velocidade da alteração.

3.8.2.4 Perturbação Estocástica de Magnitude (SMP)

A Perturbação Estocástica de Magnitude (*Stochastic Magnitude Perturbation* - SMP) envolve a aplicação de pequenas variações aleatórias na magnitude dos dados, preservando a estrutura temporal do sinal. Essa técnica ajuda o modelo a se tornar mais robusto a variações não sistemáticas na intensidade do movimento, como flutuações naturais na força aplicada durante a marcha. A SMP é implementada adicionando um fator de perturbação aleatório aos dados originais:

$$X_{\text{SMP}} = X \cdot (1 + \epsilon) \tag{3.15}$$

onde ϵ é uma pequena variável aleatória, tipicamente distribuída uniformemente dentro de um intervalo definido, como $[-0.05, 0.05]$, representando uma variação de até $\pm 5\%$ na magnitude dos dados.

3.8.2.5 Impacto das Técnicas de Aumento de Dados

A aplicação combinada dessas técnicas de aumento de dados permite uma expansão significativa do conjunto de dados, promovendo uma maior diversidade e representatividade das condições de marcha reais. Isso é fundamental para que os modelos de aprendizado de máquina desenvolvidos sejam capazes de generalizar melhor e lidar com variações não vistas durante o treinamento (TRAN; CHOI, 2020). A Figura 14 demonstra como cada técnica modifica os sinais de ângulo da coxa, ilustrando a eficácia das metodologias empregadas para aumentar a variabilidade dos dados sem comprometer a integridade das informações essenciais.

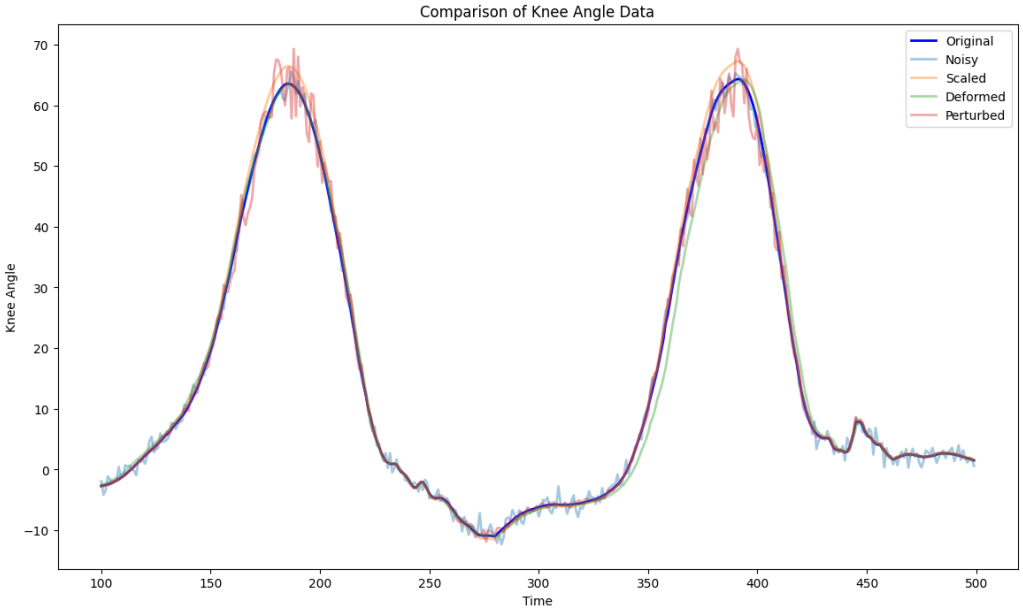


Figura 14 – Aplicação das técnicas de aumento de dados aos sinais coletados.

4 Materiais e Métodos

Nesse capítulo, busca-se fornecer uma visão detalhada das técnicas e recursos empregados, fundamentando as escolhas metodológicas que orientaram o desenvolvimento do sistema. Inicialmente, abordam-se os materiais utilizados, como os sensores inerciais, microcontroladores e demais componentes necessários para a aquisição e processamento de dados de movimento. Em seguida, detalha-se a metodologia adotada, incluindo o processo de coleta e tratamento dos dados, técnicas de *data augmentation* e a aplicação de algoritmos de aprendizado profundo. A seguir, descrevem-se as etapas de implementação e treinamento dos modelos, juntamente com as estratégias de otimização multiobjetivo para garantir precisão e eficiência computacional em sistemas embarcados.

4.1 Arquitetura da aquisição dos dados

Para implementar o filtro complementar, discutido na Seção 3.8.1 e processar os dados das IMUs, utilizou-se um microcontrolador ESP32 operando sob o ambiente FreeRTOS. Dois sensores IMU MPU6050 estão conectados ao ESP32 via interface I2C. O ESP32 desempenha múltiplas funções: inicializa os sensores, realiza a aquisição e processamento dos dados, aplica o filtro complementar e disponibiliza os dados processados para clientes externos. A Figura 15 ilustra todo processo de aquisição e envio dos dados.

4.2 Coleta de Dados dos Sensores

Utilizando a capacidade *dual-core* do ESP32 sob o FreeRTOS, a coleta e o processamento dos dados são executados de forma simultânea em núcleos separados. A aquisição dos dados é realizada por meio de temporizadores (*timers*), que ativam sistematicamente as funções de leitura de dados em intervalos específicos. Essa metodologia assegura a coleta precisa e sincronizada dos dados angulares dos sensores (FRANKLIN; MOHANA, 2020), que são então armazenados para análise.

O Core 1 lida com o sensor MPU6050 localizado na coxa, processando os dados brutos, calculando os ângulos e aplicando o filtro complementar, conforme detalhado na Seção 3.8.1. Após o cálculo do ângulo, os dados são enviados para uma fila (*queue*) que aguarda requisições externas. De forma similar, o Core 2 processa os dados do sensor localizado na panturrilha. A Figura 16 apresenta o diagrama de temporização do sistema de aquisição de dados.

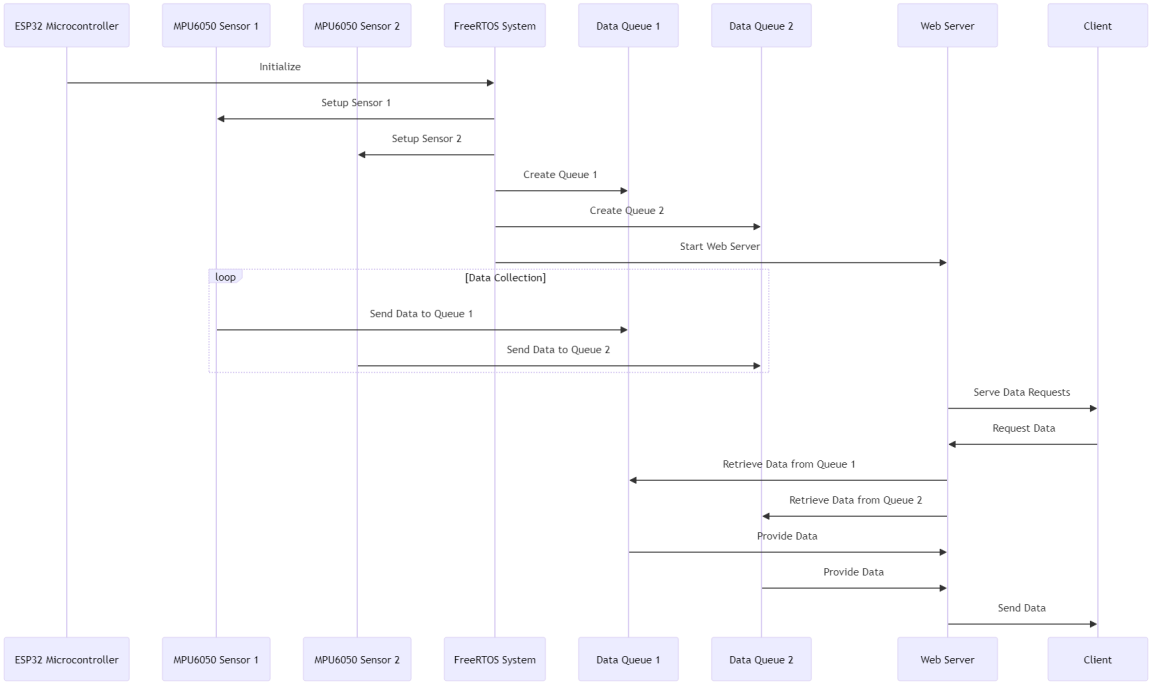


Figura 15 – Diagrama do sistema de aquisição de dados ilustrando a conexão entre o microcontrolador ESP32 e os sensores IMU MPU6050. O sistema processa dados em tempo real e os transmite para um cliente externo via Wi-Fi.

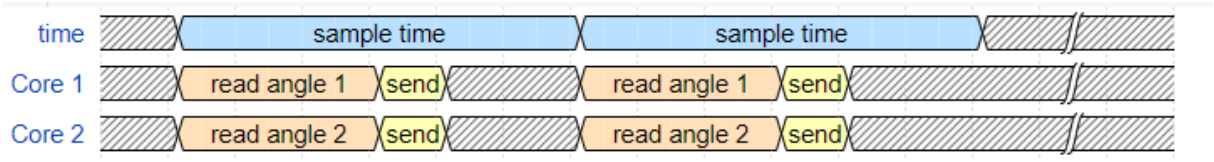


Figura 16 – Diagrama de temporização do sistema de aquisição de dados.

4.3 Mecanismo de Transmissão de Dados

A transmissão dos dados é gerenciada pelo servidor web executado no ESP32, implementado em C++ utilizando o *framework* Arduino e operando dentro do ambiente FreeRTOS. Quando um cliente externo envia uma requisição HTTP GET para o *endpoint* `/sensors`, o servidor responde fornecendo os dados mais recentes dos sensores IMU.

O servidor web interage com filas que armazenam os dados angulares processados de cada sensor. Essas filas são alimentadas pelas tarefas que executam nos respectivos núcleos do ESP32. Ao receber uma solicitação, o servidor recupera as amostras necessárias dessas filas, compila os dados em um formato estruturado e os transmite ao cliente.

4.4 Posicionamento dos Sensores e Manipulação dos Dados

Os sensores IMU foram estrategicamente posicionados na coxa e perna para capturar com precisão os movimentos cruciais para a análise de marcha, seguindo as melhores práticas

de posicionamento de sensores (Pacini Panebianco et al., 2018), conforme ilustrado na Figura 17.

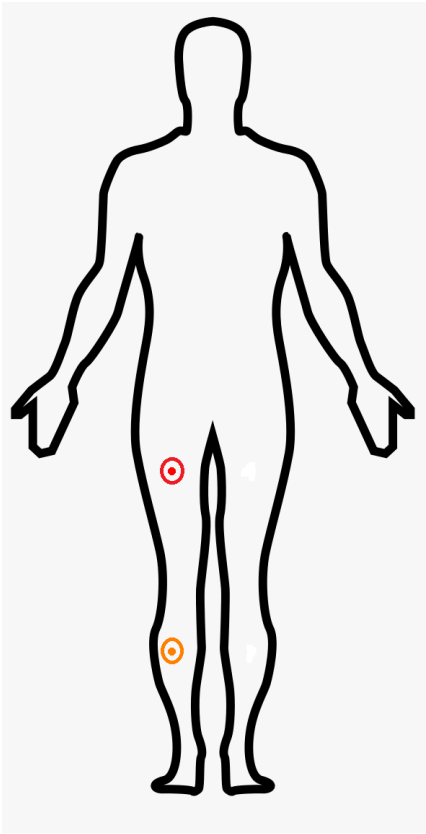


Figura 17 – Posicionamento dos sensores IMU (MPU6050) na coxa e perna do participante.

4.5 Aquisição de Dados

A coleta de dados foi realizada com quatro participantes, permitindo um controle rigoroso e consistência nos padrões de marcha observados. Entre os participantes, estavam dois homens e duas mulheres, garantindo uma diversidade básica de gênero e idade. Todos os participantes eram adultos saudáveis, sem histórico de condições musculoesqueléticas ou neurológicas que pudessem influenciar a marcha. A Tabela 2 apresenta um resumo das características demográficas dos participantes.

Gênero	**Idade (anos)**	**Altura (m)**
Masculino	21,5 ± 3,5	1,75 ± 0,05
Feminino	26,5 ± 3,5	1,63 ± 0,03

Tabela 2 – Características demográficas dos participantes do estudo (média ± desvio padrão)

Os participantes foram instruídos a realizar caminhadas com diferentes comprimentos de passada para simular diversos cenários de marcha, capturando uma ampla gama de

características representativas das condições práticas de caminhada. Os comprimentos de passada utilizados foram:

- Passadas curtas, medindo aproximadamente 0,6 metros.
- Passadas naturais, equivalentes ao comprimento médio da passada dos participantes, de 0,82 metros.
- Passadas longas, de 1 metro.

Para manter a consistência, todas as caminhadas foram realizadas em uma superfície plana e uniforme, com os participantes utilizando calçados similares para minimizar variáveis externas que pudessem afetar os padrões de marcha. Os participantes tiveram tempo suficiente para se familiarizar com os comprimentos de passada especificados antes do início da coleta de dados. A Figura 18 ilustra a variação do ângulo do joelho ao longo do ciclo da marcha para diferentes comprimentos de passada de um mesmo participante.

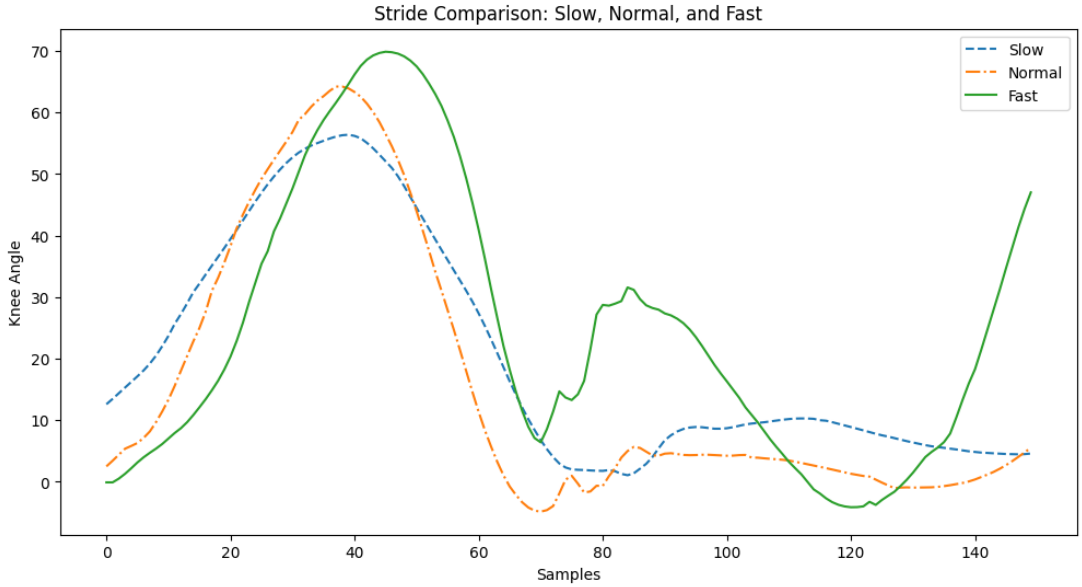


Figura 18 – Variação do ângulo do joelho durante o ciclo de marcha para passadas curtas, naturais e longas de um mesmo participante.

4.6 Processamento de Dados

Para modelar a relação entre o movimento da coxa e o comportamento do joelho, o ângulo da coxa foi utilizado como variável independente X e o ângulo do joelho como variável dependente Y. O ângulo do joelho Y foi calculado subtraindo-se o ângulo da perna do ângulo da coxa em cada ponto do ciclo da marcha:

$$Y = \text{Ângulo da Coxa} - \text{Ângulo da Perna}$$

Os dados processados foram então utilizados para treinar o modelo, visando compreender como as variações no movimento da coxa afetam a articulação do joelho durante diferentes comprimentos de passada. A Figura 19 ilustra a relação entre o ângulo da coxa e o ângulo do joelho calculado.

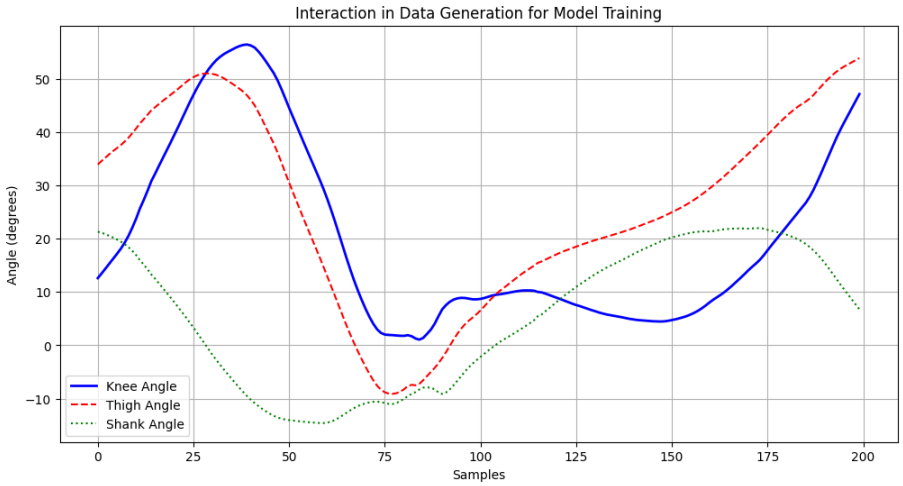


Figura 19 – Relação entre o ângulo da coxa (X) e o ângulo do joelho (Y) para diferentes comprimentos de passada.

4.7 Entrada, Saída e Parâmetros do Modelo

Para estabelecer uma relação preditiva entre o ângulo da coxa (entrada) e o ângulo do joelho (saída), é necessário definir de que forma os dados serão fornecidos ao modelo e quais serão as quantidades a serem estimadas. Nesse contexto, duas principais escolhas devem ser feitas: o tamanho da janela de entrada (quantidade de amostras históricas do ângulo da coxa que serão utilizadas para a previsão) e o horizonte de predição (quantas amostras futuras do ângulo do joelho se deseja prever).

O modelo recebe como entrada uma sequência temporal do ângulo da coxa, correspondendo a um determinado número de amostras consecutivas – a chamada "janela de entrada". Essa janela pode variar em tamanho, por exemplo, 16, 32 ou 64 amostras. A escolha desse parâmetro influencia a quantidade de informação histórica que o modelo tem para inferir o comportamento subsequente. Uma janela de entrada maior fornece mais contexto sobre o passado, enquanto uma menor foca em um trecho mais curto, possivelmente reduzindo o esforço computacional, mas potencialmente capturando menos padrões.

A saída do modelo é o ângulo do joelho previsto ao longo de um determinado "horizonte de predição", que também é um parâmetro ajustável, por exemplo, 5, 10 ou 15 amostras futuras. Um horizonte mais longo oferece uma visão mais ampla do comportamento futuro, mas aumenta a incerteza, já que a previsão é feita sobre um

período maior, sujeita a mais variáveis não controladas.uji8

Esses dois parâmetros – tamanho da janela de entrada e horizonte de predição – podem ser ajustados de acordo com os objetivos da análise e as características dos dados. Na Figura 20, ilustra-se a variação do tamanho da janela de entrada mantendo um horizonte fixo de previsão (por exemplo, 10 amostras). Nesse gráfico, aproximadamente um ciclo de marcha do ângulo da coxa e do joelho é representado e as regiões destacadas em vermelho mostram a janela de entrada em diferentes dimensões (16, 32, 64 amostras). Ao lado disso, as regiões em azul marcam o horizonte de predição fixo, permitindo visualizar como o modelo "enxerga" mais ou menos informação histórica (vermelho) para gerar a saída futura (azul).

Já a Figura 21 mostra o cenário oposto: mantendo fixo o tamanho da janela de entrada (por exemplo, 64 amostras, novamente destacadas em vermelho), variam-se diferentes horizontes de predição (5, 10 e 15 amostras, evidenciados em azul). O ciclo de marcha da coxa e joelho permanece ao fundo, fornecendo uma referência do comportamento geral. A comparação entre as diferentes extensões da área azul permite entender o impacto de se tentar prever mais ou menos passos futuros com a mesma informação histórica.

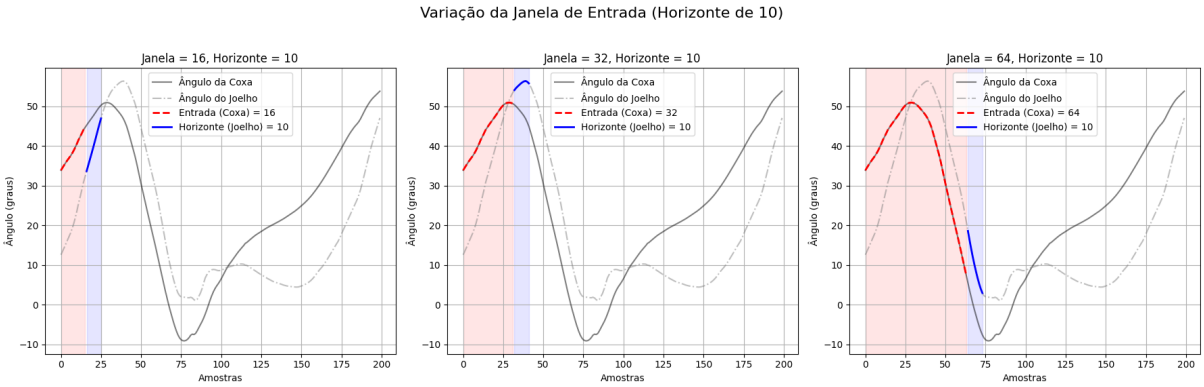


Figura 20 – Exemplo de variação do tamanho da janela de entrada (16, 32, 64) mantendo o horizonte de predição fixo em 10.

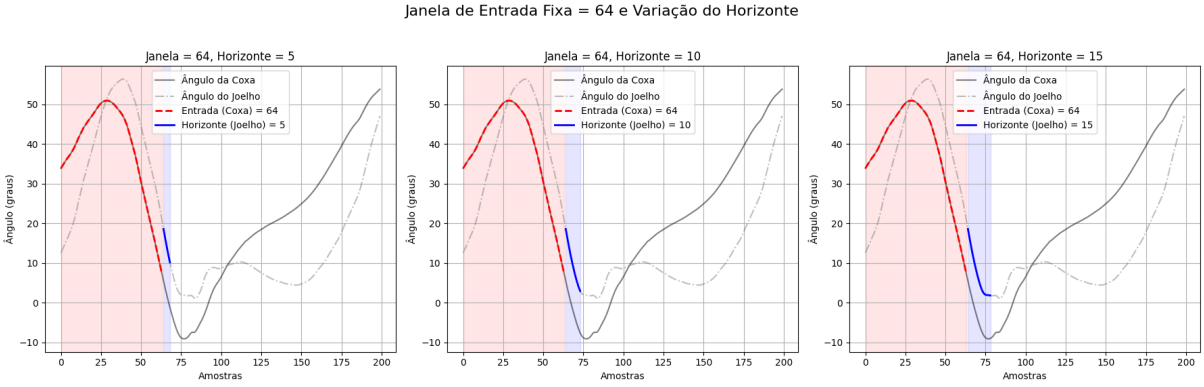


Figura 21 – Exemplo de variação do horizonte de predição (5, 10, 15) mantendo o tamanho da janela de entrada fixo em 64.

4.8 Pré-processamento e Aumento de Dados

Para abordar os desafios relacionados à quantidade limitada de dados de treinamento, estratégias de *data augmentation* foram adotadas, conforme descrito na Seção 3.8.2, e ilustrado na Figura 14. Empregando essa metodologia, o conjunto de dados foi ampliado de 33.800 para 169.000 pontos de dados. Esses dados foram processados para treinamento dos modelos conforme descrito na Seção 4.7. Essas amostras foram distribuídas em conjuntos de treinamento, validação e teste nas proporções de 70%, 20% e 10%, respectivamente.

4.9 Otimização Multiobjetivo

Controlando a arquitetura do modelo pelos parâmetros α e β (conforme discutido na Seção 3.5), foram treinados modelos com diversas janelas de entrada e horizontes de predição para identificar a solução ótima em termos de precisão e complexidade. Para isso, os métodos das Seções 3.7.1 e 3.7.2 foram combinados, utilizando os parâmetros dos modelos para encontrar a configuração que otimizasse simultaneamente múltiplos critérios: capacidade de generalização, minimização do sobreajuste, eficiência computacional e precisão preditiva.

O primeiro passo é calcular a Fronteira de Pareto, que revela um conjunto de modelos não dominados, cada um oferecendo diferentes equilíbrios entre as métricas avaliadas, especificamente o **RMSE (Erro Quadrático Médio)**, definido na Equação 3.4, que avalia a precisão preditiva (HYNDMAN; ATHANASOPOULOS, 2018), e o **tamanho do modelo em bytes**, refletindo a complexidade e eficiência computacional essenciais para ambientes restritos (ELSKEN; METZEN; HUTTER, 2019).

Para selecionar um modelo específico da Fronteira de Pareto, aplicou-se o método de Métricas Ponderadas, atribuindo pesos a cada métrica conforme sua importância relativa para os objetivos do trabalho (Tabela 3). Este método permite uma avaliação quantitativa das soluções de Pareto, facilitando a escolha do modelo que melhor atende às prioridades estabelecidas (ELBASHEER; KERMANI; RASMUSSEN, 2020).

Métrica	**Peso**
RMSE (Erro Quadrático Médio)	0,3
Tamanho do Modelo	0,3
Horizonte de Predição	0,3
Tamanho da Janela de Entrada	0,1

Tabela 3 – Pesos atribuídos a cada métrica no método de Métricas Ponderadas para seleção de modelos.

Foi dada maior importância ao RMSE e tamanho do modelo, usados na Fronteira

de Pareto, e ao horizonte de predição, indicando o número de passos futuros previstos e influenciando a utilidade prática em aplicações como próteses ativas (FARINA et al., 2014). O tamanho da janela de entrada, embora com peso menor, permanece significativo por determinar a quantidade de dados históricos utilizados e impactar a capacidade do modelo de capturar padrões temporais (GOODFELLOW; BENGIO; COURVILLE, 2016). Os pesos refletem considerações empíricas e arquiteturais (HEIM et al., 2021).

A consideração dos tamanhos da janela de entrada e do horizonte de predição é crucial para capturar a dinâmica temporal dos dados e atender aos requisitos específicos da aplicação (HEIM et al., 2021). Janelas de entrada maiores incorporam mais informações passadas, potencialmente melhorando a precisão em séries temporais com padrões de longo prazo (HYNDMAN; ATHANASOPOULOS, 2018), mas aumentam a complexidade computacional e o risco de sobreajuste (GOODFELLOW; BENGIO; COURVILLE, 2016). Janelas menores reduzem demandas computacionais, mas podem não capturar padrões essenciais. De forma similar, horizontes de predição mais longos, embora desafiadores devido à maior incerteza e acúmulo de erros (HYNDMAN; ATHANASOPOULOS, 2018), são críticos em aplicações como próteses ativas, onde é necessário prever intenções do usuário ao longo de intervalos suficientes para movimentos suaves e coordenados (FARINA et al., 2014). Previsões de curto prazo podem não fornecer informações adequadas para ajustes proativos do dispositivo, levando a respostas atrasadas ou abruptas.

As arquiteturas utilizadas foram MLP (Seção 3.1), CNN1D e CNN2D (Seção 3.2) e LSTM (Seção 3.3). No processo de otimização, os hiperparâmetros α e β foram variados de acordo com os intervalos apresentados na Tabela 4. Os valores de α eram potências de 2^n, variando de 16 a 256, enquanto β variou de 2 a 6. Esses intervalos foram selecionados para equilibrar a complexidade computacional e a eficiência de memória, com base na análise de alinhamento de memória e técnicas de otimização da literatura (HEIM et al., 2021). A escolha de potências de 2 para α aproveita o acesso eficiente à memória, enquanto o intervalo de β garante uma granularidade fina na sintonia do modelo sem aumentar excessivamente o número de operações.

Parâmetro	**Intervalo**
α	16, 32, 64, 128, 256
β	2, 3, 4, 5, 6
Horizonte de Predição	1, 3, 5, 7, 10
Tamanho da Janela de Entrada	16, 32, 64, 128, 256

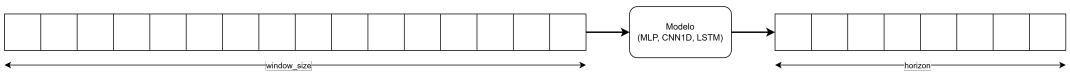
Tabela 4 – Intervalos de Busca dos Hiperparâmetros

4.10 Metodologia de Treinamento

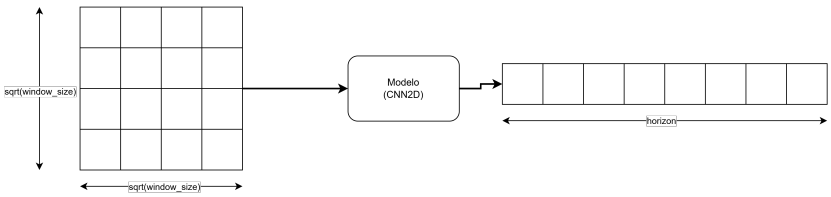
Uma variedade de modelos foi explorada, todos adaptados para utilização em sistemas embarcados com o *framework* TensorFlow Lite para Microcontroladores. Cada modelo foi treinado por um máximo de 1000 *epochs*, utilizando *Early Stopping* com paciência de 20 *epochs* para evitar sobreajuste (*overfitting*). Uma taxa de aprendizado adaptativa foi empregada, iniciando com um valor de 0,001 e utilizando decaimento.

Além disso, os dados foram estruturados de forma específica para cada tipo de modelo. Para as redes **MLP**, **LSTM** e **CNN1D**, os dados de entrada foram organizados no formato de janela temporal (*window_size*), onde cada amostra consistia em uma sequência de valores com comprimento definido por *window_size*, conforme ilustrado na Figura 22a.

Por outro lado, para a arquitetura **CNN2D**, os dados foram remodelados (*reshape*) para um formato bidimensional de $(\sqrt{\text{window\_size}}, \sqrt{\text{window\_size}})$, conforme ilustrado na Figura 22b. Essa transformação possibilitou que a CNN2D explorasse padrões espaciais nos dados, aproveitando as capacidades das camadas convolucionais bidimensionais para extrair características locais complexas a partir das representações matriciais dos dados de marcha.



(a) Estruturação dos dados para MLP, LSTM e CNN1D



(b) Estruturação dos dados para CNN2D

Figura 22 – Estruturação dos dados para diferentes arquiteturas de redes neurais e previsão do horizonte

4.11 Arquitetura Combinada e Adaptação Dinâmica da Marcha

Inicialmente, foi utilizado um modelo combinado para prever os ângulos articulares do joelho a partir de dados de movimento da coxa que englobavam todos os tipos de marcha: passos curtos, naturais e longos. Neste cenário, as sequências de movimento eram alimentadas em um único modelo treinado com todos os dados, projetado para capturar dependências temporais e características espaciais relevantes para a análise da marcha.

No entanto, essa abordagem apresentou limitações, detalhadas na Seção 5.1, pois um único modelo enfrenta dificuldades para capturar as particularidades de cada padrão de marcha, resultando em generalizações inadequadas e redução na precisão preditiva (MARCO et al., 2019; WAN; WANG; PHOHA, 2018).

Para superar essas limitações, foi adotada uma estratégia de adaptação dinâmica da marcha que envolve a classificação em tempo real do tipo de marcha e a seleção de modelos especializados para cada padrão específico—passos curtos, naturais ou longos. Essa abordagem permite que cada modelo se concentre nas características únicas de seu tipo de marcha, aprimorando a precisão preditiva e a eficiência (HAN et al., 2021; WAN; WANG; PHOHA, 2018; MARCO et al., 2019). Além de melhorar a precisão, essa estratégia facilita a escalabilidade, permitindo a inclusão de novos tipos de marcha com o treinamento de modelos adicionais e ajustes no classificador (HAN et al., 2021), e mantém a eficiência computacional ao evitar a criação de um modelo único grande e complexo, mantendo o sistema gerenciável (MARCO et al., 2019).

A Figura 23 ilustra a arquitetura do sistema na abordagem dinâmica. As sequências de movimento da coxa são inicialmente processadas por um módulo de classificação da marcha, que seleciona dinamicamente o modelo especializado apropriado para a predição dos ângulos do joelho com base no tipo de marcha identificado.

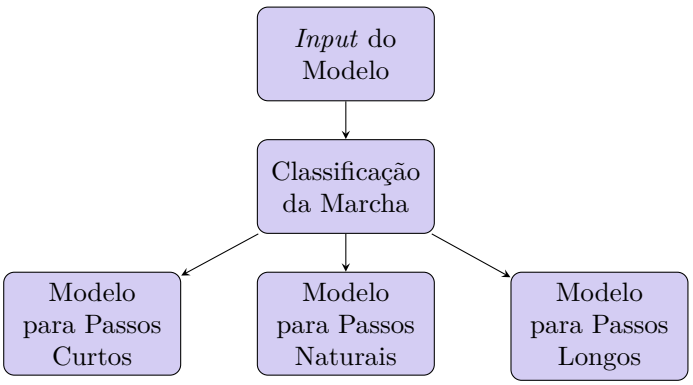


Figura 23 – Arquitetura do sistema ilustrando a adaptação dinâmica da marcha com classificação e modelos especializados.

4.12 Treinamento do Classificador e Geração de Rótulos

Para viabilizar a adaptação dinâmica da marcha, foi desenvolvido um módulo de classificação que recebe como entrada as sequências de movimento da coxa e atribui um rótulo de marcha (passos curtos, naturais ou longos) a cada intervalo temporal analisado. A construção dos rótulos para o treinamento se deu por meio da segmentação dos dados em janelas de amostras do mesmo tamanho dos modelos especialistas, onde cada janela foi classificada de acordo com o tipo de marcha dominante naquele intervalo.

Concretamente, cada amostra no conjunto de dados possuía uma anotação de marcha correspondente (ex., “Marcha Curta”, “Marcha Normal”, “Marcha Longa”). Em seguida, para cada janela de tamanho pré-definido, determinou-se o rótulo a partir da moda dessas anotações. A Figura 24 exemplifica o processo, mostrando em cores distintas o mesmo sinal de ângulo da coxa associado a diferentes rótulos, onde a classe associada foi a de Marcha Normal por ser a classe dominante da amostra.

No estágio de treinamento, as janelas rotuladas foram alimentadas em um classificador baseado em redes neurais, capaz de identificar o tipo de marcha em tempo real. Depois de treinado, o modelo classificador tornou-se responsável por analisar continuamente as leituras da coxa e decidir qual tipo de marcha (passos curtos, naturais ou longos) estava em execução. Esse rótulo, por sua vez, servia como chave para acionar o modelo regressivo mais adequado, conforme descrito na Seção 4.11.

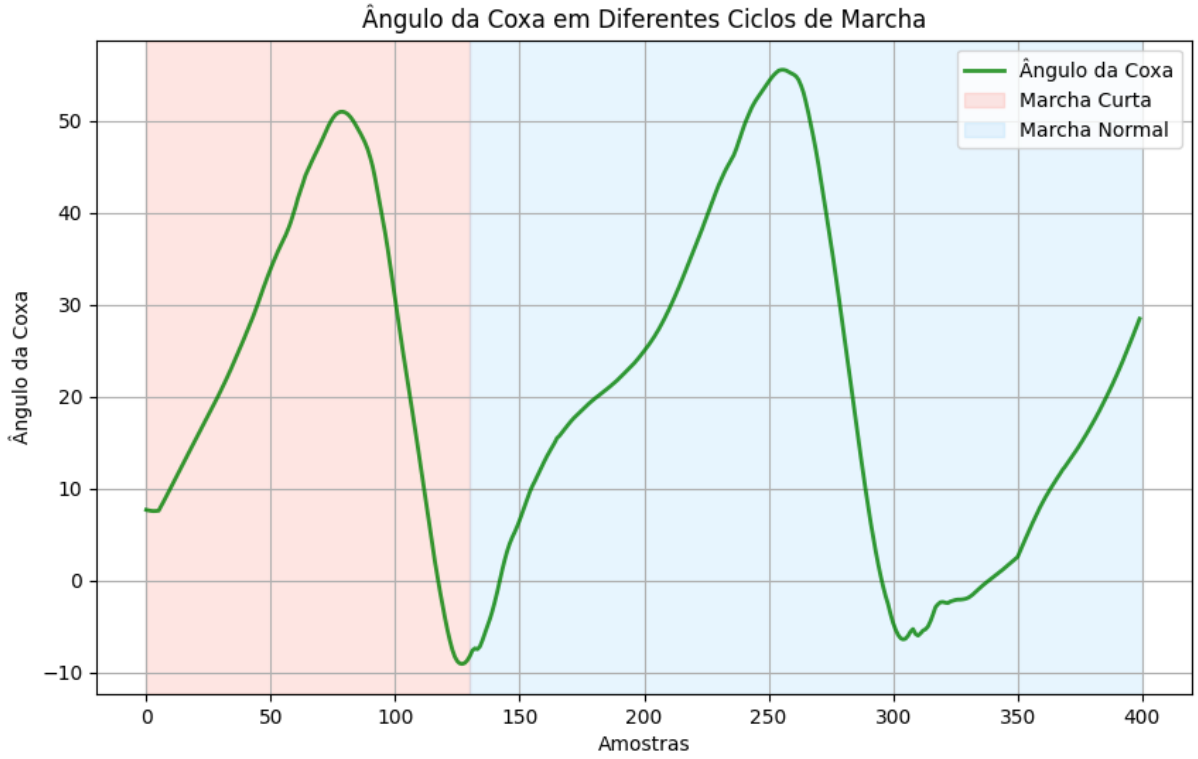


Figura 24 – Exemplo de sinal de ângulo da coxa (*verde*) segmentado em regiões classificadas como “Marcha Curta” (*rosa*) e “Marcha Normal” (*azul*).

4.13 *Hardware* Embarcado

Com base nos resultados de estudos anteriores (KARAKISH; FOUZ; ELSAWAF, 2022), que demonstraram os benefícios significativos das CNNs para a predição de trajetórias de marcha em microcontroladores embarcados, optou-se por utilizar a placa de desenvolvimento Sipeed Maix Bit, que incorpora o sistema Kendryte K210.

O K210 é um processador RISC-V de 64 bits *dual-core* que integra uma Unidade de Processamento Neural (KPU) de alto desempenho, projetada especificamente para executar CNNs de forma eficiente (KENDRYTE, 2018). A funcionalidade *dual-core* do K210 permite distribuir tarefas entre os dois núcleos. Na aplicação, o classificador opera simultaneamente no segundo núcleo, eliminando sobrecarga adicional e mantendo o sistema compacto e eficiente — essencial para aplicações em tempo real.

Além disso, o baixo custo do K210 e seu consumo energético reduzido (tipicamente menos de 1 W) tornam-no uma solução acessível e prática para aplicações embarcadas, especialmente em contextos onde os recursos são limitados. Conforme ilustrado na Tabela 5, outras plataformas como o NVIDIA Jetson Nano e o Jetson Xavier NX são aproximadamente 6 e 15 vezes mais caras, respectivamente, além de apresentarem consumos de energia significativamente maiores, entre 5–15 W. O Raspberry Pi 4B combinado com o Intel NCS2 também é cerca de 3 vezes mais caro que o K210 para aplicações específicas, mantendo um desempenho similar mas com maior custo e consumo energético.

Dessa forma, o Kendryte K210 se destaca por oferecer um excelente equilíbrio entre desempenho e eficiência energética a um custo muito mais baixo. Essa relação custo-benefício é particularmente relevante para o desenvolvimento de próteses inteligentes em larga escala, onde a redução de custos é fundamental para a viabilidade econômica e a disseminação da tecnologia em regiões com recursos limitados.

Plataforma	**Consumo de Energia (W)**	**Desempenho**
NVIDIA Jetson Nano	5–10 W	0,5 TFLOPs
NVIDIA Jetson Xavier NX	10–15 W	1,3 TFLOPs
Rasp. Pi 4B + Intel NCS2	3–6,25 W (Pi) + 1 W (NCS2)	1 TFLOPs
Kendryte K210 (Maix Bit)	<1 W (típico)	0,8 TFLOPS

Tabela 5 – Comparação de Consumo de Energia e Desempenho das Plataformas

5 Resultados e Discussão

Neste capítulo, serão apresentados os resultados obtidos a partir das análises e experimentos realizados, seguindo a metodologia descrita no capítulo anterior. Os resultados são discutidos em detalhes, comparando-os com estudos anteriores e avaliando o desempenho dos modelos propostos em termos de precisão, eficiência computacional e adequação para implementação em sistemas embarcados.

5.1 Modelo Combinado de Marcha

A Figura 25 apresenta as configurações de modelo que compõem a Fronteira de Pareto, marcadas com cruzes vermelhas (×). Essas configurações representam os melhores compromissos entre RMSE e tamanho do modelo, sem comprometer significativamente um em benefício do outro, conforme detalhado na Seção 3.7.1. O modelo selecionado pelo Método de Métricas Ponderadas, destacado com um círculo verde (○), indica um equilíbrio preferencial das métricas definidas sob os critérios de avaliação ponderada descritos na Seção 3.7.2. Esses resultados referem-se aos modelos que englobam todos os tipos de marcha analisados neste estudo.

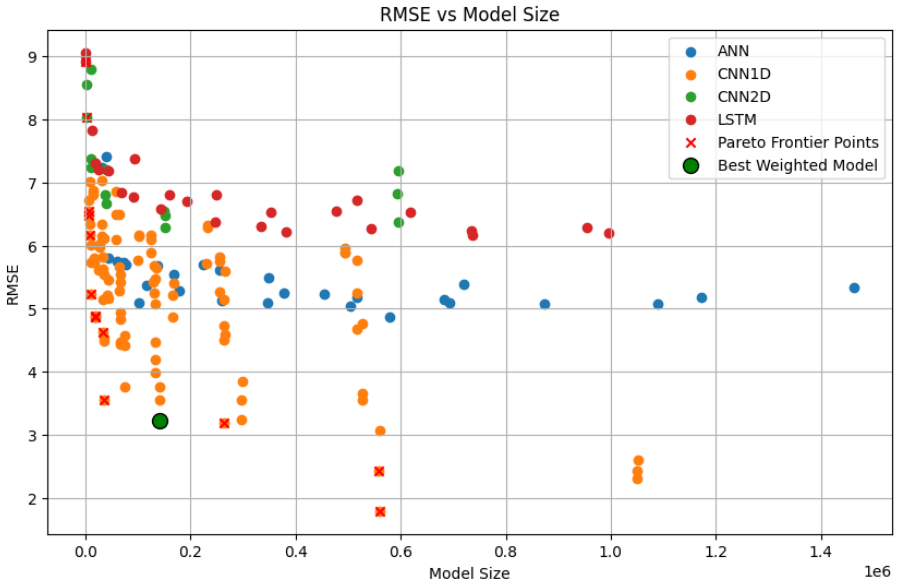


Figura 25 – Gráfico de Dispersão com Fronteira de Pareto e Método Ponderado

Os modelos listados na Tabela 6 representam aqueles na fronteira de Pareto. Para identificar o modelo ideal entre esses, foi aplicado o método ponderado a cada modelo, conforme discutido na Seção 4.9. A coluna **Score** quantifica o desempenho geral de cada modelo, com pontuações mais altas indicando um equilíbrio mais favorável de acordo com

os critérios, conforme definido na Seção 3.7.2 e com os pesos definidos na Tabela 3. A linha destacada corresponde ao modelo com a maior pontuação.

Alpha	RMSE	Tamanho do Modelo	Modelo	Tamanho da Janela	Horizonte	Beta	Score
16	7,160	8.644	CNN1D	16	1	2	0,418
16	5,630	33.220	CNN1D	64	1	2	0,509
16	7,547	6.660	CNN1D	16	1	3	0,402
16	7,484	6.796	CNN1D	16	3	3	0,555
16	6,238	10.756	CNN1D	32	1	3	0,469
16	4,894	18.948	CNN1D	64	1	3	0,506
16	4,869	19.084	CNN1D	64	3	3	0,657
16	3,166	19.220	CNN1D	64	10	4	0,807
16	3,105	35.332	CNN1D	128	1	4	0,607
32	3,063	263.044	CNN1D	128	5	2	0,500
32	3,021	140.292	CNN1D	128	3	4	0,564
64	2,424	559.108	CNN1D	128	5	4	0,374
64	1,894	560.148	CNN1D	128	10	4	0,700

Tabela 6 – Parâmetros do modelo e métricas de desempenho, incluindo a coluna Score

Com base nos resultados da Tabela 6, notou-se que as redes LSTM, tradicionalmente favorecidas para esses tipos de problemas, podem não ser a escolha mais adequada. Isso se deve principalmente a desafios na implantação desses modelos em sistemas embarcados, conforme destacado em TensorFlow (2024). Além disso, estudos como os de Gholami, Napier e Menon (2020) e Karakish, Fouz e Elsawaf (2022) demonstraram que as CNNs podem igualar o desempenho das LSTMs em cenários não limitados ao *TinyML*, o que corrobora os resultados apresentados na tabela. A Tabela 7 fornece uma visão detalhada dos parâmetros ótimos para o modelo escolhido pelo Método Ponderado.

Parâmetro	**Valor**
Alpha (α)	16
Beta (β)	4
RMSE	3,166
Tamanho do Modelo	19.220
Modelo	CNN1D
Tamanho da Janela	64
Horizonte	10

Tabela 7 – Resumo dos Parâmetros do Modelo

O desempenho do modelo esteve alinhado com os resultados apresentados em Gholami, Napier e Menon (2020). No entanto, a hipótese era de que o sistema poderia alcançar um desempenho ainda melhor se a carga de aprendizado fosse reduzida, dividindo a tarefa em modelos especializados para cada tipo de marcha, conforme discutido na Seção 4.11. Este sistema é detalhado na Seção 5.2.

5.2 Classificação de Marcha e Modelos Especializados

Os resultados apresentados na Figura 25 serviram como base para o ajuste fino de α e β dos modelos de cada passada. As Tabelas 8, 9 e 10 apresentam as configurações ótimas para cada tipo de marcha após o processo de ajuste fino do modelo combinado.

Parâmetro	**Valor**
alpha (α)	16
beta (β)	4
RMSE	2,558
Tam. do Modelo	19.220
Modelo	CNN1D
Tam. da Janela	64
Horizonte	10

Tabela 8 – Resumo do Modelo de Marcha Curta

Parâmetro	**Valor**
alpha (α)	16
beta (β)	4
RMSE	1,486
Tam. do Modelo	19.220
Modelo	CNN1D
Tam. da Janela	64
Horizonte	10

Tabela 9 – Resumo do Modelo de Marcha Normal

Parâmetro	**Valor**
alpha (α)	8
beta (β)	5
RMSE	2,108
Tam. do Modelo	15.360
Modelo	CNN1D
Tam. da Janela	64
Horizonte	10

Tabela 10 – Resumo do Modelo de Marcha Longa

A Tabela 11 ilustra a arquitetura do modelo classificador utilizado para selecionar os modelos de regressão de marcha apropriados. O classificador foi implementado usando uma rede neural MLP. Os dados de entrada foram sequências construídas independentemente do tipo de marcha, garantindo que o modelo pudesse generalizar entre diferentes padrões de marcha. Para cada sequência de dados, o rótulo foi atribuído com base na moda dos tipos de marcha presentes nessa sequência, capturando efetivamente o tipo de marcha mais frequente. Foi alcançada uma acurácia de classificação de 91,83% no conjunto de teste.

Parâmetro	**Valor**
alpha (α)	16
beta (β)	4
Acurácia	0,9183
Tam. do Modelo	9.180
Modelo	MLP
Tam. da Janela	64

Tabela 11 – Modelo classificador utilizado para selecionar os modelos de regressão de marcha apropriados. O classificador foi projetado para identificar o tipo de marcha (curta, normal ou longa) e direcionar a entrada para o modelo especializado correspondente.

5.3 Resultados do Sistema Especializado

5.3.1 Análise das Melhorias nas Abordagens de Modelagem

A abordagem especializada demonstrou melhorias significativas em cada tipo de passada e no desempenho geral do modelo, conforme ilustrado na Figura 26. Esta figura

compara as curvas de perda de validação durante o treinamento entre o Modelo Combinado de Marcha (treinado em todos os tipos de passada) e os modelos especializados para cada tipo específico de passada. Especificamente, os modelos especializados apresentaram melhorias de 48,51% para passadas curtas, 66,65% para passadas naturais e 45,06% para passadas longas em comparação com o modelo combinado.

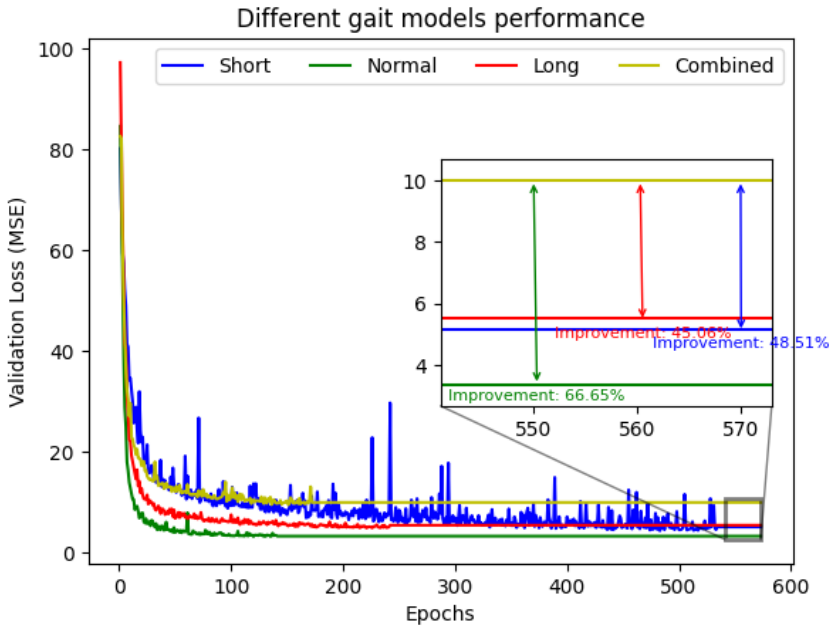


Figura 26 – Melhorias no desempenho dos modelos especializados em comparação com o modelo combinado

5.3.2 Desempenho do Modelo em Diferentes Tipos de Passada

Antes de implementar os modelos no sistema embarcado, realizou-se testes locais em uma máquina padrão para validar seu desempenho. Foi comparado o desempenho do modelo combinado e dos modelos especializados utilizando métricas de erro nos mesmos dados. A Tabela 12 apresenta os valores de Erro Quadrático Médio (MSE) para cada tipo de marcha analisado – curta, natural e longa.

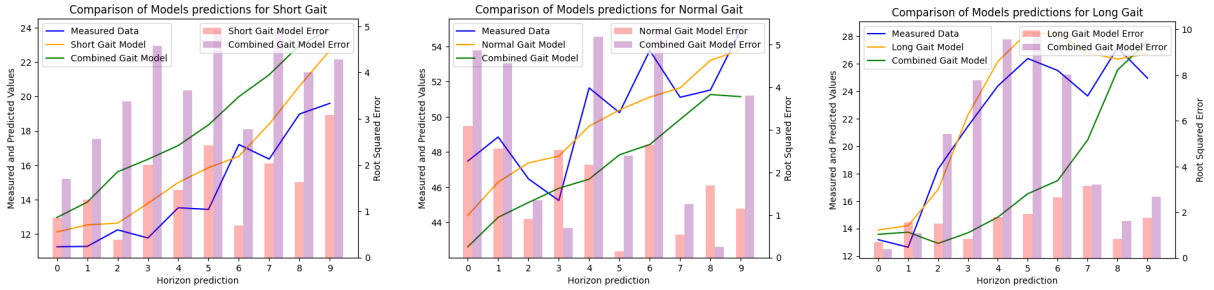
Tipo de Marcha	**MSE Modelo Combinado ($^{\circ 2}$)**	**MSE Modelo Especializado ($^{\circ 2}$)**	**Melhoria**
Curta	13,467	6,543	51,41%
Natural	6,407	2,208	65,53%
Longa	9,074	4,443	51,03%

Tabela 12 – MSE para cada tipo de marcha utilizando o modelo combinado e os modelos especializados

Os resultados indicam que a especialização dos modelos permite uma adaptação mais precisa aos diferentes padrões de marcha, reduzindo significativamente o erro de previsão. Essa melhoria é especialmente evidente para a marcha natural, onde o modelo

especializado obteve uma melhoria de 65,53% com relação ao modelo combinado. Isso sugere que, ao direcionar modelos específicos para cada tipo de passada, capturou-se nuances que um modelo combinado não consegue, aumentando assim a eficiência e a precisão geral do sistema.

A Figura 27 ilustra o desempenho de previsão e o RSE, definido na Equação 3.6, em cada passo dentro do horizonte de previsão para uma amostra de cada tipo de passada. Um eixo secundário exibe os valores do RSE. Os gráficos mostram claramente que os modelos especializados (linha amarela) consistentemente alcançam valores de RSE menores em comparação com o modelo combinado (linha verde), destacando as melhorias obtidas através da especialização. Além disso, os modelos especializados mantêm valores de RSE estáveis e menores em todos os passos de previsão quando comparados ao Modelo Combinado, indicando que as melhorias são uniformes ao longo de todo o horizonte, o que é essencial para a aplicação, como discutido na Seção 4.9.



(a) Previsões para Passada Curta (b) Previsões para Passada Natural (c) Previsões para Passada Longa

Figura 27 – Previsões e RSE para os tipos de passada curta, natural e longa usando o modelo combinado e os modelos especializados

Para ilustrar ainda mais o desempenho dos modelos em ciclos de passada, apresenta-se uma comparação detalhada para o tipo de passada natural. A Figura 28 mostra o ângulo medido do joelho (linha azul) ao lado das previsões tanto do modelo combinado (linha verde) quanto do modelo especializado (linha vermelha) ao longo de ciclos completos de passada. É evidente que o modelo especializado acompanha de perto o ângulo medido do joelho, enquanto o modelo combinado exibe discrepâncias, particularmente durante as fases de flexão e extensão rápida do joelho. Isso destaca a superior capacidade do modelo especializado em capturar as dinâmicas intrincadas da passada natural, reforçando a eficácia da especialização do modelo em melhorar a precisão preditiva.

Os resultados confirmam que os modelos especializados oferecem melhorias substanciais em relação ao modelo combinado para todos os tipos de marcha, em conformidade com os princípios de adaptação e eficiência discutidos em Han et al. (2021).

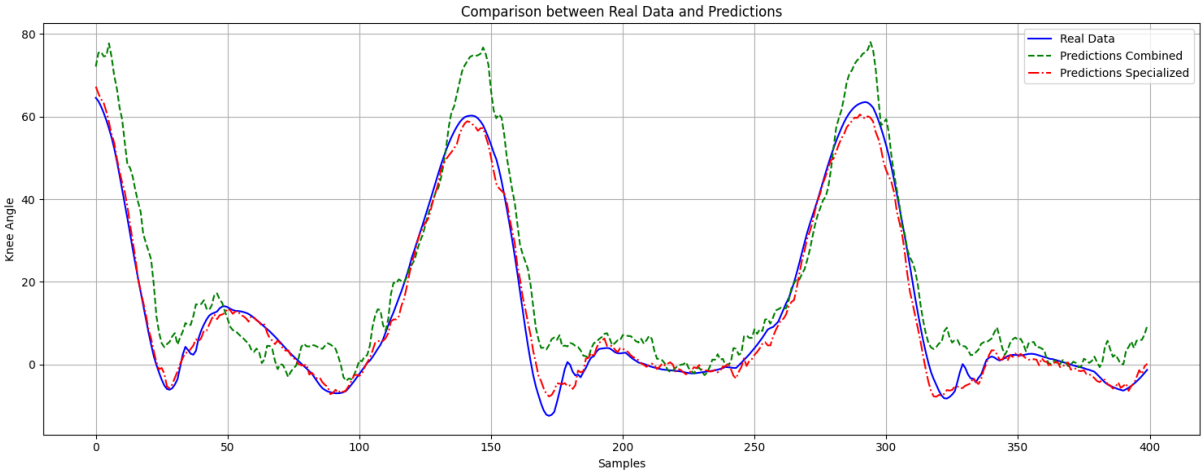


Figura 28 – Comparação das previsões dos modelos durante um ciclo de passada natural. A linha azul representa o ângulo real do joelho, a linha vermelha representa as previsões do modelo especializado, e a linha verde representa as previsões do modelo combinado.

5.3.3 Implementação no Sistema Embarcado

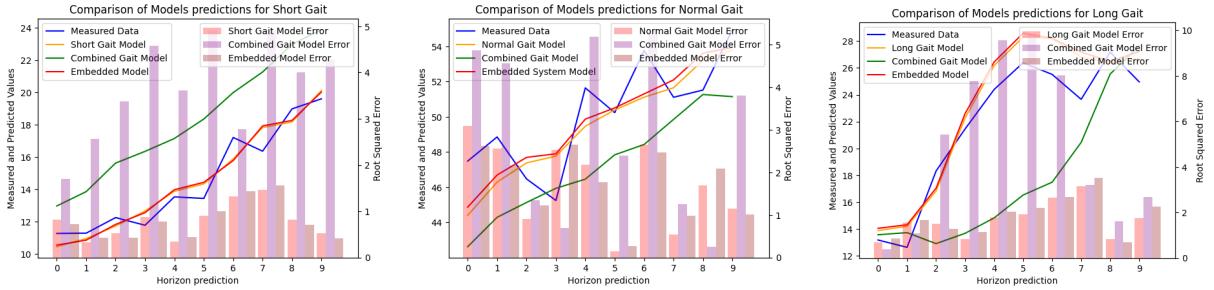
Os modelos especializados foram implementados no microcontrolador MaixBit (Apêndice D) e os resultados confirmam que as melhorias de desempenho observadas nos testes locais se mantiveram na implementação embarcada. A Tabela 13 apresenta os valores de MSE obtidos no sistema embarcado. Observa-se que os valores de MSE obtidos são muito próximos dos resultados registrados nos testes locais, indicando que a transição para o ambiente embarcado manteve a precisão dos modelos. Essa consistência sugere que o sistema proposto é robusto e capaz de preservar a qualidade das predições, mesmo com as limitações de processamento e memória do hardware utilizado.

Tipo de Marcha	**MSE Modelo Combinado** ($^{\circ 2}$)	**MSE Modelo Especializado** ($^{\circ 2}$)	**Melhoria (%)**
Curta	13,665	6,909	49,43%
Natural	6,634	2,552	61,53%
Longa	9,350	4,793	48,75%

Tabela 13 – MSE e melhoria percentual para cada tipo de marcha no sistema embarcado

A Figura 29 ilustra o desempenho de previsão e o RSE para cada passo dentro do horizonte de previsão no sistema embarcado. Os gráficos incluem as previsões feitas pelo sistema embarcado (linha vermelha), demonstrando que a implementação embarcada replica de forma bastante próxima o desempenho observado no cenário não embarcado (Figura 27).

A pequena variação entre os resultados dos testes locais e os da implementação embarcada destaca a eficiência da otimização multiobjetivo aplicada na fase de desenvolvimento, que permitiu a adaptação dos modelos às restrições de *hardware* sem comprometer a qualidade das previsões. Isso reforça o potencial da solução para aplicações em tempo



(a) Previsões de Passos Curtos (Sistema Embarcado)

(b) Previsões de Passos Naturais (Sistema Embarcado)

(c) Previsões de Passos Longos (Sistema Embarcado)

Figura 29 – Previsões e Erro Quadrático Relativo (RSE) para os tipos de passos curto, natural e longo no sistema embarcado

real em dispositivos de baixo custo, beneficiando usuários de próteses ativas com respostas precisas e confiáveis.

5.3.4 Desempenho Geral

A Tabela 14 resume o desempenho médio dos modelos considerando todos os tipos de marcha. Os modelos especializados resultaram na melhoria de todos os indicadores de desempenho, indicando que a adaptação dos modelos para padrões específicos de marcha aprimora o desempenho e as capacidades em tempo real.

Métrica	**Combinado**	**Especializado**
RMSE (°)	3,166	**2,050**
R^2	0,965	**0,980**
Tempo de Inferência (ms/previsão)	18,257	**16,845**+3,143(clf)
MSE $(°)^2$	10,023	**4,202**

Tabela 14 – Resumo das métricas de avaliação do modelo para os sistemas.

Ao comparar os modelos especializados com o modelo combinado, identificou-se melhorias significativas na precisão preditiva: o coeficiente de determinação R^2 aumentou de 0,965 para 0,980 e o RMSE reduziu de 3,166° para 2,050°. Esses resultados evidenciam um desempenho superior na predição do ângulo do joelho, refletindo a maior capacidade de cada modelo em se especializar no respectivo padrão de marcha.

Em relação ao tempo de inferência, os modelos especializados necessitam, em média, de 16,845 ms por previsão. Contudo, quando se incorpora o classificador para identificar o tipo de marcha no mesmo núcleo (sem utilizar processamento dedicado em segundo núcleo), adicionam-se 3,143 ms, totalizando 19,988 ms. Ainda assim, ao isolar apenas o tempo de inferência dos modelos regressivos (16,845 ms), observa-se uma melhora de aproximadamente 7,72% em comparação ao modelo combinado (18,257 ms), o que demonstra que a simplicidade de cada modelo especializado reduz o esforço computacional.

5.3.5 Comparação com Trabalhos Anteriores

A Tabela 15 apresenta uma comparação do método proposto com outros estudos recentes, listando a arquitetura dos modelos, o R^2, o RMSE e o tempo médio de inferência. Embora comparações diretas sejam difíceis devido às diferenças nos conjuntos de dados e nas configurações experimentais, os resultados obtidos são promissores e indicam que modelos especializados podem alcançar alta qualidade de previsão com menores demandas computacionais. A comparação posiciona este trabalho no contexto atual da pesquisa e destaca as vantagens da abordagem proposta, evidenciando a eficácia da especialização dos modelos em melhorar o desempenho e a eficiência.

Ref.	Abordagem	R^2	RMSE (°)	Tempo de Inferência (ms)
Su e Gutierrez-Farewik (2020)	LSTM	0,980	-	-
Gholami, Napier e Menon (2020)	CNN	0,980	3,405	-
Mundt et al. (2020)	LSTM	0,983	1,800	-
Huang et al. (2019b)	RNN	-	2,930	-
Saoud et al. (2023)	Transformer	0,996	1,794	-
Karakish, Fouz e Elsawaf (2022)	CNN	0,944	-	27,327
Este Trabalho (Combinado)	CNN	0,965	3,071	18,257
Este Trabalho (Especializado)	CNN	0,980	2,050	19,988

Tabela 15 – Comparação de métricas de desempenho para diferentes abordagens

Os modelos especializados demonstraram um tempo de inferência de 19,988 ms por previsão, adequando-se para aplicações em tempo real em sistemas embarcados. Em comparação, Karakish, Fouz e Elsawaf (2022) reportou um tempo de inferência de 27,327 ms utilizando uma abordagem baseada em CNN. Além disso, os modelos especializados apresentaram um RMSE de 2,050°, superando o valor de 2,930° obtido por um modelo baseado em RNN em Huang et al. (2019b), mesmo com um horizonte de predição maior de 10 em comparação a 1. Essa redução no RMSE indica uma precisão preditiva superior, essencial para a análise precisa da marcha e o controle eficaz de próteses, onde a previsão antecipada dos movimentos é crucial.

No contexto de outras pesquisas, arquiteturas como LSTMs e *Transformers* alcançaram alta precisão, porém com maior complexidade computacional. Por exemplo, Mundt et al. (2020) e Saoud et al. (2023) relataram valores de RMSE ligeiramente menores utilizando modelos LSTM e *Transformer*, respectivamente. No entanto, tais modelos podem ser considerados menos viáveis para sistemas embarcados devido às suas elevadas demandas computacionais.

De maneira geral, o método proposto neste trabalho demonstra que é possível obter resultados preditivos comparáveis a modelos mais complexos, como *Transformers* e LSTMs, utilizando modelos computacionalmente eficientes adaptados a tipos específicos de passada. Dadas as melhorias significativas observadas nos modelos CNN especializados, espera-se que aprimoramentos de desempenho similares possam ser alcançados em outros estudos,

especialmente aqueles que empregam arquiteturas semelhantes à proposta. Ao alinhar essas descobertas com as tendências observadas na literatura, contribuiu-se com *insights* valiosos para o desenvolvimento de modelos preditivos eficazes e eficientes adequados para implantação embarcada.

5.4 Principais Contribuições

O presente trabalho apresentou um sistema abrangente para o controle de próteses ativas de joelho, integrando aquisição de dados por IMUs, técnicas de *data augmentation*, arquiteturas de redes neurais otimizadas para sistemas embarcados e um classificador capaz de selecionar automaticamente modelos especializados para diferentes tipos de marcha. A seguir, destacam-se as principais contribuições alcançadas:

- **Aquisição e Processamento de Dados de Movimento**: Foram descritas soluçõess de *hardware* e *software* para aquisição confiável dos sinais de coxa e perna, incluindo o uso de um filtro complementar que combina informações de acelerômetro e giroscópio. Esse enfoque resultou em medições robustas, fundamentais para o correto funcionamento dos modelos de predição.

- **Otimização Multiobjetivo**: Foi desenvolvida uma metodologia que combina a Fronteira de Pareto e Métricas Ponderadas para conciliar a eficiência computacional (tamanho do modelo e tempo de inferência) com a precisão preditiva (RMSE, MSE e R^2). Este procedimento permitiu identificar configurações que otimizam simultaneamente o desempenho e a viabilidade de implementação em *hardware* de baixo custo.

- **Modelos Especializados para Marcha**: Em vez de treinar um único modelo para todos os padrões de marcha, foi adotada a estratégia de desenvolver arquiteturas de CNN especializadas em cada tipo de passada (curta, natural e longa). Como resultado, obteve-se redução substancial no erro de predição em comparação ao modelo combinado, reforçando a hipótese de que a segmentação por padrão de marcha melhora a exatidão e o desempenho de inferência.

- **Integração com Sistemas Embarcados de Baixo Custo**: A implantação dos modelos selecionados em um Kendryte K210 (*MaixBit*) validou a viabilidade de uso de dispositivos acessíveis e com recursos computacionais modestos para próteses de membros inferiores. O tempo de inferência para os modelos especializados manteve-se abaixo de 20 ms, evidenciando a capacidade para aplicações em tempo real.

- **Contribuição para Tecnologias Assistivas acessíveis**: Por fim, a combinação de *hardware* de baixo custo, sensores acessíveis e modelos de rede neural com alta

eficiência computacional oferece uma alternativa viável e escalável para próteses ativas, sobretudo em regiões com recursos limitados, contribuindo para o aumento da inclusão social e a melhora na qualidade de vida de indivíduos amputados.

Assim, as soluçõess propostas integram, de forma efetiva, avanços em aprendizado de máquina e sistemas embarcados, contribuindo para elevar o patamar das próteses ativas de membros inferiores no que diz respeito a precisão de predição, robustez e custo de implementação.

6 Conclusões e Trabalhos Futuros

Neste capítulo, apresentam-se as conclusões finais deste trabalho, destacando as contribuições realizadas e discutindo as limitações encontradas. Além disso, foram propostas direções para trabalhos futuros que podem ampliar e aprimorar os resultados aqui obtidos.

6.1 Conclusões

Este estudo abordou os desafios de implementar modelos preditivos eficientes e precisos para a estimativa do ângulo do joelho em próteses de membros inferiores utilizando sistemas embarcados. Ao usar IMUs de baixo custo e empregar técnicas de otimização multiobjetivo, foram desenvolvidos modelos de CNN otimizados para implantação em microcontroladores com recursos limitados e custo acessível.

Os resultados obtidos demonstram que modelos de CNN especializados, adaptados a tipos específicos de marcha—passadas curtas, naturais e longas—superam significativamente um modelo combinado treinado em todos os tipos de marcha. Os modelos especializados alcançaram um RMSE médio de $2,050°$, uma melhoria de mais de 48% em comparação com o modelo combinado, enquanto reduziram levemente o tempo de inferência em 7,72%. Esses resultados destacam a eficácia da especialização de modelos em capturar as nuances de diferentes padrões de marcha, levando a uma maior qualidade preditiva e eficiência computacional adequada para aplicações embarcadas em tempo real.

Além disso, a implementação desses modelos na placa de desenvolvimento Sipeed MaixBit, com o microcontrolador Kendryte K210, validou a viabilidade de implantar modelos avançados de aprendizado profundo em *hardware* embarcado acessível sem perda significativa de desempenho. A integração bem-sucedida de aceleradores de *hardware* e técnicas de otimização contribuiu para a redução dos tempos de inferência e utilização eficiente dos recursos.

Concluindo, esta pesquisa contribui para o campo das tecnologias assistivas vestíveis ao demonstrar que modelos eficientes e especializados de aprendizado profundo podem ser efetivamente implantados em sistemas embarcados para análise de marcha em tempo real. Os *insights* obtidos neste estudo têm o potencial de informar o desenvolvimento de dispositivos protéticos mais responsivos e adaptativos, melhorando, em última análise, a qualidade de vida de indivíduos com amputações de membros inferiores.

6.2 Trabalhos Futuros

Com base nas limitações identificadas e nos resultados obtidos, sugere-se as seguintes direções para trabalhos futuros:

- **Ampliação do Conjunto de Dados**: Realizar coletas de dados com um número maior e mais diversificado de participantes, incluindo indivíduos com diferentes características antropométricas e condições clínicas, para melhorar a generalização dos modelos.
- **Aquisição de Dados em Diferentes Terrenos**: Expandir a coleta de dados para superfícies e ambientes diversos (por exemplo, rampas, escadas, pisos irregulares e calçadas públicas), de modo a refletir melhor as condições cotidianas de uso das próteses e aumentar a robustez dos modelos.
- **Integração com Sistemas de Controle de Próteses**: Implementar os modelos em sistemas de controle de próteses reais, avaliando o desempenho em condições práticas e coletando feedback dos usuários.

6.3 Considerações Finais

Este trabalho contribuiu para o avanço no desenvolvimento de sistemas inteligentes aplicados a próteses de membros inferiores, demonstrando que é possível combinar técnicas de aprendizado de máquina e sistemas embarcados de baixo custo para alcançar predições precisas em tempo real. Os resultados obtidos indicam que a especialização de modelos e a adaptação dinâmica são estratégias promissoras para lidar com a variabilidade inerente aos padrões de movimento humano.

Espera-se que as direções propostas para trabalhos futuros incentivem a continuidade da pesquisa nesta área, levando ao desenvolvimento de próteses mais eficientes, adaptativas e acessíveis, que possam melhorar significativamente a qualidade de vida de indivíduos com amputações ou dificuldades de locomoção.

Em conclusão, além de fornecer um sistema eficiente para o controle de próteses ativas de joelho, este trabalho demonstra a viabilidade econômica da solução, que se baseia em sensores de baixo custo e hardware embarcado acessível. Essa abordagem reduz consideravelmente o custo de implementação em comparação com outros métodos que dependem de sensores EMG caros ou processadores de alto desempenho. A viabilidade econômica também amplia o potencial de aplicação do sistema para populações em regiões de baixa renda, permitindo que tecnologias avançadas de prótese estejam ao alcance de um público mais amplo. Com essa acessibilidade, o sistema proposto não só contribui para a autonomia e qualidade de vida de seus usuários, mas também tem o potencial de reduzir

custos de saúde no longo prazo ao minimizar a necessidade de reabilitações complexas e prolongadas. Portanto, a solução aqui apresentada não só preenche uma lacuna tecnológica, mas também se destaca como uma opção escalável e economicamente viável para o campo das tecnologias assistivas.

Referências

ABADADE, Y. et al. A comprehensive survey on tinyml. *IEEE Access*, v. 11, p. 96892–96922, 2023. Citado 2 vezes nas páginas 25 e 31.

ABDELJABER, O. et al. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, v. 388, p. 154–170, 2017. ISSN 0022-460X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0022460X16306204>. Citado na página 37.

ABDOLLAHPOURI, H.; BURKE, R.; MOBASHER, B. Managing popularity bias in recommender systems with personalized re-ranking. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. [S.l.]: ACM, 2019. p. (to be filled). Citado na página 45.

ADAMS, V. H. *Complementary Filters*. 2024. Lecturer of Electrical Engineering, Cornell University. Email: `vha3@cornell.edu`. Endereço: 208 Phillips Hall. Acesso em: 10 de novembro de 2024. Disponível em: <https://vanhunteradams.com/Pico/ReactionWheel/Complementary_Filters.html>. Citado na página 48.

ALJARRAH, Q. et al. Major lower extremity amputation: a contemporary analysis from an academic tertiary referral centre in a developing community. *BMC Surgery*, v. 19, n. 1, p. 170, 2019. ISSN 1471-2482. Disponível em: <https://doi.org/10.1186/s12893-019-0637-y>. Citado na página 21.

AMTMANN, D. et al. Health-related profiles of people with lower limb loss. *Archives of Physical Medicine and Rehabilitation*, v. 96, n. 8, p. 1474–1483, 2015. ISSN 0003-9993. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0003999315003354>. Citado na página 21.

AU, S.; BERNIKER, M.; HERR, H. Powered ankle-foot prosthesis to assist level-ground and stair-descent gaits. *Neural Networks*, v. 21, n. 4, p. 654–666, 2008. ISSN 0893-6080. Robotics and Neuroscience. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608008000683>. Citado na página 24.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2015. Citado na página 38.

Bionics For Everyone. *Microprocessor Knee Price List*. 2022. Acesso em: 10 de novembro de 2024. Disponível em: <https://bionicsforeveryone.com/microprocessor-knee-price-list/>. Citado na página 22.

COOPER, R. G. *Winning at New Products: Accelerating the Process from Idea to Launch*. [S.l.]: Basic Books, 2001. Citado na página 45.

DEB, K. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001. Citado na página 44.

DEB, K.; JAIN, H. An introduction to evolutionary multiobjective optimization. *In: Springer, Multi-Objective Optimization*, Springer, p. 3–34, 2014. Citado na página 44.

DU, Y. et al. Surface emg-based inter-session gesture recognition enhanced by deep domain adaptation. *Sensors*, MDPI, v. 17, n. 3, p. 458, 2017. Citado na página 39.

ELBASHEER, A.; KERMANI, B.; RASMUSSEN, H. B. Multi-objective optimization methods for training and evaluating deep learning models. In: *Proceedings of the International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2020. Citado na página 59.

ELSKEN, F.; METZEN, J. H.; HUTTER, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, v. 20, n. 55, p. 1–21, 2019. Citado na página 59.

FAN, Y. et al. Tts synthesis with bidirectional lstm based recurrent neural networks. In: *Fifteenth Annual Conference of the International Speech Communication Association*. [S.l.: s.n.], 2014. Citado na página 39.

FARINA, D.; ASZMANN, O. Bionic limbs: clinical reality and academic promises. *Science Translational Medicine*, American Association for the Advancement of Science, v. 6, n. 257, p. 257ps12, 2014. Disponível em: <https://doi.org/10.1126/scitranslmed.3010453>. Citado na página 24.

FARINA, D. et al. The extraction of neural information from the surface emg for the control of upper-limb prostheses: emerging avenues and challenges. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, IEEE Engineering in Medicine and Biology Society, v. 22, n. 4, p. 797–809, 2014. ISSN 1534-4320. Disponível em: <https://doi.org/10.1109/TNSRE.2014.2305111>. Citado na página 60.

FISCHER, T.; KRAUSS, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, Elsevier, v. 270, n. 2, p. 654–669, 2018. Citado na página 39.

FRANKLE, J.; CARBIN, M. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. Disponível em: <https://arxiv.org/abs/1803.03635>. Citado na página 43.

FRANKLIN, R. J.; MOHANA. Industry 4.0: Real time embedded system with integrated data acquisition. In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. [S.l.: s.n.], 2020. p. 637–642. Citado na página 53.

GHOLAMI, M.; NAPIER, C.; MENON, C. Estimating lower extremity running gait kinematics with a single accelerometer: A deep learning approach. *Sensors*, v. 20, n. 10, p. 2939, 2020. Disponível em: <https://doi.org/10.3390/s20102939>. Citado 3 vezes nas páginas 31, 66 e 72.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 2 vezes nas páginas 42 e 60.

GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. In: IEEE. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2013. p. 6645–6649. Citado na página 39.

GUI, P.; TANG, L.; MUKHOPADHYAY, S. Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion. In: *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*. [S.l.: s.n.], 2015. p. 2004–2009. Citado na página 29.

GUI, P.; TANG, L.; MUKHOPADHYAY, S. Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion. In: *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*. [S.l.: s.n.], 2015. p. 2004–2009. Citado na página 48.

HAFER, J. F. et al. Challenges and advances in the use of wearable sensors for lower extremity biomechanics. *Journal of Biomechanics*, v. 157, p. 111714, 2023. ISSN 0021-9290. Disponível em: <https://www.sciencedirect.com/science/article/pii/S002192902300283X>. Citado na página 25.

HAN, D.; ZHANG, W.; ZHANG, H. Adaptive critic designs for nonlinear optimal control with neural networks: A survey. *IEEE Transactions on Cybernetics*, IEEE, v. 49, n. 5, p. 1791–1804, 2019. Citado na página 35.

HAN, Y. et al. Dynamic neural networks: A survey. *CoRR*, abs/2102.04906, 2021. Disponível em: <https://arxiv.org/abs/2102.04906>. Citado 3 vezes nas páginas 31, 62 e 69.

HAN, Y. C.; WONG, K. I.; MURRAY, I. 2-point error estimation algorithm for 3-d thigh and shank angles estimation using imu. *IEEE Sensors Journal*, v. 18, n. 20, p. 8525–8531, 2018. Citado na página 29.

HEIM, L. et al. *Measuring what Really Matters: Optimizing Neural Networks for TinyML*. 2021. Citado 5 vezes nas páginas 31, 41, 42, 43 e 60.

HERR, H. Exoskeletons and orthoses: classification, design challenges and future directions. *Journal of NeuroEngineering and Rehabilitation*, v. 6, n. 1, p. 21, 2009. ISSN 1743-0003. Disponível em: <https://doi.org/10.1186/1743-0003-6-21>. Citado na página 24.

HILLSON, D. Extending the risk process to manage opportunities. *International Journal of Project Management*, Elsevier, v. 20, n. 3, p. 235–240, 2002. Citado na página 45.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>. Citado na página 38.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, p. 1735–1780, 1997. Disponível em: <https://api.semanticscholar.org/CorpusID:1915014>. Citado na página 42.

HUANG, H. et al. Continuous locomotion-mode identification for prosthetic legs based on neuromuscular–mechanical fusion. *Biomedical Engineering, IEEE Transactions on*, v. 58, p. 2867 – 2875, 11 2011. Citado na página 30.

HUANG, H. et al. Continuous locomotion-mode identification for prosthetic legs based on neuromuscular-mechanical fusion. *IEEE Transactions on Biomedical Engineering*, IEEE, v. 58, n. 10, p. 2867–2875, 2011. Disponível em: <https://doi.org/10.1109/TBME.2011.2161671>. Citado na página 24.

HUANG, Y. et al. Real-time intended knee joint motion prediction by deep-recurrent neural networks. *IEEE Sensors Journal*, IEEE, v. 19, n. 23, p. 11503–11509, 2019. Citado 2 vezes nas páginas 26 e 30.

HUANG, Y. et al. Real-time intended knee joint motion prediction by deep-recurrent neural networks. *IEEE Sensors Journal*, v. 19, n. 23, p. 11503–11509, 2019. Citado na página 72.

HUSSAIN, M.; PARK, H. Electroencephalography (eeg)-based computer-aided technique to diagnose epilepsy. *Sensors*, MDPI, v. 19, n. 7, p. 1485, 2019. Citado na página 39.

HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. 2nd. ed. Melbourne, Australia: OTexts, 2018. OTexts.com/fpp2. Citado 2 vezes nas páginas 59 e 60.

(IBGE), I. B. de Geografia e E. *Em 2021, pobreza tem aumento recorde e atinge 62,5 milhões de pessoas, maior nível desde 2012*. 2021. Acesso em: 5 de novembro de 2024. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/35687-em-2021-pobreza-tem-aumento-recorde-e-atinge-62-5-milhoes-de-pessoas-maior-nivel-desde-2012>. Citado na página 21.

Instituto Brasileiro de Geografia e Estatística (IBGE). *IBGE divulga o rendimento domiciliar per capita e o coeficiente de desequilíbrio regional de 2022*. 2024. Acesso em: 10 de novembro de 2024. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/37023-ibge-divulga-o-rendimento-domiciliar-per-capita-e-o-coeficiente-de-desequilibrio-regional-de-2022>. Citado na página 22.

JIN, Y.; SENDHOFF, B. Simultaneous structural and parameter optimization of feedforward neural networks using evolutionary algorithms. *In: Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers Inc., p. 144–151, 2001. Citado na página 44.

JOZEFOWICZ, R. et al. Exploring the limits of language modeling. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2016. p. 277–286. Citado na página 38.

KAPLAN, J. et al. *Scaling Laws for Neural Language Models*. 2020. Disponível em: <https://arxiv.org/abs/2001.08361>. Citado na página 43.

KARAKISH, M.; FOUZ, M. A.; ELSAWAF, A. Gait trajectory prediction on an embedded microcontroller using deep learning. *Sensors*, v. 22, n. 21, 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/21/8441>. Citado 5 vezes nas páginas 26, 31, 63, 66 e 72.

KENDRYTE. *Kendryte K210 Datasheet*. 2018. <https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_k210_datasheet_en.pdf>. Citado na página 64.

KIRANYAZ, S. et al. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, v. 151, p. 107398, 2021. ISSN 0888-3270.

Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327020307846>. Citado 4 vezes nas páginas 11, 36, 37 e 38.

KIRANYAZ, S.; INCE, T.; GABBOUJ, M. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, v. 63, n. 3, p. 664–675, 2016. Citado na página 37.

KLUGE, F. et al. Towards mobile gait analysis: Concurrent validity and test-retest reliability of an inertial measurement system for the assessment of spatio-temporal gait parameters. *Sensors*, v. 17, n. 7, 2017. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/17/7/1522>. Citado na página 30.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time series. In: _____. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998. p. 255–258. ISBN 0262511029. Citado na página 35.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 43.

LIM, B.; ZOHREN, S.; ROBERTS, S. Time series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 379, n. 2194, p. 20200209, 2021. Citado na página 35.

MA, X. et al. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. In: *Transportation Research Board 94th Annual Meeting*. [S.l.: s.n.], 2015. p. 1–12. Citado na página 39.

MADGWICK, S. O. H.; HARRISON, A. J. L.; VAIDYANATHAN, R. Estimation of imu and marg orientation using a gradient descent algorithm. In: *2011 IEEE International Conference on Rehabilitation Robotics*. [S.l.: s.n.], 2011. p. 1–7. Citado na página 46.

MAR, T. *Insight into LSTM*. 2023. Acessado em: 11 de novembro de 2024. Disponível em: <https://thorirmar.com/post/insight_into_lstm/>. Citado 2 vezes nas páginas 11 e 39.

MARCO, V. S. et al. Optimizing deep learning inference on embedded systems through adaptive model selection. *CoRR*, abs/1911.04946, 2019. Disponível em: <http://arxiv.org/abs/1911.04946>. Citado na página 62.

MCDONALD, C. L. et al. Global prevalence of traumatic non-fatal limb amputation. *Prosthetics and Orthotics International*, v. 45, n. 2, p. 105–114, 2021. Citado na página 21.

MUNDT, M. et al. Prediction of lower limb joint angles and moments during gait using artificial neural networks. *Medical & Biological Engineering & Computing*, v. 58, n. 1, p. 211–225, 2020. Citado na página 72.

Pacini Panebianco, G. et al. Analysis of the performance of 17 algorithms from a systematic review: Influence of sensor position, analysed variable and computational approach in gait timing estimation from imu measurements. *Gait & Posture*, v. 66, p. 76–82, 2018. ISSN 0966-6362. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0966636218304004>. Citado 2 vezes nas páginas 29 e 55.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 34.

SAATY, T. L. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, Inderscience Publishers, v. 1, n. 1, p. 83–98, 2008. Citado na página 45.

SABATINI, A. M. Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing. *Sensors*, v. 11, n. 2, p. 1489–1525, 2011. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/11/2/1489>. Citado na página 46.

SAINATH, T. N. et al. Learning the speech front-end with raw waveform cldnns. In: *Interspeech 2015*. [S.l.: s.n.], 2015. p. 1–5. ISSN 2958-1796. Citado na página 37.

SAOUD, L. S. et al. *Improving Knee Joint Angle Prediction through Dynamic Contextual Focus and Gated Linear Units*. 2023. Disponível em: <https://arxiv.org/abs/2306.06900>. Citado na página 72.

SAWICKI, G.; GORDON, K.; FERRIS, D. Powered lower limb orthoses: Applications in motor adaptation and rehabilitation. In: . [S.l.: s.n.], 2005. v. 2005, p. 206 – 211. ISBN 0-7803-9003-2. Citado na página 24.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, Elsevier, v. 61, p. 85–117, 2015. Citado na página 35.

SEEL, T.; RAISCH, J.; SCHAUER, T. Imu-based joint angle measurement for gait analysis. *Sensors (Basel)*, v. 14, n. 4, p. 6891–6909, 2014. Citado 3 vezes nas páginas 29, 46 e 49.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1409.1556>. Citado na página 42.

Sociedade Brasileira de Angiologia e de Cirurgia Vascular. *Brasil bate recorde de amputações de pés e pernas em decorrência do diabetes*. 2023. <https://sbacv.org.br/brasil-bate-recorde-de-amputacoes-de-pes-e-pernas-em-decorrencia-do-diabetes/>. Acesso em: 5 de novembro de 2024. Citado na página 21.

SU, B.; GUTIERREZ-FAREWIK, E. M. Gait trajectory and gait phase prediction based on an lstm network. *Sensors*, v. 20, n. 24, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/24/7127>. Citado 2 vezes nas páginas 30 e 72.

SUP, F.; VAROL, H. A.; GOLDFARB, M. Upslope walking with a powered knee and ankle prosthesis: initial results with an amputee subject. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, IEEE Engineering in Medicine and Biology Society, v. 19, n. 1, p. 71–78, 2011. Disponível em: <https://doi.org/10.1109/TNSRE.2010.2087360>. Citado na página 24.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2014. p. 3104–3112. Citado na página 38.

TANG, D.; QIN, B.; LIU, T. Document modeling with gated recurrent neural network for sentiment classification. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2015. p. 1422–1432. Citado na página 38.

TENSORFLOW. *Convert RNN models to TensorFlow Lite*. 2024. <https://www.tensorflow.org/lite/models/convert/rnn>. Accessed on: May 20, 2024. Citado na página 66.

TRAN, L.; CHOI, D. Data augmentation for inertial sensor-based gait deep neural network. *IEEE Access*, v. 8, p. 12364–12378, 2020. Citado 2 vezes nas páginas 49 e 51.

TSINGANOS, P. et al. Data augmentation of surface electromyography for hand gesture recognition. *Sensors*, v. 20, n. 17, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/17/4892>. Citado na página 49.

WAN, C.; WANG, L.; PHOHA, V. V. A survey on gait recognition. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 5, ago. 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3230633>. Citado na página 62.

WANG, D.; CHEN, J. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, IEEE, v. 26, n. 10, p. 1702–1726, 2018. Citado na página 35.

World Health Organization. *Assistive technology*. 2023. <https://www.who.int/news-room/fact-sheets/detail/assistive-technology>. Citado na página 21.

World Health Organization. *Disability and Health*. 2023. <https://www.who.int/news-room/fact-sheets/detail/disability-and-health>. Citado na página 21.

XIA, Y. et al. Detecting arrhythmia from 12-lead ecg using deep neural network. *Journal of Healthcare Engineering*, Hindawi, v. 2018, p. 1–8, 2018. Citado na página 39.

Xplicity. *Complementary Filtering Method*. 2023. <https://xplicity.com/complementary-filtering-method/>. Accessed: 2024-05-18. Citado 2 vezes nas páginas 11 e 48.

YOUNG, A. J.; FERRIS, D. P. State of the art and future directions for lower limb robotic exoskeletons. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 25, n. 2, p. 171–182, 2017. Citado na página 25.

YOUNG, A. J.; HARGROVE, L. J. A classification method for user-independent intent recognition for transfemoral amputees using powered lower limb prostheses. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 24, n. 2, p. 217–225, 2016. Citado na página 30.

YOUNG, A. J.; SIMON, A. M.; HARGROVE, L. J. A training method for locomotion mode prediction using powered lower limb prostheses. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 22, n. 3, p. 671–677, 2014. Citado na página 25.

ZAROUG, A. et al. Prediction of gait trajectories based on the long short term memory neural networks. *PLOS ONE*, Public Library of Science, v. 16, n. 8, p. 1–19, 08 2021. Disponível em: <https://doi.org/10.1371/journal.pone.0255597>. Citado na página 30.

ZEROUAL, A. et al. Deep learning methods for forecasting covid-19 time-series data: A comparative study. *Chaos, Solitons & Fractals*, v. 140, p. 110121, 2020. ISSN 0960-0779. Disponível em: <https://www.sciencedirect.com/science/article/pii/S096007792030518X>. Citado na página 25.

ZHANG, D.; EVANGELISTI, S.; LETTIERI, P. Multiobjective optimization of battery energy storage system in distribution grids. *IEEE Transactions on Smart Grid*, IEEE, v. 4, n. 2, p. 856–865, 2013. Citado 2 vezes nas páginas 25 e 45.

ZHANG, W. et al. A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors*, v. 17, n. 2, 2017. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/17/2/425>. Citado na página 37.

ZHENG, K.; ZHU, Y.; ZHANG, M. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, IEEE, p. 1–6, 2017. Citado na página 39.

ZIHAJEHZADEH, S.; PARK, E. J. A novel biomechanical model-aided imu/uwb fusion for magnetometer-free lower body motion capture. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, IEEE, v. 46, n. 7, p. 1005–1016, 2016. Citado 2 vezes nas páginas 46 e 49.

Apêndices

APÊNDICE A – Código Fonte do Sistema de Aquisição de Dados

Listing A.1 – Código fonte do ESP32 para aquisição de dados

```
#include <Arduino.h>
#include <WebServer.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <ArduinoJson.h>
#include "Wire.h"
#include <MPU6050_light.h>
#include <Smoothed.h>

MPU6050 mpu(Wire);
MPU6050 mpu2(Wire1);

#define CONSOLE_IP "192.168.1.2"
#define CONSOLE_PORT 4210
#define SDA_2 33
#define SCL_2 32
const char *ssid = "ESP32";
const char *password = "12345678";
WiFiUDP Udp;
IPAddress local_ip(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
WebServer server(80);

QueueHandle_t xQueue1;
QueueHandle_t xQueue2;

TimerHandle_t xTimer1;
TimerHandle_t xTimer2;

#define SAMPLE_TIME 10
#define LED 2

void timerReadAngle1(TimerHandle_t xTimer)
```

```
{
    mpu.update();
    float angle = mpu.getAngleX() - 90;
    xQueueOverwrite(xQueue1, &angle);
}

void timerReadAngle2(TimerHandle_t xTimer)
{
    mpu2.update();
    float angle = mpu2.getAngleX() - 90;
    xQueueOverwrite(xQueue2, &angle);
}

volatile bool recalibrar = false;

void taskButtonCal(void *pvParameters)
{
    while (1)
    {
        if (recalibrar)
        {
            xTimerStop(xTimer1, 0);
            xTimerStop(xTimer2, 0);
            Serial.println("Recalibrando sensores");
            digitalWrite(LED, !digitalRead(LED));
            mpu.calcOffsets();
            mpu2.calcOffsets();
            digitalWrite(LED, !digitalRead(LED));
            Serial.println("Recalibrado");
            recalibrar = false;
            xTimerStart(xTimer1, 0);
            xTimerStart(xTimer2, 0);
        }
        vTaskDelay(1000);
    }
}

void IRAM_ATTR isr()
{
    recalibrar = true;
}
```

```
void getSensorData()
{
    int numSamples = server.arg("samples").toInt();

    Serial.println("Recebi uma requisicao de " + String(
       numSamples) + " amostras");

    if (numSamples <= 0 || numSamples > 1000)
    {
        server.send(400, "text/plain", "Numero invalido de
           amostras");
        return;
    }

    float angle1;
    float angle2;

    String dataOut = "";

    for (int i = 0; i < numSamples; i++)
    {
        digitalWrite(LED, LOW);
        xQueueReceive(xQueue1, &angle1, portMAX_DELAY);
        xQueueReceive(xQueue2, &angle2, portMAX_DELAY);
        dataOut += String(angle1) + "," + String(angle2) + "\n";
    }

    digitalWrite(LED, HIGH);

    Serial.println(dataOut);

    server.send(200, "text/plain", dataOut);
}

const int buttonPin = 0;

void setup()
{
    Serial.begin(115200);

    pinMode(LED, OUTPUT);
```

```
Wire.begin();
Wire1.begin(SDA_2, SCL_2);

byte st = mpu.begin(3, 1);
while (st != 0)
{
}

delay(1000);
mpu.calcOffsets();

byte status2 = mpu2.begin(3, 1);
while (status2 != 0)
{
}

delay(1000);
mpu2.calcOffsets();

mpu.setFilterGyroCoef(0.98);
mpu2.setFilterGyroCoef(0.98);

WiFi.softAP(ssid, password);
WiFi.softAPConfig(local_ip, gateway, subnet);

server.on("/sensors", HTTP_GET, getSensorData);
server.begin();

delay(1000);

xTaskCreate(taskButtonCal, "taskButton", 4096, NULL, 1, NULL)
   ;

pinMode(buttonPin, INPUT_PULLUP);

attachInterrupt(digitalPinToInterrupt(buttonPin), isr,
   FALLING);

digitalWrite(LED, !digitalRead(LED));

xQueue1 = xQueueCreate(1, sizeof(float));
xQueue2 = xQueueCreate(1, sizeof(float));
```

```
    xTimer1 = xTimerCreate("Timer1", pdMS_TO_TICKS(SAMPLE_TIME),
       pdTRUE, (void *)0, timerReadAngle1);
    xTimer2 = xTimerCreate("Timer2", pdMS_TO_TICKS(SAMPLE_TIME),
       pdTRUE, (void *)1, timerReadAngle2);
    xTimerStart(xTimer1, 0);
    xTimerStart(xTimer2, 0);
}

void loop()
{
    server.handleClient();
}
```

APÊNDICE B – Código Fonte da Interface Gráfica (GUI)

B.1 Código Fonte da GUI

Listing B.1 – Código fonte da GUI para aquisição e leitura de dados

```
from tkinter import *
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np
from tkinter import ttk
import socket
import pandas as pd
import requests
import time

root = Tk()
root.geometry("1200x1000")

LOCAL_UDP_IP = "192.168.1.2"
SHARED_UDP_PORT = 4210

root.title("GUI Aquisição de Dados")
frameLeft = Frame(root)

SAMP_FREQ = 100

def getData():
    global data1
    global data2
    global progress
    global offset1
    global offset2

    tempoAmostragem = slider.get()
    numAmostras = int(tempoAmostragem * SAMP_FREQ)
    params = {"samples": numAmostras}
    response = requests.get("http://192.168.1.1/sensors", params=
```

```
        params)
    data = response.content

    data1 = []
    data2 = []
    lines = data.decode().strip().split("\n")
    for line in lines:
        sensor1, sensor2 = map(float, line.split(","))
        data1.append(sensor1 + offset1)
        data2.append(sensor2 + offset2)

    progress["value"] = 100
    progress.update()
    addGraph(data1, data2)

try:
    df = pd.read_csv("dados.csv")
except:
    df = pd.DataFrame(columns=["Angulo 1", "Angulo 2"])

def addData():
    global df
    print("Adicionando dados")

    df2 = pd.DataFrame({"Angulo 1": data1, "Angulo 2": data2},
        dtype=np.float32)
    df = pd.concat([df, df2], ignore_index=True)

def saveData():
    global df
    df.to_csv("dados.csv", index=False)

labelTempo = Label(frameLeft, text="Tempo amostragem (s):", font
    =("Arial", 20))
slider = Scale(
    frameLeft,
    from_=1,
    to=10,
    orient=HORIZONTAL,
    length=300,
    width=20,
    font=("Arial", 20),
```

```
)

labelFs = Label(
    frameLeft,
    text="Frequência de amostragem (Hz): " + str(SAMP_FREQ),
    font=("Arial", 20),
)

progress = ttk.Progressbar(frameLeft, orient=HORIZONTAL, length
   =300, mode="determinate")

buttonSalvar = Button(
    frameLeft,
    text="Salvar CSV",
    font=("Arial", 20, "bold"),
    pady=15,
    padx=10,
    command=saveData,
)

buttonAdicionar = Button(
    frameLeft,
    text="Adicionar dados para o dataframe",
    font=("Arial", 20, "bold"),
    pady=15,
    padx=10,
    command=addData,
)

buttonAmostrar = Button(
    frameLeft,
    text="Amostrar",
    font=("Arial", 20, "bold"),
    pady=15,
    padx=10,
    command=getData,
)

labelLeft = Label(
    frameLeft, text="Amostragem", font=("Arial", 20, "bold"),
       padx=10, pady=10
)
```

```
offset1 = 0
offset2 = 0

offset_text1 = StringVar()
offset_text2 = StringVar()

def calculate_offset():
    global offset1
    global offset2

    params = {"samples": 300}
    response = requests.get("http://192.168.1.1/sensors", params=
       params)
    data = response.content
    data1 = []
    data2 = []
    lines = data.decode().strip().split("\n")
    for line in lines:
        sensor1, sensor2 = map(float, line.split(","))
        data1.append(sensor1)
        data2.append(sensor2)

    mean1 = np.mean(data1)
    mean2 = np.mean(data2)

    offset1 = -mean1
    offset2 = -mean2

    offset_text1.set(f"Offset Sensor 1: {offset1:.2f}")
    offset_text2.set(f"Offset Sensor 2: {offset2:.2f}")

buttonOffset = Button(
    frameLeft,
    text="Calc. Offset",
    font=("Arial", 20, "bold"),
    pady=15,
    padx=10,
    command=calculate_offset,
)
buttonOffset.pack(side=BOTTOM, pady=10)
```

```
frameLeft.pack(side=LEFT, fill=Y)
labelLeft.pack()
labelTempo.pack(padx=10, pady=10)
slider.pack()
labelFs.pack()

buttonSalvar.pack(side=BOTTOM, pady=10)
buttonAdicionar.pack(side=BOTTOM, pady=10)
buttonAmostrar.pack(side=BOTTOM, pady=10)
progress.pack(pady=10, side=BOTTOM)

rightFrame = Frame(root)
labelRight = Label(
    rightFrame, text="Dados", font=("Arial", 20, "bold"), padx
       =10, pady=10
)

rightFrame.pack(side=RIGHT, fill="both", expand=True)
labelRight.pack()

def addGraph(data1, data2):
    for widget in rightFrame.winfo_children():
        widget.destroy()

    data1_array = np.array(data1)
    data2_array = np.array(data2)

    fig = plt.figure(figsize=(3, 3))
    ax1 = fig.add_subplot(311)
    ax1.plot(data1_array)
    ax1.set_title("Ângulo 1")
    ax2 = fig.add_subplot(312)
    ax2.plot(data2_array)
    ax2.set_title("Ângulo 2")

    ax3 = fig.add_subplot(313)
    ax3.plot(-(data1_array - data2_array))
    ax3.set_title("Diferença")

    canvas = FigureCanvasTkAgg(fig, master=rightFrame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=TOP, fill=BOTH, expand=True)
```

```
root.mainloop()
```

B.2 Interface de aquisição

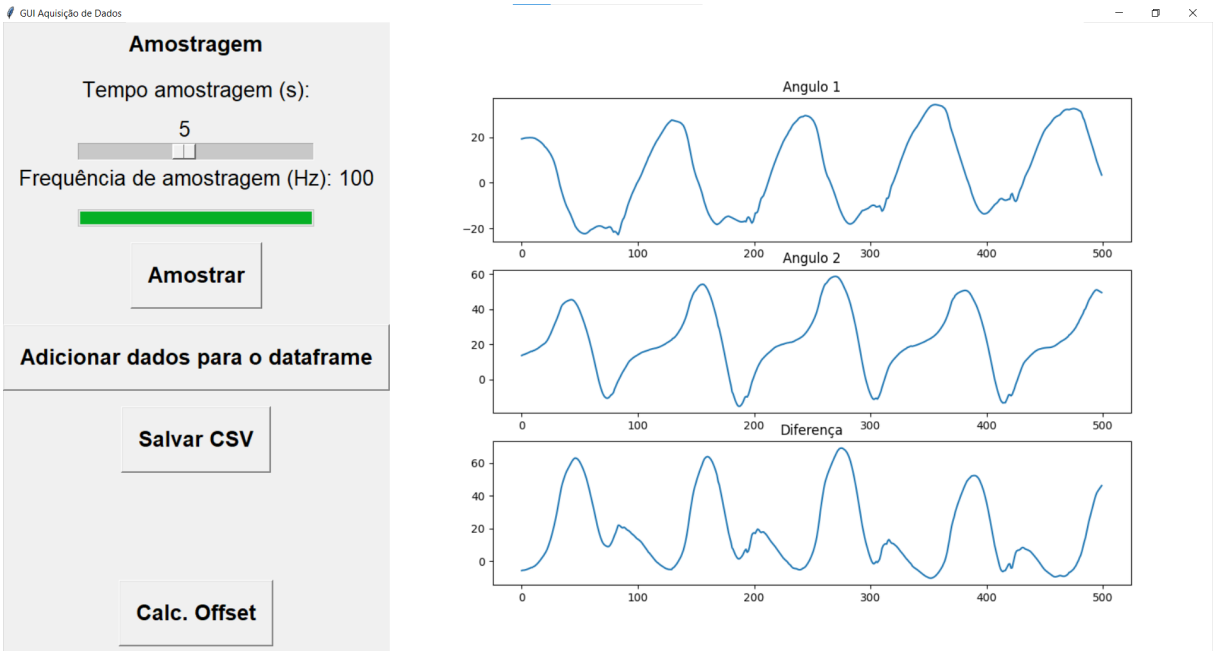


Figura 30 – Interface gráfica desenvolvida para a aquisição e leitura dos dados.

APÊNDICE C – Código Fonte das Funções de Criação de Modelos

C.1 Código Fonte para Criação dos Modelos MLP, CNN e LSTM

Listing C.1 – Funções de criação de modelos MLP, CNN e LSTM com padrões de arquitetura

```
import tensorflow as tf
from tensorflow.keras import layers, models

def create_mlp_model(input_shape, alpha, beta, horizon):
    model = models.Sequential()
    model.add(layers.Input(shape=input_shape))

    neurons = alpha
    for i in range(beta):
        model.add(layers.Dense(neurons, activation='relu'))
        neurons = max(neurons // 2, 4)

    model.add(layers.Dense(horizon, activation='linear'))

    model.compile(optimizer='adam', loss='mean_squared_error')
    return model


def create_cnn1d_model(input_shape, alpha, beta, horizon):
    model = models.Sequential()
    model.add(layers.Input(shape=input_shape))

    filters = alpha
    for i in range(beta):
        model.add(layers.Conv2D(filters, (3,1), padding='same',
           activation='relu'))
        model.add(layers.Conv2D(filters, (3,1), padding='same',
           activation='relu'))
        model.add(layers.MaxPooling2D((2,1)))
        filters = filters * 2
```

```
    model.add(layers.Flatten())
    model.add(layers.Dense(alpha, activation='relu'))
    model.add(layers.Dense(horizon, activation='linear'))

    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

def create_cnn2d_model(input_shape, alpha, beta, horizon):
    model = models.Sequential()
    model.add(layers.Input(shape=input_shape))

    filters = alpha
    for i in range(beta):
        model.add(layers.Conv2D(filters, (3,3), padding='same',
           activation='relu'))
        model.add(layers.Conv2D(filters, (3,3), padding='same',
           activation='relu'))
        model.add(layers.MaxPooling2D((2,2)))
        filters = filters * 2

    model.add(layers.Flatten())
    model.add(layers.Dense(alpha, activation='relu'))
    model.add(layers.Dense(horizon, activation='linear'))

    model.compile(optimizer='adam', loss='mean_squared_error')
    return model


def create_lstm_model(input_shape, alpha, beta, horizon):
    model = models.Sequential()
    model.add(layers.Input(shape=input_shape))

    for i in range(beta):
        return_sequences = (i < beta - 1)
        model.add(layers.LSTM(alpha, return_sequences=false))

    model.add(layers.Dense(horizon, activation='linear'))

    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

APÊNDICE D – Código Fonte para Inferência em MicroPython

D.1 Código Fonte

Listing D.1 – Código fonte em MicroPython para inferência dos dados

```
import sensor, image, time, lcd
import KPU as kpu
import _thread
import os

lcd.init(freq=15000000)
lcd.clear()

Lock = _thread.allocate_lock()

task_slow = kpu.load(0x300000)
task_normal = kpu.load(0x500000)
task_fast = kpu.load(0x700000)
task_clf = kpu.load(0x900000)

kpu.set_outputs(task_slow, 0, 1, 10, 1)
kpu.set_outputs(task_normal, 0, 1, 10, 1)
kpu.set_outputs(task_fast, 0, 1, 10, 1)
kpu.set_outputs(task_clf, 0, 1, 3, 1)

max_id = None
new_data_available = False
data_index = 0

def load_all_test_data(directory):
    test_data_list = []
    filenames = []
    for filename in os.listdir(directory):
        if filename.endswith(".txt"):
            full_path = os.path.join(directory, filename)
            with open(full_path, 'r') as f:
                lines = f.readlines()
```

```
                test_data = [float(line.strip()) for line in
                    lines]
                test_data_list.append(test_data)
                filenames.append(filename)
    return test_data_list, filenames

test_data_list, filenames = load_all_test_data("/sd/data/")

def classifier_thread():
    global max_id, new_data_available, data_index,
        current_data_to_process, current_file
    while True:
        Lock.acquire()
        if data_index >= len(test_data_list):
            Lock.release()
            break
        current_data = test_data_list[data_index]
        current_filename = filenames[data_index]
        data_index += 1
        Lock.release()

        fmap = kpu.forward(task_clf, current_data)

        plist = fmap[:]
        pmax = max(plist)
        max_id_local = plist.index(pmax)

        Lock.acquire()
        max_id = max_id_local
        new_data_available = True
        current_data_to_process = current_data
        current_file = current_filename
        Lock.release()

        time.sleep_ms(10)

def regressor_thread():
    global max_id, new_data_available, current_data_to_process,
        current_file
    while True:
        Lock.acquire()
        if new_data_available:
```

```
            local_max_id = max_id
            local_data = current_data_to_process
            local_filename = current_file
            new_data_available = False
            Lock.release()

            if local_max_id == 0:
                fmap2 = kpu.forward(task_slow, local_data)
                gait_type = "Marcha Curta"
            elif local_max_id == 1:
                fmap2 = kpu.forward(task_normal, local_data)
                gait_type = "Marcha Natural"
            elif local_max_id == 2:
                fmap2 = kpu.forward(task_fast, local_data)
                gait_type = "Marcha Longa"
            else:
                print("ID inválido para o arquivo:",
                   local_filename)
                continue

            plist2 = fmap2[:]
            print("Arquivo:", local_filename)
            print("Tipo de marcha detectado:", gait_type)
            print("Saída do regressor:", plist2)
            lcd.draw_string(0, 80, "Output: " + str(plist2))
        else:
            if data_index >= len(test_data_list):
                Lock.release()
                break
            Lock.release()
            time.sleep_ms(10)

current_data_to_process = None
current_file = None

_thread.start_new_thread(classifier_thread, ())
_thread.start_new_thread(regressor_thread, ())

print("Threads iniciadas")
while True:
    Lock.acquire()
    if data_index >= len(test_data_list) and not
```

```
        new_data_available:
            Lock.release()
            print("Processamento concluído")
            break
    Lock.release()
    time.sleep_ms(1000)
```