

Universidade de Brasília

Faculdade de Ciências e Tecnologias em Engenharia

Avaliação de Modelos Baseados em Aprendizado de Máquina para Previsão de Comportamentos de Robôs Futebolísticos

Luiz Felipe Rodrigues e Silva

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA ELETRÔNICA

Brasília

2025

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia

Avaliação de Modelos Baseados em Aprendizado de Máquina para Previsão de Comportamentos de Robôs Futebolísticos

Luiz Felipe Rodrigues e Silva

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Diogo Caetano Garcia

Brasília
2025

FICHA CATALOGRÁFICA

Rodrigues e Silva, Luiz Felipe.

Avaliação de Modelos Baseados em Aprendizado de Máquina para Previsão de Comportamentos de Robôs Futebolísticos / Luiz Felipe Rodrigues e Silva; orientador Diogo Caetano Garcia. -- Brasília, 2025.

48 p.

Trabalho de Conclusão de Curso (Engenharia Eletrônica) -- Universidade de Brasília, 2025.

1. RoboCup SSL. 2. Aprendizado de Máquina. 3. Regressão Linear. I. Caetano Garcia, Diogo, orient. II. Título.

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia

Avaliação de Modelos Baseados em Aprendizado de Máquina para Previsão de Comportamentos de Robôs Futebolísticos

Luiz Felipe Rodrigues e Silva

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, 11 de dezembro de 2025:

Prof. Dr. Diogo Caetano Garcia,
UnB/FCTE
Orientador

Prof. Dr. Cristiano Jacques Miosso
UnB/FCTE
Convidado 1

Prof. Dr. Luciano Emídio Neves da Fonseca
UnB/FCTE
Convidado 2

Dedico este trabalho aos meus pais e à minha família, pelo amor, apoio e incentivo em todos os momentos.

Agradecimentos

Agradeço, antes de tudo, a Deus, pela força e perseverança que me guiaram em cada etapa desta jornada. Aos meus pais, pelo amor incondicional, pelos incentivos aos estudos e pelos conselhos que me fortaleceram mesmo diante dos desafios, sem esse apoio e sacrifício, nada disso seria possível. Às minhas irmãs, pela cumplicidade, pelo carinho e pela alegria que sempre tornaram o caminho mais leve.

Aos meus amigos e colegas, que compartilharam comigo momentos de aprendizado, descontração e superação. Aos professores e orientadores, que dedicaram seu conhecimento e paciência para me inspirar e preparar para os desafios da graduação.

Um agradecimento especial aos projetos de robótica UnBeatables e Titans, que me proporcionaram não apenas crescimento técnico, mas também experiências memoráveis, viagens e amizades que levarei para a vida. Cada competição, noite de trabalho em equipe e conquista reforçou minha paixão pela engenharia e meu desejo de contribuir para a área.

“Cada vez que você faz uma opção
está transformando sua essência
em alguma coisa um pouco diferente
do que era antes.”
(C.S. Lewis)

Resumo

Este trabalho apresenta o desenvolvimento e a análise comparativa de modelos de aprendizado de máquina para a predição de comportamentos de robôs futebolísticos, utilizando dados reais de posicionamento provenientes de sistemas de visão computacional da *RoboCup Small Size League* (SSL). A metodologia empregada abrangeu a aquisição e conversão de dados, engenharia de atributos cinemáticos e tratamento de ruídos. Duas abordagens distintas foram avaliadas: um modelo de Regressão Linear e uma rede neural recorrente do tipo *Long Short-Term Memory* (LSTM). Os experimentos compararam o desempenho dos modelos em função da variação do tamanho da janela histórica e do horizonte de predição, utilizando como métricas o erro posicional e o tempo de inferência. O modelo de Regressão Linear atingiu um valor de raiz do erro quadrático médio (RMSE) mínimo de 2,23 mm, com tempo médio de inferência de aproximadamente 0,22 ms. A rede LSTM apresentou melhor desempenho preditivo no melhor cenário avaliado, alcançando RMSE de 1,65 mm, porém com tempos de treinamento e inferência significativamente superiores. Os resultados indicam que a Regressão Linear oferece uma solução robusta e de baixa latência para predições de curto prazo. Em contraste, a rede LSTM demonstrou uma maior capacidade de generalização para trajetórias complexas e não-lineares, com um custo computacional superior.

Palavras-chave: RoboCup SSL; Aprendizado de Máquina; Regressão Linear.

Abstract

This paper presents the development and comparative analysis of machine learning models for predicting soccer robot behavior, using real positioning data from computer vision systems in the RoboCup Small Size League (SSL). The methodology employed included data acquisition and conversion, kinematic attribute engineering, and noise treatment. Two distinct approaches were evaluated: a Linear Regression model and a Long Short-Term Memory (LSTM) recurrent neural network. The experiments compared the performance of the models as a function of the variation in the historical window size and the prediction horizon, using positional error and inference time as metrics. The Linear Regression model achieved a minimum root mean square error (RMSE) of 2.23 mm, with an average inference time of approximately 0.22 ms. The LSTM network showed better predictive performance in the best evaluated scenario, achieving an RMSE of 1.65 mm, but with significantly higher training and inference times. The results indicate that Linear Regression offers a robust, low-latency solution for short-term predictions. In contrast, the LSTM network demonstrated greater generalization capacity for complex, nonlinear trajectories, with a higher computational cost.

Keywords: RoboCup SSL; Machine Learning; Linear Regression.

Lista de figuras

Figura 2.1 Dimensões do campo da divisão B da SSL, em mm. 18
Figura 2.2 Marcadores visuais dos robôs. 19
Figura 2.3 Robô da liga SSL-EL desenvolvido pela Titans - UnB 20
Figura 2.4 Representação esquemática de um neurônio artificial. Os sinais de entrada $(x_1, \ldots, x_n)$ são ponderados pelos pesos sinápticos $(w_1, \ldots, w_n)$ e agregados na função de soma (Σ). O resultado é submetido a uma função de ativação $(\varphi(i))$, gerando a resposta de saída (y). 22
Figura 2.5 Representação da arquitetura interna de um bloco LSTM. O diagrama ilustra a interação entre a entrada (x_t), o estado recorrente (r_t) e os portões de controle (i_t, f_t, o_t) que modulam o estado da célula (c_t) para preservar dependências temporais. 23
Figura 3.1 Fluxo do processo metodológico proposto. O diagrama apresenta, de forma sequencial, as etapas de aquisição dos dados, caracterização, pré-processamento, treinamento dos modelos e avaliação dos resultados. . . 26
Figura 3.2 Representação esquemática do janelamento de uma série temporal no instante atual (t) para a previsão multi-passo. O processo utiliza uma Janela Histórica (X), composta pelos N Passos Passados (de $t - N$ a t), como entrada para o modelo. Esta entrada é usada para gerar a Janela de Previsão (Y), que abrange os K Passos Futuros (de $t+1$ a $t+K$), definindo o horizonte de previsão. 30
Figura 4.1 Mapa de calor com a Raiz do Erro Quadrático Médio (RMSE, em mm) do modelo de previsão sob diferentes configurações de janelamento. O eixo Y é a Janela de Previsão (Pred Window, em amostras), e o eixo X é a Janela de Treino (Train Window, em segundos). O mapeamento de cores demonstra que o melhor desempenho (menor RMSE, cores escuras) é atingido com Janelas de Previsão menores e Janelas de Treino maiores, indicando que um histórico de entrada mais longo é vantajoso. 35
Figura 4.2 Comparação espacial entre as posições reais (azul) e as estimadas (laranja) pelo Regressor Linear. A alta densidade de sobreposição entre os pontos evidencia a precisão do modelo em intervalos curtos de tempo. 36
Figura 4.3 Evolução do erro absoluto de previsão para as coordenadas X (azul) e Y (laranja) ao longo do tempo de experimento. O eixo vertical em escala logarítmica permite a visualização de erros que variam em diferentes ordens de grandeza. 37

Figura 4.4 Histograma de distribuição de probabilidade do erro euclidiano. O gráfico evidencia um comportamento de cauda longa, indicando que a grande maioria das previsões apresenta desvios inferiores a 5 mm. 38
Figura 4.5 Tempo de inferência vs. janela de previsão para o regressor linear. Cada curva corresponde a uma janela de treinamento. 38
Figura 4.6 Tempo de treinamento vs. janela de previsão para o regressor linear. Cada curva corresponde a uma janela de treinamento. 39
Figura 4.7 Distribuição de frequências do tempo de treinamento para o regressor linear. O histograma demonstra que a maioria das execuções se concentra em torno de 0,024 segundos (24 ms), indicando estabilidade computacional e baixo custo de processamento para o modelo de regressão. 39
Figura 4.8 Evolução da função de perda (*loss*) e da métrica MAE durante 100 épocas. Observa-se uma convergência rápida nas iterações iniciais e a ausência de distanciamento entre as curvas de treinamento e validação, indicando estabilidade e ausência de overfitting. 41
Figura 4.9 O gráfico apresenta a evolução do Erro Absoluto (em mm) para as coordenadas X (azul) e Y (vermelha) ao longo da sequência de teste. Observa-se que o modelo LSTM mantém estabilidade na maior parte da inferência, com picos de erro localizados (notadamente próximo à amostra 7.000), indicando variações pontuais na precisão da predição em relação aos dados reais. 42
Figura 4.10 Comparação entre a trajetória real e a estimada do robô no plano (XY). A curva (azul) representa os dados reais do conjunto de teste, enquanto a (vermelha) indica as posições preditas pelo modelo LSTM otimizado. . . 43
Figura 4.11 Relação entre o tempo total de treinamento (eixo Y) e o tamanho da janela de dados (eixo X). Observa-se uma tendência geral de crescimento do tempo conforme a janela aumenta, indicando que sequências temporais mais longas demandam maior processamento para a atualização dos pesos da rede LSTM. 43
Figura 4.12 Medição do tempo de inferência do modelo para diferentes janelas. Ao contrário do treinamento, o tempo de resposta apresenta oscilações significativas (picos e vales) que não seguem uma linearidade estrita, sugerindo que a latência de inferência pode ser influenciada por fatores de *overhead* do sistema além do tamanho da entrada. 44

Lista de tabelas

Tabela 2.1 Estados do jogo da SSL *RoboCup*. 19
Tabela 3.1 Composição do Vetor de Características 29
Tabela 3.2 Composição do Vetor de Características por Instante (t) 31
Tabela 3.3 Grade de Hiperparâmetros para os Experimentos de Regressão 32
Tabela 3.4 Resumo dos Hiperparâmetros de Treinamento Ótimos 33
Tabela 3.5 Ambiente de Desenvolvimento e Ferramentas 34
Tabela 4.1 Resultados ótimos obtidos para o modelo de regressão linear 36
Tabela 4.2 Resultados dos Experimentos de Predição com o Modelo LSTM. 40

Sumário

1 Introdução 14
1.1 Objetivos 15
1.1.1 Objetivos Específicos 15
1.2 Estrutura do Documento 15

2 Referencial Teórico 17
2.1 SSL (*Small Size League*) 17
2.1.1 Dimensões do Campo 17
2.1.2 Sistema de Visão e Identificação dos Robôs 18
2.1.3 *Game Controller* 18
2.1.4 Estados do Jogo e Regras 19
2.1.5 Resumo das Regras 20
2.2 Aprendizado de Máquina 20
2.2.1 Aprendizado de Máquina e Aprendizado Profundo 20
2.2.2 Regressão Linear 21
2.2.3 Redes Neurais Artificiais 22
2.2.4 LSTM (*Long Short Term Memory*) 23
2.3 Métricas de Avaliação de Desempenho 23
2.3.1 Erro Médio Absoluto 24
2.3.2 Erro Quadrático Médio 24
2.3.3 Raiz do Erro Quadrático Médio 24
2.3.4 Erro Euclidiano Médio 24
2.4 Interpolação Linear 24
2.5 Cinemática Discreta 25

3 Materiais e Métodos 26
3.1 Aquisição e Conversão dos Dados 27
3.2 Caracterização e Seleção dos Dados 27
3.3 Pré-processamento e Engenharia de Atributos 28
3.3.1 Tratamento e Limpeza dos Dados 28
3.3.2 Engenharia de Atributos e Normalização 28
3.3.3 Formulação do Problema e Janelamento 29
3.4 Definição e Treinamento dos Modelos 30
3.4.1 Definição da Arquitetura e Treinamento da Regressão Linear 30
3.4.2 Cenários de Teste para a Regressão Linear 31
3.4.3 Definição da Arquitetura e Treinamento da LSTM 32

3.4.4 Cenários de Teste para a LSTM 32
3.4.5 Estratégia de Validação 33
3.5 Métricas de Avaliação 33
3.5.1 Métricas de Precisão 34
3.5.2 Avaliação dos Modelos 34
3.6 Ambiente Computacional 34
4 Resultados 35
4.1 Modelo de Regressão Linear 35
4.2 Modelo LSTM 40
4.3 Comparação Entre Modelos 42
5 Conclusões 45
Referências 47

1 Introdução

Criada em 1997, a *RoboCup* é uma das principais competições internacionais de robótica e inteligência artificial, cujo objetivo é promover o avanço da pesquisa em sistemas autônomos por meio de desafios cooperativos e competitivos. Inspirada no futebol, a iniciativa busca fomentar o desenvolvimento de técnicas em robótica e inteligência artificial (Federation, 2025).

O surgimento da *RoboCup* está diretamente associado a um período de grandes avanços na área de inteligência artificial e robótica. A década de 1990 marcou conquistas emblemáticas, como a vitória do sistema *Deep Blue* sobre o campeão mundial de xadrez e o pouso bem-sucedido da missão Mars Pathfinder, que demonstraram o potencial de sistemas inteligentes autônomos para resolução de problemas complexos.

A ideia de utilizar o futebol como ambiente de pesquisa foi inicialmente discutida por Alan Mackworth (Mackworth, 1993) no início da década de 1990 e, de forma independente, por grupos de pesquisadores japoneses que buscavam desafios capazes de estimular avanços significativos em inteligência artificial. Esses esforços culminaram na criação da iniciativa internacional conhecida como *Robot World Cup Initiative*, posteriormente denominada *RoboCup*, reunindo pesquisadores de diferentes países e áreas do conhecimento.

Desde sua primeira edição oficial, realizada em 1997, a *RoboCup* tem se consolidado como um importante laboratório experimental para validação de técnicas de inteligência artificial, robótica e sistemas distribuídos. A competição é organizada em diferentes ligas, cada uma voltada a aspectos específicos da robótica, como visão computacional, coordenação multiagente, aprendizado de máquina e *hardware* embarcado. Entre essas ligas, destaca-se a *Small Size League* (SSL), nesta liga participam equipes formadas por estudantes e pesquisadores de diversas universidades ao redor do mundo que desenvolvem sistemas robóticos capazes de competir em partidas de futebol autônomo, nas quais os robôs devem perceber o ambiente, planejar ações e reagir de forma coordenada aos movimentos dos adversários. Esse ambiente impõe desafios técnicos significativos, exigindo soluções robustas de visão computacional, comunicação em rede, controle preditivo e tomada de decisão em tempo real (Kitano *et al.*, 1997).

Diante desse contexto, a capacidade de antecipar o movimento dos robôs e de objetos durante as partidas de futebol é determinante para o sucesso estratégico dessas equipes, permitindo ao sistema prever situações críticas, interceptar passes, otimizar posicionamentos e ajustar táticas defensivas ou ofensivas de forma adaptativa. Assim, a previsão de trajetórias torna-se um componente central dos sistemas de tomada de decisão em tempo real na SSL.

Historicamente, a estimação e a predição do estado de sistemas robóticos têm sido

abordadas por meio de métodos probabilísticos baseados em modelos dinâmicos, com destaque para o filtro de Kalman e suas extensões. Esses métodos são amplamente utilizados para a filtragem de ruído e a estimação de estados, desde que o modelo dinâmico do sistema e as características estatísticas do ruído sejam conhecidos. No contexto da SSL, isto não é completamente válido, visto que o modelo da equipe adversária não é conhecido *a priori*.

Nos últimos anos, o avanço do Aprendizado de Máquina tem impulsionado o desenvolvimento de abordagens baseadas em dados para a modelagem de dinâmicas complexas e não-lineares (LeCun; Bengio; Hinton, 2015). Arquiteturas como as Redes Neurais Recorrentes (RNNs) e, em particular, as redes com memória de curto e longo prazo (LSTMs, do inglês *Long Short-Term Memory*) (Hochreiter; Schmidhuber, 1997) têm se mostrado eficazes na captura de dependências temporais em sequências espaciais e temporais, sendo amplamente aplicadas à previsão de séries temporais.

1.1 Objetivos

O objetivo principal deste trabalho é investigar a viabilidade do uso de modelos de aprendizado de máquina para a previsão de trajetórias da bola e de agentes robóticos em partidas da SSL, considerando as restrições dinâmicas, temporais e computacionais impostas pelo ambiente competitivo.

1.1.1 Objetivos Específicos

- **Análise dos dados:** Caracterizar e pré-processar conjuntos de dados de partidas reais da SSL, aplicando técnicas de filtragem e engenharia de características.
- **Desenvolvimento do modelo:** Implementar e treinar preditores via Regressão Linear e LSTM para a previsão de trajetórias de agentes.
- **Avaliação de desempenho:** Quantificar a precisão preditiva dos modelos desenvolvidos a partir do erro de trajetória.
- **Análise de aplicabilidade:** Avaliar a viabilidade de implementação dos modelos em cenários reais de competição, considerando os aspectos computacionais de latência e a robustez.

1.2 Estrutura do Documento

Este documento está organizado da seguinte forma:

- **Capítulo 2:** Apresenta o referencial teórico sobre modelos de previsão de trajetórias, tais como modelos de regressão linear e redes neurais recorrentes.

- **Capítulo 3:** Descreve a metodologia experimental, incluindo a caracterização dos dados de entrada, seu pré-processamento e as arquiteturas dos modelos.
- **Capítulo 4:** Apresenta a análise comparativa de resultados entre os modelos de Regressão Linear e LSTM, quantificando o desempenho e a latência computacional.
- **Capítulo 5:** Conclui o trabalho com discussão dos resultados e propostas para trabalhos futuros.

2 Referencial Teórico

2.1 SSL (*Small Size League*)

A *Small Size League* (SSL) é uma das categorias da *RoboCup*, uma competição internacional que visa promover avanços em robótica e inteligência artificial através de desafios práticos e competitivos (Kitano *et al.*, 1997). Na SSL, dois times, compostos por até 6 ou 11 robôs autônomos, competem em partidas que simulam jogos de futebol. A principal característica dessa categoria é a autonomia dos robôs, que são controlados remotamente por um computador central, responsável por processar informações do ambiente e enviar comandos para a execução de estratégias predeterminadas (Zickler *et al.*, 2009).

O ambiente de jogo é monitorado por um sistema de câmeras posicionadas acima do campo, que capturam imagens em tempo real. Essas imagens são processadas por um sistema de visão computacional unificado, que identifica a posição dos robôs, da bola e de outros elementos relevantes. Com base nessas informações, o sistema central calcula as ações ideais para cada robô, permitindo que eles se movimentem e interajam de forma autônoma no campo (Rocha; Dias; Campos, 2012).

Os robôs utilizados na competição SSL são equipados com rodas omnidirecionais, permitindo-os realizarem movimentos em qualquer direção sem a necessidade de reorientação (Romero *et al.*, 2014). O controle do movimento é feito ajustando as velocidades de cada roda, o que possibilita alterar a velocidade linear, angular e a orientação do robô no campo.

A competição destaca-se pelo uso de tecnologias avançadas em robótica, sistemas embarcados e visão computacional, proporcionando um ambiente propício para o desenvolvimento e aplicação de técnicas de controle e inteligência artificial (Stone; Veloso, 2000).

2.1.1 Dimensões do Campo

O campo utilizado na Divisão B da SSL possui dimensões padronizadas de 10,4 m de comprimento por 7,4 m de largura, com uma área de jogo efetiva de 9×6 m^2. As marcações do campo, como as linhas de fundo, laterais, áreas do gol e círculo central, seguem padrões definidos pela organização da competição. Pequenas variações nas dimensões lineares (até $\pm 10\%$) são permitidas para acomodar diferentes tipos de arena (Federation, 2023).

A Figura 2.1 ilustra as dimensões e marcações do campo da Divisão B, conforme estabelecido pelas regras oficiais da RoboCup (Federation, 2023).

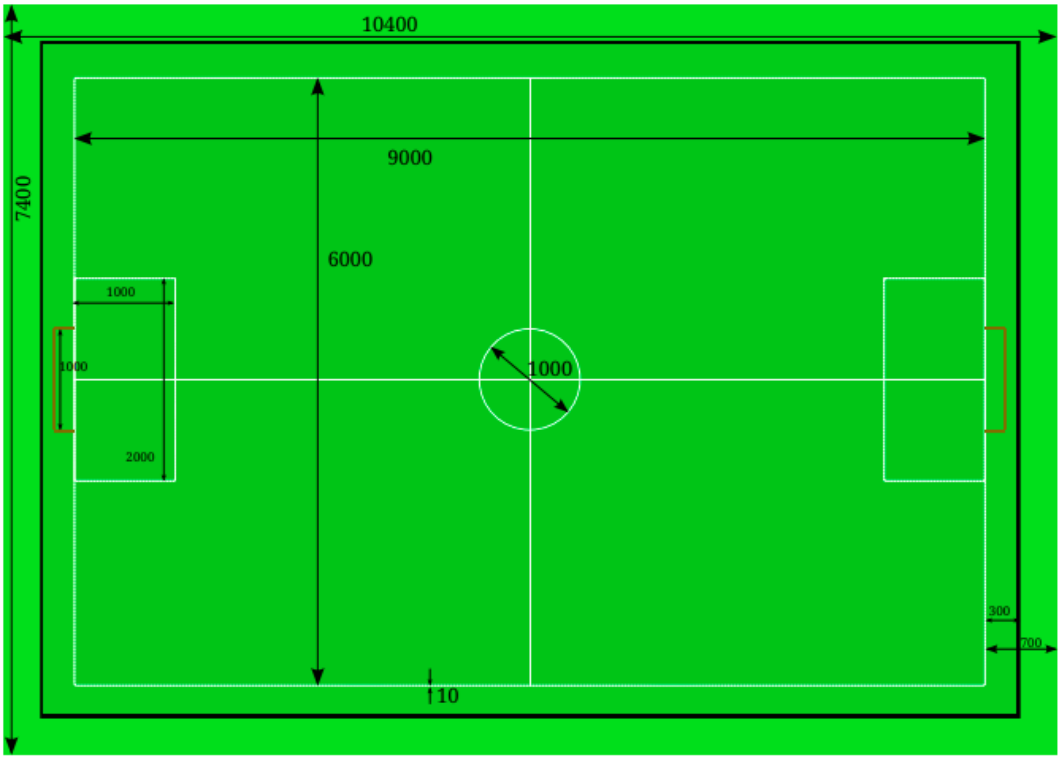


Figura 2.1 – Dimensões do campo da divisão B da SSL, em mm.
Fonte: (Federation, 2023)

2.1.2 Sistema de Visão e Identificação dos Robôs

O sistema de visão *SSL-Vision* utilizado na SSL é unificado e padronizado para todas as equipes, garantindo compatibilidade e justiça competitiva (Zickler *et al.*, 2009). Esse sistema é composto por câmeras de alta resolução posicionadas acima do campo, capturando imagens em tempo real. As imagens são processadas por algoritmos de visão computacional para identificar a posição dos robôs, da bola e de outros elementos relevantes.

Cada robô é equipado com marcadores visuais padronizados que permitem sua identificação única durante as partidas. Esses marcadores consistem em padrões de cores específicos (Figura 2.2). A diferenciação entre equipes é feita pela cor central do marcador, que pode ser azul ou amarela. As equipes devem concordar previamente sobre a escolha das cores, não permitindo que ambos os times utilizem a mesma cor central (Federation, 2023).

2.1.3 *Game Controller*

O *Game Controller* da SSL é um sistema automatizado que atua como árbitro central, gerenciando o fluxo do jogo e garantindo conformidade com as regras. Ele controla estados do jogo (e.g., *kickoff*, *penalty*), inicia e pausa partidas, define o placar e aplica penalidades por infrações (e.g., invasão de área), comunicando-se com as equipes via protocolo de rede UDP. Sua função é padronizar a execução da partida, substituindo a intervenção humana e

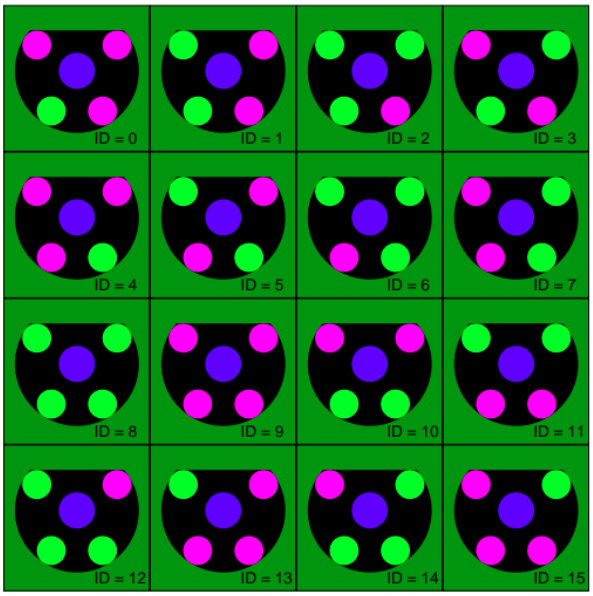


Figura 2.2 – Marcadores visuais dos robôs.
Fonte: (Federation, 2023)

permitindo que os robôs operem de forma totalmente autônoma.

Integrado ao sistema *SSL-Vision*, que rastreia posições de robôs e bola, o *Game Controller* fornece dados em tempo real para as equipes, que devem interpretar seus comandos para ajustar estratégias dinâmicas. Essa integração é essencial para manter a justiça e a sincronia em ambientes competitivos, onde a velocidade e a precisão das decisões tomadas são críticas.

2.1.4 Estados do Jogo e Regras

Os estados do jogo são controlados pelo *Game Controller* e indicam as fases operacionais da partida. Abaixo estão os principais estados do jogo:

Estado	Descrição
Halt	Jogo parado (início ou após infrações graves).
Stop	Jogo pausado (ex.: bola saiu do campo).
Running	Jogo em andamento, com robôs operando livremente.
Kickoff	Reinício no centro do campo após gol ou início de período.
Free Kick	Cobrança de falta (defensores a 0,5 m da bola).
Penalty Kick	Cobrança de pênalti (apenas cobrador e goleiro na área).
Timeout	Pausa solicitada por uma equipe (limitado a 2 por partida).

Tabela 2.1 – Estados do jogo da SSL *RoboCup*.

2.1.5 Resumo das Regras

Cada equipe pode ter até 6 robôs, obedecendo às dimensões máximas de 22 cm de altura e 18 cm. A Figura 2.3 mostra um exemplo de uma categoria de entrada, a SSL-EL da *Robocup* Brasil desenvolvido pela Equipe de Robótica Titans, da Universidade de Brasília.

Figura 2.3 – Robô da liga SSL-EL desenvolvido pela Titans - UnB



Fonte: Autor

A bola deve permanecer em movimento contínuo, e tocar robôs parados resulta em falta. Um gol é validado apenas quando a bola ultrapassa completamente a linha do gol, incluindo situações de chutes diretos ou deflexões. As infrações são categorizadas como colisões violentas, punidas com cartão amarelo ou vermelho, toque manual, que paralisa o jogo imediatamente, ou posicionamento ilegal, com remoção temporária do robô. A partida é dividida em dois períodos de 6 minutos, com intervalo de 3 minutos, e substituições são permitidas apenas durante os estados *Stop* ou *Halt*, realizadas fora da área técnica para evitar interferência no jogo.

2.2 Aprendizado de Máquina

2.2.1 Aprendizado de Máquina e Aprendizado Profundo

O Aprendizado de Máquina (ML, do inglês *Machine Learning*) é um subcampo da inteligência artificial que busca desenvolver algoritmos capazes de aprender padrões a partir de dados (Goodfellow; Bengio; Courville, 2016). Dentro desse campo, o Aprendizado Profundo se destaca por utilizar redes neurais com múltiplas camadas para aprender representações

hierárquicas e abstratas dos dados, sendo especialmente eficaz em tarefas complexas, como reconhecimento de padrões e previsão de comportamentos (LeCun; Bengio; Hinton, 2015).

2.2.2 Regressão Linear

A regressão linear é uma das formas clássicas de algoritmos de aprendizado de máquina supervisionado. O objetivo da regressão linear é construir um sistema linear capaz de receber um vetor de características $\mathbf{x} \in \mathbb{R}^{n\times 1}$ como entrada e predizer o valor de um escalar $y \in \mathbb{R}$ como saída[1]. Seja $\hat{y}$ o valor que o modelo prediz, a equação fundamental é dada por:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} \tag{2.1}$$

onde $\mathbf{w} \in \mathbb{R}^{n\times 1}$ é o vetor de parâmetros ou pesos. Os pesos controlam o comportamento do sistema; cada elemento w_i representa o coeficiente que multiplica a característica x_i. O sinal e a magnitude de w_i determinam, respectivamente, o sentido e a intensidade da influência de x_i sobre $\hat{y}$ (Goodfellow; Bengio; Courville, 2016).

É comum estender este modelo com a adição de um termo de intercepto, ou viés, b. Com isso, o modelo se torna:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b \tag{2.2}$$

Nesta configuração, o mapeamento das características para as predições é uma função afim (Goodfellow; Bengio; Courville, 2016)

O processo de treinamento da regressão linear consiste em determinar os valores ótimos de $\mathbf{w}$ (e b, se aplicável) que minimizam o erro nos dados de treino. A medida de desempenho tipicamente utilizada é o Erro Quadrático Médio (MSE, do inglês *mean square error*) (Goodfellow; Bengio; Courville, 2016).

As soluções analíticas que minimizam o MSE são conhecidas como Equações Normais (Hastie; Tibshirani; Friedman, 2009). Para a matriz de design $\mathbf{X}_{(\text{train})}$ contendo as entradas de treinamento (uma por linha) e o vetor $\mathbf{y}_{(\text{train})}$ das saídas, a solução em forma fechada para o vetor de pesos é dada por:

$$\mathbf{w} = \left(\mathbf{X}_{(\text{train})}^T \mathbf{X}_{(\text{train})}\right)^{-1} \mathbf{X}_{(\text{train})}^T \mathbf{y}_{(\text{train})} \tag{2.3}$$

Esta solução exata para $\mathbf{w}$ é utilizada para avaliar a predição $\hat{y}$ no conjunto de teste.

[1] Neste trabalho, vetores são representados em negrito, como $\mathbf{x}$ e $\mathbf{w}$, enquanto quantidades escalares, como $\hat{y}$ e b, são representadas em letra normal.

2.2.3 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano, compostos por neurônios interconectados organizados em camadas (Haykin, 2007). Cada neurônio recebe entradas, processa-as por meio de uma função de ativação e gera uma saída que é propagada para neurônios subsequentes. A estrutura básica de um RNA inclui:

Camada de Entrada: Responsável por receber os dados brutos do problema. Cada neurônio nesta camada representa uma variável de entrada do modelo, denotada por x_i, onde $i = 1, 2, ..., n$. A quantidade de neurônios nesta camada é determinada pela dimensionalidade dos dados de entrada.

Camadas Ocultas: Realizam transformações não-lineares nos dados, extraindo características relevantes. Cada neurônio recebe um conjunto de entradas ponderadas pelos pesos sinápticos w_{ij} e aplica uma função de ativação para gerar uma saída. O número de camadas ocultas e a quantidade de neurônios em cada uma influenciam a capacidade de aprendizado da rede.

Camada de Saída: Produz a predição final do modelo. A quantidade de neurônios nesta camada depende do tipo de problema: para classificação binária, geralmente há um único neurônio com função de ativação sigmóide; para classificação multiclasse, utiliza-se a função *softmax* com tantos neurônios quantas categorias; para problemas de regressão, a saída pode ser um único valor real sem ativação ou com uma ativação linear.

A escolha da função de ativação é crucial para o desempenho da rede. Funções como *Rectified Linear Unit* (ReLU) e sigmóide são comumente utilizadas para introduzir não-linearidades e permitir que a rede aprenda padrões complexos (Goodfellow; Bengio; Courville, 2016). A Figura 2.4 ilustra a estrutura básica de um neurônio artificial.

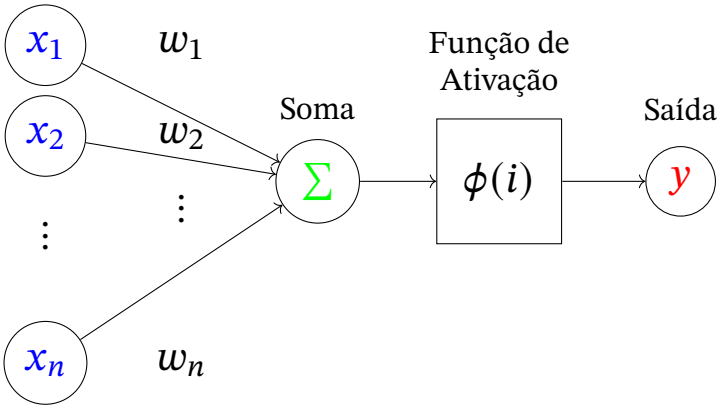


Figura 2.4 – Representação esquemática de um neurônio artificial. Os sinais de entrada $(x_1, \ldots, x_n)$ são ponderados pelos pesos sinápticos $(w_1, \ldots, w_n)$ e agregados na função de soma (Σ). O resultado é submetido a uma função de ativação $(\varphi(i))$, gerando a resposta de saída (y).

2.2.4 LSTM (*Long Short Term Memory*)

As Redes Neurais Recorrentes (RNNs) são arquiteturas projetadas para processar dados sequenciais, que no entanto enfrentam limitações como o desaparecimento do gradiente, que restringe o aprendizado de dependências de longo prazo (Hochreiter, 1998). Para contornar esse problema, as redes LSTM foram desenvolvidas com uma estrutura interna composta por células de memória (*cell state*) e um conjunto de portas lógicas (esquecimento, entrada e saída). Esses mecanismos regulam o fluxo de informações, permitindo que a rede decida seletivamente quais dados devem ser preservados ou descartados ao longo do tempo, garantindo a estabilidade do gradiente durante o treinamento. A Figura 2.5 mostra um exemplo de arquitetura desse tipo.

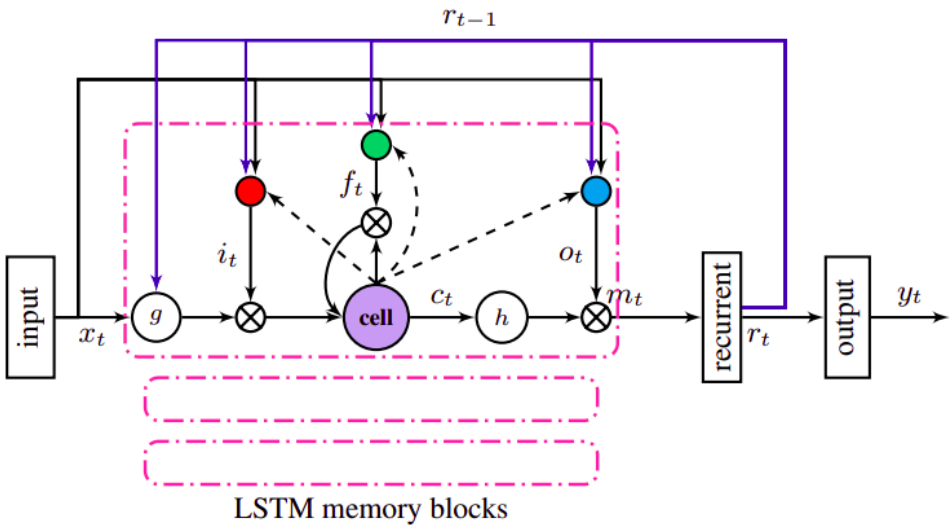


Figura 2.5 – Representação da arquitetura interna de um bloco LSTM. O diagrama ilustra a interação entre a entrada (x_t), o estado recorrente (r_t) e os portões de controle (i_t, f_t, o_t) que modulam o estado da célula (c_t) para preservar dependências temporais.

Fonte: (Sak; Senior; Beaufays, 2014)

2.3 Métricas de Avaliação de Desempenho

A avaliação é feita para verificar a precisão dos modelos de predição. No contexto de previsão de trajetórias, as métricas mensuram a discrepância entre os valores de posicionamento previstos e os reais.

Para as definições a seguir, considera-se:

- n: número total de observações na janela de teste;
- y_i: valor real da variável no instante i;
- $\hat{y}_i$: valor previsto pelo modelo no mesmo instante.

2.3.1 Erro Médio Absoluto

O Erro Médio Absoluto (MAE, do inglês *Mean Absolute Error*) quantifica a média das diferenças absolutas entre as previsões e os valores reais (Equação 2.4).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.4}$$

2.3.2 Erro Quadrático Médio

O Erro Quadrático Médio (MSE, do inglês *Mean Squared Error*) avalia a média dos quadrados das diferenças (Equação 2.5). O MSE é frequentemente utilizado como função de custo no treinamento de redes neurais devido às suas propriedades favoráveis de diferenciabilidade para otimização via gradiente.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

2.3.3 Raiz do Erro Quadrático Médio

A Raiz do Erro Quadrático Médio (RMSE, do inglês *Root Mean Squared Error*) corresponde à raiz quadrada do MSE (Equação 2.6)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.6}$$

2.3.4 Erro Euclidiano Médio

O Erro Euclidiano médio E_{euc} mede a distância linear direta média no espaço 2D entre a posição real (x_i, y_i) e a posição prevista $(\hat{x}_i, \hat{y}_i)$, combinando os erros de ambos os eixos (Equação 2.7).

$$\text{E}_{euc} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2} \tag{2.7}$$

2.4 Interpolação Linear

Em séries temporais provenientes de sistemas de rastreamento, como os sistemas de visão da SSL, a presença de dados ausentes é frequente. Essas ausências podem ocorrer devido a oclusões temporárias, falhas no processo de detecção ou ruídos inerentes ao sistema de aquisição. A inexistência de amostras contínuas pode comprometer etapas posteriores de processamento, como estimação de estado, filtragem ou análise dinâmica do movimento.

A interpolação linear é uma técnica simples e amplamente utilizada para estimar valores ausentes em séries temporais. O método assume que a variação do sinal entre dois instantes consecutivos é aproximadamente linear. Assim, um valor desconhecido $y(t)$ é estimado a partir de dois pontos válidos adjacentes, $P_1 = (t_1, y_1)$ e $P_2 = (t_2, y_2)$, considerando-se que o ponto interpolado pertence à reta que conecta esses dois pontos.

Matematicamente, o valor interpolado é obtido por meio da seguinte Equação:

$$y(t) = y_1 + (y_2 - y_1)\frac{t - t_1}{t_2 - t_1} \tag{2.8}$$

Esse procedimento é equivalente à interpolação linear clássica, também utilizada em métodos numéricos para aproximação de funções, como no método da falsa posição, cuja dedução pode ser encontrada em (Chapra; Canale, 2016).

2.5 Cinemática Discreta

Em sistemas dinâmicos contínuos, a velocidade e a aceleração são definidas como a primeira e a segunda derivada temporal da posição. Entretanto, em séries temporais discretas, nas quais as observações são obtidas em instantes igualmente espaçados, essas derivadas são aproximadas por operadores de diferenças finitas.

Seja $x[n]$ a posição no instante discreto n. A derivada contínua de primeira ordem é aproximada pela diferença finita progressiva:

$$\left.\frac{dx}{dt}\right|_{t=n\Delta t} \approx \frac{x[n] - x[n-1]}{\Delta t}, \tag{2.9}$$

o que define a velocidade discreta ao longo do eixo x:

$$v_x[n] \approx \frac{x[n] - x[n-1]}{\Delta t}. \tag{2.10}$$

De forma análoga, a aceleração corresponde à segunda derivada temporal e, em sua forma discreta, pode ser aproximada pela diferença finita de segunda ordem:

$$\left.\frac{d^2x}{dt^2}\right|_{t=n\Delta t} \approx \frac{x[n] - 2x[n-1] + x[n-2]}{(\Delta t)^2}, \tag{2.11}$$

Uma forma equivalente de expressar a aceleração é por meio da derivada discreta da velocidade:

$$a_x[n] = v_x[n] - v_x[n-1]. \tag{2.12}$$

3 Materiais e Métodos

A metodologia adotada neste trabalho[1] (Figura 3.1) para o desenvolvimento dos modelos de previsão de trajetória foi organizada em cinco etapas:

1. **Aquisição e Conversão dos Dados:** A primeira etapa consistiu na obtenção dos arquivos de registros oficiais da *RoboCup* SSL e na sua conversão do formato original *Protocol Buffers* (ProtoBuf) (Currier, 2022) para o formato CSV, a fim de facilitar a manipulação e a análise subsequente.
2. **Caracterização e Seleção dos Dados:** Com os dados em formato tabular, foi realizada uma análise exploratória para caracterizar cada partida, avaliando métricas como taxa de amostragem e volume de dados ausentes. Com base nessa análise, uma única partida foi selecionada como conjunto de dados base para o desenvolvimento inicial dos modelos.
3. **Pré-processamento, Engenharia de Atributos e Janelamento:** A partida selecionada passou por um pré-processamento, que incluiu o tratamento de valores ausentes por meio de interpolação linear e a escolha de atributos de entradas (*features*). Em seguida, os dados foram estruturados em sequências temporais para a entrada dos modelos LSTM e regressão linear.
4. **Definição e Treinamento dos Modelos:** Nesta fase, foram definidas e treinadas duas arquiteturas distintas: a Regressão Linear e a rede neural LSTM. Ambos os modelos foram treinados com o conjunto de dados de treino, utilizando as configurações específicas para cada arquitetura.
5. **Avaliação dos Modelos:** Por fim, o desempenho dos modelos treinados foi avaliado utilizando o conjunto de dados de teste. As principais métricas de avaliação foram o Erro Euclidiano, Erro Médio Absoluto, Erro Quadrático Médio e Erro Euclidiano Médio.



Figura 3.1 – Fluxo do processo metodológico proposto. O diagrama apresenta, de forma sequencial, as etapas de aquisição dos dados, caracterização, pré-processamento, treinamento dos modelos e avaliação dos resultados.

[1] O repositório *online* com os códigos desenvolvidos neste trabalho encontra-se disponível em https://github.com/luizros/trajectory-prediction.

3.1 Aquisição e Conversão dos Dados

A etapa inicial do trabalho consistiu na aquisição dos dados brutos, que são a base para o treinamento do modelo preditivo. Os dados foram obtidos do repositório público oficial da SSL, com registros de partidas da competição de 2024 (RoboCup SSL Committee, 2024).

Na SSL, todas as equipes utilizam um sistema de visão centralizado e compartilhado, cujo software padrão é o *SSL-Vision* (RoboCup SSL Community, 2024). Este sistema é responsável por capturar imagens do campo através de câmeras suspensas e fornecer os dados de localização de todos os robôs e da bola para as equipes via Ethernet. Conforme estipulado nas regras da competição, os dados sensoriais do mundo real estão sujeitos a imperfeições, como ruído, latência e, crucialmente, falhas ocasionais de detecção. Essa característica inerente ao sistema é a principal causa dos valores ausentes encontrados nos registros, um desafio que será abordado na seção de pré-processamento.

Os registros são armazenados originalmente no formato ProtoBuf, um mecanismo de serialização de dados binários otimizado para comunicação de alta performance. Para a finalidade de análise de dados e treinamento de modelos de ML, foi desenvolvido um *script* em Python para realizar a conversão. O processo envolveu a interpretação dos arquivos de registro, a extração dos dados relevantes de cada quadro (*frame*), como os instantes de captura e as coordenadas (x, y) dos objetos, e, por fim, a sua organização e armazenamento em arquivos no formato *Comma-Separated Values* (CSV).

3.2 Caracterização e Seleção dos Dados

O conjunto de dados utilizado neste trabalho consiste em 11 arquivos de registros, provenientes de partidas oficiais da SSL. Os dados foram obtidos do repositório público da competição, que disponibiliza os registros para fins de pesquisa (RoboCup SSL Committee, 2024). Cada arquivo registra as posições e orientações dos robôs e posições da bola ao longo do tempo, codificadas em formato ProtoBuf. Para compreender as características fundamentais e a qualidade dos dados brutos, foi realizada uma análise exploratória inicial. As métricas avaliadas foram a taxa de amostragem, que indica a frequência de captura dos dados, e a integridade dos dados, medida pela quantidade de valores ausentes.

A análise revelou dois pontos importantes. A taxa de amostragem apresenta uma média de 70 Hz. Essa consistência valida a uniformidade do processo de captura de dados. Em segundo, observa-se que todos os arquivos de registros contêm um volume significativo de dados ausentes, comum em sistemas de visão computacional em tempo real devido a oclusões ou falhas de detecção.

Com base nesta análise, a partida selecionada para os estudos foi a que apresentou o menor número total de células faltantes: 227.813, representando o cenário de dados mais

limpo disponível entre os analisados. As etapas subsequentes de pré-processamento, como a interpolação de dados, serão desenvolvidas e validadas utilizando este arquivo como base.

3.3 Pré-processamento e Engenharia de Atributos

Após a seleção da partida, o conjunto de dados brutos passou por um processo de preparação composto por duas fases principais: tratamento e limpeza dos dados e engenharia de características.

3.3.1 Tratamento e Limpeza dos Dados

Os registros brutos apresentam ruídos de medição e redundâncias. As etapas a seguir descrevem as técnicas aplicadas para garantir a continuidade e suavidade das trajetórias.

3.3.1.1 Filtragem e Seleção de Variáveis

O arquivo original contém metadados irrelevantes para a predição cinética (e.g., geometria do campo). Foi realizada uma filtragem para manter apenas o estado essencial: (x, y, t) para os robôs e (x, y, t) para a bola.

3.3.1.2 Imputação de Dados Ausentes

Falhas no sistema de visão computacional ocasionam perda de detecção em alguns frames. A remoção dessas linhas quebraria a continuidade temporal necessária para a LSTM. Para corrigir isso, utilizou-se a interpolação linear. Dado um valor ausente em t, delimitado por detecções válidas em $t-k_1$ e $t+k_2$, o valor P_t foi estimado assumindo velocidade constante nesse micro-intervalo.

3.3.2 Engenharia de Atributos e Normalização

Apenas as coordenadas posicionais (x, y) são insuficientes para capturar a inércia e a intenção de movimento. Nesta etapa, o vetor de dados foi expandido com propriedades cinemáticas e normalizado.

Para introduzir conceitos físicos explicitamente no modelo, foram calculadas as derivadas temporais via diferenças finitas sobre os dados suavizados:

$$v_x(t) = \frac{x(t) - x(t-1)}{\Delta t} \tag{3.1}$$

$$s = \sqrt{v_x^2(t) + v_y^2(t)} \tag{3.2}$$

$$a_x(t) = \frac{v_x(t) - v_x(t-1)}{\Delta t} \tag{3.3}$$

$$a = \sqrt{a_x^2(t) + a_y^2(t)} \tag{3.4}$$

Assumindo $\Delta t = 1$ (passo de *frame* unitário), obteve-se a velocidade e a aceleração decompostas, permitindo aos modelos detectarem tendências de frenagem ou arranque.

Com base nas variáveis calculadas, construiu-se o vetor de características que representa o estado do jogo em cada instante t. O modelo recebe simultaneamente os dados da bola e de três robôs ($rb_{0..2}$), conforme detalhado na Tabela 3.1.

Tabela 3.1 – Composição do Vetor de Características

Agente	**Variáveis Físicas**	**Dimensão**
Bola	x, y, v_x, v_y, a	5
Robôs Adversários (x3)	x, y, v_x, v_y, s, a	$3 \times 6 = 18$
Total de Features (D_{in})		**23**

Para facilitar a convergência do gradiente descendente, todos os atributos foram escalonados para o intervalo $[0, 1]$ utilizando o método Min-Max, com parâmetros (x_{min}, x_{max}) ajustados estritamente sobre o conjunto de treino para evitar vazamento de dados.

3.3.3 Formulação do Problema e Janelamento

O problema foi modelado como uma tarefa de aprendizado supervisionado em séries temporais. Nessa abordagem, mapeou-se uma sequência histórica de observações N em uma trajetória futura K por meio da estratégia de janelas deslizantes (*sliding window*) para estruturar os tensores de entrada e saída:

- **Tensor de Entrada (X):** Uma matriz de dimensão $(N \times D_{in})$, onde N é o tamanho do histórico e D_{in} é o número de atributos definidos na Figura 3.3.2.
- **Tensor de Saída (Y):** Uma matriz $(K \times 2)$, representando as coordenadas (x, y) para os próximos K passos (horizonte de previsão).

A Figura 3.2 ilustra o processo de fatiamento da série temporal, onde o alvo Y é a continuação imediata da janela X. Ou seja, o modelo atual é treinado e aplicado diretamente durante a duração da mesma partida, permitindo-o aprender padrões de comportamento dos robôs adversários.

O procedimento de janelamento percorre a série temporal para gerar as estruturas de entrada e saída. Sendo X a matriz de entrada e Y a matriz de valores-alvo, o algoritmo

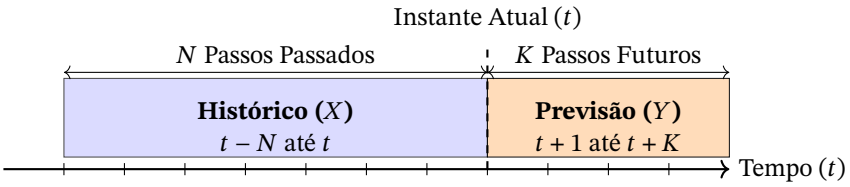


Figura 3.2 – Representação esquemática do janelamento de uma série temporal no instante atual (t) para a previsão multi-passo. O processo utiliza uma Janela Histórica (X), composta pelos N Passos Passados (de $t - N$ a t), como entrada para o modelo. Esta entrada é usada para gerar a Janela de Previsão (Y), que abrange os K Passos Futuros (de $t + 1$ a $t + K$), definindo o horizonte de previsão.

utiliza uma janela deslizante de tamanho N e um horizonte de predição h. Para cada passo temporal t, seleciona-se uma sequência $W_t \subset X$ correspondente ao intervalo $[t - N, t)$. Esta sequência é armazenada em $\mathcal{X}$ preservando sua dimensionalidade temporal. O valor-alvo y_t é extraído de Y no índice deslocado pelo horizonte h (posição $t + h$), sendo adicionado ao conjunto $\mathcal{Y}$. O Algoritmo 3.1 descreve a formação dos pares supervisionados.

Algoritmo 3.1 – Geração de sequências temporais

Dados: Entrada X (comprimento T), Alvos Y, Janela N, Horizonte h
Resultado: Conjuntos supervisionados ($\mathcal{X}$,$\mathcal{Y}$)

$\mathcal{X} \leftarrow \emptyset$
$\mathcal{Y} \leftarrow \emptyset$
para $t \leftarrow N$ ***até*** $T - h$ **faça**
  $W_t \leftarrow X[t - N : t]$
  $y_{\text{alvo}} \leftarrow Y[t + h]$
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{W_t\}$
  $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_{\text{alvo}}\}$
fim
retorna $(\mathcal{X}, \mathcal{Y})$

3.4 Definição e Treinamento dos Modelos

3.4.1 Definição da Arquitetura e Treinamento da Regressão Linear

O primeiro modelo avaliado consistiu em uma abordagem de aprendizado estatístico baseada em regressão. Para cada instante de tempo t dentro da janela de histórico, o vetor de características foi composto pelas coordenadas cartesianas da bola e dos robôs presentes em campo. Conforme detalhado na Tabela 3.2, o modelo recebeu simultaneamente a posição da bola, de três robôs do time azul (rb_0,rb_1,rb_2) e de três robôs do time amarelo (ry_0,ry_1,ry_2), totalizando 14 variáveis por quadro.

Antes da linearização, todos os atributos posicionais foram escalonados para o intervalo

Tabela 3.2 – Composição do Vetor de Características por Instante (t)

Agentes	**Variáveis (x, y)**	**Qtd. Features**
Bola	$ball_x, ball_y$	2
Robôs Time Azul (0, 1, 2)	rb_x, rb_y	6
Robôs Time Amarelo (0, 1, 2)	ry_x, ry_y	6
Total de Features por Frame (F)		**14**

$[0, 1]$, de modo a garantir que a penalidade de regularização atue uniformemente sobre todos os pesos, evitando que a escala numérica das coordenadas em milímetros enviese a otimização.

Para adequar os dados, realizou-se um processo de linearização da janela temporal. Dado uma janela de histórico de tamanho p com F variáveis por instante, o vetor de entrada x_{in} foi construído pela concatenação sequencial dos últimos p frames:

$$x_{in} = [f_{t-p}, f_{t-p+1}, \ldots, f_{t-1}] \tag{3.5}$$

O Algoritmo 3.2 detalha o procedimento exato para a criação dos pares supervisionados $(\mathcal{X}, \mathcal{Y})$, considerando a janela de histórico p.

Algoritmo 3.2 – Geração de Janelas para Regressão Linear

Dados: Sinal normalizado $S \in \mathbb{R}^{T \times d}$, janela de predição p
Resultado: Conjuntos supervisionados $(\mathcal{X},\mathcal{Y})$

$\mathcal{X} \leftarrow \emptyset$; $\mathcal{Y} \leftarrow \emptyset$;
para $t \leftarrow 0$ ***até*** $T - p - 1$ **faça**
    $W_t \leftarrow S[t : t + p]$ `// Janela deslizante de entrada`
    $x_t \leftarrow \text{flatten}(W_t)$ `// Achata para vetor 1D`
    $y_t \leftarrow S[t + p]$ `// Próximo estado futuro`
    `// Armazena nos conjuntos finais`
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_t\}$
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_t\}$
fim

A predição para o horizonte futuro h é dada pela combinação linear:

$$\hat{y}_{t+1} = \mathbf{W}^T x_{in} + \mathbf{b} \tag{3.6}$$

onde $\mathbf{W}$ é a matriz de pesos e $\mathbf{b}$ é o viés.

3.4.2 Cenários de Teste para a Regressão Linear

A avaliação foi conduzida através de uma busca em grade combinada com a validação deslizante descrita acima. O objetivo foi identificar a sensibilidade do modelo a três

fatores críticos: o histórico disponível, a distância temporal da predição e a frequência de re-treinamento.

Os experimentos cobriram as permutações dos seguintes hiperparâmetros:

Janela de Predição (p): $\{10, 15, 20, 25, 30\}$ amostras. Representa quantos instantes futuros o modelo tenta prever simultaneamente.

Janela de Treinamento (W_{train}): $\{10, 15, 30, 60\}$ segundos. Define o tamanho do histórico utilizado para ajustar o modelo.

Janela de Teste (W_{test}): 1 segundo. Intervalo de tempo em que o modelo permanece fixo antes de ser re-treinado.

A Tabela 3.3 resume a configuração da grade, totalizando 60 cenários experimentais distintos.

Tabela 3.3 – Grade de Hiperparâmetros para os Experimentos de Regressão

Parâmetro	**Símbolo**	**Valores Testados**
Janela de Predição	p	$\{10, 15, 20, 25, 30\}$ *amostras*
Janela de Treinamento	W_{train}	$\{10, 15, 30, 60\}$ *segundos*
Janela de Teste	W_{test}	$\{1\}$ *segundo*
Total de Configurações		$5 \times 4 \times 1 = 20$ cenários

3.4.3 Definição da Arquitetura e Treinamento da LSTM

O segundo modelo testado foi a LSTM, devido à sua capacidade de mitigar o problema de entender sequências temporais. A topologia da rede foi mantida constante para permitir a comparação justa entre diferentes tamanhos de janela de entrada. O fluxo de dados através do modelo é definido por:

1. **Camada de Entrada:** Recebe tensores de dimensão (N, D_{in}), onde N é o tamanho da janela histórica variável e $D_{in} = 23$ é o número de *features*.
2. **Camada LSTM Oculta:** Uma única camada recorrente composta por 50 unidades de memória. Utilizou-se a função de ativação tangente hiperbólica tanh para regular o fluxo de informações entre os estados ocultos.
3. **Camada de Saída (Densa):** Uma camada totalmente conectada (*Dense*) com 2 neurônios e ativação linear, projetando o estado final da LSTM nas coordenadas preditas $(\hat{x}_{t+1}, \hat{y}_{t+1})$.

3.4.4 Cenários de Teste para a LSTM

Foram definidos 6 cenários distintos de janelamento:

Curto Prazo: $N \in \{5, 10, 15\}$ *frames* (aprox. 0.07s a 0.21s). Focam na dinâmica instantânea.

Médio/Longo Prazo: $N \in \{30, 45, 60\}$ *frames* (aprox. 0.42s a 0.85s). Buscam capturar trajetórias mais complexas e inércia.

Para todos os cenários, utilizou-se um subconjunto de 50.000 frames da partida selecionada, garantindo que os modelos fossem treinados e validados exatamente sobre os mesmos trechos de jogo. A divisão dos dados respeitou a proporção de 70% para treinamento e 30% para teste, sem embaralhamento. A Tabela 3.4 resume as configurações de treinamento.

Tabela 3.4 – Resumo dos Hiperparâmetros de Treinamento Ótimos

Parâmetro	**Configuração**
Otimizador	Adam
Função de Perda (*Loss*)	MSE
Métrica	MAE
Batch Size	64 amostras
Épocas	100
Unidades LSTM	50
Janelas (N) testadas	5, 10, 15, 30, 45, 60
Horizonte (h)	1 quadro

3.4.5 Estratégia de Validação

Ao contrário da validação cruzada tradicional (*k-fold*), que embaralha as amostras, esta metodologia preserva a ordem temporal dos dados, refletindo um cenário real de operação em que o modelo é periodicamente re-treinado. O processo depende dos tamanhos das janelas de treino ($\mathbf{T_r}$) e teste ($\mathbf{T_s}$), além da janela de predição p utilizada na construção das sequências.

1. O modelo é treinado exclusivamente no intervalo temporal $[t,\ t + \mathbf{T_r}]$ utilizando as sequências construídas com uma janela de entrada de tamanho p;
2. A inferência é realizada no bloco imediatamente subsequente $[t + \mathbf{T_r},\ t + \mathbf{T_r} + \mathbf{T_s}]$;
3. Após essa etapa, a janela deslizante é avançada no tempo por $\mathbf{T_r} + \mathbf{T_s} + \mathbf{p}$, repetindo-se o ciclo até o final do registro da partida.

3.5 Métricas de Avaliação

Para quantificar a qualidade dos modelos propostos, adotou-se uma abordagem híbrida que considera tanto a fidelidade da trajetória prevista quanto a viabilidade de execução em tempo real. Todas as métricas de erro posicional foram calculadas sobre os dados desnormalizados, ou seja, representam distâncias reais em milímetros no campo de jogo.

3.5.1 Métricas de Precisão

A avaliação da discrepância entre a posição real do agente (p_t) e a posição estimada pelo modelo ($\hat{p}_t$) utilizou três indicadores principais previamente definidos no referencial teórico, erro euclidiano, RMSE e MAE.

3.5.2 Avaliação dos Modelos

Foram monitorados o tempo de treinamento e predição e o tempo de inferência dos modelos.

- **Tempo de Inferência (τ_{inf}):** O tempo médio necessário para o modelo processar uma única janela de entrada e gerar a predição.
- **Tempo de Treinamento (τ_{train}):** O tempo necessário para ajustar os pesos do modelo em um bloco de dados.

3.6 Ambiente Computacional

A implementação do projeto e o treinamento dos modelos foram realizados seguindo o ambiente de desenvolvimento detalhado na Tabela 3.5.

Tabela 3.5 – Ambiente de Desenvolvimento e Ferramentas

Categoria	**Especificação**
Linguagem de Programação	Python (versão 3.x)
Modelo do Computador	DELL 3240
Processador	Intel Core i7 (11ª Geração)
Memória RAM	16 GB
Framework de *Deep Learning*	TensorFlow / Keras
Biblioteca de *Machine Learning*	Scikit-Learn
Biblioteca de Processamento de Dados	Pandas e NumPy

4 Resultados

Neste capítulo, são apresentados os resultados obtidos a partir da metodologia descrita no Capítulo 3. São detalhados o desempenho de cada modelo avaliado, bem como as configurações que produziram os melhores resultados nos diferentes cenários experimentais.

4.1 Modelo de Regressão Linear

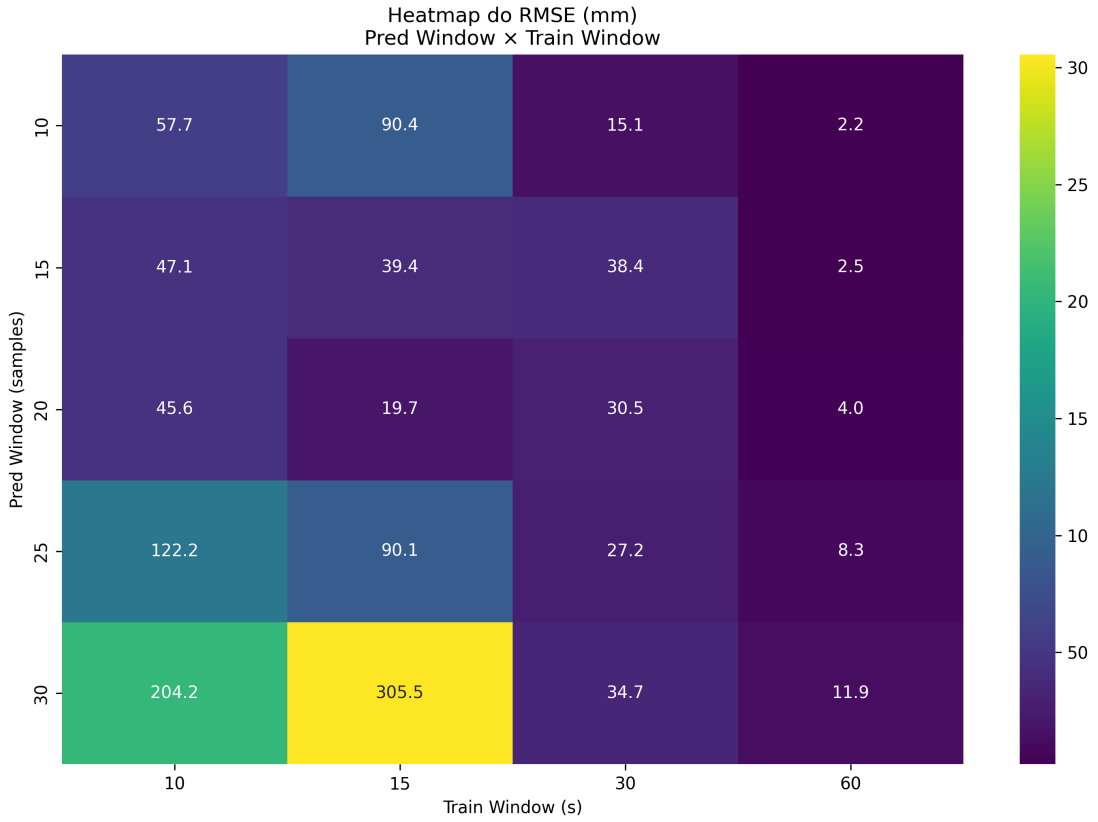


Figura 4.1 – Mapa de calor com a Raiz do Erro Quadrático Médio (RMSE, em mm) do modelo de previsão sob diferentes configurações de janelamento. O eixo Y é a Janela de Previsão (Pred Window, em amostras), e o eixo X é a Janela de Treino (Train Window, em segundos). O mapeamento de cores demonstra que o melhor desempenho (menor RMSE, cores escuras) é atingido com Janelas de Previsão menores e Janelas de Treino maiores, indicando que um histórico de entrada mais longo é vantajoso.

Foram conduzidos experimentos para a tarefa de predição de um passo à frente dentro de intervalos de teste de 1 segundo, variando dois hiperparâmetros: o tamanho da janela de observação histórica (10, 15, 30 e 60 s) e o tamanho do conjunto de dados utilizado para o ajuste do modelo (10, 15, 20, 25 e 30 amostras).

Tabela 4.1 – Resultados ótimos obtidos para o modelo de regressão linear

Janela de obs. (amostras)	Janela de treino (s)	RMSE (mm)
10	60	2,23
15	60	2,46
20	60	3,98
25	60	8,29
30	60	11,86

A Tabela 4.1 apresenta os melhores resultados encontrados para cada tamanho de janela de observação. Em todos os casos, a janela de treino ótima foi de 60 s, mostrando que o modelo linear se beneficia de uma maior quantidade de dados para treino. No entanto, o modelo não se beneficia necessariamente de um histórico de observação maior, dado que os valores RMSE aumentaram com o aumento da janela de observação.

A Figura 4.1 apresenta um mapa de calor (*heatmap*) do RMSE em função dos tamanhos das janelas de observação e de treinamento. Observa-se uma região de estabilidade quando a janela de treinamento é superior a 30 segundos. As janelas de predição menores (10 e 15 amostras) apresentaram desempenho superior às janelas maiores.

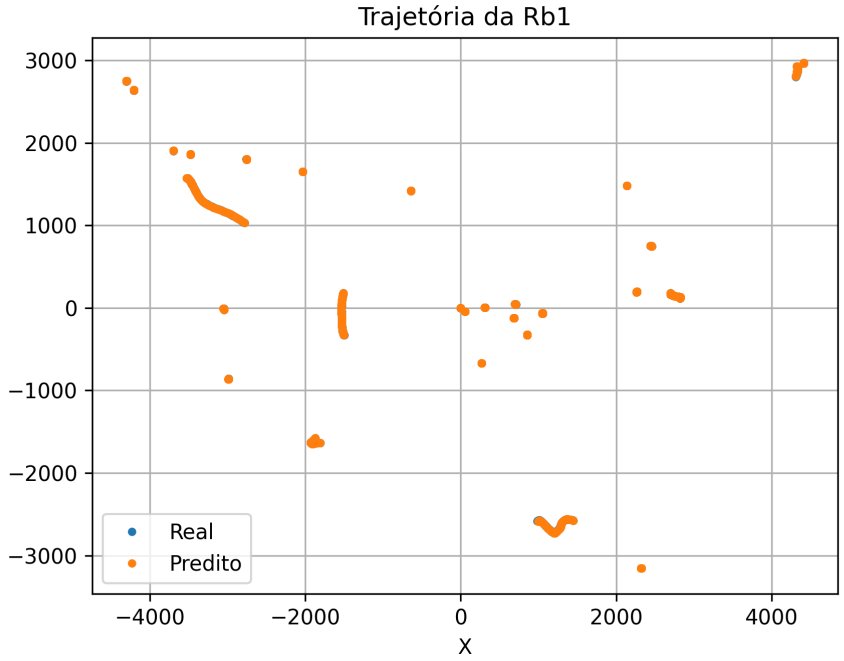


Figura 4.2 – Comparação espacial entre as posições reais (azul) e as estimadas (laranja) pelo Regressor Linear. A alta densidade de sobreposição entre os pontos evidencia a precisão do modelo em intervalos curtos de tempo.

O desempenho do regressor linear mostrou-se proporcional ao tempo de dados disponíveis para treino. Nota-se na Figura 4.1 uma queda abrupta no erro ao passar de 15 para 30 segundos de histórico de treinamento, seguida de melhoras significativas de 30 para 60

segundos. Com janelas de treino de 60 segundos, atinge-se o menor erro (2,23 mm) com a configuração mais simples de entrada.

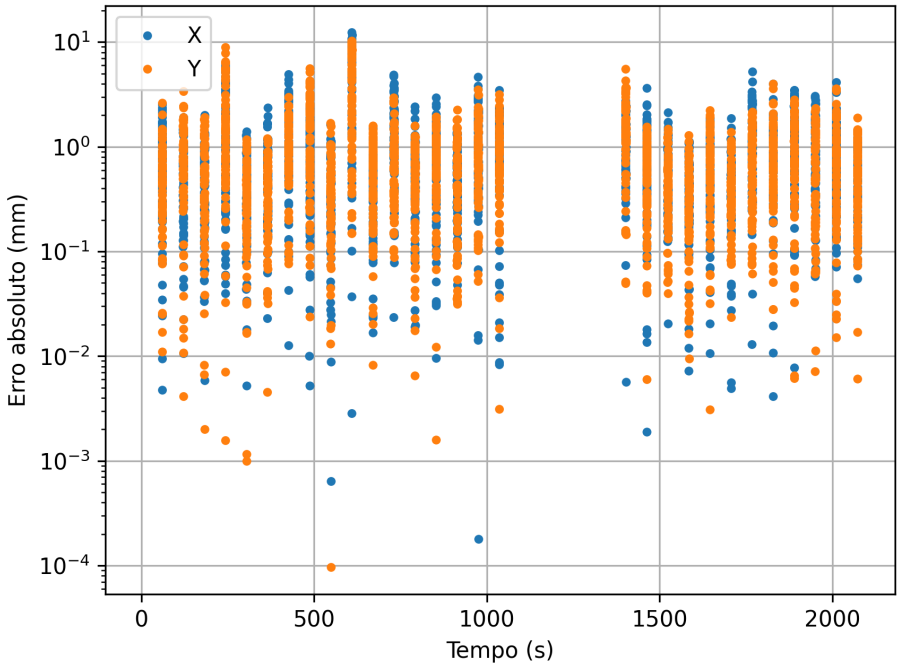


Figura 4.3 – Evolução do erro absoluto de previsão para as coordenadas X (azul) e Y (laranja) ao longo do tempo de experimento. O eixo vertical em escala logarítmica permite a visualização de erros que variam em diferentes ordens de grandeza.

Ao aumentar a quantidade de dados para predição, o modelo perde desempenho, conforme observado na Figura 4.1. O menor erro (2,23 mm) foi obtido utilizando apenas as últimas 10 amostras (aproximadamente 160 ms de histórico). Ao aumentar a janela para 30 amostras, o número de variáveis de entrada do modelo cresce significativamente, o que, para uma regressão linear simples, provavelmente introduziu ruído e dificultou a generalização, elevando o erro para 11,86 mm, mesmo com muitos dados de treino.

A Figura 4.2 apresenta a comparação visual entre a trajetória real e a predita para o melhor regressor linear dos testes. Dada a natureza da predição de curto prazo (aproximadamente 16ms), as trajetórias são muito parecidas.

A Figura 4.3 apresenta o erro absoluto em X e em Y para as diversas predições ao longo da partida avaliada. Calculando o erro euclidiano de cada predição e levantando um histograma (Figura 4.4), a distribuição correspondente apresenta um comportamento de cauda longa, concentrando a maior parte dos erros abaixo de 5 mm.

As Figuras 4.5 e 4.6 apresentam os tempos de inferência e de treinamento do regressor linear para cada janela de previsão, respectivamente. Em ambas as Figuras, cada curva corresponde a uma janela de treinamento. A Figura 4.7, por sua vez, apresenta a distribuição

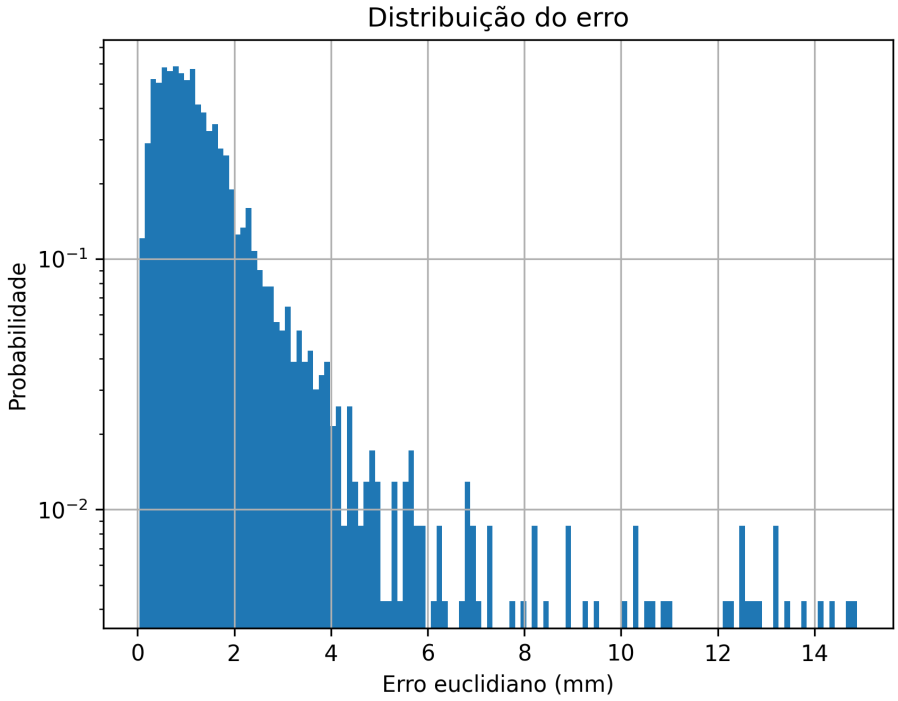


Figura 4.4 – Histograma de distribuição de probabilidade do erro euclidiano. O gráfico evidencia um comportamento de cauda longa, indicando que a grande maioria das previsões apresenta desvios inferiores a 5 mm.

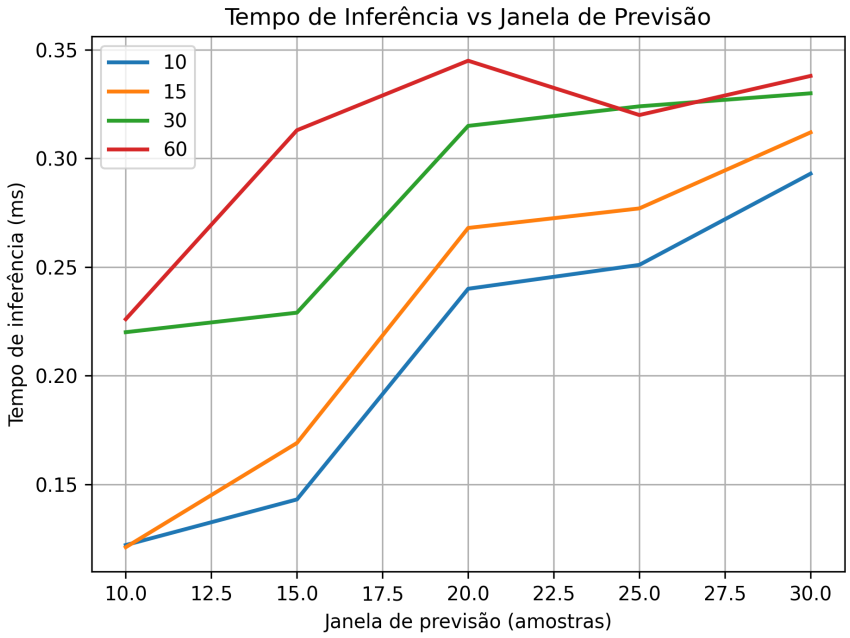


Figura 4.5 – Tempo de inferência vs. janela de previsão para o regressor linear. Cada curva corresponde a uma janela de treinamento.

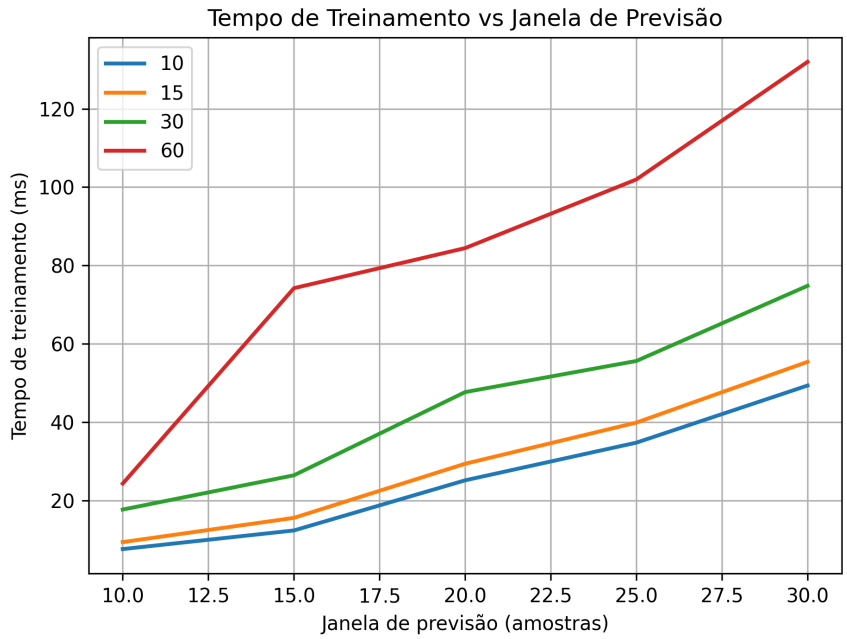


Figura 4.6 – Tempo de treinamento vs. janela de previsão para o regressor linear. Cada curva corresponde a uma janela de treinamento.

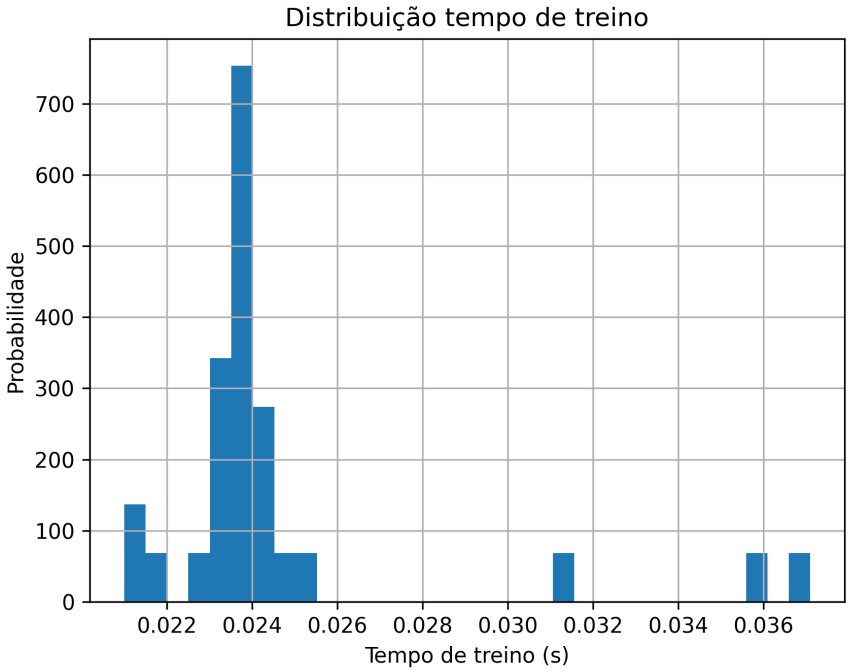


Figura 4.7 – Distribuição de frequências do tempo de treinamento para o regressor linear. O histograma demonstra que a maioria das execuções se concentra em torno de 0,024 segundos (24 ms), indicando estabilidade computacional e baixo custo de processamento para o modelo de regressão.

do tempo de treinamento. O tempo de inferência manteve-se consistentemente baixo, com uma média de 0,22 ms, o que é quase duas ordens de grandeza inferior ao período de amostragem do sistema de 16,6 ms. O treinamento completo com 60 segundos de dados leva cerca de 130 ms, permitindo re-treinamentos periódicos em *background* sem travar o *loop* de controle principal.

4.2 Modelo LSTM

Nos cenários de teste definidos no Capítulo 3 para o modelo LSTM, utilizou-se inicialmente um vetor de entrada composto por 23 variáveis, incluindo derivadas temporais. Contudo, essa alta dimensionalidade introduziu ruído excessivo, resultando em instabilidade e dificuldade de convergência do modelo. Diante disso, optou-se por reduzir a dimensionalidade dos dados e considerar apenas o posicionamento do agente como entrada. Essa simplificação resultou em uma redução significativa nos erros de predição.

Tabela 4.2 – Resultados dos Experimentos de Predição com o Modelo LSTM.

Jan. obs. (amostras)	RMSE (mm)	MAE X (mm)	MAE Y (mm)	Tempo tr. (s)	Tempo teste (s)
5	4,766	4,643	1,987	137,490	8,543
10	2,579	0,994	1,371	181,083	0,917
15	3,951	1,507	1,954	146,650	8,760
30	7,904	3,831	2,105	185,440	1,689
45	3,920	1,856	1,335	589,350	2,135
60	1,645	1,164	0,5226	709,780	2,258

Nesta seção, analisa-se o desempenho quantitativo e qualitativo da melhor rede LSTM obtida. A seleção dos hiperparâmetros baseou-se em uma análise de sensibilidade do tamanho da janela histórica (N), apresentada na Tabela 4.2. O cenário com janela de $N = 60$ frames apresentou o melhor resultado, obtendo os menores índices de erro (RMSE de 1,645 mm). Dessa forma, as análises visuais e detalhadas a seguir concentram-se nesta configuração.

A Figura 4.8 apresenta as curvas de aprendizado (*loss* e MAE) para o treinamento e a validação do modelo LSTM. Esta Figura demonstra que tanto a função de perda quanto a métrica MAE apresentaram decaimento acentuado nas primeiras épocas, estabilizando-se posteriormente.

A análise da série temporal dos erros revela o comportamento dinâmico do modelo frente às variações de trajetória. A Figura 4.9 discrimina o erro absoluto entre as coordenadas X e Y. Apesar do erro médio reduzido, identificam-se transientes de instabilidade no intervalo entre as amostras 6500 e 7500. Nota-se também uma assimetria nestes eventos: a coordenada X apresenta picos de magnitude superior à coordenada y, o que sugere uma sen-

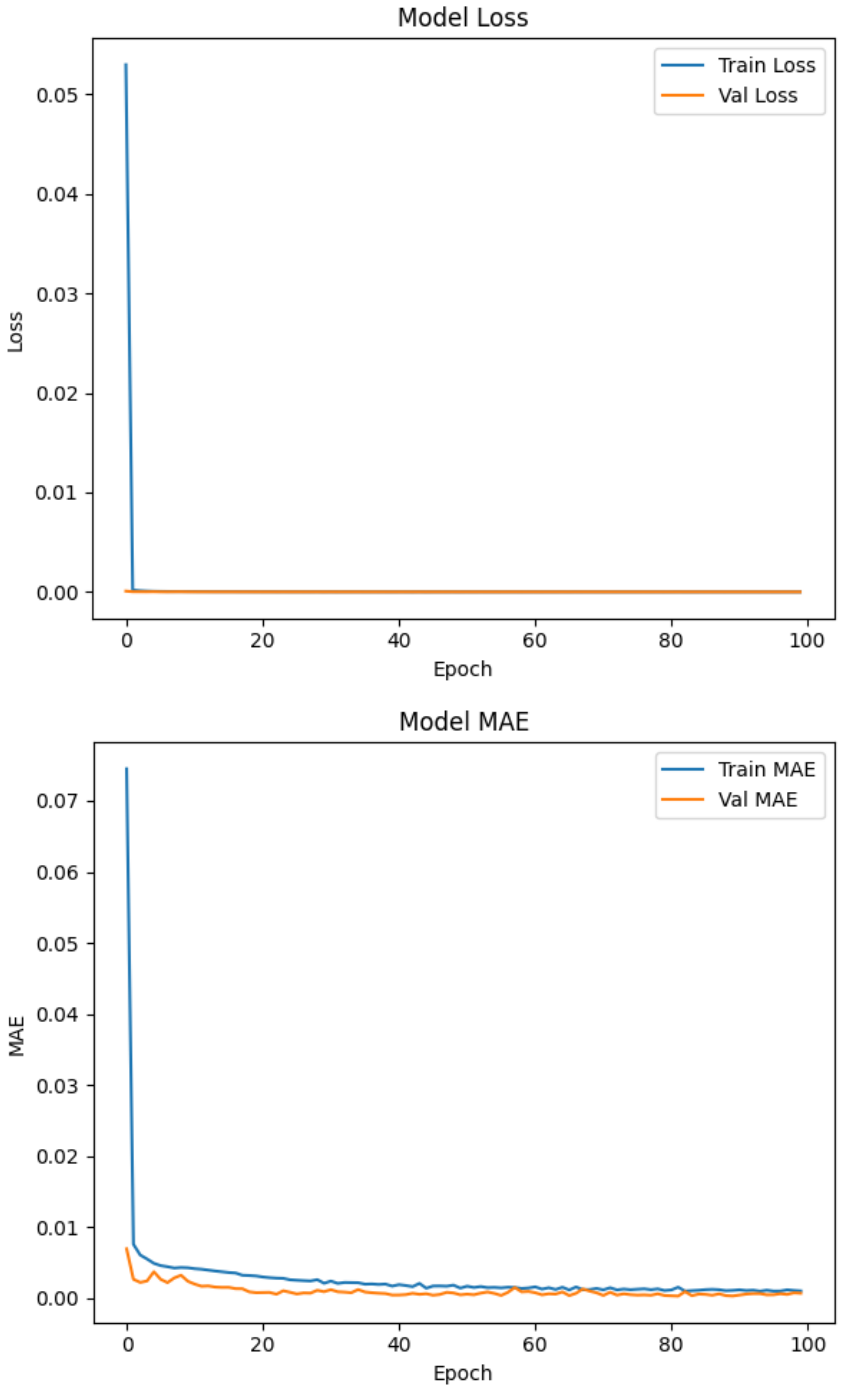


Figura 4.8 – Evolução da função de perda (*loss*) e da métrica MAE durante 100 épocas. Observa-se uma convergência rápida nas iterações iniciais e a ausência de distanciamento entre as curvas de treinamento e validação, indicando estabilidade e ausência de overfitting.

sibilidade maior do modelo a movimentos bruscos ou ruídos específicos no eixo horizontal do campo. Qualitativamente, a Figura 4.10 ilustra fidelidade de rastreamento. A trajetória predita (vermelho) adere ao caminho real (azul), antecipando corretamente curvas suaves e segmentos retilíneos.

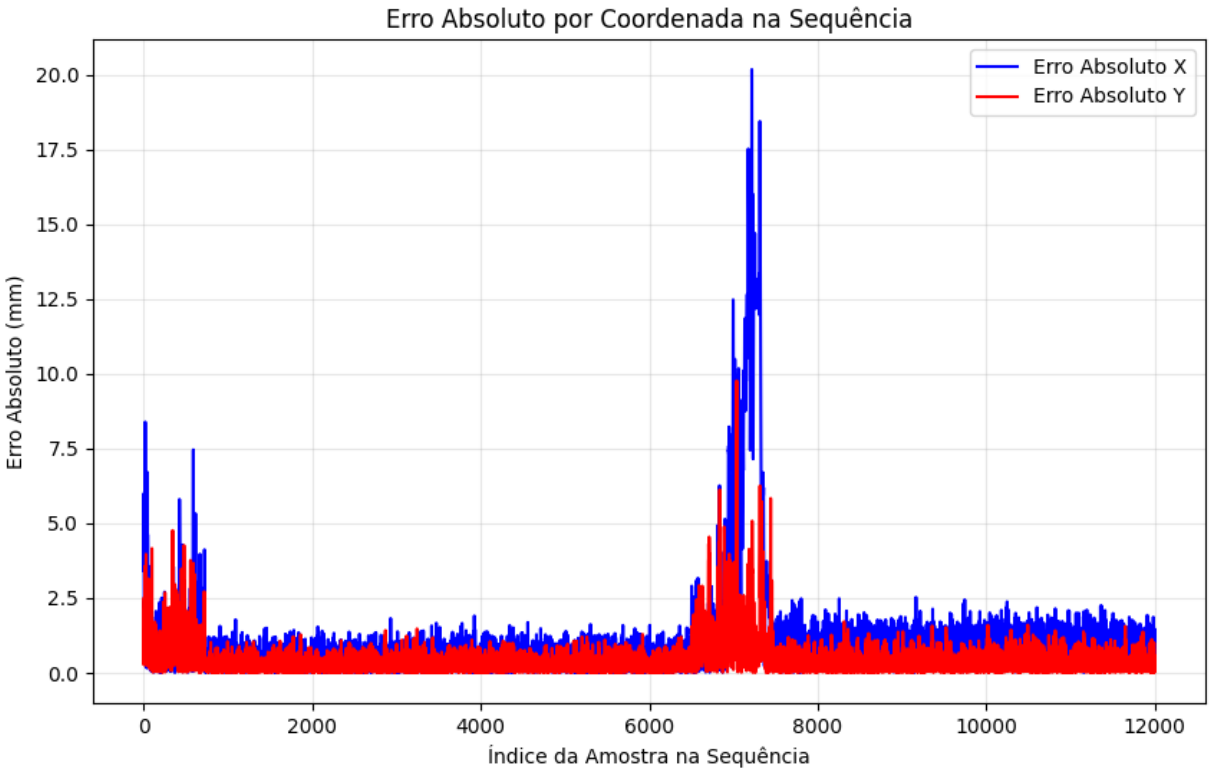


Figura 4.9 – O gráfico apresenta a evolução do Erro Absoluto (em mm) para as coordenadas X (azul) e Y (vermelha) ao longo da sequência de teste. Observa-se que o modelo LSTM mantém estabilidade na maior parte da inferência, com picos de erro localizados (notadamente próximo à amostra 7.000), indicando variações pontuais na precisão da predição em relação aos dados reais.

As Figuras 4.11 e 4.12 apresentam tempos de treinamento e inferência, respectivamente, para o modelo LSTM com diferentes janelas de treinamento. Diferentemente do regressor linear, pode-se ver que o modelo LSTM teve tempos bem mais lentos do que o período de amostragem, não permitindo re-treinamentos periódicos em *background* sem travar o *loop* de controle principal, pelo menos com o *hardware* usado nestes testes.

4.3 Comparação Entre Modelos

Na análise comparativa entre os modelos, é possível afirmar que a Regressão Linear se destacou pelo baixo custo computacional e pela rápida predição, enquanto a rede LSTM demonstrou eficiência na captura de dinâmicas não-lineares, porém apresentando baixa eficiência computacional. Contudo, o estudo enfrentou limitações, principalmente a restrição ao processamento em CPU, que impossibilitou uma busca exaustiva por hiperparâmetros

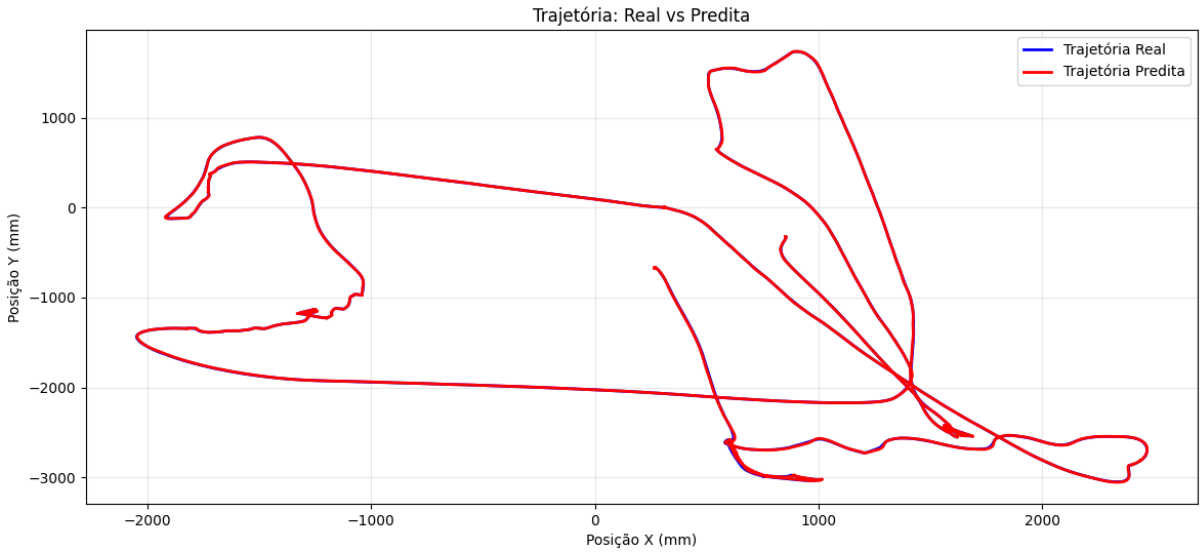


Figura 4.10 – Comparação entre a trajetória real e a estimada do robô no plano (XY). A curva (azul) representa os dados reais do conjunto de teste, enquanto a (vermelha) indica as posições preditas pelo modelo LSTM otimizado.

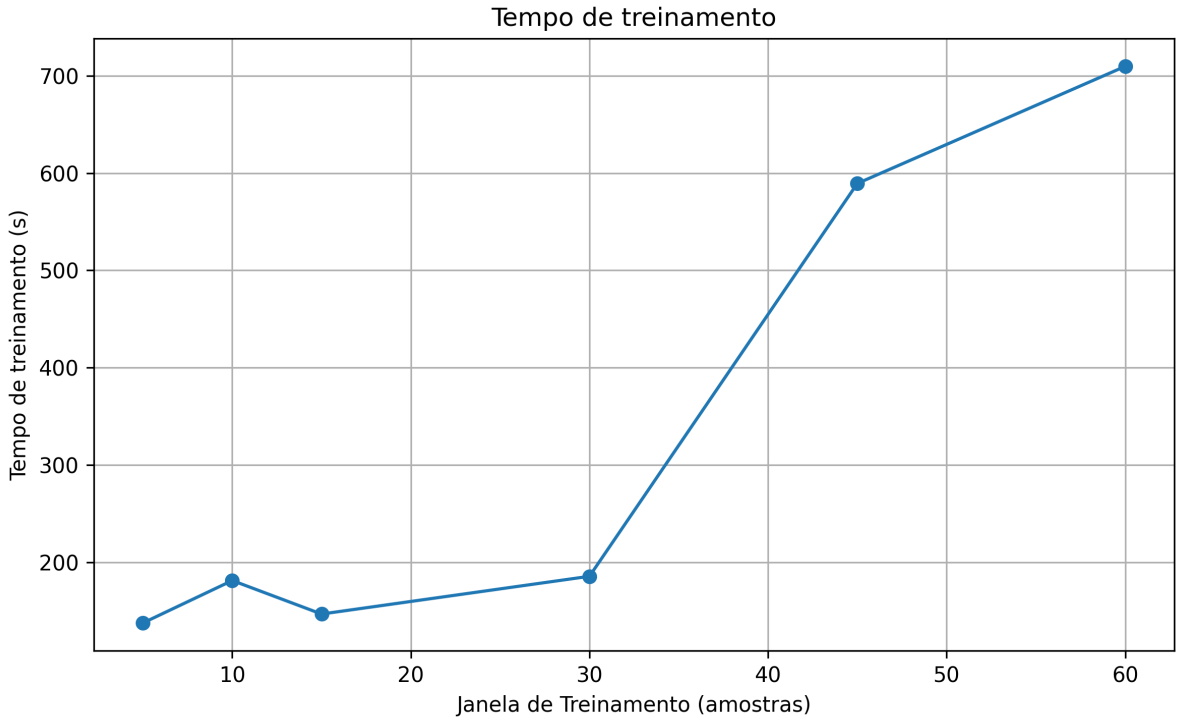


Figura 4.11 – Relação entre o tempo total de treinamento (eixo Y) e o tamanho da janela de dados (eixo X). Observa-se uma tendência geral de crescimento do tempo conforme a janela aumenta, indicando que sequências temporais mais longas demandam maior processamento para a atualização dos pesos da rede LSTM.

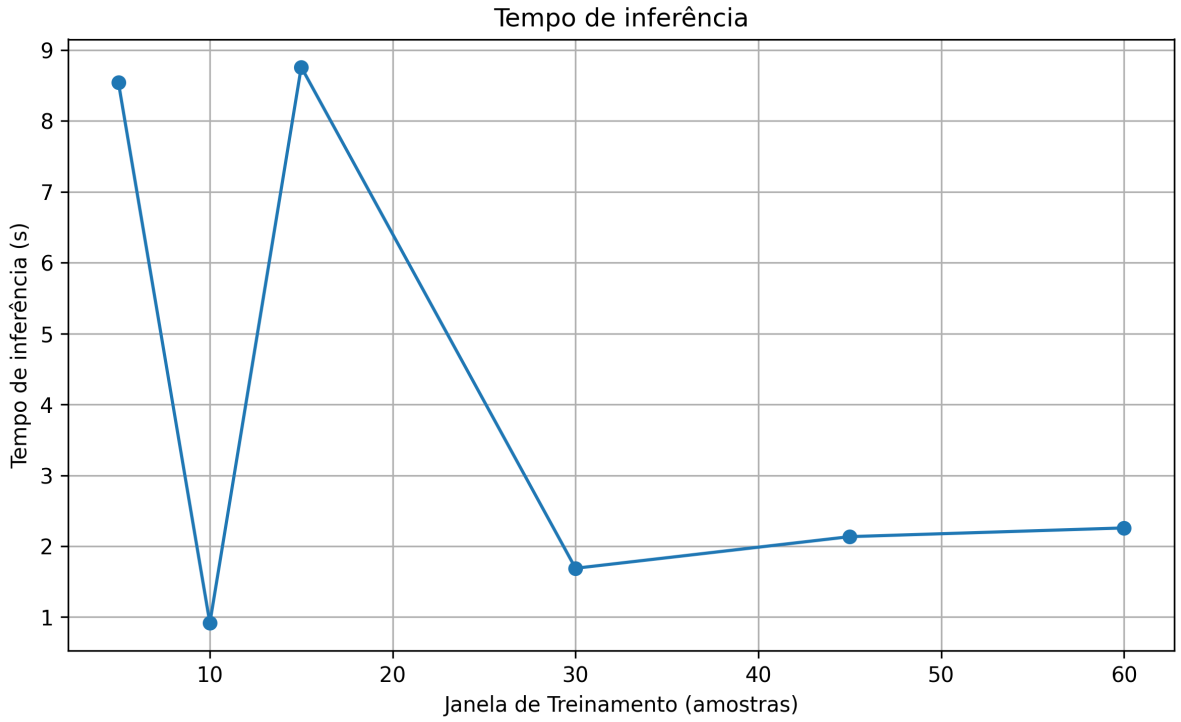


Figura 4.12 – Medição do tempo de inferência do modelo para diferentes janelas. Ao contrário do treinamento, o tempo de resposta apresenta oscilações significativas (picos e vales) que não seguem uma linearidade estrita, sugerindo que a latência de inferência pode ser influenciada por fatores de *overhead* do sistema além do tamanho da entrada.

adequados que pudessem otimizar e melhorar os resultados da LSTM, além de limitar a complexidade das arquiteturas testadas. Ademais, a representação dos dados de entrada para a LSTM, restrita a coordenadas espaciais, simplificou a modelagem, mas desconsiderou atributos cinemáticos e contextuais relevantes.

5 Conclusões

Este trabalho avaliou o uso de modelos de Aprendizado de Máquina para a predição de trajetórias de robôs futebolísticos em um cenário de previsão de um passo à frente, a partir de séries temporais de posicionamento. Foram implementados e analisados dois modelos distintos: um regressor linear e uma rede neural recorrente do tipo LSTM, os quais foram avaliados sob diferentes configurações de janelas históricas e volumes de dados, de modo a identificar, para cada modelo, as condições que resultaram no melhor desempenho na tarefa de previsão a um passo à frente.

Os resultados demonstraram que a regressão linear apresenta desempenho consistente para previsões de curto prazo, com baixo erro e custo computacional reduzido, sendo adequada para aplicações com restrições de tempo real. A LSTM, por sua vez, obteve menor erro de predição no melhor cenário avaliado, indicando maior capacidade de modelar dinâmicas não lineares, porém com custo computacional significativamente superior, o que limita sua aplicação em sistemas com necessidade de re-treinamento frequente ou inferência em tempo real sob restrições de hardware.

As análises evidenciaram que o desempenho de ambos os modelos é fortemente influenciado pelo tamanho da janela histórica utilizada como entrada, bem como pela quantidade de dados disponíveis para treinamento. Observou-se que janelas de observação maiores nem sempre resultam em melhor desempenho, especialmente para o modelo linear, onde o aumento da dimensionalidade tende a degradar a capacidade de generalização.

Apesar dos resultados obtidos, o estudo apresenta limitações importantes. A avaliação foi restrita à predição de um único passo à frente, o que não permite caracterizar o comportamento dos modelos em horizontes temporais mais longos. Além disso, a busca por hiperparâmetros foi limitada, particularmente no caso da LSTM, devido à execução exclusivamente em CPU. A base de dados utilizada também foi restrita a um conjunto reduzido de partidas, e a representação dos dados concentrou-se majoritariamente em coordenadas espaciais, com atributos cinemáticos e contextuais explorados de forma limitada.

Como trabalhos futuros, propõe-se a ampliação do horizonte de predição para múltiplos passos à frente, com o objetivo de analisar até que ponto os modelos permanecem estáveis e identificar o momento em que as predições passam a divergir significativamente da trajetória real. Essa análise permitirá comparar de forma mais precisa a degradação temporal da regressão linear e da LSTM. Recomenda-se também a adoção de métodos automáticos de otimização de hiperparâmetros, visando melhorar o desempenho e a estabilidade dos modelos, especialmente das arquiteturas recorrentes. A inclusão de dados provenientes de múltiplas partidas, bem como a expansão do conjunto de atributos de entrada com variáveis cinemáticas e informações contextuais do jogo, constitui outro caminho relevante

para aumentar a capacidade de generalização. Por fim, a investigação de arquiteturas mais complexas ou híbridas pode contribuir para um melhor compromisso entre desempenho preditivo e custo computacional.

Referências

CHAPRA, S. C.; CANALE, R. P. **Métodos Numéricos para Engenharia**. 7. ed. [*S.l.*]: McGraw-Hill, 2016. Citado na p. 25.

CURRIER, C. Protocol buffers. *In*: ____. **Mobile Forensics – The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices**. Cham: Springer International Publishing, 2022. p. 223–260. ISBN 978-3-030-98467-0. DOI 10.1007/978-3-030-98467-0$_9$. Disponível em: https://doi.org/10.1007/978-3-030-98467-0_9. Citado na p. 26.

FEDERATION, R. **RoboCup Small Size League Rules**. 2023. Disponível em: https://robocup-ssl.github.io/ssl-rules/. Citado nas pp. 17, 18 e 19.

FEDERATION, R. **RoboCup Official Website**. 2025. Accessed: 2025-01-26. Disponível em: https://www.robocup.org/objective. Citado na p. 14.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.*]: MIT Press, 2016. Citado nas pp. 20, 21 e 22.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2nd. ed. New York, USA: Springer, 2009. ,p.104. ISBN 978-0-387-84857-0. Citado na p. 21.

HAYKIN, S. **Redes Neurais: Princípios e Práticas**. [s.l.]: Bookman Editora, 2007. ISBN 978-8577800126. Citado na p. 22.

HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, v. 06, n. 02, p. 107–116, 1998. DOI 10.1142/S0218488598000094. Disponível em: https://doi.org/10.1142/S0218488598000094. Citado na p. 23.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na p. 15.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, I.; OSAWA, E.; MATSUBARA, H. Robocup: The robot world cup initiative. **Proceedings of the First International Conference on Autonomous Agents**, p. 340–347, 1997. Citado nas pp. 14 e 17.

LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep Learning**. [*S.l.*]: Nature Publishing Group, 2015. v. 521. 436–444 p. Citado nas pp. 15 e 21.

MACKWORTH, A. K. On seeing robots. *In*: **Computer Vision: Systems, Theory, and Applications**. Singapore: World Scientific, 1993. p. 1–13. Citado na p. 14.

RoboCup SSL Committee. **Game Logs**. 2024. https://ssl.robocup.org/game-logs/. Acessado em: 5 de novembro de 2025. Citado na p. 27.

RoboCup SSL Community. **ssl-vision**. 2024. Disponível em: https://github.com/RoboCup-SSL/ssl-vision. Citado na p. 27.

ROCHA, R.; DIAS, J.; CAMPOS, A. Ssl: A high-level control language for robot soccer. **RoboCup 2012: Robot Soccer World Cup XVI**, Springer, p. 1–12, 2012. Citado na p. 17.

ROMERO, R. A. F.; PRESTES, E.; OSóRIO, F. *et al.* **Robótica Móvel**. Rio de Janeiro: LTC, 2014. E-book, p.24. ISBN 978-85-216-2642-8. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/978-85-216-2642-8/. Citado na p. 17.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *In*: **15th Annual Conference of the International Speech Communication Association (INTERSPEECH)**. [*s.n.*], 2014. p. 338–342. DOI 10.21437/Interspeech.2014-80. Disponível em: https://www.isca-speech.org/archive/interspeech_2014/i14_0338.html. Citado na p. 23.

STONE, P.; VELOSO, M. Layered learning. **Machine Learning**, v. 40, n. 3, p. 255–273, 2000. Citado na p. 17.

ZICKLER, T.; LAUE, T.; LAUE, T.; LAUE, T.; LAUE, T. The shared vision system for the robocup small size league. *In*: SPRINGER. **Proceedings of the RoboCup 2009 Symposium**. 2009. p. 1–12. DOI 10.1007/978-3-642-11876-$0_3$7. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-11876-0_37. Citado nas pp. 17 e 18.


UnB