UNIVERSIDADE FEDERAL DE GOIÁS (UFG)

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO (EMC)

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO (PPGEEC)

MURILO GUIMARÃES CORREIA

Modelagem e Identificação de Sistemas Dinâmicos com Redes Recorrentes Profundas aplicados em Pedais de Distorção e Robôs Móveis

GOIÂNIA

2025

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[X] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Murilo Guimarães Correia

3. Título do trabalho

Modelagem e Identificação de Sistemas Dinâmicos com Redes Recorrentes Profundas aplicados em Pedais de Distorção e Robôs Móveis

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 18/03/2025, às 10:26, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Murilo Guimarães Correia**, **Discente**, em 18/03/2025, às 10:34, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5241015** e o código CRC **7BF80A3A**.

Referência: Processo nº 23070.003668/2025-47 SEI nº 5241015

MURILO GUIMARÃES CORREIA

Modelagem e Identificação de Sistemas Dinâmicos com Redes Recorrentes Profundas aplicados em Pedais de Distorção e Robôs Móveis

Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Engenharia Elétrica e de Computação, da Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), da Universidade Federal de Goiás (UFG), como requisito para obtenção do título de Mestre em Engenharia Elétrica e de Computação.

Área de Concentração: Engenharia de Computação.
Orientador: Professor Doutor Alisson Assis Cardoso

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Correia, Murilo Guimarães Correia
Modelagem e Identificação de Sistemas Dinâmicos com Redes Recorrentes Profundas aplicados em Pedais de Distorção e Robôs Móveis [manuscrito] / Murilo Guimarães Correia Correia. - 2025.
LXXIX, 79 f.: il.

Orientador: Prof. Dr. Alisson Assis Cardoso Cardoso.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2025.
Bibliografia.
Inclui siglas, fotografias, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Redes Neurais Profundas. 2. Modelagem de Sistemas Dinâmicos. 3. Aprendizagem Profunda. I. Cardoso, Alisson Assis Cardoso, orient. II. Título.

CDU 621.3

UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **2** da sessão de Defesa de Dissertação de **Murilo Guimarães Correia**, que confere o título de Mestre em **Engenharia Elétrica e de Computação**, na área de concentração em **Engenharia de Computação**.

Aos **vinte e um dias do mês de fevereiro de dois mil e vinte e cinco**, a partir das **18h00min.**, realizou-se a sessão pública de Defesa de Dissertação intitulada **"Modelagem e Identificação de Sistemas Dinâmicos com Redes Recorrentes Profundas aplicados em Pedais de Distorção e Robôs Móveis"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **Alisson Assis Cardoso** - (**EMC/UFG)** com a participação dos demais membros da Banca Examinadora: Professor Doutor **Gélson da Cruz Júnior** - (**EMC/UFG)**, membro titular Interno; Professor Doutor **Marcos Antônio de Sousa (EMC/UFG)** Membro Titular Externo. **cujas participações ocorreram através de videoconferência** https://meet.google.com/xgr-pvdz-gfi Durante a arguição os membros da banca **não fizeram** sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Alisson Assis Cardoso, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos **vinte e um dias do mês de fevereiro de dois mil e vinte e cinco.**

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 21/02/2025, às 18:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 21/02/2025, às 18:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Marcos Antonio De Sousa**, **Professor do Magistério Superior**, em 21/02/2025, às 18:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5171484** e o código CRC **B710FE42**.

Referência: Processo nº 23070.003668/2025-47 SEI nº 5171484

Resumo

Este trabalho trata da *Identificação e Modelagem de Sistemas Dinâmicos com o Uso de Redes Neurais Profundas*, aplicando Redes Neurais Multicamadas (MLP), Redes Neurais Recorrentes (LSTM e GRU) e a arquitetura estendida xLSTM (*Extended Long Short-Term Memory*) em dois projetos distintos: a modelagem de distorção de sinal de guitarra elétrica e o comportamento dinâmico do robô móvel TurtleBot 3. No primeiro projeto, MLP e LSTM foram utilizadas para modelar a distorção aplicada ao sinal de áudio de uma guitarra, simulando o efeito de diferentes resistências. A análise dos resultados foi realizada por meio da Função de Distribuição Acumulada (FDA), do teste Kolmogorov-Smirnov (KS) e do Erro Quadrático Médio (EQM). A LSTM demonstrou boa capacidade de capturar as dependências temporais do sinal de áudio, enquanto a MLP conseguiu modelar com precisão a relação entre as entradas e saídas. Os resultados mostraram baixos valores de EQM e uma boa correspondência no teste KS, demonstrando que ambas as arquiteturas são eficazes para modelar distorções de áudio. No segundo projeto, a modelagem do comportamento dinâmico do TurtleBot 3 foi realizada utilizando quatro modelos: MLP, GRU, LSTM e xLSTM. O objetivo foi prever suas trajetórias e velocidades com base em dados simulados. As métricas de avaliação incluíram o Erro Quadrático Médio (EQM), a Raiz do Erro Quadrático Médio (REQM), o Erro Absoluto Médio (EAM), o Erro Absoluto Mediano (MdAE) e o coeficiente de determinação R^2. Entre os modelos avaliados, a xLSTM obteve os melhores resultados em praticamente todas as métricas, destacando-se como a arquitetura mais precisa na modelagem do sistema dinâmico do robô. No entanto, a MLP também demonstrou um desempenho excelente, superando os modelos GRU e LSTM, mesmo sendo um modelo mais simples. Essa observação reforça a eficiência da MLP em capturar relações não lineares entre entradas e saídas, sendo uma alternativa competitiva para modelagem de sistemas dinâmicos. Em conclusão, os resultados obtidos em ambos os projetos evidenciam a capacidade das redes neurais profundas, incluindo MLP, LSTM, GRU e xLSTM, em modelar sistemas dinâmicos complexos. A análise das métricas confirma a robustez da metodologia utilizada, tornando-a promissora para aplicações em Gêmeos Digitais, com potencial de monitoramento e controle em tempo real.

Palavras-chave: Redes Neurais Profundas. Modelagem de Sistemas Dinâmicos. Aprendizagem Profunda.

Abstract

This work addresses the *Identification and Modeling of Dynamic Systems Using Deep Neural Networks*, applying Multilayer Perceptron (MLP), Recurrent Neural Networks (LSTM and GRU), and the extended xLSTM (*Extended Long Short-Term Memory*) architecture in two distinct projects: the modeling of electric guitar signal distortion and the dynamic behavior of the TurtleBot 3 mobile robot. In the first project, MLP and LSTM were used to model the distortion applied to a guitar audio signal, simulating the effect of different resistances. The results were analyzed using the Cumulative Distribution Function (CDF), the Kolmogorov-Smirnov (KS) test, and the Mean Squared Error (MSE). LSTM demonstrated a strong ability to capture temporal dependencies in the audio signal, while MLP effectively modeled the relationship between inputs and outputs. The results showed low MSE values and a good match in the KS test, demonstrating that both architectures are effective in modeling audio distortions. In the second project, the modeling of the dynamic behavior of the TurtleBot 3 was carried out using four models: MLP, GRU, LSTM, and xLSTM. The goal was to predict its trajectories and velocities based on simulated data. The evaluation metrics included the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Median Absolute Error (MdAE), and the coefficient of determination R^2. Among the evaluated models, xLSTM achieved the best results across almost all metrics, standing out as the most accurate architecture for modeling the robot's dynamic system. However, MLP also demonstrated excellent performance, surpassing the GRU and LSTM models despite being a simpler approach. This finding reinforces the efficiency of MLP in capturing nonlinear relationships between inputs and outputs, making it a competitive alternative for dynamic system modeling. In conclusion, the results obtained in both projects highlight the capability of deep neural networks, including MLP, LSTM, GRU, and xLSTM, in modeling complex dynamic systems. The metric analysis confirms the robustness of the proposed methodology, making it a promising approach for Digital Twin applications, with potential for real-time monitoring and control.

Keywords: Deep Neural Networks. Modeling of Dynamic Systems. Deep Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

RNPs	*Redes Neurais Profundas*
RNNs	*Redes Neurais Recorrentes*
LSTM	*Long Short-Term Memory*
xLSTM	*Extended Long Short-Term Memory*
GRU	*Gated Recurrent Units*
RNA	*Rede Neural Artificial*
EQM	*Erro Quadrático Médio*
mLSTM	*Matrix Long Short-Term Memory*
sLSTM	*Shift Long Short-Term Memory*
FDA	*Função de Distribuição Acumulada*
KS	*Teste de Kolmogorov-Smirnov*
ROS	*Robot Operating System*
REQM	*Raiz do Erro Quadrático Médio*
EAM	*Erro Absoluto Médio*
MdAE	*Median Absolute Error*

Trabalhos Submetidos e Publicados

Trabalhos aprovados e/ou publicados:

- CORREIA, MURILO G.; ALMEIDA, SAMUEL C. DE ; CARDOSO, ALISSON A. . Modelagem de Efeitos de Distorção em Sinais de Guitarra Elétrica com Uso de Redes Perceptron Multicamadas. In: Escola Regional de Informática de Goiás, 2023, Brasil. Anais da XI Escola Regional de Informática de Goiás (ERI-GO 2023).
- CORREIA, Murilo Guimarães; CARDOSO, Álisson Assis; VIEIRA, Flávio Henrique Teles; FERREIRA, Marcus Vinícius Gonzaga. EXPLORANDO REDES NEURAIS PROFUNDAS PARA MODELAGEM DE DISTORÇÃO EM SINAIS DE GUITARRA ELÉTRICA. In: LVI Simpósio Brasileiro de Pesquisa Operacional. 2024. Fortaleza, Ceará, Brasil.

Submetidos em revistas:

- CORREIA, MURILO G.; ALMEIDA, SAMUEL C. DE ; CARDOSO, ALISSON A. . Modelagem de Efeitos de Distorção em Sinais de Guitarra Elétrica com Uso de Redes Perceptron Multicamadas. In: Revista Tecnia 2023.

Sumário

Introdução

Os sistemas dinâmicos são uma classe de sistemas caracterizados por comportamentos que evoluem ao longo do tempo, podendo ser influenciados por suas condições iniciais e entradas externas. Esses sistemas são frequentemente encontrados na natureza e na engenharia, onde a modelagem e a previsão de seu comportamento são fundamentais para diversas aplicações (BRUNTON; KUTZ, 2022; NELLES; NELLES, 2020). A complexidade dos sistemas dinâmicos aumenta consideravelmente quando são não lineares, o que significa que a relação entre entrada e saída não é proporcional, dificultando sua modelagem utilizando técnicas tradicionais (NELLES; NELLES, 2020).

Para enfrentar o desafio de modelar sistemas dinâmicos não lineares, o uso de RNPs tem se mostrado promissor. As RNPs são compostas por múltiplas camadas de neurônios artificiais, capazes de capturar padrões complexos em dados (ZHANG et al., 2023). Dentre as várias arquiteturas de RNPs, as RNNs se destacam por sua capacidade de processar sequências temporais, tornando-as ideais para a modelagem de sistemas dinâmicos (HEWING et al., 2020). As RNNs, especialmente as variantes modernas como LSTM e GRU, conseguem manter informações sobre estados passados, permitindo que modelos aprendam a dinâmica temporal dos sistemas (PAN; DURAISAMY, 2020).

Recentemente, a introdução da xLSTM (*Extended Long Short-Term Memory*) representou um avanço significativo nas arquiteturas de redes neurais recorrentes. A xLSTM melhora a capacidade de aprendizado de dependências temporais de longo prazo e oferece maior estabilidade durante o treinamento, especialmente em tarefas que envolvem sistemas dinâmicos complexos (BECK et al., 2024). No contexto deste trabalho, foram realizados dois estudos focados na identificação e modelagem de sistemas dinâmicos utilizando Redes Neurais Profundas.

O primeiro estudo abordou a modelagem da distorção de sinais de guitarra elétrica, utilizando as arquiteturas MLP e LSTM. A MLP foi empregada para capturar as relações não lineares entre o sinal de entrada e sua versão distorcida, permitindo uma modelagem eficiente das transformações aplicadas ao áudio. Já a LSTM foi utilizada para explorar as dependências temporais no sinal de áudio, possibilitando uma representação mais precisa da evolução do efeito de distorção ao longo do tempo. A combinação dessas abordagens permitiu uma modelagem detalhada do comportamento do pedal de distorção, demonstrando a capacidade das redes neurais em replicar efeitos não lineares complexos.

O segundo estudo focou na modelagem do comportamento dinâmico do robô móvel TurtleBot 3, aplicando quatro arquiteturas distintas: MLP, GRU, LSTM e xLSTM. A MLP serviu como referência inicial, capturando relações diretas entre os estados do robô

e suas respectivas saídas (NELLES; NELLES, 2020). A GRU foi utilizada como uma alternativa mais eficiente às LSTMs, sendo capaz de processar sequências temporais com menor custo computacional. A LSTM, por sua vez, foi aplicada devido à sua capacidade de manter informações temporais de longo prazo, permitindo uma representação mais robusta das dinâmicas do robô. Por fim, a xLSTM foi integrada ao estudo como uma evolução das redes recorrentes, proporcionando maior estabilidade no treinamento e melhor captura das dinâmicas temporais do sistema.

A análise comparativa entre os modelos indicou que a xLSTM apresentou os melhores resultados em praticamente todas as métricas, tanto nos dados de treinamento quanto de teste, destacando-se como a arquitetura mais precisa para a modelagem do comportamento dinâmico do TurtleBot 3. A MLP também demonstrou um desempenho excelente, com valores muito próximos aos da xLSTM, especialmente no coeficiente de determinação (R^2), mas com leve inferioridade nos dados de teste. Por outro lado, a GRU e a LSTM apresentaram um desempenho mais modesto, com a LSTM mostrando maior sensibilidade aos dados de teste, especialmente nas métricas de Erro Quadrático Médio (MSE) e Raiz do Erro Quadrático Médio (RMSE). Dessa forma, os resultados obtidos evidenciam a eficácia das Redes Neurais Profundas na modelagem de sistemas dinâmicos, com a xLSTM e a MLP se destacando como as soluções mais promissoras para previsões precisas de trajetórias e velocidades, reforçando seu potencial para aplicações em Gêmeos Digitais, controle autônomo e modelagem de sistemas não lineares.

Adicionalmente, o conceito de Gêmeos Digitais (*Digital Twins*) tem ganhado destaque, especialmente no contexto da Indústria 4.0. Gêmeos Digitais são representações virtuais de sistemas físicos, capazes de simular e prever comportamentos em tempo real (KRITZINGER et al., 2018; TAO et al., 2018). A modelagem de sistemas dinâmicos com Redes Neurais Profundas, incluindo arquiteturas como a xLSTM, se alinha diretamente com essa tendência, permitindo a criação de modelos precisos e eficientes para Gêmeos Digitais (JONES et al., 2020; LENG et al., 2019). No futuro, espera-se que o uso dessas redes em Gêmeos Digitais proporcione simulações mais precisas e uma integração mais próxima entre os mundos físico e digital (TAO et al., 2018).

O presente trabalho busca, portanto, aprofundar o estudo na modelagem e identificação de sistemas dinâmicos utilizando Redes Neurais Profundas, com vistas à aplicação em Gêmeos Digitais e outras áreas correlatas, proporcionando uma base sólida para futuras pesquisas e inovações tecnológicas (BRUNTON; KUTZ, 2022).

Parte I

Referenciais teóricos

1 Sistemas Dinâmicos

1.1 Definição de Sistemas Dinâmicos

Sistemas dinâmicos referem-se a sistemas cujo comportamento evolui ao longo do tempo, influenciado por suas condições iniciais e pelas entradas aplicadas. Matematicamente, um sistema dinâmico pode ser descrito por um conjunto de equações diferenciais ou de diferença que descrevem a variação do estado do sistema ao longo do tempo. A variável de estado $x(t)$ representa o conjunto de grandezas necessárias para descrever completamente o estado do sistema em um dado instante t (ISIDORI, 1985).

Em termos gerais, a evolução de um sistema dinâmico pode ser expressa como:

$$\dot{x}(t) = f(x(t), u(t), t)$$

onde $\dot{x}(t)$ é a taxa de variação do estado ao longo do tempo, $f(\cdot)$ é uma função que descreve a dinâmica do sistema, $u(t)$ representa as entradas aplicadas ao sistema e t é o tempo. Para sistemas discretos, a equação equivalente é:

$$x(k+1) = f(x(k), u(k), k)$$

Esses sistemas podem ser encontrados em várias áreas, como engenharia, biologia, economia e física, sendo essenciais para prever e controlar comportamentos complexos (ISIDORI, 1985).

1.2 Classificação dos Sistemas Dinâmicos

Os sistemas dinâmicos podem ser classificados de diferentes formas, dependendo de suas características. As principais categorias incluem:

1.2.1 Sistemas Lineares vs. Não Lineares

Um sistema dinâmico é linear se a relação entre as entradas e saídas for proporcional e obedecer ao princípio da superposição. A equação que descreve um sistema linear é da forma:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

onde A e B são matrizes constantes que descrevem a dinâmica do sistema. Por outro lado, sistemas não lineares não seguem esse princípio, e suas equações são mais complexas. Um exemplo de sistema não linear é:

$$\dot{x}(t) = Ax(t) + Bu(t) + Cx(t)^2$$

Em sistemas não lineares, pequenas variações nas condições iniciais ou nas entradas podem causar grandes mudanças no comportamento, tornando-os mais difíceis de modelar e prever (SLOTINE; LI et al., 1991).

Para ilustrar a diferença entre sistemas lineares e não lineares, considere o seguinte diagrama de blocos genérico:

Figura 1.1 – Diagrama de Blocos de um Sistema Dinâmico Genérico

Fonte: Próprio autor.

Neste diagrama, a entrada $u(t)$ é aplicada ao sistema dinâmico, e a saída $x(t)$ representa o estado ou resposta do sistema.

1.2.2 Sistemas Determinísticos vs. Estocásticos

Sistemas determinísticos são aqueles cujo comportamento é completamente definido pelas equações de estado e pelas entradas, ou seja, o futuro do sistema é inteiramente previsível se o estado inicial for conhecido. Já em sistemas estocásticos, há incertezas envolvidas, e o comportamento futuro não pode ser previsto com exatidão, sendo modelado por processos probabilísticos. A equação de um sistema estocástico pode incluir termos de ruído, como:

$$\dot{x}(t) = f(x(t), u(t), t) + w(t)$$

onde $w(t)$ é um termo de ruído aleatório que representa incertezas ou perturbações no sistema (JAZWINSKI, 2007).

1.2.3 Sistemas Contínuos vs. Discretos

Sistemas dinâmicos contínuos são aqueles que evoluem continuamente ao longo do tempo, como mostrado pela equação diferencial:

$$\dot{x}(t) = f(x(t), u(t), t)$$

Em contrapartida, sistemas discretos evoluem em etapas de tempo definidas, sendo descritos por equações de diferença, como:

$$x(k+1) = f(x(k), u(k), k)$$

Sistemas contínuos são comuns em fenômenos físicos, enquanto sistemas discretos são frequentemente usados em controle digital e processamento de sinais (KAILATH, 1980).

1.3 Importância da Modelagem de Sistemas Dinâmicos em Engenharia e Ciência

A modelagem de sistemas dinâmicos é crucial em diversas áreas da ciência e da engenharia, uma vez que permite prever e controlar o comportamento de sistemas complexos. Exemplos práticos incluem:

- **Engenharia de Controle**: Modelos dinâmicos são a base para o projeto de sistemas de controle que garantem a estabilidade e o desempenho de processos industriais, robôs e veículos autônomos (ISIDORI, 1985).
- **Biologia de Sistemas**: A modelagem de sistemas biológicos permite compreender e prever a dinâmica de populações, redes genéticas e processos fisiológicos (ALON, 2019).
- **Economia**: Na economia, sistemas dinâmicos são usados para modelar a evolução de mercados financeiros e a dinâmica do crescimento econômico (ACEMOGLU; RESTREPO, 2018).
- **Climatologia**: Modelos dinâmicos são essenciais para prever mudanças climáticas e estudar o impacto de fatores como gases de efeito estufa e desmatamento (JR, 2010).

Em todos esses exemplos, a modelagem precisa de sistemas dinâmicos permite o desenvolvimento de soluções otimizadas, garantindo maior eficiência e desempenho.

1.4 Desafios na Modelagem de Sistemas Dinâmicos Não Lineares

A modelagem de sistemas dinâmicos não lineares apresenta desafios significativos devido à complexidade inerente desses sistemas. Entre os principais desafios, destacam-se:

- **Sensibilidade às Condições Iniciais**: Em sistemas não lineares, pequenas variações nas condições iniciais podem resultar em comportamentos drasticamente diferentes, dificultando a previsão precisa (STROGATZ, 2018).
- **Existência de Múltiplos Atratores**: Sistemas não lineares podem possuir múltiplos estados estáveis ou pontos de equilíbrio, conhecidos como atratores, tornando difícil prever para qual estado o sistema convergirá (OTT, 2002).
- **Dificuldade na Linearização**: Muitos métodos tradicionais de análise e controle dependem da linearização de sistemas ao redor de um ponto de operação. No entanto, essa abordagem nem sempre é adequada para sistemas fortemente não lineares, uma vez que a aproximação linear pode não capturar a dinâmica global do sistema (ZHU, 2023).
- **Sistemas Caóticos**: Em alguns casos, sistemas não lineares podem exibir comportamento caótico, onde o sistema parece seguir um padrão desordenado e imprevisível, mesmo que suas equações sejam determinísticas (OTT, 2002).

Para lidar com esses desafios, técnicas avançadas, como a aplicação de Redes Neurais Profundas (RNPs), têm sido utilizadas para modelar a dinâmica não linear com maior precisão (RAISSI; PERDIKARIS; KARNIADAKIS, 2019). Além disso, a utilização de métodos híbridos, que combinam modelos baseados em física com técnicas de aprendizado de máquina, tem mostrado potencial para superar as limitações dos métodos tradicionais (KARNIADAKIS et al., 2021).

1.4.1 Exemplo de Sistema Não Linear

Um exemplo clássico de sistema dinâmico não linear é o pêndulo simples. A equação que descreve o movimento de um pêndulo é dada por:

$$\ddot{\theta}(t) + \frac{g}{l}\sin(\theta(t)) = 0$$

onde $\theta(t)$ é o ângulo de oscilação, g é a aceleração gravitacional e l é o comprimento do pêndulo. Esta equação é não linear devido ao termo $\sin(\theta)$, e soluções analíticas exatas são difíceis de obter para grandes amplitudes de oscilação (STROGATZ, 2018). Para pequenas oscilações, a equação pode ser linearizada como:

$$\ddot{\theta}(t) + \frac{g}{l}\theta(t) = 0$$

A linearização simplifica a análise, mas deixa de capturar a dinâmica completa do sistema para grandes amplitudes, ilustrando um dos principais desafios na modelagem de sistemas não lineares (ZHU, 2023).

2 Identificação de Sistemas Dinâmicos

2.1 Conceito de Identificação de Sistemas

A identificação de sistemas é o processo de determinar um modelo matemático que representa o comportamento dinâmico de um sistema físico com base em dados experimentais. Este processo envolve a construção e a validação de modelos que podem prever as respostas do sistema para diferentes entradas. A identificação é crucial para a análise, controle e simulação de sistemas dinâmicos em diversas áreas, como engenharia, economia e ciências físicas (SIMPKINS, 2012; LENNART, 1999).

A modelagem geralmente começa com a coleta de dados do sistema em operação e, em seguida, utiliza técnicas estatísticas e matemáticas para ajustar um modelo que melhor descreva esses dados. O objetivo final é criar um modelo que possa ser usado para prever o comportamento do sistema sob diferentes condições (GOODWIN; SIN, 2014).

2.2 Técnicas Tradicionais de Identificação

Existem várias técnicas tradicionais de identificação de sistemas que podem ser aplicadas dependendo da complexidade e das características do sistema em questão. Entre as mais comuns estão:

2.2.1 Modelos ARX (Auto-Regressivo com Entrada Exógena)

Os modelos ARX são uma forma simples de modelar sistemas dinâmicos lineares. Eles são baseados em uma relação linear entre a saída do sistema e suas entradas passadas e saídas passadas.

A equação geral de um modelo ARX é dada por:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t) \tag{2.1}$$

onde:

- $y(t)$ é a saída do sistema,
- $u(t)$ é a entrada do sistema,
- $A(q^{-1})$ e $B(q^{-1})$ são polinômios no operador de atraso q^{-1},
- $e(t)$ é o erro de modelagem.

2.2.2 Modelos ARMAX (Auto-Regressivo com Média Móvel e Entrada Exógena)

Os modelos ARMAX são uma extensão dos modelos ARX e incluem um componente de média móvel que pode capturar comportamentos adicionais não representados pelos modelos ARX (KATAYAMA et al., 2005).

A equação de um modelo ARMAX é:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t) \tag{2.2}$$

onde:

- $C(q^{-1})$ é o polinômio de média móvel.

2.2.3 Modelos de Espaço de Estados

Modelos de espaço de estados são utilizados para descrever sistemas dinâmicos usando um conjunto de equações diferenciais lineares. Eles são particularmente úteis para sistemas multivariáveis e podem modelar comportamentos dinâmicos mais complexos (KALMAN, 1960).

O modelo de espaço de estados é descrito por:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.3}$$

$$y(t) = Cx(t) + Du(t) \tag{2.4}$$

onde:

- $x(t)$ é o vetor de estado,
- $u(t)$ é o vetor de entrada,
- $y(t)$ é o vetor de saída,
- A, B, C, e D são matrizes de sistema.

2.3 Limitações das Técnicas Tradicionais na Modelagem de Sistemas Dinâmicos Não Lineares

Embora as técnicas tradicionais como ARX, ARMAX e modelos de espaço de estados sejam eficazes para sistemas lineares, elas enfrentam limitações significativas ao lidar

com sistemas não lineares. Sistemas não lineares podem apresentar comportamentos complexos que não podem ser capturados adequadamente por modelos lineares (SLOTINE; LI et al., 1991).

As principais limitações incluem:

- **Capacidade de Representação Limitada:** Modelos lineares não conseguem capturar efeitos não lineares, como saturações e histerese (BILLINGS, 2013).
- **Complexidade Computacional:** A identificação de sistemas não lineares geralmente requer algoritmos mais complexos e computacionalmente intensivos (SJÖBERG et al., 1995).
- **Generalização:** Modelos lineares podem ter dificuldades para generalizar o comportamento do sistema para novas condições operacionais.

2.4 Critérios de Desempenho na Identificação de Sistemas

A avaliação da qualidade de um modelo identificado é crucial para garantir que ele represente adequadamente o sistema dinâmico. Vários critérios de desempenho são utilizados para avaliar a precisão e a eficácia dos modelos identificados:

2.4.1 Erro de Predição

O erro de predição é uma medida da discrepância entre as saídas previstas pelo modelo e as saídas reais do sistema. É calculado como:

$$e(t) = y_{real}(t) - y_{modelo}(t) \tag{2.5}$$

onde $y_{real}(t)$ é a saída real e $y_{modelo}(t)$ é a saída prevista pelo modelo (LENNART, 1999).

2.4.2 Validação Cruzada

A validação cruzada é uma técnica para avaliar o desempenho do modelo ao dividir os dados em conjuntos de treinamento e teste. O modelo é treinado com um conjunto de dados e testado com um conjunto diferente. Esta abordagem ajuda a evitar o overfitting e a garantir que o modelo generalize bem para novos dados (GOODWIN; SIN, 2014).

2.4.3 Critérios Estatísticos

Critérios estatísticos como o EQM e o coeficiente de determinação (R^2) são frequentemente utilizados para avaliar a qualidade do ajuste do modelo. O EQM é dado por:

$$EQM = \frac{1}{N}\sum_{i=1}^{N}\left(y_{real}(i) - y_{modelo}(i)\right)^2 \tag{2.6}$$

onde N é o número de amostras (BILLINGS, 2013).

3 Redes Neurais Profundas (RNPs)

3.1 Introdução às RNPs e seu Papel na Modelagem de Sistemas Complexos

Redes Neurais Profundas (RNPs) são uma classe de modelos de aprendizado de máquina que imitam a estrutura e funcionamento do cérebro humano para resolver problemas complexos. Elas são compostas por várias camadas de neurônios artificiais, o que permite a modelagem de padrões e relações complexas em dados. O papel das RNPs na modelagem de sistemas complexos é fundamental, especialmente quando se lida com dados não lineares e estruturas de alta dimensão (GOODFELLOW, 2016).

RNPs têm sido aplicadas com sucesso em diversas áreas, como reconhecimento de imagem, processamento de linguagem natural e modelagem de sistemas dinâmicos. Elas oferecem uma abordagem poderosa para capturar padrões e relações não lineares que modelos tradicionais podem não conseguir identificar (LECUN; BENGIO; HINTON, 2015; SCHMIDHUBER, 2015).

A Figura 3.1 apresenta a estrutura de uma RNA, ilustrando a camada de entrada, as camadas ocultas e a camada de saída. Esse diagrama destaca a interconexão entre os neurônios das diferentes camadas, enfatizando o fluxo de informações dentro da rede.

3.2 Arquiteturas Básicas de RNPs

Existem várias arquiteturas de RNPs, cada uma projetada para tipos específicos de problemas e dados. As mais comuns incluem:

3.2.1 Perceptron Multicamadas (MLP)

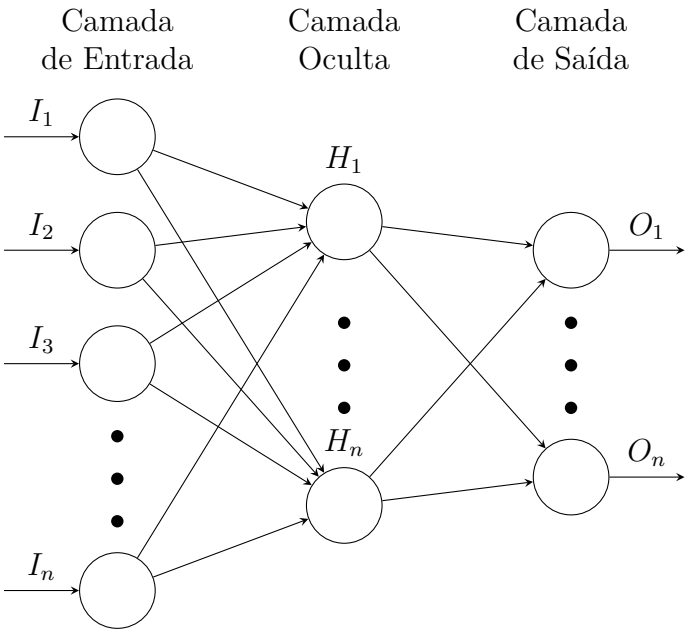
O Perceptron Multicamadas (MLP) é uma das arquiteturas mais básicas de redes neurais profundas. Consiste em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está totalmente conectado a todos os neurônios da camada seguinte (ZHANG et al., 2023).

A equação para um neurônio em uma camada oculta é dada por:

$$a_j = \sigma\left(\sum_i w_{ij} x_i + b_j\right) \tag{3.1}$$

onde:

Figura 3.1 – Diagrama de uma RNA mostrando a camada de entrada, camadas ocultas e camada de saída.



Fonte: Próprio autor.

- a_j é a ativação do neurônio j,
- w_{ij} são os pesos das conexões,
- x_i são as entradas,
- b_j é o viés,
- σ é a função de ativação (como ReLU, Sigmoid, etc.).

3.2.2 Redes Neurais Recorrentes (RNN)

Redes Neurais Recorrentes (RNN) são projetadas para lidar com dados sequenciais e temporais. Elas possuem conexões que formam ciclos, permitindo que a informação persista ao longo do tempo. Uma variante importante das RNNs são as Long Short-Term Memory (LSTM), que são projetadas para capturar dependências de longo prazo em sequências (ZHANG et al., 2023).

A equação para um passo de tempo t em uma RNN é:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b) \tag{3.2}$$

onde h_t é o estado oculto no tempo t, W_h e W_x são matrizes de pesos, x_t é a entrada e b é o viés.

3.3 Treinamento de RNPs: Técnicas de Otimização e Funções de Perda

O treinamento de redes neurais profundas envolve a otimização dos pesos da rede para minimizar a função de perda. Isso é feito através do algoritmo de retropropagação e técnicas de otimização (ZHANG et al., 2023).

3.3.1 Algoritmo de Retropropagação

O algoritmo de retropropagação é utilizado para calcular o gradiente da função de perda com relação aos pesos da rede, permitindo a atualização desses pesos para reduzir o erro.

A atualização dos pesos é dada por:

$$w_{ij} = w_{ij} - \eta \frac{\partial L}{\partial w_{ij}} \tag{3.3}$$

onde:

- η é a taxa de aprendizado,
- L é a função de perda,
- $\frac{\partial L}{\partial w_{ij}}$ é o gradiente da função de perda em relação ao peso w_{ij}.

3.3.2 Funções de Perda

As funções de perda quantificam o erro entre as previsões do modelo e os valores reais. Algumas funções de perda comuns incluem:

- **Erro Quadrático Médio (EQM):**

$$L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.4}$$

- **Entropia Cruzada (Cross-Entropy):**

$$L = -\sum_{i} y_i \log(\hat{y}_i) \tag{3.5}$$

3.4 Overfitting e Técnicas de Regularização em RNPs

Overfitting ocorre quando um modelo aprende os detalhes e ruídos do conjunto de treinamento a ponto de comprometer sua capacidade de generalização para novos dados. Para combater o *overfitting*, várias técnicas de regularização podem ser aplicadas (ZHANG et al., 2023):

3.4.1 Regularização L1 e L2

A regularização L1 e L2 adiciona um termo de penalização à função de perda para evitar que os pesos se tornem excessivamente grandes.

$$L_{L1} = \lambda \sum_i |w_i| \tag{3.6}$$

$$L_{L2} = \lambda \sum_i w_i^2 \tag{3.7}$$

onde λ é o parâmetro de regularização.

3.4.2 Dropout

O *Dropout* é uma técnica que desativa aleatoriamente uma fração dos neurônios durante o treinamento para evitar que a rede dependa excessivamente de qualquer neurônio específico (ZHANG et al., 2023). A equação para o Dropout é:

$$\hat{y} = \frac{y}{1-p} \tag{3.8}$$

onde p é a taxa de Dropout e $\hat{y}$ é a saída ajustada.

3.5 Aplicações de RNPs em Sistemas Dinâmicos

Redes Neurais Profundas têm sido amplamente utilizadas na modelagem e controle de sistemas dinâmicos. Elas oferecem uma abordagem flexível para capturar dinâmicas complexas e não lineares que são desafiadoras para modelos tradicionais. Aplicações incluem a previsão de séries temporais, controle de sistemas robóticos e otimização de processos industriais (ZHANG et al., 2023).

3.5.1 Modelagem de Sistemas Dinâmicos com RNPs

A modelagem de sistemas dinâmicos utilizando RNPs envolve o treinamento de redes neurais para prever o comportamento futuro do sistema com base em dados históricos.

Isso é especialmente útil em situações onde o sistema apresenta dinâmicas não lineares e interações complexas entre variáveis (ZHANG et al., 2023).

4 Redes Neurais Recorrentes (RNNs)

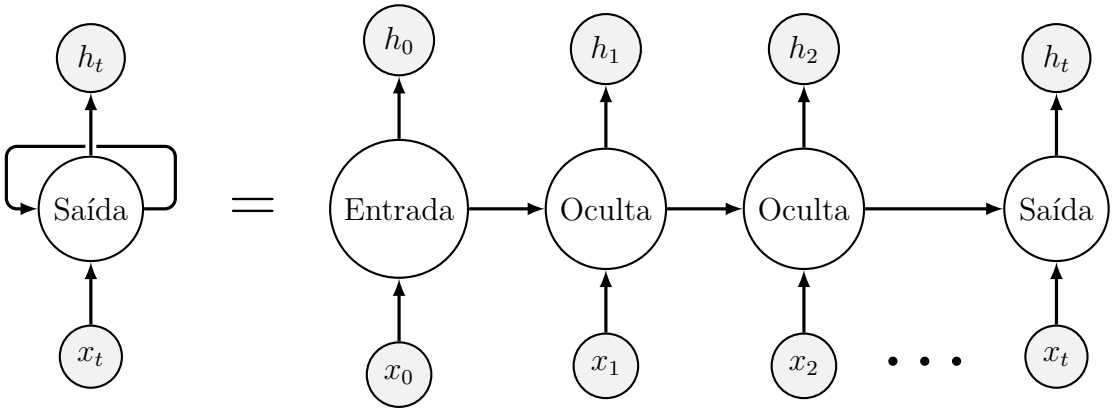
4.1 Definição e Arquitetura Básica das RNNs

RNNs são uma classe de redes neurais projetadas para modelar dados sequenciais e temporais. Diferentemente das redes neurais *feedforward*, as RNNs possuem conexões recorrentes que permitem que a saída de um neurônio em um momento seja usada como entrada para o mesmo ou outro neurônio em momentos futuros. Essa característica torna as RNNs adequadas para tarefas como modelagem de sequências e previsão de séries temporais (ZHANG et al., 2023).

A arquitetura básica de uma RNN inclui unidades recorrentes que compartilham os mesmos parâmetros em todas as etapas temporais, o que as diferencia das arquiteturas tradicionais. O principal benefício dessa estrutura é a capacidade de processar sequências de diferentes comprimentos, tornando-a útil para aplicações como reconhecimento de fala, tradução automática e análise de séries temporais.

A arquitetura básica de uma RNN pode ser ilustrada conforme mostrado na Figura 4.1:

Figura 4.1 – Arquitetura básica de uma RNN.



Fonte: Próprio autor

A saída da camada oculta h_t em um instante de tempo t é calculada a partir da entrada x_t e da saída da camada oculta no instante anterior h_{t-1} através da seguinte equação:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \tag{4.1}$$

onde W_h e W_x são matrizes de pesos, b é um vetor de bias e ϕ é uma função de

ativação não linear.

4.2 Problemas de RNNs Tradicionais

Apesar de sua flexibilidade, as RNNs tradicionais enfrentam problemas como o desvanecimento e a explosão de gradientes. O desvanecimento de gradientes ocorre quando os gradientes das funções de ativação se tornam muito pequenos durante o treinamento, o que dificulta a atualização dos pesos e impede a aprendizagem de dependências de longo prazo (ZHANG et al., 2023).

O problema da explosão de gradientes ocorre quando os gradientes se tornam muito grandes, resultando em atualizações instáveis dos pesos e dificultando a convergência do treinamento (ZHANG et al., 2023).

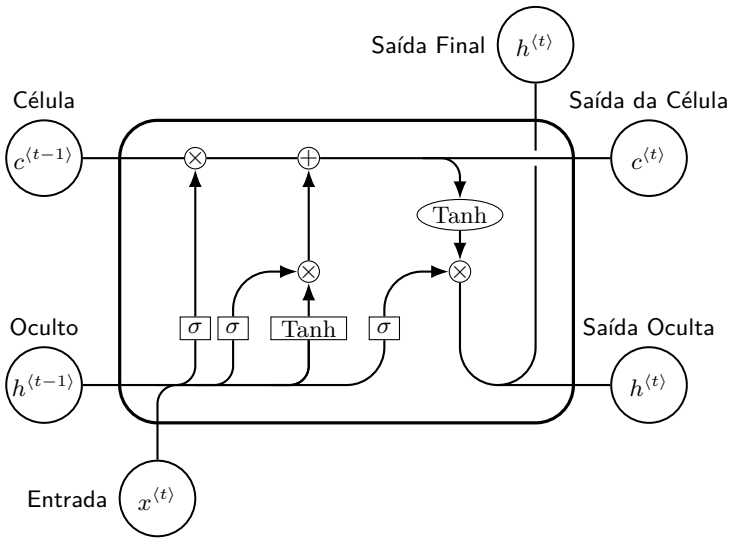
4.3 Variantes de RNNs: LSTM e GRU

Para superar esses problemas, foram desenvolvidas variantes das RNNs, como LSTM e GRU.

4.3.1 LSTM (Long Short-Term Memory)

A arquitetura LSTM foi proposta para lidar com o problema do desvanecimento de gradientes e é composta por unidades de memória que podem manter informações por períodos de tempo mais longos (ZHANG et al., 2023). A estrutura básica de um LSTM é mostrada na Figura 4.2.

Figura 4.2 – Arquitetura básica de uma unidade LSTM.



Fonte: Próprio autor.

A equação fundamental de um LSTM é a seguinte:

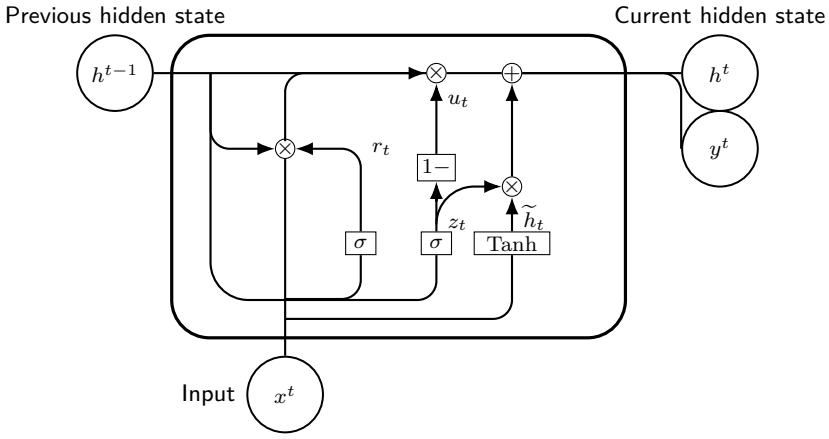
$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{4.2}$$

onde f_t, i_t, e o_t são as saídas das portas de esquecimento, entrada e saída, respectivamente, e $\tilde{C}_t$ é a candidata a memória.

4.3.2 GRU (Gated Recurrent Units)

As GRU são uma simplificação das LSTM, que combinam as portas de entrada e esquecimento em uma única porta, simplificando a arquitetura e a computação (ZHANG et al., 2023). A estrutura básica de uma GRU é mostrada na Figura 4.3.

Figura 4.3 – Arquitetura básica de uma unidade GRU.



Fonte: Próprio autor.

As equações principais para uma GRU são:

$$\begin{aligned} r_t &= \sigma(W_r[h_{t-1}, x_t] + b_r) \\ z_t &= \sigma(W_z[h_{t-1}, x_t] + b_z) \\ \tilde{h}_t &= \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned} \tag{4.3}$$

onde r_t e z_t são as portas de reset e atualização, respectivamente, e $\tilde{h}_t$ é a candidata de estado.

4.3.3 xLSTM (Extended Long Short-Term Memory)

A arquitetura LSTM tem sido amplamente utilizada para modelar sequências temporais, resolvendo problemas como o desvanecimento do gradiente, graças ao uso do "constant error carousel"e das portas de controle (HOCHREITER, 1997). No entanto, com o surgimento dos Transformers, que utilizam mecanismos de autoatenção paralelizados, as LSTMs começaram a ser superadas em escalabilidade e desempenho em diversos contextos (VASWANI, 2017). Nesse cenário, o xLSTM surge como uma evolução das LSTMs, combinando a escalabilidade moderna com novas técnicas para superar as limitações conhecidas das LSTMs tradicionais.

A arquitetura xLSTM apresenta duas principais modificações: (i) *exponential gating* com técnicas apropriadas de normalização e estabilização, e (ii) a introdução de novas estruturas de memória, como o *sLSTM* e o *mLSTM*, cada um com características distintas para otimizar a performance em diferentes cenários (BECK et al., 2024). O *sLSTM* introduz uma técnica de mistura de memória inovadora, enquanto o *mLSTM* é totalmente paralelizável, utilizando memória matricial e uma nova regra de atualização de covariância.

4.3.3.1 mLSTM (Matrix LSTM)

O componente mLSTM da xLSTM utiliza uma abordagem matricial para calcular as transformações internas da célula LSTM, tornando a rede mais eficiente em termos de representação e captura de padrões temporais. A estrutura básica de uma célula LSTM é composta por três portas principais: a porta de esquecimento (*forget gate*), a porta de entrada (*input gate*) e a porta de saída (*output gate*). Estas portas regulam o fluxo de informações, permitindo que a rede aprenda dependências de curto e longo prazo.

A equação de atualização da célula de estado c_t no mLSTM é dada por:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

onde:

- c_t é o estado da célula no tempo t,
- f_t é a porta de esquecimento, calculada como $f_t = \sigma(W_f \cdot h_{t-1} + U_f \cdot x_t + b_f)$,
- i_t é a porta de entrada, calculada como $i_t = \sigma(W_i \cdot h_{t-1} + U_i \cdot x_t + b_i)$,
- $\tilde{c}_t$ é o novo estado da célula, calculado como $\tilde{c}_t = \tanh(W_c \cdot h_{t-1} + U_c \cdot x_t + b_c)$,
- $\odot$ denota a multiplicação elemento a elemento.

A saída da célula h_t, que representa o estado oculto no tempo t, é atualizada de acordo com:

$$h_t = o_t \odot \tanh(c_t)$$

onde o_t é a porta de saída, calculada como $o_t = \sigma(W_o \cdot h_{t-1} + U_o \cdot x_t + b_o)$.

Essa estrutura é semelhante à LSTM tradicional, porém a matriz de pesos W no mLSTM é expandida, permitindo maior capacidade de generalização e armazenamento de informações complexas ao longo do tempo.

4.3.3.2 sLSTM (Shift LSTM)

Além da arquitetura matricial do mLSTM, a xLSTM também incorpora o sLSTM, que utiliza operações de deslocamento (*shifts*) nos estados internos da célula LSTM para facilitar o aprendizado de padrões com deslocamentos temporais. O componente sLSTM é eficaz em capturar mudanças de fase ou deslocamentos temporais em séries temporais que apresentam comportamentos recorrentes.

O sLSTM realiza a seguinte modificação na atualização do estado da célula:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t + S_t$$

onde S_t é o termo de deslocamento, calculado como:

$$S_t = \text{shift}(c_{t-1}, \delta_t)$$

Aqui, δ_t representa o deslocamento aplicado ao estado c_{t-1}, e a função *shift* realiza esse deslocamento de maneira controlada, permitindo que o modelo capture variações temporais de forma eficiente.

Essas inovações permitem que o xLSTM lide com tarefas desafiadoras, como previsão de tokens raros e busca por vizinhos mais próximos, superando limitações inerentes às LSTMs convencionais (BECK et al., 2024). Além disso, o xLSTM também se mostrou competitivo quando comparado a modelos de última geração, como os Transformers, em tarefas de modelagem de linguagem natural e sistemas dinâmicos complexos.

Por fim, a arquitetura xLSTM é construída sobre blocos residuais que empilham as melhorias da *sLSTM* e da *mLSTM*, tornando-a adequada para ser escalada em grandes modelos de linguagem, como demonstrado em estudos recentes (BECK et al., 2024).

4.4 Aplicações de RNNs na Modelagem de Sequências Temporais

RNNs, especialmente suas variantes LSTM e GRU, têm se mostrado eficazes em uma variedade de aplicações de modelagem de sequências temporais. Exemplos incluem a previsão de séries temporais financeiras, análise de sentimentos em textos, e tradução automática de idiomas (ZHANG et al., 2023).

4.5 Vantagens e Desvantagens das RNNs para Sistemas Dinâmicos

As RNNs oferecem várias vantagens na modelagem de sistemas dinâmicos, incluindo a capacidade de capturar dependências temporais complexas e modelar dados sequenciais com alta precisão. No entanto, elas também possuem desvantagens, como a necessidade de grandes quantidades de dados para treinamento e a dificuldade de interpretar os resultados devido à sua natureza de caixa-preta (ZHANG et al., 2023).

Em resumo, embora as RNNs sejam poderosas para tarefas envolvendo dados sequenciais, é crucial considerar suas limitações e escolher a arquitetura mais adequada para a tarefa específica em questão.

5 Materiais e Métodos

5.1 Descrição Geral das Implementações

Este capítulo descreve as metodologias utilizadas no desenvolvimento das duas implementações principais deste trabalho: a modelagem de distorção de sinais de guitarra elétrica e a modelagem do TurtleBot 3, ambas utilizando Redes Neurais Profundas. A seguir, detalha-se a visão geral do projeto, a motivação para o uso dessas arquiteturas e os desafios enfrentados.

5.1.1 Visão Geral do Projeto

Este projeto tem como objetivo modelar dois sistemas dinâmicos utilizando Redes Neurais Profundas. Os dois sistemas modelados são:

- **Modelagem de Distorção de Sinais de Guitarra Elétrica:** Um pedal de distorção de guitarra foi utilizado para gerar os dados de entrada e saída, com o intuito de modelar o comportamento não linear do circuito de distorção.
- **Modelagem do TurtleBot 3:** A dinâmica do TurtleBot 3 foi modelada com base em dados de simulação de seu comportamento cinemático, coletados no ambiente Gazebo, buscando prever a trajetória do robô.

Para ambas as implementações, foram utilizadas arquiteturas de Redes Neurais Profundas. No caso da modelagem de distorção de sinais de guitarra, foram empregados modelos MLP e LSTM, permitindo a comparação entre uma abordagem estática e uma abordagem recorrente para capturar dependências temporais. Já na modelagem do TurtleBot 3, foram implementadas quatro arquiteturas: MLP, GRU, LSTM e xLSTM, possibilitando uma análise comparativa entre diferentes abordagens para prever trajetórias e velocidades do robô. O conjunto de dados para cada sistema foi pré-processado e dividido em treinamento, validação e teste para garantir a generalização dos modelos.

5.1.2 Motivação para o Uso de Redes Neurais Profundas

A escolha por Redes Neurais Profundas foi motivada pela necessidade de capturar comportamentos dinâmicos complexos que os modelos tradicionais, como ARX ou espaço de estados, não conseguem representar adequadamente. No caso da distorção de sinais de guitarra, o circuito apresenta uma forte não linearidade, o que inviabiliza a utilização

de modelos lineares. Da mesma forma, a cinemática do TurtleBot 3 envolve interações temporais que requerem o uso de modelos capazes de lidar com sequências de dados.

A motivação para o uso das arquiteturas específicas pode ser resumida da seguinte maneira:

- **MLP:** Utilizada para a modelagem estática do sistema, a MLP é adequada para capturar relações não lineares complexas entre entrada e saída, sem a necessidade de lidar com dependências temporais. Foi empregada em ambos os sistemas como referência para comparação com modelos recorrentes.
- **LSTM:** Utilizada tanto no pedal de distorção quanto no TurtleBot 3, a LSTM foi escolhida por sua capacidade de capturar dependências temporais de longo prazo, sendo ideal para modelar variações temporais dos sinais da guitarra e os movimentos do robô.
- **GRU:** Implementada no TurtleBot 3, a GRU é uma alternativa simplificada à LSTM, possuindo menos parâmetros e, teoricamente, permitindo um treinamento mais eficiente sem comprometer significativamente o desempenho.
- **xLSTM:** Também aplicada ao TurtleBot 3, a xLSTM representa um avanço na modelagem de sistemas dinâmicos complexos, combinando vantagens das arquiteturas LSTM tradicionais com maior capacidade de aprendizado de padrões temporais complexos.

5.1.3 Desafios e Objetivos Específicos

Diversos desafios surgiram durante o desenvolvimento deste projeto. Para a modelagem de distorção de guitarra, o principal desafio foi lidar com a alta não linearidade e a ampla faixa de frequências dos sinais de entrada e saída. O circuito de distorção gera harmônicos complexos e saturação, o que exige um modelo capaz de capturar tais fenômenos. Outro ponto crítico foi a quantidade limitada de dados experimentais, o que exigiu técnicas de regularização no treinamento da rede neural.

No caso do TurtleBot 3, o desafio principal foi a simulação precisa da dinâmica de movimento e a coleta de dados relevantes para o treinamento. Além disso, garantir que o modelo fosse capaz de generalizar bem para novos cenários de movimentação do robô foi uma preocupação significativa.

Os objetivos específicos para cada sistema podem ser resumidos da seguinte forma:

- **Distorção de Guitarra:**
  - Modelar a resposta do circuito de distorção para diferentes ganhos de entrada.

  - Avaliar a capacidade da MLP e da LSTM de capturar as características não lineares.

- **TurtleBot 3:**

  - Modelar a trajetória do TurtleBot 3 com base nas velocidades lineares e angulares.
  - Avaliar a precisão da MLP, GRU, LSTM e xLSTM na previsão da movimentação futura do robô.
  - Comparar o desempenho das diferentes arquiteturas em termos de erro de predição e capacidade de generalização.

Cada um desses desafios foi abordado utilizando as técnicas descritas nos capítulos seguintes, com foco na seleção de hiperparâmetros adequados, regularização dos modelos e avaliação rigorosa dos resultados obtidos.

5.2 Modelagem de Sinal de Distorção de Guitarra Elétrica

Neste ponto do trabalho, apresenta-se a modelagem de um circuito de distorção de guitarra elétrica utilizando redes neurais profundas. O processo envolveu a coleta e pré-processamento de dados, a escolha de arquiteturas adequadas (MLP e LSTM), e a análise dos resultados obtidos com essas arquiteturas.

5.2.1 Descrição do Sistema de Distorção

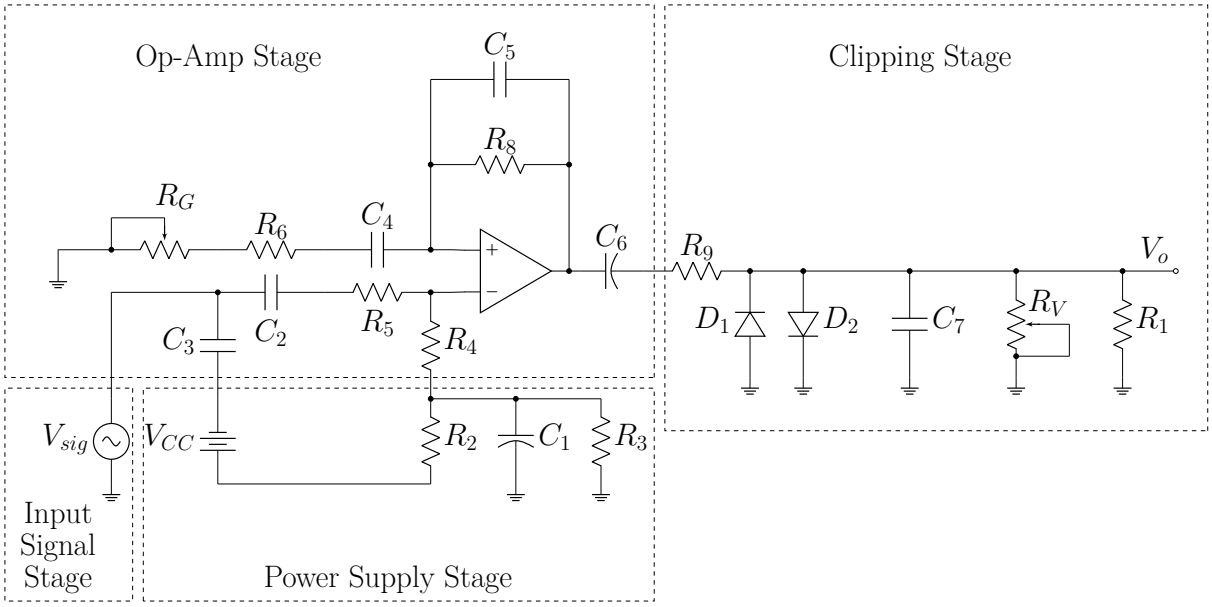
O circuito de distorção utilizado foi simulado com base em um pedal de distorção de guitarra elétrica típico. Esse circuito foi projetado para alterar o sinal de entrada de forma não linear, gerando harmônicos que resultam no som característico da distorção.

5.2.1.1 Especificação do Circuito de Distorção

A entrada do circuito foi um sinal de áudio gerado por uma guitarra elétrica, variando em amplitude e frequência. O circuito pedal foi dividido em 4 partes. A primeira dessas partes é o estágio de entrada de sinal (do inglês *input signal stage*). O sinal escolhido estava com uma tensão alta, mais alta do que costuma ser fornecido de entrada por uma guitarra. Por tal motivo, foi colocado uma fonte dependente com ganho de 0.2, valor esse que trouxe o sinal de entrada para um valor mais próximo da realidade e, na montagem do circuito analógico, o sinal da guitarra deve ser aplicado no nó sinalizado pela palavra *input*. A segunda parte do pedal é o estágio de alimentação de energia (do inglês *power supply stage*), o uso dos altos valores de resistência contribuem para uma alta impedância de entrada do circuito além de uma alta impedância para o terra do curto-circuito virtual.

O capacitor tem o papel de eliminar a variação residual conhecida como ondulação (do inglês *ripple*), que nada mais é do que um sinal AC indesejado. A Fig. 5.1 mostra o diagrama elétrico do circuito do pedal.

Figura 5.1 – Diagrama elétrico do pedal.



Fonte: Próprio autor.

Ademais, a terceira parte do pedal é o estágio de amplificação do Amp Op (do inglês *Op Amp Amplifier Stage*, ou somente *Op Amp Stage* como referido no esquemático). O amplificador se encontra em uma configuração em que gera um sinal de saída como relação dos resistores R8, R6 e RG, sendo esse último o potenciômetro que permite ao usuário controlar o ganho. Além disso, nesse estágio, há o capacitor C3 cuja função é evitar ruído de rádio frequência bem como ajudar em descargas eletroestáticas e oscilações. O capacitor C6 remove tensão DC ao próximo estágio. Os outros capacitores contribuem para a resposta em frequência com pico em torno de 1,5 KHz, característica comum em outros pedais e que ajuda a guitarra a ter destaque em relação a outros instrumentos na música ao ter maior ganho na faixa de frequência audível para os seres humanos.

O último estágio é o de recorte (do inglês *Clipping Stage*). Nele, o resistor R9 limita a corrente que chega aos diodos, protegendo tais. Os diodos possuem uma tensão de saturação e, pela disposição deles, saturam com tensão menores que o negativo da tensão de saturação e valores maiores que o positivo dessa tensão. Com isso, o sinal é cortado além desses limites, dando a sonoridade da distorção. Quanto mais abrupto o corte, mais distorcido o som fica. Finalmente, há um potenciômetro para regulagem do volume bem como a impedância de saída do pedal, essa última que não está presente na construção do pedal mas é importante para a simulação por simular a carga com máxima tranferência.

5.2.1.2 Características Não Lineares do Circuito

A não linearidade do circuito de distorção é gerada principalmente pela saturação dos amplificadores operacionais e pelo uso de diodos no estágio de corte do sinal. Isso resulta na geração de harmônicos e clipping no sinal, que é modelado pelas redes neurais. As equações que governam a não linearidade podem ser descritas como:

$$V_{\text{o}} = \begin{cases} V_{\text{sig}} & \text{se } |V_{\text{sig}}| < V_{\text{th}} \\ V_{\text{th}} \cdot \text{sign}(V_{\text{sig}}) & \text{se } |V_{\text{sig}}| \geq V_{\text{th}} \end{cases} \tag{5.1}$$

Onde V_{sig} é o sinal de entrada, V_{o} é o sinal de saída, e V_{th} é o nível de tensão de *threshold* dos diodos D1 e D2.

5.2.2 Coleta e Preparação dos Dados

Os dados de entrada e saída foram obtidos diretamente do circuito de distorção simulado utilizando o *software* de simulação de circuitos eletrônicos LTspice[1].

Na fase inicial de aquisição e preparação dos dados, adotou-se uma abordagem meticulosa para garantir a qualidade das informações coletadas. O sinal de uma guitarra genérica, sem efeitos, foi escolhido como entrada principal. Após a definição dos parâmetros apropriados, o *software* gerou o sinal de saída em formato de áudio, fornecendo assim os dados cruciais para as etapas subsequentes da pesquisa.

Os áudios da guitarra elétrica, tanto com quanto sem distorção, foram registrados a uma taxa de amostragem de 48 kHz, com duração de 30 segundos, capturando nuances sutis. A distorção foi introduzida pela aplicação de ganhos específicos, resultando em pares de áudios correspondentes: um com distorção gerada por resistores de $0k\Omega$, $10k\Omega$, $30k\Omega$, $90k\Omega$ e $200k\Omega$. Esse procedimento permitiu a comparação direta dos efeitos da distorção na saída da MLP em relação ao áudio não distorcido.

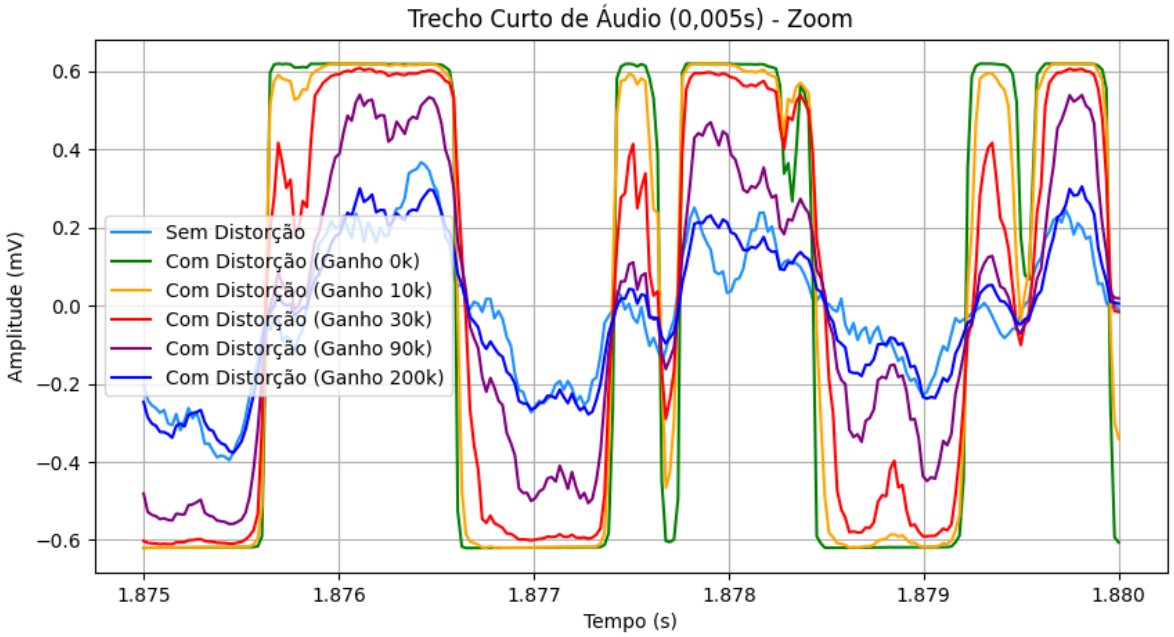
Cada áudio foi transformado em um tensor, preservando sua estrutura temporal e possibilitando uma manipulação eficiente dos dados durante as fases de treinamento e avaliação da MLP. Cada par de áudios correspondentes foi utilizado como entrada e rótulo durante o treinamento da rede. Essa abordagem refinada na preparação dos dados assegurou que a MLP fosse treinada e avaliada sob condições realistas, refletindo com precisão os desafios impostos pelas distorções presentes nos sinais da guitarra elétrica.

5.2.3 Arquitetura da MLP

A arquitetura da MLP utilizada neste projeto foi projetada para realizar a modelagem do comportamento dinâmico do sistema com base nos dados de entrada. Esta rede

[1] <https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html>

Figura 5.2 – Comparação entre os áudios com distorção e o áudio sem distorção com tempo de 0,005s.



Fonte: Próprio autor.

consiste em uma camada de entrada, duas camadas ocultas, cada uma seguida por uma função de ativação ReLU, e uma camada de saída. O objetivo da MLP é aproximar a função de distorção não linear aplicada ao sinal de entrada do sistema.

A camada de entrada da MLP possui 1.440.000 neurônios, correspondendo ao número de features do sinal de entrada, enquanto a camada de saída também tem 1.440.000 neurônios, representando as previsões do sinal de saída processado. As duas camadas ocultas possuem 32 neurônios cada uma.

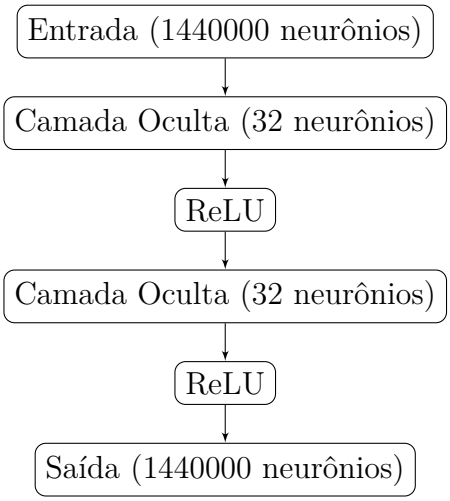
A Figura 5.3 ilustra o diagrama que representa a arquitetura da rede MLP utilizada.

5.2.4 Arquitetura da LSTM

A arquitetura da RNN com LSTM utilizada neste projeto foi projetada para modelar o comportamento dinâmico temporal dos dados. A rede consiste em duas camadas LSTM, seguidas por uma camada totalmente conectada (Linear), cujo objetivo é realizar a predição final baseada nas informações extraídas pelas camadas LSTM. A estrutura da rede permite capturar dependências temporais de longa duração, que são essenciais para a modelagem de sinais dinâmicos.

A primeira parte da rede é composta por duas camadas LSTM, cada uma com 32 neurônios, que processam sequências de dados de entrada de comprimento variável. A

Figura 5.3 – Arquitetura da MLP utilizada no projeto.

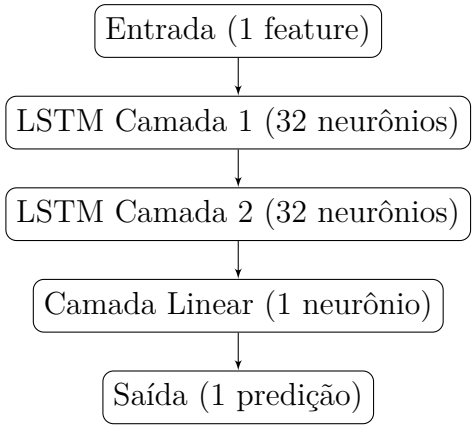


Fonte: Próprio autor.

segunda parte é uma camada totalmente conectada com um único neurônio de saída.

A Figura 5.4 ilustra o diagrama da arquitetura da rede LSTM utilizada.

Figura 5.4 – Arquitetura da RNN (LSTM) utilizada no projeto.



Fonte: Próprio autor.

5.2.5 Métodos de Análise de Desempenho

Para avaliar a eficiência do modelo implementado neste projeto, foram utilizados três métodos principais de análise de desempenho: a FDA, o EQM e o KS. Estes métodos permitem verificar a precisão do modelo e a qualidade da predição dos sinais processados.

5.2.5.1 Função de Distribuição Acumulada (FDA)

A FDA é uma função que descreve a probabilidade de uma variável aleatória ser menor ou igual a um determinado valor. No contexto deste projeto, a FDA foi utilizada

para comparar a distribuição dos valores do sinal de saída gerado pela RNN com o sinal original com distorção.

Matematicamente, a FDA pode ser representada como:

$$F(x) = P(X \leq x)$$

onde $F(x)$ é a FDA e $P(X \leq x)$ representa a probabilidade da variável aleatória X ser menor ou igual a x.

A comparação entre as FDAs dos sinais pode indicar a similaridade entre o sinal de saída gerado pelo modelo e o sinal com distorção. Se as curvas da FDA dos dois sinais forem próximas, o modelo terá um bom desempenho.

5.2.5.2 Erro Quadrático Médio (EQM)

O EQM é uma métrica amplamente utilizada para medir a diferença entre os valores previstos pelo modelo e os valores reais. Ele é definido pela média dos quadrados das diferenças entre os valores reais e os previstos:

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

onde y_i são os valores reais, $\hat{y}_i$ são os valores previstos pelo modelo e n é o número de observações.

Um valor de EQM baixo indica que o modelo está realizando boas predições. Quanto menor o EQM, melhor o desempenho do modelo.

5.2.5.3 Teste de Kolmogorov-Smirnov (KS)

O KS é um teste estatístico que verifica a hipótese nula de que duas amostras provêm da mesma distribuição. Ele é utilizado neste projeto para comparar a distribuição do sinal original com distorção e o sinal de saída gerado pela RNN.

O valor do teste KS é definido como a maior diferença entre as funções de distribuição acumulada das duas amostras:

$$D_{n,m} = \sup_x |F_1(x) - F_2(x)|$$

onde $F_1(x)$ e $F_2(x)$ são as FDAs das duas amostras.

Um valor de estatística KS ($D_{n,m}$) próximo de 0 indica que as funções de distribuição acumulada (FDAs) das duas amostras são muito semelhantes ao longo de todo

o domínio dos dados. Isso significa que a distribuição do sinal gerado pelo modelo se aproxima da distribuição do sinal original com distorção.

O p-valor representa a probabilidade de observar uma diferença tão grande (ou maior) entre as distribuições apenas por acaso, assumindo que a hipótese nula seja verdadeira (ou seja, que as distribuições sejam iguais).

- Se o p-valor for alto (tipicamente maior que 0.05), não há evidências estatísticas suficientes para rejeitar a hipótese nula, sugerindo que os sinais possuem distribuições semelhantes.
- Se o p-valor for baixo (tipicamente menor que 0.05), rejeita-se a hipótese nula, indicando que as distribuições são significativamente diferentes.

Dessa forma, um valor KS baixo e um p-valor alto indicam que o modelo conseguiu gerar um sinal cuja distribuição estatística se aproxima da distribuição real do sinal com distorção.

5.2.5.4 Análise dos Resultados

A eficiência do modelo pode ser analisada a partir dos resultados obtidos pelos três métodos descritos.

- Na FDA, se as curvas das distribuições acumuladas dos dois sinais forem semelhantes, o modelo estará reproduzindo com precisão o comportamento dinâmico do sinal.
- No EQM, quanto menor o valor obtido, mais preciso é o modelo em termos de aproximação dos valores previstos.
- No Teste KS, se o valor da estatística KS for baixo e o p-valor elevado, pode-se concluir que as distribuições dos sinais são similares, indicando um bom desempenho do modelo.

Essas métricas fornecem uma avaliação robusta da performance do modelo na tarefa de prever o comportamento dinâmico dos sinais processados.

5.3 Modelagem do TurtleBot 3

Nesta seção, será abordado o processo de desenvolvimento e implementação de RNPs para modelar o comportamento dinâmico do TurtleBot 3. O projeto tem como objetivo capturar e reproduzir as dinâmicas não lineares do robô móvel utilizando técnicas

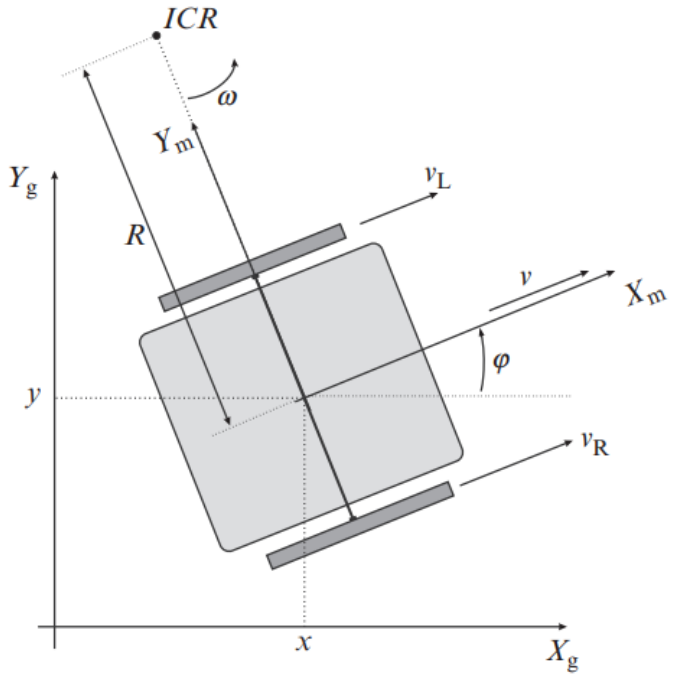
de aprendizado de máquina, explorando diferentes arquiteturas de redes neurais. Para isso, foram implementados modelos MLP, GRU, LSTM e xLSTM, permitindo uma análise comparativa entre abordagens distintas de modelagem.

Os dados utilizados para o treinamento dos modelos foram coletados a partir de simulações do TurtleBot 3, com o objetivo de prever seu comportamento cinemático, incluindo suas velocidades lineares e angulares, além da trajetória percorrida. A implementação dessas arquiteturas possibilitou investigar a eficiência de cada modelo na captura das dinâmicas temporais e não lineares do sistema, oferecendo *insights* sobre a adequação de redes neurais profundas para a modelagem de sistemas robóticos complexos.

5.3.1 Descrição do Sistema do TurtleBot 3

O TurtleBot 3 é um robô com direção diferencial, como mostra a Figura 5.5, o que significa que seu movimento é controlado pelas velocidades angulares independentes de suas duas rodas motrizes. O estudo do sistema dinâmico desse robô é crucial para entender e prever seu comportamento em diferentes condições de operação, permitindo a modelagem e controle precisos do robô em ambientes simulados e reais.

Figura 5.5 – Diagrama cinemático do TurtleBot 3.



Fonte: (KLANCAR et al., 2017).

5.3.1.1 Cinemática Direta

A cinemática direta permite determinar a posição e a orientação do TurtleBot 3 com base nas velocidades angulares das rodas. Para o robô, a posição (x, y) e a orientação

θ são calculadas a partir das velocidades v_l (roda esquerda) e v_r (roda direita), conforme descrito abaixo.

As variáveis envolvidas são:

- v_l: Velocidade angular da roda esquerda.
- v_r: Velocidade angular da roda direita.
- L: Distância entre as rodas.
- R: Raio das rodas.
- v: Velocidade linear do robô.
- ω: Velocidade angular do robô.

As equações a seguir descrevem a cinemática direta são as seguintes:

$$v = \frac{R}{2}(v_r + v_l) \tag{5.2}$$

$$\omega = \frac{R}{L}(v_r - v_l) \tag{5.3}$$

A pose do robô é atualizada com base nas equações de integração abaixo, onde Δt representa o intervalo de tempo entre cada atualização:

$$x(t+1) = x(t) + v\cos(\theta)\Delta t \tag{5.4}$$

$$y(t+1) = y(t) + v\sin(\theta)\Delta t \tag{5.5}$$

$$\theta(t+1) = \theta(t) + \omega\Delta t \tag{5.6}$$

Essas equações fornecem a base para calcular a trajetória do TurtleBot 3 em termos de sua posição e orientação, dado o controle sobre as velocidades angulares das rodas.

5.3.1.2 Cinemática Inversa

A cinemática inversa, por outro lado, envolve calcular as velocidades angulares das rodas necessárias para atingir uma velocidade linear v e angular ω desejadas. Utilizando as seguintes equações, é possível determinar as velocidades v_l e v_r:

$$v_l = \frac{v - \frac{L}{2}\omega}{R} \tag{5.7}$$

$$v_r = \frac{v + \frac{L}{2}\omega}{R} \tag{5.8}$$

A cinemática inversa é fundamental para o desenvolvimento de sistemas de controle do robô, permitindo que ele siga uma trajetória planejada, ajustando as velocidades de suas rodas para manter a direção e a velocidade desejadas.

5.3.2 Coleta e Preparação dos Dados

Os dados foram coletados em um ambiente virtual utilizando o simulador Gazebo junto com o robô TurtleBot 3, através da ferramenta de desenvolvimento ROS[2]. Essa configuração é bastante popular na comunidade científica devido à sua capacidade extensiva de emular cenários reais com alta precisão.

Primeiramente, foram realizadas todas as configurações necessárias para simular o TurtleBot 3 no simulador Gazebo[3], passos que podem ser seguidos consultando a documentação do simulador. Esse robô foi escolhido porque, além de ser de baixo custo com hardware e software de código aberto, é amplamente utilizado nas áreas de pesquisa e educação em robótica devido às suas capacidades de navegação e sensoriamento.

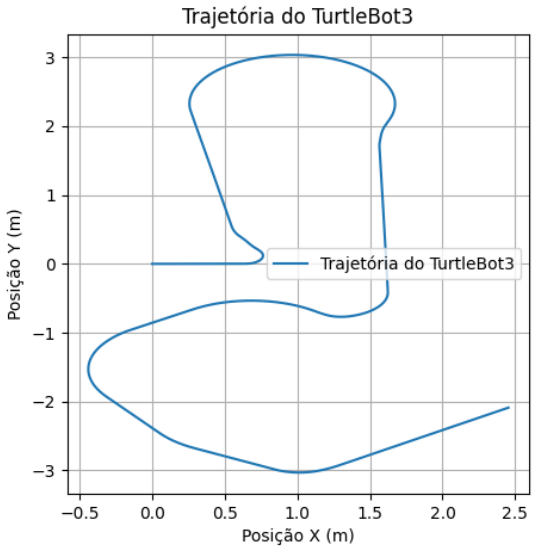
Um *script* em Python foi desenvolvido utilizando ROS2 para a coleta de dados, criando nós e facilitando a comunicação entre os diferentes sistemas que compõem o robô. Este *script*, denominado `data_collector.py`, foi implementado como um assinante do tópico `/odom`, responsável por fornecer dados espaciais do robô através da odometria. Através das mensagens `nav_msgs/msg/Odometry`, o tópico é capaz de fornecer dados como a posição e orientação do robô, bem como suas velocidades linear e angular.

Durante a simulação, o TurtleBot 3 foi controlado manualmente utilizando o pacote `teleop_keyboard` para realizar diferentes movimentos, gerando um conjunto abrangente de dados de movimentação. O script `data_collector.py` foi configurado para salvar os dados coletados em tempo real em um arquivo de valores separados por vírgula (CSV) nomeado `data.csv`. Entre os dados registrados estão: o timestamp, coordenadas de posição (x, y, z), orientações em formato de *quaternion* (x, y, z, w), e velocidades linear e angular.

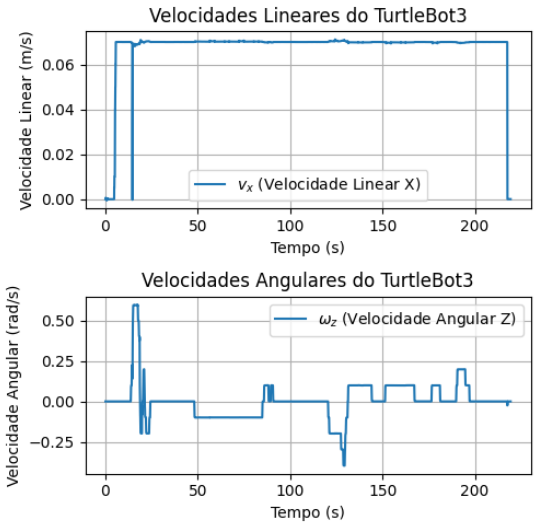
A Figura 5.6 fornece uma visão abrangente dos resultados da simulação do TurtleBot 3, da trajetória utilizada para treinamento do modelo.

[2] <https://docs.ros.org/en/foxy/index.html>
[3] <https://gazebosim.org/docs/latest/getstarted/>

Figura 5.6 – Gráficos da simulação do TurtleBot 3.



(a) Gráfico da trajetória realizada pelo TurtleBot 3 durante a simulação.



(b) Gráfico das velocidades angular e linear realizadas pelo TurtleBot 3 durante a simulação.

Fonte: Próprio autor.

A trajetória do TurtleBot 3 na simulação é mostrada no gráfico exibido na Figura 5.6a. Esse gráfico é crucial para compreender as ações e a localização do robô no estudo, proporcionando uma visão detalhada e clara de sua trajetória.

Além disso, a Figura 5.6b exibe os gráficos das velocidades angular e linear do robô para o mesmo intervalo de tempo. Esses gráficos são necessários para avaliar as alterações nas velocidades e orientações do TurtleBot 3, fornecendo informações cruciais sobre seu funcionamento e ações.

A incorporação dos dados de trajetória e velocidade durante o treinamento da rede MLP foi essencial para a melhoria do modelo. Ter acesso a esses dados permitiu ajustar o modelo para que ele refletisse com precisão os movimentos reais do TurtleBot 3.

5.3.3 Arquitetura da MLP

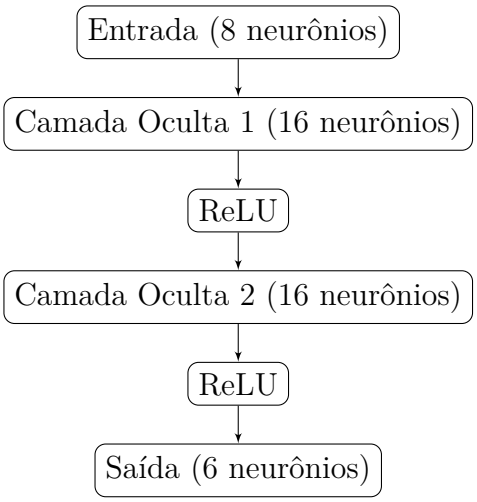
A arquitetura da MLP utilizada neste projeto foi projetada para modelar o comportamento dinâmico do TurtleBot 3 com base nos dados de entrada da simulação. Esta rede neural possui uma camada de entrada, duas camadas ocultas e uma camada de saída. O objetivo da MLP é aprender a função de mapeamento entre as entradas do robô e suas saídas desejadas.

A camada de entrada da MLP possui 8 neurônios, representando as seguintes características do TurtleBot 3: posição x, posição y, orientação x, orientação y, orientação z, orientação w, velocidade linear x e velocidade angular z. As duas camadas ocultas têm

16 neurônios cada, utilizando a função de ativação ReLU para capturar as complexidades das relações não lineares. A camada de saída possui 6 neurônios, fornecendo previsões para a posição e orientação do robô.

A Figura 5.7 ilustra o diagrama da arquitetura da rede MLP utilizada neste projeto.

Figura 5.7 – Arquitetura da MLP utilizada no projeto de modelagem do TurtleBot 3.



Fonte: Próprio autor.

5.3.4 Arquitetura da GRU

A arquitetura da GRU utilizada neste projeto foi projetada para modelar o comportamento dinâmico do TurtleBot 3, explorando sua capacidade de capturar dependências temporais e dinâmicas não lineares. A rede possui uma camada de entrada, uma camada recorrente GRU e uma camada de saída.

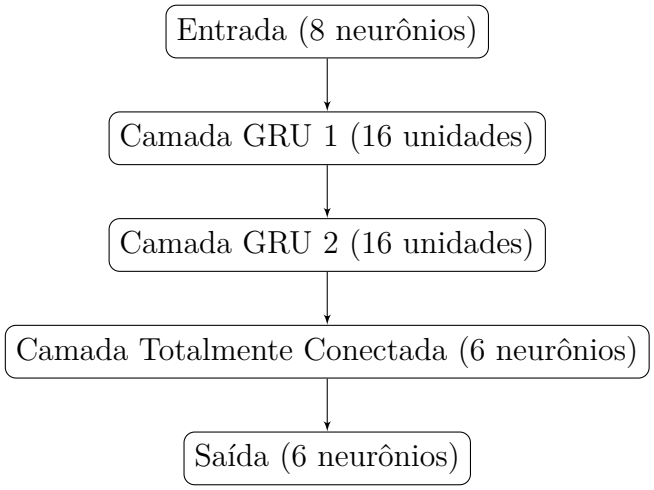
A camada de entrada da GRU possui 8 neurônios, correspondendo às características do TurtleBot 3: posição x, posição y, orientação x, orientação y, orientação z, orientação w, velocidade linear x e velocidade angular z. A camada oculta contém 32 unidades GRU, utilizando a ativação *tanh* e mecanismos de portas para reter informações relevantes da sequência temporal. A camada de saída possui 6 neurônios, responsáveis por prever a posição e orientação do robô.

A Figura 5.8 apresenta o diagrama da arquitetura da GRU utilizada neste projeto.

5.3.5 Arquitetura da LSTM

A arquitetura da LSTM foi utilizada para capturar as dinâmicas temporais do TurtleBot 3, sendo capaz de aprender sequências de movimento do robô ao longo do

Figura 5.8 – Arquitetura da GRU utilizada no projeto de modelagem do TurtleBot 3.



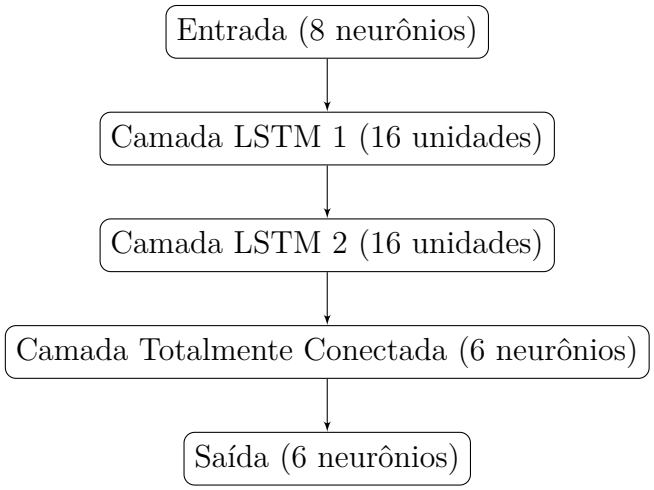
Fonte: Próprio autor.

tempo. A rede é composta por uma camada de entrada, uma camada recorrente LSTM e uma camada de saída.

A camada de entrada possui 8 neurônios, representando as mesmas características utilizadas na GRU. A camada oculta contém 32 unidades LSTM, empregando portas de entrada, esquecimento e saída para controlar o fluxo de informações ao longo da sequência temporal. A camada de saída possui 6 neurônios, responsáveis pela previsão da posição e orientação do robô.

A Figura 5.9 ilustra a estrutura da LSTM utilizada neste projeto.

Figura 5.9 – Arquitetura da LSTM utilizada no projeto de modelagem do TurtleBot 3.



Fonte: Próprio autor.

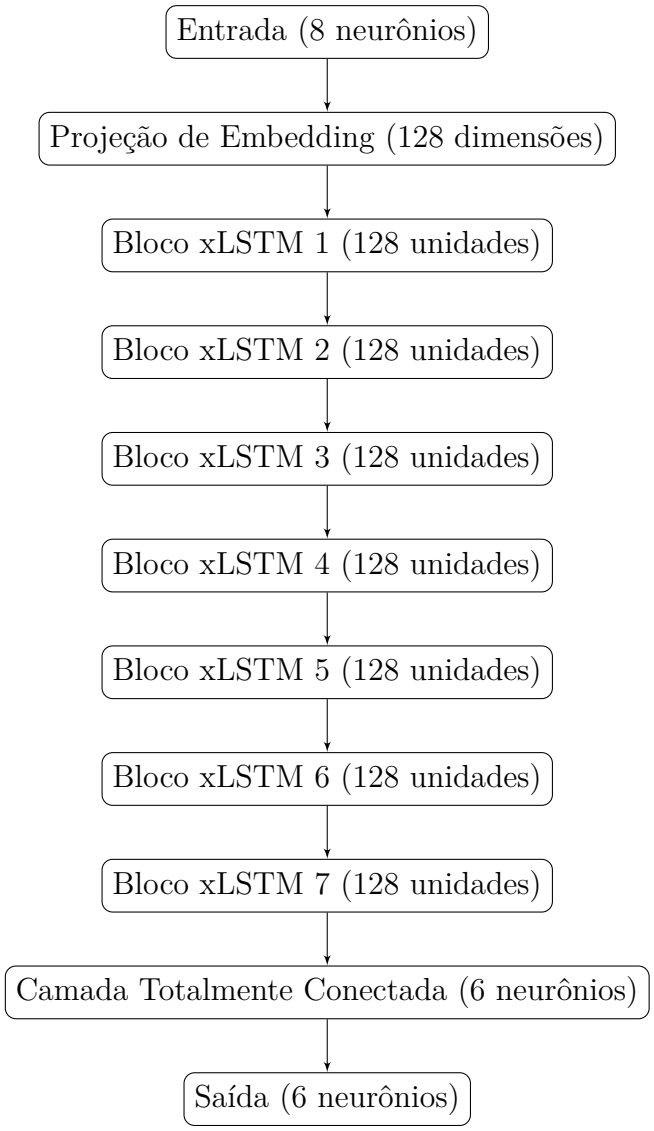
5.3.6 Arquitetura da xLSTM

A xLSTM é uma extensão da LSTM convencional, oferecendo maior estabilidade e eficiência no aprendizado de padrões temporais complexos. Essa arquitetura foi empregada para melhorar a capacidade de modelagem das dinâmicas não lineares do TurtleBot 3.

A rede possui uma camada de entrada com 8 neurônios, seguida por uma camada oculta composta por 32 unidades xLSTM. A xLSTM combina componentes de memória adicionais e mecanismos de normalização interna para aprimorar a retenção de informações ao longo da sequência temporal. A camada de saída possui 6 neurônios, responsáveis por prever a posição e orientação do robô.

A Figura 5.10 apresenta o diagrama da arquitetura da xLSTM utilizada neste projeto.

Figura 5.10 – Arquitetura da xLSTM utilizada no projeto de modelagem do TurtleBot 3.



Fonte: Próprio autor.

5.3.7 Métodos de Análise de Desempenho

Para avaliar o desempenho do modelo de rede neural MLP utilizado na modelagem do TurtleBot 3, foram empregadas as seguintes métricas de desempenho:

5.3.7.1 Erro Quadrático Médio (EQM)

O EQM é uma métrica que mede a média dos quadrados das diferenças entre os valores previstos e os valores reais. Foi utilizada no projeto anterior e apresentada na Seção 5.2.5.2. O EQM penaliza erros grandes mais severamente do que erros pequenos devido à sua natureza quadrática.

5.3.7.2 Raiz do Erro Quadrático Médio (REQM)

A REQM é a raiz quadrada do EQM e fornece uma medida da magnitude dos erros em unidades de saída do modelo. É calculada por:

$$\text{REQM} = \sqrt{\text{EQM}}$$

O REQM é útil para interpretar o desempenho do modelo em termos das unidades originais dos dados. Valores mais baixos indicam um melhor ajuste do modelo aos dados.

5.3.7.3 Erro Absoluto Médio (EAM)

O EAM calcula a média dos valores absolutos das diferenças entre os valores reais e previstos. A fórmula para o EAM é:

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

O EAM fornece uma medida direta da média dos erros e é menos sensível a grandes erros em comparação com o MSE. Valores menores indicam um melhor desempenho do modelo.

5.3.7.4 Erro Absoluto Mediano (MdAE)

O Erro Absoluto Mediano (MdAE, do inglês *Median Absolute Error*) é a mediana dos erros absolutos entre os valores reais e previstos. É calculado por:

$$\text{MdAE} = \text{mediana}(|y_i - \hat{y}_i|)$$

Esta métrica é robusta a *outliers* e fornece uma medida da tendência central dos erros absolutos.

5.3.7.5 Coeficiente de Determinação (R^2)

O Coeficiente de Determinação (R^2) mede a proporção da variância nos dados que é explicada pelo modelo. É calculado pela fórmula:

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

onde $\bar{y}$ é a média dos valores reais. O valor de R^2 varia de 0 a 1, onde valores próximos de 1 indicam que o modelo explica uma alta proporção da variância nos dados.

5.3.7.6 Análise dos Resultados

A interpretação dos resultados das métricas é essencial para avaliar a eficiência do modelo:

- *EQM e REQM:* Valores baixos de MSE e RMSE indicam que o modelo tem um bom ajuste aos dados, com erros menores em média. No entanto, o EQM e o REQM podem ser afetados por outliers, então é importante considerar outras métricas também.
- *EAM:* Um EAM baixo indica que o modelo, em média, está cometendo poucos erros absolutos. Essa métrica é mais robusta a outliers do que o EQM e o REQM.
- *MdAE:* O MdAE fornece uma visão sobre a mediana dos erros absolutos, sendo útil para identificar a presença de outliers e a dispersão dos erros. Valores baixos indicam um bom desempenho do modelo sem grandes erros absolutos.
- R^2: Um valor de R^2 próximo de 1 indica que o modelo explica bem a variância dos dados. Valores baixos sugerem que o modelo não está capturando a variância dos dados de forma eficaz.

Essas métricas devem ser analisadas em conjunto para obter uma avaliação completa do desempenho do modelo. Enquanto o EQM e o REQM fornecem informações sobre a magnitude dos erros, o EAM e o MdAE ajudam a entender o impacto dos erros em termos absolutos. O R^2 oferece uma visão sobre a capacidade do modelo de explicar a variância dos dados. Juntas, essas métricas ajudam a determinar se o modelo MLP é eficiente e adequado para a modelagem do comportamento do TurtleBot 3.

Parte II

Resultados

6 Resultados da Modelagem de Distorção de Sinal de Guitarra Elétrica

6.1 Desempenho da MLP

O modelo de rede neural MLP foi treinado utilizando diferentes configurações de ganho para a distorção do sinal de guitarra elétrica. As métricas de desempenho, como a Função de Perda, o Teste KS, o EQM e a FDA, foram utilizadas para avaliar a precisão do modelo.

6.1.1 Gráfico de Função de Perda e Acurácia por Época

A Figura 6.1 apresenta a evolução da função de perda e da acurácia durante o treinamento da MLP ao longo das épocas para diferentes valores de resistência.

A função de perda apresenta uma tendência de convergência ao longo das épocas, indicando que a MLP está aprendendo o mapeamento entre o sinal de entrada e sua versão distorcida. Além disso, observa-se que a acurácia melhora gradativamente até estabilizar, o que sugere que o modelo conseguiu se ajustar bem aos dados de treinamento.

Um aspecto relevante é a variação observada na função de perda e na acurácia nos momentos em que o *learning rate* foi resetado a cada 100 épocas. Esse comportamento é particularmente evidente nos ganhos mais elevados (resistências menores, como 0k e 10k), onde há mudanças abruptas nos gráficos de perda e acurácia. Esse ajuste foi essencial para evitar que o modelo ficasse preso em mínimos locais e garantir uma melhor adaptação à complexidade do sinal. O impacto dessa estratégia reforça a importância do ajuste dinâmico da taxa de aprendizado para otimizar o treinamento da rede neural, especialmente em cenários onde há alta não linearidade na resposta do sistema.

6.1.2 Análise da FDA

A Função de Distribuição Acumulada (FDA) foi utilizada para verificar a aderência do modelo MLP às distribuições dos dados de saída. A Figura 6.2 ilustra a comparação da FDA entre os sinais de saída da rede e os sinais com distorção real do pedal, para diferentes resistências.

A análise da FDA revela que a aderência do modelo MLP às distribuições reais varia conforme o ganho aplicado ao sinal. Para resistências menores (0k e 10k), os sinais gerados pelo modelo apresentam uma discrepância mais pronunciada em relação ao sinal real do pedal, especialmente nos extremos da amplitude (picos positivo e negativo). Esse

comportamento indica que, em situações de alta distorção, o modelo tem maior dificuldade em capturar fielmente a resposta não linear do circuito.

Por outro lado, para resistências mais altas (90k e 200k), a FDA do modelo MLP mostra uma maior similaridade com a FDA do sinal distorcido real. Isso sugere que, à medida que a distorção se torna menos intensa, o modelo consegue capturar melhor a relação entre entrada e saída, aproximando-se da resposta esperada. Esse resultado reforça que, embora a MLP seja capaz de modelar a distorção de forma satisfatória, há limitações na sua capacidade de representar com precisão os extremos da resposta não linear para ganhos mais elevados.

6.1.3 Análise dos Resultados de Desempenho da MLP

A Tabela 6.1 resume os resultados obtidos para as métricas de desempenho (Estatística KS, Valor p (KS) e Erro Quadrático Médio - EQM) para diferentes valores de resistência, com base nos sinais gerados pelo modelo MLP.

Tabela 6.1 – Resultados da MLP para diferentes valores de resistência (0k, 10k, 30k, 90k, 200k)

Resistência (k)	**Estatística KS**	**Valor p (KS)**	**Erro Quadrático Médio (EQM)**
0k	0.1239	0.0000	0.0033
10k	0.0654	0.0000	0.00099
30k	0.0192	0.0000	0.000086
90k	0.00026	1.0000	3.93×10^{-8}
200k	0.00004	1.0000	1.30×10^{-9}

Fonte: Próprio autor.

Os resultados indicam que, conforme o valor da resistência aumenta, o erro quadrático médio (EQM) diminui de forma significativa. Para resistências mais altas (90k e 200k), a Estatística KS aproxima-se de zero e o valor-p atinge 1, o que sugere que o modelo reproduziu com alta precisão a distribuição do sinal com distorção, evidenciada pela similaridade entre as Funções de Distribuição Acumulada (FDA) dos sinais gerados e dos sinais reais.

Entretanto, para resistências menores (0k e 10k), verifica-se que, nos extremos da amplitude nos picos positivos e negativos os sinais gerados pela MLP divergem do sinal original. Essa discrepância se reflete em valores mais elevados de EQM e na estatística KS, indicando que o modelo tem maior dificuldade em capturar os comportamentos extremos do sinal de distorção quando o ganho é alto.

Adicionalmente, observa-se que a cada 100 épocas o learning rate foi resetado, provocando mudanças abruptas nos gráficos de função de perda e acurácia, especialmente para os ganhos com resistores menores. Essa estratégia de ajuste do learning rate foi

fundamental para evitar que o modelo ficasse preso em mínimos locais, contribuindo para uma convergência mais robusta ao longo do treinamento.

Em resumo, a análise dos resultados confirma que a MLP apresenta uma excelente capacidade de previsão da distorção aplicada ao sinal de guitarra, com desempenho que melhora significativamente com o aumento da resistência, onde a distorção é menos intensa. No entanto, para resistências mais baixas, embora a MLP reproduza satisfatoriamente o comportamento do sistema, os extremos da amplitude ainda apresentam discrepâncias notáveis em relação ao sinal original.

6.1.4 Comparação dos Sinais de Saída

A Figura 6.3 apresenta a comparação entre o sinal original (sem distorção), o sinal com distorção real e o sinal gerado pela MLP para diferentes valores de resistência. Pode-se notar que, para resistências maiores (como 90k e 200k), a MLP reproduz com alta fidelidade a forma do sinal distorcido, evidenciando uma boa aproximação das características gerais do circuito.

Entretanto, para resistências menores (0k e 10k), onde os ganhos são maiores e o efeito de distorção é mais pronunciado, observa-se que os picos positivos e negativos do sinal gerado pela MLP apresentam pequenas divergências em relação ao sinal real. Essas discrepâncias, especialmente perceptíveis nos extremos da amplitude, corroboram a análise realizada na Seção 6.1.2, que apontou dificuldades do modelo em capturar os extremos do sinal em situações de alta distorção.

Apesar dessas variações, o desempenho geral da MLP é robusto, conforme evidenciado pelos baixos valores de EQM e pela alta correspondência nos testes KS. Esses resultados reforçam a eficácia da MLP na modelagem da distorção, mesmo diante da complexidade não linear apresentada pelo circuito do pedal.

6.2 Desempenho da LSTM

6.2.1 Gráfico de Função de Perda e Acurácia por Época

A Figura 6.4 apresenta a evolução da função de perda e da acurácia ao longo das épocas durante o treinamento da LSTM. Note que, diferentemente do treinamento da MLP, neste caso não foi necessário realizar o *reset* do *learning rate*, pois o treinamento foi realizado em apenas 30 épocas. Essa configuração permitiu que a curva de perda e acurácia se comportasse de forma mais contínua e estável, refletindo a capacidade da LSTM de aprender os padrões temporais do sinal de distorção sem interrupções abruptas na taxa de aprendizado.

6.2.2 Análise da FDA

A Figura 6.5 apresenta a Função de Distribuição Acumulada (FDA) para a LSTM, comparando o sinal com distorção real e o sinal gerado pela rede para o resistor de 30k. Diferentemente do comportamento observado com a MLP, onde as discrepâncias eram mais evidentes apenas nos extremos da amplitude (picos positivo e negativo), o sinal produzido pela LSTM apresenta diferenças distribuídas por toda a faixa de valores.

Isso indica que, ao contrário da MLP, a LSTM tem dificuldade em capturar com precisão as características estatísticas completas do sinal de distorção. Em outras palavras, a FDA da LSTM se desvia consideravelmente do sinal real, não apenas nos extremos, mas ao longo de toda a distribuição. Essa divergência pode estar relacionada à natureza dos dados.

6.2.3 Análise Quantitativa dos Resultados da LSTM para Resistência de 30k

A Tabela 6.2 apresenta os valores obtidos com a LSTM para o resistor de 30k, que foi o único valor utilizado para o treinamento deste modelo. Nesta configuração, foram analisadas três métricas: a Estatística KS, o Valor p do teste KS e o Erro Quadrático Médio (EQM).

Tabela 6.2 – Resultados da LSTM para resistência de 30k

Estatística KS	**Valor p (KS)**	**Erro Quadrático Médio (EQM)**
0.2042	0.0000	0.0077

Fonte: Próprio autor.

A análise dos resultados indica que o valor da Estatística KS de 0.2042 é relativamente elevado, sugerindo que a distribuição do sinal gerado pela LSTM difere significativamente da distribuição do sinal com distorção real. O valor-p igual a 0 reforça que essa diferença é estatisticamente significativa. Além disso, o EQM de 0.0077, embora demonstre que a LSTM seja capaz de capturar a forma geral do sinal distorcido, indica uma discrepância maior entre o sinal previsto e o sinal real, em comparação com a MLP.

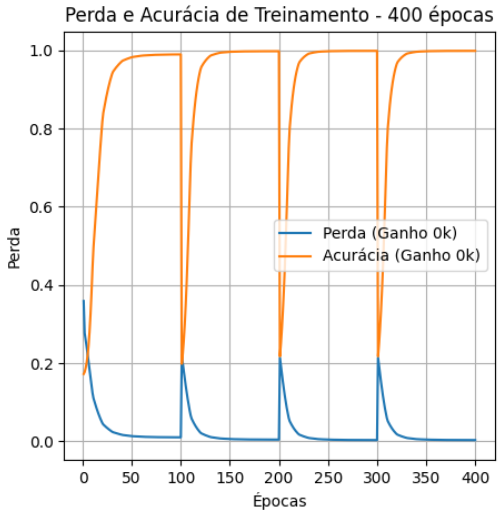
Esses achados sugerem que, ao contrário da MLP que consegue reproduzir com maior precisão os extremos da amplitude, especialmente para altos ganhos (resistores de 0k e 10k) a LSTM não apenas apresenta divergências nos picos, mas suas previsões se diferem em toda a distribuição dos valores.

Portanto, embora a LSTM capte de forma razoável as dependências temporais do sinal de distorção, seus resultados quantitativos apontam para uma performance inferior à da MLP neste cenário específico.

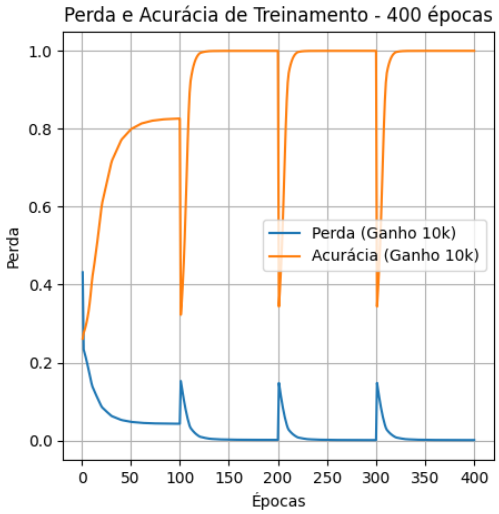
6.2.4 Comparação dos Sinais de Saída

A Figura 6.6 apresenta uma comparação detalhada entre o sinal original, o sinal com distorção real (obtido a partir do circuito elétrico do pedal) e o sinal gerado pela LSTM para o resistor de 30k. Observa-se que, embora a LSTM consiga captar a forma geral do sinal distorcido, há discrepâncias perceptíveis em toda a distribuição. Em particular, não apenas os picos positivos e negativos (extremos da amplitude) se diferem, mas também os valores intermediários, o que indica que a rede não reproduz com fidelidade todas as nuances do sinal real. Esse comportamento corrobora com as observações realizadas na Seção 6.2.2, reforçando que a LSTM, apesar de sua capacidade de modelar dependências temporais, apresenta limitações na reprodução precisa da distribuição completa do sinal de distorção.

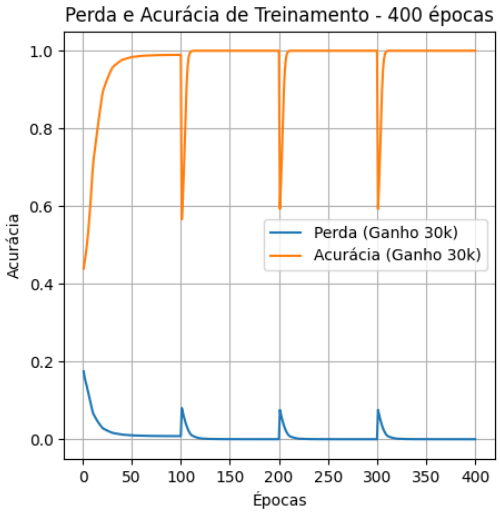
Figura 6.1 – Evolução da Função de Perda e Acurácia por Época para diferentes resistências na MLP.



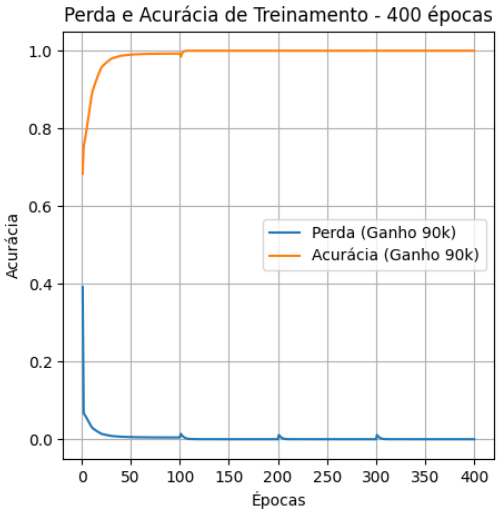
(a) Ganho com resistência de 0k.



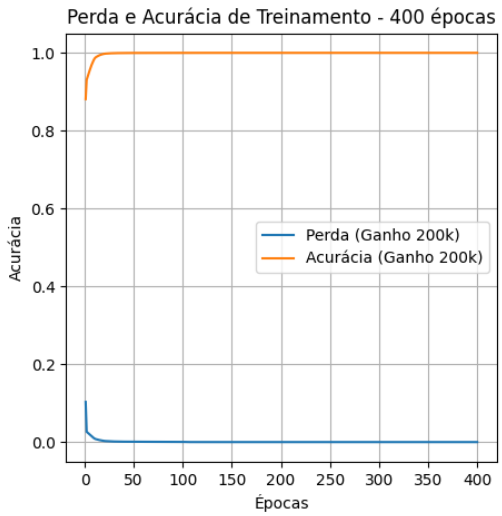
(b) Ganho com resistência de 10k.



(c) Ganho com resistência de 30k.



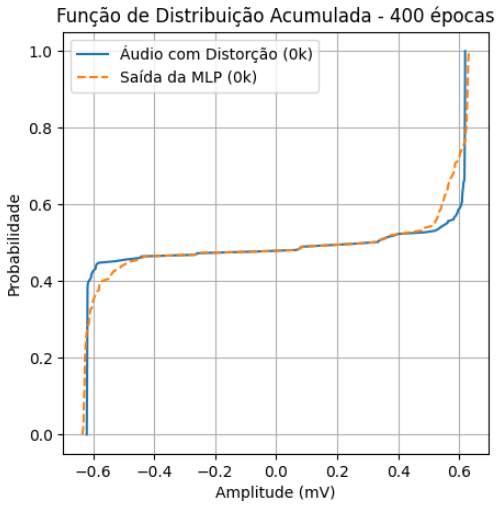
(d) Ganho com resistência de 90k.



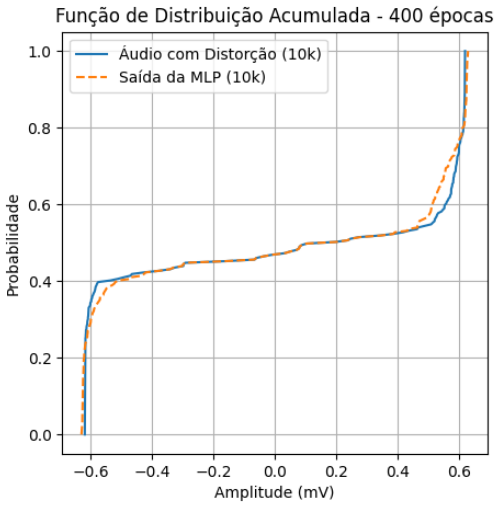
(e) Ganho com resistência de 200k.

Fonte: Próprio autor.

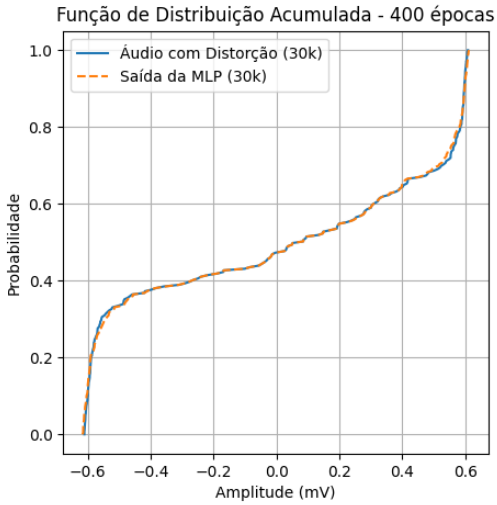
Figura 6.2 – Gráficos da Função de Distribuição Acumulada (FDA) para diferentes resistências na MLP.



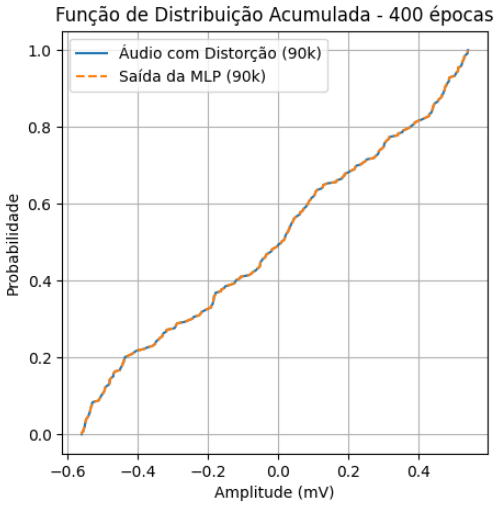
(a) Ganho com resistência de 0k.



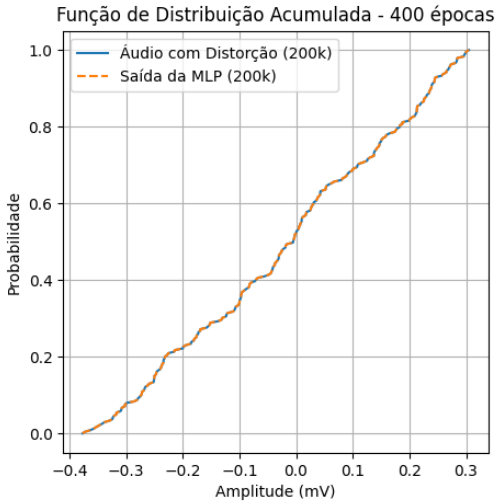
(b) Ganho com resistência de 10k.



(c) Ganho com resistência de 30k.



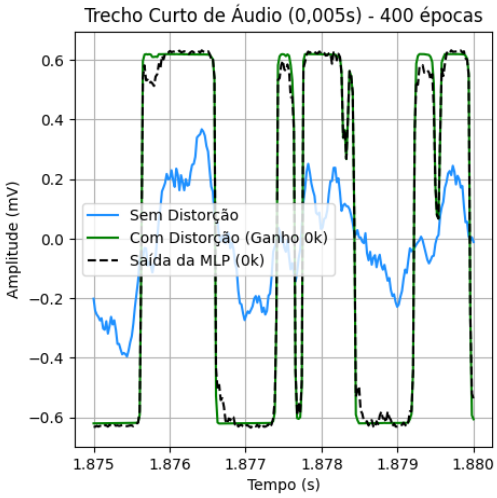
(d) Ganho com resistência de 90k.



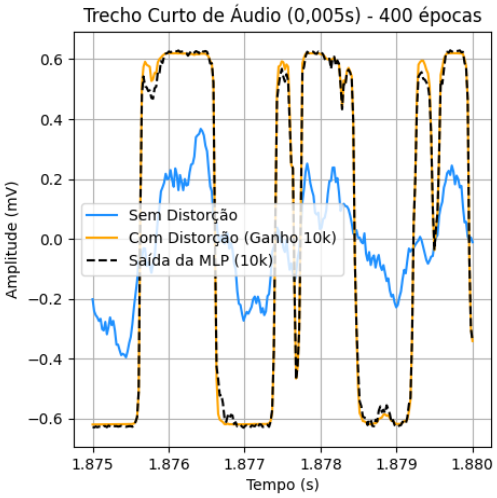
(e) Ganho com resistência de 200k.

Fonte: Próprio autor.

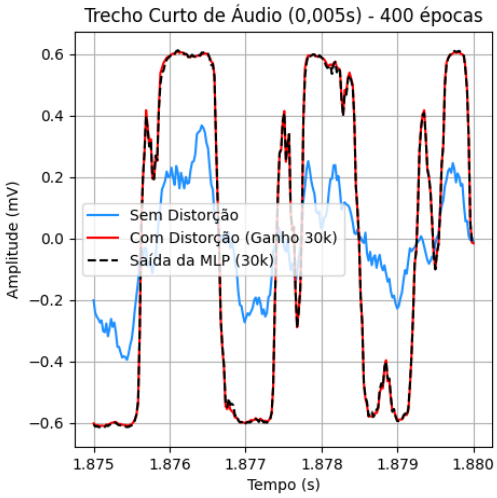
Figura 6.3 – Comparação dos Sinais (Original, Com Distorção e Saída da Rede) para diferentes valores de resistência.



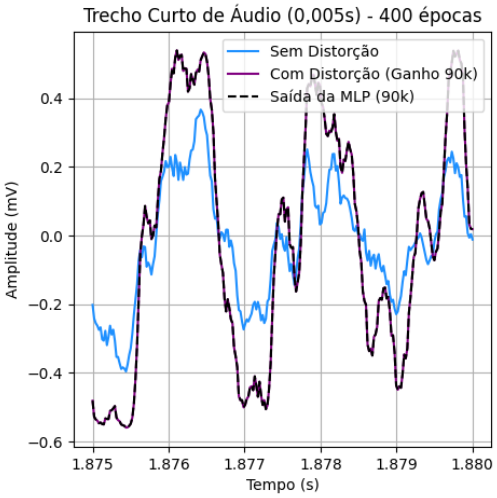
(a) Resistência de 0k.



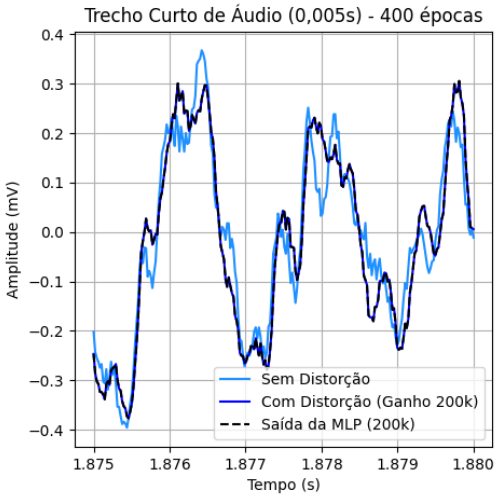
(b) Resistência de 10k.



(c) Resistência de 30k.



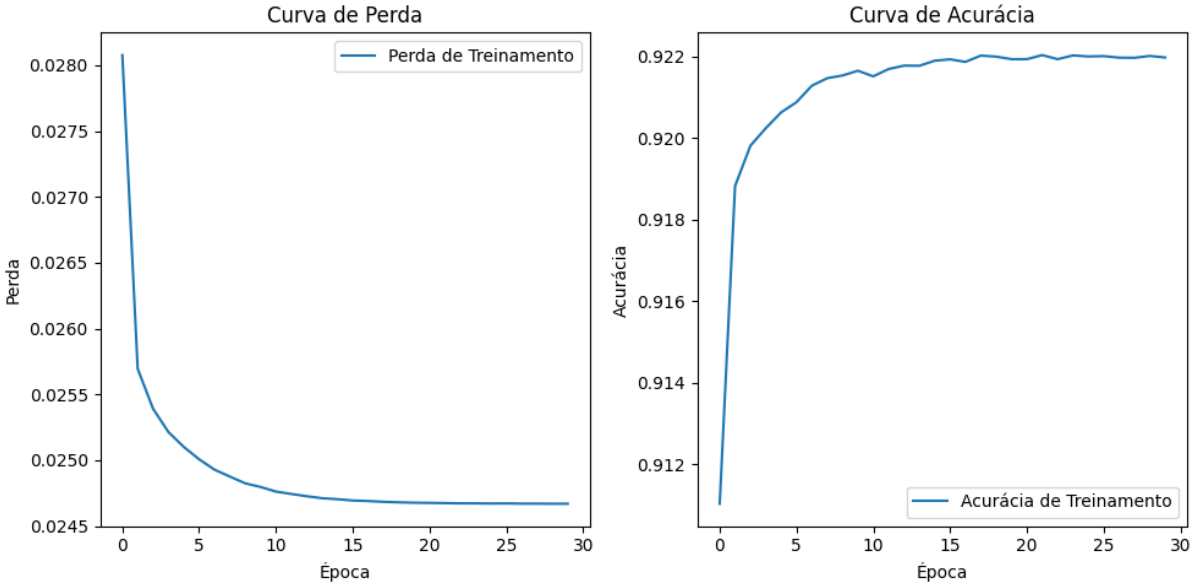
(d) Resistência de 90k.



(e) Resistência de 200k.

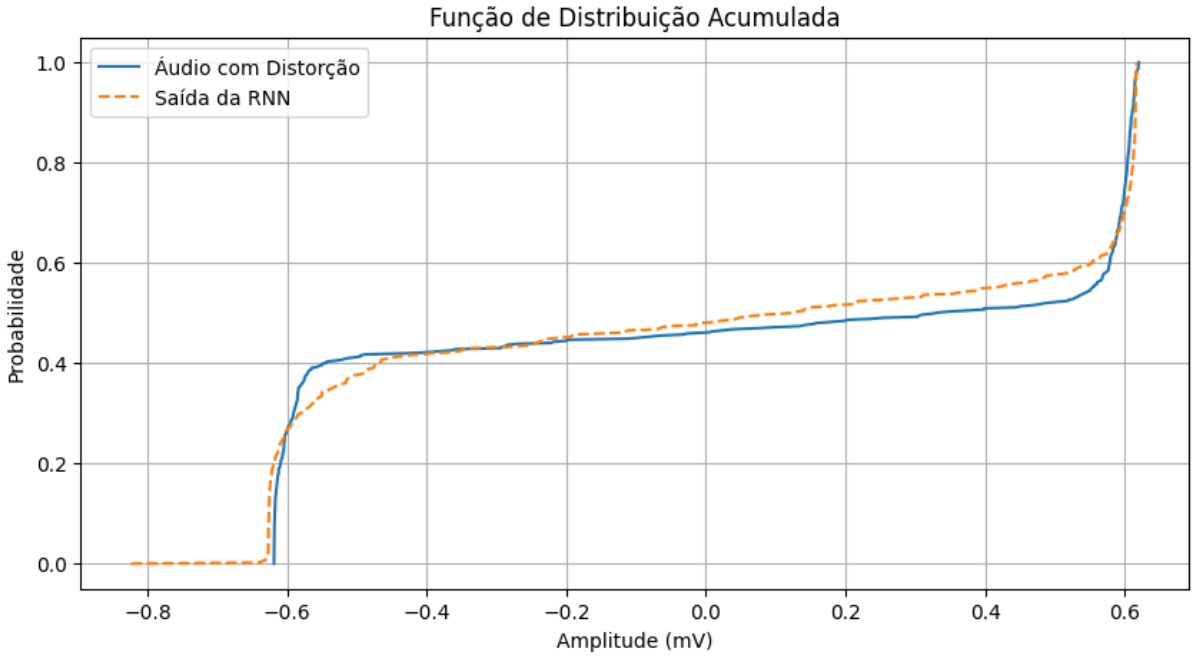
Fonte: Próprio autor.

Figura 6.4 – Evolução da Função de Perda e Acurácia por Época para a LSTM.



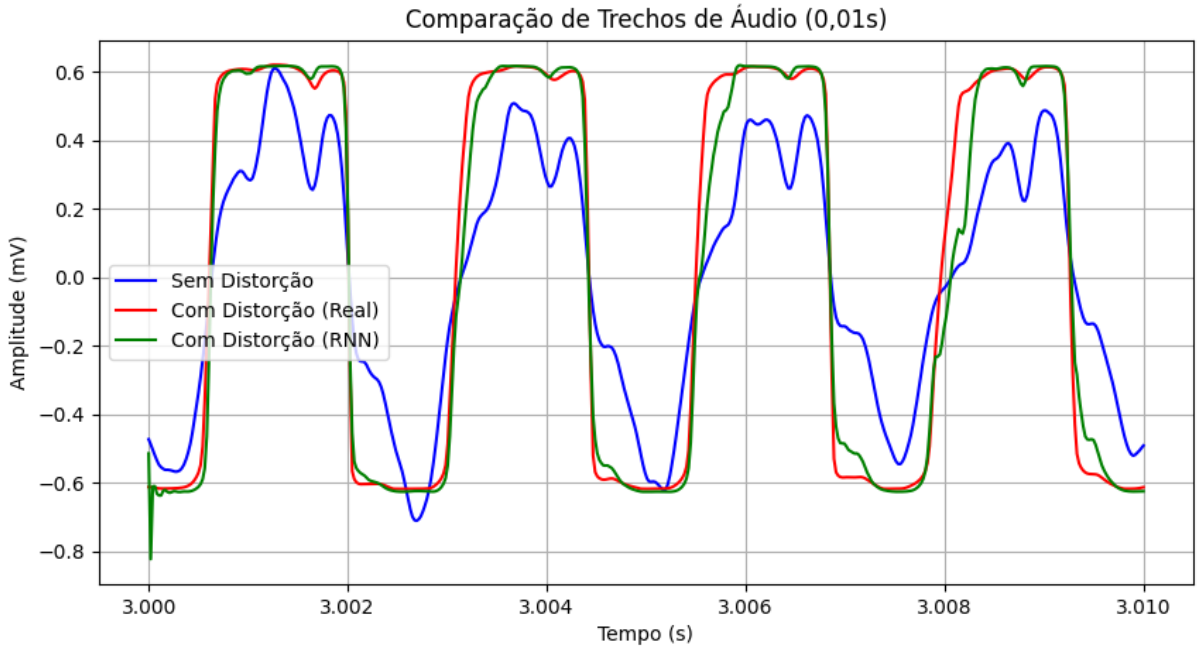
Fonte: Próprio autor.

Figura 6.5 – Gráfico da Função de Distribuição Acumulada (FDA) para a LSTM.



Fonte: Próprio autor.

Figura 6.6 – Comparação dos Sinais (Original, Com Distorção e Saída da Rede) para o resistor de 30k.



Fonte: Próprio autor.

7 Resultados da Modelagem do Comportamento Dinâmico do TurtleBot 3

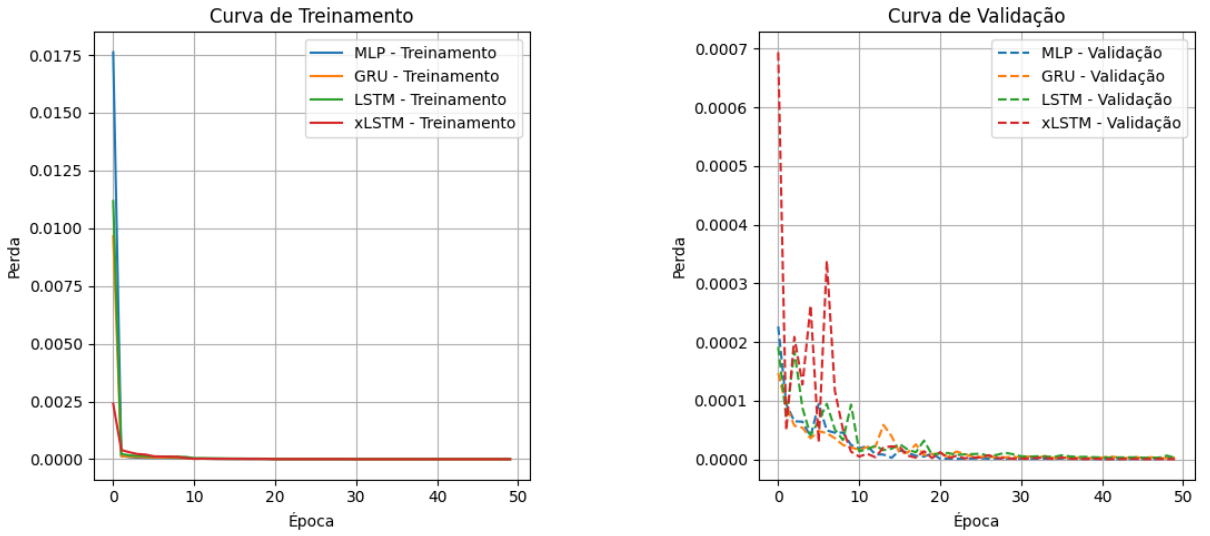
7.1 Desempenho da Modelagem

7.1.1 Função de Perda e Acurácia por Época

Nesta seção, são apresentados os gráficos da evolução da função de perda ao longo das épocas para os modelos MLP, GRU, LSTM e xLSTM. Esses gráficos permitem analisar a convergência do treinamento e a estabilidade da validação, indicando o desempenho de cada modelo na modelagem do comportamento dinâmico do TurtleBot 3.

A Figura 7.1 exibe as curvas de perda para os conjuntos de treinamento e validação. Observa-se que todos os modelos reduziram progressivamente a função de perda ao longo das épocas, demonstrando aprendizado. No entanto, diferenças na convergência entre as arquiteturas podem ser notadas, sendo a xLSTM a que apresentou a menor perda final.

Figura 7.1 – Evolução da Função de Perda por Época para a modelagem do TurtleBot 3.



(a) Curvas de perda no conjunto de treinamento.

(b) Curvas de perda no conjunto de validação.

Fonte: Próprio autor.

7.1.2 Métricas de Desempenho

Para avaliar a eficácia da modelagem do comportamento dinâmico do TurtleBot 3, foram analisadas diversas métricas de desempenho para os modelos MLP, GRU, LSTM e xLSTM. Mais detalhes das métricas são apresentados na Seção 5.3.7.

A Tabela 7.1 apresenta os resultados quantitativos obtidos para os modelos treinados e testados.

Tabela 7.1 – Métricas quantitativas para os modelos MLP, GRU, LSTM e xLSTM.

Modelo	**EQM**	**REQM**	**EAM**	**MdAE**	R^2
MLP (Treinamento)	2.73e-06	0.00165	0.00108	0.00059	0.999999
MLP (Teste)	0.00036	0.01893	0.01219	0.01045	0.999845
GRU (Treinamento)	1.37e-05	0.00370	0.00268	0.00202	0.999994
GRU (Teste)	0.00148	0.03847	0.02395	0.01216	0.999363
LSTM (Treinamento)	1.61e-05	0.00402	0.00293	0.00235	0.999993
LSTM (Teste)	0.01078	0.10383	0.07527	0.04613	0.995359
xLSTM (Treinamento)	1.15e-06	0.00107	0.00081	0.00073	0.999999
xLSTM (Teste)	0.00014	0.01180	0.00643	0.00370	0.999940

Fonte: Próprio autor.

7.1.3 Análise das Métricas

Os valores obtidos para as métricas de desempenho fornecem uma visão detalhada sobre a eficácia da modelagem do TurtleBot 3 para os datasets de treinamento e teste, considerando os modelos MLP, GRU, LSTM e xLSTM.

- **Erro Quadrático Médio (EQM):**
  - No dataset de treinamento, o menor EQM foi obtido pela xLSTM (1.15×10^{-6}), indicando um excelente ajuste aos dados. A MLP também apresentou um valor baixo (2.73×10^{-6}), seguida pela GRU (1.37×10^{-5}) e pela LSTM (1.61×10^{-5}).
  - No dataset de teste, a xLSTM manteve a menor taxa de erro (0.00014), demonstrando sua robustez. A MLP também apresentou um erro relativamente baixo (0.00036), enquanto a GRU (0.00148) e a LSTM (0.01078) tiveram desempenhos inferiores.
- **Raiz do Erro Quadrático Médio (REQM):**
  - No treinamento, os menores valores de REQM foram observados na xLSTM (0.00107) e na MLP (0.00165), seguidas pela GRU (0.00370) e pela LSTM (0.00402).
  - No teste, a xLSTM novamente obteve o menor REQM (0.01180), seguida pela MLP (0.01893), enquanto a GRU (0.03847) e a LSTM (0.10383) apresentaram um maior aumento no erro.
- **Erro Absoluto Médio (EAM):**
  - Durante o treinamento, o menor EAM foi observado na xLSTM (0.00081), com valores próximos para a MLP (0.00108), enquanto a GRU (0.00268) e a LSTM (0.00293) tiveram maiores desvios médios.

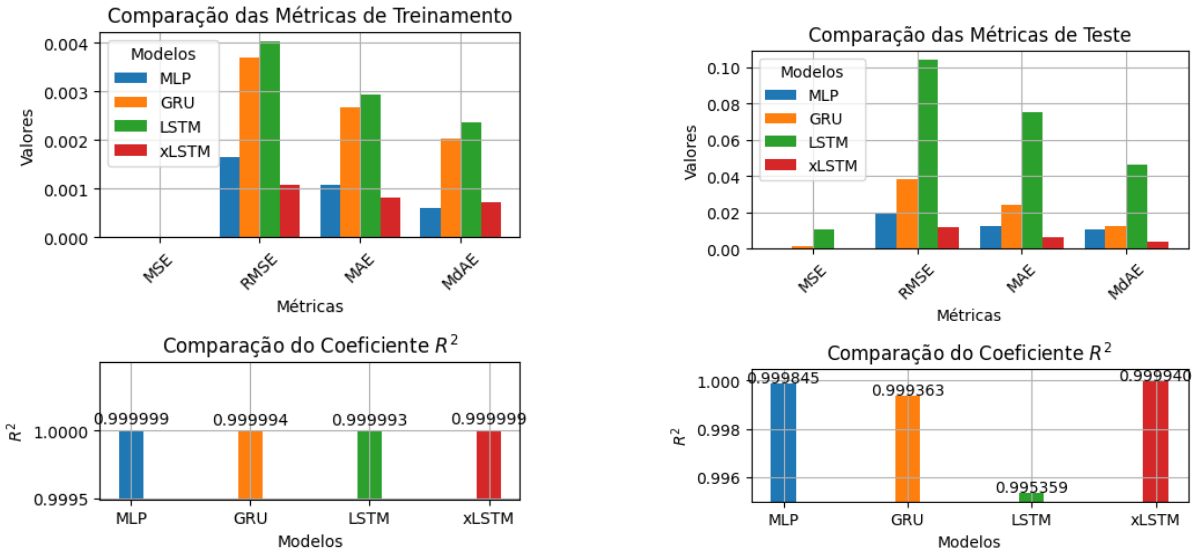
  - No teste, a xLSTM (0.00643) e a MLP (0.01219) mantiveram um desempenho superior à GRU (0.02395) e à LSTM (0.07527).

- **Erro Absoluto Mediano (MdAE):**
  - No treinamento, a xLSTM apresentou o menor erro mediano (0.00073), seguida pela MLP (0.00059), enquanto GRU (0.00202) e LSTM (0.00235) tiveram maiores valores.
  - No teste, a xLSTM (0.00370) e a MLP (0.01045) mantiveram desempenho superior à GRU (0.01216) e LSTM (0.04613).

- **Coeficiente de Determinação (R^2):**
  - No treinamento, todos os modelos apresentaram valores muito altos, com a xLSTM (0.999999) e a MLP (0.999999) alcançando os melhores resultados, seguidas pela GRU (0.999994) e LSTM (0.999993).
  - No teste, a xLSTM (0.999940) manteve a melhor generalização, seguida pela MLP (0.999845), enquanto a GRU (0.999363) e a LSTM (0.995359) apresentaram uma perda significativa de desempenho.

A Figura 7.2 apresenta os gráficos comparativos das métricas para os *datasets* de treinamento e teste.

Figura 7.2 – Comparação das métricas de desempenho dos modelos MLP, GRU, LSTM e xLSTM nos *datasets* de treinamento e teste.



(a) Métricas no *dataset* de treinamento.

(b) Métricas no *dataset* de teste.

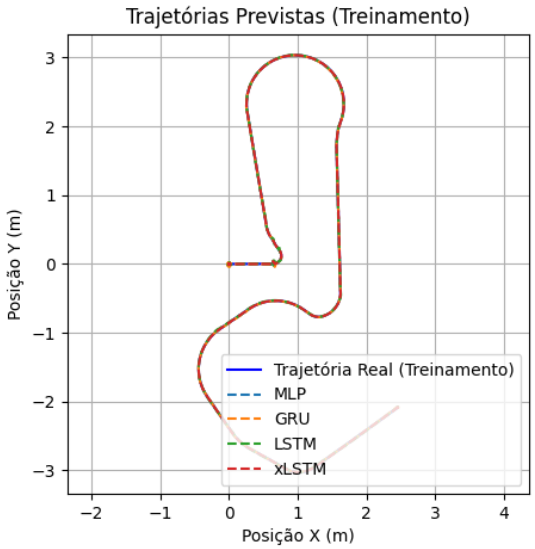
Fonte: Próprio autor.

Os resultados indicam que a xLSTM apresentou o melhor desempenho geral, com os menores valores de erro e um coeficiente R^2 mais próximo de 1 em ambos os conjuntos

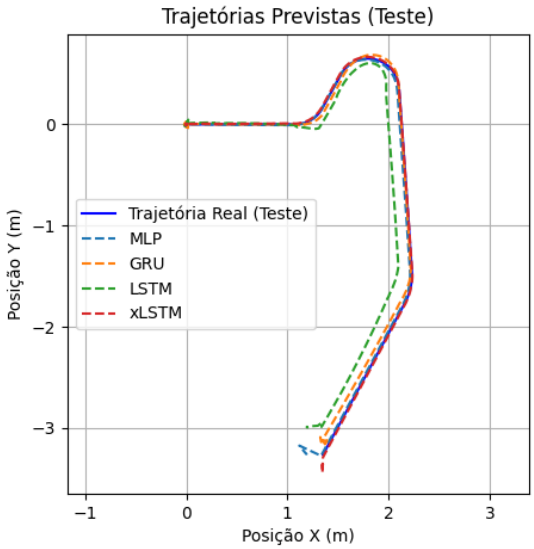
de dados. A MLP também demonstrou excelente precisão, especialmente no conjunto de teste. Já as redes GRU e LSTM apresentaram desempenhos inferiores, com a LSTM sendo a mais sensível a variações nos dados de teste, evidenciado pelos valores mais altos de MSE e RMSE.

7.1.4 Comparação das Trajetórias

Figura 7.3 – Comparação entre as trajetórias simuladas no Gazebo (*datasets* de treinamento e teste) e as trajetórias previstas pelos modelos MLP, GRU, LSTM e xLSTM para o TurtleBot 3.



(a) Comparação entre o *dataset* de treinamento e as trajetórias previstas pelos modelos.



(b) Comparação entre o *dataset* de teste e as trajetórias previstas pelos modelos.

Fonte: Próprio autor.

A comparação das trajetórias simuladas no Gazebo com as previstas pelos modelos MLP, GRU, LSTM e xLSTM permite avaliar a capacidade de generalização de cada abordagem na modelagem do comportamento dinâmico do TurtleBot 3.

No *dataset* de treinamento, observa-se que todos os modelos foram capazes de aprender e reproduzir com alta precisão a trajetória desejada. O modelo xLSTM apresentou um ajuste superior, com mínima discrepância entre os pontos simulados e previstos, seguido de perto pela MLP. A GRU e a LSTM também mostraram um bom desempenho, mas com pequenas oscilações ao longo da trajetória.

No *dataset* de teste, a xLSTM continuou apresentando a melhor aderência à trajetória simulada, evidenciando sua capacidade de generalização. A MLP também manteve um desempenho consistente, com pequenas divergências em certos pontos. Já a GRU e, principalmente, a LSTM, apresentaram maior variação na trajetória prevista, demonstrando

uma sensibilidade maior a dados não vistos durante o treinamento. Essas variações indicam que os modelos com maior capacidade de captura de dependências temporais, como a xLSTM, conseguem prever melhor a trajetória do TurtleBot 3 em cenários desconhecidos.

Os resultados reforçam a importância da escolha adequada do modelo para a modelagem de sistemas dinâmicos, destacando a eficácia da xLSTM e da MLP para a previsão da movimentação do TurtleBot 3 com maior precisão.

8 Conclusão

Este estudo analisa a identificação e modelagem de sistemas dinâmicos por meio de Redes Neurais Profundas (RNPs), comparando dois contextos distintos: a simulação de distorção em sinais de guitarra elétrica e a previsão de trajetórias do robô móvel TurtleBot 3. Os resultados evidenciam que a escolha da arquitetura neural influencia diretamente a capacidade de representação de sistemas não lineares, com implicações práticas em áreas que vão da robótica à indústria criativa.

No primeiro experimento, focado na modelagem de distorção sonora, as arquiteturas MLP e LSTM foram aplicadas de forma complementar. A MLP demonstrou eficiência na emulação de características estáticas de amplificadores - similar à forma como equalizadores gráficos reproduzem perfis tonais pré-configurados. Já a LSTM destacou-se na captura de padrões dinâmicos, reproduzindo com fidelidade efeitos como a modulação de *wah-wah*, cujo comportamento depende da interação temporal entre notas e ajustes do músico (YEH; SMITH, 2008). Essa dualidade sugere que a combinação de abordagens estáticas e sequenciais pode ser vantajosa em sistemas híbridos.

No segundo estudo, envolvendo o TurtleBot 3, a comparação entre MLP, GRU, LSTM e xLSTM revelou diferenças críticas na generalização de modelos. A xLSTM mostrou-se particularmente eficaz em cenários com variações imprevistas de trajetória, comportamento análogo aos sistemas de navegação automotiva que se adaptam a desvios de rota em tempo real. Em contraste, a MLP apresentou desempenho estável em condições controladas, porém com limitações ante perturbações não mapeadas - característica que a torna comparável a sistemas de piloto automático em ambientes estruturados. Já a GRU e a LSTM exibiram um comportamento intermediário: embora capazes de lidar com sequências temporais melhor que a MLP, mostraram-se menos robustas que a xLSTM em cenários dinâmicos, semelhante a sistemas de direção assistida que funcionam bem em estradas conhecidas, mas têm dificuldades em rotas não previstas.

A análise métrica fundamentou-se em cinco indicadores de desempenho (EQM, REQM, EAM, MdAE e R^2), confirmando a robustez da xLSTM em múltiplas dimensões de avaliação. Um aspecto notável reside na relação entre complexidade arquitetural e estabilidade preditiva: modelos com mecanismos de memória sofisticados, como a xLSTM, mostraram-se menos suscetíveis a flutuações operacionais, padrão que ecoa observações de estudos sobre controle adaptativo em robótica submarina (FOSSEN, 2011).

As aplicações potenciais estendem-se à implementação de Gêmeos Digitais em contextos industriais. Na área energética, por exemplo, a capacidade de prever estados dinâmicos permitiria simular o desgaste de turbinas eólicas sob diferentes condições mete-

orológicas, otimizando ciclos de manutenção. Paralelamente, na indústria musical, técnicas similares poderiam viabilizar sistemas de amplificação virtual com resposta dinâmica equivalente a equipamentos analógicos de alto custo, democratizando o acesso a sons profissionais.

Conclui-se que a metodologia explorada estabelece um marco para a integração entre aprendizagem de máquina e engenharia de sistemas. A versatilidade demonstrada nas aplicações - da reprodução de efeitos sonoros à navegação robótica - ressalta o potencial transdisciplinar das RNPs. Estudos futuros poderiam investigar a hibridização de arquiteturas, combinando a eficiência computacional da MLP com a capacidade adaptativa da xLSTM, estratégia que poderia equilibrar precisão e custo operacional em aplicações em tempo real.

8.1 Trabalhos Futuros

Um caminho promissor para pesquisas futuras reside na integração de Gêmeos Digitais (*Digital Twins*) com a metodologia de Redes Neurais Profundas (RNPs) aqui empregada. Esses sistemas, que replicam virtualmente entidades físicas para simulação e monitoramento em tempo real (TAO et al., 2018), ganharam relevância em setores como manufatura e saúde basta imaginar, por exemplo, como réplicas digitais de turbinas industriais podem prever desgastes meses antes de falhas críticas (JONES et al., 2020).

A metodologia baseada em RNPs incluindo MLP, LSTM e a inovadora xLSTM mostra-se particularmente adequada para essa finalidade. A razão é clara: essas arquiteturas demonstraram capacidade comprovada de capturar dinâmicas temporais complexas, como evidenciado nos experimentos com distorção sonora e navegação robótica. Para contextualizar, um Gêmeo Digital eficaz funciona como um modelo meteorológico digital: assim como previsões climáticas usam padrões históricos para antecipar tempestades, as RNPs podem mapear comportamentos de sistemas físicos para prever estados futuros.

No entanto, desafios persistem. A navegação autônoma, por exemplo, exige controle robusto frente a falhas de sensores ou interferências externas. Aqui, abordagens recentes com funções de barreira neural (ZHANG et al., 2024) sugerem caminhos interessantes. Imagine sistemas capazes de reagir a obstáculos imprevistos como um piloto automático ajusta trajetórias em tempo real essa resiliência operacional é crítica para aplicações em robótica móvel ou veículos autônomos.

As aplicações práticas dos projetos desenvolvidos ilustram bem esse potencial. No caso da modelagem de distorção em guitarras, um Gêmeo Digital poderia funcionar como um *plugin* de áudio inteligente, prevendo em tempo real como ajustes de ganho ou tonalidade afetariam o som final algo equivalente a ter um técnico de estúdio virtual. Já para o TurtleBot 3, a tecnologia permitiria monitorar o desgaste de motores durante operações

prolongadas, similar aos sistemas de diagnóstico embarcados em aeronaves comerciais.

Vale destacar que a verdadeira inovação está na sinergia entre técnicas. As RNPs não apenas modelam sistemas dinâmicos, mas criam ecossistemas digitais onde simulação e realidade física coexistem. Essa dualidade abre perspectivas para otimização contínua de processos industriais, redução de custos operacionais e até mesmo treinamento de operadores em ambientes virtuais seguros (KRITZINGER et al., 2018).

Em síntese, a metodologia aqui apresentada transcende aplicações pontuais. Ao combinar precisão preditiva com adaptabilidade operacional, estabelece as bases para sistemas ciberfísicos integrados onde modelos digitais e entidades reais não apenas coexistem, mas cooperam para otimizar resultados.

8.1.1 Modelagem de Sistemas por Gêmeos Digitais via RNPs

A integração de Redes Neurais Profundas (RNPs) em Gêmeos Digitais traz vantagens distintas, especialmente em sistemas que mudam com o tempo. Arquiteturas recorrentes como LSTM e xLSTM, por exemplo, são naturalmente adaptadas para lidar com sequências temporais pense em como previsões meteorológicas usam dados históricos para antecipar mudanças climáticas. Ao treinar esses modelos com registros operacionais de um sistema físico, cria-se uma réplica virtual que não apenas descreve o presente, mas projeta cenários futuros com precisão.

A xLSTM merece atenção especial aqui. Sua capacidade de identificar padrões temporais complexos, como variações não lineares ou atrasos em cadeias de produção, permite atualizações contínuas do Gêmeo Digital. Imagine um cenário industrial onde sensores enviam dados em tempo real: o modelo ajusta automaticamente suas previsões, funcionando como um "termômetro digital"que mede tanto o estado atual quanto riscos futuros. Essa característica é valiosa em aplicações que exigem respostas rápidas, como controle de processos químicos ou gerenciamento de tráfego urbano.

8.1.2 Cenários Aplicados e Potencial de Impacto

Os projetos desenvolvidos neste trabalho ilustram bem essa versatilidade. No caso da modelagem de distorção em guitarras, um Gêmeo Digital poderia operar como um assistente virtual para músicos: ao ajustar parâmetros como *gain* ou equalização em tempo real, o sistema preveria instantaneamente o som resultante algo semelhante a um engenheiro de áudio digital. Para o TurtleBot 3, a aplicação seria mais próxima de sistemas de diagnóstico em aviação, onde sensores monitoram o desgaste de componentes durante o voo, sugerindo manutenções preventivas antes de falhas críticas.

É relevante notar que o potencial vai além desses exemplos. Em energia eólica, por exemplo, Gêmeos Digitais baseados em RNPs poderiam simular o desgaste de turbinas sob

diferentes condições de vento, otimizando ciclos de inspeção. Já em logística, permitiriam testar rotas de entrega em ambientes virtuais antes de implantá-las fisicamente, reduzindo custos operacionais.

A metodologia aqui apresentada fundamentada na combinação entre aprendizagem profunda e modelagem dinâmica estabelece uma ponte entre simulação e realidade. Ao transformar dados brutos em *insights* preditivos, as RNPs não apenas replicam sistemas físicos, mas os aprimoram, oferecendo um caminho viável para operações mais seguras e eficientes em escala industrial (TAO et al., 2018; JONES et al., 2020).

8.1.3 Redes Neurais Profundas para Controle Robusto e Tolerante a Falhas em Navegação Autônoma

A navegação autônoma em ambientes não estruturados exige sistemas capazes de lidar com imprevistos. Robôs móveis como o TurtleBot 3, por exemplo, operam sob condições onde falhas parciais de sensores, ruídos ambientais e alterações abruptas no terreno podem comprometer a segurança operacional. Para garantir resiliência, arquiteturas de redes neurais como GRU, LSTM e xLSTM surgem como alternativas viáveis, graças à capacidade intrínseca de processar sequências temporais e padrões não lineares.

A principal vantagem desses modelos reside na adaptabilidade. Ao integrar técnicas de aprendizado por reforço, as RNPs permitem que sistemas de controle ajustem estratégias de navegação em tempo real um requisito crítico para evitar colisões em cenários caóticos.

Um aspecto relevante é a atualização contínua dos modelos. Dados operacionais coletados durante a execução de tarefas podem ser reinseridos no sistema, refinando previsões de forma similar a algoritmos de recomendação que personalizam saídas conforme interações do usuário. Essa abordagem é particularmente útil em missões de longa duração.

É fundamental destacar que a tolerância a falhas não se limita a respostas reativas. Sistemas baseados em RNPs antecipam cenários críticos por meio de simulações probabilísticas, identificando rotas alternativas antes mesmo de obstáculos surgirem.

Em síntese, a aplicação dessas redes transcende a mera otimização de trajetórias. Ao combinar previsão adaptativa com aprendizado incremental, estabelece-se um paradigma onde sistemas autônomos não apenas resistem a imprevistos, mas os integram como variáveis de operação. Avanços nessa área podem redefinir padrões de segurança em setores que vão da logística automatizada à exploração de ambientes hostis.

Referências

ACEMOGLU, D.; RESTREPO, P. Artificial intelligence, automation, and work. In: *The economics of artificial intelligence: An agenda*. [S.l.]: University of Chicago Press, 2018. p. 197–236. Citado na página 17.

ALON, U. *An introduction to systems biology: design principles of biological circuits*. [S.l.]: Chapman and Hall/CRC, 2019. Citado na página 17.

BECK, M. et al. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024. Citado 3 vezes nas páginas 12, 31 e 32.

BILLINGS, S. A. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. [S.l.]: John Wiley & Sons, 2013. Citado 2 vezes nas páginas 21 e 22.

BRUNTON, S. L.; KUTZ, J. N. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. [S.l.]: Cambridge University Press, 2022. Citado 2 vezes nas páginas 12 e 13.

FOSSEN, T. I. Handbook of marine craft hydrodynamics and motion control. *John Willy & Sons Ltd*, 2011. Citado na página 68.

GOODFELLOW, I. *Deep learning*. [S.l.]: MIT press, 2016. Citado na página 23.

GOODWIN, G. C.; SIN, K. S. *Adaptive filtering prediction and control*. [S.l.]: Courier Corporation, 2014. Citado 2 vezes nas páginas 19 e 21.

HEWING, L. et al. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, Annual Reviews, v. 3, n. 1, p. 269–296, 2020. Citado na página 12.

HOCHREITER, S. Long short-term memory. *Neural Computation MIT-Press*, 1997. Citado na página 31.

ISIDORI, A. *Nonlinear control systems: an introduction*. [S.l.]: Springer, 1985. Citado 2 vezes nas páginas 15 e 17.

JAZWINSKI, A. H. *Stochastic processes and filtering theory*. [S.l.]: Courier Corporation, 2007. Citado na página 16.

JONES, D. et al. Characterising the digital twin: A systematic literature review. *CIRP journal of manufacturing science and technology*, Elsevier, v. 29, p. 36–52, 2020. Citado 3 vezes nas páginas 13, 69 e 71.

JR, R. P. *The climate fix: what scientists and politicians won't tell you about global warming*. [S.l.]: Basic books, 2010. Citado na página 17.

KAILATH, T. *Linear systems*. [S.l.]: Prentice-Hall Englewood Cliffs, NJ, 1980. v. 156. Citado na página 17.

KALMAN, R. E. A new approach to linear filtering and prediction problems. 1960. Citado na página 20.

KARNIADAKIS, G. E. et al. Physics-informed machine learning. *Nature Reviews Physics*, Nature Publishing Group, v. 3, n. 6, p. 422–440, 2021. Citado na página 18.

KATAYAMA, T. et al. *Subspace methods for system identification*. [S.l.]: Springer, 2005. v. 1. Citado na página 20.

KLANCAR, G. et al. *Wheeled mobile robotics: from fundamentals towards autonomous systems*. [S.l.]: Butterworth-Heinemann, 2017. Citado na página 43.

KRITZINGER, W. et al. Digital twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline*, Elsevier, v. 51, n. 11, p. 1016–1022, 2018. Citado 2 vezes nas páginas 13 e 70.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Citado na página 23.

LENG, J. et al. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. *Journal of ambient intelligence and humanized computing*, Springer, v. 10, p. 1155–1166, 2019. Citado na página 13.

LENNART, L. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, v. 28, p. 540, 1999. Citado 2 vezes nas páginas 19 e 21.

NELLES, O.; NELLES, O. *Nonlinear dynamic system identification*. [S.l.]: Springer, 2020. Citado 2 vezes nas páginas 12 e 13.

OTT, E. *Chaos in dynamical systems*. [S.l.]: Cambridge university press, 2002. Citado na página 18.

PAN, S.; DURAISAMY, K. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, SIAM, v. 19, n. 1, p. 480–509, 2020. Citado na página 12.

RAISSI, M.; PERDIKARIS, P.; KARNIADAKIS, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, Elsevier, v. 378, p. 686–707, 2019. Citado na página 18.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015. Citado na página 23.

SIMPKINS, A. System identification: Theory for the user, (ljung, l.; 1999)[on the shelf]. *IEEE Robotics & Automation Magazine*, IEEE, v. 19, n. 2, p. 95–96, 2012. Citado na página 19.

SJÖBERG, J. et al. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, Elsevier, v. 31, n. 12, p. 1691–1724, 1995. Citado na página 21.

SLOTINE, J.-J. E.; LI, W. et al. *Applied nonlinear control*. [S.l.]: Prentice hall Englewood Cliffs, NJ, 1991. v. 199. Citado 2 vezes nas páginas 16 e 21.

STROGATZ, S. H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. [S.l.]: CRC press, 2018. Citado na página 18.

TAO, F. et al. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, IEEE, v. 15, n. 4, p. 2405–2415, 2018. Citado 3 vezes nas páginas 13, 69 e 71.

VASWANI, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. Citado na página 31.

YEH, D. T.; SMITH, J. O. Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations. *Proc. Digital Audio Effects (DAFx-08), Espoo, Finland*, p. 19–26, 2008. Citado na página 68.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>. Citado 10 vezes nas páginas 12, 23, 24, 25, 26, 27, 28, 29, 30 e 33.

ZHANG, H. et al. Fault tolerant neural control barrier functions for robotic systems under sensor faults and attacks. *arXiv preprint arXiv:2402.18677*, 2024. Citado na página 69.

ZHU, Q. *Nonlinear systems*. [S.l.]: MDPI-Multidisciplinary Digital Publishing Institute, 2023. Citado na página 18.