Fernando Kurike Matsumoto, José Lucas de Melo Costa

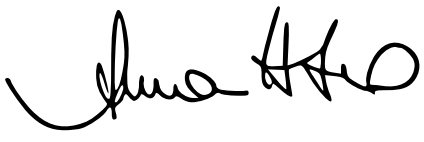
Supervisão da integridade de linhas de amarração: Neural Spectral Supervisor (NeSS)

São Paulo, SP

2023

São Paulo, 12 de dezembro de 2023.

De acordo,



Anna Helena Reali Costa

Fernando Kurike Matsumoto, José Lucas de Melo Costa

Supervisão da integridade de linhas de amarração: Neural Spectral Supervisor (NeSS)

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Profa. Dra. Anna Helena Reali Costa

Coorientador: Dr. Asdrubal do Nascimento Queiroz Filho

São Paulo, SP

2023

Catalogação-na-publicação

Matsumoto, Fernando Kurike
Supervisão da integridade de linhas de amarração: Neural Spectral Supervisor (NeSS) / F. K. Matsumoto, J. L. M. Costa -- São Paulo, 2023.
60 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.StemGNN 2.monitoramento de plataformas de petróleo 3.séries temporais multivariadas 4.aprendizado de máquina 5.segurança I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Costa, José Lucas de Melo

Resumo

Este trabalho explora o monitoramento de plataformas flutuantes de petróleo (FPSOs) e desenvolve o modelo Neural Spectral Supervisor (NeSS) para identificar falhas em linhas de amarração. Combinando aprendizado de máquina e processamento de sinais, o NeSS analisa séries temporais de posicionamento da plataforma, destacando-se por utilizar uma modelagem espectral e por capturar eficientemente correlações intra e inter-séries no domínio espectral. O modelo é alimentado por dados de simulações e o seu treinamento pode ser treinamento aprimorado por um estimador de período natural. Os resultados indicam que o NeSS apresenta desempenho comparável a outros modelos avançados, oferecendo uma solução técnica avançada e robusta para o monitoramento de infraestruturas críticas.

Palavras-chave: StemGNN. monitoramento de plataformas de petróleo. séries temporais multivariadas. aprendizado de máquina. segurança. meio ambiente.

Abstract

This study investigates the monitoring of floating oil platforms (FPSOs) and develops the Neural Spectral Supervisor (NeSS) model for detecting mooring line failures. Combining machine learning and signal processing, NeSS analyzes the platform's temporal positioning data, excelling in spectral modeling and efficiently capturing intra and inter-series correlations in the spectral domain. Powered by simulation data, its training can be enhanced with a natural period estimator. The results show that NeSS performs comparably to other advanced models, providing a sophisticated and robust technical solution for critical infrastructure monitoring

Keywords: StemGNN. oil platform monitoring. multivariate time series. machine learning. safety. environmental protection.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

No contexto da indústria de petróleo e gás, a exploração em alto mar (*offshore*) é a prática de extrair petróleo e gás natural de reservatórios localizados sob o leito do oceano. Estas operações são realizadas em ambientes marinhos, que podem variar de relativamente rasos a profundezas extremas. Nesse cenário, plataformas flutuantes de extração e armazenamento (*Floating Production, Storage, and Offloading* - FPSO) são elementos cruciais na indústria de energia, particularmente no segmento de extração de petróleo e gás em alto mar. Estas unidades são navios ou plataformas flutuantes que servem como instalações móveis para a produção e processamento de hidrocarbonetos.

Dada a complexidade e importância crítica das plataformas de petróleo offshore na atualidade (MA et al., 2019), sistemas eficientes de monitoramento dessas plataformas flutuantes são essenciais. O mal-funcionamento de uma plataforma pode levar a vazamentos de óleo ou outros incidentes que apresentam impactos devastadores (BEYER et al., 2016; ZHANG et al., 2019). Assim, sistemas capazes de prever e detectar comportamentos anômalos são não apenas uma necessidade operacional, mas também um imperativo de segurança e ambiental.

Dentre esses sistemas, uma área de pesquisa diz respeito a verificação de integridade de linhas de amarração (SAAD et al., 2021). Para manter as plataformas estáveis na mesma posição, cabos são conectados da plataforma offshore até o leito do oceano. Esses cabos são conhecidos como linhas de amarração. Esse trabalho está inserido nesse contexto crítico e visa apresentar um modelo capaz de identificar falhas no rompimento dessas linhas de amarração. Para tanto, serão utilizados modelos de aprendizado de máquina e processamento de sinais para extrair as informações necessárias para detectar uma ruptura.

1.1 Motivações e Visão Geral

Este trabalho é motivado pela importância da correta identificação do rompimento de linhas de amarração. De fato, o trabalho apresentado em (BROWN et al., 2005) discute que muitas FPSOs carecem de sistemas de detecção de falhas nas linhas de amarração, apesar de serem críticas para a segurança. Tal trabalho ressalta que a falta de monitoramento adequado e de sistemas de alerta para falhas nas linhas de amarração é um problema significativo no setor. A resolução desse problema envolve várias abordagens, dentre as quais o tratamento de séries temporais multivariadas, como os sinais de posicionamento da plataforma, sendo o foco deste trabalho.

O processamento desse tipo de dado é um tópico de pesquisa em rápido desen-

volvimento com uma ampla gama de aplicações em vários domínios, incluindo gestão da cadeia de suprimentos, finanças e saúde pública (LIM; ZOHREN, 2021). Enquanto técnicas tradicionais como ARIMA e modelos estado-espaço têm sido utilizadas (BOX; JENKINS, 1976), abordagens mais recentes baseadas em aprendizado profundo, como LSTM e GRU, mostraram resultados promissores (HOCHREITER; SCHMIDHUBER, 1997; CHO et al., 2014). No entanto, a maioria dos trabalhos existentes se concentra principalmente em capturar correlações temporais em uma única série temporal, muitas vezes negligenciando as correlações inter-séries cruciais em cenários multivariados.

Recentemente, modelos que utilizam redes convolucionais de grafos começaram a surgir, tentando capturar essas correlações inter-séries. No entanto, esses modelos frequentemente requerem uma topologia predefinida para representar as relações inter-séries (KIPF, 2016). O trabalho mais recente na área é o StemGNN, que não apenas captura as correlações intra-série e inter-séries de forma eficaz, mas também opera no domínio espectral para uma modelagem mais precisa (CAO et al., 2020). Motivado por esse novo ramo no processamento de séries temporais que baseia-se em um análise espectral, este projeto é desenvolvido no contexto de melhorar a classificação de séries temporais multivariadas. O foco será na aplicação de séries temporais de posição de uma plataforma offshore para a identificação de falhas nas suas linhas de amarração.

Como ilustrado na Figura 1, a abordagem proposta envolve o desenvolvimento de um modelo para identificar falhas nas linhas de amarração. Este modelo será alimentado com dados de posicionamento, permitindo uma avaliação da integridade das linhas. Esses dados de posicionamento, chamados de séries temporais, são tratados por meio de uma arquitetura que leva em consideração tanto as informações dentro de uma mesma série quanto as relações entre diferentes sinais, baseando-se nos recentes avanços nessa direção (CAO et al., 2020): mais precisamente, a abordagem proposta envolve utilizar a oscilação natural da plataforma, ou seja, seu período natural de oscilação, para detectar a ruptura.

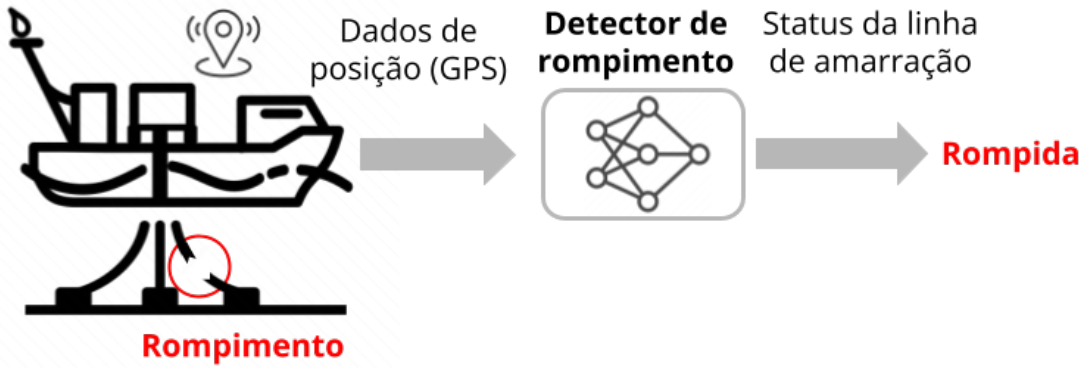


Figura 1 – Visão geral da proposta do projeto: um detector de rompimento de linhas de amarração utilizando os sinais de posicionamento da plataforma.

1.2 Objetivos e Justificativas

O trabalho de monitoramento de plataformas marítimas fixas com linhas de amarração é vital por diversas razões. Primeiramente, ele aborda uma questão crítica de segurança e ambiental: o risco de rompimentos e vazamentos de óleo, que podem ter consequências devastadoras tanto para a vida marinha quanto para as comunidades costeiras. Em segundo lugar, ele responde a uma necessidade empresarial específica, fornecendo uma solução técnica para melhorar a confiabilidade das operações.

Nesse sentido, esse projeto visa ao desenvolvimento de um modelo capaz de identificar falhas no rompimento das linhas de amarração. Este objetivo inclui o projeto, implementação e testes preliminares de um modelo que utiliza técnicas avançadas de aprendizado de máquina em um sistema de alerta sobre possíveis rompimentos de cabos em uma plataforma *offshore*. Outro objetivo é manter e atualizar a documentação do sistema, tornando-a clara e útil para desenvolvedores e *stakeholders*.

A relevância deste trabalho também se manifesta na inovação tecnológica, através do uso de algoritmos de aprendizado de máquina. Estes modelos são projetados para superar as limitações dos métodos tradicionais de monitoramento, oferecendo maior precisão na detecção de falhas ou comportamentos anômalos. A abordagem de modelagem de aprendizado de máquina segue as tendências recentes em pesquisa, destacadas em trabalhos como (BEGLI; DERAKHSHAN; KARIMIPOUR, 2019; TUMRATE et al., 2023), que exploram o uso de inteligência artificial para monitoramento de infraestrutura crítica.

1.3 Organização do Trabalho

Nesta seção, oferecemos uma visão geral da estrutura do trabalho, elucidando sua organização em capítulos para facilitar a leitura e o entendimento do conteúdo. Cada capítulo foi cuidadosamente elaborado para abordar aspectos distintos da pesquisa, desde conceitos fundamentais até considerações finais e perspectivas futuras.

- Capítulo 2 — Descrição do problema: Este capítulo apresenta o problema da identificação de rompimento de linhas de amarração.
- Capítulo 3 — Conceitos Básicos: Este capítulo fornece o alicerce teórico necessário para a compreensão do trabalho. São introduzidos os termos e conceitos essenciais que orientam o estudo.
- Capítulo 4 — Trabalhos Relacionados: Este capítulo oferece uma revisão da literatura e análise crítica de estudos e projetos relevantes ao tema do trabalho. São discutidas propostas e resultados de pesquisas similares, destacando sucessos e limitações anteriores.

- Capítulo 5 — Metodologia: Neste capítulo, descrevemos o modelo de aprendizado de máquina NeSS, assim como o processo de geração e processamento dos dados.
- Capítulo 6 — Desenvolvimento do Trabalho: Aqui, o foco é o relato do processo de desenvolvimento do sistema, incluindo desafios enfrentados, soluções adotadas e resultados alcançados.
- Capítulo 7 — Considerações Finais: Este capítulo conclui o trabalho, resumindo as principais contribuições e discutindo as limitações e potenciais melhorias futuras. Também serão apresentadas as implicações práticas do sistema desenvolvido.

Cada um desses capítulos contribui para o entendimento holístico do projeto, desde sua fundamentação teórica até suas aplicações práticas, enfatizando sua relevância e eficácia no monitoramento seguro e eficiente de plataformas fixas com cabos.

2 Descrição do Problema

2.1 O problema da amarração de plataformas

As plataformas flutuantes desempenham um papel essencial como instalações móveis na extração e armazenamento de hidrocarbonetos. As linhas de amarração, fundamentais nas operações dessas plataformas *offshore*, são cruciais para assegurar a estabilidade. Elas fixam as plataformas em uma posição estável, contrabalançando as forças variáveis do oceano, como correntes, ondas e ventos. A Figura 2 apresenta um modelo de plataforma flutuante, onde é possível perceber os cabos de amarração como linhas azuis. Esses mesmos cabos são apresentados na Figura 3 que mostra uma visão superior da disposição das linhas de amarração. É possível notar que tais linhas conectam a plataforma ao leito do oceano.

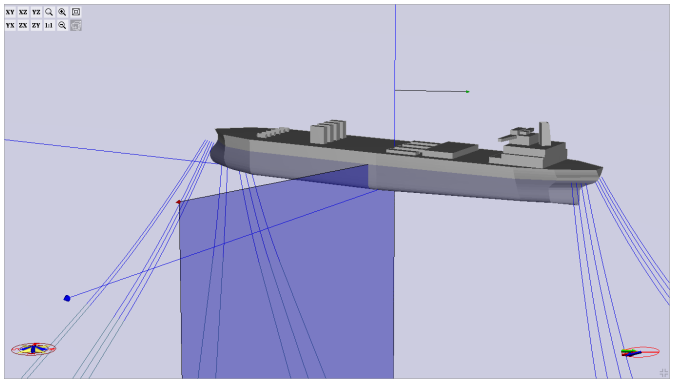


Figura 2 – Exemplo de uma plataforma flutuante e suas linhas de amarração.

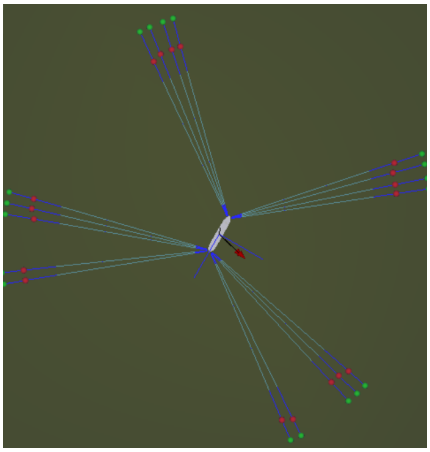


Figura 3 – Visão superior das linhas de amarração.

Entretanto, essas linhas de amarração podem sofrer desgastes e se romper. O trabalho apresentado em (FONTAINE et al., 2014) indica que 46% das causas de rompimento em linhas de amarração dizem respeito à corrosão e fadiga. Além de indicar o problema da falta de sistemas de monitoramento de FPSOs, o trabalho (BROWN et al., 2005) indica que a inspeção dessas linhas é normalmente realizada por meio de inspeção subaquática (por meio de câmeras) ou através de sensores de tensão nas linhas. Entretanto, esses mecanismos de inspeção são caros e requerem equipamentos especializados.

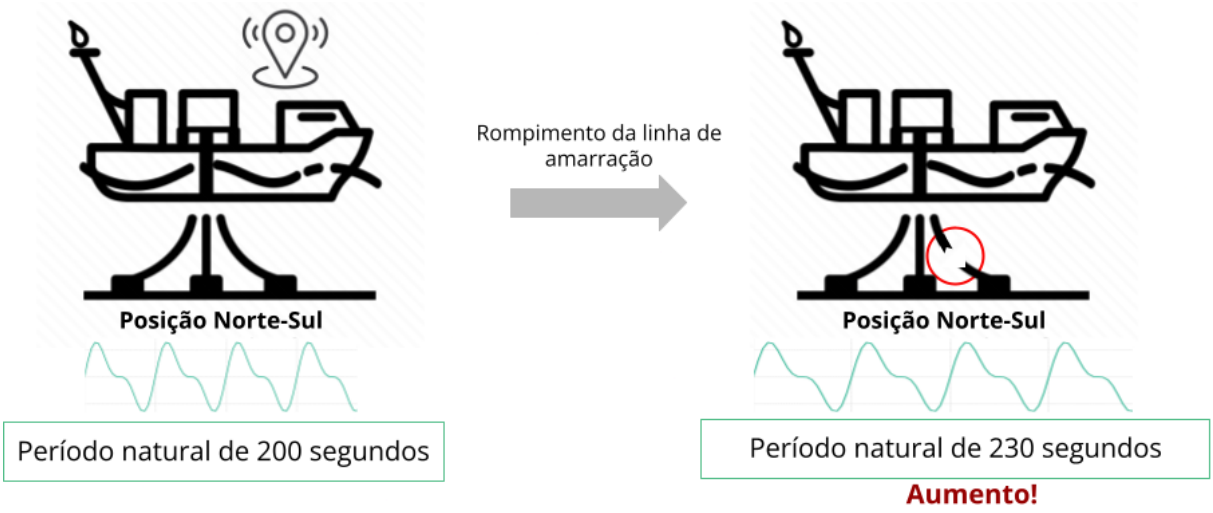


Figura 4 – Exemplo da abordagem baseada em período de oscilação natural: quando há um rompimento, espera-se que o período natural aumente e essa informação pode ser relevante para a identificação da falha na linha.

2.1.1 Importância do posicionamento da plataforma

Visando a redução de custos e a apresentação de alternativas aos mecanismos de inspeção citados, este projeto propõe a utilização de sinais de posicionamento da plataforma (*Global Positioning System* — GPS). Mais precisamente, o escopo do projeto está em utilizar a oscilação da posição da plataforma, chamada de período natural de oscilação, como um fator que caracteriza o estado das linhas de amarração: quando há uma linha rompida, espera-se que esse período de oscilação natural se altere, conforme apresentado na Figura 4.

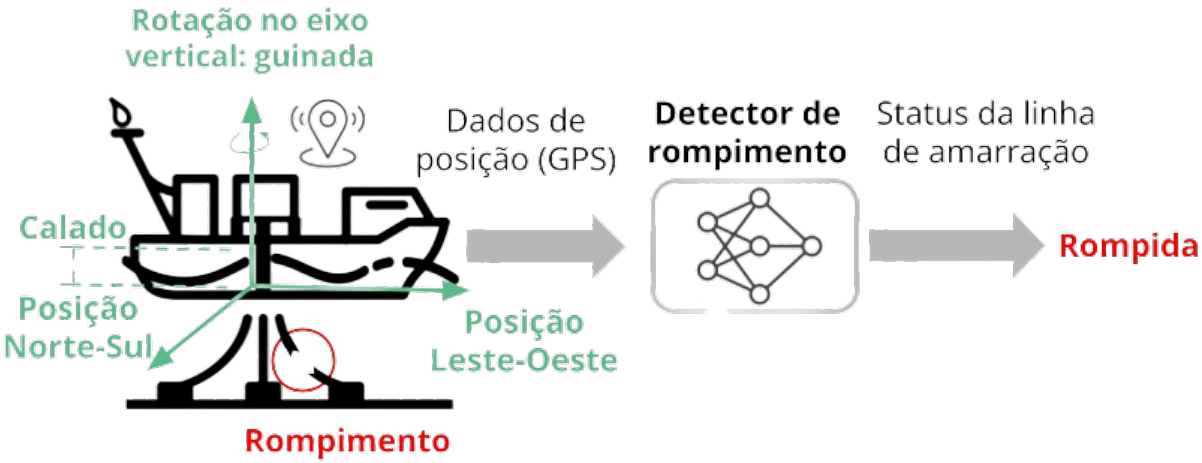


Figura 5 – Apresentação do problema da identificação de rompimento das linhas de amarração.

Contudo, a variação dos sinais quando uma linha está rompida é muito sutil, o que

torna o problema mais complexo. Assim, torna-se necessária a utilização de técnicas que mesclam processamento de sinais com métodos de aprendizagem de máquina. Conforme apresentado na Figura 5, os sinais são a entrada do sistema são: posição Norte–Sul e Leste–Oeste da plataforma, a rotação no eixo vertical (guinada) e a distância entre o nível da água e o ponto mais baixo da plataforma (calado). Esses sinais são processados por um modelo de detecção, baseando em aprendizado de máquina, e a saída é o estado da linha, ou seja, se a linha está rompida ou não.

2.2 Sistematização de Problemas em Aprendizagem de Máquina

Nesse contexto, a utilização da aprendizagem de máquina para a detecção de falhas em linhas de amarração de plataformas flutuantes oferece uma abordagem eficiente, contrastando com os métodos tradicionais de inspeção. Esta seção descreve as etapas sistemáticas envolvidas no desenvolvimento de uma solução de aprendizagem de máquina, adaptada às necessidades específicas deste projeto.

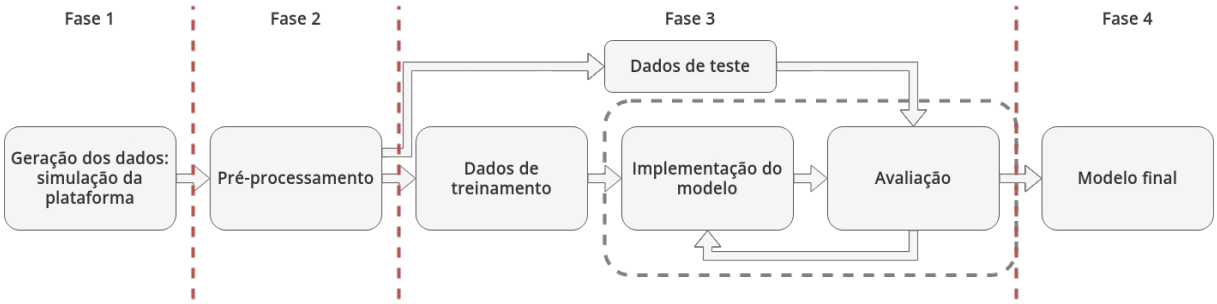


Figura 6 – Série de etapas envolvidas em um problema de aprendizado de máquina.

Conforme ilustrado na Figura 6, a implementação de um sistema de aprendizagem de máquina envolve várias etapas críticas, desde a coleta de dados até o treinamento e a validação do modelo. Cada uma dessas etapas é crucial para garantir a eficácia do sistema na detecção precisa de falhas nas linhas de amarração, baseando-se na análise dos sinais de posicionamento da plataforma. A seguir, detalhamos cada uma dessas etapas:

1. **Coleta de Dados:** A primeira etapa envolve a coleta de dados brutos, que neste caso, seriam os sinais de GPS da plataforma. Neste projeto serão utilizados dados simulados, conforme detalhado na seção 5.2
2. **Pré-processamento de Dados:** Nesta fase, os dados coletados são estruturados e preparados para o processamento.
3. **Treinamento do Modelo:** Nesta etapa, o modelo de aprendizagem de máquina é treinado usando os dados pré-processados, ajustando seus parâmetros para otimizar o desempenho.

4. **Validação e Teste do Modelo:** Avaliação do modelo em um conjunto de dados separado do utilizado no treinamento para verificar sua eficácia e precisão.

Finalmente, a arquitetura final proposta irá utilizar modelos que integram informações sobre a oscilação das plataformas em cada um dos sinais (posição Norte–Sul, posição Leste–Oeste, Guinada e Calado) ao mesmo tempo que compara as variações entre os sinais. Assim, esse processamento acontece no que é chamado de domínio espectral, através das Transformadas de Fourier tanto do grafo quanto das séries temporais. A próxima seção apresenta esses conceitos fundamentais antes de detalhar a arquitetura proposta.

3 Aspectos Conceituais

Esta seção fornece uma visão geral dos conceitos e termos fundamentais para a compreensão do trabalho. São apresentados os principais conceitos de aprendizado de máquina. Também são introduzidos os conceitos de monitoramento de plataformas fixas com cabos, incluindo os tipos de cabos e os principais riscos associados a eles.

3.1 Inteligência Artificial

A inteligência artificial (IA) permeia nossas vidas diárias, desde as recomendações que recebemos nas redes sociais até os sistemas de reconhecimento facial que desbloqueiam nossos telefones. É um conceito difícil de definir, dado seu amplo escopo e as diferentes interpretações do termo. Neste documento, utilizaremos a definição fornecida pela Comissão Europeia (Comissão Europeia, 2018):

> O conceito de inteligência artificial (IA) aplica-se a sistemas que apresentam um comportamento inteligente, analisando o seu ambiente e tomando medidas — com um determinado nível de autonomia — para atingir objetivos específicos.
>
> Os sistemas baseados em inteligência artificial podem ser puramente confinados ao software, atuando no mundo virtual (por exemplo, assistentes de voz, programas de análise de imagens, motores de busca, sistemas de reconhecimento facial e de discurso), ou podem ser integrados em dispositivos físicos (por exemplo, robôs avançados, automóveis autónomos, veículos aéreos não tripulados ou aplicações da Internet das coisas).

Neste trabalho, focaremos em sistemas de IA e, mais especificamente, nos algoritmos que são usados para alimentá-los. A disciplina de IA é composta por vários subcampos, mas existem dois ramos principais: simbólico e estatístico. A IA simbólica baseia-se na ideia de que a inteligência pode ser representada por símbolos e regras, enquanto a IA estatística baseia-se na ideia de que a inteligência pode ser aprendida a partir de dados.

O problema com a abordagem simbólica é que é difícil representar tarefas complexas que generalizam bem para uma ampla gama de entradas. É por isso que a IA estatística se tornou a abordagem dominante na última década, dependendo da disponibilidade de grandes conjuntos de dados e do aumento no poder de computação, como o processamento de propósito geral em unidades de processamento gráfico e o desenvolvimento de hardware especializado.

Dentro do ramo de IA estatística, existem dois subcampos principais: aprendizado de máquina e aprendizado profundo. Aprendizado de máquina refere-se ao uso de técnicas estatísticas para resolver uma tarefa específica, como classificação ou regressão. Aprendizado profundo é um subconjunto do aprendizado de máquina que usa redes neurais para resolver essas tarefas.

Este trabalho focará em aprendizado profundo e, mais especificamente, em um tipo de rede neural baseada em grafos. Redes neurais são apresentadas na próxima seção e redes neurais baseadas em grafos são apresentadas na subseção 3.3.3.

3.2 Redes Neurais

Conforme apresentado na seção 3.1, as redes neurais são uma ferramenta utilizada em aprendizado profundo e são o foco deste relatório de estágio. Esta seção apresentará os conceitos básicos de redes neurais e se concentrará em redes neurais convolucionais, que são usadas para classificação de imagens.

3.2.1 Conceitos Básicos

O componente básico de uma rede neural é o neurônio. Inspirado pelo neurônio biológico, é uma função matemática que recebe uma entrada e produz uma saída. A entrada é multiplicada por um peso, e um viés é adicionado ao resultado. A saída é então passada por uma função de ativação. A função de ativação é uma função não linear que permite que a rede aprenda funções complexas. A saída da função de ativação é a saída do neurônio. A representação matemática de um neurônio é dada pela Equação 3.1, e uma representação gráfica é dada pela Figura 7:

$$y = g\left(\sum_{i=1}^{n} w_i x_i + bias\right) = g\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{3.1}$$

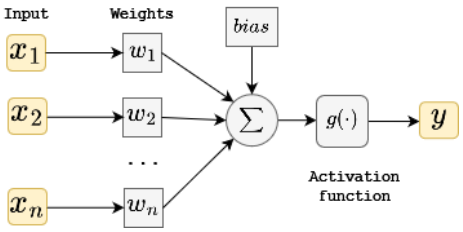


Figura 7 – Representação de um neurônio.

onde x_i é a entrada, w_i é o peso, b é o viés, g é a função de ativação, e y é a saída. Na forma vetorial, $\mathbf{w}$ é o vetor de peso, $\mathbf{x}$ é o vetor de entrada, e b é o viés.

Uma rede neural é composta de várias camadas de neurônios. A primeira camada é chamada de camada de entrada, e a última camada é chamada de camada de saída. As camadas intermediárias são chamadas de camadas ocultas. A saída de uma camada é a entrada da próxima camada. Uma única camada pode ser descrita como uma multiplicação

de matriz entre a entrada e os pesos, seguida pela adição do viés, e finalmente a aplicação da função de ativação:

$$\mathbf{y} = g\left(\mathbf{W}^T\mathbf{x} + \mathbf{b}\right) \tag{3.2}$$

onde $\mathbf{y}$ é a saída, $\mathbf{W}$ é a matriz de peso, $\mathbf{x}$ é a entrada, $\mathbf{b}$ é o viés, e g é a função de ativação.

A arquitetura descrita é chamada de rede neural feed-forward (FFN). É chamada de feed-forward porque a informação flui em uma direção, da entrada para a saída. Uma FFN também é chamada de perceptron multicamada (MLP), e pode ser representada por um gráfico acíclico direcionado (DAG), como mostrado na Figura 8.

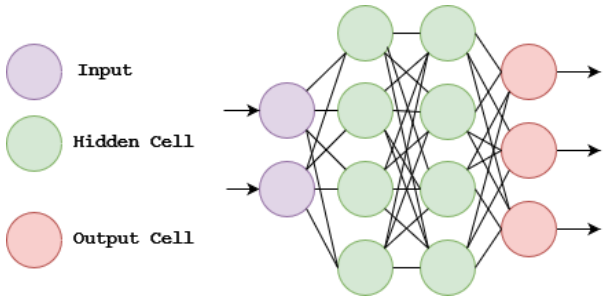


Figura 8 – Representação de um perceptron multicamada

Existem outros tipos de redes neurais, como redes neurais recorrentes, e redes neurais convolucionais, usadas para tarefas como classificação de imagens.

Um aspecto importante das redes neurais é o seu treinamento e sua capacidade de generalizar, ou seja, de obter bons resultados em dados não vistos anteriormente. Mais detalhes sobre os conceitos de treinamento de redes neurais estão elaborados no Apêndice A.

3.2.2 Camada de convolução unidimensional

A camada de convolução unidimensional, ou `Conv1D`, é uma estrutura fundamental em redes neurais convolucionais, especialmente adequada para processar dados sequenciais, como sinais temporais, séries temporais ou dados de áudio. Essa camada é utilizada para processar as séries temporais ao longo dentro do detector de rompimento.

Nesse tipo de convolução, um filtro (também chamado de *kernel*) de tamanho definido desliza ao longo do sinal de entrada, realizando uma operação de convolução. Este processo pode ser descrito matematicamente por:

$$y[n] = \sum_{m=0}^{M-1} x[n+m] \cdot w[m], \tag{3.3}$$

onde $y[n]$ é o sinal de saída, $x[n]$ é o sinal de entrada, $w[m]$ são os pesos do filtro e M é o tamanho do filtro. A convolução é aplicada em cada posição do sinal de entrada, o que permite a extração de características locais, como tendências ou padrões temporais específicos.

3.2.3 Funções de Custo para o Treinamento de Redes Neurais

O treinamento de uma rede neural, descrito no Apêndice A, é realizado por meio de uma *função de custo*, que codifica o erro da predição do modelo ($\hat{y}$) em relação ao valor real que deveria ser predito (y). O treinamento consiste em encontrar parâmetros para a rede neural que minimizem a função de custo.

Existem diversas funções de custo, cada uma adequada a diferentes tipos de problemas e dados.

3.2.3.1 Entropia Cruzada Binária

A Entropia Cruzada Binária, ou Binary Cross Entropy (BCE), é uma função de custo padrão para problemas de classificação binária. Nesse tipo de problema, o objetivo é classificar a entrada como positiva ($y = 1$) ou negativa ($y = 0$). A BCE é calculada da seguinte forma:

$$\mathrm{BCE}(y, \hat{y}) = -\sum_{i=1}^{N} \left[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \right], \tag{3.4}$$

onde y_i são as saídas corretas (0 ou 1), $\hat{y}_i$ são as previsões do modelo (valores entre 0 e 1 indicando a probabilidade de classe $y_i = 1$), e N é a quantidade de dados de treinamento.

3.2.3.2 Erro Quadrático Médio e Raiz do Erro Quadrático Médio

O Erro Quadrático Médio, ou Mean Squared Error (MSE), é uma função de custo amplamente utilizada para problemas de regressão (em que y e $\hat{y}$ assumem valores reais). Ele mede o erro entre y e $\hat{y}$ através da seguinte fórmula:

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

onde y_i são os valores reais, $\hat{y}_i$ são as previsões do modelo, e N é o número de observações. O MSE é eficaz para destacar grandes erros, pois as diferenças são elevadas ao quadrado.

A Raiz do Erro Quadrático Médio, ou Root Mean Squared Error (RMSE) é uma variação do MSE e é frequentemente utilizado por ser mais interpretável, pois está na mesma unidade que os valores preditos e reais. O RMSE é definido como:

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\mathrm{MSE}(y, \hat{y})} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}. \tag{3.6}$$

3.3 Grafos

Grafos são estruturas matemáticas usadas para modelar relações entre objetos. Um grafo G é definido como um par ordenado $G = (V, E)$, onde V é um conjunto finito de vértices (também chamados de nós) e E é um conjunto finito de arestas (relações entre os vértices). Dependendo do tipo de relação modelada, os grafos podem ser direcionados ou não direcionados. Uma aresta de um grafo não direcionado (u, v) é uma relação não ordenada, enquanto em um grafo direcionado $(u \to v)$ é uma relação ordenada. A representação matemática de um grafo é dada por:

$$G = (V, E), \quad V \subseteq \mathbb{U}, \quad E \subseteq \{(u, v) \mid u, v \in V\}. \tag{3.7}$$

3.3.1 Matriz de Adjacência e Lista de Adjacência

Há várias formas de representar grafos computacionalmente. Duas das mais comuns são a matriz de adjacência e a lista de adjacência.

A matriz de adjacência é uma matriz A de tamanho $|V| \times |V|$ onde A_{ij} é 1 se há uma aresta do vértice i para j e 0 caso contrário. Para grafos não direcionados, a matriz de adjacência é simétrica. A matriz de adjacência é definida como:

$$A_{ij} = \begin{cases} 1 & \text{se } (i, j) \in E \\ 0 & \text{caso contrário} \end{cases} \tag{3.8}$$

A lista de adjacência é uma representação mais eficiente em termos de espaço para grafos esparso. Para cada vértice v, mantém-se uma lista das arestas (v, u) onde u é um vértice adjacente a v.

3.3.2 Caminhos e Ciclos

Um caminho em um grafo é uma sequência de vértices tal que cada par adjacente de vértices é conectado por uma aresta. Um ciclo é um caminho que começa e termina no mesmo vértice e não repete nenhuma aresta ou vértice. A existência de ciclos é uma propriedade importante para muitos algoritmos de grafos, como detecção de ciclos e topological sort.

3.3.3 Grafos em Aprendizado Profundo

Grafos são particularmente úteis em aprendizado profundo para modelar relações complexas entre entidades. Redes neurais baseadas em grafos, ou Graph Neural Networks (GNNs), estendem os conceitos de redes neurais convencionais para operar sobre grafos

(VELIČKOVIĆ, 2023). Elas são especialmente úteis para tarefas como classificação de nós, classificação de grafos e previsão de links.

A atualização de um nó em uma GNN é geralmente uma função dos atributos desse nó e seus vizinhos. Formalmente, a atualização de um nó v em uma GNN pode ser descrita como:

$$\mathbf{h}_v^{(\ell+1)} = f\left(\mathbf{h}_v^{(\ell)}, \{\mathbf{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}\right) \tag{3.9}$$

onde $\mathbf{h}_v^{(\ell)}$ é a representação do nó v no instante ℓ, f é uma função de atualização, e $\mathcal{N}(v)$ é o conjunto de vizinhos do nó v.

3.3.4 Convolução atencional em grafos

A convolução atencional em grafos é uma técnica recente em GNNs que incorpora o mecanismo de atenção para aprimorar a capacidade da rede de focar em informações relevantes em um grafo. Essa abordagem é inspirada nos mecanismos de atenção utilizados em redes neurais para processamento de linguagem natural, adaptando-os ao contexto de estruturas de dados baseadas em grafos.

O mecanismo de atenção em grafos permite que a rede pondere de forma diferenciada as contribuições dos vizinhos de um nó durante o processo de agregação de informações. Em outras palavras, ele atribui pesos dinâmicos às arestas do grafo, refletindo a importância relativa de cada vizinho na atualização do estado do nó. Isso é particularmente útil em grafos com heterogeneidade ou quando a relevância dos vizinhos varia significativamente.

A operação de convolução atencional em um nó v pode ser expressa como:

$$\mathbf{h}_v^{(\ell+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv} \mathbf{W}^{(\ell)} \mathbf{h}_u^{(\ell)}\right), \tag{3.10}$$

onde $\mathbf{h}_v^{(\ell+1)}$ é a representação atualizada do nó v na camada $\ell + 1$, σ é uma função de ativação não-linear, $\mathbf{W}^{(\ell)}$ é a matriz de pesos da camada ℓ, e α_{uv} são os coeficientes de atenção que determinam a importância do vizinho u na atualização do nó v.

Os coeficientes de atenção α_{uv} são calculados com base nas características dos nós u e v, frequentemente usando uma rede neural pequena e totalmente conectada. Esse processo permite que a rede aprenda a focar nos aspectos mais informativos do grafo para a tarefa em questão, seja ela classificação de nós, de grafos, ou previsão de links.

3.4 Transformada de Fourier

3.4.1 Transformada de Fourier para Sinais Temporais

A Transformada de Fourier é uma técnica matemática usada para decompor um sinal contínuo e infinito em seus componentes de frequência. Matematicamente, para uma função contínua do tempo, $x(t)$, a Transformada de Fourier $X(f)$ é definida como:

$$\hat{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt,$$

onde f representa a frequência (PROAKIS; MANOLAKIS, 1996). Esta fórmula transforma a função do domínio do tempo para o domínio da frequência, revelando as componentes de frequência do sinal original.

No campo de processamento de sinais digitais, em que os sinais são discretos e finitos, utiliza-se uma variação dessa transformada denominada Transformada de Fourier Discreta (DFT, do inglês *Discrete Fourier Transform*). Dado um sinal discreto $x[n]$ com N amostras ($0 \leq n < N$), a DFT é definida como:

$$\hat{x}[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi \frac{kn}{N}},$$

onde $\hat{x}[k]$ é a componente de frequência correspondente à frequência discreta k, e n é o índice de tempo discreto (PROAKIS; MANOLAKIS, 1996).

3.4.2 Transformada de Fourier para Grafos

O conceito de Transformada de Fourier não se limita apenas ao domínio temporal. Uma extensão importante desse conceito é a Transformada de Fourier de Grafos (GFT, do inglês *Graph Fourier Transform*), que permite a decomposição espectral de grafos (RICAUD et al., 2019).

Na GFT, o conceito de frequência é substituído pelos autovalores e autovetores da matriz laplaciana de um grafo. Mais precisamente, considere um grafo $G = (V, E)$ com N vértices e uma função $x : V \to \mathbb{R}$ que associa cada vértice do grafo a um número real. A partir desses dados, são calculados os autovalores λ_k e autovetores μ_k da matriz laplaciana de G, definida como:

$$L_{i,j} = \begin{cases} \text{grau}(v_i) & \text{se } i = j, \\ -1 & \text{se } i \neq j \text{ e } (i,j) \in E, \\ 0 & \text{caso contrário.} \end{cases}$$

Finalmente, a GFT da função x no grafo G é $\hat{x} : \{\lambda_1, \lambda_2, \ldots, \lambda_N\} \to \mathbb{R}$, definida como:

$$\hat{x}(\lambda_\ell) = \sum_{i=1}^{N} f(i) \mu_\ell^{\intercal}(i).$$

3.5 Modelos de Aprendizado Profundo para Séries Temporais

Modelos como LSTM (Long Short-Term Memory), GRU (Gated Recurrent Units) e GLU (Gated Linear Units) têm mostrado promessa na modelagem de dependências temporais intra-série (HOCHREITER; SCHMIDHUBER, 1997; CHO et al., 2014; LEA et al., 2017). No entanto, muitos desses modelos lidam com multivariabilidade de forma ad hoc ou recorrem a prioridades predefinidas para modelar as relações inter-série.

3.5.1 *Long Short-Term Memory* - LSTM

Long Short-Term Memory (LSTM) é uma arquitetura especial de rede neural recorrente (RNN), particularmente adequada para aprender dependências de longo prazo e são amplamente utilizadas em problemas de séries temporais, processamento de linguagem natural, e outras áreas onde os dados de entrada possuem uma natureza sequencial. Nesse projeto, esse tipo de rede é utilizada para extrair o período natural de oscilação das plataformas, além de servir como base de comparação.

A arquitetura fundamental de uma LSTM consiste em uma série de blocos de memória, cada um contendo três componentes-chave: a porta de esquecimento, a porta de entrada e a porta de saída. Estas portas funcionam como mecanismos de controle que regulam o fluxo de informações dentro e fora do bloco de memória, permitindo que a LSTM armazene, modifique ou exclua informações ao longo do tempo.

- **Porta de Esquecimento:** Esta porta decide quais informações do estado anterior devem ser mantidas ou descartadas. Ela é controlada por uma função sigmóide, que pondera a relevância das informações anteriores para a tarefa atual.
- **Porta de Entrada:** Esta componente controla o fluxo de novas informações para a célula de memória. Ela inclui uma função sigmóide, que decide quais valores serão atualizados, e uma função tangente hiperbólica, que gera um novo vetor de candidatos a serem adicionados ao estado da célula.
- **Porta de Saída:** Esta porta determina quais informações da célula de memória serão usadas na saída. A função sigmóide decide quais partes do estado da célula contribuirão para a saída, e a função tangente hiperbólica processa o estado da célula para gerar a saída efetiva.

Essas portas permitem que as LSTMs mantenham um balanço entre a manutenção de informações relevantes de longo prazo e a descartação de dados irrelevantes, tornando-as eficazes para tarefas que requerem a compreensão de contextos e padrões temporais complexos. Adicionalmente, as LSTMs são capazes de mitigar o problema do desaparecimento do gradiente, possibilitando o treinamento eficiente em sequências longas de dados.

3.5.2 *Gated Linear Units* - GLU

Gated Linear Units (GLUs) são uma arquitetura de rede neural que introduz um mecanismo de controle, ou "portão", para regular o fluxo de informações através de uma unidade linear. Este mecanismo permite que as GLUs aprendam a balancear a quantidade de informação que passa pela rede, tornando-as eficientes para capturar relações complexas em dados. Nesse projeto, GLUs são usadas no interior do processamento do detector de falhas.

A estrutura de uma GLU consiste basicamente de duas partes: uma unidade linear e uma porta de controle. A unidade linear processa os dados de entrada, enquanto a porta, geralmente uma função sigmóide, determina a quantidade de informação da unidade linear que será transmitida para frente. Matematicamente, a operação de uma GLU pode ser representada como:

$$\text{GLU}(x) = x \otimes \sigma(W_g x + b_g) \tag{3.11}$$

onde x é o vetor de entrada, $\otimes$ denota a multiplicação elemento a elemento, σ é a função sigmóide, W_g e b_g são, respectivamente, os pesos e os vieses da porta de controle.

3.6 Avaliação de Modelos de Classificação Binária

As métricas de avaliação são cruciais em aprendizado de máquina, pois fornecem uma compreensão quantitativa do desempenho de modelos preditivos. Abaixo, discutimos métricas importantes para a tarefa de classificação binária, como a acurácia, a matriz de confusão, precisão, revocação e F1.

3.6.1 Acurácia

A acurácia é uma das métricas mais diretas e frequentemente usadas. Ela mede a proporção de previsões corretas em relação ao total de previsões feitas. Matematicamente, é expressa como:

$$\text{Acurácia} = \frac{\text{Número de Previsões Corretas}}{\text{Total de Previsões}}$$

Embora seja uma métrica útil, a acurácia pode ser enganosa em conjuntos de dados desbalanceados, onde uma classe é muito mais frequente que a outra. Além disso, ela é incapaz de diferenciar entre os diferentes tipos de erro, que serão descritos abaixo.

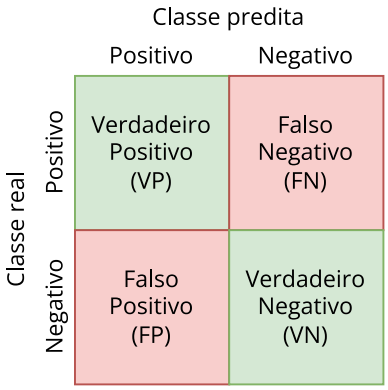


Figura 9 – Matriz de confusão

3.6.2 Matriz de Confusão

A matriz de confusão, exemplificada na Figura 9, é uma ferramenta poderosa para visualizar o desempenho de um modelo de classificação. Ela mostra as previsões corretas e incorretas detalhadas por classes. A matriz é composta por quatro elementos: Verdadeiros Positivos (VP), Falsos Positivos (FP), Verdadeiros Negativos (VN) e Falsos Negativos (FN), em função das classes reais e preditas. Essa distribuição fornece informações sobre o tipo de erros cometidos pelo modelo.

3.6.3 Precisão, Revocação e F1

A partir da matriz de confusão, diversas métricas podem ser definidas que levam em conta a distinção entre erros de tipo FP e FN.

A **precisão**, também conhecida como valor preditivo positivo, é uma métrica utilizada quando o objetivo é minimizar o número de Falsos Positivos. Esse é o caso, por exemplo, em um sistema de detecção de lixo eletrônico em e-mails, em que é importante minimizar o número de mensagens legítimas identificadas como lixo eletrônico. A precisão é definida como:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}}.$$

A **revocação**, também conhecida como sensibilidade, é utilizada quando o objetivo é minimizar o número de Falsos Negativos. Esse é o caso, por exemplo, em um exame médico, em que a prioridade é garantir que todos os pacientes doentes sejam corretamente identificados. A revocação é definida como:

$$\text{Revocação} = \frac{\text{VP}}{\text{VP} + \text{FN}}.$$

Por fim, o **F1-Score** é a média harmônica da precisão e da revocação. É útil quando queremos um equilíbrio entre as duas métricas. O uso da média harmônica significa que

o F1-Score só será alto se ambas a precisão e a revocação forem elevadas. O F1-Score é calculado como:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}.$$

4 Trabalhos Relacionados

A previsão de séries temporais multivariadas é um campo de pesquisa em constante evolução que tem recebido contribuições significativas tanto do campo das redes neurais quanto do processamento de sinais. Modelos tradicionais como ARIMA (AutoRegressive Integrated Moving Average) (BOX; JENKINS, 1976) e VAR (Vector Autoregressive) (SIMS, 1980) forneceram a base para abordagens mais modernas que buscam capturar tanto as relações temporais intra-série quanto as inter-série.

4.1 Redes Neurais de Grafos Espectro-Temporais

O StemGNN destaca-se nesta literatura ao oferecer uma abordagem unificada que aborda tanto as relações temporais (intra-série) quanto as espaciais (inter-série) simultaneamente no domínio espectral (CAO et al., 2020). Utilizando a Transformada de Fourier de Grafos (GFT, do inglês *Graph Fourier Transform*) para modelar as correlações inter-série e DFT para as dependências temporais intra-série, o StemGNN consegue capturar padrões mais complexos em ambos os aspectos. Além disso, sua capacidade de aprender automaticamente as relações inter-série sem depender de prioridades predefinidas o torna uma solução mais flexível e robusta para uma ampla gama de aplicações em previsão de séries temporais multivariadas. A Figura 10 apresenta a estrutura do modelo, que será detalhada a seguir.

4.1.1 Estrutura do Modelo

O StemGNN é estruturado para processar dados multivariados $\mathbf{X} \in \mathbb{R}^{N \times T}$, onde N é o número de nós, T é o comprimento da série temporal.

[**Bloco básico do StemGNN**]**:** O bloco StemGNN utiliza a convolução espectral em grafos para capturar relações inter-séries temporais de maneira eficaz, particularmente devido à sua capacidade de aprender representações latentes em dados de séries temporais. O bloco é composto pelos seguintes elementos principais:

- **Graph Fourier Transform (GFT)**: Uma operação que transforma os sinais do domínio do tempo para o domínio da frequência para capturar relações espectrais.
- **Spe-Seq Cell**: Uma célula aplicada após a GFT para aprender padrões temporais no domínio da frequência.
- **Inverse Graph Fourier Transform (IGFT)**: Aplicada após o processamento pela Spe-Seq Cell para transformar os sinais de volta para o domínio do tempo.

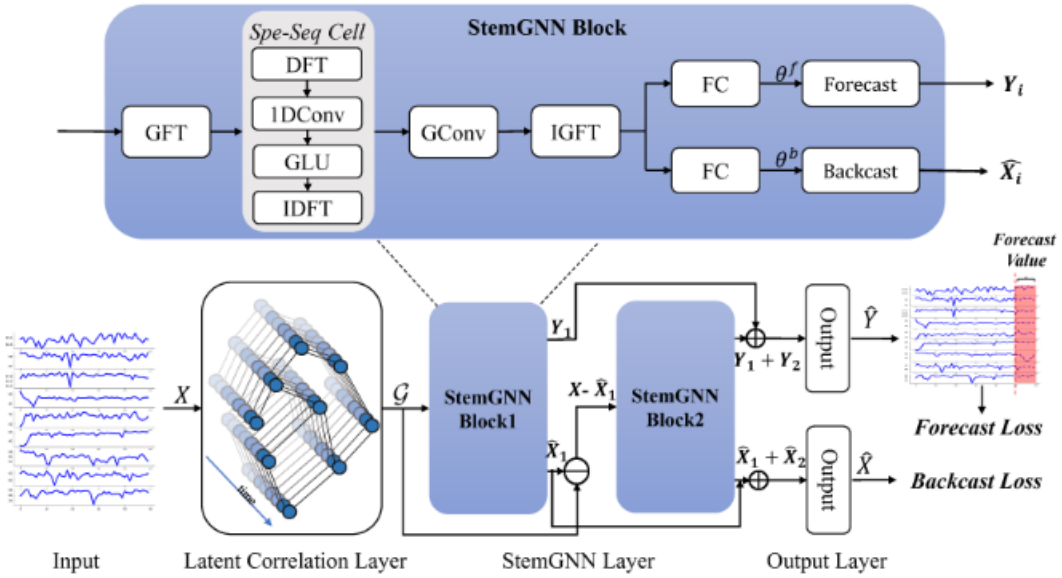


Figura 10 – Arquitetura do modelo StemGNN, retirado de Cao et al. (2020).

- **Expansão para Múltiplos Canais**: O modelo pode ser estendido para múltiplos canais, com cada canal sendo processado individualmente e combinado no final.

O bloco básico do StemGNN funciona através de um processo de várias etapas, começando com a aplicação da GFT para cada série temporal, seguido pela célula Spe-Seq para aprender padrões temporais e, finalmente, a IGFT para transformar o sinal processado de volta para o domínio do tempo.

[Célula Spectral Sequential (Spe-Seq Cell)]: A Célula Spectral Sequential, ou Spe-Seq Cell, é um componente utilizado no processamento de séries temporais multivariadas. Sua principal função é capturar padrões temporais no domínio da frequência após a aplicação da Transformada de Fourier em Grafos (GFT). A Spe-Seq Cell consiste de quatro componentes principais que operam em sequência: Discrete Fourier Transform (DFT, $\mathcal{F}$), convolução 1D, Gated Linear Unit (GLU) e Inverse Discrete Fourier Transform (IDFT, $\mathcal{F}^{-1}$). A DFT e a IDFT transformam os dados entre o domínio temporal e o domínio de frequência, enquanto a convolução 1D e o GLU aprendem representações de características no domínio da frequência.

A célula opera separadamente nas partes real e imaginária do sinal de entrada, processadas paralelamente por operadores idênticos com parâmetros distintos.

4.2 Análise espectral para identificação de rompimento

Costa et al. (2021) apresentam um sistema para supervisionar a integridade de linhas de amarração de FPSOs, baseando-se na estimativa da frequência natural de

oscilação da plataforma. Os autores propõem que mudanças na frequência natural, devido a rompimentos de linhas, podem ser um indicativo de falhas no sistema de amarração. Utilizando métodos de aprendizado de máquina e processamento de sinais, o estudo valida a hipótese de que o período natural varia perceptivelmente em caso de falha nas linhas, destacando a importância desta medida para prever falhas.

4.3 Utilização de métodos tabulares para identificação de rompimentos

No estudo de Silva et al. (2022), diversos modelos de aprendizado de máquina para classificação de séries temporais multivariadas foram avaliados na tarefa de detecção de rompimento de linhas de amarração em FPSOs. Os modelos avaliados incluem o Rocket (combinado com os modelos de Regressão Logística e XGBoost), o InceptionTime e DDTW. Esses modelos foram examinados em termos de sua acurácia, e o Rocket e InceptionTime demonstraram bons resultados, com a combinação do Rocket com Regressão Logística obtendo resultados marginalmente superiores.

Também foi realizada uma análise da robustez desses modelos ao ruído em dados de teste, quando treinados em dados sem ruído. O estudo revelou que o InceptionTime é mais robusto ao ruído, desempenhando significantemente melhor que outros modelos quando sujeito a dados de teste ruidosos.

5 Método do trabalho

Este capítulo descreve a abordagem metodológica adotada para o desenvolvimento deste trabalho, fornecendo uma visão geral das principais fases envolvidas. Além disso, são detalhadas as etapas de coleta de dados, preprocessamento de dados, e treinamento do modelo, conforme descrito na Figura 6. É importante esclarecer que alguns dos detalhes do desenvolvimento das fases e seus respectivos resultados são discutidos em outros capítulos, e não neste capítulo.

5.1 Fases do Desenvolvimento

O desenvolvimento do sistema de monitoramento eficiente e seguro de plataformas fixas com cabos é dividido nas seguintes fases:

1. **Projeto do Sistema**: Aqui, são definidas as arquiteturas de hardware e software que serão usadas para atender aos objetivos do projeto. (Ver seção 5.4).

2. **Implementação**: Nesta fase, o sistema é efetivamente construído com base no projeto. (Ver Capítulo 6).

3. **Testes e Validação**: Os modelos de aprendizado de máquina e o sistema como um todo são testados para assegurar que atendam aos requisitos e objetivos definidos. (Ver seção 6.5).

4. **Identificação de Melhorias e Trabalhos Futuros**: Esta etapa foca na análise crítica das limitações do modelo NeSS e sugere aprimoramentos, bem como direções para pesquisas futuras. (Ver Capítulo 7).

5.2 Geração dos Dados Simulados

O Dynasim, criado pelo Tanque de Provas Numérico (TPN) da Universidade de São Paulo, representa uma ferramenta sofisticada de simulação de embarcações. Sua função primordial é modelar os movimentos de estruturas marítimas como navios ou plataformas petrolíferas, considerando suas características físicas e as condições ambientais.

Neste projeto, aplicamos o Dynasim para modelar a dinâmica da plataforma de petróleo P-50. As simulações utilizaram dados hidrodinâmicos específicos da P-50, assim como um conjunto diverso de condições ambientais reais, medidas na plataforma. Especificamos também se deveria haver alguma linha de amarração rompida durante a

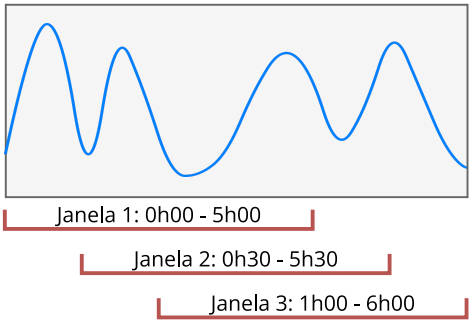


Figura 11 – Divisão de uma série temporal de 6 horas em janelas de 5 horas com passo de 30 minutos

simulação. As simulações resultaram em séries temporais complexas, as quais são utilizadas neste projeto. Nosso foco recaiu sobre dados específicos como o calado da plataforma e suas posições nos eixos Norte–Sul e Leste–Oeste, além da Guinada.

5.3 Preprocessamento dos Dados

Uma vez gerados os dados a partir do Dynasim, é crucial preprocessá-los para o treinamento do NeSS. Este processo envolve duas etapas, detalhadas a seguir.

5.3.1 Divisão em Janelas de Tempo

Os dados de cada simulação são inicialmente divididos em várias janelas de duração fixa de 5 horas. Essas janelas são criadas com um passo de 30 minutos entre o início de duas janelas consecutivas, conforme ilustrado na Figura 11. Este método tem o objetivo de capturar as características temporais dos dados de forma eficiente e, ao mesmo tempo, restringir a quantidade de dados exigida pelo modelo. A escolha da duração das janelas será discutida na seção 6.2.

5.3.2 Divisão em Conjuntos de Treino e Teste

Após segmentar os dados em janelas temporais, procedemos com a divisão em conjuntos de treino e teste. Neste projeto, utilizamos uma proporção de 20% para treino e 80% para teste. Essa distribuição foi escolhida para garantir que o modelo seja treinado e avaliado em uma quantidade substancial de dados, mantendo ao mesmo tempo um conjunto de teste amplo para uma avaliação rigorosa da performance do modelo.

Durante a divisão dos dados, asseguramos que todas as janelas temporais originadas de uma mesma simulação fossem alocadas no mesmo conjunto (de treino ou de teste). Esse método evita a ocorrência de dados da mesma simulação nos dois conjuntos, eliminando assim o risco de vazamento de informações entre treino e teste. Tal precaução é crucial

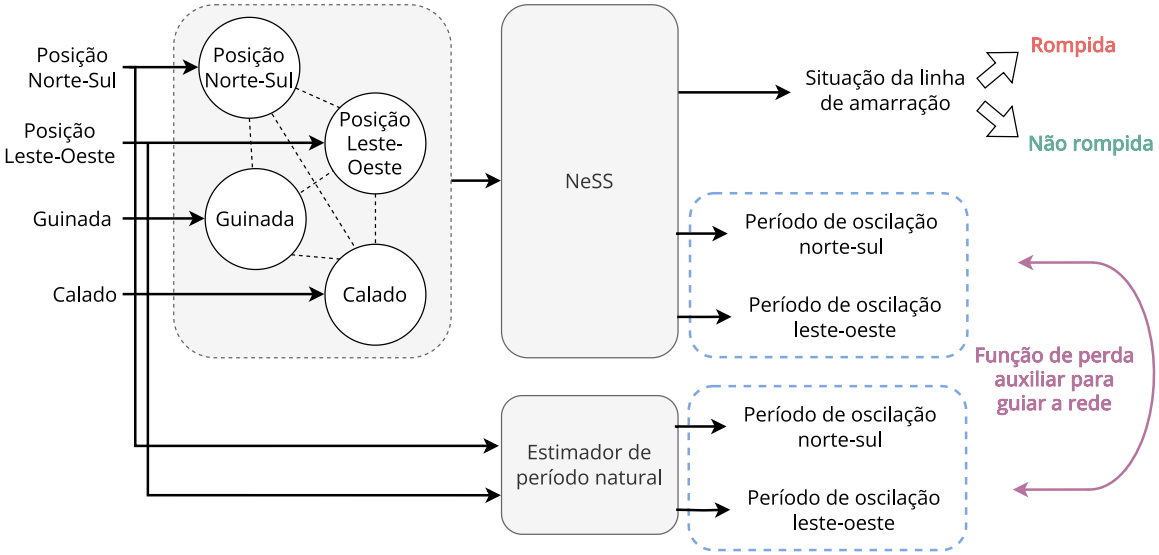


Figura 12 – Visão geral funcionamento do NeSS

para prevenir que o modelo tenha acesso a dados de teste durante a fase de treinamento, assegurando uma avaliação mais justa e realista do seu desempenho.

5.4 Neural Spectral Supervisor (NeSS)

Este projeto propõe o modelo Neural Spectral Supervisor (NeSS), projetado para oferecer uma análise eficiente e precisa de séries temporais, especificamente para monitorar a integridade de linhas de amarração. Uma visão geral do funcionamento do NeSS pode ser vista na Figura 12. Os dados de entrada preprocessados são organizados em forma de grafo e fornecidos como entrada do NeSS, que prediz a situação da linha de amarração (se ela está rompida ou intacta).

Além disso, também implementamos um estimador de período natural, que recebe um sinal ruidoso como entrada e estima o seu período natural. Este estimador não faz parte do modelo do NeSS, mas será usado durante o treinamento, conforme descrito na subseção 5.4.3.

5.4.1 Arquitetura do NeSS

O modelo NeSS é composto por três componentes, cada um desempenhando uma função específica no processamento e análise dos dados. Estes componentes estão ilustrados na Figura 13 e serão detalhados a seguir.

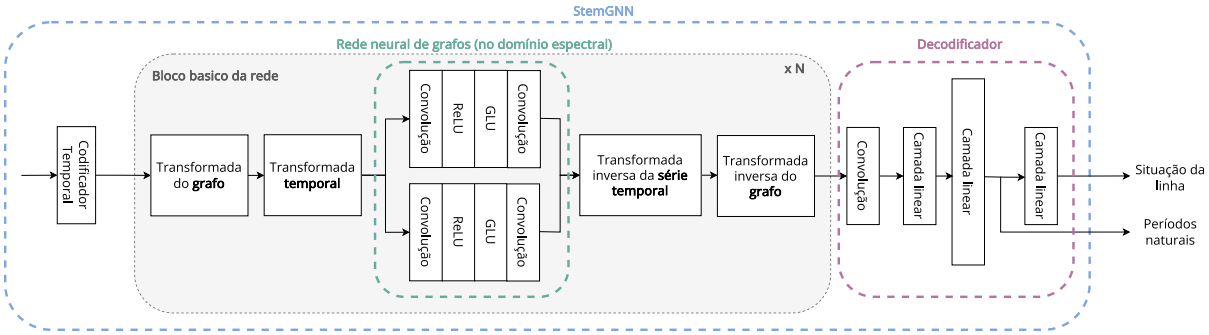


Figura 13 – Arquitetura do NeSS

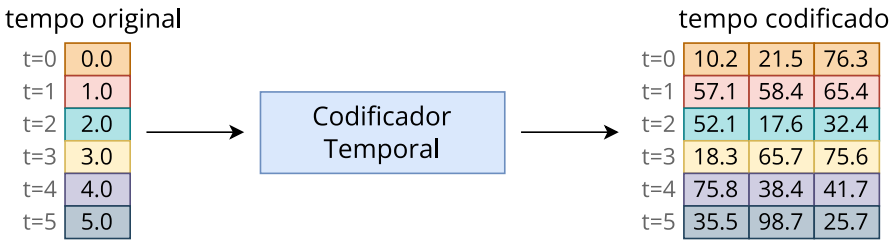


Figura 14 – Funcionamento do codificador temporal

Codificador Temporal

Este componente é responsável por codificar o tempo nas séries temporais de entrada. Dada uma série temporal, ele recebe como entrada o tempo em cada ponto da série e retorna uma representação vetorial do tempo, como ilustrado na Figura 14, para uma representação de 3 dimensões (o codificador real utiliza 32 dimensões). Esse codificador oferece uma representação mais rica dos dados de entrada, baseada em senoides e cossenoides, servindo um papel semelhante aos Positional Encoders utilizados em modelos Transformer.

Descrição da Rede Neural de Grafos

Rede Neural de Grafos: Este componente, baseado no StemGNN, processa a representação vetorial das séries temporais no domínio espectral. Para isso, são aplicadas a Transformada de Fourier de Grafos e a Transformada de Fourier Discreta sobre a entrada. Os dados são então processados por uma GNN no domínio espectral, e as transformadas inversas são aplicadas. Uma descrição detalhada do funcionamento do StemGNN é dado na seção 4.1.

Decodificador

Este componente consiste em uma rede neural simples que transforma a saída da rede neural de grafos na previsão final do estado do sistema, indicando se a linha está rompida ou intacta. Além disso, são geradas duas saídas adicionais que predizem o período de oscilação da plataforma nas direções Norte–Sul e Leste–Oeste.

5.4.2 Treinamento do Modelo

O treinamento do NeSS é uma etapa crucial que visa otimizar o modelo para realizar previsões precisas e confiáveis. O processo de treinamento envolve ajustar os parâmetros da rede neural com base nos dados de treinamento, para minimizar a seguinte função de custo:

$$\mathcal{L} = \mathrm{BCE}(y, \hat{y}) = \sum_{i=1}^{N} \left[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \right], \tag{5.1}$$

onde y são as predições do modelo (valores entre 0 e 1, representando a probabilidade de a linha estar rompida), $\hat{y}$ são as saídas corretas, e BCE é a entropia cruzada binária.

5.4.3 Estimador de Períodos Naturais

O estimador de período natural é uma rede neural LSTM, treinada com a função de custo MSE em dados sintéticos que serão descritos abaixo. Os parâmetros dos dados sintéticos foram escolhidos para gerar períodos aleatórios, mas com uma distribuição centrada em torno de 200 segundos, representando uma variedade de cenários de oscilação que o modelo NeSS pode encontrar na prática.

5.4.3.1 Geração de Dados Sintéticos de um Sistema Amortecido

Para treinar e avaliar o estimador de período natural, utilizamos dados sintéticos que simulam a resposta de um sistema mecânico com amortecimento. A geração desses dados foi realizada através da função que simula o sistema a partir de parâmetros físicos como a massa m, o coeficiente de amortecimento c e a rigidez k.

5.4.3.2 Cálculo das Características do Sistema

A partir dos parâmetros m, c e k, as características de um sistema são calculadas como segue:

- Frequência natural não amortecida:

$$\omega_n = \sqrt{\frac{k}{m}}$$

- Razão de amortecimento:

$$\zeta = \frac{c}{2\sqrt{km}}$$

- Frequência natural amortecida:

$$\omega_p = \omega_n \sqrt{1 - \zeta^2}$$

- Período natural:

$$T_n = \frac{2\pi}{\omega_p}$$

5.4.3.3 Simulação da Resposta do Sistema

A resposta do sistema é simulada ao longo de um período de tempo t, com uma entrada aleatória u representando ruído branco normal. A entrada aleatória é justificada pela aproximação de ruído branco para sistemas de deriva lenta, discutida no Capítulo 3. O tempo é discretizado em 14400 pontos, representando um intervalo de 0 a 14400 segundos (4 horas).

A função de transferência do sistema é dada por:

$$G(s) = \frac{\frac{1}{m}}{s^2 + \frac{c}{m} \cdot s + \frac{k}{m}}.$$

Usamos a função `lsim` da biblioteca SciPy para simular a saída do sistema linear $G(s)$ com entrada $u(t)$. A saída que deve ser predita pelo estimador (o período natural T_n do sistema) é então normalizada para facilitar o processamento posterior.

5.4.3.4 Treinamento do NeSS com Auxílio do Estimador de Períodos Naturais

Além do procedimento padrão de treinamento do NeSS descrito na subseção 5.4.2, implementamos um método para auxiliar o treinamento do modelo a partir do estimador de períodos naturais. Este método envolve o uso do estimador de período natural para estimar os períodos de oscilação da plataforma nas direções Norte–Sul e Leste–Oeste.

O estimador de período natural funciona paralelamente ao modelo principal, fornecendo estimativas dos períodos de oscilação do sistema. Essas estimativas são então comparadas com as previsões feitas pelo NeSS, como mostrado em Figura 12. A discrepância entre as previsões da rede e as estimativas do estimador é quantificada usando o RMSE entre os períodos calculados pelo estimador, $T^{\text{Estimador}}$, e os períodos preditos pelo NeSS, T^{NeSS}. Dessa forma, a função de custo global durante este treinamento com auxílio do estimador é uma combinação do custo de classificação e do custo auxiliar:

$$\mathcal{L} = \underbrace{\text{BCE}(y, \hat{y})}_{\text{custo de classificação}} + \lambda \cdot \underbrace{\text{RMSE}(T^{\text{NeSS}}, T^{\text{Estimador}})}_{\text{custo auxiliar}}, \tag{5.2}$$

onde λ é um parâmetro que indica o peso do auxílio no treinamento. A função de custo padrão (sem auxílio do estimador) é equivalente a utilizar $\lambda = 0$.

Este método de treinamento auxiliado pelo estimador é projetado para garantir que o NeSS não apenas preveja com precisão o estado das linhas de ancoragem, mas também adquira uma compreensão aprofundada das dinâmicas temporais do sistema. A hipótese é que esse auxílio guiará o aprendizado do modelo, de forma que ele consiga modelar melhor o problema.

6 Desenvolvimento do Trabalho

Este capítulo tem como objetivo relatar o processo de desenvolvimento do projeto. Aqui, descrevemos as tecnologias utilizadas e os experimentos realizados, especificando os objetivos de cada um e analisando seus resultados. Finalmente, apresentamos uma comparação do desempenho do NeSS com outros modelos.

6.1 Tecnologias Utilizadas

Este projeto utiliza uma variedade de tecnologias e ferramentas para facilitar a sua implementação. Cada tecnologia foi escolhida para assegurar eficiência, escalabilidade e robustez no desenvolvimento e na execução dos modelos de aprendizado de máquina. Esta seção detalha as tecnologias utilizadas.

6.1.1 Python

A linguagem de programação Python serve como a espinha dorsal do projeto. Esta linguagem de programação de alto nível é amplamente utilizada em uma variedade de aplicações, desde desenvolvimento web até aprendizado de máquina. Sua popularidade na ciência de dados é notável, suportada por uma comunidade extensa que contribui para um rico ecossistema de bibliotecas e frameworks de código aberto. Esta flexibilidade e suporte tornam o Python uma escolha ideal para a construção de soluções avançadas de aprendizado de máquina.

Além disso, durante o processo de desenvolvimento e análise, os Jupyter Notebooks desempenharam um papel essencial. Eles forneceram um ambiente interativo que facilitou a experimentação, visualização de dados e documentação, aumentando significativamente a produtividade e a clareza na execução do projeto.

6.1.2 Bibliotecas de Machine Learning

O projeto se beneficia substancialmente do uso de PyTorch, uma biblioteca de aprendizado de máquina que oferece excelente flexibilidade e eficiência, especialmente útil no treinamento de modelos de deep learning. Complementando o PyTorch, utilizamos Torch-Geometric e PyTorchTime, que são especializadas em redes neurais para grafos e análise de séries temporais, respectivamente, aprimorando significativamente a capacidade do modelo NeSS de lidar com dados complexos e temporais.

O projeto se beneficia substancialmente do uso de PyTorch, uma biblioteca de aprendizado de máquina profundo que oferece excelente flexibilidade e eficiência. Complementando o PyTorch, utilizamos Torch-Geometric, especializada em redes neurais para grafos, para facilitar a utilização de grafos. Para a avaliação de outros modelos, recorremos às bibliotecas PyTorchTime, utilizada especificamente para o modelo InceptionTime, e sktime, empregada para trabalhar com o modelo Rocket. Também utilizamos as bibliotecas scikit-learn e xgboost, que fornecem implementações de diversos modelos de aprendizado de máquina.

6.1.3 Manipulação e Visualização de Dados

Para a manipulação e análise de dados, recorremos a bibliotecas como Numpy, Pandas e Polars, que são instrumentos poderosos para manipulação de grandes bases de dados de forma eficiente. Em conjunto com H5py e PyArrow, essas ferramentas nos permitem lidar eficientemente com formatos de dados como HDF5 e Parquet. Além disso, a visualização de dados é facilitada pelo uso de Matplotlib e Seaborn, que nos ajudam a interpretar os resultados dos modelos de maneira clara e intuitiva.

6.1.4 Desenvolvimento e Gerenciamento de Experimentos

A gestão do código e o monitoramento dos experimentos são realizados com a ajuda de Git e Wandb, respectivamente. O Git é uma ferramenta indispensável para o controle de versão, enquanto o Wandb oferece uma plataforma para o rastreamento de experimentos, essencial para a análise e otimização do desempenho do modelo.

6.1.5 Desenvolvimento Web e APIs

Para a API do projeto, utilizamos FastAPI e Uvicorn, que oferecem uma maneira eficiente e escalável de construir e gerenciar APIs web, facilitando a interação com o modelo NeSS em ambientes de produção.

6.1.6 Geração de Dados Simulados

Finalmente, o projeto se beneficia do uso do Dynasim, um simulador utilizado para a geração de dados da plataforma de petróleo, essenciais para treinar e validar nosso modelo.

6.2 Análise do impacto do tamanho das janelas

Este experimento foi conduzido visando investigar como diferentes tamanhos de janelas temporais, conforme descritas na subseção 5.3.1, afetam o desempenho do nosso

Tabela 1 – Desempenho do modelo de classificação variando o tamanho da janela temporal

Tamanho da janela	Acurácia	F1
3 horas	0.9957	0.9950
4 horas	0.9964	0.9959
5 horas	0.9964	0.9959
6 horas	0.9956	0.9949

modelo. A principal hipótese testada era de que janelas curtas podem ser insuficientes para capturar adequadamente a dinâmica do sistema da plataforma de petróleo, em particular, devido à alta quantidade de ruído presente nas séries temporais utilizadas.

Conforme descrito na subseção 5.3.1, os dados de cada simulação foram inicialmente divididos em várias janelas de duração fixa de 5 horas, com um passo de 30 minutos entre janelas consecutivas. Para testar nossa hipótese, variamos sistematicamente o tamanho dessas janelas temporais entre 3 e 6 horas e analisamos o impacto dessa variação no desempenho do modelo.

Resultados e Análise

Contrariando nossas expectativas iniciais, os resultados, detalhados na Tabela 1, mostraram que o tamanho da janela escolhido teve um impacto limitado no desempenho do modelo. Apesar das variações nos tamanhos das janelas, o desempenho do NeSS manteve-se relativamente constante, sugerindo que o modelo é robusto em relação a essa variável específica. Essa descoberta foi surpreendente, pois esperávamos que o tamanho da janela fosse um fator crítico. Ela indica que o NeSS é capaz de lidar eficientemente com diferentes quantidades de dados, mantendo sua precisão e confiabilidade. Esses resultados suportam a escolha da duração de janela de 5 horas, que obteve resultados marginalmente melhores que os outros tamanhos de janela.

6.3 Análise do impacto da codificação das séries temporais

Este experimento teve como objetivo avaliar como a dimensão do espaço latente, gerado pelo Codificador Temporal, influencia o desempenho do modelo. O Codificador Temporal codifica cada instante de tempo utilizando uma representação vetorial, como descrito na seção 5.4, e associa essa representação vetorial do tempo aos pontos das séries temporais.

A hipótese que levou a este experimento era que a dimensão da codificação vetorial poderia afetar a quantidade de informações disponíveis para o modelo a respeito das séries temporais de entrada, influenciando assim a sua capacidade de aprendizado e previsão. Uma representação vetorial com maior dimensionalidade poderia teoricamente capturar

Tabela 2 – Desempenho do modelo de classificação variando o tamanho da codificação das entradas

Dimensão da codificação (por série)	Acurácia	F1
16	0.9978	0.9975
32	0.9985	0.9984
64	0.9934	0.9926

uma quantidade maior de informações temporais, levando potencialmente a previsões mais precisas. No entanto, também consideramos que o impacto dessa codificação poderia não ser muito significativo, dada a natureza dos dados utilizados (todos na mesma escala de tempo e sem dados faltantes), que já são bem condicionados para análise temporal.

Para testar essa hipótese, variamos a dimensão dos vetores produzidos pelo Codificador Temporal e observamos o impacto dessa variação no desempenho de nosso modelo. Avaliamos como diferentes dimensões afetam a capacidade do modelo de capturar e utilizar informações temporais relevantes para a previsão do estado das linhas de ancoragem.

Resultados e Análise

A análise dos resultados deste experimento, detalhados na Tabela 2, mostrou que o NeSS é capaz de extrair e utilizar eficientemente as informações codificadas pelo Codificador Temporal, sem grandes influências do tamanho da codificação utilizada. Essa descoberta sublinha que, pelo menos para os nossos dados de entrada, o modelo é capaz de extrair informações suficientes dos dados sem grande ajuda do codificador temporal. No entanto, como mencionado acima e na seção 5.4, esse resultado pode não se reproduzir com dados mais complexos, nos quais podem haver dados faltantes.

6.4 Estimador de Período Natural

6.4.1 Treinamento do estimador de período natural

O estimador de período natural consiste em uma rede neural LSTM treinada em dados sintéticos com períodos naturais conhecidos, conforme descrito na subseção 5.4.3. Após o treinamento nos dados sintéticos, o estimador apresentou um RMSE de 30,4 segundos. Este resultado indica um erro ainda alto na estimativa dos períodos de oscilação, considerando a natureza sintética dos dados e os períodos reais de 200 segundos, mas pode se justificar pela alta quantidade de ruído presente nos dados de treinamento.

Tabela 3 – Desempenho do modelo de classificação variando o coeficiente de auxílio do estimador (λ)

λ	Acurácia	F1	λ	Acurácia	F1
11×10^{-4}	0.9993	0.9992	5×10^{-4}	0.9964	0.9959
9×10^{-4}	0.9993	0.9992	3×10^{-4}	0.9971	0.9967
7×10^{-4}	0.9956	0.9951	0	0.9964	0.9959

6.4.2 Análise do impacto do estimador de período natural para auxiliar o treinamento do NeSS

Este experimento foi projetado para avaliar o impacto da integração de um estimador de período natural no processo de treinamento do NeSS. Neste experimento, a função de custo é modificada para levar consideração não apenas a predição de estado de linha (rompida/intacta), mas também as predições de período de oscilação geradas pelo NeSS, conforme descrito na subseção 5.4.3.4.

A hipótese central por trás deste experimento era que o auxílio do estimador de período natural poderia aprimorar a capacidade do NeSS de modelar as dinâmicas temporais complexas do sistema. Esperava-se que, ao treinar o modelo para predizer também os períodos de oscilação da plataforma, o modelo fosse incentivado a desenvolver uma modelagem mais profunda e precisa da dinâmica do sistema. Essa hipótese foi fundamentada em particular no trabalho de (COSTA et al., 2021), que identificou a importância do período de oscilação na detecção de ruptura de linhas de amarração.

Neste experimento, o modelo foi treinado com diversos valores do parâmetro λ, que controla o peso da função de custo auxiliar, conforme descrito na Equação 5.2.

Resultados e Análise

Os resultados do experimento, detalhados na Tabela 3, indicaram uma melhoria marginal no desempenho do modelo com a inclusão do auxílio do estimador. Embora o auxílio do estimador no treinamento não tenha levado a um aumento drástico na acurácia e F1 das previsões, os resultados indicam que o auxílio pode ser útil em problemas mais difíceis, em que o NeSS puro tenha dificuldades de aprendizado.

6.5 Resultados

Para validar o NeSS, comparamos o seu desempenho com outros modelos mencionados nos trabalhos relacionados: modelos utilizando o Rocket e o InceptionTime. Também utilizamos uma rede neural LSTM simples como um modelo básico de predição com base em séries temporais. Todos os modelos foram treinados e avaliados no mesmo conjunto de

Tabela 4 – Resultados do NeSS em comparação com outros modelos. Em negrito, os melhores valores para cada métrica.

Modelo	Acurácia	F1
NeSS	0.9964	0.9959
NeSS com estimador de período	**0.9993**	**0.9992**
Rocket + regressão logística	0.9971	0.9967
Rocket + XGBoost	0.9891	0.9877
InceptionTime	0.9964	0.9959
LSTM	0.9359	0.9252

dados, que consiste em 6560 janelas, com 80% (5248 janelas) destinadas ao treinamento e 20% (1312 janelas) ao teste. Além disso, os dados compreendem 50% de simulações com quebra de linhas de amarração (sempre no mesmo grupo de linhas) e 50% sem quebra, refletindo um cenário equilibrado de condições. Quanto ao tempo de treinamento, o NeSS demonstrou eficiência, completando o processo em aproximadamente 5 minutos.

Os resultados comparativos são mostrados na Tabela 4. O NeSS obteve um desempenho comparável aos melhores modelos da análise, especificamente o Rocket com regressão logística e o InceptionTime, com as métricas de Acurácia e F1 muito próximas. Isso demonstra a eficácia do NeSS em contextos de previsão em séries temporais, estando ao nível de outras abordagens avançadas. A versão do NeSS que inclui o auxílio do estimador de períodos naturais alcançou os melhores resultados entre todos os modelos testados, embora a diferença seja marginal. Esta observação sugere que a integração do estimador de período natural possa oferece uma leve vantagem em termos de desempenho.

Em contraste, a rede neural LSTM apresentou resultados inferiores aos outros modelos. Este desempenho mais baixo destaca a complexidade do problema e a necessidade de abordagens mais sofisticadas, como as incorporadas no NeSS e nos modelos Rocket e InceptionTime.

7 Considerações Finais

Neste capítulo final, consolidamos as realizações, contribuições e limitações do nosso estudo, delineando as conclusões-chave e propondo direções para futuras pesquisas na área.

7.1 Conclusão

Este trabalho abordou duas hipóteses principais: a eficácia das transformadas de Fourier na identificação de rompimentos em linhas de amarração (Hipótese 1) e o benefício de um estimador espectral especialista na convergência da rede (Hipótese 2). Nossos resultados indicam que o modelo espectral, centrado na Hipótese 1, alcança um desempenho comparável a outros modelos de estado da arte, embora não supere significativamente esses modelos. Quanto à Hipótese 2, observamos um aumento marginal no desempenho com o uso do estimador espectral, mas a melhoria foi limitada. Assim, para ambas as hipóteses, desenvolvemos modelos com desempenho equivalente ao estado da arte.

Contudo, devido à complexidade do NeSS e à existência de modelos mais simples com desempenhos semelhantes, a adoção desses modelos alternativos pode ser mais benéfica, considerando a possibilidade de melhor generalização e robustez. Uma limitação importante deste projeto reside na complexidade reduzida dos dados utilizados, que apresentam menos ruído e ausência de dados faltantes em comparação com dados reais, limitando o escopo da nossa análise e a aplicabilidade dos modelos em cenários reais.

7.2 Trabalhos Futuros

Para avançar na pesquisa iniciada com este projeto, propomos várias direções para trabalhos futuros:

Análise em Dados Mais Complexos Uma abordagem essencial seria treinar e avaliar o NeSS em conjuntos de dados submetidos a maiores ruídos e dados faltantes. Em cenários com dados mais complexos, onde o desempenho dos modelos tende a diminuir, os mecanismos específicos utilizados pelo NeSS, como a operação no domínio espectral e o treinamento guiado por um especialista, poderiam demonstrar vantagens mais significativas. Isso permitiria uma reavaliação mais precisa das hipóteses sob condições mais desafiadoras e próximas da realidade.

Comparação de Robustez a Ruídos Com dados mais complexos, seria interessante realizar uma análise comparativa da robustez dos diferentes modelos ao ruído, fornecendo insights sobre como eles se comportam em cenários mais realistas. Esse trabalho expandiria a análise de robustez ao ruído realizada por (SILVA et al., 2022), incluindo o NeSS como parte do estudo comparativo.

Exploração de Problemas Mais Complexos Além de melhorar a qualidade dos dados, outro caminho seria abordar problemas mais complexos, como a identificação do grupo específico de linhas em que ocorreu o rompimento, aumentando a relevância e o desafio da tarefa.

Testes em Dados Reais Um passo importante seria a aplicação e avaliação dos modelos em dados reais de plataformas de petróleo, para validar a eficácia e a aplicabilidade prática dos modelos desenvolvidos.

Referências

BEGLI, M.; DERAKHSHAN, F.; KARIMIPOUR, H. A Layered Intrusion Detection System for Critical Infrastructure Using Machine Learning. In: *2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*. Oshawa, ON, Canada: IEEE, 2019. p. 120–124. ISBN 978-1-72812-440-7. Citado na página 17.

BEYER, J. et al. Environmental effects of the Deepwater Horizon oil spill: A review. *Marine Pollution Bulletin*, v. 110, n. 1, p. 28–51, set. 2016. ISSN 0025-326X. Citado na página 15.

BOX, G. E. P.; JENKINS, G. M. *Time Series Analysis: Forecasting and Control*. [S.l.]: Holden-Day, 1976. ISBN 978-0-8162-1104-3. Citado 2 vezes nas páginas 16 e 35.

Floating Production Mooring Integrity JIP - Key Findings, All Days de *OTC Offshore Technology Conference*, (OTC Offshore Technology Conference, All Days). OTC-17499-MS p. Disponível em: <https://doi.org/10.4043/17499-MS>. Citado 2 vezes nas páginas 15 e 19.

CAO, D. et al. Spectral temporal graph neural network for multivariate time-series forecasting. In: LAROCHELLE, H. et al. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 17766–17778. Citado 4 vezes nas páginas 7, 16, 35 e 36.

CHO, K. et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1724–1734. Citado 2 vezes nas páginas 16 e 30.

Comissão Europeia. *Comunicação Da Comissão Ao Parlamento Europeu, Ao Conselho Europeu, Ao Conselho, Ao Comité Económico e Social Europeu e Ao Comité Das Regiões*. Bruxelas, 2018. Citado na página 23.

COSTA, J. L. D. M. et al. FPSO Mooring Line Integrity Supervising System Based on Motion Data and Natural Frequency Estimation. In: . American Society of Mechanical Engineers Digital Collection, 2021. Disponível em: <https://dx.doi.org/10.1115/OMAE2021-62991>. Citado 2 vezes nas páginas 36 e 51.

Industry Survey of Past Failures, Pre-emptive Replacements and Reported Degradations for Mooring Systems of Floating Production Units, Day 4 Thu, May 08, 2014 de *OTC Offshore Technology Conference*, (OTC Offshore Technology Conference, Day 4 Thu, May 08, 2014). D041S047R002 p. Disponível em: <https://doi.org/10.4043/25273-MS>. Citado na página 19.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-term Memory. *Neural computation*, v. 9, p. 1735–80, dez. 1997. Citado 2 vezes nas páginas 16 e 30.

KIPF, T. *How Powerful Are Graph Convolutional Networks?* 2016. Http://tkipf.github.io/graph-convolutional-networks/. Citado na página 16.

LEA, C. et al. Temporal Convolutional Networks for Action Segmentation and Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 156–165. Citado na página 30.

LIM, B.; ZOHREN, S. Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Royal Society, v. 379, n. 2194, p. 20200209, fev. 2021. Citado na página 16.

MA, K.-T. et al. Chapter 1 - Introduction. In: MA, K.-T. et al. (Ed.). *Mooring System Engineering for Offshore Structures*. [S.l.]: Gulf Professional Publishing, 2019. p. 1–18. ISBN 978-0-12-818551-3. Citado na página 15.

PROAKIS, J. G.; MANOLAKIS, D. G. *Digital Signal Processing: Principles, Algorithms, and Applications*. 3. ed. ed. London: Prentice-Hall International (UK), 1996. (Prentice Hall International Editions). ISBN 978-0-13-394289-7. Citado na página 29.

RICAUD, B. et al. Fourier could be a data scientist: From graph Fourier transform to signal processing on graphs. *Comptes Rendus Physique*, No longer published by Elsevier, v. 20, n. 5, p. 474–488, jul. 2019. ISSN 1631-0705. Citado na página 29.

SAAD, A. M. et al. Using neural network approaches to detect mooring line failure. *IEEE Access*, v. 9, p. 27678 – 27695, 2021. Cited by: 15; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85101476957&doi=10.1109%2fACCESS.2021.3058592&partnerID=40&md5=773a55f303056779ca215ad44953ac79>. Citado na página 15.

SILVA, A. P. S. et al. Machine learning for noisy multivariate time series classification: A comparison and practical evaluation. In: *Anais Do XIX Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2022)*. Sociedade Brasileira de Computação - SBC, 2022. p. 682–693. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/22823>. Citado 2 vezes nas páginas 37 e 54.

SIMS, C. A. Macroeconomics and Reality. *Econometrica*, [Wiley, Econometric Society], v. 48, n. 1, p. 1–48, 1980. ISSN 0012-9682. Citado na página 35.

TUMRATE, C. S. et al. Evolutionary Computation Modelling for Structural Health Monitoring of Critical Infrastructure. *Archives of Computational Methods in Engineering*, v. 30, n. 3, p. 1479–1493, abr. 2023. ISSN 1886-1784. Citado na página 17.

VELIČKOVIĆ, P. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, v. 79, p. 102538, abr. 2023. ISSN 0959-440X. Citado na página 28.

ZHANG, B. et al. Chapter 21 - Marine Oil Spills—Oil Pollution, Sources and Effects. In: SHEPPARD, C. (Ed.). *World Seas: An Environmental Evaluation (Second Edition)*. [S.l.]: Academic Press, 2019. p. 391–406. ISBN 978-0-12-805052-1. Citado na página 15.

Apêndices

APÊNDICE A – Treinamento de Redes Neurais

Os pesos e viés descritos na Seção 3 são os parâmetros da rede neural. O objetivo da rede neural é aprender os parâmetros que permitirão que ela execute a tarefa desejada. O processo de aprendizagem dos parâmetros é chamado de treinamento.

Existem dois métodos principais para treinar uma rede neural: aprendizado supervisionado e aprendizado não supervisionado. O método de aprendizado não supervisionado é usado quando a saída desejada não é conhecida. A rede é treinada para encontrar padrões nos dados. Exemplos de aprendizado não supervisionado são agrupamento e redução de dimensionalidade.

No aprendizado supervisionado, a rede é treinada usando um conjunto de dados $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ composto de entradas $\mathbf{x}_i \in \mathcal{X}$ e suas saídas correspondentes $\mathbf{y}_i \in \mathcal{Y}$, onde $\mathcal{X}$ é o espaço de entrada e $\mathcal{Y}$ é o espaço de saída. A função que a rede está tentando aprender é chamada de função alvo e é denotada por $f : \mathbf{x} \to \mathbf{y}$. O objetivo da rede é aproximar a função alvo f aprendendo os parâmetros da rede. Os parâmetros são aprendidos minimizando uma função de perda $\mathcal{L}$, que mede a diferença entre a saída da rede e a função alvo. O problema de otimização pode então ser formulado como:

$$\min_{\theta} \mathcal{L}(\mathbf{y}, h_\theta(\mathbf{x})) \tag{A.1}$$

onde θ é o conjunto de parâmetros da rede, h_θ é a função hipótese e $\mathcal{L}$ é a função de perda.

A função de perda depende da tarefa que a rede está tentando executar. Por exemplo, em classificação, a função de perda é a perda de entropia cruzada, e em regressão, a função de perda é o erro quadrático médio.

O problema de otimização na equação A.1 normalmente é resolvido usando o algoritmo de descida de gradiente e suas variações. O algoritmo de descida de gradiente é um algoritmo iterativo que atualiza os parâmetros na direção do gradiente negativo da função de perda. O algoritmo básico de descida de gradiente é dado pela equação A.2:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}(\mathbf{y}, h_\theta(\mathbf{x})) \tag{A.2}$$

onde θ_t é o conjunto de parâmetros na iteração t, α é a taxa de aprendizagem e $\nabla_\theta \mathcal{L}$ é o gradiente da função de perda em relação aos parâmetros.

Outra técnica comum para treinar redes neurais é a retropropagação (backpropagation). É um método para calcular o gradiente da função de perda em relação aos

parâmetros da rede. Baseia-se na regra da cadeia do cálculo e é usado em conjunto com a descida de gradiente. O algoritmo de retropropagação é dado pela equação A.3:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial h_\theta} \frac{\partial h_\theta}{\partial \theta} \tag{A.3}$$

onde $\frac{\partial \mathcal{L}}{\partial h_\theta}$ é o gradiente da função de perda em relação à saída da rede e $\frac{\partial h_\theta}{\partial \theta}$ é o gradiente da saída da rede em relação aos parâmetros.