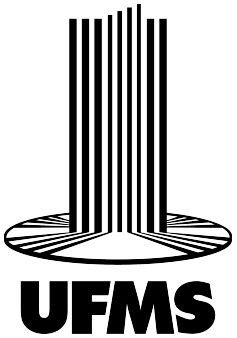
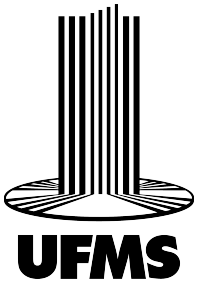


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Avaliação de Modelos de Aprendizado Profundo para Manutenção Preditiva em Sistemas de Refrigeração Industrial

Kellven Dias dos Santos Rodrigues

Campo Grande - MS
22 de Junho de 2026



UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Avaliação de Modelos de Aprendizado Profundo para Manutenção Preditiva em Sistemas de Refrigeração Industrial

Kellven Dias dos Santos Rodrigues

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharelado em Engenharia de Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Orientador: Prof. Dr. Bruno Magalhães Nogueira

Campo Grande - MS
22 de Junho de 2026

Avaliação de Modelos de Aprendizado Profundo para Manutenção Preditiva em Sistemas de Refrigeração Industrial

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharelado em Engenharia de Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Banca Examinadora:

Prof. Dr. Bruno Magalhães Nogueira

Prof. Dr. Jonathan de Andrade Silva

Profa. Dra. Vanessa Araujo Borges

Campo Grande - MS

22 de Junho de 2026

Agradecimentos

Em primeiro lugar, agradeço a Deus, pois sem ele nada disso seria possível. Foi ele quem me ajudou a enfrentar cada obstáculo e adversidade desta jornada com determinação.

Aos meus pais e familiares, que sempre me apoiaram indefinidamente. Um carinho especial à minha mãe, que sempre me incentivou a dedicar-me aos estudos e sempre me apoiou em momentos difíceis. Ao meu pai, que sempre me ofereceu suporte durante todo o processo.

A minha namorada, que nesta reta final seu apoio foi inestimável para que eu concluísse esse trabalho, além de cada conselho e ajuda preciosa.

Aos meus amigos, que estiveram ao meu lado, pela amizade e apoio ao longo do curso.

Aos meus colegas de trabalho, que me apoiaram no tema deste trabalho e que sempre foram solícitos em cada dúvida.

Aos professores, por todos os conselhos, pela ajuda e assistência que ofereceram ao longo do curso. Um carinho especial ao meu orientador, por ter desempenhado tal função com zelo, dedicação, amizade e paciência.

Resumo

A manutenção preditiva em sistemas de refrigeração industrial é fundamental para evitar paradas não planejadas e reduzir o consumo energético. Este trabalho apresenta um estudo comparativo de modelos de Inteligência Artificial voltados à análise de séries temporais provenientes de sensores industriais. Foram avaliadas arquiteturas que variam desde Redes Neurais Recorrentes (LSTM) e modelos lineares de decomposição (DLinear, NLinear), até modelos de última geração baseados em mecanismos de atenção e hierarquia, como NHITS, Autoformer e PatchTST. O objetivo central é determinar a eficácia dessas redes na identificação de comportamentos dessas máquinas e na detecção precoce de anomalias operacionais. Os resultados demonstram que modelos como o PatchTST apresentam vantagens significativas no tratamento de dependências de longo prazo, enquanto modelos lineares oferecem uma base robusta de baixo custo computacional. A pesquisa conclui que a integração dessas IAs permite uma transição eficiente da manutenção preventiva para a preditiva, otimizando a vida útil dos ativos industriais.

Palavras-chaves: Manutenção Preditiva, Séries Temporais, PatchTST, NHITS, DLinear, NLinear, LSTM, Autoformer, Refrigeração Industrial, Deep Learning.

Abstract

Predictive maintenance in industrial refrigeration systems is essential for preventing unplanned downtime and reducing energy consumption. This study presents a comparative analysis of Artificial Intelligence models tailored for time-series data from industrial sensors. This evaluated architectures range from Recurrent Neural Network (LSTM) and linear decomposition models (DLinear, NLinear) to state-of-art models based on attention mechanisms and hierarchy, such as NHITS, Autoformer, and PatchTST. The primary objective is to determine the effectiveness of these networks in identifying machine behavior and enabling the early detection of operational anomalies. The results demonstrate that models like PatchTST offer significant advantages in handling long-term dependencies, while linear models provide a robust, low-computational cost baseline. The research concludes that integrating these AI models facilitates an efficient transition from preventive to predictive maintenance, optimizing the lifespan of industril assets.

Keywords: Predictive Maintenance, Time Series, PatchTST, NHITS, DLinear, NLinear, LSTM, Autoformer, Industrial Refrigeration, Deep Learning.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contextualização, Justificativa (motivação) para o desenvolvimento do trabalho

De acordo com dados do Programa das Nações Unidas para o Meio Ambiente (UNEP), divulgados em cartilha da ABRAVAA (ABRAVA, 2023), a refrigeração industrial é um pilar essencial em setores como alimentos, bebidas e farmacêuticos, representando um alto consumo de energia elétrica, frequentemente chegando a 75% do custo operacional ao longo da vida útil desses sistemas. Dado que esses sistemas representam uma parcela significativa do consumo energético dessas indústrias, é indubitável a necessidade de mecanismos que possam auxiliar na manutenção e eficiência energética dessas máquinas, viabilizando a redução de custos operacionais, maior eficiência de produção, mais segurança e sustentabilidade.

Dessa forma, com o avanço da Indústria 4.0, a quarta revolução industrial, caracterizada pela digitalização dos processos produtivos e pela incorporação de tecnologias avançadas, como Inteligência Artificial, Computação em Nuvem, Big Data e IoT, a manutenção preditiva tem se tornado o principal componente e auxiliador na redução de custos, predição de falhas e diagnóstico de máquinas industriais (ZONTA et al., 2020). Antes, os reparos eram predominantemente realizados através de manutenção corretiva, após a falha da máquina, e manutenção preventiva. Hoje, através de sensores em tempo real e sistemas que proporcionam a integração dessas tecnologias, é possível otimizar a performance, minimizar paradas inesperadas e antecipar custos com manutenção (SCAIFE, 2024) (HANIFI et al., 2025).

Neste ponto, a análise dos dados para uma manutenção preditiva requer um grande esforço, dado que esses sensores geram dados contínuos e, na maioria das vezes, ruidosos, onde padrões normais de operação podem ser confundidos com variações fora do padrão. Além disso, uma anomalia não se caracteriza por ser um valor fora da curva, mas sim um comportamento que é proveniente do que aconteceu horas ou dias antes da máquina começar a apresentar o comportmaneto anômalo. Ademais, métodos tradicionais de análise de séries temporais como ARIMA e modelos estatísticos simples não são capazes de aprender a dependência longa entre as séries de dados (ZHAO et al., 2016).

Diante dessa complexidade analítica, surge a necessidade de empregar arquiteturas mais sofisiticadas de aprendizado profundo que superem as limitações dos métodos convencionais. O Aprendizado Profundo, diferentemente de técnicas comuns de Aprendizado de Máquina, destaca-se pela sua alta capacidade de aprender com dados não estrutu-

rados, além de possuir uma performance muito alta em grandes quantidades de dados. Portanto, a utilização desses modelos torna-se uma potencial alternativa para tarefas de manutenções preditivas.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver e avaliar o desempenho de modelos baseados em arquiteturas de *Deep Learning* para a detecção automática de anomalias operacionais do tipo "carga baixa"em máquinas de refrigeração industrial.

1.2.2 Objetivos Específicos

- Levantar na literatura o estado da arte sobre algoritmos de *deep learning* aplicados especificamente na predição de séries temporais.
- Estruturar e pré-processar o conjunto de dados históricos de sensores do sistema de refrigeração, realizando o tratamento adequado de ruídos e ausência de dados.
- Implementar e treinar diferentes arquiteturas de redes neurais utilizando os dados tratados e aplicando diferentes modelagens de predição.
- Comparar o desempenho das arquiteturas avaliadas por meio de métricas de avaliação (como Erro Quadrado Médio e Erro Médio Absoluto).
- Avaliar os impactos e *trade-offs* das escolhas de modelagem de previsões e das arquiteturas selecionadas na precisão da detecção das anomalias.

1.3 Organização do Trabalho (Resumo dos capítulos)

O Capítulo 2 apresenta a revisão da literatura referente às arquiteturas utilizadas no escopo deste trabalho, englobando desde redes neurais recorrentes até estruturas mais complexas, baseadas em *Transformers* e interpolação hierárquica.

Em seguida, o Capítulo 3 detalha a metodologia de desenvolvimento da pesquisa. Esta seção compreende a descrição do ambiente computacional e das ferramentas utilizadas, bem como a caracterização do *dataset*, os métodos de pré-processamento aplicados e as configurações de hiperparâmetros dos modelos.

O Capítulo 4 apresenta os resultados obtidos e as discussões acerca das predições, analisando o desempenho alcançado pelas diferentes abordagens e arquiteturas implementadas.

Por fim, o Capítulo 4.3 consolida as considerações finais do trabalho, destacando as principais contribuições e sugerindo perspectivas para trabalhos futuros.

2 Fundamentação Teórica

Para compreender plenamente como algoritmos e métodos de Aprendizado Profundo podem auxiliar a manutenção preditiva de máquinas industriais de refrigeração, com base em valores provenientes de sensores instalados diretamente em seus sistemas físicos, é necessária a investigação e estudo dessas arquiteturas sofisticadas de Inteligência Artificial. Os recentes avanços dessas técnicas possibilitaram uma alta produtividade em diferentes áreas de aplicação e redefiniram como o ciclo de operação é gerenciado. Dentre esses, destaca-se os modelos baseados em Redes Neurais Recorrentes capazes de processar sequências de valores, assim como modelos *State-of-art* baseados na arquitetura *Transformers* produziram uma revolução com seus mecanismos sofisticados de processamento paralelo com a técnica de *Multi-head Attention*.

Portanto, este capítulo apresenta, de forma progressiva, os principais elementos técnicos e conceituais das inovações e mecanismos dessas arquiteturas. A seção 2.1 define a arquitetura das Redes Neurais Recorrentes e, em especial, a arquitetura de redes chamadas de *Long Short Term Memory* (**LSTM**), que possibilitaram o processamento sequencial de dados. Em seguida, na seção 2.2 são apresentados dois modelos extremamente relevantes baseados em decomposição linear, **DLinear** e **NLinear**, que desafiaram a complexidade desnecessária dos *Transformers* em séries temporais simples. Já a seção 2.3 conceitua as inovações advindas de modelos *Transformers*, que introduziram os mecanismos de atenção que revolucionaram o processamento sequencial ao permitir o treinamento paralelo e capturar dependências de longo alcance entre tokens de entrada. Nesta, destacam-se os modelos **Autoformer**, desenvolvido para superar as limitações tradicionais dos Transformers em relação à complexidade computacional, e o modelo **PatchTST** que introduziu conceitos de *Patching* e *Channel Independence*. Por fim, na seção 2.4 é apresentado o modelo **NHITS** (*Neural Hierarchical Interpolation*), baseados em Hierarquia e Interpolação.

2.1 Redes Neurais Recorrentes (RNNs)

Redes Neurais Recorrentes são uma família de redes neurais especializadas no processamento de uma sequência de valores $x_1, x_2, \cdots, x_\tau$, como pontos de dados em uma série temporal. Diferentemente de modelos tradicionais de redes neurais *fully connected*, que possuem um parâmetro diferente para cada entrada, as RNNs destacam-se por sua técnica de compartilhamento de parâmetros entre a sequência de entrada, permitindo que a rede reconheça padrões e conexões entre diferentes posições no tempo através de uma "memória"interna (estado oculto) que armazena informações dos estados anteriores. Esse mecanismo pode ser mapeado como um sistema dinâmico em que o estado atual h_t

depende do estado anterior h_{t-1} e da entrada atual x_t, possibilitando que os parâmetros sejam compartilhados a cada passo de tempo.

2.1.1 BackPropagation Through Time (BPTT)

O treinamento das RNNs emprega o algoritmo *Backpropagation Through Time* (BPTT), que consiste em "desenrolar"a rede no tempo, tratando cada passo temporal como uma camada de uma rede *feedforward* profunda. A atualização dos pesos recorrentes depende da regra da cadeia aplicada à perda total retrocedendo ao longo da sequência temporal.

O desafio crítico do BPTT surge no cômputo do gradiente para sequências longas. A dependência temporal exige o cálculo de como um estado oculto no instante k influenciou o estado no instante t. Matematicamente, o gradiente completo para os pesos recorrentes W_h num instante t resulta em um produto contínuo de matrizes Jacobianas:

$$\frac{\partial L_t}{\partial W_h} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W_h}$$

A instabilidade do algoritmo reside no termo de produtório. Se os autovalores das matrizes Jacobianas $\frac{\partial h_i}{\partial h_{i-1}}$ forem sistematicamente menores que 1, o produto contínuo tenderá exponencialmente a zero, causando o Desvanecimento do Gradiente (*Vanishing Gradient*). Caso sejam maiores que 1, ocorre a Explosão do Gradiente (*Exploding Gradient*). É exatamente para mitigar essa deficiência matemática estrutural que arquiteturas dotadas de portas de controle de fluxo de informações, como as LSTMs, se tornaram o padrão no processamento de sequências.

2.1.2 Long Short Term Memory

A fim de solucionar problemas de *Vanishing Gradient* ou *Exploding Gradient* , a LSTM surge como uma ferramenta necessária e poderosa para o processamento de longas sequências de entradas. Enquanto RNNs tradicionais combinam a entrada do passo atual com o estado oculto e realizam a aplicação de uma função não linear, como Sigmoid ou Tangente Hiperbólica, as LSTMs têm como seu funcionamento principal a adoção de caminhos no grafo computacional que permitem que a informação flua através do tempo sem se degradar e em "portas"(*gates*) que gerenciam qual informação armazenar em suas células de memória (substituindo os estados ocultos), quando armazená-las e quando esquecê-las.

A arquitetura de uma LSTM (Figura 1), também chamada de **Memória de Longo e Curto Prazo** (*Long Short-Term Memory*), é composta por dois estados distintos de passagem temporal. O *Cell State*(C_t) atua como a "memória de longo prazo", na

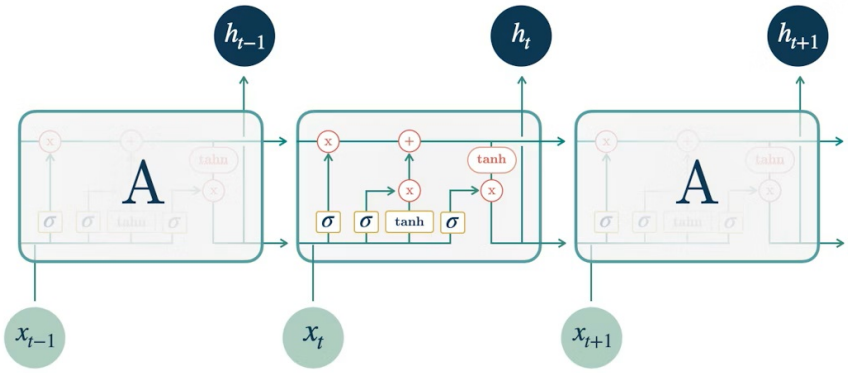


Figura 1 – Arquitetura de uma LSTM, adaptado de (NIXTLA, 2026)

qual a informação flui ao longo de toda a sequência temporal sem que perca suas informações completamente, enquanto que o estado oculto (h_t) atua como a "memória de curto prazo"usado para alimentar a próxima camada da rede e realizar predições com base nos valores obtidos pelos estados ocultos anteriores (h_{t-1}). Esses dois estados são gerenciados a partir de quatro portas fundamentais que controlam o fluxo e armazenamento das informações entre as diversas camadas da rede. Por meio de camadas de ativação sigmoide (σ) e operações de multipli- cação elemento-a-elemento (*Hadamard product* $\odot$), as portas *Forget gate*, *Input gate* e *Output gate* regulam as entradas e saídas entre as camadas da rede.

2.1.2.1 Porta de Esquecimento (Forget Gate)

A primeira etapa da LSTM é gerenciada pelo portão de esquecimento, que tem como objetivo principal determinar qual informação do estado da célula anterior deverá ser descartada. Esse processo é realizado por meio de operações entre a entrada x_t e a saída da célula anterior h_{t-1}, além uma função de ativação Sigmoid σ que mapeia os valores para o intervalo $(0, 1)$. Neste caso, um valor 0 implica que a rede deve "esquecer completamente"aqueles valores, enquanto que um valor igual a 1 implica "reter completamente". A equação de *forget gate* é dada por:

$$\mathbf{F_t} = \sigma(\mathbf{X_t}\mathbf{W_{xf}} + \mathbf{H_{t-1}}\mathbf{W_{hi}} + \mathbf{b_i})$$

onde, $\mathbf{X_t}$ corresponde a entrada, $\mathbf{H_{t-1}}$ é o estado oculto, $\mathbf{W_{xf}}$ e $\mathbf{W_{hf}}$ são parâmetros de pesos e b_i representa o viés. Ou de maneira mais enxuta:

$$f_t = \sigma(\mathbf{W_f} \cdot (h_{t-1}, x_t) + b_f)$$

2.1.2.2 Porta de entrada (Input Gate)

Já nesta etapa, a LSTM utiliza de uma porta de entrada que atua regulando a adição de novas informações na rede através de operações entre x_t e h_{t-1} e um vetor de valores candidatos $\tilde{C}_t$. A célula candidata é criada através de uma função de ativação Tangente Hiperbólica ($tanh$) que mapeia os valores entre -1 e 1:

$$\tilde{C}_t = tanh(\mathbf{W_c}(h_{t-1}, x_t) + b_C)$$

Enquanto isso, os valores de entrada passam por uma função de ativação sigmoid a fim de determinar quais valores devem ser atualizados:

$$i_t = \sigma(\mathbf{W_i}(h_{t-1}, x_t) + b_i)$$

Diante disso, esses dois processos são então combinados por meio de uma operação de produto Hadamard para, por fim, atualizar a *Cell State*:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Em vista disso, é nesse processo que as LSTMs resolvem os problemas de Vanishing Gradient e Exploding Gradient encontrados nas RNNs tradicionais no processamento de sequências longas, uma vez que as operações que atualizam a Célula de memória ocorrem através de operações **aditivas** e não por meio de multiplicações. Isso cria um caminho onde o gradiente pode fluir quase sem alterações durante o algoritmo de BPTT.

2.1.2.3 Porta de Saída (Output Gate)

Por fim, nesta última etapa do processamento da LSTM é realizado o cálculo dos valores de saída e dos valores que serão utilizados pelas camadas posteriores da rede (estado oculto). O seu funcionamento baseia-se na aplicação da função sigmoid σ através de x_t e h_{t-1} a fim de realizar uma filtragem de quais valores do *Cell State* devem ser produzidos. No cálculo do estado oculto, é utilizado uma função $tanh$ que atua como um "compressor", limitando assintoticamente o estado oculto a um intervalo seguro e padronizado, independentemente de quão grande C_t se torne internamente. Por último, o estado oculto h_t é calculado por meio do produto Hadamard da saída com os valores "formatados"do estado da célula C_t, de modo que as equações dessas duas etapas são dadas por:

$$o_t = \sigma(\mathbf{W_o}(h_{t-1}, x_t) + b_o)$$

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

2.2 Modelos de Decomposição Linear

No trabalho *"Are Transformers Effective for Time Series Forecasting?"* (ZENG et al., 2023), a fim de investigar a eficácia de metódos para tarefas de predição de séries temporais de longo prazo (**LTSF**) utilizando métodos baseados no funcionamento de *Transformers*, os autores propuseram um conjunto de modelos lineares extremamente simples, denominado **LTSF-Linear**.

Embora os *Transformers* sejam excelentes em extrair correlações semânticas em textos ou imagens utilizando de seus mecanismos de *self-attention*, eles são inerentemente independentes de permutação, isto é, eles não consideram a ordem temporal. Em tarefas de predição de séries temporais, a ordem dos passos de tempo é um fator significativo e fundamental para a tarefa de predição, uma vez que o passo de tempo t pode impactar profundamente no comportamento de $t+n$, sendo $n > 0$. Portanto, fica claro que a ordem contínua é crucial em tarefas deste tipo, e a aplicação de *Transformers* resulta em perdas de informações temporais.

Diante de tal perspectiva, a aplicação destes modelos lineares de uma única camada quando comparados com as principais arquiteturas estado-da-arte em tarefas complexas de LTSF, superaram significamente em seus resultados obtidos, reduzindo o erro entre 20% e 50%. Destacam-se como descobertas fundamentais a performance dos modelos lineares em entradas longas que, à medida que a janela de observação aumenta, a performance também cresce substancialmente, diferentemente dos Transformers onde o erro aumenta ou sofre com a estagnação. Além disso, ao embaralhar aleatoriamente os dados de entrada, o desempenho dos modelos estado-da-arte quase não oscila, enquanto que nos modelos lineares a performance cai drasticamente, indicando que esses dependem significamente da ordem temporal. Por fim, a implementação de modelos lineares simples consome menos memória e, em termos de processamento, é muito mais eficaz que os Transformers.

2.2.1 Base Linear

O conjunto **LSTF-Linear** baseia-se em uma simples regressão linear que mapeia a janela de histórico L (*look-back window*) diretamente para a janela de predição T (*forecast window*). Portanto, dada uma série de entrada $X \in \mathbb{R}^{L \times C}$, onde C representa o número de variáveis, o modelo prevê os próximos passos $\hat{Y} \in \mathbb{R}^{T \times C}$ por meio de uma transformação direta:

$$\hat{Y} = WX$$

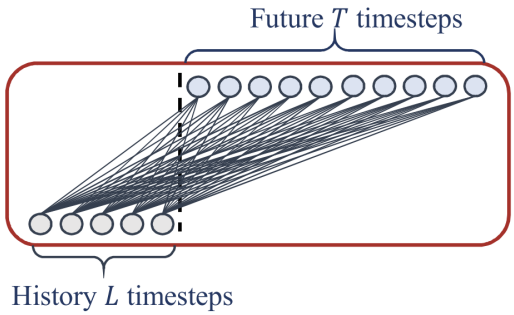


Figura 2 – Base linear. Fonte: (ZENG et al., 2023)

Onde $W \in \mathbb{R}^{T \times L}$ é a matriz de pesos aprendida. Dessa forma, o modelo capta relações temporais aprendendo um peso para cada passo de tempo do histórico em relação a cada passo do futuro (Figura 2).

2.2.2 Arquitetura DLinear e Decomposição de Séries

Diante do exposto, o modelo **DLinear** fundamenta-se adicionalmente na premissa da decomposição clássica de séries temporais. A arquitetura, representada pela Figura 3, utiliza um núcleo de decomposição simples que divide o sinal de entrada em duas componentes principais: a tendência, capturada através de um filtro de média móvel (*moving average kernel*), e a componente sazonal, obtida pela subtração da tendência do sinal original. Portanto, o objetivo central deste modelo é amplificar a capacidade do modelo linear em lidar com dados que possuem tendências e sazonalidades claras. Em um contexto de aplicação em séries temporais provenientes de máquinas industriais de refrigeração, essas características permitem ao modelo aprender o comportamento de longo prazo (tendência) como, por exemplo, o aumento gradual de variações nas correntes elétricas, bem como capturar os ciclos de funcionamento dessas máquinas (sazonalidade).

Diante disso, o seu processamento e saída é dado através da aplicação de duas camadas lineares independentes de um único nível: uma usada para processar apenas a tendência e outra dedicada a processar a sazonalidade. Essa estratégia permite que o modelo capture de forma eficaz padrões distintos de forma isolada, mitigando a perda de informação temporal que frequentemente ocorre em redes que tentam processar o sinal bruto de sensores industriais, onde o ruído de alta frequência pode mascarar tendências de degradação a longo prazo.

Dessa forma, o modelo **DLinear** estabelece que a decomposição explícita de componentes de tendência e sazonalidade, aliada à simplicidade de camadas lineares, é suficiente para superar arquiteturas complexas em diversos cenários de predição de longo prazo. No entanto, uma limitação inerente surge quando a série apresenta características de não-estacionariedade, isto é, quando a média e a variância dos dados dos sensores oscilam significamente devido a mudanças operacionais e sazonais - fenômeno conhecido

como *distribution shift*. A fim de mitigar esssas limitações técnicas, os autores propõem a implementação do modelo **NLinear** que foca na normalização dos dados de entrada em relação ao último passo de tempo observado, permitindo que a rede processe variações relativas em vez de valores absolutos, aumentando a robustez em ambientes industriais dinâmicos.

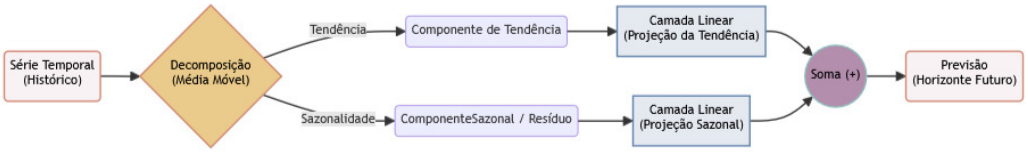


Figura 3 – Arquitetura DLinear

2.2.3 NLinear e o Tratamento de Mudanças de Distribuição

Diante das limitações técnicas do DLinear, o modelo **NLinear** surge como alternativa para *distribution shift*. Em séries temporais frutos de dados econômicos, saúde ou energia, é bastante comum que ocorra mudanças na distribuição entre os dados de treinamento e os dados de teste. Isso significa que a média ou a variância dos dados no futuro difere de maneira expressiva do histórico em que o modelo foi treinado. Diante disso, o modelo utiliza de técnicas de regressão linear e ajustes de distribuição para resolver esses impasses por meio de três processos principais: subtração, regressão linear direta e adição.

A etapa de pré-processamento tem como objetivo central normalizar a sequência em relação ao seu nível mais recente. Dada uma janela histórica de entrada L, o modelo identifica o último valor dessa sequência. A partir disso, subtrai-se este último valor de todos os pontos de dados da sequência de entrada. Ao realizar esse processo, o modelo consegue "zerar"o final da série, proporcionando uma normalização. Após esse processamento inicial, o modelo aplica uma regressão linear direta de múltiplos passos usando uma soma ponderada através de uma única camada linear. O modelo processa cada variável de forma independente, compartilhando os mesmos pesos W. Por fim, após a projeção linear calcular os passos futuros, o modelo soma de volta o mesmo último valor, que foi subtraído durante a fase inicial, a todas as previsões geradas.

Como resultado, as técnicas de ajuste de distribuição implementadas pelo NLinear proporcionam muitas vantagens e eficiência para tarefas de LSTF. Ao subtrair e depois somar o último ponto conhecido, o modelo força e alinha a previsão para se aproximar da distribuição verdadeira mais recente, evitando erros massivos por flutuações de escala. Além disso, a arquitetura é absurdamente eficiente. O número total de parâmetros é dado simplesmente por $T \times L$. Ademais, diferente de redes profundas, o NLinear conecta diretamente a entrada à saída, permitindo a captura de dependências de longo e de curto prazo de maneira eficaz.

Diante do exposto fica claro que a eficácia de modelos simples como **DLinear** e **NLinear** em tarefas de **LSTF** demonstra que a simplicidade estrutural pode, em muitos casos, superar a complexidade algorítmica. No contexto da manutenção preditiva industrial, essa característica é particularmente valiosa, pois permite a captura precisa de ciclos operacionais e degradação dos componentes mecânicos (como motores de compressores). Além disso, a aplicação desses modelos pode auxiliar na adaptação a mudanças bruscas de regime e, possivelmente, na sua utilização diretamente em dispositivos de borda (*edge computing*). Ao contrário de modelos complexos que exigem muita memória, tempo de processamento e comportamento de "caixa preta", os modelos lineares possuem um caminho de sinal $O(1)$ e uma única camada linear, permitindo que o modelo seja embarcado diretamente em CLPs (Controladores Lógicos Programáveis) avançados, gateways IoT ou computadores industriais diretamente no chão de fábrica, processando previsões em tempo real com hardware de baixo custo.

Contudo, embora modelos lineares capturam com excelência as tendências globais e variações de escala, eles podem apresentar limitações ao processar relações multivariadas extremamente complexas e dependências temporais de altíssima resolulção. Para endereçar esses nuances e explorar o máximo da atenção seletiva sobre os dados sensoriais, surge a necessidade de investigar arquiteturas baseadas em **Transformers**, que buscam otimizar o processamento de sequências longas sem abdicar da integridade da ordem temporal.

2.3 Transformers

Diante das evoluções no processamento de sequências de entrada pelas RNNs e por modelos lineares simples, o surgimento de arquiteturas baseadas em modelos **Transformers** (Figura 4) surge como uma alternativa relevante à frente de tarefas de predição de séries temporais e processamento de sequências. O artigo *Attention is All You Need* (VASWANI et al., 2023) introduz originalmente o conceito e a arquitetura base desses modelos. Diferente das Redes Neurais Recorrentes e suas variantes, que processam os dados de forma sequencial e apresentam dificuldades em manter dependências de longuíssimo prazo devido ao afunilamento de informações, os Transformers baseiam-se inteiramente no **mecanismo de atenção** (*self-attention*). Essa mudança de paradigma permite o processamento paralelo de toda a janela temporal de entrada, eliminando a necessidade de recorrência e mitigando problemas clássicos como *Vanish Gradient* discutidos anteriormente na seção 2.1. Embora tenham sido originalmente concebidos para tarefas de Processamento de Linguagem Natual (PLN), a capacidade dessas redes em capturar correlaçãoes complexas em grandes volumes de dados motivou sua adaptação para o domínio das séries temporais, dando origem a modelos otimizados como o **Autoformer** e **PatchTST**, que buscam equilibrar a alta capacidade preditiva com a eficiência dessas arquiteturas.

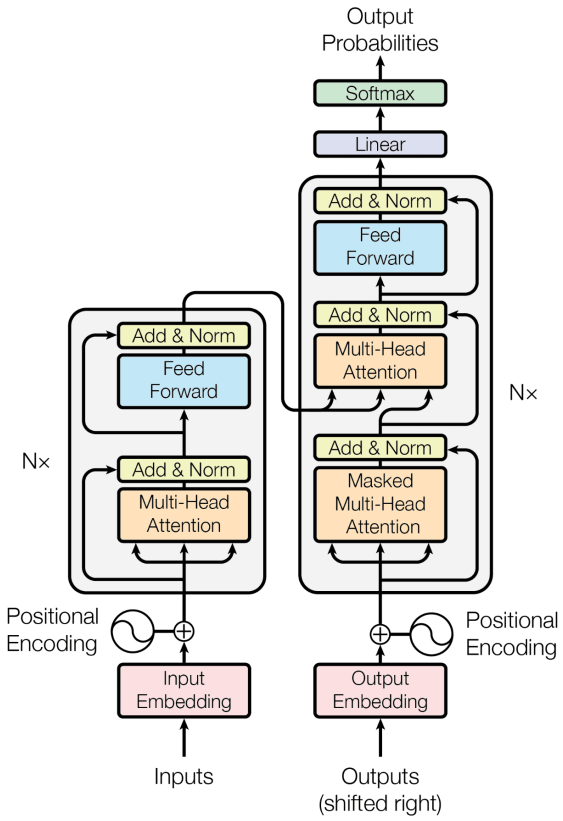


Figura 4 – Arquitetura Transformers. Fonte: (VASWANI et al., 2023)

Diante disso, a etapa inicial da arquitetura Transformers baseia-se na transformação dos dados brutos em representações vetoriais contínuas através das camadas de *Input Embeddings*. O modelo não processa o texto ou a série temporal diretamente, mas sim converte essa entrada em pequenos fragmentos chamados *tokens*. A partir disso, o papel da camada de *embedding* é converter cada um desses tokens em um vetor contínuo de alta dimensionalidade (no artigo original, $d_{model} = 512$) que carrega informações importantes da entrada, representadas por meio de valores numéricos que podem ser aprendidos durante o treinamento. No entanto, como esses modelos processam os dados de forma paralela, a arquitetura utiliza de um mecanismo chamado de *Positional Encoding* para injetar a noção de ordem e de tempo. Originalmente, esse mecanismo consiste em vetores de mesma dimensão que os *embeddings*, cujos valores são gerados por funções senoidais de diferentes frequências.

Esse mecanismo é crucial para tarefas onde a entrada são séries temporais, uma vez que essas dependem essencialmente da noção de ordem de tempo. Deste modo, arquiteturas focadas em séries temporais utilizam camadas de *embedding* adicionais e aprendíveis específicas para codificar esses carimbos de tempo como, por exemplo, o *Timestamp Encoding* implementado na arquitetura Autoformer (subseção 2.3.2).

Após a camada de entrada, os valores são passados adiante para o bloco de *Encoders*, cujo objetivo central é aplicar o algoritmo de *Multi-head attention* e uma rede neural

Position-wise feed-forward afim de extrair e refinar informações relevantes de diferentes palavras em diferentes posições. Para garantir a estabilidade do treinamento e evitar a degradação do gradiente, cada subcamada utiliza conexões residuais (*Add*) seguidas por etapas de normalização de camada (*Norm*). O resultado final do *Encoder* atua como base de conhecimento fundamental para o *Decoder*, que tem a tarefa de gerar a sequência de saída passo a passo.

2.3.1 Mecanismo de Atenção

O mecanismo de *Self-attention* é o núcleo funcional da arquitetura Transformer. Ele permite que o modelo relacione diferentes posições de uma única sequência de dados para computar uma representação rica em dependências e contexto. A atenção pode ser descrita como o mapeamento de uma matriz de consultas (*Queries*) $\mathbf{Q}$ e um conjunto de pares de chaves e valores (*Keys* e *Values*) $\mathbf{K}$, $\mathbf{V}$ para uma saída computada.

O seu funcionamento é consolidado através de um Produto Escalar Escalonado (*Scaled Dot-Product Attention*), representado na Figura 5. A primeira etapa desse processo consite na geração dos valores de *Query*, *Key* e *Value* a partir dos dados das camadas de *embedding*. A partir disso, o modelo calcula o produto escalar entre a matriz de consultas e a matriz de chaves. Um produto escalar alto indica alta similaridade e, portanto, grande similariade entre dois elementos. Os resultados desse produto são divididos por $\sqrt{d_k}$, onde d_k representa a dimensão das chaves. Em seguida, aplica-se uma função *Softmax* sobre os valores escalonados para obter os pesos finais de atenção. Essa função transforma os valores em uma distribuição probabilística de acordo com:

$$\text{Softmax}(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.1}$$

onde, $\mathbf{z}$ representa um vetor de entrada.

Por fim, a matriz de valores V é então multiplicada pelos pesos do *softmax*. O resultado é que valores de tokens irrelevantes ao contexto são multiplicados por pesos próximos de zero, enquanto características cruciais da sequência são amplificadas. A fórmula que descreve este processo é definida como:

$$\textit{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Diante dessa estrutura complexa, foi notado que realizar a equação de atenção uma única vez, utilizando toda a dimensionalidade do modelo inibia a extração de certas características. Para mitigar esse problema, os autores propuseram a técnica de *Multi-head Attention* que subdivide as projeções em h cabeças independentes (*heads*), calculando a atenção de forma paralela e reduzindo a dimensionalidade dentro de cada cabeça:

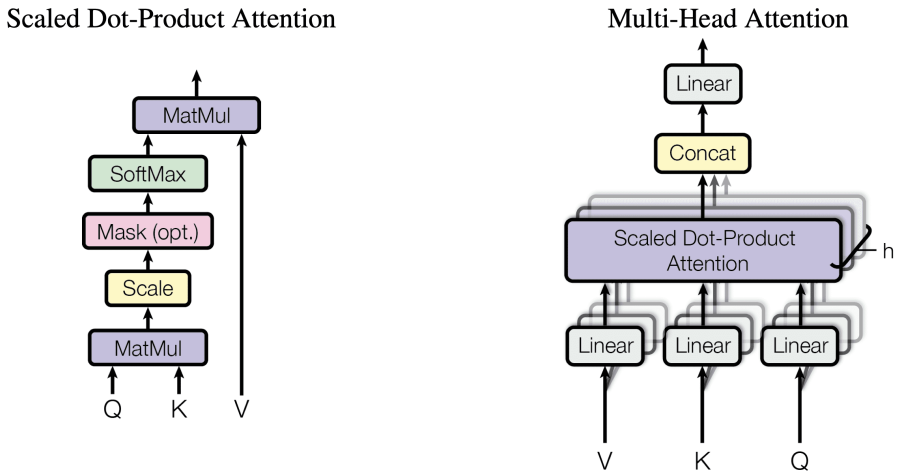


Figura 5 – Esquerda: Atenção Escalonada por Produto Escalar. Direita: Multi-head attention. Fonte: (VASWANI et al., 2023)

$$\text{MultiHead}(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

onde W_i^Q, W_i^K e W_i^V são matrizes de pesos referentes à cada parâmetro e W^O projeta o resultado concatenado de volta para a dimensão original.

Esse mecanismo de múltiplas cabeças permite que o modelo capture diferentes subespaços de representação simultaneamente. No contexto de manutenção preditiva, uma "cabeça"de atenção pode especializar-se em focar em sazonalidades de alta frequência, enquanto outra pode rastrear a degradação do equipamento ao longo do tempo. Entretanto, apesar desses mecanismos inovadores da arquitetura base Transformer a sua aplicação direta no domínio de séries temporais esbarra em um gargalo computacional: a complexidade quadrática $O(L^2)$, intrínseca ao mecanismo de atenção, em relação ao comprimento da sequência de entrada. Além disso, a arquitetura padrão é agnóstica em relação às propriedades estruturais intrínsecas de sinais contínuos, não possuindo mecanismos nativos para separar tendências de degradação e sazonalidades operacionais. Para viabilizar e mitigar esses desafios técnicos para tarefas de **LTSF**, entendeu-se como necessário o desenvolvimento de modelos especialistas como o **Autoformer**, que repensaram o paradigma de adicionar blocos de decomposição e refatoração no mecanismo de atenção, e o **PatchTST** que trouxe inovações relacionadas à própria entrada de dados, agrupando pontos em *patches* para reduzir a dimensão de processamento e isolar ruídos inconvenientes. Nas subseções subseção 2.3.2 e subseção 2.3.3 são discutidas com mais detalhes as especifações dessas arquiteturas especializadas.

2.3.2 Autoformer

Diante das limitações bases da arquitetura fundacional Transformer e do mecanismo original de Atenção, o modelo *Autoformer* (WU et al., 2022) introduz inovações profundas focadas nas particularidades e desafios de séries temporais para previsões de longo prazo por meio da incorporação de decomposição das séries através de um bloco adicional na arquitetura, permitindo que a rede separe e refine progressivamente as informações de tendência de longo prazo diretamente a partir das variáveis ocultas durante todo o processo de previsão. Além disso, o modelo traz consigo a substituição do mecanismo de *self-attention* pelo *Auto-correlation* para possibilitar o cálculo de similaridades entre os atrasos temporais simultaneamente e, ademais, as inovações proporcionaram uma reduzão significativa da complexidade computacional quadrática por uma complexidade de tempo e memória de $O(L \log L)$, onde L representa o comprimento da sequência.

2.3.2.1 Bloco de Decomposição de Séries

Ao invés de decompor a série apenas como um método de pré-processamento, o Autoformer embute a decomposição diretamente em sua arquitetura (Figura 6). Este método permite que a rede realiza a decomposição das variáveis ocultas progressivamente ao longo de todas as camadas. O bloco de decomposição utiliza uma Média Móvel para suavizar flutuações periódicas e extrair componentes de tendências (X_t). Já o componente de sazonalidade (X_s) é extraído por meio da subtração da tendência pela série original.

$$X_t = AvgPool(Padding(X))$$

$$X_s = X - X_t$$

Estas duas equações definem o bloco *SeriesDecomp* na Figura 6 que é então passado através de cada camada do Encoder e Decoder.

2.3.2.2 Encoder e Decoder

O *Encoder* do modelo Autoformer tem como objetivo principal lidar com a sazonalidade. Ele realiza isto através do mecanismo de *Auto-correlation*, que processa a entrada vinda da camada anterior. A partir disso, o resultado do algoritmo de Auto-correlação sofre uma conexão residual e passa pelo bloco de decomposição (*SeriesDecomp*), que extrai e descarta a tendência, mantendo somente a parte sazonal. Essa componente de sazonalidade passa então por uma rede *Feed-Forward*, recebe nova conexão residual e uma nova decomposição e, por fim, gera o estado sazonal final da camada, concentrando um rico histórico das sazonalidades passadas. Por sua vez, o *Decoder* divide a metade final da entrada dos dados históricos em tendência e sazonalidade. A parte que será predita é

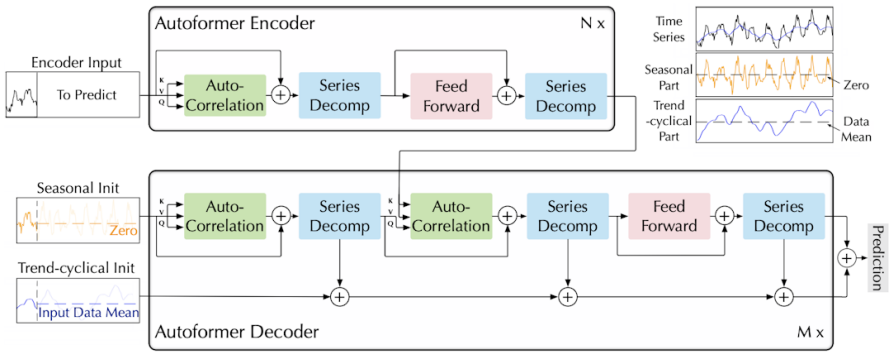


Figura 6 – Arquitetura Autoformer. Fonte: (WU et al., 2022)

preenchida com zeros para a sazonalidade e com a média móvel da série para a tendência. Para isso, esta camada implementa duas etapas de Auto-correlação. A primeira é a *Inner Auto-Correlation*, que atua sobre o estado anterior do Decoder. A segunda é a *Encoder-Decoder Auto-Correlation*, que cruza a informação para integrar o histórico sazonal. Após cada etapa do Decoder, o bloco de decomposição extrai uma nova estimativa de tendência somando-as iterativamente. Por fim, a previsão final do modelo é a simples soma da componente sazonal refinada com a componente de tendência acumulada.

2.3.2.3 Mecanismo de Auto-Correlation e Time Delay Aggregation

Diferentemente da arquitetura base Transformer, o Autoformer inova com a implementação do mecanismo de Auto-Correlação que calcula a similaridade entre sub-séries inteiras, agregando o sinal no nível das fases dos períodos. O seu processo consiste em duas fases fundamentais. A primeira delas consiste em descobrir quais são os períodos subjacentes da série calculando a autocorrelação $R_{XX}(\tau)$, que reflete o quão semelhante a série atual X_t é em relação a si mesma quando atrasada em τ passos temporais $X_{t-\tau}$, de modo que dada uma sequência de tamanho L temos:

$$R_{XX}(\tau) = \lim_{L\to\infty} \frac{1}{L} \sum_{t=1}^{L} X_t X_{t-\tau}$$

Para realizar esse processo de maneira eficiente, o modelo utiliza do **Teorema de Wiener-Khinchin** que, em termos simples, estabelece que a função de autocorrelação de um processo estacionário pode ser calculado através da **Transformada Inversa de Fourier** (iFFT) da sua densidade de potência espectral:

$$S_{Q,K} = \mathcal{F}(Q)\mathcal{F}^*(K)$$

$$R_{Q,K} = \mathcal{F}^{-1}(S_{Q,K})$$

onde, $S_{Q,K}$ é a Densidade Espectral cruzada no domínio da frequência, $\mathcal{F}$ denota a FFT (Fast Fourier Transform), $\mathcal{F}^{-1}$ sua função inversa e $\mathcal{F}^*$ sua operação conjugada.

Isto permite que, ao invés de realizar comparações entre um ponto com seus atrasos de tempo de maneira exaustiva no domínio do tempo, ele passe a realizar o cálculo das comparações de uma só vez no domínio da frequência que é infinitamente mais rápido, de modo que esse mecanismo permite a redução da complexidade fundacional dos Transformers de $O(N^2)$ para $O(N \log N)$.

Já na segunda etapa o mecanismo de *Time Delay Aggregation* atua deslocando a matriz de informações passadas por esses exatos atrasos usando uma operação de *rolling*. Primeiramente, o modelo utiliza a autocorrelação calculada $R_{Q,K}(\tau)$ entre a matriz de consultas Q e a matriz de chaves K para estimar a validade de um determinado atraso de tempo τ. Para garantir eficiência computacional e focar apenas nas periodicidades mais relevantes, o modelo seleciona os k atrasos que possuem os maiores valores autocorrelação (Topk):

$$\tau_1, \ldots, \tau_k = \text{argTopk}_{\tau \in (1,\ldots,L)}[R_{Q,K}(\tau)]$$

onde k é um hiperparâmetro definido como $k = c \times \log L$, sendo $c \in [1, 3]$ e L o comprimento da sequência.

A partir disso, os valores de autocorrelação dos k atrasos mais relevantes são convertidos em uma distribuição de probabilidade utilizando a função Softmax (Equação 2.1):

$$\hat{R}_{Q,K}(\tau_1), \ldots, \hat{R}_{Q,K}(\tau_k) = \text{SoftMax}[R_{Q,K}(\tau_1), \ldots, R_{Q,K}(\tau_k)]$$

Por fim, com os períodos τ_i identificados e com seus pesos $\hat{R}_{Q,K}(\tau_i)$ calculados, o modelo aplica uma operação de *rolling* à matriz de valores V que desloca a série V no tempo por t_i passos a fim de alinhar perfeitamente as sub-séries que compartilham a mesma fase de um período. Após o alinhamento,as sub-séries são agregadas através de uma soma ponderada. Logo, o mecanismo de **Auto-correlação** pode ser denotado como:

$$\text{Auto-Correlation}(Q, K, V) = \sum_{i=1}^{k} \text{Roll}(V, \tau_i)\hat{R}_{Q,K}(\tau_i) \tag{2.2}$$

Ao alinhar essas sub-séries defasadas com o momento atual e combiná-las através de uma soma ponderada, o modelo consegue projetar o futuro baseando-se em ciclos contínuos de dados. Na prática, no contexto de predição em sensores de máquinas industriais isso significa que a rede não tenta prever o próximo valor olhando para pontos de dados isolados, mas sim sobrepondo as fases exatas de ciclos operacionais históricos que são mais correlatos ao estado atual do equipamento. Contudo, embora o modelo consiga isolar

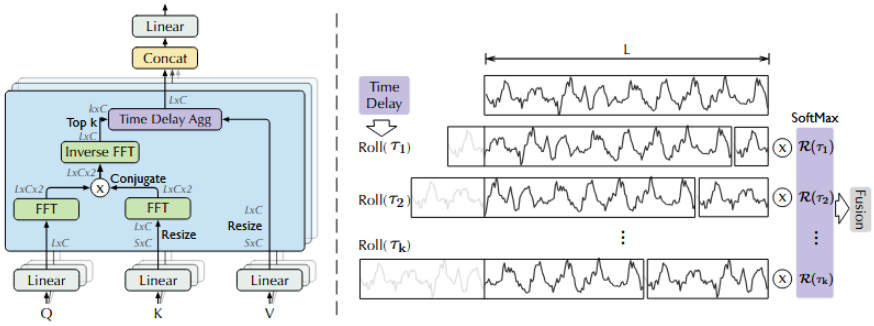


Figura 7 – Esquerda: Autocorrelação. Direita: *Time Delay Aggregation*. Fonte: (WU et al., 2022)

esses ciclos através de seu mecanismo inovador de autocorrelação, o modelo ainda opera fundamentalmente em um nível de processamento ponto a ponto. Para suprir essa lacuna e extrair representações mais ricas, a arquitetura **PatchTST** propõe uma mudança de paradigma ao agrupar passos de tempo adjacentes em blocos (os *patches*). Essa técnica de pré-processamento aliada à independência de canais (onde cada variável é tratada de forma isolada) reduz drasticamente o tamanho efetivo da sequência de entrada, estabelecendo um novo nível de eficiência e precisão para previsões de longo prazo, além de preservar a continuidade física e o contexto local do sinal.

2.3.3 PatchTST

Embora o modelo Autoformer tenha conseguido superar desafios técnicos e inovado com sua engenharia de autocorrelação e *Time Delay Aggregation* para séries temporais, o artigo (ZENG et al., 2023) demonstrou que os modelos discutidos na seção 2.2, modelos lineares simples, trouxeram muitas questões acerca da usabilidade das robustas arquiteturas baseadas em Transformers para tarefas de predição de longo de prazo. Para responder a essas questões, no artigo *A Time Series is Worth 64 Words: Long-Term Forecasting With Transformer* (NIE et al., 2023) os autores apresentam um novo modelo intitulado **PatchTST** que tem como objetivo definir se Transformers podem ser realmente úteis em tarefas de **LTSF**. Diferente de modelos como o Autoformer (subseção 2.3.2), que tentam modificar o mecanismo de atenção internamente para lidar com a complexidade computacional, o PatchTST (Figura 8) foca em como os dados são representados e alimentados na rede, introduzindo duas inovações técnicas para a arquitetura original Transformer: *Patching*, que segmenta a entrada em blocos adjacentes e o *Channel-Independence* que processa isoladamente cada série em um conjunto de dados multivariável.

2.3.3.1 Patching

Ao contrário de arquiteturas tradicionais Transformers aplicadas a séries temporais que tratam cada passo de tempo como um *token*, o PatchTST argumenta que um único ponto temporal não detém de significado semântico isolado e que focar em pontos

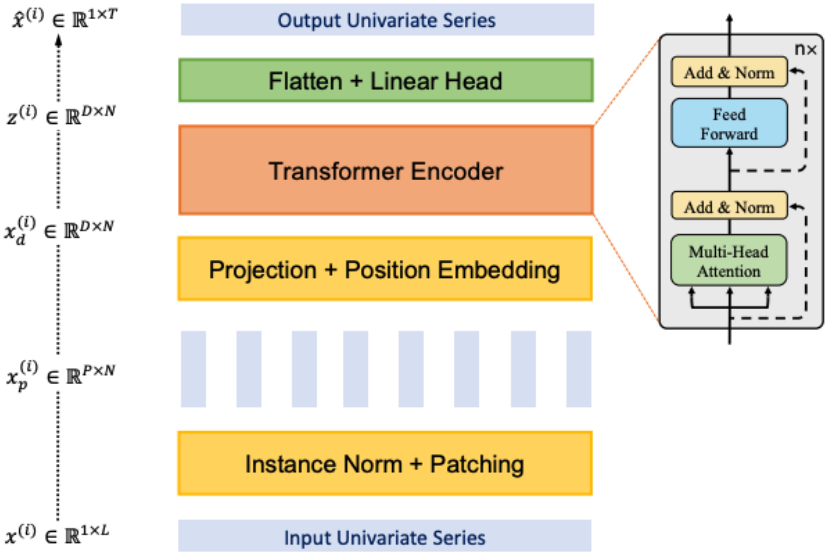


Figura 8 – Arquitetura PatchTST. Fonte: (NIE et al., 2023)

individuais ignora a estrutura local. Além disso, ao usar pontos individuais esses modelos contribuem para que a complexidade de memória e tempo seja $O(L^2)$. À frente disso, o mecanismo de *Patching* determina que a série temporal de entrada com janela histórica L seja dividida em blocos (*patches*) de tamanho P, separados por um passo (*stride*) S, de modo que essa divisão gera uma nova sequência formada por N *patches* onde:

$$N = \left\lfloor \frac{(L-P)}{S} + 2 \right\rfloor$$

Essa divisão em "sub-séries"introduz uma vantagem computacional, uma vez que o número de *tokens* de entrada cai drasticamente de L para N. Consequentemente, a complexidade computacional e de memória do mapa de atenção é reduzida quadraticamente por um fator de S. Além disso, essa eficiência permite que o modelo processe janelas históricas muito maiores sem estourar a memória da GPU, enquanto retém a riqueza semântica local dentro de cada *patch*. Após essa divisão em fragmentos, o fluxo de processamento ocorre inteiramente dentro da arquitetura baseada no clássico Transformer Encoder.

2.3.3.2 Independência de Canais (*Channel-Independence*)

Além do mecanismo de *Patching*, em séries temporais multivariadas a abordagem padrão adotada por modelos tradicionais é usar todas as variáveis em um determinado momento de tempo e projetá-las em um único *token*. Em contrapartida, o PatchTST implementa uma separação entre as variáveis, onde uma série com M variáveis é separada em M séries temporais univariadas (Figura 9). A partir disso, cada uma dessas M séries é tratada de forma independente pela arquitetura, mas com todas elas compartilhando os mesmos pesos (*embeddings* e matrizes de atenção) da rede.

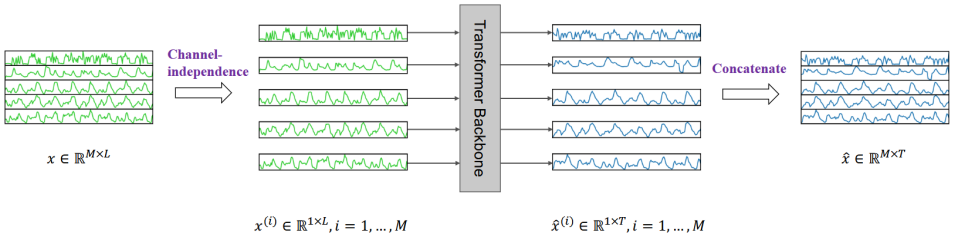


Figura 9 – PatchTST: Independência de Canais. Fonte: (NIE et al., 2023)

Esse mecanismo, ao tratar de forma isolada cada variável, permite que diferentes séries aprendam seus próprios padrões de atenção específicos para suas predições, reduzindo *overfitting* rápido, já que não exige uma quantidade enorme de dados para aprender correlações cruzadas, focando inteiramente no eixo temporal. Além disso, essa representação permite que a série não seja corrompida com ruídos excessivos em uma variável específica e se propague e corrompa toda a representação daquele espaço latente. No contexto prático da manutenção preditiva esse comportamento se destaca pela capacidade de processar longos históricos de dados de sensores isolando anomalias locais dos ciclos normais de operação. Contudo, a altíssima taxa de amostragem dos sensores industriais também incentiva a busca de alternativas focadas em diferentes escalas de tempo. É nesse cenário que modelos baseados em percepção de múltiplas frequências ganham força, abrindo espaço para a exploração do **N-HiTS**, que propõe uma inovação com um mecanismo de interpolação hierárquica.

2.4 Modelos de Interpolação Hierárquica

Os modelo **N-HiTS** (*Neural Hierarchical Interpolation for Time Series*) introduzido por (CHALLU et al., 2022) foi desenvolvido baseando-se nas inovações técnicas implementadas pelo modelo **N-BEATS** (ORESHKIN et al., 2020). Ao provar de forma contundente que arquiteturas baseadas somente em **Perceptrons de Múltiplas Camadas** (MLPs), o modelo N-BEATS pavimentou o caminho para que o N-HiTS expandisse esse paradigma. A partir disso, esse modelo trouxe à tona técnicas de amostragem multi-taxa e interpolação hierárquica a essa base MLP de forma que foi possível a mitigação da explosão dimensional em predições de longo horizonte, consolidando-se como uma alternativa robusta para o processamento de tarefas de LTSF, capturando desde micro-flutuações de alta frequência até tendências de degradação a longo prazo em regimes de operação complexos. Deste modo, na subseção subseção 2.4.1 discutimos como o N-BEATS estruturou a base para que o modelo N-HiTS pudesse desenvolver sua arquitetura discutida, posteriormente, na subseção 2.4.2.

2.4.1 N-BEATS e a Base MLP

Modelos tradicionais baseados em RNNs (seção 2.1), LSTMs e modelos *Transformers* (seção 2.3) são conhecidos por apresentarem comportamentos de *black boxes*, isto é, a saída e as "decisões"da arquitetura não podiam ser interpretadas de maneira clara e confiável. Dessa forma, a indústria relutava em adotar tais arquiteturas para missões críticas como manutenção preditiva, uma vez que a rede não explicava por que estava prevendo uma falha ou um pico de demanda. Neste contexto, o surgimento do **N-BEATS** alterou o paradigma provando que redes puramente baseadas em MLPs eram incrivelmente rápidas de treinar, escaláveis para milhares de séries e podiam ser construídas para fornecer uma saída transparente que engenheiros e estatísticos pudessem confiar.

Deste modo, a arquitetura do N-BEATS baseia-se em pilhas (*stacks*) de blocos, onde o motor principal de cada bloco ℓ é uma MLP composta por camadas totalmente conectadas com funções de ativação não-lineares. Para uma janela histórica de entrada $y_{t-L:t,\ell}$ no bloco ℓ, o MLP atua como um extrator de características para aprender um vetor de representação oculta h_ℓ:

$$h_\ell = \text{MLP}_\ell(y_{t-L:t,\ell})$$

A segunda parte da arquitetura traz como implementação camadas de projeção lineares para gerar vetores de coeficientes de expansão a partir do vetor h_ℓ. Esses vetores de expansão são divididos em dois ramos, um para o futuro (*forecast*) e outro para o passado (*backcast*). De modo que os coeficientes são dados por:

$$\theta_\ell^f = \text{LINEAR}^f(h_\ell)$$

$$\theta_\ell^b = \text{LINEAR}^b(h_\ell)$$

onde θ_ℓ^f e θ_ℓ^b são os coeficientes de previsão e reconstrução do passado, respectivamente. Após a regressão desses coeficientes, eles são passados por uma função de síntese $g(\cdot)$ para gerar o sinal no domínio do tempo. Para a arquitetura N-BEATS a função de síntese são polinômios ou séries de Fourier para interpretabilidade de tendência e sazonalidade.

A partir disso, o modelo implementa uma técnica de **Empilhamento Duplamente Residual** (*Doubly Residual Stacking*) que permite processar em paralelo tanto a filtragem do passado através do ramo de *backcast*, quanto a agregação do futuro com o ramo de *forecast*. Na etapa de *backcast*, o modelo atua extraindo características da janela de entrada do passado $x_{\ell-1}$ e gerando sua melhor estimativa ou reconstrução dessa mesma

entrada ($\hat{x}_{\ell-1}$). A conexão residual subtrai essa reconstrução do sinal original para servir como entrada para o próximo bloco:

$$x_\ell = x_{\ell-1} - \hat{x}_{\ell-1}$$

Isso permite a captura inicial de características, de forma que os próximos blocos possam extrair padrões residuais mais profundos. Paralelamente, o bloco de *forecast* gera uma predição parcial para o horizonte futuro ($\hat{y}_\ell$), de modo que a conexão no ramo do futuro atua de forma aditiva, sendo a predição final construída pela soma de todas as predições parciais geradas de forma independente por cada um dos blocos:

$$\hat{y} = \sum_\ell \hat{y}_\ell$$

Esse mecanismo permite que cada predição parcial contribua especificamente para o futuro, de modo que a previsão final seja consolidada, agregando todas essas partes de maneira hierárquica. Contudo, essa arquitetura ainda perde eficiência em longos horizontes de previsão, já que o tamanho de sua predição neural é configurada para ser exatamente igual a dimensionalidade do horizonte futuro. Ao tentar prever diretamente todos os pontos futuros passo a passo, o modelo sofre com uma expressividade ilimitada resultando em extrema volatilidade nas previsões e uma explosão desncessária na complexidade e requisitos computacionais. Em vista disso, por meio da interpolação hierárquica e amostragem de sinais de múltiplas taxas o N-HiTS foi capaz de superar esses pontos e estruturar uma nova arquitetura mais robusta e computacionalmente mais barata.

2.4.2 N-HiTS

Para resolver esses impasses relacionados a um longo histórico de predição e as limitações no N-BEATS, a arquitetura do N-HiTS (Figura 10) introduz duas novas inovações dentro de sua estrutura de blocos herdada. A **Amostragem de Múltiplas Taxas** (*Multi-Rate Signal Sampling*), que permite o modelo atuar como um filtro de frequências por meio da aplicação de camadas de *pooling* (redução de dimensionalidade) com diferentes tamanhos de janelas antes da entrada de cada bloco. Adicionalmente, o modelo introduz a **Interpolação Hierárquica** que possibilita o modelo construir a previsão apenas de um conjunto de pontos e, a partir disso, utiliza de uma função matemática de interpolação para preencher as lacunas entre esses pontos para formar a curva final completa, reduzindo drasticamente a quantidade parâmetros de aprendizado necessários para a rede.

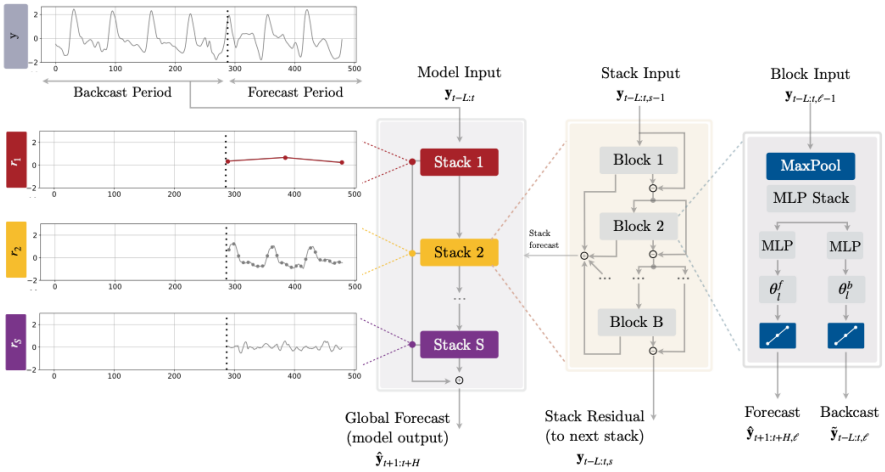


Figura 10 – Arquitetura N-HiTS. Fonte: (CHALLU et al., 2022)

2.4.2.1 Amostragem de Múltiplas Taxas

Ao invés de cada bloco da rede analisar a série temporal em sua resolução original, o N-HiTS introduz uma subamostragem na entrada por meio de uma camada de *MaxPool*, que é responsável por reduzir a dimensionalidade da entrada, preservando apenas as características mais proeminentes do sinal. Logo, para a entrada do bloco ℓ dada por $y_{t-L:t,\ell}$, temos que:

$$y^{(p)}_{t-L:t,\ell} = \text{MaxPool}(y_{t-L:t,\ell}, k_\ell)$$

onde k_ℓ representa o tamanho do *kernel* de agrupamento. Em blocos iniciais, é usado um tamanho de *kernel* grande para cortar severamente as componentes de alta frequência do sinal, obrigando o bloco a especializar-se nas variações de baixa frequência. A sequência agrupada passa, então, por uma MLP com funções de ativação assim como descrito em subseção 2.4.1.

2.4.2.2 Interpolação Hierárquica

Ao contrário da arquitetura N-BEATS que força a MLP a prever diretamente todos os H pontos do horizonte futuro, o N-HiTS usa uma **taxa de expressividade** r_ℓ para limitar a dimensionalidade da predição. Deste modo, o número de coeficientes de previsão emitidos pelo bloco ℓ é $|\theta^f_\ell| = \lceil H/r_\ell \rceil$.

Para recuperar a resolução e prever todos os pontos, os coeficientes são expandidos usando uma função de interpolação temporal contínua g, de modo que a formulação para o *forecast* e *backcast* no bloco ℓ é dada por:

$$\hat{y}_{\tau,\ell} = g(\tau, \theta^f_\ell), \forall \tau \in \{t+1, \ldots, t+H\}$$

$$\tilde{y}_{\tau,\ell} = g(\tau, \theta^b_\ell), \forall \tau \in \{t - L, \ldots, t\}$$

onde $\hat{y}_{\tau,\ell}$ é o valor final previsto interpolado no passo de tempo τ, $\tilde{y}_{\tau,\ell}$ o valor reconstruúido para o passado no passo de tempo τ, θ^b_ℓ e θ^f_ℓ representam os coeficientes de interpolação, H é o horizonte futuro de previsão, L o tamanho da janela de entrada e $g(\cdot)$ é a função de interpolação temporal contínua, que pode ser linear, cúbica, etc.

Embora seja possível usar diferentes equações de interpolação, através de seus resultados o N-HiTS provou que a interpolação linear é altamente eficiente. Ela preenche os espaços contínuos traçando retas entre os coeficientes previstos. Considerando τ como o passo de tempo a ser determinado, t_1 o índice de tempo que ocorre antes ou exatamente em τ, t_2 o índice de tempo do próximo nó de interpolação após t_1 sendo $t_2 = t_1 + 1/r_\ell$, a equação da interpolação pode ser determinada por:

$$g(\tau, \theta) = \theta[t_1] + \frac{\theta[t_2] - \theta[t_1]}{t_2 - t_1}(\tau - t_1)$$

Essa e outras inovações discutidas do modelo são fatores importantes que podem alterar o paradigma da manutenção preditiva industrial. A previsão de horizontes longos é uma tarefa crítica e de alto valor, sendo essencial para o planejamento de intervenções e agendamento de manutenções em equipamentos complexos. Neste contexto, o N-HiTS oferece soluções que podem ser integradas para os principais desafios neste setor como o isolamento do desgaste real contra ruídos operacionais através da sua técnica de Amostragem de Múltiplas Taxas, que permite o modelo isolar e se especializar em analisar o conteúdo de larga escala, capturando a verdadeira tendência de degradação do equipamento ao invés de capturar ruídos indesejáveis.

Além disso, sua arquitetura pode ser uma alternativa no diagnóstico e previsões de vida útil do sistema industrial, uma vez que seu algoritmo de interpolação hierárquica atua como um forte regularizador que garante a geração de curvas de degradação suaves e consistentes. Ademais, a sua natureza intrínseca de dividir o sinal futuro em componentes de hierarquia permite que o modelo seja altamente interpretável, pois cada bloco (*stack*) se especializa em diferentes bandas de frequência do sinal o que permite ao usuário extrair e visualizar a saída isolada de cada *stack* individualmente.

3 Metodologia

Este capítulo tem como objetivo primordial evidenciar as tecnologias usadas durante o desenvolvimento do trabalho e na obtenção dos resultados. Para isso, a pesquisa baseia-se em uma análise quantitativa e experimental, focada na avaliação comparativa de modelos de Aprendizado Profundo em tarefas de manutenção preditiva utilizando dados provenientes de sensores industriais. O fluxo metodológico compreende desde a estruturação do ambiente de desenvolvimento (seção 3.1) e o tratamento de dados provenientes de sensores reais e pré-processamento (seção 3.2 e seção 3.3), até a configuração sistemática dos hiperparâmetros necessários para o treinamento das redes (seção 3.4). O propósito dessa estrutura é permitir um ambiente experimental que permita avaliar se os referidos modelos são capazes de identificar falhas operacionais reais e comportamentos anômalos através da predição de suas séries temporais. As seções posteriores evidenciam em detalhes os procedimentos e tecnologias utilizadas.

3.1 Ambiente e Ferramentas

3.1.1 *Neuralforecast*

O treinamento de modelos de Aprendizado Profundo para predição de séries temporais é um grande desafio, uma vez que é necessário lidar com impasses críticos de engenharia de dados, implementação de arquiteturas complexas dos modelos abordados, entre outros. Neste contexto, a utilização da biblioteca *Neuralforecast* (OLIVARES et al., 2022) oferece um imenso leque de ferramentas capazes de mitigar e tornar o desenvolvimento desses experimentos muito mais prático.

A *Neuralforecast* é uma biblioteca do ecossistema *Nixtlaverse* focada especialmente em modelos de *Deep Learning* para a previsão de séries temporais. Diferente de outras bibliotecas que focam em estatísticas clássicas, a *NeuralForecast* foi desenhada para lidar com um grande volume de dados e padrões complexos. Sua biblioteca contempla os modelos mais recentes da literatura de IA, como o **N-HiTS** (subseção 2.4.2) e **PatchTST** (subseção 2.3.3), permitindo flexibilidade e praticidade no processo de treinamento. Além disso, para resolver problemas de engenharia de dados, como o janelamento temporal aplicado nas séries, toda a engenharia de tensores é implementada internamente permitindo a automação de todo o processo de *dataloading* e criação de janelas.

Portanto, a escolha fundamenta-se na sua arquitetura otimizada para o ecossistema *PyTorch*, permitindo aceleração via GPU e a implementação de modelos de *Deep Learning* focados em séries temporais de última geração. Ademais, a padronização da API permite

comparar, sob as mesmas condições de pré-processamento e *hardware*, múltiplos modelos, o que viabiliza a análise comparativa entre seus resultados, uma importante característica para o objetivo deste trabalho.

3.1.2 *Google Colab*

O ambiente *Google Colaboratory* é um serviço de nuvem, baseado no ecossistema *Jupyter Notebook*, que permite escrever e executar código *Python* diretamente no navegador. Seu ambiente oferece suporte a GPUs e TPUs de alto nível, essenciais para treinar redes neurais e modelos de *Deep Learning*, além de RAM e Disco na nuvem. Além disso, ao contrário de ambientes locais, o *Colab* já gerencia e configura a maioria dos *drivers*, gerenciadores e bibliotecas mais usadas em tarefas de Aprendizado Profundo, como *Pandas* e *Tensorflow*.

À frente disso, a implementação algorítmica deste trabalho foi inteiramente estruturada na plataforma oferecida por esse ambiente. O ambiente de execução utilizado no treinamento e produção dos resultados conta com as seguintes especificações de *hardware*:

Componente	**Especificação**
Processador Gráfico (GPU)	NVIDIA Tesla T4
Memória de Vídeo (VRAM)	15,0 GB GDDR6
Memória RAM do Sistema	12,5 GB
Armazenamento em Disco	112,6 GB
Ambiente de Desenvolvimento	Jupyter Notebook (Cloud-based)
Linguagem de Programação	Python 3.x

Tabela 1 – Especificações do Ambiente de Execução (Google Colab)

Portanto, o ambiente oferecido pelo *Google Colab* foi de extrema importância para a viabilização do treinamento dos modelos. A escolha justifica-se pela necessidade de processamento paralelo via **GPUs**, essenciais para o manejo de séries temporais de alta dimensionalidade provenientes dos sensores industriais. Além disso, a plataforma assegura a reprodutibilidade dos experimentos através de um ambiente virtualizado padronizado.

3.2 Conjunto de Dados

3.2.1 *Dataset*

O *Dataset* utilizado neste presente trabalho foi elaborado pelo autor a partir de dados reais de um sistema de refrigeração industrial armazenados por meio do banco de dados **InfluxDB**, um banco projetado especificamente para dados sequenciais e que faz parte da infraestrutura de coleta dos dados sensoriais. A opção pela elaboração do *dataset*, extraídos a partir da base histórica do *Influx*, justifica-se pela necessidade de validar o

desempenho dos modelos de Inteligência Artificial em cenários que simulam fielmente a volatibilidade do ambiente fabril, evidenciando desafios como ausência de dados, falta de conectividade, limitações dos equipamentos sensoriais, interrupções não planejadas, entre outras.

O conjunto provém de 14 máquinas de refrigeração, especificamente unidades condensadoras, que operam nas câmaras frias de uma empresa do setor alimentício. O conjunto de equipamentos está subdividido em dois grupos no que diz respeito à capacidade de instrumentação e à dimensionalidade das variáveis coletadas.

A primeira categoria engloba 9 máquinas identificadas pelo prefixo "MT". O monitoramento dessas conta com sensores que aferem exclusivamente a corrente elétrica dos três compressores de cada máquina. O segundo grupo é formado por 5 unidades designadas pelo prefixo "B". Estas contam com um sistema de sensoriamento mais amplo, capaz de registrar a corrente elétrica, tensão, frequência e fator de potência.

Além da distribuição estática inicial, a partir do ano de 2026, quatro máquinas do primeiro grupo (MT04, MT05 e MT06) tiverem seus sensores originais substituídos por hardwares da categoria "B", alterando suas respectivas identificações na base de dados para B417, B134 e B419.

3.2.2 Carga Baixa

A ocorrência do fenômeno conhecido como "carga baixa"nas unidades condensadoras é um problema de natureza multifatorial. Suas causas podem variar desde microvazamentos nas tubulações do sistema e baixo nível de fluido refrigerante, até defeitos eletromecânicos nos próprios compressores. Contudo, o escopo deste trabalho não busca realizar o diagnóstico mecânico exato de qual componente falhou. Em vez disso, o objetivo central é investigar a viabilidade de prever e identificar a ocorrência desse estado de carga baixa baseando-se exclusivamente no comportamento dos dados capturados pelos sensores.

Do ponto de vista analítico, o estado de carga baixa apresenta uma assinatura visual relativamente simples de ser identificada. O comportamento anômalo se manifesta quando a série temporal da máquina sofre uma queda sustentada, estabilizando-se em um patamar de medição visivelmente inferior aos níveis de sua operação nominal.

Para estruturar a base de dados e criar a referência necessária para avaliar essa detecção, foi preciso realizar o mapeamento cronológico dessas ocorrências. O total de falhas contabilizadas por máquina, ao longo de seus respectivos períodos de monitoramento, foi obtido a partir da extração dos registros históricos de alarmes. Esses registros foram extraídos diretamente do banco de dados da empresa responsável pelo monitoramento das câmaras frias, servindo como base para validar o comportamento observado nas séries

temporais. As informações correspondentes a cada máquina são observadas nas tabelas Tabela 2 e Tabela 3.

Conjunto de Dados			
Identificação	**Período**	**Total de Registros**	**Registros de Falhas**
MT02	14/11/2024 a 08/10/2025	≈ 30046 pontos	20
MT03	14/09/2024 a 19/02/2026	≈ 47944 pontos	45
MT04	12/09/2024 a 13/12/2025	≈ 41272 pontos	10
MT05	09/09/2024 a 27/05/2025	≈ 23845 pontos	33
MT06	09/09/2024 a 13/12/2025	≈ 43056 pontos	33
MT09	12/09/2024 a 01/04/2026	≈ 53222 pontos	10
MT13	09/09/2024 a 12/12/2025	≈ 40544 pontos	10
MT14	09/07/2025 a 29/03/2026	≈ 25033 pontos	9
MT16	08/07/2025 a 23/09/2025	≈ 6687 pontos	6

Tabela 2 – Especificações Máquinas MT

Conjunto de Dados			
Identificação	**Período**	**Total de Registros**	**Registros de Falhas**
B134	05/06/2025 a 01/04/2026	≈ 24766 pontos	13
B80	25/03/2025 a 23/02/2026	≈ 30778 pontos	25
B81	25/11/2025 a 07/04/2026	≈ 12078 pontos	14
B417	02/04/2026 a 23/04/2026	≈ 1759 pontos	2
B419	03/04/2026 a 22/04/20266	≈ 1597 pontos	1
B420	04/04/2026 a 23/04/2026	≈ 1508 pontos	3
B423	03/04/2026 a 23/04/2026	≈ 1734 pontos	1
B426	03/04/2026 a 23/04/2026	≈ 1763 pontos	1

Tabela 3 – Especificações Máquinas B

3.3 Pré-processamento

A etapa de pré-processamento compreende a transformação dos registros brutos coletados por meio dos sensores industriais em um conjunto de dados apto ao treinamento de modelos de alta performance. Dado que os dados foram coletados em um ambiente industrial real, sujeito a latências de rede e falhas de sensores, aplicou-se um protocolo de limpeza, uniformização de intervalos e normalização. Tais procedimentos asseguram a integridade matemática necessária para o funcionamento das camadas de atenção e blocos residuais dos modelos propostos.

Deste modo, a primeira etapa de pré-processamento foi incubida ao processamento e agregação dos dados sensoriais diretamente na montagem da *query* que filtra os dados históricos do Influx. Estes dispositivos de medição possuem uma taxa de amostragem na faixa de 1 a 5 segundos, o que dificulta o aprendizado de dependências de longo prazo por conta de redundância informacional e ruídos de medição. Portanto, para mitigar esses

entraves, foi utilizado uma agregação temporal para intervalos de 15 minutos através da média aritmética. Este processo de *downsampling* permite que os modelos foquem em tendências macros do sistema de refrigeração, facilitando a convergência durante o treinamento e reduzindo o custo computacional.

A partir disso, com o intuito de garantir a compatibilidade com a biblioteca **NeuralForecast**, o dataset foi submetido a uma reestruturação. Cada série temporal foi devidamente rotulada e indexada temporalmente, respeitando a convenção de nomenclatura de colunas e tipos de dados esperados pelo *framework*. Essa etapa assegurou a integridade do fluxo de dados, desde a alimentação das redes neurais até a geração das predições no horizonte de tempo definido.

Em seguida, com o dataset no formato esperado pelo *framework*, para assegurar integridade cronológica das séries, realizou-se a eliminação de registros duplicados para a combinação entre o identificador do sensor e o respectivo carimbo de tempo. Além disso, visando a continuidade das amostras, as lacunas temporais foram identificadas e preenchidas através da técnica de *Forward Fill*, permitindo que o último estado registrado pelo sensor fosse mantido durante períodos de ausência de dados.

Para divisão de treinamento do modelo, o dataset estruturado foi submetido à divisão em conjuntos de treinamento e teste. Devido à natureza estrutural das séries temporais, para evitar o vazamento de informações (*data leakage*), a divisão não adotou proporções temporais estáticas, mas sim uma separação orientada a eventos. O critério estabelecido foi a utilização do último registro de falha de cada máquina como fronteira de teste. Além disso, para cada 100 passos de treinamento, os modelos foram submetidos a uma etapa de validação. Esta escolha é estratégica para a validação da hipótese deste projeto: testar o modelo em um cenário de falha operacional. Ao isolar um período onde a carga efetivamente cai, torna-se possível provar se a arquitetura de rede neural é capaz de capturar a tendência de queda e antecipar o comportamento crítico, ou se ela simplesmente replica padrões médios de operação normal, falhando na detecção de anomalias.

Por fim, para a normalização das variáveis de entrada definiu-se o uso do ***Robust Scaler***, técnica de redimensionamento dos dados capaz de lidar com a natureza volátil dos dados sensoriais. Esse escalonador assegura que as diferentes grandezas físicas monitoradas sejam transpostas para uma escala comum sem que a precisão decimal seja comprometida por valores extremos.

3.4 Métricas e Configuração de Hiperparâmetros

A eficácia dos modelos de *Deep Learning* observados neste trabalho depende não apenas da arquitetura escolhida, mas do ajuste de seus hiperparâmetros e da seleção de métricas de avaliação. Enquanto a configuração de hiperparâmetros atua como o processo

de calibração que molda o comportamento do modelo durante o treinamento, as métricas de desempenho servem como o referencial quantitativo para mensurar a precisão das previsões e a capacidade de generalização da rede. Neste seção, detalham-se os parâmetros otimizados para os modelos analisados e os resultados métricos utilizados para validar seus comportamentos de predição.

Sobre a parte de hiperparâmetros, dado a quantidade de dados e análises, nem todas as abordagens consideraram todos os valores estipulados de configuração. Ademais, definiu-se como forma fixa o tamanho do lote (*batch size*) em 32 para todos os cenários, visando equilíbrio entre a estabilidade do gradiente e as limitações de memória do ambiente de execução. Como critério de otimização, a função de perda adotada foi o Erro Médio Absoluto (MAE).

3.4.1 Erro Médio Absoluto

O Erro Médio Absoluto (*Mean Absolute Error*) representa a média das magnitudes das diferenças entre os valores previstos pelo modelo e os valores reais observados. Por utilizar o valor absoluto, essa métrica trata erros positivos e negativos com o mesmo peso, fornecendo uma medida linear da precisão do modelo. Sua equação é descrita como:

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

onde n representa o número total de observações, y_i é o valor real e $\hat{y}_i$ é o valor previsto pelo modelo

3.4.2 Erro Quadrado Médio

O Erro Quadrado Médio (*Mean Squared Error*) calcula a média dos quadrados das diferenças entre as previsões e os valores reais. Ao elevar o error ao quadrado, esta métrica penaliza de forma mais severa os desvios de grande magnitude em comparação aos erros menores. Sua equação é descrita por:

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

onde n é o número total de observações, y_i é o valor real e $\hat{y}_i$ o valor previsto. Em alguns casos usou-se o RMSE para visualização, definido como a raíz quadrada do MSE.

3.4.3 LSTM e NHITS

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de aprendizado.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m \in [1, 7]$
Tamanho do lote.	{32}
Função de ativação.	ReLU
Camadas do *encoder*.	$E \in \{2\}$
Tamanho da camada oculta do *encoder*.	$E_s \in \{64, 128\}$
Tamanho da camada oculta do *decoder*.	$D_s \in \{64, 128\}$
Camadas do *decoder*.	$D \in \{2\}$

Tabela 4 – Configuração de Hiperparâmetros da LSTM

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de aprendizado inicial.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m \in [1, 5]$
Tamanho do Lote.	{32}
Função de Ativação.	ReLU
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1], [4, 4, 1], [24, 4, 1]\}$
Número de *Stacks*.	$S \in \{3\}$
Blocos por *stack*.	$B \in \{1\}$
Camadas MLPs.	{3}
Tamanho Oculto dos Coeficientes.	$N_h \in \{512\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[2, 2, 1], [24, 12, 1]$ $[8, 4, 1], [4, 4, 1]\}$
Estratégia de interpolação	$g(\tau, \theta) \in \{\text{Linear}\}$

Tabela 5 – Hiperparâmetros do NHITS

3.4.4 DLinear e NLinear

Para os modelos NLinear e DLinear que não exigem hiperparâmetros arquiteturais complexos o ajuste concentrou-se nos parâmetros treinamento, como taxa de aprendizado e tamanho da janela de entrada (*input size multiplier*).

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de aprendizado.	{1e-4, 1e-3}
Passos de treinamento.	{5000, 8000}
Input size multiplier ($L = m * H$).	$m \in [1, 7]$
Tamanho do Lote.	32
Escalonamento.	{identity, robust}
Janela de Média Móvel (DLinear)	{13, 25, 33}

Tabela 6 – Configuração de Hiperparâmetros do NLinear e DLinear

3.4.5 PatchTST e Autoformer

Para os modelos PatchTST e Autoformer, a configuração de hiperparâmetros atinge um nível elevado devido à natureza modular e profunda de suas arquiteturas. Por se tratarem de modelos de alta complexidade, a calibração dos parâmetros requer um esforço adicional de tempo e de recursos computacionais. O treinamento impõe uma demanda computacional significativamente superior, tornando o processo de otimização mais desafiador com recursos limitados do *Google Colab*. A necessidade de processar grandes volumes de tensores para os mecanismos de atenção e armazenamento de gradientes levam ao esgotamento da memória RAM da GPU disponível. Consequentemente, a escolha dessas configurações evidencia que a busca pelo desempenho máximo é atrelada à disponibilidade de *hardware*.

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de Aprendizado.	{1e-4, 1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m \in [1, 2]$
Number of Multi-Head's Attention.	{8, 16}
Camadas de *encoder*	3
Unidades de *Embeddings* e *encoders*	128
Unidades de Camadas Lineares	256
Tamanho do *patch*	{8, 16}
Stride do *patch*	{4, 8}
Escalonamento.	identity, robust

Tabela 7 – PatchTST: Configuração de Hiperparâmetros

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de Aprendizado.	{1e-4, 1e-3}
Passos de treinamento.	{1000, 5000}
Input size multiplier ($L = m * H$).	$m = 1$
Number of Multi-Head's Attention.	{4, 8}
Unidades de *Embeddings* e *encoders*	{64, 128}
Probsparse Attention Factor	3
Camadas de *encoder*	2
Camadas de *decoder*	1
Tamanho do filtro de média móvel	25
Scaler Type.	{identity, robust}

Tabela 8 – Autoformer: Configuração de Hiperparâmetros

3.4.6 Otimização de Hiperparâmetros

A otimização de hiperparâmetros foi conduzida através do algoritmos de busca **Hyperopt** (BERGSTRA; YAMINS; COX, 2012), integrado ao *backend* **Ray Tune** (LIAW et al., 2018). O Hyperopt é uma biblioteca voltada para a otimização de espaços de busca, utilizando algoritmos como o *Tree of Parzen Estimators* (TPE) para encontrar configurações globais de forma mais eficiente que uma busca aleatória simples. Para viabilizar esse processo, utilizou-se o Ray como motor de execução paralela, uma estrutura de computação distribuída que permite gerenciar múltiplas tentativas de treinamento de forma escalonada.

Este processo de otimização automatizada foi aplicado exclusivamente ao modelo NHITS. Tal decisão fundamentou-se no comportamento computacionalmente eficiente dessa arquitetura, permitindo múltiplas iterações de busca dentro das restrições impostas pelo ambiente de execução do projeto. Além disso, o NHITS tem demonstrado um desempenho superior e consistente em trabalhos recentes de previsão de séries temporais, justificando o investimento na sua calibração exaustiva. Para operacionalizar essa tarefa a biblioteca NeuralForecast disponibiliza versões "Auto"dos modelos. Essas implementações são projetadas nativamente para integrar a lógica de busca do Hyperopt e do Ray, automizando a seleção da melhor configuração e garantindo que o modelo final entregue o menor erro possível para os horizontes de previsão estabelecidos. À frente disso, a configuração utilizada pode ser observada na Tabela 9.

HIPERPARÂMETRO	**VALORES CONSIDERADOS**
Taxa de aprendizado.	{1e-3}
Passos de treinamento.	$\{1000, 2500, 5000\}$
Escalonamento	robust
Random seed for initialization.	DiscreteRange(1, 10)
Input size multiplier ($L = m * H$).	$m \in \{5, 7, 2\}$
Tamanho do lote.	{256}
Função de Ativação.	ReLU
Decaimento da Taxa de Aprendizado (3 *times*).	0.5
Tamanho do kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 2], [4, 4, 4], [8, 8, 8], [8, 4, 1], [16, 8, 1]\}$
Número de *Stacks*.	$S \in \{3\}$
Blocos por *stack*.	$B \in \{1\}$
Camadas MLPs.	{2}
Tamanho dos Coeficientes Ocultos.	$N_h \in \{512\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[168, 24, 1], [24, 12, 1] [180, 60, 1], [40, 20, 1], [64, 8, 1], [4, 4, 1], [8, 4, 1]\}$
Estratégia de interpolação.	$g(\tau, \theta) \in \{\text{Linear}\}$

Tabela 9 – AutoNHITS: Otimização de Hiperparâmetros.

4 Resultados e Discussões

Este capítulo compreende os resultados obtidos na aplicação dos modelos para a predição de séries temporais em sensores industriais. A análise foca na capacidade das arquiteturas em antecipar o fenômeno de carga baixa, estado crítico que compromete a integridade de produtos termossensíveis e indica falhas no maquinário. Para tanto, a seção 4.1 detalha o arranjo experimental e métricas de avaliação; a seção 4.2 discute a modelagem global; e a seção 4.3 analisa a modelagem local.

4.1 Visão Geral dos Experimentos

A condução dos experimentos preditivos foi estruturada em duas abordagens metodológicas distintas, com o intuito de comparar a capacidade de generalização frente ao grau de especialização dos algoritmos. A primeira estratégia consistiu na criação de modelos globais agrupados por categoria, resultando em um modelo treinado exclusivamente com o conjunto de máquinas da série "MT"e outro dedicado apenas aos equipamentos da série "B". Em contrapartida, a segunda estratégia adotou uma modelagem estritamente local, na qual foi instanciado e treinado um modelo especialista e individualizado para a máquina "B134", avaliando o desempenho da rede ao aprender apenas com o histórico isolado do próprio equipamento.

Para viabilizar a primeira abordagem, explorou-se a capacidade da biblioteca de lidar com múltiplas séries temporais simultaneamente através da técnica de **Modelagem Global** (*Global Forecasting*) e pela diferenciação das séries através de seus identificadores. Ao processar o histórico de vários equipamentos de mesma natureza estrutural em um único treinamento, a arquitetura da rede neural passa a utilizar uma matriz de pesos compartilhados. Em termos práticos, o modelo não se limita a decorar a dinâmica isolada de uma máquina, mas simula a extração de padrões globais de conhecimento interno, permitindo que a assinatura de uma carga baixa aprendida em uma máquina seja refletida no auxílio de predição em outra máquina de mesmo tipo.

Já na segunda abordagem, foi adotado um modelo especialista focado apenas na máquina "B134". A decisão justificou-se pela disponibilidade de um histórico mais extenso e pela presença de um sensor capaz de registrar variáveis adicionais como fator de potência, tensão e frequência, além da corrente elétrica. Além disso, ao utilizar Variáveis Exógenas nos modelos LSTM e NHITS, um volume maior de correlações pode contribuir para resultados mais expressivos. Neste contexto, o intuito dessa modelagem foi testar se o enriquecimento de contexto se traduz em um ganho direto na capacidade de aprendizado e modelagem da rede.

Nesse contexto, o critério de sucesso para os modelos, indo de encontro com a literatura recente, fundamenta-se nos valores obtidos nas métricas de Erro Absoluto Médio (MAE) e Erro Quadrático Médio (MSE), medidas que avaliam de maneira quantitativa como a série predita se aproxima da série real. Entretanto, a análise não se limita apenas à redução de métricas estatísticas globais, mas baseia-se de maneira adicional na eficácia em antecipar a queda na carga elétrica. Ao predizer com precisão que a máquina sofrerá essa redução antes que ela ocorra, o modelo demonstra ter "aprendido"os padrões que precedem a instabilidade. Essa capacidade de detecção precoce é o que valida a aplicação como uma ferramenta de manutenção preditiva, cumprindo o objetivo de transformar dados sensorias brutos em informação.

4.2 Modelagem Global

Para a etapa de modelagem global, foram selecionadas e avaliadas cinco arquiteturas preditivas com suporte a séries temporais multivariadas: LSTM, NHITS, NLinear, DLinear e PatchTST. O escopo metodológico inicial previa também a avaliação do modelo Autoformer. Porém, a sua execução foi inviabilizada devido a restrições de memória e processamento do ambiente de execução do *Google Colab*. A exclusão desse algoritmo evidencia o elevado custo computacional e a complexidade de implementação exigidos por certas arquiteturas baseadas em mecanismos de atenção mais densos, justificando a delimitação do experimento aos cinco modelos citados, os quais demonstraram uma relação viável entre capacidade preditiva e eficiência de treinamento.

4.2.1 Máquinas Tipo MT

Nesse conjunto de máquinas, onde os sensores só são capazes de medir a corrente elétrica dos seus três compressores, o desafio central para as arquiteturas de predição foi verificar se o aprendizado conjunto e o compartilhamento de pesos entre as dez séries temporais são suficientes para compensar a ausência de grandezas complementares.

Embora o modelo LSTM utilizado na análise global tenha conseguido aprender corretamente a sazonalidade das séries temporais, seu desempenho foi comprometido por um notável *overfitting* na maioria dos casos de teste, o que resultou na incapacidade de predizer com precisão os períodos de carga baixa das máquinas MT. Essa limitação preditiva ficou evidente tanto em mudanças sutis no regime operacional quanto em quedas de carga mais expressivas (Figura 11), situações com as quais a rede não foi capaz de lidar adequadamente.

Adicionalmente, as tentativas de ajuste variando a janela de *lookback* para $7H$ e $3H$ não produziram melhorias substanciais e, em algumas séries, chegaram a prejudicar a performance do modelo quando comparadas aos resultados obtidos com a janela de $5H$.

Ao aumentar camadas internas de 64 para 128 e adicionar 0.2 de *Encoder Dropout*, o modelo ainda sim não obteve nenhum ganho significativo.

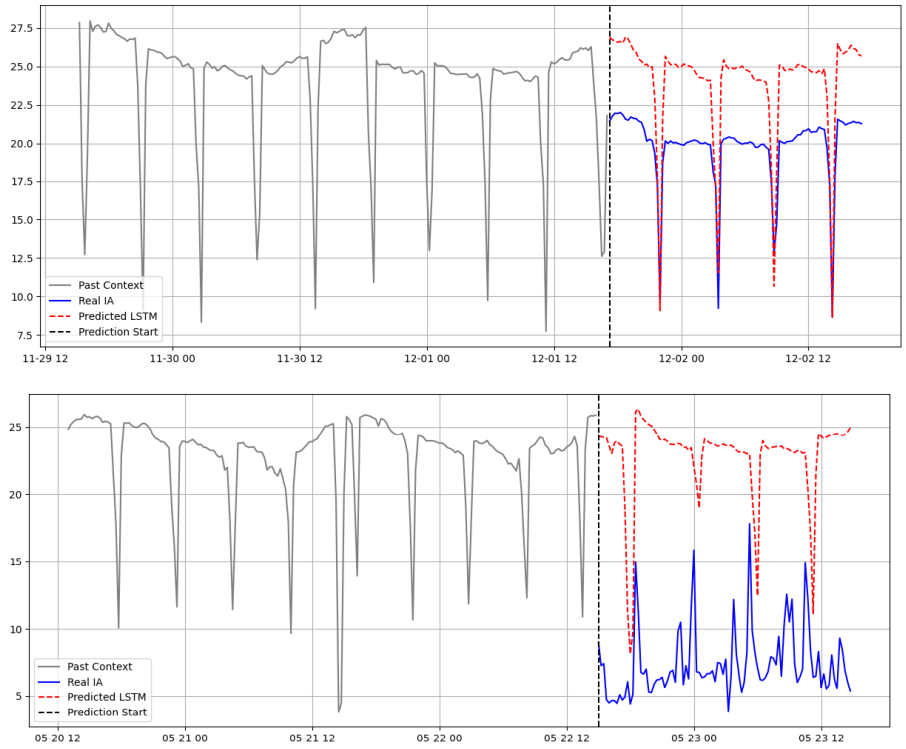


Figura 11 – LSTM: *Overfitting* nas predições. Acima: MT06. Abaixo: MT05.

Seguindo a mesma tendência de instabilidade da LSTM, as arquiteturas NLinear e DLinear também sofreram com *overfitting* na maioria dos casos de teste. A fim de mitigar os casos de sobreajuste desses modelos foram empregadas variações em seus parâmetros mais significativos como a janela de *lookback* para o NLinear e a janela de decomposição do DLinear. Porém, a otimização não produziu nenhum resultado significativo.

A arquitetura PatchTST revelou resultados expressivos, demonstrando capacidade de prever as quedas de carga operacional em determinadas máquinas. Todavia, notou-se que o modelo ainda é suscetível ao *overfitting* em alguns equipamentos, manifestando dificuldade de generalização, sobretudo diante de mudanças bruscas. Seu melhor desempenho ocorreu durante variações mais sutis (Figura 12) dentro da faixa de carga nominal, capturando nuances que podem servir como uma informação valiosa em um contexto de predição contínua. Apesar desse potencial promissor, a complexidade e a robustez inerentes à arquitetura do PatchTST resultam em um custo computacional considerável. Logo, a exploração aprofundada de técnicas de otimização para este modelo não pôde ser viabilizada com o ambiente utilizado neste trabalho.

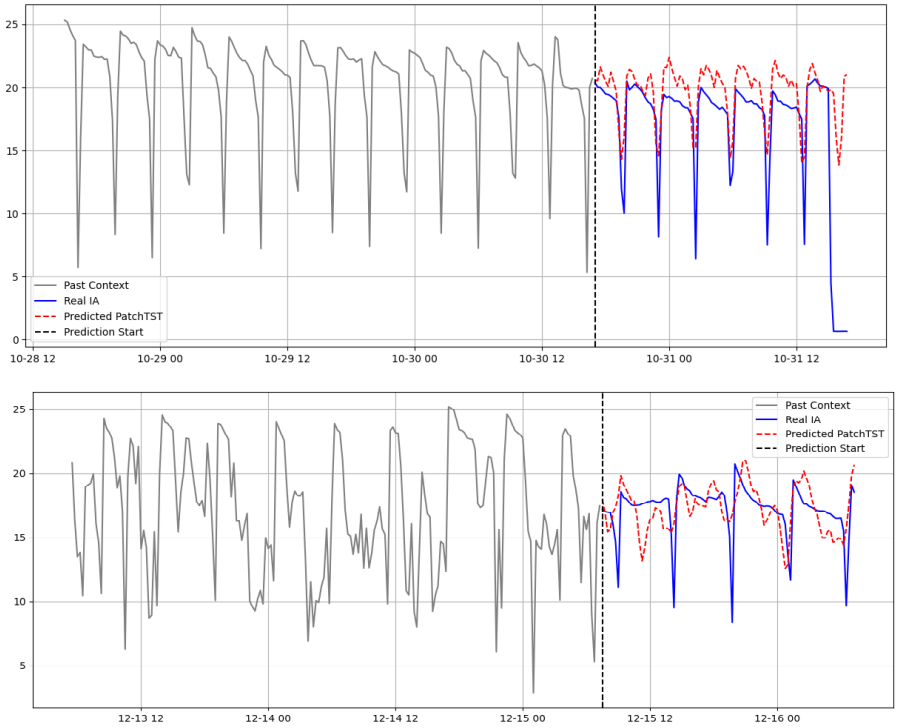


Figura 12 – Predição com PatchTST. Acima: MT04; Abaixo: MT03.

Por fim, assim como os resultados observados pela arquitetura PatchTST, o N-HITS foi capaz de produzir alguns resultados positivos, mas em outros casos sofreu com o mesmo problema de *overfitting* enfrentado pelos demais modelos. Em consonância com o PatchTST, seus resultados mais expressivos ocorreram em predições de quedas mais sutis na carga nominal e falhando em predizer mudanças mais bruscas na faixa nominal.

Na tentativa de mitigar essas limitações, procedeu-se com uma etapa de otimização de hiperparâmetros. Contudo, mesmo após a variação sistemática de configurações cruciais para a arquitetura, como o Número de Coeficientes em cada *Stack* e o Tamanho do *Kernel* de *Pooling*, o N-HITS não apresentou nenhuma mudança significativa em seu desempenho. O ajuste contínuo desses parâmetros não se mostrou suficiente para reverter a tendência de *overfitting* ou aprimorar a resposta do modelo diante das mudanças bruscas de regime.

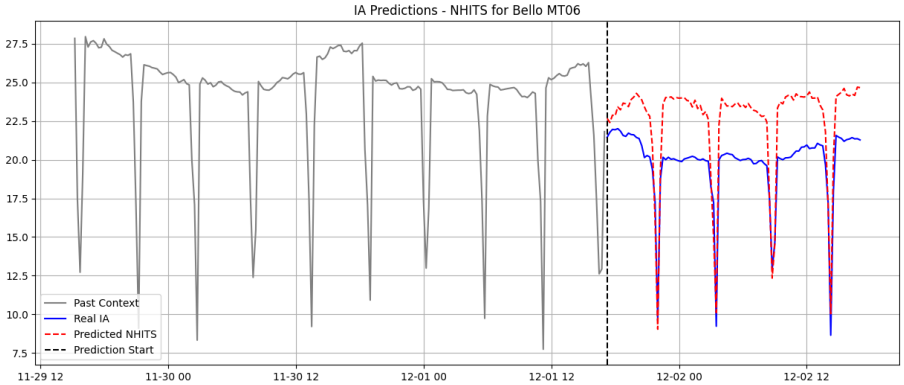


Figura 13 – NHITS: Predição MT06 com resultado positivo.

4.2.2 Máquinas Tipo B

As máquinas do tipo "B"são equipadas com sensores modernos que monitoram, além da corrente elétrica, o fator de potência, a tensão e a frequência do equipamento. Para a modelagem, as variáveis selecionadas foram os fatores de potência de cada fase, o fator de potência total e as correntes das outras duas fases. Essa seleção foi feita porque essas métricas apresentaram a maior correlação estatística com a variável que se deseja predizer, ou seja, a corrente da fase A.

Em algumas análises com os modelos LSTM, NLinear e DLinear, optou-se por utilizar um horizonte de previsão de 48 passos. Essa configuração foi necessária porque certos sensores no conjunto de dados, como o B419 e o B417, possuem um volume histórico reduzido, o que acaba restringindo o tamanho da janela de *lookback*. Ao estabelecer esse horizonte específico de 48 passos, foi possível contornar essa limitação e viabilizar o aumento da janela de *lookback*, otimizando o aproveitamento dos dados disponíveis.

Em relação ao modelo LSTM, observou-se um desempenho ligeiramente superior ao obtido nas máquinas do tipo "MT". Embora o modelo ainda apresente alguns casos de *overfitting*, duas predições se destacaram positivamente: no sensor B81 (arquitetura base, H=48 e L=5H), o modelo conseguiu prever uma queda na faixa nominal, mesmo que a curva não tenha coincidido perfeitamente com a série real (Figura 15); e no sensor B134 (arquitetura base, H=96 e L=3H), a predição indicou corretamente uma tendência de queda, apesar de os dados de teste exibirem uma queda e forma mais abrupta.(Figura 14). Além dessas, destaca-se um comportamento, evidenciado na Figura 16, onde o compartilhamento de pesos com os testes de mudanças abruptas afetaram uma predição de queda sutil no B417.

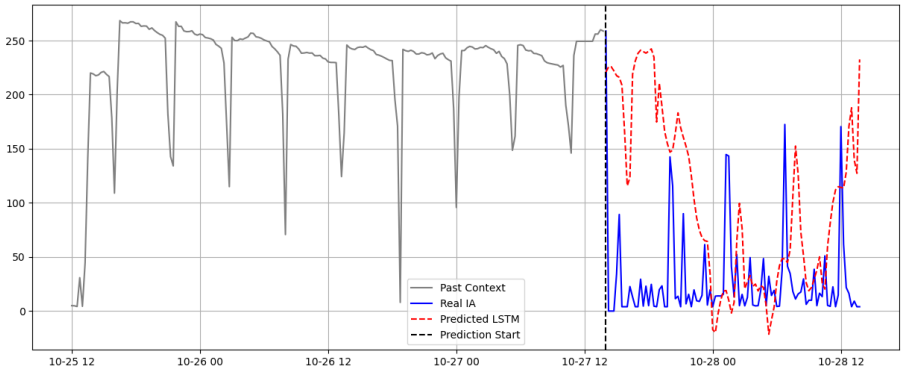


Figura 14 – LSTM: Predição B134

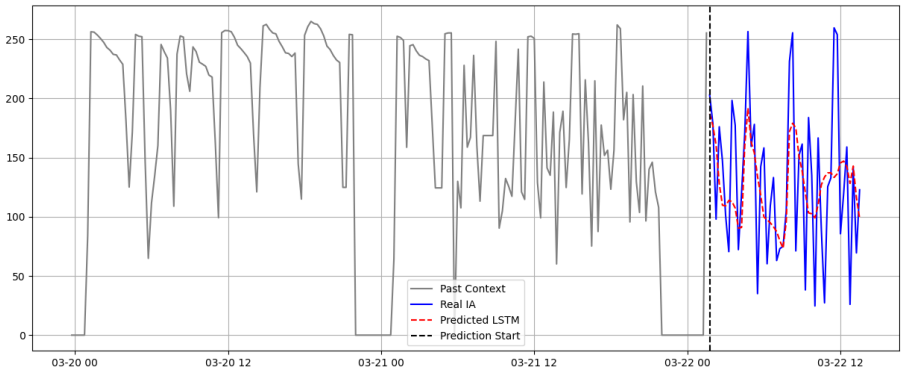


Figura 15 – LSTM: Predição B81

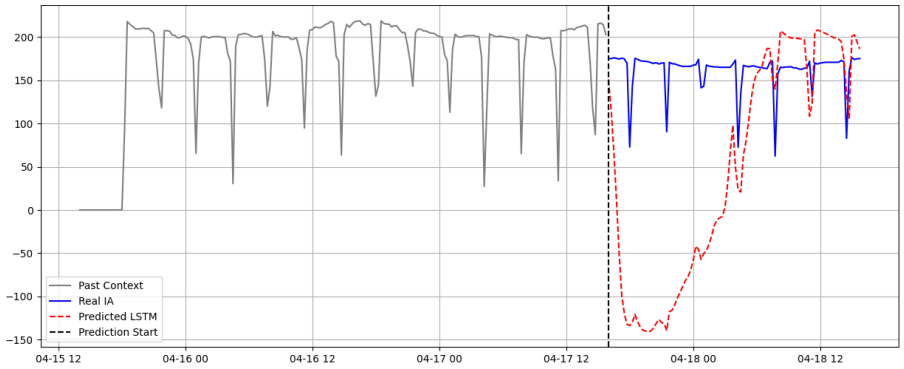


Figura 16 – LSTM: Predição B417

Os modelos NLinear e DLinear não apresentaram resultados expressivos, visto que a maioria dos testes sofreu com sobreajuste. O grande destaque, no entanto, foi a predição para o sensor B80, em que ambos os modelos conseguiram se aproximar significativamente do comportamento real da série. Independentemente da janela de *lookback* utilizada no NLinear ou do tamanho da janela de decomposição no DLinear, o Erro Médio Absoluto (MAE) na predição do B80 manteve-se estável, não apresentando nenhuma diferença significativa.

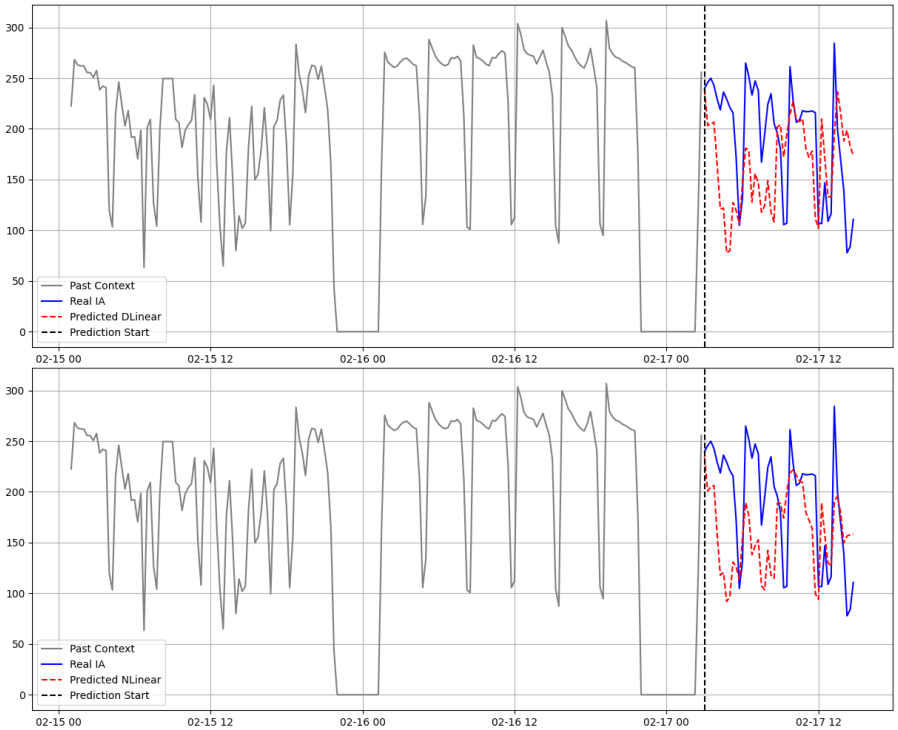


Figura 17 – DLinear e NLinear: Predição B80

Por fim, em todas as análises, o modelo PatchTST sofreu com um severo sobreajuste, produzindo apenas predições inconsistentes e nenhum resultado significativo. Em contrapartida, o modelo NHITS apresentou um desempenho superior ao observado nas máquinas "MT"ao prever mudanças mais sutis na faixa nominal dos sensores B417 e B419 (Figura 18). Contudo, o NHITS também sofreu com sobreajuste nos demais sensores e não demonstrou nenhuma variação significativa em seu comportamento, independentemente da otimização de seus principais hiperparâmetros.

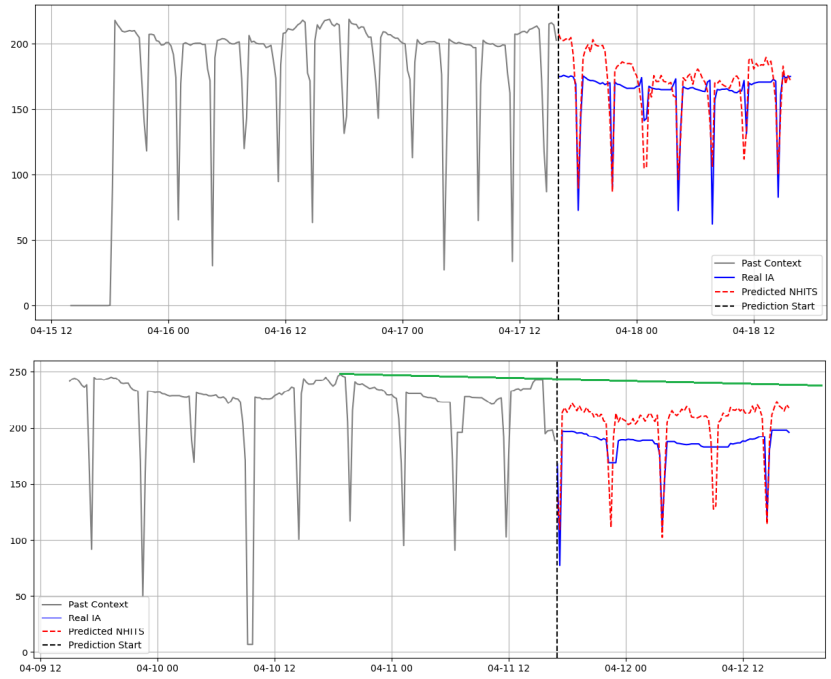


Figura 18 – Acima: Predição NHITS para B417; Abaixo: Predição NHITS para B419

Os resultados observados nas arquiteturas NHITS e, principalmente, na LSTM evidenciam um aspecto importante: embora os sensores do tipo "B"possuam um histórico temporal e de falhas mais reduzido, a utilização de variáveis exógenas no treinamento foi capaz de elevar, embora mínimo, o desempenho das predições em comparação aos sensores que medem apenas a corrente elétrica. Em contrapartida, os modelos NLinear e DLinear, apesar de bem conceituados na literatura para previsão de séries temporais, não apresentaram resultados significativos neste escopo de manutenção preditiva. Mesmo possuindo arquiteturas computacionalmente mais "baratas", ambos sofreram intensamente com o sobreajuste das séries.

Quanto ao PatchTST, o modelo obteve alguns resultados positivos nas máquinas tipo "MT", mas sobreajustou completamente nas máquinas tipo "B". Esse comportamento indica uma alta sensibilidade do modelo ao volume de dados históricos e à quantidade de falhas registradas. Além disso, por exigir um custo computacional maior para o treinamento, a otimização de seus parâmetros não foi tão aprofundada quanto a dos demais modelos, o que deixa uma margem para explorações futuras.

Por fim, o fato de os resultados não apresentarem mudanças significativas, mesmo após diversas tentativas de otimização nos modelos NLinear, DLinear, LSTM e NHITS, sugere que o gargalo do estudo está fortemente atrelado ao próprio conjunto de dados. Isso aponta para possíveis limitações na etapa de *feature engineering* ou para um baixo fator preditivo intrínseco aos dados atualmente disponíveis.

4.3 Modelagem Local

Para a etapa de Modelagem Local, o equipamento escolhido foi o sensor **B134**, cuja seleção se baseou em três disponibilidades valiosas: o seu amplo volume histórico, a presença de registros de falhas e a capacidade de correlacionar outras variáveis por se tratar de um dispositivo mais moderno. Como complemento à análise, as medidas de fator de potência e das demais correntes (fases B e C) foram selecionadas por serem as mais correlatas com a variável alvo do estudo, a corrente elétrica na fase A, assim como mostra a (Figura 19). É importante destacar, no entanto, que essas variáveis adicionais foram incorporadas exclusivamente aos modelos NHITS e LSTM, uma vez que a biblioteca utilizada oferece suporte nativo multivariado para essas arquiteturas, propriedade que se encontra ausente nos demais modelos avaliados.

A adoção da modelagem local é fundamental porque, na prática industrial, equipamentos fisicamente idênticos não operam de forma igual. Mesmo se tratando de unidades condensadoras de um mesmo modelo, fatores como ambiente térmico, variações de carga e histórico de desgaste fazem com que a assinatura elétrica de cada máquina divirja significativamente. Ao focar exclusivamente no ativo específico, essa abordagem cirúrgica captura essas particularidades operacionais únicas, evitando as generalizações imprecisas de modelos globais e garantindo um monitoramento de alta fidelidade para melhores diagnósticos.

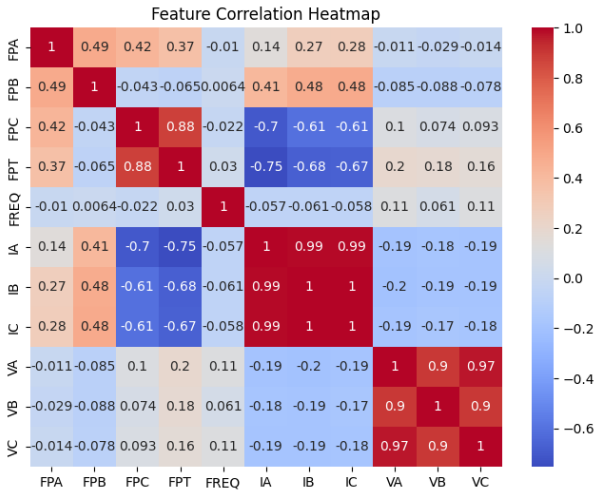


Figura 19 – Correlações entre Variáveis.

Frente à isso, os resultados de predição com os modelos em quatro horizontes diferentes ($H = \{48, 96, 192, 336\}$, equivalentes a 12h, 24h, 48h, 84h) são encontrados na Tabela 10. Cabe destacar que, para os horizontes mais extensos (192 e 336 passos), a utilização do modelo Autoformer não se mostrou viável em virtude do elevado custo computacional inerente à sua arquitetura. Por fim, as configurações dos hiperparâmetros de cada modelo, correspondentes às métricas expostas na tabela para os respectivos

horizontes, encontram-se detalhadas no Apêndice A.

		N-HiTS		LSTM		NLinear		DLinear		PatchTST		AutoNHITS		Autoformer	
	H.	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
B134	48	19.648	17.605	20.111	18.049	17.727	15.771	17.297	**15.416**	19.891	17.706	19.607	17.693	**16.813**	15.427
	96	7.290	**4.201**	6.892	5.486	16.787	15.487	17.339	16.109	15.484	13.906	**6.884**	4.410	22.488	21.324
	192	11.688	8.541	**9.940**	**7.014**	13.824	10.551	13.765	10.335	12.332	8.793	13.050	9.739	-	-
	336	**9.677**	**7.487**	15.132	11.597	12.275	10.656	11.513	9.487	11.134	7.904	-	-	-	-

Tabela 10 – Modelagem Local: Tabela Comparativa de Resultados

Para início das discussões, é crucial destacar o comportamento observado no horizonte de previsão de 48 passos ($H = 48$). Nesta janela, todos os modelos avaliados sofreram com *overfitting*, registrando resultados inconsistentes quando comparados aos desempenhos nos demais horizontes. Embora o modelo Autoformer tenha se destacado positivamente em relação aos outros nessa configuração específica, ele ainda assim não conseguiu predizer, nem mesmo de maneira aproximada, a queda abrupta da série temporal (Figura 20). Esse padrão de comportamento evidencia de forma clara que a extensão do horizonte de previsão exerce uma influência significativa e direta na performance e na capacidade de generalização das arquiteturas avaliadas.

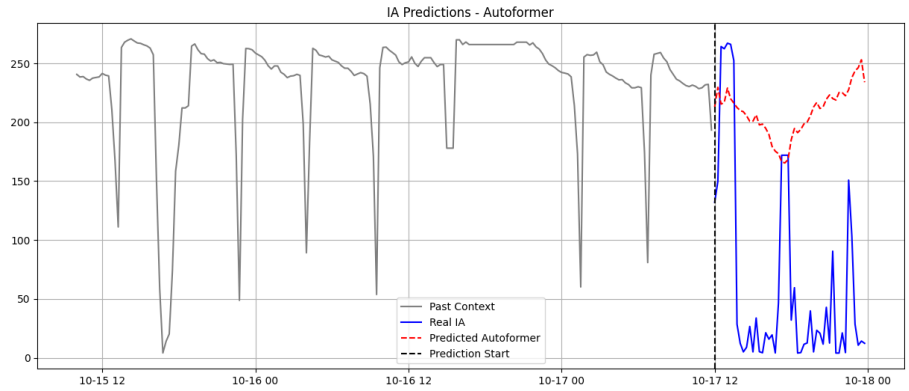


Figura 20 – Autoformer: Predição B134 com H=48.

Em contraste direto com o cenário anterior, o horizonte de predição de 96 passos se sobressaiu como o que obteve os melhores resultados gerais, apresentando a maior aproximação com a série temporal real. Além disso, é válido ressaltar o comportamento dos modelos **NLinear** e **DLinear**: embora não tenham alcançado uma sobreposição exata com os valores reais, foi possível constatar que essas arquiteturas conseguiram capturar e predizer a tendência de uma leve queda na série (Figura 21), demonstrando sensibilidade à dinâmica do sistema.

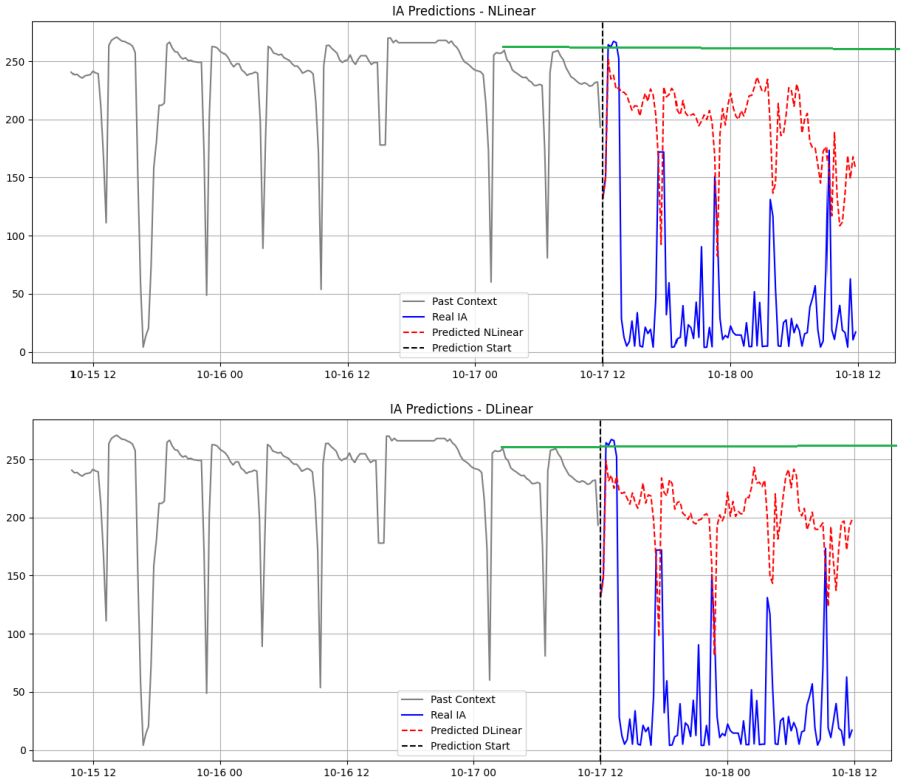


Figura 21 – NLinear e DLinear: Predição para B134 com H=96.

Consolidando o marco principal das análises, o modelo **NHITS** obteve o melhor desempenho preditivo de todo o trabalho para este horizonte de 96 passos. Cabe destacar que a acurácia desta arquitetura foi significativamente incrementada pela otimização do hiperparâmetro de número de coeficientes das *stacks* $[r_1^{-1}, r_2^{-1}, r_3^{-1}]$. A configuração ideal encontrada foi $[4, 4, 1]$, superando as alternativas de $[2, 2, 1]$ e $[24, 12, 1]$. Esse comportamento estrutural sugere que a série contém ruídos que demandam filtragem (o que justifica a rejeição do valor 2), enquanto a sua verdadeira "essência"preditiva reside em uma escala de tempo relativamente curta (explicando a ineficácia dos coeficientes mais longos e complexos, como 24).

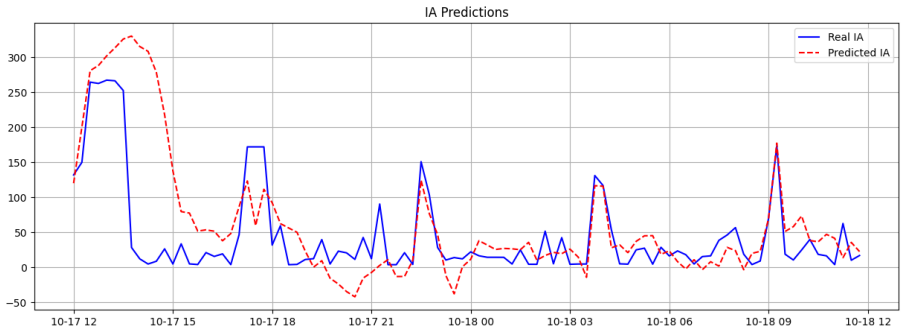


Figura 22 – NHITS: Predição B134 com H=96 (Zoom).

Outro achado determinante foi a comprovação de que as variáveis exógenas são extremamente importantes para a performance das predições. Os modelos LSTM, NHITS

e AutoNHITS destacaram-se significativamente frente aos univariados, validando a premissa de que a correlação com as demais correntes e o fator de potência enriquece a modelagem. No caso específico da rede **LSTM**, notou-se que o seu comportamento foi adicionalmente altamente sensível e influenciado pelo tamanho da janela de histórico (Figura 23), atingindo seu melhor desempenho com $L = 5 \times H$.

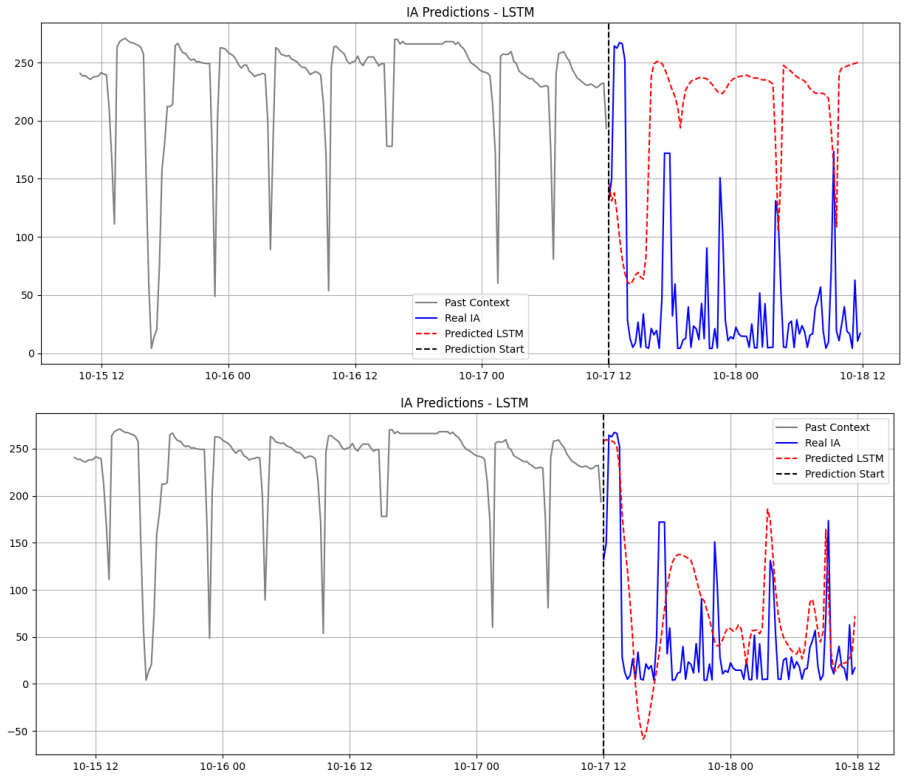


Figura 23 – Acima: Predição com LSTM com L=7H; Abaixo: Predição com LSTM com L=5H.

Por fim, a avaliação do modelo **PatchTST** trouxe insights complementares sobre a natureza dos dados. Observou-se que a arquitetura obteve um ganho de desempenho ao ter seus parâmetros simplificados: reduzindo o número de cabeças de atenção (*heads*) de 16 para 8, o tamanho do *patch* de 16 para 8 e o passo (*stride*) de 8 para 4. Esse resultado indica claramente que os dados possuem uma dinâmica marcante de curto prazo e que o emprego de estruturas arquiteturais excessivamente complexas não compactua com a complexidade intrínseca do problema estudado.

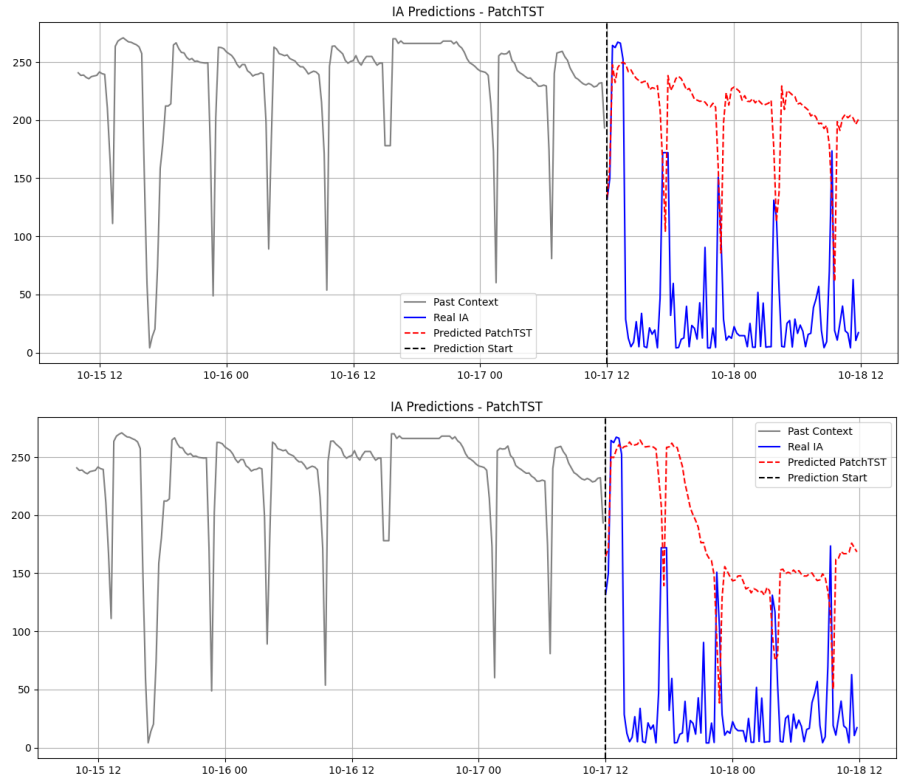


Figura 24 – Comparação entre arquiteturas; Acima: PatchTST mais robusto; Abaixo: PatchTST com arquitetura simplificada.

Para $H = 192$, o modelo LSTM obteve o maior destaque. A rede conseguiu antecipar o comportamento geral da série, prevendo a queda e a subsequente retomada de carga. Contudo, embora essa dinâmica macro tenha sido capturada mesmo sem coincidir perfeitamente com a série real, fica claro visualmente que a projeção sofreu uma considerável perda de identidade em relação à curva característica original.

Ainda neste mesmo horizonte, é essencial evidenciar o desempenho do modelo PatchTST. Mesmo não atingindo os melhores valores nas métricas quantitativas, essa arquitetura conseguiu prever o momento exato da queda e do retorno à faixa nominal, com a vantagem notável de preservar com fidelidade a curva característica da série temporal.

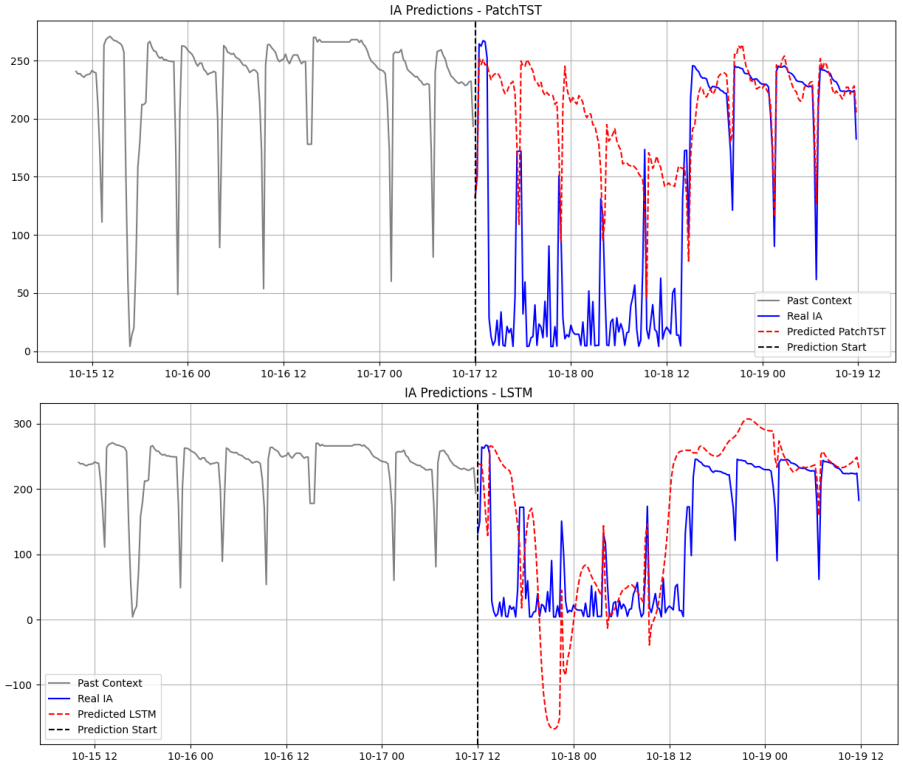


Figura 25 – Acima: Predição PatchTST para H=192; Abaixo: Predição LSTM para H=192

Vale pontuar que, na predição em que alcançaram seu melhor valor de MAE, as arquiteturas lineares sofreram com sobreajuste na tentativa de prever a falha. Em contraste, em outras configurações, embora os modelos tenham conseguido indicar a ocorrência da queda, eles apresentaram uma nítida defasagem temporal em relação à série real, como indicado na Figura 26. Esse desvio é justificado por limitações mecânicas dos algoritmos. O modelo DLinear introduz um atraso inerente ao aplicar filtros de médias móveis para extrair tendências, enquanto o NLinear sofre de inércia preditiva ao ancorar toda a sua projeção no último valor alto conhecido da janela de contexto.

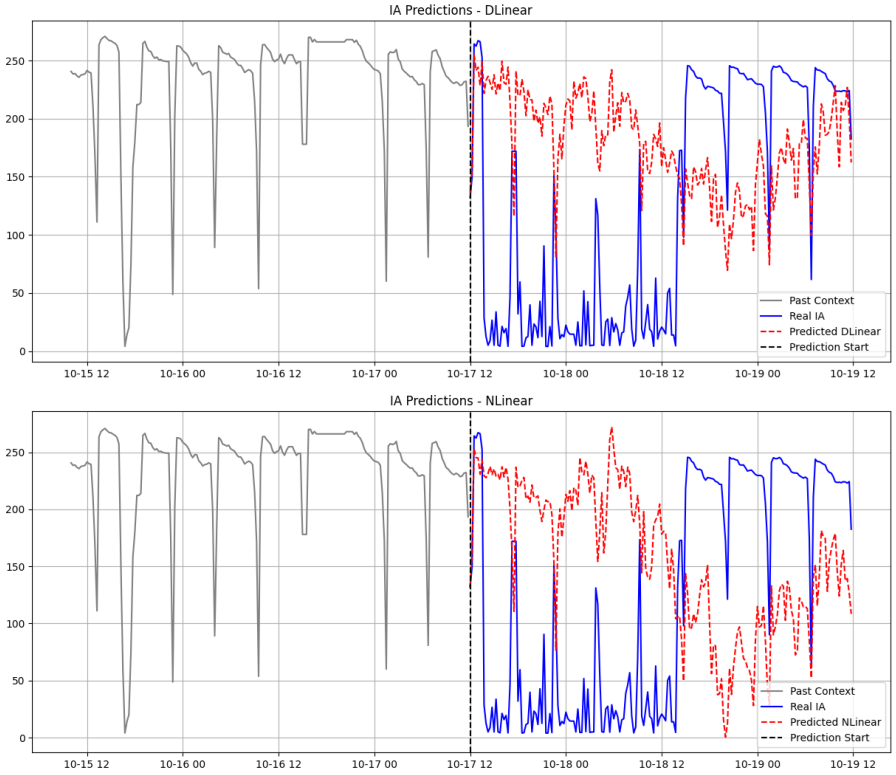


Figura 26 – Defasagem nas predições de modelos lineares

Por fim, o horizonte de 336 passos impõe uma complexidade a mais nas predições, cenário no qual os modelos NHITS e PatchTST se destacam por conseguirem predizer a queda e a volta para a faixa nominal (Figura 28), mesmo que suas curvas não sejam perfeitamente coincidentes com a série real. Em contrapartida, os modelos lineares mantiveram o mesmo comportamento já visto para $H = 192$, apresentando defasagens na predição das quedas na carga. Além disso, a rede LSTM gerou resultados completamente anômalos (Figura 27), indicando uma forte sensibilidade a horizontes preditivos maiores e evidenciando um possível problema de desvanecimento do gradiente.

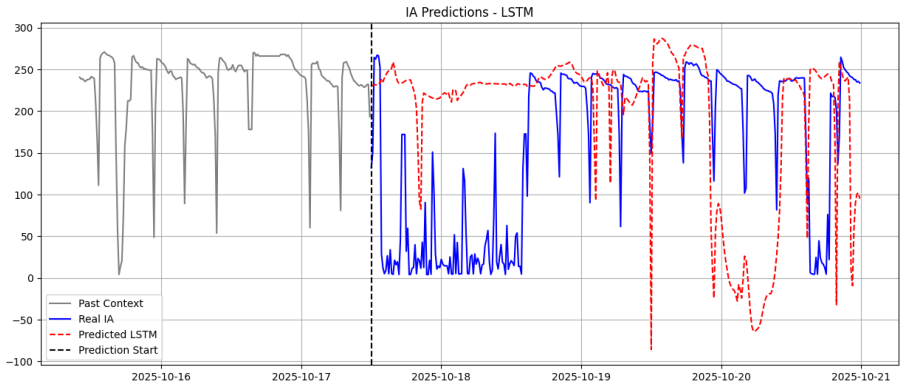


Figura 27 – LSTM: Predição anômala.

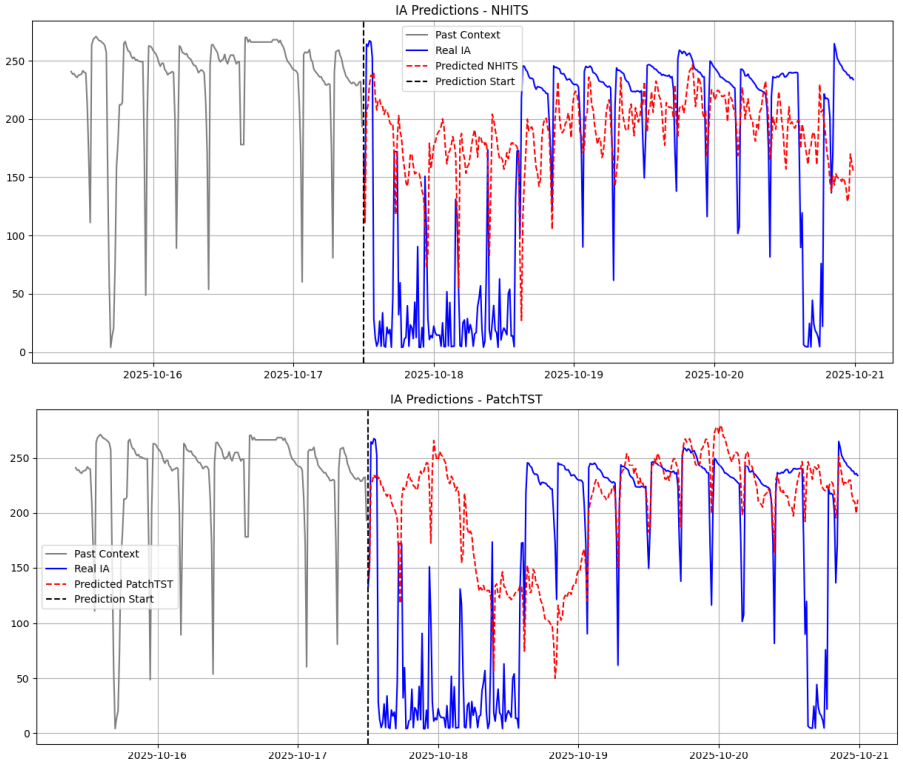


Figura 28 – NHITS e PatchTST: Predição para H=336.

Conclusão

A manutenção preditiva em sistemas de refrigeração industrial atua como um mecanismo fundamental para evitar paradas operacionais não planejadas e trazer mais eficiência energética para esse ambiente. Contudo, a análise contínua de dados sensoriais das indústrias impõe um grande desafio analítico devido ao ruído e à complexidade na identificação precoce de anomalias, como o estado operacional de falha de "carga baixa"em unidades condensadoras. Diante dessa problemática, este trabalho atingiu o seu objetivo central de avaliar e comparar a eficácia de diferentes arquiteturas de Inteligência Artificial para a predição de séries temporais reais. Através da implementação de modelos de Aprendizado Profundo, desde de arquiteturas mais clássicas como a LSTM, de modelos de decomposição linear, até redes mais robustas e modernas baseadas em mecanismos de atenção e interpolação hierárquica, o estudo comprovou a capacidade dessas ferramentas em modelar o comportamento das máquinas e viabilizar a antecipação de falhas operacionais em um cenário de produção real. O desempenho alcançado reitera a viabilidade de sua aplicação prática, uma vez que os modelos foram desafiados a superar gargalos inerentes às condições dinâmicas de operação industrial real, como presença de ruídos, lacunas temporais por problemas de conectividade e particularidades operacionais únicas decorrentes do histórico de desgaste individual de cada ativo no chão de fábrica.

Os experimentos realizados dividiram-se em duas frentes metodológicas: a modelagem global (*global forecasting*), desenvolvida para capturar padrões compartilhados entre grupos de máquinas semelhantes, e a modelagem local (*local forecasting*), focada no comportamento individualizado do ativo B134. Na abordagem global univariada para as máquinas do tipo MT, o modelo PatchTST demonstrou resultados expressivos na identificação de tendências dentro da faixa de carga nominal. Esse desempenho atribui-se à sua estratégia de amostragem em blocos locais (*patches*), que preserva a integridade semântica local da série, e ao mecanismo de independência de canais, que trata cada variável de forma isolada para mitigar o sobreajuste e evitar a propagação de ruídos cruzados no espaço latente. O modelo NHITS acompanhou essa tendência positiva em variações sutis de carga. Por outro lado, nas unidades do tipo B, a incorporação de variáveis exógenas elevou o patamar de assertividade das redes LSTM e NHITS. Esse ganho foi impulsionado pelo suporte nativo de ambas as arquiteturas para o processamento de séries temporais auxiliares diretamente em suas camadas de entrada, permitindo mapear correlações físicas de alta relevância, como o fator de potência das fases.

No entanto, os resultados mais consistentes e de maior aderência à realidade prática foram alcançados pela modelagem local no horizonte de previsão de 96 passos. Nessa configuração, o NHITS consolidou-se como o principal destaque de todo o estudo, obtendo

a maior acurácia preditiva ao se sobrepor com precisão à curva real após a calibração e otimização hierárquica dos coeficientes de suas *stacks*. Em horizontes de longo prazo, como 192 e 336 passos, os modelos PatchTST e NHITS também exibiram notável desempenho ao antecipar corretamente o momento exato da queda e o posterior retorno do equipamento às condições normais de funcionamento.

Do ponto de vista prático, o estudo comprova a viabilidade da manutenção preditiva baseada em IA para otimizar o consumo energético. Ao demonstrar a capacidade analítica de antecipar o estado de "carga baixa", o trabalho consolida uma ferramenta com potencial para evitar paradas não planejadas, garantindo a integridade dos produtos armazenados e prevenindo falhas. Já no âmbito teórico, a pesquisa enriquece a literatura ao validar algoritmos complexos em um *dataset* genuíno e ruidoso de chão de fábrica. Os resultados comprovam que arquiteturas baseadas em *patching* com independência de canais são fundamentais para preservar a semântica local das séries, enquanto mecanismos de amostragem multitaxa e interpolação hierárquica mostram-se cruciais para isolar variações de alta frequência e capturar tendências de degradação. Além disso, arquiteturas já consolidades como LSTM podem auxiliar de maneira eficaz no escopo da manutenção preditiva.

Nesse cenário, é importante pontuar que o NHITS figura como modelo principal para o emprego prático na indústria. Sua arquitetura estabelece um equilíbrio estrutural ideal, superando a simplicidade limitante de modelos lineares e RNNs, sem necessitar de uma densa complexidade e alto custo computacional exigidos pelas arquiteturas *Transformers*. Além disso, a sua natureza de dividir o sinal em componentes de hierarquia permite extrair e visualizar a saída de cada *stack* individualmente, gerando um modelo interpretável. Essa transparência mitiga problemas de "caixa preta"associado a algoritmos tradicionais, oferecendo confiabilidade e justificativa técnica. Por fim, sua flexibilidade na integração de variáveis exógenas consolida o NHITS como solução robusta, permitindo enriquecer o contexto das predições com múltiplas grandezas.

Em relação às limitações da pesquisa, as restrições de infraestrutura computacional e de *hardware* representaram um gargalo relevante para os experimentos. A demanda por recursos computacionais modernos inviabilizou a avaliação global e em horizontes extensos do modelo Autoformer, além de restringir a busca exaustiva e o aprofundamento das técnicas de otimização para o PatchTST.

Ademais, o próprio conjunto de dados do ecossistema fabril impôs barreiras quanto ao volume e distribuição cronológica das amostras. A presença de janelas históricas reduzidas e o baixo índice de falhas registradas em determinados sensores mais recentes, como o B417 e B419, limitaram o tamanho do contexto de entrada e forçaram a delimitação de cenários com horizontes mais restritos a 48 passos na modelagem global. Por fim, a persistência de *overfitting* nas redes lineares e recorrentes sinaliza a existência de

limitações técnicas na etapa de *feature engineering* ou evidencia um baixo fator preditivo intrínseco aos dados brutos disponíveis. Esse comportamento sugere que as variáveis de corrente elétrica e fator de potência, isoladamente, podem ser insuficientes para conter toda a variabilidade e dinâmica complexa que antecedem o fenômeno de carga baixa.

Diante dos resultados obtidos e das limitações identificadas, abrem-se panoramas de exploração para continuidade desta pesquisa. No que tange ao enriquecimento de dados, sugere-se o aprofundamento na etapa de engenharia de atributos, bem como a incorporação de novas variáveis físicas, mecânicas e termodinâmicas do ciclo de compressão dessas máquinas, podendo elevar o fator preditivo e mitigar casos de sobreajuste. Ademais, recomenda-se a validação de modelos baseados em mecanismos de atenção mais densos, como o Autoformer, e uma calibração exaustiva do PatchTST utilizando ambientes de alto desempenho, além de implementações (ou abertura de *issues* na biblioteca *NeuralForecast*) que viabilizem a adição de variáveis exógenas nessas arquiteturas.

Sob a perspectiva metodológica e modelagem de dados, propõe-se a evolução para uma abordagem de modelagem global direcionada a subgrupos de ativos. Essa estratégia envolveria a investigação de mecanismos avançados de comparação de séries temporais para avaliar as assinaturas elétricas e os padrões de degradação, unindo em um mesmo treinamento apenas as máquinas que apresentarem comportamentos e ocorrências de falhas altamente semelhantes. A título de validação preliminar e de maneira puramente exploratória, essa premisa foi testada mediante o agrupamento dos sensores B423 e B426 sob a arquitetura NHITS. A experimentação inicial produziu resultados promissores, evidenciando um ganho preditivo positivo em relação ao B426 (Figura 29).

Em paralelo, sugere-se a exploração de novas arquiteturas preditivas com alta capacidade nativa para lidar com contextos multivariados complexos. A adoção dessas estruturas permitiria expandir a integração de variáveis exógenas para além das métricas puramente históricas abordadas neste trabalho, possibilitando a injeção de atributos estáticos (como especificações mecânicas de cada equipamento) e covariáveis futuras conhecidas (como previsões climáticas ou cronogramas de produção). Por fim, para mitigar o viés gerado pelo desbalanceamento do *dataset*, torna-se imperativa a aplicação de mecanismos de *data augmentation* para enriquecer o treinamento de séries temporais que possuem poucos registros reais de "carga baixa", sempre preservando a lógica temporal das séries.

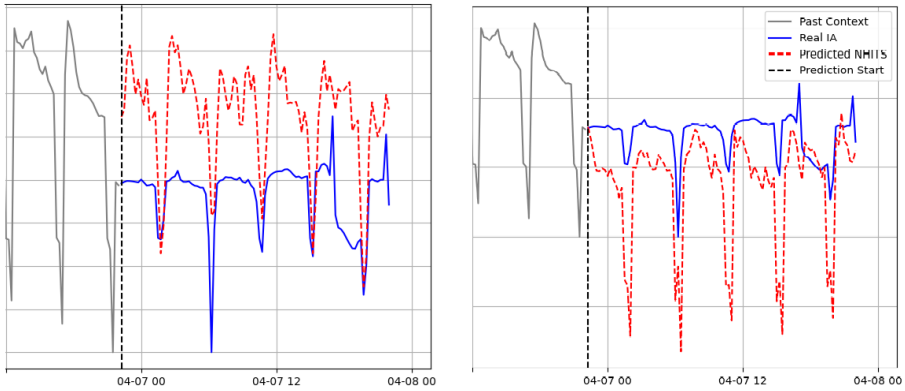


Figura 29 – Esquerda: Modelagem Global; Direita: Modelagem Global reduzida para B423 e B426.

Referências

ABRAVA. *A Relação da Refrigeração e a Eficiência Energética*. 2023. Acesso em: 22 de junho de 2026. Disponível em: <https://materiais.abrava.com.br/cartilha-eficiencia-energetica-em-refrigeracao>https://materiais.abrava.com.br/cartilha-eficiencia-energetica-em-refrigeracao.

BERGSTRA, J.; YAMINS, D.; COX, D. D. *Making a Science of Model Search*. 2012. Disponível em: <https://arxiv.org/abs/1209.5111>https://arxiv.org/abs/1209.5111.

CHALLU, C. et al. *N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting*. 2022. Disponível em: <https://arxiv.org/abs/2201-.12886>https://arxiv.org/abs/2201.12886.

HANIFI, S. et al. Advancements in predictive maintenance modelling for industrial electrical motors: Integrating machine learning and sensor technologies. *Measurement: Sensors*, v. 38, p. 101473, 2025. ISSN 2665-9174. Proceedings of the XXIV IMEKO World Congress. Disponível em: <https://www.sciencedirect.com/science/article/pii-/S2665917424004495>https://www.sciencedirect.com/science/article/pii/S2665917424004495.

LIAW, R. et al. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

NIE, Y. et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. Disponível em: <https://arxiv.org/abs/2211-.14730>https://arxiv.org/abs/2211.14730.

NIXTLA. *LSTM Model Documentation - NeuralForecast*. 2026. Acesso em: 16 de março de 2026. Disponível em: <https://nixtlaverse.nixtla.io/neuralforecast/models.lstm-.html>https://nixtlaverse.nixtla.io/neuralforecast/models.lstm.html.

OLIVARES, K. G. et al. *NeuralForecast: User friendly state-of-the-art neural forecasting models.* 2022. PyCon Salt Lake City, Utah, US 2022. Disponível em: <https://github.com/Nixtla/neuralforecast>https://github.com/Nixtla/neuralforecast.

ORESHKIN, B. N. et al. *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting*. 2020. Disponível em: <https://arxiv.org/abs/1905-.10437>https://arxiv.org/abs/1905.10437.

SCAIFE, A. D. Improve predictive maintenance through the application of artificial intelligence: A systematic review. *Results in Engineering*, v. 21, p. 101645, 2024. ISSN 2590-1230. Disponível em: <https://www.sciencedirect.com/science/article/pii-/S2590123023007727>https://www.sciencedirect.com/science/article/pii/S2590123023007727.

VASWANI, A. et al. *Attention Is All You Need*. 2023. Disponível em: <https://arxiv-.org/abs/1706.03762>https://arxiv.org/abs/1706.03762.

WU, H. et al. *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*. 2022. Disponível em: <https://arxiv.org/abs/2106-.13008>https://arxiv.org/abs/2106.13008.

ZENG, A. et al. Are transformers effective for time series forecasting? In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2023. v. 37, n. 9, p. 11121–11128.

ZHAO, R. et al. *Deep Learning and Its Applications to Machine Health Monitoring: A Survey*. 2016. Disponível em: <https://arxiv.org/abs/1612-.07640>https://arxiv.org/abs/1612.07640.

ZONTA, T. et al. Predictive maintenance in the industry 4.0: A systematic literature review. *Computers Industrial Engineering*, v. 150, p. 106889, 2020. ISSN 0360-8352. Disponível em: <https://www.sciencedirect.com/science/article/pii-/S0360835220305787>https://www.sciencedirect.com/science/article/pii/S0360835220305787.

Apêndices

APÊNDICE A – Hiperparâmetros

A.1 Apêndices - LSTM

HIPERPARÂMETRO	**VALORES**
Taxa de aprendizado.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m = 5$
Tamanho do lote.	{32}
Função de ativação.	ReLU
Camadas do *encoder*.	$E \in \{2\}$
Tamanho da camada oculta do *encoder*.	$E_s \in \{64\}$
Tamanho da camada oculta do *decoder*.	$D_s \in \{64\}$
Camadas do *decoder*.	$D \in \{2\}$

Tabela 11 – Configuração LSTM para $H = \{48, 96\}$

Para $H = 192$ e $H = 336$, o melhor resultado foi observado com $m = 7$, mantendo as demais configurações da Tabela 11.

A.2 Apêncides - NHITS

HIPERPARÂMETRO	**VALORES**
Taxa de aprendizado inicial.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m = 5$
Tamanho do Lote.	{32}
Função de Ativação.	ReLU
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1]\}$
Número de *Stacks*.	$S \in \{3\}$
Blocos por *stack*.	$B \in \{1\}$
Camadas MLPs.	{3}
Tamanho Oculto dos Coeficientes.	$N_h \in \{512\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[2, 1, 1]\}$
Estratégia de interpolação	$g(\tau, \theta) \in \{\text{Linear}\}$

Tabela 12 – Configuração NHITS para $H = \{48\}$

Utilizando como base a configuração da Tabela 15, os hiperparâmetros explorados nos demais horizontes são observados nas tabelas:

HIPERPARÂMETRO	**VALORES**
Input size multiplier $(L = m * H)$.	$m = 1$
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1]\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[4, 4, 1]\}$

Tabela 13 – Configuração NHITS para $H = \{96\}$

HIPERPARÂMETRO	**VALORES**
Input size multiplier $(L = m * H)$.	$m = 2$
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1]\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[4, 4, 1]\}$

Tabela 14 – Configuração NHITS para $H = \{192\}$

HIPERPARÂMETRO	**VALORES**
Input size multiplier $(L = m * H)$.	$m = 5$
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1]\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[2, 1, 1]\}$

Tabela 15 – Configuração NHITS para $H = \{336\}$

A.3 Apêncides - NLinear e DLinear

HIPERPARÂMETRO	**VALORES**
Taxa de aprendizado.	{1e-4, 1e-3}
Passos de treinamento.	{5000}
Input size multiplier $(L = m * H)$.	$m = 5$
Tamanho do Lote.	32
Escalonamento.	{identity, robust}
Janela de Média Móvel (DLinear)	{25}

Tabela 16 – Configuração NLinear e DLinear para $H = 48$

Tomando como base a configuração em Tabela 16, para $H = 96$ e $H = 336$ foi obtido o melhor resultado para ambos os modelos com $m = 5$. Para $H = 192$, o melhor resultado foi obtido com $m = 2$ para os dois modelos.

A.4 Apêndices - PatchTST

HIPERPARÂMETRO	**VALORES**
Taxa de Aprendizado.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m = 1$
Number of Multi-Head's Attention.	{8}
Camadas de *encoder*	3
Unidades de *Embeddings* e *encoders*	128
Unidades de Camadas Lineares	256
Tamanho do *patch*	{16}
Stride do *patch*	{8}
Escalonamento.	{robust}
Dropout.	0.3

Tabela 17 – Configuração PatchTST para $H = \{48, 336\}$

Considerando Tabela 17 como base, as configurações consideradas para $H = 96$ e $H = 192$ são observadas em Tabela 18.

HIPERPARÂMETRO	**VALORES**
Number of Multi-Head's Attention.	{8}
Tamanho do *patch*	{8}
Stride do *patch*	{4}

Tabela 18 – Configuração PatchTST para $H = \{96, 192\}$

A.5 Apêndices - Autoformer

HIPERPARÂMETRO	**VALORES**
Taxa de Aprendizado.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m = 1$
Number of Multi-Head's Attention.	{8}
Unidades de *Embeddings* e *encoders*	{64}
Probsparse Attention Factor	3
Camadas de *encoder*	2
Camadas de *decoder*	1
Tamanho do filtro de média móvel	25
Scaler Type.	{identity}

Tabela 19 – Configuração Autoformer para $H = \{48, 96\}$

A.6 Apêndices - AutoNHITS

HIPERPARÂMETRO	**VALORES**
Taxa de aprendizado inicial.	{1e-3}
Passos de treinamento.	{5000}
Input size multiplier ($L = m * H$).	$m = 1$
Tamanho do Lote.	{32}
Função de Ativação.	ReLU
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 2]\}$
Número de *Stacks*.	$S \in \{3\}$
Blocos por *stack*.	$B \in \{1\}$
Camadas MLPs.	{3}
Tamanho Oculto dos Coeficientes.	$N_h \in \{512\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[1, 1, 1]\}$
Estratégia de interpolação	$g(\tau, \theta) \in \{\text{Linear}\}$

Tabela 20 – Configuração AutoNHITS para $H = \{48\}$

Considerando a configuração base em Tabela 20, para os demais horizontes as configurações com melhores resultados são evidenciadas nas tabelas Tabela 21 e Tabela 22.

HIPERPARÂMETRO	**VALORES**
Input size multiplier ($L = m * H$).	$m = 1$
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[2, 2, 1]\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[8, 4, 1]\}$

Tabela 21 – Configuração AutoNHITS para $H = \{96\}$

HIPERPARÂMETRO	**VALORES**
Input size multiplier ($L = m * H$).	$m = 2$
Tamanho do Kernel de *Pooling*.	$[k_1, k_2, k_3] \in \{[16, 8, 1]\}$
Número de Coeficientes das *Stacks*.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in \{[24, 12, 1]\}$

Tabela 22 – Configuração AutoNHITS para $H = \{192\}$