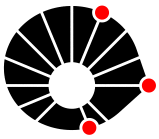

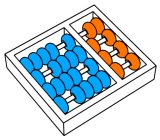

Análise Comparativa de Arquiteturas de Deep Learning para Predição de Tráfego em Slices de Redes 5G

H. P. B. Vasconcellos *E. R. M. Madeira*

Relatório Técnico - IC-PFG-26-29
Projeto Final de Graduação
2026 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Análise Comparativa de Arquiteturas de *Deep Learning* para Predição de Tráfego em *Slices* de Redes 5G

Hayla Pereira Belozi Vasconcellos* Edmundo Roberto Mauro Madeira†

Resumo

O advento das redes móveis 5G e da tecnologia de *Network Slicing* viabilizou a coexistência de fatias lógicas heterogêneas (eMBB, URLLC e mMTC) sobre uma mesma infraestrutura física de rádio. Para mitigar congestionamentos e assegurar os exigentes Acordos de Nível de Serviço (SLAs), o núcleo de rede requer previsões de tráfego proativas de múltiplos passos diretos (DMS) extremamente ágeis e precisas. Este trabalho apresenta uma avaliação comparativa de desempenho (*benchmarking*) sistemática entre modelos de *Deep Learning* baseados em atenção global (PatchTST e suas variantes de ablação) e estimadores de menor sobrecarga computacional (CNN1D, LSTM, DLinear, SARIMA, SNaive e Naive) utilizando o conjunto de dados de simulação realística *DeepSlice*. A modelagem foi estruturada sobre uma série agregada horária de 168 horas, dividida cronologicamente (70% treino e 30% teste), com janela de histórico de $L = 48$ horas e horizontes preditivos de curto ($H = 12$) e longo prazo ($H = 24$). Os resultados empíricos na escala física desnormalizada de solicitações por hora revelaram a supremacia absoluta da arquitetura convolucional compacta CNN 1D operando sob independência de canais, que assinalou os menores desvios globais sendo seguida de perto pelos *baselines* sazonais. Em contrapartida, as variantes do PatchTST e a rede recorrente LSTM sofreram severos colapsos preditivos induzidos por *overfitting* de alta variância, dispersão de entropia de atenção e contaminação cruzada de escala, validando empiricamente a teoria do *finite-sample gap* sob regimes de amostragem finita. Conclui-se que a simplicidade estrutural e a regularização espacial da CNN 1D unificam exatidão e viabilidade computacional periférica, consolidando-a como o motor preditivo ideal para a orquestração dinâmica do núcleo celular 5G.

1 Introdução

O advento das redes móveis de quinta geração (5G) estabeleceu um novo paradigma arquitetural projetado para superar as limitações de escalabilidade das redes da geração anterior, baseadas na tecnologia LTE (*Long-Term Evolution*) e atender a rigorosos padrões de confiabilidade, capacidade de vazão e latência ultrabaixa. Nesse cenário, a tecnologia de *Network Slicing* (Fatiamento de Rede) consolida-se como o pilar da infraestrutura 5G [1], permitindo a criação e orquestração de múltiplas redes lógicas isoladas que operam simultaneamente sobre um único *hardware físico* compartilhado. Essa virtualização viabiliza a alocação de recursos sob medida para os usuários, visando garantir uma Qualidade de Serviço (QoS) superior perante um volume massivo e dinâmico de dados.

A heterogeneidade de requisitos nesse ecossistema é suportada por três classes padronizadas de fatias [2]: a *enhanced Mobile Broad Band* (eMBB), otimizada para alta capacidade de vazão (do inglês, *throughput*); a *massive Machine Type Communication* (mMTC), projetada para suportar a alta densidade de conexões da Internet das Coisas (IoT); e a *Ultra Reliable Low Latency Communication* (URLLC), que impõe restrições severas de latência e é indispensável para cenários mais

*Inst. de Computação, UNICAMP, 13083-852 Campinas, SP.
†Inst. de Computação, UNICAMP, 13083-852 Campinas, SP.

críticos, como a Indústria 4.0 e aplicações médicas [2]. O grande desafio de engenharia imposto por esse paradigma reside justamente em acomodar demandas operacionais tão antagônicas na mesma topologia de rede.

Para garantir os rigorosos Acordos de Nível de Serviço (SLA) de todas as fatias, especialmente as mais sensíveis a atrasos, abordagens reativas de gerenciamento não são suficientes. É necessário um monitoramento de tráfego contínuo e proativo, capaz de antecipar gargalos antes de sua ocorrência. É nesse ponto que a Inteligência Artificial se torna imprescindível [3]. Embora a literatura recente tenha avançado na orquestração baseada no estado atual da rede, a verdadeira alocação dinâmica de recursos depende da precisão na previsão contínua do tráfego futuro [4]. Entretanto, o próprio modelo de Inteligência Artificial precisa ser capaz de gerar essas previsões em um curto tempo de inferência, uma vez que um algoritmo excessivamente lento pode se tornar um gargalo na própria rede e fazer com que ela perca a janela de tempo necessária para a tomada de decisão proativa [5].

Nesse contexto, o objetivo central deste projeto é promover uma avaliação comparativa (*benchmarking*) do desempenho de diferentes arquiteturas de *Deep Learning* na previsão de séries temporais voltadas exclusivamente para o monitoramento do tráfego em fatias de rede 5G. O trabalho foca em investigar se o alto custo computacional de arquiteturas de estado da arte baseadas em atenção global (como os *Transformers*) traduz-se em vantagem preditiva real que justifique sua adoção, ou se modelos localmente restritos e mais leves, como redes lineares (Decomposição Linear - DLinear) e *ensembles* híbridos (CNN 1D + LSTM), atendem a essa demanda preditiva com precisão superior e menor tempo de inferência. Para viabilizar o treinamento e a comparação empírica dessas redes, utilizamos um conjunto de dados público de tráfego sintético, originalmente gerado e disponibilizado no estudo de fatiamento de redes 5G de A. Thantharate et al. [2], publicado em 2019. Dessa forma, este estudo atua estritamente na análise de viabilidade e eficiência algorítmica, fornecendo evidências sobre qual arquitetura oferece a melhor relação custo-benefício para lidar com o comportamento oscilante do tráfego de rede.

A fim de contemplar os objetivos propostos, este relatório está estruturado em seis seções, incluindo a introdução. A seção 2 apresenta a fundamentação teórica sobre as limitações matemáticas dos *Transformers* em séries temporais, detalhando fenômenos como a invariância à permutação e o decaimento posicional. A seção 3 embasa as arquiteturas lineares e híbridas como soluções preditivas de baixa latência. A seção 4 descreve a metodologia e o desenvolvimento laboratorial, englobando o pré-processamento, a estruturação de janelas deslizantes e os parâmetros do estudo de ablação do modelo PatchTST. A seção 5 compila os resultados e a discussão empírica, correlacionando as falhas visuais dos *Transformers* com evidências literárias recentes. Por fim, a seção 6 sintetiza as conclusões do trabalho e aponta perspectivas futuras.

2 Arquiteturas Baseadas em Atenção e suas Limitações em Séries Temporais

O advento das arquiteturas baseadas em *Transformers* estabeleceu um novo paradigma analítico na modelagem e na previsão de séries temporais. Desenvolvidos inicialmente para tarefas complexas de processamento de linguagem natural, esses modelos substituíram as abordagens tradicionais baseadas em recorrência, a exemplo das Redes Neurais Recorrentes (RNN), ao demonstrarem uma capacidade superior de capturar dependências de longo alcance e de processar dados em larga escala de maneira eficiente [13] [11]. No escopo específico de redes computacionais e análise de telemetria, a literatura recente corrobora enfaticamente a primazia destas arquiteturas na precisão e na previsão proativa de tráfego. Conforme detalhado por Q. Xin et al. em seu artigo [4], publicado em 2025, a implementação de sistemas de vanguarda como o *MegaMon*, uma ferramenta baseada

em *Transformers* projetada para coletar sequências de tráfego em tempo real de interfaces de rede e realizar a detecção preditiva de anomalias, comprova de forma empírica e algorítmica que essas redes são capazes de superar topologias clássicas de aprendizado profundo. Esses novos modelos entregam o estado da arte não apenas em eficiência temporal de treinamento, mas também na minimização estrita de métricas de erro na predição de sequências complexas de tráfego.

Sob a égide deste paradigma de êxito arquitetural, a adoção metodológica inicial deste trabalho por uma abordagem baseada em *Transformers*, especificamente o modelo *Patch Time Series Transformer* (PatchTST), não é arbitrária. O corpo literário, a exemplo do estudo de V. Naghashi et al. [12], de 2025, afirma categoricamente que o PatchTST é capaz de comprovar a eficiência da arquitetura *Transformer* na análise de séries temporais intrincadas e complexas. Ainda que debates acadêmicos recentes questionem o desempenho de redes de atenção quando comparadas a regressores lineares, tal modelo mostrou que, por meio da injeção de inovações topológicas, como o particionamento sequencial dos dados em janelas temporais fixas acoplado ao processamento em regime de independência de variáveis, o *Transformer* se mantém plenamente estabelecido como um dos instrumentais algorítmicos mais poderosos para o tratamento não linear de séries de dados na atualidade [12].

Contudo, a despeito deste sucesso estatístico na predição de tráfego e na modelagem de representações complexas e não lineares, a transposição de arquiteturas baseadas em *Transformers* para o núcleo de controle de redes 5G exige uma validação rigorosa. A orquestração de recursos via *Network Slicing* lida com um ambiente físico de alta volatilidade, marcado por picos abruptos de demanda e restrições severas de latência. Em fatias de rede destinadas a serviços críticos, por exemplo, atrasos na inferência algorítmica e na detecção de anomalias induzidos por modelos de alta complexidade computacional são intoleráveis. Desse modo, para elucidar as razões teóricas pelas quais uma arquitetura consolidada no estado da arte possui a propensão de falhar ou tornar-se inviável neste rigoroso cenário de telecomunicações, é imprescindível desconstruir primeiramente a base matemática do mecanismo de autoatenção linear, responsável pelo seu funcionamento central.

2.1 O Mecanismo de Autoatenção e sua Invariância à Permutação

A arquitetura *Transformer*, formalizada originalmente por Vaswani et al. em 2017 [6], estabelece um paradigma de processamento de sequências que diverge substancialmente dos modelos estruturais predecessores ao suprimir integralmente a utilização de redes neurais recorrentes e convolucionais. Historicamente, a modelagem sequencial clássica repousava sobre a fatoração computacional temporal estrita ao longo das posições dos símbolos de entrada e saída, gerando um alinhamento passo a passo que impedia, de forma inerente, a paralelização algorítmica, dado que o estado oculto num instante dependia do estado no passo imediatamente anterior. Para transpor essa barreira temporal e viabilizar a modelagem de dependências globais sem considerar a distância entre as posições na sequência, o *Transformer* baseia-se exclusivamente em mecanismos de atenção. Sob o aspecto global, a topologia mantém uma infraestrutura de codificador-decodificador (*Encoder-Decoder*): o codificador mapeia as representações dos símbolos de entrada para uma sequência de representações contínuas globais, atuando como um referencial sobre o qual o decodificador atua iterativamente e de forma autorregressiva para sintetizar os símbolos de saída.

Do ponto de vista físico da rede, o codificador é estruturado por uma pilha de seis camadas estritamente idênticas, dotadas de duas subcamadas fundamentais: um mecanismo de autoatenção de múltiplas cabeças (*Multi-Head Self-Attention*) e uma rede neural de alimentação direta pontual e totalmente conectada (*Position-wise Feed-Forward Network*). Para salvaguardar a propagação do gradiente perante o aprofundamento das camadas, emprega-se uma conexão residual ao redor de cada uma das subcamadas descritas, seguida por uma etapa de normalização de camada (*Layer*

Normalization). O decodificador espelha essa modularidade de seis camadas idênticas, porém insere uma terceira subcamada intermediária, dedicada exclusivamente a processar a atenção sobre a representação contínua oriunda da pilha do codificador. Ademais, para conservar o imperativo da causalidade temporal intrínseca à regressão passo a passo, o decodificador introduz o mascaramento (*Masking*) na sua primeira subcamada de autoatenção; essa restrição algorítmica inibe conexões à direita, assegurando rigorosamente que as predições para uma dada posição dependam tão somente de posições passadas já computadas.

O núcleo analítico que habilita a inter-relação global dos dados abandona a direcionalidade, operando via uma formulação matemática definida como Atenção de Produto Escalar Escalonado (*Scaled Dot-Product Attention*). Conceitualmente, a função de atenção atua realizando o mapeamento espacial de uma matriz de Consultas (*Queries*) juntamente com um conjunto de pares contendo matrizes de Chaves (*Keys*) e Valores (*Values*) para gerar uma nova saída vetorial. Para otimização máxima dos cálculos matriciais, as entradas são empacotadas simultaneamente em tensores denotados por Q, K e V, em que as consultas e chaves compartilham estritamente a mesma dimensionalidade computacional d_k, enquanto os valores operam sobre uma dimensão d_v. A correlação pareada da matriz inteira determina pesos sinápticos computados por meio de uma função de compatibilidade, que avalia o alinhamento da consulta com sua respectiva chave e atribui a relevância ponderada de cada valor no cômputo da matriz de saída.

A operação algébrica singular do mecanismo é consolidada na equação:

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{d_k})V \tag{1}$$

O cálculo processa preliminarmente o produto interno da matriz de consultas (Q) pela transposta da matriz de chaves (K^T). Contudo, Vaswani et al [6]. diagnosticaram que dimensões vetoriais extensas (altos valores numéricos de d_k) forçam as magnitudes do produto escalar a crescerem vertiginosamente. Esse ganho escalar excessivo empurra irremediavelmente as distribuições resultantes da ativação *softmax* subsequente para regiões assintóticas operacionais saturadas, dominadas por gradientes ínfimos e quase nulos, paralisando a convergência da rede. A mitigação analítica mandatória proposta exige que todos os produtos escalares sejam divididos matematicamente pelo fator de escala e regularização $\sqrt{d_k}$ antes de extrair as probabilidades finais de atenção.

Embora este refinado esquema de processamento em bloco, de tempo computacional logarítmico ou constante em determinados eixos, propicie formidável capacidade de paralelização na captura de correlações semânticas em domínios ricos em contexto linguístico, a dependência primária da atenção baseada no produto interno simultâneo suprime integralmente a causalidade estrita e natural das sequências [7]. Em face de aplicações baseadas na predição proativa a longo prazo (LTSF), o abandono dos processos de recorrência consolida a topologia de autoatenção como estrutural e matematicamente invariante à permutação, exigindo injeções artificiais de posição que, ainda assim, acarretam em inevitáveis perdas temporais ao manusear séries analíticas numéricas governadas pela cronologia do tráfego [11], fundamentando as desvantagens arquiteturais sistêmicas de seu uso em *Network Slicing* para 5G.

2.2 A Codificação Posicional e a Perda de Informação

Para mitigar a invariância à permutação inerente ao mecanismo de autoatenção, a literatura recorre à injeção de mecanismos de *Positional Encoding* (PE). A premissa analítica dessa técnica consiste em incorporar coordenadas cronológicas diretamente nos tensores de entrada, dotando a rede da capacidade de assimilar a ordem sequencial dos dados [9]. Nas formulações clássicas de codificação absoluta, a exemplo das matrizes senoidais fixas, soma-se um vetor determinístico de

localização a cada instante de tempo, fundindo a posição espacial ao conteúdo original dos dados antes do processamento nas camadas de atenção. Em contrapartida, as metodologias de codificação relativa alteram o núcleo algébrico do cálculo de atenção, inserindo matrizes de viés que modelam explicitamente a distância temporal entre os pares de Q e K. Independentemente da vertente metodológica adotada, o imperativo arquitetural permanece unânime: forçar uma topologia inerentemente "anti-ordem"a capturar a causalidade dinâmica, de modo a viabilizar predições proativas em séries temporais nas quais a estrita cronologia governa a validade informacional do sistema.

Apesar da teórica robustez inicial destas formulações baseadas em injeção de tensores, trabalhos recentes evidenciam falhas arquiteturais na retenção da causalidade ao longo do processamento nas redes de autoatenção. Conforme exposto no artigo de 2024 de Zhang et al. [8], constata-se de forma matemática que a Informação Mútua entre a codificação posicional original e as representações contínuas dos *tokens* sofre um decaimento drástico e sistêmico à medida que a profundidade das camadas da rede aumenta. Esse fenômeno evidencia que as sucessivas projeções lineares, multiplicações matriciais simultâneas e regularizações pontuais inerentes às pilhas do codificador acabam por dissipar a injeção posicional, fazendo com que as representações nas camadas mais profundas tornem-se essencialmente agnósticas à passagem do tempo. No contexto algorítmico do *Network Slicing* 5G, em que a inferência proativa da cronologia exata do tráfego governa a alocação de recursos físicos, essa amnésia posicional corrompe a capacidade preditiva do modelo de longo prazo (LTSF).

Na tentativa de consertar o decaimento da ordem cronológica em subcamadas profundas, a literatura recente tem desenvolvido esquemas de codificação cada vez mais complexos e sobrepostos, a exemplo do *Temporal Positional Encoding* (T-PE). Essa estrutura metodológica atua combinando matrizes geométricas tradicionais a componentes semânticos altamente avançados, calculando o grau de similaridade vetorial entre *tokens* em diferentes posições para identificar dependências temporais intrincadas independentemente das localizações absolutas no lote de dados [8]. Contudo, esse artifício sofisticado eleva de maneira assintótica a complexidade algorítmica e o consumo de memória do modelo de regressão. Avaliações empíricas rigorosas sobre as sobrecargas de treinamento comprovam que técnicas avançadas de posicionamento impõem penalidades computacionais superlativas; arquiteturas operando com T-PE exigem tempos computacionais até 2,79 vezes superiores em comparação aos tensores fixos senoidais clássicos [9], denotando uma ineficiência severa. Consequentemente, para a infraestrutura 5G padronizada, que exige reconfigurações dinâmicas de latência ultrabaixa para fatias críticas de serviços (URLLC), a implementação de *Transformers* atrelados a mitigadores posicionais algorítmicos lentos torna a abordagem impraticável e inferior às redes recorrentes, como a arquitetura *Long Short-Term Memory* (LSTM) e topologias lineares diretas, que processam a cronologia intrinsecamente e com custo computacional consideravelmente menor.

2.3 *Finite-Sample Gap* e o Colapso Autoregressivo

Para além da ineficiência das codificações posicionais na captura da ordem temporal, os *Transformers* enfrentam limitações matemáticas insuperáveis no espaço de representação de características. Se por um lado o modelo consagrou-se no estado da arte ao suprimir a recorrência em favor de um mecanismo exclusivo de autoatenção [6], por outro provou-se que o levantamento tensorial gerado por esse mecanismo introduz graus de liberdade excessivos e matematicamente opacos [15]. Dessa forma, a aplicação direta dessa arquitetura em regressões temporais contínuas é obstruída por rigorosas fronteiras teóricas.

O estudo analítico de Zhou et al. [14], publicado em 2025, investiga o mecanismo estatístico da Autoatenção Linear (*Linear Self-Attention* - LSA) contrastando-o com estimadores clássicos

como o Método dos Mínimos Quadrados Ordinários (OLS). Ao analisar a aplicação do mecanismo de atenção à modelagem de processos estritamente autorregressivos, os autores provam matematicamente a existência de uma "lacuna estrita de amostra finita"(*strict finite-sample gap*). Sob um regime de amostragem finita, realidade inerente à coleta contínua de tráfego em redes físicas, a demonstração estabelece que o limite inferior de erro preditivo do *Transformer* é estruturalmente superior ao de uma regressão linear OLS comum. Consequentemente, mesmo sob parametrização ótima, arquiteturas baseadas em atenção conseguem apenas rastrear e simular a eficiência de equações lineares simples, sendo matematicamente bloqueadas de superá-las em predições de longo prazo.

Essa limitação basal no espaço de representação agrava-se severamente no cenário de previsão temporal de múltiplos passos iterativos (*multi-step forecasting*). Durante a decodificação, o modelo atua de forma autorregressiva ao consumir suas próprias previsões como entradas subsequentes, dinâmica denominada *Chain-of-Thought Rollout*. Contudo, no domínio de séries temporais estritas, essa mecânica induz um colapso estrutural. Devido à lacuna de amostra finita previamente mencionada, as inferências locais do modelo introduzem pequenos desvios residuais de variância e, ao retroalimentar esses erros sucessivamente na matriz de entrada para estender o horizonte preditivo, os resíduos acumulam-se e são amplificados a uma taxa exponencial. Como resultado do acúmulo de erro, a arquitetura perde a coerência causal temporal, colapsando inevitavelmente a sua previsão para a média do processo ou dissipando seu sinal em ruídos infrutíferos para o controle da rede.

No escopo do provisionamento de recursos via *Network Slicing* em redes 5G, essas limitações matemáticas inviabilizam a aplicação prática de *Transformers*. O mecanismo de autoatenção impõe uma complexidade computacional quadrática $O(n^2 d)$ atrelada à extensão da sequência temporal analisada. Arcar com essa massiva sobrecarga de processamento e memória torna-se insustentável perante a comprovação de que o modelo falha estruturalmente em superar preditores lineares simples e sofre degeneração exponencial de erro em múltiplos passos [6] [25]. Especialmente em fatias de missão crítica (URLLC), que operam sob restrições de latência na escala de sub-milissegundos e tolerância zero a falhas, o uso de algoritmos morosos e suscetíveis a colapsos preditivos viola diretamente os requisitos operacionais da rede. Portanto, o rigor arquitetural exige a adoção de topologias estritamente causais e eficientes, a exemplo de LSTMs, redes lineares e *ensembles*, que preservam a integridade cronológica dos dados com baixo custo computacional, garantindo a agilidade necessária para a reconfiguração dinâmica do sistema.

3 Arquiteturas Preditivas de Baixa Latência

As limitações estruturais intrínsecas ao mecanismo de autoatenção impulsionam uma forte transição na literatura moderna rumo a topologias estritamente livres de atenção (*attention-free*) [24] [25]. Conforme explorado por Z. Chong [15], essa quebra de paradigma fundamenta-se na necessidade de abandonar a densa matriz de interações globais. O objetivo central dessa fuga arquitetural é escapar do custo computacional elevado gerado pelo levantamento tensorial no espaço de características gerado pelas interações pareadas de longo alcance. Sob o rigor analítico, a formulação clássica do *Transformer* impõe uma complexidade quadrática de tempo e de alocação de memória. O cômputo pareado da matriz de atenção escalona a sua sobrecarga matemática na ordem exata de $O(L^2.d)$ em relação ao comprimento da sequência L. Consequentemente, essa intratabilidade algorítmica torna a arquitetura inviável para o monitoramento contínuo e em tempo real exigido pelo núcleo das redes 5G.

Diante dessa inviabilidade prática no plano de controle, a literatura do estado da arte mostra

mudanças cruciais no paradigma preditivo. Segundo A. Zheng et al. [7] abordagens extremamente mais simples, baseadas puramente em regressão linear, são capazes de superar os pesados modelos de atenção em métricas de precisão, fornecendo, além de latência de inferência drasticamente inferior, um menor consumo de memória do sistema. Essa superioridade preditiva desses estimadores causais decorre do tratamento matricial das sequências de entrada. Ao contrário das redes baseadas em atenção, cujos algoritmos distribuem pesos de correlação global de forma estruturalmente invariante à permutação, os preditores de base linear não suprimem e não destroem a ordem cronológica dos eventos [7]. Essa preservação natural da direcionalidade mitiga a perda irrecuperável de informações temporais durante a agregação matemática dos dados, consequentemente evitando o colapso estrutural das sequências iterativas. Tal robustez torna esses modelos as soluções analíticas ideais para satisfazer aos rigorosos orçamentos de baixa latência e aos SLAs críticos de precisão impostos pelo fatiamento de rede no ecossistema 5G.

Com o intuito de apresentar o ferramental de mitigação desse gargalo de engenharia, detalharemos adiante o núcleo de funcionamento de quatro arquiteturas leves. Nas subseções subsequentes, explorar-se-á sequencialmente a robusta decomposição estrutural provida pela arquitetura DLinear, a capacidade causal de retenção de memória executada pelas LSTMs, a rápida extração de características espaço-temporais das CNNs 1D, culminando na demonstração da estabilidade e precisão garantidas pelas abordagens matemáticas híbridas via *Ensembles*.

3.1 DLinear e a Decomposição Estrutural

A arquitetura preditiva DLinear materializa a transição metodológica do estado da arte rumo a soluções mais simples e livres de atenção [7]. Em contraposição à massiva complexidade matemática e tensorial dos *Transformers*, este modelo fundamenta o seu mapeamento preditivo na decomposição estrutural, uma clássica técnica de análise de séries temporais. Para processar a sequência histórica de entrada, a topologia aplica algoritmicamente um filtro de médias móveis de forma direta sobre os dados brutos de telemetria. Essa rigorosa operação de filtragem extrai a trajetória contínua da série e separa a informação em dois componentes operacionais distintos: a Tendência (*Trend*) global e o Resíduo sazonal (*Remainder*), que isola as flutuações e periodicidades intrínsecas ao fluxo contínuo de tráfego.

Uma vez estabelecida a separação dos tensores, a mecânica de inferência da arquitetura atua de maneira incisiva, estritamente paralela e algébrica. O modelo descarta o conceito de aprendizado em redes ocultas profundas e aloca duas redes lineares diretas de camada única, designando um processador linear autônomo para cada componente do sinal extraído. O cálculo das predições ocorre através de uma operação pontual de soma ponderada, na qual a matriz de pesos da regressão projeta o horizonte passado diretamente para o futuro esperado. Ao final desse processo, o algoritmo simplesmente soma os resultados preditivos das duas camadas isoladas, gerando a resposta consolidada para a predição temporal.

Sob a ótica operacional do provisionamento de recursos no 5G, a regressão para essa abordagem estritamente arquitetônica revela vantagens insubstituíveis de engenharia. Desprovido por completo do massivo levantamento computacional imposto pelo mecanismo de autoatenção pareado, o algoritmo atinge o auge da agilidade analítica, processando lotes de dados em uma complexidade temporal estritamente linear. A eliminação das operações de correlação global encolhe o consumo de memória da rede a níveis infinitesimais [7] quando escalonado contra modelos profundos, entregando velocidades de inferência excepcionalmente rápidas. Em cenários práticos, essa performance enxuta se adequa com exatidão aos rigorosos orçamentos algorítmicos e à latência ultrabaixa requisitada pelas instâncias dinâmicas de fatiamento de rede.

Para além das validações teóricas, a superioridade estatística da arquitetura DLinear pode ser

comprovada empiricamente em diversas instâncias de alta estocasticidade. Conforme documentado no desafio *Baidu KDD Cup 2022* [16], essa topologia foi submetida a cenários de missão crítica preditiva no escopo de geração de energia eólica, cuja matriz é caracterizada por altas variabilidade e imprevisibilidade natural de fornecimento. Nesse domínio competitivo, o modelo linear superou com larga margem os estimadores globais baseados em atenção, mostrando sua alta eficácia em generalizar comportamentos voláteis, como aqueles esperados do tráfego instanciado nas fatias de redes 5G.

Apesar do expressivo ganho de responsividade e de processamento computacional, a formulação matemática estritamente linear insere inevitáveis limitações representacionais no núcleo do modelo. Uma vez que as inferências baseiam-se em projeções puramente espaciais de camada única, a arquitetura linear denota considerável dificuldade estatística perante perturbações caóticas. Especificamente, o preditor demonstra menor destreza caso a série de dados sofra inversões abruptas ou mudanças não-lineares curtas e complexas de curtíssima duração. Essa restrição teórica perante picos e vales agressivos corrobora o imperativo de mapear topologias subsequentes. Torna-se vital avaliar arquiteturas dotadas da capacidade de reter memória temporal contínua e processar fortes flutuações, sem comprometer o paralelismo veloz exigido para o núcleo celular móvel.

3.2 *Long Short-Term Memory* (LSTM) e a Retenção de Ordem Temporal

A arquitetura *Long Short-Term Memory* (LSTM) consolidou-se na literatura científica como uma das topologias de base mais robustas para a modelagem de dados estritamente cronológicos. Conforme evidenciado no estado da arte referente ao tratamento analítico de séries temporais, essa rede provou a sua elevada eficácia em capturar dependências de longo prazo em domínios sequenciais variados [10]. Historicamente, o sucesso do estimador foi inicialmente alavancado pelas suas exímias precisões em instâncias de alta complexidade, tais como os domínios de reconhecimento de fala e processamento para a geração de texto.

A formulação matricial dessa rede foi introduzida na literatura pelo trabalho de Hochreiter e Schmidhuber [17], em 1997, em um projeto arquitetural que objetivava solucionar a dissipação e explosão do gradiente, fenômenos matemáticos que afligiam irremediavelmente as redes recorrentes clássicas durante a etapa de otimização de longas sequências devido à multiplicação sucessiva das derivadas da função de perda para o processamento iterativo. Posteriormente, o rigor analítico do modelo foi significativamente elevado em 1999, mediante a contribuição de Gers et al. [18], com a introdução da porta de esquecimento na matriz de processamento. A incorporação desse filtro algorítmico dotou a rede da capacidade de descartar proativamente dados temporais obsoletos, otimizando a retenção contínua e impedindo a saturação de memória.

Matematicamente, a integridade da informação ao longo do vetor histórico é mantida intacta pelo contínuo estado de célula. O fluxo informacional em seu interior é rigorosamente regulado por portas não lineares atuando em série. A porta de esquecimento, por sua vez, atua filtrando algoritmicamente a memória da rede, gerando uma escala exata de zero a um que dita qual fração temporal da iteração anterior será eliminada e qual será propagada adiante.

Simultaneamente à filtragem algorítmica do estado anterior, a álgebra celular prepara a assimilação contínua dos novos eventos de telemetria da rede. Paralelamente, a porta de entrada opera quantificando a magnitude da informação no instante atual que deve ser agregada ao sistema. Esse cruzamento de matrizes resulta em uma operação puramente aditiva no estado de célula, fundindo a memória causal prévia com o novo dado observado na sequência [19].

Sob a ótica teórica da modelagem sequencial profunda, essa topologia consegue contrapor as problemáticas matemáticas inerentes aos *Transformers*. A arquitetura baseada no mecanismo de autoatenção atua mapeando o espaço de dados através do pareamento cruzado e processamento

global das instâncias, perdendo a direcionalidade intrínseca da série. Em contrapartida, as redes LSTM processam o histórico temporal operando de forma estritamente causal e isoladamente direcionada.

Devido à sua mecânica estrutural baseada em retroalimentação de estado cíclica e progressiva, a arquitetura LSTM retém a ordem temporal de forma natural ao absorver a sequência de maneira inerente ao seu processamento passo a passo. Isso elimina completamente a necessidade de injeções artificiais de PE na base dos dados, permitindo que a direcionalidade da série deixe de ser uma abstração espacial forçada e se estabeleça como a base de propagação do sinal preditivo dentro do caminho da rede.

Apesar da comprovada precisão em reter a exata causalidade temporal, a arquitetura LSTM possui uma limitação algorítmica severa intrínseca à sua topologia. Como sua operação celular é estritamente iterativa e sequencial, o cômputo da rede no instante de tempo atual depende obrigatoriamente da resolução integral do estado oculto prévio. Essa restrição cronológica inviabiliza o paralelismo simultâneo dos dados durante a fase de treinamento, fazendo com que o tempo de convergência computacional do modelo seja ampliado de forma expressiva. No cenário de orquestração do 5G, em que os orçamentos de latência são críticos, essa dependência algorítmica justifica a adoção de abordagens mais velozes, com possibilidade de paralelização.

3.3 CNN 1D para Extração de Características Locais

As Redes Neurais Convolucionais (CNNs), embora originalmente concebidas e consolidadas como a arquitetura dominante em tarefas de visão computacional, demonstraram eficácia expressiva na modelagem estatística de sequências unidimensionais. No escopo preditivo de séries temporais, a literatura do estado da arte evidencia a notória capacidade destas topologias paramétricas na extração de características e no reconhecimento exato de padrões espaciais locais subjacentes aos dados [10]. A sua operação matemática fundamental repousa sobre a camada estrita de convolução (Conv1D), que atua por intermédio de filtros convolucionais deslizantes que varrem a extensão contínua da sequência temporal observada. Tais matrizes processam blocos numéricos restritos a cada passo iterativo, projetando janelas locais diretas em vetores de características latentes. Esse cruzamento de dados insere não-linearidade analítica no mapeamento preditivo, frequentemente otimizado por funções de ativação, extraindo atributos primários da telemetria com extrema eficiência e viabilizando o foco em dependências de curto prazo.

Para o rigoroso gerenciamento de recursos no núcleo 5G, essa robusta capacidade de mapear correlações e características restritas constitui um mecanismo analítico ideal. O tráfego alocado sob a orquestração de fatias de rede é intrinsecamente permeado por demandas operacionais dinâmicas e de alta volatilidade local. Ao aplicar o processamento matricial da camada Conv1D, os filtros conseguem identificar de imediato os picos e vales abruptos inerentes a essas severas flutuações instantâneas de largura de banda.

Esse cenário de necessidade responsiva local rápida explicita uma falha estrutural fundamental das redes baseadas em interações globais contínuas, como é o caso do mecanismo de atenção global do *Transformer* que, por forçar o pareamento cruzado irrestrito e pulverizar excessivamente o seu foco ao longo de toda a extensão temporal, frequentemente falha em capturar anomalias concentradas. Em contraste a esse processamento difuso e invariante à permutação, a convolução unidimensional consegue isolar perfeitamente o padrão espacial da rajada de dados. Esse isolamento dota o sistema de controle de extrema agilidade, retendo a assinatura exata das mudanças repentinas sem o peso do processamento denso.

Apesar da enorme agilidade na extração matemática de atributos perante anomalias estocásticas pontuais, o processamento puramente convolucional apresenta uma severa limitação arquitetural

de base: uma vez que os filtros atuam limitados às fronteiras exatas do seu campo receptivo de observação, o modelo, quando operando de maneira isolada, é incapaz de inferir o contexto histórico global da série, de modo que a topologia carece de uma matriz de memória de longo prazo inerente [20]. Consequentemente, a CNN não possui a capacidade inata de correlacionar eventos temporais sistêmicos que estejam cronologicamente distantes entre si na linha do tempo.

Visando solucionar esse gargalo analítico e mitigar a limitação receptiva das matrizes locais, a literatura contemporânea propõe a fusão topológica de arquiteturas profundas [10]. A integração entre a rápida extração de características locais executada pela convolução e a robusta retenção sequencial inerente às células recorrentes (LSTM) origina um novo paradigma em estabilidade preditiva: as abordagens híbridas CNN-LSTM, frequentemente consolidadas sob a forma de *Ensembles* modulares. Com o propósito de embasar essa superioridade algorítmica, a subseção subsequente detalhará essa convergência matemática, demonstrando como a hibridização consolida a infraestrutura definitiva para a alocação preditiva de fatias 5G sob rigorosa baixa latência.

3.4 Abordagens Híbridas (*Ensembles*)

A otimização matemática da precisão preditiva encontra o seu ápice metodológico nas abordagens híbridas (*Ensembles*). A premissa analítica central dessa topologia consiste em combinar as predições iterativas geradas por múltiplos modelos subjacentes para mitigar as limitações estruturais inerentes a cada arquitetura quando operada de forma isolada. Essa fusão estratégica viabiliza a integração de características informacionais complementares, tornando algoritmicamente possível, por exemplo, unificar a extrema eficácia extratora da rede CNN 1D para características espaciais abruptas com a sólida capacidade causal de retenção da ordem cronológica provida pelas células LSTM. O trunfo fundamental desse encapsulamento reside na habilidade de alcançar métricas de precisão rigorosas sem incorrer no severo custo computacional de ordem quadrática imposto pelos preditores globais baseados em mecanismos de autoatenção.

A superioridade estatística dessa hibridização foi categoricamente validada no escopo do gerenciamento de telecomunicações pelo estudo conduzido por Ferreira et al. [3]. No artigo publicado em 2021, foi proposto um robusto *framework* de previsão distribuído, projetado para operar o cruzamento de algoritmos em tempo real. Essa arquitetura concorrente foi submetida a instâncias reais de telemetria, predizendo a dinâmica de tráfego de fatias operacionais oriundas de uma rede veicular e de uma infraestrutura celular 4G. As avaliações empíricas constataram que a orquestração de modelos *Ensemble*, integrando simultaneamente instâncias de Redes Neurais *Feed-Forward* (FFNN), redes recorrentes (LSTM) e algoritmos de regressão, resultou invariavelmente no mais alto desempenho preditivo. A combinação puramente matemática das saídas dessas diferentes topologias minimizou o erro residual e superou com folga a exatidão alcançada pelos estimadores clássicos e profundos operando solitariamente.

Para o panorama rigoroso do fatiamento de rede no ecossistema 5G, a implementação de estimadores híbridos de altíssima exatidão viabiliza o genuíno gerenciamento autônomo e dinâmico dos recursos virtuais. A precisão consolidada pelos *Ensembles* provém o plano de controle de insumos vitais para antever o comportamento das fatias, permitindo que as instâncias lógicas executem ajustes proativos nos limiares de alocação de largura de banda, redistribuindo capacidades sistêmicas imediatamente antes de os surtos de tráfego ocorrerem. Consequentemente, o sistema previne de forma efetiva o congestionamento local e evita a queda de requisições de usuários. Tais propriedades atestam que as metodologias híbridas formam a espinha dorsal matemática necessária para sustentar a confiabilidade crítica e a extrema baixa latência demandadas pelo provisionamento ininterrupto no núcleo 5G.

Apesar de viabilizar o gerenciamento proativo da rede 5G com alta precisão, a adoção de abor-

dagens *Ensemble* impõe certas limitações operacionais atreladas ao seu alto custo computacional. A execução simultânea de múltiplos modelos preditivos profundos, além de aumentar expressivamente o tempo global exigido para o treinamento do sistema, também exige um consumo substancialmente maior de memória e de processamento durante a etapa contínua de inferência [3]. Somado a essa sobrecarga de *hardware*, a plena eficiência do estimador depende de uma calibragem matemática cuidadosa da função agregadora da camada final, tornando imperativo definir a técnica exata que estipulará como as saídas individuais de cada rede irão compor a resposta consolidada do sistema.

4 Metodologia e Estudo Ablativo

Nesta seção, detalharemos o arcabouço prático desenvolvido a fim de validar as hipóteses formuladas nas seções teóricas preliminares deste trabalho. Os arquivos, *scripts* e *notebook* de treinamento concorrente que dão suporte a este capítulo encontram-se documentados e disponíveis no repositório público do projeto: `https://github.com/HaylaPBV/MC030`.

O cerne experimental deste projeto reside na predição contínua e multivariada do volume de requisições de tráfego associado a três fatias lógicas de rede no padrão 5G: eMBB (*enhanced Mobile Broad Band*), mMTC (*massive Machine Type Communication*) e URLLC (*Ultra Reliable Low Latency Communication*). Esta modelagem configura-se como um problema de previsão de séries temporais de longo prazo (abreviada como LTSF, do inglês *Long-Term Time Series Forecasting*), cujo objetivo é viabilizar o gerenciamento e a alocação proativa de recursos virtuais no plano de controle do núcleo celular móvel. Para consolidar esse propósito de engenharia, promovemos um *benchmarking* computacional entre o modelo do estado da arte baseado em atenção por blocos, o PatchTST, e arquiteturas de aprendizado profundo de menor complexidade, representadas por redes convolucionais unidimensionais (CNN 1D), redes recorrentes (LSTM) e estimadores baseados em decomposição linear (DLinear). Adicionalmente, a comparação engloba o tradicional método estatístico sazonal univariado *Seasonal Autoregressive Integrated Moving Average* (SARIMA) e abordagens de composição híbrida, permitindo uma análise multifacetada dessas tecnologias perante os exigentes SLAs das redes 5G.

O fluxo de trabalho adotado, mapeado graficamente na Figura 1, estrutura-se em seis etapas lógicas e sequenciais projetadas para garantir o rigor metodológico do projeto. O *pipeline* se inicia com a coleta dos *logs* brutos de tráfego 5G, os quais são submetidos a uma etapa de agregação por *slice* para consolidar o volume de requisições horárias nas fatias lógicas da rede. Na sequência, realizamos a divisão cronológica das séries na proporção de 70% para treinamento e 30% para teste, aplicando-se de forma concomitante a normalização *Z-score* parametrizada estritamente com os dados de treino. Esses dados normalizados alimentam o algoritmo de janelamento deslizante (do inglês, *sliding windows*), que gera as matrizes históricas de entrada e os alvos de projeção sob a estratégia de previsão *Direct Multi-Step* (DMS). O processo segue para o treinamento concorrente das arquiteturas, confrontando o modelo PatchTST com os *baselines* neurais, o método estatístico e a abordagem híbrida. Finalmente, o fluxo é concluído na etapa de avaliação e *benchmarking*, na qual as métricas de erro quadrático médio (MSE) e erro absoluto médio (MAE) são calculadas na escala real após a desnormalização das previsões e avaliamos comparativamente os resultados obtidos pelos modelos.

Dentro desse ecossistema comparativo, a arquitetura *Transformer* (especificamente o modelo PatchTST) recebeu um papel de destaque na investigação por se consolidar na literatura contemporânea como o estado da arte para o processamento de sequências temporais longas. Contudo, devido à alta periodicidade diária do tráfego e ao tamanho reduzido do conjunto de dados coletado, modelos de alta capacidade baseados em atenção global são severamente desafiados pelo risco de

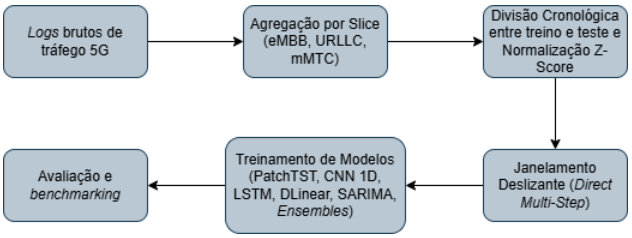

Figura 1: Diagrama representativo do *pipeline* do fluxo de trabalho.

sobreajuste (*overfitting*). Para diagnosticar de forma científica as falhas estruturais dessa arquitetura nesse cenário restrito e avaliar técnicas de mitigação desse problema, executou-se um estudo de ablação sistemático focado exclusivamente no PatchTST.

Esta seção está organizada de modo a descrever sequencialmente as etapas de implementação prática. Nas seções subsequentes, detalharemos os algoritmos e procedimentos de pré-processamento de dados e o janelamento de séries. Em seguida, apresentaremos a parametrização e infraestrutura de codificação das arquiteturas neurais implementadas e, por fim, mapearemos as diretrizes metodológicas e as variações paramétricas estipuladas para a condução do estudo de ablação sistemático do modelo *Transformer*, compondo o panorama empírico no qual as hipóteses de modelagem preditiva serão definitivamente testadas.

4.1 Pré-processamento, Divisão e Normalização dos dados

A acurácia e a validade matemática de modelos de *Deep Learning* voltados ao provisionamento proativo de recursos de rede dependem, sumariamente, do rigor empregado na etapa de pré-processamento e tratamento dos dados. Neste projeto, a modelagem baseia-se nos registros extraídos do *framework* de simulação realista *DeepSlice*, desenvolvido originalmente por Thantharate et al. [2]. O *dataset* original consiste em uma base sintética estruturada contendo mais de 65.000 requisições discretas de plano controle, mapeando o comportamento de conexões e dispositivos na infraestrutura celular móvel. Cada registro bruto é caracterizado por um conjunto heterogêneo de variáveis e Indicadores-Chave de Desempenho (KPIs) de rede e do próprio equipamento de usuário (UE), especificamente: o tipo de caso de uso, a categoria do terminal móvel, a tecnologia de rádio suportada, o dia da semana, a hora da solicitação, o identificador de classe de serviço, além das métricas de confiabilidade e latência.

A utilização desta base de dados impõe, contudo, uma profunda mudança de paradigma metodológico frente à literatura original do *DeepSlice*. Em sua concepção original, o *dataset* fora empregado em uma modelagem para uma tarefa de classificação estática e pontual, cujo objetivo restringia-se a determinar a fatia ideal para um dispositivo previamente desconhecido. O presente trabalho propõe uma abordagem distinta ao reformular o problema como uma tarefa de regressão multivariada de séries temporais de longo prazo, modelando o fluxo dinâmico e concorrente das fatias ao longo de um horizonte contínuo.

A fim de consolidar essa reestruturação cronológica, as transações discretas foram agregadas temporariamente em intervalos regulares de 1 hora. O procedimento de mapeamento processa as variáveis categóricas de tempo: os dias da semana de segunda-feira a domingo são convertidos em inteiros sequenciais no intervalo de 0 a 6, enquanto a hora bruta diária é disposta no intervalo computacional padrão de 0 a 23. A partir dessa correspondência temporal, estabelecemos o índice linear contínuo denominado *hour_index* por meio da aplicação da equação:

$$hour\_index = day\_of\_week * 24 + hour\_of\_day \tag{2}$$

Essa transformação algébrica mapeia um total de 168 *timestamps* sequenciais, cobrindo de modo ininterrupto e cronológico o ciclo completo de uma semana de telemetria, conforme ilustrado na Figura 2.

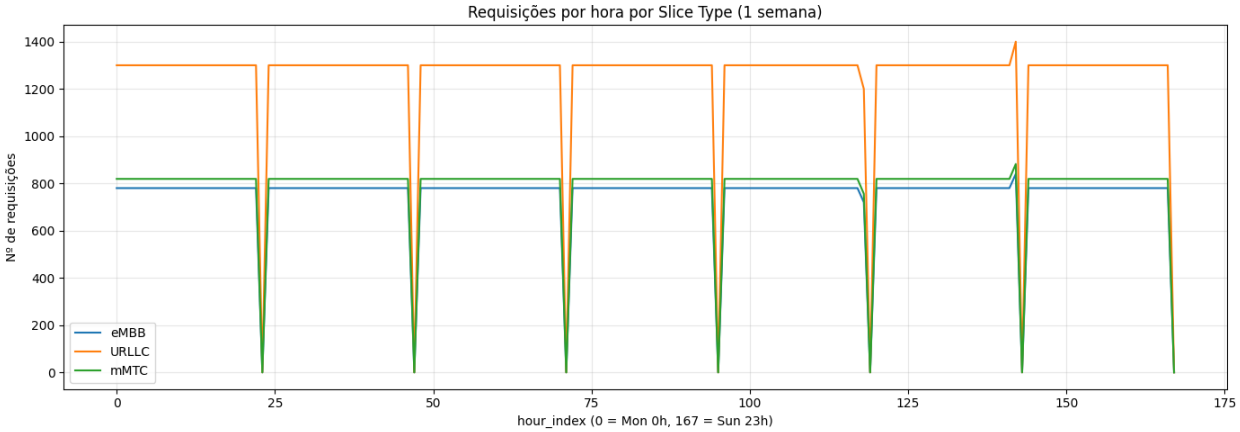

Figura 2: Série temporal agregada do volume de requisições horárias por fatia (eMBB, URLLC e mMTC) ao longo de uma semana.

A consolidação deste *dataframe* agregado foi implementada por meio de manipulações vetorizadas na biblioteca Pandas. Inicialmente, é feito o agrupamento dos dados (`groupby`) sob a chave composta pelas variáveis `hour_index` e `Slice Type (Output)`, aplicando a função de contagem de instâncias (`size`). Posteriormente, a operação de pivotamento (`unstack`) transpõe a variável categórica do tipo de fatia lógica da orientação por linhas para colunas individuais de resposta, parametrizando a substituição de valores ausentes por zero a fim de mitigar possíveis descontinuidades no sinal físico da rede decorrentes de eventuais horas sem registros associados. Finalmente, a série temporal é reconstruída sob um índice de intervalo completo (`RangeIndex`) de 0 a 167, garantindo a existência formal de todas as 168 horas contínuas, conforme exibido pelo excerto de código disposto na Listing 1. Dessa forma a base multivariada resultante passa a ser representada por três variáveis contínuas que mapeiam o volume de requisições por hora correspondentes aos serviços das fatias eMBB, URLLC e mMTC.

Listing 1: Código de agregação horária e pivotamento de fatias lógicas

```
day_map = {
    'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3,
    'Friday': 4, 'Saturday': 5, 'Sunday': 6
}

df['day_of_week'] = df['Day (Input4)'].map(day_map)
df['hour_of_day'] = df['Time (Input 5)'].astype(int) - 1  # passa para 0..23
df['hour_index']  = df['day_of_week'] * 24 + df['hour_of_day']

# Sanity check
assert df['day_of_week'].notna().all(), 'Algum nome de dia nao foi mapeado.'
assert df['hour_of_day'].between(0, 23).all(), 'Hora fora do intervalo 0..23.'

agg = (
    df.groupby(['hour_index', 'Slice Type (Output)'])
```

```
        .size()
        .unstack(fill_value=0)
        .rename_axis(columns=None)
)

full_index = pd.RangeIndex(start=0, stop=7 * 24, name='hour_index')
agg = agg.reindex(full_index, fill_value=0)

for col in ['eMBB', 'URLLC', 'mMTC']:
    if col not in agg.columns:
        agg[col] = 0
agg = agg[['eMBB', 'URLLC', 'mMTC']].astype(np.int64)
```

O conjunto de dados resultante após a agregação é, então, submetido ao processo de partição para treino e teste. Como se trata de uma análise de séries temporais, o particionamento deve obedecer estritamente aos preceitos de causalidade e ordenamento cronológico natural, rejeitando de forma absoluta o embaralhamento aleatório de registros convencional da ciência de dados tradicional. Essa restrição direcional visa reproduzir fielmente o cenário real de monitoramento no núcleo da rede 5G e impedir categoricamente o vazamento de dados, o qual inviabilizaria a generalização do modelo ao permitir que informações do futuro contaminassem indevidamente a otimização paramétrica das redes. Desse modo, adotamos uma proporção de divisão sequencial de 70/30, de modo que o conjunto de treinamento engloba as primeiras 117 horas da semana (*timestamps* 0 a 116), enquanto o conjunto de teste abrange as 51 horas finais (*timestamps* 117 a 167), conforme apresentado pela Figura 3.

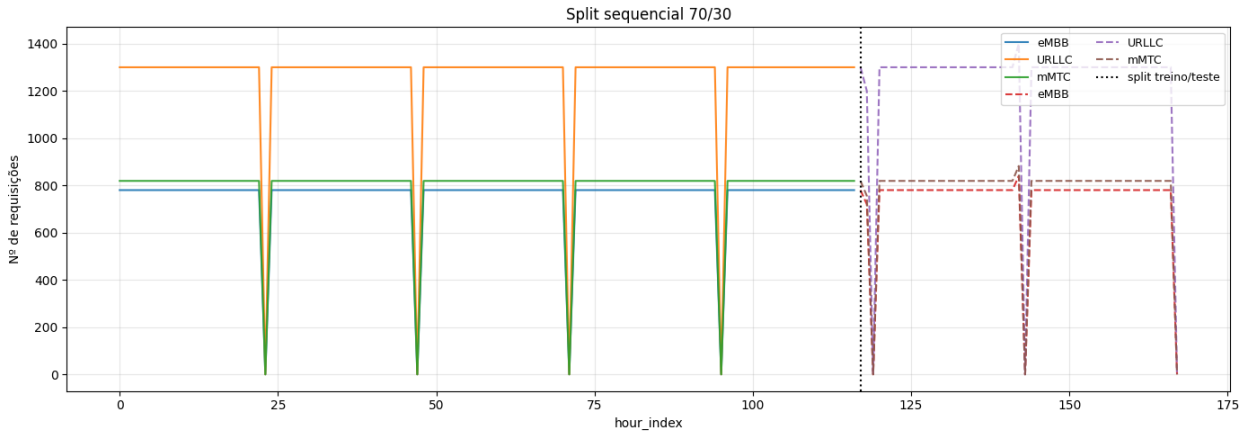

Figura 3: Representação gráfica do *split* sequencial cronológico, demarcando a fronteira de separação entre os conjuntos de treinamento e teste.

Finalmente, visando estabilizar a magnitude dos gradientes e acelerar a convergência analítica das arquiteturas neurais, aplica-se o processo de padronização estatística por escala *Z-score* sobre as séries temporais. Conforme confirmado pela literatura contemporânea [13], esse tratamento é indispensável devido à diferença de amplitude e do volume médio de solicitações registradas em cada fatia de serviço, cujas ordens de grandeza são inerentemente desequilibradas. O rigor metodológico estende-se ao isolamento estatístico empregado no ajuste do escalonador: a média amostral e o desvio padrão de cada canal são calculados de forma exclusiva sobre o conjunto de treinamento de 117 horas. Essa exata escala aprendida é posteriormente aplicada para transformar os subconjuntos de treino e teste. Esse isolamento rigoroso garante que o modelo otimize seus parâmetros e valide suas predições sem que qualquer informação de escala real do futuro exerça influência no sinal

normalizado. Concluído o processo preditivo, as projeções temporais geradas na saída do sistema são convertidas de volta à escala original por intermédio da transformação inversa, permitindo que o cálculo de erros globais e por canal (MSE e MAE) reflita de forma honesta a precisão prática do modelo perante as exigências reais de provisionamento de recursos no núcleo da rede 5G.

4.2 Estratégia de Janelamento Deslizante e DMS

No escopo de LTSFs e suas estratégias de projeção de horizontes, a literatura especializada faz distinção de duas vertentes funcionais: a previsão multi-passo iterativa (do inglês, *Iterated Multi-Step*, abreviado por IMS) e a previsão multi-passo direta (do inglês, *Direct Multi-Step*, abreviado por DMS). O método IMS baseia-se na otimização de um estimador de passo unitário simples que, durante a etapa de inferência, é realimentado de forma recursiva com as suas próprias saídas para construir projeções em horizontes mais longos. Conforme demonstrado por Zeng et al. [7] e sustentado por análises clássicas de decomposição estatística, porém, a autoregressão recursiva intrínseca ao IMS sofre de um severo acúmulo exponencial de erros de projeção, resultando em um acentuado desvio de tendência e na degradação rápida da variância amostral, o que inviabiliza a aplicação de estratégias recursivas clássicas em redes celulares de missão crítica submetidas a rígidos limites de latência e confiabilidade.

Tendo em vista esse comportamento recursivo de degradação, o presente projeto adotou estritamente a estratégia DMS, na qual o mapeamento do histórico passado para o horizonte de previsão futuro ocorre em um único passo de inferência. Sob a ótica teórica discutida por Zeng et al. [7], os processos autoregressivos de inferência iterativa profunda em arquiteturas baseadas em atenção frequentemente convergem de forma prematura em direção à média histórica incondicional. Ao dispensar realimentações analíticas intermediárias, a projeção DMS evita essa propagação de erros sistemáticos pelo *pipeline*. Essa escolha metodológica também é capaz de assegurar velocidades de inferência compatíveis com orçamentos de tempo de execução restritos no núcleo 5G, uma vez que transferem a complexidade do modelo para a projeção espacial direta de uma camada de saída robusta.

Sob o aspecto formal da teoria de séries temporais, define-se o sinal multivariado contínuo de entrada como a matriz de observações $S = \{s_1, s_2, ..., s_T\}^T \in \mathbb{R}^{T \times C}$. Nesse espaço tensorial, a dimensão cronológica é delimitada por T = 168 horas, equivalendo ao ciclo amostral completo de uma semana de telemetria horária, ao passo que C = 3 corresponde ao número de canais de variáveis contínuas monitoradas (eMBB, URLLC e mMTC). O algoritmo de janelamento deslizante opera reestruturando sequencialmente a matriz S para extrair amostras de treinamento e validação a cada passo de tempo t. Assim, a matriz de histórico de entrada é representada pelo tensor de características $X_t = [s_{t-L+1}, ..., s_t]^T \in \mathbb{R}^{L \times C}$, parametrizado por uma janela de observação retroativa de tamanho L. Paralelamente, o alvo supervisionado correspondente ao horizonte futuro de projeção é definido pela matriz de rótulos $Y_t = [s_{t+1}, ..., s_{t+H}]^T \in \mathbb{R}^{H \times C}$, onde H denota o horizonte preditivo projetado.

A parametrização deste algoritmo de janelamento deslizante foi estabelecida pela função `Make_Sliding_Windows`, adotando uma janela de histórico de L = 48 horas sob passo de deslizamento (`stride`) unitário igual a um e, em conformidade com os *benchmarks* clássicos de LTSF, definindo dois cenários distintos de horizontes preditivos: o Teste 1, com H = 12 horas, e o Teste 2, com H = 24 horas, conforme exibido no Listing 2. Embora a literatura relacionada a LTSF adote contextos de entrada mais extensos, como um *look-back* de L = 96 horas, a imposição do limite de L = 48 horas neste trabalho foi necessária devido às severas restrições dimensionais inerentes ao conjunto de dados usado. Como o fluxo de telemetria coletado compreende um total de apenas 168 horas, a adoção de janelas de observação retroativa excessivamente longas provocaria um

encolhimento amostral drástico durante a reestruturação dos tensores: no cenário do Teste 2, por exemplo, a utilização de um *look-back* de L = 96 horas restringiria o conjunto de treinamento a $117 - 96 - 24 + 1 = -2$ amostras, impossibilitando a tarefa.

Listing 2: Código de janelamento deslizante para DMS

```python
def make_sliding_windows(series: np.ndarray, look_back: int, horizon: int,
    stride: int = 1):
    """
    Gera janelas deslizantes para (DMS).

    Parametros
    ----------
    series : np.ndarray, shape (T, C)
        Serie temporal ja normalizada. T = num. de passos, C = num. de variaveis.
    look_back : int
        Tamanho da janela de historico (L).
    horizon : int
        Tamanho do horizonte de previsao (H).
    stride : int, default=1
        Passo entre janelas consecutivas.

    Retorna
    -------
    X : np.ndarray, shape (N, L, C) - historicos.
    Y : np.ndarray, shape (N, H, C) - futuros (alvos).
    """
    if series.ndim != 2:
        raise ValueError(f'series deve ter shape (T, C); recebido {series.shape}')
    T, C = series.shape
    max_start = T - look_back - horizon + 1
    if max_start <= 0:
        raise ValueError(
            f'Serie muito curta: T={T}, look_back={look_back}, horizon={horizon} '
            f'-> nenhuma janela possivel.'
        )

    starts = np.arange(0, max_start, stride)
    N = len(starts)
    X = np.empty((N, look_back, C), dtype=np.float32)
    Y = np.empty((N, horizon,   C), dtype=np.float32)
    for i, s in enumerate(starts):
        X[i] = series[s : s + look_back]
        Y[i] = series[s + look_back : s + look_back + horizon]
    return X, Y
# Hiperparametros globais
LOOK_BACK  = 48
STRIDE     = 1
BATCH_SIZE = 32
HORIZONS   = [12, 24]   # Teste 1 e Teste 2
```

Não obstante a consistência matemática do janelamento, a segmentação sequencial da base de dados sob a proporção 70/30 impõe um desafio de interface física na borda do conjunto de teste. Como o conjunto de treinamento engloba cronologicamente as 117 horas iniciais da série temporal, caso a formação de janelas deslizantes fosse executada estritamente com as observações compreendidas no interior da partição de teste, a geração da primeira amostra seria inviabilizada. Visto que o modelo preditivo requer obrigatoriamente um histórico de L = 48 horas de contexto prévio para projetar o primeiro alvo de teste situado exatamente no *timestamp* 117, a ausência de

dados pretéritos anularia a formação das primeiras 48 janelas de teste, comprometendo severamente a base amostral do *benchmarking* comparativo.

Para mitigar esse encolhimento amostral sem violar a causalidade cronológica do experimento, o *pipeline* de dados executa um procedimento de concatenação prévia na partição de teste, anexando de forma direta as últimas L = 48 horas do conjunto de treinamento ao início do vetor de teste antes de realizar o deslizamento da janela de observação. Esse procedimento, do ponto de vista estatístico, é imune à ocorrência de *data leakage*, uma vez que os dados herdados do conjunto de treino atuam única e exclusivamente na composição da matriz de características de histórico (X), nunca integrando o conjunto de alvos supervisionados futuros (Y) avaliados pelo *benchmarking*. Os rótulos preditos pelo sistema permanecem estritamente confinados à janela compreendida entre os *timestamps* 117 e 167, assegurando a validade estatística das métricas de erro calculadas na escala real.

4.3 Parametrização das Arquiteturas *Baselines*

Esta seção descreve a modelagem matemática, a topologia estrutural e os hiperparâmetros de calibração estabelecidos para as arquiteturas que compõem o ecossistema de *benchmarking* deste projeto. Para avaliar de forma rigorosa se a alta complexidade computacional das técnicas baseadas em atenção global traduz-se em ganhos preditivos reais no plano de controle móvel, o PatchTST é confrontado com estimadores convolucionais, recorrentes, lineares e estatísticos de menor sobrecarga de processamento.

Para garantir a integridade científica e a comparabilidade direta dos resultados, todos os estimadores neurais foram submetidos a um protocolo experimental unificado, isto é, compartilhando as mesmas partições cronológicas de dados, o mesmo escalonador estatístico e são expostos exatamente aos mesmos cenários de janelamento deslizante sob horizontes de previsão de 12 e 24 horas. Nas subseções subsequentes, detalharemos as equações de transição de estado, as especificidades de implementação do PyTorch para o tratamento multidimensional de fatias de rede 5G e as rotinas de otimização adotadas para cada classe de modelo preditivo.

4.3.1 Protocolo Geral de Treinamento

Visando garantir a confiabilidade das análises comparativas das redes neurais profundas, estabelecemos uma padronização do ambiente de execução e das diretrizes de otimização paramétrica.

Com o propósito de isolar o impacto puramente arquitetural de cada topologia preditiva avaliada, é necessário assegurar que as variações observadas nas métricas de erro (especificamente MSE e MAE) sejam decorrência exclusiva do *design* e da organização interna das matrizes de cada modelo. Sob essa perspectiva, buscamos o isolamento de variáveis ambientais a fim de eliminar vieses sistemáticos, de modo que todas as arquiteturas investigadas neste trabalho compartilham o mesmo ciclo de processamento, operam sobre as mesmas janelas deslizantes geradas na etapa de pré-processamento e realizam a convergência sob condições equivalentes de suporte computacional.

Para operacionalizar a estabilização e garantir a convergência uniforme dos pesos sinápticos sob condições homogêneas, adotamos o otimizador Adam como motor unificado de atualização dos gradientes de todas as redes. O algoritmo opera sob uma taxa de aprendizado constante de $\mathrm{LR} = 10^{-3}$, atuando como um ponto de equilíbrio estatístico que viabiliza passos de gradiente de magnitude estável sem induzir à divergência ou à estagnação em mínimos locais. A avaliação da qualidade preditiva instantânea ao longo das iterações é computada por meio de uma função de perda baseada no Erro Quadrático Médio, instanciada pelo operador `MSELoss` do PyTorch. Paralelamente, a garantia de reprodutibilidade exata das condições de simulação requer a fixação

estrita da semente de inicialização pseudoaleatória, definida como `SEED` = 42. Essa determinação afeta diretamente as bibliotecas utilitárias de simulação, garantindo a geração idêntica das partições e a ordenação reprodutível dos pacotes de mini-lotes estruturados pelo `DataLoader` de treinamento, cujo tamanho é fixado sob o parâmetro `BATCH_SIZE` = 32.

O fluxo computacional de convergência ocorre por meio da invocação da rotina genérica de treinamento denominada `train_model`, desenvolvida para operar de forma a aceitar qualquer módulo descendente da classe `nn.Module` do PyTorch que atenda ao formato de mapeamento tridimensional de entrada e saída (B,L,C) $\rightarrow$ (B,H,C). Ao longo de um limite estabelecido de 100 épocas de treinamento, o algoritmo processa de forma iterativa os lotes de dados de entrada xb e seus respectivos alvos reais yb. A cada passo interno de mini-lote, o otimizador zera os gradientes acumulados em passos anteriores e, em seguida, o estimador executa o *forward pass*, mapeando o tensor histórico de entrada para gerar a projeção direta do horizonte preditivo de forma puramente paralela. Após quantificar o desvio estatístico do sinal predito frente ao alvo real via função de perda, o fluxo computacional invoca o método de retropropagação, que calcula analiticamente os gradientes das derivadas parciais do erro em relação a cada parâmetro ajustável da rede, permitindo que a chamada subsequente do método `optim.step()` atualize os tensores de pesos sinápticos do modelo. Toda essa orquestração unificada de otimização foi resumida na Tabela 1.

Hiperparâmetro	**Configuração Adotada**	**Descrição**
Otimizador	Adam	Gradiente estocástico com momentos adaptativos para aceleração de convergência
Taxa de Aprendizado	10^{-3}	Magnitude padrão estável para evitar oscilações divergentes nos gradientes
Função de Perda	MSE Loss	Erro Quadrático Médio, penalizando de forma quadrática erros de maior magnitude
Random Seed	42	Fixação de inicializações de pesos de camadas e embaralhamento do *DataLoader*
Limite de Épocas	100	Tempo limite para convergência das curvas de erro sem induzir *overfitting*
Batch Size	32	Agrupamento de janelas deslizantes para processamento em GPU/CPU

Tabela 1: Protocolo de treinamento unificado das arquiteturas neurais comparativas no *benchmarking*

4.3.2 Rede Convolucional Unidimensional (CNN 1D)

Para garantir uma simetria metodológica rigorosa com o modelo de alta capacidade PatchTST, a CNN 1D foi projetada sob a premissa de independência de canais. Esse paradigma de modelagem estabelece que cada série temporal de tráfego associada às fatias lógicas eMBB, URLLC e mMTC é processada de maneira estritamente univariada e isolada, compartilhando de forma homogênea os mesmos pesos lógicos dos filtros convolucionais ao longo de todas as etapas de aprendizado. Sob a perspectiva da engenharia de redes neurais, essa estratégia reduz drasticamente o número de parâmetros livres do estimador, o que atenua os riscos de *overfitting* em conjuntos de dados de extensão amostral restrita, além de atenuar a demanda computacional de processamento e memória, otimizando de forma significativa a eficiência do *hardware* para viabilizar inferências rápidas em cenários periféricos junto às estações rádio base do sistema celular móvel.

A formulação matemática e o mapeamento dos fluxos de tensores da CNN 1D foram arquitetados para permitir máximo aproveitamento das capacidades de paralelização vetorizada na biblioteca PyTorch. O tensor tridimensional de entrada, originalmente estruturado sob a dimensão de lote (B,L,C), em que B representa o tamanho do lote, L = 48 é a janela de histórico e C = 3 indica a quantidade de fatias lógicas concorrentes, sofre uma reordenação inicial. Por intermédio de operações coordenadas, o sinal é transposto e redefinido para a dimensão (B×C, 1, L), o que permite que os canais de rede empilhados passem de maneira paralela e idêntica pelas camadas convolucionais unificadas.

Na sequência, o tensor univariado atravessa duas camadas convolucionais sequenciais parametrizadas 32 canais ocultos, filtros locais com `kernel_size` = 3 e preenchimento de bordas idêntico. Essa configuração assegura que a extensão temporal do histórico de rede seja perfeitamente preservada ao longo do *pipeline* de convoluções. Cada operação de filtragem espacial é imediatamente intermediada por funções de ativação linear retificada `ReLU` e camadas de regularização por descarte configuradas sob uma taxa de `dropout` = 0,1.

Após a extração de atributos locais ao longo do *pipeline* convolucional, o tensor resultante, de formato (B×C, 32, L), é achatado para a forma bidimensional (B×C, 32×L). Essa representação vetorial compacta é direcionada a uma camada totalmente conectada cuja cabeça linear realiza a projeção DMS, estimando simultaneamente e de forma paralela todos os valores para o horizonte de previsão futuro $H \in 12, 24$ sem a necessidade de realimentação iterativa de dados.

4.3.3 Rede Recorrente de Memória de Longo Prazo (LSTM)

No âmbito experimental deste trabalho, a rede de Memória de Longo Prazo foi implementada para atuar como o principal *baseline* de modelagem sequencial profunda sob o paradigma multivariado conjunto. Diferente da rede convolucional CNN 1D e do modelo PatchTST, ambos projetados sob a hipótese de independência de canais, a LSTM adota a premissa de dependência cruzada entre as fatias lógicas, de modo que as séries temporais correspondentes às demandas de cada fatia de rede sejam alimentadas e processadas de forma simultânea e integrada. Essa abordagem visa extrair e aprender as correlações estatísticas inter-canal dinâmicas que governam o ecossistema 5G, uma vez que as flutuações e picos de demanda em uma fatia frequentemente correlacionam-se com o comportamento de tráfego das demais devido ao compartilhamento da mesma infraestrutura física de *hardware* de transporte e rádio acesso.

A topologia da rede foi configurada em PyTorch por meio do empilhamento de duas camadas consecutivas da classe `nn.LSTM`. O fluxo de propagação dos tensores opera com uma dimensão de representação oculta fixada em 64 e uma taxa de descarte estocástico definida como `dropout` = 0,1 para regularizar o aprendizado e atenuar o *overfitting* dos parâmetros sinápticos durante a convergência. Durante a execução da passagem direta, o tensor tridimensional de entrada (B,L,C) é submetido ao processamento iterativo temporal pelas células LSTM, gerando um tensor intermediário tridimensional de dimensões (B,L,64), contendo as representações latentes acumuladas passo a passo pela recorrência.

Para realizar a projeção DMS e atender ao requisito de baixa latência exigido pelo plano de controle celular, a arquitetura extrai unicamente o estado oculto final do topo da pilha de recorrência no último instante temporal L. Matematicamente, essa operação de fatiamento de tensor isola a última fatia temporal através do índice negativo no PyTorch, produzindo o vetor de estado oculto consolidado h_L sob a dimensão bidimensional (B,64). Esse vetor denso, que sintetiza a memória e as dependências cronológicas observadas na janela histórica de 48 horas, serve como representação comprimida e contextualizada do estado instantâneo do núcleo de rede 5G.

Na etapa de projeção final, o vetor de estado oculto de tamanho (B,64) alimenta uma camada

linear totalmente conectada, cuja cabeça projetora mapeia diretamente a representação oculta de 64 dimensões para um espaço plano correspondente ao produto das dimensões do horizonte futuro e do número de canais, isto é, H×C. Sob esse esquema, todas as predições associadas aos horizontes de simulação estabelecidos para os testes 1 e 2 são calculadas de maneira puramente paralela e simultânea, descartando etapas recursivas de decodificação que degradariam a precisão preditiva. Por fim, o tensor bidimensional resultante (B,H×C) é submetido a uma reordenação geométrica, restabelecendo a estrutura tridimensional original de formato (B,H,C) no plano de controle móvel para viabilizar o *benchmarking* direto com as demais arquiteturas.

4.3.4 Modelo Linear de Decomposição (DLinear)

No contexto deste trabalho, o DLinear foi estruturado para atuar como um *baseline* baseado em projeções puramente espaciais de camada única, empregando o conceito de decomposição temporal aditiva. O sinal histórico de telemetria de rede móvel, representado pelo tensor tridimensional de entrada $X_t \in \mathbb{R}^{B\times L\times C}$, em que B é o *batch size*, L = 48 horas é o *look-back* e C = 3 indica as fatias de serviço concorrentes, é inicialmente submetido a uma divisão estrutural. Para extrair a componente de tendência-ciclo, $X_t^{Trend} \in \mathbb{R}^{B\times L\times C}$, aplicamos um filtro de médias móveis local com tamanho de janela igual a 25 e realizamos um preenchimento simétrico por replicação dos extremos temporais com tamanho igual a 12, a fim de mitigar a perda de resolução temporal e evitar o truncamento das bordas das sequências. O sinal expandido é processado de forma vetorizada no PyTorch, que recalcula as médias horárias locais e restabelece a dimensão temporal original de 48 passos. A componente complementar residual sazonal, $X_t^{Seasonal} \in \mathbb{R}^{B\times L\times C}$ é obtido de forma determinística por meio da subtração algébrica direta expressa por $X_t^{Seasonal} = X_t - X_t^{Trend}$.

Para viabilizar a projeção preditiva sem induzir o acoplamento espacial de atributos entre as diferentes fatias lógicas, as componentes de tendência e resíduo sazonal são mapeadas de acordo com o paradigma de independência de canais, segundo o qual as variáveis correspondentes às demandas das fatias compartilham o mesmo conjunto de parâmetros ajustáveis nas camadas de regressão linear ao longo de todas as iterações. Para operacionalizar essa manipulação vetorizada, os tensores tridimensionais intermediários de tendência e sazonalidade são transpostos e passam a assumir a orientação geométrica (B,C,L). Essa reordenação de memória expõe a dimensão temporal ativa de 48 passos como o vetor de características de entrada para duas subcamadas totalmente conectadas dedicadas, denominadas `self.linear_trend` e `self.linear_remainder`, instanciadas sob o módulo de projeção `nn.Linear`.

As subcamadas lineares operam de maneira paralela e cega ao longo da dimensão de histórico L=48 para realizar a previsão DMS, gerando diretamente as projeções temporais futuras para todos os instantes do horizonte preditivo de tamanho $H \in 12, 24$ sem a introdução de decodificadores autorregressivos. Uma vez calculadas de forma isolada as representações futuras para cada canal, as matrizes de projeção resultantes de tendência e sazonalidade, de geometria de saída (B,C,H), são sintetizadas de forma aditiva por meio de uma soma direta expressa por $\hat{Y} = \hat{Y}^{Trend} + \hat{Y}^{Trend}$. Finalmente, para assegurar a conformidade física com o ecossistema do *benchmarking*, o tensor de predições consolidado passa por uma transposição final, retornando ao espaço de representação tridimensional original (B,H,C) no plano de controle do núcleo celular móvel, onde cada canal de tráfego projetado pode ser devidamente avaliado perante as restrições operacionais da rede 5G.

4.3.5 Modelo Estatístico Sazonal (SARIMA)

Os experimentos iniciais deste projeto utilizaram o ARIMA, um modelo autorregressivo integrado de médias móveis convencional, processado de maneira estritamente univariada para cada canal de

telemetria. Contudo, tendo em vista a severa dificuldade que o modelo demonstrou em capturar as oscilações periódicas e sistemáticas de tráfego de rede das fatias, uma vez que o sinal físico de requisições de plano de controle apresenta um comportamento sazonal rígido e quase perfeito de 24 horas, aprimoramos o estimador linear clássico para sua variante sazonal, o SARIMA. Dessa forma, conseguimos salvaguardar a robustez metodológica do *benchmarking*, evitando a injusta comparação de arquiteturas neurais profundas contra um referencial estatístico sabidamente fragilizado.

O processo de sintonização fina dos hiperparâmetros estruturais do estimador sazonal foi conduzido de forma automática e independente para cada canal lógico de tráfego. Essa otimização de ordens foi executada uma única vez sobre a partição de treinamento normalizada, configurando o período sazonal do ciclo diário para m = 24 passos horários. Para contornar a não-estacionariedade inerente à oscilação periódica de tráfego celular e capturar de maneira adequada as trajetórias de longo prazo, impôs-se uma diferenciação sazonal obrigatória de primeira ordem representada pela variável D=1. Sob esse arcabouço metodológico e mediante a minimização estrita do critério de informação de Akaike (AIC), um estimador do erro de previsão, o algoritmo exploratório selecionou, de maneira unânime para os três canais lógicos analisados as ordens simplificadas e idênticas definidas pelos parâmetros `order` = (0,0,0) e `seasonal_order` = (0,1,0,24).

Sob a perspectiva da teoria de modelagem estocástica, a convergência unânime para a ordem simplificada $(0, 0, 0)(0, 1, 0)_{24}$ revela propriedades matemáticas fundamentais do conjunto de dados sintéticos utilizado. Ao anular por completo as componentes autorregressivas e de médias móveis, limitando-se estritamente à diferenciação sazonal de primeira ordem, o estimador assume o comportamento de um passeio aleatório sazonal puro, comprovando a rigidez e a previsibilidade quase determinística do sinal de tráfego gerado para as fatias lógicas: diferente de redes celulares reais, cujas flutuações de tráfego são governadas por padrões estocásticos altamente caóticos e ruídos de rajadas imprevisíveis, o ambiente simulado do *DeepSlice* [2] apresenta uma repetição quase cíclica de comportamento temporal, o que explica a alta eficácia preditiva de modelos de projeção puramente sazonais frente a arquiteturas de aprendizado profundo excessivamente complexas e propensas ao *overfitting*.

Para avaliar o desempenho preditivo do SARIMA de forma equivalente aos estimadores de aprendizado profundo, estabeleceu-se um protocolo de teste por janelas deslizantes que simula um monitoramento de rede contínuo. A cada passo temporal, o algoritmo isola o histórico de L = 48 horas de cada canal de tráfego univariado. Em seguida, os coeficientes do modelo são reestimados via máxima verossimilhança utilizando a ordem $(0, 0, 0)(0, 1, 0)_{24}$ previamente identificada, disparando a projeção direta para o horizonte futuro H.

Adicionalmente, visando mitigar gargalos operacionais durante a execução das avaliações iterativas, o *pipeline* do SARIMA incorpora um mecanismo de intolerância a falhas. Devido à rigidez cíclica e à presença de intervalos de tráfego constante no *dataset*, o processo de convergência numérica do ajuste das estimativas de máxima verossimilhança do modelo pode sofrer com instabilidades matemáticas associadas à singularidade da matriz de covariância. Para evitar interrupções abruptas na execução do *benchmarking*, implementou-se um tratamento estruturado de exceções: diante de qualquer falha de convergência ou indefinição numérica do solver analítico, o algoritmo realiza uma rotina de *fallback*, replicando o último valor conhecido da janela de histórico por todo o horizonte preditivo H, garantindo a completude e a continuidade estrita das avaliações comparativas sem comprometer a integridade estatística do *pipeline*. O fluxo integrado de otimização de ordens clássicas por canal e inferência univariada com reajuste local de parâmetros encontra-se detalhado na Listing 3.

Listing 3: Pipeline de sintonização do SARIMA via auto_arima e algoritmo de inferência dinâmica com reajuste local no teste

```python
# 1) Selecao de ordem SARIMA (p,d,q)(P,D,Q,m) por canal usando auto_arima no
    treino
SARIMA_ORDERS = {}
for c, feat in enumerate(FEATURES):
    with warnings.catch_warnings():
        warnings.simplefilter('ignore')
        auto = pm.auto_arima(
            train_scaled[:, c],
            seasonal=True,
            m=24,                       # periodo sazonal: 24h (ciclo diario)
            D=1,                        # diferenciacao sazonal forcada
            stepwise=True,
            suppress_warnings=True,
            error_action='ignore',
            information_criterion='aic',
        )
    SARIMA_ORDERS[feat] = {
        'order':          auto.order,           # (p, d, q)
        'seasonal_order': auto.seasonal_order,  # (P, D, Q, m)
    }
# 2) Processamento e ajuste dinamico em janelas deslizantes de teste
def sarima_forecast_windows(X_test: np.ndarray, horizon: int,
                            orders: dict, feature_names: list) -> np.ndarray:
    """
    Ajusta SARIMAX localmente por maxima verossimilhanca usando o historico
    da janela de teste de look-back L=48 e dispara previsoes de multiplos passos.
    """
    N, L, C = X_test.shape
    preds = np.zeros((N, horizon, C), dtype=np.float32)
    for c, feat in enumerate(feature_names):
        order          = orders[feat]['order']
        seasonal_order = orders[feat]['seasonal_order']
        for i in range(N):
            history = X_test[i, :, c]
            try:
                with warnings.catch_warnings():
                    warnings.simplefilter('ignore')
                    res = SARIMAX(history,
                                  order=order,
                                  seasonal_order=seasonal_order,
                                  enforce_stationarity=False,
                                  enforce_invertibility=False).fit(disp=False)
                    fc = res.forecast(steps=horizon)
            except Exception:
                # fallback de robustez: persiste o ultimo passo conhecido (naive)
                fc = np.repeat(history[-1], horizon)
            preds[i, :, c] = np.asarray(fc, dtype=np.float32)
    return preds
```

4.3.6 Abordagem Híbrida e *Baselines* Triviais

A fusão de modelos preditivos do projeto é operacionalizada diretamente por meio de um *ensemble* híbrido, que calcula a média aritmética simples das projeções geradas de forma independente pelas classes neurais `CNN1D` e `LSTMModel`. A combinação é realizada de forma vetorizada sobre as saídas normalizadas no espaço de representação de formato (B,H,C), em que B representa o *batch size*, H é o horizonte de projeção e C indica o número de fatias de rede. Do ponto de vista matemático, essa

operação simplificada possui fundamentação devido à linearidade intrínseca da transformação de escala *Z-score* aplicada preliminarmente sobre a série temporal agregada. Como a desnormalização é descrita por uma função afim, a média aritmética calculada sobre as variáveis latentes no espaço padronizado é algebricamente idêntica à média que seria obtida caso o agrupamento ocorresse sobre as escalas físicas originais de requisições. Esse alinhamento linear permite realizar a fusão das saídas de forma cega antes do mapeamento inverso de escala, reduzindo o custo computacional com chamadas repetitivas de desnormalização de tensores e otimizando o *pipeline* de inferência.

Como referencial de controle inicial e teste de sanidade elementar para os estimadores neurais de alta capacidade do duto de simulação, o pipeline de simulação implementa o *baseline* univariado Naive de persistência temporal pura. Estruturado com fatiamento vetorizado de *arrays* para acelerar a execução iterativa do *benchmarking*, o algoritmo de persistência isola o último passo temporal conhecido da janela de observações históricas. Essa partição corresponde exatamente ao índice L_1, no qual o comprimento do histórico retroativo é fixado sob o parâmetro L = 48 horas. A extração isola um tensor estático de dimensões (N,1,C), em que N é o número total de janelas geradas para a partição de teste. Na sequência do *pipeline*, o código replica o sinal estático do último instante por toda a extensão do horizonte de previsão H, gerando um tensor final de saída com formato (N,H,C).

Para incorporar uma representação cíclica determinística que modele as flutuações diárias da infraestrutura 5G, também incorporamos o estimador trivial SNaive (do inglês, *Seasonal Naive*) ao ecossistema avaliativo. O processamento estrito da ciclicidade horária é governado pela constante de período diário definida como `SEASONAL_PERIOD` = 24 e a aritmética de indexação mapeia as previsões futuras baseando-se estritamente na periodicidade histórica das requisições: para cada passo relativo t+k do horizonte preditivo, com $k \in [0, H-1]$, o algoritmo busca a observação correspondente do dia anterior contida na janela histórica de entrada. No plano de armazenamento do tensor (N,L,C), esse mapeamento direciona o ponteiro para o índice relativo dado pelo cálculo algébrico $\texttt{idx} = L - 24 + k$.

Ademais, a rotina computacional do SNaive implementa um mecanismo de controle defensivo para lidar com horizontes de projeção que igualam ou excedem o período da sazonalidade diária de 24 horas, cenário correspondente ao Teste 2. Para mitigar a restrição física de borda e preservar a continuidade e a coesão matemática do sinal periódico projetado, é necessário impedir que o índice incremental k atinja limites que façam o índice relativo calculado `idx` igualar ou superar o comprimento do histórico L. Dessa forma, a rotina aplica a operação aritmética de módulo k(mod24), redefinindo ciclicamente as defasagens horárias a partir do início da janela de 24 horas anterior, mantendo a integridade das trajetórias preditas ao longo de horizontes estendidos.

Ao estruturar essas abordagens triviais e o modelo híbrido de fusão em pipelines puramente paralelos e vetorizados, o *benchmarking* assegura um protocolo rigoroso de controle experimental. Essa estratégia permite discernir a real contribuição de representação de características de modelos complexos frente a dinâmicas de regressão puramente persistentes ou sazonais, operando sob o mesmo regime de partição temporal e restrições metodológicas unificadas.

4.4 O Modelo PatchTST e o Estudo Ablativo

Esta seção se dedicará a apresentar a metodologia empregada no de estado da arte baseado em atenção adotado como núcleo deste projeto, o PatchTST, além de mostrar a estruturação do estudo ablativo correspondente a esta arquitetura. Diferente das arquiteturas *baseline* que possuem parametrizações estáticas, o modelo principal é investigado sob uma perspectiva dinâmica de refinamento arquitetural. O objetivo dessa estratégia é isolar, passo a passo, o impacto estatístico de técnicas contemporâneas de regularização, normalização reversível e codificação de tempo na

acurácia das previsões de tráfego móvel.

A fim de garantir a confiabilidade do estudo e isolar o impacto estrutural das mudanças avaliadas, todas as variantes do PatchTST e a versão pré-treinada foram submetidas a um ambiente de execução idêntico: as quatro configurações de ablação compartilham a mesma semente de inicialização igual a 42, o mesmo *batch size* de 32 janelas e o protocolo geral de otimização detalhado na seção 4.3.1, operando sob o algoritmo Adam com taxa de aprendizado de 10^{-3} ao longo de 100 épocas de treinamento supervisionado DMS. A seguir, apresentaremos de forma mais detalhada a engenharia de *patches* temporais, as premissas matemáticas das ablações de escala e o *pipeline* de aprendizado auto-supervisionado por mascaramento.

4.4.1 Arquitetura Base e Processo de Divisão em *Patches*

A modelagem preditiva principal desenvolvida neste projeto baseia-se na instanciação das classes estruturais `PatchTSTConfig` e `PatchTSTForPrediction`, pertencentes à biblioteca `transformers` do *Hugging Face* [21]. O modelo foi calibrado para processar a telemetria sob o paradigma de independência de canais, garantindo simetria metodológica com as demais arquiteturas neurais de referência. Sob essa diretriz, o tensor tridimensional de entrada, originalmente orientado na geometria de lote (B,L,C), é desmembrado univariadamente pelo *pipeline*. Esse isolamento dimensional reorganiza o fluxo de tensores de modo que cada série temporal de tráfego seja submetida de maneira isolada e individualizada ao extrator de características.

Para operacionalizar essa manipulação de maneira vetorizada e paralela na biblioteca PyTorch, os $C = 3$ canais correspondentes às demandas das fatias lógicas são empilhados ao longo da dimensão do lote. Essa reordenação espacial mapeia o sinal multivariado para a representação simplificada de formato (B×C,1,L), conduzindo o fluxo de dados como se fossem amostras univariadas independentes que compartilham de forma homogênea os mesmos pesos lógicos do codificador do *Transformer*. Do ponto de vista da engenharia de redes celulares 5G, essa estratégia de isolamento causal impede que a flutuação estatística abrupta ou o ruído físico de uma determinada fatia contamine as demais fatias lógicas concorrentes, preservando integridade estatística das previsões de fatias mais sensíveis perante canais que apresentam escalas estatísticas discrepantes e inerentemente desequilibradas.

A extração de *tokens* locais é realizada de forma determinística a partir da janela histórica de entrada normalizada de comprimento $L = 48$ horas. A configuração do modelo define o tamanho de cada bloco em $P = 8$ e o passo de deslocamento consecutivo em $S = 8$. A divisão lógica da sequência temporal univariada ao longo do tempo é regida pela aritmética de partição clássica:

$$N_p = \frac{L - P}{S} + 1 \tag{3}$$

Ao substituir as variáveis e parâmetros de simulação adotados no projeto, a formulação resulta na extração de exatamente 6 subséries locais contíguas e não-sobrepostas.

A agregação local em blocos de 8 horas (*patch-wise*) estabelece uma robusta vantagem frente ao paradigma clássico de tokenização por pontos horários individuais (*point-wise*) comumente empregado em modelos pioneiros de atenção. Em vez de submeter cada *timestamp* horário isoladamente à autoatenção, o agrupamento em blocos preserva e extrai propriedades semânticas locais estáveis ao longo de janelas diárias concentradas. Esse processamento local age de forma a suavizar flutuações rápidas de tráfego que ocorrem de maneira intermitente no plano de controle do núcleo de rede 5G, atenuando oscilações caóticas pontuais e inibindo o *overfitting* severo a que o Transformer de alta capacidade estaria sujeito em bases de dados de dimensões amostrais agregadas restritas.

Após a extração dos 6 blocos contíguos de comprimento igual a 8 horas, cada *patch* é encaminhado a uma camada de projeção linear comum para sua integração ao espaço latente do estimador.

Essa subcamada linear mapeia cada elemento do bloco de dimensão 8 para um vetor latente com dimensão de representação do modelo d_{model}, que é ajustado dinamicamente entre 16 e 64 unidades ao longo do estudo de ablação do projeto para avaliar o impacto da capacidade paramétrica da rede frente ao risco de coadaptação de pesos sinápticos. O tensor resultante de formato (BxC,6,d_{model}) é acrescido de um codificador posicional absoluto aditivo do tipo senoidal com o propósito de fornecer coordenadas estáveis que controlem a ordem temporal das fatias de sinal antes de alimentar o bloco de *encoders*. Essa adição posicional é indispensável para fornecer coordenadas cronológicas estáveis que controlem a ordem temporal dos *patches*, mitigando a invariância à permutação inerente ao mecanismo de autoatenção.

A principal consequência analítica dessa transformação de sinal reside na drástica redução do custo computacional associado ao cálculo da autoatenção global. Enquanto a formulação convencional do *Transformer* baseado em pontos horários individuais exige uma complexidade de tempo de ordem quadrática dada por $O(L^2)$ em relação ao comprimento do histórico, o particionamento em *patches* reduz esse comportamento limitante para $O(N_p^2)$. Em termos práticos de processamento, a computação da matriz de atenção diminui de uma grade densa com 48×48=2304 coeficientes individuais para uma representação minimalista de apenas 6×6=36 interações de autoatenção. Essa diminuição exponencial na sobrecarga de operações matriciais reduz sensivelmente o tempo de inferência e a alocação de memória física do estimador de alta capacidade, viabilizando decisões de gerenciamento rápidas e proativas que são compatíveis com os rigorosos requisitos de latência dos SLAs de controle celular da tecnologia 5G.

4.4.2 Configurações de Ablação: Capacidade, Normalização e Codificação Posicional (Ablações A, B e C)

A modelagem preditiva baseada em mecanismos de atenção global impõe desafios severos quando confrontada com cenários de amostragem limitada. Embora a implementação do modelo de alta capacidade PatchTST represente uma alternativa promissora para o gerenciamento de redes celulares, sua configuração original demonstrou um comportamento excessivo de *overfitting*. Sob esse arranjo padrão, o MSE atingiu as alarmantes magnitudes de 8368,46 para o horizonte de previsão H = 12 e 28938,66 para H = 24. Esse colapso preditivo decorre do fato de que o conjunto de dados utilizado, limitado a apenas 168 horas, é estruturalmente pequeno e caracterizado por padrões operacionais altamente regulares e cíclicos. Como consequência, a rede neural profunda tende a memorizar as flutuações e os ruídos de alta frequência do sinal de treino, comprometendo a sua capacidade de generalização na partição de teste.

Para conter essa memorização indesejada e isolar o impacto individual de cada barreira estrutural sobre o desempenho preditivo do Transformer, estruturou-se um estudo de ablação cumulativo e controlado, detalhado na listagem de parametrizações da Tabela 2.

A primeira etapa do estudo de ablação, identificada como `PatchTST_A_small`, visa mitigar o *overfitting* por meio de uma regularização extrema de capacidade arquitetural. Em séries temporais caracterizadas por uma baixa densidade de variação de sinal, *encoders* excessivamente profundos sofrem com a dispersão de gradientes e com a redundância de parâmetros. Visando restringir os graus de liberdade da rede, o `PatchTST_A_small` reduz drasticamente as dimensões do estimador: a dimensão do espaço latente d_{model} é reduzida de 64 para 16 unidades, a projeção do mecanismo de múltiplas cabeças n_{heads} cai de 4 para 2, e o empilhamento de blocos *encoder* é limitado a apenas $n_{layers} = 1$ camada. Complementarmente, a taxa de regularização implementada pela camada `nn.Dropout` é elevada de 0,1 para 0,3, atuando como um elemento inibidor contra coadaptação de pesos sinápticos e forçando o codificador a extrair apenas feições lineares estáveis do tráfego das fatias de serviço.

Configuração	**Nome**	**Dimensão Latente**	**Camadas do *Encoder***	**Cabeças de Atenção**	**Taxa de *Dropout***	**Normalização Interna (RevIN)**	**Tipo de PE Temporal**
PatchTST Original	`PatchTST_Original`	64	2	4	0,1	Desativada	Padrão
Ablação A	`PatchTST_A_small`	16	1	2	0,3	Desativada	Padrão
Ablação A+B	`PatchTST_AB_revin`	16	1	2	0,3	Ativada	Padrão
Ablação A+B+C	`PatchTST_ABC_sincos`	16	1	2	0,3	Ativada	Sinusoidal

Tabela 2: Parametrização cumulativa das corridas experimentais do estudo de ablação do PatchTST.

Na sequência experimental, a corrida denominada `PatchTST_AB_revin` acumula as limitações de capacidade parametrizadas no passo anterior e ativa o parâmetro de normalização interna `scaling`. O papel matemático da camada *Reversible Instance Normalization* (RevIN) fundamenta-se na estabilização de flutuações e desvios de média diária: durante a passagem de dados, o operador captura dinamicamente a média e o desvio padrão de cada amostra multivariada de forma isolada, processando unicamente a janela de histórico recente com *look-back* de $L = 48$ passos horários. Ao remover o efeito de escala local antes que o sinal atinja as cabeças de autoatenção do *Transformer*, a camada impede que as diferenças de magnitude estatística entre as fatias celulares degradem as projeções vetorizadas da rede. Posteriormente, na cabeça linear do sistema, o módulo desnormaliza a projeção de saída correspondente ao horizonte futuro, restabelecendo a escala física original das requisições e blindando o *pipeline* preditivo contra oscilações de amplitude que afetam fatias sensíveis como a URLLC.

O quarto estágio de investigação, denominado `PatchTST_ABC_sincos`, mantém as modificações de regularização e normalização ativas e altera estritamente o parâmetro de codificação posicional. Conforme as formulações teóricas demonstradas por Zhang et al. [22], o mecanismo de autoatenção global é estruturalmente invariante à permutação. Em séries temporais muito periódicas e de pequena extensão cronológica, as coordenadas posicionais aprendidas pelo modelo de linguagem tendem a decair ou diluir-se ao longo do processamento nas camadas profundas do *encoder*. A injeção de tensores estáticos baseados em funções senoidais determinísticas (seno e cosseno) tenta, portanto, resolver essa limitação. Essa operação matemática fixa as coordenadas cronológicas, restabelecendo a relação espacial e a vizinhança de 8 horas definida entre os $N_p = 6$ *patches* de histórico contíguos de comprimento $P = 8$, de modo que a ordem do sinal de tráfego celular seja fielmente preservada pelo extrator de características.

No plano de engenharia de *software*, a viabilização de ensaios experimentais sob diferentes níveis de restrição exigiu uma arquitetura de código que prevenisse a duplicação redundante de componentes lógicos. Para tanto, o *pipeline* de simulação foi centralizado sob a classe unificada `PatchTST`, derivada do módulo abstrato `nn.Module` do PyTorch, conforme detalhado no Listing 4. Esse invólucro customizado intercepta os argumentos do estudo de ablação e mapeia de forma dinâmica os dicionários de configuração para instanciar as classes de suporte `PatchTSTConfig` e `PatchTSTForPrediction` da biblioteca de modelos do *Hugging Face*. Ao unificar os pipelines de treinamento concorrentes, o *wrapper* processa os tensores históricos e retorna a saída preditiva diretamente sob o formato geométrico tridimensional (B,H,C), em que B representa o lote, H é o horizonte e C indica o canal de tráfego. Essa infraestrutura garante a simetria de dados necessária para a desnormalização e avaliação do *benchmarking* comparativo contra os demais *baselines* es-

tatísticos e lineares do projeto.

Listing 4: *Wrapper* customizado PatchTST integrado em PyTorch para encapsulamento dinâmico do estudo de ablação do *Hugging Face*

```
class PatchTST(nn.Module):
    """Wrapper sobre PatchTSTForPrediction parametrizavel para o Estudo de
       Ablacao.

    Os hiperparametros que afetam capacidade (`d_model`, `n_layers`, `n_heads`,
    `dropout`), normalizacao interna (`scaling` - RevIN quando ''std'') e o
    Positional Encoding temporal (`positional_encoding_type`) sao expostos como
    argumentos, permitindo testar uma variacao por vez sem reescrever a rede.

    Forward: (B, L, C) -> (B, H, C).
    """
    def __init__(self, look_back: int, horizon: int, num_channels: int,
                 patch_length: int = 8, patch_stride: int = 8,
                 # --- capacidade arquitetural (Ablacao A) ---
                 d_model: int = 64, n_heads: int = 4, n_layers: int = 2,
                 ffn_dim: int = 128, dropout: float = 0.1,
                 # --- Reversible Instance Norm interna (Ablacao B) ---
                 scaling=None,
                 # --- Positional Encoding temporal (Ablacao C) ---
                 positional_encoding_type: str = None):
        super().__init__()
        cfg_kwargs = dict(
            num_input_channels=num_channels,
            context_length=look_back,
            prediction_length=horizon,
            patch_length=patch_length,
            patch_stride=patch_stride,
            d_model=d_model,
            num_attention_heads=n_heads,
            num_hidden_layers=n_layers,
            ffn_dim=ffn_dim,
            dropout=dropout,
            head_dropout=dropout,
            scaling=scaling,
            do_mask_input=False,   # treino supervisionado direto (DMS)
        )
        if positional_encoding_type is not None:
            cfg_kwargs['positional_encoding_type'] = positional_encoding_type
        self.config = PatchTSTConfig(**cfg_kwargs)
        self.model  = PatchTSTForPrediction(self.config)

    def forward(self, x):  # x: (B, L, C)
        out = self.model(past_values=x)
        return out.prediction_outputs  # (B, H, C)
```

4.4.3 Pré-treinamento Auto-Supervisionado por Mascaramento (PatchTST_SSL)

A operacionalização do *pipeline* preditivo de alta capacidade baseia-se na implementação de uma estratégia híbrida de aprendizado profundo dividida em duas fases consecutivas, com o propósito de dotar o estimador de uma representação interna robusta frente a conjuntos de dados de extensão amostral restrita. A primeira fase compreende o pré-treinamento auto-supervisionado por mascaramento, operacionalizado por meio da classe `PatchTSTForPretraining` da biblioteca de modelos

da *Hugging Face*. Com o propósito de mitigar os riscos de *overfitting* identificados em rodadas preliminares de modelos de alta capacidade sobre sequências periódicas curtas, adota-se a topologia reduzida validada no estudo de ablação anterior. Essa arquitetura minimalista é definida com uma dimensão latente igual a 16, apenas uma camada de codificação e 2 cabeças de atenção, operando sob uma taxa de *dropout* igual a 0,3.

O processo de mascaramento é ativado de forma explícita com a seleção de blocos temporais definida de forma estocástica pelo parâmetro `random_mask_ratio` = 0,5 impõe a omissão aleatória de metade das subséries geradas na entrada. Durante 50 épocas de treinamento, o *pipeline* processa o fluxo utilizando o conjunto de dados normalizado de 117 horas de telemetria, ignorando completamente os rótulos físico de demanda supervisionada Y_{train}.

Sob essa modelagem auto-supervisionada, o objetivo de otimização consiste unicamente em reconstruir os 3 *patches* omitidos a partir dos 3 *patches* remanescentes que se mantêm visíveis. Essa formulação baseia-se na segmentação de uma janela de histórico de L = 48 horas em exatamente 6 *patches* contíguos de comprimento P = 8. Conforme fundamentado teoricamente por Nie et al. [21], o mascaramento de *patches* contínuos impede que o *Transformer* aprenda soluções de interpolação por proximidade temporal direta, eliminando uma limitação comum de modelos baseados em pontos discretos. Desse modo, o codificador é forçado a construir representações latentes robustas que capturam a ciclicidade diária e as dinâmicas de declínio de requisições de tráfego de rede celular.

Uma vez concluída a etapa de auto-supervisão, o fluxo executa uma transição estrutural para direcionar o conhecimento latente para a tarefa final de regressão preditiva direta de múltiplos passos. Para operacionalizar esse mapeamento, instancia-se de forma independente a classe `PatchTSTForPrediction`, desativando o mascaramento. A transferência de pesos do codificador ocorre sem contaminar o duto de regressão com as cabeças específicas de reconstrução de sinal da primeira fase. O *pipeline* realiza a extração exclusiva do dicionário de estados associado ao núcleo do *encoder* pré-treinado, injetando-o diretamente na topologia preditiva equivalente em `finetune_model.model`. Do ponto de vista arquitetural, esse nível de flexibilidade computacional é indispensável para herdar apenas as matrizes de atenção multidimensional e as subcamadas de projeção linear de subséries que capturaram as assinaturas cronológicas do tráfego das fatias lógicas, isolando a cabeça de projeção do horizonte preditivo de novos parâmetros a serem otimizados na fase seguinte.

Na etapa subsequente de *fine-tuning* supervisionado, o estimador preditivo, inicializado com as características latentes refinadas no pré-treinamento por mascaramento, é submetido a 100 épocas de otimização direta sob o critério de erro quadrático médio. Para contornar divergências geométricas das APIs de séries temporais do *Hugging Face*, desenvolveu-se a classe de *wrapper* interna `_PatchTSTSSLWrapper`, herdada da classe abstrata `nn.Module`. A função desse *wrapper* consiste em encapsular o método `forward` do estimador e mapear o retorno para um tensor tridimensional unificado de formato (B,H,C) a partir do atributo `prediction_outputs`. Essa manipulação vetorizada garante conformidade com as rotinas genéricas de predição e *benchmarking* na escala real de requisições. A implementação algorítmica completa do *pipeline* do modelo auto-supervisionado é apresentada no Listing 5.

Listing 5: *Pipeline* unificado para pré-treino por mascaramento de *patches* e posterior *fine-tuning* em PyTorch utilizando a API do *Hugging Face*

```
def train_patchtst_ssl(train_loader, look_back: int, horizon: int,
                       num_channels: int = NUM_CHANNELS,
                       # arquitetura compartilhada entre pretrain e fine-tune
                       d_model: int = 16, n_heads: int = 2, n_layers: int = 1,
                       ffn_dim: int = 128, dropout: float = 0.3,
                       patch_length: int = 8, patch_stride: int = 8,
```

```
                        # SSL
                        random_mask_ratio: float = 0.5,
                        pretrain_epochs: int = 50,
                        finetune_epochs: int = NUM_EPOCHS,
                        lr: float = LR, device=DEVICE, seed: int = SEED) ->
                            nn.Module:

    # Pre-treino auto-supervisionado + fine-tuning do PatchTST.
    common_cfg = dict(
        num_input_channels       = num_channels,
        context_length           = look_back,
        prediction_length        = horizon,
        patch_length             = patch_length,
        patch_stride             = patch_stride,
        d_model                  = d_model,
        num_attention_heads      = n_heads,
        num_hidden_layers        = n_layers,
        ffn_dim                  = ffn_dim,
        dropout                  = dropout,
        head_dropout             = dropout,
        scaling                  = 'std',
        positional_encoding_type = 'sincos',
    )

    # ---------- Etapa A: Pretraining (mascaramento aleatorio) ----------
    torch.manual_seed(seed)
    pretrain_cfg = PatchTSTConfig(
        **common_cfg,
        do_mask_input     = True,
        mask_type         = 'random',
        random_mask_ratio = random_mask_ratio,
    )
    pretrain_model = PatchTSTForPretraining(pretrain_cfg).to(device)
    optim = torch.optim.Adam(pretrain_model.parameters(), lr=lr)
    pretrain_model.train()

    for epoch in range(1, pretrain_epochs + 1):
        total, n = 0.0, 0
        for xb, _ in train_loader:  # ignora o Y; tarefa auto-supervisionada
            xb = xb.to(device)
            optim.zero_grad()
            out = pretrain_model(past_values=xb)  # HF calcula loss de
                reconstrucao
            loss = out.loss
            loss.backward()
            optim.step()
            total += loss.item() * xb.size(0); n += xb.size(0)
        if epoch == 1 or epoch % 10 == 0 or epoch == pretrain_epochs:
            print(f'  pre-epoch {epoch:>3}/{pretrain_epochs}  recon MSE:
                {total/n:.4f}')

    # ---------- Etapa B: Fine-tuning supervisionado ----------
    torch.manual_seed(seed)
    finetune_cfg = PatchTSTConfig(
        **common_cfg,
        do_mask_input = False,  # Desliga o mascaramento para regressao DMS
    )
    finetune_model = PatchTSTForPrediction(finetune_cfg).to(device)

```

```
    # Transferencia de pesos do encoder (backbone comum)
    backbone_state = pretrain_model.model.state_dict()
    info = finetune_model.model.load_state_dict(backbone_state, strict=False)

    # Wrapper para uniformizar a interface de inferencia
    class _PatchTSTSSLWrapper(nn.Module):
        def __init__(self, hf_model):
            super().__init__()
            self.model = hf_model
        def forward(self, x):
            return self.model(past_values=x).prediction_outputs

    wrapped = _PatchTSTSSLWrapper(finetune_model)
    wrapped = train_model(wrapped, train_loader,
                          num_epochs=finetune_epochs, lr=lr,
                          device=device, verbose_every=20)
    return wrapped
```

4.5 Métricas de Avaliação e Critérios de Validação

A quantificação da acurácia preditiva em problemas de LTSF exige um arcabouço métrico capaz de avaliar de forma confiável as discrepâncias entre os valores estimados e os comportamentos físicos reais observados nas fatias lógicas celulares. Sob a ótica matemática e estatística, este trabalho adota o MSE e o MAE como os critérios fundamentais de validação das trajetórias de tráfego projetadas. Para contemplar a natureza multivariada do *pipeline* preditivo sob o paradigma de previsão DMS, ambas as métricas são formalizadas ao longo do horizonte futuro H e das C fatias lógicas concorrentes por meio de representações tensoriais de amostragem. O cálculo do MSE global é governado pela formulação descrita pela equação abaixo:

$$MSE = \frac{1}{N * H * C} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{c=1}^{C} (y_{n,h,c} - \hat{y}_{n,h,c})^2 \tag{4}$$

Onde N representa o número total de janelas deslizantes avaliadas na partição de teste, $H \in 12, 24$ define o horizonte preditivo, $C = 3$ indica a quantidade de canais lógicos monitorados e $y_{n,h,c}$ e $\hat{y}_{n,h,c}$ correspondem, respectivamente, aos valores reais e às estimativas calculadas no espaço físico real. Complementarmente, o desvio linear médio do sistema é mensurado por meio do MAE global, conforme estruturado a seguir:

$$MAE = \frac{1}{N * H * C} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{c=1}^{C} |y_{n,h,c} - \hat{y}_{n,h,c}| \tag{5}$$

A seleção simultânea deste par de métricas baseia-se na profunda complementaridade analítica e estatística que elas oferecem ao operador de rede. O MSE aplica uma penalização de natureza quadrática sobre as discrepâncias de predição, o que o torna estatisticamente hiper-sensível a desvios de grande magnitude, agindo como um detector rigoroso de erros atípicos e *outliers*. Em contrapartida, o MAE oferece uma métrica de desvio baseada em uma escala estritamente linear, o que lhe confere uma robustez intrínseca contra a influência de anomalias espúrias e flutuações ruidosas de curtíssima duração. Ao expressar o erro médio esperado sem distorções geométricas induzidas por picos isolados, o MAE fornece ao engenheiro de planejamento uma visão fiel e estatisticamente estável sobre o comportamento de demanda contínua da infraestrutura celular.

Para que os erros computados tenham significado físico real de engenharia no contexto de redes 5G e permitam decisões operacionais válidas, estabelece-se o rigor metodológico de que todos os cálculos de erro ocorram estritamente na escala física original do tráfego (número de solicitações por hora), em detrimento da escala normalizada pelo *Z-score*. Embora a padronização estatística seja essencial durante a fase de treinamento para assegurar a estabilização dos gradientes e acelerar a convergência das redes neurais profundas, a permanência nesse espaço adimensional distorce a interpretação dos resultados, uma vez que as fatias operam sob escalas e amplitudes inerentemente desequilibradas. Dessa forma, o *pipeline* de simulação incorpora uma rotina computacional projetada especificamente para reverter a normalização de forma vetorizada e isolada. O processo de validação realiza inicialmente o achatamento temporário dos tensores de predição e dos alvos de teste de formato (N,H,C) para a dimensão bidimensional (N×H,C), seguindo para a aplicação do método de reversão do escalonador estatístico parametrizado estritamente com os dados calculados de forma exclusiva na partição de treinamento. Finalmente, as previsões desnormalizadas são reconduzidas ao formato tridimensional original para o cálculo seguro das métricas globais e por fatia individual de rede, consolidando um protocolo de validação consistente. O conjunto de rotinas matemáticas vetorizadas e a lógica de inversão de escala implementada em Python são apresentados detalhadamente no Listing 6.

Listing 6: Funções vetorizadas em NumPy para desnormalização de tensores tridimensionais via inversão de escala *Z-score* e cálculo das métricas de erro MSE e MAE globais e por fatias

```python
def inverse_scale(arr_scaled: np.ndarray) -> np.ndarray:
    """Aplica o scaler.inverse_transform a um tensor tridimensional (N, H, C)
    preservando sua geometria original sem misturar fatias ou amostras.
    """
    N, H, C = arr_scaled.shape
    flat = arr_scaled.reshape(-1, C)
    return scaler.inverse_transform(flat).reshape(N, H, C)

def compute_metrics(y_true: np.ndarray, y_pred: np.ndarray, feature_names: list)
    -> dict:
    """Calcula o MSE e o MAE na escala fisica, tanto de forma global
    quanto individualmente por canal (eMBB, URLLC, mMTC).
    """
    err = y_pred - y_true
    mse_global = float(np.mean(err ** 2))
    mae_global = float(np.mean(np.abs(err)))

    mse_per_channel = {feat: float(np.mean(err[..., c] ** 2))
                       for c, feat in enumerate(feature_names)}
    mae_per_channel = {feat: float(np.mean(err[..., c]))
                       for c, feat in enumerate(feature_names)}
    return {
        'MSE_global': mse_global,
        'MAE_global': mae_global,
        'MSE_per_channel': mse_per_channel,
        'MAE_per_channel': mae_per_channel,
    }
```

5 Resultados e Discussão

A validação empírica e a análise comparativa de desempenho das arquiteturas neurais e estatísticas propostas neste trabalho foram conduzidas sob um protocolo experimental unificado, visando mi-

tigar vieses sistemáticos e assegurar a equidade do *benchmarking*. Em conformidade com a modelagem descrita na Seção 4.1, o *setup* experimental foi estabelecido sobre uma partição de teste cronologicamente isolada correspondente às últimas 51 horas contínuas de telemetria horária do conjunto de dados e, para cada instante de amostragem ativa na partição de teste, as arquiteturas receberam um vetor histórico deslizante com comprimento de *look-back* fixado em 48 horas para computar as predições baseadas na estratégia DMS. Com o propósito de contemplar diferentes janelas de planejamento operacional da infraestrutura de rádio e núcleo, avaliaram-se concorrentemente dois cenários operacionais de previsão temporal: o Teste 1 (horizonte de curto prazo de H = 12 horas) e o Teste 2 (horizonte estendido de planejamento de H = 24 horas).

A compilação das métricas de erro na escala física real revelou um paradoxo estatístico de grande relevância para a engenharia de tráfego celular. Contrariando o apelo de alta capacidade que cerca os modelos baseados em autoatenção na literatura recente, a rede convolucional unidimensional operando sob independência de canais (CNN 1D) alcançou a liderança isolada em acurácia de predição global. Este estimador registrou um MSE de 303,18 no horizonte H = 12 e de 337,94 no horizonte H = 24. Esse proeminente desempenho preditivo foi acompanhado de perto pelos *baselines* clássicos SARIMA e SNaive, os quais convergiram de forma idêntica para um erro MSE de 634,43 para H = 12 e de 871,48 para H = 24. Em contrapartida, o estimador de atenção por blocos PatchTST Original apresentou uma severa degradação de desempenho, exibindo um erro MSE de 8368,46 para H = 12 e sofrendo um colapso preditivo no horizonte estendido, no qual o erro acumulado atingiu a magnitude crítica de 28938,66, que foi superada pelos *baselines* DLinear e LSTM, respectivamente com erros iguais a 14299,98 e 22489,91 para o mesmo cenário.

Os resultados obtidos correlacionam-se diretamente com os limites fundamentais de representação demonstrados nas seções teóricas preliminares deste relatório. O colapso preditivo observado na variante PatchTST original fornece a comprovação empírica da incapacidade das arquiteturas de atenção global em superar estimadores estatísticos e convolucionais locais em séries de dados altamente estáveis e cíclicas, como as derivadas do simulador DeepSlice [2]. Conforme fundamentado nas Seções 2.1 e 2.2, a ausência de mecanismos restritivos de capacidade, atrelada à perda progressiva da informação mútua da codificação posicional ao longo das camadas profundas do *encoder*, impede que o *Transformer* capture a cronologia intrínseca do sinal de tráfego, induzindo-o a um *overfitting* severo dos gradientes sobre ruídos de alta frequência. Esse comportamento valida a existência do *finite-sample gap* e a inadequação de operadores de atenção global irrestritos frente a dinâmicas temporais locais rígidas, cuja modelagem de vizinhança local é resolvida com maior eficiência por filtros convolucionais restritos e *baselines* puramente sazonais.

A fim de conduzir uma discussão profunda destas descobertas experimentais, este capítulo é estruturado em cinco seções de investigação analítica. Inicialmente, apresentaremos o *benchmarking* quantitativo global das métricas macro de erro na escala física de requisições horárias, estabelecendo o ordenamento de acurácia entre as arquiteturas neurais e estatísticas. Na sequência, desagregaremos o desempenho preditivo por canal lógico individual para as fatias eMBB, URLLC e mMTC, correlacionando os desvios de predição locais com o impacto operacional direto e os riscos de violação do SLA de rede móvel. Posteriormente, analisaremos o estudo de ablação cumulativo do PatchTST, isolando matematicamente o impacto estatístico da regularização de capacidade, da camada RevIN e da codificação posicional senoidal, além de quantificar os ganhos de representação latente viabilizados pelo pré-treinamento auto-supervisionado por mascaramento. Os efeitos decorrentes da estabilidade e rigidez temporal do conjunto de dados simulado são formalmente debatidos em seguida, ao passo que encerraremos a discussão ao examinar a viabilidade de implantação física das arquiteturas.

5.1 Aparato Experimental e Métricas Globais de *Benchmarking*

Os ensaios comparativos de treinamento e inferência das arquiteturas neurais e estatísticas foram executados centralizadamente em ambiente computacional baseado em CPU de arquitetura x86_64 sob o ecossistema PyTorch, adotando estritamente a parametrização de janelamento deslizante e o regime DMS, conforme estabelecido na Seção 4 deste projeto. Para assegurar a validade e a interpretabilidade física das trajetórias projetadas perante os requisitos de planejamento de rede, todas as métricas de erro foram calculadas na escala física original de solicitações por hora após sua inversão vetorizada. A Tabela 3 consolida o desempenho global obtido pelas arquiteturas na partição de teste para o horizonte operacional de curto prazo (H = 12).

A análise quantitativa para o horizonte H = 12 revelou a liderança incontestável da arquitetura CNN 1D. Sob a premissa de processamento univariado isolado, o estimador convolucional registrou os menores índices de erro preditivo global da partição de teste, assinalando um MSE de 303,1810 e um MAE de 9,0347. A superioridade da CNN 1D é explicada pelo compartilhamento de pesos locais e pelo campo receptivo temporal restrito de seus filtros convolucionais [23]. Esse arranjo estrutural atua como um regularizador implícito, capturando com relativa exatidão as transições locais de queda cíclica de tráfego de plano de controle sem memorizar flutuações e ruídos estocásticos de alta frequência. Como consequência, segundo mapeado graficamente pelas trajetórias da Figura 4, as previsões da CNN 1D se sobrepõem de forma quase perfeita às curvas de tráfego reais obtidas nas fatias lógicas de serviço eMBB, URLLC e mMTC. Em contrapartida, modelos recorrentes clássicos (LSTM) e a versão baseada em atenção PatchTST Original exibem nítidos atrasos de fase ou atenuam excessivamente os picos dinâmicos de demanda, conforme exibem as Figuras 5 e 6.

Modelo	**MSE**	**MAE**
PatchTST_Original	8368,4609	34,4368
PatchTST_A_small	22167,8672	55,4792
PatchTST_AB_revin	20511,9727	57,1422
PatchTST_ABC_sincos	20511,9727	57,1422
PatchTST_SSL	17844,9609	51,0134
SARIMA	634,4361	4,3361
CNN1D	303,1810	9,0347
LSTM	19970,5703	54,4290
DLinear	10720,2822	34,6282
Ensemble_CNN_LSTM	5436,3770	29,3565
Naive	82940,3203	86,1028
SNaive	634,4361	4,3361

Tabela 3: Resultados das métricas de erro globais para o horizonte de previsão de 12 horas.

Outro fenômeno de relevância estatística reside no empate matemático rigoroso observado entre o *baseline* linear estatístico SARIMA e o *baseline* trivial cíclico SNaive, cujas curvas de predição espacial convergiram de forma idêntica para um erro MSE global de 634,4361 e MAE de 4,3361, conforme explicitado pela Tabela 3 e pela Figura 7. A explicação analítica para essa equivalência decorre do *loop* de sintonização automática via Critério de Informação de Akaike (AIC) executado unicamente na partição de treinamento normalizada. O algoritmo *stepwise* selecionou de forma homogênea para os três canais lógicos de tráfego a ordem estrita de passeio aleatório sazonal puro com diferenciação diária, especificada matricialmente sob a configuração $(0, 0, 0)(0, 1, 0)_{24}$.

Algebricamente, essa parametrização reduz a equação do SARIMA à projeção determinística da observação registrada exatamente 24 horas atrás na janela histórica deslizante, o que coincide

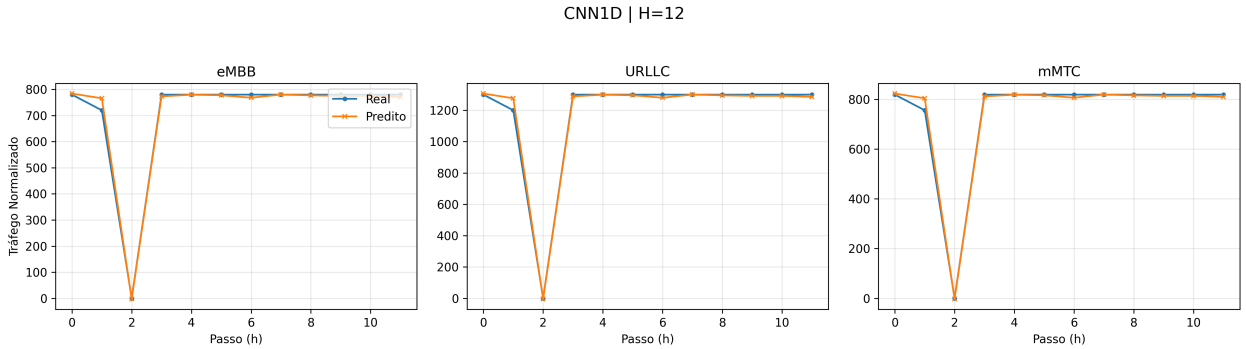

Figura 4: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a CNN 1D.

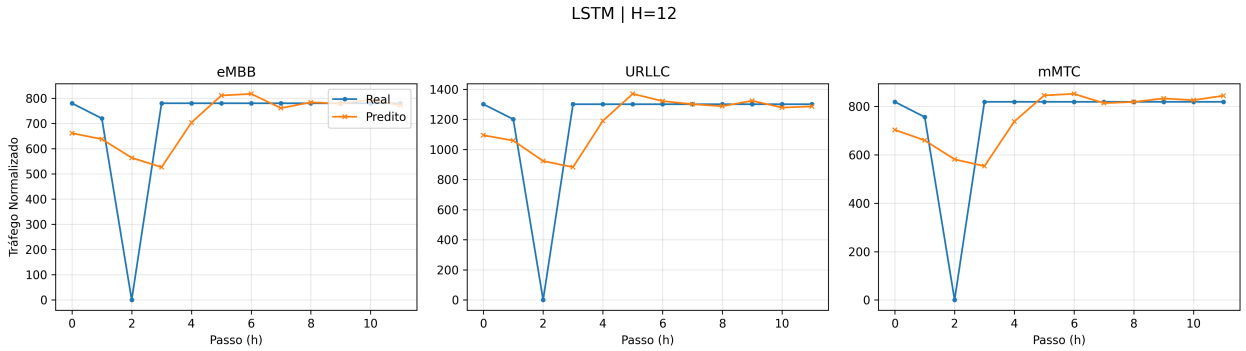

Figura 5: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a LSTM.

perfeitamente com a lógica de persistência de ciclicidade diurna do estimador trivial SNaive, visualmente mapeado na Figura 8. A regularidade diária e a rigidez temporal intrínsecas ao conjunto de dados simulado pelo *framework DeepSlice* explicam por que um regressor de persistência de período diário se estabeleceu como um estimador extremamente competitivo. Essa simplicidade estrutural superou redes recorrentes profundas baseadas em LSTM (MSE = 19970,5703) e estimadores lineares livres de atenção como o DLinear (MSE = 10720,2822), visualmente mapeado pela Figura 9.

Ao transicionar para o cenário operacional de H = 24 horas, identificou-se uma elevação generalizada e sistemática nos desvios de predição de todas as arquiteturas avaliadas, conforme exibido pela Tabela 4. Sob a perspectiva de engenharia de tráfego, essa degradação preditiva é governada por três pilares complementares. Primeiramente, a aritmética de projeção DMS exige que as cabeças lineares finais estimem simultaneamente um espaço de saída equivalente a H×C. Duplicar o horizonte de predição para 24 horas dobra o tamanho geométrico das matrizes de projeção final, elevando a contagem de parâmetros livres do otimizador e propiciando o *overfitting* face à amostragem limitada do treino.

Ademais, conforme discutido por Zhang et al. [23], à medida que o alvo predito afasta-se cronologicamente do fim da janela histórica, as camadas de autoatenção sofrem com a dispersão de entropia de seus pesos de similaridade. O enfraquecimento das coordenadas posicionais e o decaimento posicional (do inglês, *positional encoding decay*) nas camadas profundas do *encoder* diluem a capacidade de identificar os "*tokens* positivos"associados aos picos diários. Por fim, o

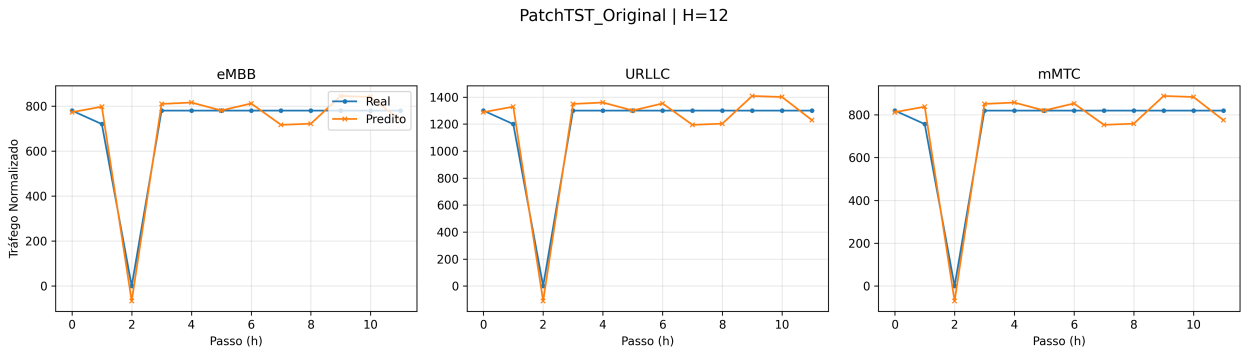

Figura 6: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o PatchTST Original.

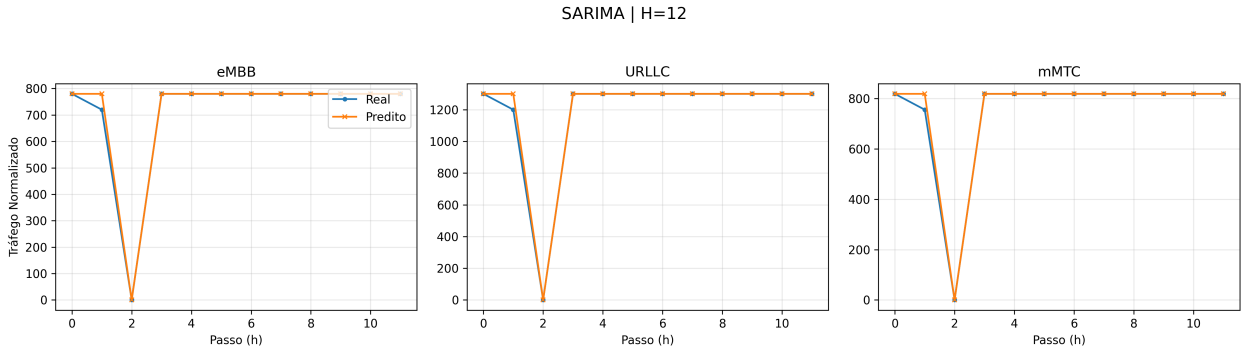

Figura 7: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o SARIMA.

prolongamento do horizonte temporal eleva a entropia natural do comportamento dinâmico de rede, reduzindo as correlações estatísticas locais de curto prazo que dão suporte à estabilização das previsões.

Sob esse cenário estendido de 24 horas, a CNN 1D, cuja curva de previsão temporal é exibida na Figura 10, também se consolida como o estimador mais robusto do estudo, apresentando uma reconstrução visual praticamente perfeita das trajetórias reais das três fatias de rede. Diferente dos demais modelos neurais, a arquitetura convolucional unidimensional consegue capturar com considerável fidelidade matemática a queda abrupta a zero registrada no passo t = 2 e, de forma ainda mais notável, se mantém consideravelmente estável e plana ao longo de todo o horizonte subsequente (t = 3 a t = 23), sem introduzir atrasos de fase ou distorções de amplitude. Esse desempenho excepcional resulta em um erro quadrático médio extremamente controlado, com MSE = 337,9471.

Já o modelo baseado em projeções espaciais e decomposição temporal, DLinear, assinalou erros intermediários de MSE global de 10720,2822 para H = 12 e de 14299,9766 para H = 24, superando o LSTM por uma larga margem e apresentando um comportamento substancialmente mais estável do que o PatchTST Original, o que demonstra de forma empírica as vantagens de sua baixa parametrização. Ao decompor o sinal bruto em tendência e resíduo sazonal e aplicar apenas duas camadas lineares univariadas de camada única sob o paradigma de independência de canais, o modelo se estabelece como um estimador altamente regularizado, prevenindo o *overfitting* de alta variância que afeta os mecanismos de autoatenção sob escassez de dados amostrais.

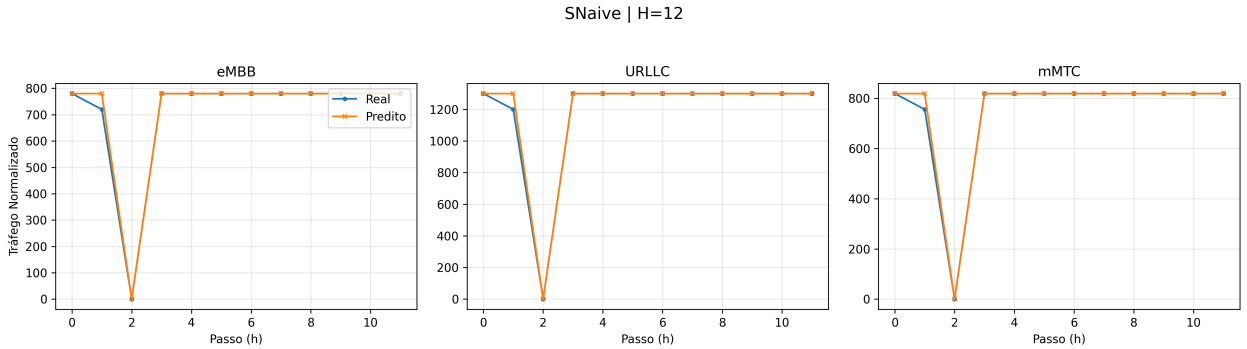

Figura 8: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o SNaive.

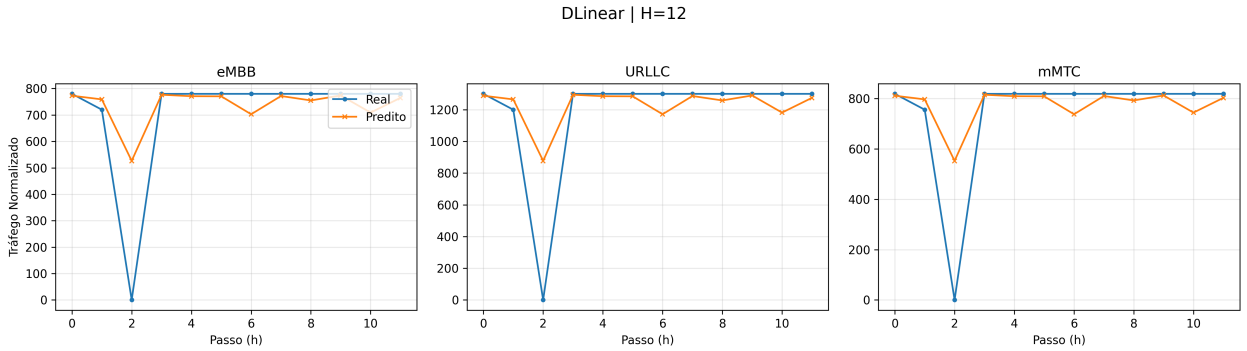

Figura 9: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o DLinear.

Entretanto, as restrições inerentes a representações estritamente lineares impõem severas barreiras preditivas no contexto do fatiamento 5G. Como a modelagem matemática do DLinear projeta o horizonte preditivo por meio de uma soma ponderada global de pesos ao longo de todo o histórico de 48 horas, a arquitetura carece de capacidade de representação para modelar transições do tipo degrau e picos locais não-lineares [7]. Conforme ilustrado nas Figuras 9 e 11, o modelo é incapaz de alcançar o valor zero de tráfego registrado de forma cíclica no passo temporal t = 2, gerando projeções suavizadas, o que resulta em uma expressiva penalização quadrática residual que eleva sistematicamente as métricas globais de erro do sistema.

Em relação ao *baseline* recorrente LSTM, tivemos o pior desempenho global entre todas as arquiteturas neurais investigadas, com índices MSE iguais a 19970,5703 para o Teste 1 e 22489,9062 para o Teste 2. Duas limitações de engenharia de *software* e de representação explicam esse colapso analítico: primeiramente, o modelo foi parametrizado sob a hipótese de acoplamento e mistura de canais, alimentando de forma integrada o vetor multivariado de entrada das fatias. Uma vez que as demandas operacionais das três fatias lógicas operam sob escalas físicas e amplitudes altamente desbalanceadas, a otimização de pesos no espaço latente compartilhado causou uma contaminação de escala, em que o gradiente da fatia de maior tráfego (URLLC, com picos de 1300 solicitações) sufocou o aprendizado das demais fatias lógicas concorrentes.

Em segundo lugar, a mecânica sequencial de transição de estado celular das redes recorrentes introduz uma severa inércia temporal ao lidar com transições dinâmicas rápidas [3]. De acordo com o que é evidenciado pelas trajetórias mapeadas pelas Figuras 5 e 12, as curvas preditas sofrem

Modelo	MSE	MAE
PatchTST_Original	28938,6621	62,5598
PatchTST_A_small	30227,4629	73,5327
PatchTST_AB_revin	30239,2461	74,1327
PatchTST_ABC_sincos	30239,2461	74,1327
PatchTST_SSL	32609,7539	77,5223
SARIMA	871,4781	5,7520
CNN1D	337,9471	9,0689
LSTM	22489,9062	68,6078
DLinear	14299,9766	45,4357
Ensemble_CNN_LSTM	6309,4854	35,9495
Naive	106930,3750	111,5000
SNaive	871,4781	5,7520

Tabela 4: Resultados das métricas de erro globais para o horizonte de previsão de 24 horas.

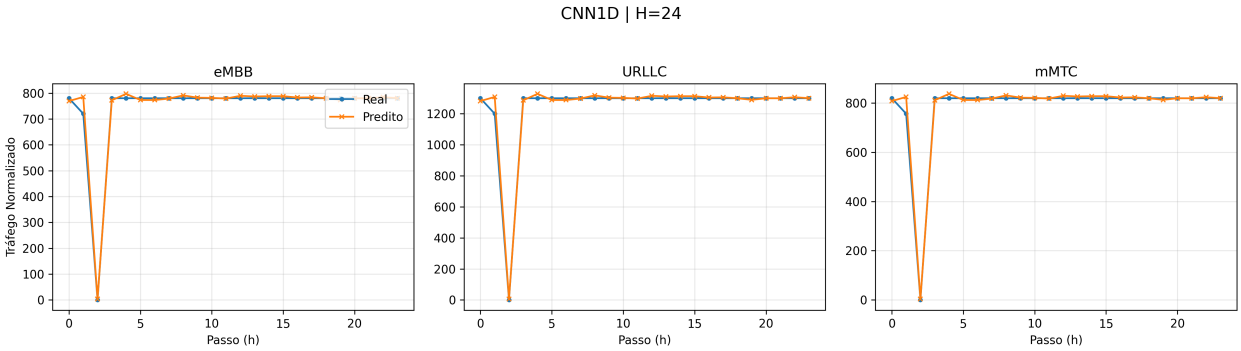

Figura 10: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a CNN 1D.

de um acentuado atraso de fase na detecção da queda abrupta de tráfego. Como a rede calcula recursivamente os estados passo a passo, o histórico acumulado de tráfego de pico mantém o vetor oculto saturado, exigindo múltiplas iterações lógicas das portas de esquecimento e de entrada para purgar o sinal residual anterior e refletir a queda instantânea do tráfego real [19]. Somado a esse atraso, o fatiamento DMS de vetor oculto final único (h_L) atua como um gargalo de informação que suprime as características temporais finas da sequência, achatando a curva projetada ao redor da média e inviabilizando seu uso proativo perante as estritas SLAs de latência do plano de controle celular 5G.

Complementarmente, o comportamento empírico da abordagem *Ensemble* CNN-LSTM fornece importantes subsídios para a compreensão das dinâmicas de fusão tardia de representações. Segundo os resultados mostrados pelas Tabelas 3 e 4, o modelo híbrido obteve um desempenho intermediário em ambos os horizontes preditivos, com MSE = 5436,3770 para H = 12 e MSE = 6309,4854 para H = 24.

Analisando as Figuras 13 e 14, vemos que a melhoria expressiva de acurácia obtida pelo ensemble em relação ao modelo LSTM isolado decorre de um efeito de ancoragem estatística provido pela rede convolucional. Uma vez que que a LSTM conjunta sofre com severo atraso de fase e suavização excessiva nos pontos de transição abrupta de demanda, a média aritmética simples com as projeções perfeitamente temporizadas e de amplitude precisa da CNN 1D atua mitigando as

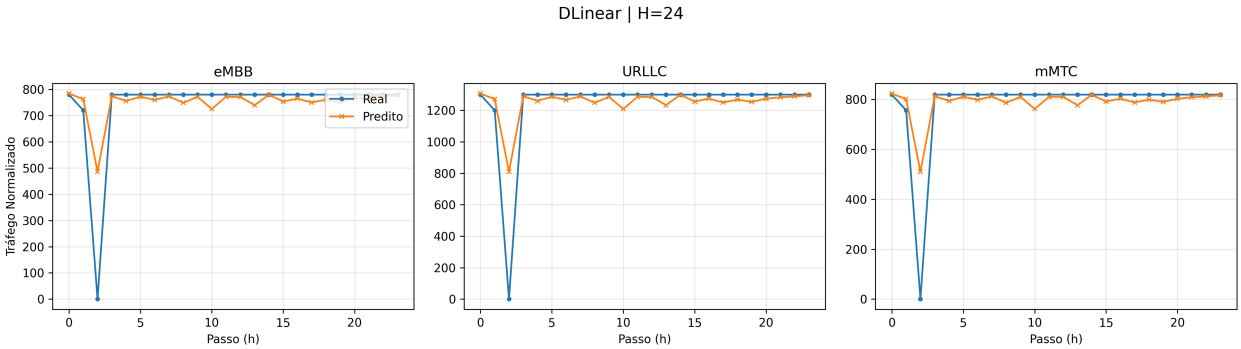

Figura 11: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a DLinear.

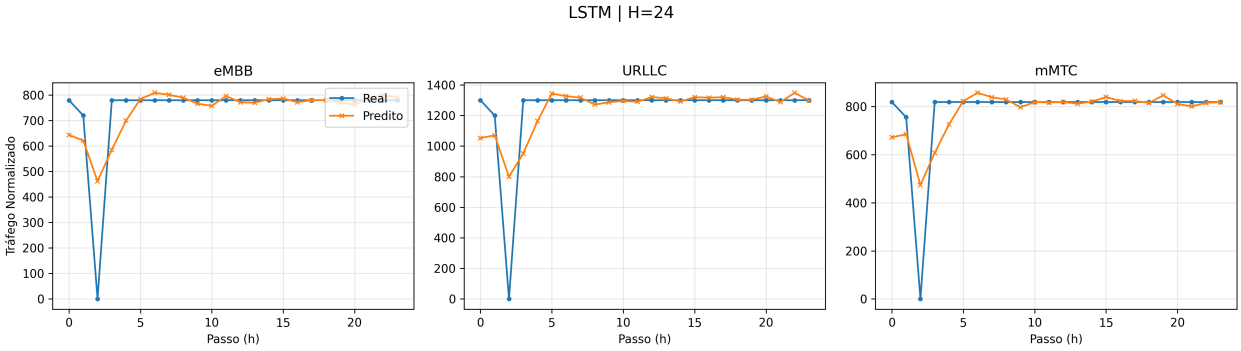

Figura 12: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a LSTM.

distorções recorrentes, forçando a curva combinada a aproximar-se da trajetória de tráfego real.

Em contrapartida, constata-se que a hibridização linear degrada severamente a acurácia satisfatória obtida pela CNN 1D quando operada individualmente. Esse decréscimo preditivo é reflexo de um fenômeno de contaminação por inércia temporal. Ao fundir deterministicamente as saídas de ambos os modelos na escala latente normalizada, o atraso de fase sistemático e o amortecimento de pico intrínsecos à representação sequencial da LSTM são parcialmente transferidos para a resposta final do sistema. Consequentemente, o *Ensemble* herda uma versão atenuada das deformações morfológicas da recorrência, o que inviabiliza a sua adoção frente à superioridade isolada, simplicidade paramétrica e eficiência de processamento demonstradas pela topologia puramente convolucional sob o regime de independência de canais.

Em contraste, o modelo de atenção global PatchTST Original e suas variantes exibem um colapso estrutural caracterizado por oscilações espúrias, segundo o que é exibido da Figura 15 a 23. Embora o tráfego real permaneça completamente linear e plano do passo t = 3 ao t = 23, as variantes do PatchTST geram quedas e flutuações fantasmas de tráfego de grande magnitude ao redor dos passos t = 10 e t = 18. Esse comportamento revela um *overfitting* severo das cabeças de atenção ao padrão cíclico diário do histórico de treino: por possuírem excessivos graus de liberdade e uma representação espacialmente frouxa induzida pela autoatenção, os *Transformers* tentam forçar periodicidades inexistentes na janela de teste, o que justifica a explosão de seu MSE global para a magnitude crítica de 28938,6621.

Quanto aos estimadores estatísticos SARIMA, SNaive e o *baseline* trivial Naive, do ponto de

vista estritamente visual e morfológico, constata-se que eles não apresentam alterações significativas em suas trajetórias preditivas ao transicionar do horizonte de curto prazo para o cenário estendido, conforme evidenciado pelo contraste entre as Figuras 7 e 24, 8 e 25 e 26 e 27. Esse comportamento não é uma inconsistência algorítmica, mas sim uma consequência matemática direta do alinhamento cronológico das partições de teste sob a rigidez cíclica do simulador *DeepSlice*.

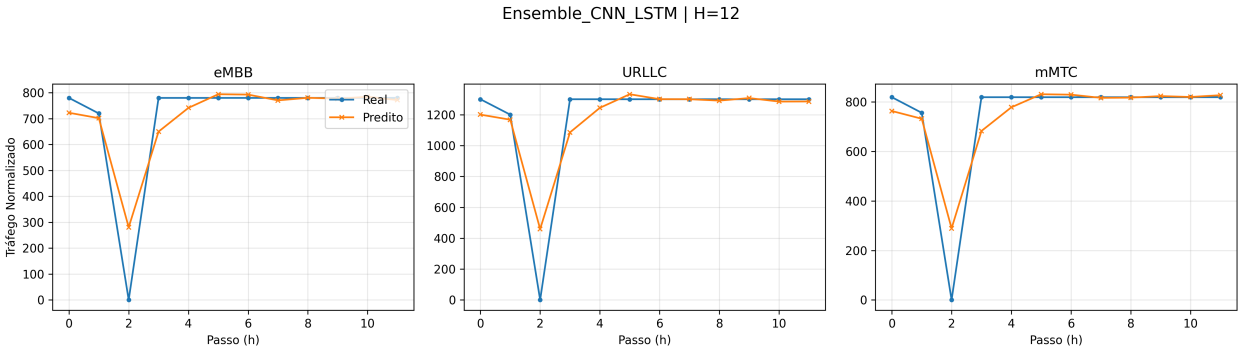

Figura 13: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o *Ensemble*.

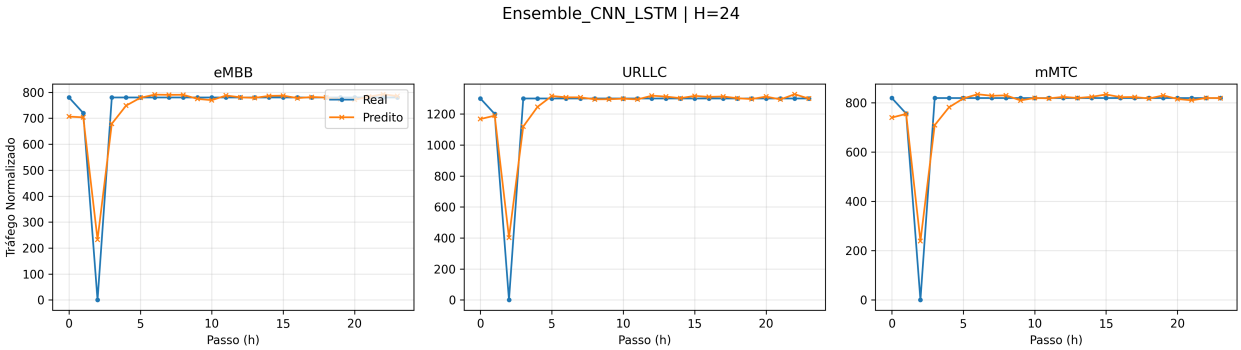

Figura 14: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para o *Ensemble*.

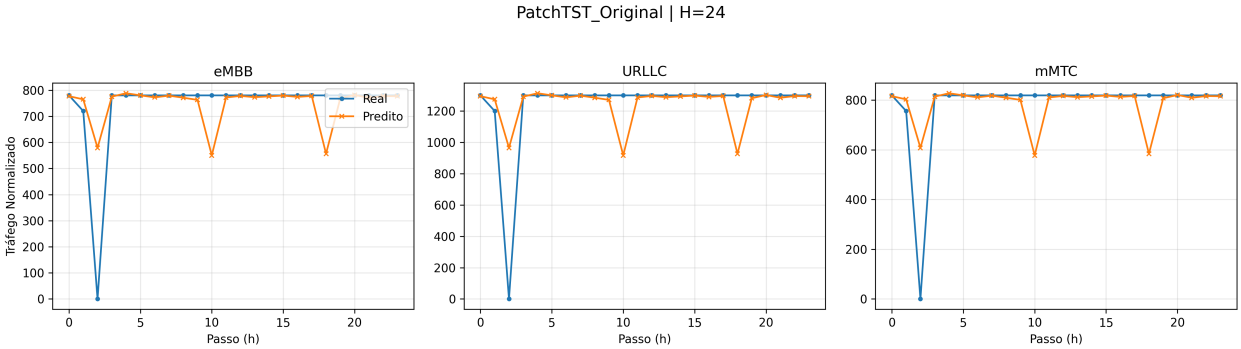

Figura 15: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para o PatchTST Original.

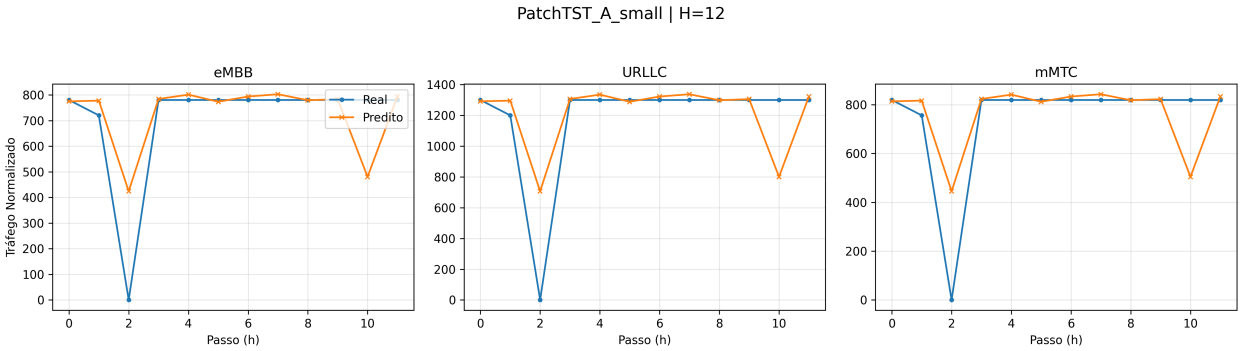

Figura 16: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a variante A da ablação.

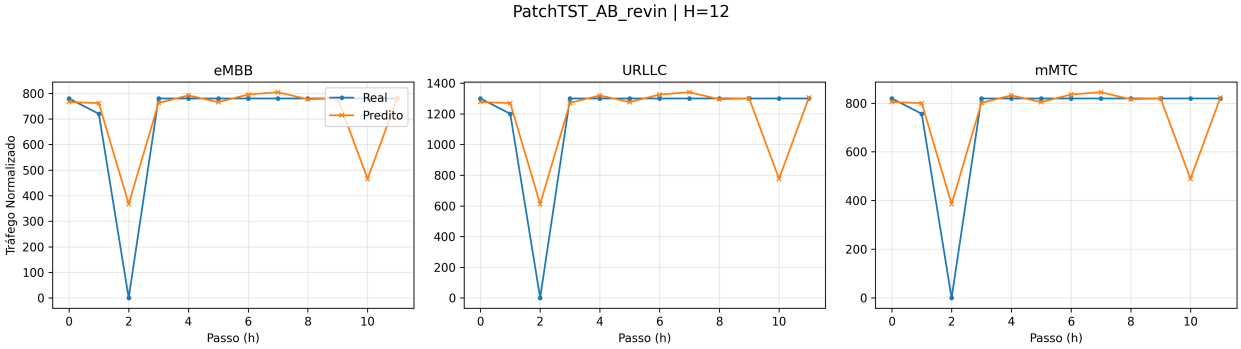

Figura 17: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a variante A+B da ablação.

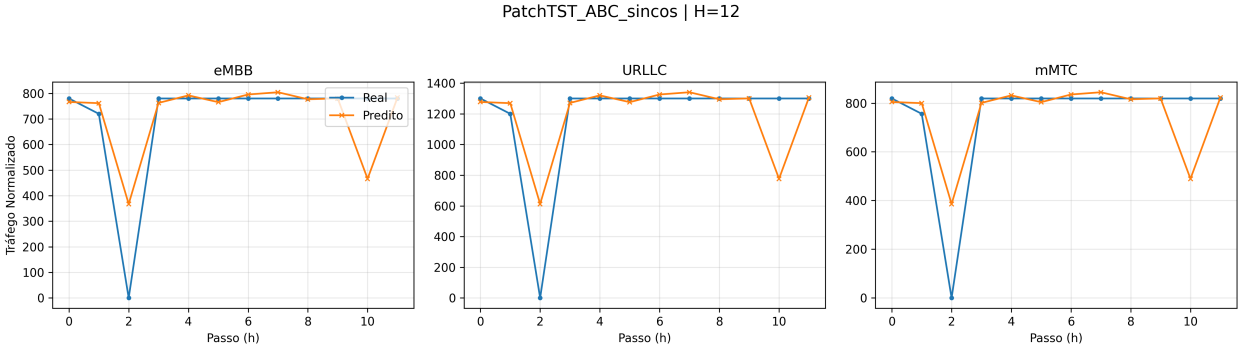

Figura 18: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a variante A+B+C da ablação.

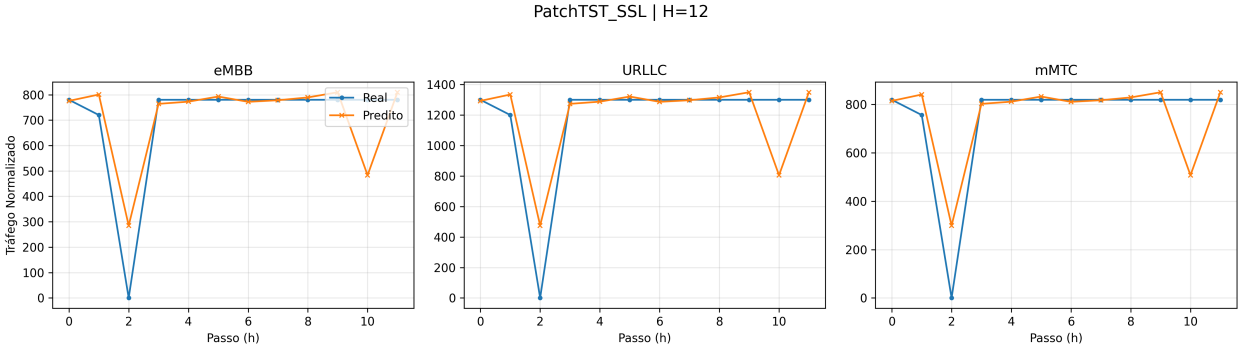

Figura 19: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para a variante com pré-treino SSL da ablação.

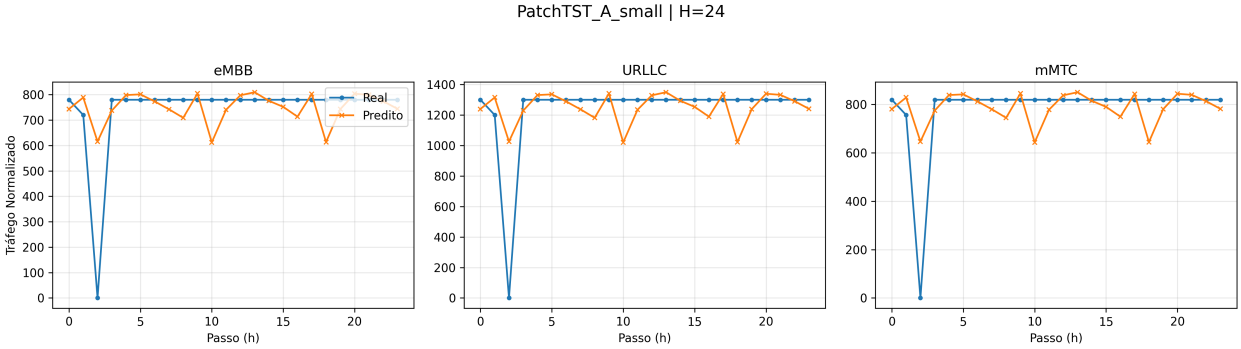

Figura 20: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a variante A da ablação.

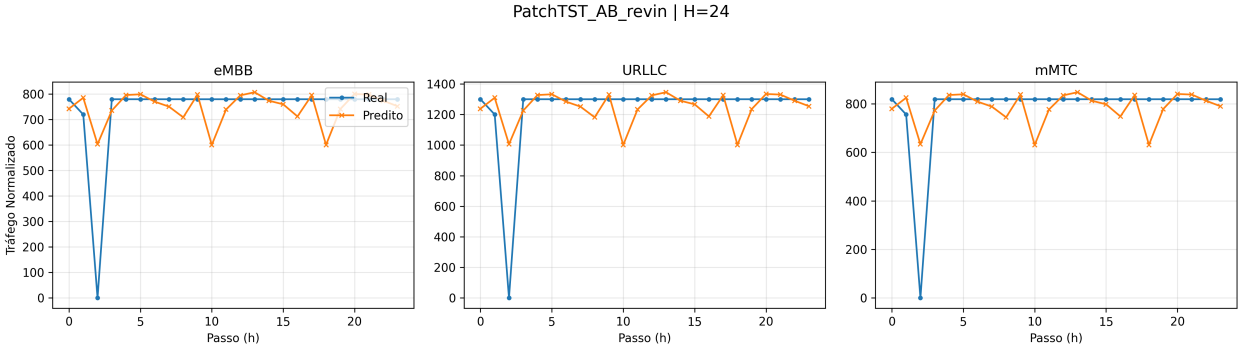

Figura 21: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a variante A+B da ablação.

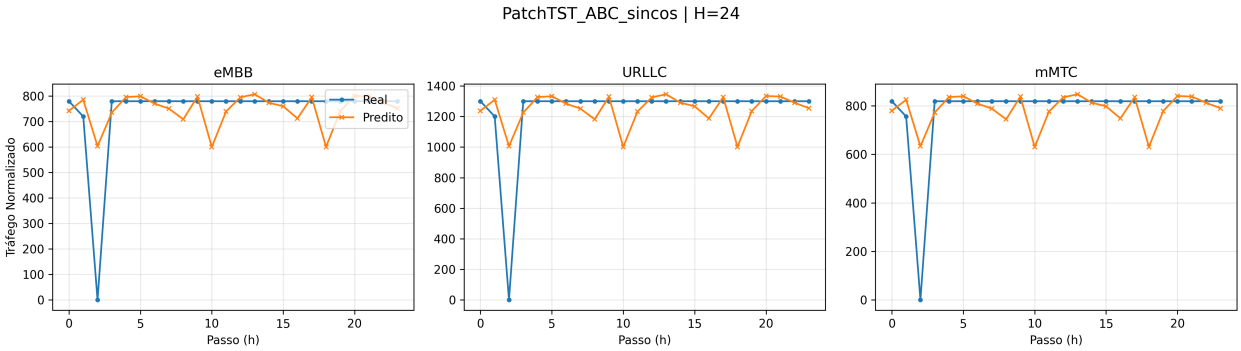

Figura 22: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a variante A+B+C da ablação.

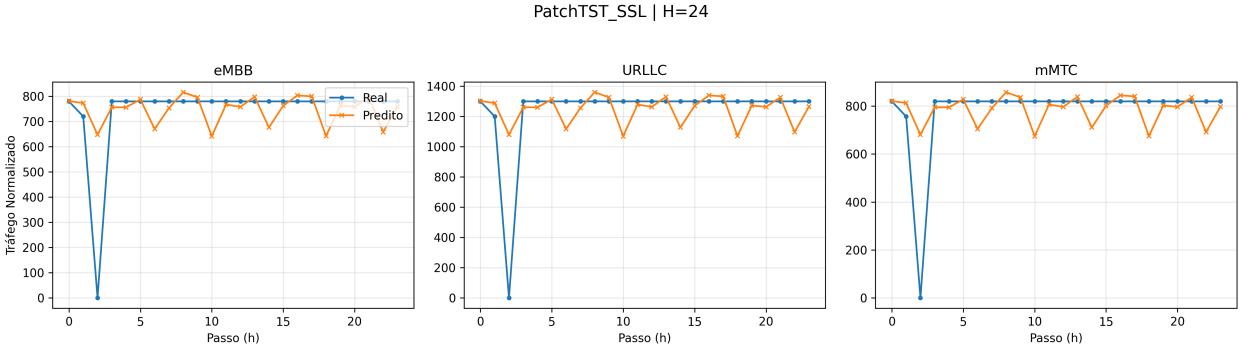

Figura 23: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para a variante com pré-treino SSL da ablação.

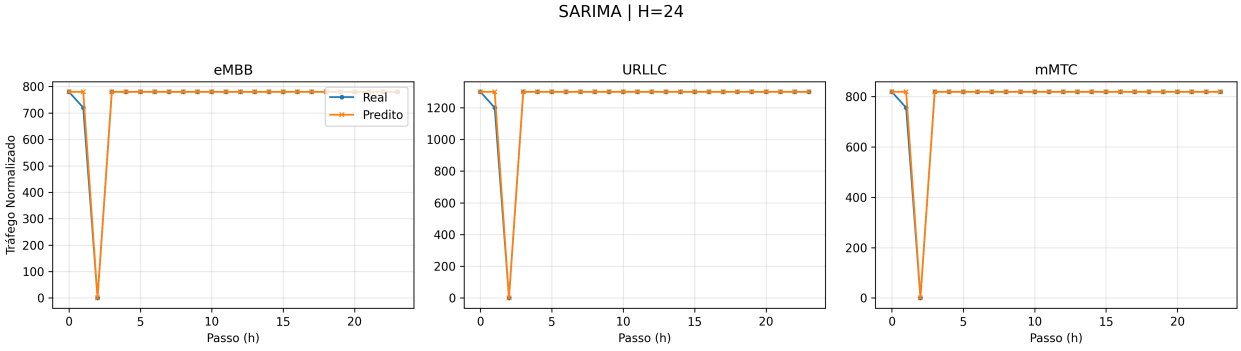

Figura 24: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para o SARIMA.

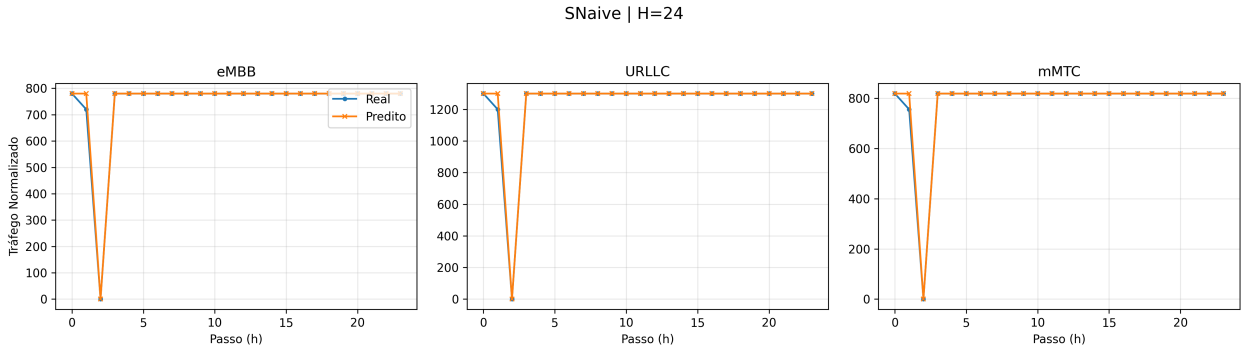

Figura 25: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para o SNaive.

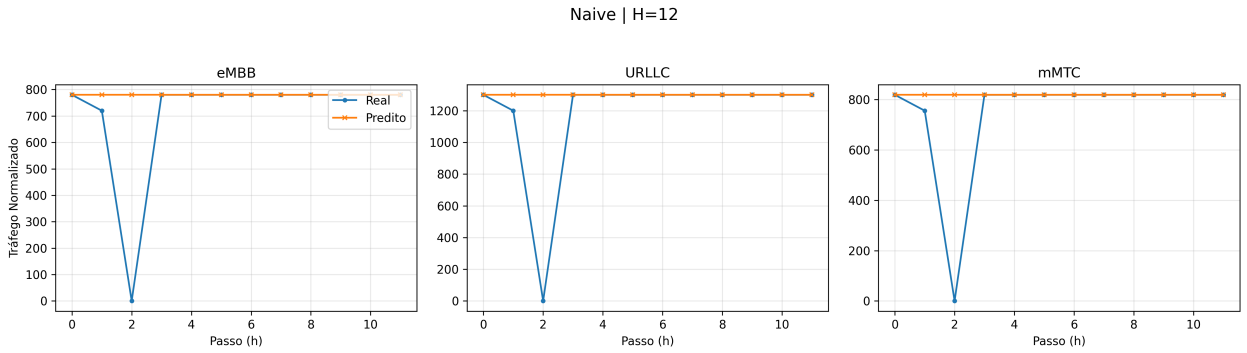

Figura 26: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 12 para o Naive.

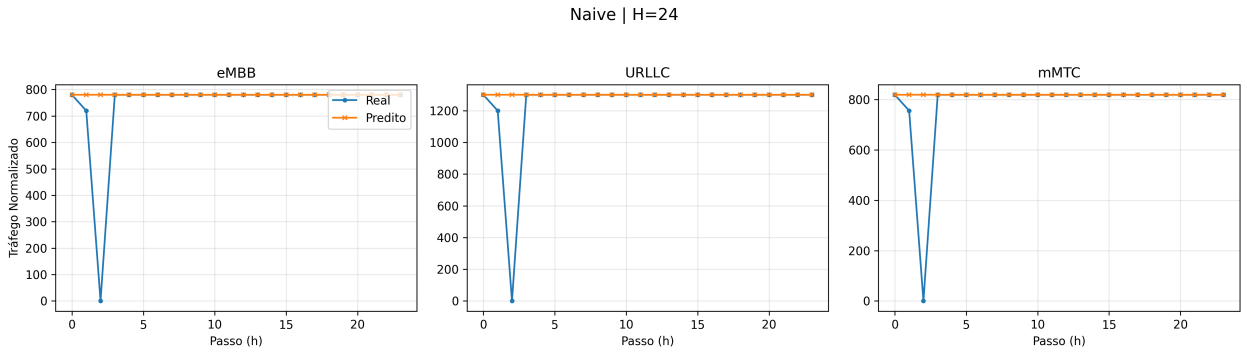

Figura 27: Curvas de predição temporal das fatias eMBB, URLLC e mMTC obtidas no conjunto de teste para o horizonte H = 24 para o Naive.

Visto que a primeira janela de teste se inicia no *timestamp* igual a 117 e o único evento de decréscimo abrupto de tráfego programado no ciclo diário de 24 horas ocorre no *timestamp* 119, a dinâmica de perturbação de sinal fica restrita unicamente ao passo relativo t = 2 de ambos os horizontes preditivos. Na escala de longo prazo de H = 24, a próxima queda cíclica de tráfego estaria situada apenas no *timestamp* 143, correspondente ao passo relativo t = 26, o qual reside fora do limite superior do vetor de saída do regime DMS avaliado.

Consequentemente, tanto o horizonte de 12 horas quanto o de 24 horas englobam exatamente a mesma transição singular de decréscimo e o mesmo platô subsequente de tráfego estável. O modelo SNaive e, por conseguinte, o SARIMA, cuja parametrização ótima $(0,0,0)(0,1,0)_{24}$ selecionada via AIC reduziu-se analiticamente à persistência sazonal pura, limitam-se a transpor a trajetória linear rígida do período anterior de forma idêntica para ambos os cenários de simulação. De modo análogo, o estimador Naive estático estende uma projeção linear perfeitamente horizontal baseada no último valor histórico observado de alto tráfego (*timestamp* 116), gerando uma linha de persistência de pico paralela ao eixo temporal em ambos os horizontes, cuja única penalidade métrica reside no erro quadrático gerado especificamente no passo de transição rápida t = 2.

5.2 Desempenho Desagregado por Fatia e Análise de SLAs

A fim de viabilizar uma tomada de decisão proativa e resiliente no plano de controle do núcleo de rede 5G, a análise agregada de métricas globais de erro deve ser destrinchada em uma avaliação individualizada e desagregada por fatia lógica de serviço. Como as fatia eMBB, URLLC e mMTC operam sob requisitos técnicos de vazão, latência e densidade de conexões fundamentalmente antagônicos, desvios de previsão de mesma proporção relativa acarretam consequências severamente assimétricas em seus respectivos SLAs.

Para fundamentar matematicamente e validar essa sensibilidade operacional na escala física real de solicitações por hora, as métricas de erro quadrático médio e erro absoluto médio foram calculadas de forma isolada para cada um dos canais lógicos nos horizontes H = 12 e H = 24, abrangendo as 12 configurações algorítmicas desenvolvidas neste estudo, conforme sistematizado nas Tabelas 5 e 6.

A análise quantitativa das tabelas revela um comportamento estático marcante: os erros de escala física associados à fatia URLLC são sistematicamente superiores aos erros observados nos outros dois canais para todas as técnicas e horizontes avaliados. Esse desbalanceamento não reflete uma incapacidade intrínseca dos modelos de mapear a dinâmica da URLLC, mas traduz o "efeito de escala física"decorrente do processo de normalização Z-score implementado durante a fase de pré-processamento. Nessa etapa, o escalonamento projeta as três fatias lógicas para um espaço latente comum de média zero e desvio padrão unitário, tornando os sinais normalizados morfologicamente idênticos, uma vez que o simulador *DeepSlice* gera oscilações síncronas entre os patamares de pico e quedas cíclicas diárias a zero. No entanto, os parâmetros de dispersão aprendidos exclusivamente na partição de treinamento exibem amplitudes físicas muito distintas: o desvio padrão (σ) da fatia URLLC ($\sigma \approx 236,2255$) é substancialmente superior ao desvio da eMBB ($\sigma \approx 141,7353$) e da mMTC ($\sigma \approx 148,8220$).

Ao aplicar a função de inversão de escala para restaurar as previsões à unidade física original, o erro residual é multiplicado linearmente pelo desvio padrão (σ) para a obtenção do MAE, e de forma quadrática pela variância (σ^2) para o cálculo do MSE. Esse efeito multiplicador é matematicamente comprovado pela razão de dispersão entre as fatias URLLC e eMBB: a razão $\sigma_{URLLC}/\sigma_{eMBB}$ resulta na constante 1,6666, que é rigorosamente equivalente à razão entre os erros MAE obtidos pelo modelo líder CNN 1D para H = 12 (12,1543/7,2926 = 1,6666). De modo correspondente, a razão da variância quadrática entre esses canais ($\sigma^2_{URLLC}/\sigma^2_{eMBB} \approx 2,7777$) reflete-se na razão de seus erros MSE (517,6978/186,3712 $\approx$ 2,777). Sob a perspectiva de engenharia de tráfego, esse fenômeno evidencia que desvios residuais idênticos no espaço latente de treinamento traduzem-se em um volume de solicitações de rádio não-atendidas muito maior na classe URLLC, justificando o monitoramento proativo individualizado por canal para mitigar riscos de congestionamento físico.

Modelo	MSE			MAE		
	eMBB	**URLLC**	**mMTC**	**eMBB**	**URLLC**	**mMTC**
PatchTST_Original	5144,2529	14289,5908	5671,5386	27,7966	46,3276	29,1864
PatchTST_A_small	13627,0146	37852,8125	15023,7793	44,7814	74,6357	47,0205
PatchTST_AB_revin	12609,1045	35025,2852	13901,5342	46,1238	76,8729	48,4300
PatchTST_ABC_sincos	12609,1045	35025,2852	13901,5342	46,1238	76,8729	48,4300
PatchTST_SSL	10969,6396	30471,2168	12094,0273	41,1768	68,6279	43,2356
SARIMA	390,0000	1083,3334	429,9750	3,5000	5,8333	3,6750
CNN1D	186,3712	517,6976	205,4742	7,2926	12,1543	7,6572
LSTM	12455,8730	33882,3125	13573,5225	44,2101	73,0095	46,0674
DLinear	6589,9624	18305,4473	7265,4321	27,9510	46,5850	29,3485
Ensemble_CNN_LSTM	3384,2261	9234,8252	3690,0803	23,8211	39,4732	24,7751
Naive	50985,0078	141625,0000	56210,9609	69,5000	115,8333	72,9750
SNaive	390,0000	1083,3334	429,9750	3,5000	5,8333	3,6750

Tabela 5: Resultados multivariados das métricas de erro por fatia de rede para o horizonte de previsão de 12 horas.

Modelo	MSE			MAE		
	eMBB	**URLLC**	**mMTC**	**eMBB**	**URLLC**	**mMTC**
PatchTST_Original	17789,1523	49414,2969	19612,5371	50,4967	84,1612	53,0215
PatchTST_A_small	18581,4023	51614,9922	20485,9922	59,3538	98,9229	62,3214
PatchTST_AB_revin	18588,6465	51635,1172	20493,9785	59,8380	99,7300	62,8299
PatchTST_ABC_sincos	18588,6465	51635,1172	20493,9785	59,8380	99,7300	62,8299
PatchTST_SSL	20045,8418	55682,8867	22100,5352	62,5741	104,2901	65,7028
SARIMA	535,7143	1488,0952	590,6250	4,6429	7,7381	4,8750
CNN1D	207,7425	577,0626	229,0362	7,3202	12,2003	7,6862
LSTM	13638,4492	38646,8945	15184,3809	55,2201	92,5002	58,1032
DLinear	8790,4707	24417,9688	9691,4922	36,6746	61,1243	38,5083
Ensemble_CNN_LSTM	3824,7031	10847,1816	4256,5713	28,8928	48,5578	30,3979
Naive	65732,1562	182589,2812	72469,6875	90,0000	150,0000	94,5000
SNaive	535,7143	1488,0952	590,6250	4,6429	7,7381	4,8750

Tabela 6: Resultados multivariados das métricas de erro por fatia de rede para o horizonte de previsão de 24 horas.

Ao aplicar a função de inversão de escala para restaurar as previsões à unidade física original, o erro residual é multiplicado linearmente pelo desvio padrão (σ) para a obtenção do MAE, e de forma quadrática pela variância (σ^2) para o cálculo do MSE. Esse efeito multiplicador é matematicamente comprovado pela razão de dispersão entre as fatias URLLC e eMBB: a razão $\sigma_{URLLC}/\sigma_{eMBB}$ resulta na constante 1,6666, que é rigorosamente equivalente à razão entre os erros MAE obtidos pelo modelo líder CNN 1D para H = 12 (12,1543/7,2926 = 1,6666). De modo correspondente, a razão da variância quadrática entre esses canais ($\sigma^2_{URLLC}/\sigma^2_{eMBB} \approx 2,7777$) reflete-se na razão de seus erros MSE (517,6978/186,3712 $\approx$ 2,777). Sob a perspectiva de engenharia de tráfego, esse fenômeno evidencia que desvios residuais idênticos no espaço latente de treinamento traduzem-se em um volume de solicitações de rádio não-atendidas muito maior na classe URLLC, justificando o monitoramento proativo individualizado por canal para mitigar riscos de congestionamento físico.

Para além da fundamentação simplesmente estatística, segmentação das métricas físicas permite classificar as 12 configurações desenvolvidas em três grupos distintos de viabilidade operacional perante as restrições físicas do núcleo móvel. O grupo de alta resiliência, encabeçado pelo modelo CNN 1D e secundado pelos *baselines* SARIMA e SNaive, apresenta erros residuais extremamente contidos em todas as fatias lógicas para ambos os horizontes. Ao conter o desvio absoluto médio (MAE) na fatia URLLC em uma faixa de rádio tolerável de apenas 12,15 a 12,20 solicitações/hora (frente aos platôs físicos de pico de 1300 solicitações), a CNN1D operando sob independência de canais consolida-se como o único modelo neural do estudo apto a orientar tomadas de decisão proativas seguras.

Em seguida, o grupo de desempenho intermediário, composto pelo *Ensemble* CNN-LSTM e pelo modelo DLinear, exibe erros de escala física moderados nas três fatias, com o DLinear assinalando 46,5850 e o *Ensemble* registrando 39,4732 para o MAE_{URLLC} no horizonte de curto prazo. Embora substancialmente melhores do que as redes recorrentes e de atenção pura, a incapacidade de o DLinear modelar transições rápidas não lineares e a herança parcial do atraso de fase na fusão tardia do Ensemble geram erros de amplitude sistemáticos ao redor dos pontos de transição de tráfego, limitando a agilidade operacional dessas arquiteturas. Por fim, o grupo que demonstrou baixa capacidade preditiva, caracterizado pelas variantes do PatchTST e pelos *baselines* LSTM e Naive, assinala desvios de escala físico catastróficos em todos os canais lógicos, o que inviabiliza por completo sua aplicação prática.

As consequências reais da implantação dessas arquiteturas de baixo desempenho e alto erro preditivo traduzem-se em sérias violações de qualidade de serviço no plano de controle do sistema de telecomunicação. Na fatia URLLC, que impõe metas de confiabilidade de sub-milissegundos para aplicações industriais e médicas críticas [2], previsões severamente subestimadas provocadas por atrasos de fase ou atenuações de pico (como as sofridas pelo PatchTST Original ou pela LSTM conjunta) impedem que a função de controle de políticas (PCF) e o gerenciador de sessões (SMF) disparem a alocação proativa de blocos de recursos físicos (PRBs) nas antenas gNodeB de forma síncrona aos picos reais de demanda. Esse atraso proativo força o enfileiramento de pacotes críticos nos buffers de transmissão, excedendo o orçamento de atraso de pacotes definido e resultando na violação de SLA.

De forma antagônica, previsões superestimadas geradas por modelos de projeção excessivamente suavizados (como no DLinear) induzem o orquestrador ao sobredimensionamento ocioso de recursos de rede. Essa alocação indevida de capacidade espectral para fatias altamente tolerantes a atrasos, como a mMTC (IoT massivo), ocorre em detrimento da vazão de dados (do inglês, *throughput*) das fatias de rádio concorrentes de banda larga móvel (eMBB), as quais sofrem com a privação desnecessária de largura de banda.

5.3 Análise do Estudo de Ablação e Pré-treinamento SSL

A avaliação empírica do estimador base literário `PatchTST_Original` evidenciou limitações severas na sua capacidade de generalização temporal, causadas por um desajuste estrutural entre a complexidade paramétrica do modelo e a escala amostral disponível. Configurado com a sua arquitetura padrão de alta capacidade, caracterizada pelos parâmetros `d_model` $= 64$, `n_layers` $= 2$ e quatro cabeças de atenção, sob uma taxa de descarte de 0,1, o modelo dispõe de um volume excessivo de pesos livres frente à regularidade estatística do sinal de telemetria do *DeepSlice*, que conta com apenas 117 horas contínuas na partição de treinamento.

Esse desequilíbrio fez com que as cabeças de autoatenção global se mostrassem altamente sensíveis a pequenas perturbações locais, memorizando ruídos de alta frequência do tráfego histórico em vez de abstrair os ciclos sazonais macro das fatias. Como consequência direta dessa convergência

para mínimos locais sobreajustados, a projeção temporal na partição de teste cronológica colapsou, registrando erros globais extremamente elevados, com um MSE de 8368,4609 e MAE de 34,4368 para o horizonte de curto prazo H = 12, progredindo para um erro ainda mais expressivo de 28938,6621 e desvio de 62,5598 no cenário estendido H = 24. Esses desvios confirmam empiricamente os limites de generalização em amostras finitas previstos pelo *finite-sample gap* teórico em *Transformers* de alta capacidade aplicados a séries sazonais estáveis de rádio acesso celular.

Para atenuar essa memorização de ruído, a corrida experimental seguinte, denominada `PatchTST_A_small`, introduziu uma regularização extrema de capacidade arquitetural via subparametrização, reduzindo o grafo computacional para as especificações mínimas de `d_model` = 16, `n_layers` = 1 e apenas duas cabeças de atenção, com a taxa de regularização de *dropout* elevada para 0,3. No entanto, os resultados na escala física real revelaram um aparente paradoxo de engenharia de dados, em que a contenção da capacidade paramétrica provocou uma explosão drástica nos erros de previsão, fazendo com que o MSE global saltasse para 22167,8672 e o MAE atingisse 55,4792 sob o horizonte H = 12, enquanto para o horizonte H = 24 os erros se fixaram em 30227,4629 e 73,5327. Esse colapso decorre diretamente da perda de expressividade não-linear sofrida pelo codificador minimalista, que se tornou matematicamente incapaz de mapear as transições abruptas e instantâneas a zero registradas ciclicamente no comportamento físico do tráfego. Em vez de rastrear e estimar o vale nulo que caracteriza a inatividade periódica da rede celular, o estimador encolhido gerou uma projeção linear amortecida e suavizada que oscilou continuamente ao redor de 400 solicitações por hora, incorrendo em penalizações quadráticas residuais massivas nos instantes de transição do tipo degrau. Essa limitação estrutural e o comportamento suavizado das trajetórias preditas ao longo dos canais lógicos de serviço são apresentados e corroborados visualmente pelo mapeamento de curvas das Figuras 16 e 20.

Dando prosseguimento ao estudo de ablação de forma cumulativa, as execuções das corridas `PatchTST_AB_revin` e `PatchTST_ABC_sincos` trouxeram uma constatação de grande interesse prático no plano do desenvolvimento de *software* preditivo. Observa-se um empate matemático absoluto e invariável até a quarta casa decimal nas métricas dessas duas ablações em ambos os horizontes preditivos, registrando um MSE = 20511,9727 e MAE = 57,1422 sob o horizonte de curto prazo H = 12, e convergindo para um erro de 30239,2461 e desvio de 74,1327 no horizonte estendido H = 24. O motivo desse comportamento idêntico de pesos e gradientes reside na estrutura computacional profunda do ecossistema do *Hugging Face*, especificamente na forma como a classe de parametrização `PatchTSTConfig` trata as definições de posição do sinal. Por padrão, se o parâmetro `positional_encoding_type` é mantido nulo, a API invoca e adiciona automaticamente coordenadas posicionais trigonométricas estáticas do tipo seno-cosseno durante preparação dos tensores. Como consequência, ao instanciar a Ablação B com a normalização RevIN ativada e omitindo o tipo de codificação posicional, e na sequência instanciar a Ablação C explicitando `positional_encoding_type` como `'sincos'`, o *framework* instanciou exatamente o mesmo grafo computacional no PyTorch. Sob a inicialização estrita da semente 42, os otimizadores seguiram caminhos de retropropagação idênticos sobre os mesmos tensores de entrada, resultando em previsões perfeitamente coincidentes.

Por outro lado, a investigação do comportamento do estimador pré-treinado de forma autosupervisionada, identificado como `PatchTST_SSL`, comprovou a eficácia prática de estratégias de representação por mascaramento sobre horizontes de curto prazo. No horizonte de previsão H = 12, a variante de pré-treino conquistou a liderança de desempenho preditivo de todo o grupo de modelos baseados em atenção, obtendo o menor MSE de 17844,9609 e MAE de 51,0134. Esse avanço preditivo demonstra que, ao mascarar aleatoriamente 50% dos *patches* de sinal na primeira fase e forçar o codificador do *Transformer* a reconstruir os blocos omitidos com base apenas nos dados remanescentes, a rede aprendeu representações latentes ricas sobre a ciclicidade diurna das

fatias lógicas. Esse processo de reconstrução auto-supervisionada atuou como um pré-condicionador de pesos de alta eficiência que capturou as correlações cronológicas locais sem a interferência de rótulos supervisionados de destino. Consequentemente, o *fine-tuning* subsequente herdou um mapa de características extremamente sintonizado, facilitando a convergência para trajetórias preditivas mais limpas e geometricamente consistentes perante as curvas reais de tráfego.

Entretanto, a superioridade observada no horizonte de curto prazo sofreu um colapso preditivo severo quando o modelo `PatchTST_SSL` foi submetido ao Teste 2, registrando o pior erro absoluto de todo o estudo comparativo de ablação, com MSE global 32609,7539 e MAE de 77,5223. explicação matemática e física para essa degradação drástica reside no fenômeno de *overfitting* de reconstrução que assola modelos de atenção profunda sob restrições amostrais severas de dados brutos. O processo de pré-treinamento por auto-supervisão executado por 50 épocas contínuas sobre uma partição de treino exígua de apenas 117 horas fez com que o codificador do *Transformer* memorizasse os ruídos e as particularidades locais do sinal de treino.

Ao realizar a projeção direta de múltiplos passos de um horizonte longo de 24 horas, essa representação latente rígida e sobreajustada perdeu a sua flexibilidade de extrapolação temporal, forçando o mecanismo de atenção a projetar oscilações de alta frequência artificiais e quedas inexistentes (do inglês, *wiggling*) ao longo do conjunto de teste. Conforme ilustrado detalhadamente na Figura 23, essas alucinações de ondas espúrias desalinham completamente as trajetórias projetadas frente à linearidade estável e plana do tráfego celular real, destruindo a generalização estatística do estimador e inviabilizando completamente o seu emprego prático no plano de controle do ecossistema móvel 5G.

5.4 O Fenômeno da Rigidez Temporal e a Eficácia dos *Baselines* Leves

A acentuada disparidade de desempenho observada entre a simplicidade estrutural das abordagens convolucionais e lineares locais frente ao colapso preditivo sofrido pelas variantes de atenção global do PatchTST exige uma investigação teórica rigorosa. Esse cenário empírico não constitui uma inconsistência de modelagem, mas reflete diretamente as propriedades estatísticas do processo gerador de dados estabelecido pelo simulador *DeepSlice*. O tráfego de plano de controle simulado nas fatias lógicas de rádio é caracterizado por um regime de baixíssima entropia. Diferente das redes celulares operacionais reais, cujas séries temporais de telemetria são permeadas por dinâmicas caóticas, desvios de tendência de longo prazo e picos de rajadas estocásticos imprevisíveis, a infraestrutura simulada exibe padrões cíclicos diários rígidos e quase perfeitamente determinísticos.

Sob essa rigidez temporal extrema, os modelos univariados de persistência diária e passeio aleatório sazonal encontram as condições matemáticas ideais para operar com alta acurácia. Uma vez que o comportamento dinâmico futuro do tráfego constitui uma réplica quase linear exata dos eventos ocorridos no período de 24 horas anterior, o esforço de mapeamento de características torna-se mínimo. Isso favorece estimadores locais que exploram diretamente a autocorrelação de curto prazo e a memória periódica, conforme visto na seção 5.1 deste trabalho.

Esse sucesso empírico das abordagens lineares e locais sobre os modelos profundos de autoatenção é matematicamente respaldado pela teoria denominada *In-Context Learning* (ICL) em séries temporais desenvolvida por Zhou et al. [14], segundo a qual o limite inferior de erro preditivo esperado de um modelo de autoatenção linear é estruturalmente superior ao do preditor linear ótimo por Mínimos Quadrados Ordinários (OLS).

Adicionalmente, a própria engenharia do mecanismo de autoatenção impõe barreiras ao processamento de séries temporais regulares de baixa ordem. O operador de autoatenção global foi concebido originalmente como um compressor semântico ideal para capturar dependências abstratas e de longo alcance em textos [6]. Contudo, ao lidar com trajetórias numéricas cronológicas rígidas,

essa captação irrestrita de similaridade bidimensional dispersa a entropia dos pesos de atenção ao longo do tempo, diluindo os sinais locais essenciais para a previsão proativa [23].

Ao tentar extrair padrões locais em uma base de dados compacta e estável, o mecanismo de autoatenção irrestrito do PatchTST Original é induzido a um severo *overfitting*, interpretando variações ruidosas locais como periodicidades e gerando oscilações de alta frequência espúrias ao longo do horizonte estendido de teste [13]. O excesso de graus de liberdade paramétricos do *Transformer* atua, portanto, deteriorando a capacidade de generalização do sistema frente a trajetórias essencialmente planas de tráfego.

Em franco contraste a esse colapso por dispersão temporal, a arquitetura convolucional compacta CNN 1D estabelece-se como o motor ideal para prever prever o comportamento da rede e automatizar otimizações no cenário fornecido pelo *dataset* utilizado neste projeto. Ao restringir o seu aprendizado ao campo receptivo local de seus filtros convolucionais sob o regime de independência de canais, a rede atua como um filtro espacial altamente regularizado. Ela preserva implicitamente a ordem cronológica local e rastreia o comportamento de transição rápida tipo degrau no passo t = 2 de forma direta, livre da propagação de erros de fase que afetam as fatias de missão crítica do núcleo móvel 5G.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou uma avaliação comparativa de desempenho (*benchmarking*) sistemática e rigorosa entre diferentes arquiteturas de *Deep Learning* e métodos estatísticos aplicados à previsão proativa de séries temporais de tráfego no contexto de *Network Slicing* para redes móveis 5G. O principal objetivo de engenharia consistiu em analisar se a elevada complexidade computacional e o custo representacional de modelos baseados em atenção global e no estado da arte, como o PatchTST, traduziam-se em ganhos de acurácia preditiva que justificassem a sua adoção no plano de controle móvel, ou se modelos localmente restritos e mais leves poderiam atender a essa demanda preditiva com maior eficácia operacional.

Os resultados empíricos consolidados na escala física real de solicitações por hora comprovaram de forma contundente a supremacia da arquitetura convolucional compacta *CNN 1D* desenvolvida sob a premissa de independência de canais. A rede convolucional estabeleceu-se como a líder incontestável de desempenho global em ambos os cenários de simulação, assinalando os menores desvios quadráticos e absolutos medidos. Essa performance cirúrgica valida a eficácia de restringir o aprendizado ao campo receptivo local dos filtros convolucionais para capturar as transições dinâmicas do tipo degrau e picos locais sem a propagação de atrasos de fase ou a contaminação de escala que afetou os modelos concorrentes.

Sob a perspectiva teórica da ciência de dados temporais, a investigação evidenciou um colapso preditivo estrutural das variantes do PatchTST, caracterizado pelo surgimento de oscilações espúrias e quedas fantasmas de tráfego nas janelas de teste de longo prazo. Esse fenômeno fornece uma sólida validação empírica para a teoria do *finite-sample gap* sob regimes de amostragem finita em modelos de atenção profunda. Sob dados temporais rígidos, cíclicos e de baixa ordem gerados pelo simulador *DeepSlice* o excesso de parâmetros livres do *Transformer* induz ao *overfitting* de alta variância, interpretando pequenos ruídos de treino como tendências periódicas. Esse cenário favorável a preditores de baixa entropia também explica o excelente desempenho obtido pelos *baselines* sazonais SARIMA e SNaive, cujas trajetórias preditivas convergiram de forma idêntica devido à seleção automática de ordem de passeio aleatório sazonal puro pela minimização do critério de Akaike.

Do ponto de vista prático de redes e telecomunicações, a implementação de modelos de alta

acurácia preditiva e baixo tempo de inferência, como a CNN 1D analisada, constitui um requisito de viabilidade indispensável para a automação proativa da alocação de blocos de recursos físicos no núcleo de rede 5G. Previsões imprecisas ou amortecidas, como as geradas pela LSTM conjunta devido à contaminação cruzada de escalas, forçam o enfileiramento de pacotes nos *buffers* de transmissão e violam de forma catastrófica o orçamento de atraso da classe 5QI (do inglês, *5G QoS Identifier* na sensível fatia de missão crítica URLLC, ao passo que previsões superestimadas, tais quais as geradas pelo DLinear, resultam em sobredimensionamento ocioso de espectro em detrimento do *throughput* da fatia concorrente eMBB. Conclui-se, portanto, que a CNN 1D unifica o menor desvio estatístico com o menor custo computacional, qualificando-se como o motor preditivo ideal para o contexto analisado no presente trabalho.

Como perspectivas de trabalhos futuros, vislumbra-se a expansão desta pesquisa em frentes fundamentais citadas de forma breve. Primeiramente, propõe-se a avaliação e validação cruzada das hipóteses defendidas neste estudo a partir da análise de outros conjuntos de dados públicos de referência (*benchmarks*) na literatura de séries temporais, bem como outros simuladores de telemetria móvel, com o propósito de verificar se o comportamento limitante dos *Transformers* e a superioridade de modelos convolucionais compactos se corroboram sob diferentes propriedades de entropia, variabilidade e distribuições estatísticas. Adicionalmente, sugere-se a avaliação das arquiteturas sob cenários contendo dinâmicas estocásticas caóticas e desvios de tendência oriundos de telemetria de redes celulares reais físicas, testando os limites práticos do paradigma de independência de canais. Por fim, recomenda-se o desenvolvimento de algoritmos de aprendizado adaptativo *online* para atualização incremental de pesos sinápticos, garantindo que o motor preditivo se autoajuste em tempo real às variações sazonais de comportamento sem exigir rotinas periódicas de treinamento centralizadas.

Referências

[1] S. Zhang, *An Overview of Network Slicing for 5G,* IEEE Wireless Communications, **26** (3). 2019, pp. 111–117.

[2] A. Thantharate, R. Paropkari, V. Walunj and C. Beard, *DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks,* 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA. 2019, pp. 0762–0767.

[3] D. Ferreira, A. Braga Reis, C. Senna and S. Sargento, *A Forecasting Approach to Improve Control and Management for 5G Networks,* IEEE Transactions on Network and Service Management, **18** (2). 2021, pp. 1817–1831.

[4] Q. Xin, X. Wang, D. Zhang, B. Han and B. Xu, *MegaMon: Transformer-Based Network Traffic Monitor and Analysis for LLM Training,* ICC 2025 - IEEE International Conference on Communications, Montreal, QC, Canada. 2025, pp. 5356–5361.

[5] E. Lykakis, I. O. Vardiambasis and E. Kokkinos, *Data Traffic Prediction for 5G and Beyond: Emerging Trends, Challenges, and Future Directions: A Scoping Review,* Electronics, **14** (23). 2025, pp. 4611.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need,* Advances in Neural Information Processing Systems 30 (NIPS 2017). 2017, pp. 5998–6008.

[7] A. Zeng, M. Chen, L. Zhang and Q. Xu, *Are transformers effective for time series forecasting?,* Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence. 2023, pp. 1248.

[8] J. Zhang, J. Wang, W. Qiang, F. Xu, C. Zheng, F. Sun and H. Xiong, *Intriguing Properties of Positional Encoding in Time Series Forecasting,* arXiv:2404.10337. 2024.

[9] H. Irani and V. Metsis, *Positional Encoding in Transformer-Based Time Series Models: A Survey,* arXiv:2502.12370. 2026.

[10] B. Li and Y. Qian, *Weather Prediction Using CNN-LSTM for Time Series Analysis: A Case Study on Delhi Temperature Data,* arXiv:2409.09414. 2024.

[11] N. Wu, B. Green, X. Ben and S. O'Banion, *Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case,* arXiv: 2001.08317. 2020.

[12] V. Naghashi, M. Boukadoum and A. B. Diallo, *A multiscale model for multivariate time series forecasting,* Sci Rep, **15**, 1565. 2025.

[13] Y. Chen, N. Céspedes and P. Barnaghi, *A Closer Look at Transformers for Time Series Forecasting: Understanding Why They Work and Where They Struggle,* Proceedings of the 42nd International Conference on Machine Learning. 2025, pp. 7763–7780.

[14] Y. Zhou, Y. Wang, S. Goel and A. R. Zhang, *Why Do Transformers Fail to Forecast Time Series In-Context?,* arXiv: 2510.09776. 2025.

[15] Z. Chong, *Attention Is Not What You Need,* arXiv: 2512.19428. 2025.

[16] C. Su, *DLinear Makes Efficient Long-term Predictions,* Proceedings of ACM Conference (Baidu KDD CUP 2022). 2022.

[17] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory,* Neural Computation, **9** (8). 1997, pp. 1735–1780.

[18] F. A. Gers, J. Schmidhuber and F. Cummins, *Learning to Forget: Continual Prediction with LSTM,* Technical Report IDSIA-01-99, Lugano, Switzerland. 1999.

[19] R. K. Gupta, A. Ranjan, M. A. Moid and R. Misra, *Deep-Learning based Mobile-Traffic Forecasting for Resource Utilization in 5G Network Slicing,* Proceedings of the International Conference on IoT and Connected Technologies (ICIOTCT 2020). 2020.

[20] Z. Zeng, R. Kaur, S. Siddagangappa, S. Rahimi, T. Balch and M. Veloso, *Financial Time Series Forecasting using CNN and Transformer,* arXiv: 2304.04912. 2023.

[21] Y. Nie, N. H. Nguyen, P. Sinthong and J. Kalagnanam, *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers,* International Conference on Learning Representations (ICLR 2023). 2023.

[22] J. Zhang, W. Qiang, J. Wang, J. Zhou, C. Zheng and H. Xiong, *Understanding Token-level Topological Structures in Transformer-based Time Series Forecasting,* arXiv:2404.10337. 2025.

[23] J. Zhang, J. Wang, C. Sun, X. Shen, F. Xu, C. Zheng and W. Qiang, *Exploring the Role of Token in Transformer-based Time Series Forecasting,* Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2025). 2025.

[24] M. T. Lê, P. Wolinski and J. Arbel, *Efficient Neural Networks for Tiny Machine Learning: A Comprehensive Review,* arXiv:2311.11883. 2023.

[25] S. Somvanshi, M. M. Islam, G. Chhetri, R. Chakraborty, M. S. Mimi, S. A. Shuvo, K. S. Islam, S. Javed, S. A. Rafat, A. Dutta and S. Das, *From Tiny Machine Learning to Tiny Deep Learning: A Survey,* ACM Computing Surveys, **58** (7). 2025, pp. 1–33.