Estudo de Ablação em Arquiteturas Seq2Seq para Predição de Séries Temporais Sintéticas

Aluno: Alexsandro Gehlen
Disciplina: Algoritmos Baseados em Dados para Problemas de Ciência e Engenharia
Semestre: 2025.2

1. Introdução

Neste trabalho, utilizo uma série temporal sintética gerada a partir do módulo tsgen da biblioteca Foreblocks, o que permite controlar explicitamente componentes como tendência, sazonalidade e nível de ruído. A partir dessa série, são aplicadas etapas de pré-processamento (filtragem, detrending e normalização) com o objetivo de torná-la mais adequada para tarefas de previsão e reduzir efeitos indesejados de não estacionariedade.

Na sequência, é realizado ablação em arquiteturas Seq2Seq, mantendo a mesma série e o mesmo protocolo de treino, e variando componentes específicos do modelo: uso ou não de camadas de Fourier, diferentes mecanismos de atenção no Transformer (como multi-head attention densa, sliding window e probSparse) e a substituição do Transformer por um encoder–decoder baseado em LSTM. O objetivo é comparar o impacto de cada escolha arquitetural na qualidade da predição, quantificado por métricas de erro em validação, e discutir quais combinações se mostram mais adequadas para o cenário considerado.

2. Desenvolvimento

2.1 Caracterização do dataset sintético

Foi criado um dataset sintético de séries temporais, gerado utilizando o módulo tsgen da biblioteca Foreblocks. O objetivo foi construir uma série com comportamento rico e realista, combinando múltiplos componentes estatísticos e determinísticos, permitindo avaliar de forma controlada os efeitos do pré-processamento e das diferentes arquiteturas Seq2Seq. A série possui uma série temporal de 1500 pontos de frequência diária.

Na figura 1 é a série gerada, destacando visualmente a evolução temporal e a presença de seus principais padrões.

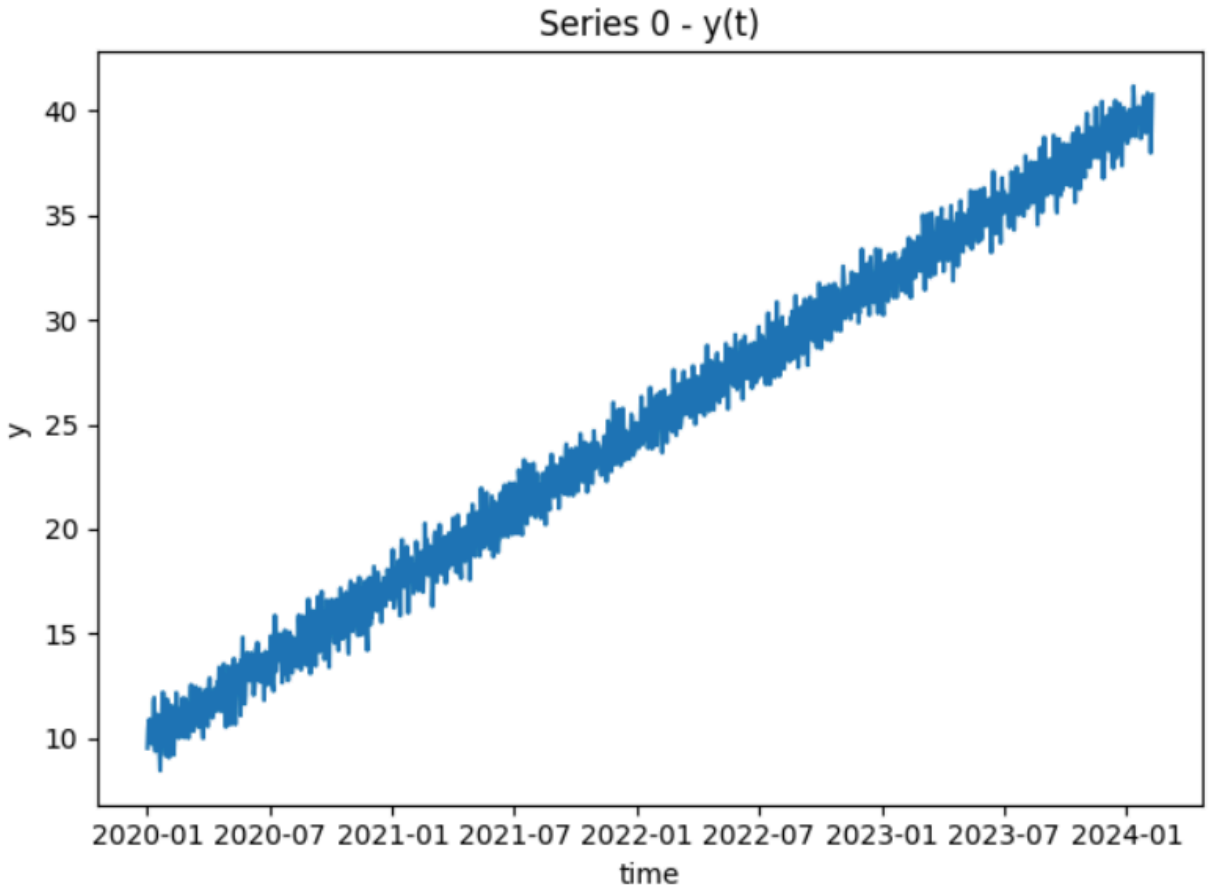

Figura 1: Série temporal sintética gerada a partir dos parâmetros definidos no módulo `tsgen`.

Na Figura 2 mostra a decomposição clássica em tendência, sazonalidade e resíduos, permitindo observar a contribuição individual de cada componente antes das etapas de filtragem e detrending. O primeiro painel mostra a tendência, representada por um crescimento linear ao longo do tempo, descrevendo o movimento de longo prazo da série. O segundo painel exibe a sazonalidade, composta por oscilações periódicas regulares neste caso, um padrão repetitivo de curta duração associado à periodicidade semanal. Por fim, o terceiro painel mostra o componente de ruído ou resíduos, que corresponde às variações irregulares não explicadas pelos componentes anteriores, incorporando a aleatoriedade intrínseca e pequenas flutuações imprevisíveis. Essa decomposição facilita a análise individual de cada componente e é fundamental para as etapas de pré-processamento e modelagem preditiva subsequentes.

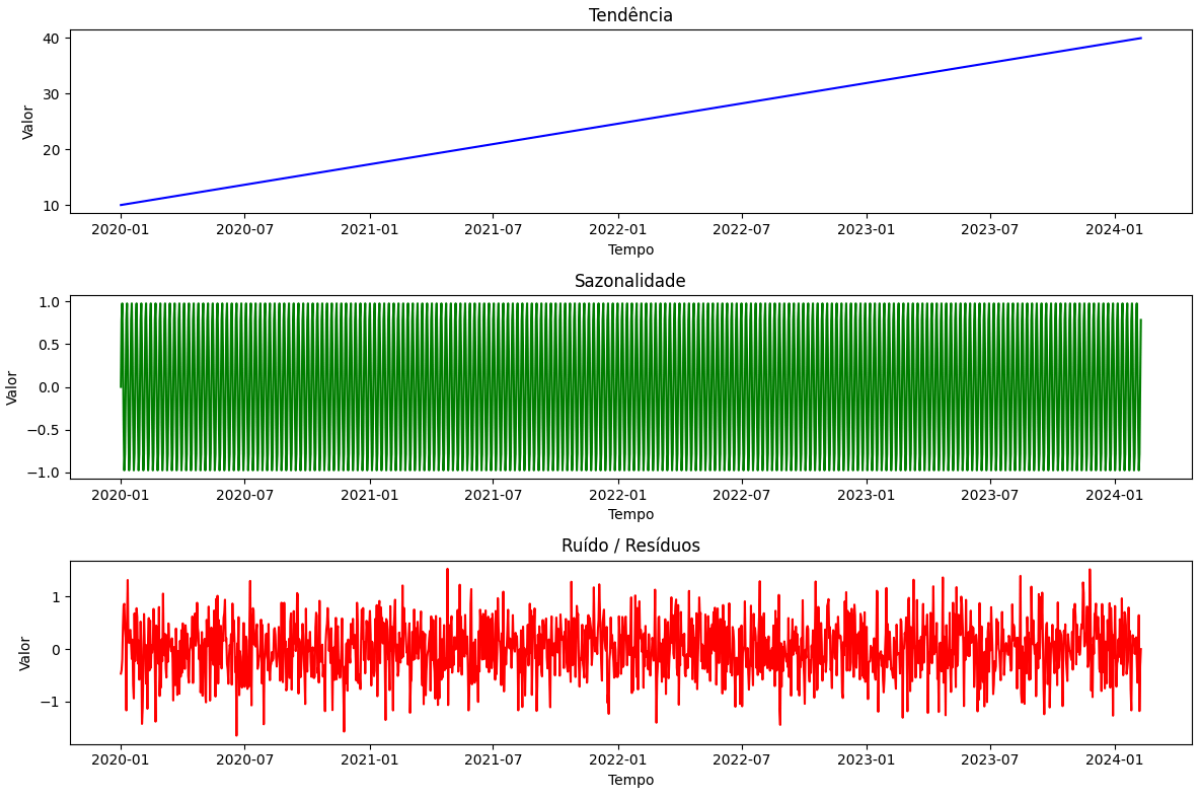

Figura 2: Decomposição da série temporal em tendência, sazonalidade e resíduos.

2.2 Preparação das janelas e divisão dos dados

Para preparar a série para os modelos Seq2Seq, foi utilizado o método model.preprocess(), que cria as janelas de entrada (X_seq) e de saída (y_seq) a partir da série original. Foram aplicadas normalização, diferenciação, remoção de tendência e EWT, e definidos dois parâmetros principais, window_size de 64 ehorizon de 12 . O modelo deve prever os próximos 12 valores. Os dados foram separados em treino e teste mantendo a ordem temporal, utilizando 80% para treino e 20% para teste. Essa preparação foi utilziada em todos os treinamentos e avaliações dos modelos.

Durante o pré-processamento, foram aplicados alguns filtros para melhorar a qualidade da série antes de formar as janelas de entrada e saída. A normalização foi utilizada para colocar os valores em uma escala mais uniforme, enquanto a diferenciação e o detrending ajudaram a remover padrões de longo prazo e reduzir a não estacionariedade. Também foi aplicada a EWT (*Empirical Wavelet Transform*), que destaca componentes frequenciais importantes e reduz ruídos. Por fim, a opção self_tune permitiu que o próprio Foreblocks ajustasse automaticamente parâmetros internos para melhorar a preparação da série. Na Figura 3 é possível ver os dados originais e tratados.

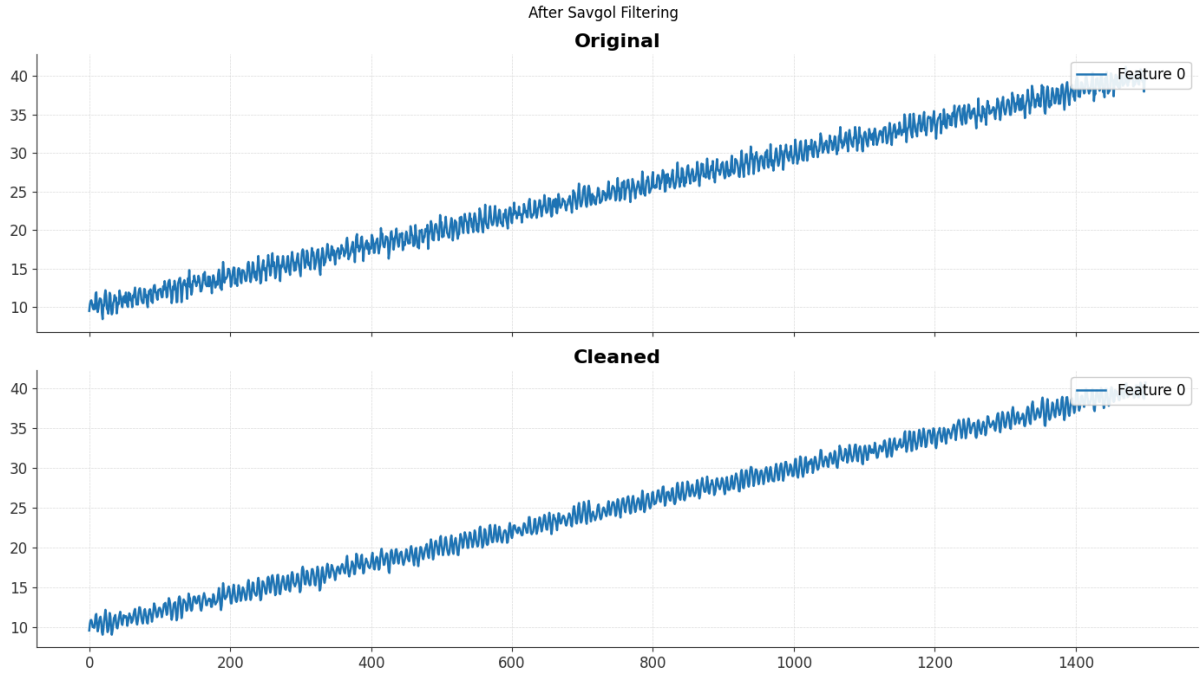

Figura 3: Dados originais e após o tratamento.

2.3 Modelos

Para iniciar a modelagem, foi definido um conjunto de parâmetros básicos utilizados como configuração padrão nos experimentos. Tanto os modelos baseados em LSTM quanto os modelos baseados em Transformers foram configurados com um tamanho de estado oculto de 64 unidades (hidden_size = 64) e com profundidade reduzida (apenas uma camada), mantendo a arquitetura leve e facilitando o treinamento. Em todos os casos, o horizonte de previsão foi fixado em 12 passos à frente (target_len = 12), de forma a comparar diretamente o desempenho das diferentes arquiteturas.

O modelo final foi construído utilizando o módulo TimeSeriesSeq2Seq, que integra de maneira modular os componentes principais: model_config, encoder, decoder, attention_module, training_config e, opcionalmente, o input_preprocessor.

Para o modelo LSTM, o encoder e o decoder são baseados em camadas recorrentes, responsáveis por processar a janela histórica e gerar a previsão futura. Quando utilizado, o pré-processador de Fourier projeta as features para um espaço frequencial antes de entrarem no encoder, com o objetivo de destacar padrões sazonais e harmônicos. O módulo de atenção, fornecido por attention_module. Esse mecanismo permite que o decoder aprenda a dar diferentes pesos aos instantes da sequência de entrada ao gerar cada ponto da previsão. No estudo de ablação, serão testadas várias variantes de atenção disponibilizadas no Foreblocks, incluindo: *dot*, *mha* (multi-head attention), *prob* (ProbSparse), *temporal*, *multiscale* e *autocorr*. Cada uma dessas alternativas possui características distintas em termos

de custo computacional e capacidade de capturar dependências temporais, permitindo avaliar como diferentes estratégias de atenção influenciam a qualidade final das previsões.

O modelo Transformer utilizado nesta implementação segue uma arquitetura encoder-only, desenvolvida sem o uso da biblioteca *foreblocks* e sem blocos adicionais de decomposição no domínio da frequência. A janela de entrada é projetada para o espaço interno do modelo e processada diretamente por camadas do TransformerEncoder, mantendo o pipeline simples, limpo e totalmente construído apenas com módulos nativos do PyTorch.

No encoder, a única forma de atenção empregada é a Multi-Head Attention (MHA) padrão das camadas Transformer, utilizada para capturar dependências temporais dentro da janela observada. Após o processamento, apenas o último estado oculto é utilizado como representação da sequência e passado a uma camada totalmente conectada que gera, de forma direta, todos os passos futuros do horizonte de previsão, resultando em uma arquitetura enxuta e eficiente para o problema.

Um componente central nas duas famílias de modelos é o módulo de atenção, fornecido por attention_module. Esse mecanismo permite que o decoder aprenda a dar diferentes pesos aos instantes da sequência de entrada ao gerar cada ponto da previsão. No estudo de ablação, serão testadas várias variantes de atenção disponibilizadas no Foreblocks, incluindo: *dot*, *mha* (multi-head attention), *prob* (ProbSparse), *temporal*, *multiscale* e *autocorr*. Cada uma dessas alternativas possui características distintas em termos de custo computacional e capacidade de capturar dependências temporais, permitindo avaliar como diferentes estratégias de atenção influenciam a qualidade final das previsões.

2.4 Resultados

Na Tabela 1 é possível comparar as redes todas com 300 épocas de treinamento. Sendo que a primeira linha contém nosso modelo base de lstm e a partir dele são realizados incrementos e variações.

Tabela 1 - Compilação da configuração e resultados dos modelos.

Base	Enc-Dec o	Attention	Fourier	MSE	RMSE	Melhoria (%) MSE	Melhoria (%) RMSE
lstm	lstmenc-dec	Não	Não	0.4569	0.6759	Base	Base
lstm	lstmenc-dec	Não	Sim	0.2828	0.5318	38,11	21,31
lstm	lstmenc-dec	MHA	Não	0.4352	0.6597	4,74	2,40
lstm	lstmenc-dec	MHA	Sim	0.2651	0.5149	41,97	23,84
lstm	lstmenc-dec	Prob	Sim	0.2666	0.5163	41,64	23,63
lstm	lstmenc-dec	Multiscal e	Sim	0.2735	0.5229	40,08	22,76
Transfor mer	transform er-enc	MHA	Sim	0.2725	0.5220	40,35	22,76
Transfor mer	transform er-enc	MHA	Não	0.2703	0.5199	40,84	23,08

Na sequência de Figuras 4, 5, 6, 7, 8, 9, 10 e 11 com o resultados dos testes(um recorte de 300 pontos).

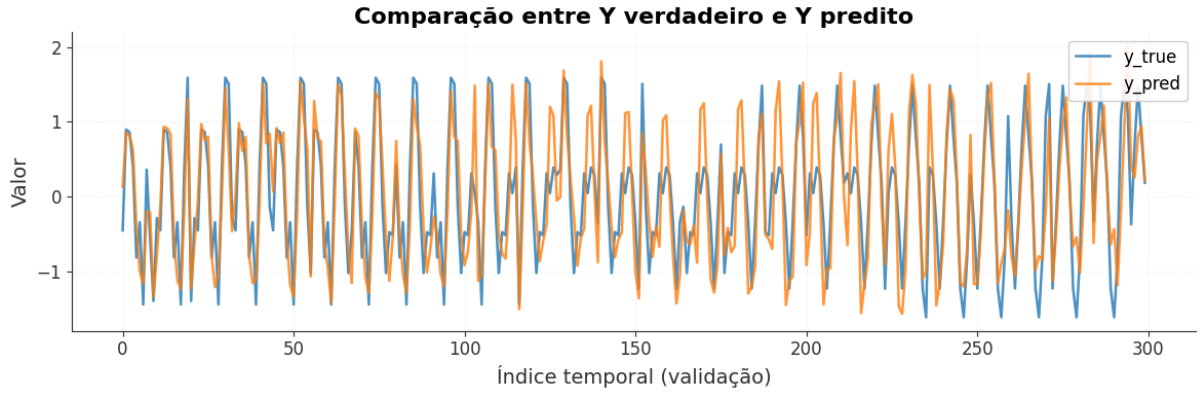

Figura 4: Desempenho do modelo base LSTM, sem atenção e sem Fourier, utilizado como referência para comparação nos experimentos.

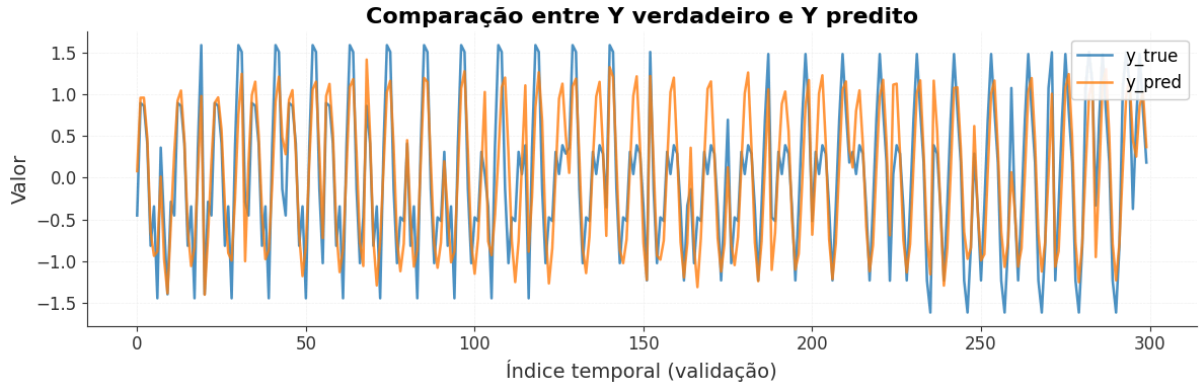

Figura 5: Desempenho do modelo LSTM com camada de Fourier.

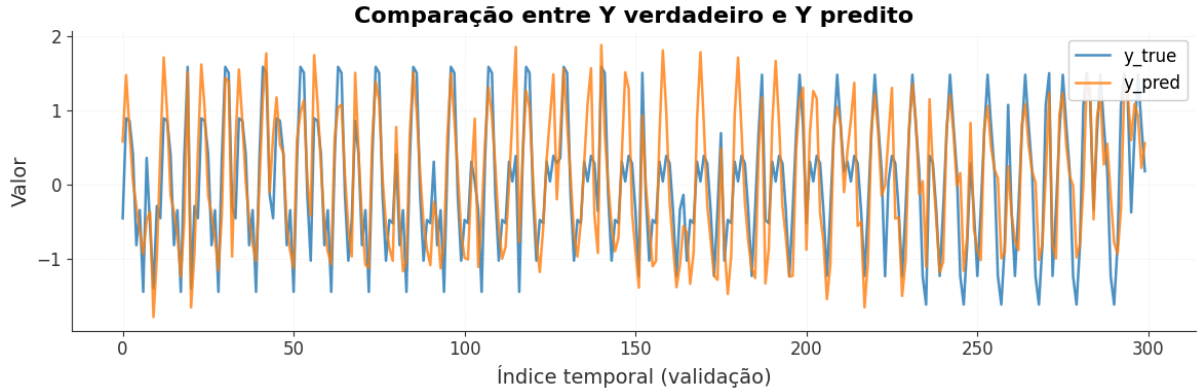

Figura 6: Desempenho do modelo LSTM com camada de Attention MHA.

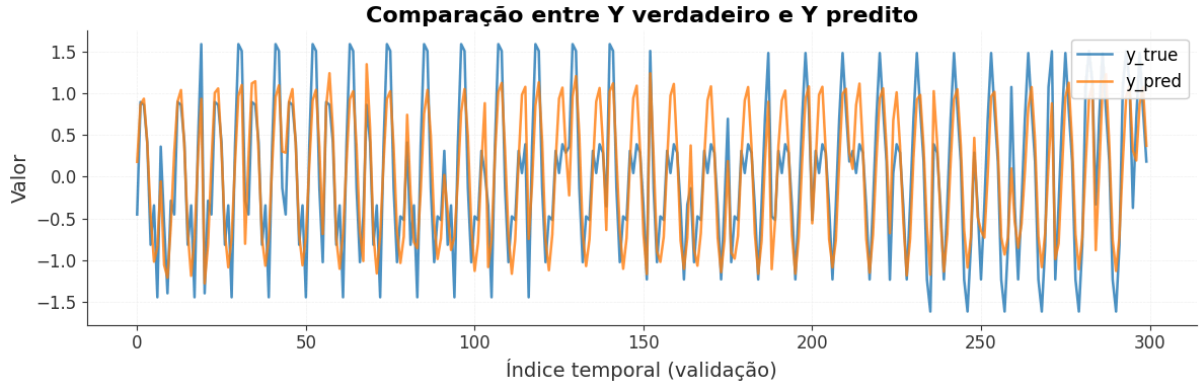

Figura 7: Desempenho do modelo LSTM com camada de Attention MHA e Fourier.

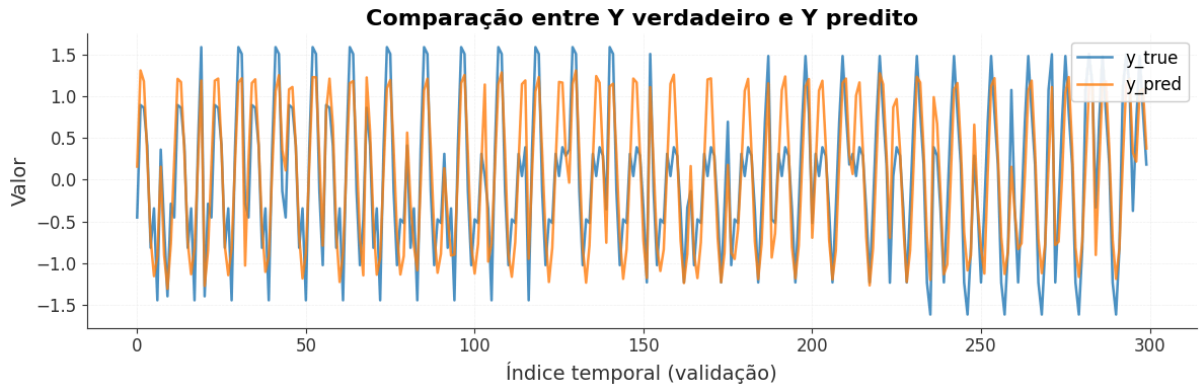

Figura 8: Desempenho do modelo LSTM com camada de Attention Prob e Fourier.

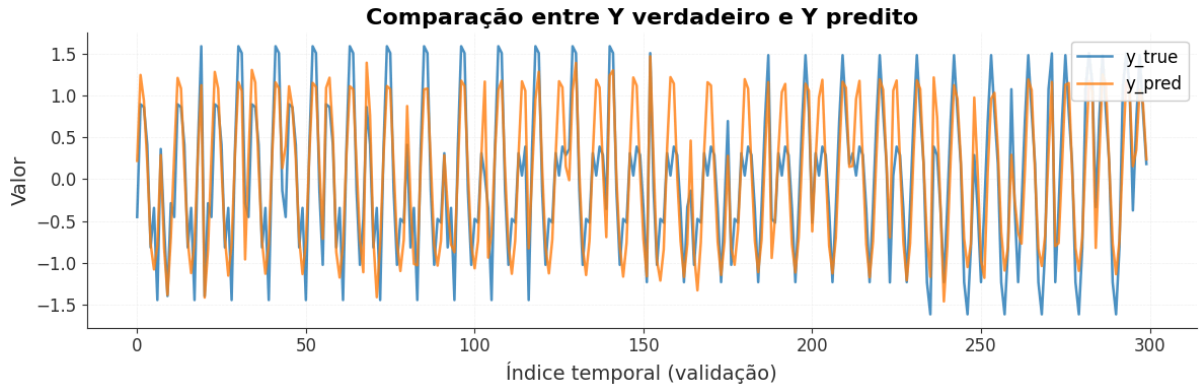

Figura 9: Desempenho do modelo LSTM com camada de Attention Multiscale e Fourier.

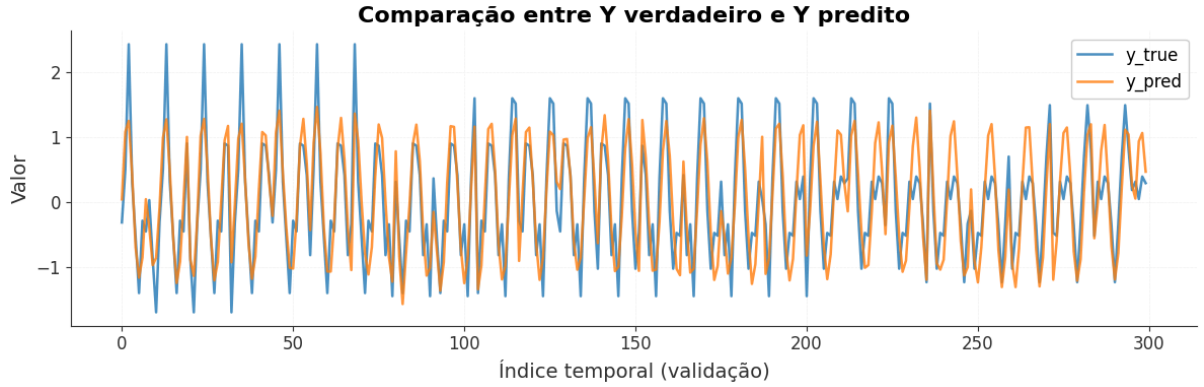

Figura 10: Desempenho do modelo Transformer com camada de MHA e Fourier.

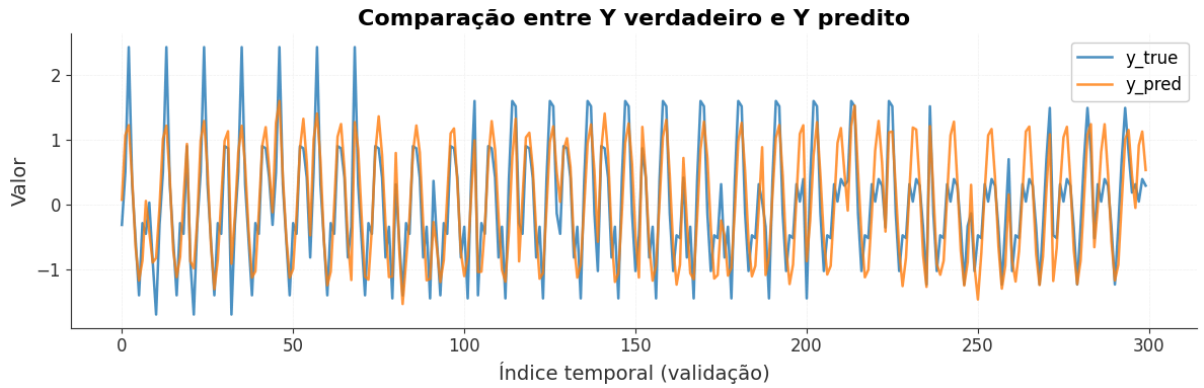

Figura 11: Desempenho do modelo Transformer com camada de MHA.

3. Conclusão

Ao longo dos experimentos, os scripts base de LSTM e Transformer foram sendo ajustados gradualmente para incorporar diferentes blocos e configurações, permitindo avaliar de forma isolada o impacto de cada componente. A arquitetura inicial LSTM encoder–decoder serviu como ponto de partida, apresentando desempenho moderado e estabelecendo a referência para comparação. As melhorias mais significativas ocorreram com a ativação do bloco Fourier, que reduziu o erro entre 38% e 42% em MSE e entre 21% e 24% em RMSE, indicando que a filtragem de baixa frequência fornece um sinal mais limpo e consistente para o modelo aprender.

A inclusão de mecanismos de atenção como MHA padrão, ProbSparse e Multiscale também foi testada nas versões modificadas dos scripts, produzindo ganhos adicionais, porém menores do que os obtidos pelo Fourier isoladamente. Já o modelo Transformer encoder-only, também ajustado ao longo dos testes, demonstrou desempenho competitivo mesmo sem decoder autoregressivo, com melhorias próximas de 40% em MSE e 23% em RMSE, tanto com Fourier quanto sem Fourier. Isso evidencia que, após as modificações estruturais aplicadas durante os experimentos, o Transformer se mostrou uma solução eficiente e consistente para previsão multistep direta.