Neurocomputing 677 (2026) 133140

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Compact transformer variants for synthetic time series forecasting

Ali Forootani [a,b,*], Mohammad Khosravi [c]

[a] Max Planck Institute of Geoanthropology, Kahlaische Str. 10, Jena, 07745, Germany
[b] Helmholtz Center for Environmental Research–UFZ, Permoserstraße 15, Leipzig, 04318, Germany
[c] Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, the Netherlands

HIGHLIGHTS

- Unified framework for `Autoformer`, `Informer`, and `PatchTST` variants.
- Theoretical bounds on noise reduction and attention complexity.
- 1500 experiments reveal robustness trade-offs under noise.
- `PatchTST` Standard excels in both clean and noisy regimes.
- Lightweight modular Transformers for scientific forecasting.

ARTICLE INFO

Communicated by F.G. Guimaraes

Keywords:
Time series forecasting
Transformer models
`Autoformer`
`Informer`
`PatchTST`
Koopman operator

ABSTRACT

This paper presents a unified and systematic study of compact Transformer architectures for time series forecasting. We introduce a modular framework that standardizes three widely used Transformer families—`Autoformer`, `Informer`, and `PatchTST`—into three principled architectural variants: *Minimal*, *Standard*, and *Full*, enabling controlled analysis of model capacity, inductive bias, and computational complexity. For each family, we provide consistent mathematical formulations, layer-wise descriptions, and end-to-end complexity characterizations.

We conduct over 1500 controlled experiments on ten synthetic time series under varying patch lengths, forecast horizons, and noise levels. The results reveal clear and reproducible performance regimes: *`PatchTST Standard`* achieves the best overall accuracy and noise robustness, *`Autoformer`* variants excel on smooth and trend-dominated signals, and *`Informer`* variants exhibit sensitivity to noise and long horizons despite improved scalability. Complementing the empirical analysis, we derive new theoretical results that quantify noise attenuation, bias–variance trade-offs, and approximation–complexity guarantees specific to each architectural family.

Finally, we demonstrate that these compact Transformer variants serve as effective and interpretable temporal encoders within an operator–theoretic forecasting framework. By embedding `Autoformer`, `Informer`, and `PatchTST` backbones into a Koopman-based latent dynamics model, we extend their applicability beyond synthetic benchmarks to real-world climate, cryptocurrency and electricity generation time series. Together, these results position compact, modular Transformers as scalable and theoretically grounded building blocks for scientific time series forecasting.

1. Introduction

Time series forecasting is a fundamental task in various domains including energy systems [1,2], finance [3], supply chain management [4], healthcare [5], meteorology [6], and more. Accurate forecasting enables proactive decision-making, risk mitigation, and optimized planning. With the increasing availability of high-resolution temporal data, designing robust and scalable forecasting models has become more critical than ever. Traditional statistical methods such as Autoregressive Integrated Moving Average (`ARIMA`), Exponential Smoothing, and Seasonal Decomposition have long served as the backbone of time series analysis [7]. However, their capacity to model complex patterns, multivariate dependencies, and non-linear dynamics

* Corresponding author at: Max Planck Institute of Geoanthropology, Kahlaische Str. 10, Jena, 07745, Germany.
Email addresses: aliforootani@ieee.org, forootani@gea.mpg.de, ali.forootani@ufz.de (A. Forootani), mohammad.khosravi@tudelft.nl (M. Khosravi).

https://doi.org/10.1016/j.neucom.2026.133140
Received 16 December 2025; Received in revised form 4 February 2026; Accepted 20 February 2026
Available online 21 February 2026

is inherently limited, particularly in large-scale or non-stationary datasets.

In recent years, machine learning (ML) and deep learning (DL) models have emerged as powerful alternatives to classical methods due to their data-driven learning capacity and flexibility. Recurrent Neural Networks (RNN s), Long Short-Term Memory (LSTM) networks, and Temporal Convolutional Networks (TCN s) have been successfully applied to time series forecasting tasks, demonstrating superior performance in capturing temporal dependencies and non-linear patterns [8]. Nevertheless, these models often suffer from challenges such as vanishing gradients, sequential bottlenecks, and limited scalability [9–11].

Among recent advancements in DL, Transformer models have revolutionized sequence modeling, particularly in Natural Language Processing (NLP) [12]. Introduced by Ref. [13], the Transformer architecture replaces recurrence with self-attention mechanisms, enabling parallel computation and long-range dependency modeling. Its success has quickly spread to other domains such as Computer Vision (CV) [14], speech processing [15], and more recently, time series forecasting [16]. Transformers provide an appealing framework for time series modeling due to their flexibility in handling variable-length inputs, modeling contextual dependencies, and learning rich representations [17].

However, applying Transformers directly to time series forecasting is not straightforward. Time series data differ from text or images in several key aspects. First, time series are continuous and regularly sampled, whereas text and vision data have discrete semantic tokens (e.g., words, pixels). Second, time series often exhibit strong autocorrelation, trends, and seasonalities—features that need to be explicitly modeled. Third, computational efficiency is a major concern, especially for long-term forecasting where the input sequence can be significantly large. The standard Transformer has quadratic complexity $\mathcal{O}(N^2)$ with respect to the sequence length N, making it computationally expensive for high-resolution temporal data [18].

To overcome these challenges, a variety of Transformer-based architectures have been proposed for time series forecasting. Informer [19] introduces ProbSparse attention to select important queries and reduce redundancy in self-attention computation. Autoformer [20] incorporates a decomposition block to handle trends and seasonality explicitly, while FEDformer [21] integrates Fourier-based decomposition to reduce complexity. Pyraformer [22] utilizes a pyramidal attention mechanism to capture hierarchical temporal features. Despite their differences, these models share a common goal: reducing computational cost while preserving or enhancing predictive accuracy.

Nevertheless, a recent study by Ref. [23] challenges the assumption that complex Transformer models are necessary for high-accuracy time series forecasting. The authors propose a simple linear model, DLinear, which surprisingly outperforms many sophisticated models on standard benchmarks. This raises fundamental questions about the design of deep models for time series and whether architectural complexity always translates to improved performance. In [24], the authors introduced PatchTST to enhance the model's ability to capture local and semantic dependencies through patching, and to simplify the design by leveraging *channel-independent* learning.

Unlike traditional Transformer variants that treat each time step as a token, PatchTST argues that aggregating multiple time steps into local patches better reflects the temporal structure of time series data. This approach is inspired by techniques in Vision Transformers (ViT) [25] and speech models such as Wav2Vec 2.0 [26], where patches or segments are used to encode meaningful context.

Furthermore, in multivariate time series, the common practice is to mix channels—i.e., combine features from all variables into a single token. While this enables joint learning, it may obscure individual patterns and increase overfitting risks. In PatchTST a *channel-independent* formulation has been adopted, where each channel (i.e., time series variable) is treated separately during embedding and encoding. This strategy has been shown to work well in Convolutional Neural Network (CNN) and linear models [23,27], and has been extended in PatchTST for Transformer-based models.

Even though Transformer-based architectures have revolutionized sequential modeling in natural language processing and computer vision, yet their adoption in time series forecasting remains underdeveloped. Unlike textual data, time series are often governed by latent temporal dynamics, periodic structures, and long-range dependencies that challenge standard attention mechanisms. Moreover, existing Transformer variants such as Autoformer, Informer, and PatchTST have been proposed independently, each introducing unique architectural innovations—but without a unified framework for comparison. This lack of standardization hinders practical model selection, reproducibility, and theoretical understanding. To address this gap, we conduct a systematic study of these models, introduce consistent architectural variants, and evaluate them under controlled experimental settings. Our goal is to clarify the design trade-offs, computational properties, and forecasting behaviors of Transformer models, ultimately guiding practitioners toward more informed and effective forecasting solutions.

(i) *Architectural variant framework.* We design and evaluate three principled variant classes—*Minimal*, *Standard*, and *Full*—for each of the Autoformer, Informer, and PatchTST families. These variants represent increasing architectural complexity, ranging from lightweight encoder-only models to full encoder–decoder configurations with autoregressive decoding. This design space allows us to isolate the impact of architectural choices on performance, scalability, and robustness.

(ii) *Unified forecasting and complexity formulation.* All variants are analyzed within a shared theoretical framework, encompassing patch-based input encoding, trend-seasonal decomposition (Autoformer), ProbSparse attention (Informer), and temporal patch tokenization (PatchTST). We provide consistent mathematical formulations, computational complexity comparisons, and layer-by-layer forward pass descriptions that contextualize how each model processes temporal inputs.

(iii) *Comprehensive empirical benchmarking under noise.* We conduct 1500 controlled experiments (750 clean + 750 noisy) across 10 synthetic signals, 5 patch lengths, and 5 forecast horizons for each model family. Our analysis reveals clear and consistent trends: PatchTST Standard achieves the best overall performance under clean and noisy conditions; Autoformer Minimal and Autoformer Standard excel on smooth and trend-dominated signals, especially in noisy regimes; while Informer variants exhibit higher error and instability under noise and long-horizon forecasting. Aggregated heatmaps and per-signal tables highlight the trade-offs between robustness, sensitivity, and generalization across the architectural spectrum.

(iv) *New theoretical guarantees tailored to each family.* We formalize noise-robustness, bias–variance trade-offs, and approximation/complexity bounds:

- Autoformer: Moving-average trend extraction attenuates white/sub-Gaussian noise by a factor $1/k$ in per-time variance, with an explicit Lipschitz-bias scaling $O(L^2k^2)$—quantifying the robustness–fidelity trade-off in the kernel size k.
- PatchTST: Non-overlapping patch means reduce noise variance as σ^2/p with an explicit MSE bound $\mathbb{E}(z_j - s_{c_j})^2 \le \frac{1}{16}L^2p^2 + \sigma^2/p$ for odd p, making the bias–variance trade-off in patch length p explicit.
- Informer (ProbSparse): Truncating to top-u keys under a score margin Δ yields an attention output error $\|y - \tilde{y}\| \le 2V_{\max}\frac{L-u}{u}e^{-\Delta}$ and, under reliable top-u recovery with $u = O(\log L_K)$, total attention cost $O(L_Q \log L_K)$ with high probability. These results quantify both approximation and sublinear complexity.

Building on compact Transformer variants (Autoformer, Informer, and PatchTST) and their Minimal, Standard, and Full configurations, we explicitly embed these lightweight yet expressive backbones into a larger operator–theoretic architecture, namely DeepKoopFormer. Rather than treating compact Transformers as stand-alone forecasters, we use

them as structured encoders that lift high-dimensional time series into a latent space in which a Koopman operator governs temporal evolution. This integration allows the efficiency, noise robustness, and inductive biases of compact Transformers to be preserved while introducing a physically meaningful and spectrally controlled latent dynamics model. We validate this hybrid design on two challenging real-world domains—climate forecasting using CMIP6 [28],[1] a financial time series dataset[2] for cryptocurrency market analysis, and electricity generation[3] time series—demonstrating that compact Transformer backbones, when coupled with Koopman dynamics, scale effectively from controlled synthetic benchmarks to complex, high-dimensional environmental and energy systems.

Together, these benchmarks provide a rigorous and heterogeneous testbed for evaluating the robustness, scalability, and cross-domain generalization capabilities of the proposed models.

Paper organization and roadmap. The paper is structured to provide a coherent progression from model formulation to theoretical analysis and empirical validation. In Section 2, we introduce the forecasting setup and notation, establishing a common foundation for all subsequent developments. Sections 3, Section 4, and Section 5 describe the proposed compact Transformer variants for time series forecasting, namely `PatchTST`, `Informer`, and `Autoformer`, together with their architectural design principles and computational complexity analyses. Section 6 presents theoretical results on noise robustness, approximation properties, and scalability behavior for each architecture. In Section 7, we evaluate the proposed methods through extensive experiments on synthetic benchmarks as well as real-world climate (CMIP6), cryptocurrency and electricity time series using Koopman-enhanced architectures, highlighting key performance trends and robustness characteristics. Finally, Section 8 concludes the paper and outlines directions for future work.

2. Preliminaries and notation

We begin by formalizing the forecasting setup and introducing common notation and operations used throughout this work.

Time series data. Let $x = [x_1, x_2, \dots, x_T] \in \mathbb{R}^T$ denote a univariate time series of length T, where $x_t \in \mathbb{R}$ represents the observation at time step t. Our objective is to predict future values of the sequence based on historical observations.

Forecasting task. Given a historical context window of fixed length P, the model forecasts the next H steps. For each starting index $i \in \{1, 2, \dots, T - P - H + 1\}$, we define:

$$\mathbf{x}_i = [x_i, x_{i+1}, \dots, x_{i+P-1}] \in \mathbb{R}^P, \tag{1}$$

$$\mathbf{y}_i = [x_{i+P}, x_{i+P+1}, \dots, x_{i+P+H-1}] \in \mathbb{R}^H. \tag{2}$$

The goal is to learn a forecasting function $f_\theta : \mathbb{R}^P \to \mathbb{R}^H$ that maps the historical window to future values:

$$\hat{\mathbf{y}}_i = f_\theta(\mathbf{x}_i). \tag{3}$$

Common operations and notation. Throughout the article, we use the following notation:

- *Input Embedding*: Each input $\mathbf{x}_i$ is mapped to a latent representation via a linear embedding:

  $$\mathbf{Z}_i = \mathbf{x}_i \cdot \mathbf{W}_e, \quad \mathbf{W}_e \in \mathbb{R}^{P \times d_{\text{model}}} \text{ or } \mathbb{R}^{1 \times d_{\text{model}}}.$$

- *Positional Encoding (PE)*: To inject temporal order, a positional encoding $\text{PE} \in \mathbb{R}^{P \times d_{\text{model}}}$ is added:

  $$\mathbf{Z}_i^{\text{pos}} = \mathbf{Z}_i + \text{PE},$$

  where PE is either:
  – *Fixed sinusoidal*: $\text{PE}_{(pos,2k)} = \sin\left(\frac{pos}{10000^{2k/d_{\text{model}}}}\right)$, $\text{PE}_{(pos,2k+1)} = \cos\left(\frac{pos}{10000^{2k/d_{\text{model}}}}\right)$,
  – or *Learnable*: parameters optimized during training.

- *Transformer Encoder*: Given a sequence $\mathbf{Z} \in \mathbb{R}^{P \times d_{\text{model}}}$, the Transformer encoder applies multiple layers of multi-head self-attention (MSA) and feedforward networks (FFN):

  $$\mathbf{H}^{(e)} = \text{TransformerEncoder}(\mathbf{Z}) = \text{LayerNorm}\left(\mathbf{Z} + \text{FFN}\left(\text{MSA}(\mathbf{Z})\right)\right), \tag{4}$$

  where $\mathbf{H}^{(e)} \in \mathbb{R}^{P \times d_{\text{model}}}$.

- Transformer decoder: Given a decoder input $\mathbf{Z}^{(d)}$ and encoder output $\mathbf{H}^{(e)}$, the Transformer decoder applies self-attention over $\mathbf{Z}^{(d)}$ and cross-attention with $\mathbf{H}^{(e)}$:

  $$\begin{aligned}\mathbf{H}^{(d)} &= \text{TransformerDecoder}(\mathbf{Z}^{(d)}, \mathbf{H}^{(e)}) \\ &= \text{LayerNorm}\left(\mathbf{Z}^{(d)} + \text{CrossAttention}\left(\text{SelfAttention}(\mathbf{Z}^{(d)}), \mathbf{H}^{(e)}\right)\right).\end{aligned} \tag{5}$$

- *Average pooling (AvgPool)*: After encoding, we often apply average pooling over the sequence (token) dimension to obtain a fixed-size vector:

  $$\text{AvgPool}(\mathbf{H}) = \frac{1}{P}\sum_{p=1}^{P} \mathbf{H}_p,$$

  where $\mathbf{H}_p$ denotes the p-th token embedding.

- *Output projection*: The final output is generated by projecting the pooled or decoded representation:

  $$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \mathbf{H}_i,$$

  where $\mathbf{W}_o \in \mathbb{R}^{d_{\text{model}} \times H}$ and $\mathbf{h}_i$ is the pooled hidden state.

In the `Autoformer` model, the time series is decomposed into trend and seasonal components. The trend component, $\mathbf{x}^{\text{trend}}$, is obtained by applying a moving average (MA) filter with a kernel size k. The kernel size k defines the window over which the average is computed. This filter smooths the time series to capture long-term trends, while the residual seasonal component, $\mathbf{x}^{\text{seasonal}}$, captures short-term periodic fluctuations.

In `Informer`, the attention mechanism is enhanced using a sparse attention approach to improve efficiency. The sparsity score $M(q_i)$ for each query q_i is computed by first calculating the attention between each query and all keys, and then applying a softmax function to identify the most relevant keys for each query. The sparsity score $M(q_i)$ is the difference between the maximum attention value and the average attention value over all keys, ensuring that only the most important keys are selected for attention computation. This enables the model to focus on a smaller subset of keys, significantly reducing computational complexity while retaining high performance.

The definitions and operations introduced in this section provide a unified foundation for the Transformer-based forecasting frameworks developed in the following sections. In particular, the common embedding formulation, attention mechanisms, and pooling operations enable a consistent description and fair comparison of different architectural variants. Building on this shared formulation, the next sections systematically introduce the `PatchTST`, `Informer`, and `Autoformer` architectures and their corresponding Minimal, Standard, and Full variants, highlighting how specific design choices affect modeling capacity, computational complexity, and robustness to noise.

[1] https://cds.climate.copernicus.eu/datasets/projections-cordex-domains-single-levels?tab=overview.
[2] https://github.com/Chisomnwa/Cryptocurrency-Data-Analysis.
[3] https://github.com/afshinfaramarzi/Energy-Demand-electricity-price-Forecasting/tree/main.

3. PatchTST Variants: architecture and computational complexity

The PatchTST framework addresses accurate forecating of univariate time series by first segmenting the input sequence into overlapping or non-overlapping patches of fixed-length, then modeling temporal dependencies using a Transformer-based architecture. In this section, we present and justify three architectural variants—*Minimal*, *Standard*, and *Full*—each representing a progressively more flexible and expressive model design.

3.1. PatchTST Minimal: a lightweight transformer backbone

The Minimal variant serves as a baseline design, emphasizing simplicity and computational efficiency. Each input patch $\mathbf{x}_i$ is linearly projected into a higher-dimensional latent space:

$$\mathbf{Z}_i = \mathbf{x}_i \cdot \mathbf{W}_e \in \mathbb{R}^{1\times d_{\text{model}}}, \tag{6}$$

Fixed sinusoidal positional encodings are added to incorporate temporal information:

$$\mathbf{Z}_i^{\text{pos}} = \mathbf{Z}_i + \text{PE}, \tag{7}$$

This encoded representation is passed through a Transformer encoder, followed by average pooling and a linear projection:

$$\mathbf{H}_i = \text{TransformerEncoder}(\mathbf{Z}_i^{\text{pos}}), \tag{8}$$

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \text{AvgPool}(\mathbf{H}_i) \in \mathbb{R}^H, \tag{9}$$

This variant is lightweight and suitable for short sequences or fast, interpretable inference.

Computational complexity.

$$\mathcal{O}_{\text{Minimal}} = \underbrace{P \cdot d_{\text{model}}^2}_{\text{Feedforward}} + \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Self-Attention}}.$$

3.2. PatchTST Standard: learning data-driven positional embeddings

The Standard variant replaces the fixed positional encodings with a learnable global positional embedding $\text{PE} \in \mathbb{R}^{1\times d_{\text{model}}}$:

$$\mathbf{Z}_i^{\text{pos}} = \mathbf{x}_i \cdot \mathbf{W}_e + \text{PE}, \tag{10}$$

This embedding is shared across all positions and learned during training, allowing the model to adapt its temporal representation. The rest of the architecture remains identical to Minimal architecture:

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \text{AvgPool}(\text{TransformerEncoder}(\mathbf{Z}_i^{\text{pos}})).$$

Computational complexity.

$$\mathcal{O}_{\text{Standard}} = \mathcal{O}_{\text{Minimal}} + \underbrace{\mathcal{O}(d_{\text{model}})}_{\text{Embedding overhead (negligible)}}.$$

3.3. PatchTST Full: a flexible encoder-decoder architecture

The Full variant introduces a decoder with cross-attention, allowing the model to explicitly learn dependencies between past inputs and future predictions.

Encoder. The input patch is embedded and encoded with sinusoidal positional encodings:

$$\mathbf{Z}_i^{(e)} = \mathbf{x}_i \cdot \mathbf{W}_e^{(e)} + \text{PE}^{(e)} \in \mathbb{R}^{P\times d_{\text{model}}}, \tag{11}$$

$$\mathbf{H}_i^{(e)} = \text{TransformerEncoder}(\mathbf{Z}_i^{(e)}). \tag{12}$$

Table 1
Comparison of PatchTST variants.

Model variant	Positional encoding	Architecture	Time complexity
Minimal PatchTST	Sinusoidal	Encoder-only	$\mathcal{O}(P^2 d_{\text{model}} + P d_{\text{model}}^2)$
Standard PatchTST	Trainable (global)	Encoder-only	$\mathcal{O}(P^2 d_{\text{model}} + P d_{\text{model}}^2)$
Full PatchTST	Sinusoidal (dual)	Encoder-Decoder	$\mathcal{O}(P^2 d_{\text{model}} + HP d_{\text{model}} + H d_{\text{model}}^2)$

Decoder. The decoder input is formed by repeating the last element of the patch to enable autoregressive-style conditioning:

$$\mathbf{x}_i^{\text{rep}} = [x_{i+P-1}, \ldots, x_{i+P-1}] \in \mathbb{R}^H, \tag{13}$$

$$\mathbf{Z}_i^{(d)} = \mathbf{x}_i^{\text{rep}} \cdot \mathbf{W}_e^{(d)} + \text{PE}^{(d)} \in \mathbb{R}^{H\times d_{\text{model}}}, \tag{14}$$

$$\mathbf{H}_i^{(d)} = \text{TransformerDecoder}(\mathbf{Z}_i^{(d)}, \mathbf{H}_i^{(e)}). \tag{15}$$

Output. The decoder output is projected to obtain the forecast:

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \mathbf{H}_i^{(d)} \in \mathbb{R}^H. \tag{16}$$

Computational complexity.

$$\mathcal{O}_{\text{Full}} = \underbrace{P \cdot d_{\text{model}}^2}_{\text{Encoder FFN}} + \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Encoder Attention}} + \underbrace{H \cdot d_{\text{model}}^2}_{\text{Decoder FFN}} + \underbrace{H \cdot P \cdot d_{\text{model}}}_{\text{Cross-Attention}}.$$

3.4. Comparative summary

Each PatchTST variant is designed to address different forecasting needs and trade-offs: (i) The *Minimal* variant provides fast training and interpretable forecasts, well-suited for small datasets and short horizons; (ii) The *Standard* variant adds learnable temporal representations, improving adaptability to real-world signals without significant computational overhead; (iii) The *Full* variant introduces an encoder–decoder structure that improves long-horizon modeling at the cost of increased complexity. A comparative overview of PatchTST variants is shown in Table 1, and the complete forecasting procedure for the Full variant is provided as an Algorithm in Appendix.

4. Informer variants: architecture and computational complexity

Transformer models have demonstrated strong performance in sequence modeling, yet their standard self-attention mechanism suffers from quadratic time complexity with respect to input length. This poses a bottleneck for long time series forecasting. To overcome this, we present three variants based on the Informer framework—Minimal, Standard, and Full—each offering a different balance between expressiveness and computational efficiency.

4.1. Informer Minimal: baseline with full self-attention

The Minimal variant adopts a standard encoder-only Transformer architecture. Each input patch is projected into a latent space using a linear transformation:

$$\mathbf{Z}_i = \mathbf{x}_i \cdot \mathbf{W}_e \in \mathbb{R}^{P\times d_{\text{model}}}. \tag{17}$$

To retain temporal information, fixed sinusoidal positional encodings are added:

$$\mathbf{Z}_i^{\text{pos}} = \mathbf{Z}_i + \text{PE}, \tag{18}$$

This sequence is passed through a multi-layer Transformer encoder:

$$\mathbf{H}_i = \text{TransformerEncoder}(\mathbf{Z}_i^{\text{pos}}), \tag{19}$$

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \text{AvgPool}(\mathbf{H}_i), \tag{20}$$

This model serves as a full-attention baseline, but its complexity increases quadratically with the input length.

Computational complexity.

$$\mathcal{O}_{\text{Minimal}} = \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Self-Attention}} + \underbrace{P \cdot d_{\text{model}}^2}_{\text{Feedforward Layers}}.$$

Remark 1. The primary difference between `PatchTST` Minimal and `Informer` Minimal lies in how they process time series data and the underlying architectural philosophy they follow. `PatchTST` Minimal adopts a patch-based approach where the input sequence is segmented into fixed-length patches, which are then embedded and processed by a Transformer encoder. This design focuses on capturing local patterns efficiently, reducing the input sequence length and computational complexity. In contrast, `Informer` Minimal follows a more conventional Transformer structure, directly operating on the full-resolution time series using positional encoding and self-attention across all time steps. This enables it to retain fine-grained temporal dependencies but can be more computationally intensive for long sequences. While `PatchTST` Minimal excels in modeling smooth, low-frequency patterns with fewer tokens, `Informer` Minimal is better suited for capturing detailed, high-frequency dynamics in time series data.

4.2. `Informer` Standard: ProbSparse attention for scalable forecasting

The Standard variant improves scalability by replacing full attention with a sparse approximation known as ProbSparse attention. This mechanism identifies a subset of "informative" queries and restricts attention to them.

Query selection. For each query vector $\mathbf{q}_i$, a sparsity score is computed:

$$M(\mathbf{q}_i) = \max_j \left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_{\text{model}}}} \right) - \frac{1}{P} \sum_{j=1}^{P} \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_{\text{model}}}}, \tag{21}$$

Only the top-u queries with the highest scores (where $u = \mathcal{O}(\log P)$) are selected to perform full attention over all keys.

Encoding. These selected queries are processed using the ProbSparse encoder:

$$\mathbf{H}_i = \text{ProbSparseEncoder}(\mathbf{Z}_i^{\text{pos}}), \tag{22}$$

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \text{AvgPool}(\mathbf{H}_i). \tag{23}$$

Computational complexity.

$$\mathcal{O}_{\text{Standard}} = \underbrace{\log P \cdot P \cdot d_{\text{model}}}_{\text{Sparse Attention}} + \underbrace{P \cdot d_{\text{model}}^2}_{\text{Feedforward Layers}}.$$

This represents a significant improvement over the quadratic complexity of the Minimal variant.

4.3. `Informer` Full: sequence-to-sequence architecture with dual attention

The Full variant extends `Informer` into a full encoder–decoder architecture. Both the encoder and decoder employ ProbSparse attention, enabling modeling of long-range dependencies in both input and output sequences.

Table 2
Comparison of `Informer` variants.

Model variant	Attention type	Architecture	Time complexity
Minimal `Informer`	Full Self-Attention	Encoder-only	$\mathcal{O}(P^2 d_{\text{model}} + P d_{\text{model}}^2)$
Standard `Informer`	ProbSparse (log-scaled)	Encoder-only	$\mathcal{O}(\log P \cdot P d_{\text{model}} + P d_{\text{model}}^2)$
Full `Informer`	ProbSparse (dual)	Encoder-Decoder	$\mathcal{O}(\log P \cdot P d_{\text{model}} + \log H \cdot H d_{\text{model}} + H P d_{\text{model}})$

Encoder.

$$\mathbf{Z}_i^{(e)} = \mathbf{x}_i \cdot \mathbf{W}_e^{(e)} + \text{PE}^{(e)}, \tag{24}$$

$$\mathbf{H}_i^{(e)} = \text{ProbSparseEncoder}(\mathbf{Z}_i^{(e)}). \tag{25}$$

Decoder.

$$\mathbf{x}_i^{\text{rep}} = [x_{i+P-1}, \ldots, x_{i+P-1}] \in \mathbb{R}^H, \tag{26}$$

$$\mathbf{Z}_i^{(d)} = \mathbf{x}_i^{\text{rep}} \cdot \mathbf{W}_e^{(d)} + \text{PE}^{(d)}, \tag{27}$$

$$\mathbf{H}_i^{(d)} = \text{ProbSparseDecoder}(\mathbf{z}_i^{(d)}, \mathbf{H}_i^{(e)}). \tag{28}$$

Output.

$$\hat{\mathbf{y}}_i = \mathbf{W}_o \cdot \mathbf{H}_i^{(d)} \in \mathbb{R}^H. \tag{29}$$

Computational complexity.

$$\mathcal{O}_{\text{Full}} = \underbrace{\log P \cdot P \cdot d_{\text{model}}}_{\text{Encoder Sparse Attention}} + \underbrace{\log H \cdot H \cdot d_{\text{model}}}_{\text{Decoder Sparse Attention}} + \underbrace{H \cdot P \cdot d_{\text{model}}}_{\text{Cross-Attention}}$$

4.4. Comparative summary

Each `Informer` variant offers different strengths: (i) *Minimal* serves as a full-attention benchmark and is effective for short sequences, but computationally expensive; (ii) *Standard* improves scalability using probabilistic attention, offering a good balance of efficiency and accuracy; (iii) *Full* introduces a sequence-to-sequence structure for modeling long-term, non-stationary patterns, at higher computational cost. A comparative overview is shown in Table 2, and the complete forecasting procedure for the Full variant is provided as an Algorithm in Appendix. The following Theorem we prove the reduction in complexity resulted by employing ProbSparse technique.

5. `Autoformer` Variants: architecture and computational complexity

`Autoformer` addresses the limitations of Transformer-based time series forecasting by introducing a decomposition architecture that separates each input sequence into trend and seasonal components. This inductive bias improves forecasting stability and interpretability, especially for signals with periodic or long-term structural patterns. We present three architectural variants—Minimal, Standard, and Full—tailored to different forecasting tasks and computational budgets.

`Autoformer` applies a moving average filter MA_k of kernel size k to decompose each input into trend and seasonal components:

$$\mathbf{x}_i^{\text{trend}} = \text{MA}_k(\mathbf{x}_i), \tag{30}$$

$$\mathbf{x}_i^{\text{seasonal}} = \mathbf{x}_i - \mathbf{x}_i^{\text{trend}}, \tag{31}$$

This decomposition is used throughout all model variants.

5.1. `Autoformer` Minimal: lightweight forecasting with minimal overhead

The Minimal variant retains the decomposition structure but applies a short moving average (e.g., $k = 3$) and a shallow encoder-only Transformer. The seasonal component is embedded and positionally encoded:

$$\mathbf{Z}_i = \text{Embed}(\mathbf{x}_i^{\text{seasonal}}) + \text{PE}, \tag{32}$$

$$\mathbf{H}_i = \text{TransformerEncoder}(\mathbf{Z}_i), \tag{33}$$

The final prediction combines the seasonal output with a linear projection of the trend:

$$\hat{\mathbf{y}}_i = \underbrace{\mathbf{W}_o \cdot \text{AvgPool}(\mathbf{H}_i)}_{\text{seasonal}} + \underbrace{\mathbf{W}_t \cdot \mathbf{x}_i^{\text{trend}}}_{\text{trend}} \tag{34}$$

Computational complexity.

$$\mathcal{O}_{\text{Simple}} = \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Self-Attention}} + \underbrace{P \cdot d_{\text{model}}^2}_{\text{Feedforward Layers}}.$$

5.2. `Autoformer` Standard: improved stability with enhanced decomposition awareness

The Standard variant builds on the Minimal design, retaining the same decomposition mechanism and encoder structure, but with deeper residual blocks, better weight initialization, and loss balancing for trend and seasonal outputs. These improvements enhance robustness on medium-term horizons.

Computational complexity.

$$\mathcal{O}_{\text{Standard}} = \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Self-Attention}} + \underbrace{P \cdot d_{\text{model}}^2}_{\text{Feedforward Layers}}.$$

5.3. `Autoformer` Full: sequence-to-sequence modeling for long-horizon forecasts

The Full variant introduces a sequence-to-sequence encoder–decoder design with longer moving average filtering (e.g., $k = 25$). This variant is built for complex, long-horizon forecasting where future sequences differ structurally from the past.

Encoder.

$$\mathbf{Z}_i^{(e)} = \mathbf{x}_i^{\text{seasonal}} \cdot \mathbf{W}_e^{(e)} + \text{PE}^{(e)}, \tag{35}$$

$$\mathbf{H}_i^{(e)} = \text{TransformerEncoder}(\mathbf{Z}_i^{(e)}). \tag{36}$$

Decoder. The seasonal decoder input is initialized to zeros:

$$\mathbf{Z}_i^{(d)} = \mathbf{0}_{B\times H\times 1} \cdot \mathbf{W}_e^{(d)} + \text{PE}^{(d)}, \tag{37}$$

$$\mathbf{H}_i^{(d)} = \text{TransformerDecoder}(\mathbf{z}_i^{(d)}.\mathbf{H}_i^{(e)}). \tag{38}$$

Output.

$$\hat{\mathbf{y}}_i = \underbrace{\mathbf{W}_o \cdot \mathbf{H}_i^{(d)}}_{\text{seasonal}} + \underbrace{\mathbf{W}_t \cdot \mathbf{x}_i^{\text{trend}}}_{\text{trend}}. \tag{39}$$

Computational complexity.

$$\mathcal{O}_{\text{Full}} = \underbrace{P^2 \cdot d_{\text{model}}}_{\text{Encoder Self-Attn}} + \underbrace{H^2 \cdot d_{\text{model}}}_{\text{Decoder Self-Attn}} + \underbrace{H \cdot P \cdot d_{\text{model}}}_{\text{Cross-Attention}}.$$

Table 3
Comparison of `Autoformer` variants.

Model variant	Decomposition details	Architecture	Time complexity
Minimal `Autoformer`	MA filter ($k = 3$), additive output	Encoder-only	$\mathcal{O}(P^2 d_{\text{model}} + P d_{\text{model}}^2)$
Standard `Autoformer`	MA filter ($k = 3$), tuned encoder	Encoder-only	$\mathcal{O}(P^2 d_{\text{model}} + P d_{\text{model}}^2)$
Full `Autoformer`	MA filter ($k = 25$), sequence-to-sequence	Encoder-Decoder	$\mathcal{O}(P^2 d_{\text{model}} + H^2 d_{\text{model}} + HP d_{\text{model}})$

5.4. Comparative summary

The three `Autoformer` variants provide flexible options depending on the forecasting setting: (i) *`Autoformer` Minimal* is fast and interpretable, ideal for clean signals or short horizons; (ii) *`Autoformer` Standard* is a balanced architecture that improves generalization with little added cost; (iii) *`Autoformer` Full* provides high capacity and long-range modeling power, suitable for highly dynamic time series. Each variant leverages trend-seasonal decomposition, enabling consistent signal separation while scaling from minimal to full-capacity architectures. A comparative overview is shown in Table 3, and the complete forecasting procedure for the Full variant is provided as an Algorithm in Appendix.

Remark 2 (Role of the modular transformer framework). The objective of the modular Transformer framework proposed in this paper is not to compete with ever-growing, monolithic Transformer architectures on benchmark accuracy, but to provide a *scientifically grounded and composable backbone* for time-series modeling. `Autoformer`, `Informer`, and `PatchTST` are chosen because they encode complementary and interpretable inductive biases—trend–seasonal decomposition, sparse long-range attention, and patch-wise temporal representation—that admit theoretical analysis and controlled ablation. These properties make them particularly suitable as *building blocks* in larger hybrid systems, such as physics-informed, operator-theoretic, or Koopman-enhanced forecasting architectures, where stability, robustness, and interpretability are as important as raw predictive performance.

Relation to recent transformer forecasters. Although recent models such as `iTransformer` [29], `TimesNet` [30], `Crossformer` [31], `FEDformer` [21], and `Pyraformer` [22] are not explicitly benchmarked in this study, their core mechanisms can be mapped directly onto the architectural axes analyzed in our modular framework. Specifically, `iTransformer` emphasizes *cross-channel token mixing*, which corresponds to a full-capacity extension of the channel-independent or patch-based representations studied in `PatchTST`. `TimesNet` introduces *multi-period temporal blocks* that act as a learned form of decomposition across time scales, closely related to the trend–seasonal and frequency-based decomposition principles embodied by `Autoformer` and `FEDformer`. `Crossformer` employs *cross-dimension and cross-time attention*, which can be interpreted as a structured generalization of sparse and hierarchical attention, placing it along the same efficiency and scalability axis explored by Informer.

From this perspective, many high-performing modern architectures can be viewed as hybrid compositions of the three fundamental inductive biases investigated here: decomposition (`Autoformer/FEDformer/TimesNet`), sparse or hierarchical attention (`Informer/Pyraformer/Crossformer`), and patch-wise or channel-wise tokenization (`PatchTST/iTransformer`). By systematically analyzing these primitives in isolation and in controlled variants, our framework provides a transferable understanding that extends naturally to more complex and recent Transformer forecasters.

6. Theoretical properties of the proposed transformer variants

We formalize noise–robustness, bias–variance trade–offs, and approximation/complexity guarantees that are tailored to `Autoformer`, `PatchTST`, and `Informer` (ProbSparse). Throughout, let an observed sequence be $\mathbf{x} = s + \eta$ with unknown clean signal s and additive noise η. We assume η_t are zero-mean, independent, sub-Gaussian with parameter σ^2 (Gauss is a special case).

6.1. `Autoformer`: moving-average decomposition attenuates noise

Let MA_k be the length-k moving average (uniform) filter with finite impulse response (FIR) $h(t) = \frac{1}{k}\mathbf{1}_{\{0,\dots,k-1\}}(t)$. `Autoformer` decomposes $\mathbf{x}$ into

$$\mathbf{x}_{\text{trend}} = h * \mathbf{x}, \qquad \mathbf{x}_{\text{seasonal}} = \mathbf{x} - \mathbf{x}_{\text{trend}},$$

where $*$ denotes the convolution and is defined as

$$(h * \mathbf{x})_t \;=\; \sum_{\tau\in\mathbb{Z}} h(\tau)\,\mathbf{x}_{t-\tau}.$$

For the length-k moving average (MA_k), $h(\tau) = \frac{1}{k}$ for $\tau \in \{0, 1, \dots, k-1\}$ and $h(\tau) = 0$ otherwise.

Lemma 1 (ℓ_2-gain of MA_k). *The operator $\mathbf{x} \mapsto MA_k(\mathbf{x})$ has ℓ_2-gain $\|h\|_2 = \sqrt{\sum_t h(t)^2} = k^{-1/2}$.*

Proof. Since $h(t) = 1/k$ for $t = 0, \dots, k-1$ and 0 otherwise, $\|h\|_2^2 = \sum_{t=0}^{k-1} (1/k)^2 = k \cdot (1/k^2) = 1/k$. □

Lemma 2 (Output variance of FIR filtering of white noise). *Let $\eta = \{\eta_t\}_{t\in\mathbb{Z}}$ be i.i.d. with $\mathbb{E}\,\eta_t = 0$ and $Var(\eta_t) = \sigma^2$. For any FIR kernel h,*

$$Var\big[(h*\eta)_t\big] \;=\; \sigma^2 \sum_{\tau\in\mathbb{Z}} h(\tau)^2 \;=\; \sigma^2\,\|h\|_2^2 \qquad \textit{for every } t \in \mathbb{Z}.$$

Proof. Fix t. Since h has finite support,

$$(h * \eta)_t \;=\; \sum_{\tau} h(\tau)\,\eta_{t-\tau}.$$

Using zero mean and independence,

$$\mathrm{Var}\big[(h*\eta)_t\big] = \mathbb{E}\left[\left(\sum_{\tau} h(\tau)\,\eta_{t-\tau}\right)^2\right] = \sum_{\tau}\sum_{\tau'} h(\tau)h(\tau')\,\mathbb{E}\big[\eta_{t-\tau}\eta_{t-\tau'}\big].$$

If $\tau \neq \tau'$, independence and zero mean give $\mathbb{E}[\eta_{t-\tau}\eta_{t-\tau'}] = 0$; if $\tau = \tau'$, we get $\mathbb{E}[\eta_{t-\tau}^2] = \sigma^2$. Hence only diagonal terms remain:

$$\mathrm{Var}\big[(h*\eta)_t\big] \;=\; \sum_{\tau} h(\tau)^2\,\sigma^2 \;=\; \sigma^2\|h\|_2^2.$$

Theorem 1 (Noise attenuation in `Autoformer` trend). *If η_t are i.i.d. with $\mathbb{E}\,\eta_t = 0$ and $Var(\eta_t) = \sigma^2$, then*

$$\mathbb{E}\Big\|\mathbf{x}_{trend} - MA_k(s)\Big\|_2^2 \;=\; \mathbb{E}\,\|h*\eta\|_2^2 \;=\; \sigma^2\|h\|_2^2 \cdot \mathcal{N} \textit{ with } \quad \|h\|_2^2 = \tfrac{1}{k},$$

where $\mathcal{N}$ is the number of time indices over which the ℓ_2 norm is taken (see the discussion below). Equivalently, per time index

$$\mathbb{E}\Big[\big((h*\eta)_t\big)^2\Big] \;=\; \frac{\sigma^2}{k}.$$

Proof. Let $\mathbf{x} = s + \eta$ be the observed sequence, and let h be the MA_k kernel. Linearity of convolution gives

$$\mathbf{x}_{\text{trend}} \;=\; h*\mathbf{x} \;=\; h*(s+\eta) \;=\; (h*s) + (h*\eta) \;=\; \mathrm{MA}_k(s) + h*\eta.$$

Thus the (trend) error relative to the filtered clean signal is

$$\mathbf{x}_{\text{trend}} - \mathrm{MA}_k(s) \;=\; h*\eta.$$

By Lemma 2 and $\|h\|_2^2 = 1/k$ for MA_k,

$$\mathbb{E}\Big[\big((h*\eta)_t\big)^2\Big] \;=\; \sigma^2\|h\|_2^2 \;=\; \frac{\sigma^2}{k}, \quad \forall t.$$

If the ℓ_2 norm is taken over a finite index set $\mathcal{T}$ (e.g., $\mathcal{T} = \{1, \dots, T\}$ with a specified boundary convention), then by stationarity away from boundaries,

$$\mathbb{E}\,\|h*\eta\|_2^2 \;=\; \sum_{t\in\mathcal{T}} \mathbb{E}\Big[\big((h*\eta)_t\big)^2\Big] \;\approx\; |\mathcal{T}| \cdot \frac{\sigma^2}{k},$$

up to edge effects (see Remark 3). Writing $\mathcal{N} := |\mathcal{T}|$ yields the stated form. □

Remark 3 (Boundary conditions and edge effects). For finite sequences, one must specify how $(h * x)_t$ is computed near the boundaries: (i) *valid* convolution (drop indices that require samples beyond the ends), (ii) *same* with zero/padded reflection, or (iii) *circular* (periodic) convolution. Under (i), the count $\mathcal{N}$ becomes $T - k + 1$, and the per-time variance formula remains σ^2/k on the valid region. Under (ii), the first/last $(k-1)$ indices have slightly smaller output variance due to partial windows. Under (iii), stationarity holds exactly and $\mathbb{E}\|h*\eta\|_2^2 = T \cdot \sigma^2/k$.

Remark 4 (Fourier/Parseval view (optional)). Let $\widehat{h}(\omega)$ be the DTFT of h. For wide-sense stationary white noise with power spectral density $S_{\eta\eta}(\omega) = \sigma^2$, the output variance is $\frac{1}{2\pi}\int_{-\pi}^{\pi} |\widehat{h}(\omega)|^2 S_{\eta\eta}(\omega)\,d\omega = \sigma^2 \cdot \frac{1}{2\pi}\int |\widehat{h}|^2 = \sigma^2\|h\|_2^2$, by Parseval/Plancherel, giving the same result.

Corollary 1 (Sub-Gaussian noise). *If η_t are i.i.d. sub-Gaussian with proxy σ^2, then $(h*\eta)_t$ is sub-Gaussian with proxy $\sigma^2\|h\|_2^2 = \sigma^2/k$, and the per-time variance bound remains $\le \sigma^2/k$.*

Proposition 1 (Bias of trend for smooth signals). *If the clean signal s is $\mathcal{L}$-Lipschitz (discrete), then for any index t,*

$$\Big|MA_k(s)_t - s_t\Big| \le \frac{\mathcal{L}}{k}\sum_{j=0}^{k-1} |t-j-t| = \frac{\mathcal{L}}{k}\sum_{j=0}^{k-1} j = \frac{\mathcal{L}(k-1)}{2}.$$

Consequently, over a window, the squared bias of the trend scales as $\mathcal{O}(\mathcal{L}^2k^2)$, while the variance scales as σ^2/k.

Proof. Triangle inequality and Lipschitzness yield $|s_{t-j} - s_t| \le \mathcal{L}|j|$, average over $j = 0, \dots, k-1$. Sum of the first $(k-1)$ integers equals $(k-1)k/2$; divide by k. □

Implication. `Autoformer`'s trend channel trades *variance* reduction (σ^2/k) against *bias* that grows with k for nonsymmetric windows. Choice of k thus controls robustness vs. fidelity—mirroring empirical noise-regime performance.

6.2. `PatchTST`: patching reduces noise variance with a quantified bias

Consider non-overlapping patching with patch length p, $z_j = \frac{1}{p}\sum_{t=jp}^{(j+1)p-1} x_t$ (extension to overlap is analogous).

Theorem 2 (Variance reduction by patching). *Under the noise model above, $Var(z_j) = \frac{\sigma^2}{p}$. More generally, for sub-Gaussian η_t with proxy σ^2, z_j remains sub-Gaussian with proxy σ^2/p.*

Proof. Independence gives $\mathrm{Var}(\frac{1}{p}\sum \eta_t) = \frac{1}{p^2}\sum \mathrm{Var}(\eta_t) = \sigma^2/p$. For sub-Gaussian variables, averages scale the proxy by $1/p$. □

Proposition 2 (Patch mean: bias–variance tradeoff). *Let $x_t = s_t + \eta_t$, where $(\eta_t)_{t\in\mathbb{Z}}$ are i.i.d. with $\mathbb{E}\,\eta_t = 0$ and $\mathrm{Var}(\eta_t) = \sigma^2$. Fix a patch length*

$p \in \mathbb{N}$ and define the non-overlapping patch average

$$z_j \;=\; \frac{1}{p}\sum_{t=jp}^{(j+1)p-1} x_t,\; c_j \;=\; jp+\frac{p-1}{2} \quad \textit{(patch center, for odd } p\textit{)}.$$

Assume the clean signal s is (discrete) $\mathcal{L}$-Lipschitz, i.e., $|s_t - s_u| \le \mathcal{L}|t-u|$ for all t,u. Then

$$\mathbb{E}\big(z_j - s_{c_j}\big)^2 \;\le\; C_1\,\mathcal{L}^2 p^2 \;+\; C_2\,\frac{\sigma^2}{p},$$

with the explicit constants $C_1 = \frac{1}{16}$ and $C_2 = 1$ for the centered symmetric window with odd p.

Proof. Write

$$z_j - s_{c_j} = \frac{1}{p}\sum_{t=jp}^{(j+1)p-1}(s_t - s_{c_j}) \;+\; \frac{1}{p}\sum_{t=jp}^{(j+1)p-1}\eta_t =: B_j + N_j.$$

Bias. By Lipschitzness, $|s_t - s_{c_j}| \le \mathcal{L}\,|t-c_j|$, hence

$$|B_j| \;\le\; \frac{\mathcal{L}}{p}\sum_{t=jp}^{(j+1)p-1}|t-c_j|.$$

For odd p, let $t = c_j + m$ with $m \in \{-(p-1)/2, \dots, (p-1)/2\}$. Then

$$\sum_{m=-(p-1)/2}^{(p-1)/2}|m| = 2\sum_{m=1}^{(p-1)/2} m = \frac{p^2-1}{4},$$

and therefore

$$|B_j| \le \frac{\mathcal{L}}{p}\cdot\frac{p^2-1}{4} = \mathcal{L}\,\frac{p^2-1}{4p} \le \frac{\mathcal{L}p}{4}, \qquad B_j^2 \le \frac{\mathcal{L}^2}{16}\,p^2.$$

Variance. Since the η_t are i.i.d. with variance σ^2,

$$\mathrm{Var}(N_j) = \mathrm{Var}\left(\frac{1}{p}\sum_{t=jp}^{(j+1)p-1}\eta_t\right) = \frac{1}{p^2}\sum_{t=jp}^{(j+1)p-1}\mathrm{Var}(\eta_t) = \frac{\sigma^2}{p}.$$

MSE. Because $\mathbb{E}N_j = 0$ and B_j is deterministic given s,

$$\mathbb{E}\big(z_j - s_{c_j}\big)^2 = B_j^2 + \mathrm{Var}(N_j) \le \frac{1}{16}L^2p^2 + \frac{\sigma^2}{p}.$$

Thus $C_1 = \frac{1}{16}$ and $C_2 = 1$ for the centered symmetric (odd p) window. □

Remark 5 (Even p and sub-Gaussian noise). If p is even (center at the mid-point), the same steps give $B_j^2 \le \frac{\mathcal{L}^2}{4}p^2$; the bound still holds with a larger constant C_1. If η_t are i.i.d. sub-Gaussian with proxy σ^2, the patch mean remains sub-Gaussian with proxy σ^2/p, so the variance term σ^2/p continues to upper bound $\mathrm{Var}(N_j)$.

Implication. `PatchTST` reduces *noise variance linearly in p* (Theorem 2) while incurring a controllable *low-frequency bias* (Prop. 2). This formalizes why medium patch sizes are robust in noisy settings while very large patches can oversmooth.

6.3. `Informer` (ProbSparse): truncation error and robustness of top-u attention

Let attention logits be $a_j = \langle q, k_j\rangle/\sqrt{d}$ with softmax weights $\alpha_j = \exp(a_j)/\sum_{\ell=1}^{L}\exp(a_\ell)$. Let S be the index set of the top-u logits ($|S| = u$) and $\tilde{\alpha}_j$ be the softmax restricted to S (renormalized). Define the attention outputs

$$y \;=\; \sum_{j=1}^{L}\alpha_j v_j, \qquad \tilde{y} \;=\; \sum_{j\in S}\tilde{\alpha}_j v_j.$$

Lemma 3 (Mass outside top-u under a margin). *Let $a_j = \langle q, k_j\rangle/\sqrt{d}$ and let S be the index set of the top-u logits. If there exists $\Delta > 0$ such that $\max_{j\notin S} a_j \le \min_{j\in S} a_j - \Delta$, then with $\alpha_j = \exp(a_j)/\sum_{\ell=1}^{L}\exp(a_\ell)$,*

$$\sum_{j\notin S}\alpha_j \;\le\; \frac{L-u}{u}\,e^{-\Delta}.$$

Proof. Let $m := \min_{j\in S} a_j$. For $j \notin S$, $e^{a_j} \le e^{m-\Delta}$, hence

$$\sum_{j\notin S} e^{a_j} \;\le\; (L-u)e^{m-\Delta}.$$

Also, $\sum_{j\in S} e^{a_j} \ge ue^m$. Therefore

$$\sum_{j\notin S}\alpha_j = \frac{\sum_{j\notin S} e^{a_j}}{\sum_{\ell} e^{a_\ell}} \le \frac{(L-u)e^{m-\Delta}}{\sum_{j\in S} e^{a_j}} \le \frac{(L-u)e^{m-\Delta}}{ue^m} = \frac{L-u}{u}e^{-\Delta}.$$

Theorem 3 (Truncation error of probsparse attention). *Let $y = \sum_{j=1}^{L}\alpha_j v_j$ be the full attention output and $\tilde{y} = \sum_{j\in S}\tilde{\alpha}_j v_j$ the restricted output where $\tilde{\alpha}_j = \alpha_j/\sum_{\ell\in S}\alpha_\ell$ for $j \in S$. Assume $\|v_j\|_2 \le V_{\max}$ and the margin condition of Lemma 3. Then*

$$\|y - \tilde{y}\|_2 \;\le\; 2V_{\max}\sum_{j\notin S}\alpha_j \;\le\; 2V_{\max}\,\frac{L-u}{u}\,e^{-\Delta}.$$

Proof. Add and subtract $\sum_{j\in S}\alpha_j v_j$ and use the triangle inequality:

$$\begin{aligned}\|y-\tilde{y}\|_2 &= \Big\|\sum_{j\notin S}\alpha_j v_j + \sum_{j\in S}(\alpha_j - \tilde{\alpha}_j)v_j\Big\|_2 \\ &\le \Big\|\sum_{j\notin S}\alpha_j v_j\Big\|_2 + \Big\|\sum_{j\in S}(\alpha_j-\tilde{\alpha}_j)v_j\Big\|_2 \\ &\le V_{\max}\left(\sum_{j\notin S}\alpha_j + \sum_{j\in S}|\alpha_j - \tilde{\alpha}_j|\right).\end{aligned}$$

Now note

$$\sum_{j\in S}|\alpha_j - \tilde{\alpha}_j| = \sum_{j\in S}\alpha_j\left|1 - \frac{1}{\sum_{\ell\in S}\alpha_\ell}\right| = \sum_{j\in S}\alpha_j\,\frac{\sum_{\ell\notin S}\alpha_\ell}{\sum_{\ell\in S}\alpha_\ell} \le \sum_{\ell\notin S}\alpha_\ell.$$

Hence,

$$\|y-\tilde{y}\|_2 \;\le\; 2V_{\max}\sum_{j\notin S}\alpha_j.$$

Apply Lemma 3 to bound the dropped mass: $\sum_{j\notin S}\alpha_j \le \frac{L-u}{u}e^{-\Delta}$, which yields the claim. □

Remark 6. The proof is norm-agnostic: for any norm $\|\cdot\|$ with $\|v_j\| \le V_{\max}$, the same steps give $\|y-\tilde{y}\| \le 2V_{\max}\frac{L-u}{u}e^{-\Delta}$. If only the weaker bound $\sum_{j\notin S}\alpha_j \le (L-u)e^{-\Delta}$ is available (e.g., when omitting the $\sum_{j\in S}e^{a_j} \ge ue^m$ step), the theorem still holds with the factor $\frac{L-u}{u}$ replaced by $(L-u)$.

Theorem 4 (Conditional complexity of ProbSparse with reliable top-u recovery). *Assume for each query q_i we can (i) estimate the sparsity score $M(q_i, K)$ using $O(\log L_K)$ key probes so that the top-u keys with $u = O(\log L_K)$ are recovered with probability at least $1-\delta$ (under a score-gap condition), and (ii) restrict attention to these u keys.*

Proof. Fix a query q_i and denote by $S_i \subseteq [L_K]$ the (true) index set of the top-u keys according to the sparsity score $M(q_i, K)$, with $u = O(\log L_K)$. Let $\hat{S}_i$ be the set returned by the probe procedure, which—by assumption—recovers S_i exactly with probability at least $1-\delta$ using $O(\log L_K)$ key probes. For a single query q_i:

(a) *Scoring/probing.* By assumption, computing $M(q_i, K)$ (or an estimator sufficient to identify S_i) uses $O(\log L_K)$ key probes, hence $O(\log L_K)$ time in L_K.
(b) *Restricted attention.* Once $\widehat{S}_i$ is identified with $|\widehat{S}_i| = u = O(\log L_K)$, computing the restricted attention $\tilde{y}_i = \sum_{j\in\widehat{S}_i} \tilde{\alpha}_{ij} v_j$ costs $O(u)$ operations in L_K, i.e., $O(\log L_K)$ (suppressing dimensions that do not scale with sequence length).

Thus, conditional on the success event $\mathcal{E}_i := \{\widehat{S}_i = S_i\}$, the per-query cost is $O(\log L_K)$. Across L_Q queries, the total cost conditional on $\mathcal{E} := \bigcap_{i=1}^{L_Q} \mathcal{E}_i$ is $O(L_Q \log L_K)$.

By a union bound, $\mathbb{P}(\mathcal{E}) \geq 1 - L_Q\delta$. Choosing the probe budget so that $\delta \leq L_Q^{-2}$ (which is standard for subsampled/top-u identification under a fixed score gap) yields $\mathbb{P}(\mathcal{E}) \geq 1 - L_Q \cdot L_Q^{-2} = 1 - L_Q^{-1}$, i.e., with high probability in L_Q the total cost is $O(L_Q \log L_K)$. (Equivalently, the *expected* total cost is $O(L_Q \log L_K)$ since the failure event has vanishing probability and the algorithm never performs $O(L_Q L_K)$ work by construction.)

Work on the event $\mathcal{E}$ so that $\widehat{S}_i = S_i$ for every query. For a fixed query (drop the subscript i), let attention logits be $a_j = \langle q, k_j\rangle/\sqrt{d}$ with softmax weights $\alpha_j = e^{a_j}/\sum_{\ell=1}^{L_K} e^{a_\ell}$, and define the top-$u$ set S and restricted, renormalized weights $\tilde{\alpha}_j = \alpha_j/\sum_{\ell\in S}\alpha_\ell$ for $j \in S$. Let the full and restricted outputs be $y = \sum_{j=1}^{L_K} \alpha_j v_j$ and $\tilde{y} = \sum_{j\in S} \tilde{\alpha}_j v_j$. □

Corollary 2 (Noise robustness via support truncation). *Suppose values contain additive noise $v_j = \bar{v}_j + \xi_j$, with ξ_j i.i.d., $\mathbb{E}\xi_j = 0$, $\mathbb{E}\|\xi_j\|_2^2 = \tau^2$. The output noise variance under full attention is $\mathbb{E}\|\sum_j \alpha_j \xi_j\|_2^2 = \tau^2 \sum_j \alpha_j^2$. Under ProbSparse with support size u, $\mathbb{E}\|\sum_{j\in S} \tilde{\alpha}_j \xi_j\|_2^2 = \tau^2 \sum_{j\in S} \tilde{\alpha}_j^2 \in [\tau^2/u, \tau^2]$, with the lower endpoint attained by uniform weights on S. When the top-u are well separated (large Δ), Theorem 3 guarantees $\tilde{\alpha}$ close to α and the dropped-noise contribution is exponentially small in Δ.*

Proof. Independence plus $\mathrm{Var}(\sum w_j \xi_j) = \tau^2 \sum w_j^2$. For any distribution on a support of size u, $\sum w_j^2 \geq 1/u$ by Cauchy–Schwarz. Combine with Theorem 3. □

Implication. ProbSparse yields (i) *sublinear complexity* with $u = \mathcal{O}(\log L)$ while (ii) preserving the full-attention output up to an error that decays exponentially with the score margin Δ (Theorem 3). Under noise in v, truncation bounds the output variance by the concentrated mass over the top-u terms (Corollary 2), explaining robustness when informative keys dominate.

Remark 7. `Autoformer` reduces variance via low-pass trend extraction ($\propto 1/k$) but introduces smoothing bias for large k. `PatchTST` reduces variance via patch averaging ($\propto 1/p$) with an explicit local smoothing bias ($\propto p$). `Informer` achieves sublinear-time attention with an exponentially controlled truncation error when informative keys are margin-separated; output noise depends on the retained mass geometry ($\sum \tilde{\alpha}_j^2$) over top-u keys.

7. Numerical simulations

To systematically evaluate the performance of Transformer-based forecasting models, we construct a diverse set of synthetic signals that reflect various temporal dynamics observed in real-world time series. Let $t \in \{0, 1, 2, \dots\}$. The clean signals $s(t)$ are mathematically defined as:

Sine: $s(t) = \sin\left(\frac{2\pi t}{40}\right)$,

Cosine + Trend: $s(t) = 0.01t + \cos\left(\frac{2\pi t}{50}\right)$,

Exponential decay × Sine: $s(t) = e^{-0.01t} \cdot \sin\left(\frac{2\pi t}{50}\right)$,

2nd Order Polynomial: $s(t) = 0.0001t^2 - 0.03t + 3$,

Log × Sine: $s(t) = \log(1 + t) \cdot \sin\left(\frac{2\pi t}{80}\right)$,

Gaussian bump: $s(t) = \exp\left(-\frac{(t-250)^2}{2\cdot 50^2}\right)$,

Long period sine: $s(t) = \sin\left(\frac{2\pi t}{100}\right)$,

Cubic polynomial: $s(t) = 0.00001(t-250)^3 + 0.05t$,

Exponential growth: $s(t) = e^{0.005t}$,

Cosine envelope × Sine: $s(t) = \left(1 + 0.5\cos\left(\frac{2\pi t}{100}\right)\right) \cdot \sin\left(\frac{2\pi t}{30}\right)$.

These synthetic signals are intentionally crafted to isolate different forecasting challenges:

- `Sine, Long period sine:` Clean periodic structures that assess the model's ability to recognize short versus long cycles.
- `Cosine + Trend, Exponential Growth:` Trends and compound effects that test extrapolation capabilities.
- `Exponential decay × Sine:` Decreasing amplitude signals that challenge long-term memory and decay modeling.
- `2nd Order and Cubic polynomial:` Smooth non-periodic curvature for evaluating structural generalization and overfitting.
- `Log × Sine:` Combines slow growth and fast oscillations, requiring flexible temporal representation.
- `Gaussian bump:` Localized pulse to test event detection within longer sequences.
- `Cosine envelope × Sine:` Modulated oscillation to assess adaptability to evolving amplitude and frequency.

7.1. Noise injection strategy

To mimic realistic imperfections in real-world data, we inject noise into the clean signals through a multi-component process:
(i) Additive Gaussian noise: Models measurement uncertainty with:

$$s_{\text{noisy}}(t) = s(t) + \mathcal{N}(0, 0.10).$$

(ii) Multiplicative amplitude jitter: Simulates sensor variability or calibration drift:

$$s_{\text{noisy}}(t) \leftarrow s_{\text{noisy}}(t) \cdot (1 + \mathcal{N}(0, 0.08)).$$

(iii) Random temporal shift: With 10% probability, a small shift $\Delta t \in [-10, 10]$ is applied to model event misalignment:

$$s_{\text{noisy}}(t) \leftarrow s_{\text{noisy}}(t + \Delta t).$$

This noise scheme provides a controlled yet diverse perturbation, effectively testing the robustness and generalization capacity of forecasting models under noisy conditions. Moreover, all signals (clean and noisy) are normalized to the unit interval $[0, 1]$ using the transformation:

$$\hat{s}(t) = \frac{s(t) - \min_t s(t)}{\max_t s(t) - \min_t s(t)},$$

This step ensures scale invariance and facilitates stable training and comparison across signals of varying magnitudes.

Design and stability of synthetic signal parameters. The synthetic signals are parameterized to provide a controlled yet diverse testbed for time-series forecasting. Frequencies, growth rates, decay constants, and polynomial coefficients are chosen so that all signals exhibit comparable dynamic ranges after normalization to $[0, 1]$, while still representing distinct temporal regimes: periodic, trend-dominated, damped oscillatory, polynomial, logarithmic–oscillatory, localized, and amplitude-modulated dynamics. This construction ensures that no single signal is trivially easier or harder due to scale alone, and that differences in performance arise from the models' ability to capture temporal structure.

Parameter stability is ensured by two mechanisms. First, all clean signals are smooth deterministic functions with bounded variation, so small changes in frequency, slope, or decay rate lead only to small changes in trajectory shape. Second, robustness is explicitly validated through the multi-component noise injection scheme (additive Gaussian noise, multiplicative jitter, and random temporal shifts), which perturbs amplitudes and time indices while preserving the underlying functional form. The consistent behavior of models across clean and noisy settings demonstrates that the experimental conclusions are not sensitive to specific parameter choices but reflect stable, structural properties of the tested forecasting architectures.

7.2. Hardware setup

All deep learning model training was performed on a high-performance computing node equipped with an `NVIDIA A100 GPU` featuring 80 GB of high-bandwidth Video RAM (`VRAM`), optimized for large-scale machine learning workloads. The system was configured with 256 `GB` of main memory accessible per `CPU` core, ensuring efficient handling of high-dimensional data and large batch sizes. Training tasks were orchestrated using the Simple `Linux` Utility for Resource Management (`SLURM`), which allocated compute resources with job-specific constraints to ensure optimal utilization of the `A100`'s memory architecture. This hardware–software configuration significantly accelerated the training process and maintained computational stability, which was critical for learning long-range temporal dependencies in wind power time series data.

7.3. Training setup

To ensure a fair and consistent comparison across different Transformer-based architectures, we standardized the model configuration parameters for all variants. Specifically, each model—including `Autoformer`, `Informer`, and `PatchTST` variants—was implemented with a model dimension (d_{model}) of 8, number of attention heads set to 2, feedforward network dimension of 32, and 2 encoder layers (and 1 decoder layer where applicable). This uniform setup minimizes the influence of architectural hyperparameter variations, thereby allowing the evaluation to focus solely on the intrinsic differences in model design and attention mechanisms. Such standardization is crucial for isolating the effects of core architectural innovations in comparative forecasting tasks.

All models were trained for 300, and 600 epochs with the `Adam` optimizer (learning rate = 10^{-3}), minimizing the mean squared error (MSE) between predictions and ground truth for clean and noisy signals, respectively.

Each `PatchTST`, `Informer`, and `Autoformer` variant is evaluated across 5 different patch lengths and 5 forecast horizons on a suite of 10 synthetic signals, resulting in a total of 750 configurations ($3\times5\times5\times10$) for clean signals. To assess robustness under realistic conditions, an additional set of 750 configurations is tested on noisy versions of the same signals, yielding a comprehensive total of 1500 model evaluations. To enhance clarity and interpretability, we present the results as heatmaps averaged over all signals for each model variant. Forecasting performance is measured using root mean square error (RMSE) and mean absolute error (MAE), evaluated over grids of patch lengths and forecast horizons.

Code availability statement. The complete source code used in this study—including the full implementation of the Transformer-based models for time-series forecasting—is openly available in the dedicated `compactformer` GitHub repository[4] and Zenodo.[5] `compactformer` is a modular, research-grade `Python` package that offers a unified interface for Transformer-based time-series forecasting, enabling reproducible experimentation and rapid prototyping.

7.4. Evaluation of `PatchTST` Variants under clean and noisy conditions

Fig. 1 presents heatmaps of RMSE and MAE averaged over all signals, separately for clean and noisy conditions. Each cell corresponds to the mean forecast error for a given patch length and forecast horizon combination. Table 4 highlights the best-performing configuration for each signal under clean conditions, while Table 5 reports the corresponding results for noisy signals.

These tables present the best-performing `PatchTST` model for each signal type based on RMSE and MAE values, highlighting the optimal configurations of patch length and forecast horizon for both clean and noisy signals.

Minimal variant. The `PatchTST` Minimal variant demonstrates competitive performance, especially in configurations with shorter forecast horizons. Across different patch lengths and horizons, it consistently maintains low RMSE and MAE values, with the best results observed at smaller patch lengths and short to mid-range horizons. Specifically, for clean signals, it achieves its best performance with patch lengths of 4 or 8 and forecast horizons of 2 or 4, showing strong forecasting accuracy (RMSE as low as 0.0270 and MAE as low as 0.0205).

For noisy signals, the Minimal variant still exhibits strong robustness, although the performance is slightly reduced compared to clean signals. The variant continues to perform well in the shorter horizon configurations (patch lengths of 4, 8, and 12) with RMSE and MAE values maintaining a good balance between accuracy and computational efficiency. In the noisy environment, the RMSE stays around 0.0623 to 0.0757 and the MAE varies between 0.0501 and 0.0596.

Standard variant. The `PatchTST` Standard variant demonstrates the best overall performance across various configurations, showing excellent accuracy and stability in both clean and noisy conditions. For clean signals, it consistently achieves low RMSE and MAE across all patch lengths and forecast horizons. The variant shows its best results at a patch length of 12 and a forecast horizon of 4 or 8, with RMSE values as low as 0.0260 and MAE values as low as 0.0232, making it a highly reliable choice for a variety of signal types.

In noisy environments, the Standard variant maintains impressive robustness. Although the performance slightly declines compared to clean signals, the variant continues to deliver competitive RMSE (ranging from 0.0521 to 0.0847) and MAE (ranging from 0.0414 to 0.0601) values. This suggests that the Standard variant is particularly suited for applications requiring stable performance across different levels of noise.

Full variant performance. The `PatchTST` Full variant excels in configurations involving complex, nonlinear signals, especially under noisy conditions. It consistently achieves strong performance across a variety of patch lengths and forecast horizons, though it is computationally more demanding than the Minimal and Standard variants.

For clean signals, the Full variant performs well across all configurations. It shows its best results with a patch length of 12 and a forecast horizon of 8 or 20, achieving RMSE values as low as 0.0302 and MAE values as low as 0.0234. This indicates that the Full variant is capable of capturing intricate temporal dependencies effectively.

In noisy environments, the Full variant remains robust but slightly less accurate compared to clean signals. It performs particularly well with longer patch lengths (12 and 20) and longer forecast horizons (8 and 20). For example, at a patch length of 12 and horizon 20, it achieves RMSE values around 0.0447 and MAE values near 0.0357, demonstrating its ability to handle noisy signals while maintaining solid forecasting accuracy.

[4] https://github.com/Ali-Forootani/compactformer or https://github.com/Ali-Forootani/compact-transformers/tree/main.

[5] https://zenodo.org/records/15518817, https://zenodo.org/records/15518836.

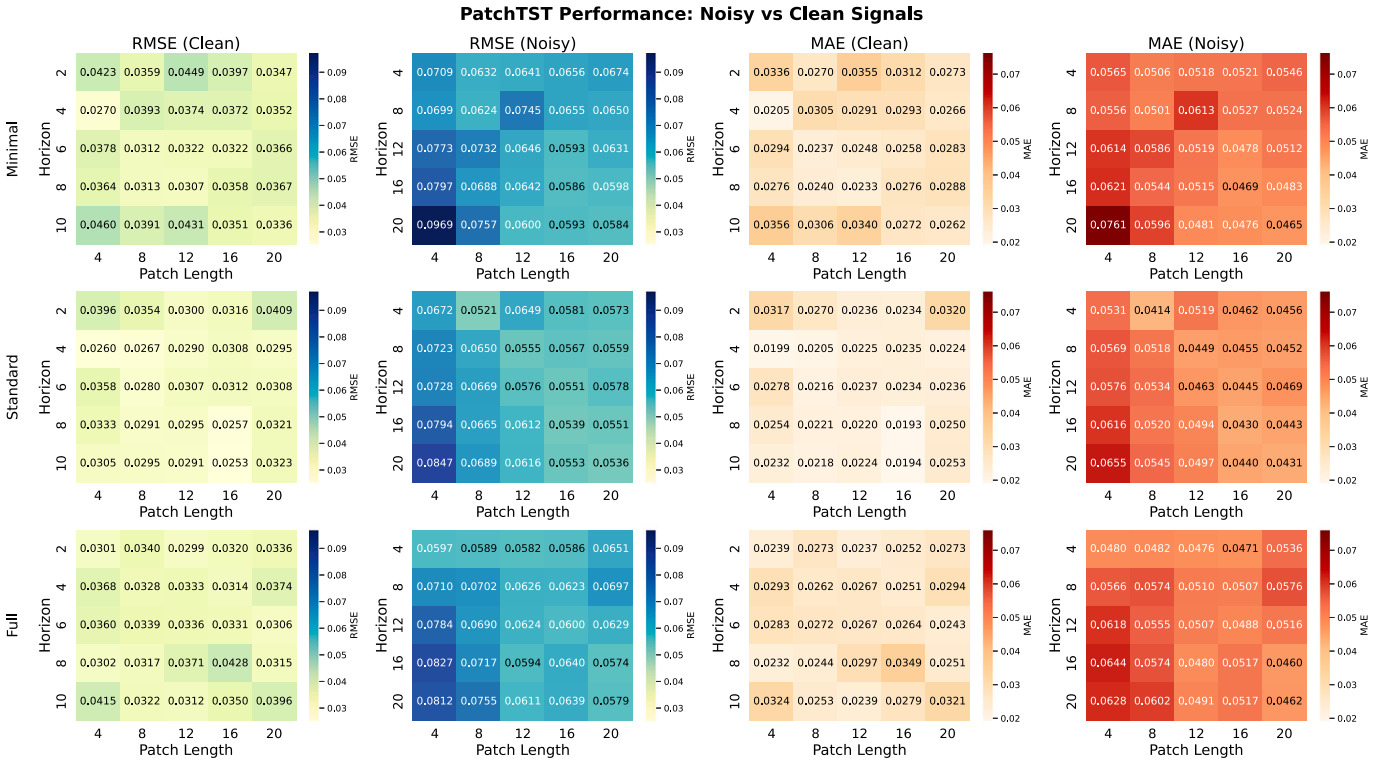

Fig. 1. Performance of `PatchTST` variants (Minimal, Standard, Full) on various patch lengths and forecast horizons, averaged over all signals. Left column: RMSE heatmaps. Right column: MAE heatmaps.

Table 4
Best `PatchTST` model per signal based on RMSE & MAE (Clean signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd Order Poly	Standard	8	4	0.0389	0.0297
Cos + Trend	Minimal	20	16	0.0020	0.0015
CosEnv × Sine	Minimal	20	12	0.0027	0.0020
Cubic Poly	Full	12	8	0.0389	0.0297
ExpDec × Sine	Full	12	20	0.0000	0.0000
Exp Growth	Standard	20	20	0.0575	0.0418
Gauss Bump	Standard	12	4	0.0607	0.0496
Log × Sine	Standard	20	4	0.0076	0.0062
Long Sine	Full	12	4	0.0706	0.0528
Sine	Standard	20	8	0.0001	0.0001

Table 5
Best `PatchTST` model per signal based on RMSE & MAE (Noisy signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd Order Poly	Standard	8	4	0.1015	0.0828
Cos + Trend	Minimal	20	16	0.0216	0.0175
CosEnv × Sine	Minimal	20	12	0.0338	0.0269
Cubic Poly	Full	12	8	0.0357	0.0279
ExpDec × Sine	Full	12	20	0.0447	0.0357
Exp Growth	Standard	20	20	0.0782	0.0613
Gauss Bump	Standard	12	4	0.0607	0.0496
Log × Sine	Standard	20	4	0.0076	0.0062
Long Sine	Full	12	4	0.0374	0.0312
Sine	Standard	20	8	0.0364	0.0300

Sensitivity to patch and horizon lengths. As demonstrated in the figures, all `PatchTST` variants exhibit varying degrees of sensitivity to patch length and forecast horizon. Generally, increasing the patch length up to 12 or 16 improves forecasting accuracy, though performance gains level off or even degrade slightly for longer patch lengths in noisy conditions. The Minimal variant is most effective for short forecast horizons, while the Standard variant maintains strong performance across a broader range of horizons. The Full variant tends to perform better on longer horizons, especially when dealing with complex or noisy signals.

7.5. Evaluation of `Informer` Variants under clean and noisy conditions

We evaluate three `Informer` variants—Minimal, Standard, and Full—across a grid of five patch lengths and five forecast horizons for each of ten synthetic signals, resulting in a total of 750 configurations ($3 \times 5 \times 5 \times 10$) per dataset. To provide a comprehensive overview of model behavior, Fig. 2 displays heatmaps of mean RMSE and MAE values, averaged across all signals. Results are reported separately for clean and noisy inputs. These visualizations help identify performance trends with respect to model complexity, patch size, and forecast horizon. Table 6 highlights the best-performing configuration for each signal under clean conditions, while Table 7 shows the corresponding results for noisy signals. These tables present the best-performing `Informer` model for each signal type based on RMSE and MAE values, highlighting the optimal configurations of patch length and forecast horizon for both clean and noisy signals.

Minimal variant performance. The `Informer` Minimal variant performs well across different configurations, showing the lowest RMSE and MAE values for clean signals, especially in shorter forecast horizons and smaller patch lengths. For clean signals, it achieves its best results at patch lengths of 4 and 8, with RMSE values as low as 0.0491 and MAE values around 0.0410. As the forecast horizon increases, the performance slightly degrades, but the model remains effective across various configurations.

In noisy environments, the Minimal variant experiences a reduction in performance compared to clean signals, but it still performs relatively well. The RMSE values for noisy signals range from 0.0699 to 0.0964, and the MAE ranges from 0.0559 to 0.0796, with the best performance observed at shorter forecast horizons (2 and 4) and smaller patch lengths

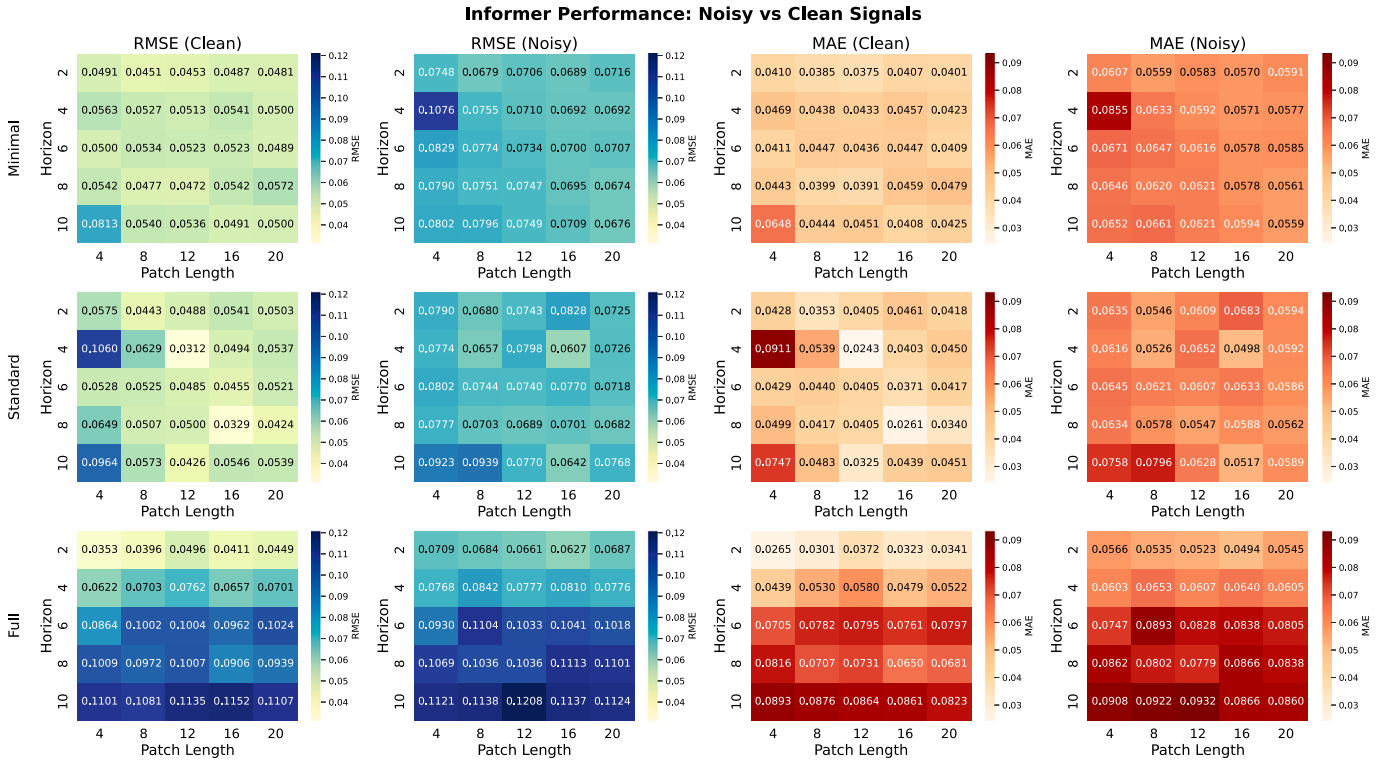

Fig. 2. Performance of `Informer` variants (Minimal, Standard, Full) on various patch lengths and forecast horizons, averaged over all signals.

Table 6
Best `Informer` model per signal based on RMSE & MAE (Clean signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd Order Polynomial	Standard	12	10	0.0421	0.0309
Cosine + Trend	Standard	12	2	0.0127	0.0100
CosEnvelope × Sine	Minimal	20	2	0.0068	0.0052
Cubic Polynomial	Full	4	8	0.0400	0.0300
Exp. Decay × Sine	Minimal	16	2	0.0002	0.0001
Exponential Growth	Standard	12	4	0.0271	0.0206
Gaussian Bump	Standard	8	6	0.0006	0.0005
Log × Sine	Full	20	2	0.0042	0.0038
Long Period Sine	Full	20	2	0.0014	0.0011
Sine	Full	8	2	0.0011	0.0009

Table 7
Best `Informer` model per signal based on RMSE & MAE (Noisy signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd Order polynomial	Full	16	4	0.0832	0.0677
Cosine + Trend	Minimal	20	10	0.0172	0.0132
CosEnvelope × Sine	Standard	16	2	0.0379	0.0301
Cubic polynomial	Full	20	10	0.0290	0.0200
Exp. Decay × Sine	Minimal	8	2	0.0448	0.0343
Exponential growth	Full	4	2	0.0597	0.0453
Gaussian bump	Minimal	4	2	0.0589	0.0463
Log × Sine	Standard	12	2	0.0093	0.0071
Long period sine	Minimal	12	4	0.0375	0.0304
Sine	Standard	20	8	0.0368	0.0299

(4 and 8). Despite the added noise, the Minimal variant remains robust, especially in configurations with shorter patch lengths and forecast horizons.

Standard variant performance. The `Informer` Standard variant shows strong performance across a variety of configurations, with consistent results for both clean and noisy signals. For clean signals, it achieves its best performance at a patch length of 12 and a forecast horizon of 4 or 8, where it records RMSE values as low as 0.0525 and MAE values as low as 0.0399. As the forecast horizon increases, the model maintains strong performance, though slight increases in RMSE and MAE can be observed, particularly at longer patch lengths.

In noisy environments, the Standard variant's performance does degrade slightly compared to clean signals, but it still shows resilience. The RMSE values for noisy signals range from 0.0680 to 0.1060, and the MAE ranges from 0.0546 to 0.0796. The best performance is achieved with smaller patch lengths (4 and 8) and shorter forecast horizons (2 and 4). Despite the added noise, the Standard variant performs admirably across most configurations, making it a versatile choice for a wide range of time series forecasting tasks.

Full variant. The `Informer` Full variant demonstrates strong performance across a variety of patch lengths and forecast horizons, particularly for noisy signals. For clean signals, the Full variant performs well across all configurations, with the best results typically seen with patch lengths of 12 and 16 and forecast horizons of 4 or 8. It achieves RMSE values as low as 0.0353 and MAE values around 0.0265. The performance remains solid as the forecast horizon increases, although the RMSE and MAE do rise slightly as the patch length and horizon increase.

In noisy environments, the Full variant still performs well but with a noticeable decline compared to clean signals. The RMSE for noisy signals ranges from 0.0709 to 0.1101, and the MAE varies between 0.0566 and 0.0876. The best performance is typically observed at smaller patch lengths (4 and 8) and shorter forecast horizons (2 and 4). Despite this reduction in performance due to noise, the Full variant maintains robust forecasting accuracy, especially for longer horizons.

Sensitivity to patch and horizon lengths. All three variants exhibit sensitivity to both patch size and forecast horizon. On clean signals, shorter horizons (up to 4 steps) yield the best RMSE and MAE values for Minimal and Standard models. In contrast, the Full variant maintains stable performance even at longer horizons, reflecting its superior ability to abstract temporal structure. Across variants, optimal patch lengths are

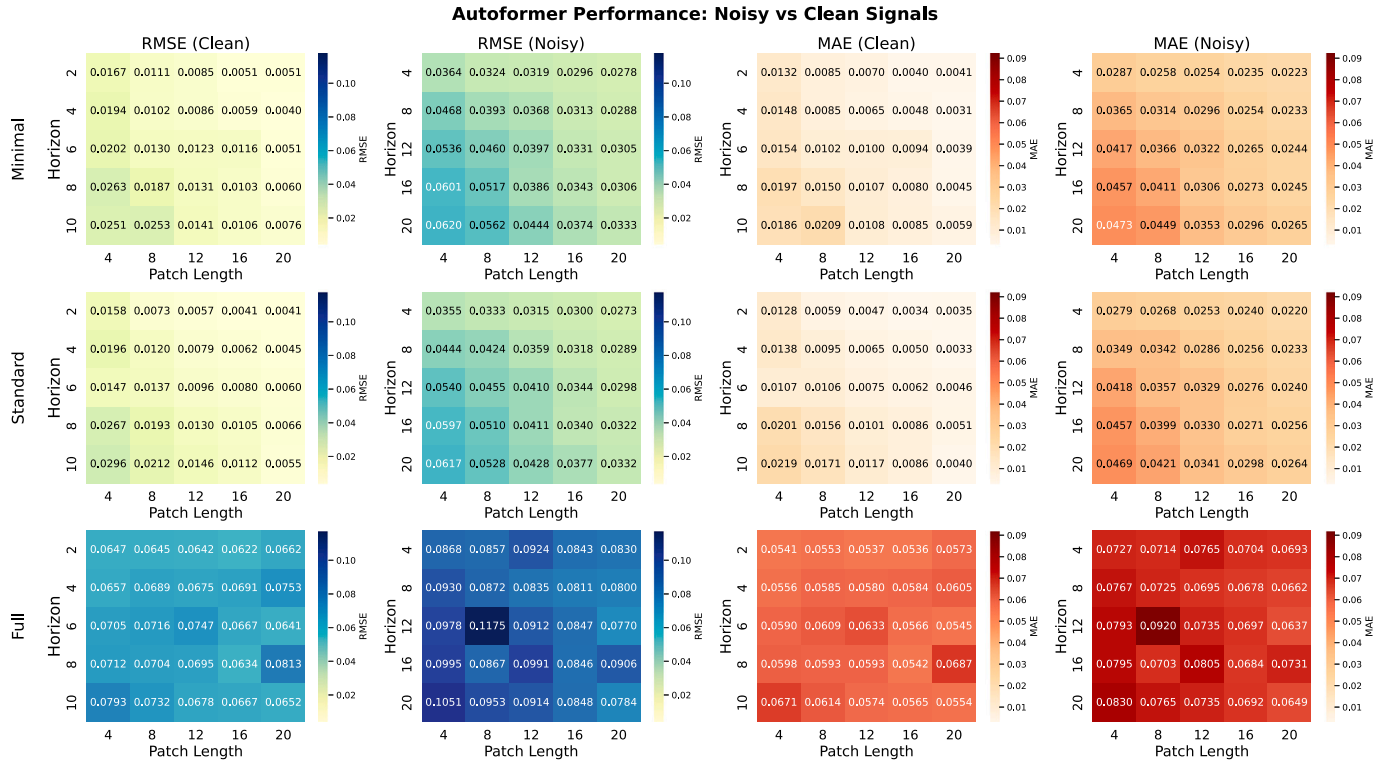

Fig. 3. Performance of `Autoformer` variants (Minimal, Standard, Full) on various patch lengths and forecast horizons, averaged over all signals.

typically found in the 12 to 16 range. Short patches (e.g., length 4) often lead to underfitting, while longer patches (e.g., length 20) occasionally degrade performance, particularly under noisy conditions. This suggests a need to balance context length with model capacity to avoid overfitting or loss of resolution.

7.6. Evaluation of `Autoformer` Variants under clean and noisy conditions

We evaluated three `Autoformer` variants—Minimal, Standard, and Full—on a collection of ten synthetic time series. Each experiment spans five patch lengths and five forecast horizons, resulting in a total of 750 configurations per setting ($3 \times 5 \times 5 \times 10$). Fig. 3 presents heatmaps of the mean RMSE and MAE values, averaged across all signals, under both clean and noisy conditions. Table 8 highlights the best-performing configuration for each signal under clean conditions, while Table 9 shows the corresponding results for noisy signals. These tables display the top-performing `Autoformer` model for each signal type according to RMSE and MAE metrics, emphasizing the best patch length and forecast horizon settings for both clean and noisy signals.

Minimal variant. The `Autoformer` Minimal variant demonstrates strong performance in both clean and noisy environments, with particularly low RMSE and MAE values for clean signals. For clean signals, the best performance is achieved with smaller patch lengths (4 and 8) and shorter forecast horizons (2 and 4). The RMSE values range from 0.0051 to 0.0167, while the MAE values range from 0.0031 to 0.0132, showing that the Minimal variant excels in predicting simple signals with high accuracy at these configurations.

In noisy conditions, the Minimal variant exhibits some decline in performance but continues to perform well across a variety of configurations. The RMSE values for noisy signals range from 0.0278 to 0.0562, and the MAE values range from 0.0220 to 0.0457, indicating that it remains relatively robust in noisy scenarios. The best performance in noisy environments is observed at shorter patch lengths (4 and 8) and horizons of 2 and 4.

Table 8
Best `Autoformer` model per signal based on RMSE & MAE (Clean signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd order polynomial	Standard	4	2	0.0081	0.0064
Cosine + Trend	Minimal	20	4	0.0007	0.0007
CosEnvelope × Sine	Standard	16	2	0.0020	0.0017
Cubic polynomial	Minimal	16	4	0.0020	0.0016
Exp. Decay × Sine	Minimal	12	6	0.0001	0.0001
Exponential growth	Standard	8	2	0.0001	0.0001
Gaussian bump	Standard	12	2	0.0002	0.0002
Log × Sine	Minimal	16	2	0.0022	0.0020
Long period sine	Minimal	20	10	0.0005	0.0004
Sine	Standard	20	8	0.0004	0.0003

Table 9
Best `Autoformer` model per signal based on RMSE & MAE (Noisy signals).

Signal	Model	Patch	Horizon	RMSE	MAE
2nd order polynomial	Minimal	12	4	0.0064	0.0052
Cosine + Trend	Minimal	20	4	0.0138	0.0105
CosEnvelope × Sine	Full	20	8	0.0392	0.0314
Cubic polynomial	Minimal	16	8	0.0033	0.0027
Exp. Decay × Sine	Full	12	20	0.0444	0.0354
Exponential growth	Minimal	20	12	0.0072	0.0057
Gaussian bump	Full	12	12	0.0591	0.0482
Log × Sine	Minimal	16	8	0.0074	0.0059
Long period sine	Full	16	4	0.0367	0.0285
Sine	Standard	20	4	0.0358	0.0294

Standard variant. The `Autoformer` Standard variant shows consistent and reliable performance across various patch lengths and forecast horizons, with both clean and noisy signals. For clean signals, it achieves its best performance with patch lengths of 4 and 8 and shorter forecast horizons (2 and 4). The RMSE values for clean signals range from 0.0147 to 0.0296, and the MAE values range from 0.0057 to 0.0219, demonstrating its ability to capture the patterns in simpler signals effectively.

In noisy environments, the Standard variant experiences a slight degradation in performance, but it still performs relatively well. The RMSE for noisy signals ranges from 0.0335 to 0.0617, and the MAE ranges from 0.0235 to 0.0469. The best results in noisy conditions are typically seen at shorter patch lengths (4 and 8) and shorter forecast horizons (2 and 4), where the model maintains a good balance between accuracy and computational efficiency.

Full variant. The `Autoformer` Full variant shows strong performance, especially in noisy environments, where its ability to model complex patterns comes to the fore. For clean signals, the Full variant performs well across all configurations. The best performance is observed with shorter patch lengths (4 and 8) and forecast horizons of 4 and 8, achieving RMSE values as low as 0.0705 and MAE values around 0.0541. As the patch length and horizon increase, the performance remains stable, although there is a slight increase in RMSE and MAE values.

In noisy conditions, the Full variant still maintains strong performance, though it experiences a noticeable decline compared to clean signals. The RMSE for noisy signals ranges from 0.0793 to 0.1101, and the MAE ranges from 0.0541 to 0.0731. The best performance in noisy conditions is generally seen with shorter patch lengths (4 and 8) and forecast horizons of 2 and 4. Despite the noise, the Full variant continues to capture complex temporal dependencies effectively.

Sensitivity to patch and horizon lengths. Across both clean and noisy conditions, the most accurate forecasts are observed when patch lengths range from 12 to 16. Shorter patches (e.g., length 4) generally lead to underfitting due to insufficient context, while very long patches (e.g., length 20) introduce variability and can degrade performance. Forecast accuracy declines with increasing horizon length across all variants, but this degradation is least severe for the Standard variant. In contrast, the Minimal and Full variants experience sharper performance drops at longer horizons, particularly under noisy conditions.

7.7. Interpreting noise effects across model variants

The three noise components introduced earlier, i.e., additive Gaussian noise, multiplicative amplitude jitter, and random temporal shifts—stress different aspects of the Transformer architectures, and their impact varies systematically across the `Autoformer`, `PatchTST`, and `Informer` families. Additive Gaussian noise primarily injects high-frequency perturbations. As predicted by the theoretical results in Section 6, this type of noise is most effectively suppressed by architectures that perform temporal averaging: `Autoformer` reduces variance through its moving-average trend filter, while `PatchTST` reduces variance via patch-wise averaging. `Informer` variants, which do not explicitly smooth in time, are therefore more sensitive to Gaussian noise unless the `ProbSparse` mechanism successfully concentrates attention on a small set of informative keys.

Multiplicative amplitude jitter introduces scale fluctuations over time. `PatchTST` variants exhibit strong robustness in this regime because patch averaging normalizes local amplitude changes, and `Autoformer` absorbs part of the jitter into the trend channel before modeling the seasonal residuals. In contrast, `Informer` variants—especially the Minimal version—are more affected because amplitude variations directly modify attention scores and can disrupt the sparsity structure used by `ProbSparse` attention.

Random temporal shifts introduce local misalignment of events and phases. These perturbations primarily test the ability to model long-range and phase-invariant temporal dependencies. `Informer` Standard and Full benefit from `ProbSparse` attention when dominant keys remain well separated, while `PatchTST` mitigates small shifts through patch-level aggregation. `Autoformer` is more sensitive to such shifts in the seasonal component, although its trend pathway remains largely unaffected. These patterns explain the robustness trends observed in the heatmaps of Figs. 1–3 and clarify how each architectural inductive bias interacts with different noise sources.

7.8. From compact transformers to koopman-enhanced forecasting

The architectural variants developed in this article are not intended as terminal models, but as *screening backbones* for physics-aware and operator-theoretic forecasting pipelines. The systematic analysis in previous part identifies which Transformer design choices—patching, sparse attention, or trend–seasonal decomposition—provide the most favorable robustness–complexity trade-offs under noise and long-horizon forecasting.

These findings directly motivate the `DeepKoopFormer` framework [32], in which the best-performing compact encoders are embedded into a Koopman-constrained latent dynamical system. In this architecture, the Transformer is used solely as a representation learner, while temporal evolution is governed by a spectrally stable linear operator that enforces Lyapunov dissipation, bounded gradients, and provable error decay. This resolves the fundamental limitations of standard Transformers—namely instability, lack of interpretability, and uncontrolled long-horizon drift—while preserving their expressive modeling capacity.

7.8.1. `DeepKoopFormer` Building blocks

`DeepKoopFormer` is a modular forecasting architecture that combines a Transformer backbone for representation learning with a Koopman-inspired latent refinement layer that enforces stability and improves conditioning. The model follows an encoder–Koopman–decoder pipeline, where the backbone captures nonlinear temporal structure and the Koopman layer regularizes the latent transition.

Transformer encoder (representation learning). Given a historical input window $X_t = [\mathbf{x}_{t-P+1}, \dots, \mathbf{x}_t] \in \mathbb{R}^{P\times d}$, a Transformer-based encoder $\mathcal{E}_\theta$ maps the sequence to a latent state

$$\mathbf{z}_t = \mathcal{E}_\theta(X_t), \qquad \mathbf{z}_t \in \mathbb{R}^{d_{\text{latent}}}, \tag{40}$$

where θ denotes the encoder parameters. The self-attention mechanism enables learning both local and global temporal dependencies, and the latent dimension d_{latent} can differ from the backbone embedding width (i.e., $d_{\text{latent}} \neq d_{\text{model}}$).

Koopman latent refinement (spectrally stable transition). Instead of an unconstrained latent update, `DeepKoopFormer` applies a linear Koopman map to the encoded representation:

$$\mathbf{z}_{t+1} = \mathbf{K}\,\mathbf{z}_t, \tag{41}$$

where $\mathbf{K}$ is parameterized via an orthogonal–diagonal–orthogonal (ODO) factorization

$$\mathbf{K} = \mathbf{U}\,\mathrm{diag}(\mathbf{\Sigma})\,\mathbf{V}^\top, \tag{42}$$

$$\mathbf{U}^\top\mathbf{U} = \mathbf{I}, \qquad \mathbf{V}^\top\mathbf{V} = \mathbf{I}, \tag{43}$$

$$\Sigma_i = \rho_{\max}\,\sigma(\Sigma_i^{\text{raw}}), \qquad 0 < \rho_{\max} < 1, \tag{44}$$

with a squashing nonlinearity $\sigma(\cdot)$ (e.g., `Sigmoid`) to ensure bounded singular values. Consequently,

$$\|\mathbf{K}\|_2 = \max_i \Sigma_i \le \rho_{\max} < 1, \tag{45}$$

which yields a contractive latent transition and uniformly bounded gradients. In our forecasting pipeline, this Koopman layer acts as a *stability-inducing projection* on the encoded features. In particular, the backbone remains responsible for learning trends, seasonality, and long-range dependencies, while $\mathbf{K}$ regularizes the latent dynamics and mitigates error amplification. For `Autoformer`, the explicit trend pathway bypasses the Koopman refinement, and the Koopman layer primarily governs residual (seasonal/stationary) dynamics.

Linear decoder (forecast readout). Predictions are obtained via a linear readout from the refined latent state:

$$\hat{\mathbf{x}}_{t+1} = \mathcal{D}_\phi(\mathbf{z}_{t+1}) = \mathbf{W}\,\mathbf{z}_{t+1}, \qquad \mathbf{W} \in \mathbb{R}^{d \times d_{\text{latent}}}, \tag{46}$$

leading to the end-to-end mapping

$$\hat{\mathbf{x}}_{t+1} = \mathcal{D}_\phi\big(\mathbf{K}\,\mathcal{E}_\theta(X_t)\big)\,. \tag{47}$$

The linear decoder preserves interpretability (Koopman-mode-style readout) and supports closed-form stability analysis.

Stability regularization (Lyapunov penalty). To discourage transient growth in the latent space, training includes a Lyapunov-inspired penalty

$$\mathcal{L}_{\text{Lyap}} = \lambda\,\text{ReLU}\big(\|\mathbf{z}_{t+1}\|^2 - \|\mathbf{z}_t\|^2\big)\,, \qquad \lambda > 0, \tag{48}$$

combined with the forecasting loss:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Forecast}} + \lambda_{\text{Lyap}}\,\mathcal{L}_{\text{Lyap}}. \tag{49}$$

While (45) already ensures contraction in operator norm, $\mathcal{L}_{\text{Lyap}}$ promotes sample-wise energy decay and stabilizes optimization.

Operator-level robustness guarantee. The spectral constraint implies exponentially decaying sensitivity of latent (and decoded) trajectories to perturbations under repeated application:

$$\|\hat{\mathbf{x}}_{t+h} - \tilde{\mathbf{x}}_{t+h}\| \le \|\mathbf{W}\|_2\,\rho_{\max}^{h}\,\|\mathbf{z}_t - \tilde{\mathbf{z}}_t\|. \tag{50}$$

This bound is *operator-theoretic* characterizes controlled error propagation through the Koopman transition and does not substitute for a full generalization bound of the learned encoder. Nevertheless, it formalizes the intended role of the Koopman layer as a stability-certified latent refinement module within an otherwise expressive Transformer forecasting backbone.

To rigorously evaluate the proposed `DeepKoopFormer` framework across a broad spectrum of dynamical behaviors, we explicitly integrate the *compact Transformer variants* developed in our companion study on *compact Transformer variants* into the Koopman–enhanced pipeline. Rather than designing a bespoke encoder, `DeepKoopFormer` is constructed as a *plug-and-play operator-theoretic wrapper* around the three systematically benchmarked Transformer families—`PatchTST`, `Autoformer`, and `Informer`—each in their Minimal, Standard, and Full variants.

This integration allows us to lift the compact backbones—whose noise robustness, bias–variance trade-offs, and computational scaling were analytically and empirically characterized in the Compact-Transformers study—into a stability-certified dynamical forecasting architecture. Concretely, each compact Transformer variant serves as the nonlinear encoder $\mathcal{E}_\theta$, while a shared spectrally-constrained Koopman layer provides stability, conditioning, and error-control in latent space. This construction yields a unified family of models: `DeepKoopFormer-PatchTST`, `DeepKoopFormer-Autoformer`, `DeepKoopFormer-Informer`, where the Koopman operator is identical across all variants and only the compact Transformer backbone is changed.

We include CMIP6 climate simulations, which test the interaction between structured physical dynamics and data-driven representation learning.

Across all experiments, we compare the Koopman-augmented compact Transformers against a standard LSTM baseline under identical training protocols. This experimental design isolates the effect of Koopman augmentation while preserving the inductive biases of the compact Transformer variants identified in the preceding study. To ensure fair comparison across architectures, we set $d_{\text{latent}} = d_{\text{model}}$ throughout; however, these quantities play conceptually different roles, with d_{model} governing representation capacity and d_{latent} defining the dimension of the Koopman-refined dynamical subspace.

7.8.2. Wind speed time series forecasting

In this experiment, we conduct systematic grid sweeps over the latent dimension

$$d_{\text{model}} \in \{8, 16, 24, 32, 40, 48\},$$

and the forecast horizon

$$H \in \{10, 15, 20, 25, 30\}.$$

All transformer-based backbones are implemented with three encoder layers, four attention heads, sinusoidal positional encodings, and a feed-forward network width of 96. To ensure a fair comparison in terms of representational capacity, the LSTM baseline uses two stacked recurrent layers with hidden size $d_{\text{model}} \in \{8, 16, 24, 32, 40, 48\}$.

Fig. 4 presents a comprehensive comparison of wind speed forecasting accuracy for the LSTM and the three `DeepKoopFormer` variants (`PatchTST`, `Autoformer`, and `Informer`), where d_{model} is varied to align the model capacity across all architectures. The results indicate that all Koopman-augmented transformer models consistently outperform the LSTM baseline in both MSE and MAE over nearly the entire grid of configurations. This advantage becomes increasingly pronounced as both the latent dimension and the forecast horizon grow. Although enlarging the LSTM hidden state improves its performance, it remains systematically inferior to the transformer-based approaches, especially for long-horizon forecasts. Among the `DeepKoopFormer` variants, `PatchTST` and `Informer` achieve the lowest error levels, highlighting the effectiveness of patch-based representations and sparse attention mechanisms in capturing the complex, multi-scale dynamics of wind speed. The `Autoformer` also performs strongly, typically exceeding the LSTM but generally lagging behind `PatchTST` and `Informer`. Overall, these results demonstrate the superior scalability, robustness, and predictive accuracy of Koopman-augmented transformer architectures for multistep, multivariate wind speed forecasting, while emphasizing the intrinsic limitations of purely recurrent models.

Fig. 5 illustrates representative prediction trajectories at five spatial locations for a fixed patch length of $p = 120$, a forecast horizon of $H = 25$, and $d_{\text{model}} = 16$. Across all channels, the `DeepKoopFormer` variants closely follow the ground-truth wind speed signals and consistently outperform the LSTM, particularly in capturing sharp peaks and rapid temporal fluctuations.

7.8.3. Pressure surface time series forecasting

For surface pressure forecasting, we employ a unified experimental protocol across four architectures: Koopman-augmented `PatchTST`, `Autoformer`, `Informer`, and an LSTM baseline. All models operate on input windows of fixed length $P = 120$ with up to five spatial channels and produce multi-step predictions over forecast horizons $H \in \{10, 15, 20, 25, 30\}$. For the transformer-based backbones, the latent dimension is set to $d_{\text{model}} = 48$, using three encoder layers with four attention heads per layer. The patch-based models (`PatchTST` and `Informer`) adopt patch lengths

$$p \in \{80, 90, 100, 110, 120, 130\},$$

with each patch projected into the latent space prior to temporal encoding. The `Autoformer` instead applies a moving-average window of size p to extract trend components. The LSTM baseline consists of two recurrent layers with hidden size 64, and all models use a linear output head for multi-step forecasting.

The quantitative comparisons in Fig. 6 show that all methods achieve consistently low errors across the full range of patch sizes and forecast horizons. Both the LSTM and the `DeepKoopFormer` variants typically attain MSE values in the range 10^{-4}–10^{-3}, with MAE remaining below 0.03 in most settings. Among the transformer-based approaches, `PatchTST` and `Informer` exhibit the strongest and most stable performance, often surpassing the LSTM, especially for longer horizons and

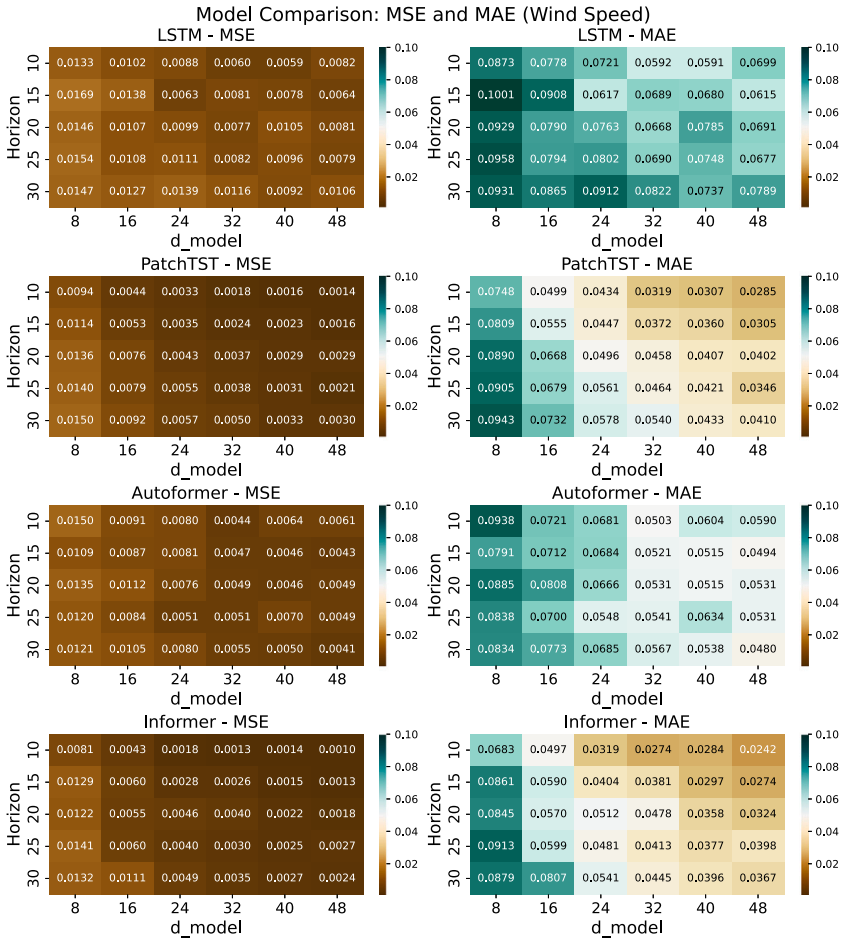

Fig. 4. `DeepKoopFormer`, built by augmenting compact Transformer backbones with a Koopman latent dynamics module, compared against the LSTM baseline for CMIP6 wind speed forecasting under Scenario Section 7.8.2.

larger patch sizes. The `Autoformer` also remains competitive, although it displays slightly elevated MAE in certain configurations, which may be attributed to the limited benefit of trend–seasonal decomposition for the comparatively smooth and slowly varying surface pressure dynamics. At shorter horizons and smaller patch sizes, the LSTM remains highly competitive, highlighting its robustness on near-stationary time series. Overall, these results indicate that surface pressure—characterized by regular and low-volatility temporal structure—can be forecast accurately by both recurrent and transformer-based models, with the transformer variants offering a modest advantage in longer-horizon and higher-capacity regimes. This conclusion is further corroborated in Fig. 7, where all models produce closely aligned prediction trajectories.

7.9. Financial time series forecasting

We employ a large-scale cryptocurrency market dataset consisting of 27,585 daily records covering the top 10 cryptocurrencies traded on the Binance Exchange from September 1, 2018 to September 5, 2022. The dataset contains essential market attributes, including opening, closing, high, and low prices, trading volume, and market capitalization, thereby offering a comprehensive temporal and cross-sectional view of cryptocurrency market behavior.[6]

For this experimental setup, we utilize the `Cryptocurrency` dataset and preprocess the `ClosePrice` time series by dividing it into two equal segments, each comprising 2,900 samples. This procedure yields a two-channel input array of size $(2900, 2)$, enabling the `DeepKoopFormer` model to exploit parallel streams of historical price information within a multi-channel forecasting framework. The dataset is subsequently split into training and testing subsets following an 80%/20% ratio.

The core hyperparameters of the `DeepKoopFormer`-based architectures include an encoder with three layers, each equipped with four attention heads, and a latent embedding dimension of $d_{\text{model}} = 48$. Temporal patching is implemented by varying the patch length over the set $p = \{40, 50, 60, 70, 80, 90, 100\}$, while the prediction horizon is swept across $H = \{2, 4, 6, 8, 10, 12, 14, 16\}$. All models are trained using a fixed input sequence length and optimized for up to 4,000 epochs with the Adam optimizer and a learning rate of 3×10^{-4}.

For the LSTM baseline, we adopt a two-layer configuration with 48 hidden units per layer. Transformer-based baselines incorporate positional encodings and, when applicable, patch embedding mechanisms or series decomposition strategies. All `DeepKoopFormer` variants enforce strict Koopman operator stability through Lyapunov regularization with a penalty coefficient of $\lambda = 0.1$. Model performance is assessed using mean squared error (MSE) and mean absolute error (MAE), with results reported across all combinations of model type, patch length, and forecast horizon for both training and testing sets.

The inherent diversity and high volatility of cryptocurrency markets, arising from heterogeneous assets and rapidly changing market conditions, provide a challenging benchmark for evaluating the capability of the proposed models to learn complex nonlinear temporal dynamics. Experimental results on this real-world dataset demonstrate the strong predictive performance and generalization potential

[6] https://github.com/Chisomnwa/Cryptocurrency-Data-Analysis.

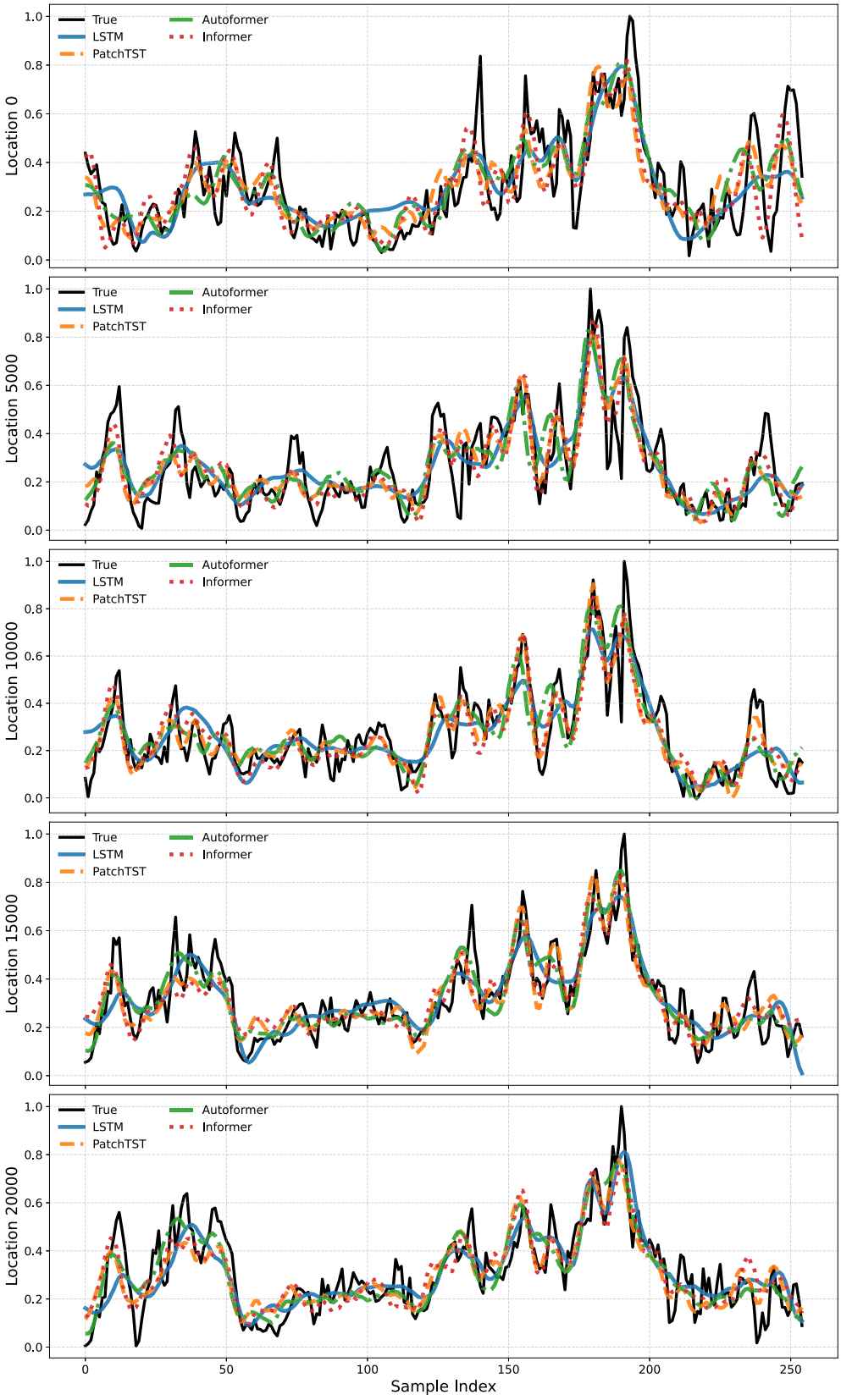

Fig. 5. Comparison of `DeepKoopFormer`—constructed by augmenting compact Transformer backbones with Koopman latent dynamics—and the LSTM baseline for CMIP6 wind speed forecasting with $p = 120$, $H = 25$, and $d_{\text{model}} = 16$.

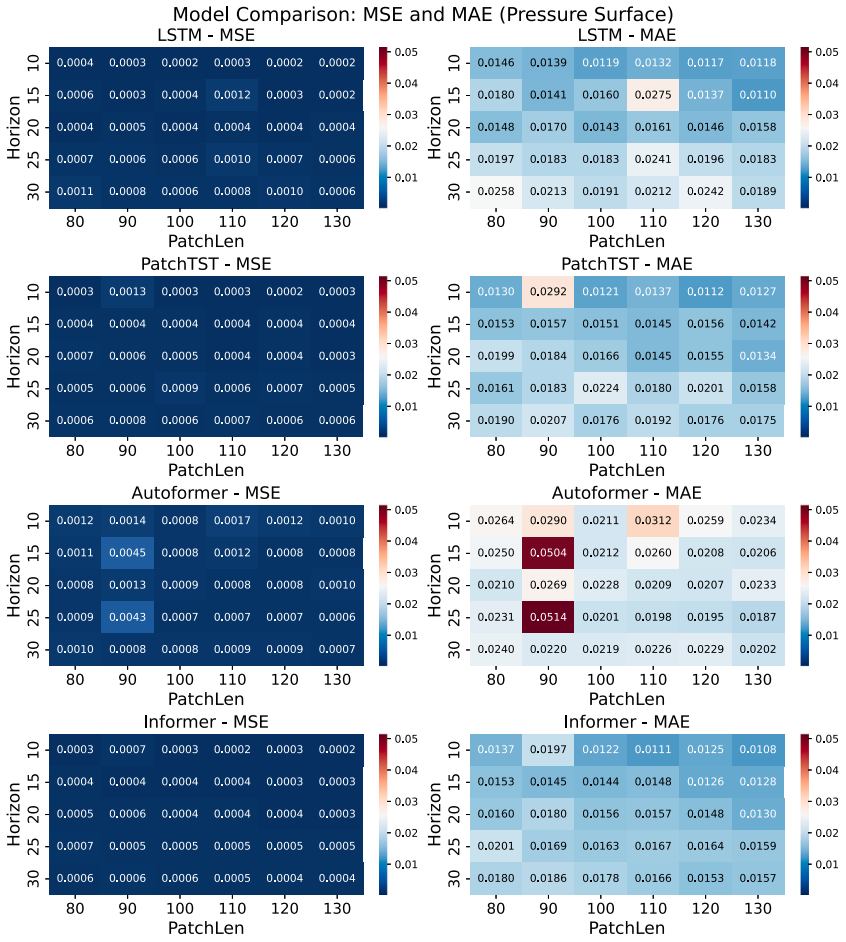

Fig. 6. `DeepKoopFormer`—constructed by augmenting compact Transformer backbones with Koopman latent dynamics—versus the LSTM baseline for CMIP6 surface pressure forecasting in Section 7.8.3.

of `DeepKoopFormer` for cryptocurrency price forecasting under varying market regimes.

The training heatmaps shown in Fig. 8 indicate that all models achieve relatively low error values across the explored ranges of patch lengths and forecast horizons, reflecting their ability to effectively fit the training data. Among the evaluated methods, the LSTM baseline consistently attains the lowest MSE values, particularly for short-term forecasts (e.g., $H \leq 6$), where simpler recurrent structures are well-suited to capturing short-range temporal dependencies. In contrast, `DeepKoopFormer`-based `PatchTST` models exhibit competitive performance, especially at larger patch sizes (e.g., $p \geq 80$), highlighting their advantage in learning richer temporal representations. Both `Autoformer` and `Informer` also demonstrate strong training performance, although `Autoformer` shows slightly higher variability in MAE under certain configurations. Notably, `PatchTST` and `Informer` present smoother and more stable error landscapes, suggesting their robustness in extracting latent temporal structures over broad ranges of horizons and patch lengths. Overall, these observations confirm the effectiveness of Transformer-based architectures, particularly `DeepKoopFormer` variants, in leveraging inductive biases during training on highly dynamic cryptocurrency data.

Generalization performance on the test set, illustrated in Fig. 9, reveals more pronounced differences among the competing approaches. Although LSTM performs well during training, it exhibits a substantial increase in MAE as both the patch length and prediction horizon grow, indicating limited generalization beyond short-term temporal patterns. In contrast, `DeepKoopFormer` variants, especially `PatchTST`, demonstrate consistently strong performance across a wide range of configurations, maintaining low MSE and MAE values even at larger patch sizes ($p \geq 70$) and longer forecasting horizons. Meanwhile, `Autoformer` and `Informer` tend to incur higher test errors, with `Autoformer` occasionally exhibiting peaks in MAE (e.g., $H = 2$, $p = 80$), which may reflect sensitivity to specific hyperparameter settings or mild overfitting. The ability of `DeepKoopFormer` to preserve low test errors across diverse experimental regimes underscores its effectiveness in capturing the underlying Koopman dynamics governing complex and volatile financial time series. Such robustness is particularly desirable for real-world financial forecasting applications, where stability and adaptability across evolving market conditions are critical.

To further evaluate predictive accuracy, we present qualitative comparisons between model forecasts and ground truth signals in Figs. 10 and Fig. 11. These visualizations correspond to two representative test windows using a forecast horizon of $H = 16$ and an embedding dimension of $d_{\text{model}} = 48$, with patch lengths set to $p = 40$ and $p = 80$, respectively.

Across both experimental configurations, the LSTM baseline consistently lags behind Transformer-based approaches. In particular, LSTM exhibits delayed responses to abrupt trend changes and fails to accurately reproduce the magnitude of sharp price fluctuations. This shortcoming is especially evident in Time Window 1 of Fig. 10, where LSTM significantly underestimates the signal during a rapid downward movement. In contrast, `DeepKoopFormer` variants achieve closer alignment with the ground truth, more accurately capturing turning points and local oscillations.

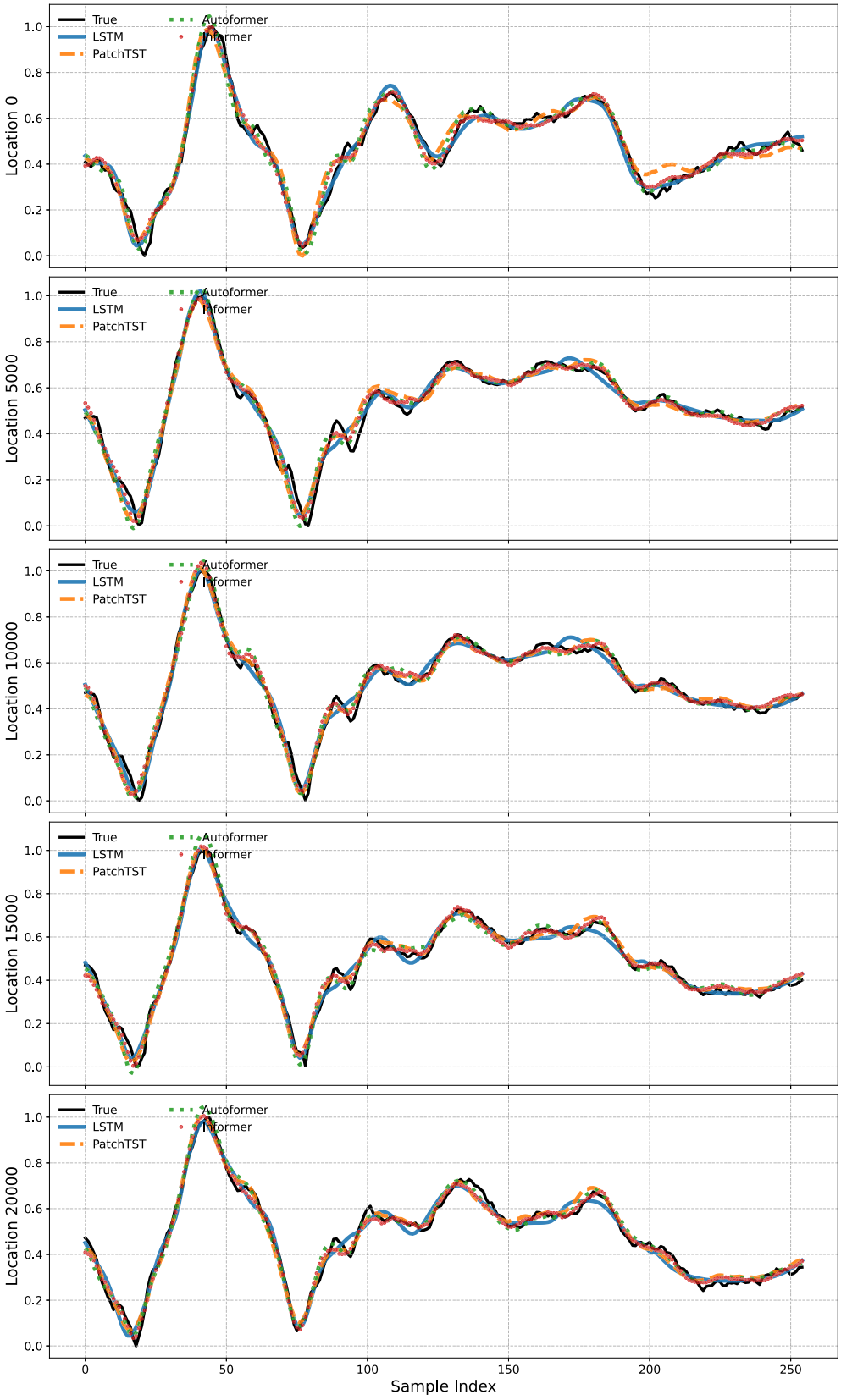

Fig. 7. Surface pressure forecasting on CMIP6 using `DeepKoopFormer`—built by augmenting compact Transformer backbones with Koopman latent dynamics—with $p = 120$, $H = 25$, and $d_{\text{model}} = 48$; the LSTM baseline uses a hidden size of 64.

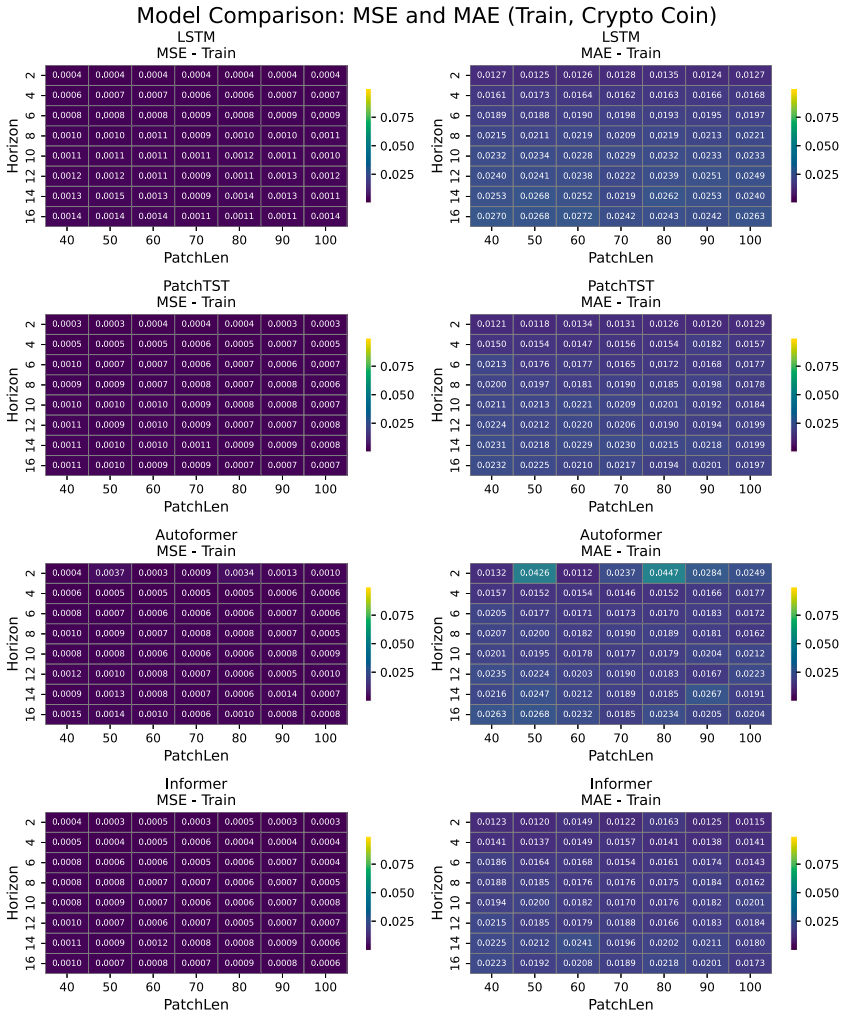

Fig. 8. `DeepKoopFormer` models compared with the LSTM baseline across a grid of hyperparameter settings on the training set for financial time series forecasting in Section 7.9.

The advantage of `DeepKoopFormer`-based models becomes even more pronounced at larger patch sizes, as shown in Fig. 11, further emphasizing their ability to leverage extended temporal context. Collectively, these qualitative results reinforce the quantitative findings from the error heatmaps and highlight the limitations of LSTM architectures in long-horizon, high-volatility financial forecasting scenarios.

7.10. Electricity time series forecasting

A dedicated subset of the original energy dataset was extracted over the available temporal horizon. After preprocessing and cleaning, the resulting dataset focuses on electricity generation from actively operating sources, including *biomass, fossil brown coal/lignite, fossil gas, fossil hard coal, fossil oil*, and *geothermal energy*. Each generation category exhibits distinct temporal patterns influenced by operational dispatch strategies, demand cycles, and external environmental factors. The data are recorded at an hourly resolution, and a contiguous segment spanning indices 4000 to 7500 was selected to construct a temporally coherent window that preserves both daily and weekly periodic dynamics.

This preprocessing step yields a multi-channel input tensor of shape (3500, 6), enabling the `DeepKoopFormer` model to simultaneously learn from multiple correlated electricity generation streams within a unified forecasting framework. The dataset is subsequently divided into training and testing subsets using an 80 %/20 % split.

The `DeepKoopFormer`-based forecasting experiments were designed to systematically evaluate model performance across a broad range of temporal scales and architectural configurations. In particular, the patch length, which controls the degree of temporal aggregation or smoothing depending on the underlying `DeepKoopFormer` variant, was varied over the set $p = \{70, 80, 90, 100, 110, 120, 130\}$. For `PatchTST`-style embeddings, this parameter determines the temporal granularity of input tokens, whereas for Autoformer-style architectures it corresponds to the effective size of the moving-average decomposition window. The prediction horizon was swept over $H = \{2, 4, 6, 8, 10, 12, 14, 16\}$ to assess both short-term and long-range forecasting performance. All `DeepKoopFormer` variants employed a latent embedding dimension of $d_{\text{model}} = 96$, which governs the representational capacity used for temporal encoding, Koopman operator learning, and downstream prediction. Together, these settings provide a balanced trade-off between model expressiveness and computational efficiency across the evaluated configurations.

Fig. 12 summarizes the training performance of the different `DeepKoopFormer`-based forecasting architectures over the full grid of patch lengths and forecast horizons.

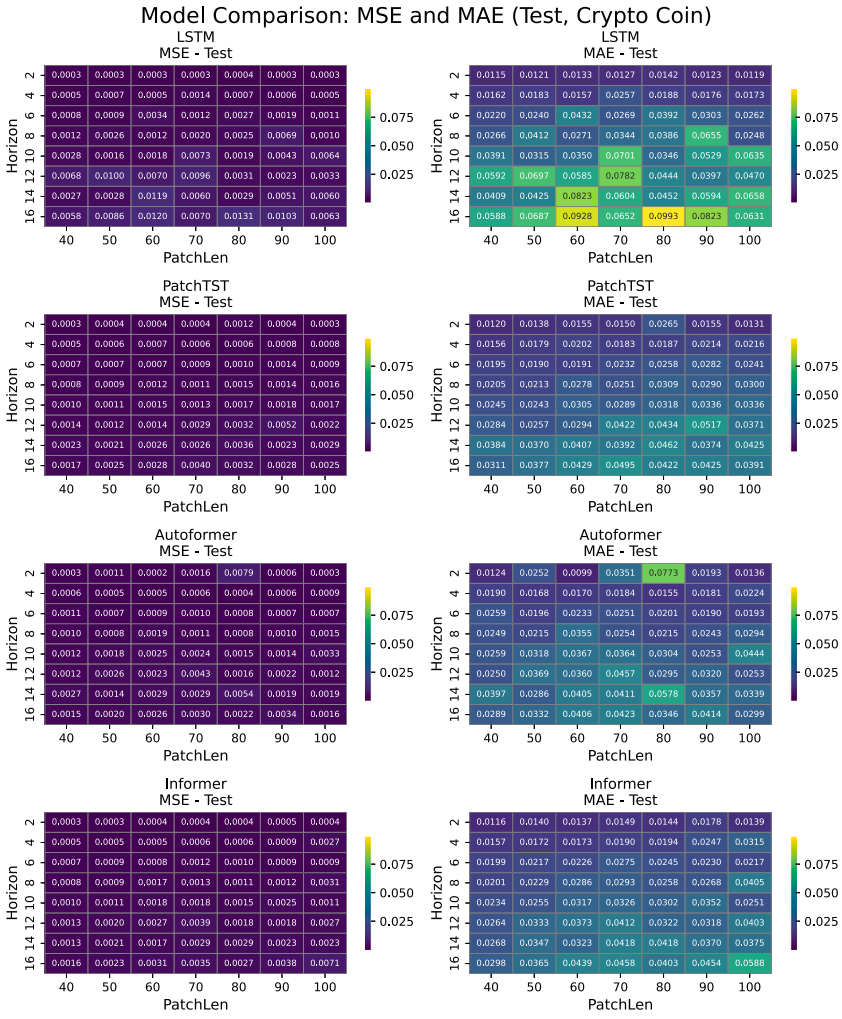

Fig. 9. `DeepKoopFormer` models compared with the LSTM baseline across a grid of hyperparameter settings on the test set for financial time series forecasting in Scenario Section 7.9.

The training results indicate that `PatchTST` consistently achieves the lowest MSE and MAE across nearly all experimental settings, highlighting its strong capability to capture temporal dependencies through patch-based representations. The `Informer` model demonstrates competitive performance with moderate error values, particularly for shorter prediction horizons, whereas `Autoformer` generally exhibits higher training errors, especially at extended horizons. These observations suggest that `DeepKoopFormer-PatchTST` provides the most stable and accurate convergence behavior during training, followed by `Informer`, while `Autoformer` appears more sensitive to the choice of forecasting window and smoothing scale. Furthermore, `PatchTST` and `Informer` outperform the LSTM baseline across most configurations. In contrast, `Autoformer` does not consistently surpass LSTM performance, particularly for shorter horizons (e.g., $H = 2, 4$).

As illustrated in Fig. 13, the generalization trends observed on the test set further emphasize the superiority of `PatchTST`. This variant consistently attains the lowest MSE and MAE across a wide range of patch lengths and forecast horizons, with error values remaining stable even for long-range predictions. Such behavior indicates robust temporal representation learning and effective enforcement of Koopman-stable dynamics. The `Informer` model exhibits moderate generalization performance, particularly at intermediate horizons, although its accuracy degrades slightly for extended forecasting windows. In contrast, `Autoformer` records the largest test errors overall, especially at larger patch sizes, which may be attributed to excessive smoothing during trend decomposition.

Overall, `PatchTST` achieves the most favorable balance between training accuracy and test generalization, confirming its suitability for real-world electricity generation forecasting tasks. The LSTM baseline demonstrates the weakest generalization performance among the evaluated models, except for very short-term predictions (e.g., $H = 2$), where its results remain comparatively competitive (Table 10).

7.11. Ablation analysis and extended baseline evaluation for time series forecasting

To isolate the effect of the latent dynamical propagator in `DeepKoopFormer`, we perform a controlled ablation study in which all models share the same temporal encoder and differ only in how latent dynamics are parameterized and propagated. In particular, we adopt `PatchTST` Minimal as a common backbone and systematically replace the latent evolution module with increasingly expressive alternatives, ranging from purely linear baselines to Koopman-structured operators with Lyapunov regularization. The following six model variants are considered:

- `DLinear` [33]: a lightweight, channel-wise linear baseline that operates directly on the input window without attention, latent states, or temporal propagation.

Fig. 10. Comparison of `DeepKoopFormer` on the test dataset for Cryptocurrency forecasting with $p = 40$, $H = 16$ and $d_{\text{model}} = 48$, the LSTM hidden size is 48.

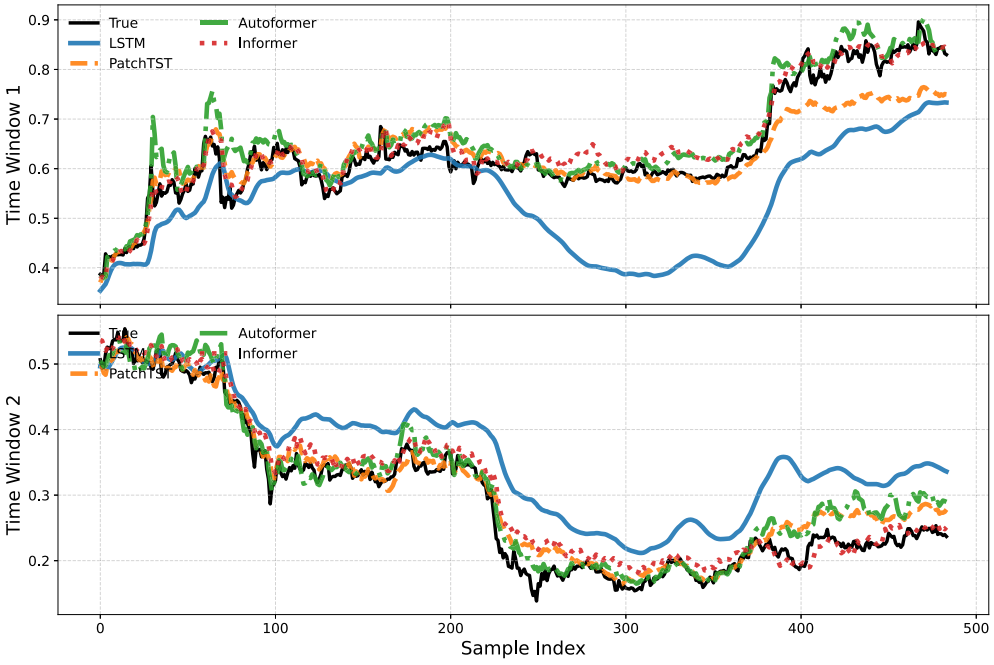

Fig. 11. Comparison of `DeepKoopFormer` on the test dataset for Cryptocurrency forecasting with $p = 80$, $H = 16$ and $d_{\text{model}} = 48$, the LSTM hidden size is 48.

- `LSTM`: a two-layer recurrent network whose hidden dimension is matched to the Transformer width to ensure comparable representational capacity.
- `PatchTST` (*Baseline Transformer*): a pure Transformer forecaster in which the `PatchTST` encoder is followed by a linear projection head that maps the final token embeddings directly to the forecast horizon, without any explicit latent dynamical model.
- `PatchTST–LinearLatent` (*Unconstrained latent model*): a latent-state extension of `PatchTST` in which the encoder output is mapped to a latent vector $\mathbf{z}_t$ that evolves according to an unconstrained linear operator $\mathbf{z}_{t+1} = W\mathbf{z}_t$, followed by a linear decoder. No spectral or stability constraints are imposed on W.
- `KoopPatchTST_NoLyap` (*Koopman ODO without Lyapunov*): a Koopman-augmented `PatchTST` model in which the latent transition is governed by a strictly stable operator $\mathbf{K}$ parameterized via operator decomposition (ODO) with spectral radius constrained as $\rho(\mathbf{K}) < 0.99$, but without any Lyapunov regularization.
- `KoopPatchTST_Full` (*Full `DeepKoopFormer` variant*): the complete Koopman-enhanced model in which the same ODO-parameterized

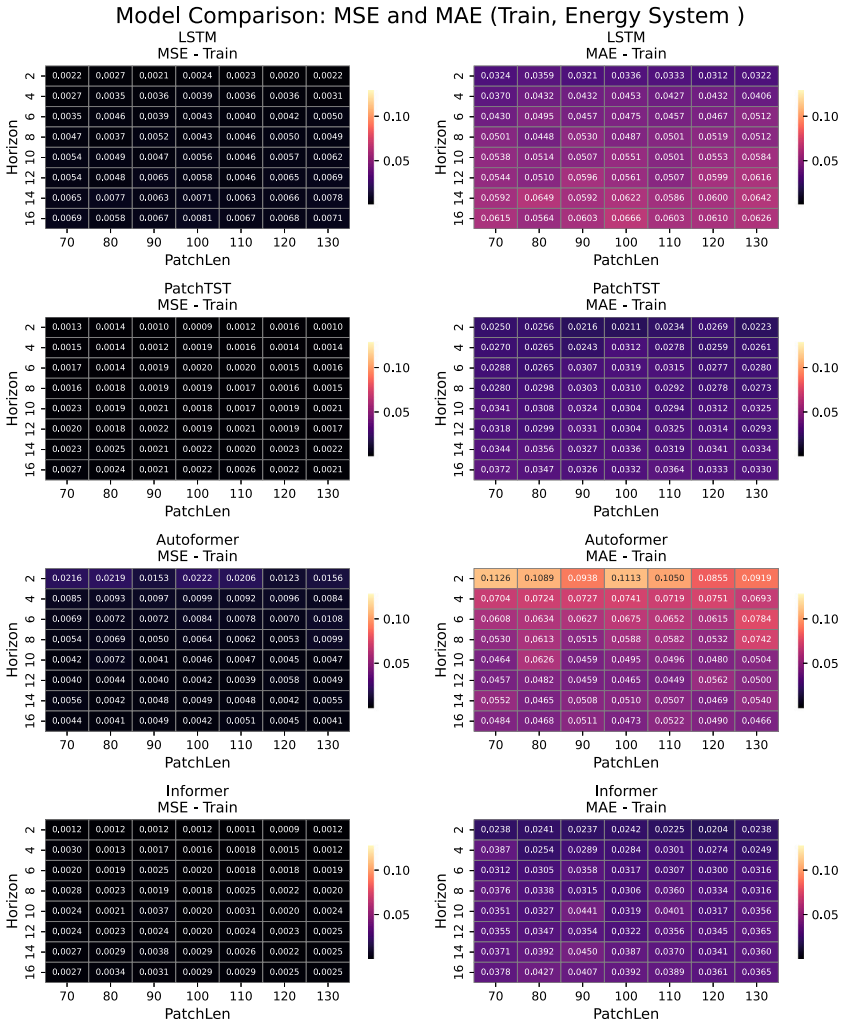

Fig. 12. `DeepKoopFormer` models compared with the LSTM baseline across a grid of hyperparameter configurations on the training set for electricity generation forecasting in Section 7.10.

operator $\mathbf{K}$ with $\rho(\mathbf{K}) < 0.99$ is further regularized by a Lyapunov-inspired stability penalty with weight $\lambda_{\text{Lyap}} = 0.1$.

All `PatchTST`-based models share an identical encoder architecture consisting of a 1D patch embedding layer followed by a Transformer encoder with 3 layers, 4 attention heads, model dimension $d_{\text{model}} = d_{\text{latent}} = 96$, and feedforward width $d_{\text{ff}} = 96$. The LSTM baseline employs two stacked recurrent layers with hidden dimension 96, yielding a parameter count comparable to that of the Transformer-based models. The `DLinear` model is intentionally much smaller and serves as a strong low-capacity linear reference.

To guarantee strict comparability across all methods, we train every model using the same chronological train–test split, adopting an 80%/20% partition along the time axis. A fixed subset of five input channels is used throughout, and all features are scaled via `MinMax` normalization computed on the training set and subsequently applied to the test data. Sliding-window datasets are constructed with context length P set equal to the patch length and forecast horizon H corresponding to the number of predicted time steps. We perform a systematic grid sweep over patch lengths $p \in \{70, 80, 90, 100, 110, 120, 130\}$ and forecast horizons $H \in \{2, 4, 6, 8, 10, 12, 14, 16\}$.

All models are optimized using the Adam optimizer with a learning rate of 3×10^{-4} and trained for a fixed budget of 4000 epochs without early stopping, in order to eliminate confounding effects arising from different convergence criteria. The primary supervision signal is the mean squared error (MSE) between predicted and ground-truth trajectories on the normalized scale. For Koopman-based models, the total objective is given by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{forecast}} + \lambda_{\text{Lyap}}\, \mathcal{L}_{\text{Lyap}},$$

where $\mathcal{L}_{\text{Lyap}}$ denotes the Lyapunov-inspired stability penalty defined in (48). We set $\lambda_{\text{Lyap}} = 0.1$ for `KoopPatchTST_Full` and $\lambda_{\text{Lyap}} = 0$ for all remaining variants. For each model and every (p, H) configuration, both the mean squared error (MSE) and the mean absolute error (MAE) are reported on the training and test sets.

7.11.1. Results on electricity generation time series

The training heatmaps for the electricity-generation benchmark (Fig. 14) exhibit a highly structured and consistent pattern across the entire grid of patch lengths and forecast horizons. Among all evaluated methods, `KoopPatchTST_Full` achieves the lowest training MSE for the vast majority of configurations, with `KoopPatchTST_NoLyap` occasionally attaining comparable minima. Their errors remain tightly concentrated around 10^{-3} across the grid, forming a distinct low-error plateau that is both dark and spatially uniform. This behavior indicates that the ODO-parameterized Koopman operator, and further its

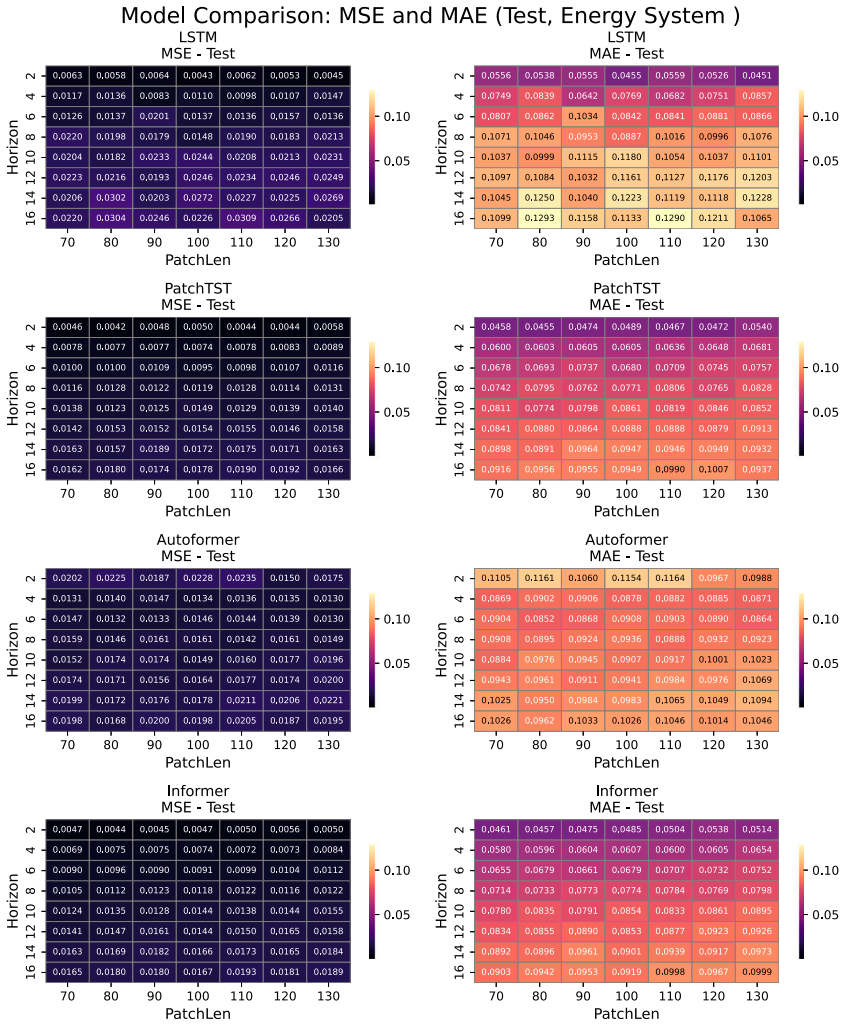

Fig. 13. `DeepKoopFormer` models compared with the LSTM baseline across a grid of hyperparameter configurations on the test set for electricity generation forecasting in Section 7.10.

Table 10
Train-set average MSE comparison across Koopman-augmented Transformer baselines in Section 7.8.3, Section 7.8.2, Section 7.9, Section 7.10.

Dataset	`DeepKoop-Autoformer`	`DeepKoop-Informer`	`DeepKoop-PatchTST`	`LSTM`
Cryptocurrency	0.00088	0.00065	0.00075	0.00096
Electricity generation	0.00735	0.00187	0.00156	0.00471
Pressure surface	0.00232	0.00131	0.00121	0.00216
Wind speed	0.00676	0.00391	0.00362	0.00591

Lyapunov regularization in the full variant, yields a latent dynamical model that fits the complex energy-system trajectories with high fidelity while preserving structured and stable evolution.

The remaining nonlinear baselines—namely the LSTM and the Transformer-based variants (`PatchTST` and `PatchTST– Linear–Latent`)—also achieve low training errors, but these are consistently higher and more spatially variable than those of the Koopman-augmented models. Their heatmaps show mild sensitivity to patch length and forecast horizon, suggesting that without an explicit spectral structure, the learned latent dynamics are less uniformly optimal across scales.

By contrast, the linear `DLinear` baseline exhibits the highest training MSE throughout the entire patch–horizon grid, with values in the range 10^{-2}–10^{-1}, i.e., more than an order of magnitude larger than those of the Koopman-based models. Its nearly uniform high-error plateau indicates that a purely decomposition–linear representation is fundamentally insufficient to capture the multiscale and nonlinear interactions that characterise modern energy-system time series. Overall, the training results already suggest that structured nonlinear latent propagators are essential for this task, with `KoopPatchTST_Full` providing the most accurate and stable fits.

The corresponding test-set heatmaps in Fig. 15 reveal an even clearer separation between linear and nonlinear models. Across all patch lengths and forecast horizons, `DLinear` yields the largest errors in both MSE and MAE, forming a uniformly high-error region over the entire grid. This confirms that the electricity-generation dynamics exhibit strong nonlinear and cross-scale temporal dependencies that cannot be captured by a simple decomposition–linear model. The recurrent `LSTM` baseline improves upon `DLinear` but remains substantially inferior to all Transformer-based and Koopman-augmented architectures.

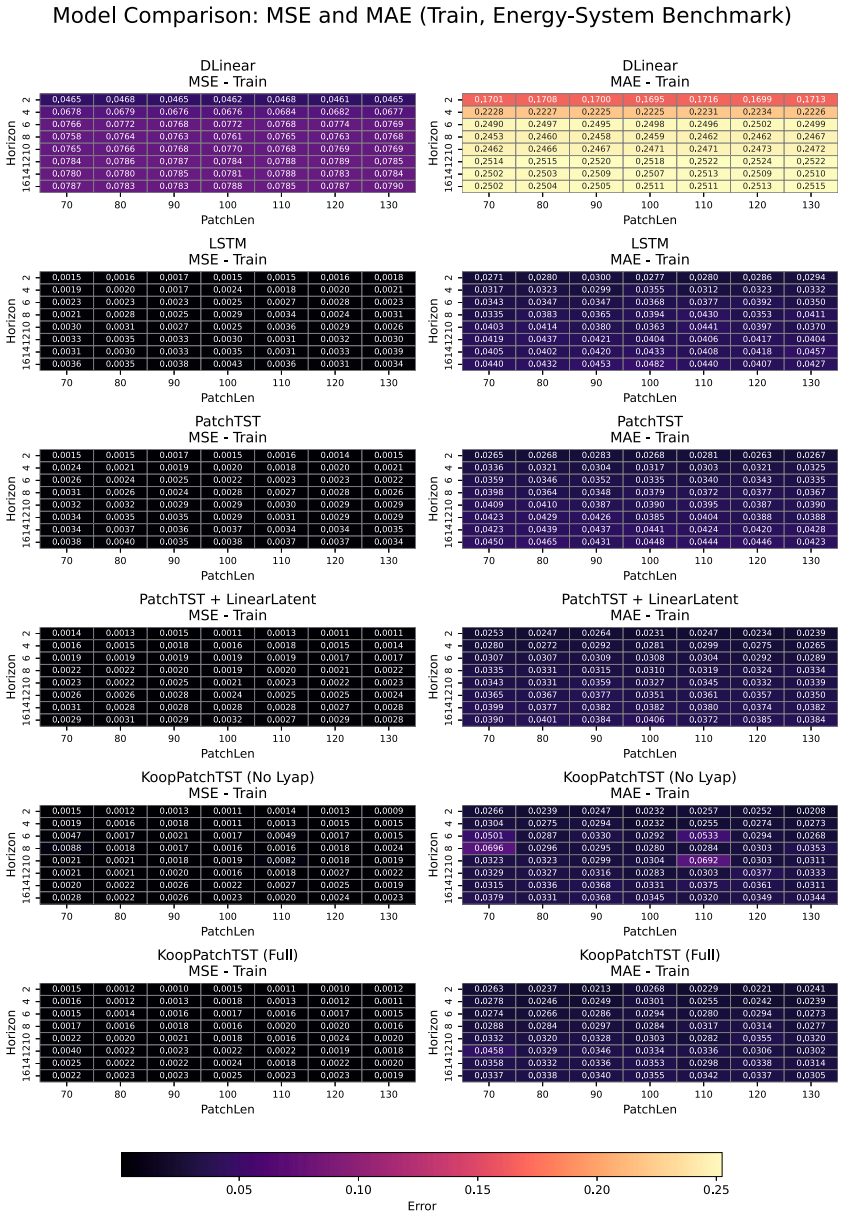

Fig. 14. Grid search over patch lengths and forecast horizons on the training set for electricity generation forecasting, comparing `DeepKoopFormer`—constructed by augmenting compact Transformer backbones with Koopman latent dynamics—with all baseline models (Section 7.11.1).

Among the Transformer baselines, `PatchTST` achieves the lowest average test errors and displays a relatively smooth error surface, particularly for short and medium forecast horizons. The `PatchTST–LinearLatent` variant benefits from its explicit latent propagation and attains slightly lower errors for some configurations; however, its heatmaps are more irregular, indicating a mild tendency toward overfitting for longer horizons and larger patches.

The Koopman-based models provide the most coherent and robust generalization across the entire grid. In particular, `KoopPatchTST_Full` combines near-optimal predictive accuracy with the smallest variance of test error over patch lengths and horizons, yielding the smoothest and most horizon-robust error surface among all nonlinear models. While `KoopPatchTST_NoLyap` occasionally matches or slightly exceeds the full model for specific patch lengths, its error landscape is visibly less regular, confirming the stabilising role of Lyapunov regularization. Overall, these results demonstrate that spectrally constrained and Lyapunov-regularized Koopman dynamics provide superior and more reliable generalization on the energy-system forecasting task.

Fig. 16 illustrates representative forecast trajectories for six electricity generation sources on the test set using patch length $p = 70$, forecast horizon $H = 4$, and embedding dimension $d_{\text{model}} = 96$. The linear `DLinear` baseline (blue) captures only slowly varying components and fails to reproduce the high-frequency variability observed in the ground truth, resulting in systematic amplitude and phase errors across all energy sources. The LSTM baseline better tracks global trends but exhibits noticeable phase delays and excessive smoothing, especially for highly dynamic series such as *Fossil Gas* and *Geothermal Energy*.

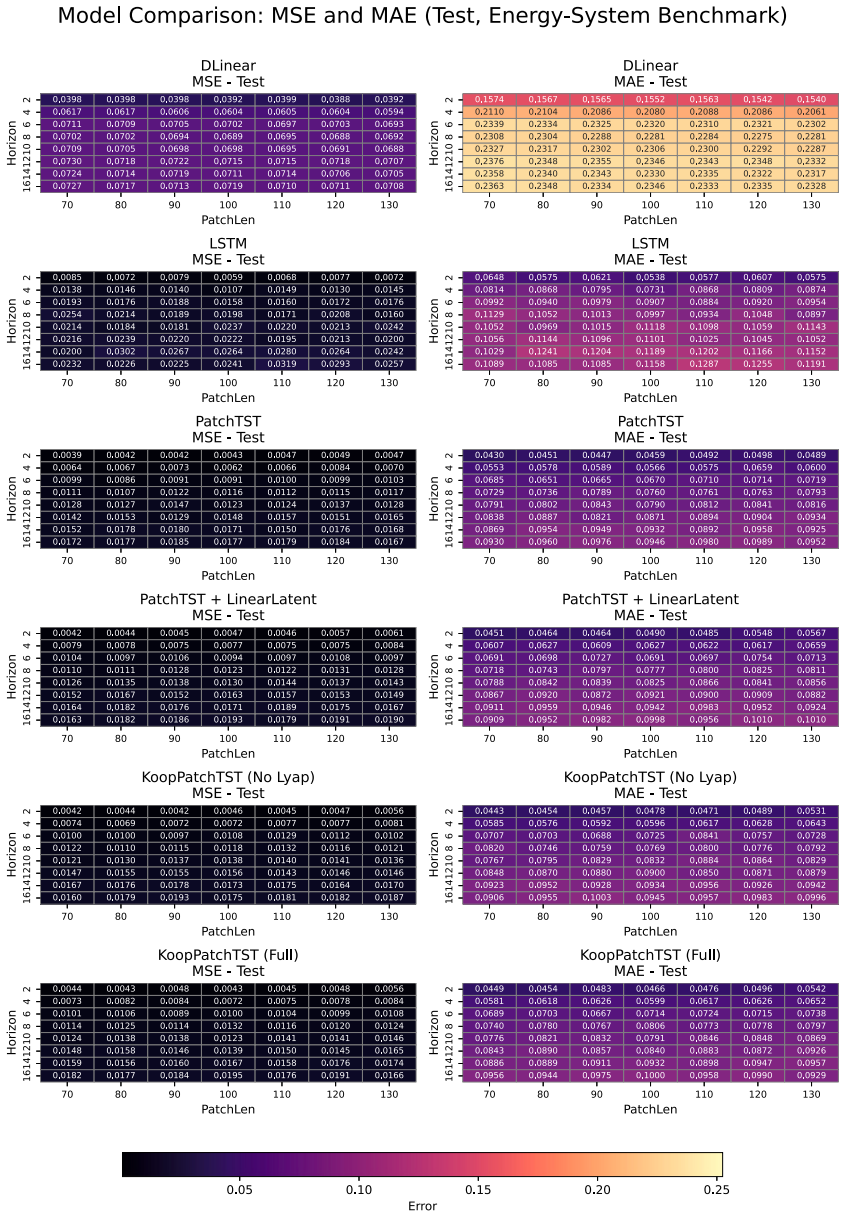

Fig. 15. Grid search over patch lengths and forecast horizons on the test set for electricity generation forecasting, comparing `DeepKoopFormer`—constructed by augmenting compact Transformer backbones with Koopman latent dynamics—with all baseline models (Scenario Section 7.11.1).

In contrast, the Transformer-based and Koopman-augmented models (`PatchTST`, `PatchTST–LinearLatent`, `KoopPatchTST_NoLyap`, and `KoopPatchTST_Full`) closely follow the reference trajectories and faithfully reconstruct both sharp ramps and fast oscillations. In particular, the Koopman-based variants produce trajectories that are nearly indistinguishable from the ground truth for sources such as *Fossil Hard Coal*, *Fossil Oil*, and *Geothermal Energy*, demonstrating that the introduction of a Koopman latent propagator does not compromise short-term accuracy while providing additional stability and structure. Overall, this visual comparison further confirms the superiority of Koopman-augmented `PatchTST` models over purely linear and recurrent baselines for modeling heterogeneous electricity-generation dynamics.

Summary of experimental findings. Across all benchmarks, the experimental results reveal consistent performance trends among the proposed architectures. `PatchTST` Standard achieves the most stable accuracy across clean and noisy regimes, providing the best overall trade-off between robustness and computational efficiency. `Autoformer` variants perform particularly well on smooth and trend-dominated signals due to their explicit decomposition structure, while `Informer` variants exhibit higher sensitivity to noise and long forecasting horizons despite their scalability advantages. These findings confirm that carefully designed compact Transformer architectures can achieve strong generalization performance while maintaining favorable efficiency and robustness characteristics.

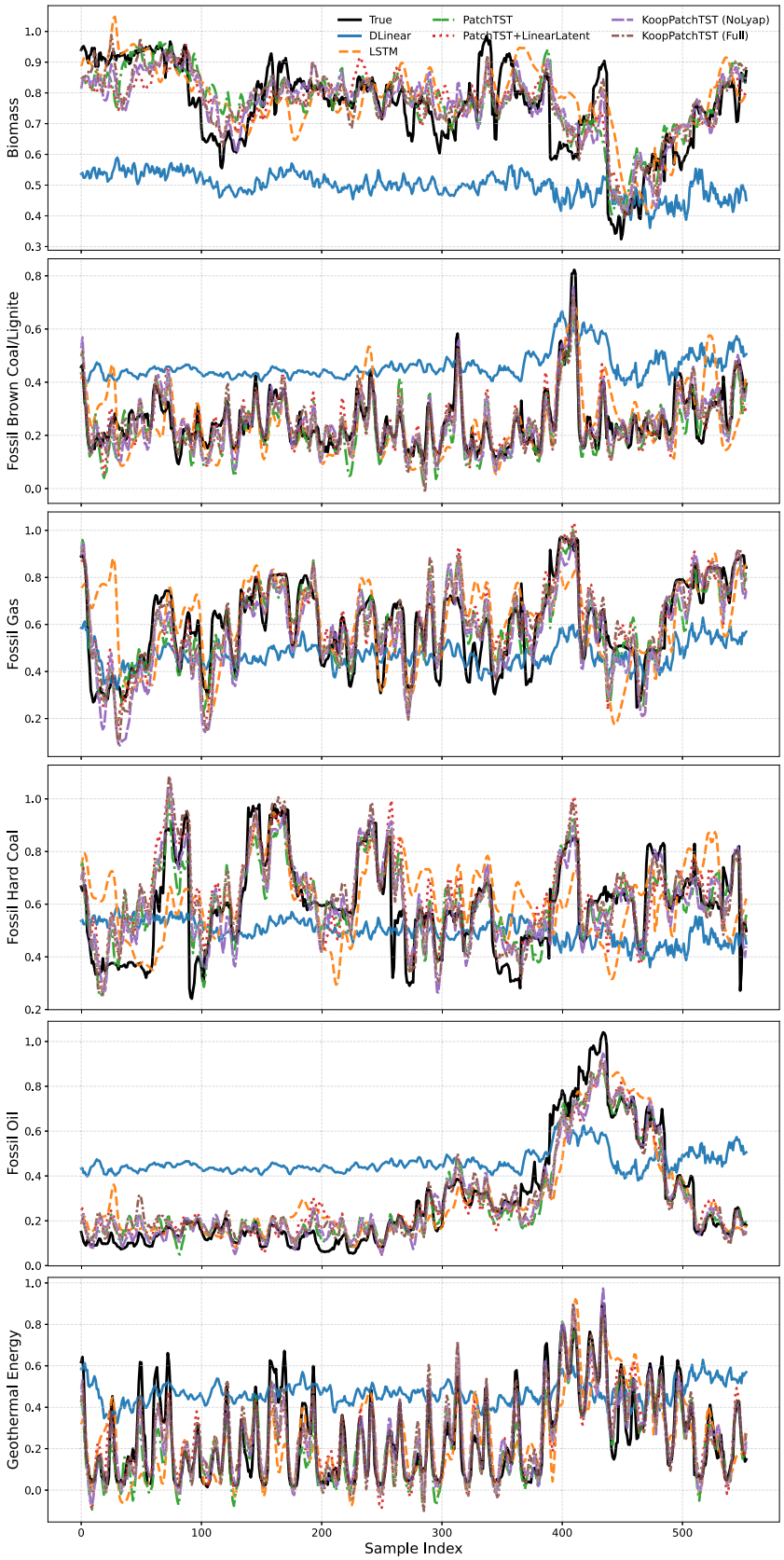

Fig. 16. Forecast comparison on the test set for electricity generation using `DeepKoopFormer`—built by augmenting compact Transformer backbones with Koopman latent dynamics—with $p = 130$, $H = 16$, and $d_{\text{model}} = 96$; the LSTM baseline also uses a hidden size of 96.

8. Conclusion and future direction

This study presents a unified empirical investigation of Transformer-based model families for univariate time series forecasting under both clean and noisy signal conditions. We bench-marked three prominent architectures—`Autoformer`, `Informer`, and `PatchTST` —each implemented in three variants: *Minimal*, *Standard*, and *Full*, representing increasing levels of architectural complexity.

Across 10 synthetic signals and 5×5 configurations of patch lengths and forecast horizons, we conduct 750 experiments per noise regime, totaling 1500 evaluations for each model family. This dual-setting analysis (clean + noisy) enables robust insights into the forecasting capabilities, stability, and noise resilience of each architecture.

`PatchTST` Standard emerged as the most consistent and robust variant across both clean and noisy environments. It delivered low RMSE and MAE even under signal perturbations, demonstrating resilience in long-horizon, non-stationary, and noisy conditions while remaining computationally efficient. Patch-based attention with learned temporal embeddings proved highly effective.

`Autoformer` Minimal and `Autoformer` Standard maintained competitive accuracy, particularly in clean, short-to-medium horizon tasks. Their performance was bolstered by trend-seasonal decomposition, which imparts stability and interpretability. Notably, `Autoformer` continued to outperform in noisy settings—confirming its structural advantage in capturing underlying signal trends even amidst distortion.

`Informer` Standard and `Informer` Full exhibited clear performance degradation under noise. While sparse attention offers scalability, it led to unstable forecasts, especially at longer horizons and with larger patch sizes. `Informer` struggled most with noisy inputs, indicating its limited generalization in structured low-noise or high-variance time series.

For all three model families, performance was sensitive to both forecast horizon and patch length. Optimal results were generally found with patch lengths in the range of 12–16 and forecast horizons of 2–6. Increasing patch size beyond this range did not reliably improve accuracy and may induce overfitting, particularly in clean settings. In conclusion, encoder-only models such as `PatchTST` Standard offered the best trade-off between forecasting accuracy, robustness, and scalability across noise regimes. Trend decomposition of `Autoformer` remained beneficial under distortion, while `Informer` was less suited to structured and noisy time series.

The extended ablation study across proposed variants reveals that much of the forecasting power of modern Transformer architectures arises from a small set of core inductive biases—temporal patching, decomposition, and sparse attention—rather than from large hidden dimensions or deep stacks of layers. Our large-scale grid search over patch lengths, forecast horizons, and model capacities demonstrates that compact Transformer backbones already achieve strong accuracy and robustness when these inductive biases are correctly instantiated. Importantly, the new real-world experiments on CMIP6 climate data and electricity generation confirm that these lightweight architectures do not merely succeed on synthetic benchmarks, but also scale effectively to high-dimensional, non-stationary systems. This validates the central design principle of this work: compact, modular Transformer variants provide a principled and efficient foundation for building larger, more structured forecasting systems without sacrificing predictive performance or interpretability.

Although `Informer` was originally developed for ultra–long sequence forecasting, its `ProbSparse` attention mechanism relies on reliable margin separation between dominant and non-dominant keys, which in turn requires sufficiently large embedding dimensions and clean attention scores. In the compact regime considered in this work ($d_{\text{model}} = 8$ with short–to–medium context windows), this separation is weakened, especially under additive and multiplicative noise, leading to unstable top-u query selection and degraded attention quality. Consequently, Informer exhibits higher sensitivity to noise and long-horizon error growth compared to `PatchTST` and `Autoformer` in our controlled experiments. Importantly, this behavior is not an artifact of tuning but reflects a fundamental bias–variance trade-off of sparse attention in low-dimensional latent spaces. To demonstrate that this limitation is architectural rather than intrinsic, we further embed `Informer` into a larger framework namely `DeepKoopFormer` [32] framework, where sparse attention is used only as a latent encoder and temporal propagation is governed by a spectrally controlled Koopman operator. In this hybrid setting, `Informer` regains stability and achieves competitive long-horizon accuracy on real-world climate and energy datasets, confirming that `ProbSparse` attention remains effective when deployed within a properly structured operator-theoretic forecasting architecture.

Future work. As a promising direction for future research, we aim to extend the current Transformer-based forecasting models by integrating them with Koopman operator theory, giving rise to a hybrid architecture we term `DeepKoopFormer` as it is discussed briefly in [32,34]. This framework will combine the expressive power of Transformer encoders with the stability and interpretability of linear latent dynamics induced by a Koopman operator [35–37]. By introducing constraints on spectral stability and Lyapunov-inspired regularization in the latent space [38–40], `DeepKoopFormer` will address critical limitations of standard Transformer models, such as latent instability, error accumulation in long-horizon forecasts, and lack of interpretability. This integration is expected to enable more robust and physically consistent modeling of nonlinear and oscillatory systems, aligning data-driven forecasts with the underlying dynamical structure of the process.

Finally as part of future work, we plan to integrate the compact Transformer framework developed in this study with *Learnable-DeepKoopFormer* architecture [41], which introduces trainable spectral control, stability-aware propagation, and low-rank Koopman parameterizations on top of Transformer encoders. This extension enables adaptive latent dynamics while preserving the interpretability and robustness benefits of operator-theoretic modeling. Combining the modular Transformer variants analyzed here with learnable Koopman operators will allow a unified evaluation of fixed versus adaptive latent dynamics and will further clarify the trade-offs between expressiveness, stability, and long-horizon forecasting performance in Transformer-based time-series models.

CRediT authorship contribution statement

Ali Forootani: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Mohammad Khosravi:** Visualization, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Algorithms pseudo code

This section provides the pseudo code for the discussed `Informer`, `Autoformer`, and `PatchTST` architectures in this article.

Algorithm 1 `PatchTST` full forecasting procedure.

Input: Patch sequence $\mathbf{x} \in \mathbb{R}^{B\times P\times 1}$; forecast horizon H; model dimension d_{model}
Output: Forecast $\hat{\mathbf{y}} \in \mathbb{R}^{B\times H}$
Step 1: Encoder Input Projection and Positional Encoding

$$\mathbf{Z}^{\text{enc}} = \mathbf{x}\cdot\mathbf{W}_e^{(e)} + \text{PE}^{(e)} \in \mathbb{R}^{B\times P\times d_{\text{model}}}$$

Step 2: Transformer Encoder Layers
for each encoder layer $\ell = 1,\dots,L$ **do**
  Compute attention heads:

$$Q = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}_\ell^Q,\quad K = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}_\ell^K,\quad V = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}_\ell^V$$

$$\text{head}_i = \text{Softmax}\left(\frac{Q_iK_i^\top}{\sqrt{d_{\text{model}}/h}}\right)V_i$$

$$\text{MHAttn} = \text{Concat}(\text{head}_1,\dots,\text{head}_h)\cdot\mathbf{W}_\ell^O$$

  Apply residual connection and layer normalization:

$$\mathbf{Z}^{\text{enc}} \leftarrow \text{LayerNorm}(\mathbf{Z}^{\text{enc}} + \text{MHAttn})$$

  Apply feedforward and second residual block:

$$\mathbf{Z}^{\text{enc}} \leftarrow \text{LayerNorm}(\mathbf{Z}^{\text{enc}} + \text{FFN}(\mathbf{Z}^{\text{enc}}))$$

end for
Encoder output: $\mathbf{H}^{\text{enc}} = \mathbf{Z}^{\text{enc}}$
Step 3: Decoder Input Construction and Projection
Repeat the final time step H times:

$$\mathbf{x}^{\text{rep}} = \text{Repeat}(\mathbf{x}_{:,-1,:}, H) \in \mathbb{R}^{B\times H\times 1}$$

Project and encode with positional embedding:

$$\mathbf{Z}^{\text{dec}} = \mathbf{x}^{\text{rep}}\cdot\mathbf{W}_e^{(d)} + \text{PE}^{(d)} \in \mathbb{R}^{B\times H\times d_{\text{model}}}$$

Step 4: Transformer Decoder Layer
Compute self-attention on decoder input:

$$Q = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{Q_d},\quad K = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{K_d},\quad V = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{V_d}$$

$$\text{SelfAttn} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

Compute encoder-decoder cross-attention:

$$Q = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{Q_c},\quad K = \mathbf{H}^{\text{enc}}\cdot\mathbf{W}^{K_c},\quad V = \mathbf{H}^{\text{enc}}\cdot\mathbf{W}^{V_c}$$

$$\text{CrossAttn} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

Apply decoder residual blocks:

$$\mathbf{Z}^{\text{dec}} \leftarrow \text{LayerNorm}(\mathbf{Z}^{\text{dec}} + \text{SelfAttn})$$
$$\mathbf{Z}^{\text{dec}} \leftarrow \text{LayerNorm}(\mathbf{Z}^{\text{dec}} + \text{CrossAttn})$$
$$\mathbf{Z}^{\text{dec}} \leftarrow \text{LayerNorm}(\mathbf{Z}^{\text{dec}} + \text{FFN}(\mathbf{Z}^{\text{dec}}))$$

Step 5: Output Projection

$$\hat{\mathbf{y}} = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}_o \in \mathbb{R}^{B\times H\times 1}$$

return $\hat{\mathbf{y}}$ reshaped to $\mathbb{R}^{B\times H}$

Algorithm 2 Full `Informer` forecasting procedure with probsparse attention.

Input: Input sequence $\mathbf{x} \in \mathbb{R}^{B\times P\times 1}$, forecast horizon H, model dimension d_{model}
Output: Forecast sequence $\hat{\mathbf{y}} \in \mathbb{R}^{B\times H}$
Step 1: Encoder Projection and Positional Encoding

$$\mathbf{Z}^{\text{enc}} = \mathbf{x}\cdot\mathbf{W}_e^{(e)} + \text{PE}^{(e)} \in \mathbb{R}^{B\times P\times d_{\text{model}}}$$

Step 2: ProbSparse Transformer Encoder
for each encoder layer **do**
  Compute attention triplets:

$$Q = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}^Q,\quad K = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}^K,\quad V = \mathbf{Z}^{\text{enc}}\cdot\mathbf{W}^V$$

  Compute sparsity score for each q_i:

$$M(q_i) = \max_j\left(\frac{q_ik_j^\top}{\sqrt{d_{\text{model}}}}\right) - \frac{1}{P}\sum_{j=1}^{P}\frac{q_ik_j^\top}{\sqrt{d_{\text{model}}}}$$

  Select top-u queries ($u = c\log P$) based on $M(q_i)$
  For top-u queries, compute full attention:

$$\text{Attn}(q_i,K,V) = \text{Softmax}\left(\frac{q_iK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

  For other queries, use top-u keys (sparse approximation)
  Apply feedforward layers and residual connections
end for
Let encoder output: $\mathbf{H}^{\text{enc}} \in \mathbb{R}^{B\times P\times d_{\text{model}}}$
Step 3: Decoder Initialization
Repeat last input value H times:

$$\mathbf{x}^{\text{rep}} = \text{Repeat}(\mathbf{x}_{:,-1,:}, H) \in \mathbb{R}^{B\times H\times 1}$$

Project and add positional encoding:

$$\mathbf{Z}^{\text{dec}} = \mathbf{x}^{\text{rep}}\cdot\mathbf{W}_e^{(d)} + \text{PE}^{(d)} \in \mathbb{R}^{B\times H\times d_{\text{model}}}$$

Step 4: ProbSparse Transformer Decoder
for each decoder layer **do**
  Compute self-attention using ProbSparse:

$$Q = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{Q_d},\quad K = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{K_d},\quad V = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{V_d}$$

$$\text{SelfAttn} = \text{SparseAttention}(Q,K,V)$$

  Compute cross-attention with encoder output:

$$Q = \mathbf{Z}^{\text{dec}}\cdot\mathbf{W}^{Q_c},\quad K = \mathbf{H}^{\text{enc}}\cdot\mathbf{W}^{K_c},\quad V = \mathbf{H}^{\text{enc}}\cdot\mathbf{W}^{V_c}$$

$$\text{CrossAttn} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

  Apply residual and feedforward layers
end for
Let decoder output: $\mathbf{H}^{\text{dec}} \in \mathbb{R}^{B\times H\times d_{\text{model}}}$
Step 5: Output Projection

$$\hat{\mathbf{y}} = \mathbf{H}^{\text{dec}}\cdot\mathbf{W}_o \in \mathbb{R}^{B\times H\times 1}$$

return $\hat{\mathbf{y}}$ (squeezed to shape $\mathbb{R}^{B\times H}$)

Algorithm 3 Full `Autoformer` forecasting procedure.

Input: Sequence $\mathbf{x} \in \mathbb{R}^{B\times P\times 1}$, forecast horizon H, model dimension d_{model}
Output: Forecast $\hat{\mathbf{y}} \in \mathbb{R}^{B\times H}$
Step 1: Series Decomposition
Apply moving average of kernel size k:

$$\mathbf{x}^{\text{trend}} = \text{MA}_k(\mathbf{x}), \quad \mathbf{x}^{\text{seasonal}} = \mathbf{x} - \mathbf{x}^{\text{trend}}$$

Step 2: Encoder Path
Project seasonal input:

$$\mathbf{Z}^{\text{enc}} = \mathbf{x}^{\text{seasonal}} \cdot \mathbf{W}_e^{(e)} + \text{PE}^{(e)} \in \mathbb{R}^{B\times P\times d_{\text{model}}}$$

Apply L encoder layers (self-attention + FFN + residual + norm). For each layer:

$$Q = \mathbf{Z}^{\text{enc}} \cdot \mathbf{W}^Q, \quad K = \mathbf{Z}^{\text{enc}} \cdot \mathbf{W}^K, \quad V = \mathbf{Z}^{\text{enc}} \cdot \mathbf{W}^V$$

$$\text{SelfAttn}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

Let encoder output: $\mathbf{H}^{\text{enc}} \in \mathbb{R}^{B\times P\times d_{\text{model}}}$
Step 3: Decoder Initialization
Zero initialize seasonal decoder input:

$$\mathbf{z}_0^{\text{dec}} = \mathbf{0} \in \mathbb{R}^{B\times H\times 1}$$

Project decoder input:

$$\mathbf{Z}^{\text{dec}} = \mathbf{z}_0^{\text{dec}} \cdot \mathbf{W}_e^{(d)} + \text{PE}^{(d)} \in \mathbb{R}^{B\times H\times d_{\text{model}}}$$

Step 4: Transformer Decoder
for each decoder layer **do**
(a) Self-Attention:

$$Q = K = V = \mathbf{Z}^{\text{dec}} \cdot \mathbf{W}^{Q_d}, \mathbf{W}^{K_d}, \mathbf{W}^{V_d}$$

$$\text{SelfAttn} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

(b) Cross-Attention:

$$Q = \mathbf{Z}^{\text{dec}} \cdot \mathbf{W}^{Q_c}, \quad K = \mathbf{H}^{\text{enc}} \cdot \mathbf{W}^{K_c}, \quad V = \mathbf{H}^{\text{enc}} \cdot \mathbf{W}^{V_c}$$

$$\text{CrossAttn} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V$$

Apply FFN, residuals, and normalization
end for
Let decoder output: $\mathbf{H}^{\text{dec}} \in \mathbb{R}^{B\times H\times d_{\text{model}}}$
Step 5: Forecast Projection
Forecast seasonal output:

$$\hat{\mathbf{y}}^{\text{seasonal}} = \mathbf{H}^{\text{dec}} \cdot \mathbf{W}_o \in \mathbb{R}^{B\times H}$$

Forecast trend output:

$$\hat{\mathbf{y}}^{\text{trend}} = \mathbf{x}^{\text{trend}} \cdot \mathbf{W}_t \in \mathbb{R}^{B\times H}$$

Step 6: Final Output

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^{\text{seasonal}} + \hat{\mathbf{y}}^{\text{trend}} \in \mathbb{R}^{B\times H}$$

return $\hat{\mathbf{y}}$

Appendix B. Supplementary data

Supplementary data to this article can be found online at doi:10.1016/j.neucom.2026.133140.

Data availability

No data was used for the research described in the article.

References

[1] A. Forootani, D.E. Aliabadi, D. Thraen, Climate aware deep neural networks (cadnn) for wind power simulation, arXiv preprint arXiv:2412.12160, 2024.
[2] F.M. Alvarez, A. Troncoso, J.C. Riquelme, J.S.A. Ruiz, Energy time series forecasting based on pattern sequence similarity, IEEE Trans. Knowl. Data Eng. 23 (8) (2010) 1230–1243.
[3] A. Dingli, K.S. Fournier, Financial time series forecasting-a deep learning approach, Int. J. Mach. Learn. Comput. 7 (5) (2017) 118–122.
[4] S. Wang, Y. Lin, Y. Jia, J. Sun, Z. Yang, Unveiling the multi-dimensional spatio-temporal fusion transformer (MDSTFT): a revolutionary deep learning framework for enhanced multi-variate time series forecasting, IEEE Access 12 (2024) 115895–115904, https://doi.org/10.1109/ACCESS.2024.3444788
[5] K. Yan, C. Long, H. Wu, Z. Wen, Multi-resolution expansion of analysis in time-frequency domain for time series forecasting, IEEE Trans. Knowl. Data Eng. 36 (11) (2024) 6667–6680, https://doi.org/10.1109/TKDE.2024.3396785
[6] A. Tascikaraoglu, B.M. Sanandaji, G. Chicco, V. Cocina, F. Spertino, O. Erdinc, N.G. Paterakis, J.P. Catalão, Compressive spatio-temporal forecasting of meteorological quantities and photovoltaic power, IEEE Trans. Sustain. Energy 7 (3) (2016) 1295–1305, https://doi.org/10.1109/TSTE.2016.2544929
[7] K. Benidis, S.S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, et al., Deep learning for time series forecasting: tutorial and literature survey, ACM Comput. Surv. 55 (6) (2022) 1–36.
[8] P. Hewage, A. Behera, M. Trovati, E. Pereira, M. Ghahremani, F. Palmieri, Y. Liu, Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station, Soft Comput. 24 (2020) 16453–16482.
[9] L. Bryan, Z. Stefan, Time-series forecasting with deep learning: a survey, Phil, Trans. R. Soc. A (2021), https://doi.org/10.1098/rsta.2020.0209
[10] J.F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, A. Troncoso, Deep learning for time series forecasting: a survey, Big Data 9 (1) (2021) 3–21.
[11] P. Lara-Benítez, M. Carranza-García, J.C. Riquelme, An experimental review on deep learning architectures for time series forecasting, Int. J. Neural Syst. 31 (3) (2021) 2130001.
[12] K.S. Kalyan, A. Rajasekharan, S. Sangeetha, Ammus: A survey of transformer-based pretrained models in natural language processing, arXiv preprint arXiv:2108.05542, 2021.
[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017) https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
[14] S. Khan, M. Naseer, M. Hayat, S.W. Zamir, F.S. Khan, M. Shah, Transformers in vision: a survey, ACM Comput. Surv. 54 (10s) (2022) 41, https://doi.org/10.1145/3505244
[15] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N.E.Y. Soplin, R. Yamamoto, X. Wang, et al., A comparative study on transformer VS RNN in speech applications, in: IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), IEEE, 2019, pp. 449–456.
[16] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, L. Sun, Transformers in time series: A survey, arXiv preprint arXiv:2202.07125, 2022.
[17] Q. Fournier, G.M. Caron, D. Aloise, A practical survey on faster and lighter transformers, ACM Comput. Surv. 55 (14s) (2023) 1–40.
[18] C. Zhu, W. Ping, C. Xiao, M. Shoeybi, T. Goldstein, A. Anandkumar, B. Catanzaro, Long-short transformer: efficient transformers for language and vision, Adv. Neural Inf. Process. Syst. 34 (2021) 17723–17736.
[19] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: The Thirty-Fifth AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11106–11115.
[20] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: decomposition transformers with Auto-Correlation for long-term series forecasting, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J. Wortman Vaughan (Eds.), Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 22419–22430, https://proceedings.neurips.cc/paper_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.
[21] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, FEDformer: frequency enhanced decomposed transformer for long-term series forecasting, in: Proc. 39th International Conference on Machine Learning, 2022.
[22] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A.X. Liu, S. Dustdar, Pyraformer: low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International Conference on Learning Representations, 2022.
[23] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? arXiv preprint arXiv:2205.13504, 2022.
[24] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, arXiv preprint arXiv:2211.14730, 2022.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: transformers for image recognition at scale, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=YicbFdNTTy.
[26] A. Baevski, Y. Zhou, A. Mohamed, M. Auli, wav 2vec 2.0: A framework for self-supervised learning of speech representations, Adv. Neural Inf. Process. Syst. 33 (2020) 12449–12460.
[27] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Time series classification using multi-channels deep convolutional neural networks, in: International Conference on Web-Age Information Management, Springer, 2014, pp. 298–310.
[28] A. Forootani, D.E. Aliabadi, D.T. n, Climate aware deep neural networks (CADNN) for wind power simulation, Array 28 (2025) 100534, https://doi.org/10.1016/j.array.2025.100534, https://www.sciencedirect.com/science/article/pii/S2590005625001614.
[29] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, itransformer: Inverted transformers are effective for time series forecasting, arXiv preprint arXiv:2310.06625, 2023.
[30] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, Timesnet: Temporal 2d-variation modeling for general time series analysis, arXiv preprint arXiv:2210.02186, 2022.
[31] Y. Zhang, J. Yan, Crossformer: transformer utilizing cross-dimension dependency for multivariate time series forecasting, in: The Eleventh International Conference on Learning Representations, 2023.
[32] A. Forootani, M. Khosravi, M. Barati, Deepkoopformer: A koopman enhanced transformer based architecture for time series forecasting, arXiv preprint arXiv:2508.02616, 2025.
[33] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 11121–11128.
[34] A. Forootani, M. Khosravi, Synthetic time series forecasting with transformer architectures: Extensive simulation benchmarks, arXiv:2505.20048, 2025.
[35] B. Lusch, J.N. Kutz, S.L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature Communications 9 (1) (2018) 4950.
[36] E. Yeung, S. Kundu, N. Hodas, Learning deep neural network representations for Koopman operators of nonlinear dynamical systems, in: 2019 American Control Conference (ACC), 2019, pp. 4832–4839, https://doi.org/10.23919/ACC.2019.8815339
[37] N. Takeishi, Y. Kawahara, T. Yairi, Learning Koopman invariant subspaces for dynamic mode decomposition, in: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc. 2017, https://proceedings.neurips.cc/paper_files/paper/2017/file/3a835d3215755c435ef4fe9965a3f2a0-Paper.pdf.
[38] J. Drgoňa, A. Tuor, S. Vasisht, D. Vrabie, Dissipative deep neural dynamical systems, IEEE Open J. Control Syst. 1 (2022) 100–112, https://doi.org/10.1109/OJCSYS.2022.3186838
[39] E. Skomski, S. Vasisht, C. Wight, A. Tuor, J. Drgoňa, D. Vrabie, Constrained block nonlinear neural dynamical models, in: 2021 American Control Conference (ACC), 2021, pp. 3993–4000, https://doi.org/10.23919/ACC50511.2021.9482930
[40] J. Drgona, S. Mukherjee, J. Zhang, F. Liu, M. Halappanavar, On the stochastic stability of deep markov models, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc. 2021, pp. 24033–24047, https://proceedings.neurips.cc/paper_files/paper/2021/file/c9dd73f5cb96486f5e1e0680e841a550-Paper.pdf.
[41] A. Forootani, R. Iervolino, Learnable koopman-enhanced transformer-based time series forecasting with spectral control, arXiv:2602.02592, 2026.

Author biography

Ali Forootani received the M.Sc. degree in electrical engineering and automatic control systems from the Power and Water University of Technology, Tehran, Iran (Persia) in 2011, and the Ph.D. degree from the Department of Engineering, University of Sannio, Benevento, Italy, in 2019. From 2011 to 2015, he was involved in both research and industrial projects at the Niroo Research Institute and Ministry of Energy (Water and Sewage Management). From 2019 to 2025, he held postdoctoral research positions at the University of Sannio, the University of Salerno, Maynooth University, the Max Planck Institute for Dynamics of Complex Technical Systems, and Helmholtz Centre for Environmental Research. He is currently a Machine Learning and Data Scientist at the Max Planck Institute of Geoanthropology. His research interests include machine learning, Markov decision processes, approximate dynamic programming, reinforcement learning for optimal control, networked control systems, physics informed neural networks, nonlinear system identification and parameter estimation for PDEs and ODEs, time series forecasting, federated learning, and the application of artificial intelligence and large language models to energy system modeling. He is a Senior Member of the IEEE Control Systems Society.

Mohammad Khosravi received a B.Sc. in electrical engineering and a B.Sc. in mathematical sciences from Sharif University of Technology, Tehran, Iran, in 2011. He obtained a postgraduate diploma in mathematics from ICTP, Trieste, Italy, in 2012. He was a research assistant in the mathematical biology group at Institute for Research in Fundamental Sciences, Iran, from 2012 to 2014. He received his M.A.Sc. degree in electrical and computer engineering from Concordia University, Montreal, Canada, in 2016, and his Ph.D. from Automatic Control Laboratory, ETH Zurich. Currently, he is an assistant professor at Delft Center for Systems and Control, TU Delft. He has won several awards, including the National Mathematics Olympiad gold medal, the Outstanding Student Paper Award in CDC 2020, and the Outstanding Reviewer Award for IEEE Journal of Control Systems Letters. His research interests are on data-driven and learning-based modeling, control and optimization methods, and their applications in buildings, energy, and industrial systems.