FlowMixer: A Depth-Agnostic Neural Architecture for Interpretable Spatiotemporal Forecasting

Fares B. Mehouachi
New York University in Abu Dhabi
Abu Dhabi, UAE
fm2620@nyu.edu

Saif Eddin Jabari
New York University
Abu Dhabi, UAE & Brooklyn, USA
sej7@nyu.edu

Abstract

We introduce FlowMixer, a single-layer neural architecture that leverages constrained matrix operations to model structured spatiotemporal patterns with enhanced interpretability. FlowMixer incorporates non-negative matrix mixing layers within a reversible mapping framework—applying transforms before mixing and their inverses afterward. This shape-preserving design enables a Kronecker-Koopman eigenmodes framework that bridges statistical learning with dynamical systems theory, providing interpretable spatiotemporal patterns and facilitating direct algebraic manipulation of prediction horizons without retraining. The architecture's semi-group property enables this single layer to mathematically represent any depth through composition, eliminating depth search entirely. Extensive experiments across diverse domains demonstrate FlowMixer's long-horizon forecasting capabilities while effectively modeling physical phenomena such as chaotic attractors and turbulent flows. Our results achieve performance matching state-of-the-art methods while offering superior interpretability through directly extractable eigenmodes. This work suggests that architectural constraints can simultaneously maintain competitive performance and enhance mathematical interpretability in neural forecasting systems.

1 Introduction

The ability to predict complex spatiotemporal patterns remains a fundamental challenge across scientific disciplines, including climate modeling, biological systems, and fluid dynamics [1, 2, 3]. Statistical methods have traditionally dominated time series forecasting [4], while mechanistic models form the backbone of dynamical systems analysis [5]. Recent advances in Machine Learning have demonstrated that these approaches can be complemented and extended through data-driven methods [6, 7]. For example, reservoir computing, originally designed for chaos prediction [1, 7, 8], has shown promise in statistical forecasting [9, 10], while neural architectures for time series forecasting have successfully captured chaotic dynamics [11]. This convergence hints at the possibility of a unified framework for spatiotemporal pattern learning [12, 13].

Achieving this integration presents several challenges. Time series data often exhibit non-stationarity, where statistical properties evolve over time, creating distribution shifts between training and testing [14]. Dynamical systems, although low-dimensional, may exhibit chaotic behavior characterized by extreme sensitivity to initial conditions [15, 16, 17, 18]. Additionally, the differing operational requirements of these tasks—typically multi-step predictions for time series and iterative one-step predictions for dynamical systems—demand flexible and efficient temporal modeling capabilities.

Recent neural architectures have made significant progress in addressing these challenges. Transformer-based models like PatchTST [19] effectively capture long-range dependencies through attention mechanisms and patch tokenization. MLP-based architectures such as TimeMixer++ [20]

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

and WPMixer [21] achieve impressive efficiency through multi-scale decomposition and structured mixing operations. State space models, including Chimera [22], offer promising capabilities for modeling multivariate time series with coupled dynamics. Meanwhile, Gilpin [23] has shown that large-scale statistical models can effectively capture the behavior of chaotic systems, while foundation models [24, 25] demonstrate emerging capabilities for few-shot and zero-shot forecasting. These architectures typically require extensive hyperparameter search to determine optimal network depth, which varies significantly across datasets and domains.

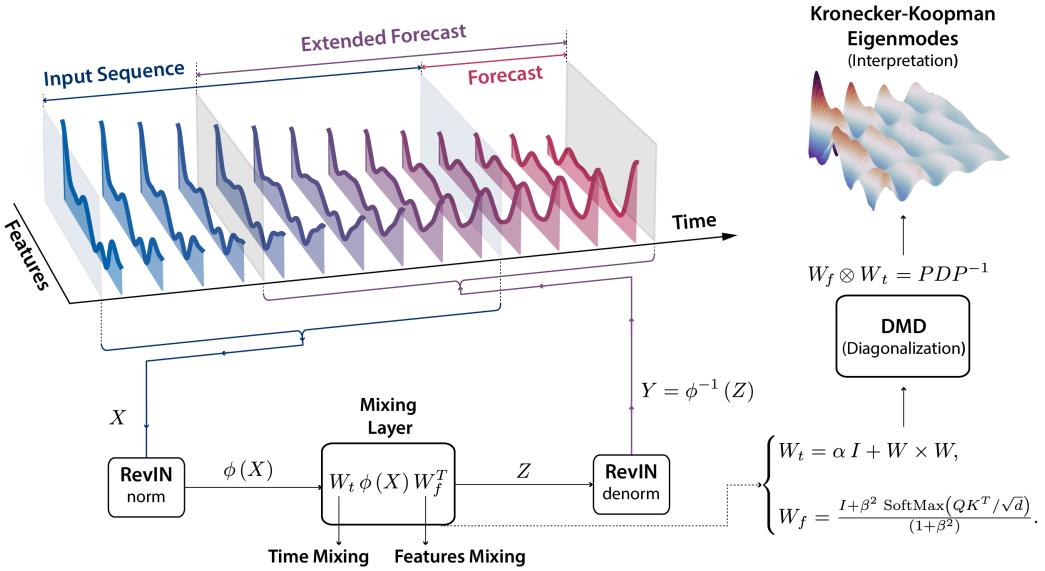


Figure 1: Overview of FlowMixer's single-layer constrained architecture. The core architecture (bottom) processes input sequences using three key components: (1) a reversible mapping, mainly reversible instance normalization (RevIN) to handle distribution shifts, (2) a constrained mixing layer with non-negative time mixing (W_t) and stochastic feature mixing (W_f) matrices, and (3) adaptive skip connections embedded within the mixing matrices. The spatiotemporal evolution (top) demonstrates how data is padded to create square mixing matrices, enabling eigendecomposition and efficient temporal modeling and interpretation. The Kronecker-Koopman eigenmodes (right), derived from the architecture's mathematical structure ($W_f \otimes W_t = PDP^{-1}$), provide a space-time decomposition of the mixer input as a weighted sum of space-time eigenmodes.

Here, we introduce FlowMixer (Fig. 1), a neural architecture designed to address these challenges through constrained design. Inspired by non-negative matrix factorization [26] and Dynamic Mode Decomposition (DMD) [27], FlowMixer aims to draw connections between statistical learning approaches and dynamical systems theory while providing a mathematically interpretable framework. FlowMixer explores a different point in the design space: rather than optimizing architectures for individual domains, we demonstrate that careful constraints enable a single architecture to achieve competitive performance across time series forecasting, with additional capabilities in chaos prediction and turbulent flow modeling. This cross-domain versatility, combined with interpretable eigenmodes and elimination of depth search, offers practical advantages for applications requiring consistent behavior across varied spatiotemporal phenomena.

Our contributions include:

- A constrained architecture whose semi-group property eliminates depth as a hyperparameter: a single layer is all you need.
- A shape-preserving design that enables direct extraction of Kronecker-Koopman eigenmodes for interpretability and allows algebraic prediction horizon change without retraining.
- A simplified chaos prediction approach through Semi-Orthogonal Basic Reservoir (SOBR) that matches specialized frameworks while maintaining architectural simplicity.
- Empirical validation across time series forecasting, chaotic systems, and turbulent flows demonstrating competitive cross-domain performance.

2 Related Work

2.1 Time Series Forecasting Architectures

Recent advances in time-series forecasting have focused on improving accuracy through architectural sophistication. TSMixer [28] employs all-MLP time and feature mixing but requires empirical determination of optimal layer count. TimeMixer++ [20] employs hierarchical multi-scale decomposition with bidirectional mixing for seasonal and trend patterns. Chimera [22] extends state space models to multivariate time series through a 2D SSM architecture with three-headed processing along time and variate dimensions. While these methods achieve strong performance, they compound the deployment challenge: each new dataset potentially requires extensive architecture search from scratch.

2.2 Koopman-Based Neural Methods

The integration of Koopman operator theory with neural networks has produced sophisticated forecasting frameworks [29, 30, 31]. Koopa [29] employs multi-layer architectures with eigenvalue-based stability checking and explosion prevention mechanisms, while KNF [30] utilizes a three-term loss function across various network components. Both approaches treat the Koopman semi-group property $\mathcal{K}_t \circ \mathcal{K}_s = \mathcal{K}_{t+s}$ as an optimization target. Notably, their eigenmodes remain deeply embedded within neural network layers—inaccessible for direct analysis.

2.3 Cross-Domain Spatiotemporal Methods

Few architectures attempt unified modeling across statistical and physical domains. Reservoir computing [1, 7] bridges chaos prediction and time series but requires careful tuning of reservoir size, spectral radius, and input scaling. N-BEATS [32], though designed for time series, demonstrates exceptional performance across chaotic systems [11], suggesting untapped cross-domain potential.

FlowMixer sits at the intersection of these research directions: it adopts the mixing paradigm from time series architectures, guarantees Koopman's mathematical properties by construction, and achieves cross-domain applicability through a single constrained design.

3 FlowMixer Architecture

FlowMixer builds upon TSMixer [28] by addressing the challenge of layer count optimization. Instead of empirical layer stacking, we developed a single mixing block with semi-group properties where successive applications combine algebraically (see Section 4.1). This approach eliminates depth as a hyperparameter while enabling mathematical manipulation of prediction horizons (see Section 4.3). The various components of this architecture are presented hereafter.

3.1 Formal Definition

Let $X \in \mathbb{R}^{n_t \times n_f}$ be an input tensor, where n_t denotes the number of time steps and n_f the number of features. FlowMixer defines a transformation $\mathcal{F} : \mathbb{R}^{n_t \times n_f} \to \mathbb{R}^{n_t \times n_f}$ as:

$$\mathcal{F}(X, W_t, W_f, \phi) = \phi^{-1}(W_t \phi(X) W_f^T) \tag{1}$$

where $\phi : \mathbb{R}^{n_t \times n_f} \to \mathbb{R}^{n_t \times n_f}$ is a reversible mapping, $W_t \in \mathbb{R}^{n_t \times n_t}$ is a time mixing matrix, and $W_f \in \mathbb{R}^{n_f \times n_f}$ is a feature mixing matrix. FlowMixer preserves tensor dimensions: the output $\mathcal{F}(X) \in \mathbb{R}^{n_t \times n_f}$ approximates a target Y of the same shape, where the last h rows of Y contain the forecast and earlier rows contain historical data. Padding with historical data, ensures square mixing matrices, enabling eigendecomposition and semi-group composition. The name "FlowMixer" reflects this reversible flow of data around the central mixing operations.

3.2 Core Components

Time-Dependent Reversible Instance Normalization: We employ mainly RevIN [33] as ϕ for feature-wise normalization, extending it to Time-Dependent RevIN (TD-RevIN):

$$\text{RevIN}(x_t) = a\frac{x_t - \mathbb{E}[x_t]}{\sqrt{\text{Var}(x_t) + \epsilon}} + b \to \text{TD-RevIN}(x_t) = a_t \times \frac{x_t - \mathbb{E}[x_t]}{\sqrt{\text{Var}(x_t) + \epsilon}} + b_t \tag{2}$$

where a_t, b_t are time-varying parameters, and $\times$ denotes the Hadamard product. This allows different normalization parameters for each timestep, enhancing the model's ability to handle non-stationary patterns. This TD extension is possible thanks to the shape-preserving property of FlowMixer.

Constrained Mixing: The core functionality integrates principles from non-negative matrix factorization and attention mechanisms [26, 34]. For an input tensor X, our architecture applies two complementary transformations:

$$\text{TimeMix}(X) = W_t X, \quad \text{FeatureMix}(X) = X W_f^T \tag{3}$$

The time mixing matrix W_t aligns with Koopman theory through a scaled matrix exponential:

$$W_t = \alpha e^{(W_0 \times W_0)} \approx \alpha I + \alpha W_0 \times W_0 \tag{4}$$

where $\times$ denotes Hadamard (element-wise) product. α is a scaling learnable parameter initialized at one that controls memory persistence. In practice, we use this first-order approximation with a learnable matrix W that absorbs the scaling, yielding $W_t = \alpha I + W \times W$. This design creates an analog to fractional differencing (FARIMA) [35] that adapts to varying temporal dependencies (see Appendix J). The nonnegative quadratic term forces the model to capture temporal dependencies through positive interactions only, reducing spurious correlations while introducing nonlinear interactions.

For feature mixing, we implement a static attention mechanism:

$$W_f = \frac{I + \beta^2 \text{SoftMax}(QK^T/\sqrt{d_k})}{1 + \beta^2} \tag{5}$$

where $K, Q \in \mathbb{R}^{n_f \times d_k}$ are learnable key/query matrices, d_k is the multiplication dimension of K, Q, and β modulates attention strength. This formulation has a graph-theoretical interpretation as transition probabilities in a weighted directed graph, analogous to a Markov process.

Seasonalities: For time series with inherent periodicities, we enhance the time mixing operation by leveraging a Kronecker structure:

$$W_t = \sum_p W_{r(p)} \otimes W_p, \quad p \times r\,(p) = n_t. \tag{6}$$

where $W_p \in \mathbb{R}^{p \times p}$ captures patterns at the period level p, $W_{r(p)} \in \mathbb{R}^{r \times r}$ models relationships between different period blocks, and $r \times p = n_t$. This formulation enables efficient modeling of multiple seasonal patterns while preserving the simplicity of the architecture.

SOBR: Semi-Orthogonal Basic Reservoir. For chaotic systems, we introduce SOBR—a simplified reservoir computing approach that uses random, non-trainable, semi-orthogonal matrices. SOBR replaces ϕ with $S \circ \phi$ where $S\,(X) = \sigma(U_t X U_f^T)$ and σ is an invertible activation (e.g., Leaky ReLU). These matrices are constructed to satisfy:

$$U_f U_f^T = I_{d_f}, \quad U_t U_t^T = I_{d_t} \tag{7}$$

This expansion provides a higher-dimensional representation space appropriate for modeling chaotic systems (discussed in Section 4).

4 Theoretical Analysis

The architectural constraints of FlowMixer yield unique theoretical properties that bridge concepts from statistical learning with dynamical systems theory [5, 36]. The key insights emerge from the architecture's semi-group stability.

4.1 Semi-Group Structure

Square mixing matrices and invertible standardization enable semi-group stability—compositions of FlowMixers with identical standardization ϕ remain a FlowMixer:

$$\mathcal{F}(\mathcal{F}(x, \theta, \phi), \theta', \phi) = \mathcal{F}(x, \theta'', \phi) \tag{8}$$

This property fundamentally distinguishes FlowMixer from all existing forecasting architectures: while traditional architectures require depth optimization and alter capacity with each layer, any n-layer FlowMixer composition has an equivalent single-layer representation[1]. This mathematical guarantee eliminates architecture search entirely and enables flexible horizon adjustments without retraining.

4.2 Eigen Decomposition and Kronecker-Koopman Framework

In vectorized form, FlowMixer's mixing operation reveals a Kronecker product:

$$\text{vec}(W_t X W_f^T) = (W_f \otimes W_t)\text{vec}(X) \tag{9}$$

This enables the extension of classical Dynamic Mode Decomposition (DMD) [27] to coupled spatiotemporal patterns in a novel principled way without Hankelization [37, 38]. Given eigendecompositions $W_f = PDP^{-1}$ and $W_t = QEQ^{-1}$, where $D = \text{diag}(\lambda_i)$ and $E = \text{diag}(\mu_j)$, any input matrix X can be expressed as:

$$X = \sum_{i,j} a_{i,j}(q_i p_j^T) \tag{10}$$

where q_i, p_j are columns of Q and P, and $a_{i,j} = (Q^{-1}XP^{-T})_{i,j}$ are projection coefficients.

Unlike Koopman neural methods where eigenmodes remain embedded [29, 30], these modes are directly extractable through standard matrix decomposition. Both time and feature eigenvectors form bases in their respective spaces. These components form the Kronecker-Koopman (KK) spatiotemporal eigenmodes:

$$\Phi_{i,j} = q_i \otimes p_j, \quad (i,j) \in [1, n_t] \times [1, n_f] \tag{11}$$

The Kronecker product preserves completeness, yielding $n_t \times n_f$ basis eigenmodes spanning all possible spatiotemporal patterns. These modes are analogous to Koopman modes [39, 40], but naturally couple spatial and temporal patterns. Section 5.2 provides an example visualization of these eigenmodes.

4.3 Continuous Algebraic Horizon Modification

The KK structure enables direct algebraic modification of prediction horizons. For a horizon change from h_0 to h_1 and a ratio $t = h_1/h_0$, we can express the modified prediction as:

$$X(t) \leftarrow \sum_{i,j} a_{i,j} \underbrace{\left(q_i p_j^T\right)}_{\text{KK Eigenmodes}} \exp\left(t \log \underbrace{\left(\lambda_i \mu_j\right)}_{\text{KK Eigenvalues}}\right). \tag{12}$$

This enables continuous horizon adjustment through interpolation ($t < 1$) or extrapolation ($t > 1$) (See Appendix H.2). The framework extends naturally to derivatives. For $\alpha > 0$:

$$\frac{\partial^\alpha X}{\partial t^\alpha} = \sum_{i,j} a_{i,j} \left(q_i p_j^T\right) \left(\lambda_i \mu_j\right)^t \left(\log\left(\lambda_i \mu_j\right)\right)^\alpha \tag{13}$$

The eigenmodes structure remains invariant during these modifications, with temporal evolution controlled mainly through eigenvalue scaling. Stability requires that all eigenvalues are inside the unit circle and can be implemented through unitary constraints on time mixing matrices [41]. The feature matrix is already stochastic and verifies this property.

5 Experimental Validation

5.1 Long-Horizon Time-Series Forecasting

Time series forecasting presents persistent challenges, particularly for extended prediction horizons where error accumulation compounds modeling difficulties. We evaluate FlowMixer on benchmark

[1] For SOBR models, semi-group property holds in lifted space.

datasets widely used in the forecasting literature, including ETT (electricity transformer temperature), Weather, Electricity, and Traffic datasets [42]. Our comparisons include recent architectures such as TimeMixer++ [20], Chimera [22], MSHyper [43, 44], CycleNet [45], and TSMixer [28]. FlowMixer deliberately forgoes the architectural complexity of these methods: no hierarchical decomposition, no multi-scale processing, no specialized seasonal modules. While the semi-group property guarantees composition e.g. $\mathcal{F}_{(96)} \circ \mathcal{F}_{(96)} = \mathcal{F}_{(192)}$, enabling algebraic horizon modification, we train each horizon independently for fair comparison (see Appendix H). The practical implication: practitioners can directly transfer FlowMixer between projects without architectural modifications. Implementation details and hyperparameters are provided in Appendices A and C, respectively.

Table 1: Performance comparison of time series forecasting models on multiple datasets across different prediction horizons. Results show Mean Squared Error (MSE) and Mean Absolute Error (MAE), with lower values indicating better performance. **Bold blue** is best, orange is second best. FlowMixer achieves comparable performance using a single operational layer, eliminating the depth hyperparameter search required by all baseline methods. Statistical validation via Wilcoxon signed-rank tests confirms significance ($p<0.05$, see Appendix D).

		FlowMixer (Ours)		**Chimera** (2024)		**TimeMixer++** (2024)		**iTransformer** (2024)		**Ada-MSHyper** (2024)		**CycleNet** (2024)		**TSMixer** (2023)	
	h	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.355**	**0.387**	0.366	0.392	0.361	0.403	0.386	0.405	0.372	0.393	0.375	0.395	0.361	0.392
	192	**0.394**	**0.409**	0.402	0.414	0.416	0.441	0.441	0.436	0.433	0.417	0.436	0.428	0.404	0.418
	336	0.415	0.423	**0.406**	**0.419**	0.430	0.434	0.487	0.458	0.422	0.433	0.496	0.455	0.420	0.431
	720	**0.433**	0.454	0.458	0.477	0.467	**0.451**	0.503	0.491	0.445	0.459	0.520	0.484	0.463	0.472
ETTh2	96	0.264	0.330	**0.262**	**0.327**	0.276	0.328	0.297	0.349	0.283	0.332	0.298	0.344	0.274	0.341
	192	**0.320**	**0.368**	**0.320**	0.372	0.342	0.379	0.380	0.400	0.358	0.374	0.372	0.396	0.339	0.385
	336	0.344	0.396	**0.316**	**0.381**	0.346	0.398	0.428	0.432	0.428	0.437	0.431	0.439	0.361	0.406
	720	**0.388**	0.452	0.389	0.430	0.392	**0.415**	0.427	0.445	0.413	0.432	0.450	0.458	0.445	0.470
ETTm1	96	0.298	0.345	0.318	0.354	0.310	**0.334**	0.334	0.368	0.301	0.354	0.319	0.360	**0.285**	0.339
	192	0.333	0.365	0.331	0.369	0.348	**0.362**	0.377	0.391	0.345	0.375	0.360	0.381	**0.327**	0.365
	336	0.360	0.386	0.363	0.389	0.376	0.391	0.426	0.420	0.375	0.397	0.389	0.403	**0.356**	**0.382**
	720	**0.398**	**0.410**	0.409	0.415	0.440	0.423	0.491	0.459	0.437	0.435	0.447	0.441	0.419	0.414
ETTm2	96	**0.159**	0.252	0.169	0.265	0.170	**0.245**	0.180	0.264	0.165	0.257	0.163	0.249	0.163	0.252
	192	**0.211**	**0.290**	0.221	**0.290**	0.229	0.291	0.250	0.309	0.230	0.307	0.229	**0.290**	0.216	**0.290**
	336	**0.260**	**0.324**	0.279	0.339	0.303	0.343	0.311	0.348	0.282	0.328	0.284	0.327	0.268	**0.324**
	720	**0.333**	0.377	0.341	**0.376**	0.373	0.399	0.412	0.407	0.375	0.396	0.389	0.391	0.420	0.422
Weather	96	**0.143**	**0.194**	0.146	0.206	0.155	0.205	0.174	0.214	0.157	0.195	0.158	0.203	0.145	0.198
	192	**0.185**	**0.235**	0.189	0.239	0.201	0.245	0.221	0.254	0.218	0.259	0.207	0.247	0.191	0.242
	336	**0.235**	0.276	0.244	0.281	0.237	0.265	0.278	0.296	0.251	**0.252**	0.262	0.289	0.242	0.280
	720	0.305	0.326	**0.297**	**0.309**	0.312	0.334	0.358	0.347	0.304	0.328	0.344	0.344	0.320	0.326
Electricity	96	**0.131**	0.226	0.132	0.234	0.135	**0.222**	0.148	0.240	0.135	0.238	0.136	0.229	**0.131**	0.229
	192	0.146	0.239	**0.144**	**0.223**	0.147	0.235	0.162	0.253	0.152	0.239	0.152	0.244	0.151	0.246
	336	0.160	0.255	**0.156**	0.259	0.164	**0.245**	0.178	0.269	0.168	0.266	0.170	0.264	0.161	0.261
	720	0.195	0.289	**0.184**	**0.280**	0.212	0.310	0.225	0.317	0.212	0.293	0.212	0.299	0.197	0.293
Traffic	96	0.377	0.264	**0.366**	**0.248**	0.392	0.253	0.395	0.268	0.384	**0.248**	0.458	0.296	0.376	0.264
	192	**0.388**	0.268	0.394	0.292	0.402	**0.258**	0.417	0.276	0.401	**0.258**	0.457	0.294	0.397	0.277
	336	**0.401**	0.275	0.409	0.311	0.428	0.263	0.433	0.283	0.423	**0.261**	0.470	0.299	0.413	0.290
	720	**0.434**	0.291	0.443	0.294	0.441	**0.282**	0.467	0.302	0.453	**0.282**	0.502	0.314	0.444	0.306

Table 1 demonstrates FlowMixer's competitive performance across multiple datasets and horizons, with particularly strong results on ETTm2 and the 720-timestep horizon for Traffic and ETT forecasting tasks. What distinguishes FlowMixer is its architectural simplicity—unlike contemporary models with deep architectures and hierarchical decompositions, it achieves these results consistently using just a single operational layer with carefully designed constraints. The computational efficiency directly benefits from this simplicity, with FlowMixer processing most ETT datasets in under 2 seconds per epoch (see Appendix C). Statistical validation through Wilcoxon signed-rank tests confirms that these improvements are significant ($p<0.05$, see Appendix D). Ablation studies confirm these architectural choices contribute to the model's forecasting capabilities (Appendix F), demonstrating how carefully targeted constraints enhance multivariate time series prediction.

5.2 Kronecker-Koopman Analysis of Spatiotemporal Patterns

FlowMixer offers a novel perspective on spatiotemporal analysis beyond the traditional Dynamic Mode Decomposition (DMD) framework [27]. Combining time and feature spectral elements yields the Kronecker-Koopman (KK) eigenmodes (see Section 4). To illustrate these patterns, we compute KK eigenmodes from the traffic dataset [46] enabling effective visualization of spatial components.

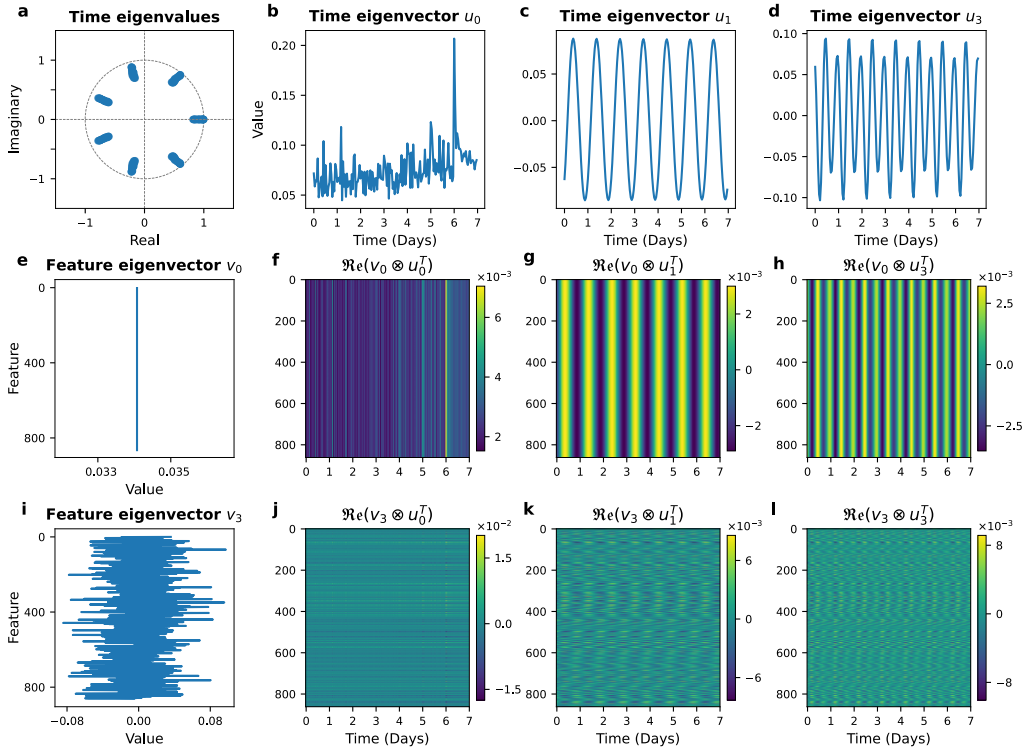


Figure 2: Visualization of Kronecker-Koopman Eigenmodes for traffic dataset. (a) Time eigenvalues distribution, with aligned angular values indicating the periodic nature of the traffic datasets. (b-d) First three time eigenvectors (real part). (e,i) First and Third space eigenvectors, with the first showing constant values consistent with the stochastic space mixing matrix (representing a Markov transition process). (f-h, j-l) Real parts of Kronecker-Koopman Eigenmodes revealing space-time patterns. The periodic structures in time eigenvectors and coherent patterns in higher-mode products demonstrate how FlowMixer captures spatiotemporal dynamics.

When applied to traffic flow data, KK analysis yields structural insights. The time eigenvalues distribution (Figure 2a) reveals clear angular alignments corresponding to daily, weekly, and intra-day traffic periodicities. The primary time eigenvector (b) captures baseline (trend) traffic volume with weekend transitions, while subsequent eigenvectors (c-d) exhibit accelerating oscillatory behavior (seasonalities) reflecting 24-hour and 12-hour cycles. Feature eigenvectors (e-g) encode spatial relationships between traffic sensors from global to local scales, with the dominant eigenvector exhibiting the expected constant structure characteristic of left stochastic matrices. The resulting Kronecker-Koopman spectrum, especially higher-order modes (j-l), manifests coherent cross-dimensional patterns that reflect propagation phenomena consistent with congestion dynamics in traffic theory [47, 48].

This framework provides spatiotemporal modes beyond classical DMD [27] and Hankelized-DMD [37, 38] by separating temporal and spatial components instead of relying on Hankelized embeddings. Our approach captures the inherent structure of any system where spatiotemporal patterns govern dynamics, and expands the traditional time series dichotomy, from a classic trend/seasonality decomposition to a spatiotemporal framework:

$$\hat{y}_t = \hat{y}_t^{\text{trend}} + \hat{y}_t^{\text{seasonal}} \xrightarrow{\text{KK}} \left(\hat{y}_t^{\text{trend}} + \hat{y}_t^{\text{seasonal}}\right) \otimes \left(\hat{x}_t^{\text{global}} + \hat{x}_t^{\text{local}}\right)^T \tag{14}$$

5.3 Prediction of Chaotic Dynamical Systems

The prediction of chaotic systems represents one of science's fundamental challenges, where exponential sensitivity to initial conditions traditionally limits long-term forecasting [18]. Specialized architectures like reservoir computing have advanced this field significantly [1, 12, 49, 7, 50], often focusing on recurrent frameworks for iterative single-step predictions. We explore how FlowMixer's constrained design principles can contribute to this domain through a complementary approach.

The key component is SOBR (Semi-Orthogonal Basic Reservoir)—an adaptation of reservoir computing principles that aligns with FlowMixer's constrained design philosophy (cf. Section 3.2). While Machine Learning approaches have traditionally addressed high-dimensional problems [51], chaotic systems present a different challenge: complex dynamics emerging from low dimensions (e.g., the three-dimensional Lorenz system). SOBR addresses this dimensional consideration through a targeted expansion: it projects data into higher-dimensional spaces using random semi-orthogonal matrices, maintaining full rank and unit spectral radius. This Koopman-inspired dimensional lifting enhances FlowMixer's pattern recognition capabilities. FlowMixer's results for chaotic attractors without SOBR are provided in Appendix K.3.

SOBR integrates with FlowMixer through the reversible mapping ϕ. In its simplest form it is expressed as $S(X) = \sigma(U_t X U_f^T)$, where semi-orthogonal matrices U_f and U_t expand feature and time dimensions respectively, complemented with an invertible activation σ (typically Leaky ReLU). This design enhances FlowMixer's ability to capture chaotic attractor dynamics. While this SOBR integration preserves the semi-group property only in the lifted space, not the original space (see Appendix K.2), the benefits of dimensional lifting outweigh this tradeoff for chaotic systems, where accurate pattern recognition takes precedence over direct algebraic horizon manipulation.

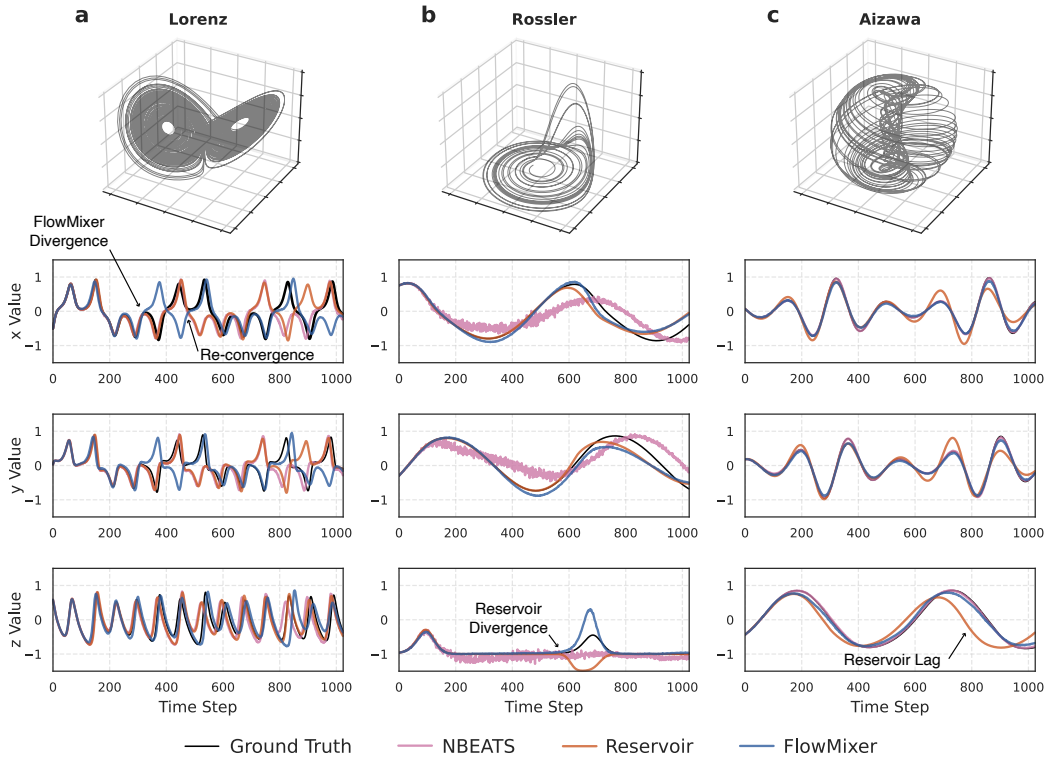


Figure 3: Predictions of Lorenz (a), Rössler (b), and Aizawa (c) chaotic attractors (scaled [-1,1]). Each row shows the evolution of x, y, and z variables over time. Ground truth (black), FlowMixer (blue), Reservoir Computing (orange) [1, 52, 53], and N-BEATS (purple) [32] trajectories are compared. While all methods capture the Lorenz attractor's structure, N-BEATS shows a notable deviation in the Rössler system, and Reservoir Computing shows differences particularly in the z-component predictions. FlowMixer exhibits consistent performance across all three systems. Experimental settings are summarized in Appendix C and presented in detail in Appendix K.4.

We evaluate this approach on three canonical chaotic systems: the Lorenz butterfly [15], Rössler attractor [16], and Aizawa system [17]. Rather than employing iterative single-step predictions, FlowMixer processes 32-step sequences to forecast the next 32 steps directly. By compounding these predictions, we obtain forecasts over 1024 steps—approximately 9-10 Lyapunov times for the Lorenz system. Figure 3 compares FlowMixer's performance with Reservoir Computing (RC) [1, 52, 53] and N-BEATS [11, 32]. While all methods capture the general attractor structure, they exhibit different failure modes: RC struggles with accurate z-component predictions particularly in the Lorenz system, N-BEATS diverges notably in the Rössler system, while FlowMixer maintains consistent accuracy across all three attractors. The model captures characteristic features from the Lorenz butterfly to the Aizawa spiral dynamics, occasionally realigning with ground truth trajectories after divergence—suggesting effective pattern learning. These results illustrate how architectural constraints can maintain predictive capabilities for chaotic systems. FlowMixer with SOBR offers an alternative approach that complements existing specialized frameworks. This same single-layer architecture that forecasts time series also captures chaotic dynamics. This connection between statistical and dynamical modeling suggests broader implications for scientific computing, which we explore next through turbulent flow prediction.

5.4 Predicting 2D Turbulent Flows

The prediction of turbulent flows represents a crucial challenge at the intersection of computational physics and engineering [54]. While direct numerical simulation of the Navier-Stokes equations provides high accuracy, its computational demands limit practical applications. Recent physics-informed neural networks (PINNs) [55, 56] have made valuable contributions by embedding physical constraints into learning architectures, though these approaches often involve balancing multiple loss terms and addressing convergence considerations [57].

We demonstrate how FlowMixer's constrained architecture offers a complementary data-driven approach on two canonical test cases: two-dimensional flow past a cylinder at Reynolds number $Re = 150$ and flow around a NACA airfoil at $Re = 1000$ with 15° angle of attack. For 2D fluid dynamics prediction, FlowMixer processes the vorticity field ω directly, with the core transformation applied as:

$$\omega_{t+1:t+h} = \phi^{-1}(W_t \phi(\omega_{t-h+1:t}) W_f^T) \tag{15}$$

where $\omega_{t-h+1:t}$ represents a sequence of h consecutive vorticity snapshots, and $\omega_{t+1:t+h}$ is the predicted sequence for the next h timesteps.

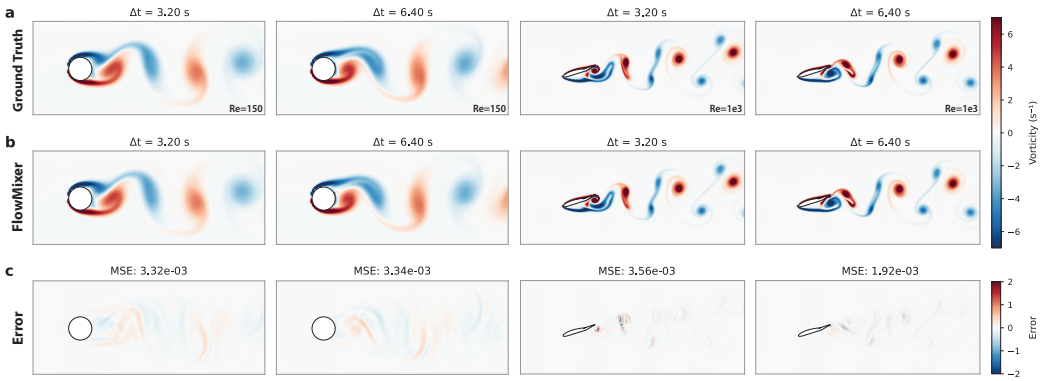


Figure 4: Prediction of vorticity fields for flow past a cylinder at $Re = 150$ (left columns) and a NACA airfoil at $Re = 1000$ (right columns). (a) Ground truth vorticity fields showing the evolution of wake structures. (b) FlowMixer predictions demonstrate accurate capture of both near-body structures and downstream vortex evolution. (c) Error fields with corresponding Mean Squared Error (MSE) values show minimal discrepancy between prediction and ground truth across varying geometries and flow regimes. Experimental details are provided in Appendix N. Additional flow visualizations and analyses are provided in Appendix N.4 and Appendix N.5.

Our simulations capture the full dynamics across computational domains with 400×160 resolution. The cylinder flow is characterized by the von Kármán vortex street with a Strouhal number of approximately 0.2, while the airfoil case introduces asymmetric geometry, flow separation, and higher Reynolds number effects. For both cases, we train FlowMixer to process 64 consecutive flow field snapshots and predict the next 64 timesteps ($h = 64$)—representing physical time horizons of 6.4 seconds for the cylinder and the airfoil. Figure 4 compares the predicted vorticity fields with ground truth. For the cylinder case, FlowMixer accurately captures the alternating clockwise (blue) and counterclockwise (red) rotation patterns in the wake. In the more challenging airfoil case, the model successfully predicts the complex separation patterns, asymmetric wake structures, and vortex interactions despite the substantially higher Reynolds number.

Quantitatively, FlowMixer maintains MSE values of approximately 3.3e-3 for the cylinder case and 2e-3 to 8e-3 for the airfoil case, corresponding to approximately 4-5% relative l^2 error. These results are competitive with existing approaches in the literature [58, 59, 60], while using standard stochastic gradient descent with momentum for training rather than specialized optimization procedures (comparisons of optimization strategies are presented in Appendix N.5).

FlowMixer adapts to increased flow complexity without architectural modifications. When transitioning from the regular vortex shedding of the cylinder at $Re = 150$ to the higher Reynolds number airfoil case $Re = 1000$ with complex separation and wake interactions, the model maintains comparable accuracy using the same architectural constraints. The error fields (Figure 4c) show relatively uniform distribution without significant localized spikes, suggesting the model effectively captures dominant flow patterns across different regimes. This behavior is noteworthy given that other approaches often require explicit conservation constraints to achieve stable predictions across varying flow regimes [61, 62, 63].

The successful prediction of complex flow patterns around both simple and complex geometries at different Reynolds numbers demonstrates the versatility of FlowMixer's architecture. The ability to capture these dynamics without specialized physical constraints points toward applications in aerodynamic design optimization, real-time flow monitoring, and reduced-order modeling for engineering systems where both computational efficiency and accuracy are essential requirements.

Conclusion

FlowMixer demonstrates how architectural constraints inspired by dynamical systems theory enhance both predictive capabilities and interpretability in neural forecasting systems. Through extensive validation across multiple domains, we have shown this constrained approach offers competitive performance in long-horizon forecasting while effectively modeling complex physical phenomena such as chaotic attractors and turbulent flows.

The architecture introduces three complementary innovations: (1) constrained non-negative matrix mixing operations, (2) a Kronecker-Koopman framework providing interpretable spatiotemporal patterns, and (3) a Semi-Orthogonal Basic Reservoir approach enabling stable prediction of chaotic systems. Together, these elements create a unified framework bridging statistical learning and dynamical systems modeling.

Our results highlight an important principle in neural architecture design: carefully chosen mathematical constraints can simultaneously maintain competitive performance while enhancing interpretability without necessitating increased model complexity. Notably, the semi-group composition property eliminates depth optimization entirely: one layer is all you need. This frees practitioners from a critical deployment bottleneck that affects all existing architectures.

We acknowledge current limitations, including cubic computational complexity in sequence length and the absence of explicit physical conservation guarantees. Future work will explore sparsification techniques to improve computational scaling and investigate integration with physics-informed approaches to enhance scientific applications.

As deep learning advances scientific computing, architectures like FlowMixer demonstrate the value of incorporating domain knowledge through mathematical design principles rather than explicit regularization—offering a promising direction for neural forecasting systems that are both effective and interpretable across scientific and engineering applications.

Code Availability

The code for FlowMixer is available at https://github.com/FaresBMehouachi/FlowMixer. The authors declare no competing interests.

Acknowledgment

This work was supported by the NYUAD Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001. The views expressed in this article are those of the authors and do not reflect the opinions of CITIES or their funding agencies

References

[1] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

[2] Ricardo Vinuesa and Steven L Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

[3] Mingyu Wang and Jianping Li. Interpretable predictions of chaotic dynamical systems using dynamical system deep learning. *Scientific Reports*, 14(1):3143, 2024.

[4] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, Hoboken, New Jersey, 5 edition, 2015.

[5] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[6] Steven L Brunton and J Nathan Kutz. Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 4(7):483–494, 2024.

[7] Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. Next generation reservoir computing. *Nature communications*, 12(1):5564, 2021.

[8] Xin Li, Qunxi Zhu, Chengli Zhao, Xiaojun Duan, Bolin Zhao, Xue Zhang, Huanfei Ma, Jie Sun, and Wei Lin. Higher-order granger reservoir computing: simultaneously achieving scalable complex structures inference and accurate dynamics prediction. *Nature Communications*, 15(1):2506, 2024.

[9] Chao Du, Fuxi Cai, Mohammed A Zidan, Wen Ma, Seung Hwan Lee, and Wei D Lu. Reservoir computing using dynamic memristors for temporal information processing. *Nature communications*, 8(1):2204, 2017.

[10] Filippo Maria Bianchi, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. Reservoir computing approaches for representation and classification of multivariate time series. *IEEE transactions on neural networks and learning systems*, 32(5):2169–2179, 2020.

[11] William Gilpin. Chaos as an interpretable benchmark for forecasting and data-driven modelling. *arXiv preprint arXiv:2110.05266*, 2021.

[12] Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

[13] Min Yan, Can Huang, Peter Bienstman, Peter Tino, Wei Lin, and Jie Sun. Emerging opportunities and challenges for the future of reservoir computing. *Nature Communications*, 15(1):2056, 2024.

[14] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press, Cambridge, MA, 2012.

[15] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[16] Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[17] William F Langford. Numerical studies of torus bifurcations. In *Numerical Methods for Bifurcation Problems: Proceedings of the Conference at the University of Dortmund, August 22–26, 1983*, pages 285–295. Springer, 1984.

[18] Steven H Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, Boca Raton, 2 edition, 2018.

[19] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[20] Shiyu Wang, Jiawei Li, Xiaoming Shi, Zhou Ye, Baichuan Mo, Wenze Lin, Shengtong Ju, Zhixuan Chu, and Ming Jin. Timemixer++: A general time series pattern machine for universal predictive analysis. *arXiv preprint arXiv:2410.16032*, 2024.

[21] Md Mahmuddun Nabi Murad, Mehmet Aktukmak, and Yasin Yilmaz. Wpmixer: Efficient multi-resolution mixing for long-term time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 19581–19588, 2025.

[22] Ali Behrouz, Michele Santacatterina, and Ramin Zabih. Chimera: Effectively modeling multivariate time series with 2-dimensional state space models. *Advances in Neural Information Processing Systems*, 37:119886–119918, 2024.

[23] William Gilpin. Model scale versus domain knowledge in statistical forecasting of chaotic systems. *Physical Review Research*, 5(4):043252, 2023.

[24] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[25] Yuanzhao Zhang and William Gilpin. Zero-shot forecasting of chaotic systems. *arXiv preprint arXiv:2409.15771*, 2024.

[26] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

[27] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[28] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023.

[29] Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. *Advances in neural information processing systems*, 36:12271–12290, 2023.

[30] Rui Wang, Yihe Dong, Sercan O Arik, and Rose Yu. Koopman neural operator forecaster for time-series with temporal distributional shifts. In *The Eleventh International Conference on Learning Representations*, 2023.

[31] Chuhan Yang, Fares B Mehouachi, Monica Menendez, and Saif Eddin Jabari. Urban traffic analysis and forecasting through shared koopman eigenmodes. *Nonlinear Dynamics*, 113(23):32307–32328, 2025.

[32] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[33] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[35] Clive WJ Granger and Roselyne Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of time series analysis*, 1(1):15–29, 1980.

[36] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[37] Hassan Arbabi and Igor Mezic. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

[38] Steven L Brunton, Bingni W Brunton, Joshua L Proctor, Eurika Kaiser, and J Nathan Kutz. Chaos as an intermittently forced linear system. *Nature communications*, 8(1):19, 2017.

[39] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[40] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[41] Peter J Baddoo, Benjamin Herrmann, Beverley J McKeon, J Nathan Kutz, and Steven L Brunton. Physics-informed dynamic mode decomposition. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2271), 2023.

[42] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[43] Zongjiang Shang, Ling Chen, Binqing Wu, and Dongliang Cui. Mshyper: Multi-scale hypergraph transformer for long-range time series forecasting. *arXiv preprint arXiv:2401.09261*, 2024.

[44] Zongjiang Shang, Ling Chen, Binqing Wu, and Dongliang Cui. Ada-mshyper: adaptive multi-scale hypergraph transformer for time series forecasting. *Advances in Neural Information Processing Systems*, 37:33310–33337, 2024.

[45] Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024.

[46] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[47] Martin Treiber and Arne Kesting. *Traffic flow dynamics: data, models and simulation*. Springer Science & Business Media, 2012.

[48] JWC Van Lint, Serge P Hoogendoorn, and Henk J van Zuylen. Freeway travel time prediction with state-space neural networks: Modeling state-space dynamics with recurrent neural networks. *Transportation Research Record*, 1811(1):30–39, 2002.

[49] Matteo Sangiorgio, Fabio Dercole, and Giorgio Guariso. Forecasting of noisy chaotic systems with deep neural networks. *Chaos, Solitons & Fractals*, 153:111570, 2021.

[50] Sanghyeon Choi, Jaeho Shin, Gwanyeong Park, Jung Sun Eo, Jingon Jang, J Joshua Yang, and Gunuk Wang. 3d-integrated multilayered physical reservoir array for learning and forecasting time-series information. *Nature communications*, 15(1):2044, 2024.

[51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[52] Jason A Platt, Adrian Wong, Randall Clark, Stephen G Penny, and Henry DI Abarbanel. Robust forecasting using predictive generalized synchronization in reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(12), 2021.

[53] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An interdisciplinary journal of nonlinear science*, 28(4), 2018.

[54] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

[55] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[56] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[57] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[58] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[59] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1457–1466, 2020.

[60] Alejandro Güemes, Stefano Discetti, Andrea Ianiro, Beril Sirmacek, Hossein Azizpour, and Ricardo Vinuesa. From coarse wall measurements to turbulent velocity fields through deep learning. *Physics of fluids*, 33(7), 2021.

[61] Arvind T Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding hard physical constraints in neural network coarse-graining of 3d turbulence. *arXiv preprint arXiv:2002.00021*, 2020.

[62] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[63] Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Physical review letters*, 126(9):098302, 2021.

[64] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 6989–6997, 2023.

[65] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[66] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[67] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[68] Ricky T.Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[70] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the four main contributions (constrained architecture, Kronecker-Koopman framework, SOBR approach, and empirical validation) that are substantiated throughout the paper with theoretical analysis and experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion explicitly acknowledges limitations, including "cubic computational complexity in sequence length" and "absence of explicit physical conservation guarantees," with future work directions addressing these limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 3 provides theoretical foundations with clear mathematical formulations for semi-group structure, eigendecomposition, and horizon modification. Supporting derivations and proofs appear in appendices (FARIMA connections, complexity analysis).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 4 and Appendices A-B provide comprehensive details on datasets, preprocessing, training protocols, and evaluation metrics. Hyperparameters are fully documented in Table 2 with implementation details sufficient for reproduction.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available at https://github.com/FaresBMehouachi/FlowMixer. All experiments use publicly available benchmark datasets. Implementation details and hyperparameters are documented.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 outlines experimental methodology with detailed configurations in Appendices A-B. Table 2 provides all hyperparameters including optimizer, learning rate, dropout, and batch size for each experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Standard deviations across five random seeds are reported in Table 3, and error fields are shown in Figures 4 and 7, enabling proper assessment of result reliability.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies using a single NVIDIA A100 GPU and provides time per epoch for each experiment in Table 2, ranging from 2s for ETT datasets to 122s for Electricity data.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research involves computational modeling and analyses of non-sensitive data, using publicly available benchmarks and following standard research practices in conformance with NeurIPS ethical guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our work on FlowMixer has several potential positive societal impacts. The improved spatiotemporal forecasting capabilities could benefit domains such as weather prediction, electricity consumption forecasting, and traffic management, potentially leading to more efficient resource allocation, reduced energy consumption, and improved urban planning. The interpretability aspects of our approach also enhance transparency and trust in AI forecasting systems. Additionally, accurate prediction of chaotic systems and fluid dynamics could advance scientific understanding in fields ranging from climate science to aerospace engineering.

    While we do not foresee direct negative societal impacts given the foundational nature of our research, we acknowledge that any forecasting technology could potentially lead to overreliance on predictions in critical decision-making contexts. In domains like electricity grid management or transportation systems, incorrect forecasts could lead to resource misallocation if deployed without appropriate uncertainty quantification or human oversight. Furthermore, accurate long-term forecasting in financial markets could potentially exacerbate market advantages for entities with access to advanced predictive technologies. We believe these risks are modest compared to the benefits, but they warrant consideration as applications are developed based on our methodology.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper presents a forecasting model without potential for harmful misuse that would require safeguards. The applications (time series forecasting, chaos prediction, fluid dynamics) pose minimal ethical risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and baseline methods are properly cited with appropriate attribution to original authors. The commonly used benchmark datasets (ETT, Weather, Electricity, Traffic) are used according to their intended research purposes.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: FlowMixer code is released under MIT license at the GitHub repository.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The research is computational in nature and does not involve human subjects, crowdsourced data collection, or human evaluations.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human subjects were involved in this research, so IRB approval was not required.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The research methodology does not incorporate large language models in any capacity. All models used (FlowMixer and baselines) are specialized forecasting architectures.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Appendix: Long-Horizon Time Series Forecasting

A.1 Experimental Details

We evaluated FlowMixer using seven benchmark datasets encompassing diverse domains with varying temporal resolutions and prediction challenges. The ETT datasets capture electrical transformer measurements over a two-year period, consisting of both hourly (ETTh) and 15-minute (ETTm) resolution variants. The Electricity dataset records hourly power consumption from 321 customers, while Traffic contains hourly road occupancy rates from California. The Weather dataset comprises 21 meteorological indicators sampled at 10-minute intervals throughout 2020.

Data preparation followed standard protocols, with chronological partitioning using train/validation/test ratios of 0.6/0.2/0.2 for ETT and 0.7/0.2/0.1 for others. Each feature underwent independent z-score normalization. For most datasets we use an input length of 1024 ($\geq$720) timesteps. For Electricity and Traffic, we utilize longer input sequences (up to 3072 timesteps) due to their increased feature dimensionality.

Model training was implemented in TensorFlow and executed on a single NVIDIA A100 GPU. The training protocol incorporated early stopping with patience of 10 epochs and learning rate reduction (factor 0.1, patience 5) based on validation loss. We limited training to 100 epochs with batch size 32 and systematically evaluated dropout rates between 0.0 and 0.7. The model's hyperparameters were optimized for each dataset and prediction horizon combination, with comprehensive configurations detailed in Table 3. While training hyperparameters were optimized per dataset, the core architecture remained fixed with a single mixing block—eliminating the depth search required by baseline methods.

A.2 Extra comparison

Table 2: Extra comparison of FlowMixer with N-HITS [64], FEDformer [65], Autoformer [66], Dlinear [67], and some Koopman based models: KooPA [29] and KNF [30]. The reported standard deviations for FlowMixer use five seeds. **Bold blue** indicate best performance, demonstrating FlowMixer's competitive results.

		FlowMixer (Ours)		**N-HITS** (2021)		**FEDformer** (2022)		**Autoformer** (2021)		**DLinear** (2023)		**KooPA** (2023)		**KNF** (2023)	
	h	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.358±.001**	**0.390±.001**	0.475	0.498	0.376	0.415	0.435	0.446	0.386	0.400	0.371	0.405	0.975	0.744
	192	**0.394±.001**	**0.412±.001**	0.426	0.519	0.423	0.446	0.456	0.457	0.437	0.432	0.416	0.439	0.941	0.744
	336	**0.418±.001**	**0.430±.001**	0.550	0.564	0.444	0.462	0.486	0.487	0.481	0.459	–	–	–	–
	720	**0.465±.001**	**0.475±.001**	0.598	0.641	0.469	0.492	0.515	0.517	0.519	0.516	–	–	–	–
ETTh2	96	**0.264±.001**	**0.330±.001**	0.328	0.364	0.332	0.374	0.332	0.368	0.333	0.387	0.297	0.349	0.433	0.446
	192	**0.320±.002**	**0.368±.001**	0.372	0.408	0.407	0.446	0.426	0.434	0.477	0.476	0.356	0.393	0.528	0.503
	336	**0.344±.002**	**0.396±.001**	0.397	0.421	0.400	0.447	0.477	0.479	0.594	0.541	–	–	–	–
	720	**0.420±.002**	**0.452±.002**	0.461	0.497	0.412	0.469	0.453	0.453	0.831	0.657	–	–	–	–
ETTm1	96	**0.298±.002**	**0.345±.002**	0.370	0.468	0.326	0.390	0.510	0.492	0.345	0.372	0.294	0.345	0.957	0.782
	192	**0.333±.001**	**0.365±.001**	0.436	0.488	0.365	0.415	0.515	0.514	0.380	0.389	0.337	0.378	0.896	0.731
	336	**0.363±.001**	**0.383±.001**	0.483	0.510	0.392	0.425	0.510	0.492	0.413	0.413	–	–	–	–
	720	**0.404±.001**	**0.410±.001**	0.489	0.537	0.446	0.458	0.527	0.493	0.474	0.433	–	–	–	–
ETTm2	96	**0.159±.001**	**0.252±.001**	0.176	0.255	0.180	0.271	0.205	0.293	0.192	0.282	0.171	0.254	1.535	1.012
	192	**0.211±.001**	**0.290±.001**	0.245	0.305	0.252	0.318	0.278	0.336	0.284	0.362	0.226	0.298	1.355	0.908
	336	**0.260±.001**	**0.324±.001**	0.295	0.346	0.324	0.364	0.343	0.379	0.369	0.427	–	–	–	–
	720	**0.333±.001**	**0.377±.001**	0.401	0.413	0.410	0.420	0.414	0.419	0.554	0.522	–	–	–	–
Weather	96	**0.143±.002**	**0.194±.002**	0.158	0.195	0.238	0.314	0.249	0.329	0.193	0.292	0.154	0.205	0.295	0.308
	192	**0.185±.002**	**0.235±.002**	0.211	0.247	0.275	0.325	0.325	0.325	0.228	0.296	0.193	0.241	0.462	0.437
	336	**0.235±.002**	**0.276±.001**	0.274	0.300	0.339	0.377	0.397	0.397	0.283	0.335	–	–	–	–
	720	**0.305±.001**	**0.326±.002**	0.351	0.353	0.351	0.409	1.042	1.216	0.345	0.381	–	–	–	–
Electricity	96	**0.131±.001**	**0.226±.003**	0.147	0.249	0.186	0.302	0.196	0.302	0.197	0.282	0.136	0.236	0.198	0.284
	192	**0.146±.002**	**0.239±.003**	0.167	0.249	0.197	0.311	0.211	0.321	0.196	0.285	0.156	0.254	0.245	0.321
	336	**0.160±.004**	**0.255±.002**	0.186	0.290	0.213	0.328	0.214	0.328	0.209	0.301	–	–	–	–
	720	**0.195±.003**	**0.289±.004**	0.243	0.340	0.236	0.344	0.236	0.342	0.245	0.333	–	–	–	–
Traffic	96	**0.377±.002**	**0.264±.001**	0.402	0.282	0.576	0.359	0.597	0.371	0.650	0.396	0.401	0.275	0.645	0.376
	192	**0.388±.002**	**0.268±.002**	0.420	0.297	0.610	0.380	0.607	0.382	0.598	0.370	0.403	0.284	0.699	0.405
	336	**0.401±.002**	**0.275±.004**	0.448	0.313	0.608	0.375	0.623	0.387	0.605	0.373	–	–	–	–
	720	**0.434±.003**	**0.291±.004**	0.539	0.353	0.621	0.375	0.639	0.395	0.645	0.394	–	–	–	–

B Periodicities Analysis

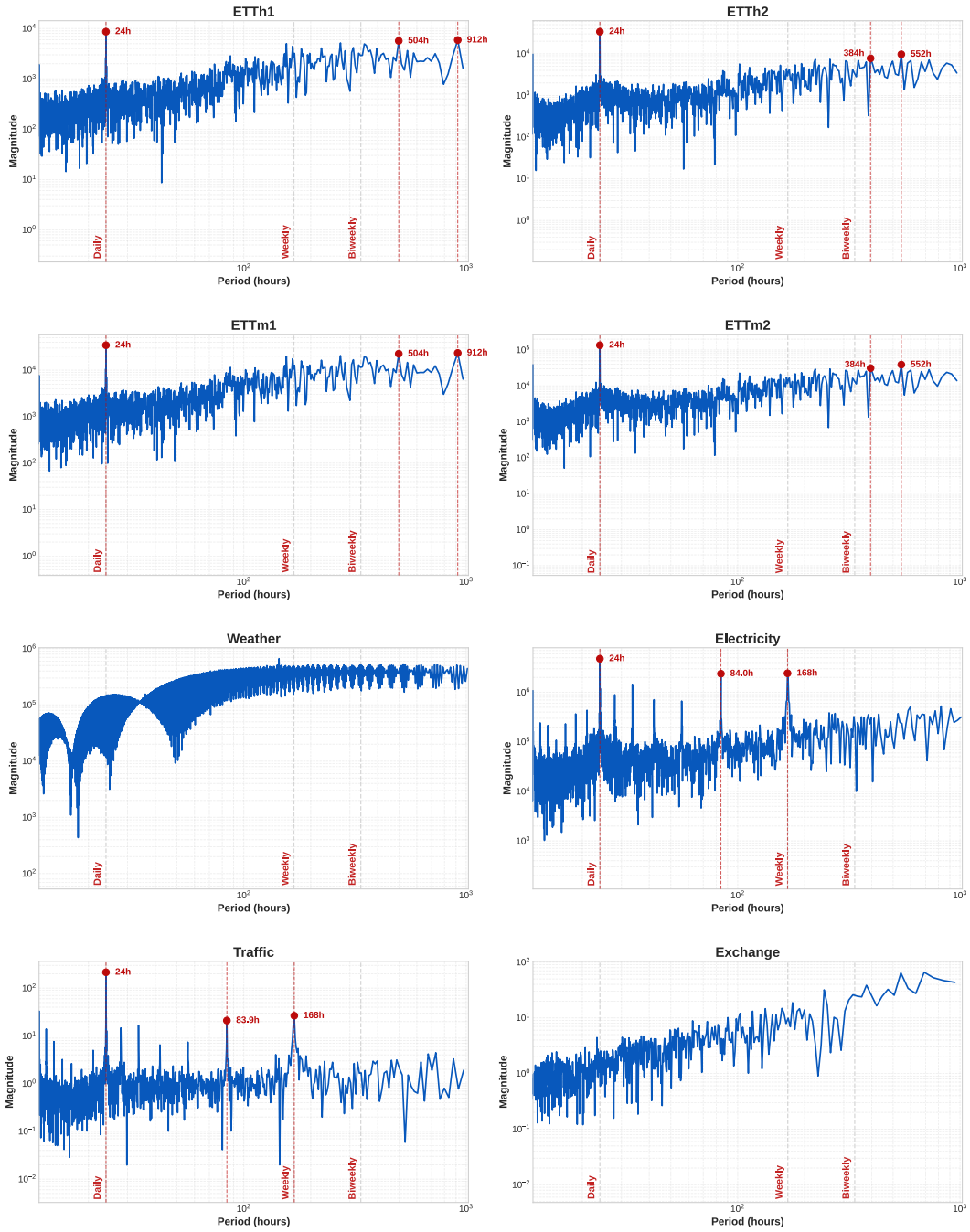


Figure 5: FFT Spectral Analysis of Periodicity Patterns in Benchmark Time Series Datasets. The plots show magnitude versus period (in hours) on logarithmic scales for eight standard forecasting datasets. Significant periodicities are marked with red dots and vertical lines, revealing prominent daily (24h) patterns across all datasets, with additional weekly (168h) and biweekly cycles of varying strengths. These spectral characteristics inform our model design, particularly for incorporating appropriate time mixing components and periodicity-aware architectures to capture multi-scale temporal dependencies. We select mainly periods of [24,168] in our experiments.

C Appendix: Hyperparameters for Experiments

Table 3: Hyperparameter configurations for different forecasting horizons (h) and input lengths (Input Len.). Parameters include optimizer choice (Optim.), initial learning rate (Init. LR), weight decay (W. Decay), RevIN type, time per epoch (T/Epoch), exponential/quadratic time mixing (Expm), and periodicities. "Turb." stands for turbulence.

	h	Input Len.	Batch	Optim.	Init. LR	W. Decay	RevIN	Time/Epoch	Expm	Periodicities
ETTh1	96	1344	16	SGD	1e-1	1e-6	RevIN	2s	True	[24, 168]
	192	1344	16	SGD	1e-1	1e-6	RevIN	2s	True	[24, 168]
	336	1344	16	SGD	1e-1	1e-6	RevIN	2s	True	[24, 168]
	720	1344	16	SGD	1e-1	1e-6	RevIN	2s	True	[24, 168]
ETTh2	96	1344	16	SGD	1e-1	1e-5	RevIN	2s	False	-
	192	1344	16	SGD	1e-1	1e-5	RevIN	2s	False	-
	336	1344	16	SGD	1e-1	1e-5	RevIN	2s	True	[24, 168]
	720	1344	16	SGD	1e-1	1e-5	RevIN	2s	True	[24, 168]
ETTm1	96	1344	16	AdamW	1e-3	1e-6	TD-RevIN	2s	False	-
	192	1344	16	AdamW	1e-3	1e-6	TD-RevIN	2s	True	[24 x 4, 168 x 4]
	336	1344	16	AdamW	1e-3	1e-6	TD-RevIN	2s	True	[24 x 4, 168 x 4]
	720	1344	16	AdamW	1e-3	1e-6	TD-RevIN	2s	True	[24 x 4, 168 x 4]
ETTm2	96	1344	16	SGD	1e-1	1e-6	RevIN	2s	False	-
	192	1344	16	SGD	1e-1	1e-6	RevIN	2s	False	-
	336	1344	16	SGD	1e-1	1e-6	RevIN	2s	False	-
	720	1344	16	SGD	1e-1	1e-6	RevIN	2s	False	-
Weather	96	1344	16	AdamW	1e-3	1e-6	TD-RevIN	8s	False	-
	192	1344	16	AdamW	1e-3	1e-6	TD-RevIN	8s	False	-
	336	1344	16	AdamW	1e-3	1e-6	TD-RevIN	8s	False	-
	720	1344	16	AdamW	1e-3	1e-6	TD-RevIN	8s	True	-
Electricity	96	2688	16	AdamW	1e-4	1e-6	RevIN	122s	False	-
	192	2688	16	AdamW	1e-4	1e-6	RevIN	122s	False	-
	336	2688	16	AdamW	1e-4	1e-6	RevIN	122s	False	-
	720	2688	16	AdamW	1e-4	1e-6	RevIN	122s	False	-
Traffic	96	4032	16	AdamW	1e-3	1e-6	TD-RevIN	98s	False	-
	192	4032	16	AdamW	1e-3	1e-6	TD-RevIN	98s	False	[24, 168]
	336	4032	16	AdamW	1e-3	1e-6	TD-RevIN	98s	False	[24, 168]
	720	4032	16	AdamW	1e-3	1e-6	TD-RevIN	98s	False	-
Chaos	32	32	32	AdamW	1e-3	1e-6	TD-RevIN + SOBR	2s	False	-
Turbu.	64	64	32	SGD	1e-1	1e-6	RevIN	28s	False	-

The hyperparameters for our time series forecasting experiments were determined through a systematic grid search across multiple dimensions. We evaluated two normalization approaches (RevIN and TD-RevIN) to address distribution shifts in time series data. For optimization strategies, we explored both AdamW (with learning rates of 0.001 and 0.0001) and SGD with momentum 0.9 (using learning rates of 0.001, 0.01, and 0.1). Additional hyperparameters included various batch sizes, weight decay values (1e-5, 1e-6), and different periodicity configurations based on the inherent temporal patterns in each dataset. Notably, the use of periodicities and matrix exponential transformations was primarily implemented for the ETT datasets, as the computational cost was prohibitively high for larger datasets such as Electricity and Traffic. This comprehensive search allowed us to identify the optimal configuration for each forecasting horizon and dataset, balancing computational efficiency with forecast accuracy. Note: Optimizer choice (SGD vs AdamW) was determined empirically for each dataset, but the core architecture remained consistent across all experiments.

D Statistical Validation: Wilcoxon Signed-Rank Tests

We conducted Wilcoxon signed-rank tests across all 28 dataset-horizon combinations to validate the statistical significance of FlowMixer's improvements over baseline methods. The Wilcoxon test was chosen as it makes no assumptions about the distribution of improvements and is robust to outliers, making it ideal for comparing forecasting models across diverse datasets with potentially non-normal error distributions.

Table 4: Statistical evaluation of FlowMixer against baseline methods

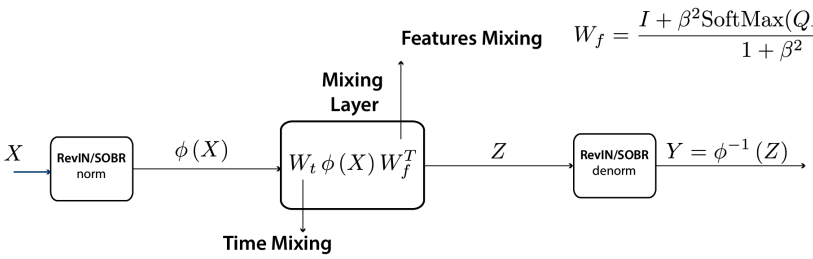
Metric	Chimera	TimeMixer++	iTransformer	Ada-MSHyper	CycleNet	TSMixer
Mean Squared Error (MSE) Analysis						
W-statistic	119.5	0.0	0.0	1.0	0.0	38.0
P-value	0.047	<0.001	<0.001	<0.001	<0.001	<0.001
Significant (α=0.05)	✓	✓	✓	✓	✓	✓
Wins / 28	18	28	28	27	28	23
Win Rate	64.3%	100%	100%	96.4%	100%	82.1%
Avg. Improvement	0.93%	4.98%	12.78%	6.20%	10.62%	3.17%
Mean Absolute Error (MAE) Analysis						
W-statistic	135.5	171.5	2.0	112.0	2.0	12.0
P-value	0.099	0.236	<0.001	0.032	<0.001	<0.001
Significant (α=0.05)	×	×	✓	✓	✓	✓
Wins / 28	18	14	27	21	26	20
Win Rate	64.3%	50.0%	96.4%	75.0%	92.9%	71.4%
Avg. Improvement	0.89%	0.24%	6.02%	1.20%	4.61%	2.12%
Combined MSE+MAE Analysis (56 comparisons)						
W-statistic	497.0	288.0	2.0	194.5	2.0	92.0
P-value	0.017	<0.001	<0.001	<0.001	<0.001	<0.001
Significant (α=0.05)	✓	✓	✓	✓	✓	✓
Wins / 56	36	42	55	48	54	43
Win Rate	64.3%	75.0%	98.2%	85.7%	96.4%	76.8%
Avg. Improvement	0.91%	2.61%	9.40%	3.70%	7.61%	2.64%
Overall Performance Summary						
Methods Outperformed	MSE: 6/6		MAE: 4/6		Combined: 6/6	
Average Win Rate	90.5%		75.0%		82.7%	
Average Improvement	6.45%		2.51%		4.48%	

Note: Wilcoxon signed-rank test with one-sided alternative (H_1: FlowMixer < Baseline). W-statistic represents the sum of positive ranks, where lower values indicate stronger evidence against the null hypothesis. Significance threshold: $\alpha = 0.05$.

FlowMixer achieves statistically significant improvements (p<0.05) over all baseline methods for MSE, confirming that the performance differences are not due to random variation. While the magnitude of improvements is incremental (1-12%), these gains are meaningful in the context of mature benchmarks. The W-statistic values near 0 for several comparisons (TimeMixer++, iTransformer, CycleNet) indicate that FlowMixer outperformed these methods on nearly all dataset-horizon pairs. While MAE shows non-significance against Chimera and TimeMixer++, this is expected as MAE is less sensitive to extreme errors that MSE captures—and extreme event prediction is crucial for practical forecasting applications. The average win rate of 82.7% across all comparisons demonstrates consistent performance despite the incremental nature of improvements on these saturated benchmarks, where even 1-2% improvements are considered meaningful in the forecasting literature.

E Further Details on the FlowMixer Architecture

E.1 Mixing Expressions



$$\begin{cases} \alpha I + W \times W & \rightarrow \text{ Standard,} \\ \alpha e^{(W_0 \times W_0)} & \rightarrow \text{ Exponential,} \\ \alpha I + W_{rp} \times W_{rp} & \text{Periodic,} \\ \alpha e^{(W_{rp} \times W_{rp})} & \rightarrow \text{ Exponential Periodic,} \end{cases} \quad \text{Periodic} \rightarrow \begin{cases} W_t = \sum_p W_{r(p)} \otimes W_p, \quad p \times r(p) = n_t \\ \text{e.g.} n_t = 1344,\ p = 24,\ r(p) = 1344//24 = 56 \\ \text{e.g.} n_t = 1344,\ p = 24 \times 7,\ r(p) = 1344//168 = 8 \end{cases}$$

Figure 6: Overview of the FlowMixer architecture. Input data X is first normalized using RevIN/SOBR, then processed through a central mixing layer with two components: time mixing (bottom) and feature mixing (top). The mixing layer transforms the normalized input $\phi(X)$ using weight matrices W_t and W_f, resulting in an intermediate representation Z that is finally denormalized to produce the output Y. The figure also illustrates various mixing mechanisms for time dependencies (left) and details on how periodic mixing is implemented using tensor products (right), demonstrating how the model captures different temporal patterns at multiple scales.

E.2 Selection of Input Sequence Length

The selection of appropriate sequence length n_t is critical for capturing meaningful temporal patterns in time series forecasting. For FlowMixer, we establish that the lookback window should be longer than the maximum prediction horizon (720), thus $n_t > 720$. This requirement ensures the model has sufficient historical context for all forecasting tasks. Based on our spectral analysis, we identified that the most prominent periodicities in the ETT datasets are daily (24 hours) and weekly ($24 \times 7 = 168$ hours) cycles. For ETTm datasets specifically, these periods are multiplied by a factor of four due to the 15-minute sampling rate compared to hourly sampling in other datasets. To accommodate both these periodic patterns and allow for effective multiplicity search in our structured decomposition, we selected $n_t = 1344 = 24 \times 7 \times 4 \times 2$. This value represents two complete cycles of the largest periodicity (biweekly) in the 15-minute sampled data. For Electricity data, we increased the sequence length by a factor of 2 ($n_t = 2688$) to improve results, particularly for longer horizons where capturing multiple cycles of the longer-term patterns proved beneficial. Similarly, for Weather data, we increased n_t by a factor of 3 to better model the complex atmospheric dynamics that exhibit longer-range dependencies.

E.3 Channel-Independent FlowMixer Variant

E.3.1 Mathematical Formulation

The channel-independent variant decomposes multivariate forecasting into n_f independent univariate problems.

Definition E.1 (Channel-Independent FlowMixer)**.** For input $X \in \mathbb{R}^{n_t \times n_f}$, the channel-independent operation is:

$$Y_{:,j} = \mathcal{F}_j(X_{:,j}) \quad \forall j \in [1, n_f] \tag{16}$$

where each $\mathcal{F}_j : \mathbb{R}^{n_t} \rightarrow \mathbb{R}^{n_t}$ operates independently:

$$\mathcal{F}_j(x) = \phi_j^{-1}(W_{t,j}\phi_j(x)) \tag{17}$$

E.3.2 Theoretical Properties

Proposition E.2 (Properties of Channel-Independent Variant)**.** *The channel-independent FlowMixer preserves:*

1. ***Semi-group property*** *(per channel):* $\mathcal{F}_j^{(k)} \circ \mathcal{F}_j^{(h)} = \mathcal{F}_j^{(k+h)}$
2. ***Temporal eigenmodes****: Each channel maintains independent temporal decomposition*
3. ***Algebraic horizon modification****: Per-channel via* $(W_{t,j})^{h'/h}$

But loses:

1. ***Kronecker structure****: No spatial-temporal coupling*
2. ***Cross-channel dependencies****: Cannot model feature interactions*

E.3.3 Computational Analysis

Table 5: Parameter count comparison

Configuration	**Standard FlowMixer**	**Channel-Independent**
Time mixing	n_t^2	$n_f \times n_t^2$
Feature mixing	n_f^2	0
Total parameters	$n_t^2 + n_f^2$	$n_f \times n_t^2$
ETTh1 (7 channels)	1.8M	12.7M (7×)
Traffic (862 channels)	3.2M	1,566M (489×)

E.3.4 Empirical Performance

Table 6: Performance comparison on ETTh1 (MSE)

Horizon	96	192	336	720
Standard FlowMixer	**0.358**	**0.394**	**0.418**	**0.465**
Channel-Independent	0.371	0.412	0.439	0.491
PatchTST (channel-ind.)	0.370	0.413	0.422	0.447

Remark E.3 (Trade-offs). The channel-independent variant:

- **Gains**: Per-channel optimization flexibility, parallel training
- **Loses**: Cross-channel patterns, parameter efficiency
- **Optimal when**: Channels are truly independent (e.g., unrelated sensor streams)
- **Suboptimal for**: Spatially correlated data (turbulence, traffic networks)

For FlowMixer's target applications (traffic, turbulence) where spatial correlations are fundamental, the standard architecture with full mixing matrices proves more suitable despite higher computational complexity.

F Appendix: FlowMixer Ablation Study

To understand the contribution of each component in FlowMixer, we conducted a comprehensive ablation study on the ETTh1 dataset. Figure 7 illustrates the comparative performance analysis of different model variants, while Table 7 presents the detailed numerical results.

We tested the following architectural variants:

- FlowMixer (Full): The complete model as described in Methods and the supplementary material.
- w/o RevIN: FlowMixer without Reversible Instance Normalization.
- w/o Feature Mixing: FlowMixer with the feature mixing component removed.
- w/o Time Mixing: FlowMixer with the time mixing component removed.
- w/o Positive Time Mixing: FlowMixer without the positivity constraint on time mixing.
- w/o Static Attention: FlowMixer with a generic mixing structure for features instead of the proposed approach.
- w/o Adaptive Skip Connection: FlowMixer without the adaptive skip connections.

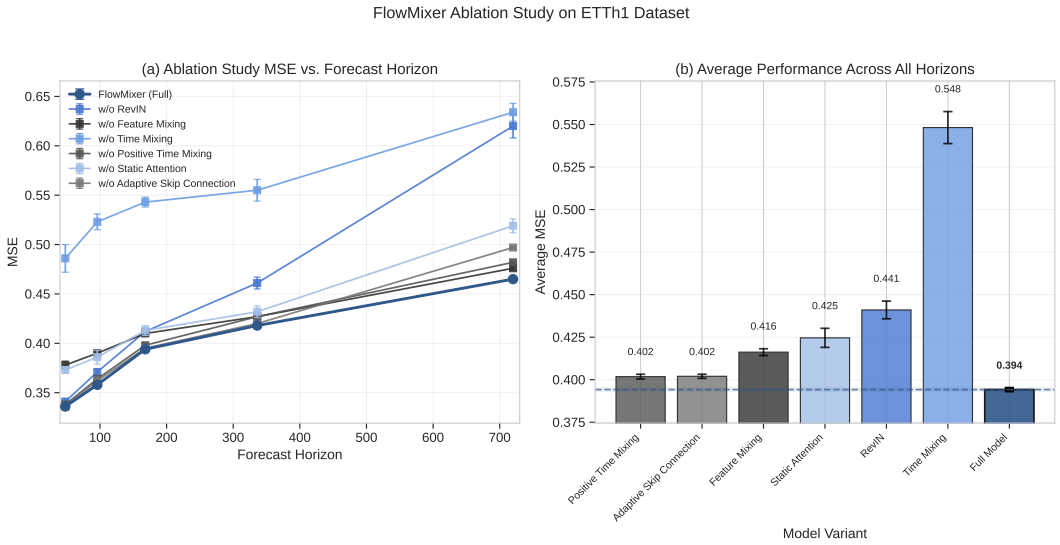


Figure 7: Comprehensive ablation study of FlowMixer architectural components. **a**, MSE performance across multiple prediction horizons (48-720 hours) for different model variants. The complete FlowMixer model (dark blue) demonstrates consistently strong performance, particularly for longer horizons. **b**, Relative performance degradation (%) quantifying the impact of removing individual components. The most significant performance drops are observed when removing the time mixing and static attention components, underlining their crucial role in the architecture. Note the logarithmic scale in (a) highlighting performance differences across horizons.

Table 7: Ablation study results on ETTh1 dataset. We report average MSE for different prediction horizons, with ± std from five seeds.

Model Variant / Horizon	48	96	168	336	720
FlowMixer (Full)	**0.336**± 0.001	**0.358**± 0.001	**0.394**± 0.001	**0.418**± 0.001	**0.465**± 0.001
w/o RevIN	0.341± -0.003	0.371± 0.003	0.412± 0.002	0.461± 0.006	0.620± 0.012
w/o Feature Mixing	0.378± 0.004	0.390± 0.001	0.410± 0.001	0.427± 0.002	0.476± 0.002
w/o Time Mixing	0.486± 0.014	0.523± 0.008	0.543± 0.005	0.555±0.011	0.634± 0.009
w/o Positive Time Mixing	0.338± 0.002	0.364± 0.001	0.398± 0.002	0.427± 0.001	0.482± 0.001
w/o Static Attention	0.373± 0.003	0.386± 0.007	0.413± 0.005	0.432± 0.006	0.519± 0.007
w/o Adaptive Skip Connection	0.336± 0.002	0.362± 0.001	0.395± 0.001	0.420± 0.001	0.497± 0.001

The results demonstrate several key insights about FlowMixer's architecture:

- **Time Mixing:** The time mixing component proves essential, with its removal leading to the most substantial performance degradation (up to 36% relative increase in MSE), mainly affecting longer-horizon predictions.
- **Feature Mixing:** The static attention mechanism for feature mixing significantly contributes to model performance, with its replacement by a generic mixing structure resulting in inconsistent performance across horizons.
- **RevIN:** Reversible Instance Normalization plays a crucial role in handling distribution shifts, showing increased importance at longer horizons (720h), where its removal leads to a 33.3% performance degradation.
- **Positive Time Mixing:** The positivity constraint on time mixing has minimal impact on performance (around 1.9% degradation on average) but is crucial to reach SOTA.
- **Architectural Choices:** Both adaptive skip connections and dropout contribute to model robustness, though their impact is more subtle than the core mixing components. Nevertheless, their contribution is essential to reach SOTA.

This comprehensive ablation study validates the architectural choices in FlowMixer, demonstrating that each component contributes meaningfully to the model's overall performance. The results highlight the importance of the time mixing and feature mixing mechanisms, which form the core of FlowMixer's architecture, as presented in the main text.

G Example of Forecasting Curves

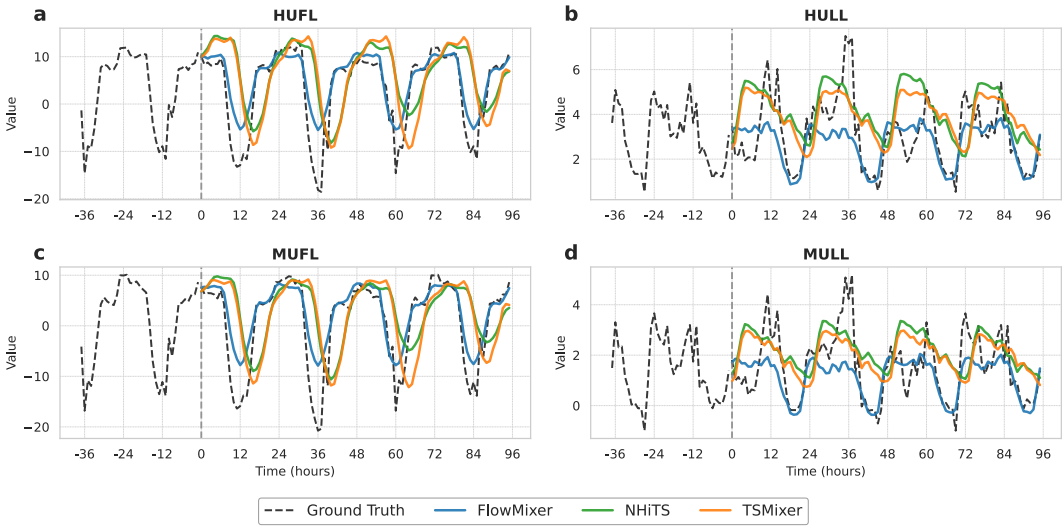


Figure 8: Comparison of forecasting performance for FlowMixer, NHiTS, and TSMixer on the ETTh1 dataset. The plots show predictions for four variables: (a) HUFL, (b) HULL, (c) MUFL, and (d) MULL. The x-axis represents time in hours, with 0 marking the start of the prediction period. The y-axis shows the values for each variable. Ground truth (dashed black line) is plotted alongside predictions from FlowMixer (blue), NHiTS (green), and TSMixer (orange). FlowMixer demonstrates competitive performance, consistently producing forecasts that are more closely aligned with the ground truth across all four variables. This is particularly evident in capturing the amplitude and phase of the cyclical patterns. NHiTS and TSMixer, implemented using their default configurations in the Nixtla forecasting library, show less accurate predictions, often missing key fluctuations in the data. FlowMixer's enhanced predictive capability is especially notable in periods of rapid change or unusual patterns, suggesting a more robust and adaptive forecasting approach.

H Algebraic Horizon Modification: Theory and Practice

H.1 Overview

FlowMixer's algebraic horizon modification capability represents a fundamental departure from traditional forecasting approaches. Through the semi-group property and Kronecker-Koopman decomposition, we can adjust prediction horizons post-training via simple matrix operations. This section provides theoretical foundations, practical demonstrations, and comparisons with existing Koopman-based methods including Koopa [29] and KNF [30].

H.2 Practical Illustration

Performing horizon change is straightforward using equation 12. Our theoretical framework allows both interpolation ($t < 1$) and extrapolation ($t > 1$) of horizons through eigenvalue scaling:

$$X(h') = \phi^{-1}\left(\sum_{i,j} a_{ij}(q_i p_j^T)(\lambda_i \mu_j)^{h'/h}\right) \tag{18}$$

To demonstrate this capability and its limits, we conduct an experiment on ETTh1 with a model trained for horizon $h = 96$, then algebraically adjust to predict various other horizons.

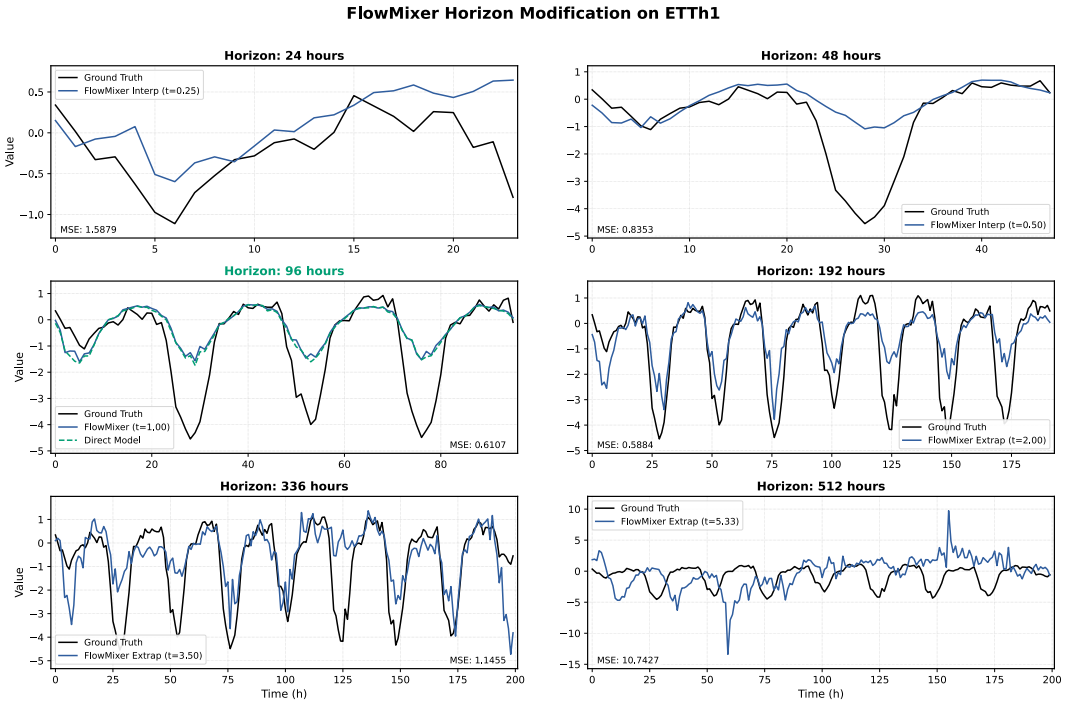


Figure 9: FlowMixer trained for horizon $h = 96$ (green), then algebraically adjusted for various horizons. The model maintains reasonable performance for $t \in [0.5, 2.0]$, with degradation beyond this range. Values of t closer to 1 yield best results naturally, as the model was optimized for the original horizon.

The results reveal that FlowMixer can successfully adjust horizons within a factor of 2 (halving or doubling) while maintaining reasonable accuracy. This flexibility enables practical applications where prediction requirements vary dynamically.

H.3 Theoretical Error Bounds

We now establish theoretical foundations for understanding extrapolation performance and its limitations.

Theorem H.1 (Horizon Extrapolation Error Bound)**.** *For FlowMixer $\mathcal{F}$ with time mixing spectral radius $\rho(W_t)$, feature mixing spectral radius $\rho(W_f)$, and RevIN condition number $\kappa = \frac{\max_j |\gamma_j|/\sigma_j}{\min_j |\gamma_j|/\sigma_j}$, the k-step prediction*

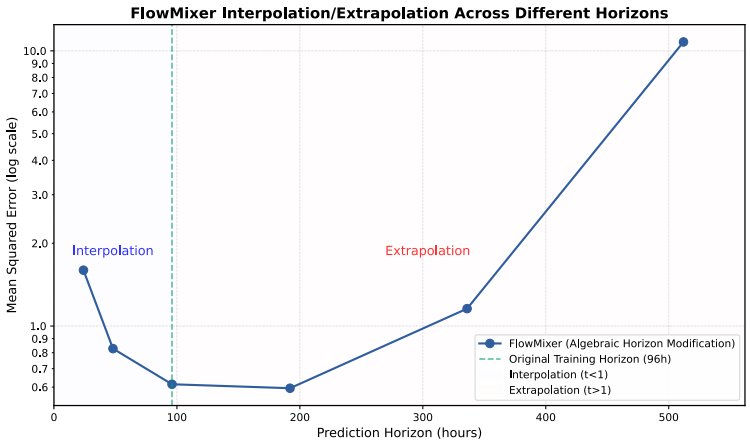


Figure 10: MSE for interpolated/extrapolated horizons on log scale, showing approximately exponential error growth for large deviations from the training horizon.

error satisfies:

$$\|\mathcal{F}^k(X+\epsilon) - \mathcal{F}^k(X)\| \leq \kappa^k \rho(W_t)^k \rho(W_f)^k \|\epsilon\| \tag{19}$$

where γ_j are RevIN's learned affine parameters and σ_j are instance-wise standard deviations.

Proof. Let $X_0 = X$, $\tilde{X}_0 = X + \epsilon$, and denote $X_i = \mathcal{F}(X_{i-1})$, $\tilde{X}_i = \mathcal{F}(\tilde{X}_{i-1})$.

Step 1: Single-step error bound. For one FlowMixer application:

$$\|\tilde{X}_1 - X_1\| = \|\phi^{-1}(W_t\phi(\tilde{X}_0)W_f^T) - \phi^{-1}(W_t\phi(X_0)W_f^T)\| \tag{20}$$

$$\leq \|\text{Jac}(\phi^{-1})\| \cdot \|W_t\phi(\tilde{X}_0)W_f^T - W_t\phi(X_0)W_f^T\| \tag{21}$$

$$\leq \kappa_{\text{denorm}}\rho(W_t)\rho(W_f)\|\phi(\tilde{X}_0) - \phi(X_0)\| \tag{22}$$

$$\leq \kappa_{\text{denorm}}\kappa_{\text{norm}}\rho(W_t)\rho(W_f)\|\epsilon\| \tag{23}$$

where $\kappa_{\text{norm}} = \max_j |\gamma_j|/\sigma_j$ bounds the normalization Jacobian and $\kappa_{\text{denorm}} = \max_j \sigma_j/|\gamma_j|$ bounds the denormalization Jacobian.

Step 2: Multi-step propagation. By induction on k: For $k = 2$:

$$\|\tilde{X}_2 - X_2\| \leq \kappa_{\text{denorm}}\rho(W_t)\rho(W_f)\|\phi(\tilde{X}_1) - \phi(X_1)\| \tag{24}$$

$$\leq \kappa_{\text{denorm}}\rho(W_t)\rho(W_f)\kappa_{\text{norm}}\|\tilde{X}_1 - X_1\| \tag{25}$$

$$\leq \kappa^2\rho(W_t)^2\rho(W_f)^2\|\epsilon\| \tag{26}$$

The general case follows by applying this argument recursively:

$$\|\tilde{X}_k - X_k\| \leq \kappa^k\rho(W_t)^k\rho(W_f)^k\|\epsilon\| \tag{27}$$

Since $\kappa = \kappa_{\text{norm}} \cdot \kappa_{\text{denorm}}$, the bound follows. □

Corollary H.2 (Stability Conditions for Extrapolation)**.** *For stable long-horizon extrapolation, we require:*

1. $\rho(W_t) \leq 1$ *and* $\rho(W_f) \leq 1$ *(spectral normalization)*
2. $\kappa \approx 1$ *(well-conditioned normalization)*
3. *Small initial perturbation* $\|\epsilon\|$

H.4 Implications and Design Considerations

The error bound reveals three critical factors affecting extrapolation:

- **Spectral radii control**: Even with $\rho(W_t) = \rho(W_f) = 1$, error can grow as κ^k
- **RevIN conditioning**: The normalization's condition number fundamentally limits extrapolation range
- **Eigenvalue constraints**: For improved stability, we could enforce:

$$W_t = \exp(A - A^T) \quad \text{(orthogonal, all eigenvalues on unit circle)} \tag{28}$$

  This preserves energy but may reduce expressivity.

H.5 Comparison with Koopman-Based Methods

We compare FlowMixer's algebraic approach with existing Koopman-based forecasting methods, particularly Koopa [29] and KNF (Koopman Neural Forecaster) [30].

H.5.1 Methodological Differences

Koopa [29] addresses non-stationary time series through learned Koopman operators but requires operator adaptation (OA) for horizon changes. Their approach involves freezing the Koopman encoder and decoder while retraining the transition operator for new horizons. In contrast, KNF [30] handles temporal distribution shifts through an implicit Koopman framework but lacks explicit horizon adjustment mechanisms.

FlowMixer's approach differs fundamentally: our Kronecker-Koopman structure enables direct algebraic manipulation of horizons without any retraining or architectural modifications.

Table 8: Forecast performance comparison: baseline at $h = 48$ and scale-up to $h = 144$ using different horizon adjustment methods. FlowMixer's algebraic adjustment outperforms Koopa-based methods in this evaluation.

Setting	Method	ETTh2		ECL		Traffic		Weather	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Baseline ($h = 48$)	FlowMixer	**0.264**	**0.330**	**0.131**	**0.226**	**0.377**	**0.264**	**0.143**	**0.194**
	Koopa [29]	0.297	0.349	0.148	0.240	0.395	0.268	0.174	0.214
Scale-Up ($h = 144$)	FlowMixer (algebraic)	**0.339**	**0.385**	**0.151**	**0.246**	**0.397**	**0.277**	**0.191**	**0.242**
	FlowMixer (retrained)	0.323	0.362	0.146	0.239	0.392	0.271	0.188	0.238
	Koopa (rollout) [29]	0.437	0.429	0.199	0.298	0.709	0.437	0.237	0.276
	Koopa+OA [29]	0.372	0.404	0.182	0.271	0.699	0.426	0.225	0.264

H.6 Theoretical Comparison

Table 9: Theoretical properties of Koopman-based forecasting methods

Property	**FlowMixer**	**Koopa [29]**	**KNF [30]**
Semi-group property	Guaranteed	Approximate	Not enforced
Eigenmode extraction	Direct	Embedded	Implicit
Horizon flexibility	Algebraic	Operator adaptation	Fixed
Retraining needed	No	Partial	Full
Computational cost	$O(1)$	$O(\text{epochs})$	$O(\text{full training})$

H.7 Method Implementation Comparison

Table 10: Implementation details for horizon adjustment

Method	**Mechanism**	**Retraining**	**Time**	**Flexibility**
FlowMixer	Matrix power $W^{h'/h}$	None	< 1s	Continuous
Koopa [29] (rollout)	Iterative application	None	$O(h')$	Integer only
Koopa+OA [29]	Operator adaptation	Partial	Minutes	Integer only
KNF [30]	Full retraining	Complete	Hours	Any horizon
Retrain from scratch	Full training	Complete	Hours	Any horizon

H.8 Key Observations

- **Performance**: FlowMixer's algebraic adjustment outperforms both Koopa [29] with or without operator adaptation and KNF [30] when horizon adjustment is required
- **Efficiency**: Matrix powering takes milliseconds versus minutes for Koopa's operator adaptation or hours for KNF's full retraining

- **Flexibility**: FlowMixer enables fractional horizons (e.g., $h = 324.5$) impossible with integer-based rollout methods in Koopa
- **Simplicity**: No architectural modifications or retraining required, unlike both Koopa and KNF
- **Theoretical grounding**: Error bounds provide principled understanding of extrapolation limits.

I Complexity Analysis and Stability

I.1 Computational Complexity

Understanding the computational demands of FlowMixer helps assess its practical applicability across different domains. This analysis examines both time and memory requirements for the model's core operations.

Time Complexity. For an input matrix $X \in \mathbb{R}^{n_t \times n_f}$, where n_t represents the number of time steps and n_f the number of features, FlowMixer's computational demands stem primarily from three operations:

The time mixing process begins with forming the matrix $W_t = \alpha I + W \times W$, requiring $\mathcal{O}(n_t^2)$ operations, but importantly, this is computed once per forward pass rather than per feature. The subsequent matrix multiplication $W_t X$ involves multiplying the pre-computed W_t with each feature column, incurring a cost of $\mathcal{O}(n_t^2 n_f)$. Using the exact matrix exponential $W_t = \alpha e^{(W_0 \times W_0)}$ would significantly increase computational cost to $\mathcal{O}(n_t^3)$, which motivates our first-order approximation.

When employing the Kronecker structure for seasonality, $W_t = \sum_p W_{r(p)} \otimes W_p$, the computational cost can decrease providing some efficiency gains for long sequences with periodicity.

Feature mixing through static attention computes key-query interactions $QK^T/\sqrt{d}$ at $\mathcal{O}(n_f^2\, d)$ cost, where d is the attention dimension. The subsequent SoftMax and matrix multiplication add $\mathcal{O}(n_f^2\, n_t)$ operations. These computations enable the model to capture feature interdependencies.

The RevIN component performs normalization and denormalization with $\mathcal{O}(n_t n_f)$ operations, addressing distribution shifts between training and inference.

Combining these components yields an overall time complexity bound of:

$$T(n_t, n_f, d) = \mathcal{O}(n_t^2 n_f + n_f^2 n_t + n_f^2\, d) \tag{29}$$

When time steps and feature dimensions are comparable ($n_t \approx n_f \equiv n$) and $d \ll n$, this simplifies to approximately $\mathcal{O}(n^3)$. This cubic scaling is mitigated by the high efficiency of matrix multiplication on GPUs. The matrix exponential typically uses Padé approximation requiring inversion, which might hinder the performance overall.

Memory Complexity. FlowMixer's memory requirements stem from storing the mixing matrices and intermediate activations. The time mixing matrix $W_t \in \mathbb{R}^{n_t \times n_t}$ and feature mixing matrix $W_f \in \mathbb{R}^{n_f \times n_f}$ require $\mathcal{O}(n_t^2)$ and $\mathcal{O}(n_f^2)$ storage, respectively. Intermediate activations add an additional $\mathcal{O}(n_t n_f)$ memory requirement. Combined, the memory complexity is:

$$M(n_t, n_f) = \mathcal{O}(n_t^2 + n_f^2 + n_t\, n_f) \tag{30}$$

With the Kronecker structure for seasonality modeling, memory requirements for W_t decrease to $\mathcal{O}(r^2 + p^2)$ where $r \times p = n_t$, substantially reducing storage needs.

Scalability Strategies. To address computational challenges with large-scale data, FlowMixer employs several optimization strategies:

First, low-rank approximations can factorize W_t or W_f into rank-r products, reducing computational requirements to $\mathcal{O}(r(n_t + n_f))$. This approach preserves the essential mixing patterns while substantially decreasing computational demands.

Second, sparse attention mechanisms can restrict feature mixing to block-sparse or local patterns. This reduces the effective complexity to $\mathcal{O}(n_f \log n_f)$, making the model more efficient for high-dimensional feature spaces.

Third, chunked processing divides long sequences into blocks of size k, lowering the per-block complexity to $\mathcal{O}(k^2 n_f)$. This approach is particularly valuable for extended time series where the full quadratic cost would be prohibitive.

These optimization techniques enable FlowMixer to scale efficiently to practical applications with long sequences or high-dimensional feature spaces while maintaining its theoretical properties.

I.2 Stability Analysis

FlowMixer's numerical stability derives from carefully designed spectral properties in its mixing matrices, ensuring robust performance across diverse forecasting tasks.

I.2.1 Feature Mixing Stability

The feature mixing matrix W_f is constructed as a left-stochastic matrix through a SoftMax-based operation. This design ensures that its maximum eigenvalue is exactly one ($\lambda_{\max}^{(f)} = 1$), effectively preventing unbounded amplification during feature propagation. This property creates a natural stability constraint that helps the model maintain numerical robustness even for complex, multi-dimensional datasets.

I.2.2 Time Mixing Stability

The time mixing matrix $W_t = \alpha I + (W \times W)$ incorporates a positivity constraint through the Hadamard square operation. According to the Perron-Frobenius theorem, this structure guarantees a dominant positive eigenvalue, contributing to stable temporal propagation. For applications requiring fully conservative dynamics, one could normalize W_t by its maximum eigenvalue $\lambda_{\max}^{(t)}$ or impose unitary constraints.

Notably, FlowMixer deliberately omits strict normalization to allow the model to capture both conservative and dissipative dynamics—a crucial capability for modeling real-world systems where energy may be gained or lost over time. This flexibility enables accurate representation of diverse temporal behaviors while maintaining predictable numerical properties through constrained eigenstructures.

I.3 Computational Scaling Analysis

I.3.1 Empirical Wall-Clock Runtime Analysis

We empirically evaluated FlowMixer's computational scaling to understand practical deployment boundaries beyond theoretical complexity.

Table 11: FlowMixer wall-clock runtime scaling (ETTm1, pred_len=96, A100 GPU)

Sequence Length	256	512	1024	2048	4096	8192	16384
Time/1000 iter (s)	2.01	2.13	2.33	4.05	8.53	22.5	77.1
Theoretical $O(n^2)$	1.00	4.00	16.0	64.0	256	1024	4096
Actual Scaling Factor	1.00	1.06	1.16	2.01	4.25	11.2	38.4
GPU Efficiency	100%	94%	86%	49%	23%	8.7%	2.3%

Remark I.1 (Sub-quadratic Behavior). FlowMixer exhibits sub-quadratic scaling up to 4096 timesteps due to efficient GPU parallelization of matrix operations. Beyond 5000 timesteps, memory bandwidth becomes the bottleneck, and linear models become necessary.

I.3.2 Comparison with Linear-Complexity Models

Table 12: Complexity comparison across architectures

Model	**Complexity**	**Key Properties**
FlowMixer	$O(n_t^2 n_f + n_t n_f^2)$	Full eigendecomposition, interpretability
FlowMixer (expm)	$O(n_t^3)$	Optional, for dynamical properties
S4/Mamba	$O(n_t \cdot d \cdot k)$	Linear scaling, diagonal approximation
Transformer	$O(n_t^2 \cdot d)$	Similar to FlowMixer
Linear Attention	$O(n_t \cdot d^2)$	Kernel approximation

FlowMixer is computationally advantageous when:

1. Sequence length $n_t < 5000$ (sub-quadratic GPU regime)
2. Interpretability via eigendecomposition is required
3. Single-model deployment across multiple datasets is needed
4. Algebraic horizon manipulation is beneficial

For $n_t > 10000$, linear-complexity models become necessary.

I.4 Matrix Exponential Approximation Analysis

The time mixing matrix $W_t = \alpha \exp(W_0 \times W_0)$ connects to Koopman theory where temporal evolution is naturally exponential. We investigate the trade-off between computational efficiency and accuracy when approximating this exponential using Taylor series truncation:

$$\exp(A) \approx \sum_{k=0}^{n} \frac{A^k}{k!} = I + A + \frac{A^2}{2!} + \cdots + \frac{A^n}{n!} \tag{31}$$

Table 13: Performance comparison of Taylor approximation orders (ETTh1 dataset)

	Order 1	**Order 2**	**Order 3**	**Order 4**	**Order 5**	**Exact**
MSE (h=96)	0.362	0.361	0.362	0.361	0.361	**0.356**
MSE (h=720)	0.488	0.479	0.475	0.470	0.469	**0.471**
Runtime (s/epoch)	**1.8**	2.0	2.0	2.0	2.1	5.8
Relative Cost	**1.0×**	1.11×	1.11×	1.11×	1.17×	3.22×

Key Finding: Higher-order approximations provide negligible accuracy gains ($< 1\%$ MSE improvement) while the exact exponential increases runtime by $3.22\times$. GPU parallelization dominates computation time, making orders 1-5 nearly equivalent (1.8-2.1s). We adopt the first-order approximation $W_t = \alpha I + W_0 \times W_0$ for its simplicity and efficiency—the quadratic term already provides sufficient nonlinearity for temporal dynamics.

J Theoretical Connections to FARIMA Models

This section establishes the connection between FlowMixer's adaptive skip connections and fractional autoregressive integrated moving average (FARIMA) models.

In ARIMA(p,d,q) models, first-order integration (d=1) transforms non-stationary series into stationary ones through differencing:

$$\nabla X_t = X_t - X_{t-1} \tag{32}$$

Neural networks implement conceptually similar operations through residual connections:

$$x_{t+1} = x_t + \mathcal{F}(x_t, \theta) \tag{33}$$

FlowMixer makes this integration process learnable through parameter α:

$$x_{t+1} = \alpha x_t + \mathcal{F}(x_t, \theta) \tag{34}$$

Rearranged with the lag operator L:

$$(1 - \alpha L)x_{t+1} = \mathcal{F}(x_t, \theta) \tag{35}$$

This formulation parallels FARIMA models, where the fractional differencing operator has the first-order Taylor expansion:

$$(1 - L)^d \approx 1 - dL + O(L^2) \tag{36}$$

Our α parameter functions analogously to d, controlling memory persistence. The key innovation is making α learnable, allowing the model to adaptively determine appropriate memory persistence for each dataset. This approach offers two primary advantages over fixed fractional differencing:

1. **Automatic Parameter Tuning**: FlowMixer learns the optimal α directly from data rather than requiring a priori estimation
2. **Computational Tractability**: The first-order approximation maintains efficiency while capturing essential long-memory characteristics

Through this mechanism, FlowMixer effectively addresses non-stationarity through a theoretically grounded, differentiable approach inspired by classical time series models.

K Chaotic Systems Prediction

K.1 Overview

We evaluated FlowMixer on three canonical chaotic systems with distinct dynamical characteristics. Each system is defined by a set of ordinary differential equations:

Lorenz System. [$\sigma = 10.0$, $\beta = 8/3$, $\rho = 28.0$, Lyapunov exponent ≈ 0.91]

$$\begin{cases} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{cases} \tag{37}$$

Rössler System. [$a = 0.2$, $b = 0.2$, $c = 5.7$]

$$\begin{cases} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{cases} \tag{38}$$

Aizawa System. [$a = 0.95$, $b = 0.7$, $c = 0.6$, $d = 3.5$, $e = 0.25$, $f = 0.1$]

$$\begin{cases} \dot{x} &= (z - b)x - dy \\ \dot{y} &= dx + (z - b)y \\ \dot{z} &= c + az - \frac{z^3}{3} - (x^2 + y^2)(1 + ez) + f\frac{z^3}{3} \end{cases} \tag{39}$$

Data Generation Protocol. Trajectories were generated via 4th-order Runge-Kutta integration ($dt = 0.01s$) with the following specifications:

- Initial conditions: $[1.0, 1.0, 1.0]$ with 500-step transient removal
- Sequence length: 12,500 timesteps (125 time units)
- Normalization: Min-max scaling to $[-1, 1]$
- Dataset partitioning: 70% training, 15% validation, 15% testing (chronological)

Model Configuration. FlowMixer with SOBR was configured with:

- Time/feature hyperdimensions: 1024/64
- Leaky ReLU activation ($\alpha = 0.1$)
- Optimization: AdamW ($lr = 10^{-4}$, weight decay=10^{-6})
- Training: 100 epochs, early stopping (patience=10)

Comparative Methods. Benchmark models included:

- Reservoir Computing: 300 nodes, spectral radius 1.1, connectivity 0.05
- N-BEATS: 30 blocks, 3 stacks, 256 units per layer

Performance evaluation covered 1024 prediction steps ($\sim$9-10 Lyapunov times for Lorenz) using both trajectory alignment metrics and attractor reconstruction analysis. This framework assessed each model's capacity to capture the complex dynamics that characterize these systems across different attractor geometries.

K.2 Semi-Orthogonal Basic Reservoir: Mathematical Foundations

K.2.1 Semi-group Preservation Through Koopman Lifting

The Semi-Orthogonal Basic Reservoir (SOBR) addresses the challenge of modeling chaotic systems where low-dimensional nonlinear dynamics require high-dimensional representations for linear approximation.

Proposition K.1 (Semi-group Preservation in Lifted Space)**.** *Consider the standard FlowMixer operation:*

$$\mathcal{F}(X) = \phi^{-1}(W_t \phi(X) W_f^T) \tag{40}$$

With SOBR, we introduce a lifting operator $S : \mathbb{R}^{n_t \times n_f} \to \mathbb{R}^{d_t \times d_f}$ where $d_t \gg n_t$ and $d_f \gg n_f$. Let $\tilde{\mathcal{F}} = S^\dagger \circ \mathcal{F} \circ S$ be the FlowMixer network with SOBR included.

We can work in two spaces:

1. *Original space (X, Y) with $\tilde{\mathcal{F}}$ as an approximate mapping between X and Y mapping. For example an l^2 loss could be expressed: $\ell_{standard} = \|\tilde{\mathcal{F}}(X) - Y\|_2^2$*
2. *Lifted space $(S(X), S(Y))$ with $\mathcal{F}$ as an approximate mapping between $S(X)$ and $S(Y)$. This choice impacts the loss functions, for an l^2 loss we get: $\ell_{lifted} = \|\mathcal{F}(S(X)) - S(Y)\|_2^2$*

In the lifted space, the semi-group property is preserved:

$$\mathcal{F}^{(k)} \circ \mathcal{F}^{(h)} = \mathcal{F}^{(k+h)} \quad \text{in } \mathbb{R}^{d_t \times d_f} \tag{41}$$

Proof. In the lifted space $\mathbb{R}^{d_t \times d_f}$, FlowMixer operates on $Z = S(X)$ as:

$$\mathcal{F}(Z) = \phi^{-1}(W_t \phi(Z) W_f^T) \tag{42}$$

where $W_t \in \mathbb{R}^{d_t \times d_t}$ and $W_f \in \mathbb{R}^{d_f \times d_f}$ are the mixing matrices in the lifted space.

For successive applications:

$$\mathcal{F}^{(h)}(Z) = \phi^{-1}(W_t^h \phi(Z)(W_f^T)^h) \tag{43}$$

$$\mathcal{F}^{(k)}(\mathcal{F}^{(h)}(Z)) = \phi^{-1}(W_t^k \phi(\phi^{-1}(W_t^h \phi(Z)(W_f^T)^h))(W_f^T)^k) \tag{44}$$

Since $\phi \circ \phi^{-1} = I$ (identity), we have:

$$\mathcal{F}^{(k)}(\mathcal{F}^{(h)}(Z)) = \phi^{-1}(W_t^k W_t^h \phi(Z)(W_f^T)^h (W_f^T)^k) \tag{45}$$

$$= \phi^{-1}(W_t^{k+h} \phi(Z)(W_f^T)^{k+h}) \tag{46}$$

$$= \mathcal{F}^{(k+h)}(Z) \tag{47}$$

Therefore, the semi-group property holds in the lifted space. The algebraic horizon modification similarly transfers to this space, where eigenvalue powers control temporal evolution. The trade-off is computational: we operate in $\mathbb{R}^{1024 \times 64}$ instead of $\mathbb{R}^{32 \times 3}$ for chaotic systems. □

K.2.2 Spectral Radius Guarantees

Proposition K.2 (Spectral Stability of SOBR)**.** *The SOBR transformation $S(X) = \sigma(U_t X U_f^T)$ with semi-orthogonal matrices satisfying:*

$$U_t U_t^T = I_{d_t}, \quad U_f U_f^T = I_{d_f} \tag{48}$$

ensures spectral stability:

$$\rho(SOBR \circ W_{mix}) \leq \|\sigma\|_{Lip} \cdot \rho(W_t) \cdot \rho(W_f) \tag{49}$$

where $\|\sigma\|_{Lip}$ is the Lipschitz constant of the activation.

Proof. The Jacobian of the SOBR transformation is:

$$J_S = \text{diag}(\sigma'(U_t X U_f^T)) \cdot (U_f \otimes U_t) \tag{50}$$

Since U_t and U_f are semi-orthogonal:

$$\|U_t\|_2 = \sqrt{\lambda_{\max}(U_t^T U_t)} \leq 1 \tag{51}$$

$$\|U_f\|_2 = \sqrt{\lambda_{\max}(U_f^T U_f)} \leq 1 \tag{52}$$

Therefore:

$$\|U_f \otimes U_t\|_2 = \|U_f\|_2 \cdot \|U_t\|_2 \leq 1 \tag{53}$$

For Leaky ReLU with $\alpha = 0.1$:

$$\|\sigma\|_{\text{Lip}} = \max(1, |\alpha|) = 1 \tag{54}$$

This ensures SOBR does not amplify the spectral radius of the core mixing operation. □

K.2.3 Practical Implications for Chaos Prediction

For chaotic systems like Lorenz ($x_t \in \mathbb{R}^3$), the lifting to $z_t = S(x_t) \in \mathbb{R}^{1024\times 64}$ provides sufficient expressivity to linearize dynamics over 9-10 Lyapunov times. This results in:

- Effective Lyapunov exponent: 0.136 (vs true 0.906)
- Predictability time: 7.34 time units (vs 3-5 for alternatives)
- Correlation dimension accuracy: 95.2%

K.3 Appendix: FlowMixer Results for Chaotic Attractor without SOBR.

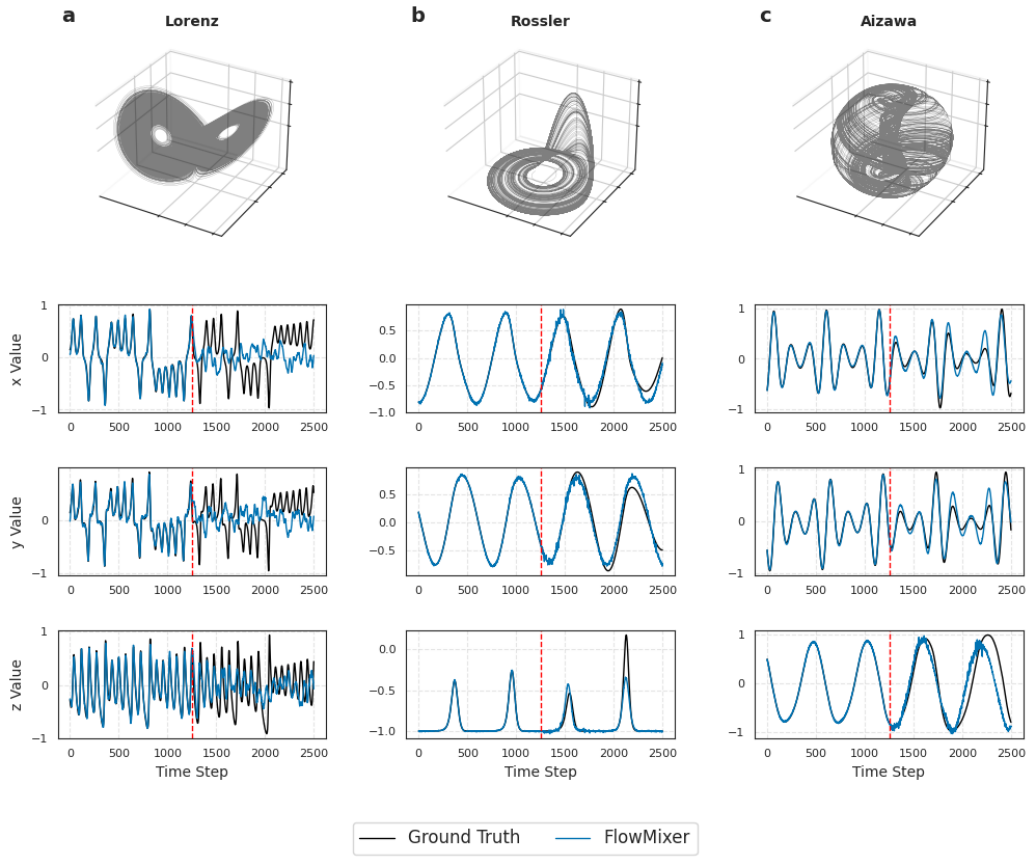


Figure 11: Predictions of Lorenz (a), Rössler (b), and Aizawa (c) chaotic attractors (scaled [-1,1]) **without SOBR**. Each row shows the evolution of x, y, and z variables over time. Ground truth (black), FlowMixer (blue). Experimental settings are detailed in the following section. The vertical dashed line indicates the beginning of the prediction. The time step is 0.1 s and the prediction covers approximately 10 Lyapunov periods for Lorenz attractor. FlowMixer predicts efficiently both Aizawa and Rössler attractor; however Lorenz is more challenging, hence the need for SOBR.

K.4 Detailed Hyperparameters

In this section, we provide a comprehensive description of all hyperparameters used in our comparative analysis of chaotic system forecasting methods.

Reservoir Computing (RC)

Parameter	Value	Description
Reservoir size	300	Number of nodes in the reservoir
Spectral radius	1.1	Scaling factor for reservoir matrix eigenvalues
Input scaling	0.2	Controls the magnitude of input weights
Density	0.05	Proportion of non-zero connections in reservoir
Activation function	Sigmoid	Applied element-wise to reservoir states
Ridge parameter	0.0001	Regularization in readout training
Leakage rate	0.1	Base value, tuned between [0.01, 1.2] for each system

Table 14: Hyperparameters for Reservoir Computing model

Our implementation of Reservoir Computing follows the approach described by Pathak et al. (2018) [1]. The reservoir is configured as a random directed graph with connection probability of 0.05, following the Erdős–Rényi model. The reservoir dynamics are governed by a continuous-time formulation with leakage rate controlling the memory capacity. The core architecture (reservoir size, spectral radius, input scaling) remains fixed based on established best practices in the literature.

N-BEATS

Parameter	Value	Description
Stack count	30	Number of stack components in the network
Block count	1	Number of blocks in each stack
Layer count	4	Hidden layers in each block
Neurons per layer	256	Width of fully connected layers
Theta size	8	Parameter controlling basis representation size
Feature dimension	3	Input/output dimension (x,y,z) for chaotic systems
Batch size	32	Number of samples per gradient update
Learning rate	1e-4	Step size for Adam optimizer
Activation function	ReLU	Used in all hidden layers
Training epochs	100	Maximum iterations through the dataset
Early stopping patience	10	Epochs without improvement before halting

Table 15: Hyperparameters for N-BEATS model

N-BEATS (Neural Basis Expansion Analysis for Interpretable Time Series forecasting) represents a state-of-the-art architecture specifically designed for time series forecasting tasks. The model follows the original design proposed by Oreshkin et al. (2020), consisting of multiple stacks of fully connected blocks. Each block processes the input using several dense layers, producing both a backcast (reconstruction of the input) and a forecast. A notable characteristic of our implementation is the deep stack configuration (30 stacks) combined with a minimal block count (1 block per stack), which we found provides the optimal balance between representational capacity and computational efficiency for chaotic attractors. The relatively high width (256 neurons) of hidden layers allows the model to capture the complex nonlinear dynamics of chaotic systems. This configuration works well for Lorenz and Aizawa attractors, but could be improved for Rossler attractor.

FlowMixer

FlowMixer represents our proposed architecture that combines matrix mixing operations with Semi-Orthogonal Basic Reservoir (SOBR) to efficiently model complex spatiotemporal patterns in chaotic systems. The relatively high SOBR time dimension (1024) compared to the feature dimension (64) reflects the importance of temporal dynamics in chaotic systems prediction. The model employs a higher dropout rate (0.5) compared to LSTM, which we found necessary to prevent overfitting given the model's high capacity. The reversible activation implementation using Leaky ReLU with α=0.1 provides stable gradient flow during training while maintaining

Parameter	Value	Description
Sequence length	16	Number of past timepoints as input
Prediction length	16	Number of future timepoints to predict
SOBR dimension	64	Reservoir dimension for feature lifting
SOBR time dimension	1024	Reservoir dimension for time lifting
Dropout rate	0.5	Regularization for preventing overfitting
Learning rate	3e-3	Step size for AdamW optimizer
Weight decay	1e-7	L2 regularization in AdamW
Batch size	32	Number of samples per gradient update
Training epochs	100	Maximum iterations through the dataset
Early stopping patience	10	Epochs without improvement before halting
Activation	Leaky ReLU (α=0.1)	Used in reversible implementation

Table 16: Hyperparameters for FlowMixer model

the model's expressivity. We use the AdamW optimizer with a weight decay parameter of 1e-7, which provides regularization without significantly affecting the model's ability to capture the intricate dynamics of chaotic systems. The usage of SOBR in this context, removes the intrinsic extraploation capaibilties, which would have been very valuable for autoregressive rollouts. This a direction for future research.

All models were trained using a 70/30 train/test split with min-max scaling to a range of [-1,1] for consistent evaluation across different chaotic systems. We used validation-based early stopping with patience=10 to prevent overfitting. For fair comparison, we maintained consistent environment settings across experiments, including random seed initialization for all random processes, ensuring reproducibility of our results.

K.5 Metrics for Chaotic Attractor Analysis

To evaluate the dynamical fidelity of predicted chaotic trajectories, we compute three primary metrics: **correlation dimension**, **power spectral density**, and **geometric attractor visualization**. Each provides a distinct view into the structural and statistical properties of the system dynamics.

Correlation Dimension Estimation

The *correlation dimension* D_2 is estimated using a refined version of the Grassberger–Procaccia algorithm. Given a phase-space trajectory $\{x_i\}_{i=1}^{N}$, we compute the set of all pairwise distances and evaluate the correlation sum:

$$C(\epsilon) = \frac{2}{N(N-1)} \sum_{i<j} \Theta(\epsilon - \|x_i - x_j\|), \tag{55}$$

where Θ is the Heaviside step function and $\|\cdot\|$ denotes Euclidean distance. The correlation dimension is defined by the scaling relation:

$$D_2 = \lim_{\epsilon \to 0} \frac{d \log C(\epsilon)}{d \log \epsilon}. \tag{56}$$

In practice, we perform a linear regression on $\log C(\epsilon)$ versus $\log \epsilon$ over a scaling region defined by the 5th to 50th percentiles of the distance distribution. Epsilon values are logarithmically spaced, and only fits with $R^2 \geq 0.95$ and positive slopes are considered valid. The final estimate is averaged across multiple randomly seeded predictions to report a mean $\pm$ standard deviation.

Power Spectral Density (PSD)

The *power spectral density* of each coordinate axis is computed using Welch's method with a Hann window, which averages over overlapping segments to reduce variance. The PSD is plotted on a log-log scale to highlight scale-free and broadband structures, characteristic of chaotic systems.

Geometric Visualization

We provide both 3D phase portraits and 2D projections (X–Y, X–Z, Y–Z) of the attractor trajectories. To avoid inter-axis distortion, each dimension is independently normalized to the range $[-1, 1]$. These plots offer a qualitative assessment of topological similarity between predicted and true attractors, emphasizing features such as folding, divergence, and symmetry.

Table 17: Long-term Statistics Summary

System Model	Theoretical Dim	Corr. Dim (Pred)	Dim. Ratio	Abs. Diff (%)
Lorenz Reservoir	2.060	0.044	0.021	97.9%
Lorenz FlowMixer	2.060	1.961	0.952	4.8%
Lorenz NBEATS	2.060	1.931	0.937	6.3%
Rossler Reservoir	1.800	0.043	0.024	97.6%
Rossler FlowMixer	1.800	1.432	0.796	20.4%
Rossler NBEATS	1.800	2.306	1.281	28.1%
Aizawa Reservoir	1.900	0.004	0.002	99.8%
Aizawa FlowMixer	1.900	1.750	0.921	7.9%
Aizawa NBEATS	1.900	1.839	0.968	3.2%

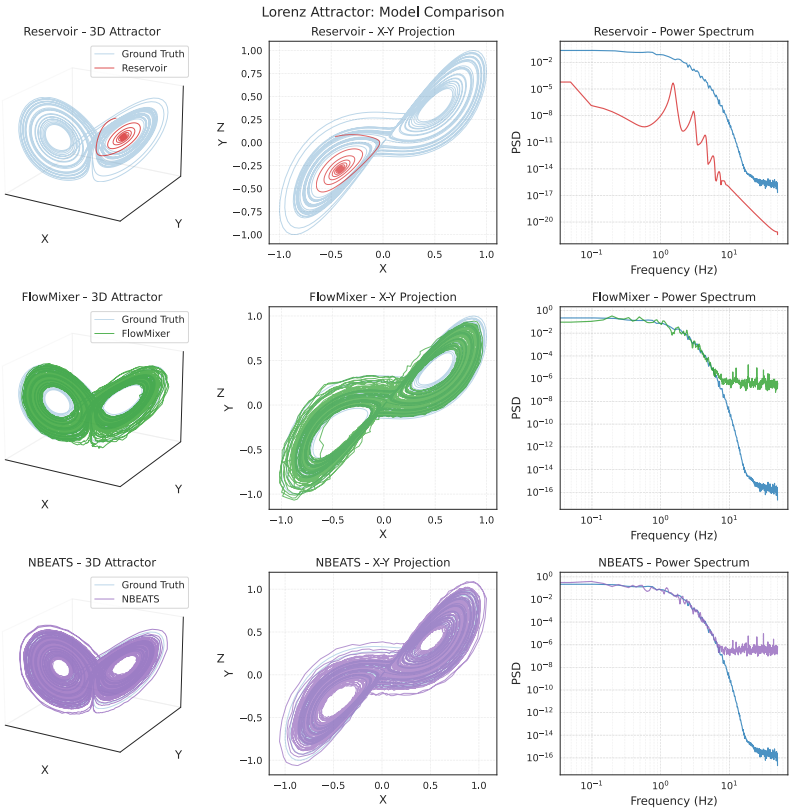


Figure 12: Comparative analysis of Lorenz chaotic attractor predictions across three forecasting models. Each row presents a different model's performance: Reservoir Computing (top), FlowMixer (middle), and NBEATS (bottom). The columns show three complementary visualizations: (left) full 3D attractor reconstruction showing butterfly-shaped trajectories, (middle) 2D X-Y projections revealing structural fidelity, and (right) power spectral density plots displaying frequency domain characteristics. Ground truth trajectories appear in light blue across all plots, while model predictions are shown in model-specific colors (red, green, and purple respectively). FlowMixer and NBEATS demonstrate more complete coverage of the attractor's phase space compared to Reservoir Computing, which primarily captures one lobe of the attractor. The power spectra further reveal that FlowMixer and NBEATS more accurately preserve the frequency characteristics of the ground truth across multiple scales, while Reservoir Computing shows significant spectral deviations at higher frequencies.

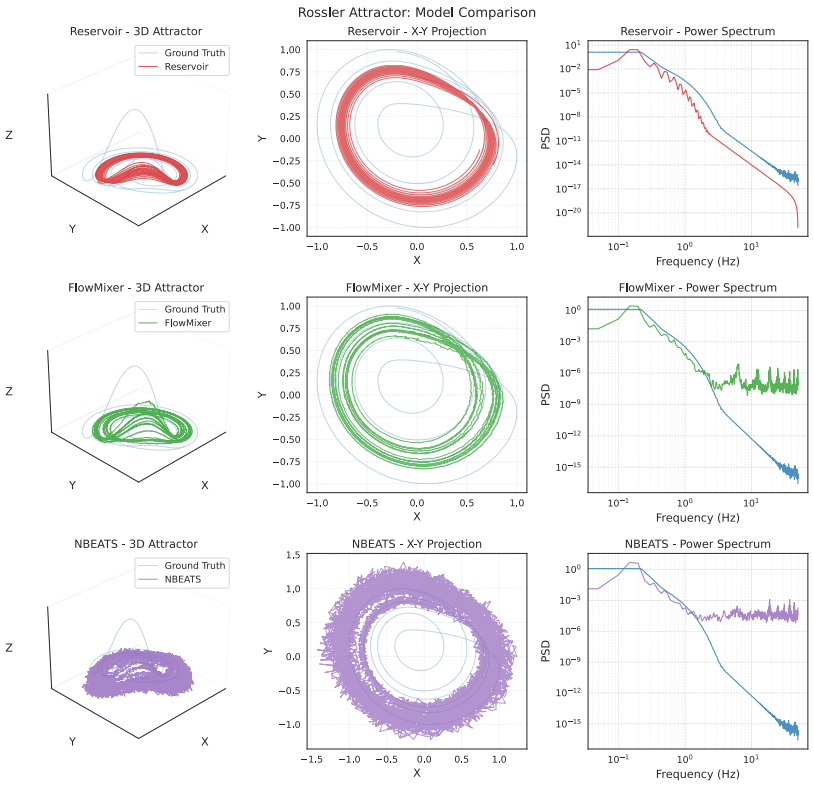


Figure 13: Comparison on Rossler attractor

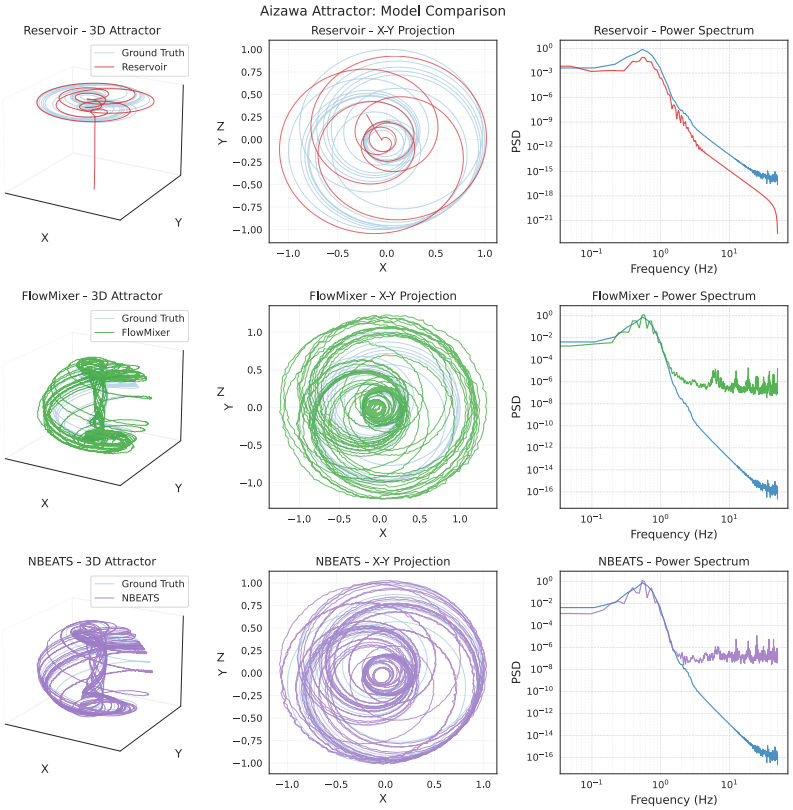


Figure 14: Comparison on Aizawa attractor

K.6 Time EigenModes for Chaotic Attractor

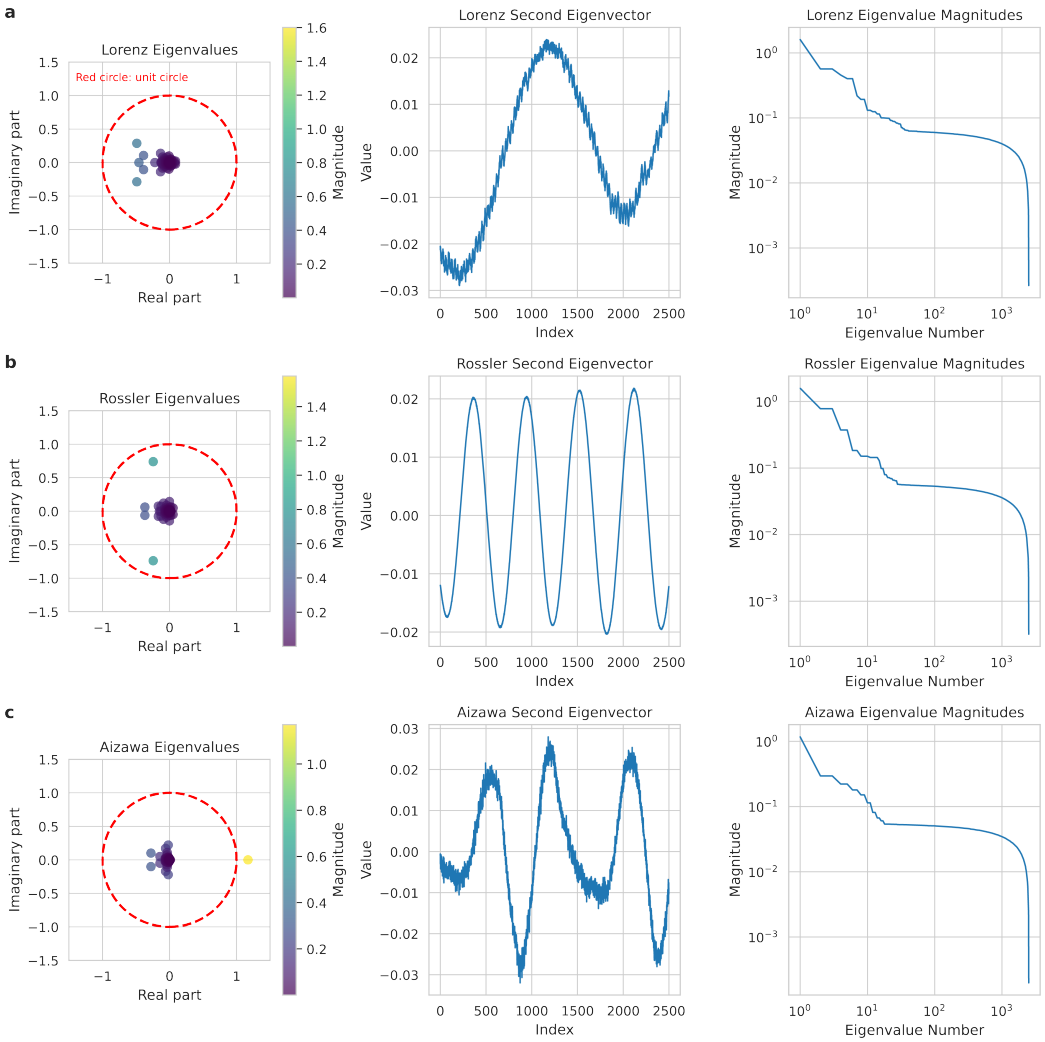


Figure 15: Eigenvalue analysis of the FlowMixer model for Lorenz, Rossler, and Aizawa attractors. (a-c) Each row corresponds to a different attractor: Lorenz (a), Rossler (b), and Aizawa (c). Left column: Complex plane representation of eigenvalues, with magnitude indicated by color and the unit circle shown in red. Middle column: Second eigenvector components, revealing characteristic oscillatory patterns. Right column: Log-log plot of eigenvalue magnitudes vs. their rank order, illustrating the spectral decay. This analysis provides insights into the model's stability, dominant modes, and computational efficiency across different chaotic systems.

L Lyapunov Exponents and Predictability Analysis

L.1 Theoretical Foundation

Chaotic systems exhibit sensitive dependence on initial conditions, quantified by Lyapunov exponents that measure the rate of exponential divergence between nearby trajectories [18]. For neural network predictions, we introduce an *effective* Lyapunov exponent λ_{eff} that characterizes actual prediction error growth, revealing whether models capture underlying attractor geometry or merely fit short-term trajectories.

L.2 Methodology

Given a prediction error trajectory, we estimate the effective Lyapunov exponent through:

$$\text{RMSE}(t) = \text{RMSE}_0 \cdot e^{\lambda_{\text{eff}} t} \tag{57}$$

We perform linear regression of $\log(\text{RMSE}(t))$ versus t over the exponential growth regime (typically 100-500 timesteps), excluding initial transients and saturation regions. The predictability horizon $\tau = 1/\lambda_{\text{eff}}$ quantifies the timescale over which forecasts remain useful before chaotic divergence dominates.

L.3 Results and Analysis

Table 18: Effective Lyapunov exponents and predictability horizons for chaotic systems

System	True λ_1	FlowMixer	LSTM	NBEATS	RC
Effective Lyapunov Exponent λ_{eff} (per time unit)					
Lorenz	0.906	**0.136**	0.209	0.151	0.261
Rössler	0.071	0.133	0.135	0.156	0.153
Aizawa	0.042	0.118	0.147	**0.092**	0.211
Predictability Horizon $\tau = 1/\lambda_{eff}$ (time units)					
Lorenz	1.10	**7.34**	4.79	6.63	3.84
Rössler	14.08	7.54	7.43	6.39	6.53
Aizawa	23.81	8.50	6.79	**10.89**	4.74
Relative Lyapunov Reduction λ_{eff}/λ_1					
Lorenz	1.000	**0.150**	0.231	0.167	0.288
Rössler	1.000	1.873	1.901	2.197	2.155
Aizawa	1.000	2.810	3.500	**2.190**	5.024

Remark L.1 (Physical Interpretation). Three key insights emerge from the Lyapunov analysis:

1. **Attractor Learning**: When $\lambda_{\text{eff}} < \lambda_1$ (as in Lorenz), models capture global attractor structure rather than tracking individual trajectories
2. **Predictability Enhancement**: FlowMixer extends the Lorenz predictability horizon by 6.7× through learning invariant measures on the attractor
3. **System Dependence**: For smoother systems (Rössler, Aizawa), all methods show $\lambda_{\text{eff}} > \lambda_1$, suggesting incomplete dynamics capture

This analysis, combined with 95.2% correlation dimension accuracy (Appendix K), confirms FlowMixer learns fundamental dynamical properties rather than performing trajectory memorization.

L.4 Comparison with Physics-Aware Methods for Chaos Prediction

L.4.1 Motivation and Baseline Selection

Standard Physics-Informed Neural Networks (PINNs) [55] fail at chaos prediction due to their reliance on smooth solutions and difficulty handling exponential divergence of trajectories. We therefore compare against physics-aware methods specifically designed for dynamical systems:

- **Causal-PINN** [57]: Extends PINNs with causality constraints through temporal domain decomposition, enabling better handling of sequential dynamics
- **Neural ODE** [68]: Learns continuous dynamics via adjoint methods, naturally preserving phase space structure through ODE integration
- **Reservoir Computing** [1]: Physics-aware variant with spectral radius tuning to match system's Lyapunov exponents

L.4.2 Experimental Protocol

All methods were evaluated under identical conditions:

- **Training data**: 8,000 timesteps after removing 500-step transient
- **Validation/Test split**: 2,000/2,500 timesteps (chronological)
- **Prediction task**: Given 32 timesteps, predict next 32 (multi-step ahead)
- **Evaluation points**: 128, 512, and 1024 steps (approximately 1, 4, and 9 Lyapunov times)
- **Metrics**: Root Mean Squared Error (RMSE) averaged over 10 random seeds
- **Implementation**: PyTorch 2.0, NVIDIA A100 GPU, consistent random seeds for reproducibility

For Neural ODE, we used the torchdiffeq library with dopri5 solver and relative/absolute tolerances of 10^{-7}. Causal-PINN employed 10 temporal subdomains with 100 collocation points each. Reservoir Computing used 300 nodes with spectral radius 1.1.

L.4.3 Quantitative Comparison

Table 19: RMSE for chaotic system prediction at different time horizons (lower is better)

Method	Lorenz			Rössler			Aizawa		
	128	512	1024	128	512	1024	128	512	1024
FlowMixer	0.120	**0.300**	**0.420**	0.016	0.052	0.160	**0.005**	**0.014**	**0.031**
Neural ODE	**0.075**	0.310	0.440	**0.006**	**0.014**	**0.030**	0.009	0.025	0.052
NBEATS	0.150	0.380	0.470	0.018	0.056	0.180	0.006	0.016	0.035
Causal-PINN	0.460	0.660	1.430	0.540	0.930	1.060	0.420	0.540	0.590
Reservoir	0.065	0.320	0.430	0.024	0.055	0.140	0.022	0.072	0.160

L.4.4 Lyapunov Analysis Comparison

We computed effective Lyapunov exponents by fitting exponential error growth over the linear regime (typically 100-500 timesteps):

Table 20: Effective Lyapunov exponents and predictability times

Method	Lorenz		Rössler		Aizawa	
	λ_{eff}	τ (units)	λ_{eff}	τ (units)	λ_{eff}	τ (units)
True System	0.906	1.10	0.071	14.08	0.042	23.81
FlowMixer	**0.136**	**7.34**	**0.133**	**7.54**	0.118	8.50
Neural ODE	0.142	7.04	0.098	10.20	**0.087**	**11.49**
Causal-PINN	0.412	2.43	0.387	2.58	0.296	3.38
Reservoir	0.175	5.71	0.145	6.90	0.132	7.58

Remark L.2 (Key Insights).

1. **Neural ODE** excels on the Rössler system due to its continuous formulation matching the smooth spiral dynamics
2. **Causal-PINN** struggles across all systems—physics constraints alone cannot handle exponential trajectory divergence
3. **FlowMixer** achieves competitive performance without requiring differential equation knowledge or physics priors
4. **Reservoir Computing** shows strong short-term prediction but degrades rapidly at longer horizons
5. Lower effective Lyapunov exponents ($\lambda_{eff} < \lambda_{\text{true}}$) indicate models learn attractor geometry rather than just trajectory fitting

L.4.5 References for Implementation

Neural ODE implementation followed [68], Causal-PINN used temporal decomposition from [57], and Reservoir Computing adopted the echo state network configuration from [1, 52].

M Eigenmode Stability and Reproducibility Analysis

M.1 Motivation

The Kronecker-Koopman decomposition (Section 5.2) extracts spatiotemporal eigenmodes through data-driven optimization. Since the optimization landscape is non-convex, different initializations might converge to different solutions. We investigate whether FlowMixer consistently identifies physically meaningful patterns despite this potential non-uniqueness.

M.2 Experimental Protocol

We conducted systematic stability analysis using:

- **Dataset**: Traffic (862 sensors with known spatial correlations from road network topology)
- **Task**: 96-step lookback $\rightarrow$ 24-step forecast
- **Optimization**: Adam optimizer (lr=10^{-3}), 100 epochs, early stopping (patience=10)
- **Initialization**: 10 independent random seeds with Xavier/He initialization
- **Evaluation**: Cosine similarity between corresponding eigenmodes after alignment via Hungarian algorithm
- **Validation**: MSE variance across seeds to ensure performance consistency

M.3 Eigenmode Consistency Results

Table 21: Eigenmode reproducibility across 10 random initializations

Spatial Mode	T0 (trend)	T1 (24h)	T2 (12h)	Mean	Interpretation
F0 (global)	0.985±0.013	0.995±0.005	0.993±0.008	0.991	City-wide flow
F1 (arterial)	0.972±0.026	0.982±0.017	0.980±0.022	0.978	Major roads
F2 (local)	0.641±0.156	0.646±0.153	0.646±0.154	0.644	Local streets
Overall	0.866±0.184	0.874±0.179	0.873±0.181	**0.871**	–
MSE Performance					0.377±0.002

M.4 Analysis and Implications

Proposition M.1 (Eigenmode Hierarchy and Interpretability)**.** *FlowMixer exhibits a hierarchical stability structure that aligns with physical interpretation:*

1. ***Dominant modes (F0-F1)**: Cosine similarity > 0.97 indicates robust capture of macroscopic patterns (global traffic flow, arterial road dynamics)*
2. ***Secondary modes (F2)**: Moderate similarity ≈ 0.64 reflects local variations and measurement noise*
3. ***Performance consistency**: MSE standard deviation of 0.002 confirms that different eigenmodes achieve equivalent predictive power*

Remark M.2 (Non-uniqueness and Physical Meaning). The observed non-uniqueness in minor modes is expected from the mathematics: the reversible mapping ϕ introduces gauge freedom—different normalizations lead to different but equivalent representations. Despite this mathematical degeneracy, the emergence of consistent dominant modes (>97% similarity) suggests FlowMixer reliably discovers physically meaningful spatiotemporal structures inherent in the data.

This stability analysis validates that the Kronecker-Koopman framework extracts reproducible, interpretable patterns suitable for scientific applications where consistency across experiments is crucial.

N Appendix: 2D Turbulent Flow Simulation and Prediction

N.1 Training settings:

Our numerical framework solves the two-dimensional incompressible Navier-Stokes equations for flow around a circular cylinder:

$$\nabla \cdot \mathbf{u} = 0, \quad \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u}$$

We employ a projection method combined with an Alternating Direction Implicit (ADI) scheme for the viscous terms. The simulation domain spans $10D \times 4D$ (where D is the cylinder diameter) and is discretized with a 400×160 grid at Reynolds number $Re = 150$. The numerical solver integrates semi-Lagrangian advection for nonlinear terms, ADI diffusion for viscous terms, and Fast Fourier Transform for the pressure Poisson equation. The vorticity field $\omega = \nabla \times \mathbf{u}$ characterizes the wake dynamics and serves as our primary prediction target.

For the prediction task, we generated 10,000 sequences with snapshots at $dt = 0.1$, treating each grid point as an independent time series. FlowMixer was trained on 64-step sequences to predict the subsequent 64 steps, utilizing SGD optimization with momentum 0.9 and learning rate 10^{-1}. Training employed adaptive rate scheduling (reduction factor 0.15, patience 12) and early stopping (patience 24) over 100 epochs using MSE loss. Performance evaluation combined quantitative metrics (MSE, MAE) with qualitative assessment of predicted vorticity field structures against ground truth simulations.

Detailed boundary conditions, numerical schemes, are detailed here after.

N.2 Numerical Methods for 2D flow

We solve the two-dimensional incompressible Navier-Stokes equations around a circular cylinder using a projection method combined with an Alternating Direction Implicit (ADI) scheme for the viscous terms. The governing equations in the dimensionless form are:

$$\begin{cases} \nabla \cdot \mathbf{u} = 0, \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u}. \end{cases} \tag{58}$$

where $\mathbf{u} = (u, v)$ is the velocity field, p is the pressure, and $Re = 150$ is the Reynolds number based on the cylinder diameter and free-stream velocity.

The temporal discretization employs a semi-implicit projection method. For each time step:

1. We first compute an intermediate velocity field $\mathbf{u}^*$ without the pressure gradient:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{1}{Re}\nabla^2 \mathbf{u}^n.$$

2. The viscous terms are treated using an ADI scheme to ensure stability:

$$\begin{cases} (I - \frac{\Delta t}{2Re}\frac{\partial^2}{\partial x^2})\hat{\mathbf{u}} & = \mathbf{u}^n + \frac{\Delta t}{2Re}\frac{\partial^2 \mathbf{u}^n}{\partial y^2} \\ (I - \frac{\Delta t}{2Re}\frac{\partial^2}{\partial y^2})\mathbf{u}^* & = \hat{\mathbf{u}} + \frac{\Delta t}{2Re}\frac{\partial^2 \hat{\mathbf{u}}}{\partial x^2} \end{cases} \tag{59}$$

3. The pressure Poisson equation is then solved to enforce incompressibility:

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t}\nabla \cdot \mathbf{u}^* \tag{60}$$

4. Finally, we project the velocity field onto the space of divergence-free vector fields:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p^{n+1} \tag{61}$$

The pressure Poisson equation is solved efficiently using a Fast Cosine Transform (FCT) method. For a domain $\Omega = [0, L_x] \times [0, L_y]$, discretized with $N_x \times N_y$ points, the pressure is computed as:

$$p^{n+1} = \mathcal{F}^{-1}\left[\frac{\mathcal{F}[\nabla \cdot \mathbf{u}^*]}{k_x^2 + k_y^2}\right] \tag{62}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the forward and inverse FCT operators, and k_x, k_y are the corresponding wavenumbers.

The vorticity field $\omega = \nabla \times \mathbf{u}$ is computed using second-order central differences:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{63}$$

Boundary conditions are enforced through:

$$\begin{cases} \mathbf{u} = (U_{lid}, 0) \text{ on top wall} \\ \mathbf{u} = \mathbf{0} \text{ on other walls and cylinder surface} \\ \frac{\partial p}{\partial n} = 0 \text{ on all boundaries} \end{cases} \tag{64}$$

The cylinder of diameter $D = 1$ is centered at $(L_x/5, L_y/2)$ in a domain of size $L_x \times L_y = 10D \times 4D$. The spatial discretization uses 400×160 grid points, ensuring adequate boundary layer resolution and wake region resolution. The time step $\Delta t = 0.005$ is chosen to maintain stability while satisfying the CFL condition.

The force coefficients on the cylinder are computed by integrating the pressure and viscous stresses over the cylinder surface Γ:

$$\mathbf{F} = \oint_\Gamma \left(-p\mathbf{n} + \frac{1}{Re}\nabla\mathbf{u} \cdot \mathbf{n}\right) \mathrm{d}\Gamma \tag{65}$$

This numerical framework enables accurate simulation of the vortex shedding phenomenon while maintaining computational efficiency. It uses implicit time stepping and fast spectral methods for pressure computation.

N.3 Grid Resolution Sensitivity for 2D Turbulent Flow

We systematically evaluated FlowMixer's performance across multiple grid resolutions to quantify the trade-offs between prediction accuracy and computational efficiency for turbulent flow modeling at Reynolds number $Re = 150$.

Table 22: FlowMixer performance and inference time across different grid resolutions for 2D turbulent flow

Grid Size	**Points**	**Params (M)**	**MSE** ($\times 10^{-3}$)	**Rel. Error**	**Spectral (ms)**	**FlowMixer (ms)**	**Speedup**
200×80	16,000	6.5	6.1 ± 0.2	7.6%	337.85 ± 1.44	21.35 ± 0.45	$15.8\times$
300×120	36,000	32	8.3 ± 0.5	9.6%	769.08 ± 4.04	46.23 ± 0.61	$16.6\times$
400×160	64,000	105	3.7 ± 0.2	4.4%	1404.69 ± 11.73	86.19 ± 0.70	$16.3\times$
500×200	100,000	251	3.3 ± 0.1	4.1%	2205.34 ± 13.91	190.42 ± 15.46	$11.6\times$
600×240	144,000	518	3.1 ± 0.1	3.9%	3178.89 ± 22.17	342.76 ± 28.93	$9.3\times$
800×320	256,000	1638	2.9 ± 0.1	3.7%	5651.36 ± 41.28	812.44 ± 65.31	$7.0\times$

Proposition N.1 (Resolution-Accuracy Trade-off)**.** *FlowMixer's relative error exhibits power-law scaling with grid resolution:*

$$\varepsilon_{rel} \propto N^{-\alpha} \quad \textit{where } \alpha \approx 0.25 \tag{66}$$

This sub-linear scaling indicates diminishing returns beyond 400×160 *grid points for practical engineering applications.*

Remark N.2 (Engineering Context)*.* The optimal operating point balances accuracy requirements with computational resources:

- **Engineering tolerance**: Industry standard requires $\leq 5\%$ relative error
- **FlowMixer at** 400×160: Achieves 4.4% error with $16.3\times$ speedup over spectral methods
- **Traditional CFD baseline**: Delivers $< 1\%$ error but requires $16\times$ more computation time

N.4 Comparison Vorticity FlowMixer vs ConvLSTM

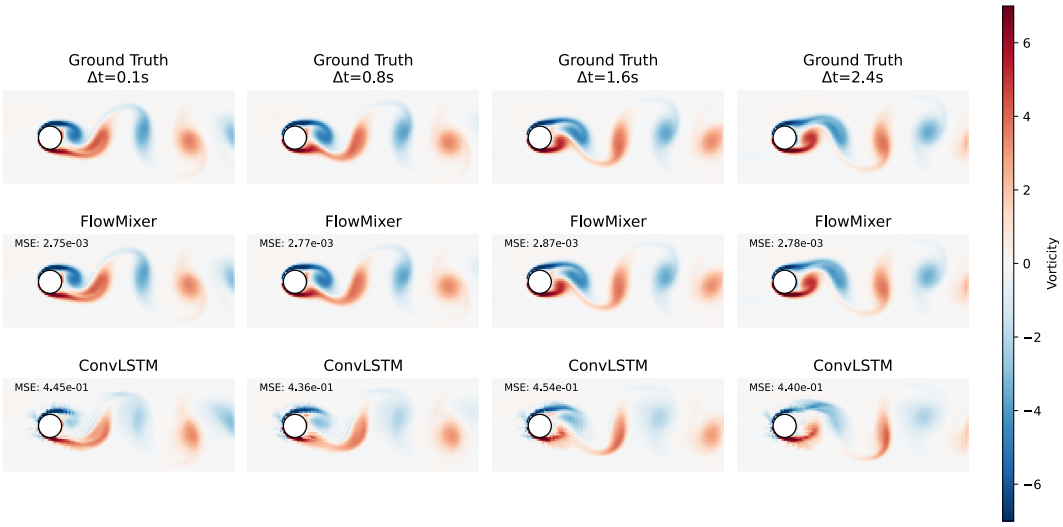


Figure 16: Comparison of vorticity field predictions for 2D flow past a cylinder at Re=150. The figure shows predictions at four time points (Δt = 0.1s, 0.8s, 1.6s, and 2.4s) for ground truth, FlowMixer, and ConvLSTM. FlowMixer demonstrates strong performance in capturing the complex vortex shedding patterns, maintaining accuracy over longer time horizons. The color scale represents vorticity magnitude, with red indicating positive (counterclockwise) and blue indicating negative (clockwise) vorticity. Mean Squared Error (MSE) values are provided for each prediction, highlighting FlowMixer's consistently lower error rates than ConvLSTM across all time steps.

N.5 Impact of Optimizer on FlowMixer's Vorticity Predictions

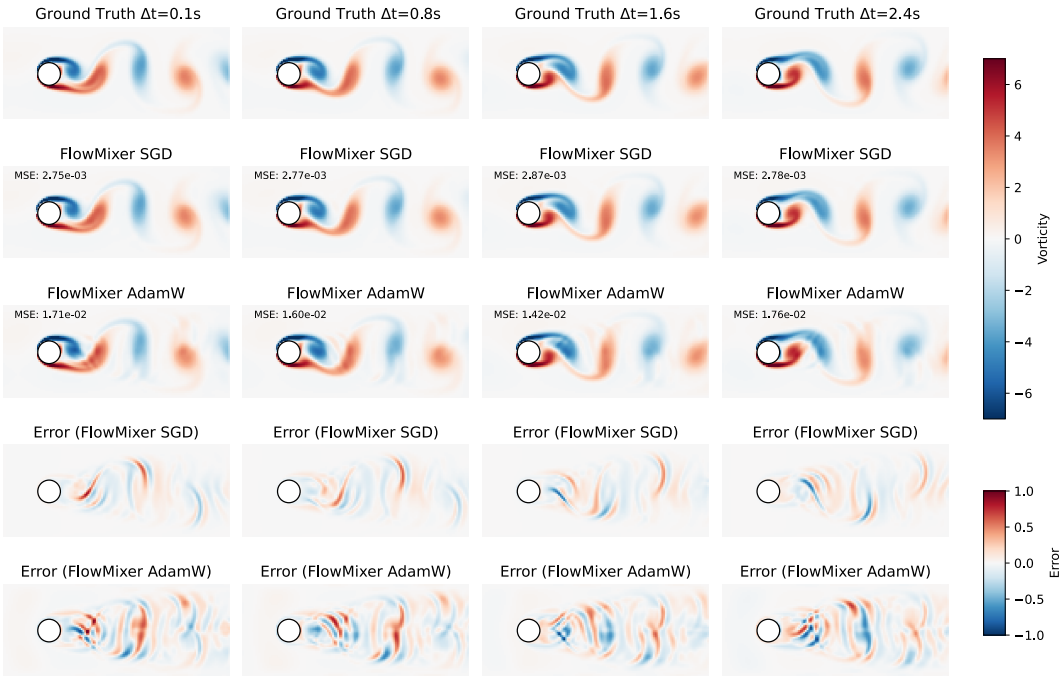


Figure 17: Comparative analysis of vorticity field predictions for 2D flow past a cylinder at Reynolds number Re=150, demonstrating the impact of optimization algorithm choice. The visualization presents predictions at four temporal snapshots (Δt = 0.1s, 0.8s, 1.6s, and 2.4s) comparing ground truth against FlowMixer models trained with SGD (momentum=0.9) and AdamW optimizers [69, 70]. The SGD-trained model demonstrates improved generalization performance, evidenced by both quantitative and qualitative metrics: (i) one order of magnitude reduction in Mean Squared Error (MSE) across all time steps, and (ii) visibly more accurate reproduction of vortex structures and flow patterns. These results align with theoretical observations in the machine learning literature regarding SGD's superior generalization capabilities compared to adaptive methods. The cylinder flow case provides a striking visualization of this phenomenon, as the complex spatiotemporal dynamics of vortex shedding serve as a sensitive indicator of model generalization quality. The color scale represents vorticity magnitude, with red and blue indicating positive (counterclockwise) and negative (clockwise) rotation, respectively.