LEARNING TO FORECAST ON CONVOLUTION-INDUCED GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Delay-coordinate methods provide a principled way to expose the geometry of dynamical systems from observed time series. However, classical constructions typically choose coordinates to recover or characterize the dynamics of observed trajectories, rather than to optimize a downstream forecasting objective. We ask whether such coordinates can instead be learned directly for prediction. Building on the connection between convolutional coordinates and finite-dimensional Koopman representations, we show that a causal convolution provides a learnable projection of a delay embedding, with Hankel-SVD convolutional coordinates arising from a particular choice of filters. This motivates a neural forecaster that learns delay coordinates end-to-end for prediction. We further use a Koopman regularizer to encourage approximately linear evolution in the learned coordinate space while retaining efficient direct multi-horizon prediction. Experiments show that the model achieves strong performance on standard forecasting benchmarks. More importantly, ablations and representation analyses reveal that forecasting supervision gives the learned coordinates predictive structure, while Koopman regularization helps organize their temporal evolution. Overall, our results show that strong neural forecasting and an explicit, analyzable dynamical representation can be achieved within the same learned coordinate space, connecting predictive accuracy with the study of what the forecaster learns.

1 INTRODUCTION

Neural time-series forecasting requires models to extract intrinsic temporal structures, such as trends and seasonality Nie et al. (2022); Zeng et al. (2023); Oreshkin et al. (2019), while also accounting for cross-variate interactions that indicate future evolution Liu et al. (2024c). Much progress has largely focused on strengthening these capabilities through architectures for long-range temporal and cross-variable modeling, task-specific inductive biases Wang et al. (2025); Lu et al. (2025), and, more recently, large-scale foundation models trained across heterogeneous time-series collections Ansari et al. (2025); Vishwas & Macharla (2025); Khwaja et al. (2026); Liu et al. (2026). In parallel, a growing line of research has revisited forecasting from a dynamical-systems perspective Durstewitz et al. (2026), asking whether neural models can recover representations that more explicitly reflect the processes underlying observed trajectories Gilpin (2020); Park et al. (2024); Brenner et al. (2025). Classical embedology provides a natural basis for this perspective: delay-coordinate constructions use temporal history to reconstruct a state-space geometry Packard et al. (1980) that, under appropriate conditions, preserves the dynamics of the underlying system Sauer et al. (1991). Related Koopman approaches further seek observable representations in which nonlinear evolution can be represented, or approximated, by simpler linear dynamics Bevanda et al. (2021). This viewpoint is compatible with neural representation learning, where observations are transformed into task-adapted latent states. This raises a central question: can a neural forecaster learn an explicit, analyzable coordinate representation that supports strong prediction, and when does its dynamical structure help?

One route toward dynamics-aware forecasting is to first reconstruct a phase-space representation from the observed series using delay-coordinate techniques Kamb et al. (2020); Young & Graham (2023). However, such reconstruction is typically specified before forecasting optimization. Its quality can depend strongly on design choices such as the delay and embedding dimension, which are commonly determined by separate reconstruction criteria or treated as hyperparameters rather than optimized for the downstream forecasting objective. More importantly, even a faithful reconstruction of the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

observed dynamics does not necessarily provide the coordinates most useful for prediction. When the coordinate construction is fixed beforehand, the forecaster must therefore adapt to a geometry that is not optimized for its prediction objective.

Motivated by this limitation, we propose Forecasting on Convolution-Induced Geometry (FCIG), which learns convolutional coordinates directly from forecasting supervision. Crucially, we view causal convolution not merely as a feature extractor, but as a shared learnable projection of local delay observations, providing a neural counterpart of classical delay-coordinate transformations. An auxiliary Koopman regularization encourages approximately linear temporal evolution of the learned coordinates. The Koopman matrix serves only as a structural regularizer, rather than a recursive forecast generator. Forecasting is performed in the learned coordinate space: an MLP maps historical coordinate sequences to future latent trajectories, which are then linearly decoded into measurements. This integrates coordinate learning, dynamical regularization, and direct forecasting within a coherent training framework. Our contributions are threefold:

- **Learnable convolutional coordinates.** We introduce a learnable convolutional coordinate representation for neural time-series forecasting. By interpreting causal convolution as a learnable projection of local delay observations, the proposed formulation allows the coordinate geometry to be optimized directly for the forecasting objective rather than prescribed beforehand.
- **Dynamics-informed direct forecasting.** We formulate a compact dynamics-informed forecaster in which forecasting supervision and an auxiliary Koopman constraint jointly shape the learned coordinates. Unlike Koopman forecasting models that rely on recursive latent evolution, the learned operator is used only to regularize temporal structure, while multi-horizon predictions are generated directly from the learned coordinates.
- **Forecasting performance and representation analysis.** Experiments show strong forecasting performance against both general-purpose and dynamics-oriented baselines. Phase-space, spectral, and neighborhood analyses reveal temporal patterns in the learned coordinates and show how training and regularization affect their geometry and predictive consistency.

2 Preliminaries and Motivation

2.1 Forecasting from Measurements

Let $\xi_t \in \mathcal{M}$ denote an unobserved state evolving under $\xi_{t+1} = F(\xi_t)$, with measurements $\mathbf{x}_t = g(\xi_t) \in \mathbb{R}^V$. In practical forecasting, the state ξ_t and evolution law F are generally unknown. Given a lookback window $X = [\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{V \times T}$, a neural forecaster learns

$$\widehat{Y} = f_\theta(X), \qquad Y = [\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+H}] \in \mathbb{R}^{V \times H}, \tag{1}$$

where T is the lookback length, H is the forecasting horizon, and V is the number of observed variables. Although f_θ may transform the observations into latent features, patches, or tokens, the forecasting task is defined from observed history to future measurements.

2.2 Hankel Delay Embedding

Consider one observed variable within a lookback window, written as the univariate sequence $x_{1:T}$ after reindexing. Its k-dimensional delay-coordinate vector with lag τ is defined as

$$\mathbf{d}_t = \left[x_t, x_{t-\tau}, x_{t-2\tau}, \ldots, x_{t-(k-1)\tau}\right]^\top \in \mathbb{R}^k. \tag{2}$$

Rather than representing the system using only the instantaneous measurement x_t, the delay vector augments it with regularly spaced measurements from its temporal history.

Takens' embedding theorem provides a foundation for representing dynamical states through delayed observations. In its classical setting, for a generic pair consisting of a smooth diffeomorphism on a compact d-dimensional manifold and a smooth scalar observation function, a delay map with $k \geq 2d + 1$ is an embedding Takens (2006). The state manifold is therefore diffeomorphic to its image in delay-coordinate space. In practice, the quality of a finite delay representation also depends on the lag τ and dimension k.

Collecting delay vectors along the observed trajectory gives the delay-coordinate matrix

$$\mathbf{H}_\tau = [\mathbf{d}_{t_0}, \mathbf{d}_{t_0+1}, \ldots, \mathbf{d}_T], \qquad t_0 = 1 + (k-1)\tau, \tag{3}$$

or explicitly,

$$\mathbf{H}_\tau = \begin{bmatrix} x_{t_0} & x_{t_0+1} & \cdots & x_T \\ x_{t_0-\tau} & x_{t_0+1-\tau} & \cdots & x_{T-\tau} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_0-(k-1)\tau} & x_{t_0+1-(k-1)\tau} & \cdots & x_{T-(k-1)\tau} \end{bmatrix}. \tag{4}$$

For unit lag $\tau = 1$, reversing the row order of $\mathbf{H}_\tau$ gives the standard Hankel delay matrix.

2.3 Convolutional Coordinates and Koopman Dynamics

Delay coordinates can be further transformed by projecting local trajectory segments onto basis functions. Following Kamb et al. (2020), consider a continuous-time flow Φ^s and a complete differentiable orthonormal basis $\{\phi_r\}_{r=1}^\infty$ of real-valued functions on a finite time interval I. For a scalar component g_v of the measurement function g, the corresponding convolutional coordinates are

$$w_r(\xi) = \int_I \phi_r(s)\, \mathcal{K}^s g_v(\xi)\, ds, \qquad \mathcal{K}^s g_v(\xi) = g_v(\Phi^s(\xi)). \tag{5}$$

Here, the associated Koopman operator $\mathcal{K}^s$ acts on measurement functions by composing them with the state evolution. Along a trajectory $\xi(t)$, this becomes

$$w_r(\xi(t)) = \int_I \phi_r(s)\, g_v(\xi(t+s))\, ds, \tag{6}$$

showing that a convolutional coordinate is a projection of a local delay trajectory onto a basis function.

Under the regularity and convergence conditions of Kamb et al. (2020), the Koopman generator $\mathcal{G}$ has the representation

$$\mathcal{G} w_r = \sum_{j=1}^\infty A_{rj} w_j, \qquad A_{rj} = \int_I \phi_r(s) \phi_j'(s)\, ds, \tag{7}$$

where ϕ_j' denotes the derivative of ϕ_j with respect to s.

For a fixed basis, the coefficients A_{rj} depend on the basis functions, while the accuracy of a finite-dimensional approximation depends on how well the retained coordinates represent the observed trajectories. Hankel-SVD provides a data-adapted basis for compact dynamical representations Brunton et al. (2017); Kamb et al. (2020). However, fixing this basis leaves it outside forecasting optimization, while recomputing it during training introduces matrix decompositions and differentiation through singular vectors. We instead parameterize delay projections with learnable causal filters, retaining the connection to convolutional coordinates while enabling direct optimization through forecasting supervision. This formulation also supports stacked nonlinear encoders and benefits from optimized convolution implementations.

3 Method

We first show that Hankel-SVD projection coordinates are a special case of a linear causal convolutional transform. This motivates replacing the prescribed SVD basis with learnable filters. We then build a direct multi-horizon forecaster on the learned coordinates and introduce Koopman Autoregressive Regularization (KAR) to encourage approximately linear coordinate evolution.

3.1 Learning Forecasting-Oriented Convolutional Coordinates

For the delay vector $\mathbf{d}_t$ defined in Eq. (2), a learnable filter $\mathbf{h}_r \in \mathbb{R}^k$ produces the coordinate

$$z_{r,t} = \mathbf{h}_r^\top \mathbf{d}_t + b_r = \sum_{j=0}^{k-1} h_{r,j} x_{t-j\tau} + b_r. \tag{8}$$

Thus, a causal dilated convolution implements a shared affine projection of local delay observations.

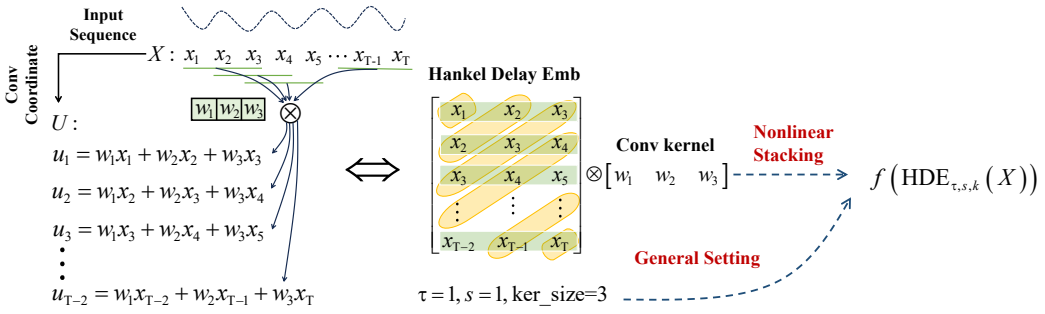


Figure 1: Connection between delay-coordinate projections and causal convolution. A linear projection of each delay vector can be implemented by a shared convolutional filter.

Proposition 3.1 (Hankel-SVD coordinates as a special case)**.** *Let*

$$\mathbf{H} = [\mathbf{d}_{t_1}, \ldots, \mathbf{d}_{t_M}] = \mathbf{U}_H \mathbf{\Sigma}_H \mathbf{V}_H^\top \tag{9}$$

be the delay matrix and its singular value decomposition. For its r-th left singular vector $\mathbf{u}_r$, choosing $\mathbf{h}_r = \mathbf{u}_r$ and $b_r = 0$ in Eq. (8) yields

$$[z_{r,t_1}, \ldots, z_{r,t_M}] = \mathbf{u}_r^\top \mathbf{H} = \sigma_r \mathbf{v}_r^\top. \tag{10}$$

Hence, the SVD projection coordinates $\sigma_r \mathbf{v}_r^\top$ are a special case of the linear convolutional parameterization.

Proof. With $\mathbf{h}_r = \mathbf{u}_r$ and $b_r = 0$, Eq. (8) gives $z_{r,t} = \mathbf{u}_r^\top \mathbf{d}_t$. Stacking these values and applying the SVD gives $\mathbf{u}_r^\top \mathbf{H} = \sigma_r \mathbf{v}_r^\top$. □

Rather than fixing the SVD basis, we optimize the convolutional filters jointly with the forecasting objective. In practice, we extend the linear construction with stacked convolution layers:

$$Z = E_\theta(X), \; Z \in \mathbb{R}^{V \times D \times T} \tag{11}$$

where $X \in \mathbb{R}^{V \times T}$ is the multivariate input, E_θ is the coordinate encoder. We refer to the resulting representation as the Convolution-Induced Coordinate Space (CICS), with D coordinates per stream. The proposition establishes the linear special case; the full encoder provides a more flexible nonlinear map, which is detailed in Appendix B.2.

3.2 Direct Forecasting on the Learned Coordinates

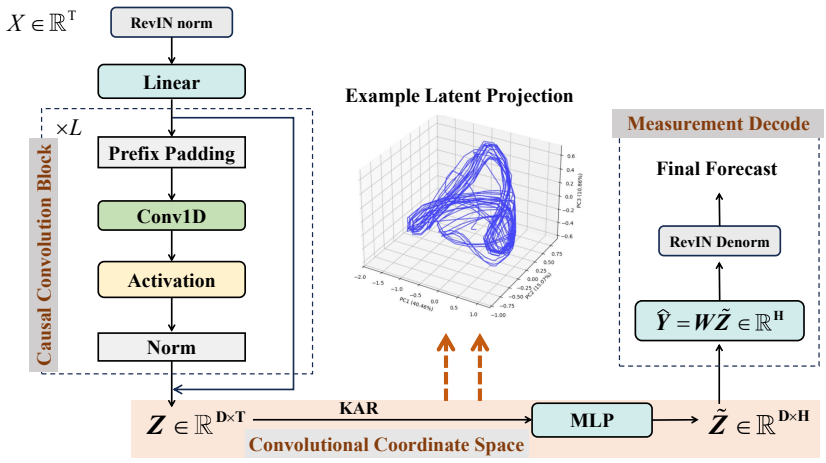


Figure 2: Overview of a forecasting block in FCIG, shown for one variate. The central plot illustrates a projection of learned coordinates.

A Koopman-based forecaster can recursively propagate coordinates through a fitted operator and then decode them into observations. Such rollout can accumulate prediction errors, while training through

repeated transitions can introduce optimization difficulties (Kumar & Isola, 2026). We instead use the dynamical constraint to shape the coordinates and generate forecasts through a direct predictor:

$$\widehat{Y} = F_\phi(Z), \qquad \widehat{Y} \in \mathbb{R}^{V \times H}. \tag{12}$$

Here, H is the forecasting horizon, and F_ϕ includes the MLP prediction head, measurement decoder, and inverse normalization where applicable.

The full model follows the Boosted Direct Output strategy (Lu et al., 2026). In our two-block configurations, the first block predicts $H/2$ steps. The second block encodes the history concatenated with this intermediate forecast and directly predicts all H steps. Overall, the encoder learns the coordinate representation, and the prediction head and decoder learn how to use it for forecasting.

3.3 Koopman Regularization of Coordinate Dynamics

Forecasting supervision encourages useful predictive coordinates but does not explicitly constrain their temporal evolution as in Eq. (7). We therefore introduce KAR to encourage an approximately linear relation between coordinate states.

For one coordinate stream, let $\mathbf{z}_t \in \mathbb{R}^D$ denote its state and let $\mathbf{K} \in \mathbb{R}^{D \times D}$ denote the transition matrix for regularization.

$$\mathbf{z}_{t+\Delta} \approx \mathbf{K}\mathbf{z}_t, \tag{13}$$

where Δ is the sampling interval used by the regularizer. We constrain both one-step and two-step transitions:

$$\mathcal{L}_{\mathrm{K}} = \frac{1}{2}\sum_{q=1}^{2} \frac{1}{D|\mathcal{T}_q|} \sum_{t \in \mathcal{T}_q} \|\mathbf{K}^q \mathbf{z}_t - \mathbf{z}_{t+q\Delta}\|_2^2, \tag{14}$$

where $\mathcal{T}_q$ contains the sampled time indices with valid q-step targets. The loss is also averaged over samples and coordinate streams.

Each forecasting block has its own matrix. These matrices provide auxiliary constraints and are not used to generate the final forecasts. Forecasting supervision and KAR therefore encourage predictive information and approximately linear coordinate evolution, respectively.

Alternating optimization of the Koopman constraint. In the default coordinate-update mode, we stop gradients through $\mathbf{K}$. For a single transition, the corresponding term is

$$\mathcal{L}_{\mathrm{K}}^{\mathrm{coord}} = \|\mathrm{sg}(\mathbf{K})\mathbf{z}_t - \mathbf{z}_{t+\Delta}\|_2^2, \tag{15}$$

where sg denotes stop-gradient. Without operator updates, $\mathbf{K}$ remains at its initialization.

We optionally alternate this mode with a target-detached operator-update mode:

$$\mathcal{L}_{\mathrm{K}}^{\mathrm{op}} = \|\mathbf{K}\mathbf{z}_t - \mathrm{sg}(\mathbf{z}_{t+\Delta})\|_2^2. \tag{16}$$

The same gradient rules apply to the two-step term in Eq. (14). In the second mode, gradients update $\mathbf{K}$ and still reach the encoder through the source state $\mathbf{z}_t$; only the future target is detached. This is therefore an alternating gradient scheme rather than strict block-coordinate optimization Boyd et al. (2011). Forecasting supervision remains active in both modes.

3.4 Training Objective

For each forecasting head, the primary supervision measures prediction error in the observation space:

$$\mathcal{L}_{\mathrm{fore}} = \ell(\widehat{Y}, Y), \tag{17}$$

where ℓ combines the time-domain MAE with optional frequency-domain loss (Wang et al., 2024a).

We also compute first-order temporal differences (Majeedi et al., 2025). Let $\widehat{S} = [X_{\mathrm{tail}}, \widehat{Y}]$ and $S = [X_{\mathrm{tail}}, Y]$, where X_{tail} is a shared suffix of the observed history. Then

$$\mathcal{L}_{\mathrm{diff}} = \mathrm{MAE}\left(\nabla_t \widehat{S}, \nabla_t S\right), \tag{18}$$

where ∇_t denotes adjacent temporal differences. This includes the transition from the last observation to the first prediction, as well as changes within the forecast.

After aggregating the losses across forecasting heads, the complete objective is

$$\mathcal{L}_{\text{total}} = (1-\lambda_1)\mathcal{L}_{\text{fore}} + \lambda_1\mathcal{L}_{\text{diff}} + \lambda_2\mathcal{L}_{\text{K}}, \tag{19}$$

where λ_1 mixes the observation-space losses, λ_2 weights KAR. Head-wise aggregation, the history suffix, and other configs are specified in Appendix B.3.

4 Experiment

4.1 Experimental Setup

Datasets. We evaluate FCIG on seven widely used multivariate Long-term Time Series Forecasting (LTSF) datasets: ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity, and Traffic. The four ETT datasets contain seven operational variables, sampled hourly for ETTh and every 15 minutes for ETTm. Weather contains 21 meteorological variables recorded every 10 minutes, while Electricity and Traffic comprise 321 electricity-consumption series and 862 road-occupancy series, respectively, both sampled hourly. Detailed dataset statistics and preprocessing procedures are provided in Table. 3.

Baselines. Existing dynamics-oriented neural forecasters do not consistently match the predictive accuracy of general-purpose state-of-the-art models on standard forecasting benchmarks. We therefore organize the baselines into two groups to examine two complementary questions. First, we compare FCIG with representative state-of-the-art forecasters under the standard long-term forecasting protocol to assess its overall forecasting accuracy. Second, we compare FCIG specifically with dynamics-oriented forecasters to determine whether it remains competitive within this model family without sacrificing general forecasting performance. Following the multivariate forecasting protocol of DeepEDM (Majeedi et al., 2025) (forecast horizons $H \in \{48, 96, 144, 192\}$ and lookback length $T = 2H$), this group includes DeepEDM, Koopa (Liu et al., 2023), KNF (Wang et al., 2022), and Attraos (Hu et al., 2024), which respectively build on empirical dynamic modeling, Koopman predictors, Koopman neural operators, and attractor-based representations. For each dataset, we reuse the model and loss hyperparameters from its $H = 96$ LTSF configuration for horizons 48 and 96, and from its $H = 192$ configuration for horizons 144 and 192, without additional hyperparameter search.

4.2 Forecasting Performance

Table 1: Average long-term forecasting results under model-specific hyperparameter selection. Results are averaged over prediction lengths $\{96, 192, 336, 720\}$. Lower values are better. The **best** and second-best results are highlighted; ties receive the same rank.

Models	FCIG ours		ReNF (2026)		TimeBridge (2025a)		DUET (2025)		TimeDistill (2026)		Timer-XL (2025b)		iTransformer (2024b)		TimeMixer (2024b)		PatchTST (2022)		Crossformer (2023)		DLinear (2023)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.381**	**0.403**	0.391	0.416	0.401	0.426	0.401	0.420	0.430	0.441	0.409	0.430	0.439	0.448	0.427	0.441	0.419	0.436	0.439	0.461	0.425	0.439
ETTh2	**0.313**	**0.367**	0.327	0.379	0.345	0.386	0.336	0.384	0.345	0.395	0.352	0.402	0.370	0.403	0.349	0.397	0.351	0.395	0.894	0.680	0.470	0.468
ETTm1	**0.327**	**0.360**	0.331	0.364	0.349	0.380	0.338	0.369	0.348	0.380	0.359	0.382	0.361	0.390	0.356	0.380	0.349	0.381	0.464	0.456	0.356	0.379
ETTm2	**0.242**	**0.300**	0.243	0.301	0.247	0.305	0.248	0.308	0.250	0.312	0.271	0.322	0.269	0.327	0.257	0.318	0.256	0.314	0.501	0.505	0.259	0.325
Weather	**0.211**	**0.243**	0.214	0.247	0.220	0.250	0.219	0.253	0.221	0.269	0.240	0.273	0.232	0.269	0.226	0.264	0.224	0.262	0.232	0.294	0.242	0.293
Electricity	**0.145**	**0.237**	**0.145**	**0.237**	0.152	0.247	0.157	0.248	0.157	0.254	0.155	0.246	0.163	0.258	0.185	0.284	0.171	0.271	0.171	0.263	0.167	0.264
Traffic	0.373	0.248	0.365	**0.245**	**0.357**	0.248	0.393	0.256	0.387	0.271	0.374	0.255	0.395	0.279	0.409	0.279	0.397	0.275	0.522	0.282	0.418	0.287

As shown in Table 1, FCIG is highly competitive with strong recent forecasters under the standard long-term forecasting protocol. When averaged across the four prediction horizons, FCIG achieves the lowest or tied-lowest MSE and MAE on six of the seven datasets. Compared with ReNF, the strongest competing baseline on ETTh and Weather, FCIG reduces average MSE and MAE by approximately 2.6% and 3.1% on ETTh1, and 1.4% and 1.6% on Weather, respectively. These improvements hold across all four prediction horizons, showing nontrivial consistent gains on mature forecasting benchmarks. Besides, we also note that on most datasets, FCIG achieves comparable or better accuracy with narrower MLP prediction heads, suggesting that the learned coordinates may reduce the capacity required by the downstream predictor.

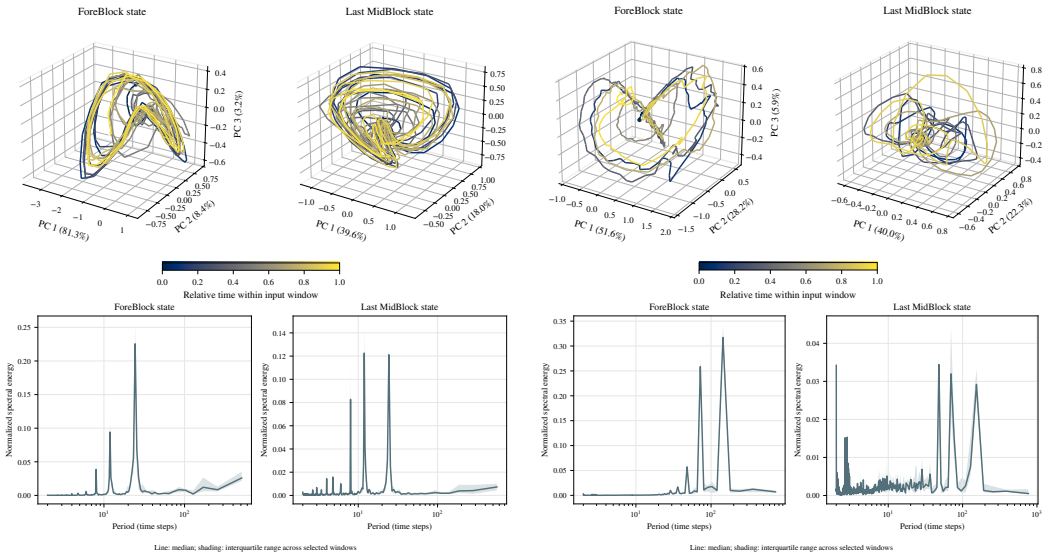
More importantly, Table 2 compares FCIG directly with dynamics-oriented neural forecasters under the DeepEDM protocol. FCIG achieves the lowest average MSE and MAE on every evaluated dataset. The full results in Table 6 further show that FCIG obtains the lowest or tied-lowest error in 51 of the 56 dataset–horizon–metric settings, substantially outperforming DeepEDM, Koopa, KNF, and Attraos in most matched comparisons. This result is particularly relevant because explicitly incorporating dynamical structure has often made it difficult for dynamics-oriented forecasters to remain competitive with general-purpose forecasting models on public benchmarks.

Although the two comparisons follow different forecasting protocols, they provide complementary evidence: FCIG remains competitive with general-purpose state-of-the-art forecasters while outperforming existing dynamics-oriented methods under a matched protocol. These results suggest that introducing an explicit dynamical inductive bias does not necessarily come at the cost of forecasting accuracy. Having established its forecasting competitiveness, we next investigate when and how the learned dynamical information contributes to forecasting.

Table 2: Multivariate forecasting performance against dynamics-oriented forecasters under the DeepEDM; Baseline results are taken directly from (Majeedi et al., 2025). All results are averaged over four forecast horizons: $\{48, 96, 144, 192\}$. The **best** and second-best results are highlighted in red and blue, respectively. Full results are reported in Appendix Table 6.

Models	FCIG ours		DeepEDM		Koopa		KNF		Attraos	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**0.158**	**0.197**	0.165	0.199	0.161	0.210	0.338	0.358	0.176	0.221
Electricity	**0.131**	**0.222**	0.149	0.240	0.143	0.243	0.205	0.292	0.162	0.252
Traffic	**0.380**	**0.247**	0.399	0.266	0.404	0.277	0.662	0.391	0.474	0.321
ETTm1	**0.294**	**0.332**	0.299	0.336	0.309	0.356	0.950	0.766	0.327	0.364
ETTm2	**0.178**	**0.253**	0.182	0.257	0.184	0.265	1.212	0.855	0.189	0.270
ETTh1	**0.362**	**0.383**	0.371	0.390	0.382	0.407	0.898	0.715	0.393	0.407
ETTh2	**0.280**	**0.335**	0.297	0.340	0.303	0.356	0.447	0.445	0.315	0.358

4.3 Analysis of the Learned Coordinate Space

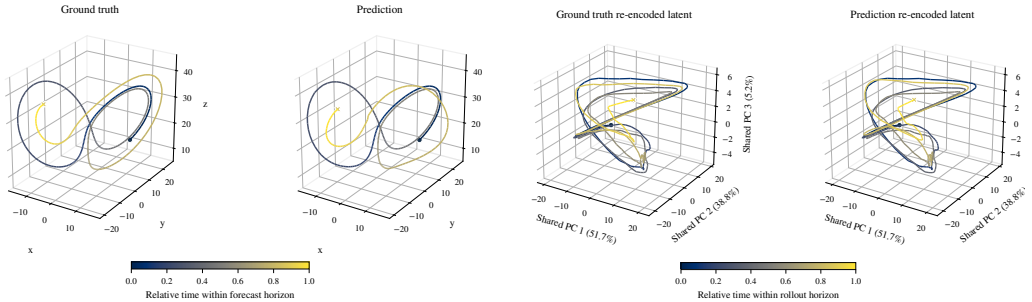


(a) ETTh1–MUFL (b) Weather–Tlog

Figure 3: Visualization of the learned dynamical coordinates on (a) ETTh1–MUFL and (b) Weather–Tlog. The upper panels show their phase-space projections, while the lower panels show the corresponding frequency-space representations.

We qualitatively inspect the Koopman states produced by the first and last forecasting blocks of FCIG. For each variable, we uniformly sample eight windows from the test set. The coordinate visualization applies a moving average with a width of five and projects one representative trajectory onto its first three principal components with PCA. Color indicates relative time within the input window. For the frequency analysis, we report the median normalized spectral energy across the eight windows.

Observations on real-world data. Figure 3 shows that the learned coordinates form smooth and temporally ordered loops rather than diffuse point clouds or tangled lines. Their spectra capture clear periodic patterns: ETTh1–MUFL shows a dominant 24-step daily cycle along with 12- and 8-step intra-day harmonics, corresponding to regular electricity-load activity; Weather–Tlog shows periods around 144 and 72 steps, corresponding to daily and half-daily temperature variations at its ten-minute sampling rate. Across forecasting blocks, ETTh1–MUFL redistributes spectral energy among these harmonics and develops a richer multi-loop geometry, whereas Weather–Tlog retains its dominant cycles while forming a more compact and regular coordinate structure. This contrast suggests that FCIG adapts its coordinates to variable-specific temporal patterns rather than applying a uniform filter or compression. Additional examples in Appendix D.3 further demonstrate parameter-dependent coordinate structures and their learning process.



(a) Original phase space (b) z-channel learned coordinate

Figure 4: Closed-loop prediction of the Lorenz63 system over 336 steps. The ground-truth and predicted trajectories are compared in (a) the original phase space and (b) the learned z-channel coordinate.

Closed-loop prediction on the Lorenz system. We further examine FCIG on the Lorenz63 system, a typical chaotic system with a well-known two-lobe phase-space structure. The model is trained in a multivariate setting with 192 input steps and 96 prediction steps. To obtain a longer and clearer trajectory, we extend the forecast to 336 steps in a closed rollout: each predicted 96-step is appended to the last input and used to predict the next block, without using future ground-truth values. As shown in Figure 4, even in this direct output setting, the predicted long trajectory follows the ground truth of a chaotic Lorenz system. We further re-encode the true and predicted sequences and visualize the learned coordinate associated with the z variable under a shared PCA projection. Its trajectory shows an early outline of the two-lobe geometry. The close match between the true and predicted coordinate paths suggests that FCIG preserves important geometric structure during short closed-loop prediction, although this does not imply accurate long-term prediction of a chaotic system.

4.4 Effects of Coordinate Learning and Koopman Regularization

To quantitatively evaluate how temporal convolutions and Koopman regularization affect the predictive relevance and local temporal consistency of the learned coordinates, we use two metrics adapted from neighborhood-based analyses of state-space reconstructions (Uzal et al., 2011; Gilpin, 2020). KNN future consistency (KNN-FC) measures whether nearby coordinate representations correspond to similar observed future trajectories. Latent transition consistency (LTC) measures whether nearby representations evolve in similar directions, drawing on local trajectory alignment (Pecora et al., 1995; Ouyang et al., 2021). For both metrics, we exclude temporally adjacent samples from neighbor selection to reduce spurious correlations, following the Theiler-window approach (Theiler, 1986). Detailed definitions are provided in Appendix A.2.

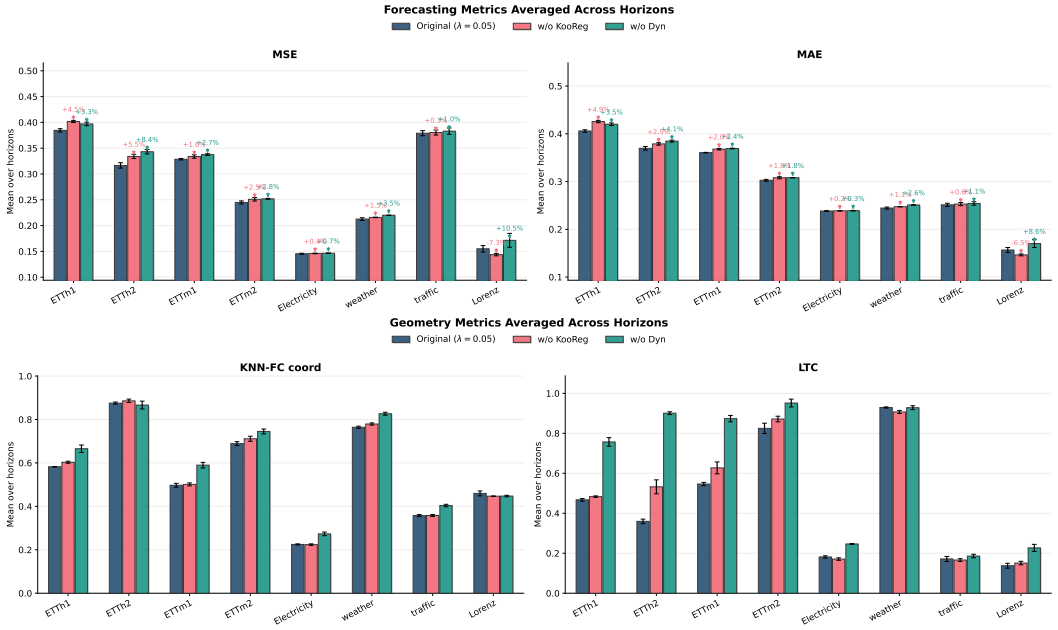


Figure 5: Effects of temporal convolutions and Koopman regularization on forecasting accuracy (Above) and coordinate geometry (Bottom). Results are averaged over four prediction lengths ($H \in \{96, 192, 336, 720\}$) and four random seeds.

Figure 5 reports the mean over four prediction lengths and four random seeds, with error bars showing the standard deviation. *w/o KooReg* removes KAR, while *w/o Dyn* further sets the convolution kernels and dilations to one, reduces the aggregation of delayed observations through temporal convolutions to dense projections while retaining the other encoder operations.

As shown in Figure 5, both ablations increase MSE and MAE on all real-world datasets, with *w/o Dyn* causing larger errors in most cases. On Lorenz63, removing KAR improves forecasting accuracy, whereas additionally removing temporal convolution increases the error relative to the full model. In addition, *w/o Dyn* worsens both geometry metrics on most datasets. Removing KAR generally worsens KNN-FC and LTC. These results suggest that aggregating delayed observations through temporal convolutions helps learn coordinates with more consistent local transitions and more similar observed futures among neighbors. KAR further improves neighborhood future consistency on most datasets.

5 Conclusion

We presented FCIG, an interpretable neural forecaster that learns convolutional coordinates from forecasting supervision and uses Koopman regularization to encourage approximately linear coordinate evolution. The connection between delay-coordinate projections and causal convolution motivates a learnable coordinate construction that integrates dynamical structure into direct forecasting. Experiments show that FCIG remains highly competitive with strong general-purpose forecasters and outperforms dynamics-oriented baselines in most evaluated settings. Coordinate visualizations and ablations further show how temporal convolutions and regularization affect the learned representations, revealing recurring temporal patterns and changes in local consistency. These findings also distinguish geometric regularity from forecasting accuracy: improvements in one do not always imply improvements in the other. Overall, FCIG demonstrates a practical way to bring dynamical structure into neural forecasting while retaining strong predictive performance and making the learned coordinates available for direct analysis.

AI USE STATEMENT

In this work, we used generative AI tools to assist with checking and refining mathematical formulations, coding for repetitive experiments and statistical computations. We also used AI to provide feedback on experimental design. Additionally, these tools supported language refinement, content organization, and LaTeX formatting. All scientific decisions and final conclusions were made by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

The main text describes the coordinate construction, forecasting architecture, and training objective. Appendices B.2 and B.3 provide the encoder structures, dataset-specific hyperparameters, and detailed loss computations. We specify the datasets and evaluation protocols for both forecasting comparisons and report per-horizon results in Appendix C. The geometry metrics are defined in Appendix A.2, while the additional analyses describe the coordinate visualization and comparison procedures.

REFERENCES

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Manuel Brenner, Elias Weber, Georgia Koppe, and Daniel Durstewitz. Learning interpretable hierarchical dynamical systems models from time series data. In *International Conference on Learning Representations*, volume 2025, pp. 85616–85652, 2025.

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.

Xiaoyuan Cheng, Wenxuan Yuan, Yiming Yang, Yuanzhao Zhang, Sibo Cheng, Yi He, and Zhuo Sun. Information shapes koopman representation. In *International Conference on Learning Representations*, volume 2026, pp. 151380–151429, 2026.

Daniel Durstewitz, Christoph Jürgen Hemmer, Florian Hess, Charlotte Ricarda Doll, and Lukas Eisenmann. Position: A dynamical systems perspective is needed to advance time series modeling. *arXiv preprint arXiv:2602.16864*, 2026.

William Gilpin. Deep reconstruction of strange attractors from time series. *Advances in neural information processing systems*, 33:204–216, 2020.

Tao Han, Zhibin Wen, Zhenghao Chen, Dazhao Du, Song Guo, and Lei Bai. Benchmarking physics-informed time-series models for operational global station weather forecasting. *arXiv preprint arXiv:2406.14399*, 2024.

Christoph Jürgen Hemmer and Daniel Durstewitz. True zero-shot inference of dynamical systems preserving long-term statistics. *Advances in Neural Information Processing Systems*, 38:13497–13540, 2026.

Jiaxi Hu, Yuehong Hu, Wei Chen, Ming Jin, Shirui Pan, Qingsong Wen, and Yuxuan Liang. Attractor memory for long-term time series forecasting: A chaos perspective. *Advances in Neural Information Processing Systems*, 37:20786–20818, 2024.

Mason Kamb, Eurika Kaiser, Steven L Brunton, and J Nathan Kutz. Time-delay observables for koopman: Theory and applications. *SIAM Journal on Applied Dynamical Systems*, 19(2):886–917, 2020.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, et al. Toto 2.0: Time series forecasting enters the scaling era. *arXiv preprint arXiv:2605.20119*, 2026.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Akarsh Kumar and Phillip Isola. Pretraining recurrent networks without recurrence. *arXiv preprint arXiv:2606.06479*, 2026.

Jeffrey Lai, Anthony Bao, and William Gilpin. Panda: A pretrained forecast model for chaotic dynamics. In *International Conference on Learning Representations*, volume 2026, pp. 104651–104695, 2026.

Peiyuan Liu, Beiliang Wu, Yifan Hu, Naiqi Li, Tao Dai, Jigang Bao, and Shu-Tao Xia. Timebridge: Non-stationarity matters for long-term time series forecasting. In *International Conference on Machine Learning*, pp. 39815–39840. PMLR, 2025a.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024a.

Yiding Liu, Yifan Hu, Hongjie Xia, Peiyuan Liu, Hongzhou Chen, Xilin Dai, Zewei Dong, and Jiang-Ming Yang. Falcon-x: A time series foundation model for heterogeneous multivariate modeling. *arXiv preprint arXiv:2605.27286*, 2026.

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.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2024b.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024c.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. *arXiv preprint arXiv:2410.04803*, 2025b.

Yihang Lu, Yangyang Xu, Qitao Qin, and Xianwei Meng. Timecapsule: Solving the jigsaw puzzle of long-term time series forecasting with compressed predictive representations. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 1987–1998, 2025.

Yihang Lu, Xianwei Meng, and Enhong Chen. Renf: Rethinking the design of neural long-term time series forecasters. *arXiv preprint arXiv:2509.25914*, 2026.

Abrar Majeedi, Viswanatha Reddy Gajjala, Satya Sai Srinath Namburi Gnvv, Nada Magdi Elkordi, and Yin Li. Lets forecast: Learning embedology for time series forecasting. *arXiv preprint arXiv:2506.06454*, 2025.

Juntong Ni, Zewen Liu, Shiyu Wang, Ming Jin, and Wei Jin. Timedistill: Efficient long-term time series forecasting with mlp via cross-architecture distillation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1113–1124, 2026.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

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.

Jiahong Ouyang, Qingyu Zhao, Ehsan Adeli, Edith V Sullivan, Adolf Pfefferbaum, Greg Zaharchuk, and Kilian M Pohl. Self-supervised longitudinal neighbourhood embedding. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 80–89. Springer, 2021.

Norman H Packard, James P Crutchfield, J Doyne Farmer, and Robert S Shaw. Geometry from a time series. *Physical review letters*, 45(9):712, 1980.

Jeongjin Park, Nicole T Yang, and Nisha Chandramoorthy. When are dynamical systems learned from time series data statistically accurate? *Advances in Neural Information Processing Systems*, 37:43975–44008, 2024.

Louis M Pecora, Thomas L Carroll, and James F Heagy. Statistics for mathematical properties of maps between time series embeddings. *Physical Review E*, 52(4):3420, 1995.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. Duet: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1185–1196, 2025.

Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3): 579–616, 1991.

Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pp. 366–381. Springer, 2006.

James Theiler. Spurious dimension from correlation algorithms applied to limited time-series data. *Physical review A*, 34(3):2427, 1986.

Lucas C Uzal, Guillermo L Grinblat, and Pablo F Verdes. Optimal reconstruction of dynamical systems: A noise amplification approach. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 84(1):016223, 2011.

Banglore Vijay Kumar Vishwas and Sri Ram Macharla. Timesfm: Time series forecasting using decoder-only foundation model. In *Time Series Forecasting Using Generative AI: Leveraging AI for Precision Forecasting*, pp. 195–210. Springer, 2025.

Hao Wang, Licheng Pan, Zhichao Chen, Degui Yang, Sen Zhang, Yifei Yang, Xinggao Liu, Haoxuan Li, and Dacheng Tao. Fredf: Learning to forecast in the frequency domain. *arXiv preprint arXiv:2402.02399*, 2024a.

Rui Wang, Yihe Dong, Sercan Ö Arik, and Rose Yu. Koopman neural forecaster for time series with temporal distribution shifts. *arXiv preprint arXiv:2210.03675*, 2022.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024b.

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. In *International Conference on Learning Representations*, volume 2025, pp. 16980–17016, 2025.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S Yu. From observations to states: Latent time series forecasting. *arXiv preprint arXiv:2602.00297*, 2026.

Charles D Young and Michael D Graham. Deep learning delay coordinate dynamics for chaotic attractors from partial observable data. *Physical Review E*, 107(3):034215, 2023.

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, pp. 11121–11128, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

A Appendix

A.1 Related Work

Neural Time Series Forecasting. Modern end-to-end neural forecasting models have explored diverse architectural inductive biases for modeling temporal dependencies and cross-variate interactions (Zhang & Yan, 2023), including patch-based temporal modeling (Nie et al., 2022), inverted variable-wise attention (Liu et al., 2024c), multi-scale mixing (Wang et al., 2025), and frequency-aware representations (Wang et al., 2024a). More recently, inspired by the tremendous success of advanced LLMs, the community has shifted the focus toward building time-series foundation models that can scale to cross-domain generalization, such as TimesFM (Vishwas & Macharla, 2025), Chronos (Ansari et al., 2025), and Moirai (Liu et al., 2024a), pretrained on large and heterogeneous time-series corpora with time-series-specific tokenization and forecasting objectives, enabling zero- or few-shot prediction on unseen datasets (Khwaja et al., 2026; Liu et al., 2026). While these approaches substantially expand forecasting capacity and generalization, their predictive success does not by itself require an explicit representation of the dynamical state underlying the observed trajectory. This is especially important for time series from natural physical systems and their related applications like weather nowcasting and turbulence forecasting, which often exhibit long-term chaotic behavior.

Dynamics-aware Time Series Forecasting. A complementary line of research in forecasting assumes that observed time series are generated by underlying evolution laws, and argues that neglecting the dynamical system would be a potential bottleneck to the progress of forecasting in the long run Durstewitz et al. (2026). Here we focus specifically on neural forecasters that exploit reconstructed state geometry or learned dynamical representations, rather than the broader literature on physics-informed learning Kovachki et al. (2023); Han et al. (2024). Attraos reconstructs phase-space coordinates using non-parametric phase-space reconstruction and models their evolution through attractor memory and frequency-domain dynamics Hu et al. (2024). DeepEDM instead constructs time-delay embeddings, maps them into a learned latent space, and performs forecasting through differentiable kernel regression and a learned decoder Majeedi et al. (2025). These studies demonstrate the value of connecting delay reconstruction with neural forecasting. However, their initial delay-coordinate construction remains prescribed rather than learned directly from the forecasting objective. Our work instead learns the filters defining the delay-coordinate representation itself, allowing forecasting supervision to determine which local historical projections are useful for prediction. At a larger scale, recent models study generalization directly across dynamical systems. For instance, DynaMix Hemmer & Durstewitz (2026) targets zero-shot dynamical-system reconstruction with an emphasis on long-term statistics. Panda Lai et al. (2026) pretrains on a large synthetic collection of chaotic systems for zero-shot forecasting. These works have also demonstrated a promising ability to recognize and forecast diverse dynamical systems, emphasizing transferable dynamical structure across systems. Our study considers a complementary setting in which a task-trained forecaster learns an explicit predictive coordinate geometry directly from observed time series.

Dynamical Representation Learning. Recent studies have also begun to examine the internal representations learned for temporal prediction. LatentTSF identifies that accurate observation-space forecasts can coexist with temporally disordered latent representations and instead performs prediction in a structured latent state space Yang et al. (2026). From the Koopman perspective, recent work studies how finite-dimensional representations balance temporal coherence, predictive sufficiency, structural simplicity, and expressiveness Cheng et al. (2026). These observations motivate examining not only forecasting accuracy but also the dynamical properties of the learned representation. In this work, we use Koopman linearity as an auxiliary structural constraint, which allows us to examine

how forecasting supervision and dynamical regularization jointly shape the learned convolutional coordinate space.

A.2 Geometry Metrics

Let $\mathbf{z}_i(t)$ denote the learned coordinate sequence of the i-th input window and let $\mathbf{y}_i \in \mathbb{R}^{H\times V}$ denote its true future trajectory. We standardize each dimension of the coordinate representations before computing distances. To avoid selecting highly overlapping windows as neighbors, samples within a temporal exclusion window W are removed from the candidate set.

KNN Future Consistency (KNN-FC). KNN-FC measures whether nearby samples in the learned coordinate space have similar future trajectories. For each sample, we summarize its coordinate sequence as

$$\mathbf{q}_i = [\mathbf{z}_i(T),\, \mathrm{Mean}_t\, \mathbf{z}_i(t),\, \mathrm{Std}_t\, \mathbf{z}_i(t),\, \mathbf{z}_i(T) - \mathbf{z}_i(1)]^\top .$$

Let $\mathcal{N}_K(i)$ be the K nearest valid neighbors of $\mathbf{q}_i$. The normalized future distance is

$$d_F(i,j) = \frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2/(HC)}{\mathrm{Var}(\mathbf{Y}) + \epsilon},$$

where $\mathrm{Var}(\mathbf{Y})$ is computed over all evaluated future values. We then compute

$$D_{\mathrm{KNN}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \frac{1}{|\mathcal{N}_K(i)|} \sum_{j\in\mathcal{N}_K(i)} d_F(i,j).$$

To make the metric comparable across datasets, we normalize it using the same distance computed from K randomly selected valid neighbors, denoted by D_{rand}:

$$\text{KNN-FC} = \frac{D_{\mathrm{KNN}}}{D_{\mathrm{rand}} + \epsilon}.$$

A lower value indicates that nearby coordinate states lead to more similar futures. A value below one means that the learned neighbors are more future-consistent than random neighbors.

Latent Transition Consistency (LTC). LTC measures whether nearby coordinate states follow similar local transition directions. Using chronologically ordered input windows, we define

$$\mathbf{q}_i = [\mathbf{z}_i(T)]\,, \qquad \Delta\mathbf{q}_i = \mathbf{q}_{i+1} - \mathbf{q}_i.$$

For each $\mathbf{q}_i$, let $\mathcal{N}_K(i)$ denote its K nearest valid neighbors. LTC is defined as

$$\mathrm{LTC} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \frac{1}{|\mathcal{N}_K(i)|} \sum_{j\in\mathcal{N}_K(i)} \left(1 - \frac{\Delta\mathbf{q}_i^\top \Delta\mathbf{q}_j}{\|\Delta\mathbf{q}_i\|_2 \|\Delta\mathbf{q}_j\|_2 + \epsilon}\right).$$

A lower LTC means that nearby states have more aligned transition directions, indicating a smoother and more consistent local dynamical flow.

In our experiments, we use $K = 10$ and $W = \max(H, K+1)$. KNN-FC is evaluated on at most $5{,}000$ evenly sampled windows, while LTC uses a contiguous block of at most $5{,}000$ windows to preserve temporal order.

B Additional Experimental Details

B.1 Datasets

We evaluate FCIG on seven public multivariate forecasting datasets and two controlled dynamical systems. ETTh1, ETTh2, ETTm1, and ETTm2 contain seven measurements from electricity transformers, recorded hourly or every 15 minutes. Weather contains 21 meteorological variables recorded every 10 minutes in Germany. Electricity records the hourly electricity consumption of 321 clients, while Traffic contains hourly road occupancy measurements from 862 sensors. We follow the standard chronological splits used in long-term forecasting. Each variable is standardized using statistics computed from the training split.

Table 3: Dataset statistics. Length denotes the number of observations in the raw series. The public benchmarks use the standard long-term forecasting protocol.

Dataset	Length	Variates	Prediction length	Split	Interval	Domain
ETTh1, ETTh2	17,420	7	$\{96, 192, 336, 720\}$	$6:2:2$	1 hour	Electricity
ETTm1, ETTm2	69,680	7	$\{96, 192, 336, 720\}$	$6:2:2$	15 min	Electricity
Weather	52,696	21	$\{96, 192, 336, 720\}$	$7:1:2$	10 min	Environment
Electricity	26,304	321	$\{96, 192, 336, 720\}$	$7:1:2$	1 hour	Electricity
Traffic	17,544	862	$\{96, 192, 336, 720\}$	$7:1:2$	1 hour	Transportation
Lorenz-63	50,000	3	$\{96, 192, 336, 720\}$	$7:1:2$	$\Delta t = 0.01$	Synthetic dynamics
Van der Pol	10,000	2	336	$7:1:2$	$\Delta t = 0.01$	Synthetic dynamics

We additionally use the Lorenz-63 and Van der Pol systems for controlled analysis. Lorenz-63 is generated with $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, and a sampling step of 0.01. Van der Pol is generated with $\mu = 1$ and the same sampling step. Both systems are noise-free and divided chronologically using a $7:1:2$ ratio.

B.2 Model Configurations

Note that we implement the coordinate map with a special temporal convolutional encoder, although the proposed coordinate-level constraint is not tied to this particular convolutional stack. For prediction in the measurement space, we adopt the BDO direct-output structure of ReNF (Lu et al., 2026) to provide a strong forecasting path alongside the auxiliary dynamical constraint.

Model structure. All benchmark configurations use two forecasting blocks following the BDO strategy of ReNF (Lu et al., 2026). The ForeBlock encodes the observed sequence and directly predicts the first $H/2$ future steps. The MidBlock then encodes the observed sequence concatenated with this intermediate forecast and directly predicts all H future steps. This gives the second block access to both the observed history and an initial estimate of its continuation, allowing it to revise the near-term prediction while extending the forecast horizon. The two blocks use separate coordinate encoders, forecasting heads, and transition matrices because they operate on different inputs and need not share the same representation. The transition matrices provide auxiliary training constraints, while the MLP heads generate the forecasts directly.

Convolutional coordinate encoders. The encoders construct coordinates from temporally ordered windows. Each temporal convolution uses left-only padding, so its filters combine the current and preceding steps while preserving sequence length. We pad by repeating the first available value rather than inserting zeros, reducing the artificial boundary jump that zero padding can introduce into the coordinates. This boundary treatment is intended to limit padding effects, rather than remove them entirely.

The ForeBlock uses a deeper encoder to extract patterns over different temporal spans. A 1×1 input projection is followed by L_c residual convolutional layers with kernel size k_1 and dilations $d_1^0, \ldots, d_1^{L_c-1}$. Increasing dilation broadens the temporal range covered by the filters without increasing their kernel sizes. Each layer combines convolution, GELU, normalization, and a residual connection, allowing it to add temporal features while retaining the incoming representation. A final linear 1×1 projection produces c_1 coordinates per stream without an additional activation.

The MidBlock uses a shallower encoder to build a new representation of the history and intermediate forecast without repeating the full convolutional stack. It applies one undilated, left-padded temporal convolution with kernel size k_2. Thus, L_c and d_1 in Table 4 describe only the ForeBlock encoder.

Koopman initialization. The ForeBlock Koopman matrix is initialized orthogonally, so its initial linear mapping preserves vector norms and does not start by amplifying or shrinking the coordinates. The MidBlock matrix uses Xavier uniform initialization, providing a different starting point for the representation built from the extended input.

Table 4: Main hyperparameters of FCIG-MLP. All configurations use two forecasting blocks ($N = 2$). Each row corresponds to one prediction length H.

Dataset	H	T	$\mathbf{c}$	$\mathbf{k}$	d_1	L_c	d_{ff}	α	λ_1
ETTh1	96	512	$(64, 64)$	$(3, 32)$	2	4	128	0	0.7
	192	512	$(64, 64)$	$(3, 5)$	2	4	128	0	0.8
	336	512	$(64, 64)$	$(3, 1)$	2	4	256	0	0.8
	720	336	$(64, 64)$	$(3, 1)$	2	4	256	0	0.7
ETTh2	96	512	$(64, 64)$	$(3, 7)$	2	4	512	0	0.5
	192	512	$(64, 64)$	$(3, 15)$	2	4	512	0	0.5
	336	512	$(64, 64)$	$(3, 24)$	2	4	512	0	0.5
	720	512	$(64, 64)$	$(3, 24)$	2	4	512	0	0.5
ETTm1	96	720	$(16, 16)$	$(3, 7)$	2	4	512	0	0.9
	192	720	$(16, 16)$	$(3, 7)$	2	4	512	0	0.9
	336	720	$(16, 16)$	$(3, 7)$	2	4	512	0	0.9
	720	720	$(16, 16)$	$(3, 15)$	2	4	512	0	0.9
ETTm2	96	512	$(16, 16)$	$(3, 7)$	2	4	64	0	0.9
	192	512	$(16, 16)$	$(7, 12)$	2	4	256	0	0.9
	336	512	$(16, 16)$	$(7, 15)$	2	4	256	0.2	0.9
	720	512	$(16, 16)$	$(3, 15)$	2	4	256	0.8	0.9
Weather	96	720	$(64, 64)$	$(12, 16)$	2	4	64	0.5	0.9
	192	720	$(64, 64)$	$(12, 16)$	2	4	64	0.1	0.9
	336	720	$(64, 64)$	$(12, 16)$	2	4	64	0.1	0.9
	720	720	$(64, 64)$	$(12, 16)$	2	4	512	0.1	0.9
Electricity	96	512	$(12, 12)$	$(1, 5)$	1	2	1024	0.7	0.9
	192	512	$(12, 12)$	$(1, 5)$	1	2	1024	0.7	0.9
	336	512	$(12, 12)$	$(1, 5)$	1	2	1024	0.7	0.9
	720	512	$(12, 12)$	$(1, 5)$	1	2	1024	0.7	0.9
Traffic	96	512	$(8, 8)$	$(3, 24)$	2	2	1024	0.2	0.1
	192	512	$(8, 8)$	$(3, 24)$	2	4	1024	0.2	0.1
	336	512	$(8, 8)$	$(3, 24)$	2	4	1024	0.2	0.1
	720	512	$(8, 8)$	$(3, 24)$	2	4	1024	0.1	0.3

B.3 Training Objective Details

For an N-block BDO predictor, let $\widehat{Y}^{(j)}$ denote the output of the j-th forecasting head, whose prediction length is $H_j = jH/N$. When frequency-domain supervision is enabled, the forecasting loss of each head is

$$\mathcal{L}_{\text{fore}}^{(j)} = (1 - \alpha)\mathcal{L}_{\text{time}}^{(j)} + \alpha\mathcal{L}_{\text{freq}}^{(j)}, \tag{20}$$

where α controls the frequency-domain component.

Following the definitions in the main text, the complete BDO objective is

$$\mathcal{L}_{\text{total}} = \sum_{j=1}^{N} \frac{1}{j} \left[(1 - \lambda_1)\mathcal{L}_{\text{fore}}^{(j)} + \lambda_1\mathcal{L}_{\text{diff}}^{(j)}\right] + \lambda_2 w_e \sum_{j=1}^{N} \mathcal{L}_{\text{K}}^{(j)}. \tag{21}$$

Here, λ_1 controls the temporal-difference loss, λ_2 controls the Koopman regularization, and w_e is an optional warm-up factor. We default to using $\lambda_2 = 0.05$ in all main experiments.

Table 5: Long-term forecasting results of competing methods under model-specific hyperparameter selection. The look-back window is selected from $\{336, 512, 720\}$, except that Timer-XL uses a window length of 672 following its original paper. Avg. reports the mean over prediction lengths $\{96, 192, 336, 720\}$. Lower values are better. The **best** and second-best results are highlighted; ties receive the same rank.

Models		FCIG ours		ReNF (2026)		TimeBridge (2025a)		DUET (2025)		TimeDistill (2026)		Timer-XL (2025b)		iTransformer (2024b)		TimeMixer (2024b)		PatchTST (2022)		Crossformer (2023)		DLinear (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.342**	**0.375**	0.350	0.383	0.355	0.391	0.353	0.386	0.373	0.401	0.364	0.397	0.386	0.405	0.372	0.401	0.377	0.397	0.411	0.435	0.379	0.403
	192	**0.380**	**0.398**	0.385	0.408	0.389	0.414	0.398	0.409	0.411	0.426	0.405	0.424	0.424	0.440	0.413	0.429	0.409	0.425	0.409	0.438	0.408	0.419
	336	**0.396**	**0.409**	0.405	0.425	0.415	0.435	0.415	0.428	0.439	0.444	0.427	0.439	0.449	0.460	0.438	0.450	0.431	0.444	0.433	0.457	0.440	0.440
	720	**0.405**	**0.429**	0.422	0.449	0.443	0.462	0.436	0.458	0.495	0.493	0.439	0.459	0.495	0.487	0.486	0.484	0.457	0.477	0.501	0.514	0.471	0.493
	Avg.	**0.381**	**0.403**	0.391	0.416	0.401	0.426	0.401	0.420	0.430	0.441	0.409	0.430	0.439	0.448	0.427	0.441	0.419	0.436	0.439	0.461	0.425	0.439
ETTh2	96	**0.255**	**0.322**	0.261	0.329	0.270	0.331	0.271	0.335	0.273	0.336	0.277	0.343	0.297	0.348	0.281	0.351	0.274	0.337	0.728	0.603	0.300	0.364
	192	**0.310**	**0.363**	0.320	0.370	0.338	0.375	0.335	0.376	0.334	0.381	0.348	0.391	0.372	0.403	0.349	0.387	0.348	0.384	0.723	0.607	0.387	0.423
	336	**0.330**	**0.382**	0.346	0.394	0.370	0.402	0.354	0.398	0.363	0.415	0.375	0.418	0.388	0.418	0.366	0.413	0.377	0.416	0.740	0.628	0.490	0.487
	720	**0.357**	**0.403**	0.381	0.423	0.402	0.434	0.384	0.426	0.408	0.446	0.409	0.458	0.424	0.444	0.401	0.436	0.406	0.441	1.386	0.882	0.704	0.597
	Avg.	**0.313**	**0.367**	0.327	0.379	0.345	0.386	0.336	0.384	0.345	0.395	0.352	0.402	0.370	0.403	0.349	0.397	0.351	0.395	0.894	0.680	0.470	0.468
ETTm1	96	**0.267**	**0.321**	0.270	0.325	0.288	0.339	0.279	0.333	0.285	0.344	0.290	0.341	0.300	0.353	0.293	0.345	0.289	0.342	0.314	0.367	0.300	0.345
	192	**0.310**	**0.349**	**0.310**	0.352	0.326	0.368	0.320	0.358	0.331	0.368	0.337	0.369	0.341	0.380	0.335	0.372	0.329	0.368	0.374	0.410	0.336	0.366
	336	**0.336**	**0.369**	0.343	0.373	0.363	0.394	0.348	0.377	0.359	0.386	0.374	0.393	0.374	0.396	0.368	0.386	0.362	0.390	0.413	0.432	0.367	0.387
	720	**0.395**	**0.401**	0.400	0.405	0.417	0.419	0.405	0.408	0.415	0.416	0.437	0.428	0.429	0.430	0.426	0.417	0.416	0.423	0.753	0.613	0.419	0.417
	Avg.	**0.327**	**0.360**	0.331	0.364	0.349	0.380	0.338	0.369	0.348	0.380	0.359	0.382	0.361	0.390	0.356	0.380	0.349	0.381	0.464	0.456	0.356	0.379
ETTm2	96	**0.156**	**0.240**	0.157	0.241	0.157	0.243	0.162	0.249	0.163	0.255	0.175	0.257	0.175	0.266	0.165	0.256	0.165	0.255	0.296	0.391	0.164	0.256
	192	0.213	0.280	**0.212**	**0.279**	0.218	0.284	0.215	0.288	0.220	0.294	0.242	0.301	0.242	0.312	0.225	0.298	0.221	0.293	0.369	0.416	0.224	0.304
	336	**0.259**	**0.313**	0.262	0.315	0.270	0.321	0.267	0.321	0.269	0.328	0.293	0.337	0.282	0.337	0.277	0.332	0.276	0.327	0.588	0.600	0.277	0.337
	720	**0.338**	**0.368**	0.341	**0.368**	0.344	0.372	0.348	0.373	0.346	0.369	0.376	0.390	0.375	0.394	0.360	0.385	0.362	0.381	0.750	0.612	0.371	0.401
	Avg.	**0.242**	**0.300**	0.243	0.301	0.247	0.305	0.248	0.308	0.250	0.312	0.271	0.322	0.269	0.327	0.257	0.318	0.256	0.314	0.501	0.505	0.259	0.325
Weather	96	**0.135**	**0.176**	0.138	0.180	0.144	0.184	0.146	0.191	0.145	0.204	0.157	0.205	0.157	0.206	0.147	0.198	0.150	0.200	0.143	0.210	0.170	0.230
	192	**0.179**	**0.220**	0.182	0.224	0.186	0.226	0.188	0.231	0.188	0.247	0.206	0.250	0.200	0.248	0.192	0.242	0.191	0.239	0.195	0.261	0.216	0.273
	336	**0.230**	**0.263**	0.231	0.266	0.237	0.267	0.235	0.269	0.240	0.286	0.259	0.291	0.252	0.287	0.247	0.284	0.242	0.279	0.254	0.319	0.258	0.307
	720	**0.299**	**0.313**	0.304	0.318	0.312	0.321	0.308	0.319	0.310	0.338	0.337	0.344	0.320	0.336	0.318	0.330	0.312	0.330	0.335	0.385	0.323	0.362
	Avg.	**0.211**	**0.243**	0.214	0.247	0.220	0.250	0.219	0.253	0.221	0.269	0.240	0.273	0.232	0.269	0.226	0.264	0.224	0.262	0.232	0.294	0.242	0.293
Electricity	96	0.120	0.212	**0.118**	**0.210**	0.122	0.216	0.128	0.219	0.128	0.225	0.127	0.219	0.134	0.230	0.153	0.256	0.143	0.247	0.134	0.231	0.140	0.237
	192	**0.138**	**0.229**	**0.138**	**0.229**	0.143	0.238	0.145	0.235	0.145	0.241	0.145	0.236	0.154	0.250	0.168	0.269	0.158	0.261	0.146	0.243	0.154	0.251
	336	**0.150**	**0.244**	0.151	**0.244**	0.163	0.258	0.163	0.255	0.161	0.258	0.159	0.252	0.169	0.265	0.189	0.291	0.168	0.267	0.165	0.264	0.169	0.268
	720	**0.170**	**0.265**	0.173	0.266	0.178	0.274	0.193	0.281	0.195	0.291	0.187	0.277	0.194	0.288	0.228	0.320	0.214	0.307	0.237	0.314	0.204	0.301
	Avg.	**0.145**	**0.237**	**0.145**	**0.237**	0.152	0.247	0.157	0.248	0.157	0.254	0.155	0.246	0.163	0.258	0.185	0.284	0.171	0.271	0.171	0.263	0.167	0.264
Traffic	96	0.341	0.228	0.335	**0.226**	**0.332**	0.237	0.360	0.238	0.358	0.256	0.340	0.238	0.358	0.258	0.369	0.257	0.370	0.262	0.526	0.288	0.395	0.275
	192	0.364	0.243	0.356	**0.239**	**0.343**	**0.239**	0.383	0.249	0.374	0.264	0.360	0.248	0.382	0.271	0.399	0.272	0.386	0.269	0.503	0.263	0.407	0.280
	336	0.375	0.250	0.366	**0.246**	**0.360**	0.249	0.395	0.259	0.389	0.271	0.377	0.256	0.396	0.277	0.407	0.272	0.396	0.275	0.505	0.276	0.417	0.286
	720	0.413	0.269	0.402	**0.267**	**0.392**	0.268	0.435	0.278	0.428	0.292	0.418	0.279	0.445	0.308	0.461	0.316	0.435	0.295	0.552	0.301	0.454	0.308
	Avg.	0.373	0.248	0.365	**0.245**	**0.357**	0.248	0.393	0.256	0.387	0.271	0.374	0.255	0.395	0.279	0.409	0.279	0.397	0.275	0.522	0.282	0.418	0.287

C FULL FORECASTING RESULTS

Table 6: Multivariate forecasting results against dynamics-oriented forecasters under the DeepEDM protocol;results are taken directly from (Majeedi et al., 2025). The forecast horizon is $H \in \{48, 96, 144, 192\}$, and the lookback length is set to $T = 2H$. Avg. reports the mean over the four forecast horizons. Lower values are better. The **best** and second-best results are highlighted; ties receive the same rank. Baseline results are taken from DeepEDM (Majeedi et al., 2025).

Models		FCIG		DeepEDM		Koopa		KNF		Attraos	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	48	0.329	0.359	**0.324**	**0.357**	0.336	0.377	0.876	0.709	0.341	0.371
	96	**0.359**	**0.382**	0.365	0.384	0.371	0.405	0.975	0.744	0.387	0.402
	144	**0.376**	**0.392**	0.388	0.398	0.405	0.418	0.801	0.662	0.415	0.422
	192	**0.384**	**0.400**	0.407	0.421	0.416	0.429	0.941	0.744	0.429	0.434
	Avg.	**0.362**	**0.383**	0.371	0.390	0.382	0.407	0.898	0.715	0.393	0.407
ETTh2	48	**0.223**	0.290	0.225	**0.288**	0.226	0.300	0.385	0.376	0.230	0.301
	96	**0.273**	**0.330**	0.289	0.333	0.297	0.349	0.433	0.446	0.302	0.350
	144	**0.308**	**0.356**	0.324	0.362	0.333	0.381	0.441	0.456	0.355	0.383
	192	**0.316**	**0.366**	0.351	0.377	0.356	0.393	0.528	0.503	0.373	0.399
	Avg.	**0.280**	**0.335**	0.297	0.340	0.303	0.356	0.447	0.445	0.315	0.358
ETTm1	48	**0.277**	**0.317**	**0.277**	0.318	0.283	0.333	1.026	0.792	0.312	0.353
	96	**0.282**	**0.324**	0.288	0.328	0.294	0.345	0.957	0.782	0.314	0.355
	144	**0.301**	**0.339**	0.308	0.344	0.322	0.366	0.921	0.760	0.332	0.368
	192	**0.314**	**0.348**	0.322	0.353	0.337	0.378	0.896	0.731	0.349	0.378
	Avg.	**0.294**	**0.332**	0.299	0.336	0.309	0.356	0.950	0.766	0.327	0.364
ETTm2	48	**0.130**	**0.218**	0.133	0.221	0.134	0.226	0.621	0.623	0.139	0.236
	96	**0.164**	**0.244**	0.169	0.248	0.171	0.254	1.535	1.012	0.174	0.259
	144	**0.199**	**0.267**	0.203	0.271	0.206	0.280	1.337	0.876	0.209	0.284
	192	**0.219**	**0.282**	0.224	0.289	0.226	0.298	1.355	0.908	0.233	0.302
	Avg.	**0.178**	**0.253**	0.182	0.257	0.184	0.265	1.212	0.855	0.189	0.270
Electricity	48	**0.122**	**0.215**	0.161	0.247	0.130	0.234	0.175	0.265	0.192	0.268
	96	**0.126**	**0.218**	0.137	0.232	0.136	0.236	0.198	0.284	0.150	0.244
	144	**0.135**	**0.225**	0.145	0.239	0.149	0.247	0.204	0.297	0.151	0.246
	192	**0.141**	**0.231**	0.151	0.244	0.156	0.254	0.245	0.321	0.154	0.249
	Avg.	**0.131**	**0.222**	0.149	0.240	0.143	0.243	0.205	0.292	0.162	0.252
Traffic	48	**0.391**	**0.241**	0.448	0.286	0.415	0.274	0.621	0.382	0.612	0.396
	96	0.387	**0.254**	**0.383**	0.259	0.401	0.275	0.645	0.376	0.439	0.300
	144	**0.364**	**0.240**	0.380	0.258	0.397	0.276	0.683	0.402	0.423	0.294
	192	**0.378**	**0.251**	0.387	0.262	0.403	0.284	0.699	0.405	0.421	0.295
	Avg.	**0.380**	**0.247**	0.399	0.266	0.404	0.277	0.662	0.391	0.474	0.321
Weather	48	0.130	**0.165**	0.138	0.168	**0.126**	0.168	0.201	0.288	0.149	0.191
	96	**0.150**	**0.190**	0.157	0.192	0.154	0.205	0.295	0.308	0.168	0.214
	144	**0.167**	**0.208**	0.174	0.210	0.172	0.225	0.394	0.401	0.184	0.231
	192	**0.184**	**0.225**	0.191	0.226	0.193	0.241	0.462	0.437	0.202	0.249
	Avg.	**0.158**	**0.197**	0.165	0.199	0.161	0.210	0.338	0.358	0.176	0.221
1st Count		51		5		1		0		0	

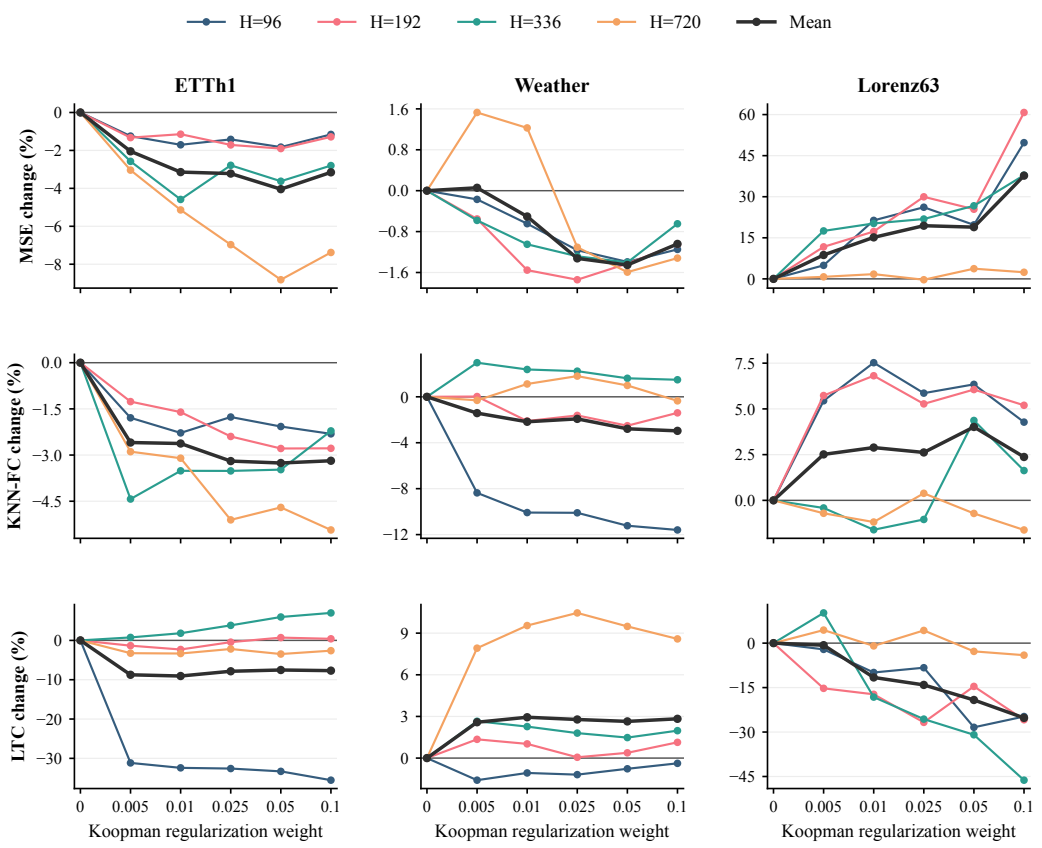


Figure 6: Sensitivity to the Koopman regularization weight on ETTh1, Weather, and Lorenz63. Values show percentage changes relative to $\lambda_2 = 0$.

D More Analysis

D.1 Sensitivity of Koopman Regularization Weight

We use a Koopman regularization weight of $\lambda_2 = 0.05$ in all main experiments. In this section, we vary the Koopman regularization weight $\lambda_2 \in \{0, 0.005, 0.01, 0.025, 0.05, 0.1\}$ on ETTh1, Weather, and Lorenz63. Figure 6 reports the percentage change from $\lambda_2 = 0$ at each forecast horizon. Each point averages three seeds, and the black curve averages the changes across four horizons.

On ETTh1, every nonzero weight reduces the mean forecasting MSE. The largest reduction is 4.0% at $\lambda_2 = 0.05$, where the coordinate KNN-FC ratio and LTC also decrease. Weather shows a smaller MSE reduction of 1.5% at the same weight: its KNN-FC ratio decreases, but LTC rises slightly. Lorenz63 shows a different trade-off. At $\lambda_2 = 0.05$, LTC decreases by 19.2%, while forecasting MSE increases by 18.9%. These results show that a lower dynamical metric does not necessarily lead to a more accurate forecast; the optimal regularization weight varies and should balance the two objectives.

D.2 More demonstrations of CICS

Figures 11, 12, and 13 show further examples of the learned CICS across datasets. In each row, the left plot projects the coordinates from a representative input window onto the first three principal components, with color indicating relative time. The right plot shows selected coordinate channels from the same window in their original dimensions, without PCA. Both ForeBlock and last MidBlock states are shown where available. For display, the real-world traces use a five-point moving average, while the synthetic-system traces are unsmoothed.

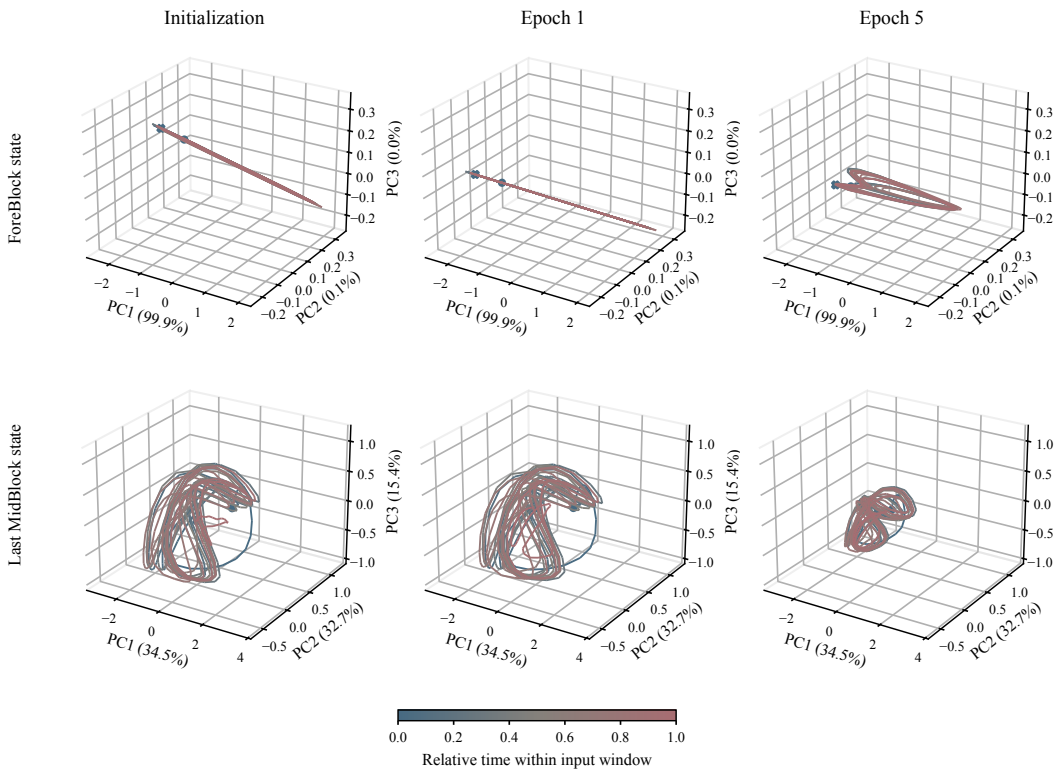


Figure 7: Illustations of the learning process of Traffic's CICS in different training stages (indicated by training epochs). Rows show the ForeBlock and Last MidBlock states under a shared PCA basis and axis limits.

The coordinate traces in these figures provide another view of the learned CICS. Across the real-world datasets, both blocks retain recurring patterns, but they emphasize different temporal components. The ForeBlock often shows broader variations, while the last MidBlock makes finer oscillations and local changes more visible. For example, ETTh1 and Electricity retain a clear main cycle in both blocks, despite differences in their finer fluctuations. This suggests complementary representations rather than a strict separation into low- and high-frequency bands.

D.3 Evolution of Learned Latent States

We examine the learned coordinates during training on Traffic and across two controlled comparisons on Weather. Within each dataset, **we use the same test windows and a shared PCA basis with fixed axis limits**. For Weather, the independently trained states are also aligned to remove differences in rotation and translation, while preserving their scales. These projections allow us to compare the same directions, although differences outside the displayed dimensions may remain hidden.

Changes during training. Figure 7 shows the Traffic states at initialization, Epoch 1, and Epoch 5. The Last MidBlock trajectories are similar at initialization and Epoch 1. By Epoch 5, they occupy a smaller region while retaining a broadly similar curved and looping structure. The clearest change is therefore a reduction in scale, rather than the appearance of an entirely new shape. Meanwhile, the online test MSE decreases from 0.4588 at Epoch 1 to 0.3636 at Epoch 5. The ForeBlock does not show the same contraction, suggesting that the two blocks adjust their representations differently during training.

Effects of convolution and KAR. Figure 8 compares the Weather configurations. With the main convolution unchanged, removing Koopman autoregressive regularization (KAR) greatly increases the trajectory extent in both blocks, while test MSE rises from 0.1354 to 0.1378. The scale difference

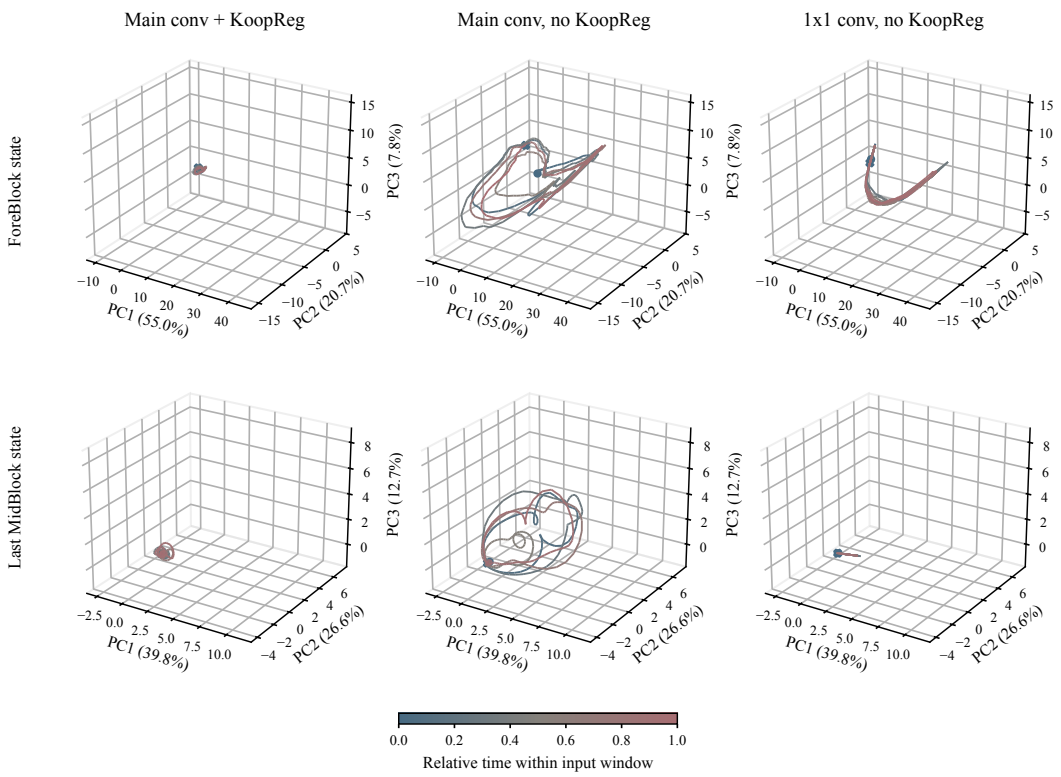


Figure 8: Illustrations of the learned CICS of Weather under different dynamical configs. Rows show the ForeBlock and Last MidBlock states under a shared PCA basis and axis limits.

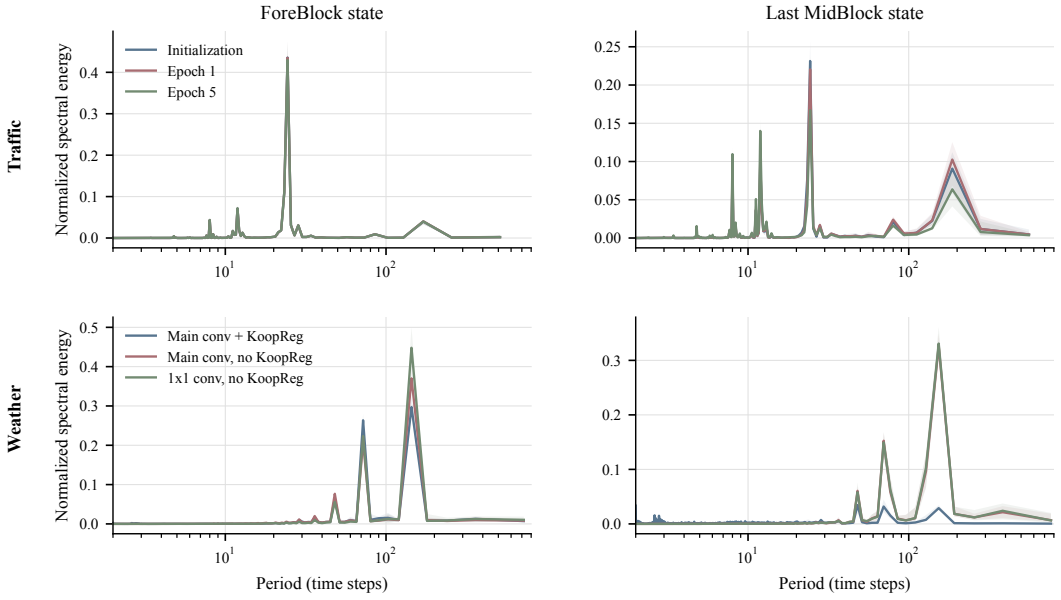


Figure 9: The upper panels show the variations of the frequency spectrum of Traffic along the training process, while the bottom panels show this under different dynamical configs.

makes the regularized trajectories appear very small under the shared axes, so this figure alone cannot establish whether their detailed shapes are better preserved. With KAR disabled, replacing the main convolution with 1×1 kernels produces a narrow Last MidBlock projection but further increases

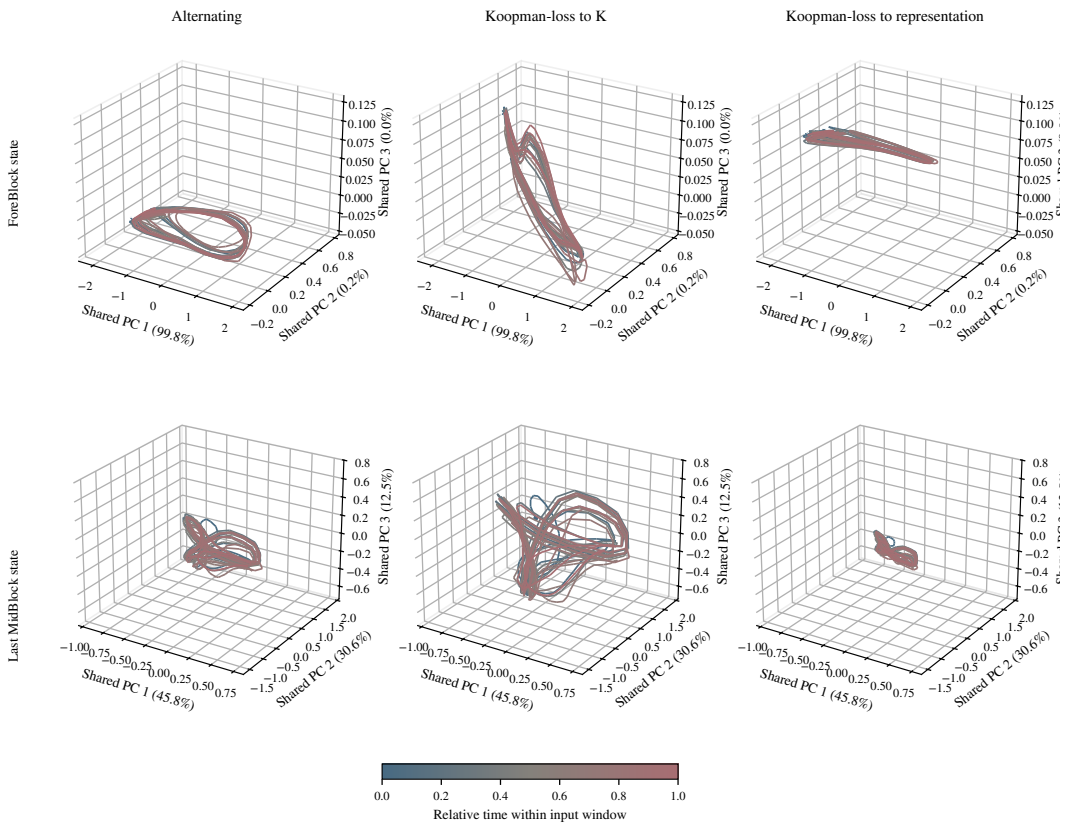


Figure 10: Effect of alternating optimization on Traffic latent states. Columns compare alternating updates with operator-update-only and coordinate-update-only training. Rows show the ForeBlock and Last MidBlock states under a shared PCA basis and axis limits.

MSE to 0.1427. A smaller or simpler projection therefore does not necessarily indicate a better forecasting representation.

Changes beyond overall scale. Figure 9 complements the projections with normalized spectra. On Traffic, the ForeBlock spectra nearly overlap. The Last MidBlock retains its main peak near 24 time steps, but Epoch 5 gives more relative weight to shorter periods around 8–12 steps and less to some longer periods. On Weather, the main spectral peaks remain at similar locations in the ForeBlock, whereas KAR reduces several prominent long-period peaks in the Last MidBlock. Because the spectra are normalized, these differences cannot be explained by uniformly shrinking the coordinates. They show that training and KAR also change the relative contributions of different temporal patterns.

Together, the figures show changes in both coordinate scale and frequency content. These changes accompany improved forecasting in the tested settings, but neither a compact trajectory nor a weaker spectral peak alone establishes more accurate learned dynamics.

Effect of alternating optimization. We compare the alternating gradient scheme in Section 3.3 with using either update mode alone. In operator-update-only training, KAR updates the Koopman matrix and the encoder through the source state, while the target state is detached. In coordinate-update-only training, KAR updates the encoder while keeping the Koopman matrix fixed. Forecasting supervision remains active in all settings.

Figure 10 shows the clearest difference in the Last MidBlock. Operator-update-only training produces the largest trajectories, whereas coordinate-update-only training produces the smallest. Alternating updates give an intermediate scale while retaining a broadly similar looping structure. This compari-

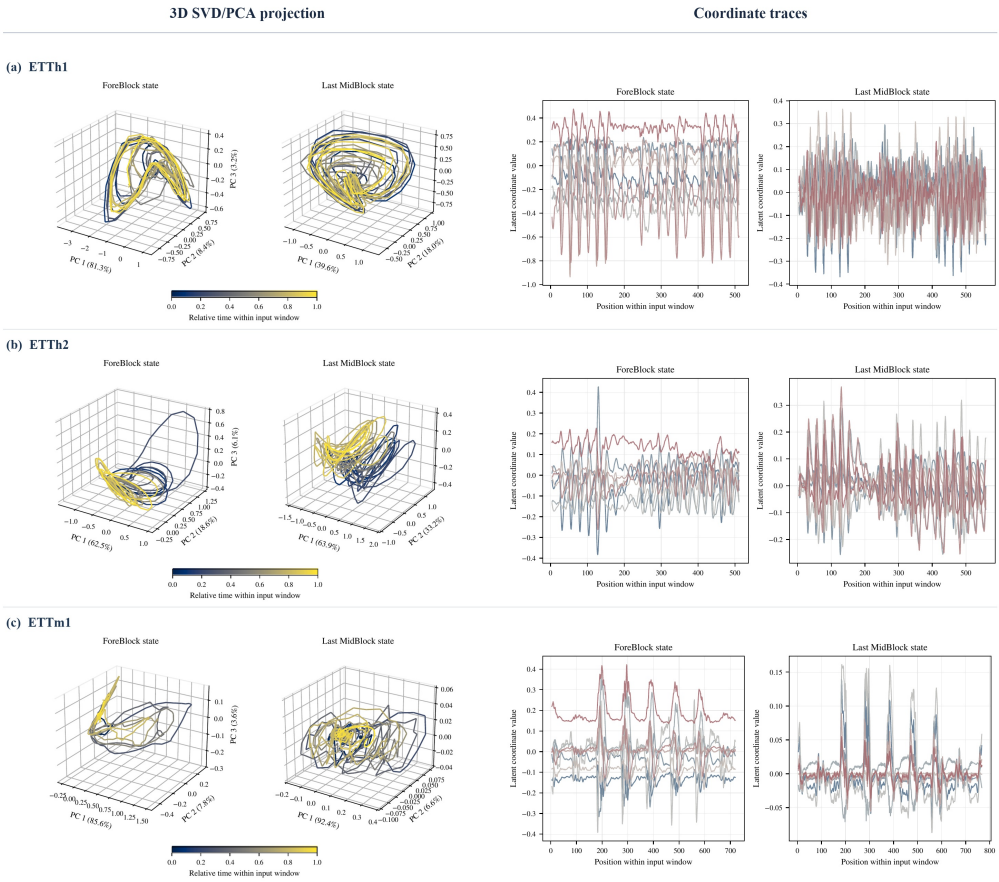


Figure 11: Representative coordinate projections and traces on ETTh1, ETTh2, and ETTm1.

son mainly shows differences in coordinate scale: smaller coordinates can reduce the unnormalized KAR loss, but a smaller projection alone does not indicate better dynamics or forecasting accuracy.

E LIMITATIONS

Dynamical priors do not necessarily benefit forecasting across all real-world time series. Our empirical observations suggest that FCIG can favor a periodic shortcut, relying on recurring patterns rather than fully capturing more complex temporal changes. This tendency may be useful when stable periodic structure dominates, but less effective when prediction depends on irregular or changing patterns. These observations suggest that the benefits of the proposed prior depend on the temporal structure of the data. Understanding when periodic patterns provide sufficient predictive information, and when additional modeling is needed, remains an important direction for future work.

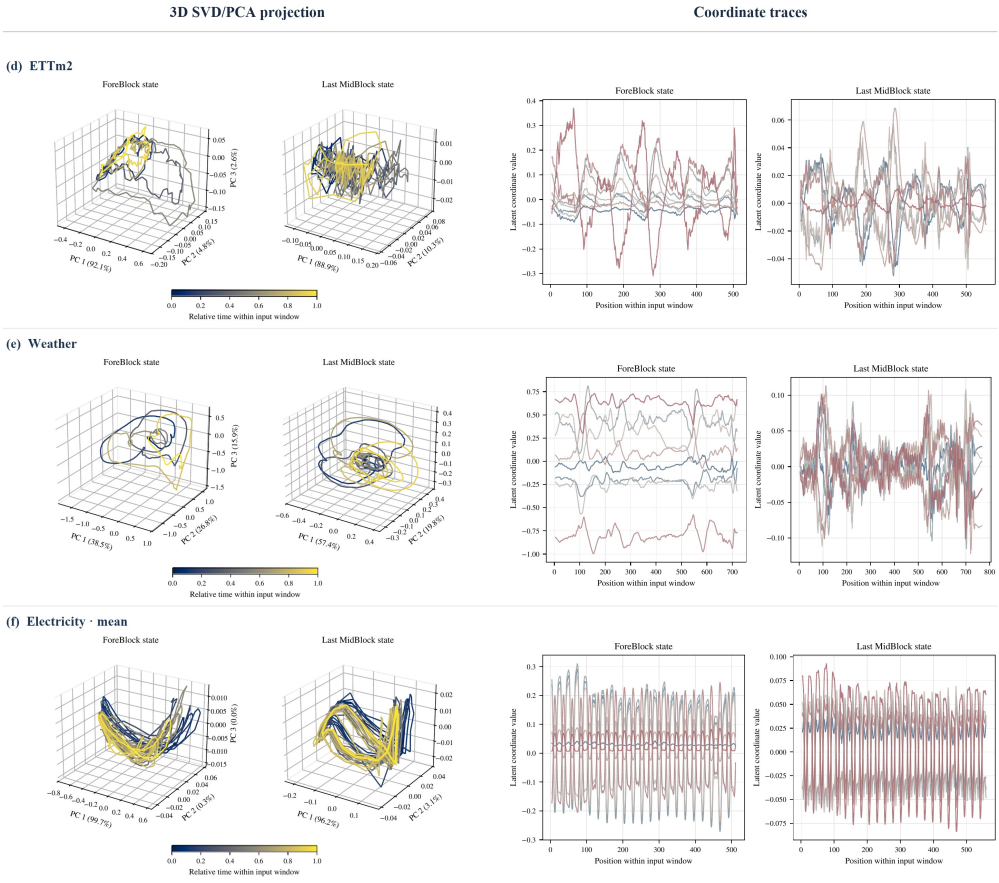


Figure 12: Representative coordinate projections and traces on ETTm2, Weather, and Electricity.

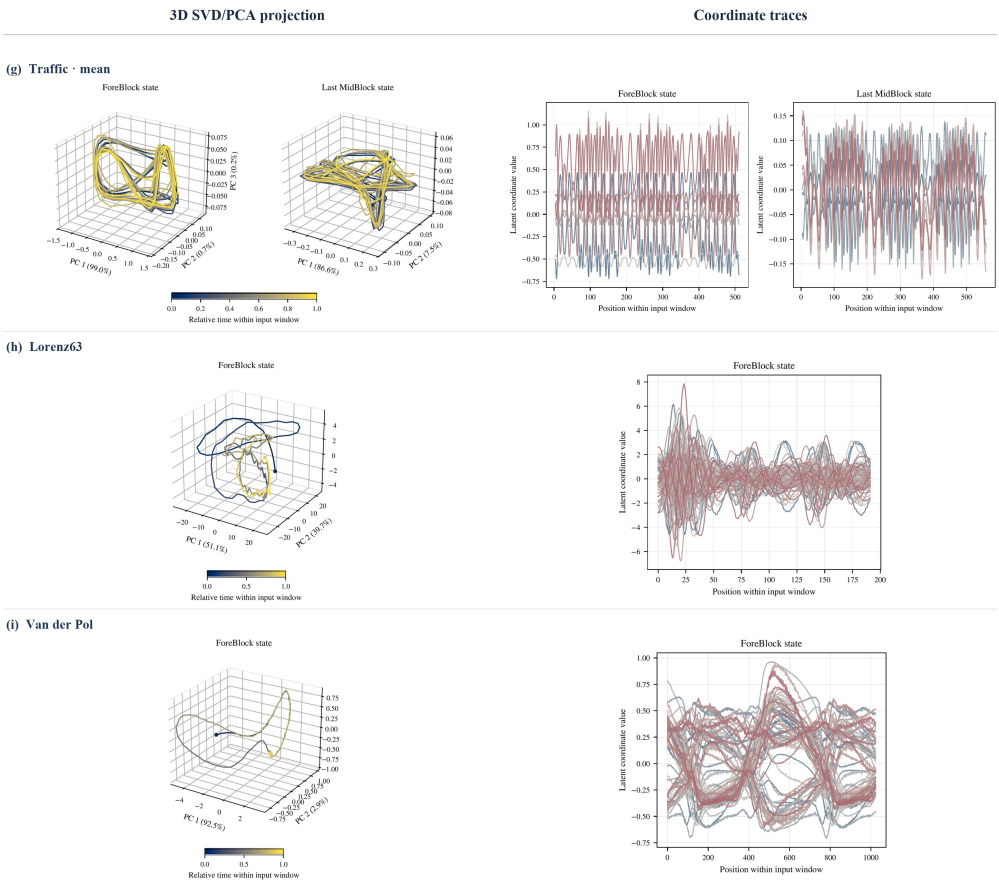


Figure 13: Representative coordinate projections and traces on Traffic, Lorenz63 (input192-horizon96), and Van der Pol.