PRINCIPLED KOOPMAN REPRESENTATIONS WITH KALMAN INFERENCE FOR EFFICIENT TIME-SERIES PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The Koopman operator has been widely used for time-series prediction in dynamical systems. However, prior work that learns latent "Koopman spaces" using neural networks often does not construct a representation consistent with the operator-theoretic formulation or expose the intrinsic structure of the dynamics. We introduce K^{2}SVD, a point-forecasting method that learns leading singular functions of the Koopman operator by optimizing a Hilbert-Schmidt objective. The resulting compact representation supports an interpretable low-dimensional linear rollout. K^{2}SVD then refines that rollout with a lightweight Kalman-style latent correction and decodes the corrected states into multi-step point forecasts. Across real-world and dynamical-system benchmarks, K^{2}SVD achieves competitive forecasting accuracy with substantially shorter inference time and lower computational cost than the evaluated efficiency-focused models.

1 INTRODUCTION

Time-series forecasting is a fundamental problem across a wide range of scientific and industrial domains, often requiring models to handle highly nonlinear transformations and accumulated long-horizon error. Recent deep learning models, including transformers Li et al. (2019), sequence-to-sequence architectures, and foundation models Das et al. (2024); Ansari et al. (2024), have achieved strong empirical performance. However, in domains such as economics, weather, and energy, where explainable and expressive latent spaces are crucial, purely black-box prediction models are often insufficient.

Koopman operator Lan and Mezić (2013) offers a compelling alternative to black-box models by providing a mathematically grounded and expressive latent space. By lifting data into an infinite-dimensional space of measurement functions, the Koopman framework represents nonlinear dynamics through a linear operator that maps measurements to their next-step evolution. This perspective also enables spectral methods to analyze the widely observed low-rank structure in system dynamics, bridging classical dynamical systems theory and modern machine learning. In particular, recent Koopman singular value decomposition (SVD)-based methods Jeong et al. (2025); Kostic et al. (2024); Wu and Noé (2020) learn the leading singular functions of the operator, demonstrating the effectiveness of Koopman representations for modeling dynamical systems.

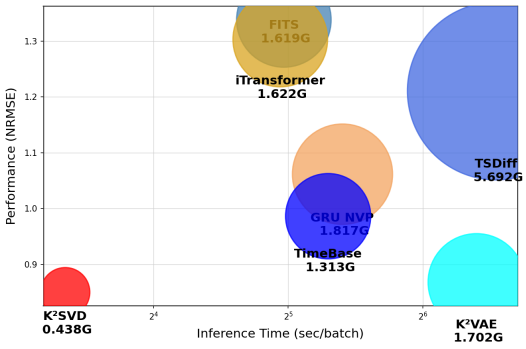


Figure 1: Comparison of predictive performance, GPU usage, and inference time on the Electricity dataset. K^{2}SVD outperforms competing methods in time and space efficiency, as well as overall performance. Circle size indicates peak allocated GPU memory during training.

Despite these advances, many Koopman-based time-series forecasting methods Liu et al. (2026; 2023) rely on heuristic or loosely defined latent representations. Their embeddings are commonly optimized primarily for reconstruction or forecast

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.

error rather than for recovering the latent structure. Consequently, the learned "Koopman spaces" may not admit a clear operator interpretation and can introduce redundant latent dimensions and costly prediction modules.

To address these limitations, we propose K^{2}SVD, a principled framework that explicitly enforces a low-rank Koopman structure by learning approximations to its leading singular functions. Rather than treating the latent space as an unconstrained embedding, our method constructs a theoretically grounded representation by optimizing a Hilbert-Schmidt objective that directly targets the best rank-k approximation of the operator. This yields a latent space with well-defined linear dynamics, enabling faithful spectral characterization and linear temporal evolution, which also reduces the computation dramatically. In the Koopman latent space, temporal evolution is first represented by a compact linear rollout. Because a static rollout can accumulate approximation error over a long horizon, we apply a trainable Kalman-style correction that deterministically combines the recurrent latent forecast with a Koopman-derived reference trajectory before decoding the filtered point forecast.

The overall model is trained using a two-stage scheme that first establishes the Koopman subspace and then refines the transition and correction components for point forecasting in an end-to-end manner. This design combines theoretical rigor with practical scalability, yielding a model that captures low-rank dynamics while maintaining strong forecasting performance and parameter efficiency. As shown in Figure 1, K^{2}SVD achieves competitive prediction performance with substantially low GPU memory usage and shorter inference time than existing methods. Our contribution is as follows:

- We propose K^{2}SVD, a principled Koopman forecasting framework that learns a low-rank Koopman space by approximating the leading singular functions of the Koopman operator through a Hilbert-Schmidt objective, yielding a mathematically grounded latent space.
- We integrate the learned Koopman representation with a Kalman-style latent correction, enabling efficient multi-step point forecasting while limiting rollout error.
- We empirically show that K^{2}SVD achieves competitive forecasting performance across multiple dynamic systems and real-world time-series benchmarks while requiring substantially fewer computational resources and shorter inference time than existing methods, demonstrating the benefit of explicitly modeling low-rank Koopman structure.

2 Related Works

Time-series forecasting. Time-series forecasting has evolved from classical autoregressive models to neural sequence models. Architectures such as N-BEATS Oreshkin et al. (2020) and transformer-based forecasters learn flexible representations for multi-horizon point prediction Lim et al. (2021); Zhou et al. (2021); Rasul et al. (2021). More recent work incorporates structured dynamical priors to improve long-horizon accuracy, efficiency, and interpretability.

Koopman operator learning. The Koopman operator framework provides a principled approach to analyzing nonlinear dynamical systems by lifting them into a linear space. Classical works establish their spectral foundations and connections to ergodic theory and modal decomposition Mezić (2005; 2013); Rowley et al. (2009). Building on this theory, data-driven approximations such as dynamic mode decomposition (DMD) and its extensions enable finite-dimensional representations of the Koopman operator directly from data Schmid (2010); Tu (2013); Williams et al. (2015a;b). In particular, low-rank approximations based on SVD play a central role in extracting dominant Koopman modes, and more recently, learning-based approaches leverage neural networks to recover Koopman subspaces Wu and Noé (2020); Mardt et al. (2018); Kostic et al. (2024). These methods highlight the importance of low-rank Koopman representations and SVD approximations for high-dimensional dynamical systems.

Koopman-based forecasting. Koopman-based models use lifted linear dynamics to forecast nonlinear time series. KNF adapts Koopman embeddings to temporal shifts Wang et al. (2022), while Koopa decomposes non-stationary dynamics into invariant and time-varying components Liu et al. (2023). Other methods combine Koopman lifting with latent state correction or sequence reconstruction, including K-ESKF Huang et al. (2024) and K^{2}VAE Wu et al. (2025). These approaches motivate structured latent evolution, while our focus is a compact operator-derived representation for efficient point forecasting.

3 PRELIMINARIES

Koopman theory. Consider a stochastic discrete-time dynamical system

$$\mathbf{x}_{t+1} = \xi(F(\mathbf{x}_t), \epsilon_t). \tag{1}$$

Here, $F : \mathcal{X} \to \mathcal{X}$ is a possibly nonlinear mapping for a domain $\mathcal{X} \subseteq \mathbb{R}^d$, and $\epsilon_t \sim \mathcal{D}$ denotes independent random noise. Assuming that F and the noise distribution $\mathcal{D}$ are time-invariant, the process is a time-homogeneous Markov process with transition density $p(\mathbf{x}' \mid \mathbf{x})$, i.e., $\Pr(\mathbf{X}_{t+1} \in A \mid \mathbf{X}_t = \mathbf{x}) = \int_A p(\mathbf{x}' \mid \mathbf{x})\, d\mathbf{x}'$ for all measurable sets $A \subseteq \mathcal{X}$ and all $t \geq 0$.

In this stochastic setting, the dynamics are fully characterized by the transition density $p(\mathbf{x}' \mid \mathbf{x})$ induced by $(\xi \circ F)(\cdot)$. The Koopman operator is therefore the conditional expectation operator: for an observable $g : \mathcal{X} \to \mathbb{R}$,

$$(\mathcal{K}g)(\mathbf{x}) \triangleq \mathbb{E}_{p(\mathbf{x}'|\mathbf{x})}[g(\mathbf{x}')]. \tag{2}$$

By the Markov property, repeated application of the Koopman operator gives the multi-step posterior mean of $g(\mathbf{X}_t)$ conditioned on $\mathbf{X}_0 = \mathbf{x}_0$, i.e., $(\mathcal{K}^t g)(\mathbf{x}_0) = \mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}[g(\mathbf{x}_t)]$. We assume that the operator is compact throughout this paper.

Problem setting. Given historical data $X = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T) \in \mathbb{R}^{N \times T}$, our goal is to model the stochastic process and predict the future sequence $Y = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \cdots, \mathbf{x}_{T+t})$, with each point sampled from its conditional distribution $p(\mathbf{x}_t \mid \mathbf{x}_0)$. For two states $\mathbf{x}$ and $\mathbf{x}'$ at successive times, their marginal distributions ρ_0 and ρ_1 satisfy $\rho_1(\mathbf{x}') \triangleq \mathbb{E}_{\rho_0(\mathbf{x})}[p(\mathbf{x}' \mid \mathbf{x})]$.

Let $\mathcal{X}$ be a measurable space and let ρ_0 and ρ_1 be finite measures on $\mathcal{X}$ representing the distributions of the current and future states, respectively. For any measure ρ on $\mathcal{X}$ define $L^2_\rho(\mathcal{X}) \triangleq \{f : \mathcal{X} \to \mathbb{C} \mid \int |f(\mathbf{x})|^2 \rho(d\mathbf{x}) < \infty\}$. Equipped with the inner product $\langle f, g \rangle_\rho \triangleq \int f(\mathbf{x}) g(\mathbf{x}) \rho(d\mathbf{x})$, $L^2_\rho(\mathcal{X})$ is a Hilbert space. The Koopman operator $\mathcal{K}$ is then a mapping from $L^2_{\rho_1}(\mathcal{X})$ to $L^2_{\rho_0}(\mathcal{X})$.

SVD of Operators. Given $\mathcal{K} : L^2_{\rho_1}(\mathcal{X}) \to L^2_{\rho_0}(\mathcal{X})$, the Singular Value decomposition(SVD) decomposes $\mathcal{K}$ into a countable sum of rank-1 operators:

$$\mathcal{K} = \sum_{i=1}^{\infty} \sigma_i \, f_i \otimes g_i, \tag{3}$$

where the outer product operator acts as $(f_i \otimes g_i)h = \langle g_i, h \rangle_{\rho_1} f_i$. The singular values and singular functions satisfy:

$$\mathcal{K} g_i = \sigma_i f_i, \qquad \mathcal{K}^* f_i = \sigma_i g_i, \tag{4}$$

with orthogonality conditions:

$$\langle g_i, g_j \rangle_{\rho_1} = \delta_{ij}, \qquad \langle f_i, f_j \rangle_{\rho_0} = \delta_{ij}, \tag{5}$$

where $\{g_i\} \subset L^2_{\rho_1}(\mathcal{X})$ forms an orthonormal basis of the input space (observables at current state $\mathbf{x}_t$), and $\{f_i\} \subset L^2_{\rho_0}(\mathcal{X})$ forms an orthonormal basis of the output space (observables at future state $\mathbf{x}_{t+1}$), with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. In a real-world scenario, the transition often processes **Low-Rank (LoRA) properties**, which implies σ_i will decline to 0 rapidly as i increases.

Second-moment matrices are key quantities in SVD. We thus introduce the following shorthand notation for convenience. The *second-moment matrix* of vector-valued functions $\mathbf{h}_1(\cdot)$ and $\mathbf{h}_2(\cdot)$ with respect to a distribution ρ is denoted as

$$\mathsf{M}_\rho[\mathbf{h}_1, \mathbf{h}_2] \triangleq \mathbb{E}_{\rho(\mathbf{x})}[\mathbf{h}_1(\mathbf{x}) \mathbf{h}_2(\mathbf{x})^\top], \tag{6}$$

and we write $\mathsf{M}_\rho[\mathbf{h}] \triangleq \mathsf{M}_\rho[\mathbf{h}, \mathbf{h}]$ for shorthand. The *joint second-moment matrix* of $\mathbf{h}_1(\cdot)$ and $\mathbf{h}_2(\cdot)$ over the joint distribution $\rho_0(\mathbf{x}) p(\mathbf{x}' \mid \mathbf{x})$ is defined as

$$\mathsf{T}[\mathbf{h}_1, \mathbf{h}_2] \triangleq \mathbb{E}_{\rho_0(\mathbf{x}) p(\mathbf{x}'|\mathbf{x})}[\mathbf{h}_1(\mathbf{x}) \mathbf{h}_2(\mathbf{x}')^\top], \tag{7}$$

which satisfies the identities $\mathsf{T}[\mathbf{h}_1, \mathbf{h}_2] = \mathsf{M}_{\rho_0}[\mathbf{h}_1, \mathcal{K}\mathbf{h}_2] = \mathsf{M}_{\rho_1}[\mathcal{K}^*\mathbf{h}_1, \mathbf{h}_2]$. In practice, these quantities are to be estimated with trajectories, and their empirical estimates are denoted with hats ($\hat{}$), i.e., $\hat{\mathsf{M}}_{\rho_0}[\mathbf{f}]$, $\hat{\mathsf{M}}_{\rho_1}[\mathbf{g}]$, and $\hat{\mathsf{T}}[\mathbf{f}, \mathbf{g}]$.

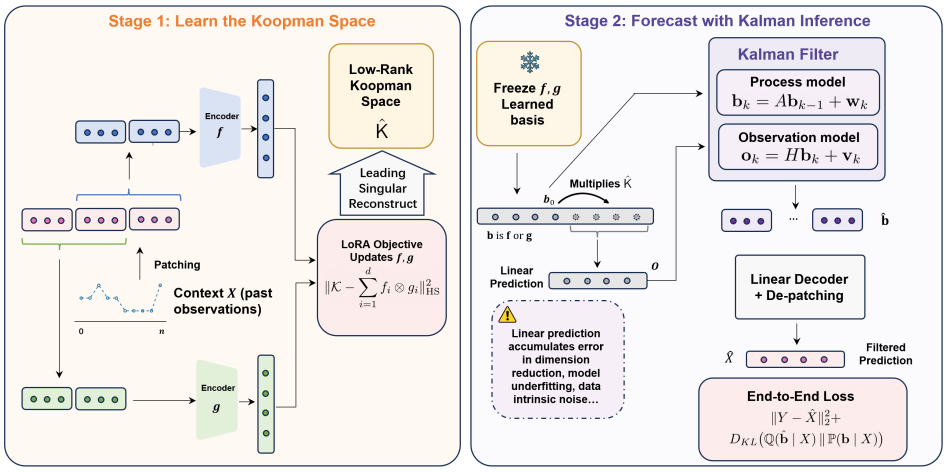


Figure 2: Overview of the two-stage K^{2}SVD procedure for learning a low-rank Koopman space and performing Kalman inference.

4 Proposed Method

We propose K^{2}SVD, a time-series forecasting framework that combines Koopman representations with Kalman inference to model complex nonlinear dynamics through linear evolution in a latent space. Specifically, K^{2}SVD learns a neural representation that lifts observations into a low-rank Koopman space, where state transitions are modeled as linear dynamics perturbed by Gaussian noise. This enables efficient and principled inference via the Kalman filter.

K^{2}SVD is trained with a two-stage algorithm: first, it learns the low-rank Koopman space by optimizing the Hilbert–Schmidt objective; second, it freezes the learned Koopman basis and trains the Kalman inference and observation components for forecasting. The predicted latent states are then mapped back to the observation space through a linear projection. This design combines neural expressivity with the analytical tractability of linear dynamical systems. The full procedure is illustrated in Figure 2 and the pipeline is summarized in Appendix B.

4.1 First Stage: Learning the Koopman Space

4.1.1 Encoder

We consider **multivariate patches** as tokens to implicitly model the cross-variable interaction during state transition. We divide the context series

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T] \in \mathbb{R}^{N \times T}$$

into non-overlapping patches:

$$X^P = [\mathbf{x}_1^P, \mathbf{x}_2^P, \cdots, \mathbf{x}_n^P] \in \mathbb{R}^{N \times s \times n},$$

where $s = T/n$ denotes the patch size, n is the number of patches, and patch $\mathbf{x}_i^P \in \mathbb{R}^{N \times s}$. Similarly, we patchwise the horizon as well with the same patch size s as follows:

$$Y^P = [\mathbf{x}_1^P, \mathbf{x}_2^P, \cdots, \mathbf{x}_m^P] \in \mathbb{R}^{N \times s \times m},$$

In the following, we capture the singular functions of the desired Koopman space with neural networks: $\mathbf{f}(X^P) = [\mathbf{f}(\mathbf{x}_1^P), \mathbf{f}(\mathbf{x}_2^P), \cdots, \mathbf{f}(\mathbf{x}_n^P)]$, and for similar cases $\mathbf{g}(X^P) = [\mathbf{g}(\mathbf{x}_1^P), \mathbf{g}(\mathbf{x}_2^P), \cdots, \mathbf{g}(\mathbf{x}_n^P)]$ where $\mathbf{f}, \mathbf{g}$ are desired neural network corresponding to the intrinsic property of data.

4.1.2 LoRA Loss Function

The best rank-d approximation of $\mathcal{K}$ in the Hilbert-Schmidt norm is:

$$\mathcal{K} \approx \sum_{i=1}^{d} \sigma_i \, f_i \otimes g_i \tag{8}$$

with the learned f_i and g_i spans the optimal approximated rank d subspace $\mathcal{H}^d$ of the output and input space. To bypass the numerical instability and optimization challenges, Jeong et al. (2025) propose to directly minimize the low-rank approximation error $\|\mathcal{K} - \sum_{i=1}^{d} f_i \otimes g_i\|_{\text{HS}}^2$ to find the singular functions of the low-rank Koopman operator Hilbert space, where $\|\cdot\|_{HS}$ denotes the Hilbert–Schmidt loss. Succinctly, the learning objective can be expressed as:

$$\mathcal{L}_{\text{lora}}(\tilde{\mathbf{f}}, \tilde{\mathbf{g}}) \triangleq -2\,\text{tr}(\mathsf{T}[\tilde{\mathbf{f}}, \tilde{\mathbf{g}}]) + \text{tr}(\mathsf{M}_{\rho_0}[\tilde{\mathbf{f}}]\mathsf{M}_{\rho_1}[\tilde{\mathbf{g}}]), \tag{9}$$

where $\tilde{\mathbf{f}} = [\mathbf{f}(\mathbf{x}_2^P), \mathbf{f}(\mathbf{x}_3^P), \cdots, \mathbf{f}(\mathbf{x}_n^P)]$, and $\tilde{\mathbf{g}} = [\mathbf{g}(\mathbf{x}_1^P), \mathbf{g}(\mathbf{x}_2^P), \cdots, \mathbf{g}(\mathbf{x}_{n-1}^P)]$. A detailed derivation of the loss is provided in Appendix A. Noting that the LoRA loss does not require ground-truth prediction for calculating the gradient, we can train it separately in a stage before inference to stabilize and well-structure the Koopman space.

For forecasting, we select a learned basis $\mathbf{b} \in \{\mathbf{f}, \mathbf{g}\}$ and write $\mathbf{b}^P = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$. Following the justification of Jeong et al. (2025), we treat the dominant singular-function subspace learned by the Hilbert–Schmidt objective as an approximately Koopman-invariant subspace, so the action of $\mathcal{K}$ on a basis $\mathbf{b}$ of this subspace admits a finite-dimensional linear representation $\mathsf{K}_{\mathbf{b}} : \mathcal{H}^d \to \mathcal{H}^d$ such that

$$(\mathcal{K}\mathbf{b})(\mathbf{x}) = \mathbb{E}_{p(\mathbf{x}'|\mathbf{x})}[\mathbf{b}(\mathbf{x}')] \approx \mathsf{K}_{\mathbf{b}}\mathbf{b}(\mathbf{x}). \tag{10}$$

Ideally, if the data obeys the dynamics given by equation 1, i.e., has no temporal distribution shift, is Markovian, and has sufficient latent dimension for linearization, prediction can be performed by a simple linear rollout. Given the linear relation, $\mathsf{K}_{\mathbf{b}}$ can be approximately solved with linear regression $\mathsf{K}_{\mathbf{b}}^{\text{ols}} \triangleq (\mathsf{M}_{\rho_0}[\mathbf{b}])^{+}\mathsf{T}[\mathbf{b}] \in \mathbb{R}^{k\times k}$ and we multiply it repeatedly to the last state in the training data until the prediction fills $m = (t+T)/s$ time steps to get the linearly reconstructed base vectors of horizon data,

$$\mathbf{b}^L = [\mathsf{K}_{\mathbf{b}}^{\text{ols}}\mathbf{b}(\mathbf{x}_n^P), (\mathsf{K}_{\mathbf{b}}^{\text{ols}})^2\mathbf{b}(\mathbf{x}_n^P), \cdots, (\mathsf{K}_{\mathbf{b}}^{\text{ols}})^{m-n}\mathbf{b}(\mathbf{x}_n^P)]. \tag{11}$$

Noted that $\mathbf{b}^L$ has the same length $(m-n)$ as the prediction, and we treat it as a preliminary estimate. As shown in the ablation study in Section 5.3.2, it is not sufficiently accurate, motivating the use of a Kalman filter.

4.2 Second Stage: Forecast with Kalman Correction

4.2.1 Latent Kalman Correction

As described above, in real-world datasets, nonlinear residuals can cause misalignment between the actual base vector in the state space and its linear expansion $\mathbf{b}^L$. This noise can arise from dimension reduction, nonlinear or non-Markovian dynamics, random perturbations in the data, and errors in the linear regression estimate $\mathsf{K}_{\mathbf{b}}^{\text{ols}}$. Despite these nonlinear residuals, we posit that the learned Koopman space is sufficiently representative so that the residual components carry limited predictive information and can be mitigated as noise during inference. Considering the recursive nature of time-series prediction, the Kalman filter provides a lightweight, closed-form inference procedure that minimizes estimation error under linear Gaussian dynamics and inspired us to apply it as follows.

For $k \geq n+1$, its processing model is:

$$\mathbf{b}_k = A\mathbf{b}_{k-1} + \mathbf{w}_k, \tag{12}$$

where $A \in d \times d$ is the transition matrix, which we initialize with $\mathsf{K}_{\mathbf{b}}^{\text{ols}}$ by definition. $\mathbf{w}_k \sim N(\mathbf{0}, Q)$ is the process noise and Q is its covariance matrix. Accordingly, the observation model is:

$$\mathbf{o}_k = H\mathbf{b}_k + \mathbf{v}_k. \tag{13}$$

Here $H \in d \times d$ is the observation matrix, which we initialize with I. $\mathbf{o}_k$ is the pseudo-observation. Since we have no access to the actual future states, we assign the expanded linear component to it $\mathbf{o}_k = \mathbf{b}^L[k]$ as a unfiltered noisy state, and $\mathbf{v}_k \sim N(\mathbf{0}, R)$. We then conduct the prediction and update steps iteratively. The prediction step is formulated as:

$$\hat{\mathbf{b}}_{k|k-1} = A\hat{\mathbf{b}}_{k-1|k-1}, \tag{14}$$

$$\Sigma_{k|k-1} = A\Sigma_{k-1|k-1}A^T + Q, \tag{15}$$

where $\hat{\mathbf{b}}_{k|k-1}$ is the predicted state and $\Sigma_{k|k-1}$ is the predicted covariance matrix of the process uncertainty. The update step uses the Kalman gain G_k to balance the prediction and observation, thereby refining the latent state:

$$G_k = \Sigma_{k|k-1} H^T (H \Sigma_{k|k-1} H^T + R)^{-1}, \tag{16}$$

$$\hat{\mathbf{b}}_{k|k} = \hat{\mathbf{b}}_{k|k-1} + G_k(\mathbf{o}_k - H\hat{\mathbf{b}}_{k|k-1}), \tag{17}$$

$$\Sigma_{k|k} = (I - G_k H)\Sigma_{k|k-1}. \tag{18}$$

We note that A, H, Q, R are trainable parameters in the above process. Iteratively, we obtain the refined predicted singular vectors $\hat{\mathbf{b}} = [\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \cdots, \hat{\mathbf{b}}_m]$.

4.2.2 DECODER

Given the learned base functions $\mathbf{b}$ and an observable h, we compute the projection coefficient $\mathbf{z}_{\mathbf{b}}^{\text{ols}} \triangleq (\mathsf{M}_{\rho_0}[\mathbf{b}])^{\dagger} \langle h, \mathbf{b} \rangle_{\rho_0} \in \mathbb{R}^{k \times N \times s}$ by solving the linear regression $h(\mathbf{x}) \approx \mathbf{z}^{\top}\mathbf{b}(\mathbf{x})$. This coefficient is fixed throughout the forecasting process. In our case, we take $h(\mathbf{x}) = \mathbf{x}$, so that the predicted data can be decoded from the learned base functions as follows:

$$\hat{X}^P = \hat{\mathbf{b}}^{\top} \mathbf{z}_{\mathbf{b}}^{\text{ols}}. \tag{19}$$

After prediction, we de-patch the outputs to obtain $\hat{X}$ and optimize an MSE-based reconstruction loss together with an additional KL-divergence term on the filter output distribution $\hat{\mathbf{b}}$. This regularization encourages a well-structured latent representation and helps preserve the orthogonality of the learned base functions, leading to the following loss:

$$\mathcal{L}_e = \|Y - \hat{X}\|_2^2 + D_{KL}\big(\mathbb{Q}(\hat{\mathbf{b}} \mid X) \,\|\, \mathbb{P}(\mathbf{b} \mid X)\big), \tag{20}$$

where $\mathbb{P}(\mathbf{b} \mid X) = \mathcal{N}(0, \mathbf{I})$, and $\hat{X}$ and $\hat{\mathbf{b}}$ are induced by the trainable Kalman filter.

4.3 OVERALL ALGORITHM

We adopt a two-stage training procedure for learning the Koopman representation.

Train encoders with LoRA objective. First, we independently train the Koopman module using the LoRA objective to capture the low-rank dynamics and establish a stable latent space for linear evolution. We then freeze the learned functions, assuming that the low-rank structure has been adequately captured, and estimate the Koopman matrix.

Dynamically updated matrix. We estimate the Koopman operator $\mathsf{K}_{\mathbf{b}}$ via linear regression and initialize the transition matrix A of the Kalman filter using $\mathsf{K}_{\mathbf{b}}^{\text{ols}}$, averaged over 128 training batches. Because the patched training data provide too few samples for reliable regression, we instead treat the finite-dimensional Koopman matrix K_θ as trainable and optimize it jointly in an end-to-end manner using the forecasting objective $\mathcal{L}_e$. The Koopman matrix is initialized using the static least-squares estimate $\mathsf{K}_\theta \leftarrow \mathsf{K}_{\mathbf{b}}^{\text{ols}}$. The proposed K^{2}SVD preserves the principled operator-consistent latent space learned by LoRA, while allowing the transition dynamics to adapt to the downstream multi-step forecasting objective, mitigating finite-sample error in the regression. Since $\mathsf{K}_\theta \in \mathbb{R}^{d \times d}$ is low-dimensional, making it trainable adds only a small computational and parameter overhead.

Appendix B summarizes when each component is initialized, optimized, or held fixed. All learned parameters remain fixed during inference.

5 EXPERIMENTS

In this section, we present empirical results that validate the effectiveness and performance of K^{2}SVD. We first examine the singular-value spectra of the learned Koopman spaces to characterize the intrinsic rank structure of the datasets. We evaluate forecasting on both low- and high-dimensional datasets, with an emphasis on computationally demanding high-dimensional cases. Finally, we report efficiency results and ablation studies to assess the contribution of each module.

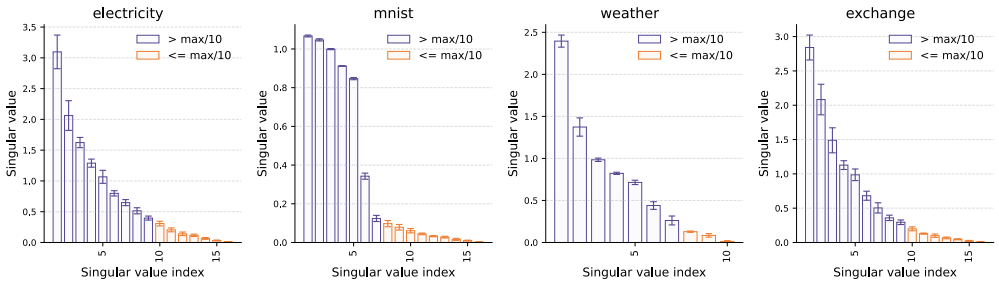


Figure 3: The singular values of the Koopman space exhibit low-rank properties on multiple datasets.

5.1 Experiment Setup

We evaluate K^{2}SVD against state-of-the-art (SOTA) efficient and precise baselines on real-world and dynamic forecasting tasks. To make the comparison clear, we organize the datasets by their dimensionality.

Datasets. We conduct experiments on nine forecasting datasets: eight real-world datasets from the ProbTS benchmark Zhang et al. (2024) and one dynamical-system dataset following the experimental setting of Jeong et al. (2025). The high-dimensional group consists of Electricity-L, Weather-L, and Ordered-MNIST. Electricity-L contains the consumption of electricity in kWh, Weather-L contains multivariate climatological measurements, and Ordered-MNIST represents image-based dynamic observations. With tens or hundreds of dimensions, these datasets better assess whether the learned low-rank Koopman space improves scalability and efficiency. The low-dimensional group includes ETTh1-L, ETTh2-L, ETTm1-L, ETTm2-L, Exchange-L, and ILI-L. These datasets contain relatively few variables and evaluate whether K^{2}SVD remains competitive when the observation space is already compact. Detailed dataset statistics are reported in Table 7.

The context and horizon length for the main results are displayed in table 7 and additional results for longer term time series prediction are shown in the Appendix D.1.

Baselines. We compare K^{2}SVD with seven sequence-forecasting baselines on real-world and dynamical-system data: K^2VAE Wu et al. (2025), FITS Xu et al. (2024), iTransformer Liu et al. (2024), Koopa Liu et al. (2023), TSDiff Kollovieh et al. (2023), GRU NVP Rasul et al. (2021), and TimeBase Huang et al. (2025). For every baseline, we evaluate the model's point prediction withNRMSE; implementation details are provided in Appendix C.

Evaluation Metrics. We evaluate point forecasts using NRMSE (Normalized Root Mean Square Error) as defined below. We conduct model agnostic evaluation of the performance on raw data space. We also report model computational complexity and inference time in the efficiency section to highlight the benefits of the LoRA structure recovered by the proposed K^{2}SVD algorithm.

$$\text{NRMSE} = \frac{\sqrt{\text{MSE}}}{\frac{1}{N}\sum_{i=1}^{N}|Y_i|}, \text{ with MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2.$$

5.2 Main Results

Empirical evidence for the LoRA property. We observe low-rank structure across both low-dimensional and high-dimensional datasets, a property that most existing models do not explicitly capture or exploit. As shown by the Koopman-space spectrum in Figure 3, the singular values decay toward zero as the rank increases across all scenarios. Most datasets exhibit this decay within roughly 10-30 dimensions. These observations motivate an adaptive choice of latent dynamic dimension, where we use dynamic dimension 16 for most datasets instead of the commonly chosen 128 in baselines. The ablation study further shows that increasing the latent dimension has only a minor effect on precision but substantially increases computational cost. One exception is ILI-L, which requires approximately 60 Koopman dimensions before the singular values vanish, indicating a relatively high-rank structure despite its low observation dimension. Results for ILI-L are reported in the Appendix D.3.

Table 1: Results of models using the NRMSE metric on the high-dimensional datasets and Exchange-L. Best results are shown in **bold**, and second-best results are underlined. '-' indicates that the run took too long to finish.

Model	Electricity-L	MNIST	Weather-L	Exchange-L
FITS	1.3386 ± 0.0038	2.0438 ± 0.0007	0.6226 ± 0.0133	0.0912 ± 0.0001
TimeBase	0.9864 ± 0.1110	1.8188 ± 0.0011	0.2969 ± 0.0031	0.0476 ± 0.0126
iTransformer	1.3035 ± 0.0033	2.0486 ± 0.0014	0.3617 ± 0.0006	0.1050 ± 0.0001
Koopa	1.3977 ± 0.0183	2.0510 ± 0.0007	0.3656 ± 0.0004	0.1051 ± 0.0002
TSDiff	1.2107 ± 0.0587	-	0.3101 ± 0.0082	0.1196 ± 0.0060
GRU NVP	1.0617 ± 0.0299	-	0.7841 ± 0.1476	0.0662 ± 0.0002
K^2VAE	0.8682 ± 0.0106	1.9817 ± 0.0379	**0.2853 ± 0.0042**	0.0470 ± 0.0009
K^{2}SVD	**0.8502 ± 0.0014**	**1.7844 ± 0.0007**	0.3054 ± 0.0062	**0.0419 ± 0.0004**

Prediction performance. The prediction results in Table 1 show that K^{2}SVD performs strongly on Electricity-L, Weather-L, Ordered-MNIST, and Exchange-L. For fairness, a CNN encoder will replace all the encoders when models are applied on MNIST dataset. As described in the dataset setup and summarized in Table 7, the former three datasets form the high-dimensional group, with observations consisting of many interacting variables or image-like dynamic states. K^{2}SVD is well-suited to these settings because it directly identifies a compact transition space from the underlying dynamics, reducing redundant latent computation while correcting accumulated rollout error with its lightweight latent update.

For the low-dimensional group, K^{2}SVD also remains competitive, with full results reported in Appendix D.1. Exchange-L is included because it is a low-dimensional but challenging financial dataset characterized by weak periodicity and nonstationary dynamics. Together, our results show that the principled Koopman transition space is beneficial across different observation scales, with the efficiency advantage to be evaluated in the next section.

Table 2: Comparison of inference time per validation batch in seconds, and GPU usage in GBs on the high-dimensional datasets. For each metric, the best results are shown in **bold**, and the second-best results are underlined. "-" indicates that the run took too long to finish.

Model	Inference Time			GPU Usage		
	Electricity-L	MNIST	Weather-L	Electricity-L	MNIST	Weather-L
FITS	31.261 ± 1.083	43.497 ± 0.199	4.072 ± 0.026	1.619	4.313	0.140
TimeBase	39.276 ± 0.760	209.821 ± 0.602	5.898 ± 0.062	1.313	3.479	0.116
iTransformer	30.674 ± 0.421	43.276 ± 0.115	4.177 ± 0.032	1.622	4.314	0.150
Koopa	29.559 ± 0.067	43.097 ± 0.229	4.026 ± 0.020	1.635	4.432	0.149
TSDiff	93.217 ± 0.572	-	13.677 ± 0.102	5.692	-	2.276
GRU NVP	42.229 ± 0.164	-	4.802 ± 0.069	1.817	-	0.126
K^2VAE	84.202 ± 0.167	127.390 ± 0.255	8.150 ± 0.126	1.702	4.437	0.160
K^{2}SVD	**10.144 ± 0.069**	**13.662 ± 0.205**	**2.787 ± 0.041**	**0.438**	**0.954**	**0.048**

Efficiency and compactness. From an efficiency perspective, K^{2}SVD establishes outstanding performance in both inference time and GPU memory usage among the evaluated methods, with its advantage becoming particularly pronounced on high-dimensional datasets. All models are evaluated on the same GPU under an identical software environment, and the results are reported in Table 2. K^{2}SVD achieves the fastest inference time on all three datasets. On the two most computationally demanding benchmarks, Electricity and MNIST, it is approximately three times faster than every competing baseline. In terms of memory usage, K^{2}SVD requires less than half of the peak GPU memory allocated by the most efficient baseline models. Notably, K^{2}SVD substantially outperforms efficiency oriented models FITS and TimeBase in both inference speed and memory usage. This comparison demonstrates that the efficiency of K^{2}SVD arises from its low-rank transition space and from concentrating iterative prediction within this space. Consequently, K^{2}SVD achieves both superior predictive accuracy and substantially lower inference cost, yielding a more favorable accuracy-efficiency trade-off.

5.3 Ablation Study

5.3.1 Ablation on the Dimension of Dynamic Space

We conduct an ablation study to examine the sufficiency of the latent Koopman dimension. Specifically, we vary the rank of the learned Koopman subspace from 16 to 64 to investigate whether reducing the rank harms performance. As shown in Table 3, performance in prediction precision does not improve significantly yet sometimes drops as the dimension increases, while the total computation measured as FLOPs (floating-point operations per second) booms. Although inference time does not visibly increase, we infer that it is limited by memory bandwidth rather than compute capacity, as the models require no more than 1 GFLOP while the GPU provides over 100 TFLOPS of compute. These results suggest that the dominant system dynamics can be effectively captured in a compact, low-rank Koopman subspace, making higher-dimensional representations redundant.

Table 3: Ablation study on the Koopman latent dimension. Low-dimensional latent spaces achieve competitive accuracy, indicating redundancy in higher-rank representations.

Dataset	Metric	Rank 16	Rank 32	Rank 64
Exchange-L	NRMSE	0.0441 ± 0.0016	0.0449 ± 0.0002	0.0428 ± 0.0001
	Inference time (s)	0.656 ± 0.027	0.625 ± 0.044	0.629 ± 0.036
	FLOPs	3M	6M	33M
Electricity-L	NRMSE	0.8502 ± 0.0014	0.906 ± 0.008	0.8690 ± 0.0120
	Inference time (s)	9.786 ± 0.088	9.980 ± 0.181	9.856 ± 0.031
	FLOPs	40M	58M	137M

5.3.2 Ablation on Kalman Module

As argued in Section 4.2, we designed Kalman module to handle approximation and recursive rollout errors accumulated in the latent path. To test the necessity, we remove it, decode the linear rollout directly, and compare the resulting point forecasts. This setting is identical to directly applying Jeong et al. (2025)'s forecasting theme on our benchmarks. The performance decline shows that the Kalman module contributes beyond the linear rollout alone.

Table 4: Ablation of the latent correction module. Directly decoding the linear rollout yields less accurate point forecasts.

Ablation	Electricity-L	MNIST	Weather-L	Exchange-L
Linear	3.0902 ± 0.3347	1.8829 ± 0.0054	0.4228 ± 0.0119	0.0447 ± 0.0005
Filtered	**0.8502 ± 0.0014**	**1.8125 ± 0.0010**	**0.3124 ± 0.0011**	**0.0419 ± 0.0004**

6 Conclusion

This work revisits time-series forecasting from the perspective of operator structure rather than only predictive capacity. The central observation is that many forecasting benchmarks contain a compact transition structure that can be exposed through the spectrum of the learned Koopman space. By building the forecasting model around this structure, K^2SVD reduces the latent dynamic dimension substantially while retaining competitive prediction accuracy and achieving the best inference efficiency, strongly exceeding baseline models.

The experiments also show results with clear distinction in intrinsically different datasets, revealing information about the data and providing a visible judgment through the learned spectrum rather than empirical observations.

More broadly, these results suggest that efficient forecasting should not only compress model parameters, but also identify which part of the temporal dynamics is worth modeling explicitly. Future work can extend this direction by allowing the Koopman rank and transition operator to adapt across time and by developing robust point-correction mechanisms for changing dynamics.

AI USE STATEMENT

We used generative AI tools to create or edit software code and to improve the readability of the manuscript. We did not use generative AI tools to generate synthetic datasets, develop theoretical or conceptual frameworks, formulate mathematical claims, contribute to proofs, propose hypotheses, design the research methodology or experiments, implement the proposed methods, translate content, process datasets, analyze qualitative data, or interpret experimental results; other required disclosure activities were not applicable to this work. All AI-assisted code was reviewed, tested, and validated by the authors, and all AI-assisted edits to the manuscript were checked for technical accuracy, appropriate attribution, and consistency with the underlying results. The authors take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the Language of Time Series, 2024. URL https://arxiv.org/abs/2403.07815.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A Decoder-Only Foundation Model for Time-Series Forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10148–10167. PMLR, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Peng Huang, Ketong Zheng, and Gerhard P. Fettweis. Data-Driven Koopman Operator-Based Error-State Kalman Filter for Enhanced State Estimation of Quadrotors in Agile Flight. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10983–10989, 2024. doi: 10.1109/IROS58592.2024.10802457.

Qihe Huang, Zhengyang Zhou, Kuo Yang, Zhongchao Yi, Xu Wang, and Yang Wang. Timebase: The power of minimalism in efficient long-term time series forecasting. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=GhTdNOMfOD.

Minchan Jeong, Jongha Jon Ryu, Se-Young Yun, and Gregory W. Wornell. Efficient Parametric SVD of Koopman Operator for Stochastic Dynamical Systems. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kL2pnzClyD.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Wang. Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 36, pages 28341–28364, 2023.

Vladimir R. Kostic, Pietro Novelli, Riccardo Grazzi, Karim Lounici, and Massimiliano Pontil. Learning Invariant Representations of Time-Homogeneous Stochastic Dynamical Systems. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=twSnZwiOIm.

Yueheng Lan and Igor Mezić. Linearization in the Large of Nonlinear Systems and Koopman Operator Spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013. doi: 10.1016/j.physd.2012.08.017.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021. doi: 10.1016/j.ijforecast.2021.03.012.

Ningxin Liu, Shuigen Liu, Xin T. Tong, and Lijian Jiang. Estimate of Koopman Modes and Eigenvalues with Kalman Filter. *SIAM Journal on Scientific Computing*, 48(2):A512–A539, 2026. doi: 10.1137/24M1684207.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning Non-Stationary Time Series Dynamics with Koopman Predictors. In *Advances in Neural Information Processing Systems*, volume 36, pages 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. In *International Conference on Learning Representations*, 2024.

Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets for Deep Learning of Molecular Kinetics. *Nature Communications*, 9(1):5, 2018. doi: 10.1038/s41467-017-02388-1.

Igor Mezić. Spectral Properties of Dynamical Systems, Model Reduction and Decompositions. *Nonlinear Dynamics*, 41:309–325, 2005. doi: 10.1007/s11071-005-2824-x.

Igor Mezić. Analysis of Fluid Flows via Spectral Properties of the Koopman Operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013. doi: 10.1146/annurev-fluid-011212-140652.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting. In *International Conference on Learning Representations*, 2020.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In *International Conference on Learning Representations*, 2021.

Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral Analysis of Nonlinear Flows. *Journal of Fluid Mechanics*, 641:115–127, 2009. doi: 10.1017/S0022112009992059.

Peter J. Schmid. Dynamic Mode Decomposition of Numerical and Experimental Data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217.

Jonathan Huy Tu. *Dynamic Mode Decomposition: Theory and Applications*. PhD thesis, Princeton University, 2013.

Rui Wang, Yihe Dong, Sercan Ö. Arik, and Rose Yu. Koopman Neural Forecaster for Time Series with Temporal Distribution Shifts, 2022. URL `https://arxiv.org/abs/2210.03675`.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015a. doi: 10.1007/s00332-015-9258-5.

Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A Kernel-Based Method for Data-Driven Koopman Spectral Analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015b. doi: 10.3934/jcd.2015005.

Hao Wu and Frank Noé. Variational Approach for Learning Markov Processes from Time Series Data. *Journal of Nonlinear Science*, 30(1):23–66, 2020. doi: 10.1007/s00332-019-09567-y.

Xingjian Wu, Xiangfei Qiu, Hongfan Gao, Jilin Hu, Bin Yang, and Chenjuan Guo. K^2VAE: A Koopman-Kalman Enhanced Variational AutoEncoder for Probabilistic Time Series Forecasting, 2025. URL `https://arxiv.org/abs/2505.23017`.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling Time Series with $10k$ Parameters. In *International Conference on Learning Representations*, 2024.

Jiawen Zhang, Xumeng Wen, Zhenwei Zhang, Shun Zheng, Jia Li, and Jiang Bian. ProbTS: Benchmarking Point and Distributional Forecasting across Diverse Prediction Horizons. In *Advances in Neural Information Processing Systems*, volume 37, pages 48045–48082, 2024.

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. doi: 10.1609/aaai.v35i12.17325.

A Proof to the LoRA Loss

Here, we provide the proof to justify the LoRA loss proposed in Jeong et al. (2025).

Recall that $\mathcal{L}_{\text{lora}}(\mathbf{f}, \mathbf{g}) \triangleq -2\,\mathrm{tr}(\mathrm{T}[\mathbf{f}, \mathbf{g}]) + \mathrm{tr}(\mathrm{M}_{\rho_0}[\mathbf{f}]\mathrm{M}_{\rho_1}[\mathbf{g}])$. To prove, first note that

$$\left\| \mathcal{K} - \sum_{i=1}^{k} f_i \otimes g_i \right\|_{\text{HS}}^2 - \|\mathcal{K}\|_{\text{HS}}^2 = -2\sum_{i=1}^{k} \langle f_i, \mathcal{K} g_i \rangle_{\rho_0} + \sum_{i=1}^{k}\sum_{j=1}^{k} \langle f_i, f_j \rangle_{\rho_0} \langle g_i, g_j \rangle_{\rho_1}. \tag{21}$$

Here, the first term can be rewritten as

$$\sum_{i=1}^{k} \langle f_i, \mathcal{K} g_i \rangle_{\rho_0} = \sum_{i=1}^{k} \mathbb{E}_{\rho_0(\mathbf{x})p(\mathbf{x}'|\mathbf{x})}\left[f_i(\mathbf{x}) g_i(\mathbf{x}') \right] \tag{22}$$

$$= \mathrm{tr}\big(\mathbb{E}_{\rho_0(\mathbf{x})p(\mathbf{x}'|\mathbf{x})}\left[\mathbf{f}(\mathbf{x}) \mathbf{g}(\mathbf{x}')^\top \right] \big) = \mathrm{tr}(\mathrm{T}[\mathbf{f}, \mathbf{g}]). \tag{23}$$

Likewise, the second term is

$$\sum_{i=1}^{k}\sum_{j=1}^{k} \langle f_i, f_j \rangle_{\rho_0} \langle g_i, g_j \rangle_{\rho_1} = \mathrm{tr}\big(\mathbb{E}_{\rho_0(\mathbf{x})}\left[\mathbf{f}(\mathbf{x}) \mathbf{f}(\mathbf{x})^\top \right] \mathbb{E}_{\rho_1(\mathbf{x}')}\left[\mathbf{g}(\mathbf{x}') \mathbf{g}(\mathbf{x}')^\top \right] \big) \tag{24}$$

$$= \mathrm{tr}(\mathrm{M}_{\rho_0}[\mathbf{f}]\mathrm{M}_{\rho_1}[\mathbf{g}]). \tag{25}$$

This concludes the proof.

B Algorithm

Algorithm 1 summarizes the two-stage training procedure described in the method section. The first stage learns an operator-consistent low-rank Koopman space using the LoRA objective. The second stage freezes this learned Koopman basis and trains the Kalman forecasting module. The static variant K^{2}SVD-S keeps the regression-induced Koopman matrix fixed, while K^{2}SVD further updates the finite-dimensional Koopman matrix during downstream forecasting training.

Additionally, we provide table 5 to clarify the activation of parameters in 2 stages and inference.

Table 5: Initialization and parameter-update schedule. "Optimize" denotes gradient-based updating in the indicated stage; "fixed" denotes use without parameter updates.

Parameter	Initialization	Stage 1: Koopman space	Stage 2: forecasting	Inference
$\mathbf{f}$	Network initialization	Optimize with $\mathcal{L}_{\text{lora}}$	Fixed	Fixed encoder
$\mathbf{g}$	Network initialization	Optimize with $\mathcal{L}_{\text{lora}}$	Fixed	Fixed encoder
$\mathsf{K}_{\mathbf{b}}^{\text{ols}}$	Estimate after Stage 1 from 128 batches	Not applicable	Fixed reference	Fixed reference rollout
K_θ	$\mathsf{K}_{\mathbf{b}}^{\text{ols}}$	Inactive	Optimize with $\mathcal{L}_e$ (dynamic variant)	Fixed reference rollout
A/F	$\mathsf{K}_{\mathbf{b}}^{\text{ols}}$	Inactive	Optimize with $\mathcal{L}_e$	Fixed recurrent transition
H	Identity	Inactive	Optimize with $\mathcal{L}_e$	Fixed observation map
Q	Identity covariance	Inactive	Optimize with $\mathcal{L}_e$	Fixed process covariance
R	Identity covariance	Inactive	Optimize with $\mathcal{L}_e$	Fixed observation covariance
Decoder $\mathbf{z}_{\mathbf{b}}^{\text{ols}}$	Least squares after Stage 1	Inactive	Optimize with $\mathcal{L}_e$	Fixed linear decoder

C Experimental Settings

All experiments are conducted on a single GPU. We run each experiment with seeds 2, 42, and 2026 and aggregate the three independent runs. Validation is performed after every epoch, and the selected checkpoint is evaluated once on the held-out test split. All configurations use Adam with learning rate 10^{-3}, gradient clipping at 0.5, standard input scaling, multistep prediction, and eight data-loading workers. The test batch size is 32. The encoders contain three hidden layers of width 256, and the model directly decodes one corrected latent trajectory into one point forecast per input.

Algorithm 1: Two-Stage K^{2}SVD Training and Forecasting

Input: Training pairs (X, Y); patch length s; Koopman rank d; variant $\nu \in \{\text{static}, \text{dynamic}\}$
Output: Encoders $\mathbf{f}, \mathbf{g}$; Koopman matrix K; Kalman parameters A, H, Q, R; decoder coefficients $\mathbf{z}_{\mathbf{b}}^{\text{ols}}$

Stage 1: Learn a low-rank Koopman space;
for *each training batch* X **do**
  Split X into non-overlapping patches $X^P = [\mathbf{x}_1^P, \ldots, \mathbf{x}_n^P]$;
  Compute shifted Koopman features $\tilde{\mathbf{f}} = [\mathbf{f}(\mathbf{x}_2^P), \ldots, \mathbf{f}(\mathbf{x}_n^P)]$ and $\tilde{\mathbf{g}} = [\mathbf{g}(\mathbf{x}_1^P), \ldots, \mathbf{g}(\mathbf{x}_{n-1}^P)]$;
  Evaluate $\mathcal{L}_{\text{LoRA}} \leftarrow -2\,\text{tr}(\mathsf{T}[\tilde{\mathbf{f}}, \tilde{\mathbf{g}}]) + \text{tr}(\mathsf{M}_{\rho_0}[\tilde{\mathbf{f}}]\mathsf{M}_{\rho_1}[\tilde{\mathbf{g}}])$;
  Update $\mathbf{f}, \mathbf{g}$ by minimizing $\mathcal{L}_{\text{LoRA}}$;

Initialize finite-dimensional dynamics;
Choose the learned basis $\mathbf{b} \in \{\mathbf{f}, \mathbf{g}\}$ used for forecasting;
Estimate the regression-induced Koopman matrix $\mathsf{K}_{\mathbf{b}}^{\text{ols}} \leftarrow \big(\hat{\mathsf{M}}_{\rho_0}[\mathbf{b}]\big)^{\dagger}\hat{\mathsf{T}}[\mathbf{b}]$ using training batches;
Initialize $A \leftarrow \mathsf{K}_{\mathbf{b}}^{\text{ols}}$, $H \leftarrow I$, and noise covariances Q, R;
if $\nu =$ *dynamic* **then**
  Initialize trainable $\mathsf{K}_\theta \leftarrow \mathsf{K}_{\mathbf{b}}^{\text{ols}}$ and set $\mathsf{K} \leftarrow \mathsf{K}_\theta$;
else
  Set $\mathsf{K} \leftarrow \mathsf{K}_{\mathbf{b}}^{\text{ols}}$ and keep K fixed;
Compute decoder coefficients $\mathbf{z}_{\mathbf{b}}^{\text{ols}} \leftarrow (\mathsf{M}_{\rho_0}[\mathbf{b}])^{\dagger}\langle h, \mathbf{b}\rangle_{\rho_0}$ with $h(\mathbf{x}) = \mathbf{x}$;

Stage 2: Train Kalman forecasting module;
Freeze $\mathbf{f}, \mathbf{g}$;
for *each training batch* (X, Y) **do**
  Encode the final context patch to obtain the initial latent state $\hat{\mathbf{b}}_{n|n}$;
  Roll out the regression-induced observation sequence $\mathbf{b}^L$ using $\mathsf{K}_{\mathbf{b}}^{\text{ols}}$;
  for *each forecast patch index* $k = n+1, \ldots, n+m$ **do**
    Predict latent state and covariance: $\hat{\mathbf{b}}_{k|k-1} \leftarrow A\hat{\mathbf{b}}_{k-1|k-1}$, $\Sigma_{k|k-1} \leftarrow A\Sigma_{k-1|k-1}A^\top + Q$;
    Set observation $\mathbf{o}_k \leftarrow \mathbf{b}^L[k]$ and compute Kalman gain $G_k \leftarrow \Sigma_{k|k-1}H^\top(H\Sigma_{k|k-1}H^\top + R)^{-1}$;
    Update latent state and covariance: $\hat{\mathbf{b}}_{k|k} \leftarrow \hat{\mathbf{b}}_{k|k-1} + G_k(\mathbf{o}_k - H\hat{\mathbf{b}}_{k|k-1})$, $\Sigma_{k|k} \leftarrow (I - G_kH)\Sigma_{k|k-1}$;
  Decode patch forecasts $\hat{X}^P \leftarrow \hat{\mathbf{b}}^\top\mathbf{z}_{\mathbf{b}}^{\text{ols}}$ and de-patch to obtain $\hat{X}$;
  Compute $\mathcal{L}_e \leftarrow \|Y - \hat{X}\|_2^2 + D_{KL}\Big(\mathbb{Q}(\hat{\mathbf{b}} \mid X) \,\|\, \mathbb{P}(\mathbf{b} \mid X)\Big)$;
  if $\nu =$ *dynamic* **then**
    Jointly update K_θ and A, H, Q, R using $\mathcal{L}_e$;
  else
    Update A, H, Q, R by minimizing $\mathcal{L}_e$;

Two-Stage Training and Dataset-Specific Configuration. Stage 1 optimizes the two encoders with the Hilbert–Schmidt objective for the number of epochs listed in Table 6. The encoders are then frozen, and Stage 2 optimizes the forecasting components for the remaining epochs. Most epochs contain 100 training batches; Ordered-MNIST uses 60. Table 6 reports all dataset-specific K^{2}SVD settings that differ from the common configuration.

Electricity-L, ETTh2-L, ETTm1-L, and ETTm2-L use cosine annealing with minimum learning rate 10^{-6}; $T_{\max} = 100$ except for ETTm2-L, where $T_{\max} = 75$. The remaining datasets use a constant learning rate. Ordered-MNIST replaces the multilayer-perceptron encoder with the CNN encoder described in the main text. Model checkpoints and logs are stored in the configured results directory.

Table 6: Dataset-specific K^{2}SVD hyperparameters. "Epochs" is the total training budget, "Stage 1" is the number of Hilbert–Schmidt representation-learning epochs, and "Batches" is the maximum number of training batches per epoch. The training batch size is denoted by BS.

Dataset	Patch	Latent dim.	Epochs	Stage 1	Batches	BS	β	Koopman initialization
Electricity-L	24	16	70	15	100	64	0.01	dynamic
Weather-L	24	10	30	15	100	64	0.01	dynamic
Ordered-MNIST	16	10	70	10	60	64	0.01	dynamic
Exchange-L	24	16	50	10	100	64	0.01	dynamic
ETTh1-L	24	16	50	15	100	64	0.01	dynamic
ETTh2-L	24	10	50	15	100	32	0.01	dynamic
ETTm1-L	24	16	50	15	100	64	0.01	dynamic
ETTm2-L	24	10	50	15	100	64	0.025	dynamic
ILI-L	6	64	30	15	100	64	0.01	mixed

Dataset Statistics. Table 7 summarizes the dataset domains, dimensions, sampling frequencies, data ranges, and context and forecast horizons.

Table 7: Statistics and Description of the nine datasets.

Dimension Group	Dataset	#var.	range	freq.	Context	Horizon	Description
High-dimensional	Electricity-L	321	$\mathbb{R}^+$	H	96	96-720	Electricity consumption (Kwh)
	Ordered-MNIST	28*28	$(0, 255)$	NA	48	48-360	Periodic change of handwritten figures
	Weather-L	21	$\mathbb{R}^+$	10min	96	96-720	Local climatological data
Low-dimensional	ETTh1/h2-L	7	$\mathbb{R}^+$	H	96	96-720	Electricity transformer temperature per hour
	ETTm1/m2-L	7	$\mathbb{R}^+$	15min	96	96-720	Electricity transformer temperature every 15 min
	Exchange-L	8	$\mathbb{R}^+$	Busi. Day	96	96-720	Daily exchange rates of 8 countries
	ILI-L	7	$(0, 1)$	W	36	24-48	Ratio of patients seen with influenza-like illness

D Additional Empirical Results

D.1 Additional Results on Long-Term Time Series Prediction

Tables 9 and 10 summarize the NRMSE results for prediction lengths 96 and 720. Table 9 reports the low-dimensional datasets, including Exchange-L and the ETT series. These datasets have compact observation spaces, so the main question is whether the learned Koopman transition remains stable when the prediction horizon increases. K^{2}SVD performs competitively across this group and obtains the best results on Exchange-L for both prediction lengths, showing that the learned operator-induced transition space is useful even when the raw data dimension is small. On the ETT datasets, K^{2}SVD is strongest on ETTh1-L at prediction length 96 and on ETTh1-L, ETTh2-L, and ETTm2-L at prediction length 720, while K^2VAE and Koopa remain strong competitors on several shorter-horizon or lower-variance cases.

Table 10 reports the high-dimensional datasets, including Electricity-L, Ordered-MNIST, and Weather-L. These datasets are the most relevant for evaluating the computational motivation of K^{2}SVD, because the observations contain many coupled variables or image-like dynamic states. K^{2}SVD achieves the best results on Electricity-L and Ordered-MNIST at prediction length 96 and remains competitive at prediction length 720. On Weather-L, K^2VAE attains the best NRMSE, while K^{2}SVD remains close to the strongest baselines. Together with the efficiency results in the main text, these results suggest that K^{2}SVD provides a favorable accuracy-efficiency trade-off: it is not uniformly best on every dataset, but it is especially effective when a compact Koopman transition captures the dominant dynamics.

D.2 Additional Results on Long Prediction Efficiency

We further evaluate long-horizon inference efficiency to examine whether the computational advantage of K^{2}SVD persists when prediction requires many recursive rollout steps. The comparison includes FITS, iTransformer, Koopa, and K^2VAE because they represent the main classes of efficiency-competitive baselines used in this work. This set therefore compares K^{2}SVD against both general-purpose long-sequence predictors and methods that are structurally closest to it.

Table 8: Comparison of inference time in seconds and GPU usage in GBs on the selected high and low dimensional datasets when inference length is $T = 480$ for MNIST and $T = 720$ for the rest. For each metric, the best results are shown in **bold**, and the second-best results are underlined. "-" indicates that the run took too long to finish.

Model	Inference Time			GPU Usage		
	Electricity-L	MNIST	Weather-L	Electricity-L	MNIST	Weather-L
FITS	168.5 ± 4.654	564.423 ± 12.195	17.968 ± 4.434	8.723	24.407	0.938
TimeBase	251.976 ± 1.759	880.621 ± 15.625	23.327 ± 0.152	7.006	14.781	0.751
iTransformer	143.538 ± 1.071	380.082 ± 0.901	15.104 ± 4.055	8.724	24.408	0.951
Koopa	131.898 ± 1.382	453.949 ± 21.723	15.010 ± 3.968	8.739	24.433	0.948
K^2VAE	488.162 ± 8.377	2228.906 ± 74.189	45.667 ± 6.598	8.810	24.477	0.965
K^{2}SVD	**44.999 ± 0.063**	**120.644 ± 0.014**	**6.184 ± 0.039**	**1.910**	**5.052**	**0.210**

Table 9: Results of models using the NRMSE metric on low-dimensional datasets with prediction lengths 96 and 720. Best results are shown in **bold**, and second-best results are underlined. '*' indicates that the model encounters invalid computation due to instability.

Model	Exchange-L		ETTh1-L		ETTh2-L		ETTm1-L		ETTm2-L	
	96	720	96	720	96	720	96	720	96	720
FITS	0.0912 ± 0.0001	0.1233 ± 0.0030	0.6673 ± 0.0026	0.7967 ± 0.0160	0.3349 ± 0.0006	0.6068 ± 0.0277	0.5985 ± 0.0016	0.6674 ± 0.0087	0.2937 ± 0.0027	0.9151 ± 0.0695
TimeBase	0.0912 ± 0.0001	0.1209 ± 0.0030	0.6526 ± 0.0050	0.7433 ± 0.0033	0.2995 ± 0.0007	0.3669 ± 0.0019	0.5339 ± 0.0054	0.6835 ± 0.0007	0.2210 ± 0.0016	0.3203 ± 0.0008
iTransformer	0.1050 ± 0.0001	0.1581 ± 0.0020	0.6592 ± 0.0011	0.7152 ± 0.0010	0.3397 ± 0.0015	2.9661 ± 0.9167	0.6643 ± 0.0071	0.8523 ± 0.0278	0.3132 ± 0.0036	0.7747 ± 0.0898
Koopa	0.1051 ± 0.0002	0.1831 ± 0.0029	0.6832 ± 0.0040	0.7117 ± 0.0045	0.3568 ± 0.0010	0.3897 ± 0.0049	0.6204 ± 0.0066	**0.6376 ± 0.0016**	*	0.3572 ± 0.0052
TSDiff	0.1196 ± 0.0060	0.1632 ± 0.0087	0.8040 ± 0.0059	1.0635 ± 0.0437	0.5095 ± 0.0191	0.6274 ± 0.0022	0.6668 ± 0.0172	0.9958 ± 0.0154	0.4048 ± 0.0245	0.5154 ± 0.0097
GRU NVP	0.0662 ± 0.0002	0.1935 ± 0.0097	0.8307 ± 0.0249	0.9291 ± 0.0509	0.6020 ± 0.0639	8.1696 ± 2.1317	1.0492 ± 0.0669	3.6860 ± 0.7297	0.3811 ± 0.0319	2.0532 ± 1.0939
K^2VAE	0.0470 ± 0.0009	0.1145 ± 0.0012	0.6231 ± 0.0062	0.7199 ± 0.0144	**0.2895 ± 0.0035**	0.4678 ± 0.0353	**0.5330 ± 0.0054**	0.6810 ± 0.0090	**0.2131 ± 0.0006**	0.3662 ± 0.0125
K^{2}SVD	**0.0419 ± 0.0004**	**0.1139 ± 0.0010**	**0.6124 ± 0.0091**	**0.7064 ± 0.0012**	0.3251 ± 0.0157	**0.3476 ± 0.0006**	0.5555 ± 0.0010	0.6973 ± 0.0008	0.2179 ± 0.0011	**0.3192 ± 0.0004**

Table 8 shows that the advantage of K^{2}SVD becomes more pronounced under long prediction lengths. K^{2}SVD achieves the fastest inference on all three datasets, reducing inference time by about $2.4\times$–$5.5\times$ compared with the strongest non-K^{2}SVD baseline. Its GPU usage is also consistently the lowest, especially on MNIST, where the compact Koopman transition avoids carrying the high-dimensional observation space through each rollout step. These results support the central design of K^{2}SVD: by explicitly learning a compact operator-induced transition space, the model improves not only predictive accuracy but also the time, memory, and arithmetic cost of long-horizon forecasting.

Table 10: Results of models using the NRMSE metric on high-dimensional datasets with prediction lengths 96 and 720. Best results are shown in **bold**, and second-best results are underlined. '-' indicates that the model takes too long to finish training.

Model	Electricity-L		MNIST		Weather-L	
	96	720	48	360	96	720
FITS	1.3386 ± 0.0038	2.2749 ± 0.1578	2.0438 ± 0.0007	2.0486 ± 0.0004	0.6226 ± 0.0133	0.7892 ± 0.0018
TimeBase	0.9864 ± 0.1110	1.5179 ± 0.0863	1.8188 ± 0.0011	1.9886 ± 0.0012	0.2969 ± 0.0031	0.3107 ± 0.0006
iTransformer	1.3035 ± 0.0033	1.3560 ± 0.0063	2.0486 ± 0.0014	2.0336 ± 0.0003	0.3617 ± 0.0006	0.3769 ± 0.0000
Koopa	1.3977 ± 0.0183	1.3979 ± 0.0026	2.0510 ± 0.0007	2.0428 ± 0.0004	0.3656 ± 0.0004	0.3912 ± 0.0043
TSDiff	1.2107 ± 0.0587	-	-	-	0.3101 ± 0.0082	0.3623 ± 0.0237
GRU NVP	1.0617 ± 0.0299	-	-	-	0.7841 ± 0.1476	0.6221 ± 0.1136
K^2VAE	0.8682 ± 0.0106	**1.0890 ± 0.0084**	1.9817 ± 0.0379	2.0353 ± 0.00002	**0.2853 ± 0.0042**	**0.3084 ± 0.0025**
K^{2}SVD	**0.8502 ± 0.0014**	1.3035 ± 0.0155	**1.7844 ± 0.0007**	**1.9423 ± 0.0029**	0.3054 ± 0.0062	0.3470 ± 0.0009

D.3 Discussion of High-rank Dataset

Although ILI-L is low-dimensional in its observation space, Figure 4 shows that its Koopman spectrum decays more slowly than the other datasets. We therefore report it separately as a high-rank case. Table 11 shows that K^{2}SVD remains comparable but does not dominate this dataset. This behavior is consistent with the spectral evidence: when the effective Koopman rank is high, a compact latent space necessarily discards more non-negligible singular directions, so the resulting finite-dimensional transition cannot preserve as much of the predictive dynamics as it does on low-rank datasets.

ILI-L also exhibits properties that are less aligned with the assumptions behind the compact Koopman approximation used in K^{2}SVD. Influenza-like illness time series can undergo abrupt changes across epidemic phases, changes in dominant temporal patterns, and shifts between context and forecast windows. Prior Koopman forecasting work has emphasized that temporal shifts and non-stationary dynamics are central challenges in real-world time-series prediction Wang et al. (2022); Liu et al.

(2023). In such cases, a single low-dimensional transition operator is harder to estimate reliably: the regression-induced operator may be biased toward the context-window dynamics, and recursive long-horizon rollout can amplify this mismatch. Methods with stronger local adaptation or larger effective latent capacity can therefore be more favorable on ILI-L. We view this result as an informative limitation rather than a contradiction of the main findings: K^{2}SVD is designed to exploit datasets with dominant low-rank Koopman structure, while ILI-L represents a high-rank, temporally shifted regime where that structural advantage is weaker.

Table 11: Results of models using the NRMSE metric on ILI-L with prediction lengths 24 and 60. Best results are shown in **bold**, and second-best results are underlined.

Model	24	60
FITS	0.7467 ± 0.0288	0.7199 ± 0.0256
iTransformer	0.7834 ± 0.0125	0.7791 ± 0.0221
Koopa	1.0881 ± 0.0822	0.9026 ± 0.0146
TSDiff	0.6929 ± 0.0246	0.7332 ± 0.0068
GRU NVP	**0.2934 ± 0.0269**	0.4328 ± 0.0258
CSDI	0.7214 ± 0.0416	0.7856 ± 0.0211
K^2VAE	0.2971 ± 0.0103	**0.4246 ± 0.0051**
K^{2}SVD	0.4579 ± 0.0042	0.4630 ± 0.0049

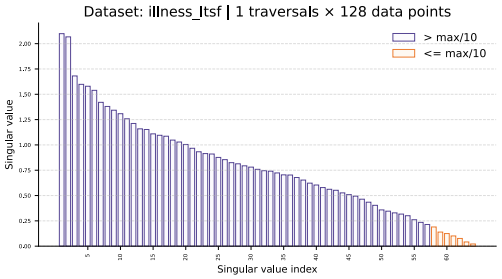


Figure 4: The singular values of the Koopman space exhibit higher rank properties on ILI-L.

D.4 Additional Ablation Study

D.4.1 Ablation on Estimation of Koopman

As described in the method section, K^{2}SVD uses a trainable finite-dimensional Koopman matrix initialized from the regression estimate. To show its benefit, we ablate this update by fixing the Koopman matrix to $\mathsf{K}_{\mathbf{b}}^{\mathrm{ols}}$ and optimizing only the latent correction parameters. We denote this static-matrix scheme as K^{2}SVD-S and report its results on multiple datasets. Table 12 shows that K^{2}SVD consistently improves over K^{2}SVD-S on the evaluated datasets. This improvement is consistent with the motivation in the overall algorithm: the static regression-induced matrix is obtained from finite-sample one-step latent regression and can be affected by approximation error and model mismatch.

Table 12: Ablation study on the two training schemes using the NRMSE metric. K^{2}SVD-S uses the regression-induced Koopman matrix, while K^{2}SVD dynamically updates the Koopman matrix during downstream forecasting training.

Model	Electricity-L	MNIST	Weather-L	Exchange-L
K^{2}SVD-S	1.0417 ± 0.0692	2.0134 ± 0.0017	0.3209 ± 0.0016	0.0441 ± 0.0016
K^{2}SVD	**0.8502 ± 0.0014**	**1.8125 ± 0.0010**	**0.3124 ± 0.0011**	**0.0419 ± 0.0004**

D.4.2 Ablation on LoRA Objective

We argue that optimizing the encoder with the LoRA objective provides an efficient and expressive Koopman space. In comparison to end-to-end methods, the encoder trained with the HS objective and frozen in later steps creates a more stable low-rank space, and as the inference length increases it limits accumulated rollout error and improves reconstruction. To support this claim, we set the context $t = 96$, horizon $T = 720$, remove Stage 1, and directly train the encoder, Koopman matrix K_θ, and correction parameters end-to-end using only $\mathcal{L}_e$. As shown in Table 13, the two-stage scheme achieves better performance under the same hidden-space dimension on MNIST, Electricity, and Exchange, and comparable results on Weather.

Table 13: Ablation study on LoRA objective. NRMSE dropps on 3 out of 4 dataset shows its advantage on long term forcasting.

Ablation	Electricity-L	MNIST	Weather-L	Exchange-L
End-to-end	1.6286 ± 0.1900	1.9759 ± 0.0718	**0.3380 ± 0.0013**	0.1239 ± 0.0019
2-Stage	**1.3035 ± 0.0155**	**1.9423 ± 0.0029**	0.3470 ± 0.0009	**0.1139 ± 0.0010**