scientific reports

OPEN

Temporally consistent Koopman autoencoders for forecasting dynamical systems

Indranil Nayak[1,2,4,5], Ananda Chakrabarti[2,5], Mrinal Kumar[1,3], Fernando L. Teixeira[1,2] & Debdipta Goswami[2,3]✉

Absence of sufficiently high-quality data often poses a key challenge in data-driven modeling of high-dimensional spatio-temporal dynamical systems. Koopman autoencoders (KAEs) harness the expressivity of deep neural networks (DNNs), the dimension reduction capabilities of autoencoders, and the spectral properties of the Koopman operator to learn a reduced-order feature space with simpler, linear dynamics. However, the effectiveness of KAEs is hindered by limited and noisy training datasets, leading to poor generalizability. To address this, we introduce the temporally consistent Koopman autoencoder (tcKAE), designed to generate accurate long-term predictions even with limited and noisy training data. This is achieved through a consistency regularization term that enforces prediction coherence across different time steps, thus enhancing the robustness and generalizability of tcKAE over existing models. We provide analytical justification for this approach based on Koopman spectral theory and empirically demonstrate tcKAE's superior performance over state-of-the-art KAE models across a variety of test cases, including simple pendulum oscillations, kinetic plasma, and fluid flow data.

Keywords Koopman, Machine learning, Neural networks

Long-term forecasting of sequential time-series data generated from a nonlinear dynamical system is a central problem in engineering. For classification or prediction of such time-series data, a precise analysis of the underlying dynamical system provides valuable insights to generate appropriate features and interpretability to any computational algorithm. This gives rise to a family of physics-constrained learning (PCL) algorithms that incorporate constraints arising from physical consistency of the governing dynamical system.

Recently, a physics constrained approach for time-series learning based on Koopman methods[1] has been introduced. This approach uses an infinite-dimensional linear operator that encodes a nonlinear dynamics completely. The Koopman operator[2] provides a computationally preferable linear method to analyze and forecast dynamical systems. However, Koopman operator maps between infinite-dimensional function spaces, and hence, cannot be *in general* computationally represented without a finite-dimensional projection. Machine learning techniques utilize a finite-dimensional approximation of the Koopman operator by assuming the existence of a finite-dimensional Koopman invariant function space. This is usually achieved by an autoencoder network that maps the state-space into a latent space where the dynamics can be linearly approximated by the finite-dimensional Koopman operator encoded via a linear layer. This Koopman autoencoder (KAE) approach is attractive as it strikes a balance between the expressivity of deep neural networks and interpretability of PCL. Also, Koopman modal approximation can be readily used for stability analysis[3] and control[4,5] of the underlying dynamical system in a data-driven fashion. Recent literature[6] also investigates the existence of a backward Koopman operator in order to impose an extra consistency constraint on the latent space linear map, giving rise to consistent Koopman autoencoder (cKAE) algorithm.

This manuscript proposes a new algorithm for consistent long-term prediction of time-series data generated from a nonlinear (possibly high-dimensional) dynamical system using a temporal consistency constraint with Koopman autoencoder framework. This algorithm, dubbed as temporally consistent Koopman autoencoder (tcKAE), compares the predictions from different initial time-instances to a final time in the *latent space*. This is

[1]ElectroScience Laboratory, The Ohio State University, Columbus, OH 43212, USA. [2]Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA. [3]Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH 43210, USA. [4]Present address: SLAC National Accelerator Laboratory, Stanford University, Menlo Park, CA 94025, USA. [5]Indranil Nayak and Ananda Chakrabarti contributed equally to this work. ✉email: goswami.78@osu.edu

in contrast with prior KAE methodologies, where the enforcement of multi-step look-ahead prediction loss is reliant on *labeled* data and/or the latent space representation of the labeled data[6–8]. Typically larger look-ahead step during training results in greater prediction accuracy, at the cost of training time. However, having large look-ahead step is a constraint when such data is scarce. The proposed tcKAE is different since it evaluates predictions relative to each other, bypassing direct comparison with the encoded version of the *labeled* data. This alleviates limitations on the maximum look-ahead step imposed by a limited training dataset. However, as discussed in Methods section, selecting arbitrarily large look-ahead steps to enforce consistency among predictions is neither practical nor optimal. Instead, a relatively small number of look-ahead steps can yield comparable or even superior results compared to state-of-the-art Koopman neural network models. The key advantage of the tcKAE algorithm over cKAE[6] lies in its ability to operate without assuming the existence of backward dynamics. It is analytically shown that a KAE spanning a Koopman invariant subspace must satisfy the temporal consistency constraint and by enforcing it, tcKAE leads to higher expressivity and generalizability. This enforcement of temporal consistency in effect reduces the sensitivity of the KAE to noise, decreases the variance or uncertainty in future predictions, and make it more robust with less training data as shown in the experiments. Consistency regularization have been used for semi-supervised learning in classification problems[9–11], but to the best of our knowledge, this is the first attempt to regularize multi-step temporal consistency for dynamical system learning.

Background

Recurrent neural networks

The idea of forecasting dynamical systems using neural networks, particularly using recurrent neural networks (RNNs), dates back to several decades [12,13]. Recent advances in big data, machine learning algorithms and computational hardware, have rekindled the interest in this domain. Emergence of innovative architectures such as Long Short-Term Memmory (LSTM)[14], or gated neural networks (GRUs)[15] have enhanced the prediction capabilities of NN models significantly. However, training an RNN is fraught with vanishing and exploding gradient problems[16] which can be solved by approaches like analyzing stability[17] or using unitary hidden weight matrices[18]. But these can affect the short term modeling capability by reducing the expressivity of the RNNs[19]. Another drawback of RNNs is the lack of generalizability and interpretability especially when used for predicting a physical system. To make RNN physically consistent, a number of methods are presented to take physical constraints into account. These range from making a physics guided architecture[20], relating it to dynamical systems[21] or differential equations[22]. Hamiltonian neural networks[23] aims to learn the conserved Hamiltonian as a physical constraint but works only for lossless systems. In this paper, we use the linear operator-based recurrent archiecture along with a physical prediction constraint to develop a more generalizable prediction model.

Koopman-based methods

The Koopman operator-based approach exploits the linearity[1] in an infinite dimensional function space to yield a linearly recurrent prediction scheme. A number of non-neural dictionary-based approach relying on the dynamic mode decomposition (DMD)[24] are proposed[4,5,25–27]. However, these methods implicitly assumes a dictionary spanning a Koopman invariant subspace. Neural approaches using Koopman autoencoder (KAE) provide a better alternative to define a Koopman invariant function space where the dynamics can be linearly approximated[7,8,28]. [29] showcased the superiority of Koopman-based spectral methods over LSTM, Gated Recurrent Unit (GRU), and Echo State Networks (ESNs) due its ability to extract "slow" frequencies which can have significant effect for long-term predictions. Furthermore, the autoencoder architecture is particularly well-suited (order reduction) for modeling high-dimensional physical systems, the primary goal of this work. However, most of the KAE models just focus on multi-step prediction errors and without any test on consistency of predictions. Recent work[6] on backward Koopman operator provides a consistency test by making the predictions forward and backward consistent, but works only when a backward dynamics is well-defined.

Koopman theory: an overview

Consider a discrete-time dynamical system on a N_d-dimensional compact manifold $\mathscr{M}$, evolving according to the flow-map $\mathrm{f} : \mathscr{M} \mapsto \mathscr{M}$:

$$\mathrm{x}_{n+1} = \mathrm{f}(\mathrm{x}_n), \quad \mathrm{x}_n \in \mathscr{M}, \quad n \in \mathbb{N} \cup \{0\}. \tag{1}$$

Let $\mathscr{F}$ be a Banach space of complex-valued observables $\psi : \mathscr{M} \to \mathbb{C}$. The discrete-time *Koopman operator* $\mathscr{K} : \mathscr{F} \to \mathscr{F}$ is defined as

$$\mathscr{K}\psi(\cdot) = \psi \circ \mathrm{f}(\cdot), \quad \text{with} \quad \psi(\mathrm{x}_{n+1}) = \mathscr{K}\psi(\mathrm{x}_n) \tag{2}$$

where $\mathscr{K}$ is infinite-dimensional, and linear over its argument. The scalar observables ψ are referred to as the Koopman observables. Koopman eigenfunctions ϕ are special set of Koopman observables that satisfy $(\mathscr{K}\phi)(\cdot) = \lambda\phi(\cdot)$, with an eigenvalue $\lambda \in \mathbb{C}$. Considering the Koopman eigenfunctions span the Koopman observables, a vector valued observable $\mathrm{g} \in \mathscr{F}^p = [\psi_1 \ \psi_2 \ \dots \ \psi_p]^\mathrm{T}$ can be expressed as a sum of Koopman eigenfunctions $\mathrm{g}(\cdot) = \sum_{i=1}^{\infty} \phi_i(\cdot)\mathrm{v}_i^\mathrm{g}$, where $\mathrm{v}_i^\mathrm{g} \in \mathbb{R}^p, i = 1, 2, \dots,$ are called the *Koopman modes* of the observable $\mathrm{g}(\cdot)$. This modal decomposition provides the growth/decay rate $|\lambda_i|$ and frequency $\angle\lambda_i$ of different Koopman modes via its time evolution

$$\mathrm{g}(\mathrm{x}_t) = \sum_{i=1}^{\infty} \lambda_i^t \phi_i(\mathrm{x}_0)\mathrm{v}_i^{\mathrm{g}}. \tag{3}$$

The Koopman eigenvalues and eigenfunctions are properties of the dynamics only, whereas the Koopman modes depend on the observable.

Koopman modes can be analyzed to understand the dominant characteristics of a complex dynamical system and getting traction in fluid mechanics[1], plasma dynamics[30,31], control systems[4], unmanned aircraft systems[32], and traffic prediction[33]. In addition, it is also being used for machine learning tasks and training deep neural networks[34]. Several methods have also been developed to compute the Koopman modal decomposition, e.g., DMD and EDMD[24,25], Ulam-Galerkin methods, and Koopman autoencoder (KAE) deep neural networks[28,35]. In this paper, we primarily focus on long-term prediction of high-dimensional autonomous dynamical systems using Koopman modes with autoencoder networks.

Temporally consistent Koopman autoencoders (tcKAE) is a KAE neural network architecture developed for long-term prediction of high-dimensional nonlinear dynamical systems, primarily aimed for data-scarce scenarios. It enforces the temporal-consistency among predictions across different time-steps. This consistency requirement enhances the robustness of the resultant algorithm against noise, reduces variance, improves the prediction accuracy, and reduce the amount of training data required for similar level of prediction accuracy.

Methods

Prediction via Koopman invariant subspace: Koopman autoencoders (KAE)

Koopman operator, being an infinite-dimensional one, must be projected onto a finite-dimensional basis for any practical prediction algorithm. The equation in (3) can be truncated to finite set of terms in the presence of a finite-dimensional Koopman invariant subspace. However, discovering such finite-dimensional Koopman invariant subspaces purely from data is a challenging task, and an active area of research. Recent works[6–8,28] leverage the nonlinear function approximation capabilities of neural networks to find a suitable transformation from the state space $\mathscr{M}$ to a Koopman invariant subspace via a latent state (z). These models are encompassed within the broader umbrella of Koopman autoencoders (KAE) which strives to find a suitable transformation from state space to the Koopman observable space where the dynamics is linear, and can be easily learned. The basic operation of KAE can be decomposed into three components (4), *i)* encoding $(\Psi_e(\cdot)) : \mathscr{M} \to \mathbb{R}^{N_l}$ that transforms the original state x to a Koopman observable $\mathrm{z} \in \mathbb{R}^{N_l}$, *ii)* advancing the dynamics in that transformed space through a linear operator $(K \in \mathbb{R}^{N_l \times N_l})$, and *iii)* decoding $(\Psi_d(\cdot))$ back to the original state space:

$$\mathrm{z}_n \approx \Psi_e(\mathrm{x}_n) \quad \rightarrow \quad \mathrm{z}_{n+k} = K^k \cdot \mathrm{z}_n \quad \rightarrow \quad \mathrm{x}_{n+k} \approx \Psi_d(\mathrm{z}_{n+k}) = \hat{\mathrm{x}}_{n+k}, \tag{4}$$

where K is the finite-dimensional restriction of $\mathscr{K}$, which advances z by k time-steps (or time samples), and $\hat{\mathrm{x}}$ denotes the KAE reconstruction of the state. Note that for traditional autoencoders the encoding operation typically result in dimensionality reduction which is beneficial when dealing with high-dimensional dynamical systems. Traditionally, since we are mostly interested in future prediction of the state, KAE is trained by minimizing the difference between $\hat{\mathrm{x}}_{n+k}$ and x_{n+k}, with $k \in \mathbb{Z}_+ \cup \{0\}$. The loss functions traditionally used for training KAEs are the identity loss $\mathscr{L}_{\mathrm{id}}$ $(k=0)$, and the forward loss L_{fwd} $(k = 1, 2, \ldots)$[6].

$$\mathscr{L}_{id} = \frac{1}{2M}\sum_{n=1}^{M} ||\hat{\mathrm{x}}_n - \mathrm{x}_n||_2, \quad \mathscr{L}_{fwd} = \frac{1}{2k_m M}\sum_{k=1}^{k_m}\sum_{n=1}^{M} ||\hat{\mathrm{x}}_{n+k} - \mathrm{x}_{n+k}||_2^2, \tag{5}$$

where $||\cdot||_2$ is the 2-norm, M is the number of samples over which we want to enforce the loss, and k_m is the maximum value of k, i.e. the maximum look-ahead step for multi-step training. Recent work in[6] demonstrated that including backward dynamics $(K_b \in \mathbb{R}^{N_l \times N_l},\ \Psi_d \circ K_b^k \circ \Psi_e(\mathrm{x}_n) = \hat{\mathrm{x}}_{n-k})$ as shown in Fig. 1 leads to better stability resulting in more accurate long-term predictions. The idea is to incorporate a backward loss term $(\mathscr{L}_{\mathrm{bwd}})$ with forward-backward consistency loss $(\mathscr{L}_{\mathrm{con}})$,

$$\mathscr{L}_{bwd} = \frac{1}{2k_m M}\sum_{k=1}^{k_m}\sum_{n=1}^{l} ||\hat{\mathrm{x}}_{n-k} - \mathrm{x}_{n-k}||_2^2 \tag{6}$$

$$\mathscr{L}_{con} = \sum_{i=1}^{N_b} \frac{1}{2i}||K_{bi\star}K_{\star i} - I_{N_l}||_F + \frac{1}{2i}||K_{\star i}K_{bi\star} - I_{N_l}||_F, \tag{7}$$

where N_l is the latent space dimension, $K_{bi\star}$ is the upper i rows of K_b, $K_{\star i}$ is the i left most columns of K, $||\cdot||_F$ denotes the Frobenius norm, and $I_{N_l} \in \mathbb{R}^{N_l \times N_l}$ is the identity matrix of dimension $N_l \times N_l$.

Temporally consistent Koopman autoencoder (tcKAE)

In this section, we describe our proposed temporally consistent Koopman autoencoder (tcKAE) architecture. The key idea, as illustrated in Fig. 2, is to introduce a consistency regularization term for training KAE in order to enforce consistency among future predictions. The fundamental idea derives from the time-invariance property of the autonomous dynamical systems. In the context of (1), the temporal consistency can be stated as the direct result of the following:

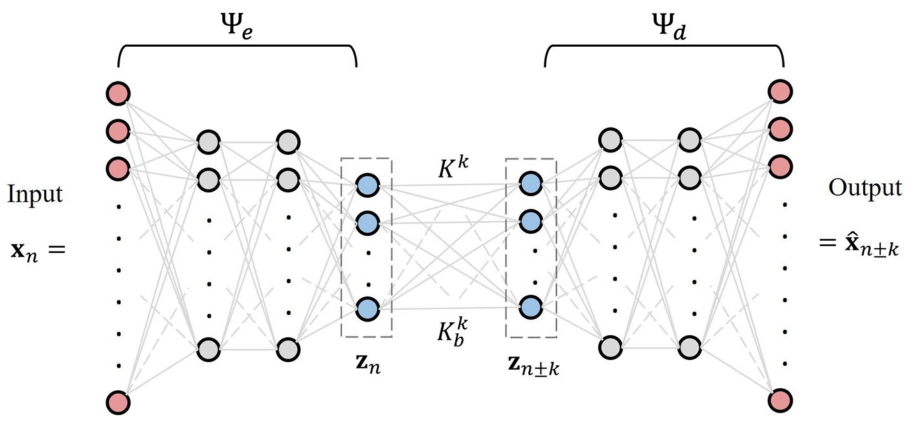


Fig. 1. cKAE architecture where the layers with red nodes indicate the input and output layer in the original state space. The latent space is represented by the bottleneck layer with blue nodes. The hidden layers are shown by the gray nodes. The encoder and decoder layers are denoted by Ψ_e and Ψ_d respectively. The linear layers represent the linear forward (K) and backward dynamics (K_b).

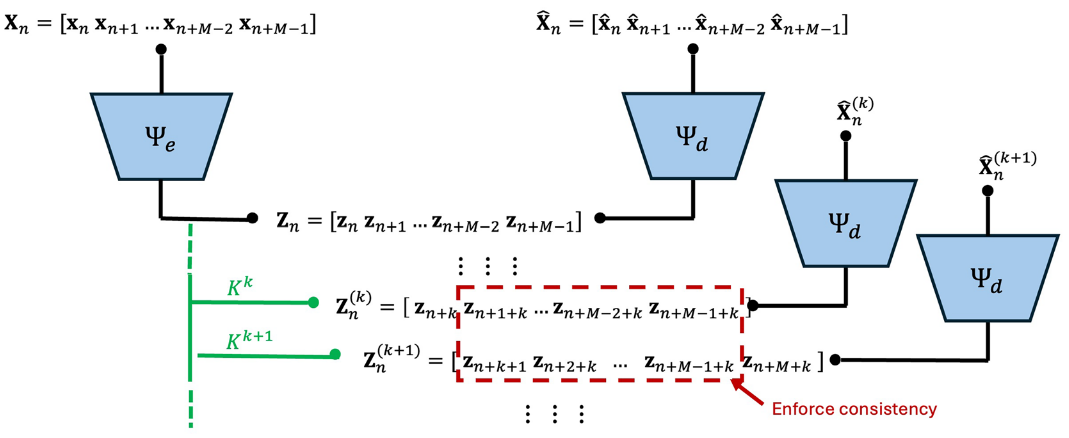

$\mathbf{X}_n = [\mathbf{x}_n\ \mathbf{x}_{n+1} \dots \mathbf{x}_{n+M-2}\ \mathbf{x}_{n+M-1}]$
$\hat{\mathbf{X}}_n = [\hat{\mathbf{x}}_n\ \hat{\mathbf{x}}_{n+1} \dots \hat{\mathbf{x}}_{n+M-2}\ \hat{\mathbf{x}}_{n+M-1}]$
Ψ_e
Ψ_d
$\hat{\mathbf{X}}_n^{(k)}$
$\hat{\mathbf{X}}_n^{(k+1)}$
$\mathbf{Z}_n = [\mathbf{z}_n\ \mathbf{z}_{n+1} \dots \mathbf{z}_{n+M-2}\ \mathbf{z}_{n+M-1}]$
K^k
K^{k+1}
$\mathbf{Z}_n^{(k)} = [\mathbf{z}_{n+k}\ \mathbf{z}_{n+1+k} \dots \mathbf{z}_{n+M-2+k}\ \mathbf{z}_{n+M-1+k}]$
$\mathbf{Z}_n^{(k+1)} = [\mathbf{z}_{n+k+1}\ \mathbf{z}_{n+2+k} \dots \mathbf{z}_{n+M-1+k}\ \mathbf{z}_{n+M+k}]$
Enforce consistency

Fig. 2. Illustration of enforcement of temporal consistency among predictions in Koopman invariant latent space.

$$\mathrm{f}^k(\mathrm{x}_n) = \mathrm{f}^{k+\kappa}(\mathrm{x}_{n-\kappa}), \quad \forall n, k \in \mathbb{Z}_+ \cup \{0\}, \quad \kappa = 1, 2, \dots, n, \tag{8}$$

where $\mathrm{f}^k = \mathrm{f} \circ \mathrm{f} \circ \dots (k \text{ times}) \dots \circ \mathrm{f}$, with $\circ$ being the composition operator. One key aspect of our approach is that instead of enforcing this consistency in the original state space, we do so in the Koopman invariant subspace (latent space). This helps to avoid computation in high-dimensional space and provides robustness to noise. We justify enforcing such consistency in the latent space by summarizing the desired property of a consistent autoencoder Ψ_e and constructing the architecture of this framework. The autoencoder $\Psi_e(\cdot) = [\psi_1(\cdot), \dots, \psi_{N_l}(\cdot)]$ defines a vector-valued observable of the state space $\mathscr{M}$ where each ψ_i is a scalar-valued function. Let $\mathscr{G}$ be the span of $\{\psi_1, \dots, \psi_{N_l}\}$. In order for the linear recurrent matrix K to asymptotically approximate the projection of the Koopman operator $\mathscr{K}$ on $\mathscr{G}$, the latter must be Koopman invariant, i.e., $\mathscr{K}\mathscr{G} \subsetneq \mathscr{G}$. The Koopman invariance will ensure the linear recurrent structure

$$\hat{\mathrm{x}}_{n+1} = \Psi_d \circ K \circ \Psi_e(\mathrm{x}_n). \tag{9}$$

The proposed architecture introduces a multi-step temporal consistency loss (not applied against the directly encoded version of the labeled data) in the latent space in order to strengthen the Koopman invariance of the latent function space $\mathscr{G}$. It is important to emphasize that, for calculating temporal consistency, comparisons are made between multiple forward predictions in the latent space. These predictions are not necessarily constrained by the encoded versions of the available training data. In the following, we summarize the desirable temporal consistency of the encoder functions upon which the proposed method is constructed.

Theorem 1 Let $\Psi_e(\cdot) = [\psi_1(\cdot), \ldots, \psi_{N_l}(\cdot)]^T \in \mathscr{F}^{N_l}$ denote a vector valued function (encoder) comprised of scalar functions $\psi_i(\cdot) \in \mathscr{F}$. The latent space $\mathscr{G} = \text{span}\{\psi_1(\cdot) \ldots, \psi_{N_l}(\cdot)\}$ forms a Koopman invariant subspace with respect to the system dynamics (1) if and only if there exists $K \in \mathbb{R}^{N_l \times N_l}$ such that

$$\Psi_e(\mathrm{x}_{n+k}) = K^k \Psi_e(\mathrm{x}_n) \tag{10}$$

for all $n \geq 0$ and $k \geq 1$.

Proof ($\Rightarrow$) Suppose $\Psi_e(\cdot)$ is defined such that $\mathscr{G}$ is a Koopman-invariant subspace. We prove this part through induction. For any $n \geq 0$, (10) is satisfied for $k = 1$ with a matrix K where $K = \mathscr{K}_{\mathscr{G}}$, i.e., the restriction of the Koopman operator $\mathscr{K}$ on $\mathscr{G}$. Now suppose that it is satisfied for a fixed $k \in \mathbb{N}$. Then, $\Psi_e(\mathrm{x}_{n+k+1}) = \Psi_e \circ \mathrm{f}(\mathrm{x}_{n+k}) = \mathscr{K}\Psi_e(\mathrm{x}_{n+k}) = \mathscr{K}K^k\Psi_e(\mathrm{x}_n) = K^k\mathscr{K}\Psi_e(\mathrm{x}_n) = K^k\Psi_e \circ \mathrm{f}(\mathrm{x}_n) = K^k\Psi_e(\mathrm{x}_{n+1}) = K^{k+1}\Psi_e(\mathrm{x}_n)$, where the first two steps are by the definition of Koopman operator, next two steps are by the linearity of the same, and the last step is by the induction anchor.

($\Leftarrow$) Suppose $\exists\ K \in \mathbb{R}^{N_l \times N_l}$ such that (10) is satisfied for all $n \geq 0$ and $k \geq 1$. Then it must be satisfied for $n = 0$ and $k = 1$, i.e., $\Psi_e(\mathrm{x}_1) = K\Psi_e(\mathrm{x}_0)$. Since x_0 can be anything in $\mathscr{M}$, this yields $\Psi_e \circ \mathrm{f}(\mathrm{x}) = K\Psi_e(\mathrm{x})$ for all $\mathrm{x} \in \mathscr{M}$, i.e., $\Psi_e \circ \mathrm{f}(\cdot) = K\Psi_e(\cdot)$. Let $g \in \mathscr{G}$ be an observable. Then, $\exists$ $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{N_l}] \in \mathbb{R}^{N_l}$ such that $g(\cdot) = \sum_{i=1}^{N_l} \alpha_i \psi_{N_i}(\cdot) = \boldsymbol{\alpha}^\top \Psi_e(\cdot)$. Now, $\mathscr{K}g(\cdot) = g \circ \mathrm{f}(\cdot) = \boldsymbol{\alpha}^\top \Psi_e \circ \mathrm{f}(\cdot)$ $= \boldsymbol{\alpha}^\top K\Psi_e(\cdot) = \sum_{j=1}^{N_l}\left(\sum_{i=1}^{N_l} \alpha_i K_{ij}\right)\psi_j(\cdot)$, i.e., $\mathscr{K}g \in \mathscr{G}$. Hence, $\mathscr{G} = \text{span}\{\psi_1(\cdot) \ldots, \psi_{N_l}(\cdot)\}$ forms a Koopman-invariant subspace. □

Theorem 1 ensures that, in the latent space, encoded state at any time instant must be equal to the encoded state of any previous time multiplied with K^k where *k* denotes the shift in terms of time samples between those two instances. It also forces the latent state to have linear time-invariant dynamics with a constant state-transition matrix *K*. Theorem 1 provides a method to ensure the autoencoder span a Koopman invariant subspace by enforcing temporal consistency as follows.

Corollary 1 For a KAE $\Psi_e : \mathscr{M} \to \mathbb{R}^{N_l}$, if the associated function space $\mathscr{G}$ is Koopman invariant, then for any $n \geq 0$ and $k \geq 1$, $K^{k-p}\Psi_e(\mathrm{x}_{n+p}) = K^{k-q}\Psi_e(\mathrm{x}_{n+q})$ for all $p, q = 1, \ldots, k-1$, $(p \neq q)$ i.e.,

$$K^{k-1}\Psi_e(\mathrm{x}_{n+1}) = K^{k-2}\Psi_e(\mathrm{x}_{n+2}) = \ldots = K\Psi_e(\mathrm{x}_{n+k-1}).$$

Corollary 1 is essentially heart of the temporal consistency loss. In order to define our loss function, let us consider the reference time-step to be *n*. For a batch size of *M* (in latent space $\mathrm{Z}_n = [\mathrm{z}_n\ \mathrm{z}_{n+1} \ldots\ \mathrm{z}_{n+M-1}]$), let the KAE advance the dynamics up to k_{tm} time-steps, giving rise to a set of batches $\mathrm{Z}_n^{(k)} = [\mathrm{z}_{n+k}^{(k)}\ \mathrm{z}_{n+1+k}^{(k)} \cdots\ \mathrm{z}_{n+M+k-1}^{(k)}]$ with $k = 1, 2, \ldots, k_{tm}$, where the superscript (*k*) denotes the number of operations of *K*. The consistency loss $\mathscr{L}_{\text{tc}}$ (Fig. 2) can be defined as,

$$\mathscr{L}_{\text{tc}} = \frac{1}{2(k_{tm}-1)} \sum_{q=1}^{k_{tm}-1} \mathscr{L}_q, \quad \mathscr{L}_q = \frac{1}{(k_{tm}-q)} \sum_{k=1}^{k_{tm}-q} \frac{1}{(M-q)} \sum_{p=q}^{M-1} ||\mathrm{z}_{n+k+p}^{(k)} - \mathrm{z}_{n+k+p}^{(k+q)}||_2^2, \tag{11}$$

where $\mathscr{L}_q$ takes into account the common terms between two batches shifted by *q* time samples. The total loss ($\mathscr{L}_{\text{tot}}$) for training tcKAE is given by,

$$\mathscr{L}_{\text{tot}} = \gamma_{\text{id}}\mathscr{L}_{\text{id}} + \gamma_{\text{fwd}}\mathscr{L}_{\text{fwd}} + \gamma_{\text{bwd}}\mathscr{L}_{\text{bwd}} + \gamma_{\text{con}}\mathscr{L}_{\text{con}} + \gamma_{\text{tc}}\mathscr{L}_{\text{tc}}, \tag{12}$$

where $\gamma_{\text{id}}, \gamma_{\text{fwd}}, \gamma_{\text{bwd}}, \gamma_{\text{con}}$ and γ_{tc} are the corresponding scaling factors.

Remark 1 The temporal consistency loss enforces a Koopman invariant subspace by virtue of the time-homogeneity of the latent-state dynamics, i.e., the Koopman operator is time-invariant for an autonomous system. This provides a *physics constrained learning* method for dynamical system prediction even from a smaller dataset.

Simulation setup

In this section we discuss the simulation setup in details for both the oscillating electron beam and flow past cylinder.

Oscillating electron beam

A two-dimensional (2-D) electron beam (Fig. 3a) is simulated inside a square cavity of dimension $1\text{ cm} \times 1\text{ cm}$ using a charge-conserving electromagnetic particle-in-cell (EMPIC) algorithm[36]. The electron beam is propagating along the +ve *y* direction under the influence of an oscillating transverse magnetic flux. The solution domain is discretized using irregular triangular mesh (grey lines in Fig. 3a) with $N_0 = 844$ nodes, $N_1 = 2447$ edges and $N_2 = 1604$ elements (triangles). The blue dots in Fig. 3a represent the superparticles which are discretized representation of the phase-space of electrons (delta distribution in both position and velocity). The superparticles are essentially the point charges with charge $q_{sp} = r_{sp}q_e$, and mass $m_{sp} = r_{sp}m_e$,

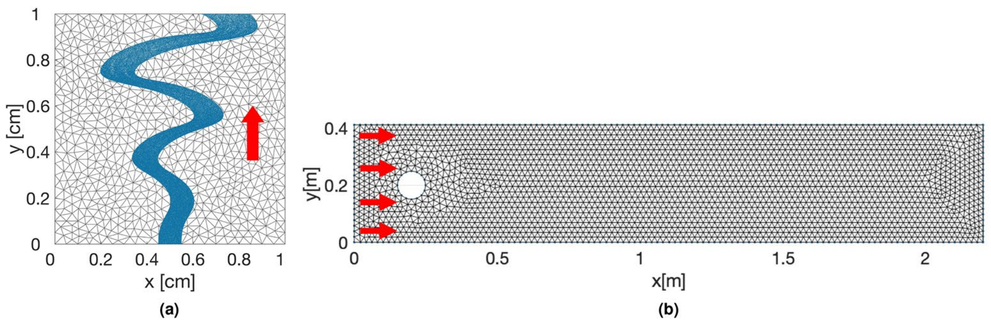


Fig. 3. Simulation setup for benchmark dynamical systems: (**a**) Snapshot of the 2D electron beam (blue dots indicates the charged particles) at $t = 8$ ns with unstructured triangular mesh. Red arrow indicates the beam direction.; (**b**) Irregular triangular mesh for flow-past cylinder finite element simulation. Red arrows indicate the flow direction.

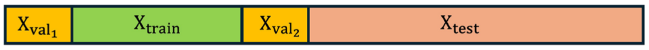


Fig. 4. Splitting of time-series dataset for training, validation and testing.

where r_{sp} is the superparticle ratio with q_e and m_e respectively representing the charge and mass of an electron. We select $r_{sp} = 5000$. The superparticles are injected from the bottom of the cavity randomly with uniform distribution over the region $[0.45 \text{ cm}, \ 0.55 \text{ cm}]$ with the injection rate of 50 superparticles per time-step. The superparticles are injected along the vertical direction with $v_y = 5 \times 10^6$ m/s. The external oscillating magnetic flux can be represented by $\mathscr{B}_{ext}(t) = \mathscr{B}_0 \ \sin(2\pi t/T_{osc}) \ \hat{z}$ with $\mathscr{B}_0 = 2.5 \times 10^{-2}$ T, and $T_{osc} = 0.8$ ns. The time-step for the EMPIC simulation is taken to be $\Delta t = 0.2$ ps.

Flow past cylinder
The 2-D solution domain $(2.2 \text{ m} \times 0.41 \text{ m})$ is discretized using irregular triangular mesh (Fig. 3b) with number of nodes $N_0 = 2647$, and number of elements (triangles) $N_2 = 5016$. The cylinder has the diameter of 0.1 m with center of the cylinder located at $(0.2 \text{ m}, 0.2 \text{ m})$. The flow is assumed to be incompressible, and governed by the Navier-Stokes equations with u, v denoting the horizontal and vertical component of the velocity respectively while p denotes the pressure field. The density of the fluid is set to $\rho = 1$ Kg/m^3, and dynamic viscosity $\mu = 0.001$ Kg/m s. The flow is unsteady with a maximum velocity of 1 m/s and mean velocity $\frac{2}{3}$ of the maximum velocity. The simulation starts with initial conditions $\mathrm{u}_0 = \mathrm{v}_0 = 0$ and $\mathrm{p}_0 = 0$. For the boundary conditions, the leftmost boundary is set as an inlet with a parabolic velocity profile. This is representative of fully developed laminar flow at the inlet. The rightmost boundary is set as an outflow (pressure boundary), where we specify the pressure but do not specify the velocity, allowing the flow to exit naturally based on the internal flow field. All other boundaries are treated as walls with a no-slip condition, which means the fluid velocity at the walls is zero. The simulation runs for a total of 80 seconds, with a time-step size of 0.01 seconds.

Network architecture & training strategy

The overall KAE network architecture is shown in Fig. 1 and the tcKAE architecture is shown in Fig. 2. Note that the network topology is same as that in[6]. The number of nodes in the input and output layer (red) is indicated by N_{in} and N_{out} ($N_{\text{in}} = N_{\text{out}}$). N_l denotes the number of nodes in the bottleneck layer (blue), which represents the approximation of Koopman-invariant latent space. Number of nodes in each hidden layers (two hidden layers for each encoder and decoder) is represented by N_h (gray nodes). We keep N_h same for both the hidden layers and it is tuned in multiple of 16. We use the *tanh* activation function for each layer except the bottleneck layer.

The experiments were carried out on the Ohio Supercomputer's (OSC)[37] Owens cluster, consisting of 842 Dell Nodes and 648 compute nodes with Dell PowerEdge C6320 two-socket servers, Intel Xeon E5-2680 v4 (Broadwell, 14 cores, 2.40 GHz) processors and 128 GB memory. As shown in Fig. 4, we partition the dataset X into training ($\mathrm{X}_{\text{train}}$), validation ($\mathrm{X}_{\text{val}_1}, \mathrm{X}_{\text{val}_2}$) and testing ($\mathrm{X}_{\text{test}}$) set. For a set of hyperparameters, the tcKAE architecture is trained to reduce the training loss $\mathscr{L}_{\text{tot}}$ as defined in 12. The set of hyperparameters which provides lowest validation loss is chosen to be the optimal set of hyperparameters for each case. The validation loss is computed as the average relative 2-norm error over $\mathrm{X}_{\text{val}_2}$, using predictions initialized from points in $\mathrm{X}_{\text{val}_1}$ as initial conditions. We analyze four different scenarios for each test case with clean and noisy (30 dB SNR) data for two different N_{train} values. Instead of enforcing $\mathscr{L}_{\text{tc}}$ from the very first epoch of training, we enforce it after certain number of epochs e_s, which is a hyperparameter for training the tcKAE. That is, we keep $\gamma_{\text{tc}} = 0$ for the first e_s number of epochs. Up to e_s we train without enforcing the consistency regularization, and switch to nonzero γ_{tc} from e_s onward. The look-ahead step k_{tm} for enforcing temporal consistency is an important hyperparameter. Ideally, a larger k_{tm} should lead to better accuracy, since we are looking further into the future. However, as seen in (11), increasing k_{tm} also increases the number of terms in the summation for the temporal consistency loss. Since in practice we are using a numerical stochastic optimization routine, the increasing complexity (and possibly non-convexity) of the loss term increases the possibility of the optimization algorithm to get stuck in a local minima, thereby resulting in higher error and increased training time. As a result, in practice, we typically avoid selecting very large k_{tm} unless there is a significant improvement in accuracy. The weights for different loss components are one of the most crucial hyperparameters. Although, there are no fixed rule for selecting these weights, it is not entirely ad-hoc either. Since we are interested in forward prediction from the trained model, γ_{fwd} and γ_{id} are most crucial. In order to reduce the number of variables, we set $\gamma_{\text{id}} = 1$ and vary other weights accordingly. Note that the advantage of tcKAE over DAE and cKAE comes at the cost of extra hyperparameters and increased training time. Training time, detailed training parameters, ablation study, and effect of k_{tm} on prediction accuracy is provided in the supplementary material (see Supplementary Tables 1-12).

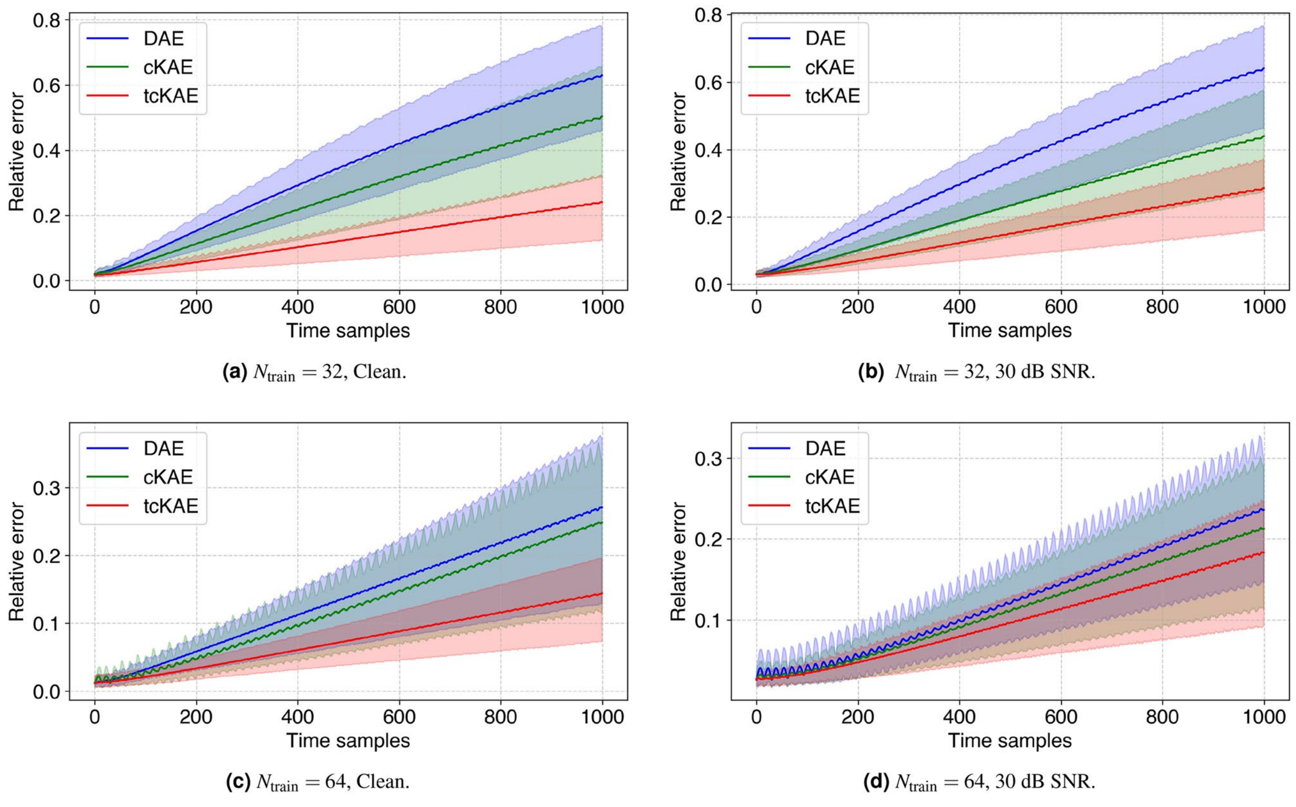
Results

Baseline

The performance of the proposed temporally consistent Koopman autoencoder (tcKAE) is compared against the state-of-the-art reduced-order KAE forecasting models for high-dimensional physical systems. As discussed earlier, several neural-network based models such as recurrent neural networks (RNNs)[15,22,38], Hamiltonian neural networks (HNNs)[23] have already been applied successfully in forecasting dynamical systems. However, these models do not fare well for very long-term predictions[29], and very high-dimensional systems. Koopman-based autoencoder models[7,28] have been shown to be effective in capturing this long-term behavior, especially for high-dimensional systems. However, recently proposed consistent Koopman autoencoder[6] which we will refer to as cKAE, has been shown to outperform all the mentioned methods for long-term predictions in terms of stability. We will consider cKAE as the baseline for benchmarking our proposed tcKAE for several datasets. We will also compare our results with dynamic autoencoders (DAEs), the term used in[6] to describe the models in[7,28].

Performance on benchmark datasets

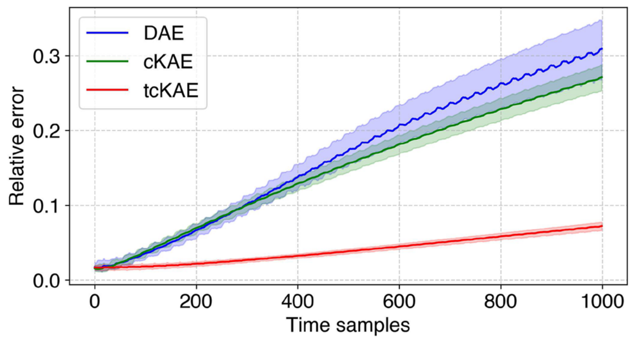
In order to streamline the discussion, we adopt the following notation: the complete dataset is denoted by $\mathrm{X} \in \mathbb{R}^{N_{\text{dim}} \times N_{\text{tot}}}$, where N_{dim} is the dimension of the state-space, and N_{tot} is the total number of time



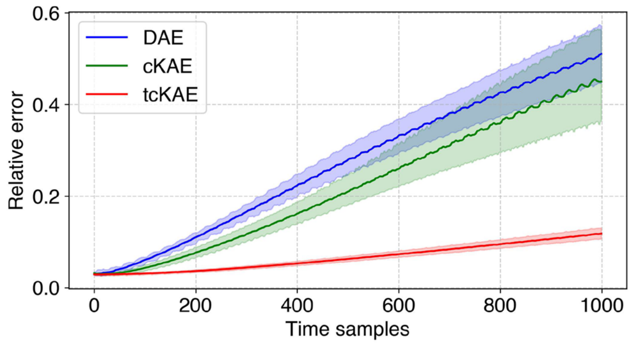
(a) $N_{\text{train}} = 32$, Clean. **(b)** $N_{\text{train}} = 32$, 30 dB SNR. **(c)** $N_{\text{train}} = 64$, Clean. **(d)** $N_{\text{train}} = 64$, 30 dB SNR.

Fig. 5. Comparison between DAE, cKAE, and tcKAE for long-term extrapolation performance of pendulum dynamics. The shaded portion denotes the 90% confidence interval.

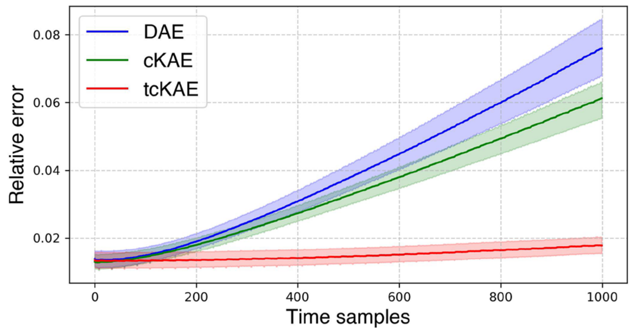
samples [1]. We denote the training set as $\mathrm{X}_{\mathrm{train}} \in \mathbb{R}^{N_{\mathrm{dim}} \times N_{\mathrm{train}}}$, validation set as $\mathrm{X}_{\mathrm{val}} \in \mathbb{R}^{N_{\mathrm{dim}} \times N_{\mathrm{val}}}$, and testing set as $\mathrm{X}_{\mathrm{test}} \in \mathbb{R}^{N_{\mathrm{dim}} \times N_{\mathrm{test}}}$ with N_{train}, N_{val} and N_{test} being the number of time samples in training, validation and testing set respectively. Note that validation for long-term time-series prediction is nontrivial, and authors in[6] does not mention any validation set. However, for practical applications, validation is crucial, and we validate using the following method: We divide the validation set in two sets $\mathrm{X}_{\mathrm{val}_1} \in \mathbb{R}^{N_{\mathrm{dim}} \times N_{\mathrm{val}_1}}$ and $\mathrm{X}_{\mathrm{val}_2} \in \mathbb{R}^{N_{\mathrm{dim}} \times N_{\mathrm{val}_2}}$, separated by $\mathrm{X}_{\mathrm{train}}$ as shown in Fig. 4, where $N_{\mathrm{val}} = N_{\mathrm{val}_1} + N_{\mathrm{val}_2}$. We chose $N_{\mathrm{val}_1} = N_{\mathrm{val}_2} = \frac{1}{4} N_{\mathrm{train}}$ for all the datasets. The datapoints in $\mathrm{X}_{\mathrm{val}_1}$ are used as initial conditions to predict over the span of $\mathrm{X}_{\mathrm{val}_2}$. The error metric we use in this work is the 2-norm relative error defined (at n^{th} time sample) by $\delta_n = \frac{||\hat{\mathrm{x}}_n - \mathrm{x}_n||_2}{||\mathrm{x}_n||_2}$, where x is the original state and $\hat{\mathrm{x}}$ is the predicted state. The validation loss is determined by calculating the average relative 2-norm error across $\mathrm{X}_{\mathrm{val}_2}$, where predictions are generated



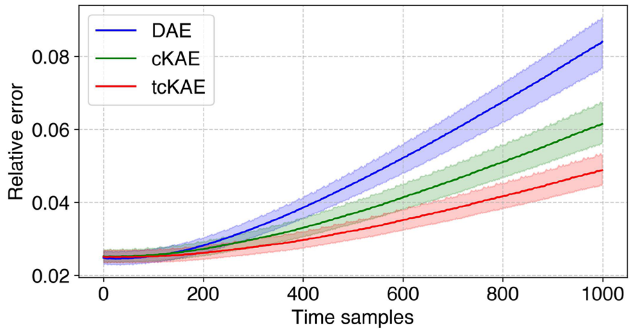
(a) $N_{\mathrm{train}} = 28$, Clean.



(b) $N_{\mathrm{train}} = 28$, 30 dB SNR.



(c) $N_{\mathrm{train}} = 52$, Clean.



(d) $N_{\mathrm{train}} = 52$, 30 dB SNR.

Fig. 6. Comparison between DAE, cKAE, and tcKAE for long-term extrapolation performance of oscillating electron beam. The shaded portion denotes the 90% confidence interval.

using initial conditions from points in $\mathrm{X}_{\mathrm{val}_1}$. The test error is calculated as the 2-norm relative error in the extrapolation region. To generate an error bar, we use the variance quantified by the 90% confidence interval, defined by the 5th and 95th percentiles of the error. This provides a robust summary of the typical error range by excluding the extreme 5% values at both ends of the distribution. We follow the statistical analysis approach used

[1]Note that we distinguish between time steps and time samples. By time step, we refer to the discrete time instances used by high-fidelity numerical solvers to generate data. This high-fidelity data is typically sampled at specific time steps and used for KAE training.

	$N_{\mathrm{train}} = 32$		$N_{\mathrm{train}} = 64$	
Method	**Clean**	**30 dB SNR**	**Clean**	**30 dB SNR**
DAE	34.411 (20.12)	35.055 (18.83)	13.884 (12.10)	12.421 (9.75)
cKAE	26.415 (18.52)	23.202 (15.71)	12.388 (11.82)	11.316 (9.75)
tcKAE	12.561 (10.20)	15.147 (10.92)	7.491 (6.36)	9.826 (7.91)

Table 1. Error (%) comparison for different N_{train} and noise levels for pendulum data. The quantities inside parenthesis show the width of 90% confidence interval.

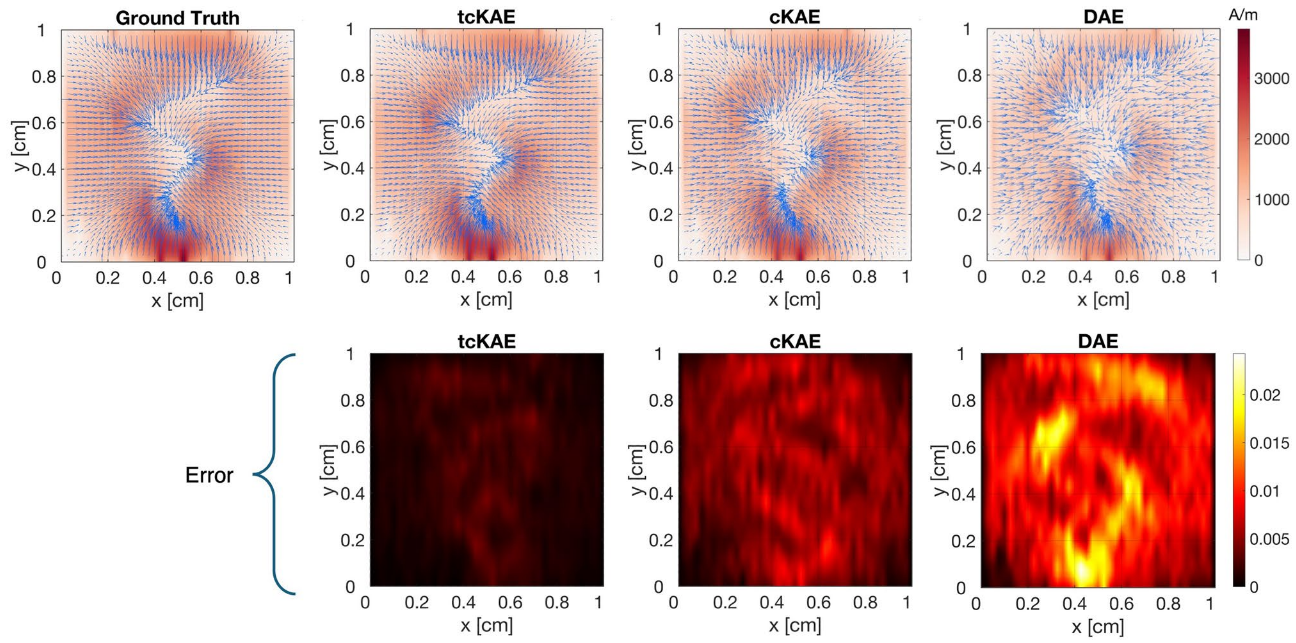


Fig. 7. Predictions (top) and corresponding error profile (bottom) for oscillating electron beam example ($N_{\text{train}} = 28$, SNR = 30 dB) at 1000th time sample in the test-region.

in[6]. Specifically, we take the first N_s time steps ($N_s = N_{\text{train}}$) from the testing set and use the corresponding states as initial conditions. The trained model is then used to predict up to a fixed number of time steps. For each initial condition, we compute the 2-norm error by comparing the predictions with the ground truth. These errors are then used to generate the confidence interval. The reported results represent the average of neural network training outcomes across 10 different random seeds.

Undamped pendulum
The undamped pendulum is a classic textbook example of nonlinear dynamical system for benchmarking performance of data-driven predictive models[6,23,39–41]. Although, the pendulum dynamics do not fall under typical PDE-governed high-dimensional systems, it serves as an excellent illustrative example due to the wide range of physical systems they represent, ranging from orbital mechanics, micorwave cavities to select biological and quantum systems. We use the elliptical functions[6,42] for exact solution of the motion governed by the ODE $\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0$, where $\theta \in [0, 2\pi]$ denotes the angular displacement from equilibrium $\theta = 0$ in radians. The gravitational constant and length of the pendulum is denoted by g and l respectively. We consider $g = 9.8$ m/s^2, and $l = 1$ m with initial condition $\theta = \theta_0$ and $\frac{d\theta}{dt} = \dot{\theta} = 0$. We sample the solution at interval of 0.1 s upto 220 s, resulting in the dataset $\Theta \in \mathbb{R}^{2\times 2200}$ with 2200 time samples. Note that the dimensionality of the state-space is two, since we have two states θ and $\dot{\theta}$. In order to mimic a high-dimensional system, we use a random orthogonal transformation[6] to rotate it to higher dimensional ($N_{\text{dim}} = 64$) space, i.e. $\mathrm{X} = \mathrm{P} \cdot \Theta$, where $\mathrm{P} \in \mathbb{R}^{64\times 2}$ is a random orthogonal matrix. The final dataset $\mathrm{X} \in \mathbb{R}^{64\times 2200}$ consists of 2200 time samples of a state with dimension 64. We generate the data using the initial condition $\theta_0 = 2.4$. One cycle of oscillation is covered approximately by 31 time steps (same as time samples). We perform the tests for $N_{\text{train}} = 32$ and 64, for clean and noisy (30 dB SNR) data. For $N_{\text{train}} = 32$, we take $N_{\text{val}_1} = N_{\text{val}_2} = 8$, and for $N_{\text{train}} = 64$, $N_{\text{val}_1} = N_{\text{val}_2} = 16$ is taken. We extrapolate 1000 future time samples using the trained model to evaluate the test error. We use $N_s = N_{\text{train}}$ initial conditions in test set to generate the 90% confidence interval. The results are summarized in Table 1.

As can be seen in Table 1, and Fig. 5, our proposed tcKAE has clear advantage over DAE or cKAE for clean data. This advantage is magnified for limited dataset of $N_{\text{train}} = 32$, which covers approximately one and half cycle of oscillation of the pendulum. The advantage tends to diminish with increasing size of training set. Large N_{train} not only makes learning with larger look-ahead step k_m possible, but also reduces overfitting, resulting in enhanced stability for long-term predictions for DAE and cKAE. Table 1 also shows that the variation of error (values inside parenthesis) is lowest for tcKAE. We see a similar trend when noise is included. However, interestingly with addition of noise in some cases the relative error actually decreases. This should not be surprising since models tend to overfit with limited data, and adding noise is one of the solutions to avoid such overfitting. Overall, tcKAE provides more accurate estimate of the state for long-term prediction.

Oscillating electron beam
Plasma systems are excellent candidates for data-driven modeling due to their high-diemnsionality, and inherent nonlinearity arising from the complex wave-particle interaction. A two dimensional electron beam (Fig. 3a) oscillating under the influence of an external transverse magnetic flux is simulated using charge-conserving

	$N_{\text{train}} = 28$		$N_{\text{train}} = 52$	
Method	**Clean**	**30 dB SNR**	**Clean**	**30 dB SNR**
DAE	16.570 (4.22)	27.006 (6.20)	3.956 (0.87)	4.782 (0.74)
cKAE	14.958 (2.15)	21.901 (8.28)	3.370 (0.61)	3.900 (0.64)
tcKAE	4.003 (0.73)	6.575 (1.23)	1.489 (0.46)	3.376 (0.52)

Table 2. Error (%) comparison for different N_{train} and noise levels for oscillating electron beam data. The quantities inside parenthesis show the width of 90% confidence interval.

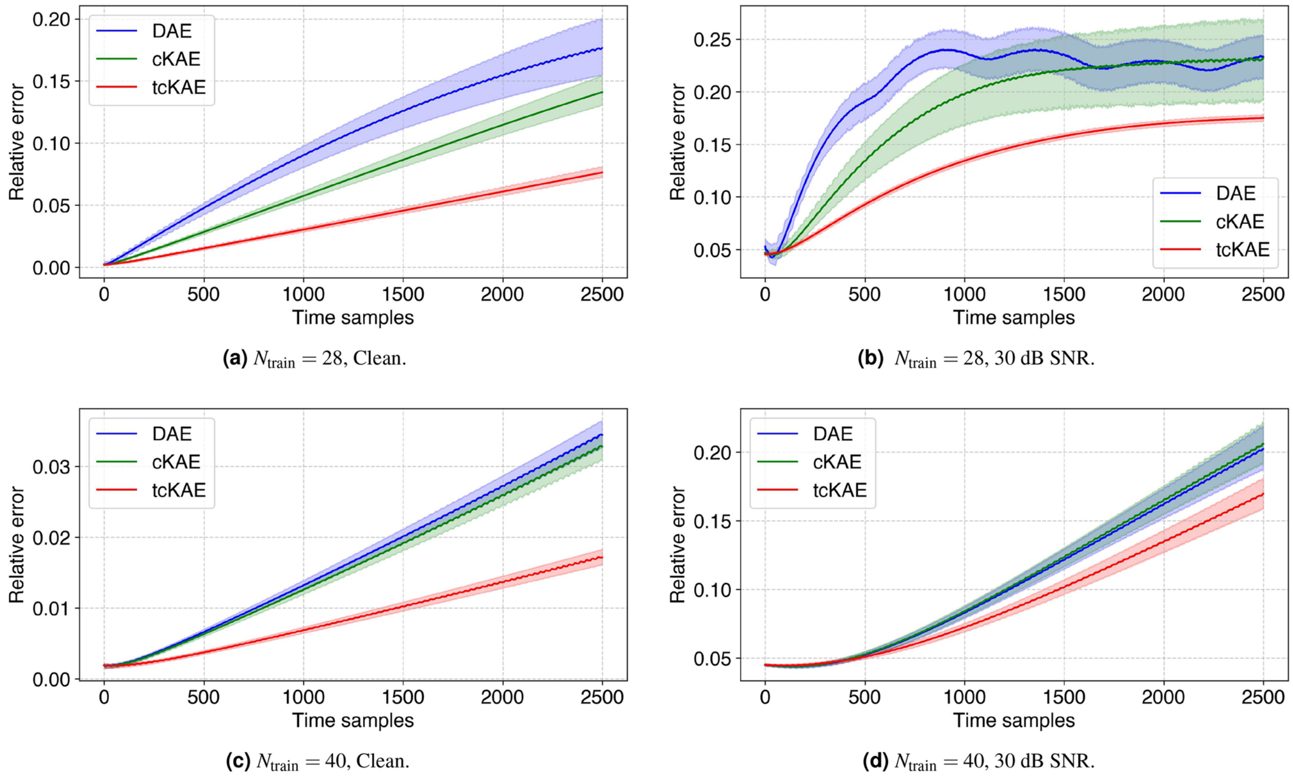


Fig. 8. Comparison between DAE, cKAE, and tcKAE for long-term extrapolation performance of flow past cylinder. The shaded portion denotes the 90% confidence interval.

electromagnetic particle-in-cell (EMPIC) algorithm[36]. The solution domain (1 m × 1 m) is discretized using unstructured triangular mesh with 844 nodes, 2447 edges, and 1604 elements (triangles). The electric field data ($\mathrm{e} \in \mathbb{R}^{2447}$) is sampled at every 0.32 ns. Starting from 80 ns, total 1751 time samples are collected with the complete dataset $\mathrm{X} \in \mathbb{R}^{2447\times1751}$. More details are provided in Methods: Simulation Setup section .

For the electron beam oscillation, one period approximately comprises of 25 time samples. In order to assess the performance of the Koopman models for limited dataset, we train the models for $N_{\text{train}} = 28$ and 52, each for clean and noisy (30 dB SNR) data. For $N_{\text{train}} = 28$; $N_{\text{val}_1} = N_{\text{val}_2} = 7$, and for $N_{\text{train}} = 52$; $N_{\text{val}_1} = N_{\text{val}_2} = 13$. We extrapolate up to 1000 time samples for evaluating the test error. Growth of the relative prediction error over 1000 time samples is depicted in Fig. 6. The snapshots of the electric field distribution and corresponding spatial error profiles for long-term prediction is shown in Fig. 7. Note that the spatial error is calculated as following: $\delta(x,y) = \frac{|\mathrm{x}(x,y)-\hat{\mathrm{x}}(x,y)|}{||\mathrm{x}||_2}$. As can be seen from Fig. 6 and Table 2, the advantage of tcKAE is prominent for limited dataset i.e. $N_{\text{train}} = 28$.

Flow past cylinder

Flow past cylinder is another common dynamical system used for benchmarking data-driven models. The simulation was carried out using MATLAB's FEAtool (Fig. 3b). A cylinder with diameter 0.1 m is located at a height of 0.2 m. The fluid is characterized by its density $\rho = 1$ Kg/m^3, and dynamic viscosity $\mu = 0.001$ Kg/m s. The flow is unsteady with a maximum velocity of 1 m/s and mean velocity $\frac{2}{3}$ of the maximum velocity. The simulation is run for 80 s until the steady state is achieved, and the horizontal u component of the velocity is probed at every 0.02 s starting from 20 s. The complete dataset $\mathrm{X} \in \mathbb{R}^{2647\times3001}$ consists of 3001 time samples. More details regarding the simulation setup is presented in Methods: Simulation Setup section. Growth of

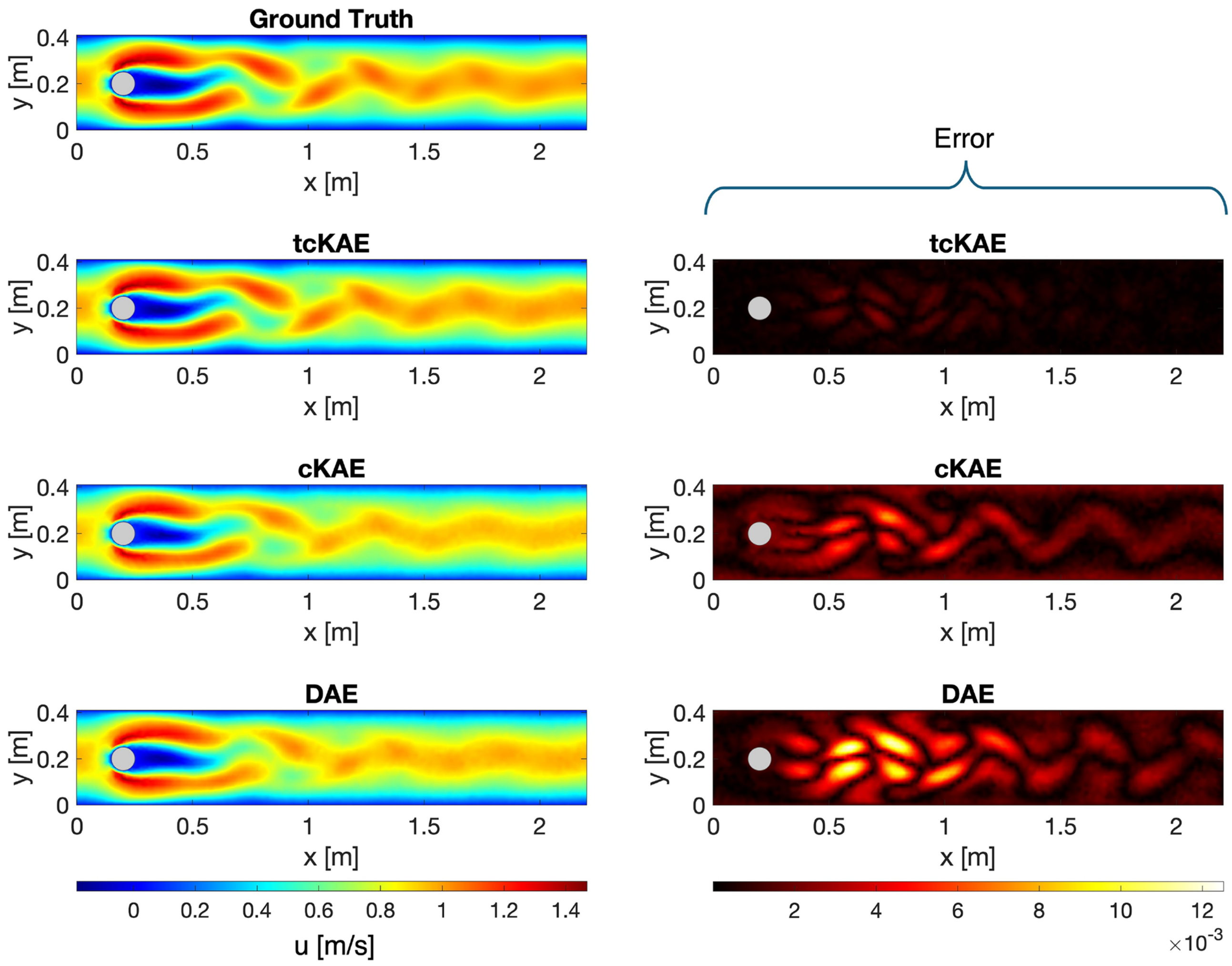


Fig. 9. Predictions (left) and corresponding error profile (right) for flow past cylinder example ($N_{\text{train}} = 28$, clean) at 2500th time sample in the test-region. The grey circle represents the cylinder.

	$N_{\text{train}} = 28$		$N_{\text{train}} = 40$	
Method	**Clean**	**30 dB SNR**	**Clean**	**30 dB SNR**
DAE	10.182 (2.24)	20.724 (3.60)	1.695 (0.18)	10.775 (1.29)
cKAE	7.153 (1.06)	18.376 (5.58)	1.616 (0.17)	10.933 (1.24)
tcKAE	3.808 (0.45)	13.311 (0.53)	0.874 (0.11)	9.275 (0.94)

Table 3. Error (%) comparison for different N_{train} and noise levels for flow past cylinder data. The quantities inside parenthesis show the width of 90% confidence interval.

relative error over 2500 time samples is shown in Fig. 8. Predicted velocity field snapshots and corresponding error profile for long-term prediction is shown in Fig. 9. Here also we observe that tcKAE provides significant advantage for limited training data.

In case of flow past cylinder, since one cycle approximately contains 29 time samples, we consider $N_{\text{train}} = 28$ and 40. For $N_{\text{train}} = 28$, we take $N_{\text{val}_1} = N_{\text{val}_2} = 7$, and for $N_{\text{train}} = 40$, $N_{\text{val}_1} = N_{\text{val}_2} = 10$ is taken. We evaluate the test error for 2500 time samples in the extrapolation region. Growth of relative error over 2500 time samples is shown in Fig. 8. Predicted velocity field snapshots and corresponding error profile for long-term prediction is shown in Fig. 9. Table 3 provides the relative (percentage) time-averaged test error. Here also we observe that tcKAE provides significant advantage for limited training data.

Discussion

The proposed tcKAE algorithm advances the state-of-the-art by enforcing multi-step temporal consistency in the latent space. Unlike prior KAE architectures such as consistent Koopman autoencoders (cKAE)[6], which rely on backward dynamics assumptions, tcKAE sidesteps this limitation by introducing a consistency regularization term in the forward direction. The ablation study in the supplementary material further demonstrates that, even without the backward and consistency losses, the combination of forward, identity, and temporal consistency losses achieves comparable or even superior accuracy to cKAE. This innovation strengthens the Koopman invariance of the latent space, enhancing robustness to limited date and noise, improving generalizability. Furthermore, empirical results demonstrate that tcKAE outperforms both cKAE and dynamic autoencoders (DAEs)[7] in terms of long-term prediction accuracy and stability, particularly for scenarios where we have limited training data. The proposed tcKAE not only achieved lower error margins in benchmark problems such as pendulum oscillation, oscillating electron beam, and flow past a cylinder, but also demonstrated narrower confidence intervals, thereby inspiring greater confidence in the predictions. It is important to note that the proposed tcKAE does not guarantee unconditional stability. The tcKAE predictions are *more* stable for long-term predictions compared to DAE and cKAE. However, the secular growth in relative error indicates that if we were to predict 10 or 20 times further, the large error would likely render the results useless for all the models.

Despite its advantages, tcKAE has some limitations. With sufficient training data, the performance gap between tcKAE and vanilla KAEs narrows significantly, reducing the need for additional consistency constraints. Furthermore, the inclusion of temporal regularization introduces computational overhead, increasing training time compared to simpler DAE method. These trade-offs highlight the necessity of tcKAE primarily in scenarios with scarce or noisy data, where its robustness and enhanced prediction accuracy provide a critical edge. This makes tcKAE particularly attractive for modeling high-dimensional nonlinear systems, where computationally costly numerical solvers limit the amount of training data available. Note that although, the proposed idea of temporal consistency is explored in the context of KAEs, it can be easily extended to other models. Future work may also explore control applications with tcKAE and hybrid models that balance the computational cost of tcKAE with the efficiency of traditional approaches, potentially by dynamically adapting regularization strength based on data quality and training progress.

Data availability

The code for generating the data and results of this paper will be made available upon request. Any such request should be addressed to A.C. It is also partially available at the following dynamically updated github repository: https://github.com/SOARLabOSU/tcKAE.

Received: 31 December 2024; Accepted: 2 June 2025
Published online: 01 July 2025

References

1. Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P. & Henningson, D. S. Spectral analysis of nonlinear flows. *J. Fluid Mech.* **641**, 115–127 (2009).
2. Koopman, B. O. Hamiltonian systems and transformation in Hilbert space. *Proceed. Nat. Acad. Sci.* **17**, 315–318 (1931).
3. Mauroy, A. & Mezić, I. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Trans. Autom. Control* **61**, 3356–3369 (2016).
4. Peitz, S., Otto, S. E. & Rowley, C. W. Data-driven model predictive control using interpolated Koopman generators. *SIAM J. Appl. Dyn. Syst.* **19**, 2162–2193 (2020).
5. Goswami, D. & Paley, D. A. Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A koopman spectral approach. *IEEE Trans. Autom. Control* **67**, 2715–2728. https://doi.org/10.1109/TAC.2021.3088802 (2022).
6. Azencot, O., Erichson, N. B., Lin, V. & Mahoney, M. Forecasting sequential data using consistent koopman autoencoders. in *International Conference on Machine Learning*, pp. 475–485 (PMLR, 2020).
7. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **9**, 4950 (2018).
8. Takeishi, N., Kawahara, Y. & Yairi, T. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in neural information processing systems* **30** (2017).
9. Sajjadi, M., Javanmardi, M. & Tasdizen, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 29 (Curran Associates, Inc., 2016).
10. Englesson, E. & Azizpour, H. Consistency regularization can improve robustness to label noise. arXiv:2110.01242 (2021).
11. Fan, Y., Kukleva, A., Dai, D. & Schiele, B. Revisiting consistency regularization for semi-supervised learning. *Int. J. Comput. Vision* **131**, 626–643. https://doi.org/10.1007/s11263-022-01723-4 (2023).
12. Bailer-Jones, C. A. L., MacKay, D. J. C. & Withers, P. J. A recurrent neural network for modelling dynamical systems. *Netw. Comput. Neural Syst.* **9**, 531–547. https://doi.org/10.1088/0954-898X_9_4_008 (1998).
13. Aussem, A. Dynamical recurrent neural networks towards prediction and modeling of dynamical systems. *Neurocomputing* **28**, 207–232 (1999).
14. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
15. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555 (2014).
16. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks* **5**, 157–16. https://doi.org/10.1109/72.279181 (1994).
17. Miller, J. & Hardt, M. Stable recurrent models. in *International Conference on Learning Representations* (2019).
18. Arjovsky, M., Shah, A. & Bengio, Y. Unitary evolution recurrent neural networks. in Balcan, M. F. & Weinberger, K. Q. (eds.) *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48 of *Proceedings of Machine Learning Research*, 1120–1128 (PMLR, New York, New York, USA, 2016).
19. Kerg, G. *et al.* Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. in Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 2019).
20. Jia, X. *et al.Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles*, pp. 558–566.
21. Sussillo, D. & Barak, O. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Comput.* **25**, 626–649 (2013).

22. Chang, B., Chen, M., Haber, E. & Chi, E. H. Antisymmetricrnn: A dynamical system view on recurrent neural networks. arXiv:1902.09689 (2019).
23. Greydanus, S., Dzamba, M. & Yosinski, J. Hamiltonian neural networks. *Advances in neural information processing systems* **32** (2019).
24. Schmid, P. J. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **656**, 5–28 (2010).
25. Williams, M. O., Kevrekidis, I. G. & Rowley, C. W. A data-driven approximation of the Koopman operator: extending Dynamic Mode Decomposition. *J. Nonlinear Sci.* **25**, 1307–1346 (2015).
26. Williams, M. O., Hemati, M. S., Dawson, S. T. M., Kevrekidis, I. G. & Rowley, C. W. Extending data-driven Koopman analysis to actuated systems. *IFAC-PapersOnLine* **49**, 704–709 (2016).
27. Goswami, D. & Paley, D. A. Global bilinearization and controllability of control-affine nonlinear systems: A Koopman spectral approach. in *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 6107–6112 (2017).
28. Otto, S. E. & Rowley, C. W. Linearly recurrent autoencoder networks for learning dynamics. *SIAM J. Appl. Dyn. Syst.* **18**, 558–593 (2019).
29. Lange, H., Brunton, S. L. & Kutz, J. N. From fourier to koopman: Spectral methods for long-term time series prediction. *J. Mach. Learn. Res.* **22**, 1881–1918 (2021).
30. Nayak, I., Teixeira, F. L. & Kumar, M. Koopman autoencoder architecture for current density modeling in kinetic plasma simulations. in *2021 International Applied Computational Electromagnetics Society Symposium (ACES)*, 1–3 (IEEE, 2021).
31. Nayak, I., Kumar, M. & Teixeira, F. L. Koopman autoencoders for reduced-order modeling of kinetic plasmas. *Advances in Electromagnetics Empowered by Artificial Intelligence and Deep Learning* 515–542 (2023).
32. Narayanan, S. S. K. S., Tellez-Castro, D., Sutavani, S. & Vaidya, U. Se(3) koopman-mpc: Data-driven learning and control of quadrotor uavs (2023). arXiv:2305.03868.
33. Avila, A. M. & Mezić, I. Data-driven analysis and forecasting of highway traffic dynamics. *Nat. Commun.* **11**, 2090. https://doi.org/10.1038/s41467-020-15582-5 (2020).
34. Dogra, A. S. & Redman, W. Optimizing neural networks via koopman operator theory. in Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 2087–2097 (Curran Associates, Inc., 2020).
35. Yeung, E., Kundu, S. & Hodas, N. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. in *2019 American Control Conference (ACC)*, 4832–4839, https://doi.org/10.23919/ACC.2019.8815339 (2019).
36. Na, D., Moon, H., Omelchenko, Y. A. & Teixeira, F. L. Local, explicit, and charge-conserving electromagnetic particle-in-cell algorithm on unstructured grids. *IEEE Trans. Plasma Sci.* **44**, 1353–1362 (2016).
37. Ohio supercomputer center (osc). https://osc.edu
38. Kerg, G. *et al.* Non-normal recurrent neural network (nnrnn): Learning long time dependencies while improving expressivity with transient dynamics. *Advances in neural information processing systems* **32** (2019).
39. Chen, Z., Zhang, J., Arjovsky, M. & Bottou, L. Symplectic recurrent neural networks. in *International Conference on Learning Representations* (2020).
40. Bounou, O., Ponce, J. & Carpentier, J. Online learning and control of dynamical systems from sensory input. in *NeurIPS 2021-Thirty-fifth Conference on Neural Information Processing Systems Year* (2021).
41. Azari, B. & Erdogmus, D. Equivariant deep dynamical model for motion prediction. in *International Conference on Artificial Intelligence and Statistics*, 11655–11668 (PMLR, 2022).
42. Azencot (2020) code. https://github.com/erichson/koopmanAE.

Acknowledgements
This work was funded in part by the Department of Energy Grant DE-SC0022982 through the NSF/DOE Partnership in Basic Plasma Science and Engineering and in part by the Ohio Supercomputer Center Grants PAS-0061 and PAS-2709.

Author contributions
I.N. and D.G. conceived the algorithm. I.N. developed the code. A.C. conducted the experiments. I.N. and A.C. analyzed the results. F.T., M.K., and D.G. advised I.N and A.C. and acquired the funding and resources. All authors reviewed the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-05222-7.

Correspondence and requests for materials should be addressed to D.G.