Ground-Displacement Forecasting from Satellite Image Time Series via a Koopman-Prior Autoencoder

Takayuki Shinohara and Hidetaka Saomoto
National Institute of Advanced Industrial Science and Technology
Tsukuba, Japan
shinohara.takayuki@aist.go.jp, h-saomoto@aist.go.jp

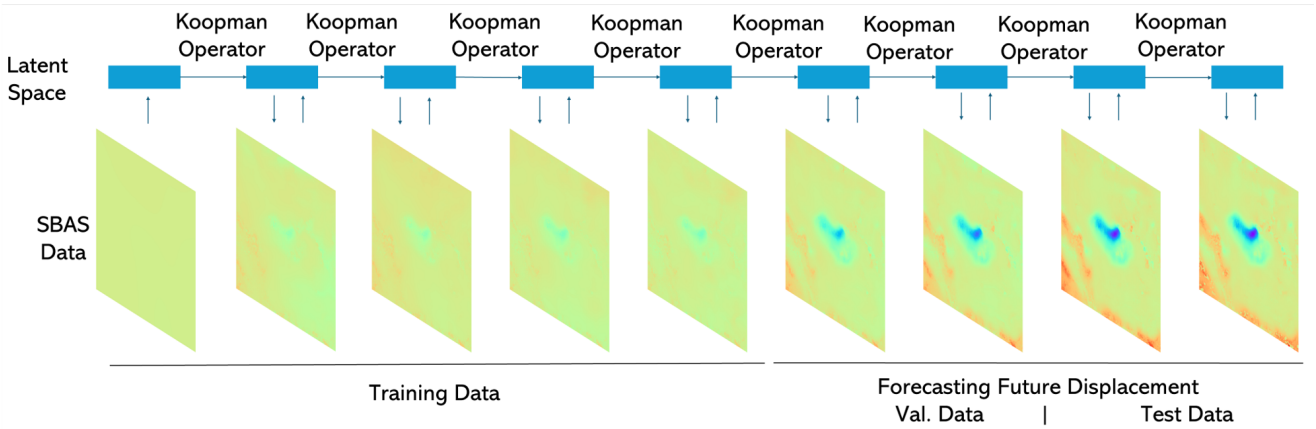


Figure 1. Overview of our method and visualization of one of the time series displacement satellite image calculation by Small Baseline Subset (SBAS) algorithm in our dataset.

Abstract

Accurate yet lightweight forecasting of ground-displacement image sequences is key for real-time disaster mitigation and urban-planning workflows. We introduce the Koopman-Prior Autoencoder (KPA), an efficient deep-learning framework that embeds a physics-inspired prior—a Koopman operator acting along the time axis—directly into the model's latent dynamics. A convolutional encoder first distills each displacement image frame into a compact representation. Rather than learning arbitrary nonlinear recurrence, we constrain temporal evolution to follow a single linear operator whose spectrum is regularised to ensure stability. This Koopman prior captures the dominant, quasi-linear components of crustal deformation, enabling long-horizon predictions with orders-of-magnitude fewer parameters and FLOPs than Transformer or diffusion baselines. Trained on nationwide Japanese archived displacement images and evaluated on geographically distinct test sites (Turkey, Italy, Hawaii), KPA matches or exceeds state-of-the-art accuracy while slashing inference cost, demonstrating that a carefully chosen physical prior can unlock scalable ground-motion forecasting on modest hardware.

1. Introduction

Ground-surface deformation (subsidence, uplift, and fault slip) poses direct risks to critical infrastructure and population centres. Small Baseline Subset (SBAS) Satellite Image Time Series [7] delivers millimetre-level displacement maps at continental scale, yet remains strictly observational. For proactive hazard mitigation and urban-planning workflows we must forecast future deformation fields, a task that requires capturing highly nonlinear spatio-temporal dynamics while remaining computationally affordable across decades of imagery.

Recent advancements in deep learning, including powerful sequence models like Transformers and generative approaches such as diffusion models, have demonstrated success in various forecasting tasks. However, their direct application to large-scale satellite image forecasting, particularly for SBAS data, often faces significant hurdles. Transformer-based models typically exhibit quadratic complexity with respect to sequence length due to their self-attention mechanisms, limiting scalability. Diffusion models, while capable generators, can be computationally expensive to train and sample from, and may not inherently enforce the underlying physical consistency desirable for geophysical phenomena.

Consequently, there is a pressing need for lightweight, yet accurate and physically-informed algorithms capable of efficiently processing SBAS data for robust predictive modeling.

To bridge this gap, we turn to Koopman operator theory [23], which offers a powerful framework for representing complex nonlinear dynamical systems using infinite-dimensional linear operators. For practical implementation, the central challenge lies in finding finite-dimensional approximations of this operator that capture the essential dynamics. A promising machine learning approach involves learning a mapping, typically via an autoencoder network, from the high-dimensional input space to a lower-dimensional latent space where the dynamics are approximately linear and governed by a finite-dimensional Koopman operator [25, 32]. This approach aligns with the principles of Physics-Constrained Learning (PCL), aiming to embed system properties into the model architecture to enhance interpretability and predictive accuracy with efficient parameterization.

In this paper, we propose a novel Koopman-Prior Autoencoder (KPA) framework specifically tailored for forecasting ground displacement from SBAS data(Figure 1). Our key contribution lies in designing an autoencoder architecture and associated learning objectives that effectively identify a Koopman-invariant latent subspace suitable for robust long-term prediction of spatiotemporal satellite data. By explicitly optimizing for linear dynamics within this learned latent space, KPA combines the representational power of deep learning with the structure and potential stability guarantees offered by Koopman theory [15, 26, 33]. This results in a lightweight and interpretable forecasting model that overcomes the computational limitations of larger architectures, offering a practical solution for large-scale displacement prediction tasks critical for risk management and sustainable development.

2. Related Work

2.1. Koopman theory

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

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

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

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

where $\mathcal{K}$ 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 $(\mathbf{K}\phi)(\cdot) = \lambda\phi(\cdot)$, with an eigenvalue λ. Considering the Koopman eigenfunctions span the Koopman observables, a vector valued observable $\mathbf{g} \in \mathcal{F}^p = [\psi_1 \ \psi_2 \ \dots \ \psi_p]^{\mathrm{T}}$ can be expressed as a sum of Koopman eigenfunctions $\mathbf{g}(\cdot) = \sum_{i=1}^{\infty} \phi_i(\cdot)\mathbf{v}_i^{\mathbf{g}}$, where $\mathbf{v}_i^{\mathbf{g}} \in \mathbb{R}^p, i = 1, 2, \dots,$ are called the *Koopman modes* of the observable $\mathbf{g}(\cdot)$. This modal decomposition provides the growth/decay rate $|\lambda_i|$ and frequency $\angle\lambda_i$ of different Koopman modes via its time evolution

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

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 [34], plasma dynamics [31], control systems [33], unmanned aircraft systems [30], and traffic prediction [3]. In addition, it is also being used for machine learning tasks and training deep neural networks [12]. Several methods have also been developed to compute the Koopman modal decomposition, e.g., DMD and EDMD [35, 42], Ulam-Galerkin methods, and deep neural networks [32, 44]. In this paper, we primarily focus on long-term prediction of autonomous dynamical systems using Koopman modes with autoencoder networks.

2.2. Timeseies Prediction

Neural network-based methods: Forecasting dynamical systems using neural networks, especially recurrent neural networks (RNNs), has been a topic of interest for decades [2, 5]. Advances in data availability, machine learning techniques, and computational power have significantly renewed focus on this area. Novel architectures like Long Short-Term Memory (LSTM) networks [19] and Gated Recurrent Units (GRUs) [11] have greatly improved the predictive capabilities of neural networks. However, training RNNs remains challenging due to issues like vanishing and exploding gradients [6], which can be mitigated through strategies such as stability analysis [27] or unitary hidden weight matrices [1]. Nevertheless, these approaches often trade off the expressive power of RNNs, potentially hindering short-term modeling [22]. Another limitation of RNNs lies in their lack of generalizability and interpretability when applied to physical systems. To address this, various methods have been developed to incorporate physical constraints, including physics-guided architectures [20], connections to dynamical systems [38], and formulations based on differential equations [9]. Hamiltonian neural networks [16] aim to learn conserved quantities like the Hamiltonian but are limited to lossless systems. In this work, we propose a linear

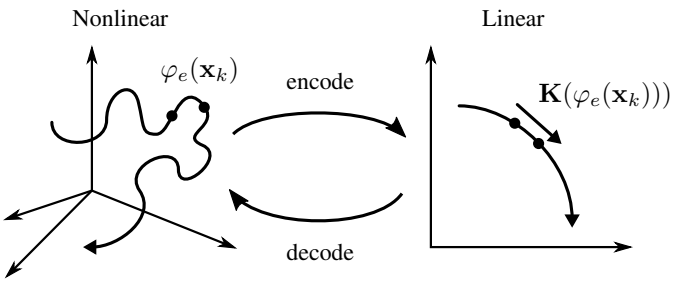


Figure 2. Illustration of the data transformation that maps the high-dimensional states $\mathbf{x}_k$ which evolve on a nonlinear trajectory via φ_e to a new space where the dynamics are linear and given by $\mathbf{K}$.

operator-based recurrent architecture augmented with physical prediction constraints to enhance generalizability and accuracy in forecasting the displacement of SBAS data as dynamical systems.

Koopman-based methods: Exploiting the linear nature of the Koopman operator in an infinite-dimensional function space [34], Koopman-based methods offer a linear framework for recurrent predictions. Early approaches relying on dynamic mode decomposition (DMD) [35] have been extended to include dictionary-based techniques [14, 15, 33, 42, 43], though these approaches often assume a dictionary that spans a Koopman-invariant subspace. Neural network-based Koopman autoencoders overcome this limitation by learning invariant spaces where dynamics can be approximated linearly [25, 32, 40]. The advantage of the Koopman spectral method has been demonstrated over recurrent networks like LSTM, GRU, and ESNs, emphasizing their ability to identify slowly evolving frequencies crucial for accurate long-term predictions [24]. Furthermore, autoencoders are particularly advantageous for modeling and reducing the dimensionality of high-dimensional systems, which is central to this work. Despite these benefits, existing KAE models often prioritize minimizing multi-step prediction errors and fail to address prediction consistency. Recent work [4] introduces the concept of the past state Koopman operator, which enforces future state and past state prediction consistency. However, this technique is effective only in cases where past state dynamics can be clearly defined.

3. Method

3.1. Problem Statement

Satellite SBAS time series encode millimetre–scale ground displacement. Let $\mathbf{x}_i \in \mathcal{M} \subset \mathbb{R}^{H\times W}$ denote the i-th frame. We model the sequence as a discrete, time-invariant dynamical system

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i), \tag{4}$$

where the unknown map f is highly nonlinear. A naive computer-vision approach would train a deep network to approximate f directly.

Koopman theory, however, guarantees that every nonlinear system admits a change of coordinates under which the evolution becomes *linear*. As shown in Figure **??** if we can learn an encoder φ_e such that

$$\mathbf{z}_{k+1} = \mathbf{K}\,\mathbf{z}_k, \qquad \mathbf{z}_k = \varphi_e(\mathbf{x}_k), \tag{5}$$

then forecasting l steps ahead reduces to $\mathbf{z}_{k+l} = \mathbf{K}^l\mathbf{z}_k$ followed by a decoder φ_d that reconstructs $\mathbf{x}_{k+l} \approx \varphi_d(\mathbf{z}_{k+l})$. Our goal is therefore to learn the encoder–decoder pair (φ_e, φ_d) and a compact transition matrix $\mathbf{K} \in \mathbb{R}^{N_l\times N_l}$.

3.2. Koopman-Prior Autoencoder (KPA)

KPA couples a lightweight autoencoder with an explicit Koopman layer:

1. **Encoder.** A shallow CNN φ_e maps each frame to a latent vector $\mathbf{z}_n \in \mathbb{R}^{N_l}$.
2. **Koopman Layer.** Latent dynamics are *forced to be linear*: $\mathbf{z}_{n+k} = \mathbf{K}\mathbf{z}_n$, where $\mathbf{K}$ is a learnable $N_l \times N_l$ matrix.
3. **Decoder.** A symmetric CNN–upsampling decoder φ_d reconstructs the forecast $\hat{\mathbf{x}}_{n+k}$ from $\mathbf{z}_{n+k}$.

To embed *physical reversibility* directly into the latent dynamics, we extend the model with a *backward* evolution operator so that it can predict the past as well as the future. Both $\mathbf{K}$ and its backward counterpart $\mathbf{K}_b \in \mathbb{R}^{N_l\times N_l}$ are trainable. The latter generates past states via $\mathbf{z}_{n-k} = \mathbf{K}_b\mathbf{z}_n$ and is regularised towards $\mathbf{K}^{-1}$. If $\mathbf{K}$ is constrained to be orthogonal, we simply set $\mathbf{K}_b = \mathbf{K}^\mathsf{T}$, incurring no extra parameters while preserving numerical stability. The resulting bidirectional pair enforces exact round-trip consistency, embeds the near-reversible nature of crustal deformation, and enables a unified spectral analysis of forward and backward error.

To train KPA, we jointly minimise *reconstruction* error on the input frames and *prediction* error on both future and past frames, while also penalising discrepancies in latent space to promote numerical stability. Concretely, we minimise four ℓ_2–norm losses:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_{\text{in}}}\sum_{n=1}^{N_{\text{in}}}\left\|\hat{\mathbf{x}}_n - \mathbf{x}_n\right\|_2^2, \tag{6}$$

$$\mathcal{L}_{\text{fwd}} = \frac{1}{k_m N_{\text{in}}}\sum_{k=1}^{k_m}\sum_{n=1}^{N_{\text{in}}}\left\|\hat{\mathbf{x}}_{n+k} - \mathbf{x}_{n+k}\right\|_2^2, \tag{7}$$

$$\mathcal{L}_{\text{bwd}} = \frac{1}{k_m N_{\text{in}}}\sum_{k=1}^{k_m}\sum_{n=1}^{N_{\text{in}}}\left\|\hat{\mathbf{x}}_{n-k} - \mathbf{x}_{n-k}\right\|_2^2, \tag{8}$$

$$\mathcal{L}_{\text{lat}} = \frac{1}{N_{\text{in}}}\sum_{n=1}^{N_{\text{in}}}\left\|\hat{\mathbf{z}}_{n\pm k} - \mathbf{z}_{n\pm k}\right\|_2^2, \tag{9}$$

where k_m is the maximum prediction horizon. The total objective combines four complementary errors, each scaled by a weight $\gamma_\star$. The reconstruction term $\mathcal{L}_{\text{rec}}$ compels the autoencoder to reproduce every input frame, preserving all pixel-level detail in the latent code. The forward term $\mathcal{L}_{\text{fwd}}$ penalises mismatches between predicted and true frames up to k_m steps ahead, teaching the Koopman matrix $\mathbf{K}$ to advance latent dynamics reliably along the time axis. To inject reversibility, the backward term $\mathcal{L}_{\text{bwd}}$ imposes the same penalty for k_m steps into the past via the operator $\mathbf{K}_b$, thereby regularising $\mathbf{K}_b$ toward $\mathbf{K}^{-1}$ and stabilising long roll-outs. Finally, the latent-consistency term $\mathcal{L}_{\text{lat}}$ aligns trajectories predicted by $\mathbf{K}$ (or $\mathbf{K}_b$) with the encoder's latent codes, tightening Koopman invariance and further improving numerical stability. The total loss is

$$\mathcal{L} = \gamma_{\text{rec}}\mathcal{L}_{\text{rec}} + \gamma_{\text{fwd}}\mathcal{L}_{\text{fwd}} + \gamma_{\text{bwd}}\mathcal{L}_{\text{bwd}} + \gamma_{\text{lat}}\mathcal{L}_{\text{lat}}, \qquad (10)$$

whitch are tuned to balance pixel-level fidelity, forecasting accuracy, temporal reversibility, and latent coherence..

3.3. Network Architecture

While the fundamental training methodology has been outlined in the preceding sections, this subsection provides a detailed examination of our deep learning architecture and the specific hyperparameter configurations employed for displacement imagery prediction. We focus on the technical implementation details that enable our model to effectively learn temporal patterns in displacement data.

The architecture of our displacement imagery prediction network is illustrated in Figure 3, building upon the framework presented in [4]. The network processes high-dimensional displacement data through input($\mathbf{x_n} \in \mathbb{R}^{\text{H}\times\text{W}}$) and predict $\mathbf{x_{n+1}} \in \mathbb{R}^{\text{H}\times\text{W}}$ (shown in blue rectangle). In this case, our network trains the first N_{in} frames and predicts N_{out} frames respectively.

Encoder. The transformation between input spaces and latent space is facilitated by the encoder (green triangle in Figure 3) and decoder (magenta triangle in Figure 3)), each containing a simple CNN-based network.

First, we show the encoder network. We adopt a three–stage *FasterNet* backbone, which alternates depth-wise–separable 3×3 convolutions with point-wise linear projections and LayerNorm, following the design of original FasterNet[10]. Each stage halves the spatial resolution ($H\times W \rightarrow \frac{H}{2}\times\frac{W}{2}$) so that, after the third down-sampling, the feature map is $\frac{H}{8}\times\frac{W}{8}$ with C channels. A global average pooling and a linear layer project this tensor to the N_l-dimensional Koopman latent vector $\mathbf{z}_n = \varphi_e(\mathbf{x}_n)$. Except for the bottleneck, all layers use `tanh` activations, yielding a parameter-efficient yet expressive encoder.

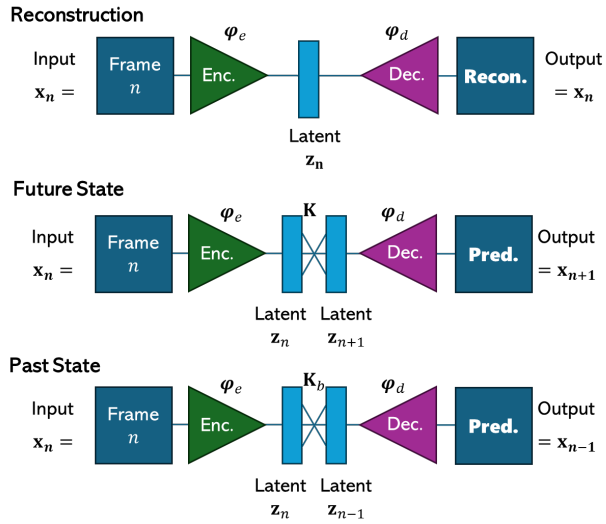


Figure 3. Overview of the proposed KPA architecture. A single, weight–shared **encoder** first maps each SBAS frame to a latent vector. The latent code is then routed through three heads: (*i*) a **reconstruction** head that decodes the same frame, (*ii*) a **future-state** head that applies the forward Koopman matrix $\mathbf{K}$ up to k_m steps and decodes the resulting vectors, and (*iii*) a **past-state** head that applies the backward matrix $\mathbf{K}_b$ and decodes past frames. All three heads share an identical **decoder**, ensuring that only the latent dynamics—not the image generator—differ across tasks. Dashed arrows indicate the linear propagation performed by $\mathbf{K}$ or $\mathbf{K}_b$; solid arrows correspond to convolutional operations. The entire graph is trained end-to-end with the loss terms described in Sec. 3.2, enforcing pixel fidelity, forward prediction, backward consistency, and latent invariance.

Decoder. The decoder(φ_d) inverses the encoder process. The latent code $\mathbf{z}_n \in \mathbb{R}^{N_l}$ or $\mathbf{z}_{n\pm k} \in \mathbb{R}^{N_l}$ is first reshaped to a square tensor of size $\frac{H}{8}\times\frac{W}{8}\times C$. We then apply three identical *up-sampling blocks*: nearest-neighbour interpolation by a factor of 2 followed by a 3×3 depthwise convolution, a point-wise convolution, and LayerNorm. After the final block the resolution is restored to $H\times W$; a 1×1 convolution produces the predicted displacement map $\hat{\mathbf{x}}_{n\pm k}$. All decoder activations are `tanh`, ensuring that the reconstructed range matches the input dynamic range.

Koopman Layer. The latent bottleneck vector (cyan) holds the latent state $\mathbf{z}_n \in \mathbb{R}^{N_l}$; its dimension N_l is chosen as a multiple of 16 to balance expressiveness and efficiency. Temporal evolution is computed *purely linearly* by the learnable Koopman matrix $\mathbf{K} \in \mathbb{R}^{N_l\times N_l}$:

$$\mathbf{z}_{n+1} = \mathbf{K}\,\mathbf{z}_n, \qquad \mathbf{z}_{n+k} = \mathbf{K}^k\,\mathbf{z}_n \;\; (k \geq 1). \qquad (11)$$

Thus a single learnable matrix–vector multiply advances the latent code by one time step, while repeated application of $\mathbf{K}$

yields multi-step predictions with negligible computational overhead.

3.4. Baseline Methods

To evaluate the performance of KPA, we compare it against five baseline models. The first baseline employs Dynamic Mode Decomposition (DMD) combined with an AutoEncoder to capture the underlying temporal structures. DMD [36, 41] has been widely used for analyzing spatiotemporal patterns. The second utilizes a State Space Model, a state-space model architecture, in combination with an AutoEncoder for sequence prediction. In this paper, we use Mamba [17] as an SSM method. Mamba, a recently proposed state-space model, enables efficient sequence modeling with linear-time complexity. The third and fourth baselines directly apply DMD and SSM, respectively, without an AutoEncoder to assess their standalone forecasting capabilities. Dynamic Mode Decomposition (DMD) is used for time-series image prediction by representing image sequences as linear dynamical systems. Each image is reshaped into a 1D vector before applying DMD, enabling the extraction of dominant spatiotemporal patterns. The trained model then extrapolates future image frames based on learned modes and dynamics. Finally, the fifth baseline leverages a direct Koopman operator approach to predict future states in the latent space. These baselines provide a comprehensive evaluation of KPA's predictive performance against both classical and modern forecasting techniques.

3.5. Training strategy

We implement a training methodology based on [4], adapted specifically for SBAS displacement prediction. During training, each time–series sample is temporally split into two parts: the first N_{in} frames and the subsequent N_{out} frames. The initial N_{in} frames are used to train the network via two objectives—(i) frame reconstruction and (ii) autoregressive prediction of future states. The following N_{out} frames serve exclusively as validation targets: after the model has processed the N_{in} context frames, it must predict these unseen valudation and test N_{out} frames (see Figure 1), and we measure accuracy by the mean-squared error between the forecasts and ground-truth displacements.

4. Experimental Results

4.1. Training details

The critical hyperparameters for future prediction of satellite images using the proposed model include the learning rate (l_r), the decay rate of the learning rate (l_{rd}), the associated decay schedule, the maximum prediction step k_m, and the weights of the individual loss components: $\gamma_{\text{id}}, \gamma_{\text{fwd}}, \gamma_{\text{bwd}}, \gamma_{\text{con}}$, and γ_{lat}. Structural hyperparameters such as N_l also play an important role. The specific values of these hyperparameters are chosen for different scenarios in the satellite image prediction task. Note that $N_{\text{in}}, N_{\text{out}}$ are determined by the dimensionality of the input data and are not tunable. Training is performed for 600 epochs.

The model is trained for a total of 600 epochs. The initial learning rate is set to $l_r = 1 \times 10^{-4}$ and is decayed by a factor of $l_{rd} = 0.5$ every 200 epochs. The maximum prediction step calculating loss function is defined as $k_m = 5$. The number of input frames and the number of predicted output frames are determined by the structure of the dataset and are not subject to optimization. Regarding the loss function, the reconstruction loss is weighted by $\gamma_{\text{rec}} = 1.0$, the forward prediction loss by $\gamma_{\text{fwd}} = 0.5$, the backward prediction loss by $\gamma_{\text{bwd}} = 0.5$, and the latent consistency loss by $\gamma_{\text{lat}} = 0.1$. To stabilize early training, the latent consistency loss is disabled for the first 100 epochs by setting $\gamma_{\text{lat}} = 0$ during that period, and enabled thereafter. The dimensionality of the latent space, which determines the size of the Koopman operator, is set to $N_l = 64$, providing a compact yet expressive representation of the high-dimensional displacement imagery.

4.2. Displacement Datasets

The Small BAseline Subset (SBAS) technique is employed to generate time series satellite images, capturing ground displacement patterns with high temporal and spatial resolution. In this experiment, we use the public Japanese SBAS dataset [28](Table. 1), which contains the deformation data for 191 SBAS of velocities(mm/y). This dataset was calculated using Sentinel-1 by LiCSBAS [29]. Preprocessing steps involve phase unwrapping, noise filtering, and the exclusion of outliers. This dataset serves to evaluate the method's applicability to geophysical phenomena. Additionally, each input image was resized into 64×64 pixels. Specific dataset details are described in the supplemental material 6.2. To structure the temporal forecasting task, we divide the full SBAS time series into non-overlapping sequences. These sequences are then partitioned as follows: the earliest 30-frame sequence is used for training(N_{in}) and the middle 30-frame sequence for validation(N_{out}). Additionally, the most recent 20-frame sequence for testing. These validation and test data are predicted from the N_{in} frames used for training. To address the common challenge of limited SBAS data availability, we evaluate three different scenarios with varying N_{in} and N_{out} values. To that end, we additionally construct two extended-horizon variants of the dataset: one with $N_{\text{in}} = 10$ and $N_{\text{out}} = 50$, and another with $N_{\text{in}} = 20$ and $N_{\text{out}} = 40$. In both cases, the set of test frames is kept identical to the original split, ensuring a consistent evaluation protocol. For a detailed explanation, please see the Figure 6 in the supplementary material 6.2. This chronological split ensures realistic evaluation by mimicking practical forecasting scenarios where future observations are unknown during

Table 1. Displacement dataset in this experiment was targeted in Japan. This Displacement was calculated by the SBAS algorithm by LicSBAS [29] using Sentinel-1 and openly published [28].

FrameID	Date (yyyymmdd)		# Img.
	Start	End	
017D_05353_130400	20151118	20200308	109
017D_05514_131312	20141123	20200308	133
039A_05193_040711	20150430	20200304	113
039A_05372_151515	20150430	20200304	116
046D_04690_121308	20141207	20200322	128
046D_04907_081313	20141231	20200310	128
046D_05096_121313	20141231	20200310	126
046D_05292_131313	20141125	20200310	125
046D_05469_071311	20141125	20200322	132
068A_04667_071312	20150526	20200330	116
083A_05602_131212	20151205	20200224	105
090D_05485_121414	20151123	20200301	110
090D_05651_051011	20141116	20200301	132
112A_05300_030808	20150505	20200321	108
112A_05454_131213	20150505	20200226	117
119D_05226_070704	20150117	20200303	124
119D_05372_141313	20150728	20200327	117
119D_05525_040809	20141130	20200303	131
141A_04925_131313	20150507	20200228	120
141A_05118_131311	20150507	20200228	121
141A_05337_160700	20151209	20200228	103
148D_04631_101009	20141120	20200129	123
156A_05796_080500	20150508	20200405	101
163D_05579_100804	20141121	20200306	118
163D_05736_131313	20141121	20200330	136
163D_05934_091414	20141121	20200330	134

training. This setup enables the model to learn long-term temporal dynamics from historical deformation patterns and assess its ability to extrapolate future ground deformation based on past trends.

4.3. Forcasting Future State

We generated the future frames of test data using the trained KPA. We perform the tests for $N_{\text{in}} = 10, 20$ and 30 frames to predict test frames. The Mean Square Error(MSE) is calculated average over all predicted frames in only the test time data. The results are summarized in Table 2 and in Figure 4. We evaluate our proposed Koopman-Prior AutoEncoder (KPA) against several baseline methods on the SBAS dataset for time-series image prediction. Table 2 presents the MSE displacement in millimeters across different input sequence lengths (N_{in}).

Our experimental results demonstrate the superiority of KPA over traditional methods. Integrating an autoencoder significantly enhances performance, reducing prediction error by approximately 70% for both SSM and DMD, underscoring the importance of efficient latent space representation. KPA achieves competitive accuracy across all training sequence lengths, with MSE values of 39.192 mm, 35.929 mm, and 34.708 mm for N_{in} of 10, 20, and 30 frames, respectively. This matches state-of-the-art deep learning models like ConvLSTM while surpassing the Transformer, especially for shorter sequences. Notably, KPA outperforms its non-autoencoder counterpart (Koopman w/o AE) by up to 65%, validating our hypothesis that combining Koopman dynamics with autoencoder-based dimensionality reduction enhances spatiotemporal pattern prediction. Moreover, KPA maintains strong performance even with limited training context data ($N_{\text{in}} = 10$), a crucial advantage in scenarios where extensive training sequences are unavailable.

A key consideration is that Transformer architectures typically require substantial amounts of training data to fully leverage their self-attention mechanisms and realize their potential. The regional nature of our SBAS dataset, which is limited to observations over Japan, may not provide sufficient scale for optimal Transformer training. This aligns with previous findings in the literature that highlight the data-hungry nature of Transformer models [13, 21]. We hypothesize that the Transformer model's performance could potentially improve significantly if trained on a global-scale dataset incorporating SBAS measurements from multiple geographical regions. This would provide the model with the volume and diversity of data typically required for Transformer architectures to achieve their characteristic scaling benefits [8]. In contrast, our proposed KPA method demonstrates robust performance despite the limited dataset size. This efficiency in data utilization can be attributed to the inherent properties of Koopman operators, which effectively capture the underlying dynamical system with fewer examples due to their physically motivated mathematical foundation. This characteristic makes our approach particularly valuable for specialized applications where large-scale displacement is impractical or unavailable. The ability of KPA to achieve superior results with limited training data suggests that it offers a more practical solution for regional-scale applications, where data availability might be constrained by geographical or temporal limitations. This advantage becomes especially relevant in operational scenarios where rapid deployment with limited historical data is required.

4.4. Computational Efficiency Analysis

To evaluate the computational efficiency of our proposed KPA model using 20 frames as context input, we conduct comprehensive comparisons with state-of-the-art models, including Transformers and ConvLSTM networks. All experiments were performed on a single NVIDIA RTX 3080 GPU to ensure fair comparison. Specific experimental details are described in the supplemental material 8.1.

As shown in Table 3, our KPA achieves remarkable effi-

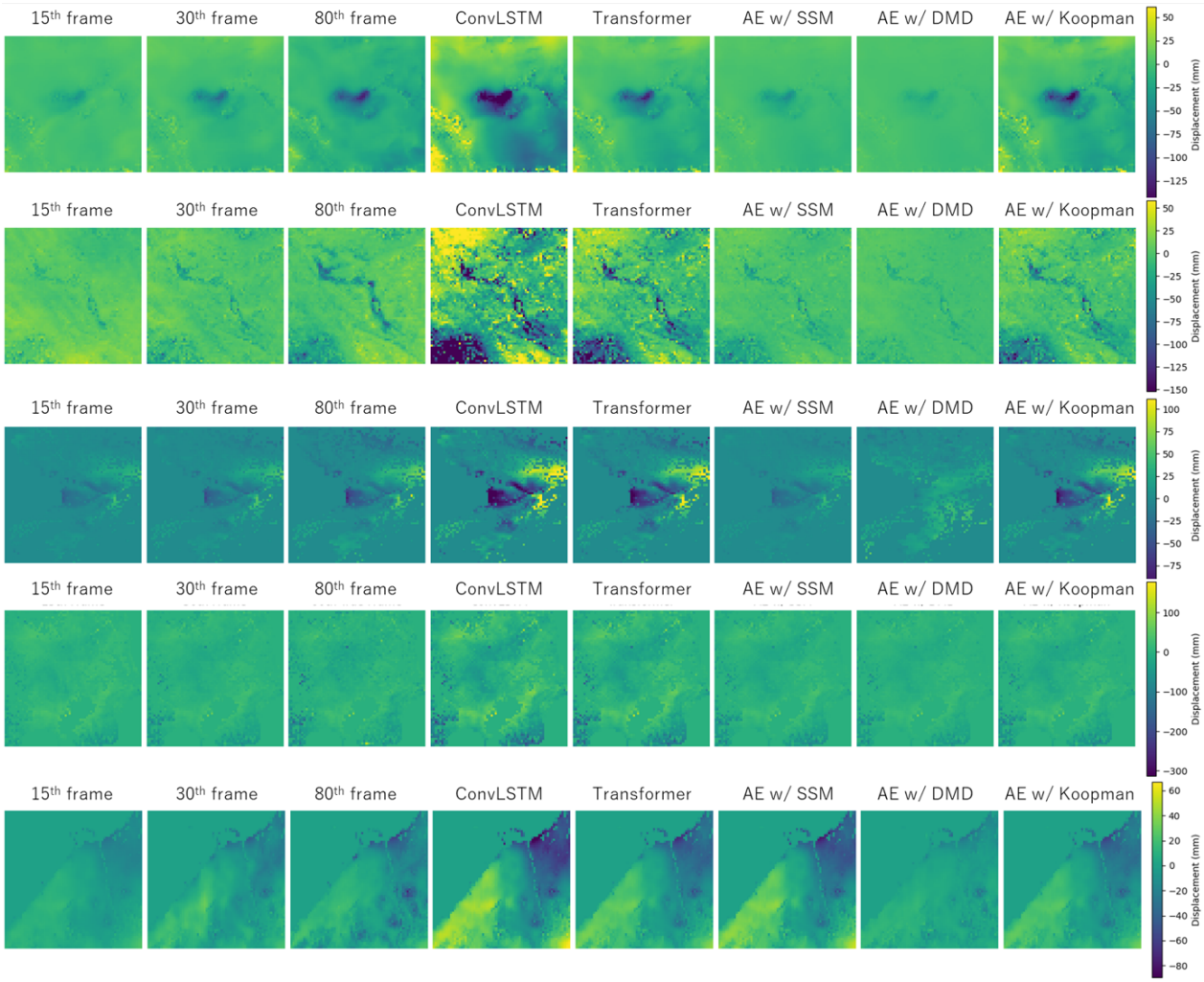


Figure 4. Starting the initial displacement image(the first frame of training data), we compared our KPA model(AE w/Koopman), Dynamic Mode Decomposition-based AutoEncoder(AE w/ DMD) and State Space Model-based Auto Encoder(AE w/ SSM) to forecast the future displacement of SBAS data for the 80th frames(the last frames of test data). All of the methods used the first 30 frames as input context frames and forecasted the next 50 frames.

Method	$N_{\text{in}} = 10$	$N_{\text{in}} = 20$	$N_{\text{in}} = 30$
SSM w/o AE	143.348	141.438	138.156
SSM w AE	41.782	39.238	36.593
DMD w/o AE	153.854	151.289	143.662
DMD w AE	45.721	42.299	38.327
Koopman w/o AE	129.101	102.716	100.243
Koopman w AE(Ours)	39.192	35.929	34.708
ConvLSTM	40.122	35.524	33.182
Transformer	83.316	45.625	40.537

Table 2. Prediction displacement of Mean Square Error (mm) comparison at test 20 frames: Our Koopman-Prior Auto Encoder(KPA) and baseline algorithms on SBAS dataset. N_{in} is used as input frames of SBAS dataset.

ciency improvements across all computational metrics. The parameter count of our model is merely 0.2M, which represents a reduction of 99.3% compared to the Transformer baseline (28M parameters) and 98.0% compared to ConvLSTM (10M parameters). This dramatic reduction in model complexity is attributed to our efficient Koopman operator design and the utilization of spectral methods through FFT.

In terms of computational complexity, KPA demonstrates exceptional efficiency, requiring only 19M FLOPs compared to 31.3G for Transformers and 76.5M for ConvLSTM. This represents a substantial 99.9% reduction in computational requirements compared to the Transformer baseline. The inference speed of our approach demonstrates a particular advantage in real-time applications, processing each frame in 1- 3ms compared to 15- 30ms for Transformers and 5-

Model	Parameters	FLOPs	Inference Time
Transformer	28M	31.3G	15-30ms
ConvLSTM	10M	76.5M	5-10ms
KPA (Ours)	**0.2M**	**19M**	**1-3ms**

Table 3. Computational Resource Requirements of our method and baseline models using the same 20 frames as context input.

10ms for ConvLSTM. This translates to a 10× speedup over Transformers and 3× over ConvLSTM, while maintaining competitive accuracy as shown in our previous experiments. This efficiency stems from the spectral representation and the linear nature of operations in the Koopman-embedded latent space, eliminating the need for complex attention mechanisms or recurrent computations.

4.5. Robustness Evaluation on Unseen Regions

To assess the robustness of our proposed Koopman Operator Autoencoder (KPA) in predicting ground deformation in unseen regions, we conducted evaluations on three independent volcanic areas: Mauna Loa in Hawaii (Sentinel-1 frame 087D_07004_060904), Mount Etna in Italy (Sentinel-1 frame 124D_05291_081406), and Mount Ararat in Turkey (Sentinel-1 frame 152D_04960_131313) (shown in Supplementary material 6.3). These locations were not included in the training dataset, providing a rigorous test of the model's generalization capability.

Our experimental results demonstrate that KPA effectively generalizes to these unseen locations, accurately predicting ground deformation patterns derived from SBAS data(Figure 5). The model using 20 frames as context input successfully reconstructs spatiotemporal deformation dynamics despite differences in geological characteristics, indicating its adaptability to diverse terrains. We evaluated on Mauna Loa the mean absolute error (MAE) was 525.2 mm, on Mount Etna it was 36.5 mm, and on Mount Ararat it reached 39.1 mm—each exceeding the Japanese SBAS data.

In the case of Mauna Loa, we observed instances where the predicted deformation deviated significantly from historical trends. This suggests that our approach could potentially be utilized for anomaly detection, as such deviations may indicate abnormal volcanic activity. This capability highlights the potential of KPA not only for standard forecasting tasks but also for early warning systems in volcanic monitoring applications. Overall, these findings confirm that our model maintains high robustness and applicability even in regions with no prior training data, supporting its effectiveness in real-world geophysical and remote sensing scenarios.

5. Conclusion

In this paper, we introduced Koopman-Prior AutoEncoder (KPA), an efficient and lightweight model for forecasting time-series displacement of SBAS data. KPA leverages the Koopman operator to map the encoder-extracted nonlinear latent space of displacement data into a linear latent space. A key aspect is enforcing temporal consistency in the latent variables by exploiting the time-invariance property of the Koopman operator for autonomous dynamical systems. We evaluated KPA on Japanese SBAS datasets, comparing it against physics-based techniques and recent time series prediction methods, including Diffusion models [18, 37]. Our method demonstrated superior performance with shorter training times and faster inference compared to state-of-the-art approaches. KPA's ability to handle limited data is significant for computationally demanding simulations of high-dimensional physical systems. Furthermore, this method can be extended to non-autonomous control systems using a bilinearly recurrent physics-based architecture based on [15].

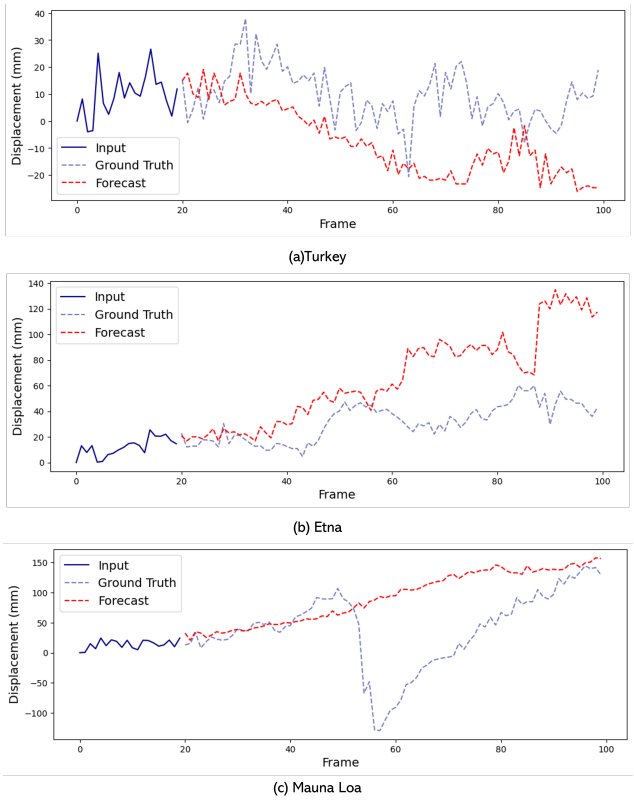


Figure 5. Predicted ground deformation on unseen regions by trained KPA using 20 frames as context input. The model successfully captures deformation trends across different geological environments.

Acknowledgements

This study makes use of displacement time–series data calculated by Sentinel-1 using LicSBAS. We used ABCI 3.0 provided by AIST and AIST Solutions[39].

References

[1] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1120–1128, New York, New York, USA, 2016. PMLR. 2

[2] Alex Aussem. Dynamical recurrent neural networks towards prediction and modeling of dynamical systems. *Neurocomputing*, 28(1-3):207–232, 1999. 2

[3] A. M. Avila and I. Mezić. Data-driven analysis and forecasting of highway traffic dynamics. *Nature Communications*, 11 (1):2090, 2020. 2

[4] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020. 3, 4, 5

[5] Coryn A L Bailer-Jones, David J C MacKay, and Philip J Withers. A recurrent neural network for modelling dynamical systems. *Network: Computation in Neural Systems*, 9(4): 531–547, 1998. PMID: 10221578. 2

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. 2

[7] Paolo Berardino, Gianfranco Fornaro, Riccardo Lanari, and Eugenio Sansosti. A new algorithm for surface deformation monitoring based on small baseline differential sar interferograms. *IEEE Transactions on geoscience and remote sensing*, 40(11):2375–2383, 2002. 1

[8] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 6

[9] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019. 2

[10] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: Chasing higher flops for faster neural networks. *arXiv preprint arXiv:2303.03667*, 2023. 4

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2

[12] Akshunna S. Dogra and William Redman. Optimizing neural networks via koopman operator theory. In *Advances in Neural Information Processing Systems*, pages 2087–2097. Curran Associates, Inc., 2020. 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 6

[14] D. Goswami and D. A. Paley. Global bilinearization and controllability of control-affine nonlinear systems: A Koopman spectral approach. In *2017 IEEE 56th Annual Conference on Decision and Control*, pages 6107–6112, 2017. 3

[15] Debdipta Goswami and Derek A. Paley. Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A koopman spectral approach. *IEEE Transactions on Automatic Control*, 67(6):2715–2728, 2022. 2, 3, 8

[16] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019. 2

[17] Albert Gu, Tri Dao, Stephan Gönner, Atri Rudra, and Christopher Ré. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2305.14327*, 2023. 5

[18] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022. 8

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[20] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. *Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles*, pages 558–566. 2

[21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 6

[22] Giancarlo Kerg, Kyle Goyette, Maximilian Puelma Touzel, Gauthier Gidel, Eugene Vorontsov, Yoshua Bengio, and Guillaume Lajoie. Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 2

[23] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of National Academy of Sciences*, 17:315–318, 1931. 2

[24] Henning Lange, Steven L Brunton, and J Nathan Kutz. From fourier to koopman: Spectral methods for long-term time series prediction. *The Journal of Machine Learning Research*, 22(1):1881–1918, 2021. 3

[25] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018. 2, 3

[26] A. Mauroy and I. Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, 2016. 2

[27] John Miller and Moritz Hardt. Stable recurrent models. In *International Conference on Learning Representations*, 2019. 2

[28] Yu Morishita. Nationwide urban ground deformation monitoring in japan using sentinel-1 licsar products and licsbas. *Progress in Earth and Planetary Science*, 8(1):1–23, 2021. 5, 6

[29] Yu Morishita, Milan Lazecky, Tim J Wright, Jonathan R Weiss, John R Elliott, and Andy Hooper. Licsbas: An open-source insar time series analysis package integrated with the licsar automated sentinel-1 insar processor. *Remote Sensing*, 12(3):424, 2020. 5, 6

[30] Sriram S. K. S. Narayanan, Duvan Tellez-Castro, Sarang Sutavani, and Umesh Vaidya. Se(3) koopman-mpc: Data-driven learning and control of quadrotor uavs, 2023. 2

[31] Indranil Nayak, Fernando L Teixeira, and Mrinal Kumar. Koopman autoencoder architecture for current density modeling in kinetic plasma simulations. In *2021 International Applied Computational Electromagnetics Society Symposium (ACES)*, pages 1–3. IEEE, 2021. 2

[32] Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019. 2, 3

[33] Sebastian Peitz, Samuel E. Otto, and Clarence W. Rowley. Data-driven model predictive control using interpolated Koopman generators. *SIAM Journal on Applied Dynamical Systems*, 19(3):2162–2193, 2020. 2, 3

[34] 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. 2, 3

[35] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. 2, 3

[36] Peter J. Schmid. *Dynamic mode decomposition of numerical and experimental data*. Cambridge University Press, 2010. 5

[37] Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Mostgan-v: Video generation with temporal motion styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5652–5661, 2023. 8

[38] David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, 2013. 2

[39] Ryousei Takano, Shinichiro Takizawa, Yusuke Tanimura, Hidemoto Nakada, and Hirotaka Ogawa. Abci 3.0: Evolution of the leading ai infrastructure in japan, 2024. 8

[40] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in neural information processing systems*, 30, 2017. 3

[41] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014. 5

[42] 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, 2015. 2, 3

[43] Matthew O. Williams, Maziar S. Hemati, Scott T. M. Dawson, Ioannis G. Kevrekidis, and Clarence W. Rowley. Extending data-driven Koopman analysis to actuated systems. *IFAC-PapersOnLine*, 49(18):704–709, 2016. 3

[44] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839, 2019. 2

Ground-Displacement Forecasting from Satellite Image Time Series via a Koopman-Prior Autoencoder

Supplementary Material

6. Dataset

6.1. Train/Val/Test

Figure 6 schematically summarizes the three chronological split strategies (A, B, C) used in our experiments. The top blue bar represents the full, ordered sequence of SBAS frames; the coloured segments beneath it indicate how each strategy partitions those frames into training (blue), validation (orange), and test (green) sets.

Strategy A (30/30/20). This strategy is used for long context length. The first 30 frames are used for training, the next 30 frames for validation, and the final 20 frames for testing.

Strategy B (20/40/20). To investigate a baseline context length, we allocate the first 20 frames to training, the subsequent 40 frames to validation, and keep the same 20-frame test window as in Strategy A. Within this split, we train the model with

Strategy C (10/50/20). Here, the shortest training context remains 10 frames, while the validation horizon is extended to 50 frames; the last 20 frames again serve as a fixed test set.

Across all three strategies, the test portion (right-most green segment) is identical, guaranteeing that every model is evaluated on exactly the same 20 unseen frames. The differing sizes of the training and validation windows allow us to study how context length and prediction horizon affect learning dynamics and generalisation performance.

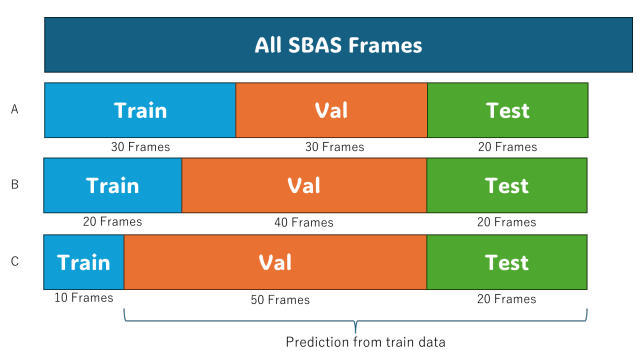


Figure 6. Data splitting strategies for training, validation, and testing. We train a model using only frames of training data, and using that learned model, we predict frames of validation and testing data.

6.2. SBAS dataset in Japan

We evaluate our method using the Small Baseline Subset (SBAS) dataset, which consists of time-series synthetic aperture radar (SAR) interferograms capturing ground deformation over a specified region. The SBAS technique reconstructs surface displacement by minimizing temporal and spatial decorrelation effects, making it well-suited for studying geophysical phenomena such as seismic activity and land subsidence.

For our experiments, we use SBAS-derived displacement maps with a spatial resolution of X meters and a temporal resolution of Y days, covering a total of Z frames. The dataset provides a rich source of spatiotemporal patterns, enabling us to assess the effectiveness of our KPA model in forecasting surface deformation.

6.3. Out of distribution dataset

For training, we utilize the SBAS dataset from Japan. To evaluate the robustness of our method, we test it on SBAS data from regions not included in training, specifically Turkey, Italy, and Hawaii. This cross-region evaluation allows us to assess the generalization capability of our approach in diverse geophysical settings.

Figure 7 illustrates the mean displacement velocity of the SBAS data. The points in the figure represent sampled locations corresponding to the evaluation sites discussed in the text.

7. Method

7.1. Koopman Layer

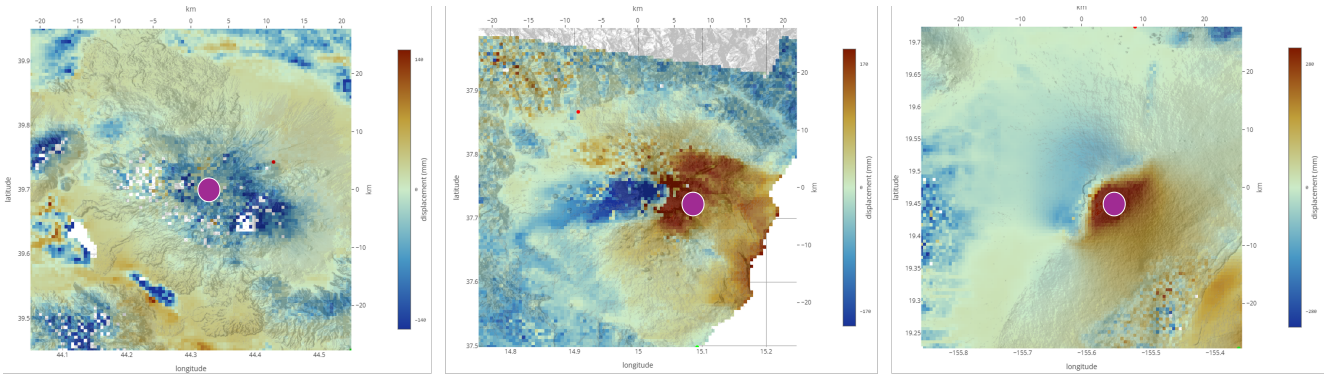
The latent state $\mathbf{z}_n \in \mathbb{R}^{N_l}$ resides in the bottleneck (cyan); a *single* learnable matrix $\mathbf{K} \in \mathbb{R}^{N_l \times N_l}$ then advances the dynamics:

$$\mathbf{z}_{n+1} = \mathbf{K}\,\mathbf{z}_n, \qquad \mathbf{z}_{n+k} = \mathbf{K}^k\,\mathbf{z}_n \;(k \geq 1). \tag{12}$$

Because $\mathbf{K}$ is declared as a parameter of the network it is updated by back-propagation alongside all convolutional weights. In PyTorch the layer can be written concisely:

```python
import torch, torch.nn as nn

class KoopmanLayer(nn.Module):
    def __init__(self, latent_dim: int):
        super().__init__()
        # initialise K as (noisy) identity; requires_grad=True by default
        self.K = nn.Parameter(torch.eye(latent_dim))
```



(a)Turkey (b) Etna (c) Mauna Loa

Figure 7. SBAS data for evaluating the robustness of our trained model.

```
    def forward(self, z, steps: int = 1):
        """Propagate latent code z forward by `steps`."""
        if steps == 1:
            return z @ self.K.T
        Kk = torch.matrix_power(self.K, steps)
        return z @ Kk.T
```

Here `self.K` is a fully learnable weight matrix; the call `z @ self.K.T` realises the update $\mathbf{z}_{n+1} = \mathbf{K}\mathbf{z}_n$. Multi-step propagation uses `torch.matrix_power` to obtain $\mathbf{K}^k$ without explicit loops, keeping the operation both GPU-friendly and differentiable.

8. Experimental Results

8.1. Model Efficiency Analysis

Comparison of Model Architectures for Time Series Image Prediction. In this supplementary material, we provide a detailed analysis of the computational efficiency of our proposed model compared to conventional ConvLSTM and Transformer-based approaches for the task of predicting the next frame ($64 \times 64 \times 1$) from time series image data ($64 \times 64 \times 20$ch).

Parameter Count Comparison Table 4 shows the parameter count for each model architecture with default hyperparameter settings. Our model achieves a significant reduction in parameter count compared to traditional approaches. Specifically, our model requires only 0.2 million parameters, which is approximately 50 times fewer than ConvLSTM (10M) and 140 times fewer than Transformer-based architectures (28M). This dramatic reduction in model size is primarily attributed to the efficient Koopman operator representation and the utilization of spectral methods through FFT, which enables capturing complex dynamics with fewer parameters.

Table 4. Parameter Count Comparison

Model	Parameter Count	Relative Size
Ours	0.2M	1×
ConvLSTM	10M	50×
Transformer	28M	140×

Computational Complexity Analysis. Table 5 presents the theoretical computational complexity in terms of floating point operations (FLOPs) required for a single forward pass with batch size 1. The computational advantage of our approach is particularly evident when compared to Transformer-based models, requiring approximately 1,647 times fewer operations. Our model also offers a 4-fold reduction in computational requirements compared to ConvLSTM architectures, making it significantly more efficient for real-time applications and deployment on resource-constrained devices.

Table 5. Computational Complexity Comparison

Model	FLOPs	Relative Computation
Ours	19M	1×
ConvLSTM	76.5M	4×
Transformer	31.3G	1,647×

Inference Time Estimation. Table 6 provides estimated inference times on modern hardware for a single input sample. These measurements demonstrate that Ours not only has theoretical efficiency advantages but also translates to practical performance benefits, with approximately 5-10× faster inference times on both GPU and CPU hardware compared to Transformer models.

Table 6. Estimated Inference Time Comparison

Model	GPU Time (RTX 3080)	CPU Time
Ours	1-3 ms	10-30 ms
ConvLSTM	5-10 ms	50-100 ms
Transformer	15-30 ms	200-400 ms