Physics-Informed Deep Neural Network for Backward-in-Time Prediction: Application to Rayleigh–Bénard Convection

MOHAMAD ABED EL RAHMAN HAMMOUD,[a] HUMAM ALWASSEL,[a] BERNARD GHANEM,[a] OMAR KNIO,[a] AND IBRAHIM HOTEIT[a]

[a] *King Abdullah University of Science and Technology, Thuwal, Saudi Arabia*

(Manuscript received 19 October 2022, in final form 1 February 2023)

ABSTRACT: Backward-in-time predictions are needed to better understand the underlying dynamics of physical fluid flows and improve future forecasts. However, integrating fluid flows backward in time is challenging because of numerical instabilities caused by the diffusive nature of the fluid systems and nonlinearities of the governing equations. Although this problem has been long addressed using a nonpositive diffusion coefficient when integrating backward, it is notoriously inaccurate. In this study, a physics-informed deep neural network (PI-DNN) is presented to predict past states of a dissipative dynamical system from snapshots of the subsequent evolution of the system state. The performance of the PI-DNN is investigated using several systematic numerical experiments and the accuracy of the backward-in-time predictions is evaluated in terms of different error metrics. The proposed PI-DNN can predict the previous state of the Rayleigh–Bénard convection with an 8-time-step average normalized ℓ_2 error of less than 2% for a turbulent flow at a Rayleigh number of 10^5.

KEYWORDS: Data assimilation; Dynamical system model; Nonlinear models; Deep learning

1. Introduction

Accurate state representation of physical fluid flow systems is essential for thorough analyses of the underlying dynamics. In general, numerical models are developed to propagate the solution of a dynamical system forward in time to provide forecasts of the system state. Backward-in-time models have been shown to improve the accuracy of forward-in-time forecasts (Auroux and Blum 2008). These models were used in a data assimilation setting with large-scale ocean and atmospheric models. For example, Ruggiero et al. (2015) implemented a forward–backward nudging data assimilation algorithm using a negative diffusion coefficient to stabilize the Nucleus for European Modelling of the Ocean (NEMO) model over a short period of time. Forward–backward nudging algorithms were also examined to incorporate high-frequency remotely sensed observations of low-level wind into a high-resolution MesoNH mesoscale model (Boilley and Mahfouf 2012), to analyze the Lorenz'05 (Osborne 2021) and the two-dimensional shallow-water model (Auroux et al. 2016). In numerical weather prediction, the short-term estimation of the system state was also improved by nudging the solution variables in backward-in-time integration while disregarding the diffusion term (Ma et al. 2015). These advances motivate the use of backward-in-time predictions in a data assimilation setting to improve the forecast accuracy.

The most common methods for mitigating the impact of diffusion in backward integration in environmental flow applications involve either disregarding the diffusion term (e.g., Zodiatis et al. 2012) or setting the diffusion coefficient to a negative value (e.g., De Dominicis et al. 2013a,b; Dung and Thien 2014). However, both techniques often diverge from the actual solution or yield unreliable past predictions (Herty et al. 2020). Little research has been performed on developing robust backward-in-time integration models that provide consistent-in-time solutions (Chaturvedi et al. 2013). These efforts are hindered because the diffusion time scale limits the backward integration period, raising numerical difficulties in simulating the quantity and making the problem of obtaining accurate backward-in-time forecasts "uniquely challenging" (Clement 2010; Sun and Sun 2017). This problem was also explicitly noted by Ruggiero et al. (2015) stating that the key limitation of their proposed nudging algorithm is the accuracy of the backward-in-time integration.

This study leverages advances in artificial intelligence to train a deep neural network that provides accurate backward-in-time predictions without modifying the diffusion parameter. The popularity of deep learning can be attributed to the ability of neural networks (NNs) to learn complex relationships between the input and output fields (Goodfellow et al. 2016). NNs are deployed to perform various tasks such as image classification (Deng 2014), semantic segmentation (Li et al. 2019), temporal action detection (Alwassel et al. 2018), and autonomous driving (Amini et al. 2020). Deep learning has been recently applied to model complex physical phenomena, including modeling fluid flows (Beintema et al. 2020; Jiang et al. 2020), subgrid-scale parameterization of climate models (Rasp et al. 2018), subgrid-scale parameterization of wall models in large eddy simulations (Yang et al. 2019), modeling heat transfer (Sheikholeslami et al. 2019; Tamaddon-Jahromi et al. 2020), and other computational engineering applications (Saha et al. 2021; Haghighat and

Authors Hammoud and Alwassel contributed equally to this work.

Supplemental information related to this paper is available at the Journals Online website: https://doi.org/10.1175/AIES-D-22-0076.s1.

Corresponding author: Ibrahim Hoteit, ibrahim.hoteit@kaust.edu.sa

DOI: 10.1175/AIES-D-22-0076.1 e220076

Juanes 2021). Physics-informed neural networks (PINNs) are also introduced for enforcing physical constraints by appending the total loss function, which is minimized to train the network, with the norm of the residual of governing equations, boundary, and initial conditions (Raissi et al. 2019). A comprehensive review of recent developments in PINNs and their applications was presented by Karniadakis et al. (2021).

Here, the proposed physics-informed deep neural network (PI-DNN) is trained to predict the past states of a physical flow system from the snapshots of subsequent evolution of the fluid flow field. To the best of our knowledge, this is the first study to use an PI-DNN for backward-in-time predictions of dissipative fluid flows to mitigate the problems caused by diffusion. The Rayleigh–Bénard convection is used as a test bed, where the corresponding equations are solved for several distinct initial conditions at different Rayleigh numbers (Ra) to generate training and validation data. The NN comprises a U-net for feature extraction and a multilayer perceptron (MLP) for predicting the previous states and calculating the partial derivatives of the solution variables with respect to the coordinates, which are subsequently used to enforce the governing equations.

The combination of a feature extractor followed by a task-performing network is extensively applied in the deep learning–related literature (Long et al. 2015). In the context of spatiotemporal coordinates, this method was first proposed by Chen and Zhang (2019) and Saito et al. (2019) for three-dimensional (3D) point cloud applications and was later adapted to PINNs (Jiang et al. 2020). A feature extractor can extract correlations in the input fields and provide a grid of latent features, which includes information about the input data that cannot be measured directly. The latent feature vectors are then input to the task-performing network to provide the context of the input fields and enhance the predictability of the task-performing network. This method has the advantages of simplicity and effectiveness, as demonstrated in the deep learning–related literature (e.g., Goodfellow et al. 2016). Other kernel- or dictionary-based techniques are also expected to perform well. In particular, several techniques have been developed to learn differential operators, a map between different function spaces, based on NNs, for example, by using Koopman operators (Lusch et al. 2018), DeepONets (Lu et al. 2021), or neural operators Li et al. (2020, 2021). Fourier neural operators, a subclass of neural operators, showed performance comparable to that of DeepONets (Lu et al. 2021). Recently, Bakarji et al. (2022) introduced the application of autoencoders in conjunction with sparse system identification for nonlinear dynamics (Brunton et al. 2016) to recover the system of nonlinear equations using partial observations of the system states. A direct comparison of the dictionary-, kernel-, and convolution-based methods has yet to be performed. However, based on the literature, they all perform comparably and reasonably well, provided the network is appropriately trained, as all the abovementioned methods represent surrogate models that describe a given phenomenon.

Numerical experiments conducted in the present study examine the accuracy and robustness of the proposed framework at providing backward predictions for flows at different Ra. Furthermore, the robustness of a well-trained model is evaluated to predict the past states of a flow at an Ra different from that used to train the NN. Here, high accuracy refers to achieving low error values and robustness refers to the generalizability of the trained network to solution trajectories not observed during training. The study also investigates the sensitivity of the NN to the time lag between the start of the input and that of the output. It further examines the effect of the frequency of observing the system state for a fixed physical prediction time on the accuracy of the NN. The potential to obtain long-term backward-in-time predictions by recursively evaluating the trained model is also explored. The numerical results suggest that for $\mathrm{Ra} = 10^6$, a well-trained model can reach down to 5% averaged normalized ℓ_2 prediction error, while a model trained at a lower Ra of 10^5 can achieve down to 2% averaged normalized ℓ_2 prediction error. Integral quantities, such as the total energy and the Nusselt number, are accurately predicted with negligible error values. Although the framework is established on the basis of the Rayleigh–Bénard convection, the physical model can be substituted with a more practical model to extend the scope of the framework in future work.

The study is organized as follows: section 2 describes the Rayleigh–Bénard convection system, the numerical schemes applied to solve the governing equations, and the datasets used for training and validation. Section 3 outlines the NN architecture and describes the training process. Section 4 presents the numerical experiments and analysis of the results. Finally, section 5 summarizes the major findings and discusses future work.

2. Rayleigh–Bénard convection

The Rayleigh–Bénard convection is a popular canonical flow that models the thermal instability of a fluid constrained between a warm bottom plate and a relatively cold top plate (Pandey et al. 2018). The Ra of a fluid affects the complexity of the Rayleigh–Bénard flow. The flow becomes chaotic and is characterized by distinct vortical structures beyond a critical Ra (Curry et al. 1984; Paul et al. 2007).

Consider a 2D periodic rectangular channel with a horizontal period and a vertical height, $\tilde{L}_x$ and $\tilde{L}_y$, respectively. The system of equations on the fundamental domain $\Omega = [0, \tilde{L}_x/\tilde{L}_y] \times [0, 1]$ can be solved by recasting the governing equations to their dimensionless form in the Boussinesq limit with the proper scaling (Le Quéré 1991; Le Maître et al. 2002). The equations are expressed as follows:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \frac{\mathrm{Pr}}{\sqrt{\mathrm{Ra}}} \nabla^2 \mathbf{u} + \mathrm{Pr}\Theta \mathbf{e}_y, \tag{2}$$

$$\frac{\partial \Theta}{\partial t} + (\mathbf{u} \cdot \nabla)\Theta = \frac{1}{\sqrt{\mathrm{Ra}}} \nabla^2 \Theta + \mathbf{u} \cdot \mathbf{e}_2, \tag{3}$$

where $\mathrm{Ra} = (g\tilde{\alpha}\Delta\tilde{T}\tilde{L}_y^3)(\tilde{\nu}\tilde{\kappa})^{-1}$, g is the gravitational acceleration, $\tilde{L}_y$ is the vertical spacing between the plates, $\Delta\tilde{T}$ is the absolute temperature difference between the plates, $\tilde{\alpha}$ is the thermal expansion coefficient of the fluid, $\tilde{\nu}$ is the kinematic viscosity, $\tilde{\kappa}$ is the thermal diffusivity coefficient, $\mathrm{Pr} = \tilde{\nu}\tilde{\kappa}^{-1}$ is

the Prandtl number, t is time, and $\mathbf{e}_y$ is the standard basis vector in the vertical direction. The solution fields are described by the nondimensional velocity vector $\mathbf{u} = (u, v)$, pressure p, and nondimensional temperature Θ.

The governing equations can be solved once the initial and boundary conditions are specified. In this study, all solution fields are initialized with random noise beginning from unique seeds to promote diversity in the evolution of the flow, providing the network with various distinct features for learning and testing. Specifically, at an Ra larger than the critical Ra, the solution becomes more sensitive to the initial conditions where small perturbations lead to noticeable differences in the solution fields (Chertovskih et al. 2015). The boundary conditions are kept constant for all simulations with periodic conditions along the vertical boundaries. At the top and bottom boundaries, no-slip, no-penetration, and isothermal conditions are enforced. The following are the initial and boundary conditions:

$$\mathbf{u}(0; x, y) \sim \mathcal{U}(-0.1, 0.1)\,\forall (x, y) \in \Omega, \tag{4}$$

$$\Theta(0; x, y) \sim \mathcal{U}(-0.1, 0.1)\,\forall (x, y) \in \Omega, \tag{5}$$

$$\mathbf{u}(t; x, 0) = \mathbf{u}(t; x, 1) = 0, \quad \text{and} \tag{6}$$

$$\Theta(t; x, 0) = \Theta(t; x, 1) = 0, \tag{7}$$

where $\mathcal{U}(a, b)$ represents a uniform distribution on the interval (a, b).

a. Numerical simulation

A finite-difference technique is used to numerically solve the system of equations on a uniformly spaced, staggered Cartesian grid. Other discretization schemes could be used to solve the governing equations provided they are stable and the resulting solution is sufficiently accurate; in other words, a stable mesh-independent solution should be achieved (Seeni et al. 2021). The domains of the horizontal length L_x and vertical length $\tilde{L}_y$ are discretized into n_x and n_y numerical grid points, respectively, resulting in a computational cell resolution of $\Delta x = \tilde{L}_x/(\tilde{L}_y/n_x)$ in the horizontal direction and $\Delta y = 1/n_y$ in the vertical direction.

A third-order Adam–Bashforth scheme is implemented for time integration with a time step Δt and a second-order central differencing scheme is used for the advection and diffusion terms. A pressure-projection method is employed to satisfy the incompressibility condition, and a fast Fourier transform-based algorithm is used to solve the associated elliptic pressure equation. Figure 1 illustrates snapshots of the temperature fields for Ra of 10^5, 10^6, and 10^7. More intricate features are observed with increasing Ra, which are manifested in terms of fine-scale features and fast dynamics.

b. Training and validation datasets

The training and validation datasets are generated by solving the governing equations multiple times for distinct initial conditions to diversify the trajectories of the solution fields. Note that the Rayleigh–Bénard convection admits multiple unstable states for Ra values greater than the critical value (Cao et al. 2022). Moreover, we reiterate that the Rayleigh–Bénard convection solution strongly depends on the initial conditions of the system, where different initial conditions would yield different solution trajectories (Greenside and Coughran 1984; van der Poel et al. 2011; Chertovskih et al. 2015; Yigit et al. 2015).

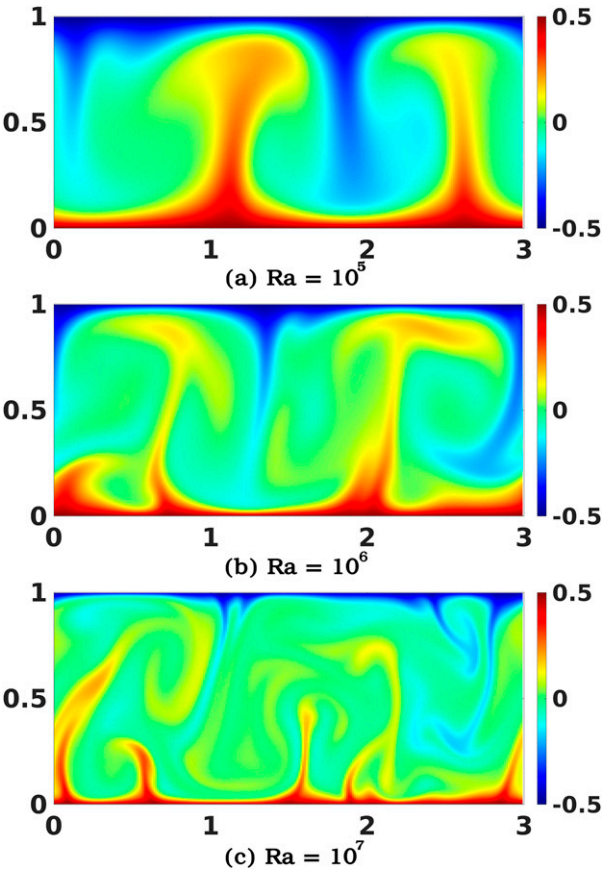


FIG. 1. Snapshots of Rayleigh–Bénard solution fields. Snapshots of the temperature field of the Rayleigh–Bénard convection fields corresponding to flows with Ra of (a) 10^5, (b) 10^6, and (c) 10^7. The illustrations describe the chaotic dynamics of the system for increasing Ra, where more fine-scale features appear with greater Ra.

Datasets were generated for flows with Ra = 10^5, 10^6, and 10^7, where each training and validation dataset consists of 20 and 5 unique solution trajectories, respectively, each starting from a unique initial field. Datasets corresponding to different Ra values enable analyzing the framework for various magnitudes of chaos in the system, where for a high Ra value, the flow resembles atmospheric convection (Weidauer et al. 2010; Chillà and Schumacher 2012). Each solution trajectory was generated by solving the governing equations over a domain with $L_x = 3$ and $L_y = 1$, discretized into $[n_x, n_y] = [384, 128]$ grid points. A fine time step $\Delta t = 5 \times 10^{-4}$ is used, which results in a maximum grid Reynolds number of approximately $\mathcal{O}(10)$, and Courant–Frechet–Louie number below 0.2, which ensures that the generated solution is stable.

The transient of each solution trajectory was discarded from the dataset; thus, a temporal subset of the evolution trajectory within the fully developed regime and corresponding to $t \in [15, 60]$ was extracted for training and validation. Specifically, the PI-DNN is trained to obtain predictions in the fully developed regime and when the solution is evolving on the attractor. Although the numerical solution is obtained at a high temporal resolution, the solution fields at temporal increments of δt_o are considered within this time span, where $\delta t_o = 0.05$ for $\mathrm{Ra} = 10^7$ and $\delta t_o = 0.1$ for a smaller Ra. This allows for variation in the solution fields over different time steps. This setup weakly translates to environmental flow applications, which have long been hypothesized to evolve on a chaotic attractor (Hastings et al. 1993; Angelbeck and Minkara 1994).

In the present work, the input and output to the PI-DNN is called a clip and is comprised of a finite number of snapshots of the solution fields over the spatial domain. In other words, the input and output clips have the dimensions (n_C, n_x, n_y, n_t), where n_C is the number of solution variables, n_x and n_y and the number of spatial points along the horizontal and vertical, and n_t is the number of snapshots. To augment the training and validation dataset, and as typically performed in the computer-vision literature, the PI-DNN is trained using spatiotemporal crops or subsets selected from the entire solution domain (Simonyan and Zisserman 2015). Spatial subsets are extracted for training, where the training and validation clips have the dimensions of $(n_C, n_{x,c}, n_{y,c}, n_t)$ with $n_{x,c}$ and $n_{y,c}$ are the number of points along the horizontal and vertical. It is noteworthy to mention that although training and validation are performed on the spatial crops, evaluation is performed on the entire spatial domain, where an evaluation clip has the dimensions (n_C, n_x, n_y, n_t). As described in the problem statement, the input clip corresponds to the future evolution of the solution fields, whereas the output clip corresponds to the solution field at a past time.

We manipulate the temporal resolution and overlap between the input and output clips by relying on dimensionless quantities, namely, the subsampling rate and stride, that are commonly used in the deep learning literature [e.g., Hou et al. (2017) and Scheidegger et al. (2017)]. The temporal subsampling rate (sr) is introduced as the temporal frequency at which data are sampled, such that the time step between consecutive frames $\delta t = \delta t_o \times \mathrm{sr}$. The sampled clips are also selected such that the output and input clips start at times t_1 and t_2, respectively. The time lag between the start of the input clip and that of the output clip is characterized by the stride, $\mathrm{st} \equiv (t_2 - t_1)/\delta t$. Note that sr refers to the temporal resolution of the input data, whereas st describes the time lag between the start of the input and output fields.

3. Neural network architecture

We propose a two-stage PI-DNN for the backward-in-time prediction of the dynamic dissipative fluid flow state. The deep neural network (DNN) comprises a 3D U-net, used for extracting the spatiotemporal features, followed by an MLP, used for the backward-in-time prediction. The network architecture constitutes building blocks that have been extensively validated in the relevant literature. The efficiency of a U-net (Ronneberger et al. 2015) in extracting features and an MLP network (Rosenblatt 1961; Chen and Chen 1995) in solving regression problems makes this combination attractive for solving the problem at hand. The architecture of the feature extractor is discussed in section 3a and that of the MLP in section 3b. Furthermore, the training pipeline and a more detailed schematic of the architecture of the network are illustrated in Fig. 2. The total loss function and its components are presented in section 3c. Section 3d presents the error metrics used to evaluate the performance of the proposed network. Finally, the training procedure is described in section 3e.

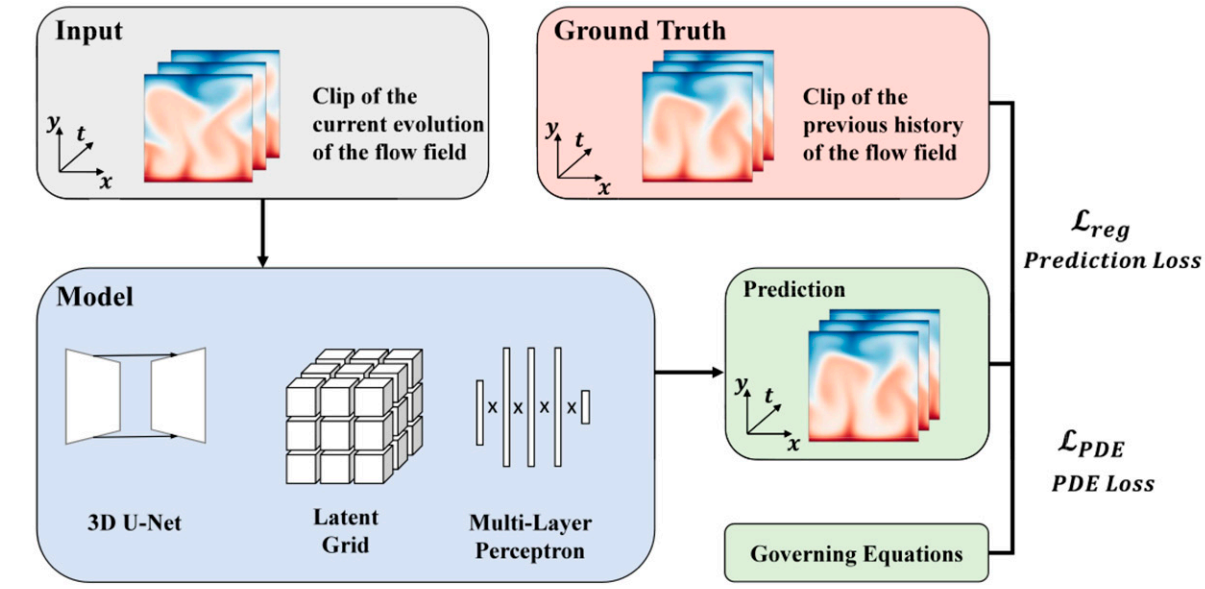
a. Feature extraction

The first stage of the pipeline comprises feature extraction using a convolutional neural network. In particular, a modified U-net architecture is used to extract features from the input clips. Although U-nets were initially employed for semantic segmentation (Ronneberger et al. 2015), their applications extend to other computer vision tasks, including image classification (Cao and Zhang 2020), image superresolution (Lim et al. 2017), and dense object tracking (Bozek et al. 2018). In our application, the U-net is used to extract spatiotemporal features from a multichannel 3D input (x, y, t), with temperature, pressure, and the two velocity components as channels.

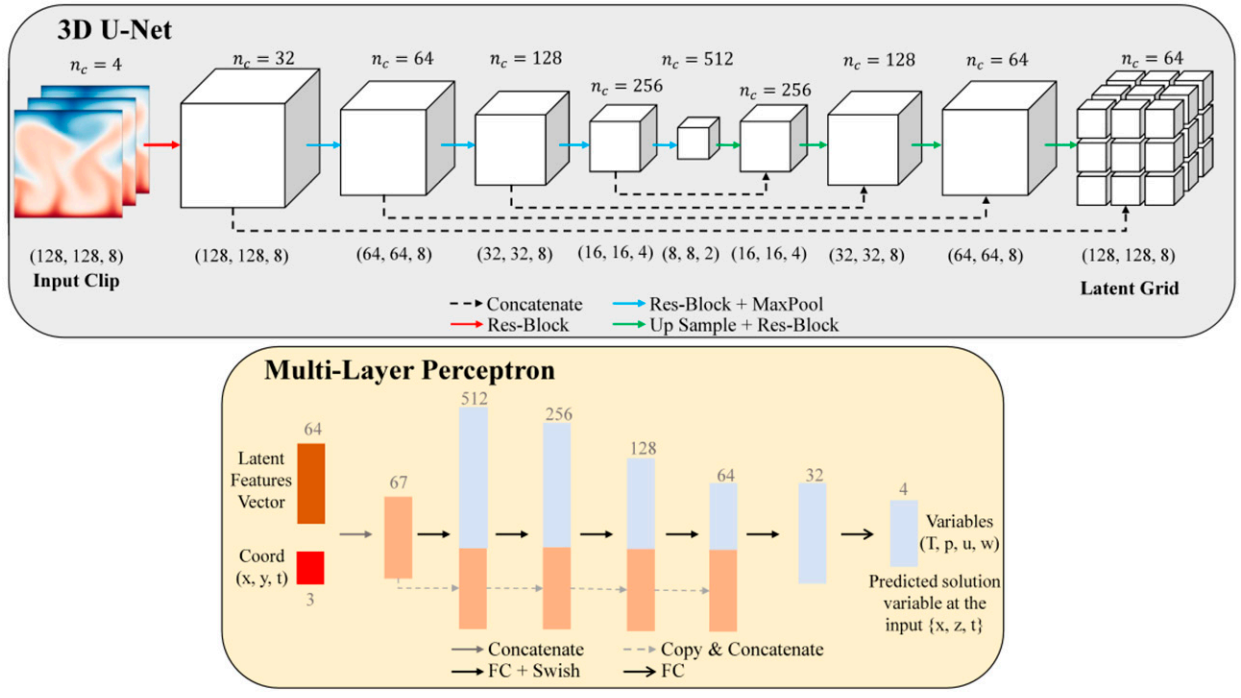
Our U-net involves 3D convolution operations instead of the 2D convolutions suggested in the original implementation. In addition, we use residual blocks (ResBlocks) to benefit from improved training properties (He et al. 2016). The network is composed of an encoder followed by a decoder. The encoder comprises many stages of convolutional ResBlocks that are immediately followed by a max-pooling layer with a stride of 2. Each ResBlock is characterized by a sequence of three convolutional layers of kernel sizes $1 \times 1 \times 1$, $3 \times 3 \times 3$, and $1 \times 1 \times 1$. Each convolution layer is paired with a batch normalization (BatchNorm) layer and rectified linear activation unit (ReLU) nonlinearity. The input to each convolution layer is zero-padded to maintain the spatiotemporal dimensions constant throughout the convolution layer. The decoder mimics the encoder part of the network with nearest-neighbor upsampling substituting the max-pooling layers. A skip connection concatenates the features from each layer of the encoder to those of the same level from the decoder; this is similar to the original U-net implementation. These skip connections promote strong gradient propagation and maintain information from early layers (Ronneberger et al. 2015; He et al. 2016).

b. Multilayer perceptron network

The second stage of the proposed PI-DNN is an MLP network, which takes as input the spatiotemporal features from the U-net and the relative spatiotemporal coordinates of the input clip and predicts on a point-by-point basis the temperature, pressure, and velocity components of the previous clip. The relative spatiotemporal coordinates need to be passed to the MLP to allow the calculation of partial derivatives of the output variables with respect to space and time. This approach was first introduced by Raissi et al. (2019), where



(a) Schematic of the training procedure



(b) Schematic of the neural network architecture

FIG. 2. Overview of the training pipeline. (a) Schematic of the training procedure. The PI-DNN uses the current evolution of the flow field as the input and predicts the past evolution of the flow field. Specifically, the input clip is transformed using a 3D U-net to produce a latent grid of spatiotemporal features. These features are then passed to an MLP to predict the states of the system backward in time. We compare the predicted clip with the ground truth history of the flow field using a regression loss ($\mathcal{L}_{\text{reg}}$) and employ a PDE loss ($\mathcal{L}_{\text{PDE}}$) to enforce the physics of the governing equations. (b) Schematic of the 3D U-net architecture applied for extracting spatiotemporal features and the MLP used for providing backward-in-time predictions.

TABLE 1. Performance results. The PI-DNN is evaluated on the dataset with Ra = 10^6, sr = 2, $\delta t = 0.2$, and st = 8. The training and validation losses for different combinations of LR and γ used for training are presented along with the RRMSEs of the solution variables. The best LR and γ settings are highlighted in bold.

γ	LR	RRMSE_T	RRMSE_p	RRMSE_u	RRMSE_v	RRMSE_{T_0}	$\text{RRMSE}_{T_{-1}}$	Training loss	Validation loss
0.000	0.001	0.1155	0.0981	0.0809	0.0768	0.0861	0.0824	0.0676	0.0685
0.000	0.010	0.0811	0.0737	0.0601	0.0643	0.0588	0.0577	0.0383	0.0520
0.000	0.050	0.1032	0.0947	0.0960	0.0812	0.0792	0.0806	0.0436	0.0807
0.000	0.100	0.0620	0.0604	0.0672	0.0595	0.0430	0.0417	0.0307	0.0463
0.000	0.200	0.0609	0.0564	0.0688	0.0588	0.0422	0.0407	0.0336	0.0456
0.010	0.001	0.1193	0.0988	0.0826	0.0764	0.0908	0.0874	0.0713	0.0715
0.010	0.010	0.0929	0.0782	0.0705	0.0641	0.0697	0.0679	0.0487	0.0605
0.010	0.050	0.0759	0.0678	0.0710	0.0611	0.0551	0.0526	0.0411	0.0530
0.010	**0.100**	**0.0552**	**0.0517**	**0.0626**	**0.0596**	**0.0367**	**0.0364**	**0.0268**	**0.0434**
0.010	0.200	0.0596	0.0557	0.0724	0.0593	0.0398	0.0390	0.0337	0.0459
0.050	0.001	0.1183	0.0962	0.0832	0.0721	0.0910	0.0854	0.0717	0.0710
0.050	0.010	0.0803	0.0680	0.0644	0.0631	0.0563	0.0551	0.0400	0.0515
0.050	0.050	0.0962	0.0901	0.0858	0.0795	0.0758	0.0736	0.0412	0.0775
0.050	0.100	0.0649	0.0600	0.0692	0.0647	0.0451	0.0433	0.0385	0.0481
0.050	0.200	0.1243	0.1266	0.1191	0.0967	0.1069	0.1050	0.0453	0.0964
0.100	0.001	0.1215	0.1003	0.0833	0.0742	0.0928	0.0883	0.0724	0.0725
0.100	0.010	0.0827	0.0705	0.0632	0.0636	0.0586	0.0575	0.0446	0.0532
0.100	0.050	0.0664	0.0585	0.0680	0.0602	0.0454	0.0441	0.0313	0.0474
0.100	0.100	0.0703	0.0651	0.0736	0.0566	0.0511	0.0487	0.0420	0.0506
0.100	0.200	0.0582	0.0549	0.0640	0.0535	0.0380	0.0373	0.0304	0.0433
0.200	0.001	0.1144	0.0935	0.0779	0.0723	0.0859	0.0818	0.0720	0.0663
0.200	0.010	0.0820	0.0701	0.0596	0.0633	0.0597	0.0579	0.0456	0.0513
0.200	0.050	0.0746	0.0688	0.0655	0.0624	0.0519	0.0506	0.0417	0.0514
0.200	0.100	0.0672	0.0590	0.0735	0.0602	0.0458	0.0446	0.0343	0.0483
0.200	0.200	0.0582	0.0577	0.0662	0.0568	0.0386	0.0375	0.0336	0.0456
0.500	0.001	0.1186	0.1004	0.0802	0.0758	0.0903	0.0872	0.0829	0.0714
0.500	0.010	0.0791	0.0681	0.0642	0.0605	0.0533	0.0527	0.0398	0.0508
0.500	0.050	0.0648	0.0649	0.0631	0.0630	0.0432	0.0426	0.0301	0.0477
0.500	0.100	0.0640	0.0575	0.0693	0.0566	0.0414	0.0412	0.0288	0.0472
0.500	0.200	0.1383	0.1220	0.1297	0.1067	0.1125	0.1114	0.0494	0.1102
0.800	0.001	0.1194	0.0953	0.0831	0.0733	0.0931	0.0879	0.0825	0.0685
0.800	0.010	0.0827	0.0738	0.0712	0.0601	0.0579	0.0567	0.0428	0.0579
0.800	0.050	0.0808	0.0685	0.0678	0.0655	0.0572	0.0566	0.0423	0.0532
0.800	0.100	0.0648	0.0625	0.0671	0.0591	0.0434	0.0419	0.0321	0.0479
0.800	0.200	0.0558	0.0473	0.0642	0.0546	0.0361	0.0342	0.0317	0.0419
1.000	0.001	0.1237	0.0998	0.0856	0.0785	0.0941	0.0926	0.0792	0.0741
1.000	0.010	0.0787	0.0683	0.0634	0.0623	0.0547	0.0536	0.0411	0.0515
1.000	0.050	0.0661	0.0603	0.0654	0.0644	0.0443	0.0433	0.0319	0.0488
1.000	0.100	0.1196	0.1175	0.1057	0.0858	0.0918	0.0906	0.0426	0.0885
1.000	0.200	0.0558	0.0525	0.0669	0.0609	0.0364	0.0340	0.0309	0.0438

backpropagation was employed to approximate derivatives with high accuracy. By computing the norm of the residual from the governing equations and including it as a loss term, the physics of the system can be enforced.

As in Chen and Zhang (2019), the features with the spatiotemporal coordinates are concatenated before passing them to the MLP. In addition, the spatiotemporal features and relative coordinates are concatenated with each of the hidden layers of the MLP to maintain the information at the following layers. An infinitely differentiable Softplus activation function is incorporated into the MLP. Specifically, the MLP predicts the temperature, pressure, and velocity components at the input spatiotemporal coordinate provided context from the feature extractor and the spatiotemporal coordinates. The predicted clip is then obtained by evaluating the MLP for every computational grid point of this clip using the appropriate feature vector. That is, given a spatiotemporal grid of feature vectors resulting from propagating the input through the U-net, the MLP predicts the solution variables point-by-point resulting in the output clip. Finally, the entire framework can be used for predictions over longer periods by evaluating the network recursively. In particular, the PI-DNN is evaluated recursively by using the predicted clip as input to the PI-DNN for the next prediction step. This process begins by propagating an input clip through the PI-DNN to obtain a predicted clip, which is considered as an input to the second model evaluation that yields a predicted clip that is further in the past.

The combination of a feature extractor with a subsequent task-performing MLP has been established in the deep

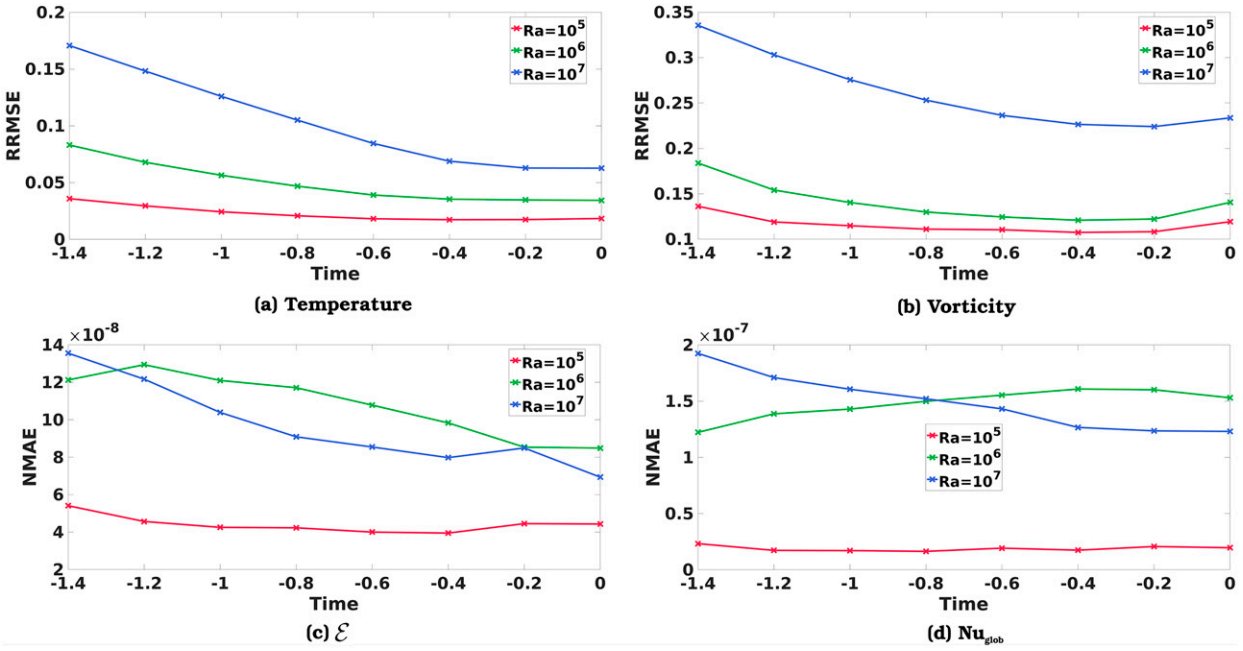


FIG. 3. Evolution of error metrics. The evolution of the RRMSE of the predicted (a) temperature and (b) vorticity fields, and the NMAE of the predicted (c) $\mathcal{E}$ and (d) Nu_{glob}. The error metrics are presented for Ra of 10^5, 10^6, and 10^7, as indicated in the legend.

learning literature (e.g., Chen and Zhang 2019; Saito et al. 2019). This approach was further applied for physics-informed superresolution in the context of data compression (Jiang et al. 2020). Although similarities exist between the present network and that of Jiang et al. (2020), the objectives are different, resulting in important differences, including the dimensions of the input and output fields and their time stamps. In Jiang et al. (2020), the training and validation datasets are randomly sampled from the same set of solution trajectories, allowing data to be leaked between the datasets. Here, the input and output to the network have the same dimensions, and even though the MLP predicts the past states of the system, the PDE loss is calculated on the forward-in-time evolution of the states. Moreover, Jiang et al. (2020) evaluate their loss function at a finite number of points, whereas in the present work, only the PDE loss is computed for a finite number of points due to limitations in the graphical processing unit (GPU) memory as further discussed in the following section. Most importantly, the primary objective of our study consists of providing a backward-in-time prediction of the system state for which no backward-in-time operator can be derived. However, superresolution in the context of data compression, tackled by Jiang et al. (2020), has been extensively investigated in the literature.

c. *Loss functions*

The proposed PI-DNN is trained end-to-end by minimizing a weighted sum of the regression loss ($\mathcal{L}_{reg}$) and the PDE loss ($\mathcal{L}_{PDE}$). The $\mathcal{L}_{reg}$ determines how well the neural network can fit the data, whereas $\mathcal{L}_{PDE}$ determines how well the predicted solution satisfies the governing equations. Specifically, the total loss ($\mathcal{L}$) is calculated as

$$\mathcal{L} = \mathcal{L}_{reg} + \gamma \mathcal{L}_{PDE}, \quad \text{and} \tag{8}$$

$$= \|\mathbf{y}_{true} - \hat{\mathbf{y}}\|_p + \gamma \|\Gamma \hat{\mathbf{y}} - f\|_p, \tag{9}$$

where $\|\cdot\|_p$ is the p norm, $\mathbf{y}_{true} = \{\Theta, p, u, v\}$ is the true system state, $\hat{\mathbf{y}}$ is the predicted system state, γ is a positive constant scaling the PDE loss, and $\Gamma\hat{\mathbf{y}} - f$ is the residual of the system of governing equations [Eqs. (1)–(3)], where Γ is the differential operator and f is the source term. The $\mathcal{L}_{PDE}$ is evaluated by calculating the partial derivatives involved in the residual of the governing equations using backpropagation.

During training, $\mathcal{L}_{reg}$ is calculated over the predicted clips, which are represented using a finite number of snapshots of spatial crops of the computational domain. By contrast, $\mathcal{L}_{PDE}$ is calculated over 2048 randomly selected points from the computational mesh. For each point, a computational graph is used to examine the derivatives involved in the PDE, which allows the evaluation of how well the predicted solution at a given point satisfies the PDE. A finite number of points are selected for calculating the PDE loss owing to memory limitations in the GPU, where $\mathcal{L}_{PDE}$ requires a copy of the computational graph for each point at which $\mathcal{L}_{PDE}$ is evaluated. It is worth pointing out that using random points from the spatial crops rather than the whole domain to verify the PDE loss is meaningful because the prediction must satisfy the governing equation at arbitrary spatiotemporal coordinates. Since over the training period, a sufficient number of arbitrary samples cover a wide variety of spatial areas, the PDE loss is statistically significant. In the current setup, the losses do not include a penalty on boundary information. Adding a loss term on the boundary conditions is fairly straightforward (Raissi et al. 2019)

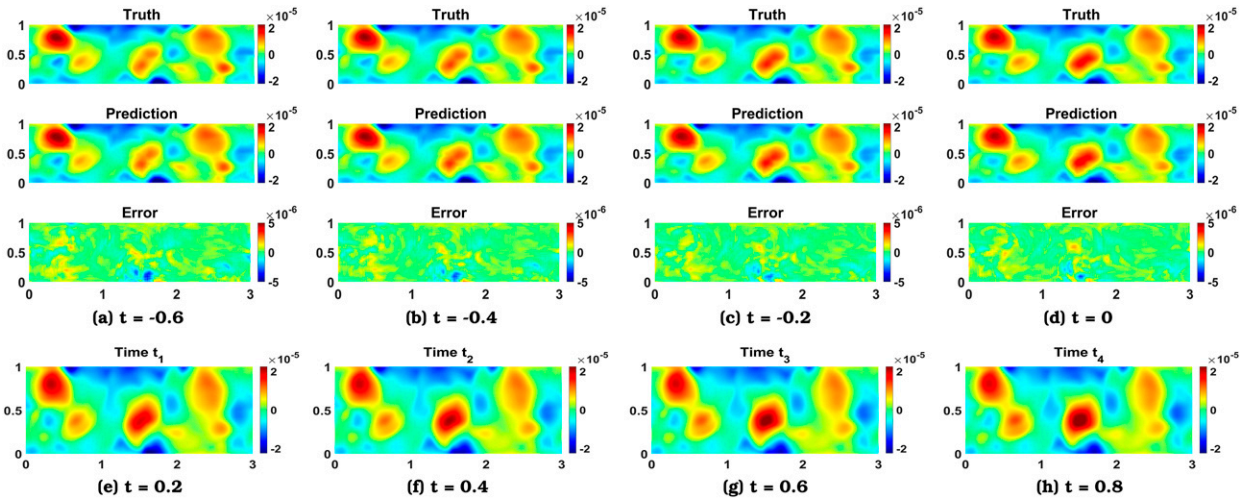


FIG. 4. Qualitative results. (a)–(d) Snapshots of the true, predicted, and error pressure distribution at the indicated time steps, shown from top to bottom. (e)–(h) Snapshots of the input pressure fields at the specified time steps.

and might help reduce errors near the boundaries, however, to achieve such improvements the loss terms must be suitably scaled leading to additional complexities during training.

d. Error metrics

The performance of the trained model is evaluated by calculating error metrics that determine the discrepancy between the predicted and the true solution fields. Consequently, we use the relative root-mean-squared error (RRMSE), a measure of the normalized ℓ_2-error between the predicted and reference solutions, defined as

$$\mathrm{RRMSE}_g = \frac{\|g(x, y; t) - \hat{g}(x, y; t)\|_2}{\|g(x, y; t)\|_2}, \tag{10}$$

where g and $\hat{g}$ are an arbitrary reference variable its predicted counterpart, respectively. The RRMSE of each solution variable is calculated independently and is averaged over Ω, the period of the predicted clip, and all the randomly selected assessment clips. In addition, the RRMSE of the predicted temperature fields at the two immediate previous frames, represented by RRMSE_{T_0} and $\mathrm{RRMSE}_{T_{-1}}$, are reported. Finally, the training and validation losses are also presented. The RRMSE of the state variables is an approximate normalized energy norm. Therefore, it is an efficient metric to evaluate the performance of numerical models (Moin 2010; Mao et al. 2020), further justifying its application in the present context.

For a comprehensive analysis and illustrate the performance of the proposed framework, errors associated with physically derived integral quantities are calculated. Consequently, the normalized ℓ_1 errors (NMAEs) of the mathematical energy $\mathcal{E}$ and the global Nusselt number (Nu) are calculated using their definitions retrieved from (Howard 1963, 1972; Doering and Constantin 1996, 2001; Mhina et al. 2021). The NMAE follows from the definition of the RRMSE, where the ℓ_2 norms are substituted by the ℓ_1 norm as follows:

$$\mathrm{NMAE}_g = \frac{\|g(x, y; t) - \hat{g}(x, y; t)\|_1}{\|g(x, y; t)\|_1}. \tag{11}$$

The mathematical energy $\mathcal{E}$ is defined as

$$2\mathcal{E}(t) = \frac{1}{\mathrm{Pr}}\int_\Omega |\mathbf{u}|^2 d\Omega + \int_\Omega (\Theta - \Theta_0) d\Omega, \tag{12}$$

where Θ_0 is the initial temperature profile. Finally, $\mathrm{Nu}_{\mathrm{glob}}$ is given as follows:

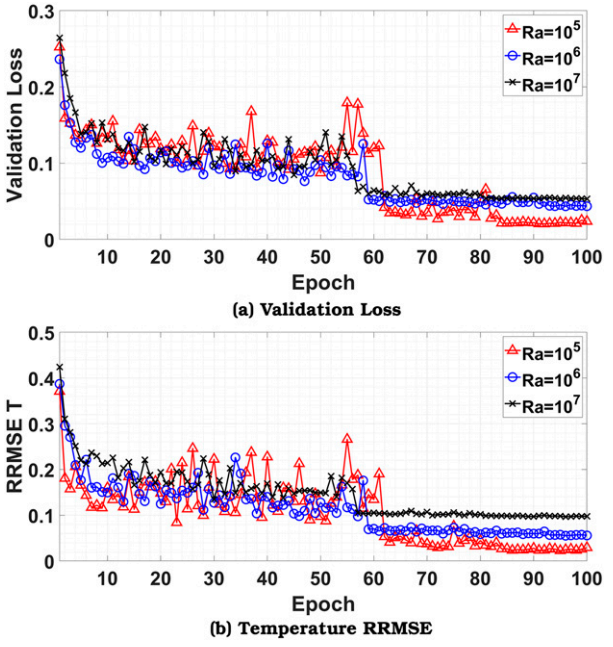


FIG. 5. Training curves. Plots of the (a) validation loss and (b) temperature RRMSE for the best PI-DNN trained on Rayleigh–Bénard flows at different values of Ra. The backward-in-time prediction becomes more challenging as Ra increases.

TABLE 2. Generalizability of the PI-DNN to different values of Ra. The PI-DNN is trained using Rayleigh–Bénard flows with Ra = 10^6. The performance of the PI-DNN is evaluated on flows with different Ra. The results corresponding to the Ra number trained on are highlighted in bold.

Ra	RRMSE_T	RRMSE_p	RRMSE_u	RRMSE_v	RRMSE_{T_0}	$\text{RRMSE}_{T_{-1}}$
1×10^5	0.0682	0.0911	0.0836	0.0523	0.0359	0.0352
7×10^5	0.0381	0.0399	0.0448	0.0592	0.0271	0.0268
8×10^5	0.0445	0.0484	0.0468	0.0631	0.0304	0.0296
9×10^5	0.0622	0.0602	0.0657	0.0712	0.0384	0.0392
1×10^6	**0.0552**	**0.0517**	**0.0626**	**0.0596**	**0.0367**	**0.0364**
1.1×10^6	0.0560	0.0508	0.0574	0.0734	0.0381	0.0383
1.2×10^6	0.0727	0.0619	0.0666	0.0792	0.0453	0.0459
1.5×10^6	0.0750	0.0676	0.0705	0.0828	0.0474	0.0480
2×10^6	0.1398	0.1238	0.1215	0.1671	0.0783	0.0814

$$\text{Nu}_{\text{glob}}(t) = 1 + \int_{\Omega} \Theta \mathbf{u} \cdot \mathbf{e}_y d\Omega. \tag{13}$$

The NMAE of $\mathcal{E}$ and Nu_{glob} were computed for 1000 different predictions, and their average was reported in the study.

The predicted clip has the exact dimensions as the input clip, implying that the backward-in-time prediction comprises n_t snapshots of the past states. The RRMSEs for the two immediate previous snapshots are presented to help evaluate the performance of the model at improving the estimate of the initial conditions.

e. Training setup

Each model was trained for 100 epochs, where each epoch involves 3000 randomly selected clips. Unless specified otherwise, both the input and output clips used for training and validation comprise $n_t = 8$ frames, each with a spatial dimension of $(n_{x,c}, n_{y,c}) = (128 \times 128)$ arbitrarily cropped from the domain of magnitude $[n_x, n_y] = [384, 128]$. The PI-DNN was evaluated on inputs covering the entire spatial domain with spatial dimensions of $(n_x, n_y) = (384, 128)$. The crops include the top and bottom boundaries and might overlap across arbitrary samples corresponding to different epochs. Although the random samples do not cross the periodic boundaries, the horizontal periodicity allows taking crops across the periodic boundary. The loss function was minimized using the stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 10^{-4} (Sutskever et al. 2013). After testing the ℓ_1, ℓ_2, and Huber norms for the loss terms, the ℓ_1 norm was adopted for both $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{PDE}}$ losses because it results in models that achieve the lowest RRMSE. A parameter search was conducted over the learning rate (LR) and γ; these findings are discussed in the following section. The LR was reduced on plateau by a factor of 0.1 with a patience of 10 epochs. Unless specified otherwise, models were trained using sr = 2 (i.e., $\delta t = 0.2$ between consecutive time steps) and st = 8.

4. Experiments and results

Numerical experiments were first conducted to test the proposed methodology for various Ra values to assess the ability of the PI-DNN to predict past solution trajectories. Next, we investigate the generalizability of the PI-DNN to flows corresponding to flows with Ra values different from the one adopted for training. The sensitivity of the network to sr and st was then evaluated for several conditions. We emphasize here that the results presented in the following sections correspond to training and validation of the PI-DNN performed on a spatial crop of the domain, and evaluation is performed on solution trajectories of the entire spatial domain.

a. Learning solution trajectories

The ability of the neural network to predict the states of the Rayleigh–Bénard system was assessed by first training different PI-DNNs for each of the datasets described in section 2b. Table 1 lists the RRMSE of each solution variable and those

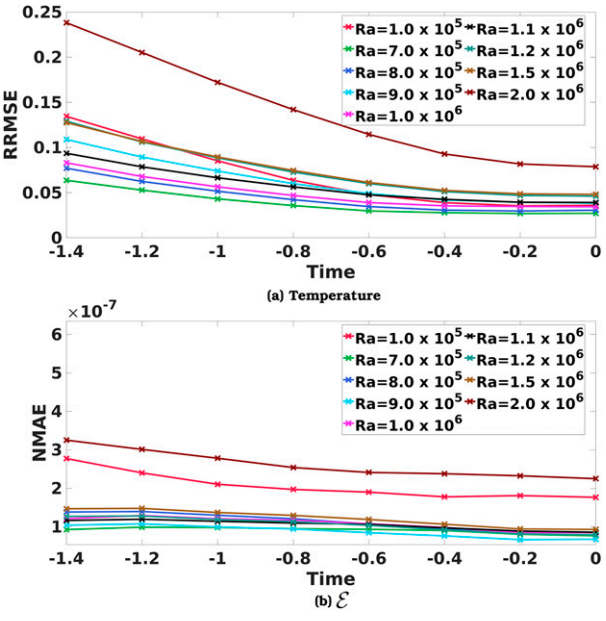


FIG. 6. Evolution of error metrics. The evolution of (a) the RRMSE of the predicted temperature and (b) the NMAE of $\mathcal{E}$. The error metrics are presented for a PI-DNN trained using a dataset of solution trajectories of the Rayleigh–Bénard convection with Ra = 10^6 and evaluated on solution trajectories at different Ra values, as indicated by the legend.

TABLE 3. Sensitivity of the PI-DNN to sr for a constant period of backward-in-time prediction. Average error metrics for the best trained PI-DNN for each n_t. Results correspond to a constant physical prediction time of 1.6 at different combinations of n_t and sr.

n_t	γ_{opt}	LR_{opt}	$RRMSE_T$	$RRMSE_p$	$RRMSE_u$	$RRMSE_v$	$RRMSE_{T_0}$	$RRMSE_{T_{-1}}$	Training loss	Validation loss
32	1.0	0.05	0.0804	0.0784	0.0794	0.0773	0.0684	0.0668	0.0282	0.0549
16	1.0	0.10	0.0652	0.0850	0.0744	0.0762	0.0521	0.0506	0.0310	0.0523
8	0.1	0.20	0.0663	0.0735	0.0786	0.0707	0.0406	0.0402	0.0331	0.0496
4	0.1	0.20	0.0799	0.0775	0.0633	0.0692	0.0399	0.0615	0.0379	0.0505

of the temperature solutions for the two immediate snapshots prior to the input as well as the training and validation losses. The table presents the outcomes of evaluating a neural network trained on a Rayleigh–Bénard flow with $Ra = 10^6$, using st = 8 and sr = 2. The results indicate that the best (LR, γ) pair is (0.1, 0.01) and the corresponding results are highlighted in Table 1. The resulting prediction of the trained network has an RRMSE of approximately 6% averaged over the eight prediction time steps. The temperature field for the immediate frame prior to the input is predicted with an RRMSE of approximately 4%. The results further suggest that for this experiment, a small γ value achieves the best trained network; however, the RRMSEs corresponding to nearby values of γ are comparable, suggesting that the PI-DNN's training is slightly sensitive to γ. In the limit of large γ, the network is trained to minimize the PDE loss only, similar to conventional PINNs (Raissi et al. 2019). This, however, may not achieve the desired solution because the predicted field may not be consistent with the reference solution, as seen during training. In the limit of γ approaching zero, the network is trained to minimize the regression loss only; the predicted fields thus capture the large scales more accurately than the fine scales. This suggests that calibrating γ generally improves the performance of the trained model, as further discussed in light of the experiments presented below.

The table confirms a strong relationship between γ and the resulting RRMSE; differing from the observation of Jiang et al. (2020). In particular, Jiang et al. (2020) tackled the issue of superresolution in the context of data compression, where the training and validation datasets corresponded to the same solution trajectories. This enabled them to realize spatial and temporal compression ratios that are not theoretically possible through downscaling, which is the process of obtaining high-resolution solution fields using low-resolution information (Azouani et al. 2013; Farhat et al. 2015). Data compression would then require a larger weight on the reconstruction loss as opposed to the PDE loss, which was numerically validated in their work. However, the training and validation datasets in our study correspond to unique solution trajectories because the primary objective is to identify a mapping that could be employed as a surrogate to the intractable backward-in-time operator. This is necessary for the problem at hand because the backward-in-time operator cannot be derived for this dissipative system; hence, the forward-in-time governing equations are employed to assess the predictions of the past states, in their correct chronological order. Enforcing the forward-in-time dynamic avoids making modifications to the viscosity term, especially in terms of taking nonpositive values. Accordingly, the PDE loss is an important term to obtain accurate predictions, as affirmed by the results in Table 1. In a practical setting, such as large-scale environmental models, the PDE loss might be challenging to implement. However, invariants and/or conservation laws could be used instead, similarly to Ling et al. (2016), Mao et al. (2020), and Bode et al. (2021).

Figure 3 shows the evolution of the RRMSE of the predicted temperature and vorticity fields as well as the NMAE of $\mathcal{E}$ and Nu_{glob} for a PI-DNN trained and evaluated on datasets corresponding to different Ra values. The figure indicates that the RRMSEs of temperature and vorticity are lower for smaller Ra, which is expected because the system becomes more chaotic as Ra increases. The RRMSE of the vorticity is greater than that of the temperature because the network was trained with temperature data, while vorticity is a derived quantity, which will inherit the errors of both velocity components. In addition, the figure indicates that the further the prediction is backward in time, the higher are the prediction errors. Moreover, the rate of increase of errors increases with higher Ra, implying that the lead time corresponding to low prediction errors becomes shorter with higher Ra. Both $\mathcal{E}$ and Nu_{glob} suggest a different behavior, with no discernible pattern. The errors corresponding to $\mathcal{E}$ and Nu_{glob} are negligible where both are less than 10^{-6}. These results suggest that the total network prediction accurately reflects the flow dynamics, with a small difference in the fine scales resulting in higher reconstruction errors but minimal errors for integral quantities. This behavior resonates with the numerical methods literature, where using different metrics yields the same conclusions (Farhat et al. 2015; Jiang et al. 2020). The discussion on the vorticity and Nu_{glob} will be omitted in subsequent sections because it is similar to those on the predicted temperature and $\mathcal{E}$, respectively.

TABLE 4. As in Table 3, but for PI-DNNs trained with fixed training parameters [scaled based on batch size (Goyal et al. 2017)].

n_t	γ	LR	$RRMSE_T$	$RRMSE_p$	$RRMSE_u$	$RRMSE_v$	$RRMSE_{T_0}$	$RRMSE_{T_{-1}}$	Training loss	Validation loss
32	0.1	0.05	0.0952	0.0996	0.0876	0.0879	0.0843	0.0831	0.0340	0.0642
16	0.1	0.10	0.0778	0.0755	0.0733	0.0727	0.0639	0.0632	0.0320	0.0520
8	0.1	0.20	0.0663	0.0735	0.0786	0.0707	0.0406	0.0402	0.0331	0.0496
4	0.1	0.40	0.9991	0.9992	1.0000	0.9992	0.9992	0.9992	0.7443	0.7414

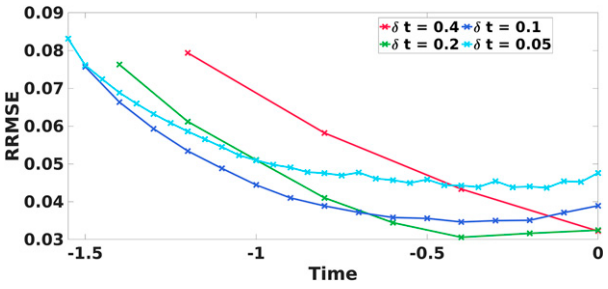


FIG. 7. Evolution of error metrics. The evolution of the RRMSE of the predicted temperature. All PI-DNNs are trained to predict at a constant prediction period with different temporal frequencies of the input snapshots. The results are shown here for varying n_t, while keeping the prediction length fixed.

Figure 4 illustrates the evolution, in chronological order, of the temperature field corresponding to the reference field, predicted field, and their residual fields along with the temperature fields that were used to evaluate the PI-DNN. Although the PI-DNN input and output clips consisted of eight snapshots, only four snapshots of the PI-DNN's input and output clips, centered over their combined time period, are presented for clarity. The figure outlines small-scale structures in the domain and a qualitative comparison of the prediction of the network against the true field. Although the large-scale features are often well predicted with no noticeable difference relative to the reference solution fields, the results suggest that the error increases the further the prediction is in the past. The figure shows that the predicted fields capture the formation of a plume that later detaches near the middle of the top boundary with an error approximately an order of magnitude less than the range of values in the domain. The predicted fields also regulate the magnitude of the field across the domain; that is, the center high-pressure blob was predicted to have a smaller pressure value, with an increasing amplitude moving forward in time. Finally, the comparison of the various structures indicates that the input and output clips have different large- and small-scale features, which indicates that the network is genuinely performing a backward-in-time prediction. The backward-in-time prediction can predict the correct position and size of the fields with an error at least an order of magnitude less than the background signal. The supplemental material illustrates the other solution variables for the same experiment.

The training parameter search for the Rayleigh–Bénard flows at Ra of 10^5 and 10^7 is shown in the supplemental material for brevity. Specifically, for Ra = 10^5, the best LR and γ are 0.2 and 1.0, respectively, yielding a maximum temperature RRMSE of 0.0314. By contrast, for Ra = 10^7 and sr = 1, the best LR and γ are 0.1 and 0.5, respectively, yielding a maximum temperature RRMSE of 0.0967, indicating that the resulting backward-in-time predictions become less accurate as Ra increases. Prediction errors may be decreased using a more complex DNN and/or a larger training dataset. Finally, the validation loss and the RRMSE corresponding to the predicted temperature fields are demonstrated in Fig. 5. The results indicate that, for a fixed network width and depth, the RRMSE of the predicted fields increases for larger Ra; this suggests that it is more challenging to predict the system states at previous times when Ra increases.

b. *Generalizability to different* Ra

In a practical setting, the value of Ra may not be accurately known a priori. This requires the trained PI-DNN to be robust to flows with slightly perturbed Ra. To evaluate our PI-DNN's performance in such a setting, we used the PI-DNN trained for Ra = 10^6 to conduct backward-in-time predictions on Rayleigh–Bénard flows at Ra of {10^5, 7×10^5, 8×10^5, 9×10^5, 1.1×10^6, 1.2×10^6, 1.5×10^6, 2×10^6}. A set of 3000 arbitrarily sampled clips for each Ra was employed to investigate this problem. The predictions were conducted over the entire spatial domain and eight snapshots in time, and the error metrics were averaged over the random samples.

Table 2 lists the average error metrics over these arbitrarily selected clips for each tested Ra. For smaller Ra, the trained PI-DNN could maintain the accuracy obtained on the original validation datasets. Specifically, for Ra 10%, 20%, and 30% below the Ra on which the PI-DNN was trained, the temperature RRMSEs are 6.2%, 4.45%, and 3.81%, respectively. The temperature RRMSE is 6.82% when Ra is 10 times smaller than that on which the PI-DNN was trained. This implies that the PI-DNN can generalize to Ra up to 10 times smaller than the trained one, with an additional error of less than 2%. The predicted pressure and velocity fields also show similar outcomes.

For higher Ra, the performance of the PI-DNN is different than that for lower Ra. In particular, for a Ra 10% greater than the one trained on, the PI-DNN yields an RRMSE of 5.6%. At Ra of 1.2×10^6 and 1.5×10^6, the RRMSE of the temperature increases to 7.27% and 7.5%, respectively, implying a satisfactory performance at moderately high values of Ra. By contrast, when Ra is doubled, the network fails to yield accurate backward-in-time predictions, with the RRMSE of the predicted

TABLE 5. Sensitivity of the PI-DNN to sr at constant n_t. Comparison of the PI-DNNs trained with the best parameters using different temporal subsampling rates sr and a fixed number of predicted frames at $n_t = 8$. The RRMSE of all the solution variables increases linearly with sr. This observation indicates that slower dynamics are simpler to predict than faster dynamics.

sr	γ_{opt}	LR_{opt}	$RRMSE_T$	$RRMSE_p$	$RRMSE_u$	$RRMSE_v$	$RRMSE_{T_0}$	$RRMSE_{T_{-1}}$	Training loss	Validation loss
1	0.05	0.2	0.0172	0.0453	0.0342	0.0323	0.0157	0.0149	0.0148	0.0223
2	0.05	0.2	0.0274	0.0513	0.0261	0.0375	0.0215	0.0208	0.0196	0.0241
3	0.3	0.2	0.0451	0.0467	0.0628	0.0577	0.0314	0.0292	0.0287	0.0359
4	0.1	0.2	0.0628	0.0677	0.0758	0.0730	0.0364	0.0366	0.0333	0.0473
5	0.3	0.2	0.0905	0.0983	0.0859	0.0909	0.0499	0.0505	0.0416	0.0633
6	0.5	0.2	0.1192	0.1168	0.1109	0.1098	0.0589	0.0646	0.0420	0.0804

TABLE 6. As in Table 5, but for constant training parameters.

sr	γ	LR	RRMSE$_T$	RRMSE$_p$	RRMSE$_u$	RRMSE$_v$	RRMSE$_{T_0}$	RRMSE$_{T_{-1}}$	Training loss	Validation loss
1	0.1	0.2	0.0187	0.0440	0.0321	0.0312	0.0175	0.0164	0.0135	0.0220
2	0.1	0.2	0.0306	0.0615	0.0374	0.0366	0.0252	0.0234	0.0226	0.0283
3	0.1	0.2	0.0613	0.0928	0.0691	0.0920	0.0436	0.0415	0.0425	0.0565
4	0.1	0.2	0.0628	0.0677	0.0758	0.0730	0.0364	0.0366	0.0333	0.0473
5	0.1	0.2	0.1016	0.0958	0.0930	0.0916	0.0650	0.0646	0.0499	0.0665
6	0.1	0.2	0.1207	0.1254	0.1119	0.1145	0.0643	0.0694	0.0420	0.0827

temperature fields reaching approximately 13.98%. The predicted pressure and velocity fields show similar variations in RRMSEs as those of the temperature. These results can be explained by the fact that finer features appear for flows with higher Ra. Because the network was not exposed to these features during training, it cannot predict them accurately.

Figure 6 shows the evolution of the RRMSE of the predicted temperature fields and the NMAE of $\mathcal{E}$. The further the prediction is back in time, the higher is the RRMSE of the prediction. Similar values of the RRMSE are obtained for all Ra values below or equal to 1.5×10^6. However, the RRMSE is noticeably higher for Ra $= 2 \times 10^6$. Moreover, the rate of increase of errors is also comparable across all Ra values considered. The figure also shows that the NMAE of $\mathcal{E}$ is of the order 10^{-7} for all Ra values considered, which implies that the network can predict the total energy in the system with sufficient accuracy.

c. *Sensitivity to sr for a fixed prediction period*

In this section, we investigate the sensitivity of the PI-DNN to the sr while keeping the prediction period fixed. Specifically, the experiment was conducted by varying the (sr, n_t) pairs to maintain a prediction period of 1.6 and then training a network for each chosen pair. This corresponds to observing the state of the system at different temporal frequencies while maintaining the prediction period constant. To achieve a constant total physical period of the clip, the number of snapshots is varied. The dataset corresponding to Ra $= 10^6$ with $\delta t_o =$ 0.05 was used. Finally, the stride was set to st $= n_t$ such that no snapshots overlap between the input and predicted clips.

Tables 3 and 4 outline the error metrics, training, and validation loss for each trained PI-DNN with the indicated γ and LR. Table 3 presents the error metrics corresponding to the best trained PI-DNN according to a grid search for the best (γ, LR) pair. The results indicate that for the best training parameters and for $n_t = 32$ and 4, the PI-DNN predicts the temperature fields with an average RRMSE of approximately 8%. By contrast, for $n_t = 8$ and 16, the temperature RRMSE is approximately 6.52% and 6.63%, respectively. This indicates that there is only a slight difference in the PI-DNN's performance when observing a small number of frames at a coarse temporal period and when observing several frames at a fine temporal period.

Table 4 outlines the error metrics and losses for a fixed (γ, LR) pair. Because the batch size decreases with increasing n_t (due to memory restrictions), a linear scaling was used for the LR (Goyal et al. 2017). The results signify that, for constant training parameters, the lowest RRMSE corresponds to the PI-DNN with $n_t = 8$ separated by $\delta t = 0.2$. Similar to the conclusions drawn from Table 3, for $n_t = 4$ and 32, the RRMSEs of all solution variables are slightly larger than those corresponding to both $n_t = 8$ and 16. For $n_t = 4$ and $\delta t = 0.4$, the model diverges, while for $n_t = 32$, the PI-DNN outputs a prediction with a temperature RRMSE of 9.52%. In comparison, for $n_t = 8$, the temperature RRMSE is the lowest, approximately 6.63%.

Figure 7 presents the RRMSE of the predicted temperature fields, providing more insight into the evolution of the errors over the backward-in-time prediction. The curves are translated along the time axis to show the immediate prediction at $t = 0$. The figure indicates that for all values of δt, the prediction errors increase when the prediction is further back in time. These results support the previous findings that the related errors are higher for $\delta t = 0.05$ and 0.4 than for $\delta t = 0.1$ and 0.2, suggesting that highly coarse or fine observations of the system state might degrade the performance of the PI-DNN. The figure also indicates that the errors increase almost linearly when $\delta t = 0.4$ and nonlinearly for the finer temporal resolution of the input clip.

d. *Sensitivity to sr at fixed n_t*

This section analyzes the impact of sr on the performance of the PI-DNN. The number of input and output frames is set to $n_t = 8$ with st $= 8$. This configuration, with a fixed n_t and a variable sr examines the effect of the temporal frequency of the input data on the backward-in-time prediction errors and is different from the previous experiment in terms of the backward-in-time prediction period. Here, we rely on the dataset corresponding to Ra $= 10^6$, with $\delta t_o = 0.1$, and vary sr by unit increments from 1 to 6.

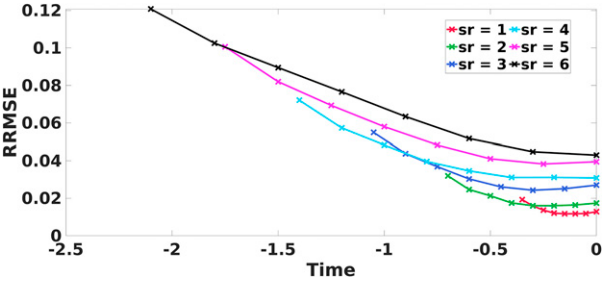


FIG. 8. Evolution of error metrics. The evolution of the RRMSE of the predicted temperature. All PI-DNNs are trained to predict at a constant prediction period with different temporal frequencies of the input snapshots. The results are presented here for a fixed n_t resulting in different prediction periods.

TABLE 7. Sensitivity of the PI-DNN to st. Comparison of the PI-DNNs trained with the best parameters and using different stride values, st. The RRMSE of the backward-in-time prediction decreases as the number of shared frames between the input and output clips increases.

st	γ_{opt}	LR_{opt}	$RRMSE_T$	$RRMSE_p$	$RRMSE_u$	$RRMSE_v$	$RRMSE_{T_0}$	$RRMSE_{T_{-1}}$	Training loss	Validation loss
2	0.3	0.1	0.0126	0.0125	0.0120	0.0190	0.0125	0.0103	0.0148	0.0095
4	0.1	0.1	0.0242	0.0275	0.0234	0.0279	0.0189	0.0160	0.0185	0.0178
5	0.5	0.1	0.0313	0.0335	0.0282	0.0355	0.0217	0.0192	0.0199	0.0220
6	0.5	0.2	0.0400	0.0419	0.0422	0.0443	0.0273	0.0248	0.0252	0.0288
7	0.1	0.2	0.0520	0.0526	0.0584	0.0573	0.0325	0.0311	0.0316	0.0378
8	0.5	0.2	0.0648	0.0737	0.0709	0.0686	0.0377	0.0370	0.0332	0.0480
9	0.1	0.2	0.0811	0.0867	0.0890	0.0811	0.0490	0.0498	0.0423	0.0582
10	0.05	0.2	0.1080	0.1107	0.1020	0.0999	0.0743	0.0768	0.0489	0.0738

Tables 5 and 6 list the RRMSE of each solution variable, the RRMSE for the first two backward-in-time temperature predictions, respectively denoted by T_0 and T_{-1}, and the training and validation losses. Table 5 outlines the aforementioned error metrics for the best trained PI-DNN according to a grid search on different (γ, LR) pairs. The results indicate that the RRMSE of all the solution variables increases linearly with sr, confirming that it is easier to perform short-term predictions where the input snapshots are closely spaced in time. This suggests that the accuracy of backward-in-time prediction is directly proportional to δt, the temporal resolution of the input clip. The results further indicate that the input to the NN should be provided at a finer δt, which implies that more frequent observations of the system state are required to train the proposed PI-DNN efficiently. Table 6 lists the error metrics for PI-DNNs trained using a constant (γ, LR) pair. The conclusions drawn from Table 5 hold for constant training parameters, where the average error metrics are approximately linear with sr.

The evolution of the RRMSE of the predicted temperature fields over the backward-in-time prediction period is illustrated in Fig. 8. The figure shows the RRMSE at each predicted time step for different sr values, maintaining constant n_t. The curves are translated along the time axis such that the prediction extends from the time origin backward. As sr increases, the RRMSE increases; however, the prediction period is also larger. The RRMSE is usually lower for a lower sr value at a given time step. However, exceptions might arise at the final prediction time step. This agrees with the results of Tables 5 and 6.

e. *Sensitivity to st*

Although correlations in the data are generally considered to not carry much additional information (Homan and Gelman 2014), in this experiment, we investigate whether correlations in the training input–output pairs, in the form of overlapping snapshots, would improve the predictability of the network. We then examine if the network can output accurate predictions while skipping one or more frames. In particular, the network was trained for multiple values of st while fixing $n_t = 8$. An st value smaller than n_t implies that the input and output clips have $n_t - \text{st}$ frames in common. By contrast, an st value greater than n_t implies that $\text{st} - n_t$ frames are skipped between the input and output clips. All snapshots that were not shared with the input will be called "new" snapshots to differentiate them from those shared with the input clip when $\text{st} < n_t$.

Tables 7 and 8 list the error metrics for each solution variable, the RRMSE of the predicted temperature field for the immediate two steps backward, and training and validation losses. The two tables show the results corresponding to the PI-DNN trained with the best (γ, LR) pair in Table 7 and with a constant (γ, LR) training pair in Table 8. In both tables, for all solution variables, the RRMSE increases linearly with st for all $\text{st} \leq 6$. By contrast, for $\text{st} > 6$, the RRMSE increases quadratically with st. The results also indicate that the RRMSE decreases as the number of common frames between the input and output clips increases. For st = 2, the lowest RRMSEs are obtained; for example, the average RRMSE of the predicted temperature is less than 2%. However, for st = 10, the average RRMSE is above 10% for all variables, indicating that when multiple frames are skipped, predictability is rapidly lost. The results also imply that a single missing frame would result in only a 2% error increase compared to the case when backward prediction starts immediately before the input (i.e., no common frames).

TABLE 8. As in Table 7, but for fixed training parameters.

st	γ	LR	$RRMSE_T$	$RRMSE_p$	$RRMSE_u$	$RRMSE_v$	$RRMSE_{T_0}$	$RRMSE_{T_{-1}}$	Training loss	Validation loss
2	0.5	0.2	0.0159	0.0192	0.0155	0.0221	0.0167	0.0130	0.0126	0.0122
4	0.5	0.2	0.0244	0.0293	0.0323	0.0363	0.0186	0.0159	0.0198	0.0206
5	0.5	0.2	0.0329	0.0339	0.0332	0.0356	0.0220	0.0201	0.0200	0.0234
6	0.5	0.2	0.0400	0.0419	0.0422	0.0443	0.0273	0.0248	0.0252	0.0288
7	0.5	0.2	0.0545	0.0662	0.0682	0.0647	0.0339	0.0314	0.0321	0.0449
8	0.5	0.2	0.0648	0.0737	0.0709	0.0686	0.0377	0.0370	0.0332	0.0480
9	0.5	0.2	0.0818	0.0811	0.0848	0.0860	0.0491	0.0496	0.0396	0.0577
10	0.5	0.2	0.1125	0.1196	0.1009	0.1151	0.0779	0.0796	0.0576	0.0801

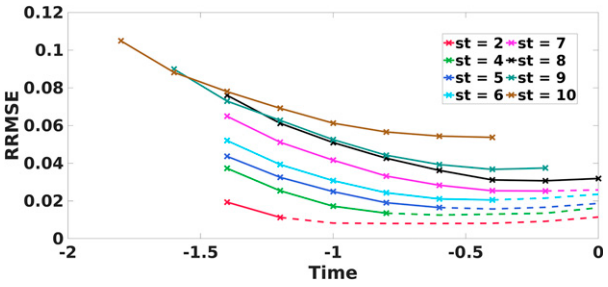


FIG. 9. Evolution of error metrics. The evolution of the RRMSE of the predicted temperature. All PI-DNNs were trained to predict at a constant prediction period with different stride values. The RRMSE of the shared snapshots between the input and output clips with a dashed line between the marks and the "new" ones with a solid line.

Figure 9 illustrates the behavior of the RRMSE of the predicted temperature field for different st values. For all values of st, the figure indicates that the RRMSE increases, but the prediction goes further back in time. For a given time step, the RRMSEs of the "new" snapshots are generally lower for smaller st, indicating that correlations in the input help better predict past states. When a lag of one snapshot is present between the input and output clips, the errors are comparable to the case when no snapshots are shared between the input and output clips. If more snapshots are missing, the PI-DNN fails to provide appropriately accurate estimates of the past states where the resulting prediction errors exceed 8%.

f. Sensitivity to longer prediction periods

In this last experiment, we evaluate the performance of the trained PI-DNN for providing extended backward-in-time predictions. For this purpose, the trained PI-DNN was evaluated recursively using the output of the previous evaluation as input to the subsequent evaluation. We used the dataset corresponding to Ra $= 10^6$, $\delta t = 0.2$, and st values of 2, 4 and 8. Note that $n_t = 8$ snapshots are predicted in each experiment. However, only st past snapshots are not overlapping with the snapshots in the input. In the current setup, each PI-DNN evaluation predicts n_t snapshots of the past state, where st output snapshots do not overlap with the input snapshots. That is, when the PI-DNN is recursively evaluated N_{eval} times, the network predicts a total of $N_{\text{eval}} \times$ st snapshots that do not appear in the original input clip.

Table 9 summarizes the performance metrics for different values of st and number of recursive model evaluations. Note that the product st $\times N_{\text{eval}}$ corresponds to the number of predicted frames. The results imply that when st $\times N_{\text{eval}} = 8$, st $= 4$, and $N_{\text{eval}} = 2$, the PI-DNN yields the lowest average RRMSE of 3.93% for the predicted temperature field. For st values of 2 and 8, the RRMSE of each solution variable is approximately 1% greater than those for st $= 4$, indicating that a recursive model evaluation is a valid solution for a short prediction time. When st $\times N_{\text{eval}}$ is 16 or 24, the PI-DNNs trained with st $= 4$ achieve the lowest error averaged over the solution variables. In particular, for the case of 16 predicted frames, the PI-DNN trained with st $= 4$ achieves a predicted temperature RRMSE of 12%, whereas the PI-DNNs trained with st $= 2$ and 8 yield a temperature RRMSE of 15.1% and 13.5%, respectively. As for the case of st $\times N_{\text{eval}} = 24$, the average RRMSE of the solution variables is the lowest for the case of st $= 4$ averaging at 15%, compared to 20% and 16% for st $= 2$ and 8, respectively. This implies that while correlations in the input–output pair enhance the accuracy of the backward-in-time predictions, too strong correlations or no correlation might adversely affect the accuracy of the backward-in-time predictions.

Finally, Fig. 10 shows the behavior of the RRMSE of predicted temperature and the NMAE of the predicted $\mathcal{E}$ over the past prediction period. From left to right, the subplots present the cases of st $\times N_{\text{eval}} = 8$, 16, and 24, where different experiments are indicated by the value of st. Note that the figure only shows the RRMSE of the "new" snapshots without showing the overlapping time steps. When st $\times N_{\text{eval}} = 8$, the RRMSE for the cases of st $= 2$ and 4 are initially lower than those for st $= 8$. The RRMSE, however, increases rapidly for smaller st values, resulting in the cases of st $= 2$ and 4 to have a larger RRMSE further in the past time steps. The RRMSE noticeably increases for predictions that are further back in time as the number of model evaluations increases. Over the whole integration period, the RRMSE increases for decreasing st values, which indicates that correlations between the input and output clips might not be valuable for accurate long backward-in-time predictions. By contrast, the NMAE of $\mathcal{E}$ achieves NMAE values of $\mathcal{O}(10^{-6})$ for all values of st $\times N_{\text{eval}}$.

TABLE 9. Recursive model evaluation. Error metrics corresponding to the backward-in-time prediction for recursively evaluating a trained PI-DNN for the indicated stride (st) and the number of evaluation times (N_{eval}). Note that the product st $\times N_{\text{eval}}$ corresponds to the number of frames predicted.

st $\times N_{\text{eval}}$	st	N_{eval}	RRMSE$_T$	RRMSE$_p$	RRMSE$_u$	RRMSE$_v$	RRMSE$_{T_0}$	RRMSE$_{T_{-1}}$
8	2	4	0.0442	0.0337	0.0271	0.0314	0.0112	0.0090
	4	2	0.0393	0.0377	0.0284	0.0305	0.0163	0.0134
	8	1	0.0495	0.0500	0.0500	0.0673	0.0343	0.0346
16	2	8	0.1514	0.1159	0.0764	0.0706	0.0111	0.0088
	4	4	0.1195	0.0864	0.0644	0.0583	0.0163	0.0134
	8	2	0.1350	0.1022	0.0830	0.0983	0.0342	0.0345
24	2	12	0.2706	0.2508	0.1477	0.1334	0.0102	0.0081
	4	6	0.2226	0.1571	0.1199	0.0978	0.0161	0.0132
	8	3	0.2183	0.1666	0.1236	0.1340	0.0339	0.0341

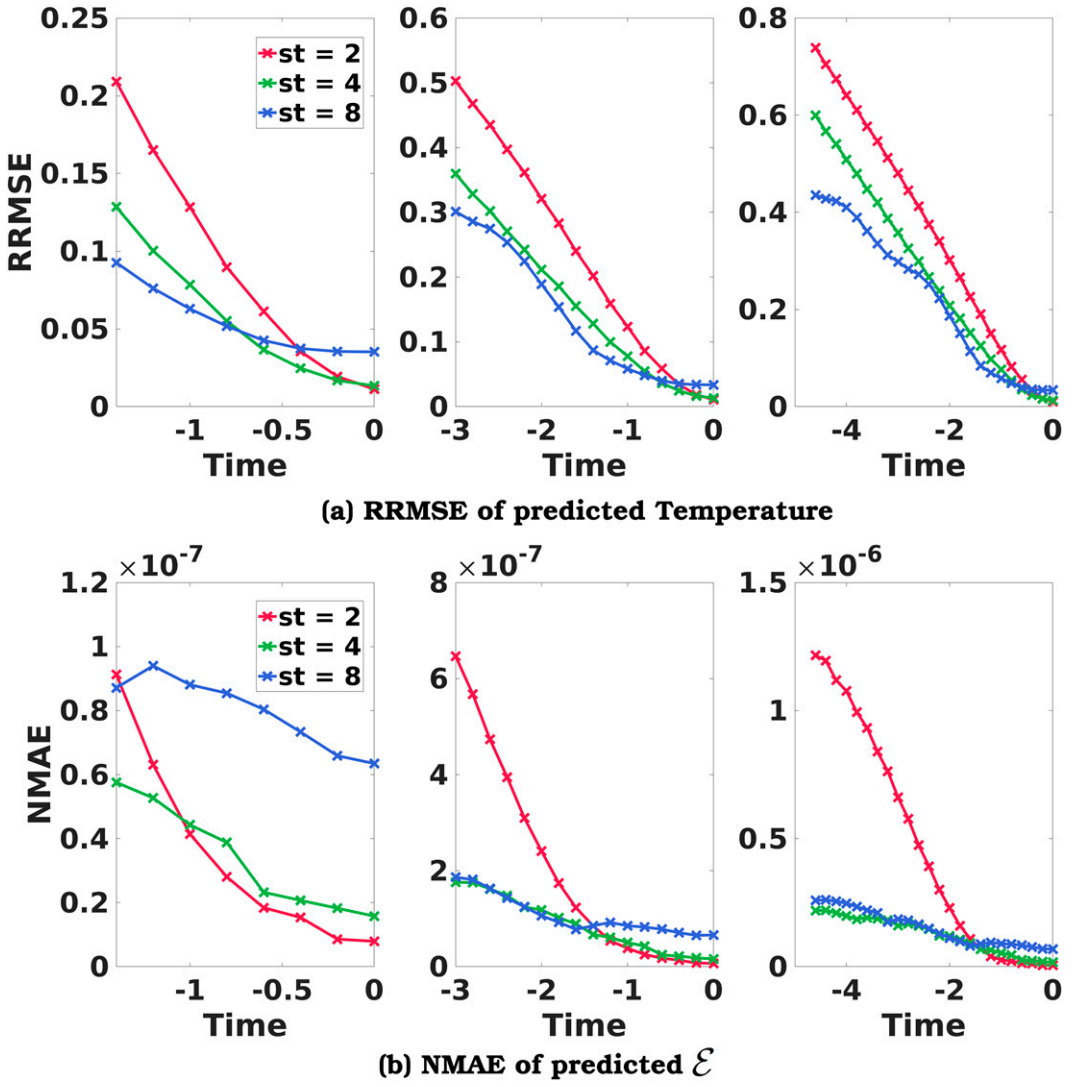


FIG. 10. Evolution of error metrics. The evolution of the RRMSE of the predicted temperature. All PI-DNNs are trained to predict a constant prediction period, which is achieved by recursively evaluating the PI-DNN. Each PI-DNN was trained with a different value of st.

For st $\times N_{\text{eval}} = 8$, the NMAE for the case of st = 8 is the largest. Nevertheless, the NMAE is negligible for all cases presented. As the number of model evaluations increases, the NMAE of $\mathcal{E}$ remains similar for the cases of st = 4 and 8. However, the NMAE corresponding to the case of st = 2 increases quickly for predictions further back in time.

5. Discussion and conclusions

We presented a new approach for backward-in-time predictions of a dissipative dynamical system by employing a PI-DNN and evaluated its performance by applying it to the 2D Rayleigh–Bénard convection. The network was trained to predict the previous system states over several time steps on the basis of the current evolution of the system state. As opposed to the existing backward-in-time integration techniques, the proposed approach does not disregard diffusion or assume a negative diffusion coefficient. Instead, the diffusion term is maintained at the same level as in the forward run to ensure that the predicted solutions are physically consistent. The proposed methodology was evaluated through several numerical experiments to demonstrate the efficiency of the DNN in predicting the past states of chaotic dissipative system. In particular, the framework was tested for learning the solution fields in the past for flows with increasing Rayleigh numbers (Ra) and for robustness against flows from perturbed Ra numbers. Several sensitivity experiments to different model parameters, such as the subsampling rate and stride, were further performed. Finally, the potential of predicting the system state further in the past by recursively evaluating the trained model was tested.

The performance of the model was evaluated using error metric that assess the similarity between the prediction and

reference fields, namely, the normalized ℓ_2 error of the solution variables and the normalized ℓ_1 error of integral quantities, such as the mathematical energy and global Nusselt number. Additional attention was given to the backward-in-time prediction of the first two frames just before the input to provide further insights into the network's performance for enhancing estimates of the initial conditions. The DNN provided backward-in-time predictions with approximately 3% error for flows at $\mathrm{Ra} = 10^5$. For a fixed network depth and width, the network achieved about 5% and 10% average RRMSE for flows at $\mathrm{Ra} = 10^6$ and 10^7, respectively. The optimum trained model maintained its accuracy for flows with an Ra 10 times smaller than the one employed in the training dataset, indicated by an RRMSE increase less than 1%. For higher Ra, however, the network RRMSE increased by 2.5% for a 50% increase in Ra, and accuracy was lost when Ra doubled.

For a fixed prediction period, the numerical experiment results suggest that both low and high subsampling rates would yield a lower accuracy than models trained with an intermediate subsampling rate. This indicates that observing the system state at a high or low frequency slightly impacts the model's prediction accuracy. In particular, observing the Rayleigh–Bénard system state at periods of 0.1 and 0.2 showed a steady 2%–3% enhancement in accuracy relative to the case when the state was observed every 0.05 and 0.4. By contrast, for a fixed number of predicted frames, the accuracy decreases linearly with an increasing subsampling rate, suggesting that the accuracy of predicting slow dynamics is higher than that of predicting fast dynamics. Finally, the sensitivity to stride indicates that correlations in the data help improve the accuracy of past predictions. The RRMSEs of the solution variables were also found to behave linearly in relation with stride for stride values less than 6 and in a quadratic relation for stride values greater than 6 for 8 predicted frames.

We further assessed the potential of recursively evaluating the model to provide long-term predictions using the output from the first model evaluation as input to the second one and so on. The investigation indicates that including some correlations improves the overall prediction accuracy; however, if several time steps are common between the input–output pairs, the prediction accuracy is lost at a faster rate. For the cases of no common or four common frames between the input and output, the normalized mean absolute error of integral quantities is comparable, even for long time predictions (i.e., 4.6 units of time). The reconstruction errors are also comparable; however, beyond 2.6 units of time, including no correlations provides smaller errors.

Future work will investigate the proposed network for backward-in-time predictions for practical environmental flows, focusing on realistic oceanic and atmospheric applications. We will also extend the scope of the work to study the impact of uncertainties in the inputs and/or models on the reliability of the backward-in-time predictions. In addition, we plan to explore the use of this approach within a data assimilation framework to estimate the initial conditions of the model according to the available data.

Acknowledgments. This research was supported by the Office of Sponsored Research (OSR) at King Abdullah University of Science and Technology (KAUST) CRG Award CRG2020-4336 and CRG2020-4077 and Virtual Red Sea Initiative Award REP/1/3268-01-01.

Data availability statement. The codes and datasets for reproducing the findings of this study are available in https://datadryad.org/stash/share/oJu106-GRZewNpaD1vVmqPtSXB8ETRN4eECqLLCnM0E.

REFERENCES

Alwassel, H., F. Caba Heilbron, and B. Ghanem, 2018: Action search: Spotting actions in videos and its application to temporal action localization. *Computer Vision–ECCV 2018*, V. Ferrari et al., Eds., Springer, 253–269.

Amini, A., I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, 2020: Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Rob. Autom. Lett.*, **5**, 1143–1150, https://doi.org/10.1109/LRA.2020.2966414.

Angelbeck, D. I., and R. Y. Minkara, 1994: Strange attractors and chaos in wastewater flow. *J. Environ. Eng.*, **120**, 122–137, https://doi.org/10.1061/(ASCE)0733-9372(1994)120:1(122).

Auroux, D., and J. Blum, 2008: A nudging-based data assimilation method: The back and forth nudging (BFN) algorithm. *Nonlinear Processes Geophys.*, **15**, 305–319, https://doi.org/10.5194/npg-15-305-2008.

——, ——, and G. Ruggiero, 2016: Data assimilation for geophysical fluids: The diffusive back and forth nudging. *Mathematical Paradigms of Climate Science*, F. Ancona et al., Eds., Springer, 139–174.

Azouani, A., E. Olson, and E. S. Titi, 2013: Continuous data assimilation using general interpolant observables. *J. Nonlinear Sci.*, **24**, 277–304, https://doi.org/10.1007/s00332-013-9189-y.

Bakarji, J., K. Champion, J. N. Kutz, and S. L. Brunton, 2022: Discovering governing equations from partial measurements with deep delay autoencoders. arXiv, 2201.05136v1, https://doi.org/10.48550/arXiv.2201.05136.

Beintema, G., A. Corbetta, L. Biferale, and F. Toschi, 2020: Controlling Rayleigh–Bénard convection via reinforcement learning. *J. Turbul.*, **21**, 585–605, https://doi.org/10.1080/14685248.2020.1797059.

Bode, M., M. Gauding, Z. Lian, D. Denker, M. Davidovic, K. Kleinheinz, J. Jitsev, and H. Pitsch, 2021: Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. *Proc. Combust. Inst.*, **38**, 2617–2625, https://doi.org/10.1016/j.proci.2020.06.022.

Boilley, A., and J.-F. Mahfouf, 2012: Assimilation of low-level wind in a high-resolution mesoscale model using the back and forth nudging algorithm. *Tellus*, **64A**, 18697, https://doi.org/10.3402/tellusa.v64i0.18697.

Bozek, K., L. Hebert, A. S. Mikheyev, and G. J. Stephens, 2018: Towards dense object tracking in a 2D honeybee hive. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, Institute of Electrical and Electronics Engineers, 4185–4193, https://doi.org/10.1109/CVPR.2018.00440.

Brunton, S. L., J. L. Proctor, and J. N. Kutz, 2016: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. USA*, **113**, 3932–3937, https://doi.org/10.1073/pnas.1517384113.

Cao, K., and X. Zhang, 2020: An improved Res-UNet model for tree species classification using airborne high-resolution images. *Remote Sens.*, **12**, 1128, https://doi.org/10.3390/rs12071128.

Cao, Y., M. S. Jolly, and E. S. Titi, 2022: Determining form for the 2D Rayleigh–Bénard problem. *Pure Appl. Funct. Anal.*, **7**, 99–132.

Chaturvedi, V., S. Kim, S. J. Smith, L. Clarke, Z. Yuyu, P. Kyle, and P. Patel, 2013: Model evaluation and hindcasting: An experiment with an integrated assessment model. *Energy*, **61**, 479–490, https://doi.org/10.1016/j.energy.2013.08.061.

Chen, T., and H. Chen, 1995: Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Networks*, **6**, 911–917, https://doi.org/10.1109/72.392253.

Chen, Z., and H. Zhang, 2019: Learning implicit fields for generative shape modeling. *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, Institute of Electrical and Electronics Engineers, 5932–5941, https://doi.org/10.1109/CVPR.2019.00609.

Chertovskih, R., E. V. Chimanski, and E. L. Rempel, 2015: Route to hyperchaos in Rayleigh-Bénard convection. *Europhys. Lett.*, **112**, 14001, https://doi.org/10.1209/0295-5075/112/14001.

Chillà, F., and J. Schumacher, 2012: New perspectives in turbulent Rayleigh-Bénard convection. *Eur. Phys. J. E*, **35**, 58, https://doi.org/10.1140/epje/i2012-12058-1.

Clement, T. P., 2010: Complexities in hindcasting models-when should we say enough is enough? *Ground Water*, **49**, 620–629, https://doi.org/10.1111/j.1745-6584.2010.00765.x.

Curry, J. H., J. R. Herring, J. Loncaric, and S. A. Orszag, 1984: Order and disorder in two- and three-dimensional Bénard convection. *J. Fluid Mech.*, **147**, 1–38, https://doi.org/10.1017/S0022112084001968.

De Dominicis, M., N. Pinardi, G. Zodiatis, and R. Lardner, 2013a: MEDSLIK-II, a Lagrangian marine surface oil spill model for short-term forecasting—Part 1: Theory. *Geosci. Model Dev.*, **6**, 1851–1869, https://doi.org/10.5194/gmd-6-1851-2013.

——, ——, ——, and R. Archetti, 2013b: MEDSLIK-II, a Lagrangian marine surface oil spill model for short-term forecasting—Part 2: Numerical simulations and validations. *Geosci. Model Dev.*, **6**, 1871–1888, https://doi.org/10.5194/gmd-6-1871-2013.

Deng, L. and D. Yu, 2014: *Deep Learning: Methods and Applications*. Vol. 7, Now Publishers, 197–387.

Doering, C. R., and P. Constantin, 1996: Variational bounds on energy dissipation in incompressible flows. III. Convection. *Phys. Rev. E*, **53**, 5957–5981, https://doi.org/10.1103/PhysRevE.53.5957.

——, and ——, 2001: On upper bounds for infinite Prandtl number convection with or without rotation. *J. Math. Phys.*, **42**, 784–795, https://doi.org/10.1063/1.1336157.

Dung, V. B., and D. V. Thien, 2014: The equation of backward diffusion and negative diffusivity. *J. Phys.*, **537**, 012011, https://doi.org/10.1088/1742-6596/537/1/012011.

Farhat, A., M. S. Jolly, and E. S. Titi, 2015: Continuous data assimilation for the 2D Bénard convection through velocity measurements alone. *Physica D*, **303**, 59–66, https://doi.org/10.1016/j.physd.2015.03.011.

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep Learning*. MIT Press, 800 pp.

Goyal, P., and Coauthors, 2017: Accurate, large minibatch SGD: Training ImageNet in 1 hour. arXiv, 1706.02677v2, https://doi.org/10.48550/arXiv.1706.02677.

Greenside, H. S., and W. M. Coughran Jr., 1984: Nonlinear pattern formation near the onset of Rayleigh-Bénard convection. *Phys. Rev. A*, **30**, 398–428, https://doi.org/10.1103/PhysRevA.30.398.

Haghighat, E., and R. Juanes, 2021: SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Comput. Methods Appl. Mech. Eng.*, **373**, 113552, https://doi.org/10.1016/j.cma.2020.113552.

Hastings, A., C. L. Hom, S. Ellner, P. Turchin, and H. C. J. Godfray, 1993: Chaos in ecology: Is mother nature a strange attractor? *Annu. Rev. Ecol. Syst.*, **24**, 1–33, https://doi.org/10.1146/annurev.es.24.110193.000245.

He, K., X. Zhang, S. Ren, and J. Sun, 2016: Deep residual learning for image recognition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, Institute of Electrical and Electronics Engineers, 1063–6919, https://doi.org/10.1109/CVPR.2016.90.

Herty, M., G. Puppo, S. Roncoroni, and G. Visconti, 2020: The BGK approximation of kinetic models for traffic. *Kinetic Relat. Models*, **13**, 279–307, https://doi.org/10.3934/krm.2020010.

Homan, M. D., and A. Gelman, 2014: The No-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, **15**, 1593–1623.

Hou, R., C. Chen, and M. Shah, 2017: Tube convolutional neural network (T-CNN) for action detection in videos. *Proc. IEEE Int. Conf. on Computer Vision*, Venice, Italy, Institute of Electrical and Electronics Engineers, 5823–5832, https://doi.org/10.1109/ICCV.2017.620.

Howard, L. N., 1963: Heat transport by turbulent convection. *J. Fluid Mech.*, **17**, 405–432, https://doi.org/10.1017/S0022112063001427.

——, 1972: Bounds on flow quantities. *Annu. Rev. Fluid Mech.*, **4**, 473–494, https://doi.org/10.1146/annurev.fl.04.010172.002353.

Jiang, C. M., and Coauthors, 2020: MeshfreeFlowNet: A physics-constrained deep continuous space-time super-resolution framework. *SC'20: Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, Institute of Electrical and Electronics Engineers, 9, https://dl.acm.org/doi/10.5555/3433701.3433712.

Karniadakis, G. E., I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, 2021: Physics-informed machine learning. *Nat. Rev. Phys.*, **3**, 422–440, https://doi.org/10.1038/s42254-021-00314-5.

Le Maître, O. P., M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, 2002: A stochastic projection method for fluid flow: II. Random process. *J. Comput. Phys.*, **181**, 9–44, https://doi.org/10.1006/jcph.2002.7104.

Le Quéré, P., 1991: Accurate solutions to the square thermally driven cavity at high Rayleigh number. *Comput. Fluids*, **20**, 29–41, https://doi.org/10.1016/0045-7930(91)90025-D.

Li, G., M. Müller, A. Thabet, and B. Ghanem, 2019: DeepGCNs: Can GCNs go as deep as CNNs? *Proc. IEEE/CVF Int. Conf. on Computer Vision*, Seoul, South Korea, Institute of Electrical and Electronics Engineers, 9266–9275, https://doi.org/10.1109/ICCV.2019.00936.

Li, Z., N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, 2020: Fourier neural operator for parametric partial differential equations. arXiv, 2010.08895v3, https://doi.org/10.48550/ARXIV.2010.08895.

——, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, 2021: Physics-informed neural operator for learning partial differential equations. arXiv, 2111.03794v2, https://doi.org/10.48550/ARXIV.2111.03794.

Lim, B., S. Son, H. Kim, S. Nah, and K. Mu Lee, 2017: Enhanced deep residual networks for single image super-resolution. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Honolulu, HI, Institute of Electrical and Electronics Engineers, 1132–1140, https://doi.org/10.1109/CVPRW.2017.151.

Ling, J., A. Kurzawski, and J. Templeton, 2016: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.*, **807**, 155–166, https://doi.org/10.1017/jfm.2016.615.

Long, J., E. Shelhamer, and T. Darrell, 2015: Fully convolutional networks for semantic segmentation. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, Institute of Electrical and Electronics Engineers, 3431–3440, https://doi.org/10.1109/CVPR.2015.7298965.

Lu, L., P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, 2021: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, **3**, 218–229, https://doi.org/10.1038/s42256-021-00302-5.

Lusch, B., J. N. Kutz, and S. L. Brunton, 2018: Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.*, **9**, 4950, https://doi.org/10.1038/s41467-018-07210-0.

Ma, H.-Y., and Coauthors, 2015: An improved hindcast approach for evaluation and diagnosis of physical processes in global climate models. *J. Adv. Model. Earth Syst.*, **7**, 1810–1827, https://doi.org/10.1002/2015MS000490.

Mao, Z., A. D. Jagtap, and G. E. Karniadakis, 2020: Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.*, **360**, 112789, https://doi.org/10.1016/j.cma.2019.112789.

Mhina, H., S. S. Hassane, and M. McCurdy, 2021: A numerical investigation of Rayleigh-Benard convection with an obstruction. arXiv, 2110.11470v2, https://doi.org/10.48550/ARXIV.2110.11470.

Moin, P., 2010: *Fundamentals of Engineering Numerical Analysis*. 2nd ed. Cambridge University Press, 241 pp.

Osborne, J., 2021: Nonlinear data assimilation for ocean forecasting. Tech. Rep. NRL/7320/FR–2021/1, 19 pp., https://apps.dtic.mil/sti/pdfs/AD1124365.pdf.

Pandey, A., J. D. Scheel, and J. Schumacher, 2018: Turbulent superstructures in Rayleigh-Bénard convection. *Nat. Commun.*, **9**, https://doi.org/10.1038/s41467-018-04478-0.

Paul, M. R., M. I. Einarsson, P. F. Fischer, and M. C. Cross, 2007: Extensive chaos in Rayleigh-Bénard convection. *Phys. Rev. E*, **75**, 045203, https://doi.org/10.1103/PhysRevE.75.045203.

Raissi, M., P. Perdikaris, and G. E. Karniadakis, 2019: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, **378**, 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.

Rasp, S., M. S. Pritchard, and P. Gentine, 2018: Deep learning to represent subgrid processes in climate models. *Proc. Natl. Acad. Sci. USA*, **115**, 9684–9689, https://doi.org/10.1073/pnas.1810286115.

Ronneberger, O., P. Fischer, and T. Brox, 2015: U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*, N. Navab et al., Eds., Springer, 234–241.

Rosenblatt, F., 1961: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Tech. Rep. 1196-G-8, 626 pp., https://safari.ethz.ch/digitaltechnik/spring2018/lib/exe/fetch.php?media=neurodynamics1962rosenblatt.pdf.

Ruggiero, G. A., Y. Ourmières, E. Cosme, J. Blum, D. Auroux, and J. Verron, 2015: Data assimilation experiments using diffusive back-and-forth nudging for the NEMO ocean model. *Nonlinear Processes Geophys.*, **22**, 233–248, https://doi.org/10.5194/npg-22-233-2015.

Saha, S., and Coauthors, 2021: Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering. *Comput. Methods Appl. Mech. Eng.*, **373**, 113452, https://doi.org/10.1016/j.cma.2020.113452.

Saito, S., Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, 2019: PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. *Proc. IEEE/CVF Int. Conf. on Computer Vision*, Seoul, South Korea, Institute of Electrical and Electronics Engineers, 2304–2314, https://doi.org/10.1109/ICCV.2019.00239.

Scheidegger, F., L. Cavigelli, M. Schaffner, A. C. I. Malossi, C. Bekas, and L. Benini, 2017: Impact of temporal subsampling on accuracy and performance in practical video classification. *2017 25th European Signal Processing Conf. (EUSIPCO)*, Kos, Greece, Institute of Electrical and Electronics Engineers, 996–1000, https://doi.org/10.23919/EUSIPCO.2017.8081357.

Seeni, A., P. Rajendran, and H. Mamat, 2021: A CFD mesh independent solution technique for low Reynolds number propeller. *CFD Lett.*, **11**, 15–30.

Sheikholeslami, M., M. B. Gerdroodbary, R. Moradi, A. Shafee, and Z. Li, 2019: Application of neural network for estimation of heat transfer treatment of AL_2O_3-H_2O nanofluid through a channel. *Comput. Methods Appl. Mech. Eng.*, **344**, 1–12, https://doi.org/10.1016/j.cma.2018.09.025.

Simonyan, K., and A. Zisserman, 2015: Very deep convolutional networks for large-scale image recognition. arXiv, 1409.1556v6, https://doi.org/10.48550/arXiv.1409.1556.

Sun, W.-Y., and O. M. Sun, 2017: Backward integration of diffusion equation. *Aerosol Air Qual. Res.*, **17**, 278–289, https://doi.org/10.4209/aaqr.2016.06.0271.

Sutskever, I., J. Martens, G. Dahl, and G. Hinton, 2013: On the importance of initialization and momentum in deep learning. *Proc. 30th Int. Conf. on Machine Learning*, Atlanta, GA, PMLR, 1139–1147, https://proceedings.mlr.press/v28/sutskever13.html.

Tamaddon-Jahromi, H. R., N. K. Chakshu, I. Sazonov, L. M. Evans, H. Thomas, and P. Nithiarasu, 2020: Data-driven inverse modelling through neural network (deep learning) and computational heat transfer. *Comput. Methods Appl. Mech. Eng.*, **369**, 113217, https://doi.org/10.1016/j.cma.2020.113217.

van der Poel, E. P., R. J. A. M. Stevens, and D. Lohse, 2011: Connecting flow structures and heat flux in turbulent Rayleigh-Bénard convection. *Phys. Rev. E*, **84**, 045303, https://doi.org/10.1103/PhysRevE.84.045303.

Weidauer, T., O. Pauluis, and J. Schumacher, 2010: Cloud patterns and mixing properties in shallow moist Rayleigh–Bénard convection. *New J. Phys.*, **12**, 105002, https://doi.org/10.1088/1367-2630/12/10/105002.

Yang, X. I. A., S. Zafar, J.-X. Wang, and H. Xiao, 2019: Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Phys. Rev. Fluids*, **4**, 034602, https://doi.org/10.1103/PhysRevFluids.4.034602

Yigit, S., R. J. Poole, and N. Chakraborty, 2015: Effects of aspect ratio on laminar Rayleigh–Bénard convection of power-law fluids in rectangular enclosures: A numerical investigation. *Int. J. Heat Mass Trans.*, **91**, 1292–1307, https://doi.org/10.1016/j.ijheatmasstransfer.2015.08.032.

Zodiatis, G., R. Lardner, D. Solovyov, X. Panayidou, and M. De Dominicis, 2012: Predictions for oil slicks detected from satellite images using MyOcean forecasting data. *Ocean Sci.*, **8**, 1105–1115, https://doi.org/10.5194/os-8-1105-2012.