Generalizability of reservoir computing for flux-driven two-dimensional convection

Florian Heyder,[1] Juan Pedro Mellado,[2] and Jörg Schumacher[1,3]

[1]*Institut für Thermo- und Fluiddynamik, Technische Universität Ilmenau, Postfach 100565, D-98684 Ilmenau, Germany*

[2]*Meteorologisches Institut, Universität Hamburg, Bundesstraße 55, D-20146 Hamburg, Germany*

[3]*Tandon School of Engineering, New York University, New York, New York 11201, USA*

(Received 16 May 2022; revised 17 August 2022; accepted 21 September 2022; published 9 November 2022)

We explore the generalization properties of an echo state network applied as a reduced-order model to predict flux-driven two-dimensional turbulent convection. To this end, we consider a convection domain with constant height with a variable ratio of buoyancy fluxes at the top and bottom boundaries, which break the top-down symmetry in comparison to the standard Rayleigh-Bénard case, thus leading to highly asymmetric mean and fluctuation profiles across the layer. Our direct numerical simulation model describes a convective boundary layer in a simple way. The data are used to train and test a recurrent neural network in the form of an echo state network. The input of the echo state network is obtained in two different ways, either by a proper orthogonal decomposition or by a convolutional autoencoder. In both cases, the echo state network reproduces the turbulence dynamics and the statistical properties of the buoyancy flux, and is able to model unseen data records with different flux ratios.

DOI: 10.1103/PhysRevE.106.055303

I. INTRODUCTION

Machine learning (ML) methods are known for their exceptional capabilities in the classification of comprehensive data records and data-driven modeling. In fluid mechanics, ML thus found its way into the analysis and control of turbulent flows [1–6]. Applications now cover a broad spectrum of problems, such as the subgrid scale modeling in Reynolds-averaged Navier-Stokes equations [7] or large-eddy simulations [8,9], the exploration of inertial manifolds in the phase space of the systems [10], and the control of their spatiotemporal dynamics [11], or the reconstruction and generation of partially missing turbulent data by deep neural networks [12,13]. The aim of such ML applications is usually to reduce the computational cost which comes with solving the governing equations of motion in direct numerical simulations or analyzing high-resolution experimental data [14]. Among other fluid flows which have been studied in Refs. [15–18], turbulent convection has been chosen as a prominent application case. We mention the classification of convective heat flux patterns [19], spectral nudging methods to reconstruct the flow fields from temperature measurements [20], the application of reinforcement learning to control the heat transport in a Rayleigh-Bénard cell [21], or the prediction and reconstruction of turbulent dry and moist convection flows by recurrent neural networks [22–25]. Convection plays a prominent role in geophysical flows [26,27]. Machine learning is then applied for parametrizations of unresolved convection processes in the oceans [28] and atmosphere [29,30]. Given the strong variability in the environmental conditions in atmospheric flows, particular interest lies in the development of robust ML methods which can model configurations that are different from the training cases with respect to the parameter setting [31]. This particular point sets the stage for the present work which consists of two major parts.

In the first part, we discuss a two-dimensional Rayleigh-Bénard convection (RBC) model [32] that is driven by heat or buoyancy fluxes from the top and bottom. In our direct numerical simulations (DNS), we consider a convective cell with constant height H in which bottom and top fluxes are chosen such that the cell as a whole is differently strongly heated from the bottom and the top. This configuration can be understood as a simplified model of a convective boundary layer (CBL) in cloud-free and shear-free conditions. In this model, we retain the entrainment of fluid from the free troposphere into the turbulent region by prescribed top flux into the convective domain as well as heating from below, i.e., from the heated ground. A difference between this model and a real CBL is that H in our model remains constant, whereas the height of the atmospheric layer increases slowly with increasing time, i.e., the CBL grows into the free troposphere [33–36]. Given this setup, the top-down symmetry of a standard RBC flow will be broken; highly asymmetric mean profiles of buoyancy, convective buoyancy flux, and velocity fluctuations follow. This clearly challenges the reproduction of statistical properties by the ML algorithm.

In the second part, we use the DNS data as a training data base to study the performance of reduced-order models of turbulent convection *dynamics* based on recurrent neural network architectures, i.e., neural networks with a short-term memory. These ML algorithms will then be applied to data that have a different ratio of boundary fluxes as the training configuration, i.e., a different set of system parameters. Our study thus addresses one important point of supervised ML algorithms, namely, how well do they perform with respect to unseen data with changed system parameters—known as the generalization property or generalizability [31]. More specifically, we apply echo state networks (ESNs) which are one implementation of reservoir computing [37,38]. The ESN approach has found interest recently in inferring states of

a nonlinear dynamical system. Applications concerned the Rössler and Lorenz 63 systems [39,40], the Lorenz 96 model [41], and Galerkin models of plane shear flows [42]. Moreover, hybrid models which combine both data-driven (ESN) and knowledge-based methods, i.e., solving the mathematical equations, have already been proposed [43,44] and tested in terms of a global atmospheric forecast model [45]. Further, reservoir computing techniques, due to their computationally inexpensive training routine, could serve as a substitute for conventional parametrization schemes. The performance of ESNs in two-dimensional dry and moist turbulent Rayleigh-Bénard convection has already shown great promise, as low-order statistics of buoyancy and liquid water fluxes are successfully reproduced [22–24]. Here, we want to apply this framework to a case that is a bit closer to real atmospheric flows than standard RBC.

Even two-dimensional DNS data records are still too large to be directly processed by the ESN. Thus, a data reduction step is required. We suggest two methods here: (1) the proper orthogonal decomposition (POD) and (2) the convolutional autoencoder (CAE) [24,46–49]. As a consequence, the present ML algorithm is a combination of two building blocks, i.e., the encoder-decoder module and the dynamical core in the form of an ESN which advances the convection flow in time in the low-dimensional latent space. We will refer to this as the combined POD/CAE–ESN algorithm in the following.

It is found that despite smaller differences in flux statistics and reconstruction of the fields, both models perform well. We thus investigate (1) the performance of two data reduction methods, namely, POD and CAE, on turbulent convection data, (2) the generalization capability of the ESN to data with a different heat flux ratio (which will be defined in the next section), and (3) the combined application of data reduction and reservoir computing to flux-driven highly asymmetric Rayleigh-Bénard convection flow.

The outline of the manuscript is as follows. In Sec. II, we describe the two-dimensional convection model and define all parameters, in particular, the ratio of the buoyancy fluxes at the top and bottom boundaries, β, which is the major control parameter. Section III introduces the individual modules of the combined POD/CAE–ESN algorithm. We provide details of the training and the generalization performance of the ESN. We summarize our results and give a brief outlook in Sec. IV. Technical details of ML and further results are listed in Appendices A to D.

II. FLUX-DRIVEN CONVECTION MODEL

A. Governing equations and model parameters

We use the Boussinesq approximation to the two-dimensional Navier-Stokes equations. For convenience, we formulate the problem in terms of the buoyancy field b, which is given by

$$b(x,z,t) \equiv \alpha g T(x,z,t), \tag{1}$$

where α, g, and T are the thermal expansion coefficient, the acceleration due to gravity, and the temperature field, respectively. We consider a cell of height H and length L (see Fig. 1). In the vertical direction, we consider no-slip boundary conditions for the velocity and constant-flux boundary conditions

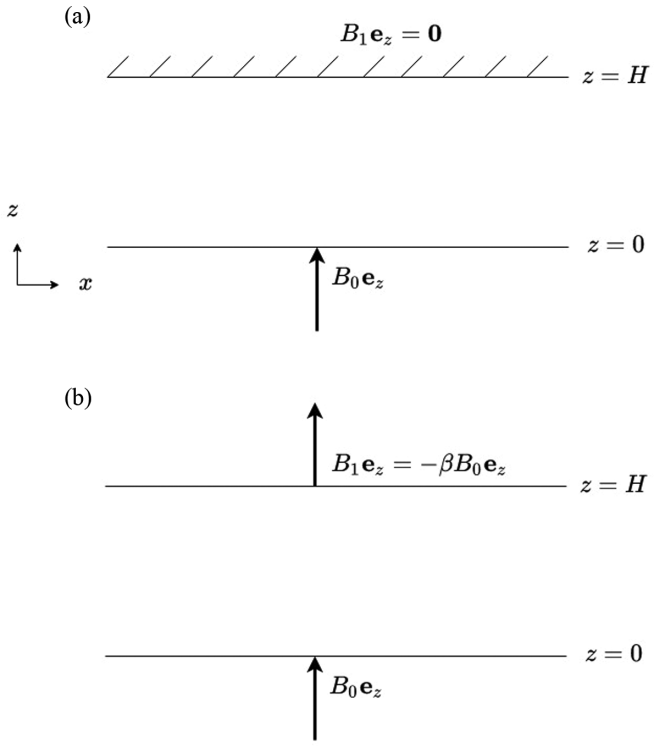

FIG. 1. Scheme of the two-dimensional Rayleigh-Bénard setup with constant buoyancy-flux boundary conditions. The bottom of the cell is heated by the incoming flux B_0. We explore the effect of asymmetric boundary conditions by imposing a different flux $B_1 = -\beta B_0$ ($\beta > 0$) at the top. The values $\beta \in \{0.1, 0.2, 0.3\}$ are representative values for convective boundary layers. (a) Adiabatic top wall with $\beta = 0$ and (b) warming flux at the top with $\beta > 0$.

for the buoyancy. We impose the fluxes B_0 and B_1 at the bottom and top, respectively. In the horizontal direction, we consider periodic boundary conditions.

The resulting evolution equations are given by

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_z}{\partial z} = 0, \tag{2}$$

$$\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_z\frac{\partial u_x}{\partial z} = -\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial z^2}\right), \tag{3}$$

$$\frac{\partial u_z}{\partial t} + u_x\frac{\partial u_z}{\partial x} + u_z\frac{\partial u_z}{\partial z} = -\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial z^2}\right) + b, \tag{4}$$

$$\frac{\partial b}{\partial t} + u_x\frac{\partial b}{\partial x} + u_z\frac{\partial b}{\partial z} = \kappa\left(\frac{\partial^2 b}{\partial x^2} + \frac{\partial^2 b}{\partial z^2}\right). \tag{5}$$

In these equations, u_x and u_z are the horizontal and vertical components of the velocity, p is the modified pressure divided by the constant density, ν is the kinematic viscosity, and κ is the molecular diffusivity. The boundary conditions are $u_x = 0$ and $u_z = 0$ at $z = 0$ and $z = 1$, together with

$$\frac{\partial b}{\partial z}(x, z = 0, t) = -B_0/\kappa, \tag{6}$$

$$\frac{\partial b}{\partial z}(x, z = 1, t) = -B_1/\kappa. \tag{7}$$

For the sake of generality, we will present the analysis in a nondimensional form. Choosing H and B_0 as reference scales, one finds the following characteristic scales: the convective velocity $(B_0H)^{1/3}$, the convective time $(H^2/B_0)^{1/3}$, and the convective buoyancy $(B_0^2/H)^{1/3}$ [50]. The resulting four system parameters are the aspect ratio $\Gamma = L/H$, the Prandtl number

$$\mathrm{Pr} = \frac{\nu}{\kappa}, \tag{8}$$

the convective Rayleigh number

$$\mathrm{Ra}_c = \frac{B_0 H^4}{\nu\kappa^2}, \tag{9}$$

and the buoyancy-flux ratio

$$\beta = -\frac{B_1}{B_0}. \tag{10}$$

As further explained below, we are interested in the cases $B_0 > 0$ and $B_1 < 0$, and hence $\beta > 0$, i.e., the fluid is heated from the bottom and the top.

The buoyancy difference,

$$\Delta b = \langle b\rangle_x(z=0,t) - \langle b\rangle_x(z=1,t), \tag{11}$$

between the two plates is a dependent variable for configurations with constant-flux boundary conditions and needs to be diagnosed from experimental or simulation data (angle brackets indicate an averaging operation and the subscript indicates the variable with respect to which the averaging operation is performed, in this case, the horizontal coordinate x). Therefore, the Dirichlet Rayleigh number,

$$\mathrm{Ra}_f = \frac{\Delta b H^3}{\nu\kappa}, \tag{12}$$

is a diagnostic variable as well. From Eqs. (2)–(5), one can derive the vertical buoyancy profile, up to a constant, for the purely conductive case to be

$$b_{\mathrm{cond}} = \frac{B_0 H}{\kappa}\frac{(z/H-1)^2 + \beta(z/H)^2}{2} + \frac{B_0}{H}(1+\beta)t + \mathrm{const.} \tag{13}$$

For $\beta = -1$, we recover the steady, linear solution that corresponds to the problem with Dirichlet boundary conditions. In the Neumann case, the profile of b has a parabolic shape and it grows linearly in time (given that we heat from below and from above). Nonetheless, it is quasisteady in the sense that the shape of the profile remains constant in time. The buoyancy difference between the bottom and top plates for this case is

$$\Delta b_{\mathrm{cond}} = \frac{B_0 H}{\kappa}\frac{1-\beta}{2}. \tag{14}$$

B. Direct numerical simulations

We fix the Prandtl and convective Rayleigh number to $\mathrm{Pr} = 1$ and $\mathrm{Ra}_c = 3\times10^8$ and consider extended layers with an aspect ratio $\Gamma = L/H = 24$. The control parameter that we vary is the flux-ratio parameter β defined by Eq. (10). In the atmospheric CBL over land, one typically finds the conditions $B_0 > 0$ and $B_1 < 0$, which represent the surface warming and the entrainment warming of the CBL, respectively. Hence, we are mainly interested in the case $\beta > 0$. Typical atmospheric conditions correspond to the range of β ranging approximately from 0.1 to 0.3 [26,51]. As β increases further to a value of 0.4, the upper region of the convective layer is increasingly stabilized (positive mean buoyancy gradient; see, also, Fig. 5), while an unstable layer of decreasing area fraction at the bottom continues to exhibit convective motion. In fact, preliminary simulations (not shown) indicate that at $\beta \approx 0.5$, the buoyancy difference Δb changes the sign which is in line with a different dynamics. Therefore, we will consider the cases $\beta \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$ in our CBL model (see, also, Fig. 1). The case $\beta = 0$ corresponds to an upper adiabatic wall. This case is considered as a first step to understand the effect of asymmetries in the boundary conditions in the results obtained from Rayleigh-Bénard convection with constant-buoyancy boundaries.

TABLE I. Simulation parameters. The time average has been calculated over the last 500 free-fall times. The buoyancy and time values in the second and third columns are given in units of convective buoyancy $(B_0^2/H)^{1/3}$ and convective time $(H^2/B_0)^{1/3}$, respectively. The last column lists the large-scale eddy turnover time $\tau_{\mathrm{eddy}} = H/u_{\mathrm{rms}}$ with $u_{\mathrm{rms}} = \langle u_i^2\rangle_{V,t}^{1/2}$. This timescale is given in units of the free-fall time T_f.

β	$\langle\Delta b\rangle_t$	$\langle T_f\rangle_t$	$\langle \mathrm{Nu}_f\rangle_t$	$\langle \mathrm{Ra}_f\rangle_t$	τ_{eddy}
0.0	22.0 ± 0.4	0.21	15.2 ± 0.3	9.9×10^6	3.08
0.1	19.0 ± 0.4	0.23	15.9 ± 0.3	8.5×10^6	3.03
0.2	15.4 ± 0.3	0.26	17.4 ± 0.4	6.9×10^6	2.92
0.3	10.7 ± 0.5	0.31	21.9 ± 0.9	4.8×10^6	2.74
0.4	4.1 ± 0.51	0.50	49.8 ± 6.6	$1.84\cdot10^6$	1.94

The Boussinesq equations (2)–(5) are discretized by a high-order spectral-like compact finite-difference method. The time evolution is treated by a low-storage fourth-order Runge-Kutta scheme. The pressure-Poisson equation is solved with a Fourier decomposition in the horizontal planes and a factorization of the resulting difference equations in the vertical direction. More details of the numerical method can be found in Mellado and Ansorge [52]. The software used to perform the simulations is freely available; see Ref. [53].

The grid size is $N_x \times N_z = 2400 \times 150$. The horizontal grid spacing is uniform. The vertical grid spacing follows a hyperbolic tangent profile: it is equal within 1.2% to the horizontal grid spacing in the center of the convection cell, and diminishes by a factor of 2.5 next to the wall. The time steps are in the range $\Delta t \approx 0.0012$–$0.0016\,(H^2/B_0)^{1/3}$, with the specific value depending on the simulation. They are defined to obtain data exactly every 0.25 free-fall times (definition follows) and satisfy the stability constraints of the numerical algorithm described in the previous paragraph. Since the free-fall time is a derived variable in the case of the constant-flux boundaries considered in this study, preliminary simulations were performed to obtain the free-fall time in each case, and we repeated the simulations with the appropriate Δt. Table I summarizes important parameters of the five simulation runs.

C. Cellular convection patterns and vertical profiles at different flux ratios

Integrating the evolution equation for b yields that the volume-averaged buoyancy $\langle b\rangle_{x,z}$ increases as

$$\langle b\rangle_{x,z} = \frac{B_0}{H}(1+\beta)t. \tag{15}$$

Hence, in the turbulent case, the fluid warms linearly with increasing time as in the conduction case. The mean vertical profile, however, is different from the pure conduction profile and, as mentioned above, a major dependent variable is the buoyancy difference Δb across the cell. After an initial transient, this quantity becomes statistically stationary, as can be seen in Fig. 2(a) for all five simulations. The free-fall time $T_f = \sqrt{H/\Delta b}$ and free-fall velocity $U_f = \sqrt{H\Delta b}$ can be computed and used as scales for better comparison to the more common case of Rayleigh-Bénard convection with constant-buoyancy boundaries. Moreover, we can express the buoyancy difference in terms of a Nusselt number,

$$\mathrm{Nu} = \frac{\Delta b_{\mathrm{cond}}}{\Delta b} = \frac{1-\beta}{2}\frac{B_0 H}{\kappa \Delta b}, \tag{16}$$

defined here as the ratio between the buoyancy difference in the purely conductive case Δb_{cond} [see Eq. (14)] and the fully convective case, i.e., Δb. For $\beta = -1$, we again recover the functional relationship corresponding to Rayleigh-Bénard convection with constant-buoyancy boundaries. The relaxation to a statistically stationary state for the buoyancy difference and the Nusselt number are demonstrated in Fig. 2 for all five cases.

Figures 3 and 4 show snapshots of the normalized buoyancy, which is given by

$$b^*(x,z) = \frac{b(x,z) - \langle b\rangle_{x,t}(z=1)}{\langle \Delta b\rangle_t}, \tag{17}$$

and the vertical flux $u_z'(x,z)b'(x,z)$ in the statistically stationary regime. For $\beta = 0.0$, the flux at the top is zero and no thermal boundary layer is present. This changes when the warming flux at the top becomes greater than zero, i.e., $\beta > 0$. With increasing warming flux at the top, we find a thermal boundary layer at $z = 1$, which increases in thickness as β increases. For $\beta = 0.4$, the convection is confined to a smaller domain near the bottom plate. From there, thermal plumes detach and rise into the bulk that is increasingly stabilized from the top, thus causing the strongly fluctuating time series of the Nusselt number. Naturally, the structures in the buoyancy flux are also affected by the change of the top flux. As more buoyant fluid is transported from the top into the center of the turbulent region, the cellular order is increasingly dissolved, which can be seen by prominent thermal plumes in both figures; compare Figs. 3(a), 4(a) and Figs. 3(e), 4(e).

We show the line-time average vertical profiles, denoted as $\langle\cdot\rangle_{x,t}(z)$, of b^* in Fig. 5(a). All profiles show the tendency towards a constant mean value in the central part of the domain, implying a layer of well-mixed fluid. Contrary to the common Rayleigh-Bénard case with constant-buoyancy boundary conditions [32], constant-flux boundary conditions break the top-down symmetry of the mean buoyancy profile. Furthermore, for $\beta > 0$, the incoming warming flux at $z = 1$ results in positive buoyancy gradients and hence a stable layer

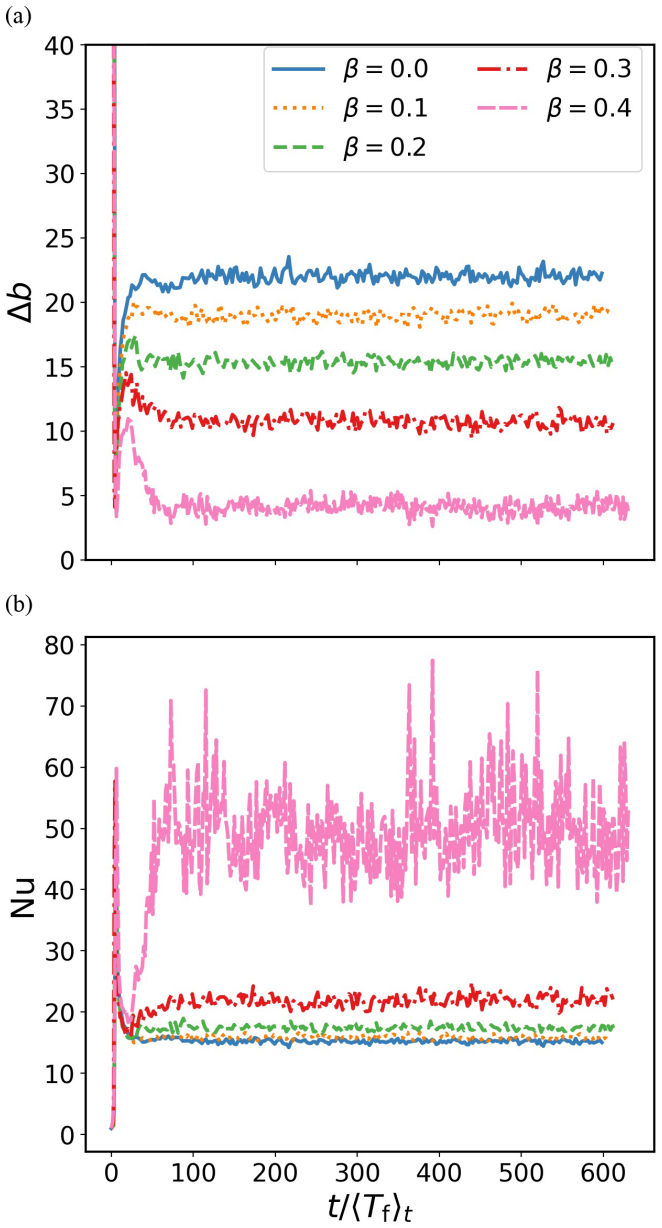

FIG. 2. Temporal variation of (a) the buoyancy difference Δb [see Eq. (11)] and (b) the Nusselt number Nu [see Eq. (16)]. Both quantities become statistically stationary after an initial transient. The case $\beta = 0.4$ exhibits the strongest fluctuations, in particular for the Nusselt number in (b). This indicates a different dynamics in comparison to $\beta \in [0.1, 0.3]$ and the adiabatic top configuration. Note that both time axes are normalized by the time mean of the free-fall time $T_f = \sqrt{H/\Delta b}$ in the statistical stationary regime.

at the top. We examine the variability of the velocity and buoyancy fields fields and decompose both into their volume mean $\langle u_x\rangle_{x,z}$, $\langle u_z\rangle_{x,z}$, $\langle b\rangle_{x,z}$ and their fluctuations u_x', u_z', b'. They are given by

$$u_x(x,z,t) = \langle u_x\rangle_{x,z}(t) + u_x'(x,z,t), \tag{18}$$

$$u_z(x,z,t) = \langle u_z\rangle_{x,z}(t) + u_z'(x,z,t), \tag{19}$$

$$b(x,z,t) = \langle b\rangle_{x,z}(t) + b'(x,z,t). \tag{20}$$

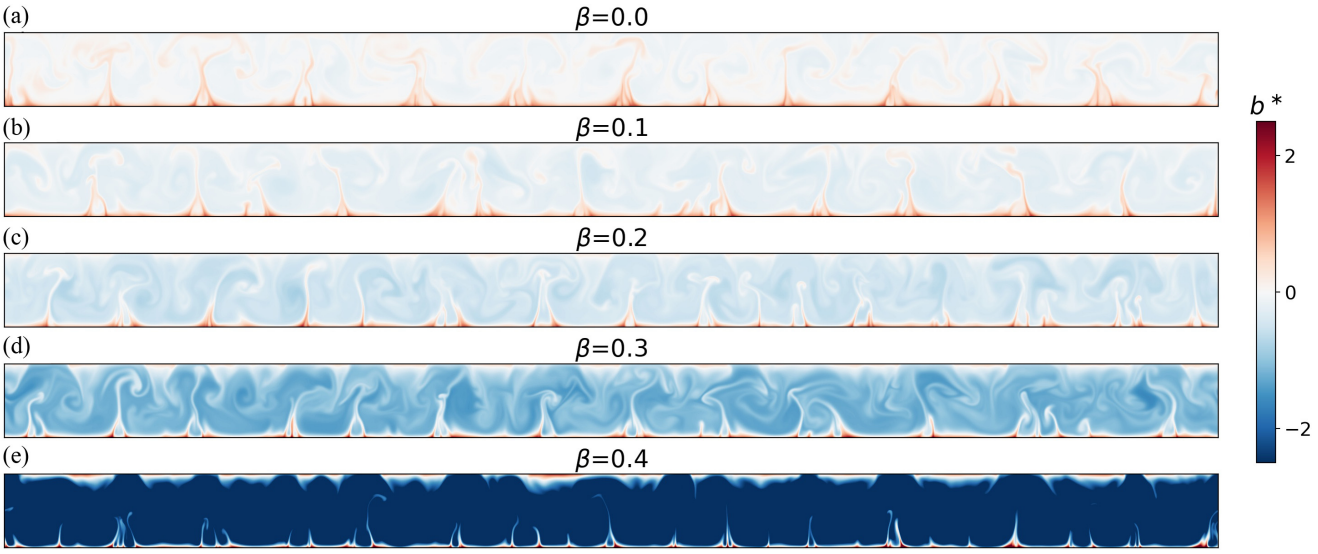

FIG. 3. Instantaneous snapshot of the normalized buoyancy field $b^* = (b - \langle b\rangle_{x,t}(z=1))/\langle \Delta b\rangle_t$ in the statistically stationary state. The five different top boundary conditions ($\beta = 0.0, 0.1, 0.2, 0.3, 0.4$) differ in their width of the top thermal boundary layer. For the adiabatic top $\beta = 0.0$, no such layer is present. For $\beta = 0.4$, the convective motion is confined to a small layer near the bottom plate. Note that with increasing β, the range of b^* increases.

Note that $\langle b\rangle_{x,z}$ depends on time, as it incorporates the linear warming of the fluid. Meanwhile, $\langle u_x\rangle_{x,z}$ and $\langle u_z\rangle_{x,z}$ are statistically stationary and vary weakly about their zero mean. The vertical profiles of the root mean square (rms) of u_z' and b' are shown in Figs. 5(b) and 5(c). The rms's of the fluctuations of the buoyancy differ greatly in their magnitude and trend in the upper portion of the domain. The vertical rms velocity component $\langle u_z'^2\rangle_{x,t}^{1/2}$, on the other hand, does not vary too much while changing β. Additionally, the total buoyancy flux,

$$F_b = \langle u_z' b'\rangle_{x,t} - \kappa \frac{\partial \langle b\rangle_{x,t}}{\partial z}, \tag{21}$$

normalized by its bottom value is shown in Fig. 5(d). We find that the flux decreases linearly with increasing height. As indicated by Fig. 5(a), the molecular terms mostly contribute to the near-wall regions. The turbulent transport (not shown), on the other hand, declines linearly over the middle of the domain and results in negative contributions near the top. This is expected by the stabilization by entrainment warming in the CBL [26,51], here considered by imposing the negative buoyancy flux B_1 at the top boundary. One goal of this study is to ascertain the capability to reproduce these vertical profiles of the turbulent contributions by the recurrent neural network, which will be presented in the next section.

In the following, we use the DNS data of $\beta = 0.1$ to train a recurrent neural network and make subsequent predictions for unseen data with flux ratios $\beta = 0.2$, 0.3, and 0.4, respectively. This is done to explore the generalization

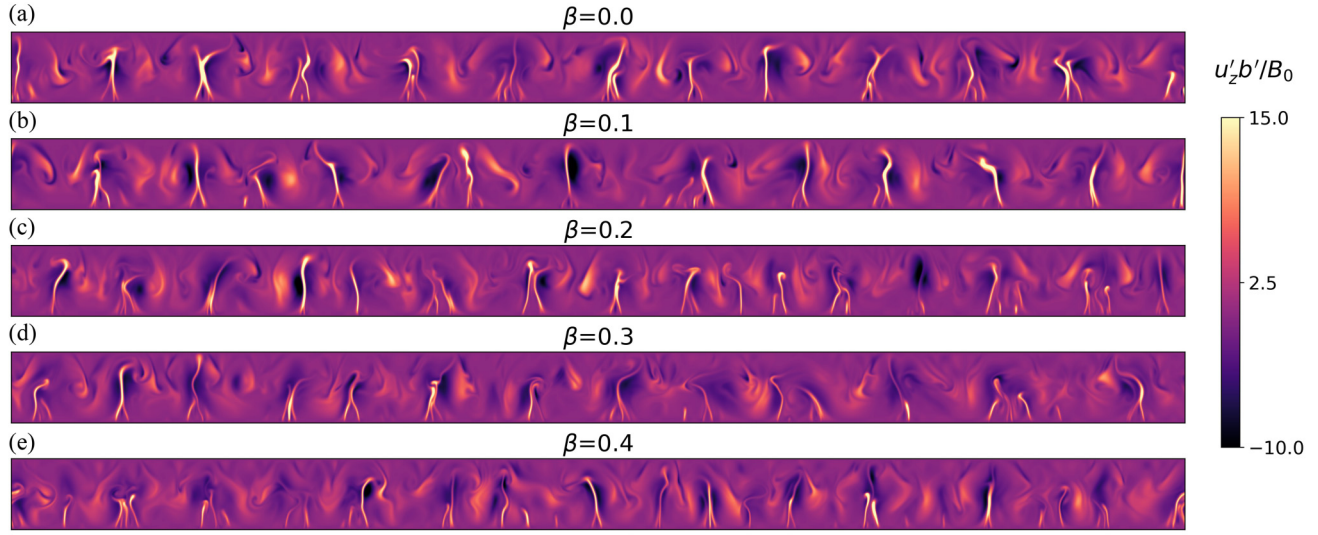

FIG. 4. Instantaneous snapshot of the vertical buoyancy flux $u_z'(x,z,t_0)b'(x,z,t_0)$ in the statistically stationary state. The cellular order is increasingly dissolved with growing parameter β. Panels (a) to (e) correspond to those of Fig. 3.

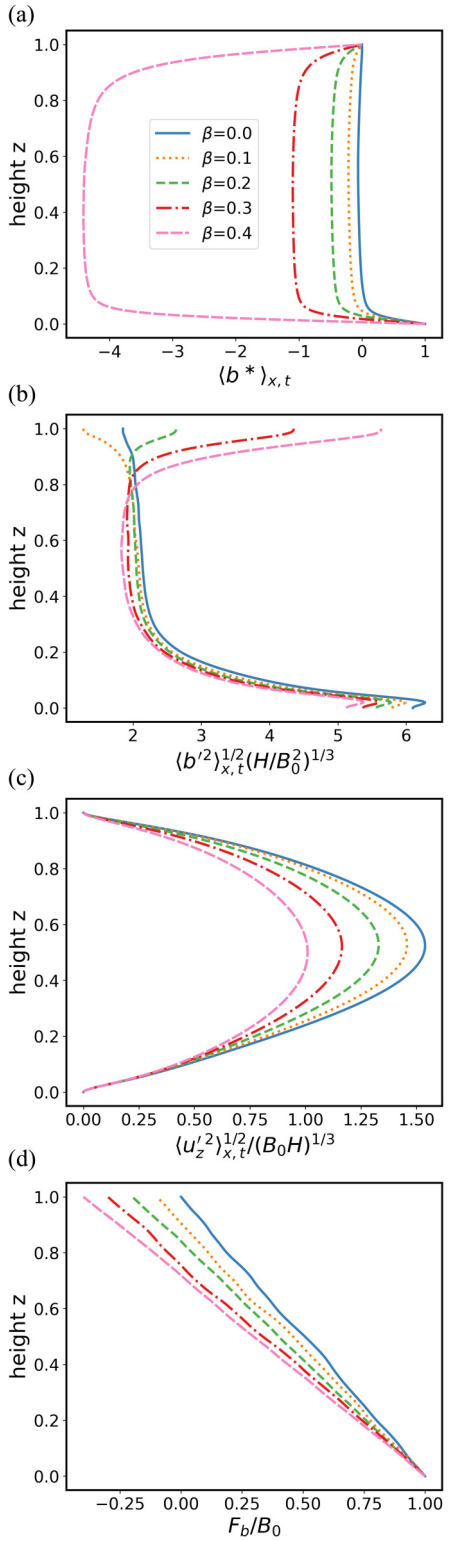

FIG. 5. Vertical profiles of (a) the normalized buoyancy $b^* = (b - \langle b \rangle_{x,t}(z=1))/\langle \Delta b \rangle_t$, (b) buoyancy fluctuations, (c) vertical velocity fluctuations, and (d) normalized total buoyancy flux $F_b(z)/F_b(z=0) = F_b(z)/B_0$. While the boundary conditions significantly affect the buoyancy and its fluctuations, the influence on the vertical velocity profiles is less important. The fluxes show linear variation across the cell. The legend shown in (a) is valid for all graphs shown.

properties of the echo state networks. We therefore interpolate all fields from the nonuniform grid with 2400 × 150 points to a 720 × 30 uniform grid by cubic splines. This grid will be denoted as the coarse-grained grid, and the data as coarse-grained DNS data.

III. CONVECTION PREDICTION FROM MACHINE LEARNING ALGORITHM

In the following, we describe the architecture of the ESN module of the combined POD/CAE–ESN machine learning algorithm and its training in more detail before discussing the predicted results for DNS at different parameters than the training data. Details of the architecture of the CAE and the POD snapshot method are outlined in Appendices A and B, respectively.

A. Echo state network and echo state property

In the following, we specify the architecture of the ESN that will be applied to process the DNS data of the CBL model described in the previous section. The discrete-time reservoir state dynamics is given by

$$\mathbf{r}(n) = (1-\gamma)\mathbf{r}(n-1) + \gamma \tanh[W^{\mathrm{r}}\mathbf{r}(n-1) + W^{\mathrm{in}}\mathbf{x}(n) + d\mathbf{1}], \quad (22)$$

where $\mathbf{r}(n) \in \mathbb{R}^{N_{\mathrm{r}}}$, $\mathbf{x}(n) \in \mathbb{R}^{N_{\mathrm{in}}}$ are the reservoir state and input at time step n, respectively. $W^{\mathrm{r}} \in \mathbb{R}^{N_{\mathrm{r}} \times N_{\mathrm{r}}}$, $W^{\mathrm{in}} \in \mathbb{R}^{N_{\mathrm{r}} \times N_{\mathrm{in}}}$ are the reservoir and input weight matrices and $\gamma \in [0, 1]$, d are the constant leaking rate and constant bias. The reservoir output $\hat{\mathbf{y}} \in \mathbb{R}^{N_{\mathrm{in}}}$ is computed by a linear mapping of the extended reservoir state $\tilde{\mathbf{r}}(n) = [d, \mathbf{x}(n), \mathbf{r}(n)]$ (vertical concatenation of bias, reservoir input, and state),

$$\hat{\mathbf{y}}(n) = W_*^{\mathrm{out}}\tilde{\mathbf{r}}(n). \quad (23)$$

The fitted output weights $W_*^{\mathrm{out}} \in \mathbb{R}^{N_{\mathrm{in}} \times (1+N_{\mathrm{in}}+N_{\mathrm{r}})}$ are chosen as to minimize the mean square cost function,

$$C(W^{\mathrm{out}}) = \sum_{n=-T_L}^{-1} \|\mathbf{y}(n) - W^{\mathrm{out}}\tilde{\mathbf{r}}(n)\|_2^2 + \lambda \|W^{\mathrm{out}}\|_F^2, \quad (24)$$

where $\mathbf{y}$ are the target outputs, which are part of the training data. T_L is the number of training time steps and $\|\cdot\|_F$ denotes the Frobenius norm. The last term penalizes large values of the output weight matrix by adjusting the regression parameter λ. This is one possibility to avoid overfitting, where the machine learning algorithm learns the training data by heart, consequently performing poorly when operating on data outside the training data set. The solution to this L^2-penalized linear regression problem is given by

$$W_*^{\mathrm{out}} = YR^T(RR^T + \lambda I)^{-1}, \quad (25)$$

where the nth column of $Y \in \mathbb{R}^{N_{\mathrm{in}} \times T_L}$, $R \in \mathbb{R}^{N_{\mathrm{r}} \times T_L}$ are $\mathbf{y}(n)$ and $\tilde{\mathbf{r}}(n)$, respectively. $I \in \mathbb{R}^{N_{\mathrm{r}} \times N_{\mathrm{r}}}$ denotes the identity matrix and $(\cdot)^T$, $(\cdot)^{-1}$ are the transpose and inverse. After the training phase, an initial input is given at $n = 0$ and the reservoir output at time step $n \geqslant 0$ is fed back to the input layer, by letting $\mathbf{x}(n) = W_*^{\mathrm{out}}\tilde{\mathbf{r}}(n-1)$. During this testing phase, the ESN autonomously predicts the next T_T iterations of the initial

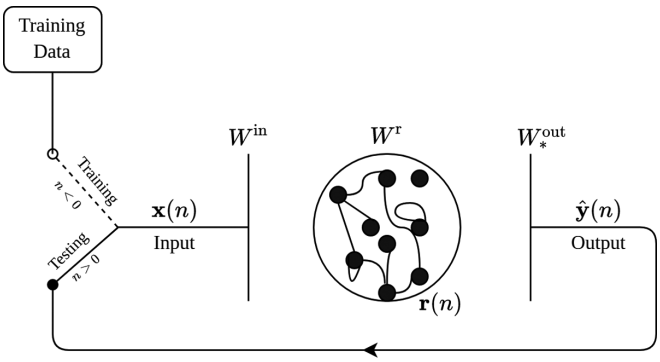

FIG. 6. Echo state network architecture. For time $n < 0$, the network learns the dynamics of the training data by computing the output weights in W^{out}_*. In the testing phase ($n > 0$), it runs in a mode of autonomous prediction, where the last network output is fed back to the input layer to be used as new input for the next prediction step.

input. Figure 6 summarizes the architecture of the ESN in a sketch.

This inexpensive training procedure comes at a cost of finding a suitable set of hyperparameters, i.e., parameters which are not learned and have to be tuned beforehand. Here we restrict ourselves to $h = \{\gamma, \lambda, N_{\text{r}}, D, \varrho\}$. The last two quantities are the reservoir density D and spectral radius ϱ. They are algebraic properties of the reservoir weight matrix and represent the number of nonzero elements and largest absolute eigenvalue of W^{r}, respectively. Finding an optimal set of these hyperparameters is crucial, as they influence the memory capacity of the reservoir [54]. In [55], a necessary condition for an effective reservoir was proposed: the *echo state property*. A reservoir is said to possess echo states when two different reservoir states $\mathbf{r}_1(n-1)$, $\mathbf{r}_2(n-1)$ converge to the same reservoir state $\mathbf{r}(n)$, provided the same input $\mathbf{x}(n)$ is given and the system has been running for many iterations n. This property highly depends on the data one uses, a suitable set of hyperparameters h, as well as the reservoir initialization [56]. So far, no universal rule for the presence of echo states has been proposed.

Note also that the echo state property is a necessary condition and that no feasible sufficient condition has yet been found, as discussed in [57]. We will keep using reservoir initializations and hyperparameter ranges, which have shown good results, e.g., in [22] or [23]. We initialize the input and reservoir weights randomly, i.e., $W^{\text{in}} \sim \mathcal{U}[-0.5, 0.5]$ and $W^{\text{r}} \sim \mathcal{U}[0, 1]$. W^{r} is then normalized by its largest absolute eigenvalue and is subsequently scaled by ϱ. Afterwards, randomly selected entries of this matrix are set to zero to get the specified value of the reservoir density D. The specific value of each of the quantities in h is chosen by a grid search procedure, which will be discussed further below. For this study, the ESN was implemented in *Python* using the library *turbESN* [58].

B. ESN training for case $\beta = 0.1$

In the following, we explore whether we can use the ESN to infer changes in the convective flow, induced by changes in the buoyancy flux at the top of the two-dimensional domain. A trained network is thus exposed to unseen data at a different physical parameter set. Such a procedure probes the generalization properties of the ESN. The subject is also connected to a transfer of the learned parameters from one task to a similar one, which is known as transfer learning [59]. Due to the computationally inexpensive training scheme of ESNs, transfer learning is not often applied for this class of algorithms, even though implementations have been proposed very recently [60].

Here, we take a different approach, which is sketched in Fig. 7. A reservoir is trained with the reduced data of one case of constant-flux boundary conditions at $z = 1$, namely, $\beta = 0.1$. An intermediate reservoir washout phase clears the reservoir memory of recent $\beta = 0.1$ information and leads to a transition of the reservoir state to one of three different and unseen convection flows with buoyancy flux parameters $\beta = 0.2$, 0.3, or 0.4. Finally, we use the trained network for the prediction of the dynamics and the statistical properties of the unseen regimes.

The DNS data possess many degrees of freedom, so that we have to introduce a preprocessing step before passing the convection data to the reservoir. We propose two common reduced-order modeling techniques, the (1) proper orthogonal decomposition (POD) and the (2) convolutional autoencoder (CAE). The former is well known in fluid mechanics as a linear method, where the data reduction is realized by a truncation to a set of Galerkin modes [46]. The CAE, on the other hand, represents a deep convolutional neural network, commonly used in deep learning tasks, such as feature extraction in image processing [48]. We stress that the same neural network architecture of the CAE (number of neurons and hidden layers) is used for each β; this CAE network is, however, trained for each β separately. That is, we do not test the generalization property of the data reduction module. For brevity, we only mention major aspects of both data reduction methods in the main text and move the details to Appendices A and B.

For both data reduction approaches, we sample 700 time steps of our coarse-grained DNS data in an interval of $0.25T_f$ for the simulation of $\beta = 0.1$ in the statistically stationary regime. Also, snapshots of 700 further time steps with the same sampling interval are gathered for the unseen target simulations at $\beta = 0.2$, 0.3, and 0.4,

$$b'(x, z, t) = \langle b' \rangle_t(x, z) + b''(x, z, t). \tag{26}$$

Finally, we apply both POD and CAE to the vector $\mathbf{g} = (u'_x, u'_z, b'')^T$ for each value of β, such that we end up with four separate POD computations and four separate CAE networks. Both methods are chosen to reduce the dimensionality of this vector to $N_{\text{POD}} = N_{\text{CAE}} = 300$ features per snapshot. The total number of degrees of freedom is thus reduced from three fields on a grid with size 2400×150 in the original DNS (that corresponds to $N_{\text{dof}} = 1.08 \times 10^6$) via coarse-grained data of grid size 720×30 to 300 modes in the latent space by a factor of 3600. With this choice of N_{POD}, the POD reduction captures about 80% of the original energy (for more details, see the Appendix).

We refer to this reduced data as POD time coefficients $\mathbf{a}(n) = [a_1(n), a_2(n), \ldots, a_{N_{\text{POD}}}(n)]^T$ for the data reduction via POD and as encoding space $\xi(n) = [\xi_1(n), \xi_2(n), \ldots, \xi_{N_{\text{CAE}}}(n)]^T$ for the one via CAE.

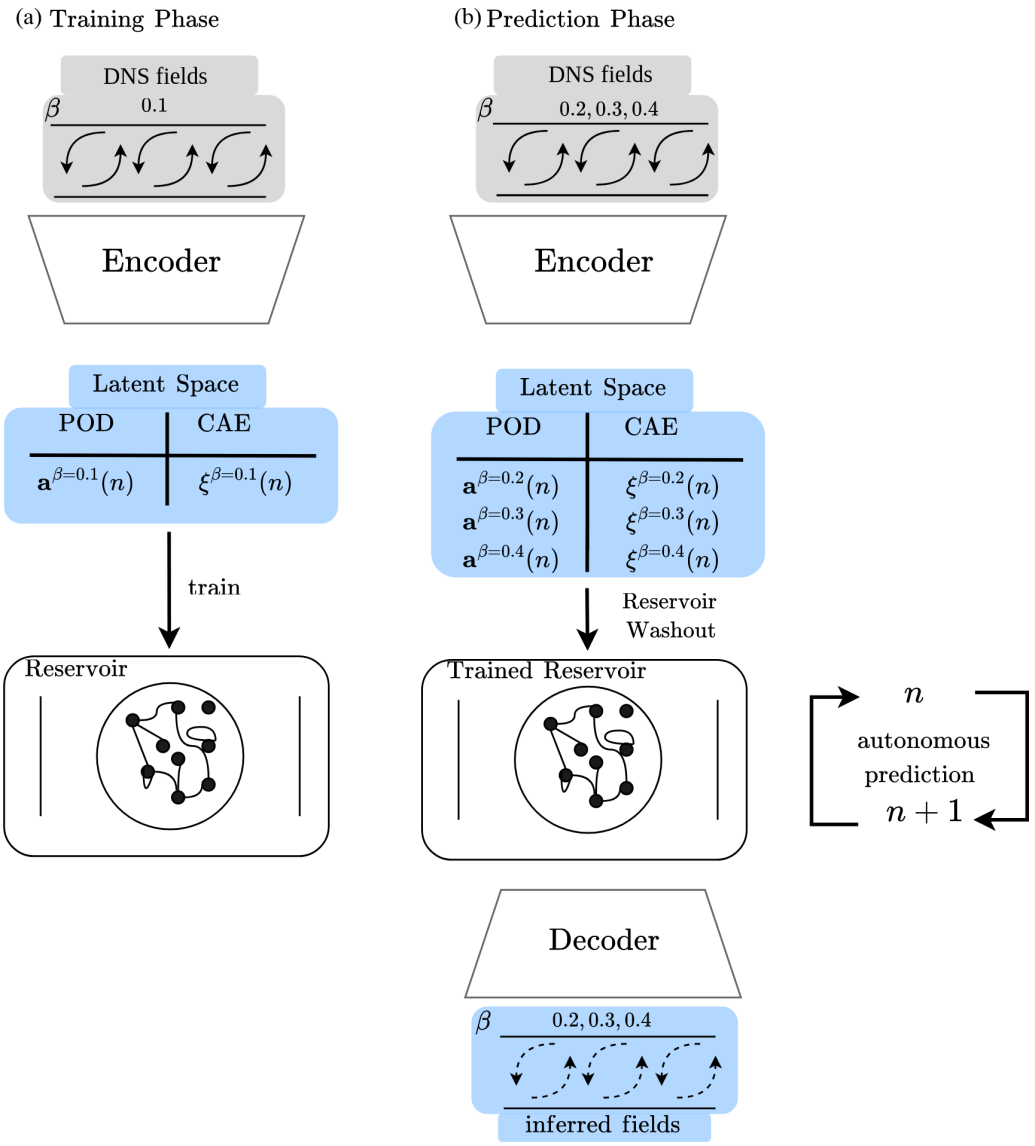

FIG. 7. Sketch of the transfer learning concept. (a) During the training phase, 700 snapshots of the simulation data for $\beta = 0.1$ are encoded into the latent space, either via the reduction by POD (denoted as $\mathbf{a}^{\beta=0.1}$) or via the one by a CAE (denoted as $\xi^{\beta=0.1}$). A reservoir is subsequently trained with the latent space. The network learns the dynamics; the optimal output weights are obtained. (b) In the prediction phase, the reservoir is then used to infer the dynamics of the target latent spaces at $\beta = 0.2, 0.3, 0.4$ and predicts either $\mathbf{a}^{\beta\neq0.1}$(POD) or $\xi^{\beta\neq0.1}$ (CAE). Snapshots of the convection flow can then be reconstructed and validated by the corresponding decoder to obtain fully resolved fields for the cases of $\beta = 0.2,\ 0.3$, and 0.4.

We construct the training data set for our ESN by taking 700 instances of $\mathbf{a}(n)$ or $\xi(n)$ of $\beta = 0.1$. We recall from Sec. II B that the spacing between outputs n and $n+1$ is 0.25 T_f. The total training length of $T_L = 700$ corresponds to 58 eddy turnover times. This timescale is given by $\tau_{\text{eddy}} = H/u_{\text{rms}} \approx 3T_f$ (see Table I). During this phase, the reservoir is trained to predict the respective next time instance of the POD expansion coefficients $\mathbf{a}(n+1)$ or the encoding variables $\xi(n+1)$; see, again, Eq. (25).

In the next paragraph, we explain how these trained ESNs can be used to predict the time coefficients (or encoding space) of the three cases with different heat flux parameter, namely, $\beta = 0.2,\ 0.3$, and 0.4. Finally, in Sec. III C, the individual prediction performance of both POD and CAE methods together with the ESN will be examined.

C. Prediction for unseen cases at $\beta = 0.2$–0.4

Once the ESN has learned to process the data in the latent space (which are obtained either by POD or CAE) for the case of $\beta = 0.1$, it is exposed to unseen data of the three CBL model cases, $\beta = 0.2,\ 0.3$, and 0.4, without further training adjustments. For this, we initialize a new reservoir state which is iterated for 50 steps following Eq. (22), i.e., the reservoir input for 50 time steps is either $\mathbf{a}^{\beta=0.2}$, $\mathbf{a}^{\beta=0.3}$, and $\mathbf{a}^{\beta=0.4}$ for POD-ESN or $\xi^{\beta=0.2}$, $\xi^{\beta=0.3}$, and $\xi^{\beta=0.4}$ for CAE-ESN, respectively. See, also, Fig. 8 where we display this crossover dynamics for one example. As discussed above, with this *washout phase*, we intend to transition to the run with a new β. Starting from this reservoir state, the ESN will autonomously predict $T_T = 700$ future time steps with its output weights that

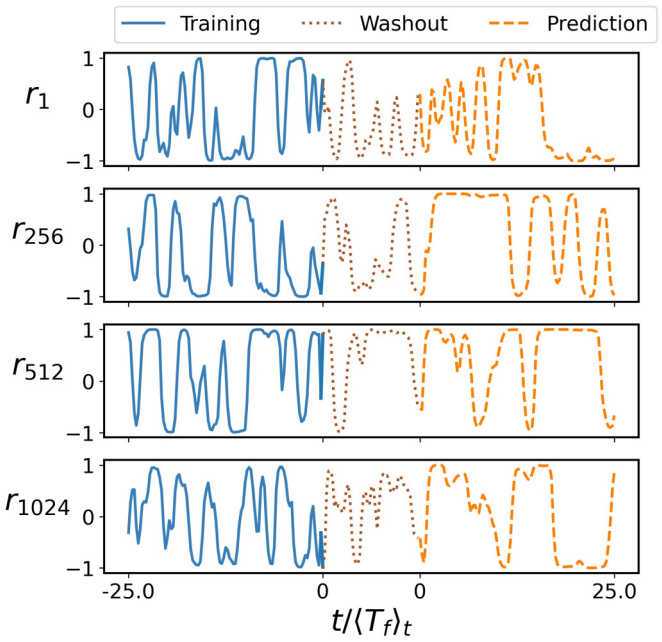

FIG. 8. Time evolution of individual components of the reservoir state vector $\mathbf{r}$ for inference of $\beta = 0.3$ in the POD-ESN algorithm. During training, the reservoir is exposed to $\mathbf{a}^{\beta=0.1}$. After the training phase, W^{out} is held fixed and the memory on the training inputs is cleared, while 50 inputs of $\mathbf{a}^{\beta=0.3}$ are given to the reservoir. Simultaneously, the reservoir builds up a memory of the unseen $\beta = 0.3$ case. Finally, the new initialized state and an initial input of $\mathbf{a}^{\beta=0.3}$ is used to start the autonomous prediction phase. Line styles for the different phases are given above.

TABLE II. Choice of optimal ESN hyperparameters h_*. These are the leaking rate γ, the regression parameter λ, the number of reservoir nodes, N_r, the reservoir density D, and the spectral radius ϱ of the reservoir network matrix W^{r}. The values were chosen according to a grid search. See Appendix C for more detailed information on the grid search. We also present the hyperparameters for the base case $\beta = 0.1$ for reference. We list the hyperparameter sets for each β that gave the lowest NARE with respect to the buoyancy flux. Note that the reservoir density D and the number of reservoir nodes, N_r, remain unchanged. Also, the spectral radius $\varrho > 1$ for several cases.

	β	γ	λ	N_{r}	D	ϱ
POD	0.1	0.8	0.5	1024	0.84	1.42
POD	0.2	0.8	0.5	1024	0.84	1.60
POD	0.3	0.8	0.5	1024	0.84	1.85
POD	0.4	0.9	0.5	1024	0.84	0.14
CAE	0.1	0.7	0.5	1024	0.84	0.25
CAE	0.2	0.4	0.5	1024	0.84	1.98
CAE	0.3	0.2	0.5	1024	0.84	0.81
CAE	0.4	0.1	0.5	1024	0.84	1.98

were learned for $\beta = 0.1$. We validate these predictions by a direct comparison with either $\mathbf{a}^{\beta=0.2}(n)$, $\mathbf{a}^{\beta=0.3}(n)$, $\mathbf{a}^{\beta=0.4}(n)$ or $\xi^{\beta=0.2}(n)$, $\xi^{\beta=0.3}(n)$, $\xi^{\beta=0.4}(n)$ with $n \in [1, T_T]$, respectively. For this, we apply the *mean squared prediction error* (MSE) which, e.g., for the specific case of $\beta = 0.2$, is given by

$$\text{MSE}_{\text{h}} = \frac{1}{T_T} \sum_{n=1}^{T_T} \|\hat{\mathbf{y}}(n) - \mathbf{a}^{\beta=0.2}(n)\|_2^2. \tag{27}$$

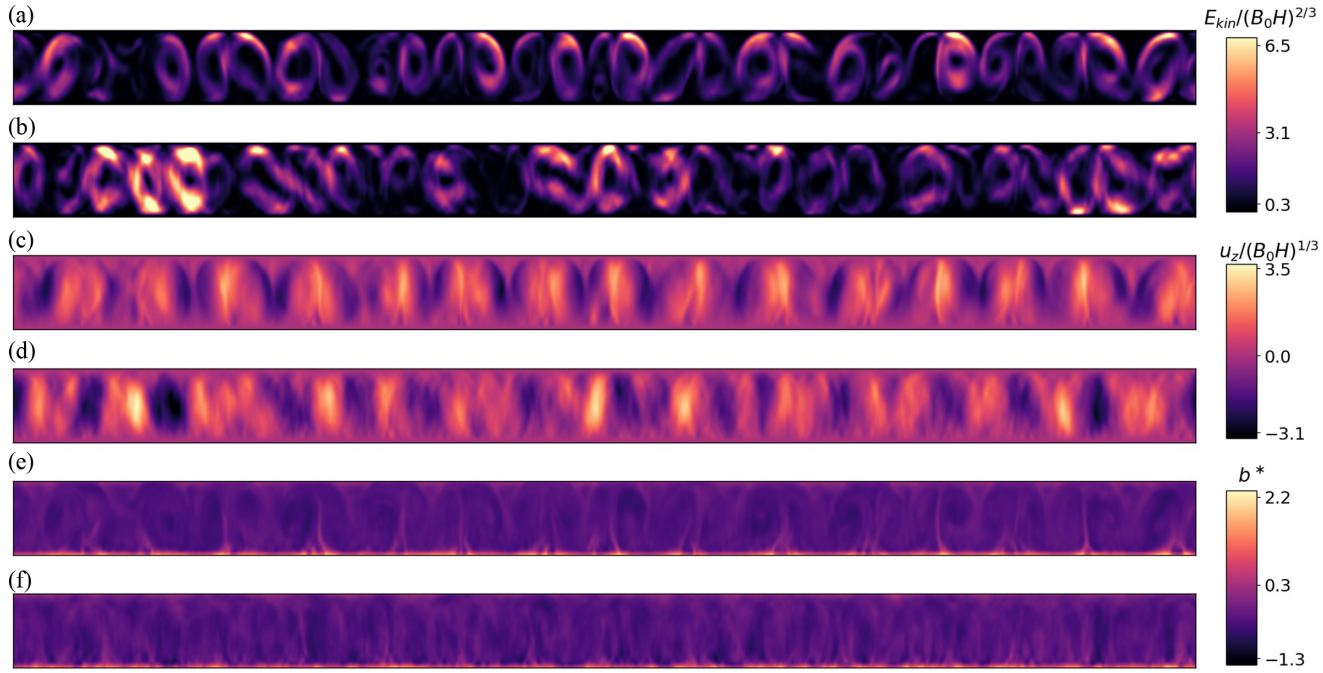

FIG. 9. POD case for inferring $\beta = 0.2$. Instantaneous snapshots of (a),(b) the local turbulent kinetic energy $E_{\text{kin}}(x, z, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) POD reconstructions with the most energetic N_{POD} modes of $\beta = 0.2$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

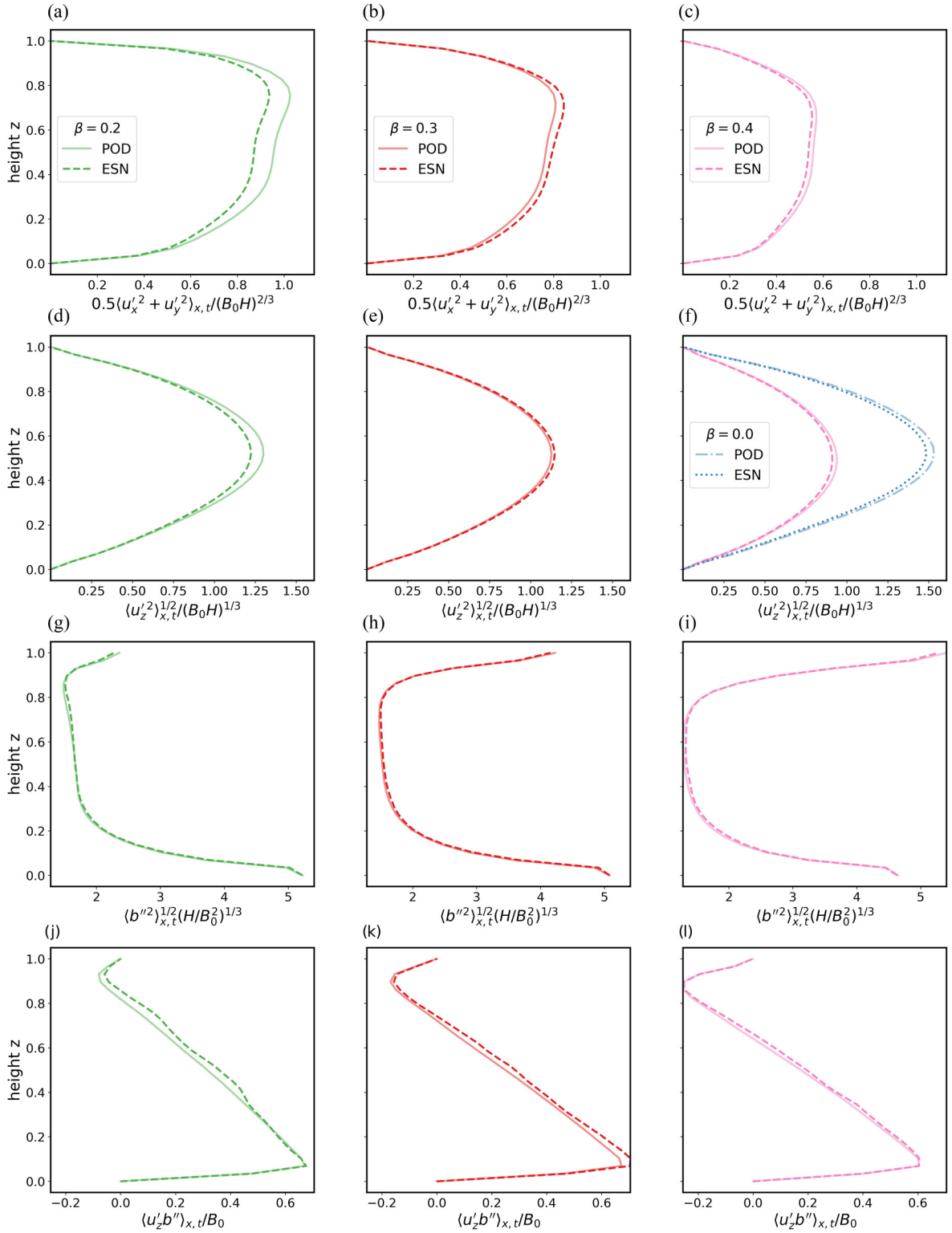

FIG. 10. Combined line-time average profiles of central quantities. (a)–(c) Turbulent kinetic energy, (d)–(f) root-mean-square profile of vertical velocity fluctuations, (g)–(i) root-mean-square buoyancy fluctuations, and (j)–(l) convective buoyancy flux. The ESN predictions (dashed lines) were chosen as to hold the median buoyancy flux NARE. They reproduce some low-order statistics of the truncated POD reconstruction (solid lines) of $\beta = 0.2$ in the first column, $\beta = 0.3$ in the second column, and $\beta = 0.4$ in the third column. For the quantity $\langle u_z'^2\rangle_{x,t}^{1/2}$, we also show the performance of the case of $\beta = 0.0$ with a different convection dynamics in (f).

TABLE III. Normalized average relative errors [see Eq. (28)] of the inferred line-time average profiles shown in Figs. 10 and 13 for the reduction by POD and CAE, respectively.

	β	$E_{h_*}[0.5\langle u_x'^2 + u_z'^2\rangle_{x,t}]$	$E_{h_*}[\langle u_z'^2\rangle_{x,t}]$	$E_{h_*}[\langle b''^2\rangle_{x,t}]$	$E_{h_*}[\langle u_z' b''\rangle_{x,t}]$
POD	0.0	0.034	0.026	0.003	0.010
POD	0.1	0.022	0.008	0.002	0.013
POD	0.2	0.033	0.024	0.002	0.018
POD	0.3	0.015	0.009	0.003	0.018
POD	0.4	0.014	0.013	0.003	0.012
CAE	0.0	0.094	0.057	0.007	0.035
CAE	0.1	0.005	0.006	0.011	0.028
CAE	0.2	0.019	0.028	0.013	0.057
CAE	0.3	0.012	0.018	0.018	0.038
CAE	0.4	0.080	0.045	0.017	0.029

In addition, we take the *normalized average relative error* (NARE) of the reconstructed fields u_z', b'', and $u_z'b''$. The definition follows the work of [22,61] and is given, for example, for $u_z'b''$ by

$$E_{\mathrm{h}}[\langle u_z' b''\rangle_{x,t}] = \frac{1}{C_{\max}} \int_0^1 \left| \langle u_z' b''\rangle_{x,t}^{\mathrm{ESN}}(z) - \langle u_z' b''\rangle_{x,t}^{\mathrm{POD}}(z) \right| dz, \tag{28}$$

with the constant

$$C_{\max} = 2 \max_{z\in[0,1]} \left(\left| \langle u_z' b''\rangle_{x,t}^{\mathrm{POD}} \right| \right). \tag{29}$$

The superscript indicates whether the field is reconstructed [see Eq. (A2)] from the N_{POD} expansion coefficients (POD) or the ESN predictions (ESN). This measure quantifies errors in the line-time average profiles of the physical fields. Similarly, one can define MSE and NARE for the CAE case by using ξ instead of $\mathbf{a}$, and the CAE instead of the POD reconstruction.

Our choice of the optimal ESN hyperparameters h_* is listed in Table II. We conducted grid searches of N, D, γ, and ϱ. See Appendix C for more details. For each setting, we additionally took 100 random realizations of the same reservoir setting and computed $\mathrm{MSE_h}$ and $E[u_z'b'']$. This sums up to 40 million individual training runs of the reservoir. The final setting h_* was chosen according to the lowest third quartile of $E[u_z'b'']$ of all 100 samples. We deliberately choose the third quartile over the median, as it assures robust reservoir outputs for different random weights W^{in}, W^{r} and therefore more reliable predictions. Furthermore, we choose the NARE of the buoyancy flux, due to its physical relevance, as opposed to the MSE. Moreover, it is comprised of two quantities which are prone to prediction errors.

1. Results for the POD-ESN algorithm

We reconstruct each component of the physical fields u_x, u_z, and b via Eq. (A2) using the decompositions (18)–(20) and (26). For the validation, we use the expansion coefficients of the most energetic N_{POD} modes of the corresponding data. For $\beta = 0.2$, instantaneous snapshots in the middle of the prediction phase (the time step is $n = 350$) of the *local* turbulent kinetic energy,

$$E_{\mathrm{kin}}(x,z,t) = \tfrac{1}{2}\left[u_x^2(x,z,t) + u_z^2(x,z,t)\right], \tag{30}$$

the vertical velocity component $u_z(x,z)$, and the normalized buoyancy $b^*(x,z)$ can be seen in Fig. 9. See, also, Appendix D for $\beta = 0.3$ and 0.4. The ground truth, i.e., the POD data, is shown for comparison. We find common features in the predicted and the validation fields. Even though some magnitudes deviate, roll patterns in the kinetic energy can be identified clearly in the prediction case. In the velocity field component, vertical up- and downdrafts can be clearly identified. Their width and shape differ slightly from the ground truth. Moreover, the thermal boundary layer at $z = 1$ is reproduced in the predicted buoyancy field. Thermal plumes, which detach primarily from the bottom wall, can also be identified. It is clear that some features are not perfectly reproduced, but the qualitative picture agrees fairly well. We emphasize that these results were obtained for one particular realization out of the 100 reservoirs with the same hyperparameter setting, which were taken typically. Nevertheless, both results are exemplary for their setting h_*, as they correspond to the median NARE of the buoyancy flux.

We now investigate the generalization capability of the reservoir by computing line-time average profiles $\langle\cdot\rangle_{x,t}$ of the fluctuations of the corresponding fields for $\beta = 0.2$, 0.3, and 0.4. These are important parameters of simulations of large-scale turbulence. The profiles are given in Fig. 10 and their corresponding NARE values are listed in Table III. We find that in this setting h_*, the average reservoir produces reasonable approximations to the profiles of the true low-order statistics of all three β values. Despite some deficiency in the profiles of turbulent kinetic energy and vertical velocity for $\beta = 0.3$, the asymmetry due to the boundary conditions is captured in all profiles. The buoyancy fluctuations are reproduced especially well. While the ESN reproduces the linear decrease of the convective buoyancy flux, $\langle u_z'b''\rangle_{x,t}^{\mathrm{ESN}}$, it overshoots near the bottom of the cell for $\beta = 0.2$ as well as in the upper cell for $\beta = 0.3$. Nevertheless, the inferred profiles match the ground truth to a reasonable extent.

Surprisingly, the profiles of the $\beta = 0.4$ case are also captured very well, even though the dynamics from the training and testing setup differ greatly, as discussed in Sec. II. This suggests that the ESN-POD algorithm is capable to generalize to substantially different convection setups. In order to test the generalization capabilities to another distinct convection system, we also investigated the adiabatic top lid case $\beta = 0.0$. It is found that even though the adiabatic top lid case exhibits

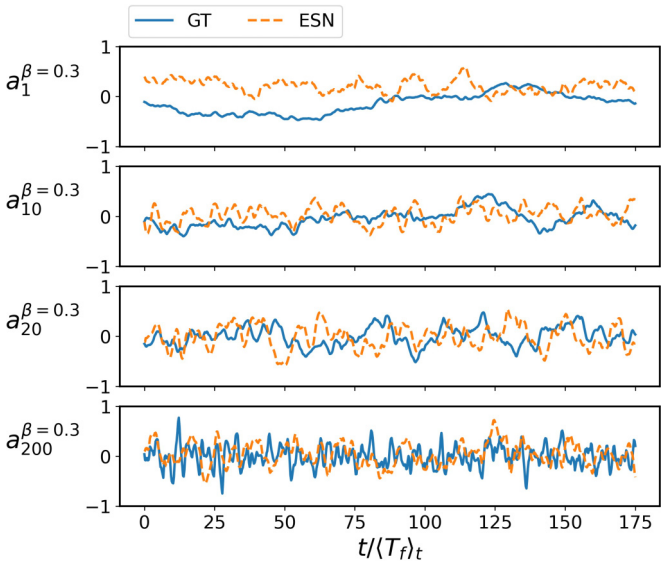

FIG. 11. Time series of selected POD expansion coefficients for the case of $\beta = 0.3$. We compare the ESN prediction and the ground truth (GT).

different convection dynamics, the trained ESN generalizes well for the POD case. For brevity, we only show the inferred root-mean-square fluctuations of the vertical velocity component in Fig. 10(f) and list all NARE values in Table III.

Overall, the ESN generalizes well to unseen convection data with similar boundary conditions, when using the low-order POD model. The inferred fields of $\beta = 0.2$–0.4 (see, also, Appendix D) reproduce features such as the thermal boundary layer, up- and downdrafts, as well as roll patterns. In the next section, we investigate how the ESN performs when we combine it with a trained convolutional autoencoder.

Finally, we show examples of the temporal evolution of selected POD expansion coefficients $a_i^{\beta=0.3}(t)$ in Fig. 11. It is seen that the time series of the expansion coefficients of the ground truth and the ESN output do not coincide for this unseen case. The characteristic frequency of $a_i^{\beta=0.3}(t)$ increases with growing index i for both ground truth and reconstruction. This is one reason why the statistical properties are eventually reproduced well.

2. *Results for the CAE-ESN algorithm*

By decoding the inferred latent spaces using Eq. (B2) and Eqs. (18)–(20) as well as Eq. (26), we reconstruct the fields u_x, u_z, and b. Figure 12 shows instantaneous snapshots of turbulent kinetic energy, vertical velocity, and normalized buoyancy of inferred fields (ESN) and ground truth (CAE). See, also, Appendix D for the cases $\beta = 0.3$ and 0.4. Here, we find predicted and true fields almost indistinguishable in terms of their features. Roll patterns, up- and downdrafts, as well as thermal plumes detaching from the bottom are reproduced very naturally. While the POD method introduces some deviations in the inferred fields, the autoencoder reproduces the small-scale features of the convection patterns well. Figure 13 shows the inferred line-time averaged profiles of the physical fields. Their corresponding NARE values are listed in Table III. Differently from the linear POD method, the CAE is trained by a gradient descent procedure, which introduces artifacts in the statistical profiles (solid lines). The loss of information in the encoder-decoder structure thus impacts the statistical features of the reconstructed flow. As a conse-

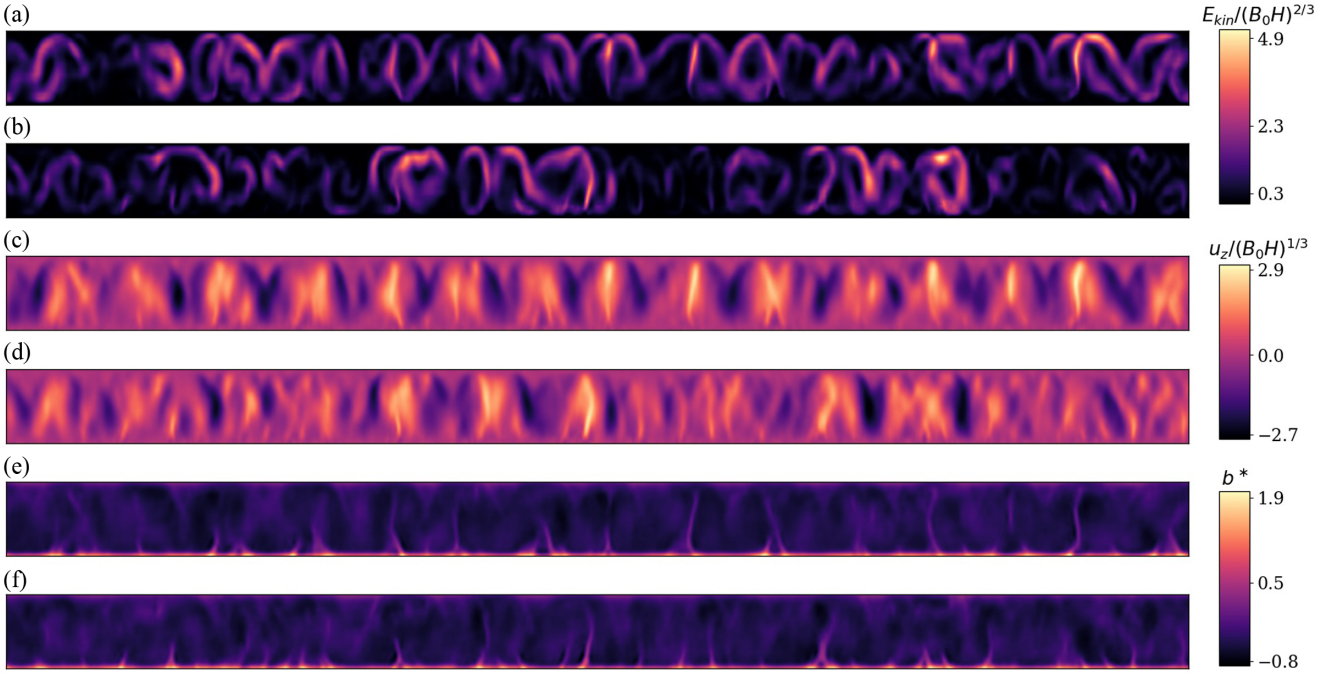

FIG. 12. CAE case for inferring $\beta = 0.2$. Instantaneous snapshots of (a),(b) the turbulent kinetic energy $E_{\text{kin}}(x, z, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) CAE reconstructions of $\beta = 0.2$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

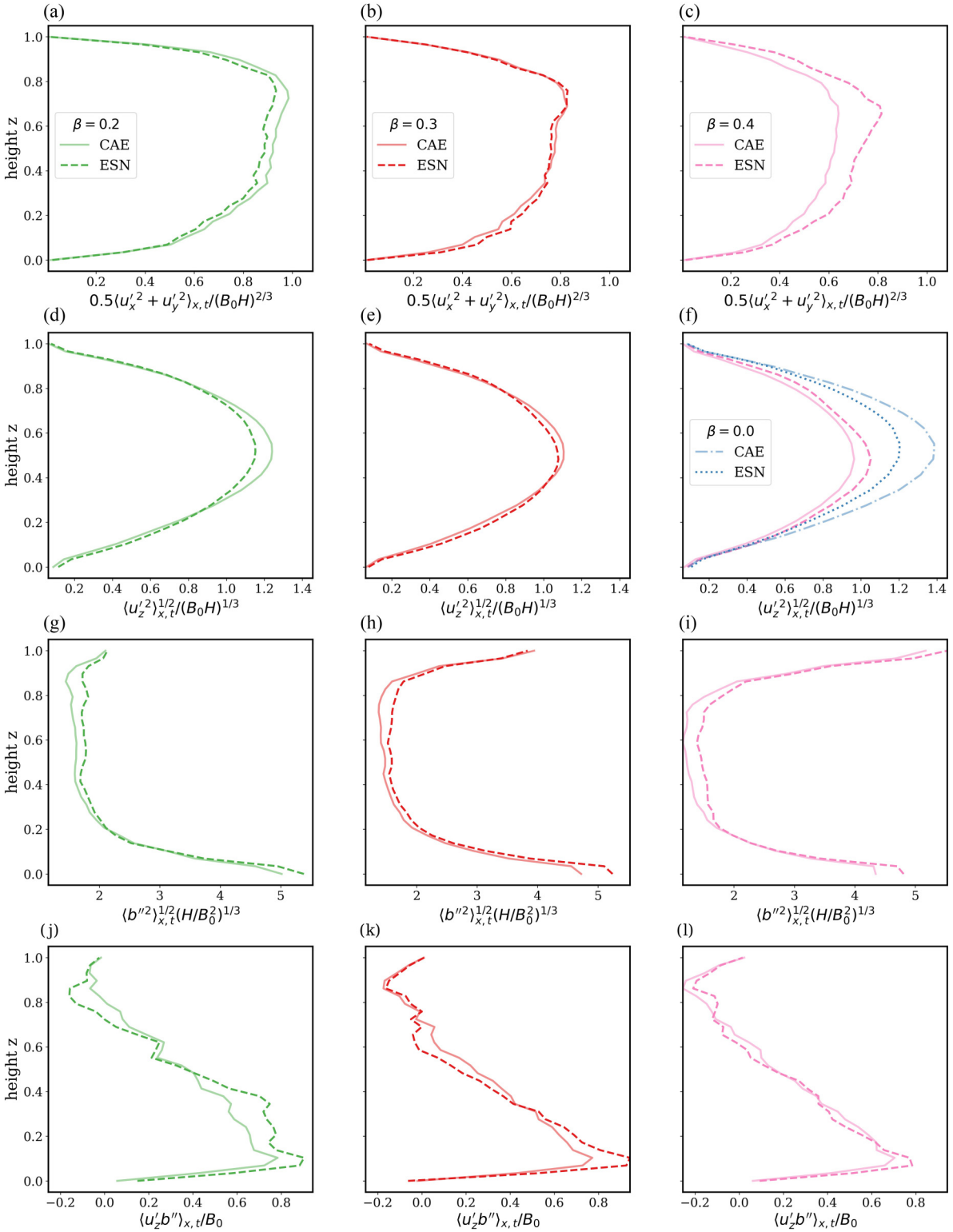

FIG. 13. Combined line-time average profiles of central quantities. (a)–(c) Turbulent kinetic energy, (d)–(f) root-mean-square profile of vertical velocity fluctuations, (g)–(i) root-mean-square buoyancy fluctuations, and (j)–(l) convective buoyancy flux. The autoencoder introduces artifacts in the statistical profiles (solid lines) of $\beta = 0.2$ in the first column, $\beta = 0.3$ in the second column, and $\beta = 0.4$ in the third column. The ESN predictions (dashed lines) were chosen such to hold the median buoyancy flux with respect to NARE. For the quantity $\langle u_z'^2\rangle_{x,t}^{1/2}$, we also show the performance of the case of $\beta = 0.0$ with a different convection dynamics in (f).

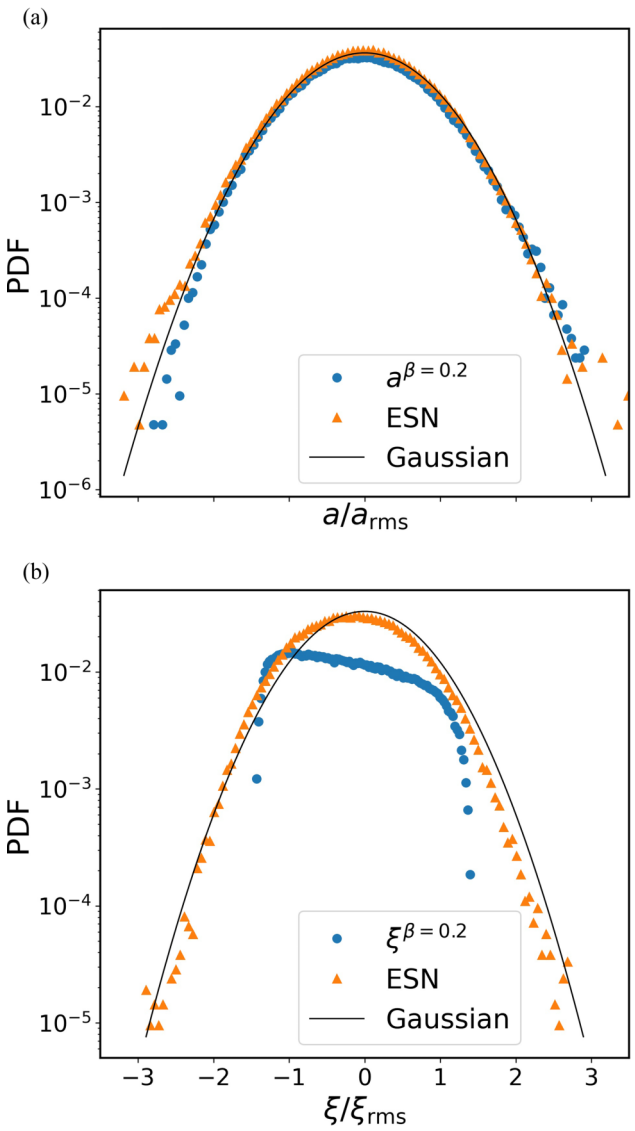

FIG. 14. Probability density functions (PDF) of the ESN predictions for $\beta = 0.2$ in comparison to the output of the data reduction module, (a) the time coefficients $\mathbf{a}^{\beta=0.2}$ of the POD approach, and (b) the latent space state vector $\boldsymbol{\xi}^{\beta=0.2}$ of the CAE approach, respectively. While the ESN captures the symmetric Gaussian PDF of the POD time coefficients, the asymmetric PDF obtained from the encoder is not captured. Note that all quantities are normalized by their corresponding root-mean-square values, such that they are directly comparable to a Gaussian PDF which is indicated by a solid line.

quence, larger magnitudes of this measure can be observed for most entries of the table.

It is possible that this error can be reduced by introducing an additional term to the loss function of the CAE that penalizes large deviations from the mean profiles. This is not applied here. The biggest artifacts can be seen in the buoyancy flux in Figs. 13(j)–13(l). Nevertheless, we find the differences acceptable, as the asymmetry and shape of the true profiles are retained. The reservoir manages to reproduce the overall trend of the line-time average profiles. It is seen that the $\langle u_z' b'' \rangle_{x,t}$ profiles are less well predicted in the lower boundary layer, where the maxima are overestimated by the reservoir. Furthermore, we also find that the CAE-ESN algorithm manages to reproduce the strongly fluctuating case of $\beta = 0.4$ fairly well. The velocity and buoyancy fluctuations are slightly underestimated in the middle of the domain.

Additionally, we find that the generalization to the adiabatic top lid case at $\beta = 0$ overestimates the amplitudes of the profiles of turbulent kinetic energy (not shown) and vertical velocity fluctuations; see Fig. 13(b) and Table III. The reason for these deviations might be that a CAE always extracts prominent features for the latent space which correspond to the strong localized up- and downdrafts in the convection layer. At the end of this section, we compare the probability density function (PDF) of the ESN output to the target latent spaces for the case $\beta = 0.2$ to better understand the dynamics in the latent space. The POD-ESN case in Fig. 14(a) exhibits symmetric Gaussian PDFs which are reproduced by the echo state network prediction. The results of the CAE-ESN case in Fig. 14(b), on the other hand, result in a strongly skewed distribution that seems to be connected to sharp edges of extracted features. These features are not captured in the PDF of the ESN prediction, even though the distribution is also slightly skewed. This may explain the poor ESN performance in the CAE approach. The complex latent space of the CAE might pose a harder learning problem for the ESN.

IV. CONCLUSIONS AND OUTLOOK

In this work, we explored the generalization property of a machine learning method applied to a more complex convection flow than standard RBC. In particular, we considered echo state network algorithms applied to two-dimensional convection with different constant-flux boundary conditions at the top and bottom. To this end, we impose buoyancy fluxes at the top and bottom boundaries which can be understood as entrainment from the top and surface heating from the bottom in an atmospheric convective boundary layer. The model is hence characterized by the buoyancy flux ratio β, beside Rayleigh and Prandtl numbers. An increasing value of β quantifies a counterheating that stabilizes the top layer and results in negative values of the mean convective buoyancy flux close to the top boundary. Thus our model resembles properties that are absent in a standard Rayleigh-Bénard setup with uniform temperatures at the top and bottom. In particular, the top-down symmetry of the boundary layers is broken; in this respect, the present model is similar to a complex non-Boussinesq convection flow. On the one hand, it is thus an ideal testing bed for dynamic parametrizations of the buoyancy flux and its low-order moments by machine learning algorithms. On the other hand, it is still a simplification of an atmospheric layer, in particular in view to its two-dimensionality.

We conducted a series of direct numerical simulations for values of β that vary between 0 and 0.4, a range that represents mid-day atmospheric conditions over land. The spectrum reaches from an adiabatic top boundary without incoming and outgoing flux ($\beta = 0$) to a rare case with a very strong buoyancy gradient at the top ($\beta = 0.4$). The five simulations result in flows with distinct features which are not common in Rayleigh-Bénard convection. The mean buoyancy is nearly constant throughout the middle of the domain, which

resembles a mixed layer inside the convective cell. Further, positive buoyancy gradients at the top and a linear decline with height of the covariance of vertical velocity and buoyancy can be observed, both features which are also observed in an atmospheric boundary layer. The five simulations also display different dynamics and convection patterns, which demonstrates the impact of the incoming top flux. These differences become evident when considering the low-order statistics of the buoyancy and its vertical flux. As β increases, so does the thickness and magnitude of the stable layer at the top of the convection cell, and the intensity of the buoyancy fluctuations. At $\beta = 0.4$, the convective motion is confined to a small region near the bottom plate only, while the stably stratified fluid at the top inhibits a significant fluid motion in the upper domain.

At the core of our combined POD/CAE–ESN machine learning method is a recurrent neural network in the form of an echo state network to predict the dynamics and low-order statistics for the *unseen* simulation data at $\beta = 0.2$, 0.3, and 0.4. The echo state network is trained with simulation data records at $\beta = 0.1$. In this way, we can explore the generalization properties of the recurrent neural network or, in other words, the performance of the machine learning algorithm to unseen data with different physical parameters.

We use two common approaches to reduce the amount of DNS data for the prediction task: (1) the proper orthogonal decomposition and the (2) convolutional autoencoder. Both methods reduce the data to 300 degrees of freedom per snapshot. We find that the training of the echo state network with data of the low-magnitude flux ($\beta = 0.1$) at the top yields good approximations of the dynamics of the higher-magnitude turbulent flux cases at $\beta = 0.2$, 0.3, and 0.4. This is the case for both data reduction methods. We are also able to reconstruct velocity and buoyancy fields very well. This is in line with a low-order statistics of these fields which is also properly reconstructed, for example, for the vertical profiles of the buoyancy flux. In particular, the features of the $\beta = 0.4$ case are captured by the model, even though the training and target setups exhibit very different dynamics. We also studied the case of zero incoming flux at the top boundary, $\beta = 0.0$, to test the generalizability of our machine algorithm for a more distinct convection dynamics. The results indicate a good performance for the POD-ESN case and a somewhat poorer quantitative agreement for the CAE-ESN case.

We point out that the two low-order models differ in their compression technique and hence yield different performances, when combined with the reservoir computing model in the latent space. While the POD preserves line-time average profiles, the autoencoder introduces small artifacts to the statistics. Also, the quality of the predicted spatial features differs among the methods. The predicted POD time coefficients capture coarse convection features, while the convolutional autoencoder reproduces the natural convection patterns, i.e., the prominent features, very well. Moreover, the POD-ESN case reproduces the statistics in the latent space well, while the distribution of the latent space variables of the CAE-ESN case becomes asymmetric. This can affect the training and generalization capability of the ESN, which requires further investigation. We can conclude that for our setup, the data emerging from one case with constant flux boundary conditions can be used to infer at least statistical and spatial features of three different cases with different conditions. The echo state network can thus serve as a reduced-order and scalable model that generates the appropriate turbulence statistics without solving the underlying Navier-Stokes equation of the flow.

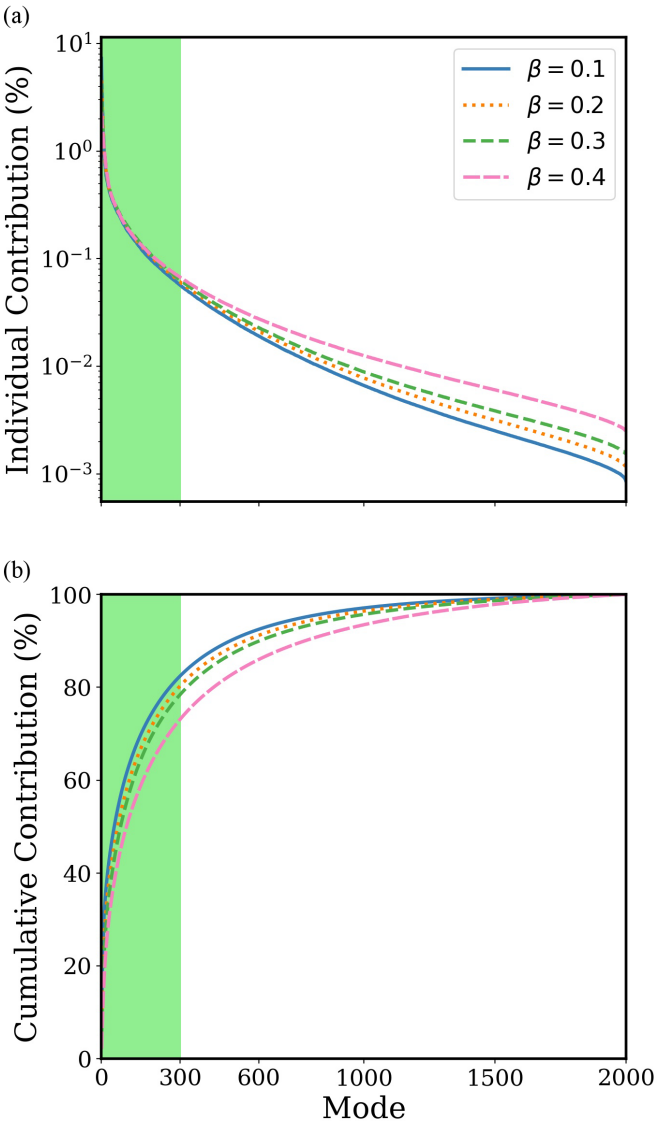

FIG. 15. Spectrum of eigenvalues of the POD modes. (a) Individual contribution of each POD mode to the total energy. (b) Cumulative contribution. The green shaded area marks the contribution of the first 300 modes.

The present study can be considered as one step in the development of efficient reduced-order models of convection dynamics by machine learning methods. Several directions for future research are possible from this point. First, an extension to the three-dimensional case is desirable. This requires a stronger reduction of the data, which can be achieved by even deeper convolutional encoder-decoder networks in combination with spatial filtering of the direct numerical simulation data. Such a reduction could lead to a dynamical model of a recent approach by [19], which reduced the convective turbulent heat transport across a convection layer to a dynamic planar network. We also suggest to incorporate physical laws or known flow properties in the training routine of the

TABLE IV. Mean squared error loss of the CAE reconstruction after 1200 epochs of training. Each CAE was trained on 8000 snapshots of (u_x', u_z', b'') for their corresponding β. The validation loss was computed on 2000 different snapshots.

β	0.0	0.1	0.2	0.3	0.4
Training loss (%)	6.0×10^{-4}	6.2×10^{-4}	7.1×10^{-4}	8.1×10^{-4}	5.4×10^{-3}
Validation loss (%)	3.1×10^{-3}	2.66×10^{-3}	2.76×10^{-3}	2.72×10^{-3}	1.37×10^{-2}

autoencoder, as mere mimicking of the input fields produces artifacts in the statistical features of the reconstruction. This has been done in several works, e.g., in Ref. [62]. Moreover, one should keep the balance between the demand of physical reality and computational expense as to keep the use of a low-order model meaningful.

The approach presented in this manuscript was only concerned with the generalization capability of the recurrent neural network. Future works will also include the data reduction method into the analysis of the generalization capabilities. Here, one might follow the conventional transfer learning approach of neural networks [63], where a single neural network is pretrained such that the actual training to the different physical systems does not require as much data, as if one would train a CAE for all systems, as it is done in this work. Furthermore, by definition, neural networks are not designed to process data that live on a continuum of different lengths and times, a property which is immanent to turbulent flows. Architectures which can represent the multiscale nature of turbulence are required. Studies in these directions are currently underway and will be reported elsewhere.

Code used for this work available on GitHub [58].

ACKNOWLEDGMENTS

This work is supported by Project No. P2018-02-001 "DeepTurb – Deep Learning in and of Turbulence" of the Carl Zeiss Foundation. Partial support for J.P.M. was provided by Grant No. PID2019-105162RB-I00 funded by MCIN/AEI/10.13039/501100011033. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. [64] for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at the Jülich Supercomputing Centre (JSC).

APPENDIX A: PROPER ORTHOGONAL DECOMPOSITION

Technical details of both data reduction techniques are discussed in the following appendices to keep the manuscript self-contained. We apply the POD in the form of the method of snapshots [46,47] on the vector $\mathbf{g} = (u_x', u_z', b'')^T$, such that its kth component can be written as

$$g_k(x, z, t) = \sum_{i=1}^{N_{\text{dof}}} a_i(t)\Phi_i^{(k)}(x, z). \tag{A1}$$

This linear method decomposes the scalar field g_k into time-dependent coefficients $a_i(t)$ and spatial modes $\Phi_i^{(k)}(x, z)$, such that the truncation error is minimized. The degrees of freedom N_{dof} can then be reduced, by taking only $N_{\text{POD}} \ll N_{\text{dof}}$ modes and coefficients with the largest variance into account,

$$g_k(x, z, t) \approx \sum_{i=1}^{N_{\text{POD}}} a_i(t)\Phi_i^{(k)}(x, z). \tag{A2}$$

As mentioned above, we consider the $N_{\text{POD}} = 300$ most energetic POD time coefficients as the input for the ESN. The cumulative contributions of the first $N_{\text{POD}} = 300$ POD modes for the four cases of $\beta = 0.1$, 0.2, 0.3, and 0.4 capture then 82.4%, 80.2%, 78.4%, and 73% of the original total energy (kinetic energy plus temperature variance), respectively, see also Fig. 15.

APPENDIX B: CONVOLUTIONAL AUTOENCODER

An autoencoder is a feed-forward neural network which is trained to reproduce its network input $\mathbf{g}$ as network output [31,48]. Since the network should not copy its inputs to the output layer only, an intermediate bottleneck structure is introduced, such that the original information is compressed to an encoding or latent low-dimensional space. Therefore, the autoencoder consists of two parts which are trained as one network. The encoder $\mathbf{f}$ compresses the high-dimensional inputs to a low-dimensional representation,

$$\xi = \mathbf{f}_{\theta_{\text{encoder}}}(\mathbf{g}), \tag{B1}$$

where $\xi \in \mathbb{R}^{N_{\text{CAE}}}$ is the encoding or latent space and θ_{encoder} includes all trainable weights and biases of the encoder network.

TABLE V. Size of convolutional channels, kernel, and max pooling (MP) kernel for each layer in the autoencoder network. The channels are given in the form (input channel, output channel), while both the convolutional and MP kernel are given by (height, width). The shape of the input data was (3,30,720), where 3 stands for the included turbulence fields.

Layer	Conv. No. 1	Conv. No. 2	Conv. No. 3	Conv. No. 4	Conv. No. 5	Conv. No. 6	Conv. No. 7	Conv. No. 8	Conv. No. 9
Channels	(3,8)	(8,16)	(16,16)	(16,32)	(32,16)	(16,16)	(16,8)	(8,3)	(3,3)
Kernel	(7,7)	(5,5)	(3,3)	(3,3)	(3,3)	(3,3)	(3,3)	(5,5)	(7,7)
MP kernel	(2,1)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,1)	

The decoder $\mathbf{h}$ then attempts to decode the encoded latent space and reconstruct the original information,

$$\mathbf{g}^{\mathrm{AE}} = \mathbf{h}_{\theta_{\mathrm{decoder}}}(\xi) = \mathbf{h}_{\theta_{\mathrm{decoder}}}[\mathbf{f}_{\theta_{\mathrm{encoder}}}(\mathbf{g})]. \tag{B2}$$

Here, $\mathbf{g}^{\mathrm{AE}}$ is the autoencoder reconstruction and $\theta_{\mathrm{decoder}}$ includes all trainable weights and biases of the decoder network. We use a convolutional autoencoder (CAE) which makes use of convolutional layers that have proven to be extremely useful in pattern detection and classification of images [65]. While the ξ can be understood as a low-dimensional representation of the input, similar to the POD time coefficients $\mathbf{a}$, the trained weights and biases correspond to the POD spatial modes which contain information on how to decode the latent space. The training of the CAE requires backpropagation of errors through the convolutional networks. An optimally working CAE minimizes the difference between original input and final output, $\mathbf{g}^{\mathrm{AE}} \approx \mathbf{g}$.

As for the POD approach, we take snapshots of $\mathbf{g} = (u'_x, u'_z, b'')^T$ of $\beta = 0.1$–0.4 as input for their own CAE. Finally, one can use the trained encoder to translate the flow dynamics into dynamics of the latent space $\xi(t)$. We choose an encoding dimension of $N_{\mathrm{CAE}} = 300$ and train the network with 8000 snapshots of $\mathbf{g}$ and use 2000 further snapshots to validate its performance. The training and validation mean square error loss of each CAE is listed in Table IV. Out of the 2000, to the CAE unseen snapshots, we sample the 700 time steps of $\xi^{\beta=0.1}(t)$, used for training the ESN, and 700 time steps of $\xi^{\beta=0.2}(t)$ and $\xi^{\beta=0.3}(t)$, used for validation of the ESN predictions.

We use a CAE with four convolutional layers and one dense layer in the encoder and five convolutional layers and one dense layer in the decoder. Except for the last layer in the decoder, each convolutional layer is complemented by a max-pooling (MP) operation, in order to downsample the input data. Further, all layers are followed by a batch normalization and dropout layer. We find that batch normalization stabilizes the training process and dropout reduces the effect of overfitting, where the neural network shows poor performance on the validation data. The activation function of the last layer in both the encoder and decoder was sigmoid, while all other layers were followed by a *parametric rectified linear unit* (PReLU) [66]. The channel size, as well as convolutional and max pooling kernels, are listed in Table V. Using this architecture, the total number of trainable weights and biases amounts to 4.96×10^6.

The autoencoder is trained using the *ADAM* optimizer [67] with a learning rate 10^{-5}, batch size 64, and a L_2-norm penalty term with penalty parameter 10^{-6}. The loss function that was minimized was chosen to be the mean square error between the input and output fields. Moreover, the input data were scaled to the range [0, 1] before they were passed to the input layer of the CAE. Finally, each network was trained for 1200 epochs on 2 GPUs and took about 106 min to finish.

TABLE VI. ESN grid search range of each hyperparameter that was studied. The number of samples indicates how many different values of each hyperparameter were studied.

	N,	D,	γ	ϱ
Range	[512, 4096]	[0.1, 1.0]	[0.1, 1.0]	[0.1, 2.0]
No. samples	4	100	10	100

APPENDIX C: ECHO STATE NETWORK GRID SEARCH PROCEDURE

In order to find an optimal reservoir for both reduction methods and both β values, we conducted grid searches on four important reservoir hyperparameters, namely, N, D, γ, and ϱ, when training the case of $\beta = 0.1$. The range and number of different values of each hyperparameter study are listed in Table VI.

APPENDIX D: FIELDS FOR $\beta = 0.3$ AND 0.4

For completeness, we show in Figs. 16–19 the results for $\beta = 0.3$ and $\beta = 0.4$ that correspond to those for the case $\beta = 0.2$ in the main text. See Figs. 9 and 12 for comparison.

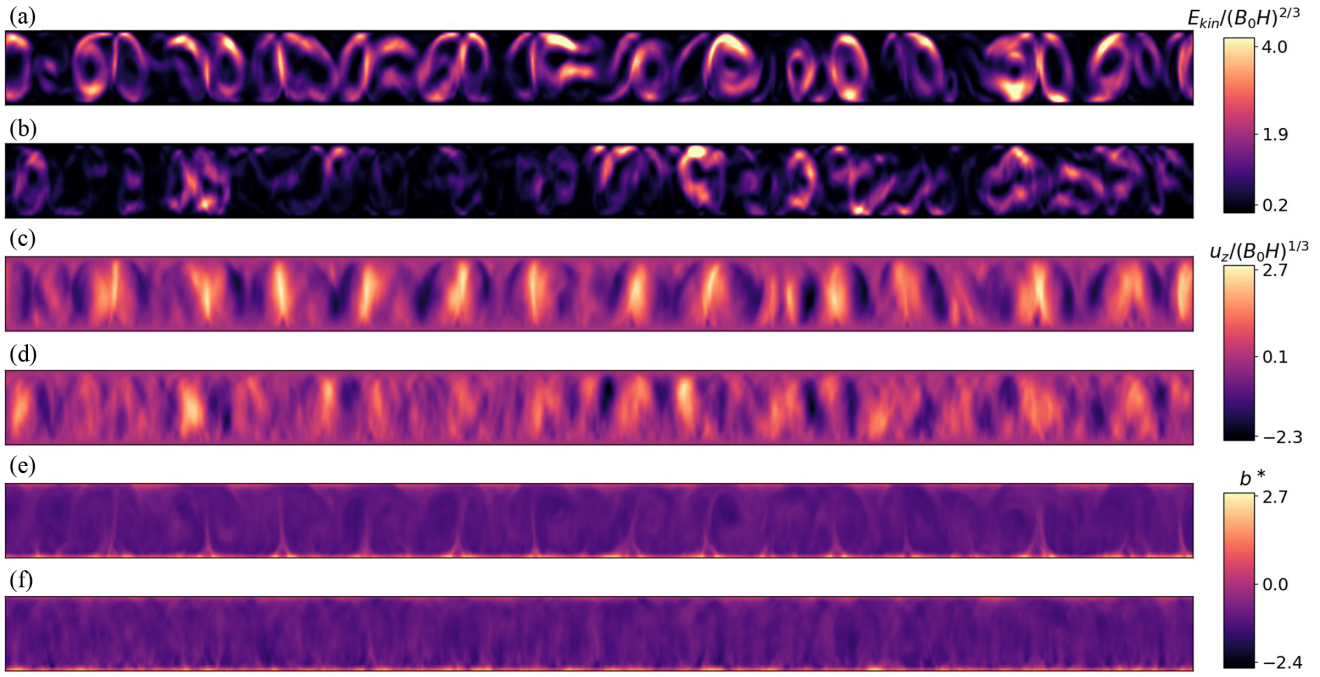

FIG. 16. POD case for inferring $\beta = 0.3$. Instantaneous snapshots of (a),(b) the local turbulent kinetic energy $E_{\mathrm{kin}}(x, y, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) POD reconstructions with the most energetic N_{POD} modes of $\beta = 0.3$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

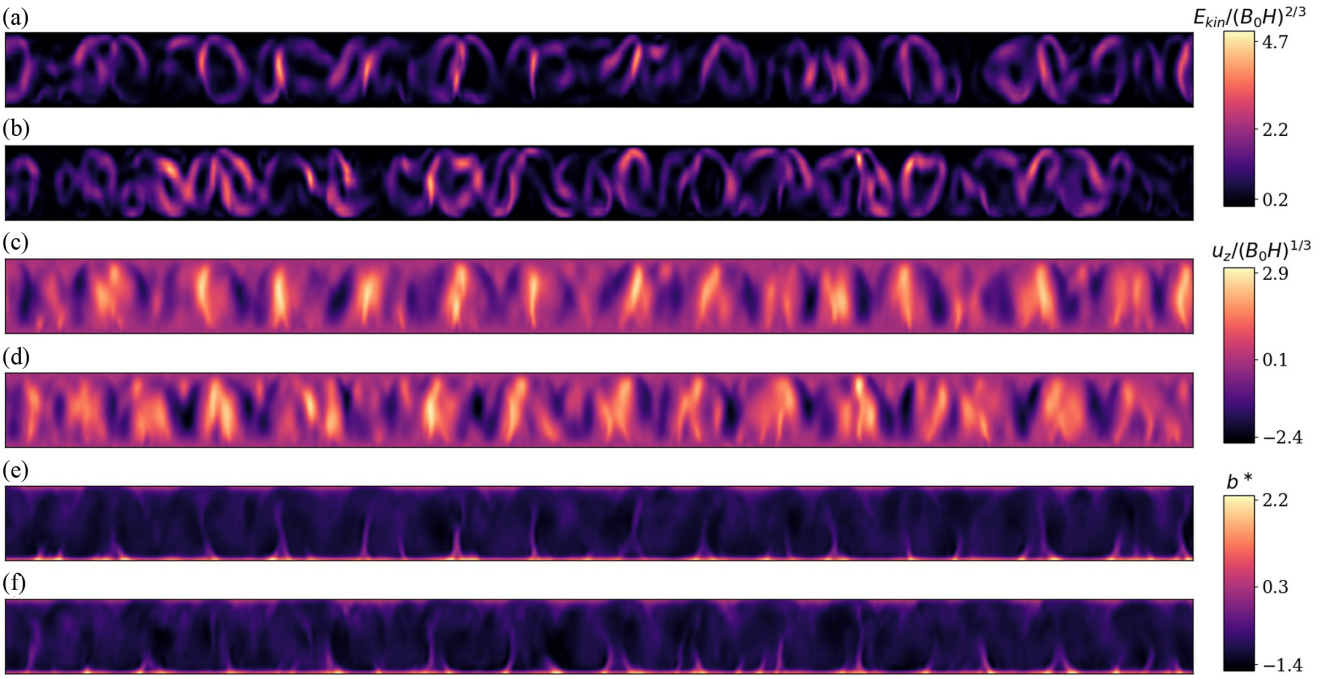

FIG. 17. CAE case for inferring $\beta = 0.3$. Instantaneous snapshots of (a),(b) the turbulent kinetic energy $E_{\mathrm{kin}}(x, y, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) CAE reconstructions of $\beta = 0.3$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

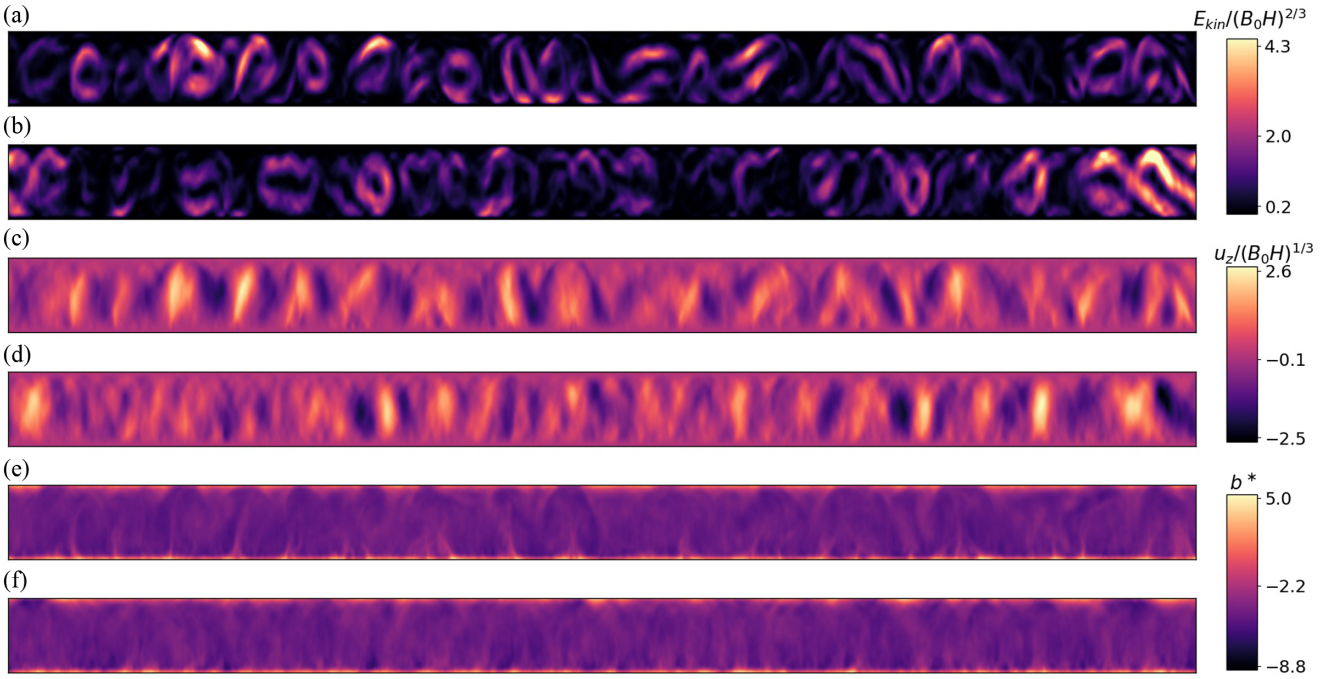

FIG. 18. POD case for inferring $\beta = 0.4$. Instantaneous snapshots of (a),(b) the local turbulent kinetic energy $E_{\mathrm{kin}}(x, y, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) POD reconstructions with the most energetic N_{POD} modes of $\beta = 0.4$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

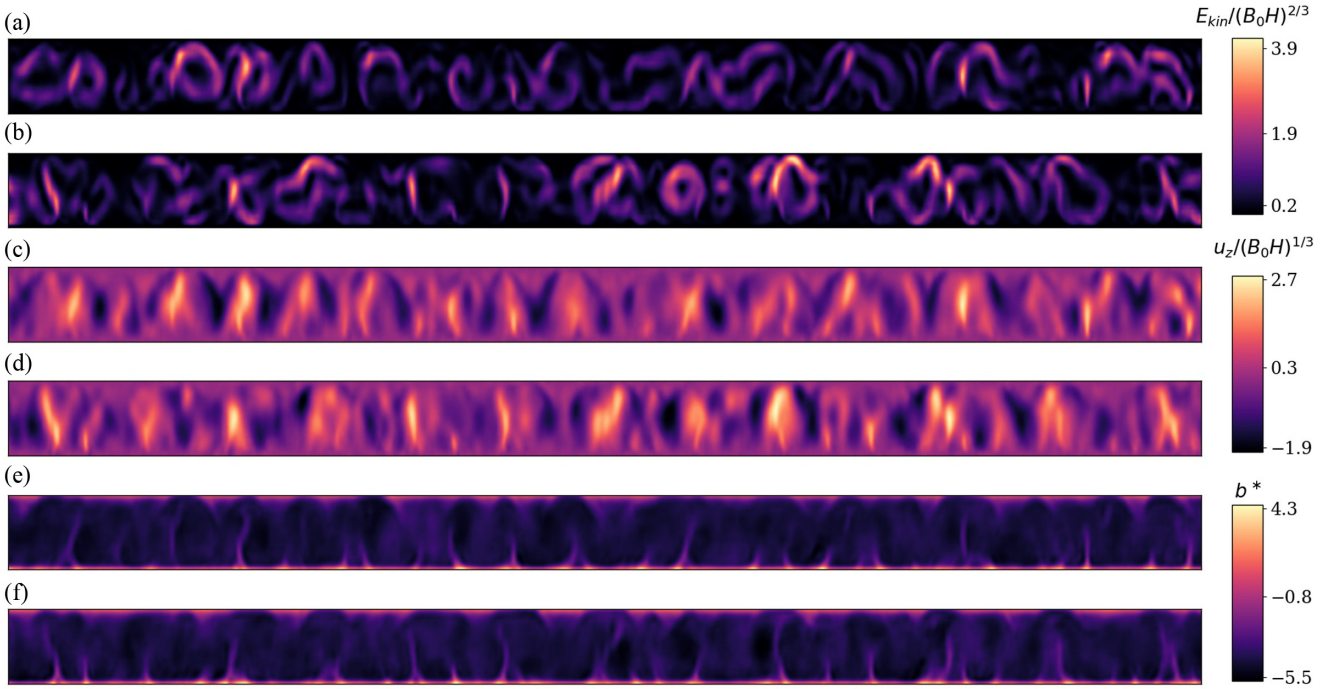

FIG. 19. CAE case for inferring $\beta = 0.4$. Instantaneous snapshots of (a),(b) the turbulent kinetic energy $E_{\mathrm{kin}}(x, y, t_0)$, (c),(d) the vertical velocity component u_z, and (e),(f) the normalized buoyancy b^*, at time step $n = 350$ in the prediction phase. (a),(c),(e) CAE reconstructions of $\beta = 0.4$ (validation snapshot); (b),(d),(f) the corresponding ESN predictions.

[1] J. N. Kutz, J. Fluid Mech. **814**, 1 (2017).
[2] M. P. Brenner, J. D. Eldredge, and J. B. Freund, Phys. Rev. Fluids **4**, 100501 (2019).
[3] K. Duraisamy, G. Iaccarino, and H. Xiao, Annu. Rev. Fluid Mech. **51**, 357 (2019).
[4] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Annu. Rev. Fluid Mech. **52**, 477 (2020).
[5] S. Pandey, J. Schumacher, and K. R. Sreenivasan, J. Turbul. **21**, 567 (2020).
[6] A. Beck and M. Kurz, GAMM-Mitteilungen **44**, e202100002 (2021).
[7] J. Ling, A. Kurzawski, and J. Templeton, J. Fluid Mech. **807**, 155 (2016).
[8] A. Beck, D. Flad, and C. Munz, J. Comput. Phys. **398**, 108910 (2019).
[9] G. Novati, H. de Laroussilhe, and P. Koumoutsakos, Nat. Mach. Intell. **3**, 87 (2021).
[10] A. J. Linot and M. D. Graham, Phys. Rev. E **101**, 062209 (2020).
[11] K. Zeng and M. D. Graham, Phys. Rev. E **104**, 014210 (2021).
[12] M. Buzzicotti, F. Bonaccorso, P. C. Di Leoni, and L. Biferale, Phys. Rev. Fluids **6**, 050503 (2021).
[13] K. Fukami, K. Fukagata, and K. Taira, J. Fluid Mech. **909**, A9 (2021).
[14] C. Lagemann, K. Lagemann, S. Mukherjee, and W. Schröder, Nat. Mach. Intell. **3**, 641 (2021).
[15] E. Hamidreza, V. Hadi, N. M. Hossein, and E. Vahid, Phys. Fluids **32**, 105104 (2020).
[16] K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata, Theor. Comput. Fluid Dyn. **34**, 367 (2020).
[17] R. Maulik, B. Lusch, and P. Balaprakash, Phys. Fluids **33**, 037106 (2021).
[18] P. R. Vlachas, G. Arampatzis, C. Uhler, and P. Koumoutsakos, Nat. Mach. Intell. **4**, 359 (2022).
[19] E. Fonda, A. Pandey, J. Schumacher, and K. R. Sreenivasan, Proc. Natl. Acad. Sci. USA **116**, 8667 (2019).
[20] L. Agasthya, P. C. D. Leoni, and L. Biferale, Phys. Fluids **34**, 015128 (2022).
[21] G. Beintema, A. Corbetta, L. Biferale, and F. Toschi, J. Turbul. **21**, 585 (2020).
[22] S. Pandey and J. Schumacher, Phys. Rev. Fluids **5**, 113506 (2020).
[23] F. Heyder and J. Schumacher, Phys. Rev. E **103**, 053107 (2021).
[24] S. Pandey, P. Teutsch, P. Mäder, and J. Schumacher, Phys. Fluids **34**, 045106 (2022).
[25] V. Valori, R. Kräuter, and J. Schumacher, Phys. Rev. Res. **4**, 023180 (2022).
[26] J. C. Wyngaard, *Turbulence in the Atmosphere* (Cambridge University Press, Cambridge, UK, 2010).
[27] J. P. Mellado, Annu. Rev. Fluid Mech. **49**, 145 (2017).
[28] L. Zanna and T. Bolton, Geophys. Res. Lett. **47**, e2020GL088376 (2020).
[29] C. Wang, G. Tang, and P. Gentine, Geophys. Res. Lett. **48**, e2020GL092032 (2021).
[30] J. Yuval, P. A. O'Gorman, and C. N. Hill, Geophys. Res. Lett. **48**, e2020GL091363 (2021).
[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[32] F. Chillà and J. Schumacher, Eur. Phys. J. E **35**, 58 (2012).
[33] R. J. Adrian, R. T. D. S. Ferreira, and T. Boberg, Expt. Fluids **4**, 121 (1986).
[34] S. S. Zilitinkevich, *Turbulent Penetrative Convection* (Avebury Technical, Aldershot, England, 1991).
[35] Z. Sorbjan, J. Atmos. Sci. **53**, 101 (1996).
[36] K. Fodor, J. P. Mellado, and M. Wilczek, Boundary-Layer Meteorol. **172**, 371 (2019).
[37] H. Jaeger and H. Haas, Science **304**, 78 (2004).
[38] M. Lukoševičius, H. Jaeger, and B. Schrauwen, Künstl. Intell. **26**, 365 (2012).
[39] Z. Lu, J. Pathak, B. R. Hunt, M. Girvan, R. Brockett, and E. Ott, Chaos **27**, 041102 (2017).
[40] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, Chaos **27**, 121102 (2017).
[41] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, Neural Networks **126**, 191 (2020).
[42] N. A. K. Doan, W. Polifke, and L. Magri, Proc. R. Soc. A **477**, 20210135 (2021).
[43] J. Pathak, A. Wikner, R. Fussel, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, Chaos **28**, 041101 (2018).
[44] A. Wikner, J. Pathak, B. R. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott, Chaos **30**, 053111 (2020).
[45] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, Geophys. Res. Lett. **47**, e2020GL087776 (2020).
[46] L. Sirovich, Q. Appl. Math. **45**, 561 (1987).
[47] J. Bailon-Cuba and J. Schumacher, Phys. Fluids **23**, 077101 (2011).
[48] P. Baldi, in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, edited by I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Proceedings of Machine Learning Research (PMLR, Bellevue, Washington, USA, 2012), Vol. 27, pp. 37–49.
[49] F. J. Gonzalez and M. Balajewicz, arXiv:1808.01346.
[50] J. W. Deardorff, J. Atmos. Sci. **27**, 1211 (1970).
[51] R. B. Stull, *An Introduction to Boundary Layer Meteorology* (Kluwer Academic, Dordrecht, The Netherlands, 1988).
[52] J. P. Mellado and C. Ansorge, Z. Angew. Math. Mech. **92**, 380 (2012).
[53] https://github.com/turbulencia/tlab.
[54] M. Hermans and B. Schrauwen, in *Proceedings of the International Joint Conference on Neural Networks, (IJCNN) 2010, Barcelona, Spain, 18–23 July, 2010* (IEEE, Piscataway, NJ, 2010), pp. 1–7.
[55] H. Jaeger, GMD-Forschungszentrum Informationstechnik Technical Report **148** (2001).
[56] M. Lukoševičius, Lect. Notes Comput. Sci. **7700**, 659 (2012).
[57] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, Neural Networks **35**, 1 (2012).
[58] turbESN library, v0.0.1.8.3 (2022), available at https://github.com/flohey/turbESN.
[59] S. J. Pan and Q. Yang, IEEE Trans. Knowl. Data Eng. **22**, 1345 (2010).
[60] M. Inubushi and S. Goto, Phys. Rev. E **102**, 043301 (2020).
[61] P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, and R. Vinuesa, Phys. Rev. Fluids **4**, 054603 (2019).

[62] M. Raissi, A. Yazdani, and G. E. Karniadakis, Science **367**, 1026 (2020).
[63] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, Proc. IEEE **109**, 43 (2021).
[64] www.gauss-centre.eu.
[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in Neural Information Processing Systems*, edited by F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc., 2012), Vol. 25.
[66] K. He, X. Zhang, S. Ren, and J. Sun, arXiv:1502.01852.
[67] P. D. Kingma and J. Ba, arXiv:1412.6980.