This paper explores the applicability of machine learning methods to model convective boundary layers. The focus is on data reduction and the generalization to configurations different from the training configurations. To this aim, the convective boundary layer is modeled as a convection domain of a fixed height and a variable ratio of buoyancy fluxes between the top and the bottom boundaries, which is used to mimic different atmospheric conditions. Data are generated by direct numerical simulations. The analysis is restricted to two-dimensional cases to ease the study of different data reduction techniques before considering the more demanding three-dimensional case. Despite these simplifications, the model reproduces the main features of the mean, variances and covariances of the convective boundary layer. The data are used to train and test a recurrent neural network which is realized by an echo state network. The input to the echo state networks is obtained in two different ways, either by a proper orthogonal decomposition in the form of a snapshot method 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. **Keywords** — Machine Learning, Echo state networks, Convective boundary layer
Turbulent convection prevails in the diurnal atmospheric boundary layer over land in weak-wind conditions. The incoming solar radiation warms the surface, and the heat transfer to the air in contact with it creates a superadiabatic temperature profile that leads to convective instability. We refer to this regime of turbulent convection in the atmospheric boundary layer as the convective boundary layer (CBL) [Wyngaard 2010]. Because of its relevance, the CBL has been extensively studied during the last century, but important questions remain open regarding non-local effects induced by the large-scale organization of the flow, its parametrization in larger-scale atmospheric circulation models, and its sensitivity to changes in environmental conditions [Fodor et al. 2019; LeMone et al. 2019; Edwards et al. 2020]. In this paper, we explore the applicability of machine learning methods to address these issues.

Machine learning (ML) methods have changed paradigms of processing data and building data-driven parametrizations of unresolved turbulent processes in fluid mechanics [Brenner et al. 2019; Brunton et al. 2020; Pandey et al. 2020], oceanography [Zanna and Bolton 2020], and atmospheric science [Gentine et al. 2018; O’Gorman and Dwyer 2018; Bony et al. 2020]. In the latter case, they might be able to provide a step forward in the representation of essential properties of CBLs on coarser numerical models. Given the strong variability in the environmental conditions, we are particularly interested in how robust are machine-learning methods when applied to unseen configurations that are different from the training configurations, an essential aspect that directs to generalization properties of supervised learning methods [Goodfellow et al. 2016]. This sets the stage for the present work which consists of two major parts.

In the first part, we introduce and validate a two-dimensional convective cell model of the atmospheric CBL in cloud-free and shear-free conditions. The cell has a constant height, and we impose constant buoyancy fluxes at the bottom and top boundaries. Such a system has been considered in the past in the atmospheric context [Sorbjan 1996; Fodor et al. 2019], but only for a few particular values of controlling parameters. In this paper, we exploit dimensional analysis to consider an arbitrary combination of parameters. A major difference between this CBL model and the atmospheric CBL is that the depth of the CBL model is constant (non-penetrative convection), whereas the depth of the atmospheric CBL increases with increasing time as the CBL grows into the free troposphere (penetrative convection; e.g., Adrian et al. 1986; Zilitinkevich 1991). However, this growth is slow compared to the large eddy turnover times, which explains that important aspects of the structure and dynamics of the CBL are well represented, at least qualitatively, by the present CBL model, such as the well-mixed properties in the center of the convective region, the large convective cell organization, and the structure of the surface layers [Sorbjan 1996; Fodor et al. 2019]. Although the present CBL model does not represent the entrainment of fluid from the free troposphere into the turbulent region that occurs in the atmospheric CBL, the model retains the effect of entrainment warming at the CBL top, which creates an upper layer of stable fluid that is important for the CBL dynamics and a challenge for the parametrization of mixing in atmospheric models [Edwards et al. 2020].

This CBL model is similar to classical Rayleigh-Bénard convection (RBC) in that it represents turbulent convection between two solid plates [Chillà and Schumacher 2012]. For this reason, we refer to the present CBL model as an RBC-like configuration. Differently to most RBC studies, however, the present CBL model considers constant-flux conditions at the bottom and top boundaries instead of constant-temperature conditions. These boundary conditions provide a better approximation to the atmospheric CBL [Stull 1988; Wyngaard 2010]. The advantage of using RBC-like configurations is their statistically-steady state, which allows detailed analyses under controlled conditions and better statistical convergence compared with the unsteady and quasi-steady CBL [Fodor et al. 2019].

We use direct numerical simulation (DNS) to solve the governing equations and generate the data used in the training and testing of the ML methods described below. As a first step, this paper is restricted to two-dimensional
cases to facilitate the study of different data reduction techniques and their potential for modeling three-dimensional cases. Although the final goal is the application of ML methods to three-dimensional cases, two-dimensional surrogates can provide fast first estimates of the possible trade-offs between cost and performance of ML models, which help to ascertain their applicability to the more computationally demanding three-dimensional cases. In the first part of the paper, we validate that the turbulent properties in the two-dimensional case are representative of those observed in the three-dimensional case. In the second part of the paper, we use the DNS data as a training base and study the performance of dynamical reduced models of turbulent convection based on recurrent neural network architectures – neural networks with a short-term memory. These ML algorithms will be applied to systems that are different from the training configuration. Our study thus addresses an important open point of supervised machine learning algorithms, namely how well do they perform with respect to unseen data at changed conditions – a point known as the generalization property of a machine learning algorithm, as mentioned already above. The aim is to ascertain the potential of RBC-like configurations to develop machine learning-based models of turbulent mixing in oceanic and atmospheric boundary layers in convective regimes.

More specifically, we apply echo state networks (ESN) which are one implementation of reservoir computing (Jaeger and Haas, 2004; Lukoševičius et al., 2012). Reservoir computing (RC) uses a simple nonlinear dynamical system with recurrent connections for time series prediction. For this, the recurrent network, called the reservoir, maps a given input signal to a high-dimensional space. The dynamical state of the reservoir is then subject to external forcing by this input. After a sufficiently long propagation phase, a linear output rule is computed which maps the reservoir state of each iteration to the target output which is then the input at the next time step. The learned output rule and the initially created reservoir can then be used for the task of prediction. The RC approach is in contrast to conventional neural networks, where all parameters, i.e., weights and biases, are tuned by an optimization scheme such as stochastic gradient descent. The concept was simultaneously proposed by (Jaeger, 2001) and (Maass et al., 2002). Since its introduction, a large variety of such reservoir computers have been proposed (Nakajima, 2020). We mention the use of water waves and their interferences (Fernando and Sojakka, 2003), photonic (Vandoorne et al., 2011), and spintronic systems (Tsunegi et al., 2019) as well as novel quantum computing approaches (Fujii and Nakajima, 2017, 2020).

The ESN approach has found wide interest recently in inferring states of a nonlinear dynamical system. Applications showed for example that the dynamics of two of the three degrees of freedom of the Rössler system can be inferred from the evolution of the third one (Lu et al., 2017). Further, the Lyapunov exponents of the dynamical system that a trained ESN represents have been shown to match the exponents of the data generating system (Pathak et al., 2017). Moreover, hybrid models which combine both data driven (ESN) and knowledge based methods, i.e. solving the mathematical equations, have already been proposed (Pathak et al., 2018; Wikner et al., 2020) and tested in terms of a global atmospheric forecast model (Arcomano et al., 2020). Further, RC techniques, due to their computationally inexpensive training routine, could serve as a lightweight substitute for conventional parameterization schemes. Other neural network architectures have already been tested as subgrid scale parameterization (Pawar and San, 2021) or for the analysis of flow experiments (Moller et al., 2020). The performance of ESNs in two-dimensional dry and moist turbulent Rayleigh-Bénard convection have already shown great promise, as low-order statistics of buoyancy and liquid water fluxes are successfully reproduced (Pandey and Schumacher, 2020; Heyder and Schumacher, 2021). This paper extends this previous work to a configuration that is more representative of the atmospheric CBL.

Even two-dimensional simulation 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) by means of the proper orthogonal decomposition (POD) and (2) the convolutional autoencoder (CAE). As a consequence, the present ML algorithm is a combination of two building blocks, the encoder-decoder module and the dynamical core in the form of an ESN which advances with respect to time in the low-dimensional latent space. It is found that, despite smaller differences in flux statistics and
reconstruction of the fields, both models perform well.

The outline of the manuscript is as follows. In section 2, we describe the two-dimensional convective cell model of the atmospheric CBL and define all parameters, in particular, the ratio of the buoyancy fluxes at the top and bottom boundaries, $\beta$, which is the major control parameter. The four DNS runs at different $\beta$ and the resulting statistical properties are analysed and discussed. Section 3 introduces first the POD and architectures of CAE and ESN. It is followed by details on the training and the results of the trained echo state network in comparison to the test data. We summarize our results and give a brief outlook in the final section 4. Technical details on ML are listed in the appendix.

## 2 | CONVECTIVE CELL MODEL OF A CONVECTIVE BOUNDARY LAYER

### 2.1 | Governing equations

We use the Boussinesq approximation to the two-dimensional Navier-Stokes equations. For convenience and generality in case water-vapor effects become relevant, we formulate the problem in terms of the buoyancy $b \equiv \alpha g (\theta_v - \theta_v,0) / \theta_v,0$, where $\alpha$, $g$, $\theta_v$ are the thermal expansion coefficient, gravitational acceleration and virtual potential temperature, respectively. At laboratory scales and for temperature-only driven configurations, the buoyancy can be related to the temperature $T$ simply by $b \equiv \alpha g T$. We consider a cell of height $H$ and length $L$ (see figure 1). In the vertical direction, we consider no-slip boundary conditions for the velocity and constant-flux boundary conditions 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

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_z}{\partial z} = 0$$  \hspace{1cm} (1)

$$\frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + u_z \frac{\partial u_x}{\partial z} = -\frac{\partial p}{\partial x} + \nu \left( \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial z^2} \right)$$  \hspace{1cm} (2)

$$\frac{\partial u_z}{\partial t} + u_x \frac{\partial u_z}{\partial x} + u_z \frac{\partial u_z}{\partial z} = -\frac{\partial p}{\partial z} + \nu \left( \frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial z^2} \right) + b$$  \hspace{1cm} (3)

$$\frac{\partial b}{\partial t} + u_x \frac{\partial b}{\partial x} + u_z \frac{\partial b}{\partial z} = \kappa \left( \frac{\partial^2 b}{\partial x^2} + \frac{\partial^2 b}{\partial z^2} \right)$$  \hspace{1cm} (4)

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

$$\frac{\partial b}{\partial z} (x, z = 0, t) = -B_0 / \kappa$$  \hspace{1cm} (5)

$$\frac{\partial b}{\partial z} (x, z = 1, t) = -B_1 / \kappa$$  \hspace{1cm} (6)

For the sake of generality, we will present the analysis in a non-dimensional form. Choosing $H$ and $B_0$ as reference scales, one finds the following characteristic scales: the convective velocity $(B_0 H)^{1/3}$, the convective time $(H^2 / B_0)^{1/3}$, and the convective buoyancy $(B_0^2 H)^{1/3}$ [Deardorff, 1970]. The resulting four controlling parameters are the aspect
ratio $\Gamma = L/H$, the Prandtl number

$$Pr = \frac{\nu}{\kappa}, \quad (7)$$

the convective Rayleigh number

$$Ra_c = \frac{B_0 H^4}{\nu \kappa^2}, \quad (8)$$

and the buoyancy-flux ratio

$$\beta = -\frac{B_1}{B_0}. \quad (9)$$

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 from the top.

The buoyancy difference

$$\Delta b = \langle b \rangle_{x}(z = 0, t) - \langle b \rangle_{x}(z = 1, t) \quad (10)$$

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

$$Ra_f = \frac{\Delta b H^3}{\nu \kappa} \quad (11)$$

is a diagnostic variable as well. From eqns. (1–4) one can derive the vertical buoyancy profile, up to a constant, for the purely conductive case to be

$$b_{\text{cond}} = \frac{B_0 H}{\kappa} (\frac{z}{H} - 1)^2 + \beta (\frac{z}{H})^2 + \frac{B_0}{H} (1 + \beta) t + \text{constant}. \quad (12)$$

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 quasi-steady in the sense that the shape of the profile remains constant in time. The buoyancy difference between bottom and top plate for this case is

$$\Delta b_{\text{cond}} = \frac{B_0 H}{\kappa} \frac{1 - \beta}{2}. \quad (13)$$

### 2.2 Direct numerical simulations

We fix the Prandtl and convective Rayleigh number to $Pr = 1$ and $Ra_c = 3 \cdot 10^8$ and consider extended layers with an aspect ratio $\Gamma = L/H = 24$. The control parameter that we vary is the flux-ratio parameter $\beta$ defined by eq. (9). In
Heyder et al.

**FIGURE 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. Here we consider a) Adiabatic top ($\beta = 0$) and b) warming flux at the top ($\beta > 0$).

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 interested in the case $\beta > 0$. Typical atmospheric conditions correspond to the range $\beta \approx 0.1 - 0.3$ [Stull 1988; Wyngaard 2010]. As $\beta$ increases, the upper region of the convective cell increasingly stabilizes (positive mean buoyancy gradient, see also later in figure 5), and preliminary simulations (not shown) indicate that the dynamics strongly change for values $\beta \approx 0.4$. Therefore, we consider the cases $\beta \in \{0.0, 0.1, 0.2, 0.3\}$ in our CBL model (see also figure 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.

The Boussinesq equations (1) – (4) 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 on the numerical method can be found in Mellado and Ansorge (2012). The software used to perform the simulations is freely available at [https://github.com/turbulencia/tlab](https://github.com/turbulencia/tlab).

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 $\Delta t \approx 0.0012 - 0.0016 \left( \frac{H^2}{B_0} \right)^{1/3}$, 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 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 $\Delta t$. Table 1 summarizes important parameters of the four simulation runs.

2.3 | Cellular structure of the convection layer and vertical profiles at different $\beta$

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

$$\langle b \rangle_{x,z} = \frac{B_0}{H} (1 + \beta) t.$$  (14)
Hence, in the turbulent case, the fluid warms linearly with increasing time as in the conduction case. The mean vertical profile, however, is different to the pure conduction profile and, as mentioned above, a major dependent variable is the buoyancy difference $\Delta b$ across the cell. After an initial transient, this quantity becomes statistically stationary, as can be seen in figure 2a) for all four 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

$$\text{Nu} = \frac{\Delta b_{\text{cond}}}{\Delta b} = \frac{1 - \beta B_0 H}{2 \kappa \Delta b}.$$  

(15)

defined here as the ratio between the buoyancy difference in the purely conductive case $\Delta b_{\text{cond}}$ (see eq. 13), and the fully convective case, i.e. $\Delta b$. 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 figure 2 for all four cases.

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

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

(16)

---

**Table 1** 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.

| $\beta$ | $\langle \Delta b \rangle_t$ | $\langle T_f \rangle_t$ | $\langle \text{Nu}_f \rangle_t$ | $\langle \text{Ra}_f \rangle_t$ |
|---------|------------------|-----------------|----------------|-----------------|
| 0.0     | $22.0 \pm 0.4$   | 0.21            | 15.2 $\pm$ 0.3 | 0.99 $\times 10^7$ |
| 0.1     | $19.0 \pm 0.4$   | 0.23            | 15.9 $\pm$ 0.3 | 0.85 $\times 10^7$ |
| 0.2     | $15.4 \pm 0.3$   | 0.26            | 17.4 $\pm$ 0.4 | 0.69 $\times 10^7$ |
| 0.3     | $10.7 \pm 0.5$   | 0.31            | 21.9 $\pm$ 0.9 | 0.48 $\times 10^7$ |
and the vertical flux $u'_x(x, z) b'(x, z)$ 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 $\beta$ increases. 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 panels (a) and (d).

**FIGURE 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 four different top boundary conditions ($\beta = 0.0, 0.1, 0.2, 0.3$) differ in their width of the top thermal boundary layer. For the adiabatic top $\beta = 0.0$ no such layer is present. Note that with increasing $\beta$ the range of $b^*$ increases.

**FIGURE 4** Instantaneous snapshot of the vertical buoyancy flux $u'_x(x, z) b'(x, z)$ in the statistically stationary state. The cellular order is increasingly dissolved with growing parameter $\beta$.

We show the line-time average vertical profiles $\langle \cdot \rangle_{x,t}(z)$ of $b^*$ in figure [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, constant-flux boundary conditions break the
**FIGURE 5** Vertical profiles of a) the normalized buoyancy
\( \nu^* = (b - \langle b \rangle_x)(z = 1)/\langle \Delta b \rangle_x \), b) buoyancy fluctuations, c) vertical velocity fluctuations, d) normalized total buoyancy flux \( F_b/\langle F_b \rangle \). 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.

Note that \( \langle b \rangle_x \) depends on time, as it incorporates the linear warming of the fluid. Meanwhile, \( \langle u_x \rangle_x \) and \( \langle u_z \rangle_x \) are statistically stationary and vary weakly about their zero mean. The vertical profiles of the root mean square (r.m.s.) of \( u'_x \) and \( b' \) are shown in figures 5(b) and (c). The r.m.s. of the fluctuations of the buoyancy differ greatly in their

\begin{align}
  u_x(x, z, t) &= \langle u_x \rangle_x + u'_x(x, z, t), \\
  u_z(x, z, t) &= \langle u_z \rangle_x + u'_z(x, z, t), \\
  b(x, z, t) &= \langle b \rangle_x + b'(x, z, t).
\end{align}

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 at the top.
magnitude and trend in the upper portion of the domain. The vertical r.m.s. velocity component \( \langle u_z' \rangle_{x,t} \), on the other hand, does not vary too much while changing \( \beta \). Additionally, the total buoyancy flux
\[
F_b = \langle u_z' b' \rangle_{x,t} - \kappa \frac{\partial \langle b \rangle_{x,t}}{\partial z}
\]
normalized by its bottom value is shown in figure 5(d). We find that the flux decreases linearly with increasing height. As indicated by figure 5(b), 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 \cite{Stull1988, Wyngaard2010}, 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.

For now, we can conclude that the present two-dimensional model of a convective boundary layer incorporates already several important physical properties that are known from the atmospheric CBL in cloud-free and wind-free conditions. To underline this remark, we add profiles of a CBL case as dashed brown lines to all 4 panels of figure 5. This CBL is a 2D version of 3D configurations used by \cite{Fodor2019} to compare the CBL with RBC, and represents penetrative convection, i.e., it retains entrainment and the growth of the CBL depth that are present in the atmospheric CBL in case of a linearly stratified free troposphere. One important property, which goes beyond the classical Rayleigh-Bénard convection case \cite{Chilla2012} is the top-down asymmetry which is obvious from the mean vertical profiles.

In the following, we use the DNS data of \( \beta \neq 0 \) to train a recurrent neural network and make subsequent predictions for an unseen data set. This is done to explore the generalization properties of the echo state networks. We therefore interpolate all fields from the non-uniform grid with 2400 × 150 points to a 720 × 30 uniform grid by cubic splines. This grid will be denoted as the coarse-grained grid, the data as coarse-grained DNS data.
3 | CONVECTIVE BOUNDARY LAYER STATISTICS FROM AN ECHO STATE NETWORK

3.1 | Echo state network and echo state property

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

$$C(W_{\text{out}}) = \sum_{n=-T_L}^{-1} \|y(n) - W_{\text{out}}^{\text{out}}r(n)\|_2^2 + \lambda \|w_{\text{out}}^i\|_2^2, \quad (23)$$

where $y$ are the target outputs, which are part of the training data. $T_L$ is the number of training time steps, $W_{\text{out}}^i$ is the $i$th row of $W_{\text{out}}$ and $\| \cdot \|_2$ denotes the $L^2$ norm. The last term penalizes large values of the rows of the output weight matrix by adjusting the regression parameter $\lambda$. This concept is one possibility to counter the problem of 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

$$W_{\text{out}}^* = Y R^T \left( R R^T + \lambda I \right)^{-1}$$  \hspace{1cm} (24)

where the $n^{th}$ column of $Y \in \mathbb{R}^{N_t \times T_L}$, $S \in \mathbb{R}^{N_t \times T_L}$ are $y(n)$ and $\tilde{r}(n)$ respectively, $I \in \mathbb{R}^{N_t \times N_t}$ 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 $n \geq 0$ is fed back to the input layer, by letting $x(n) = W_{\text{out}}^* \tilde{r}(n - 1)$. During this testing phase the ESN autonomously predicts the next $T_T$ iterations of the initial 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 $h = \{y, \lambda, N_t, D, \varphi\}$. The last two quantities are the reservoir density $D$ and spectral radius $\varphi$. They are algebraic properties of the reservoir weight matrix and represent the number of non-zero elements and largest absolute eigenvalue of $W^r$, respectively. Finding a right setting of these hyperparameters is crucial, as they influence the memory capacity of the reservoir (Hermans and Schrauwen, 2010). In Jaeger, 2001 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 $r_1(n - 1)$, $r_2(n - 1)$ converge to the same reservoir state $r(n)$, provided the same input $x(n)$ 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 (Lukoševičius, 2012). So far, no universal rule for the presence of echo states has been proposed. On top of that, the echo state property is merely a necessary condition and no feasible sufficient condition has yet been found as discussed in Yildiz et al. (2012). We will keep using reservoir initializations and hyperparameter ranges, which have shown good results, e.g., in Pandey and Schumacher (2020) or Heyder and Schumacher (2021). We initialize the input and reservoir weights as random, i.e., $W^\text{in} \sim \mathcal{U}[-0.5,0.5]$ and $W^r \sim \mathcal{U}[0,1]$. $W^r$ is then normalized by its largest absolute eigenvalue and is subsequently scaled by $\varphi$. Afterwards, randomly selected entries of this matrix are set to zero to assure the specified value of the reservoir density $D$ is obtained. The specific value of each of the quantities in $h$ is chosen by a grid search procedure which will be discussed further below.

### 3.2 | Network training with data from DNS 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 (Pan and Yang, 2010). 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 (Inubushi and Goto, 2020).

Here, we take a different approach which is sketched in Figure 7. A reservoir is trained with data of one case of buoyancy boundary conditions at $z = 1$, namely $\beta = 0.1$. Finally, we use the trained network for predicting the dynamics and statistical properties of two different and unseen convective flows with buoyancy flux parameter $\beta = 0.2$ and $0.3$.

The DNS data possesses 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 modelling techniques, the (1)
| β  | 0.1 | 0.2 | 0.3 |
|-----|-----|-----|-----|
| Cumulative Contribution (%) | 82.4 | 80.2 | 78.4 |

**TABLE 2** Cumulative contribution of the first $N_{\text{POD}} = 300$ POD modes for the three values of the boundary condition parameter $\beta$, which are used in our RC approach.

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. The CAE on the other hand, represents a deep convolutional neural network, commonly used in deep learning tasks, such as image processing. In the following we discuss similarities as well as differences between both methods in terms of their encoding-decoding mechanism. Finally in 3.3 their individual prediction performance with the ESN will be examined.

### 3.2.1 Proper Orthogonal Decomposition for data reduction

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 $\beta = 0.2$ and 0.3. Before reducing the dimensionality of the data, we decompose the buoyancy fluctuations further

$$b'(x,z,t) = \langle b' \rangle_t(x,z) + b''(x,z,t).$$

Finally, we apply the POD with the methods of snapshots [Sirovich 1987; Bailon-Cuba and Schumacher 2011] on the vector $g = (u'_x, u'_z, b'')^T$, such that its $k^{th}$ component can be written as

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

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 $N_{\text{dof}}$ can then be reduced, by taking only $N_{\text{POD}} \ll N_{\text{dof}}$ modes and coefficients with the most variance into account.

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

The individual and cumulative contribution of each mode can be seen in figure [8] and in table [2]. In the following we will consider the $N_{\text{POD}} = 300$ most energetic POD time coefficients as input for the ESN. The total number of degrees of freedom (dof) is thus reduced from three fields on a grid with size $2400 \times 150$ in the original DNS (that corresponds to $N_{\text{dof}} = 1.08 \times 10^9$) via coarse grained data of grid size $720 \times 30$ for the POD input to 300 modes in the latent space by a factor of 3600. With this choice of cut-off mode, we capture about 80% of the original energy. We construct the training data set for our ESN by taking the 700 instances of time coefficients $a(n) = (a_1(n), a_2(n), \ldots, a_{N_{\text{POD}}}(n))^T$ of $\beta = 0.1$. This results to a total training length of $T_L = 700$. During this phase the reservoir is trained to predict the respective next time instance of the POD expansion coefficients $a(n+1)$, see again eq. [24].
Figure 7 Sketch of the transfer learning concept. During the training phase (a), a reservoir is trained with simulation data for the case of $\beta = 0.1$ in the low-dimensional latent space, either with $a_{\beta=0.1}(n)$ for the reduction by POD or $\xi_{\beta=0.1}(n)$ for the one by a CAE. The network learns the dynamics; the optimal output weights are obtained. In the prediction phase (b) the reservoir is then used to infer the dynamics of the target latent spaces at $\beta = 0.2, 0.3$ and predicts either $a_{\beta=0.2,0.3}(n)$ (POD) or $\xi_{\beta=0.2,0.3}(n)$ (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 $\beta = 0.2$ and $\beta = 0.3$. 
3.2.2 Convolutional Autoencoder for data reduction

As an alternative to the data reduction by a snapshot POD, we investigate now the application of an encoder-decoder network, also known as autoencoder. An autoencoder is a feed-forward neural network which is trained to reproduce its network input \( g \) as network output \cite{Goodfellow2016}. In order for the network not just copy its inputs to the output layer, an intermediate bottleneck structure is introduced, s.t. the original information is compressed to an encoding space. Therefore, the autoencoder consists of two parts which are trained as one network. The encoder \( f \) compresses the high-dimensional inputs to a low-dimensional representation

\[
\xi = f_{\text{encoder}}(g) \tag{28}
\]

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

The decoder \( h \) then attempts to decode the encoded latent space and reconstruct the original information

\[
g^{\text{AE}} = h_{\text{decoder}}(\xi) = h_{\text{decoder}}(f_{\text{encoder}}(g)) \tag{29}
\]

Here \( g^{\text{AE}} \) is the convolutional reconstruction and \( \theta_{\text{decoder}} \) includes all trainable weights and biases of the decoder network. Here 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 \cite{Krizhevsky2012}. While the \( \xi \) can be understood as a low-dimensional representation of the input, similar to the POD time coefficients \( a \), 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, \( g^{\text{AE}} \approx g \).

Similar to the POD approach, we take snapshots of \( g = (u'_x, u'_z, b'')^T \) of \( \beta = 0.1 – 0.3 \) as input for their own CAE.

\[\text{FIGURE 8} \quad \text{POD spectrum: a) individual contribution of each POD mode to the total energy, b) cumulative contribution. The green shaded area marks the first 300 modes. See also table 2.}\]
Finally one can use the trained encoder to translate the CBL dynamics into dynamics of the latent space $\xi(t)$. We choose an encoding dimension of $N_{CAE} = 300$ 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 3. More information on the CAE architecture and our choice of its hyperparameters can be found in the appendix.

The ESN training procedure is the same as mentioned above: out of the 2000, to the CAE unseen snapshots, we sample 700 time steps of $\xi^{\beta=0.1}(t)$ in an interval of $0.25T_f$. These will then be used to train the ESN, while 700 time steps of $\xi^{\beta=0.2}(t)$ and $\xi^{\beta=0.3}(t)$ are used for validation of the ESN predictions.

3.3 | Inferring the cases $\beta = 0.2, 0.3$ from trained echo state network

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 two CBL model cases, $\beta = 0.2, 0.3$. For this, we initialize a new reservoir state which is preceded by 50 iterations of Eq. (21), where the reservoir input is given by 50 time steps of $\mathbf{a}^{\beta=0.2}, \mathbf{a}^{\beta=0.3}$ (POD) or $\xi^{\beta=0.2}, \xi^{\beta=0.3}$ (CAE) (see Figure 7b). With this washout phase, we intend to transition to the new regime of $\beta = 0.2$ or 0.3. Starting from this state of the convective flow, the ESN will autonomously predict $T_f = 700$ future time steps with its learned output weights. We validate these predictions by a direct comparison with $\mathbf{a}^{\beta=0.2}(n), \mathbf{a}^{\beta=0.3}(n)$ and $\xi^{\beta=0.2}(n), \xi^{\beta=0.3}(n)$ with $n \in [1, T_f]$, 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

$$\text{MSE}_h = \frac{1}{T_f} \sum_{n=1}^{T_f} \| \hat{\mathbf{y}}(n) - \mathbf{a}^{\beta=0.2}(n) \|^2_2.$$  \hfill (30)

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 Srinivasan et al. (2019) and is given for example for $u'_z b''$ by

$$E_h \left[ \langle u'_z b'' \rangle_{x,t} \right] = \frac{1}{C_{\text{max}}} \int_0^1 \left| \langle u'_z b'' \rangle_{x,t}^{\text{ESN}}(y) - \langle u'_z b'' \rangle_{x,t}^{\text{POD}}(z) \right| dz$$  \hfill (31)

with

$$C_{\text{max}} = \frac{1}{2B_0 \max_{z \in [0,1]} \| (u'_z b'')_{x,t}^{\text{POD}} \|^2}.$$  \hfill (32)

The superscript indicates whether the field is reconstructed, see Eq. (27), from the $N_{\text{POD}}$ POD time coefficients (POD) or the ESN predictions (ESN). This measure quantifies errors in the line-time average profiles of the physical fields.

| $\beta$ | 0.1 | 0.2 | 0.3 |
|---------|-----|-----|-----|
| Training Loss (%) | $6.2 \cdot 10^{-4}$ | $7.1 \cdot 10^{-4}$ | $8.1 \cdot 10^{-4}$ |
| Validation Loss (%) | $2.66 \cdot 10^{-3}$ | $2.76 \cdot 10^{-3}$ | $2.72 \cdot 10^{-3}$ |

**TABLE 3** Mean squared error loss of the CAE reconstruction after 1200 epochs of training. Each CAE was trained on 8000 snapshots of $(u'_z, u'_z, b'')$ for their corresponding $\beta$. The validation loss was computed on 2000 different snapshots.
Similarly, one can define MSE and NARE for the CAE case by using $\xi$ instead of $\alpha$ and the CAE instead of the POD reconstruction.

Our choice of the optimal ESN hyperparameters $h^*$ is listed in Table 4. We conducted grid searches of $N$, $D$, $\gamma$ and $\varrho$. See the appendix for more details. For each setting, we additionally took 100 random realizations of the same reservoir setting and computed MSE$_h$ and $E[u'_z b'']$ of all 100 samples. We deliberately choose the third quartile over the median, as it assures robust reservoir outputs for different random weights $W^m$, $W^r$ 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.

### 3.3.1 Results for Proper Orthogonal Decomposition-Echo State Network

We reconstruct each component of the physical fields $u_x$, $u_z$ and $b$ via eq. (27) using the decompositions (17) – (19) and (25). For the validation, we use the expansion coefficients of the first $N_{POD}$ modes of the $\beta = 0.2$ and $\beta = 0.3$ 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

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

(33)

the vertical velocity component $u_z(x, z)$, and the normalized buoyancy $b^*(x, z)$ can be seen in figure [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 in the prediction case. In the velocity field component, vertical up- and downdrafts can be clearly identified. Their width and shape differs 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 well. We also compute the normalized root mean square error (NRMSE) of these three fields at each grid point and averaged over all time steps of the test period. It quantifies the error between predicted and POD fields. For example, for the vertical velocity component, the NRMSE is defined as

$$NRMSE_{u_z}(x, z) = \frac{1}{\sqrt{\left\langle u_z^{(POD)}(x, z) \right\rangle_t}} \sqrt{\left\langle \left( u_z^{(ESN)}(x, z) - u_z^{(POD)}(x, z) \right)^2 \right\rangle_t}.$$
FIGURE 9  POD case for inferring $\beta = 0.2$. Instantaneous snapshots of the local turbulent kinetic energy $E_{\text{kin}}(x, z, t_0)$ in panels (a,b), the vertical velocity component $u_z$ in panels (c,d) and the normalized buoyancy $b^*$ in panels (e,f) at time step $n = 350$ in the prediction phase. POD reconstructions with the most energetic $N_{\text{POD}}$ modes of $\beta = 0.2$ (validation snapshot) are shown in panels (a), (c), and (e). The corresponding ESN predictions are displayed in panels (b), (d), and (f).

| POD  | $\langle \text{NRMSE}_{E_{\text{kin}}} \rangle_{x,z}$ | $\langle \text{NRMSE}_{u_z} \rangle_{x,z}$ | $\langle \text{NRMSE}_{b^*} \rangle_{x,z}$ |
|------|--------------------------------------------------|---------------------------------|-------------------------|
| $\beta$ |                                                  |                                 |                          |
| 0.2  | 1.01                                             | 1.45                            | 0.17                     |
| 0.3  | 1.07                                             | 1.42                            | 0.13                     |

TABLE 5  Volume average of the normalized time-averaged root mean square error (NRMSE) at each grid point of the turbulent kinetic energy $E_{\text{kin}}$, the vertical velocity component $u_z$, and the normalized buoyancy $b^*$, see eq. (34).

Figure 10 shows the spatial distribution of the NRMSE for the three fields of figure 9. Its volume average is additionally listed in table 5. We find that the bulk domain for $E_{\text{kin}}$ and $u_z$ shows the biggest differences. This is where the strongest up- and downflows are present. Bigger deviations are observed for the boundary layer regions of $b^*$, i.e., in the regions of the convection layer where the plumes detach or impact.

We emphasize that these results were obtained for one particular realization out of the 100 reservoirs with the same hyperparameter setting, that 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 further, by computing line-time average profiles $\langle \cdot \rangle_{x,t}$ of the fluctuations of the corresponding fields for $\beta = 0.2$ and 0.3. These are important parameters of simulations of large-scale turbulence. The profiles are given in figure 11 and their corresponding NARE values are listed in table 6.

We find that in this setting $h_*$, the average reservoir produces reasonable approximations to the profiles of the true low-order statistics of the both $\beta$ values. Despite some deficiency in the profiles of turbulent kinetic energy and vertical
FIGURE 10 Normalized time-averaged root mean square error (NRMSE) of the inferred POD reconstruction. Shown is the result for $\beta = 0.2$. (a) Local turbulent kinetic energy $E_{\text{kin}}^{(\text{ESN})}$. (b) Vertical velocity component $u_z^{(\text{ESN})}$. (c) Normalized buoyancy field $b^{*}^{(\text{ESN})}$.

POD

| $\beta$ | $E_{h_x} \left[ 0.5(u_x^2 + u_z^2)_{x,t} \right]$ | $E_{h_x} \left[ (u_x^2)_{x,t} \right]$ | $E_{h_x} \left[ (u_z^2)_{x,t} \right]$ | $E_{h_x} \left[ (b^{*})_{x,t} \right]$ | $E_{h_x} \left[ (u_x b^{''})_{x,t} \right]$ |
|--------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 0.2    | 0.033           | 0.024           | 0.002           | 0.018           | 0.018           |
| 0.3    | 0.015           | 0.009           | 0.003           | 0.018           | 0.018           |

TABLE 6 Normalized average relative errors, see eq. (31), of the inferred line-time average profiles shown in figure 11.

velocity for $\beta = 0.3$, the asymmetry due to the boundary conditions is captured in all profiles. Especially the buoyancy fluctuations are reproduced well. While the ESN reproduces the linear decrease of the convective buoyancy flux, $(u_x' b^{''})_{x,t}^{(\text{ESN})}$, 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.

Overall, the ESN generalizes well to unseen convection data with similar boundary conditions, when using the low-order POD model. The inferred fields of $\beta = 0.2$ and 0.3 (see appendix) reproduce features like 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.

3.3.2 Results for Convolutional Autoencoder-Echo State Network

By decoding the inferred latent spaces using eq. (29) and eqns. (17)–(19) as well as eq. (25), 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). 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 coarse artefacts in the inferred fields, the autoencoder reproduces the small-scale features of the convection patterns well.
FIGURE 11  Line-time average profiles. (a) Turbulent kinetic energy. (b) Root mean square profile of vertical velocity fluctuations. (c) Root mean square buoyancy fluctuations profile of $b''$. (d) Convective buoyancy flux. The ESN predictions (dotted 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 $\beta = 0.2$ (blue) and $\beta = 0.3$ (orange).
**FIGURE 12** CAE case for inferring $\beta = 0.2$. Instantaneous snapshots of the turbulent kinetic energy $E_{\text{kin}}(x, z, t_0)$ in panels (a,b), the vertical velocity component $u_z$ in panels (c,d) and the normalized buoyancy $b^*$ in panels (e,f) at time step $n = 350$ in the prediction phase. CAE reconstructions of $\beta = 0.2$ (validation snapshot) are shown in panels (a), (c), and (e). The corresponding ESN predictions are displayed in panels (b), (d), and (f).

As before, we measure the prediction performance by use of the NRMSE (see eq. (34)). Again, we find that deviations from the ground truth (CAE) are centered in the bulk of the fluid fields ($E_{\text{kin}}$ and $u_z$), while the boundary layer is the main source of error for $b^*$. Table 7 lists the volume average of the NRMSE. These global performance measures have a magnitude that is comparable to the POD case.

**FIGURE 13** Normalized time-averaged root mean square error (NRMSE) of the inferred CAE reconstruction. Shown is the result for $\beta = 0.2$. (a) Turbulent kinetic energy $E_{\text{kin}}^{(\text{ESN})}$. (b) Vertical velocity component $u_z^{(\text{ESN})}$. (c) Normalized buoyancy field $b^*^{(\text{ESN})}$.

Figure 14 shows the inferred line-time averaged profiles of the physical fields. Their corresponding NARE values...
TABLE 7 Volume average of the normalized time-averaged root mean square error (NRMSE) at each grid point of the kinetic energy $E_{\text{kin}}$, the vertical velocity component $u_z$, and the normalized buoyancy $b^*$, see eq. (34).

| $\beta$ | $\langle \text{NRMSE}_{E_{\text{kin}}} \rangle_{x,z}$ | $\langle \text{NRMSE}_{u_z} \rangle_{x,z}$ | $\langle \text{NRMSE}_{b^*} \rangle_{x,z}$ |
|--------|---------------------------------|---------------------------------|---------------------------------|
| 0.2    | 1.14                            | 1.34                            | 0.16                            |
| 0.3    | 1.17                            | 1.41                            | 0.13                            |

TABLE 8 Normalized average relative errors, see eq. (31), of the line-time average profiles shown in figure 14.

| $\beta$ | $E_h \left[ 0.5(u'^2_z + u'^2_z)_{x,t} \right]$ | $E_h \left[ \langle u'^2_z \rangle_{x,t} \right]$ | $E_h \left[ \langle b''^2 \rangle_{x,t} \right]$ | $E_h \left[ \langle u'_z b'' \rangle_{x,t} \right]$ |
|--------|-----------------------------------------------|-----------------------------------------------|-----------------------------------------------|-----------------------------------------------|
| 0.2    | 0.053                                         | 0.040                                         | 0.007                                         | 0.013                                         |
| 0.3    | 0.027                                         | 0.029                                         | 0.017                                         | 0.035                                         |

are listed in table 8. Other than the linear POD method, the CAE is trained by a gradient descent procedure, which introduces artefacts 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 consequence, larger magnitudes of this measure can be observed for most entries of the table.

We speculate that this result might be resolved by introducing an additional term to the loss function of the CAE that penalizes large deviations from the mean profiles which we have not applied here. The biggest artefacts can be seen in the buoyancy flux in figure 14(d). 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. While for $\beta = 0.3$, turbulent kinetic energy, vertical velocity and buoyancy fluctuations seem to be harder to match for the ESN, the buoyancy flux shows good agreement. The $\langle u'_z b'' \rangle_{x,t}$ profile of the intermediate case $\beta = 0.2$ is poorly predicted in the lower boundary layer, where the maximum value is overestimated by the reservoir.

4 | CONCLUSIONS AND OUTLOOK

In this work, we explored machine learning methods to model the convective boundary layer. In particular, we considered echo state network algorithms and a two-dimensional convective cell model of the atmospheric boundary layer. We impose a buoyancy flux at that top and bottom boundaries mimicking entrainment and surface fluxes in atmospheric convective boundary layers. The model is hence characterized by the buoyancy flux ratio $\beta$ between the magnitudes of the top and the bottom fluxes. An increasing value of this parameter quantifies a counter-heating that stabilizes the top layer and results in negative values of the convective buoyancy flux close to the top boundary. This prescribed flux at the top mimics the warming associated with the entrainment of free tropospheric air in atmospheric convective boundary layers. On the one hand, our model thus 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 it is similar to a non-Boussinesq flow. 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 strong simplification, in particular in view to its two-dimensionality.
**FIGURE 14** Line-time average profiles. (a) Turbulent kinetic energy. (b) Root mean square profile of vertical velocity fluctuations. (c) Root mean square buoyancy fluctuations profile of $b'$. (d) Convective buoyancy flux. The autoencoder introduces artefacts in the statistical profiles (solid lines) of $\beta = 0.2$ (blue) and $\beta = 0.3$ (orange). The ESN predictions (dotted lines) were chosen as to hold the median buoyancy flux NARE.
We conducted a series of direct numerical simulations for values of $\beta$ that vary between 0 and 0.3, a range that represents midday atmospheric conditions over land. An adiabatic top boundary, i.e., zero incoming and outgoing flux ($\beta = 0$), is also considered for comparison. The four simulations reproduce well-known features of atmospheric convective boundary layers. The mean buoyancy is constant throughout the middle of the domain, which resembles the mixed layer inside the convective boundary layer. The positive buoyancy gradients at the top can be seen as approximations to the entrainment zone between mixed layer and free atmosphere. Moreover, the covariance of vertical velocity and buoyancy show that the well-known linear decline with height is reproduced by our model. The four 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 $\beta$ 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.

Regarding the machine learning method, we employ 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 $\beta = 0.2$ and 0.3. 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 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 $\beta = 0.2$ and 0.3. 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.

We point out that the two low-order models differ in their compression technique and hence yield different performances, when combined with reservoir computing model in the latent space. While the POD preserves line-time average profiles, the autoencoder introduces small artefacts to the statistics. The quality of the predicted spatial features differ also among the methods, as the predicted POD time coefficients capture coarse convection features, while the convolutional autoencoder reproduces natural convection patterns. We can conclude that for our setup, data emerging from one case with constant flux boundary conditions can be used to infer at least statistical and spatial features of two different cases with different conditions. The echo state network can thus serve as a reduced-order and scalable dynamical model that generates the appropriate turbulence statistics without solving the underlying Navier-Stokes equation of the flow.

The present study should be considered as a first step in the development of efficient reduced dynamical models of convection processes by machine learning methods. Several directions for the 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 version of a recent approach by Fonda et al. (2019) which reduced the convective turbulent heat transport across a convection layer to a dynamic planar network. We also assess that one should incorporate physical laws or known flow properties in the training routine of the autoencoder, as mere mimicking of the input fields produces artefacts in the statistical features of the reconstruction. 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. 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.

5 | ACKNOWLEDGEMENTS

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

6 | CONFLICT OF INTEREST

The authors report no conflict of interest.
A | APPENDIX

A.1 | Echo state network grid search procedure

In order to find an optimal reservoir for both reduction methods and both $\beta$ values, we conducted grid searches on four important reservoir hyperparameters, namely $N$, $D$, $\gamma$ and $\varphi$. The range and no. values of each hyperparameter study are listed in Table 9.

| Hyperparameter | Range       | No. samples |
|----------------|-------------|-------------|
| $N$            | [512, 4096] | 4           |
| $D$            | [0.1, 1.0]  | 100         |
| $\gamma$       | [0.1, 1.0]  | 10          |
| $\varphi$      | [0.1, 2.0]  | 100         |

**Table 9** ESN grid search range of each hyperparameter that was studied. The number of samples indicates how many different values of each hyperparameter were studied.

A.2 | Autoencoder

A.2.1 | Architecture

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 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 encoder and decoder was sigmoid, while all other layers were followed by a Parametric Rectified Linear Unit (PReLU) He et al. (2015). The channel size, as well as convolutional and max pooling kernels are listed in Table 10. Using this architecture the total number of trainable weights and biases amounts to $4.96 \times 10^6$.

| Layer | Conv#1 | Conv#2 | Conv#3 | Conv#4 | Conv#5 | Conv#6 | Conv#7 | Conv#8 | Conv#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)  | -      |

**Table 10** Size of convolutional channels, kernel and max pooling 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).

A.2.2 | Training routine

The autoencoder is trained using the ADAM optimizer Kingma and Ba (2017) with a learning rate $10^{-5}$, batch size 64 and a $L_2$ penalty term with penalty parameter $10^{-6}$. The loss function that was minimized was chosen to be the mean square error between input and output fields. Moreover, the input data was scaled to the range $[0, 1]$ before it was passed to the input layer of the CAE. Finally, the network was trained for 1200 epochs on 2 GPUs and took about 106min.
A.3  |  Results for $\beta = 0.3$

**Figure 15**  POD case for inferring $\beta = 0.3$. Instantaneous snapshots of the local turbulent kinetic energy $E_{\text{kin}}(x, y, t_0)$ in panels (a,b), the vertical velocity component $u_z$ in panels (c,d) and the normalized buoyancy $b^*$ in panels (e,f) at time step $n = 350$ in the prediction phase. POD reconstructions with the most energetic $N_{\text{POD}}$ modes of $\beta = 0.2$ (validation snapshot) are shown in panels (a), (c), and (e). The corresponding ESN predictions are displayed in panels (b), (d), and (f).

**Figure 16**  Normalized time-averaged root mean square error (NRMSE) of the inferred POD reconstruction. Shown is the result for $\beta = 0.3$. (a) Local turbulent kinetic energy $E_{\text{kin}}^{(\text{ESN})}$. (b) Vertical velocity component $u_z^{(\text{ESN})}$. (c) Normalized buoyancy field $b^*^{(\text{ESN})}$. 
FIGURE 17  CAE case for inferring $\beta = 0.3$. Instantaneous snapshots of the turbulent kinetic energy $E_{\text{kin}}(x, y, t_0)$ in panels (a,b), the vertical velocity component $u_z$ in panels (c,d) and the normalized buoyancy $b^*$ in panels (e,f) at time step $n = 350$ in the prediction phase. CAE reconstructions of $\beta = 0.3$ (validation snapshot) are shown in panels (a), (c), and (e). The corresponding ESN predictions are displayed in panels (b), (d), and (f).

FIGURE 18  Normalized time-averaged root mean square error (NRMSE) of the inferred CAE reconstruction. Shown is the result for $\beta = 0.3$. (a) Turbulent kinetic energy $E_{\text{kin}}^{(\text{ESN})}$. (b) Vertical velocity component $u_z^{(\text{ESN})}$. (c) Normalized buoyancy field $b^*^{(\text{ESN})}$. 
References

Adrian, R. J., Ferreira, R. T. D. S. and Boberg, T. (1986) Turbulent thermal convection in wide horizontal fluid layers. *Exp. Fluids*, 4, 121.

Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R. and Ott, E. (2020) A machine learning-based global atmospheric forecast model. *Geophys. Res. Lett.*, 47, e2020GL087776.

Bailon-Cuba, J. and Schumacher, J. (2011) Low-dimensional model of turbulent rayleigh-bénard convection in a cartesian cell with square domain. *Phys. Fluids*, 23, 077101.

Bony, S., Schulz, H., Vial, J. and Stevens, B. (2020) Sugar, gravel, fish, and flowers: Dependence of mesoscale patterns of trade-wind clouds on environmental conditions. *Geophys. Res. Lett.*, 47, e2019GL085988.

Brenner, M. P., Eldredge, J. D. and Freund, J. B. (2019) Perspective on machine learning for advancing fluid mechanics. *Phys. Rev. Fluids*, 4, 100501.

Brunton, S. L., Noack, B. R. and Koumoutsakos, P. (2020) Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.*, 52, 477.

Chillà, F. and Schumacher, J. (2012) New perspectives in turbulent rayleigh-bénard convection. *Eur. Phys. J. E*, 35, 58.

Deardorff, J. W. (1970) Convective velocity and temperature scales for the unstable planetary boundary layer and for Rayleigh convection. *J. Atmos. Sci.*, 27, 1211.

Edwards, J., Beljaars, A., Holtslag, A. and Lock, A. (2020) Representation of boundary-layer processes in numerical weather prediction and climate models. *Boundary Layer Meteorol.*, 177, 511.

Fernando, C. and Sojakka, S. (2003) Pattern recognition in a bucket. In *Advances in Artificial Life*, 588.

Fodor, K., Mellado, J. P. and Wilczek, M. (2019) On the role of large-scale updrafts and downdrafts in deviations from monin–obukhov similarity theory in free convection. *Boundary Layer Meteorol.*, 172, 371.

Fonda, E., Pandey, A., Schumacher, J. and Sreenivasan, K. R. (2019) Deep learning in turbulent convection networks. *Proc. Natl. Acad. Sci. U.S.A.*, 116, 8667.

Fujii, K. and Nakajima, K. (2017) Harnessing disordered-ensemble quantum dynamics for machine learning. *Phys. Rev. Applied*, 8, 024030.

Fujii, K. and Nakajima, K. (2020) Quantum reservoir computing: a reservoir approach toward quantum machine learning on near-term quantum devices. *arXiv:2011.04890*.

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G. and Yacalis, G. (2018) Could machine learning break the convection parameterization deadlock? *Geophys. Res. Lett.*, 45, 5742.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press, Cambridge, USA.

He, K., Zhang, X., Ren, S. and Sun, J. (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. *arXiv:1502.01852*.

Hermans, M. and Schrauwen, B. (2010) Memory in reservoirs for high dimensional input. *IJCNN*, 1.

Heyder, F. and Schumacher, J. (2021) Echo state network for two-dimensional turbulent moist rayleigh-bénard convection. *Phys. Rev. E*, 103, 053107.

Inubushi, M. and Goto, S. (2020) Transfer learning for nonlinear dynamics and its application to fluid turbulence. *Phys. Rev. E*, 102, 043301.
Jaeger, H. (2001) The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. GMD-Forschungszentrum Informationstechnik Technical Report, 148.

Jaeger, H. and Haas, H. (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science, 304, 78.

Kingma, P. D. and Ba, J. (2017) Adam: A method for stochastic optimization. arXiv:1412.6980.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012) Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst., 25.

LeMone, M. A., Angevine, W. M., Bretherton, C. S., Chen, F., Dudhia, J., Fedorovich, E., Katsaros, K. B., Lenschow, D. H., Mahrt, L., Patton, E. G., Sun, J., Tjernström, M. and Weil, J. (2019) 100 years of progress in boundary layer meteorology. Meteorol. Monogr., 59, 1.

Lu, Z., Pathak, J., Hunt, B. R., Girvan, M., Brockett, R. and Ott, E. (2017) Reservoir observers: model-free inference of unmeasured variables in chaotic systems. Chaos, 27, 041102.

Lukoševičius, M. (2012) A practical guide to applying echo state networks. Lect. Notes in Comput. Sci., 7700, 659.

Lukoševičius, M., Jaeger, H. and Schrauwen, B. (2012) Reservoir computing trends. Künstl. Intell., 26, 365.

Maass, W., Natschläger, T. and Markram, H. (2002) Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput., 14, 2531.

Moller, S., Resagk, C. and Cierpka, C. (2020) On the application of neural networks for temperature field measurements using thermoehromic liquid crystals. Exp. Fluids, 61, 111.

Nakajima, K. (2020) Physical reservoir computing: an introductory perspective. Jpn. J. Appl. Phys., 59, 060501.

O’Gorman, P. A. and Dwyer, J. G. (2018) Using machine learning to parameterize moist convection: potential for modeling of climate, climate change, and extreme events. J. Adv. Model Earth Sy., 10, 2548.

Pan, S. J. and Yang, Q. (2010) A survey on transfer learning. IEEE Trans. Knowl. Data Eng., 22, 1345.

Pandey, S. and Schumacher, J. (2020) Reservoir computing model of two-dimensional turbulent convection. Phys. Rev. Fluids, 5, 113506.

Pandey, S., Schumacher, J. and Sreenivasan, K. R. (2020) A perspective on machine learning in turbulent flows. J. Turbul., 21, 567.

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M. and Ott, E. (2017) Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. Chaos, 27, 121102.

Pathak, J., Wikner, A., Fussel, R., Chandra, S., Hunt, B. R., Girvan, M. and Ott, E. (2018) Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. Chaos, 28, 041101.

Pawar, S. and San, O. (2021) Data assimilation empowered neural network parametrizations for subgrid processes in geophysical flows. Phys. Rev. Fluids, 6, 050501.

Sirovich, L. (1987) Turbulence and the dynamics of coherent structures. i - coherent structures. Q. Appl. Math., 45, 561.

Sorjjan, Z. (1996) Numerical study of penetrative and "solid lid" nonpenetrative convective boundar layers. J. Atmos. Sci., 53, 101.

Srinivasan, P. A., Guastoni, L., Azizpour, H., Schlatter, P. and Vinuesa, R. (2019) Predictions of turbulent shear flows using deep neural networks. Phys. Rev. Fluids, 4, 054603.
Stull, R. B. (1988) *An introduction to boundary layer meteorology*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Tsunegi, S., Taniguchi, T., Nakajima, K., Miwa, S., Yakushiji, K., Fukushima, A., Yuasa, S. and Kubota, H. (2019) Physical reservoir computing based on spin torque oscillator with forced synchronization. *Appl. Phys. Lett.*, **114**, 164101.

Vandoorne, K., Dambre, J., Verstraeten, D., Schrauwen, B. and Bienstman, P. (2011) Parallel reservoir computing using optical amplifiers. *IEEE Trans. Neural Netw.*, **22**, 1469.

Wikner, A., Pathak, J., Hunt, B. R., Girvan, M., Arcomano, T., Szunyogh, I., Pomerance, A. and Ott, E. (2020) Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos*, **30**, 053111.

Wyngaard, J. C. (2010) *Turbulence in the atmosphere*. Cambridge University Press, Cambridge, UK.

Yildiz, I. B., Jaeger, H. and Kiebel, S. J. (2012) Re-visiting the echo state property. *Neural Netw.*, **35**, 1.

Zanna, L. and Bolton, T. (2020) Data-driven equation discovery of ocean mesoscale closures. *Geophys. Res. Lett.*, **47**, e2020GL088376.

Zilitinkevich, S. S. (1991) *Turbulent penetrative convection*. Avebury Technical.