M.Sc. Florian Heyder

born 17.11.1995 in Schwabach, Germany

Reduced Order Modeling of thermal convection flows: A Reservoir Computing approach

Dissertation
for award of the academic degree Doktoringenieuer (Dr.-Ing.)

carried out at the
Institute of Thermodynamics and Fluid Mechanics
and submitted at the
Department of Mechanical Engineering
University of Technology Ilmenau.

Date of submission: 26 September 2023
Date of doctoral defense: 20 February 2024

Reviewer 1: Prof. Dr. rer. nat. habil. Jörg Schumacher (Ph.D. adviser / Doktorvater)
Reviewer 2: Prof. Dr.-Ing. Patrick Mäder
Reviewer 3: Prof. Dr. rer. nat. Juan Pedro Mellado

DOI: 10.22032/dbt.59964
URN: urn:nbn:de:gbv:ilm1-2024000106

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Bei der Auswahl und Auswertung folgenden Materials haben mir die nachstehend aufgeführten Personen in der jeweils beschriebenen Weise unentgeltlich geholfen:

- M.Sc. Philipp Pfeffer, Institut für Thermo- und Fluiddynamik, TU Ilmenau:
  - Beitrag zu den Abschnitten 3.3.1 und 3.3.2:
    Bereitstellung der Quantum Reservoir Computing Ergebnisse
- Prof. Dr. Juan Pedro Mellado, Meteorologisches Institut, Universität Hamburg:
  - Beitrag zu den Abschnitten 5.1 und 5.2:
    Bereitstellung der numerischen Simulationsergebnisse

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch bewertet wird und gemäß §7 Abs. 10 der Promotionsordnung den Abbruch des Promotionsverfahrens zur Folge hat.

Ilmenau, den 21.09.2023

Florian Heyder

Abstract

This thesis explores the potential of machine learning (ML) algorithms to enhance subgrid-scale parameterizations in large-scale atmospheric simulations. Traditional approaches often rely on simplifications or computationally expensive methods. This work aims to introduce a more physically consistent and computationally efficient approach using Reservoir Computing (RC) and data reduction techniques to extract subgrid-scale features from direct numerical simulations (DNS) of thermal convection. To this end, the high-fidelity simulation data is pre-processed by a Proper Orthogonal Decomposition (POD) or an Autoencoder (AE) network to reduce the amount of data. An RC model is subsequently trained on this reduced data space to predict future flow states without solving the governing nonlinear equations of motion. The combined POD-RC model's predictions are thoroughly validated by original simulations. It is found that the model accurately replicates spatial organization, structural features, and low-order statistics of dry and moist convection flows, opening new ways for the dynamic parameterization of subgrid-scale transport in larger-scale circulation models. Furthermore, this work investigates the generalization property of an AE-RC model based on a flux-driven two-dimensional turbulent convection system. It is found that the machine learning model can correctly reproduce spatial features and the statistical properties of the physical fields. Finally, this work focuses on the parameterization of the convective boundary layer (CBL) by means of a Generative Adversarial Network (GAN), which is trained on high-fidelity DNS data of a three-dimensional CBL. It is shown that a physics-informed rescaling of the limited amount of training data enables the method to reproduce the CBL growth and the related pattern formation. The GAN results agree with standard mass-flux schemes and additionally provide the granule-type horizontal organization of the turbulent flow, which cannot be obtained with the mass-flux approach. Although the primary focus is not on implementing ML-based parameterization schemes in large-scale models, this work advances our understanding of the potential and limitations of the aforementioned models in the context of climate modeling and numerical weather prediction.

Zusammenfassung

In dieser Arbeit wird das Potenzial von Machine-Learning-Algorithmen (ML) zur Verbesserung der Parametrisierung von großskaligen atmosphärischen Simulationen untersucht. Herkömmliche Ansätze verwenden oft Vereinfachungen oder rechenintensive Methoden. Diese Arbeit beabsichtigt, einen physikalisch konsistenten und rechnerisch effizienten Ansatz einzuführen, der Reservoir Computing (RC) und Datenkompression nutzt, um subgitter-skalige Merkmale aus direkten numerischen Simulationen (DNS) der thermischen Konvektion zu extrahieren. Hierbei wird der hochaufgelöste Simulationsdatensatz zuerst durch Proper Orthogonal Decomposition (POD) oder ein Autoencoder-Netzwerk (AE) vorverarbeitet, um die Datenmenge zu reduzieren. Anschließend wird ein RC-Modell auf diesem reduzierten Datenraum trainiert, um zukünftige Strömungszustände ohne die Lösung der nichtlinearen Bewegungsgleichungen vorherzusagen. Die Vorhersagen des kombinierten POD-RC-Modells werden anhand der Originalsimulationen validiert. Das Modell reproduziert die strukturellen und statistischen Merkmale von trockenen und feuchten Konvektionsströmungen und eröffnet somit neue Wege für die dynamische Parametrisierung des subgrid-skaligen Transports in grob aufgelösten Zirkulationsmodellen. Des Weiteren untersucht die Studie die Verallgemeinerungseigenschaften eines AE-RC-Modells basierend auf einem wärmeflussgetriebenen zweidimensionalen turbulenten Konvektionssystem. Dabei zeigt sich, dass das AE-RC-Modell die räumliche Struktur und statistischen Eigenschaften der ungesehenen physikalischen Felder korrekt wiedergibt. Schließlich liegt der Fokus auf der Parametrisierung der konvektiven Grenzschicht (CBL) mithilfe eines Generative Adversarial Networks (GAN), das auf hochaufgelösten DNS-Daten einer dreidimensionalen CBL trainiert wird. Es wird gezeigt, dass die Methode durch eine physikalisch informierte Reskalierung der begrenzten Trainingsdaten in der Lage ist, das CBL-Wachstum und die damit verbundene Musterbildung zu reproduzieren. Die GAN-Ergebnisse stimmen mit Standard Mass-Flux Parametrisierungen überein und liefern zusätzlich die horizontale Anordnung der turbulenten Strömung, die mit dem Mass-Flux-Ansatz nicht erreicht werden kann. Obwohl die Implementierung von ML-basierten Parametrisierungsschemata in großskaligen Modellen nicht im Fokus steht, trägt diese Arbeit dazu bei, unser Verständnis des Potenzials und der Grenzen dieser Modelle im Kontext der Klimamodellierung und numerischen Wettervorhersage zu vertiefen.

Danksagung

Die Gefühlslage, in der sich ein Doktorand nach Abschluss seiner Promotion befindet, wird maßgeblich von den Menschen beeinflusst, mit denen er während seiner Promotionszeit interagiert hat. Dazu gehören neben dem Doktorvater bzw. der Doktormutter auch die Kollegen und Kolleginnen, mit denen er zu tun hat. Ich kann bereits im Voraus sagen, dass ich in all diesen Bereichen während meiner Promotion am Fachgebiet Strömungsmechanik der TU Ilmenau außerordentlich viel Glück hatte.

Prof. Jörg Schumacher war mein Doktorvater, also mein Betreuer bzw. Mentor. Zudem war er auch mein Vorgesetzter. Diese Konstellation bildet nicht unbedingt eine ideale Voraussetzung für ein harmonisches Mentor-Mentee-Verhältnis, was den meisten Promovierenden, wie auch mir, erst im Laufe der Promotion bewusst wird. Jörg Schumacher, allerdings, verkörperte beide Rollen in genau dem richtigen Verhältnis. Zu keiner Zeit fühlte ich mich unzureichend beraten oder alleingelassen. Seine Herangehensweise in der Betreuung und Zielsetzung von neuen Forschungsschwerpunkten empfand ich stets erfrischen. Schließlich möchte ich Jörg herzlich für die hervorragende Betreuung danken, sowie für seine gelassene und entschlossene Arbeitsweise, von der ich viel für mich mitnehmen konnte. Als Nächstes möchte ich Colette Wilhelm, der Teamassistentin am Fachgebiet Strömungsmechanik, meinen Dank aussprechen. Ihre Unterstützung bei allen universitären Bürokratieangelegenheiten war äußerst wertvoll. Bereits während meiner Zeit als Masterand am Fachgebiet wurde mir bewusst, dass man mit einem solchen Sekretariat gut aufgehoben ist. Ich bin Colette sehr dankbar für ihre herzliche Art und ihren professionellen Umgang mit den bürokratischen Anforderungen, ohne die meine Zeit als Doktorand um einiges komplizierter geworden wäre. Zudem möchte ich Henning Schwanbeck für seine flexiblen Beratungsleistungen im Zusammenhang mit dem Rechencluster der TU Ilmenau meinen Dank aussprechen. Ohne die Nutzung des Clusters hätten viele meiner Forschungsergebnisse nicht erzielt werden können.
Innerhalb des Kollegiums möchte ich zuallererst Philipp Pfeffer meinen Dank aussprechen, mit dem ich an zwei Veröffentlichungen zusammenarbeiten durfte. Unsere Diskussionen über maschinelles Lernen und Quantencomputing aus der Sicht eines anderen Doktoranden waren erfrischend und haben mir stets neue Ideen vermittelt. Ebenfalls möchte ich Theo Käufer meine Anerkennung aussprechen, mit dem ich die Betreuung von Studierenden in ihren Projektarbeiten übernehmen durfte. Diese gemeinsame Betreuung hat mich unter anderem gelehrt, wie man zielgerichtet studentische Arbeiten leitet. Philipp Teutsch gebührt mein Dank für die äußerst effektive und offene Zusammenarbeit im Rahmen des Zeiss-Projekts. Zudem konnte ich von ihm viele praktische Kniffe in der Programmierung neuronaler Netze in Python lernen. Ähnliches gilt für Johannes Viehweg, der mir einen reibungslosen Einstieg in das Thema des Reservoir Computings ermöglichte. Weiterhin möchte ich Dr. Thomas Boeck, Dr. Priyanka Maiti und Dr. Dmitry Krasnov für ihre wertvollen Ratschläge während meiner Zeit als Doktorand meinen aufrichtigen Dank aussprechen.
Ich danke Prof. Juan Pedro Mellado von der Universität Hamburg, mit dem ich an zwei Veröffentlichungen arbeiten durfte. Während dieser Zusammenarbeit konnte ich viel über die Simulation von Klima und Wetter lernen. Im Hinblick auf die Auswirkungen der Klimakatastrophe auf uns Menschen faszinierten mich besonders die Einblicke in den Stand der aktuellen Klimamodelle. Juan Pedros professionelle und offene Arbeitsweise, empfand ich stets inspirierend, was die Kooperationen jederzeit angenehm gestaltete. Zusätzlich möchte

ich Prof. Patrick Mäder für die bereichernden Gespräche während der Zeiss-Projekttreffen danken, bei denen ich viel über den aktuellen Stand des maschinellen Lernens erfahren konnte. Weiterhin danke ich Prof. Ludvig Lizana und Prof. Rikard Eriksson für die Zusammenarbeit in Umeå und die tolle Zeit im Norden Schwedens.
Ein besonderer Dank gebührt auch Stefan Heyder für seine kontinuierliche Beratung. Ob es um die Entscheidung zur Promotion, mathematische Fragen zur Statistik oder die nächste Kaffeebestellung ging – Stefans Ratschläge und seine eigene Erfahrung haben es mir stets ermöglicht, etwas Neues zu lernen. In diesem Sinne 'Thanks for always being older than me'.
Des Weiteren danke ich Tara Bellut, die mir Kraft und Durchhaltevermögen während meiner Promotion und auch während der Pandemie gab. Ich schätze mich wirklich glücklich, eine solche Partnerin an der Seite zu haben. Abschließend möchte ich selbstverständlich meinen Eltern meinen Dank aussprechen. Sie haben mich immer in meinen Entscheidungen bestärkt und standen mir stets mit offenen Ohren zur Seite.

Schlussendlich noch ein paar Worte über die Wissenschaft und meine Zukunft nach der Promotion. Mir hat Wissenschaft unablässig Spaß gemacht. Klar gab es manchmal Durchhänger, aber im Großen und Ganzen bin ich immer zufrieden morgens zur Arbeit gegangen. Die Entscheidung, keinen Post-Doc zu machen, fiel mir nicht einfach. Das Arbeiten in der Wissenschaft hat viele Vorteile. Vor allem allerdings reizt mich doch der Anspruch der Grundlagenforschung, dass ihre Ergebnisse nicht gewinnbringend, sondern lediglich dem Zwecke der Wissenschaft dienen sollen. Gerne würde ich mich noch weitere Jahre der tatsächlichen Simulation von Klima- und Wetterereignissen widmen. Desto mehr ich über diese Themen in den letzten Jahren gelernt habe, desto mehr habe ich mich für ihre Details interessiert. Schlussendlich habe ich mich nun aber gegen eine solche Karriere entschieden. Das soll aber nicht bedeuten, dass ich mich nicht weiter mit den Themen Wetter und Klima beschäftigen werde. Zusammenfassend waren die letzten drei Jahre, trotz globaler Krisen, eine Anhäufung von tollen Erfahrungen, an die ich mich gerne zurückerinnern werde.

Florian Heyder
Ilmenau, September 2023

Contents

Nomenclature

Abbreviations

ABL	Atmospheric boundary layer
AE	Autoencoder
CBL	Convective boundary layer
CNN	Convolutional neural network
CRCP	Cloud-Resolving Convective Parameterization
CSA	Condensation in saturated ascent
DNS	Direct numerical simulation
ED	Eddy diffusivity
EDMF	Eddy diffusivity mass flux
ESN	Echo State Network
ESP	Echo state property
FFNN	Feed forward neural network
GAN	Generative Adversarial Network
GCM	General Circulation Model / Global Climate Model
GLL	Gauss-Lobatto-Legendre nodes
GT	Ground truth
LOM	Low-order model
LSM	Liquid State Machine
MF	Mass-flux
ML	Machine learning
MRBC	Moist Rayleigh-Bénard convection
MSE	Mean squared error
NARE	Normalized average relative error
NN	Neural network
NN	Numerical weather prediction

PCA	Principle Component Analysis
PDF	Probability density function
PI	Physics informed
POD	Proper Orthogonal Decomposition
QC	Quantum computing
QRC	Quantum Reservoir Computing
RBC	Rayleigh-Bénard convection
RC	Reservoir Computing
RNN	Recurrent neural network
ROM	Reduced Order Model
SP	Super-parameterization
WGAN	Wasserstein Generative Adversarial Network

Latin Symbols

$(\cdot)$	Discriminator/ critic network (GAN)
$(\cdot)$	Generator network (GAN)
$\hat{Y}$	reservoir target output matrix
$\mathbf{r}$	reservoir state
$\mathbf{u}$	velocity field
$\mathbf{w}, w$	neural network parameter/ weight
$\mathbf{x}$	reservoir input, critic input
$\mathbf{x}_G$	generator output
$\mathbf{y}$	reservoir output
$\mathcal{E}_{\mathrm{MSE}}$	mean squared error
$\mathcal{E}_{\mathrm{NARE}}$	normalized average relative error
$\mathcal{E}_{\mathrm{NRMSE}}$	normalized root mean squared error
$\mathcal{H}_T$	temperature scale-height of the atmosphere
$\mathcal{L}$	loss function
$\mathbf{y}$	reservoir output signal
$\overline{\mathcal{E}_{\mathrm{NARE}}}$	average normalized average relative error
a_k	k^{th} POD time coefficient or k^{th} AE latent space variable

b	buoyancy
B_0	bottom buoyancy flux
B_1	top buoyancy flux
b_{bg}	buoyancy background profile
C_1	adjustable parameter of EDMF parameterization
c_{in}	reservoir input bias
c_{out}	reservoir output bias
D_{r}	reservoir density
E_{kin}	turbulent kinetic energy
$F_{u_z'b'}$	flattness of the buoyancy flux distribution
H	vertical extent of fluid domain
h	height of the convective boundary layer
L	horizontal extent of fluid domain
L_0	Ozmidov length scale
N	polynomial order inside of the spectral element method
n	discrete time coordinate (time step)
N_0	buoyancy frequency
N_e	number of spectral elements
N_x, N_y, N_z	number of grid points in x, y, z direction on an interpolated grid
N_{in}	reservoir input dimension
N_{out}	reservoir output dimension
N_{r}	reservoir dimension
p, p', p_0, p_1	pressure, kinematic pressure, adiabatic ground state pressure, superadiabatic correction pressure
R	reservoir state matrix
r	relative Rayleigh number parameter of the Lorenz model
R^*	ideal gas constant divided by molecular mass of gas constituents
s_i	i^{th} eigenvalue of the POD covariance matrix
$S_{u_z'b'}$	skewness of the buoyancy flux distribution
T, T_0, T_1	temperature, adiabatic ground state temperature, superadiabatic correction temperature

t	continous time coordinate
t_f	free fall time
$U(\cdot,\cdot,\cdot), U(\cdot,\cdot)$	unitary transformation
U_f	free fall velocity
u_x, u_y	horizontal velocity fields
u_z	vertical velocity field
x, y, z	cartesian coordinates
X	data matrix

Greek Symbols

α	thermal expansion coefficient at constant pressure
β	buoyancy-flux ratio
$\Delta b, \Delta T$	buoyancy, temperature difference
δ	geometric parameter of the Lorenz model
ϵ	superadiabaticity parameter in anelastic approximation
η	learning rate
Γ	aspect ratio of fluid domain
γ_{r}	ESN leaking rate
κ	thermal diffusivity
λ_{r}	ESN regression parameter
μ	adjustable parameter of EDMF parameterization
ν	kinematic viscosity
Ω	spatial domain of the convective flow
ω	vorticity
Φ, ϕ	spatial POD modes, spatial modes matrix
ρ, ρ_0, ρ_1	fluid mass density, adiabatic ground state density, superadiabatic correction density
Σ	eigen value matrix
σ	nonlinear activation function
σ_{r}	ESN input scaling
θ	temperature deviation from linear conduction profile
ε	entrainment rate parameter in the EDMF model

ϱ_{r}	ESN spectral radius
ξ	random variable
Ξ_{r}	ESN hyperparameter set
ζ	stream function

Dimensionless numbers

Ma	Mach number
Nu	Nusselt number for Dirichlet boundary conditions
Nu_M	moist Nusselt number (Dirichlet boundary conditions)
Nu_N	Nusselt number for Neumann boundary conditions
Pr	Prandtl number
Ra	Rayleigh number
Ra_c	convective Rayleigh number
$\mathrm{Ra}_{\mathrm{crit}}$	critical Rayleigh number of the onset of convection
Re	Reynolds number
Re_0	buoyancy Reynolds number
Re_M	moist Reynolds number

Mathematical Symbols

δ_{ij}	Kronecker delta
$\lvert\psi\rangle$	quantum state in Dirac notation
$\langle\cdot\rangle_x$	mean over variable x
$\mathbb{C}$	complex numbers
$\mathbb{R}$	real numbers
$\mathcal{F}$	Fourier transform
$\mathcal{H}$	hilbert space
$\mathcal{N}(\mu,\sigma)$	Normal distribution with mean μ, standard deviation σ
$\mathcal{U}[a,b]$	Uniform distribution on the interval $[a,b]$
∇_x	differential operator w.r.t. x
$\otimes$	tensor product
σ_p	standard deviation of the variable p
$\tanh(\cdot)$	element-wise hyperbolic tangent
A^T	matrix inverse
A^T	matrix transpose
I	identity matrix

Herzlich liebe ich die Physik.
Es ist so eine Art persönlicher Liebe, wie gegen einen Menschen, dem man sehr viel verdankt.

- Lise Meitner (1879-1968),
Austrian physicist and first female physics professor in Germany

In God we trust. All others must bring data.

- William Edwards Deming (1900-1993),
US American statistician

Chapter 1

Introduction

1.1 Thermal convection in our atmosphere

Thermal convection describes the fluid motion driven by local temperature differences between fluid layers. The resulting buoyant flow leads to heat transport and can even become turbulent for sufficiently large temperature gradients. It is a widespread, ubiquitous process in nature, evident in occurrences such as the creation of clouds in Earth's atmosphere, the emergence of patterns on the Sun's surface [1], and the gradual movement of solid mantle rock within the Earth's crust [2]. Furthermore, it affects the energy distribution within the fluid system by distributing heat via up- and downdrafts. It is often coupled to other physical processes such as radiation [3] and phase change [4], which can act as vents for excess heat. Most notably, this phenomenon is conspicuous in Earth's atmosphere, where the solar energy received through radiation gets converted into the kinetic energy of air masses. This convective fluid motion can lead to pressure differences, which initiate horizontal winds. Moreover, it promotes the transport of water vapor and other greenhouse gases like CO_2 that interact with long-wave radiation emitted from our planet's surface. Hence, thermal convection is a central process in Earth's complex climate system.

1.1.1 Rayleigh-Bénard convection

The system that most directly encapsulates the characteristics of thermal convection is Rayleigh-Bénard convection (RBC) [5], illustrated in Figure 1.1. Within the Rayleigh-Bénard framework of convection, a fluid is confined between two horizontal, non-penetrable plates. These plates maintain a constant temperature, with the lower plate being set at a higher value than the upper one. Due to the thermal expansion of fluid parcels at the bottom and thermal contraction near the top, the fluid becomes buoyant, and a fluid motion is induced. This seemingly simple system has been studied extensively over a century and dates back to the experiments of Henri Bénard [6] in the year 1900. Note that Bénard's setup differed from the now commonly used model, as he studied the convection patterns arising in a thin fluid layer over a free surface[1]. The second major contributor was Lord Rayleigh, who explained the transition of pure conduction to convection in his linear stability analysis in 1916 [8]. He introduced an upper lid to the setup of Bénard and shaped the Rayleigh-Bénard model, as it is known today. The configurations of the most basic Rayleigh-Bénard convection setups frequently exhibit rectangular or cylindrical forms, enabling a concentrated examination of the convection process. Bénard's and Rayleigh's work paved the way for a paradigm that has significantly advanced our understanding of heat transport, pattern formation [9–13], turbulence [14] and atmospheric convection processes.

[1]His setup is better described by the Bénard-Marangoni system, where the surface tension at the top is the driving force for the fluid motion [7].

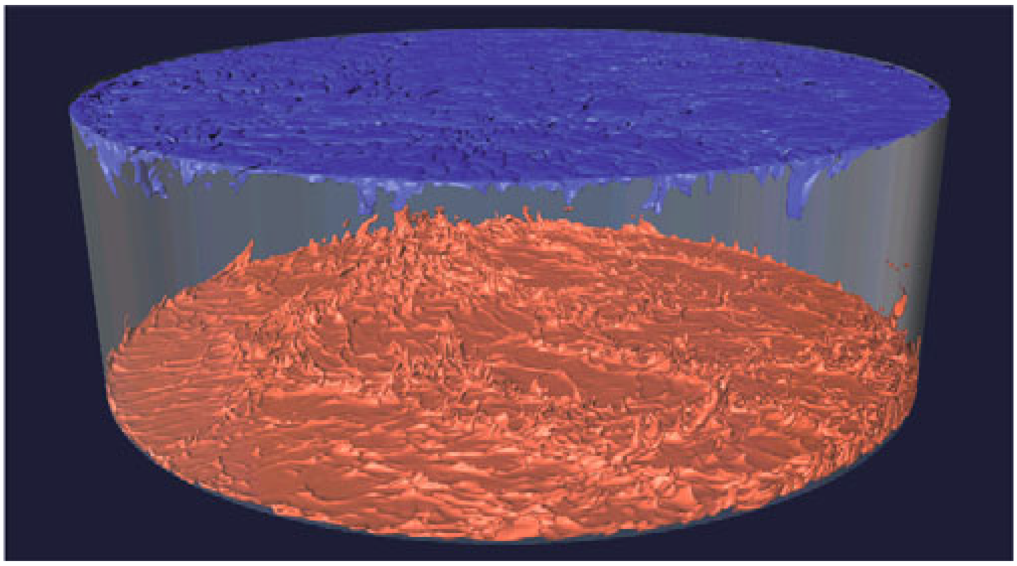

Figure 1.1: **Rayleigh-Bénard convection** in a cylindrical tank. A fluid trapped between two impermeable plates is heated from below and cooled from the top. The fluid near the top and bottom plates become buoyant, leading to warm and cold parcels rising and falling. From ref. [14].

1.1.2 The atmospheric boundary layer

The most prominent example of convection in our atmosphere is situated in the lower part of the troposphere, which is in direct contact with the Earth's surface: the atmospheric boundary layer (ABL). In this region, anthropogenic activities take place, and weather phenomena directly impact our everyday life. The state of the ABL highly depends on the time of day. During the daytime, the incoming solar radiation heats the ground. In turn, the air layers above the ground become buoyant and rise. Convective fluid motion is initiated. Hence this state of the ABL is also referred to as the convective boundary layer (CBL). The convective flow eventually becomes turbulent and creates a well-mixed layer above the ground that can extend from tens of meters in the morning to $1 - 3$km in the afternoon. In this daytime regime, the ABL can be divided into three layers: an unstable surface layer (about 10% of the CBL thickness [15]), a mixed layer in the bulk (about 35 - 80% of the CBL thickness [16]) and the boundary between the CBL and the free atmosphere: the entrainment zone. After sunset, when the incoming solar radiation has ceased, convection stops, and a stably stratified nocturnal boundary layer forms near the surface. Above, the remanence of the mixed layer is found in terms of a residual layer that is weakly stratified. When the Sun rises, the CBL starts to grow into the residual layer, and the cycle begins again. Figure 1.2 displays the diurnal cycle of the ABL in terms of backscattering measurements of aerosols at different times of the day. It can be observed that during the growth of the CBL, thermals overshoot into the capping inversion layer. Consequently, part of the vertical motion is deflected, leading to the entrainment of warm, non-turbulent air into the ABL. This has implications for the energy and material transport inside the ABL. Scalar variables like pollutants, temperature, or moisture are well-mixed as the turbulent motion smoothens their gradients. The free atmosphere above often acts as a cork for these variables, isolating them from reaching higher levels of the troposphere. Moreover, clouds can form at the inversion line if moisture is present, leading to shallow cumulus clouds. On the other hand, deep convective clouds like the cumulonimbus can reach far into the upper troposphere and cause heavy rain and storms. Therefore, the description of this part of the atmosphere is crucial for weather forecasts and the global circulation.

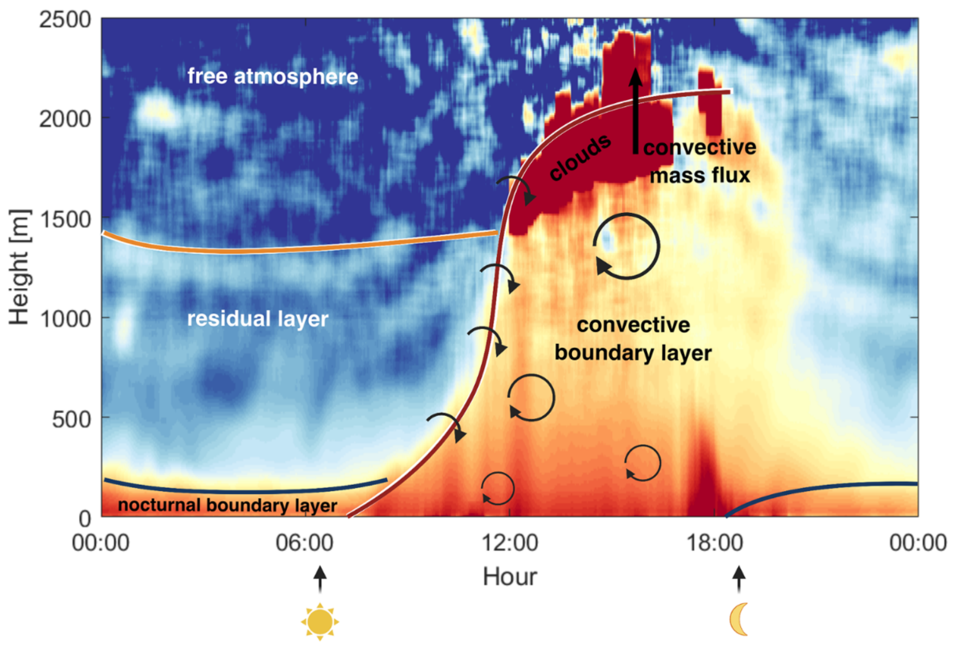

Figure 1.2: Diurnal cycle of the **atmospheric boundary layer**. Shown are lidar backscatter measurements of aerosols at the Southern Great Plains Atmospheric Observatory in Oklahoma, U.S.A. Red signifies high and blue low backscattering amplitudes. Sunrise and sunset are indicated on the time axis. Before sunrise, a stable nocturnal boundary layer and a residual layer can be observed. After sunrise, Earth's surface is heated by solar radiation. The heated ground subsequently initiates a convective fluid motion. The resulting convective boundary layer grows into the stably stratified remanence of the residual layer. The buoyant flow eventually becomes turbulent, creating a well-mixed layer above the warm surface. At noon, moist air vapor condenses at the top of the CBL, leading to the formation of clouds. From ref. [15].

1.1.3 Parameterization of thermal convection in our atmosphere

While the equations that describe the physical processes, like convection, inside the ABL are known, they can not directly be incorporated into a General Circulation Model (GCM) that numerically models Earth's climate for several decades. This is due to the climate system's vast range of spatial and temporal scales. Weather and climate phenomena range from microphysical processes like droplet formation inside clouds to large-scale precipitation events like the Madden-Julien-Oscillation over the Indian and Pacific Oceans. Figure 1.3 gives an overview of the spatial and temporal variability of the atmosphere. The classification of atmospheric convection places it in the realm referred to as the *mesoscale*, which roughly spans spatial scales ranging from 10 to 100 kilometers. This classification positions it between the microscale and the macroscale. As discussed above, these mesoscale processes substantially impact the atmosphere's energy budget and, therefore, its general circulation. Nevertheless, it is apparent that a numerical simulation representing all of the scales shown in Figure 1.3, while performing multi-decade or even multi-century predictions, is still far away from being feasible, even with the current computational capabilities. Nowadays, the

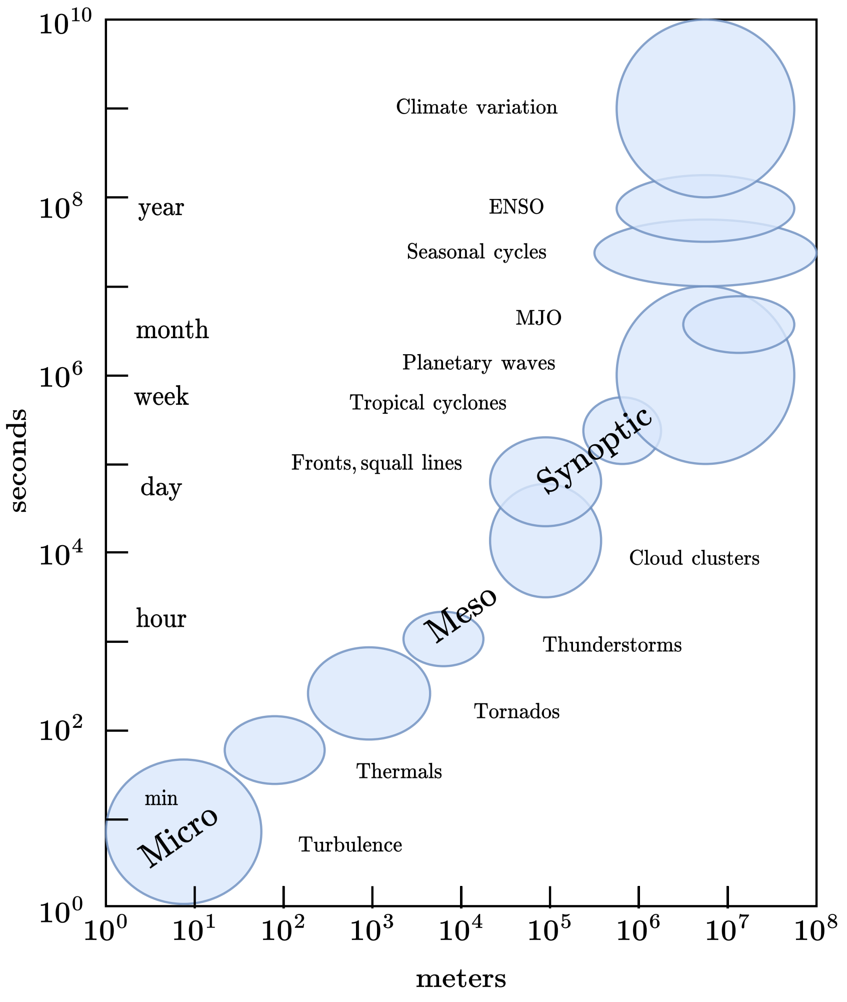

Figure 1.3: **Spatial and temporal scales** of Earth's climate system. The scales vary over vast amounts of magnitudes, rendering a numerical climate model, which represents all of these processes, unfeasible for the near future. Adapted from ref. [17].

horizontal resolution of climate models[2] ranges from $\sim$ 100km down to $\sim$ 40 $-$ 50km [19] or even $\sim$ 10km [20] which only partially covers horizontal turbulent structures in the ABL with a size of about 1 $-$ 3km [21]. Insufficient numerical resolution is a problem that has existed since the beginning of climate simulations. Traditionally, subgrid-scale parameterizations, theories that model the impact of the unresolved scales on the resolved scales, are used to take small-scale effects into account. To understand the concept of subgrid-scale parameterizations, let's consider a scalar variable $\varphi(\mathbf{x}, t)$, e.g., temperature or humidity, which follows the prognostic equation:

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\mathbf{u}\varphi) = q. \tag{1.1}$$

Here $\mathbf{u} = (u_x, u_y, u_z)^T$ denotes the fluid velocity field, $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T$ the differential operator and $\frac{\partial}{\partial t}$ the partial time derivative. The term q comprises all sources that contribute to a change of φ, like diffusion or radiation. Now consider that we seek to solve for φ in an atmospheric model, which can not resolve all relevant physical spatial scales for the above reasons. Therefore, the numerical grid on which we would like to solve for φ will be too coarse to resolve all relevant scales of turbulence. This model will then resolve merely the box-mean or filtered versions of φ, $\mathbf{u}$ and q, denoted by $\langle\varphi\rangle$, $\langle\mathbf{u}\rangle$ and $\langle q\rangle$. Applying the filter to eq.(1.1) yields

$$\frac{\partial \langle\varphi\rangle}{\partial t} + \nabla \cdot (\langle\mathbf{u}\varphi\rangle) = \langle q\rangle. \tag{1.2}$$

The term $\langle\mathbf{u}\varphi\rangle$ poses a problem, as it is an unclosed expression. However, eq.(1.2) can be rearranged to form

$$\frac{\partial \langle\varphi\rangle}{\partial t} + \nabla \cdot (\langle\mathbf{u}\rangle\langle\varphi\rangle) = \langle q\rangle + \nabla \cdot \mathbf{F}^{\mathrm{SGS}} \tag{1.3}$$

where the subgrid-scale (SGS) flux $\mathbf{F}^{\mathrm{SGS}} = \langle\mathbf{u}\rangle\langle\varphi\rangle - \langle\mathbf{u}\varphi\rangle$ has been introduced. In this way, the unresolved (small) scales act as a source or forcing term in the equations of motion of the resolved (large) scales. Finally, the goal of subgrid-scale parameterization is to find a closed form of $\mathbf{F}^{\mathrm{SGS}}$, i.e., an expression in terms of $\langle\varphi\rangle, \langle\mathbf{u}\rangle$. In the following, some common parameterization models are briefly discussed.

One of the simplest parameterization schemes is the eddy-diffusivity approach or K-Theory, which dates back to Boussinesq in 1877 [22]. It assumes that the SGS flux follows the local gradient of the scalar in order to homogenize the flow by mixing

$$\mathbf{F}^{\mathrm{SGS}} = -K_\varphi \nabla\langle\varphi\rangle. \tag{1.4}$$

Here K_φ is the eddy-diffusivity parameter to φ modeled depending on the physical interpretation of φ and the stability of the fluid layer [16]. It is often a function of the vertical position and the velocity field. Examples include mixing length models [23], Smagorinsky models [24], and the K-profile method [25]. Further, K-theory is commonly used in modeling the surface layer in the ABL [26–29]. A drawback of this parameterization technique is that

[2]Note that even weather forecast models which run only for a few days and therefore permit a smaller grid spacing of $\sim$ 6km, $\sim$ 2km [18] run into problems. As the resolution of these models becomes even higher, eventually reaching the size of the coherent structures of the ABL, turbulence will be partially resolved. This so-called "grey zone" poses a problem as the assumptions of common parameterization schemes break down.

it fails when large eddies are present in the flow. From eq.(1.4), it can be seen that counter-gradient transport of φ, e.g., near the capping ABL inversion, can not be represented. Furthermore, regions where the gradient vanishes, as is the case in the bulk of the ABL, are also not captured by the model.

This sets the stage for another established approach: the mass-flux parameterization [30–32]. This scheme decomposes the columns of the convection domain into strong thermals and a complementary quiescent environment. The subgrid-scale flux can then be written as

$$\mathbf{F}^{\mathrm{SGS}} = M_\varphi \left(\langle \varphi \rangle_u - \langle \varphi \rangle \right), \tag{1.5}$$

where φ_u is the average over the values φ associated with strong updrafts [33] and M_φ is the mass-flux parameter. This approach makes use of the fact that strong, narrow updrafts dominate vertical transport, while at each height, most of the horizontal plane is covered by slowly subsiding air. It is therefore only necessary to model the mass-flux M_φ and the updraft fields $\langle u_z \rangle_u$, $\langle \varphi \rangle_u$ by a plume model to represent the subgrid-scale transport of φ. This scheme is often employed for the convective transport inside cumulus clouds [34–36]. While this method allows scalar transport in regions where the mean gradient vanishes, it does not capture the spatial organization of individual plumes that are the basis for the formation of clouds. Finally, the unified eddy-diffusivity mass-flux (EDMF) parameterization [33] makes use of both the aforementioned methods and can represent local gradient and non-local counter-gradient transport. Despite their simplicity, these approaches have greatly improved climate [29] and weather [29, 37] simulations.

The schemes above have been around for decades and have been used extensively in numerical weather prediction (NWP) and Global Climate Models. Their emergence can be traced back to when computational capabilities for weather and climate predictions were limited. Therefore, more complex parameterizations are built upon the existing schemes by adding more and more physical processes. In the scientific cloud community, this has led to several critical arguments against solely focusing on this path of parameterization development. In [38], Arakawa points out that due to historical reasons, the development of parameterizations of cumuliform and stratiform clouds has made the simultaneous representation of both cloud formations in a single model incompatible. The lack of a unified general framework "[...] represents the most serious practical problem in the conventional approach of formulating model physics" [38]. Future parameterizations should allow all modeled physics to be represented in the same framework. Moreover, 20 years ago, conventional cloud parameterizations' progress rate was labeled "unacceptably slow" [39]. Randall *et al.* argue that the convection processes in our atmosphere are inherently complicated, and including more and more prognostic variables of such processes into new parameterization schemes will not increase their physical expressivity. In addition to the gradually evolving conventional approaches, it is essential to explore novel avenues of parameterization. One such novel scheme was the Cloud-Resolving Convective Parameterization (CRCP) introduced by Grabowski and Smolarkiewicz [41, 42] in 1999 and 2001. Their model intended to resolve some of the small-scale features explicitly, therefore directly computing the SGS flux $\mathbf{F}^{\mathrm{SGS}}$ in expression (1.3) from a small-scale-resolving simulation that is synchronized with the climate model. CRCP uses an embedded lateral two-dimensional cloud-resolving model in the vertical columns of a large-scale General Circulation Model. The concept can be seen in Fig. 1.4. The two-dimensional model does not fill the whole large-scale grid cell but aligns with the vertical parent grid while increasing the resolution in one of the lateral directions (the original work in [42] used an east-west orientation). Both models then interact via forcing terms similar to the one in 1.3, where $\mathbf{F}^{\mathrm{SGS}}$ comprises the grid-box averages of the small-scale thermodynamics. Another flux term in the equations of motion of the small scales

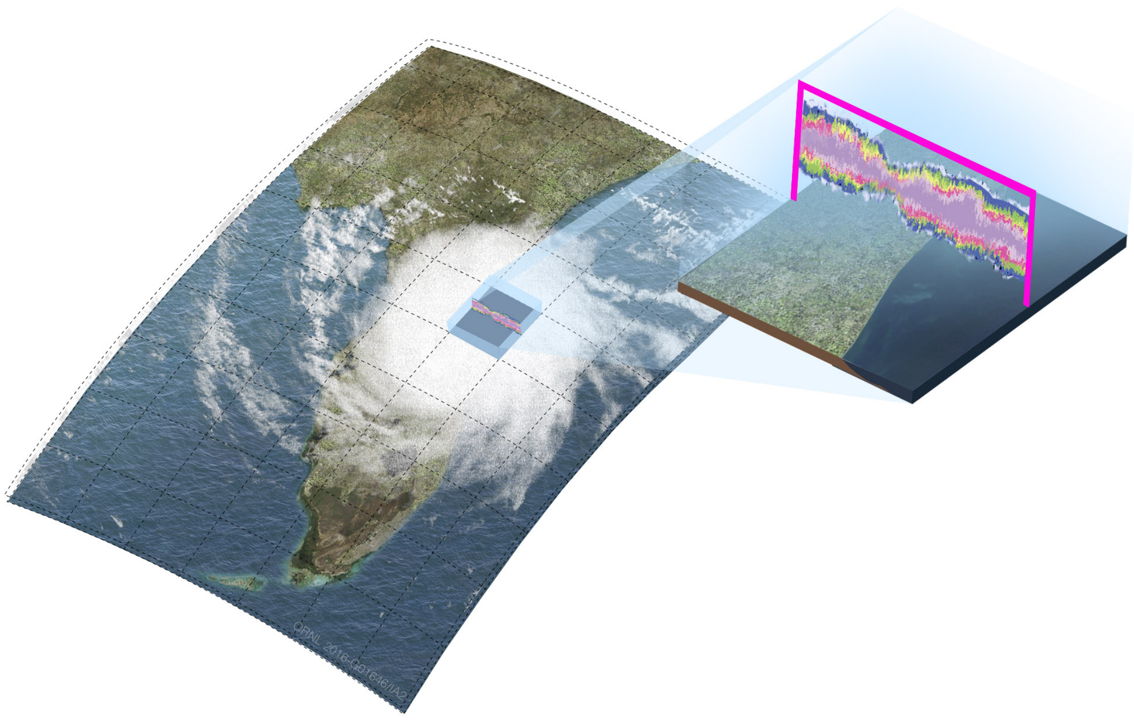

Figure 1.4: **Cloud-resolving convective parameterizaton** or super-parameterization concept. In each grid column of a General Circulation Model sits a two-dimensional cloud-resolving model that takes on the role of conventional parameterization schemes. Both models are coupled through lateral averages of the fine-scale model and large-scale drag from the GCM. From ref. [40].

introduces a forcing by the advective tendencies of the large-scale GCM. In this way, both simulations exchange information, and the prognostic variables of the large scale and averages of the small scale relax onto each other. In the end, the objective is to ensure that, for any coupled quantity, the horizontal average $\langle\cdot\rangle_{x,y}$ in the small-scale model at a height z corresponds to the value at the same height in the large-scale model. Hence, the vertical properties of both flows become crucial coupling parameters. The CRCP scheme was first used in a climate model by Khairoutdinov and Randall [43], where it was re-labeled super-parameterization (SP). To this day, SP is widely used [44–48]. While this approach has the downside of a higher computational cost than its conventional counterparts, it leads to more physically realistic predictions as parts of the sub-grid scale are explicitly resolved. Further, it is a natural first step towards a global cloud-resolving model.

Both parameterization frameworks introduced above have advantages: while conventional parameterization methods are computationally lightweight, super-parameterizations explicitly resolve mesoscale flow features. However, while the former lacks physical expressiveness, the latter comes at the computational cost of running a small-scale-resolving simulation in each GCM grid cell, making a multi-century simulation unfeasible. This begs the question: Can thermal convection processes be represented at low computational cost while retaining physical accuracy?
This lays the foundation for the current thesis, which seeks to address this inquiry using a machine learning-driven model.

1.2 A gentle introduction to neural networks

The undeniable success of machine learning (ML) and artificial neural networks (NN) during the last decade has immensely impacted our everyday lives. Common examples include recommendation systems in streaming services [49] or web shops, language processing in the form of translator apps [50] and voice assistants [51] as well as computer vision tasks like face recognition in online social networking platforms [52]. Furthermore, ML has sparked new research directions in natural science and engineering. After careful preparation of the numerical data, disciplines that naturally have to deal with large amounts of data can now leverage the potential of machine learning models. ML summarizes many classes of computational models that all have in common that they are tasked with performing a particular job without explicitly being programmed to do so. They rather self-teach or learn the task themselves by observing, primarily large, sets of data. In this way, these models learn by "gathering knowledge from experience" and therefore "avoid[...] the need for human operators" [53].

A big part of ML's success story stems from the concept of artificial neural networks and the field of *Deep Learning*. Motivated by the architecture of biological neural networks inside animal brains, these models are built from several connected units, called neurons or perceptrons [54–56]. Figure 1.5a) shows the concept of a single neuron. It takes an input signal $\mathbf{x}$ and produces an output $\mathbf{y}$ that is a nonlinear function of the weighted sum of its inputs. This is similar to biological neurons, which receive electrical impulses, become activated, and "fire" an output signal if the input signal surpasses a specific threshold value. The weights w_i of a perceptron are adaptable parameters resembling its ability to learn. Stacking several such artificial neurons and providing them with the input signals results in a single-layer network, shown in Figure 1.5b). If several such layers are followed by each other (see Figure 1.5c)), then the output of the current neuron layer is the input of the next layer, s.t. the input information is processed in a feed-forward fashion. Even though there is no clear def-

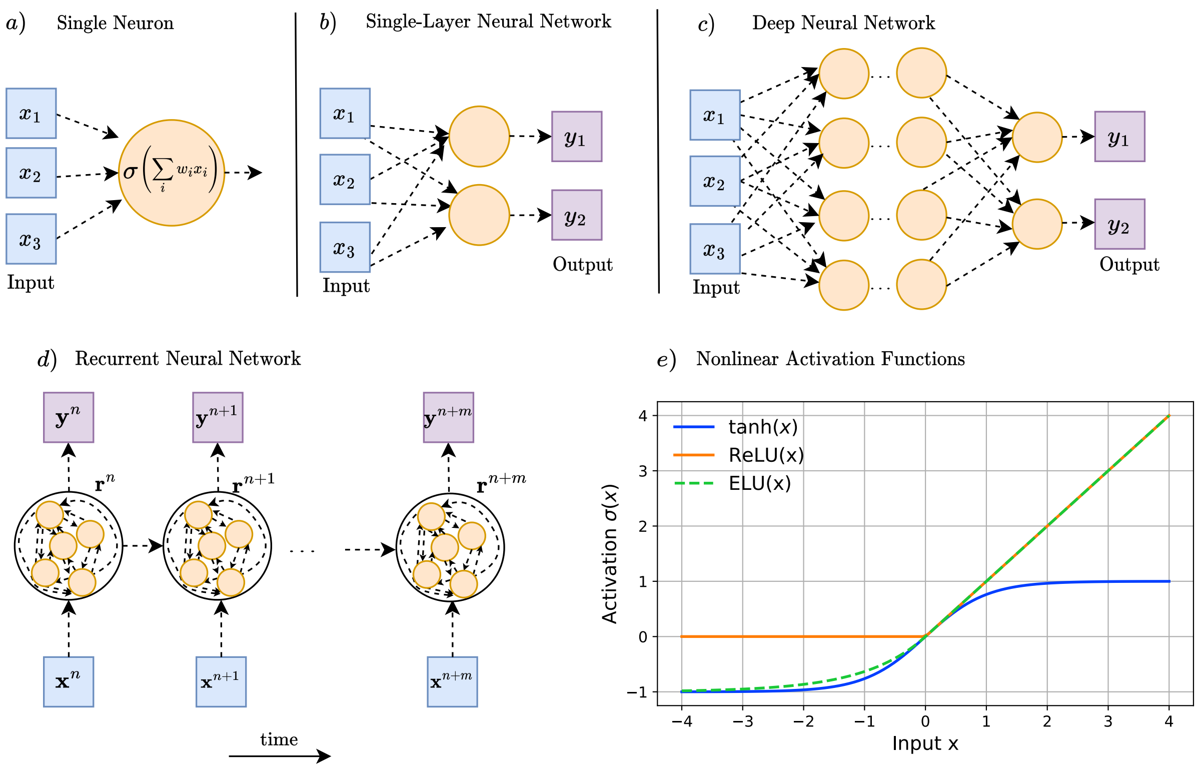

Figure 1.5: **Neural network building blocks.** (a): a single neuron (orange) receives a numerical input signal x_1, x_2, x_3 of three values. The neuron computes the weighted sum of the three inputs, applies a nonlinear activation function $\sigma(\cdot)$ (see e) to the sum, and outputs the result. Note that in practice, an additional scalar bias term is added to the sum before $\sigma(\cdot)$ is applied. (b): two neurons arranged in a single layer. Every neuron receives all information on the input and outputs an individual signal y_1 and y_2. (c): a multi-layered deep neural network. Four single-layer neurons receive the input signal. The output of each neuron is then passed to the neurons in the next layer. The final layer consists of two neurons, which result in two numerical output values y_1, y_2. (d): a recurrent neural network consists of a reservoir of connected neurons. It possesses an internal state $\mathbf{r}$, which is dynamically updated at time step n. This new state is then used in the next RNN run. (e) Examples of commonly used nonlinear activation functions σ.

inition, often, the neural network is already considered deep if the number of layers exceeds two. By applying a nonlinear activation function to the output of each neuron, the network's output $\mathbf{y}$ becomes a complex, highly parameterized representation of the input $\mathbf{x}$. Note that without this transformation at each neural output, the network output would behave as a single neuron, as $\mathbf{y}$ would again be a linear combination of the $\mathbf{x}$. Commonly used nonlinear activation functions are shown in 1.5e). The combination of "squashing" activation functions and sufficiently deep, fully connected layers of neurons enables the network to approximate "any continuous function on a closed and bounded subset of $\mathbb{R}^n$" [53]. This fact is known as the *universal approximation theorem* [57, 58].

Now that the feed-forward neural network model has been introduced let's consider how these networks learn. A trained NN corresponds to fixed weights and biases optimized in a preceding training phase on a prepared training dataset. Training is done by minimizing a loss or cost function $\mathcal{L}$ and adapting the weights and biases of the network. The loss function is task-specific and usually compares the network's output $\mathbf{y}$ with user-labeled ground truth (GT) target data $\hat{\mathbf{y}}$. This methodology is denoted as supervised learning. For example, take the mean squared error (MSE) loss for a dataset with N samples

$$\mathcal{L}_{\mathbf{w}}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 . \tag{1.6}$$

It measures the quality of the model output in a least-square sense compared to the ground truth. The subscript indicates that the loss is parameterized by the weights $\mathbf{w} = (w_1, w_2, ...)$ of the NN. Learning then requires minimizing this scalar loss function. This, on the other hand, requires a high-dimensional optimization procedure for all parameters $\mathbf{w}$ of the NN. For this, one relies on the gradient descent algorithm [59], an iterative process where one follows down the steepest direction of the loss-gradient w.r.t. the network parameters until a minimum of $\mathcal{L}_{\mathbf{w}}$ is reached. Then, the updated or learned weights, after $n+1$ optimization steps (or epochs), are given by

$$\mathbf{w}^{n+1} = \mathbf{w}^{n} - \eta \nabla_{\mathbf{w}} \mathcal{L}_{\mathbf{w}}, \tag{1.7}$$

where $\nabla_{\mathbf{w}}$ is the gradient w.r.t. to the NN weights, and η is the learning rate, a hyperparameter that defines their update speed. It is common to perform this updating step not for each of the N samples but for batches, i.e., subsets of the training dataset. This procedure is referred to as minibatch gradient descent and has two advantages. First, it increases the computational efficiency, as the amount of update steps after which the whole training dataset has been used is drastically reduced compared to updating after each sample. Secondly, updating the weights for multiple batches of the training set introduces noise, which helps escape local minima in the loss landscape, similar to stochastic gradient descent. Nowadays, advanced versions of eq. (1.7) have greatly improved the performance and stability of neural network training [60, 61].

Note that equation (1.7) requires the knowledge of the the gradients $\frac{\partial \mathcal{L}_{\mathbf{w}}}{\partial w_i}$ of each layer. However, as the output of each layer is the input to the next, the gradient w.r.t. of an initial layer requires the gradient of all layers succeeding it. This is equivalent to the chain rule, i.e., finding the derivative of a nested function $f(g(x))$ w.r.t. the inner-most argument x, which can be written as

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}. \tag{1.8}$$

Likewise, the output of the network, and hence $\mathcal{L}_\mathbf{w}$ is a nested function of each weight of the neural network. Therefore, the necessary term in eq. (1.7) can be computed similarly to (1.8). Take, for example, the loss gradient w.r.t. to a weight w_i in the l-th layer of a NN with a total of L layers. Let $\mathbf{y}^{(i)}$ be the output of the i-th layer. The gradient $\frac{\partial \mathcal{L}_\mathbf{w}}{\partial w_i}$ is then obtained by

- recording the gradient of the output $\frac{\partial \mathbf{y}^{(l)}}{\partial w_i}$ in the l-th layer
- recording the gradient $\frac{\partial \mathbf{y}^{(l+1)}}{\partial \mathbf{y}^{(l)}}$, i.e. the gradient of the subsequent layer's output w.r.t. to its input
- recording these gradients until the last layer's output $\frac{\partial \mathbf{y}^{(L)}}{\partial \mathbf{y}^{(L-1)}}$
- computing the gradient of the loss function w.r.t. to the final output $\frac{\partial \mathcal{L}}{\partial \mathbf{y}^{(L)}}$

The derivative of the loss w.r.t. to the weight w_i can then be constructed using the chain rule

$$\frac{\partial \mathcal{L}_\mathbf{w}}{\partial w_i} = \frac{\partial \mathbf{y}^{(l)}}{\partial w_i} \frac{\partial \mathbf{y}^{(l+1)}}{\partial \mathbf{y}^{(l)}} \frac{\partial \mathbf{y}^{(l+2)}}{\partial \mathbf{y}^{(l+1)}} \cdots \frac{\partial \mathbf{y}^{(L)}}{\partial \mathbf{y}^{(L-1)}} \frac{\partial \mathcal{L}_\mathbf{w}}{\partial \mathbf{y}^{L}} \tag{1.9}$$

Note that the gradients $\frac{\partial \mathbf{y}^{(l+1)}}{\partial \mathbf{y}^{(l)}}$ can easily computed, as the derivative of the layer's activation function is known. This back-propagation algorithm was first used in neural networks by Rumelhart in 1986 [62] and is now standard in programming packages that implement neural networks.

Feed-forward neural networks (FFNNs) have captured the attention of researchers in the field of fluid dynamics as well. See, e.g., [63], [64] or [65] for comprehensive reviews. FFNNs find utility across a diverse range of subjects, like in [66] and [67], which employ neural networks for the subgrid-scale modeling of turbulence in large eddy simulations. In [68], Fonda *et al.* employ a deep neural network for the analysis of heat transport in Rayleigh-Bénard convection. Moreover, FFNNs are used for flow control problems [69, 70]. Recently, many deep-learning models have found applications in super-resolution tasks, i.e., the reconstruction of high-resolution fields from their low-resolution counterparts [71]. In [72], Mohan *et al.* employ an Autoencoder network, a type of FFNN, to compress simulation data of three-dimensional homogeneous isotropic turbulence to a low-dimensional latent space. Similar architectures have also been used for the feature extraction of the unsteady flow past a cylinder [73]. Finally, novel physics-informed neural networks that incorporate physical laws into their loss function have been proposed [74, 75].

1.3 Recurrent neural networks: learning sequential data

The aforementioned FFNNs are suitable for static data where subsequent input data slices do not have a temporal relationship. This is suitable for classification tasks, where all the information required for the network's output is stored in the current input sample. Yet, most scientific measurements or simulations of physical processes are stored in terms of temporal sequences. When trained on data of such systems, FFNNs have shown poor performance as they do not possess an internal memory that can connect two subsequent samples. A class of neural networks that can extract information from previous inputs is the recurrent neural network (RNN), illustrated in Fig. 1.5e). While FFNNs have different parameters for each input feature, RNNs share their weights across multiple time instances, making it a profound neural network. Training an RNN is done by unrolling the computational graph over the training time steps, resulting in a deep FFNN with shared weights and applying the back-propagation algorithm. This is also referred to as back-propagation through time (BPTT) [53]. Although their architecture theoretically enables them to capture long-term dependencies, RNNs are inherently challenging to train. The BPTT results either in a vanishing or exploding gradient, as due to the chain rule, it involves the multiplication of many Jacobians of the network parameters [76–79].

One way to circumvent this problem was proposed in 2001 by Herbert Jaeger [80] in the machine learning community and 2002 by Wolfgang Maass [81] in computational neuroscience. The two independent works proposed randomly initializing the RNNs input and recurrent weights and only training the connections from recurrent to output neurons to leverage the potential of RNNs for sequential pattern analysis. While Jaeger labeled the approach *Echo State Network* (ESN), Maass referred to it as *Liquid State Machine* (LSM). Both seminal models are now summarized under the name of *Reservoir computing* (RC) models. The fact that only the weights in the readout layer are trained has several advantages. Firstly, a regression method is sufficient to find optimal weights in the mean-square sense. This avoids costly gradient computations and circumvents the original problem of RNN training. Secondly, the training time is heavily reduced, as the solution of the regression problem can be computed in a one-shot computation as opposed to the epochal and batch-wise training of stochastic gradient descent methods. To this day, RC models have shown great performance in a variety of applications, e.g. as deadbeat controller of a robot [82], as low-latency high accuracy detector of epileptic seizures [83] or for speech recognition [84]. Recently, reservoir computers have been applied to the task of forecasting nonlinear dynamical systems. There the model is trained with trajectories of the dynamical system. Finally, the neural network can autonomously generate trajectories, which do not require solving the underlying equations of motion. Recent works include chaotic systems like the Kuramoto-Sivashinsky equations [85, 86], the Lorenz 63' model [86, 87] or a low-order model of a shear flow [88]. These applications suggest that RC can perform well on chaotic dynamical systems while keeping the computational cost minimal. This makes this framework a suitable candidate for an application on turbulent convection flows, which require solving the underlying highly complex equations of motion. If a trained reservoir can produce key aspects of the turbulent flow, like the low-order heat and moisture transport statistics or the spatial organization of thermal plumes, these models could make viable candidates for future ML-based parameterization schemes.

Finally, it should be mentioned that besides the Reservoir Computing approach, other solutions to the vanishing/exploding gradient problem of RNNs have been proposed. Most prominently, gated RNNs like the *Long-Short Term Memory* (LSTM) [89] and the *Gated Recurrent unit* (GRU) [90] have seen great success in the recent years. Both network architectures use internal gating mechanisms that augment the internal memory of the RNN. While the LSTM comes with a forget, update, and output gate, the GRU uses only an update and reset gate. The controlled internal RNN state prevents the gradient from vanishing and exploding during training. However, as BPTT trains these networks, the time taken to train these networks is significantly longer than for a reservoir computer.

1.4 Scientific contributions of this doctoral thesis

This dissertation investigates the potential of machine learning algorithms for the potential improvement of subgrid-scale parameterizations in large-scale atmospheric simulations.

As discussed above, parameterization of subgrid-scale processes is a key challenge in atmospheric modeling. Conventional approaches are often based on simplifying assumptions, while super-parameterization schemes become computationally expensive. By leveraging the prowess of neural networks, this thesis aims towards a more physically consistent approach for representing the subgrid processes while retaining low computational costs. For this, a combination of an Echo State Network and state-of-the-art reduced-order modeling techniques will be employed in order to learn subgrid-scale features from direct numerical simulations of thermal convection. Specifically, this work aims to answer the following four questions.

(i) How can a Reservoir Computing model be used for spatial, temporal, and statistical feature generation of Rayleigh-Bénard convection, the paradigm of temperature-driven turbulence?

(ii) How does the RC model performance change when the complexity of the underlying convection flow is increased?

(iii) How can an RC model exhibit generalization when transitioning from one flux-driven convection flow characterized by a specific set of system parameters to another flow with distinct parameters?

(iv) How can a generative artificial neural network reproduce the transient pattern formation and associated transient statistics of a convective boundary layer?

All simulations performed for this thesis fully resolve the turbulent fluid motion down to the smallest eddies, which allows an evaluation of a combined reduced order model Reservoir Computing model in terms of reproducing crucial aspects of the turbulent flow, like its low-order statistics, as well as transport properties of heat and moisture. While this work is not concerned with implementing a machine learning-based parameterization scheme into an actual large-scale-resolving model, it aims to advance our understanding of the capabilities and limitations of neural networks for improving the accuracy and efficiency of such schemes. Potential applications lie in weather forecasting and climate modeling.

The work is structured as follows. Chapter 2 introduces the mathematical fundamentals of thermal convection flows in 2.1, the Reservoir Computing paradigm and its various implementations in 2.2, as well as the two Reduced Order Models (ROMs) for data compression in 2.3. Finally, the combined ROM-RC application to turbulent flow data and the evaluation criteria are presented in 2.4. Chapter 3 then proceeds to apply an ESN to the chaotic dynamics of a low-order model of thermal convection in 3.2. Moreover, a novel Quantum Reservoir Computing model's performance will be evaluated in 3.3. After that, in Chapter 4, a ROM-RC model will be applied to two-dimensional dry and moist Rayleigh-Bénard convection in 4.1 and 4.2 respectively. The results are discussed in terms of their implications for an ML-based super-parameterization scheme. Chapter 5 shifts the focus from classical RBC flow to a convective boundary layer during daytime conditions. First, in 5.1, an RBC-like CBL model in two spatial dimensions will be used to test the generalization capabilities of the ROM-RC model. Eventually, an alternative to the RC approach will be presented in 5.2, where a generative network will be trained with simulation data from a three-dimensional dry convective boundary layer. Moreover, it will be shown how a physics-informed data augmentation can help boost the network's performance. Furthermore, the approach is compared to the results of a conventional eddy-diffusivity mass-flux parameterization scheme. Finally, in Chapter 6, a summary of all results will be given, followed by an outlook on future research directions.

Chapter 2

Reservoir Computing: Methodology and application to turbulent thermal convection

2.1 Mathematical model of thermal convection

2.1.1 Atmospheric convection

The diagnostic equations of mass, momentum, and energy give the most general description of thermal convection. For the sake of simplicity, this section will be restricted to dry convection only. Comments on the moist convection will be made when appropriate. The mass density enters the momentum equation via the gravitational body force term ρg. This force triggers buoyant up- and downdrafts that lead to convective motion. Here, the mass density ρ is a thermodynamic variable connected to pressure p and temperature T via an equation of state $\rho(T,p)$. For atmospheric flows, it is often justified [91] to assume the ideal gas law to hold. It may be written as

$$\rho(T,p) = \frac{p}{R^*T}. \tag{2.1}$$

Here, R^* denotes the ideal gas constant divided by the molecular mass of the gas constituents. The fully compressible equations for mass, momentum, and energy are inherently complex. However, some simplifications can be introduced in the context of atmospheric flows. First, the atmospheric motion can be assumed relative to an adiabatic atmosphere in a state of hydrostatic equilibrium [92]. The assumption of adiabaticity can be understood by the low heat conductibility of air, which leads to negligible heat fluxes across the boundaries of air parcels. Further, for the parcel to be mechanically stable, the pressure inside the parcel must be the same as the environmental pressure at the parcel height. The thermodynamic system can then be described by the ground state profiles p_0, T_0, ρ_0 and their superadiabatic corrections p_1, T_1, ρ_1

$$p = p_0(z) + \epsilon p_1(\mathbf{x}, t), \tag{2.2}$$
$$T = T_0(z) + \epsilon T_1(\mathbf{x}, t), \tag{2.3}$$
$$\rho = \rho_0(z) + \epsilon \rho_1(\mathbf{x}, t), \tag{2.4}$$

with

$$\frac{\partial p_0}{\partial z} = -\rho_0 g \tag{2.5}$$

where ϵ denotes the superadiabaticity parameter, approximately given by $\epsilon \approx \mathrm{Ma}^2$ [93]. Here, Ma is the Mach number, which relates the characteristic velocity of convection, see (2.14) below, to the medium's speed of sound. Plugging this expansion into the fully compressible equations of motion and letting $\epsilon \to 0$ leads to the *anelastic approximation* [94, 95]. Note that this restricts the convective flow to subsonic velocities, as is the case in e.g. Earth's atmosphere. Furthermore, sound waves are effectively filtered out by these assumptions. This model is often applied in astrophysical processes [1, 96], as well as in deep atmospheric convection [97].

A further simplification can be made if the temperature scale height $\mathcal{H}_T$, i.e., the height over which pressure and density variations increase by one order of magnitude, is substantially larger than the height H of the convective region. This is known as the *Oberbeck-Boussinesq approximation* [98, 99]. This assumption has several implications. Most notably, the adiabatic density profile ρ_0 becomes independent of height, and the flow becomes incompressible. Furthermore, it can be shown that heating due to viscous dissipation becomes small compared to the internal energy variations. Similarly, the pressure contribution to the energy budget equation becomes negligible. Further, the pressure p_1 loses its thermodynamic meaning altogether as the equation of state for the density becomes

$$\rho_1 = -\alpha\rho_0 T_1, \tag{2.6}$$

which is now dependent on the temperature T_1 only. Here, α denotes the thermal expansion coefficient at constant pressure. Finally, the material properties of the fluid, like kinematic viscosity ν and thermal diffusivity κ, become independent of temperature and pressure. Despite these simplifications, the Boussinesq approximation can be used in the modeling of various natural systems. For example, shallow convection in the atmospheric boundary layer occurs in a narrow slab $H \approx 1-2$km above the Earth's surface. The scale height, however, reaches far into the troposphere $\mathcal{H}_T \approx 7.5-10$km [91, 92]. To ease the notation, the temperature T_1 and the pressure p_1 will, from here on, be referred to as T and p.

2.1.2 Equations of motion

The simplifications in the Oberbeck-Boussinesq-limit give rise to the following incompressible equations of motion

$$\nabla\cdot\mathbf{u} = 0 \tag{2.7}$$

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p' + \nu\nabla^2\mathbf{u} + b\hat{\mathbf{e}}_z \tag{2.8}$$

$$\frac{\partial T}{\partial t} + (\mathbf{u}\cdot\nabla)T = \kappa\nabla^2 T. \tag{2.9}$$

Here $\mathbf{u} = (u_x, u_y, u_z)^T$ is the three-dimensional velocity field, ν is the kinematic viscosity of the fluid and κ its thermal diffusivity. Note that eq. (2.9) is sometimes written in terms of the deviations θ from the purely conductive temperature profile T_{diff}

$$\theta(x,y,z,t) = T(x,y,z,t) - T_{\text{diff}}(z), \tag{2.10}$$

where T_{diff} depends on the choice of boundary conditions of T. For the Dirichlet case, see below, this profile becomes

$$T_{\text{diff}}(z) = T_{\text{bottom}} - \frac{T_{\text{bottom}} - T_{\text{top}}}{H}z, \tag{2.11}$$

where T_{bottom} and T_{top} denote the temperature values at the bottom and top boundary respectively. As a consequence of the simplified equation of state (2.6), the kinematic pressure $p' = p/\rho_0$ is fully determined by velocity and temperature field. As discussed above, the energy equation (2.9) has neither contributions from pressure nor from viscous dissipation. The buoyancy term b in (2.8) incorporates the density variations ρ_1 and is given by

$$b = -g\frac{\rho_1}{\rho_0} = \alpha g T. \tag{2.12}$$

It is common to reformulate the diagnostic equation for T in terms of b, as it allows for a flexible switch between dry convection, as considered here, and moist convection. Even though in the latter case, additional thermodynamic scalars like the liquid water content connect to the equation of state (2.12), the momentum equation (2.8) is left unchanged. A moist convection system will be introduced in Section 4.2. To keep the considerations general, the following part of this section will use the buoyancy formulation.

Many physical quantities in the governing equations (2.7 - 2.9) are not dimensionally independent, and one can identify a few non-dimensional parameters describing the system. The derivation of these nondimensional numbers follows the *Buckingham-Pi theorem* [100], which states that a system with n physical quantities, of which k are dimensionally independent, is described by $n-k$ dimensionless numbers. For this, one has to specify characteristic scales in which the physical quantities will be expressed. Here, the spatial scale is the vertical domain height H. The choice of buoyancy scale is usually considered to be

$$\Delta b = \alpha \Delta T g, \tag{2.13}$$

where $\Delta T = T_{\text{bottom}} - T_{\text{top}}$, is the difference of temperature values at the bottom and the top of the convection domain. Moreover, a common choice for the velocity scale is the free-fall velocity

$$U_f = \sqrt{\Delta b H}, \tag{2.14}$$
$$\tag{2.15}$$

where the associated free fall time scale follows as

$$t_f = H/U_f. \tag{2.16}$$

With this, the Boussinesq system can be written in a non-dimensional form

$$\nabla_* \cdot \mathbf{u}_* = 0 \tag{2.17}$$
$$\frac{\partial \mathbf{u}^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla_*)\mathbf{u}^* = -\nabla_* \hat{p}'^* + \sqrt{\frac{\mathrm{Pr}}{\mathrm{Ra}}}\nabla_*{}^2\mathbf{u}^* + b^*\hat{\mathbf{e}}_z \tag{2.18}$$
$$\frac{\partial b^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla_*)b^* = \sqrt{\frac{1}{\mathrm{RaPr}}}\nabla_*^2 b^*, \tag{2.19}$$

where the asterisk denotes non-dimensional physical variables and derivatives. The resulting non-dimensional numbers are the Prandtl number

$$\mathrm{Pr} = \frac{\nu}{\kappa}, \tag{2.20}$$

and the Rayleigh number

$$\mathrm{Ra} = \frac{\Delta b H^3}{\nu\kappa}. \tag{2.21}$$

The former compares the coefficients of momentum diffusion to the thermal diffusion coefficient. Typical values range from $\mathrm{Pr} = 0.7$ for moist air and $\mathrm{Pr} = 1$ for dry air to $\mathrm{Pr} = 7$

for water[101]. The latter represents the diffusion time scale in relation to the time scale of thermal convection. Therefore, the fluid is set in motion if Ra exceeds a critical value. In Earth's atmosphere typical values of Ra range from $10^{18} - 10^{22}$ [14]. A further non-dimensional parameter is the aspect ratio of the convection cell

$$\Gamma = \frac{L}{H}, \tag{2.22}$$

which relates the domain's characteristic horizontal length L to its height. As discussed above, ABL processes can extend to several hundred kilometers in the lateral directions and $1 - 2$km in the vertical, which gives typical values of $\Gamma \sim 10^2$. Another important dimensionless number is the Reynolds number Re of the flow, which compares the forces of inertia with the flow's viscous forces and hence characterizes the momentum transport. In the context of Rayleigh-Bénard convection, it is often defined as

$$\mathrm{Re} = \sqrt{\frac{\mathrm{Ra}}{\mathrm{Pr}}} \langle u_x^2 + u_y^2 + u_z^2 \rangle_{V,t}^{1/2}, \tag{2.23}$$

where $\langle \cdot \rangle_{V,t}$ denotes the volume-time average, and the characteristic velocity is chosen as the root-mean-square velocity [10]. Further, the Nusselt number characterizes the heat transfer through the convective domain. It is given by

$$\mathrm{Nu} = 1 + \sqrt{\mathrm{RaPr}} \langle u_z b \rangle_{V,t} \geq 1 \tag{2.24}$$

It denotes the ratio of the total buoyancy flux due to turbulent convection and buoyancy diffusion to the diffusive buoyancy flux. Hence, it is always greater or equal to one. Note that both Re and Nu are diagnostic parameters that require knowledge of the physical fields. Both quantities will be presented for each DNS run throughout this thesis.

The next subsection will introduce the boundary conditions for the present Boussinesq model. After this, that is, after 2.1.3, the asterisk-notation is dropped, and x, y, z, t, $\mathbf{u}$, p, T and b are assumed to be dimensionless unless stated otherwise.

2.1.3 Boundary conditions

The Boussinesq equations (2.17 - 2.19) are completed by the specification of boundary conditions on $\mathbf{u}$ and b. These should incorporate the physical environment of the convective flow and can be quite different for various buoyancy-driven flows in nature. Moreover, the geometry of the model has to be specified. A popular choice for buoyancy-driven flows is a rectangular domain, often used for numerical purposes, as it allows for a straightforward numerical implementation. This work will consider rectangular cells of size $L \times H$ (two-dimensional convection) or $L \times L \times H$ (three-dimensional convection).

As the mean flow is directed vertically, one should distinguish between the boundary conditions at the lateral and vertical boundaries. The mechanical boundary conditions at the vertical boundaries allow for a stress and stress-free formulation. The stress, or no-slip, condition is given by

$$u_x|_{z=0,H} = 0, \tag{2.25}$$
$$u_y|_{z=0,H} = 0, \tag{2.26}$$

and assumes that the fluid sticks to the bottom and top plate. Due to momentum diffusion, vertical stress $\frac{\partial u_x}{\partial z}$, $\frac{\partial u_y}{\partial z}$ is induced to the neighboring fluid layers. This configuration corresponds to Dirichlet boundary conditions. Further, these conditions are typical for laboratory experiments of thermal convection with finite viscosity working fluids. The stress-free or free-slip boundary condition can be assumed when friction between the boundary and fluid can be neglected. This is, e.g., the case for boundaries between two fluids. Here, the horizontal velocities are not constant at the boundary but obey

$$\left.\frac{\partial u_x}{\partial z}\right|_{z=0,H} = 0, \tag{2.27}$$

$$\left.\frac{\partial u_y}{\partial z}\right|_{z=0,H} = 0, \tag{2.28}$$

$$\tag{2.29}$$

making them Neumann boundary conditions. An example can be the top of the atmospheric boundary layer, which is capped by a non-turbulent atmosphere that allows for non-zero horizontal velocities. Further, as the vertical boundaries are considered impenetrable, the vertical velocity has to obey the no-penetration conditions

$$u_z|_{z=0,H} = 0. \tag{2.30}$$

In practice, the horizontal boundary conditions are given by no-slip and no-penetration boundaries. However, in numerical experiments, treating the spatial domain as a unit cell embedded in a large, horizontally extended environment is convenient. One example is the two-dimensional cloud-resolving model embedded inside the grid cells of a Global Climate Model in the super-parameterization scheme (see again Fig. 1.4). There, periodic boundary conditions are implemented at the horizontal boundaries of the two-dimensional model. In this way, the flow is not restricted by impermeable lateral boundaries, which would lead to undesired boundary effects. Periodic boundary conditions for the velocity fields read

$$u_x(x+L, y+L, z, t) = u_x(x, y, z, t), \tag{2.31}$$

$$u_y(x+L, y+L, z, t) = u_y(x, y, z, t), \tag{2.32}$$

$$u_z(x+L, y+L, z, t) = u_z(x, y, z, t). \tag{2.33}$$

Throughout this thesis, only periodic boundary conditions in the lateral directions will be considered.
Similarly to the velocity fields, Dirichlet and Neumann boundary conditions can also be considered for the buoyancy field. The former is given by

$$b|_{z=0} = b_{\text{bottom}}, \tag{2.34}$$

$$b|_{z=H} = b_{\text{top}} \tag{2.35}$$

with the constant boundary values $b_{\text{bottom}} > b_{\text{top}}$ that lead to a buoyancy difference $\Delta b > 0$ between both plates. This corresponds to the classic Rayleigh-Bénard setup, where the bottom plate is heated, and the top plate is cooled, s.t. a constant temperature difference is found between both boundaries. Dirichlet boundaries will be considered in 3 and 4. The most general Neumann case prescribes two heat fluxes $B_0 > 0$, B_1 at the boundaries

$$\left.\frac{\partial b}{\partial z}\right|_{z=0} = -\frac{B_0}{\kappa}, \tag{2.36}$$

$$\left.\frac{\partial b}{\partial z}\right|_{z=H} = -\frac{B_1}{\kappa}. \tag{2.37}$$

Note that if the incoming buoyancy flux at the bottom leaves at the top boundary, i.e., $B_1 = B_0$, this case becomes similar to the classical RBC setup. Even though Δb becomes non-constant, its lateral mean $\langle \Delta b \rangle_{x,y}$ becomes statistically stationary. Furthermore, this choice of boundary conditions has been shown to impact the turbulent flow and the associated long-term pattern formation [13]. However, both buoyancy fluxes will not be the same in the setting of an atmospheric boundary layer, where the bottom boundary is the heated ground. In contrast, the top boundary, even though not present, can be modeled to represent the influence of entrained warm, non-turbulent air from above the mixed layer (see 5.1). In this way, the fluid will also be heated from the top by

$$B_1 = -\beta B_0, \tag{2.38}$$

with the buoyancy-flux ratio $\beta > 0$ and hence $B_1 < 0$. This model will be discussed in more detail in 5.1. Finally, to be consistent with the mechanical boundary conditions, b will be considered periodic in the lateral directions, i.e.,

$$b(x + L, y + L, z, t) = b(x, y, z, t). \tag{2.39}$$

From here, all notation related to thermal convection is assumed to refer to the corresponding non-dimensional form with the scales presented in 2.1.2.

2.2 Reservoir Computing

2.2.1 The Reservoir Computing paradigm

Since its introduction over 20 years ago, Reservoir Computing has developed into a unified computational framework [102, 103] that summarizes many computational models. Most prominently Echo State Networks [80], Liquid State Machines [81], Backpropagation-Decorrelation models [104], novel quantum reservoir computers [105–108] as well as physical reservoir computers [103]. They all share the concept of the reservoir as a temporal kernel. This can be understood in the context of kernel-based methods (see Figure 2.1), which project their input to a high-dimensional space[1] for enhanced separability. Finally, a linear output rule is used to compute the desired outputs. In the RC case, the high-dimensional feature space is given by a "reservoir of rich dynamics" [80], from which the output dynamics can be linearly combined. In this way, the reservoir dynamics are given by the nonlinear system

$$\mathbf{r}(n + 1) = \mathbf{f}^{\mathrm{r}}(\mathbf{r}(n), \mathbf{x}(n)), \tag{2.40}$$

for the internal reservoir state $\mathbf{r}$ at the n+1-th iteration. The nonlinear function $\mathbf{f}^{\mathrm{r}}(\cdot)$ incorporates the reservoir state from the last iteration and the forcing of an external input signal $\mathbf{x}$. The system's output $\mathbf{y}$ is then computed via

$$\mathbf{y}(n) = \mathbf{f}^{\mathrm{out}}(\mathbf{r}(n)), \tag{2.41}$$

which relates the reservoir neurons to one or multiple output values. As mentioned in 1.3, training is only applied to the parameters of the function $\mathbf{f}^{\mathrm{out}}(\cdot)$. While this function does not have to be linear in $\mathbf{r}$ and can involve terms like $\mathbf{r}^2$ as in e.g.[85], the output is always constructed as a linear combination of these, possibly nonlinear, terms. This allows for a simple regression for finding the optimal parameters in $\mathbf{f}^{\mathrm{out}}(\cdot)$. This cheap training routine and low computational cost make RC a compelling computational framework. Note that the choice of $\mathbf{f}^{\mathrm{r}}(\cdot)$ and $\mathbf{f}^{\mathrm{out}}(\cdot)$ is dependent on the specific implementation of the reservoir computer. The following subsections will introduce the diverse landscape of reservoir computers, starting with the Echo State Network in Section 2.2.2, the main model used in this thesis. In Section 2.2.3, hardware implementations of a reservoir computer will be illuminated. Finally, a novel Reservoir Computing technique based on digital gate-based quantum computers will be discussed in Section 2.2.4.

[1]Contrary to these kernel methods, in the RC case, the high-dimensional representation must be computed explicitly, i.e., there is no kernel trick.

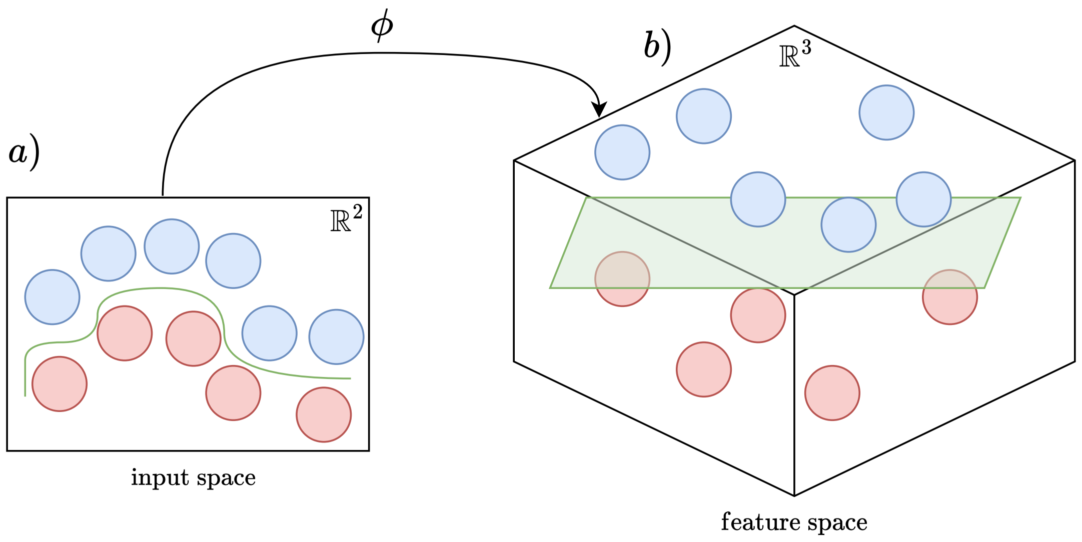

Figure 2.1: **Concept of kernel methods** for a binary classification task. The two classes, red and blue, in the two-dimensional domain a) can not be linearly separated. However, this issue can be solved by projecting the problem into a feature space of higher dimensions, here, $\mathbb{R}^3$, via a transformation ϕ, see b). Reservoir Computing uses a similar concept: an input signal is projected into a high-dimensional reservoir state space, and a linear output rule allows for the construction of the target output. This figure was inspired by the scheme in ref. [109].

2.2.2 The echo state approach

Jaeger's echo state approach is the most well-studied RC implementation to date. It has been studied extensively on a variety of problems. Nonlinear prediction tasks include the Kuramoto-Sivashinsky equation [85–87, 110], the Lorenz 63' model [86, 87, 110], the Rössler system [110]. Industrial applications cover motor control [82], fuel cells [111] and visual place recognition [112]. Recently, their application to turbulent flows has gained attention. This includes flows past a cylinder [113, 114], two-dimensional Kolmogorov flow [115], shear flow [64, 88] and Rayleigh-Bénard convection [116–120]. This subsection will introduce the ESN implementation, its training procedure, and the practical insights gained over the years.

2.2.2.1 Reservoir dynamics and training procedure

In the ESN formulation the reservoir state dynamics in eq.(2.40) take the following form[2]

$$\mathbf{r}(n+1) = (1-\gamma_r)\mathbf{r}(n) + \gamma_r \tanh\left(W^{\text{in}}\mathbf{x}(n) + W^{\text{r}}\mathbf{r}(n)\right), \tag{2.42}$$

where $\mathbf{r} \in \mathbb{R}^{N_\text{r}}$ and N_r is the reservoir dimension. The first part of the update rule is linear in the previous reservoir state $\mathbf{r}(n)$ and corresponds to a perfect memory from the last iteration. The second part introduces a nonlinearity in the shape of an entry-wise $\tanh(\cdot)$ function. The nonlinearity acts on two contributions. The first summand corresponds to a forcing of the reservoir by an external input signal $\mathbf{x} \in \mathbb{R}^{N_\text{in}}$, which often is a sample of a chaotic dynamical system that is to be learned by the ESN. The second term resembles the neuronal interactions within the reservoir. By applying a saturating activation function to the sum of

[2]Often an additional input bias $\mathbf{c}^\text{in}$ is added, i.e. $\mathbf{r}(n+1) = (1-\gamma)\mathbf{r}(n) + \gamma \tanh\left(W^\text{in}\mathbf{x}(n) + \mathbf{c}^\text{in} + W^\text{r}\mathbf{r}(n)\right)$, where $\mathbf{c}^\text{in} \in \mathbf{R}^{N_\text{r}}$ and $\mathbf{c}^\text{in} \sim \mathcal{U}[-0.5, 0.5]$ or $\mathbf{c}^\text{in} \sim \mathcal{N}(0,1)$.

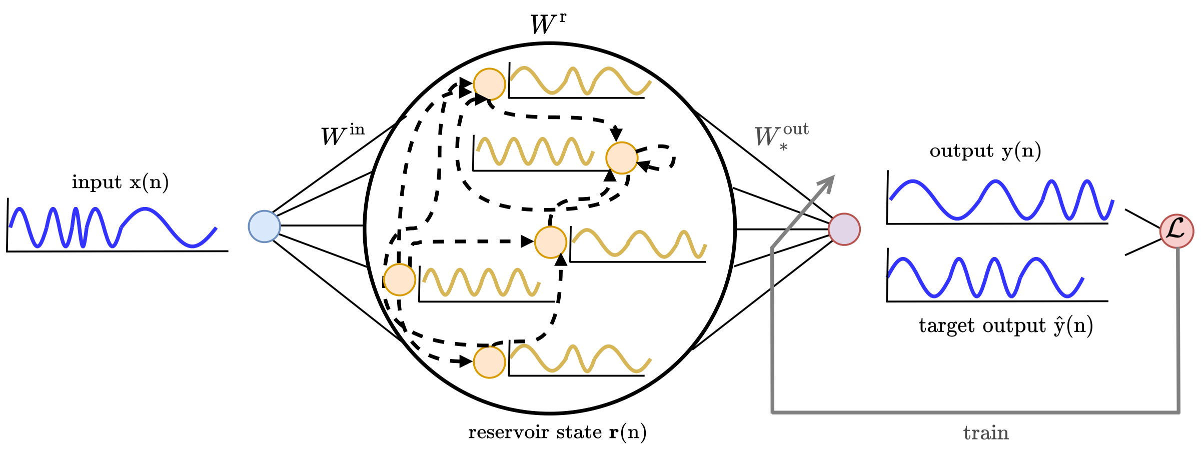

Figure 2.2: **Echo State Network** processing a one-dimensional input and output. The ESN architecture is that of a recurrent neural network. The input weights W^in provide the external input signal $\mathbf{x}$ to the reservoir (solid lines). The recurrent connections (dashed lines) inside the reservoir represent the reservoir weights W^r, which link the neurons to each other. Finally, a trained readout layer W_*^out computes the reservoir output $\mathbf{y}$. In the Reservoir Computing paradigm, only the components of W^out are learned via linear regression, while $W^\mathrm{in}, W^\mathrm{r}$ are kept fixed. The time series next to each node signifies the associated dynamics. This sketch was inspired by ref. [121].

these terms, the second part of the reservoir update becomes a nonlinear memory. It allows the reservoir to share information across nodes and take in information from outside the network. The entries of the input and recurrent weights $W^\mathrm{in} \in \mathbb{R}^{N_\mathrm{r}\times N_\mathrm{in}}$, $W^\mathrm{r} \in \mathbb{R}^{N_\mathrm{r}\times N_\mathrm{r}}$ are usually chosen from a symmetric random distribution like a uniform distribution $\mathcal{U}\,[-0.5, 0.5]$ or a standard normal distribution $\mathcal{N}\,(0,1)$ and are kept fixed after their initialization. This way, the communication channels into and inside the reservoir become random. However, a good reservoir should not depend on a certain realization of these parameters but should show good performance for a certain choice of hyperparameters, see next section, independent of the reservoir initialization. Evaluating the performance of a reservoir for different random realizations of W^in, W^r will, therefore, be a crucial step for determining a robust reservoir setting. Finally, linear and non-linear memory terms are balanced by the scalar leaking rate hyperparameter[3] $\gamma_r \in (0,1]$.

As discussed above, the reservoir dimension is chosen much larger than the input signal dimension, i.e., $N_\mathrm{r} \gg N_\mathrm{in}$, s.t. the reservoir state $\mathbf{r}$, becomes as a nonlinear high-dimensional representation of the input $\mathbf{x}$. The ESN output $\mathbf{y} \in \mathbb{R}^{N_\mathrm{out}}$ is then constructed by a linear combination of this complex representation. For this[4,5], trained output weights $W_*^\mathrm{out} \in \mathbb{R}^{N_\mathrm{out}\times N_\mathrm{r}}$ are used in eq. (2.41)

$$\mathbf{y}(n) = W_*^\mathrm{out}\mathbf{r}(n). \tag{2.43}$$

Finally, in Figure 2.2 the Echo State Network and its components are illustrated.

[3]The initial model of Jaeger [80] used a formulation corresponding to $\gamma_r = 1$ during evaluation.

[4]Some authors use the formulation $W^\mathrm{out}(\mathbf{r}, \mathbf{r}^2)^T$, e.g., ref. [85] to eliminate symmetries that hinder the reservoir in learning certain dynamical systems. Nevertheless, the training scheme stays linear.

[5]Also here an additional bias term can be added: $\mathbf{y}(n) = W_*^\mathrm{out}\mathbf{r}(n) + \mathbf{c}^\mathrm{out}$. This bias is also learned during the training procedure.

To obtain the output weights W_*^{out}, the network has to be trained. To this end, the reservoir is provided with a sequence of N_{train} input signals $\mathbf{x}$, such that the reservoir state evolves according to (2.42). The resulting sequence of reservoir states is then recorded. Further, for each sample of $\mathbf{x}$, the ESN must be presented with a corresponding target output $\hat{\mathbf{y}}$, which the reservoir should ideally reproduce. In this way, a mean square error loss function can be constructed

$$\mathcal{L}_{W^{\text{out}}} = \frac{1}{N_{\text{train}} N_{\text{out}}} \sum_{n=1}^{N_{\text{train}}} \left\| \hat{\mathbf{y}}(n) - W^{\text{out}} \mathbf{r}(n) \right\|_2^2 + \lambda_{\text{r}} \left\| W^{\text{out}} \right\|_F . \tag{2.44}$$

The optimal output weights W_*^{out} are then obtained by minimizing $\mathcal{L}_{W^{\text{out}}}$ w.r.t. the parameters in W^{out}. The final term adds an $L2$ penalty to the optimization problem in the form of the Frobenius norm $\|\cdot\|_F$ applied to W^{out}. Moreover, the regression parameter λ_{r} controls the strength of the penalty and poses an additional ESN hyperparameter. In regression problems, adding such a regularization term is common to avoid overfitting the training data. Overfitting is encountered when the fitted model perfectly reproduces the training data but generalizes poorly to data points outside the training data set. In the context of temporal inference, this term receives another interpretation concerning the stability of the reservoir output. Large values of W_*^{out} amplify small differences between the reservoir dimensions r_k. This can lead to undesirable large reservoir output values. When the reservoir operates in an autoregressive fashion (see the closed-loop scenario in 2.2.2.3), this will result in saturated activations in eq. (2.42) and therefore, in undesirable reservoir performances. Therefore, sufficient large values of λ_{r} are required to maintain good ESN dynamics.

Luckily, the minimization of expression (2.44) can be expressed in an analytical form [122]

$$W_*^{\text{out}} = \underset{W^{\text{out}}}{\operatorname{argmin}} \ \mathcal{L}_{W^{\text{out}}} = \hat{Y} R^T \left(R R^T + \lambda_{\text{r}} I_r \right)^{-1} . \tag{2.45}$$

Here, the n^{th} column of $R \in \mathbb{R}^{N_r \times N_{\text{train}}}$, $\hat{Y} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{train}}}$ are the recorded reservoir state and target output at the n^{th} training time step. Moreover, $I_{\text{r}} \in \mathbb{R}^{N_{\text{r}} \times N_{\text{r}}}$ denotes the identity matrix and $(\cdot)^{-1}$ the matrix inverse. A few things can be observed: the computational complexity of training is $\mathcal{O}\left(N_{\text{train}}^2 N_{\text{r}} + N_{\text{r}}^3\right)$. Additionally, to avoid the issue of underdetermination in the regression problem, it is necessary to have a sufficiently large number of training samples, s.t. $N_{\text{train}} \gtrsim N_{\text{r}}$. Moreover, the output weights are acquired through a single-step process, eliminating the need for multiple batches or epochs, following which the network adjusts its weights accordingly. This makes the training of ESNs extremely fast, as no backward pass through the network has to be performed. However, eq.(2.45) holds only for the penalized mean square error loss function in eq. (2.44). This contrasts with optimization methods such as gradient descent, where individual loss metrics can be applied. Section 2.4.2 will discuss problems arising from this restriction when inferring turbulent flow using ESNs.

2.2.2.2 Reservoir hyperparameters and ESN tricks of the trades

Over the years of ESN research, several reservoir preparation guidelines have been established. This section will briefly overview the most common ESN hyperparameters, i.e., parameters not optimized by the aforementioned training procedure. A correct choice of these variables can lead to a drastic improvement in reservoir performance. As part of the author's doctoral studies, he developed a *Python* package *turbESN* [123] in which all of the below-mentioned details are implemented.

Before a reservoir can be initialized, its dimensions have to be specified. While the input and output dimensions N_{in}, N_{out} are pre-set by the external database of $\mathbf{x}$ and $\hat{\mathbf{y}}$, the reservoir size N_{r} is a free parameter. In the spirit of kernel methods, it is common to choose N_{r} to exceed N_{in} by at least one order of magnitude [124]. Moreover, it should also chosen so as not to exceed the number of training samples N_{train} to avoid underfitting. Furthermore, In the original formulation of the Echo State Networks, the adjacency matrix W^{r}, which describes the intra-reservoir connections, is commonly kept sparse, i.e., most matrix elements are set to zero. However, Jaeger's initial suspicions about improving the ESN performance via a reduced number of inter-reservoir communication channels have not been confirmed. On the contrary, for most tasks, the reservoir performance is virtually independent of the reservoir sparsity. Presently, a sparse weight matrix W^{r} is used to reduce the computational cost of eq. (2.42). Moreover, in practice, the complementary reservoir density D_{r}, the fraction of non-zero weights, is specified. It takes on values between 0.2 and 0.5 [116, 117, 119]. In ref. [80], Jaeger points out that for a reservoir to perform reliably, its internal state $\mathbf{r}$ should converge asymptotically, w.r.t. time, under the influence of the input $\mathbf{x}$. This means that two initially different reservoir states $\mathbf{r}^{(1)}(0)$, $\mathbf{r}^{(2)}(0)$ with the same reservoir matrices should converge under the influence of the same input signal to the same state $\mathbf{r}(n)$ for a sufficiently large number of iterations n. It embodies the fading memory characteristic that an Echo State Network (ESN) should possess, enabling it to disregard information from distant past time steps. The ESN is then uniquely defined by the input signal. This property is called the *echo state property* (ESP). Fulfilling the ESP is, therefore, desired for good reservoir performances. It is connected to the algebraic properties of the reservoir, as well as the properties of the driving input. Following Jaeger's seminal paper, many works restrict the largest absolute eigenvalue of W^{r}, i.e., its spectral radius ϱ_{r}, to values strictly smaller than one. Jaeger argued $\varrho_{\text{r}} < 1$ to be a sufficient condition for the ESP. However, Lukoševičius and Jaeger [125] discuss that this condition is neither necessary nor sufficient for achieving the ESP, a conclusion to which other works have come as well [126, 127]. Presently, new attempts towards reformulating the ESP in terms of *Generalized Synchronization* [128] have been made. Nevertheless, the spectral radius ϱ_r and the leaking rate γ_r determine the reservoir update speed in eq. (2.42), such that both variables are commonly considered hyperparameters in ESN runs. Moreover, it is convenient to introduce an input scaling parameter σ_{r} that is pre-multiplied to W^{in} to control the non-linearity of the reservoir activation in (2.42), as well as the strength of the input forcing.

These hyperparameters, together with the regression parameter in (2.45), define an ESN setting $\Xi_{\text{r}} = \{N_{\text{r}}, \varrho_{\text{r}}, \gamma_{\text{r}}, \lambda_{\text{r}}, D_{\text{r}}, \sigma_{\text{r}}\}$. They must be chosen before training starts[6] and highly depend on the input data $\mathbf{x}$. A practical guide for delimiting the range of values of these variables can be found in [124], while a systematic review of these hyperparameters for several benchmark problems is listed in [129]. Finally, hints on how to set up hyperparameter grid search procedures can be found in [130].

2.2.2.3 ESN operation modes

After training, an ESN can be used in several ways. The two most prominent operation modes are the continuously available data (open-loop) and the autonomous prediction (closed-loop) scenario. Figure 2.3 illustrates both scenarios. In the former, the reservoir receives a prior known input signal $\mathbf{x}$, which is continuously supplied from outside the reservoir. After its state $\mathbf{r}$ has been updated and the corresponding reservoir output $\mathbf{y}$ computed, the reservoir receives the next external signal. In terms of prediction, this can be compared to a

[6]This is except for the regression parameter, which can be adapted after the matrix R has been constructed.

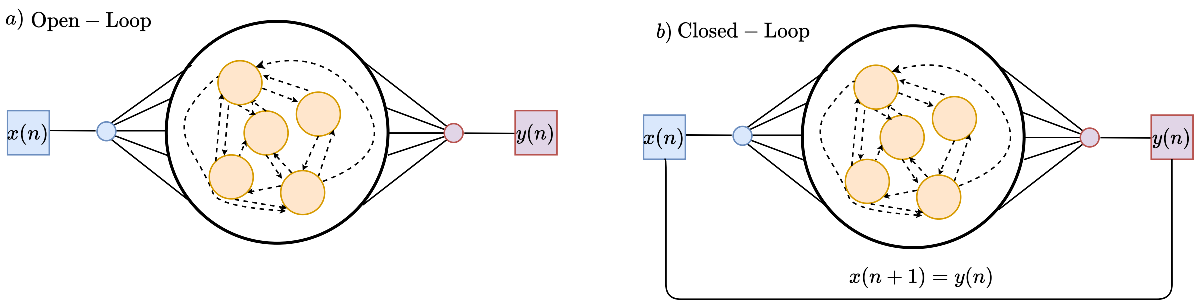

Figure 2.3: **Operation modes** of the ESN. (a) Open loop mode, where the input data $\mathbf{x}$ is continuously provided at each iteration step. This mode allows for reconstruction tasks where the dimensionality of the input does not need to match the output dimension. (b) Closed-loop mode, where the reservoir runs autoregressively. While this scenario requires $N_{\text{in}} \equiv N_{\text{out}}$, it allows for predictions past the horizon of the current database.

one-day weather forecast: given the known weather conditions today, what will the weather be like tomorrow? This scenario has the advantage that the reservoir state is continually updated with a guiding teacher signal that does not allow the state to diverge too far from its previous instance. Incidentally, the reservoir is trained precisely for this mode in (2.45). Note that this scenario allows for unequal input and output dimensions, i.e., $N_{\text{in}} \neq N_{\text{out}}$, and can therefore be used for inferring missing information, as in [110], where one of the Lorenz 63 dimensions is used as input, and the ESN infers the missing two values. A similar approach will be taken in 3.3. Moreover, similar attempts have been made to apply an open-loop ESN to spatial reconstruction tasks [114, 119]. Nevertheless, on occasions, it becomes intriguing to understand the events that transpire significantly further into the future beyond the confines of the existing database records. In this case, no continuous external information can be provided to the ESN. However, in terms of temporal prediction tasks, one can rely on the recent reservoir output and use it as a basis for the next reservoir update. In this closed-loop scenario, the reservoir's outputs are fed back to the input layer, i.e., $\mathbf{x}(n+1) = \mathbf{y}(n)$. This way, the reservoir runs autoregressively using its past predictions, starting from an initial user-provided starting point. Note that, unlike the open-loop case, this requires the input and output dimensions to be the same, i.e., $N_{\text{in}} \equiv N_{\text{out}}$. Furthermore, this can be interpreted as a long-term weather or climate forecast. The latter is attractive for potential ML-based super-parameterization applications, as a trained ESN could run in closed-loop mode, taking over the role of the subgrid-scale resolving fluid simulation. In this way, several snapshots could be generated with low computational costs. In this spirit, this thesis will primarily focus on the closed-loop operation mode for two-dimensional convection flows. However, the open-loop scenario will be employed in sections 3.3.1 and 3.3.2.

2.2.3 Physical Reservoir Computing

Echo State Networks operate as computer programs, i.e., *in silico*. However, recently, it was noted that "the basic concept of RC is exploiting the intrinsic dynamics of the reservoir by outsourcing learning [...] to the readout part. According to this unique setting, reservoirs do not have to be an RNN anymore but can be any dynamical system." [131]. From this idea, novel physical reservoir computers that exploit physical dynamics for computation tasks have emerged. This latest RC trend is worth mentioning as this emerging field builds on the knowledge and success of current ESN applications. Moreover, opposed to software implementations, like the ESN, these hardware realizations are particularly interesting, as they might yield high computational speed while maintaining low power consumption [103]. Recent examples, include photonic [132–135], electromechanical [136], opto-electronic [137, 138] or spintronic [139–141] reservoirs. Many of these implementations use delay-based reservoirs, a concept introduced by Appeltant *et al.* in ref. [137]. They are easy to implement in a hardware application, as they only require a single nonlinear node and a delayed feedback loop [142]. Furthermore, for good reservoir performances, the underlying physical reservoir should possess steady-state dynamics and be close to a bifurcation [143]. In contrast to the original formulation of RC, where many neurons interact with each other, the delay-based reservoir consists only of a single node that interacts with itself over different instances of time via a delay loop. Therefore, the reservoir state **r** consists of a collection of time samples of the response of the same node gathered via a time-multiplexing procedure. As the reservoir dimensions are scattered over time and not a collection of physically implemented nodes, they are also referred to as virtual nodes. In this case, the reservoir is given by a dynamical system with time-delayed feedback s.t. eq.(2.40) becomes

$$\mathbf{r}(n+1) = \mathbf{f}^{\mathrm{r}}(\mathbf{r}(n), \mathbf{r}(n-\tau), \mathbf{x}(n)), \tag{2.46}$$

where τ is the delay time of the dynamical system.

Despite their hardware-friendly design, physical reservoirs face similar problems as the classical ESN, as tuning of several hyperparameters is required. Moreover, despite the high processing speed of these reservoirs, the overall process is bounded by the external training procedure, which is still mostly done *in silico*. While physical Reservoir Computing will not be applied in this dissertation, it may be an interesting future research direction, that can make use of the results, which will be presented in the next chapters.

2.2.4 Quantum Reservoir Computing

A further descendant of the original RC formulation has found its way to the field of quantum machine learning [144], a symbiosis between quantum computing (QC) and classical machine learning. Besides the ESN implementation of a reservoir computer, this thesis will consider a state-of-the-art Quantum Reservoir Computing (QRC) model in 3.3. Therefore, the next paragraphs will be devoted to the architecture and concept of QRC. However, a pedagogical introduction to quantum computing will be given first, after which the details on a digital gate-based QRC implementation will be given.

2.2.4.1 A primer on quantum computing

Quantum computing, or quantum information science, is an interdisciplinary field of quantum physics and information science that uses the principles of quantum mechanics to process information [145]. The notion of a quantum computer dates back to Richard Feynman [146], who speculated that simulating complex quantum systems with a computer that exploits quantum mechanical phenomena is more straightforward than using a conventional computer. Since then, quantum computing hard- and software development have rapidly improved. Nowadays, researchers can run their code, often written in scripting languages like *Python*, on a quantum device via commercial cloud services [147]. The quantum hardware is typically implemented as ion traps [148, 149] or superconducting circuits [150], which reach up to a few hundred of qubits, the basic unit of computation (see below). However, these intermediate-scale devices are still prone to computational errors, which occur due to insufficient control of qubits and their decoherence. For this reason, the current era of QC is called the noisy intermediate scale quantum (NISQ) era [151]. While still in its early stages and vulnerable to decoherence, the progress made by current quantum technology plays a vital role in shaping the future of quantum applications.

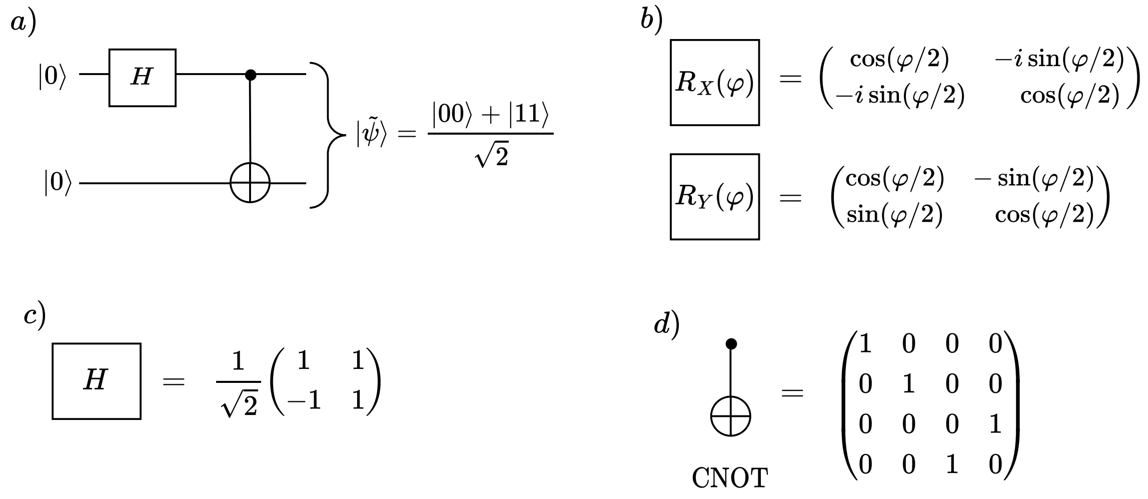
a)
$|0\rangle$
H
$|0\rangle$
$|\tilde{\psi}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$
b)
$R_X(\varphi) = \begin{pmatrix} \cos(\varphi/2) & -i\sin(\varphi/2) \\ -i\sin(\varphi/2) & \cos(\varphi/2) \end{pmatrix}$
$R_Y(\varphi) = \begin{pmatrix} \cos(\varphi/2) & -\sin(\varphi/2) \\ \sin(\varphi/2) & \cos(\varphi/2) \end{pmatrix}$
c)
$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$
d)
$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CNOT

Figure 2.4: Basic **quantum gate circuitry**. (a) A quantum circuit processing $N_{\text{qubit}} = 2$ qubits in the initial state $|0\rangle$. The two qubits become entangled after applying a Hadamard and controlled not gate. The final state $|\tilde{\psi}\rangle$ is an entangled Bell-state. (b) Definition of rotational gates R_X, R_Y with rotational angel φ. (c) Definition of the Hamadard gate used in a). (d) Definition of the two-qubit CNOT gate, used in (a).

The basic unit of information on a quantum computer is known as a qubit. It resembles a two-level, and therefore the smallest non-trivial, quantum system and is described by the quantum state

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.47}$$

As given by the axioms of quantum mechanics, the state vectors $|0\rangle$ and $|1\rangle$ are basis functions of a two-dimensional Hilbert space $\mathcal{H}$. The complex amplitudes $\alpha, \beta \in \mathbb{C}$ denote probability amplitudes and must obey

$$|\alpha|^2 + |\beta|^2 = 1, \tag{2.48}$$

i.e., their absolute squares denote the probability of finding the qubit in the state $|0\rangle$, $|1\rangle$ respectively. One advantage of quantum computing can be understood when considering the combined state of two qubits $|\psi_1\rangle = \alpha_1\,|0\rangle + \beta_1\,|1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle = \alpha_2\,|0\rangle + \beta_2\,|1\rangle \in \mathcal{H}_2$, i.e.,

$$|\psi_1\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \eta_{00}\,|00\rangle + \eta_{01}\,|01\rangle + \eta_{10}\,|10\rangle + \eta_{11}\,|11\rangle \quad \in \mathcal{H}_1 \otimes \mathcal{H}_2, \tag{2.49}$$

where $\otimes$ denotes the tensor product and $|00\rangle = |0\rangle \otimes |0\rangle$, $|01\rangle = |0\rangle \otimes |1\rangle$, etc. Note that the amplitudes of the composite state are given by $\eta_{00} = \alpha_1\alpha_2$, $\eta_{01} = \alpha_1\beta_2$, etc. In this way, the state space of a quantum system grows exponentially with the number of qubits N_{qubit}. This exponential state space allows for complex feature representations and can be leveraged when unitary quantum operators augment the combined quantum state. Figure 2.4a) shows a basic quantum circuit of a $N_{\text{qubit}} = 2$ qubit system, which is augmented from left to right by a Hadamard gate and controlled not (CNOT) gates. Both gates are defined in Fig. 2.4c), d). While the initial quantum state is given by the product state in eq. (2.49), the final state can not be written as the product of two qubits. This phenomenon is known as quantum entanglement. When two states are entangled, they become "highly correlated", i.e., the operations on the one qubit also affect the other. This non-local interaction of qubits paired with their exponentially growing state space has made quantum devices highly interesting for machine learning tasks, where complex representations are leveraged.

The application of QC can roughly be classified into three fields [152]: (1) simulation of quantum systems for physical or chemical processes. (2) Solving classical optimization problems. Interest in this field has risen since theoretical considerations proposed a quantum speed-up of classical optimization problems. Most notably, these considerations are based on the works of Shor and Grover about the factorization of prime numbers [153] and combinatorial optimization involving an exhaustive search [154]. (3) Data processing on quantum computers, which includes the topic of Quantum Reservoir Computing. The success of machine learning and deep neural networks, which extract features effectively by finding complex data representations, has sparked the notion of quantum machine learning. Biamonte *et al.* describe the motivation behind QML by the following idea: "If small quantum information processors can produce statistical patterns that are computationally difficult for a classical computer to produce, then perhaps they can also recognize patterns that are equally difficult to recognize classically" [144]. For this, novel quantum algorithms are required. These algorithms subsequently use a quantum state's large state space and entanglement property to encode a classical input $\mathbf{x}$ into a complex high-dimensional representation. This sets the stage for the present Quantum Reservoir Computing algorithm.

2.2.4.2 A digital gate-based quantum reservoir computer

The research on QRC proceeds along two major model frameworks: (i) Analog frameworks that simulate the Hamiltonian dynamics of an interacting boson or fermion many-particle quantum system. Examples are spin ensembles [105, 106] or arrays of Rydberg atoms [155]. (ii) Digital gate-based frameworks, where the quantum reservoir comprises a circuit of universal quantum gates implemented on a NISQ device. Such an implementation will be introduced below. Applications of quantum reservoir computers are still very limited but have shown good performance for the one-step prediction of nonlinear dynamics of the Mackey-Glass time-delay differential equation [105] as well as of the nonlinear autoregressive moving-average time series [156, 157].

Philipp Pfeffer developed the following QRC algorithms, labeled H1 and H2, at the Institute of Thermo- and Fluid Dynamics at TU Ilmenau. They were published in two joint works [108, 158] and make use of a digital gate-based approach, sketched in Figure 2.5. The following paragraphs will introduce the two quantum reservoir architectures employed in Section 3.3. In both methodologies, the traditional input $\mathbf{x}$ is incorporated into the quantum circuit through an amplitude encoding process. For this purpose, the input is appropriately adjusted, for example, within the range of $\mathbf{x} \in [-\pi, \pi]$. Ultimately, the individual elements of $\mathbf{x}$ are translated into rotational angles for the application of R_Y gates (see again Fig. 2.4b)). One iteration of the circuit then comprises the unitary evolution of the initial quantum state $|\psi\rangle$ and a measurement of the final state $|\tilde{\psi}\rangle$ w.r.t. to the Pauli Z-basis. This process has to be repeated several times in order to obtain a probability distribution $\mathbf{p} = (p_1, p_2, ..., p_{2^{N_\mathrm{qubit}}})^T$ of all 2^{N_qubit} basis states. The dynamics of both algorithms are motivated by the classical ESN dynamics in (2.42). H1 makes use of the following hybrid classical-quantum approach

$$\mathbf{r}_{H1}(n+1) = (1-\gamma_\mathrm{r})\mathbf{r}_{H1}(n) + \gamma_\mathrm{r}\mathbf{p}(n+1) \tag{2.50}$$

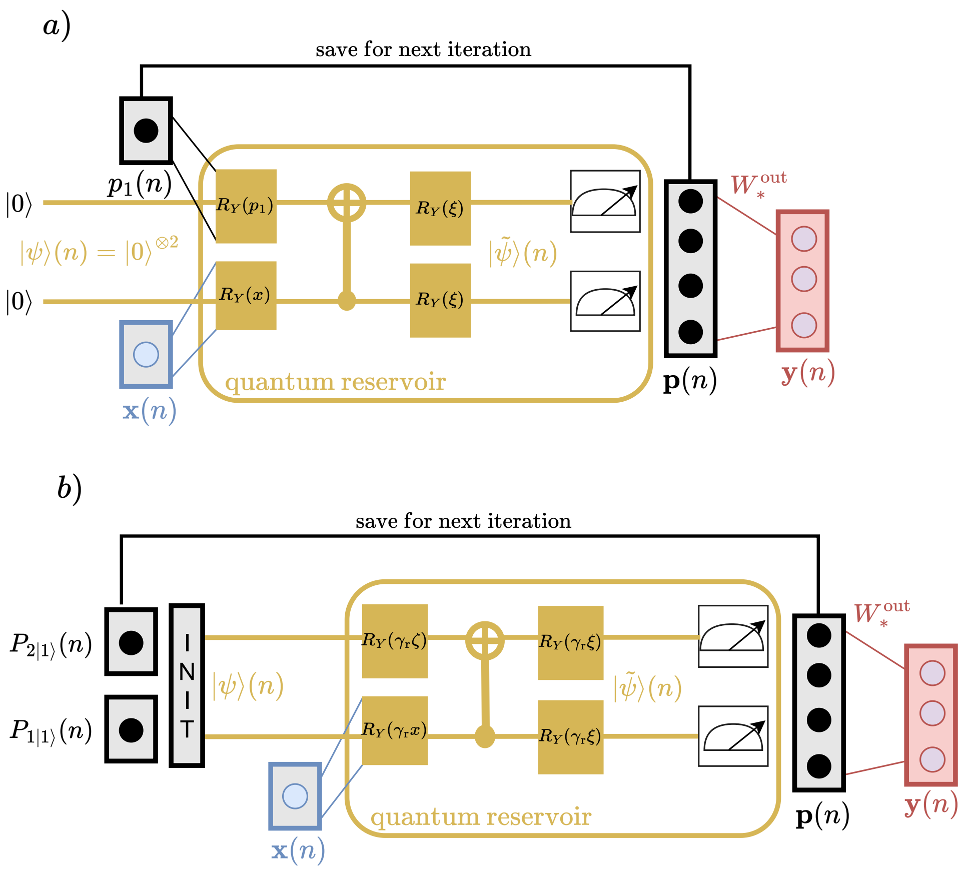

Figure 2.5: Digital gate-based **quantum reservoir computer**. The quantum reservoir is implemented as a sequence of unitary quantum gates. a) Algorithm H1: the initial quantum state at every iteration is given by $|0\rangle^{\otimes N_{\text{qubit}}}$. Further, the input $\mathbf{x}$ is encoded as an angle of rotation into the rotational gate R_Y. Additional rotational gates take in random variables ξ and a subset of previously measured probabilities p_i. The final quantum state of the system $|\tilde{\psi}\rangle$ is then measured several times to obtain a distribution $\mathbf{p}$ of all basis states. b) Algorithm H2: Contrary to H1, the quantum state $|\psi\rangle$ is initialized with the previously measured probabilities $P_{i,|1\rangle}$ to find the i^{th} qubit in $|1\rangle$. Moreover, the leaking rate γ_{r} is also passed to the quantum gates. Again, the input and random parameters ξ are introduced to the reservoir. Note that the quantum circuits are far more complex than these simplified schemes. More details follow in 3.3.

with

$$p_j(n+1) = |\langle e_j| \, U\left(\xi, \mathbf{x}(n), \mathbf{r}_{H1}(n)\right) |e_1\rangle|^2, \tag{2.51}$$

where e_j is the j^{th} basis vector of the N_{qubit} Hilbert space $\mathbb{C}^{\otimes N_{\text{qubit}}}$. Moreover, $e_1 = |0\rangle^{\otimes N_{\text{qubit}}}$ is the basis vector for which all qubits are in the state $|0\rangle$. The quantum circuit U is parameterized by the random variables $\xi \sim \mathcal{U}\,[0, 4\pi]$, the current input $\mathbf{x}$ and the previous reservoir state $\mathbf{r}_{H1}(n) \in [0,1]^7$. $\gamma_{\text{r}} \in (0,1]$ is again the leaking rate parameter. For execution on a real quantum device, it is preferred to use only a subset of $\mathbf{r}_{H1}$, instead of the full $2^{N_{\text{qubit}}}$ values, as it speeds up the computation of the whole quantum circuit. Nevertheless, the subset should be large enough to act as nonlinear memory in the reservoir dynamics. The quantum circuit consists of rotational gates that incorporate the mentioned parameters, along with interspersed CNOT layers. Note the similarlities between expressions (2.42) and (2.50). In this way, H1 employs a linear memory that requires a classical device for storage of $\mathbf{r}_{H1}(n)$. Further, the nonlinear quantum memory is realized by the embedding of r_{H1} into R_Y.

The H2 algorithm sets itself apart from this approach by conducting the entire execution directly on the quantum device. The reservoir is not parameterized by the reservoir state but is projected on an approximate reservoir vector

$$|\mathbf{r}_{\hat{H2}}(n)\rangle = \bigotimes_{i=1}^{N_{\text{qubit}}} \left[\sqrt{1 - P_{i,|1\rangle}(n)}\,|0\rangle + \sqrt{P_{i,|1\rangle}(n)}\,|1\rangle\right], \tag{2.52}$$

that incorporates the probability $P_{i,|1\rangle}$ of measuring the i^{th} qubit in the basis state $|1\rangle$. Then, the dynamics become

$$\mathbf{r}_{H2}(n+1) = \mathbf{p}(n+1), \tag{2.53}$$

with

$$p_j(n+1) = |\langle e_j| \, U\left(\gamma_{\text{r}}\xi, \gamma_{\text{r}}\mathbf{x}(\mathbf{n})\right) |\hat{r}_{H2}\rangle|^2 \quad \text{and} \quad U(0,0) = I_{2^{N_{\text{qubit}}}}. \tag{2.54}$$

Here $I_{2^{N_{\text{qubit}}}} \in \mathbb{R}^{2^{N_{\text{qubit}}} \times 2^{N_{\text{qubit}}}}$ denotes the identity matrix. By integrating the leakage rate γ_{r} into the unitary operator U, the external memory, as seen in H1, is effectively transferred within the reservoir. Moreover, the perfect memory for $\gamma_{\text{r}} = 0$ is retained.

Finally, as in the classical case, a trained output matrix yields the reservoir outputs

$$\mathbf{y}(n) = W_*^{\text{out}} \mathbf{r}_{H1/H2}(n). \tag{2.55}$$

The optimal weights W_*^{out} are again computed via the minimization of (2.44). Note that the training procedure (2.45) is done on a classical device. Hence both H1 and H2 are hybrid classical-quantum RC approaches. Finally, a comparison between ESN and the just introduced QRC formulation is in order. Table 2.1 lists the ESN hyperparameters and their quantum analogs. While the leaking rate and regression parameter are reminiscences of the classical reservoir, the reservoir dimension of the quantum reservoir becomes exponentially dependent on the number of qubits, i.e., $N_r = 2^{N_{\text{qubit}}}$. Moreover, it should be noted that the spectral radius, as introduced for the ESN, drops out of the quantum algorithm, as the largest absolute eigenvalue of the unitary U is unity. Finally, as mentioned above, the classical squashing activation function is substituted by the sine and cosine entries of the rotational gate R_Y.

[7]In practice the state is multiplied by π, s.t., $\mathbf{r}_{H1} \in [0, \pi]$.

Quantity	Classical RC (ESN)	Quantum RC
Reservoir dimension	N_r	$N_\mathrm{r} = 2^{N_\mathrm{qubit}}$
Leaking rate	γ_r	γ_r
Regression parameter	λ_r	λ_r
Spectral radius of reservoir	ϱ_r	1
Reservoir state at time n	$\mathbf{r}(n) \in \mathbb{R}^{N_\mathrm{r}}$	$\mathbf{r}_{H1/H2}(n) \in \mathbb{C}^{N_\mathrm{r}}$
Reservoir model nonlinearity	$\tanh(\cdot)$	$R_Y(\cdot)$

Table 2.1: Comparison of classical ESN and the gate-based Quantum Reservoir Computing models. The number of qubits is N_qubit. Since unitary transformations are norm-preserving, the quantum case's spectral radius ϱ_r always equals 1. Note that the ESN possesses two additional hyperparameters: reservoir density D_r and input scaling σ_r, which have no direct QRC counterpart.

2.3 Reduced order modeling of thermal convection flows

Traditional kernel methods have the advantage that the transformation to the high-dimensional feature space can be avoided through the "kernel trick" [159]. However, ESNs do not offer such an easy way out. Instead, a reservoir, i.e., its weight matrices W^in, W^r, W^out have to be explicitly computed and stored in computer memory. Furthermore, as mentioned above, to obtain a "reservoir of rich dynamics" [80], the reservoir size should be chosen s.t. $N_r \gg N_\mathrm{in}$. Common choices pick reservoir dimensions at least an order of magnitude larger than the input dimension [124]. However, this becomes a challenge when the input signal possesses many degrees of freedom. Take the two-dimensional RBC flow considered in Section 4.1. It has a grid size of $N_x \times N_z = 256 \times 64$. That is approximately $1.6 \cdot 10^4$ float values per physical field and snapshot. This would require a reservoir of at least $N_r \sim 10^5$ neurons to satisfy the aforementioned condition. This substantially decelerates the dynamics and training process of the ESN, resulting in memory-intensive and time-consuming training sessions and exhaustive searches for hyperparameters. Common literature values of N_r range from tens to thousands of neurons [129].

The main part of this thesis will be concerned with the use of ESNs to infer information about turbulent thermal convection. Hence, numerical data on turbulent convection must be provided to the network. However, turbulent flows cover a wide range of spatial scales. These scales are encoded in the instantaneous flow fields and manifest, e.g., as eddies or thermal plumes. However, it has been observed that spatial features tend to organize themselves into coherent structures that undergo characteristic temporal variations [160, 161]. Moreover, a sufficiently large database of flow snapshots can be used to extract said spatial features and save them into a numerical library. Furthermore, the individual elements of this library, referred to as modes, can be sorted according to their contributions, e.g., to the turbulent kinetic energy or thermal variance of the physical fields. Complexity and data can be heavily reduced by merely considering the most important modes, i.e., the largest spatial scales. Such modeling techniques that confine many degrees of freedom to a low-dimensional manifold are termed Reduced Order Models (ROMs). Such models make a suitable preprocessing step for the present Reservoir Computing approach to turbulent convection flows. By limiting the reservoir input to the most crucial flow features, one can leverage the temporal kernel of the reservoir while keeping computational and memory costs at a low level. At first glance, ignoring the small-scale sounds counter-intuitive to the motivation in Chapter 1, which introduced the idea of using an ML model to provide the sub-grid scale features

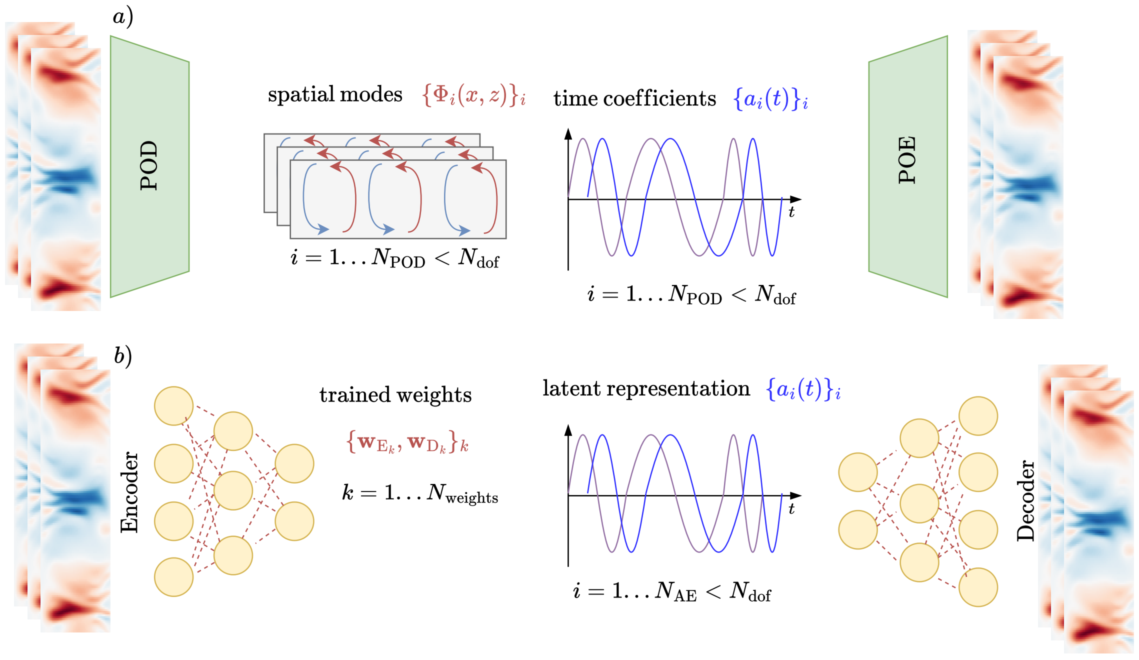
a)
POD
spatial modes $\{\Phi_i(x,z)\}_i$
$i = 1 \ldots N_{\mathrm{POD}} < N_{\mathrm{dof}}$
time coefficients $\{a_i(t)\}_i$
t
$i = 1 \ldots N_{\mathrm{POD}} < N_{\mathrm{dof}}$
POE
b)
Encoder
trained weights
$\{\mathbf{w}_{\mathrm{E}_k}, \mathbf{w}_{\mathrm{D}_k}\}_k$
$k = 1 \ldots N_{\mathrm{weights}}$
latent representation $\{a_i(t)\}_i$
t
$i = 1 \ldots N_{\mathrm{AE}} < N_{\mathrm{dof}}$
Decoder

Figure 2.6: **Reduced Order Model** schemes for a two-dimensional fluid flow. (a) A Proper Orthogonal Decomposition compresses the input snapshot of a physical field to a set of spatial modes $\Phi_i(x,z)$ and their temporal coefficients $a_i(t)$. (b) An Autoencoder neural network decomposes the data into the latent representation $a_i(t)$. Moreover, the Encoder and Decoder network's trained weights w_E, w_D are trained. While the POD decomposes the physical field into a linear combination of modes and coefficients, the AE can be seen as a nonlinear extension of the linear POD approach.

to a large-scale model. However, it should be noted that this thesis will only consider direct numerical simulations that resolve all turbulent scales down to the smallest eddy. Further, conventional cloud-resolving models used in SP do not fully resolve all scales of turbulence themselves, as they can be compared to large eddy simulations. Therefore, concentrating only on the most prominent spatial features of the below-presented convection flows is a legitimate pre-processing strategy.

This section introduces two standard ROMs that will be employed in chapters 3, 4 and 5: (i) the *Proper Orthogonal Decomposition* (POD) is a well-known method in fluid mechanics that decomposes snapshots of physical fields into data-driven optimal spatial modes and their temporal coefficients. (ii) The convolutional *Autoencoder* (AE) is a feed-forward neural network with a bottleneck structure. The process involves taking a DNS snapshot as input, compressing it into a lower-dimensional representation, and then expanding it back to its original size. Through training the network to reconstruct the original snapshot, it is forced to capture the fundamental spatial characteristics of the flow in the low-dimensional representation. In the following two subsections, both methods will be presented. Further, their similarities and differences will be discussed.

2.3.1 Proper Orthogonal Decomposition

The idea of decomposing a dataset of flow measurements into a collection of modes dates back to Lumely in 1967 [162]. He brought forth the notion of Principal Component Analysis (PCA) [163, 164] to the realm of fluid dynamics, establishing it as a conventional technique for reducing dimensionality. In this field, it has since come to be recognized as Proper Orthogonal Decomposition. Similar to the PCA in statistics, it has become a standard tool for research in fluid dynamics [165]. As mentioned above, the POD decomposes a series of snapshots into a set of orthogonal spatial modes and their corresponding temporal coefficients. While the spatial modes can be interpreted as dominant spatial patterns of the underlying flow, the temporal coefficients describe their temporal evolution. Applications range from data reduction in Rayleigh-Bénard [166, 167] and moist convection [168] to wall-bounded shear flows [169] and flows past a cylinder [113, 169]. Moreover, it is commonly used for flow control [170, 171].

Figure 2.6a) illustrates the concept of the POD for a two-dimensional flow with horizontal and vertical velocity fields $u_x(x,z,t)$ and $u_z(x,z,t)$: a collection of snapshots is decomposed into a set of spatially dependent modes $\Phi_i = (\Phi_i^{u_x}(x,z),\, \Phi_i^{u_z}(x,z))^T$ and their temporal coefficients $a_i(t)$. This decomposition can be written as a Galerkin projection formulation

$$u_x(x,z,t) \approx \sum_{i=1}^{\infty} a_i(t)\Phi_i^{u_x}(x,z), \tag{2.56}$$

$$u_z(x,z,t) \approx \sum_{i=1}^{\infty} a_i(t)\Phi_i^{u_z}(x,z), \tag{2.57}$$

i.e., the fluid flow is decomposed into a linear combination of dominant spatial patterns of each field and their shared temporal modulation. The spatial modes are orthonormal functions, i.e.,

$$(\Phi_i \Phi_j)_{L^2} = \delta_{ij}, \tag{2.58}$$

where δ_{ij} denotes the Kronecker delta. The time coefficients are obtained by projecting the spatial modes on the function $\mathbf{u} = (u_x, u_z)^T$:

$$a_k = (\Phi_k, \mathbf{u})_{L^2}\,, \tag{2.59}$$

where $(\cdot,\cdot)_{L^2}$ is the inner product of the square integrable function space $L^2(\Omega,\mathbb{C}^n)$. For two functions $\mathbf{f}(x,z),\mathbf{g}(x,z)\in L^2(\Omega,C^n)$ it is defined as

$$(\mathbf{f},\mathbf{g})_{L^2}=\sum_{i=1}^{n}\int_{\Omega} f_i(x,z)g_i^*(x,z)dxdz, \tag{2.60}$$

where f_i, g_i $(i=1,2,..,n)$ are the components of $\mathbf{f},\mathbf{g}$ and Ω denotes the two-dimensional spatial domain. Further, the asterisk denotes the complex conjugate.

The POD aims to find a set of modes that captures the maximal kinetic energy of the original flow. Specifically, it minimizes the loss functional

$$\mathcal{L}(\Phi_i)=\left\langle\left\|\mathbf{u}-\sum_{i=1}^{\infty}(\mathbf{u},\Phi_i)_{L^2}\Phi_i\right\|_{L^2}^2\right\rangle_t \qquad \text{subject to } \|\Phi_i\|_{L^2}^2=1, \tag{2.61}$$

where $\langle\cdot\rangle_t$ denotes the time average and $||\cdot||_{L^2}$ is the norm induced by $(\cdot,\cdot)_{L^2}$. It can be shown that this leads to an eigenvalue problem of the time average of the two-point correlation tensor of $\mathbf{u}$ [172]. For further details, the work of Volkwein [173] is recommended.

When dealing with discretized data, the minimization problem (2.61) translates to the least-squares loss [172]

$$\mathcal{L}(\phi)=\left\|X-\phi\phi^T X\right\|_F, \tag{2.62}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. When the spatial grid is of size $N_x\times N_z$ and N_t snapshots of the flow have been collected, the two fields can be stored in a data matrix $X\in\mathbb{R}^{2N_xN_z\times N_t}$ with $N_xN_z=N_x\times N_z$. One row of this snapshot matrix then captures the spatial information of the underlying physical fields at a fixed instance of time. One column of X, on the other hand, describes the temporal variation of one of the physical fields at a fixed point in space. The aim of the POD is then to extract the most prominent correlations of this data matrix. Moreover , the i^{th} spatial mode Φ_i is given by the i^{th} column of the spatial mode matrix $\phi\in\mathbb{R}^{2N_xN_z\times 2N_xN_z}$. Note that if ϕ is known, the temporal coefficient matrix $A\in\mathbb{R}^{2N_xN_z\times N_t}$ is computed by projecting the flow data onto the mode basis

$$A=\phi^T X. \tag{2.63}$$

By differentiating (2.62) the minimization becomes an eigenvalue problem [173]

$$XX^T\phi=\phi\Sigma \tag{2.64}$$

where XX^T is the covariance matrix of X. Its eigenvalues[8] s_i are the elements of the diagonal matrix $\Sigma\in\mathbb{R}^{2N_xN_z\times 2N_xN_z}$. Note that the i^{th}eigenvalue represents the individual contribution of the i^{th} mode to the kinetic energy of the flow. However, in practice, the grid size of a sampled flow exceeds its number of snapshots, i.e., $N_x\times N_z\gg N_t$. In this case, XX^T can become an extensive matrix. The complexity of the task can be reduced by using the method of snapshots introduced by Sirovich and Park [174, 175]. This equivalent method converts the original eigenvalue problem (2.64) into one for a $N_t\times N_t$ covariance matrix

$$X^TXA=A\Sigma, \tag{2.65}$$

[8]Note that these eigenvalues are the square of the singular values of X.

where now $A \in \mathbb{R}^{N_t \times N_t}$ and $\Sigma \in \mathbb{N}_{\approx} \times \mathbb{N}_{\approx}$. Therefore the POD computation boils down to the following steps[9]: (i) Collect flow data into a data matrix X, (ii) Compute the snapshot covariance matrix $X^T X$, (iii) Compute its eigenvectors, i.e., the temporal coefficients A, (iv) Compute the spatial modes Φ by projecting X onto A. (v) Truncate the number of spatial modes and time coefficients to a subset $N_{\text{POD}} < N_t$ for data reduction. The resulting linear combination of N_{POD} POD modes produces an approximation to the initial flow fields u_x and u_z that is optimal in the mean square sense. The POD can also be applied to other physical fields than the velocity. Convection flows, which are the topic of this thesis, deal with at least one more additional scalar field, the buoyancy. For these problems, the POD will be applied not only to the flow field but, e.g., to the vector $(u_x(x,z,t), u_z(x,z,t), b(x,z,t))^T$. Therefore, the eigenvalues s_i of the covariance matrix denote the amount of kinetic energy and buoyancy variance captured by the corresponding spatial modes.

The POD is a useful tool that can heavily reduce the degrees of freedom, as will be shown in Section 3.3.2, 4.1 and 4.2 for two-dimensional convection. However, the POD is expected to run into problems when the flow exhibits a broader range of scales. This will naturally occur when the flow becomes highly turbulent, or one considers a full three-dimensional flow, where more complex flow structures can occur. In these cases, the POD will have to, in order to capture enough variance of the flow, consider an increased number of modes N_{POD}. This is a drawback of this linear method. By allowing a nonlinear representation of the physical fields, one might get away with fewer modes and time coefficients. This sets the stage for the next section.

2.3.2 Autoencoder networks

Autoencoder networks are feed-forward neural networks that, similar to the POD, aim to compress the amount of input data. They possess a bottleneck structure in the middle, which involves compression and decompression of the processed input data. A sketch can be seen in Figure 2.6b). AEs consist of two parts: an Encoder and a Decoder network. The former receives an input snapshot, e.g., $\mathbf{x} = (u_x(x,z), u_z(x,z), b(x,z))^T$ and subsequently reduces the number of spatial points via e.g. a pooling operation, where only the maximum data values of a local sliding window are kept. This process is referred to as *Max-Pooling*[10]. In this process of spatial compression, the network learns to extract the main characteristics of the three fields. In the end, the Encoder outputs a low-dimensional latent representation

$$f_{\mathbf{w}_{\text{E}}}(\mathbf{x}) = \mathbf{a} \in \mathbb{R}^{N_{\text{AE}}}, \tag{2.66}$$

where $f_{\mathbf{w}_{\text{E}}}$ denotes the Encoder network and $\mathbf{w}_{\text{E}}$ its trainable weights. Here, $\mathbf{a}$ can be seen as a nonlinear version of the POD time coefficients[11], where the latent dimension N_{AE} takes the place of the number of retained POD modes N_{POD}. Similar to the POD reconstruction, the Decoder network $g_{\mathbf{w}_{\text{D}}}(\cdot)$ receives $\mathbf{a}$ as input and learns to reverse the encoding process.

$$g_{\mathbf{w}_{\text{D}}}(\mathbf{a}) = \mathbf{x}_{\text{AE}}. \tag{2.67}$$

Due to the bottleneck structure of the combined network, the reconstruction process is lossy, i.e., the Decoder outputs will not match the Encoder inputs exactly. This is similar to the

[9]It should be noted that in this way, the first spatial mode will always be the time-mean field $\langle \mathbf{u}(x,z)\rangle_t$, while its time coefficient a_1 will be a constant. This is why it is convenient to subtract the time mean value in a zeroth-step before performing the algorithm.

[10]Another possibility is to retain only the average of this window. This is referred to as *Average-Pooling*.

[11]Both POD and AE latent representations are denoted by the same symbols $\mathbf{a}$. This is to emphasize that the AE is merely a nonlinear extension of the linear POD approach. As POD and AE do not appear in the same section after this chapter, there is no risk of confusion.

reconstruction from a subset of POD modes that only capture a certain variance of the input snapshot. Therefore the AE can be seen as an extension of the linear POD decomposition in (2.56, 2.57) to nonlinear functions (2.66, 2.67).

Finally, the Encoder and Decoder networks are trained in a combined process to minimize the mean-square reconstruction loss

$$\mathcal{L}_{\mathbf{w}_E,\mathbf{w}_D} \sim \|\mathbf{x} - \mathbf{x}_{\mathrm{AE}}\|_2^2 = \|\mathbf{x} - g_{\mathbf{w}_\mathrm{D}}(f_{\mathbf{w}_\mathrm{E}}(\mathbf{x}))\|_2^2 \,. \tag{2.68}$$

As is outlined in 1.2, the weights of the feed-forward neural network are trained via stochastic gradient descent in an iterative process over several epochs and data samples, contrary to the one-shot POD algorithm. Note that linear functions $f_{\mathbf{w}_\mathrm{E}}$ and $g_{\mathbf{w}_\mathrm{D}}$ consisting of a single fully connected layer each would result in (2.62). However, as Encoder and Decoder use nonlinear activation functions $\sigma(\cdot)$, they can approximate a vast amount of functions that may allow for a higher data compression $N_{\mathrm{AE}} < N_{\mathrm{POD}}$, as compared to their linear counterpart. This makes the AE network a promising alternative to the POD approach in fluid dynamics.

A fully connected deep neural network, as shown in Fig. 1.5c), has the flaw that the number of parameters increases rapidly with the number of input features. The latter, however, relies on the spatial resolution of the direct numerical simulation, which must be sufficiently fine to capture phenomena such as turbulent eddies. For this reason, the majority of Autoencoders utilize convolutional layers to capture the primary spatial characteristics of their input. In a convolutional layer, a trainable filter or kernel is convolved with the input snapshot. This is opposed to assigning individual weights to each input data point for a fully connected layer. Take a two-dimensional input array $X \in \mathbb{R}^{N_z \times N_x}$, e.g., a snapshot of the vertical velocity field, with a corresponding convolutional kernel $\kappa \in \mathbb{R}^{n_z \times n_x}$, where $n_x \ll N_x$ and $n_z \ll N_z$. The output of the convolutional layer at some final image position (i, j) can then be written as [53]

$$(X * \kappa)(i,j) = \sum_{p=-\lfloor n_z \rfloor}^{\lfloor n_z \rfloor} \sum_{o=-\lfloor n_x \rfloor}^{\lfloor n_x \rfloor} X(i-p, j-o)\kappa(p,o), \tag{2.69}$$

where $(X * \kappa)$ denotes the convolution of X with κ. For coordinates (i, j) located close to the edges of X, where the kernel partially extends beyond the image, it's common to apply padding with constant values. This ensures that the convolution operation described above remains well-defined. This is similar to the ghost cell concept from computational fluid dynamics. Eq. (2.69) implies that the output of the convolution operation only incorporates local information of X. This locality principle has the advantage that the AE learns not the position of a feature but rather the feature itself. Take an updraft in a flow field, for example. This updraft's position varies with time. Therefore it is meaningful not to learn where updrafts are located but how to identify them no matter their spatial position. This principle of *translation invariance* makes convolutional layers suited for pattern identification. Furthermore, as the receptive field of the kernel is chosen to be much smaller than the dimensions of X, they result in a much smaller number of parameters $\mathbf{w}$ than their fully-connected counterparts. This results in a speed-up of the AE's training process and a reduced amount of memory required to run the neural network. Note that (2.69) can be generalized to input data with several physical fields, by adding a channel dimension to X. Furthermore, the kernel usually maps the output to either an increased number of channels, as is done in the Encoder part, or a decreased number of channels, as is done in the Decoder part. This way, the spatial information is compressed into artificial physical field dimensions during encoding, while the opposite is done during the decoding phase.

While the POD is an established tool for reduced order modeling of fluid flows, Autoencoder networks have only recently been explored in the fluid dynamics community. Such works include ROMs of the viscous Burger flow as well as the inviscid shallow water equations [176]. Other works combine the AE with a long short-term memory network for temporal prediction tasks [177], similar to the approach of this thesis with a Reservoir Computing model. More sophisticated AE make use of physical laws that they embed in their architecture. Such physics-informed networks have recently been applied to reduce the degrees of freedom of three-dimensional homogeneous isotropic turbulence in ref. [72]. Finally, Variational Autoencoders, a form of generative networks where the latent space is sampled from a learned distribution, have shown good performance in terms of sub-grid scale parameterization with the Community Atmosphere Model in ref. [178]. Similar approaches have been explored for the generation of small-scale storms by Mooers *et al.* in ref. [179].

2.4 Combined ROM-RC model for thermal convection flows

Now that the concepts of all machine learning methods have been introduced, their combined application in a ROM-RC model to direct numerical simulation data will be explained in the next section. Additionally, the turbulent flow inferred by this model must be assessed with regard to its predictive performance and its alignment with physical accuracy. This will be done in 2.4.2.

2.4.1 Convection data pipeline: DNS to the predicted turbulent flow

Figure 2.7 shows the pipeline of the thermal convection data for applying the ROM-RC model. The process can be summarized in six steps: (i) Data generation by means of direct numerical simulation of thermal convection, (ii) DNS data post-processing (time sampling, grid interpolation), (iii) Data compression via a ROM, either POD or Encoder network, (iv) Establishing a cross-validation dataset, which involves dividing the data into training, validation, and testing subsets, (v) Latent space prediction via RC model, (vi) Decompression via inverse POD or Decoder network. After the final step, the predicted convection flow can be analyzed and compared to the validation and testing snapshots.

In the first step, the convection data is generated via direct numerical simulations, i.e., the solution of the underlying nonlinear equations of motions. For example, the two-dimensional RBC data in Chapter 4 stems from solving the coupled Boussinesq system (2.17 - 2.19) for a parameter set of Γ, Ra and Pr. For these cases, the spectral element code *Nek5000* [180] was used. The spectral element method [181] decomposes the problem domain into N_e non-overlapping elements. Further, inside each element, the solution fields are given by a local expansion into orthogonal polynomials, e.g., Legendre polynomials, of the order N evaluated at the *Gauss-Lobatto-Legendre* (GLL) quadrature nodes. This spectral representation and the non-uniform GLL nodes result in an approximation error that decreases exponentially with the polynomial degree [182]. The resulting physical fields are then, in the second step, interpolated from the original unstructured element mesh to a uniform grid with resolution $N_x \times N_z$. Moreover, the sampling interval Δt of the DNS snapshots is kept constant. In this way, N_t snapshots are gathered. The simulations presented in Chapter 5 differ in the numerical methods used for solving the underlying dynamics. For these simulations, the finite differences code *tlab* [183] was employed. The post-processing, however, is the same as mentioned above. In the third step, the DNS data is reduced via POD (or AE network) to a set of $N_{\text{POD}} \ll N_x \times N_z$ (or $N_{\text{AE}} \ll N_x \times N_z$) temporal modes with N_t snapshots. Out of these, a subset of length N_{train} will be used for training the reservoir later on. Two further,

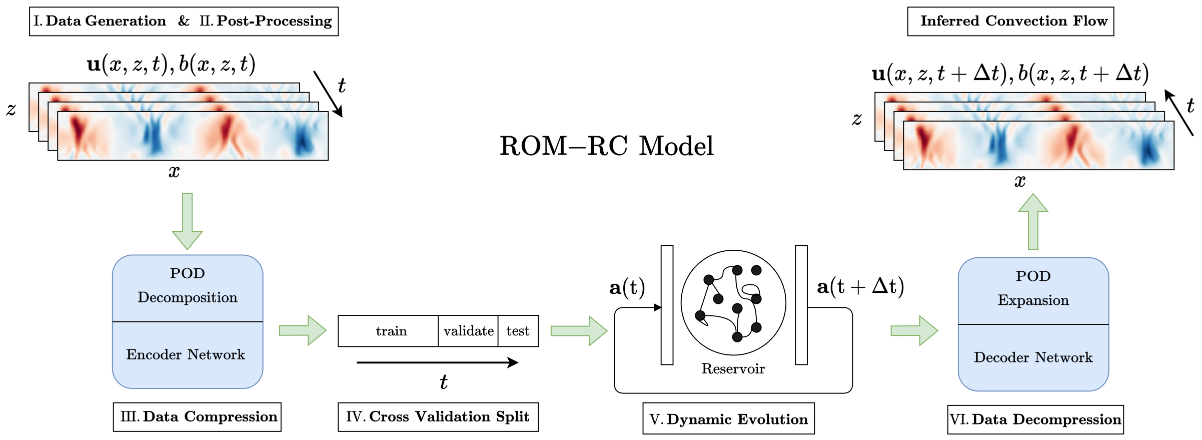

Figure 2.7: **Convection data pipeline** using the example of two-dimensional dry convection: I. Snapshots of u_x, u_z and b are generated by means of direct numerical simulations. II. The data is post-processed, e.g., interpolated to a uniform grid of size $N_x \times N_z$. III. The post-processed data is compressed to a low-dimensional space $\mathbf{a}$ by means of a POD or an Encoder network. Hence, the convection dynamics are translated to dynamics of the latent space $\mathbf{a}(t)$. IV. The low-dimensional time series data is split into a training, validation, and testing dataset. The resulting datasets can be used to cross-validate the ML model's performance. V. A RC model is trained with the training data of $\mathbf{a}(t)$. It is then used to infer subsequent time instance $\mathbf{a}(t+\Delta t)$. VI. The inferred latent space is decompressed using the inverse POD or a Decoder network. In this way, the initial snapshot of u_x, u_z, and b are advanced in time by means of a Reduced Order Model and a Reservoir Computing model.

non-overlapping subsets of length N_{val} and N_{test} will be used for cross-validating the final ROM-RC model[12]. After that, the RC model is trained with the training dataset. Then, it can be used for the task of predicting the latent space variables $\mathbf{a}$ at further time steps. The inferred variables can then be expanded to the physical space again, s.t., the corresponding inferred velocities, and buoyancy can be analyzed. In this context, the validation dataset is utilized to identify the optimal hyperparameters for the ESN, which are then tested on the testing dataset.

2.4.2 Assessing the accuracy of the inferred flow

Here, the methodologies used to measure the predictive algorithm's accuracy and overall performance are explored, permitting the user to optimize the ESN's hyperparameters. As mentioned above, it is common to prepare, besides a training dataset, a validation dataset with which the model output can be reconciled. Several metrics qualify for the aforementioned purpose. The dynamic core in step V. of Fig. 2.7 generates a multivariate time series of either POD time coefficients or AE latent space variables. Therefore, a direct comparison of inferred and ground truth dynamics is a natural choice for assessing the accuracy of the ML method. For this, the mean squared error between reservoir output $\mathbf{y}(n)$ and the ground truth signal $\mathbf{a}(n)$ is suitable

$$\mathcal{E}_{\text{MSE}}(\mathbf{a}, \mathbf{y}) = \frac{1}{N_{\text{val}}} \sum_{n=1}^{N_{\text{val}}} \|\mathbf{a}(n) - \mathbf{y}(n)\|^2. \tag{2.70}$$

Alternatively, using the normalized root-mean-squared error (NRMSE) is common, defined as[13]

$$\mathcal{E}_{\text{NRMSE}}(\mathbf{a}, \mathbf{y}) = \frac{\sqrt{\mathcal{E}_{\text{MSE}}(\mathbf{a}, \mathbf{y})}}{\max(\mathbf{a}) - \min(\mathbf{a})}. \tag{2.71}$$

As the NRMSE allows for a better comparison between datasets, it will be employed in parts of this thesis. However, using the MSE and NRMSE in the context of temporal prediction tasks has a major drawback. As will be elaborated upon in the upcoming chapters, the dynamics inferred by an ESN using the closed-loop mode will rapidly diverge from the actual ground truth signal. Characteristic temporal frequencies may, however, be retained. Consequently, this will lead to a higher NRMSE value, as the NRMSE quantifies merely the proximity between the two signals $\mathbf{a}$ and $\mathbf{y}$. Moreover, it has been observed that a constant ESN output signal $\mathbf{y} = \mathbf{const}$ often results in a lower absolute error value than predictions which reproduce physical properties [113]. Similar results have been observed for the NRMSE measure in the course of the author's doctoral studies. A simple example is two sinusoidal time series with a constant non-zero phase shift. If the phase shift is sufficiently large, the mean square deviation between both signals might become larger than a constant signal without any dynamics. This can lead to misleading conclusions, as a minimum in $\mathcal{E}_{\text{NRMSE}}$ need not correspond to the best ESN prediction. It is, therefore, important to evaluate the ESN performance also in regard to the statistical and physical properties of the reconstructed flow.

One such measure can be derived on the basis of the super-parameterization scheme, presented in 1.1.3. There, it was mentioned that the mean vertical properties of flow and thermodynamic scalars are particularly interesting for super-parameterizations, as they are the

[12]In case of the Autoencoder training procedure, a similar split is done. More details in 5.1.

[13]Other definitions use the mean or standard deviation of the ground truth signal in the denominator.

link over which the cloud-resolving model and GCM share information. Hence, it is important for future ROM-RC models to replicate these essential properties of the underlying ground truth flow. Therefore, it is logical to define an error measure via the vertical profiles, i.e., the lateral-time averages $\langle\cdot(z)\rangle_{x,t}$ (for the two-dimensional case) of ground truth and the reconstructed ESN predictions. For this, the normalized average relative error (NARE) [184] $\mathcal{E}_{\text{NARE}}\left(\cdot\right)$ can be introduced. For the buoyancy flux $u_z b$, it is defined as

$$\mathcal{E}_{\text{NARE}}\left(\langle u_z b\rangle_{x,t}\right) = \frac{1}{C_{\max}} \int_0^1 \left| \langle u_z b(z)\rangle_{x,t}^{\text{ESN}} - \langle u_z b(z)\rangle_{x,t}^{\text{GT}} \right| dz\,, \tag{2.72}$$

with the constant

$$C_{\max} = 2 \max_{z\in[0,1]} \left(\left| \langle u_z b(z)\rangle_{x,t}^{\text{GT}} \right| \right). \tag{2.73}$$

Here, the superscripts indicate whether the profile stems from the reconstructed ESN predictions or the ground truth (POD or AE). Moreover, the NARE for the buoyancy flux is particularly interesting, as it incorporates the error of two fields. Finally, in order to measure whether several profiles are matched, it is convenient to compute the arithmetic average value of their NARE values. For example, the error of the root mean square profiles of vertical and horizontal velocities is given by the average NARE

$$\overline{\mathcal{E}_{\text{NARE}}} = \frac{1}{2}\left(\mathcal{E}_{\text{NARE}}\left(\langle u_x^2\rangle_{x,t}^{1/2} \right) + \mathcal{E}_{\text{NARE}}\left(\langle u_z^2\rangle_{x,t}^{1/2} \right) \right). \tag{2.74}$$

Now that all methods have been introduced, the next chapter will present results for an ESN applied to the dynamics of a low-order model of thermal convection. It will introduce the reader to common procedures, like hyperparameter tuning, and familiarize them with the network's capabilities. Furthermore, the low-dimensional system will be used to test the QRC algorithm introduced above and compare its performance with the original ESN.

Chapter 3

Reservoir Computing of a low-order model of convection

A low-order model (LOM) describes a system of ordinary differential equations with few degrees of freedom derived from the basic equations of atmospheric motion [185]. The first LOM originated during the 1960s [186, 187] when computational resources were scarce and mostly unavailable to meteorologists [185]. By using LOMs, researchers were able to reproduce qualitative aspects of the general circulation while keeping the computational cost low. Even after the exponential increase in computational power and the availability of computational resources in the subsequent decades, the interest in LOMs prevailed. Their simplicity allowed researchers to focus on understanding the basic mechanisms of atmospheric motion and turbulence. To this day, many LOMs have been proposed. Amongst others, this includes the turbulent thermal convection models by Saltzman [188], Lorenz[1] [187], Howard and Krishnamurti [189], Grossmann and Lohse [190], Gluhovsky and Tong [191] and Gluhovsky *et al.* [192]. Further examples comprise multi-scale atmospheric flow as introduced by Lorenz[2] [193], shell models of homogeneous isotropic turbulence by Eggers and Grossmann [194] and Grossmann and Lohse [195], as well as a turbulent shear flow LOM by Moehlis *et al.* [196]. Their simplistic nature and few degrees of freedom have made LOMs an ideal testing bed for Echo State Networks for the prediction of nonlinear dynamics. Applications to the Lorenz 63 dynamics include the correct reproduction of its chaotic attractor and non-negative Lyapunov exponents [87, 197], as well as the reconstruction of two missing modes by a continuously available measurement of the third [110]. Moreover, attempts of transfer learning, a generic term for the generalization capability of a neural network, between two Lorenz systems with different Rayleigh numbers have been made in [198]. In [88], Pershin *et al.* show that an ESN can correctly infer the turbulent-to-laminar transitions in Mohelis *et al.*'s shear model. Furthermore, LOMs are often used as proof of principle for new RC architectures [86, 115, 198, 199] or new theoretical approaches towards a consistent definition of the echo state property [128]. In this spirit, this chapter will apply an Echo State Network to a LOM with eight degrees of freedom. The results of this chapter were published in two works [108] and [158].

3.1 An eighth-order Lorenz model of thermal convection

One of the first LOMs was developed by Edward N. Lorenz in 1963 [187], therefore often referred to as the Lorenz 63 model. It describes two-dimensional incompressible thermal convection between two impermeable parallel plates with Dirichlet boundary conditions for the temperature and free-slip boundaries for the velocity field. For a detailed derivation, the reader is referred to Appendix A.1.1. The starting point for the derivation is the two-dimensional form of the Boussinesq system (2.8 - 2.9). The LOM is obtained by expanding

[1]This LOM is commonly known as Lorenz 63 model.
[2]This LOM is commonly known as Lorenz 96 model.

the temperature deviation from the diffusive profile $\theta(x,z,t)$ and stream function $\zeta(x,z,t)$ into a finite series of trigonometric Fourier modes and keeping only the quadratic nonlinearities. By restricting the expansions to only a few modes, a system of coupled ordinary differential equations is obtained. This chapter will consider four modes for the temperature field and another four modes for the stream function. This results in an eighth-order Lorenz model, as introduced in [192]. It reads

$$\frac{dA_1}{dt} = \mathrm{Pr}\,(B_1 - A_1) - c_{A_{1,1}} A_2 A_3 + c_{A_{1,2}} A_3 A_4, \tag{3.1}$$
$$\frac{dA_2}{dt} = -\frac{\delta \mathrm{Pr}}{4} A_2 - \frac{3}{2\sqrt{2}} A_1 A_3, \tag{3.2}$$
$$\frac{dA_3}{dt} = -c_{A_{3,1}} \mathrm{Pr} A_3 - c_{A_{3,2}} \mathrm{Pr} B_3 + c_{A_{3,3}} A_1 A_2 + c_{A_{3,4}} A_1 A_4, \tag{3.3}$$
$$\frac{dA_4}{dt} = -\frac{9\delta \mathrm{Pr}}{4} A_4 - \frac{1}{2\sqrt{2}} A_1 A_3, \tag{3.4}$$
$$\frac{dB_1}{dt} = -B_1 + rA_1 - A_1 B_2 + \frac{1}{2} A_2 B_3 + \frac{3}{2} A_4 B_3, \tag{3.5}$$
$$\frac{dB_2}{dt} = -\delta B_2 + A_1 B_1, \tag{3.6}$$
$$\frac{dB_3}{dt} = -c_{B_{3,1}} B_3 - A_2 B_3 + \sqrt{2} r A_3 + 3 A_4 B_1 - 2\sqrt{2} A_3 B_4, \tag{3.7}$$
$$\frac{dB_4}{dt} = -4\delta B_4 + \frac{3\sqrt{2}}{4} A_3 B_3, \tag{3.8}$$

where r is the Rayleigh number relative to its critical value $\mathrm{Ra}_{\mathrm{crit}}$ for the onset of convection [200] and δ is a geometric parameter connected to the aspect ratio Γ of the convection domain. $c_{A_{ij}}$ and $c_{B_{ij}}$ are constants. The full LOM with the definitions of these constants is given in Appendix A.1.2. Note that the classical Lorenz 63 model[3] is obtained when letting $A_2 = A_3 = A_4 = 0$ and $B_3 = B_4 = 0$. Furthermore, the system is defined by the parameter space (Pr, r, δ).

Considering only these eight modes is a severe reduction of the complexity of the original Boussinesq system. Nevertheless, in the manner of LOMs, this truncation retains several major physical effects: (i) Conservation of energy

$$E = \frac{1}{2\Omega} \int_\Omega ((\nabla\zeta)^2 - 2\mathrm{Ra}\mathrm{Pr}\theta z) d\Omega\,, \tag{3.9}$$

with a kinetic and potential energy term. Here Ω again represents the convection domain. Gluhovsky *et al.* [192] showed that eqs. (3.1-3.8) can be transformed into a system of coupled gyrostats and hence conserve expression (3.9). (ii) Tilted convictions rolls, i.e., vertical shear motion, can be observed in this extension. This property was also addressed by the sixth-order Lorenz-like model of Howard and Krishnamurti [189], which introduced two additional temperature modes and a new shear mode to the Lorenz 63 model. However, their model did not conserve energy. (iii) Finally, the present model conserves vorticity [201] in the convection domain

$$\frac{1}{\Omega} \int_\Omega \omega d\Omega\,, \tag{3.10}$$

[3]The formulation that is found in literature is obtained by letting $X = A_1$, $Y = B_1$, $Z = B_2$, $\sigma = \mathrm{Pr}$ and $b = \delta$.

with vorticity $\omega = -\nabla^2\zeta$. While property (i) also holds for the original model proposed by Edward N. Lorenz, properties (ii) and (iii) are not captured. The lack of shearing motion can be seen in Figure 3.1 (a,b,d,e), which compares the reconstructed velocity and temperature fields for the present model with eight degrees of freedom and the third-order Lorenz model. Figure 3.1 (c,f) shows the chaotic attractor spanned by A_1, B_1, B_2 for both models.

To assess the ESN's prediction horizon, it is convenient to express the time coordinate in a meaningful time scale. The inverse of the maximum Lyapunov exponent Λ_1 introduces such a time scale for chaotic systems. The Lyapunov exponent Λ_1 quantifies the deterministic chaos of a dynamical system. After one Lyapunove time Λ_1^{-1}, two initially close trajectories will have diverged by a factor of e^{-1}. Thus, an effective predictor can be characterized by its ability to produce outputs that closely match the actual trajectory for a duration of at least one period, or potentially even longer. It can be estimated numerically by the method of Sprott [202]. There, the trajectory of the dynamical system is perturbed in one of its dimensions while maintaining a copy of the original, unperturbed trajectory. The evolving difference in the state is recorded. The perturbed trajectory is then renormalized to avoid computational overflow. By averaging this process over many time instances and repeating the procedure for all eight state dimensions, one ends up with an estimate for the leading Lyapunov exponent. For the system (3.1-3.8), with the parameter set listed below, it was estimated as $\Lambda_1 = 0.825$, which is a bit smaller than the value for the Lorenz 63 model [203]. In the next section, this model will be used to train an Echo State Network for the task of prediction.

3.2 Reservoir Computing of an eighth-order Lorenz model

Equations (3.1 - 3.8) are solved for $(\mathrm{Pr}, r, \delta) = (10, 28, 8/3)$ with a fourth-order Runge-Kutta scheme for a period of 163 Lyapunov times ${\Lambda_1}^{-1}$. Further information on the chosen Lorenz parameters is detailed in Table 3.1. This setting corresponds to a common Lorenz 63 configuration known to exhibit chaotic behavior. Here, $\delta = 8/3$ corresponds to a convection cell with aspect ratio $\Gamma = 2\sqrt{2}$, the critical wavelength at which the flow becomes linearly unstable. Eqs. (3.1-3.8) are easily solved numerically. Hence, there is no computational benefit in applying an ML model to avoid solving the systems equations of motion. Nevertheless, this aids in comprehending specific mechanisms of the machine learning model. This encompasses aspects like the prediction horizon, dependencies on hyperparameters, and the variation among ensembles of the RC model. Only upon recognizing these mechanisms can conclusions be drawn regarding the performance of the ESN on a real turbulent flow, as discussed in Chapter 4.

An ESN is trained using a subset of the generated data in the time interval $[0, t_{\mathrm{train}}]$ with $t_{\mathrm{train}} = 7.1\Lambda_1^{-1}$. It is discretized by a sampling interval of $\Delta t = 1.65 \cdot 10^{-4}{\Lambda_1}^{-1}$. Specifically, the ESN is employed in the closed-loop setting and is tasked to output $\mathbf{y}(n) = \mathbf{x}(n+1)$ when receiving $\mathbf{x}(n)$. The reservoir input comprises all eight Lorenz modes

$$\mathbf{x}(n) = (A_1(n), A_2(n), A_3(n), A_4(n), B_1(n), B_2(n), B_3(n), B_4(n))^T . \tag{3.11}$$

After training, the ESN runs autoregressively, using its output as the next input. A grid search is utilized to make a statement about hyperparameter dependencies. Luckily, the low degrees of freedom allow for an exhaustive grid search of ESN hyperparameters while maintaining a low computation time. The following hyperparameters are chosen for the search: leaking rate and spectral radius are important parameters for capturing the correct time

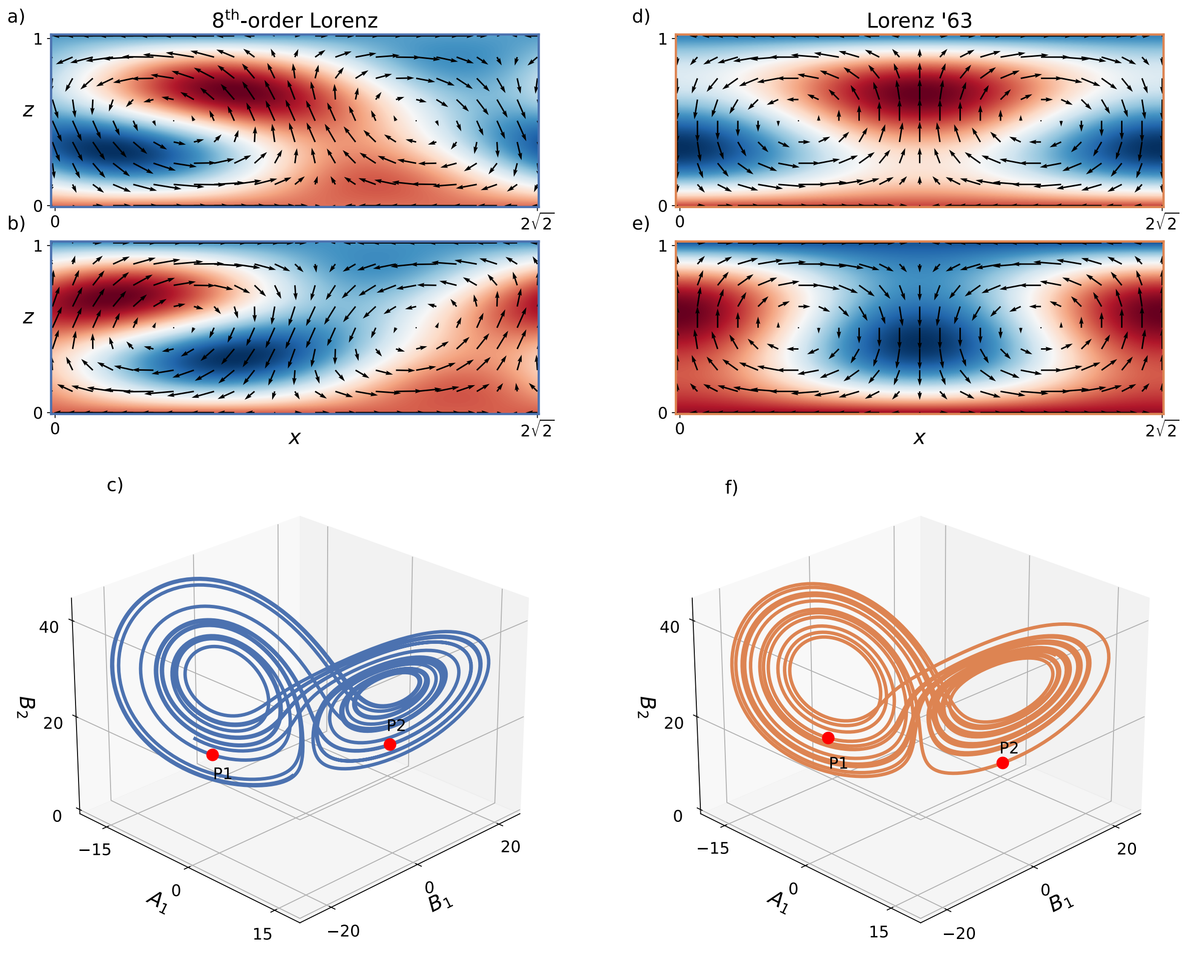

Figure 3.1: Comparison of **two Lorenz models** with eight (a-c) and three modes (d-f). The chaotic attractor spanned by (A_1,B_1,B_2) is shown in (c,f). The reconstructed temperature field $\theta(x,z)$ is shown in (a,b,d,e) at the two times P1, P2 (red symbols in c,f). The reconstructed two-dimensional velocity field $(u_x(x,z), u_z(x,z))$ is superimposed. The convection rolls in (a,b) are tilted, signifying the shear motion captured by the additional modes in $\zeta(x,z,t)$. Table 3.1 lists the corresponding Lorenz parameters.

Pr	r	δ	Ra	$\mathrm{Ra}_{\mathrm{crit}}$	Γ	Λ_1	$\Lambda_1 t_{\mathrm{solve}}$	$\Lambda_1 \Delta t_{\mathrm{solve}}$
10	28	8/3	$1.84 \cdot 10^4$	$27\pi^4/4$	$2\sqrt{2}$	0.825	163	$1.65 \cdot 10^{-4}$

Table 3.1: **Lorenz model simulation parameters** (left) and resulting physical parameters (right). The model parameters are the Prandtl number Pr, the relative Rayleigh number r, and the geometric parameter δ. The resulting parameters are the Rayleigh number Ra, its critical value for the onset of convection $\mathrm{Ra}_{\mathrm{crit}}$, and the aspect ratio of the convection domain Γ. Λ_1 is the leading order Lyapunov exponent of the eighth-order system for the current parameter setting. Its inverse corresponds to the time scale after which two initially close trajectories depart from each other. Eqs. (3.1 - 3.8) were solved by means of a fourth-order Runge-Kutta scheme. The time stepping was $\Delta t_{\mathrm{solve}}$. The solution trajectory comprises a period of length t_{solve}.

scales in the data. In contrast, the regression parameter stabilizes the delicate closed-loop reservoir dynamics and, therefore, will be considered as well. Moreover, the size of the reservoir will be examined to determine if the model exhibits improvement as the number of neurons increases. Finally, each hyperparameter setting was run for 100 different random initialization of W^r and W^{in}. Further details on the grid search are listed in Table 3.2.

γ_r	ϱ_r	λ_r	N_r
$[0.1, 1.0, 0.1]$	$[0.0, 1.5, 0.1]$	$\{0.05, 0.5, 5.0\}$	$\{64, 128, 256, 512, 1024\}$

Reservoir realizations	$\Lambda_1 t_\text{train}$	$\Lambda_1 t_\text{val}$	$\Lambda_1 t_\text{test}$
100	7.10	5.00	7.00

Table 3.2: Closed-loop ESN **hyperparameter grid search** procedure for the eight-mode Lorenz system. The reservoir density was fixed at $D_\text{r} = 0.2$, i.e., only 20% of intra-reservoir links are active. The input scaling was $\sigma_\text{r} = 2$. Square brackets indicate the range of the searched hyperparameter: $[\text{start}, \text{end}, \text{stepsize}]$. Braces indicate an array of values that were used. $t_\text{train}, t_\text{val}, t_\text{test}$ are the length of the training, validation, and testing interval, respectively. The total number of different reservoir settings is $10 \times 16 \times 3 \times 5 = 2400$, while a total of $100 \times 2400 = 2.4 \cdot 10^5$ reservoirs were run. The grid search run time was 18.6h using the *turbESN* library [123].

Figure 3.2a) shows an excerpt of the $(\varrho_\text{r}, \gamma_\text{r}, N_\text{r})$ landscape of the NRMSE between reservoir predictions and ground truth Lorenz dynamics during the validation phase. From a practical point of view, one is interested in the average reservoir performance, i.e., given a reservoir setting Ξ_r, what performance can you expect independent of the random realization of W^{in}, W^r. For this purpose, the median reservoir realization's $\mathcal{E}_\text{NRMSE}$ values are shown. Several things can be observed: sufficiently low leaking rate and spectral radius values are desirable as NRMSE values reach a minimum. Further, the ESN performance increases as the number of neurons increases. See the expanding low-error region at low $(\varrho_\text{r}, \gamma_\text{r})$ values. This seems natural, as a higher number of neurons should increase the approximation capabilities of a neural network. However, the minimum value, marked by the green star symbol, is not found at the grid search's maximum value $N_\text{r} = 1024$, but at 512. Nevertheless, the high-error region for large $(\varrho_\text{r}, \gamma_\text{r})$ values is increasingly dissolved as the number of neurons increases. Overall the loss-landscape shows the complex interdependencies of the ESN hyperparameters.

γ_r	ϱ_r	λ_r	N_r	D_r	σ_r	$\mathcal{E}_\text{NRMSE}$
0.10	0.30	5.00	512	0.20	2.00	0.272

Table 3.3: Best **ESN hyperparameter settings** Ξ_r after analysis of the normalised root-mean-square error in a hyperaparameter grid search. This setting is also marked by the green star symbol in the error landscape in Fig. 3.2a).

Based on this error landscape, the best hyperparameter set Ξ_r can be identified. It is listed in Tab. 3.3 together with the corresponding NRMSE value. Fig. 3.2b) shows the outputs of this reservoir setting during the validation phase together with the ground truth (blue). The orange curves mark the outputs of all 100 reservoir realizations of $W^{\text{in}}, W^\text{r}$. It can be

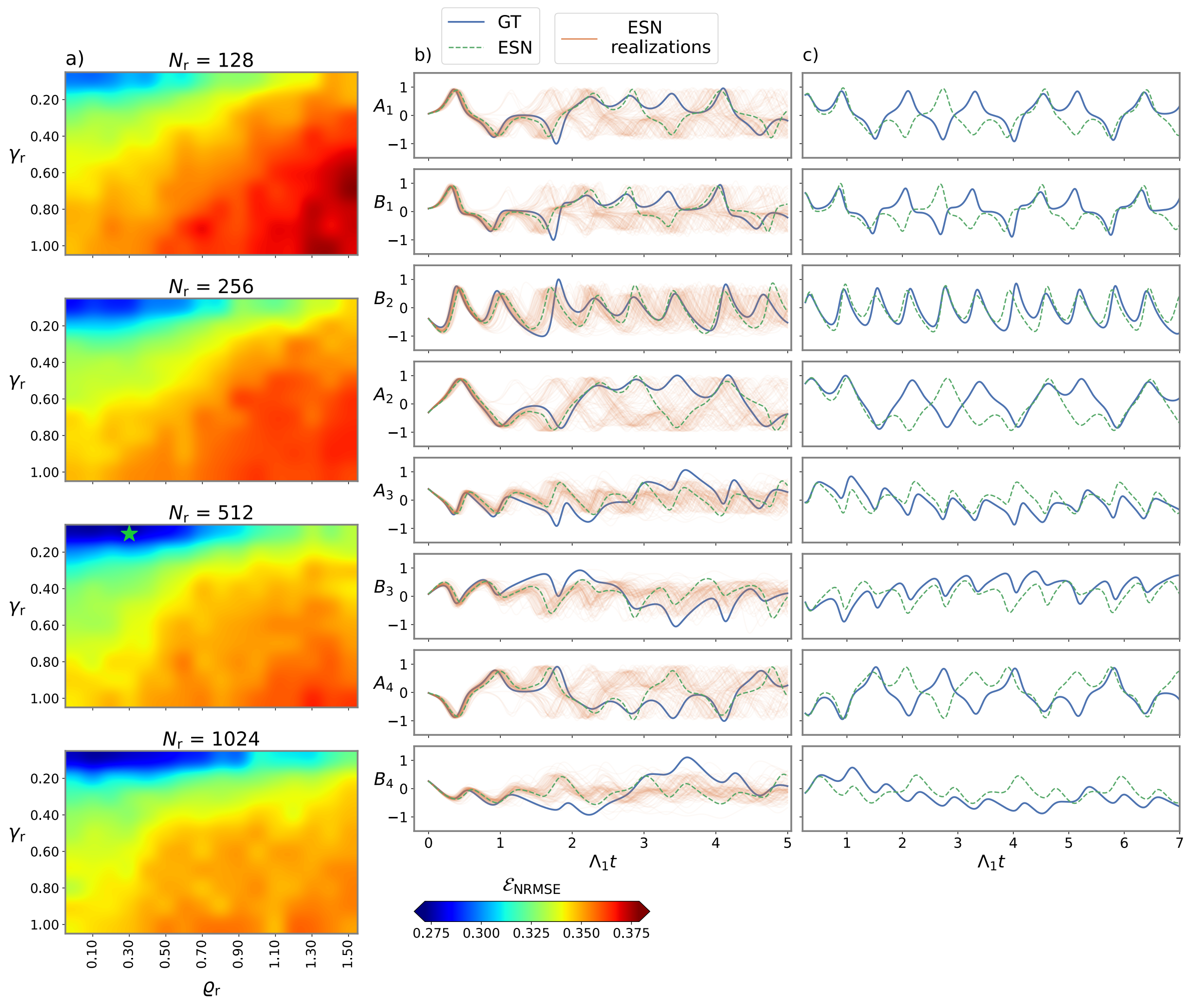

Figure 3.2: Results of the **grid search** described in Table 3.2. (a) Validation $\mathcal{E}_{\text{NRMSE}}$ landscape for ESN hyperparameters $(\varrho_\text{r}, \gamma_\text{r}, N_\text{r})$ $(\lambda_\text{r} = 5.0)$. Shown is the median value of all reservoir realizations. (b) ESN outputs during the validation phase. Shown are the ground truth (blue) and the with a) associated best ESN prediction (green). The corresponding hyperparameter setting is marked by the green star symbol in (a). Moreover, the outputs of the remaining 99 reservoir realizations (orange) with this hyperparameter configuration are shown as well.

seen that all reservoirs reproduce the ground truth reasonably well up to 2 Lyapunov times. After that, the predictions start to deviate from the reference curve. After $2{\Lambda_1}^{-1}$, a spread among the ensemble members can be observed. As a reference, the best of these ESN predictions is shown in green. It corresponds to the lowest NRMSE value among the realizations. Here, it can be seen that even though the accumulated prediction error leads to deviations from the real dynamics, the inferred dynamics reproduce the temporal features of the LOM. This example demonstrates nicely how delicate and complex the ESN performance can be. While the closed-loop scenario permits autonomous predictions for longer than several Lyapunov times, it depends on the right combination of reservoir hyperparameters. Moreover, the random realization of the input and recurrent weight matrices generally leads to differing ESN output trajectories. However, choosing the average hyperparameter setting leads to the same initial dynamics. Furthermore, the ESN should not be expected to reproduce the exact same trajectory as the chaotic reference signal. Due to the chaoticity of the LOM, even small deviations would inadvertently amplify. The same goes for the Echo State Network, which is prone to prediction errors that accumulate over the autonomous prediction phase. However, the inferred dynamics should be seen as a new realization of the learned chaotic system. In this way, the reservoir can be understood as a generative network that produces synthetic time series. This can checked by looking at Fig. 3.2c), which tests the chosen Ξ_r during a testing phase. A quick deviation from the reference curve can be observed while the characteristic features of each mode are reproduced. The ESN, therefore, seems to mimic the ground truth dynamics. This hypothesis can be checked by analyzing further characteristics of the inferred signal. For the present LOM, it is convenient to analyze the reproduction of the chaotic attractor spanned by A_1, B_1, B_2 and the reconstructed physical fields. In 3.3.2 and the next chapters, this analysis will also consider the statistical properties of the reconstructed flow fields. Fig. 3.3 shows the three snapshots of the reconstructed physical fields during the testing phase. It can be observed that the ESN outputs also reproduce the characteristic Lorenz attractor with its butterfly wing shape. Note that even though the three time instances (blue, orange, and green markers) are not found at the same position, the reconstructed temperature and velocity fields match reasonably well. The shearing fluid motion is inferred to a good extent as well, while minor differences can be seen in 3.3(e,f).

Finally, this proof of concept is a good way to analyze the machinery of the ESN that produced the results in figures 3.2 and 3.3. In Fig. 3.4 the three reservoir weight matrices W^r, W^in and W_*^out are shown. The hyperparameter set was Ξ_r, as listed in table 3.3. As described in 2.2.2.2, the components of W^r are generated randomly. The entries describe the connection strength between two or the same reservoir nodes. Further, the reservoir density of $D_\mathrm{r} = 0.2$ leads to most of these connections being zero (green color code). Note that the magnitude of the entries depends on the chosen spectral radius ϱ_r. The input weights W^in are also generated randomly. However, it is a dense matrix that distributes the Lorenz input information $\mathbf{x}$ on all N_r nodes. The magnitude of its entries is controlled by the input scaling parameter σ_r. Finally, the entries of the output matrix W_*^out are obtained by the linear regression procedure (2.45) and link the reservoir state to the reservoir output, i.e., the inferred Lorenz modes. Fig. 3.5 shows the dynamics of six reservoir state components r_i during training ($t \leq 0$) and testing phase ($t > 0$). It can be observed that the reservoir state exhibits similar dynamics to the underlying nonlinear Lorenz model. Note that these signals, together with the output weights shown in Fig 3.4c) produce the reservoir outputs in Fig. 3.2b).

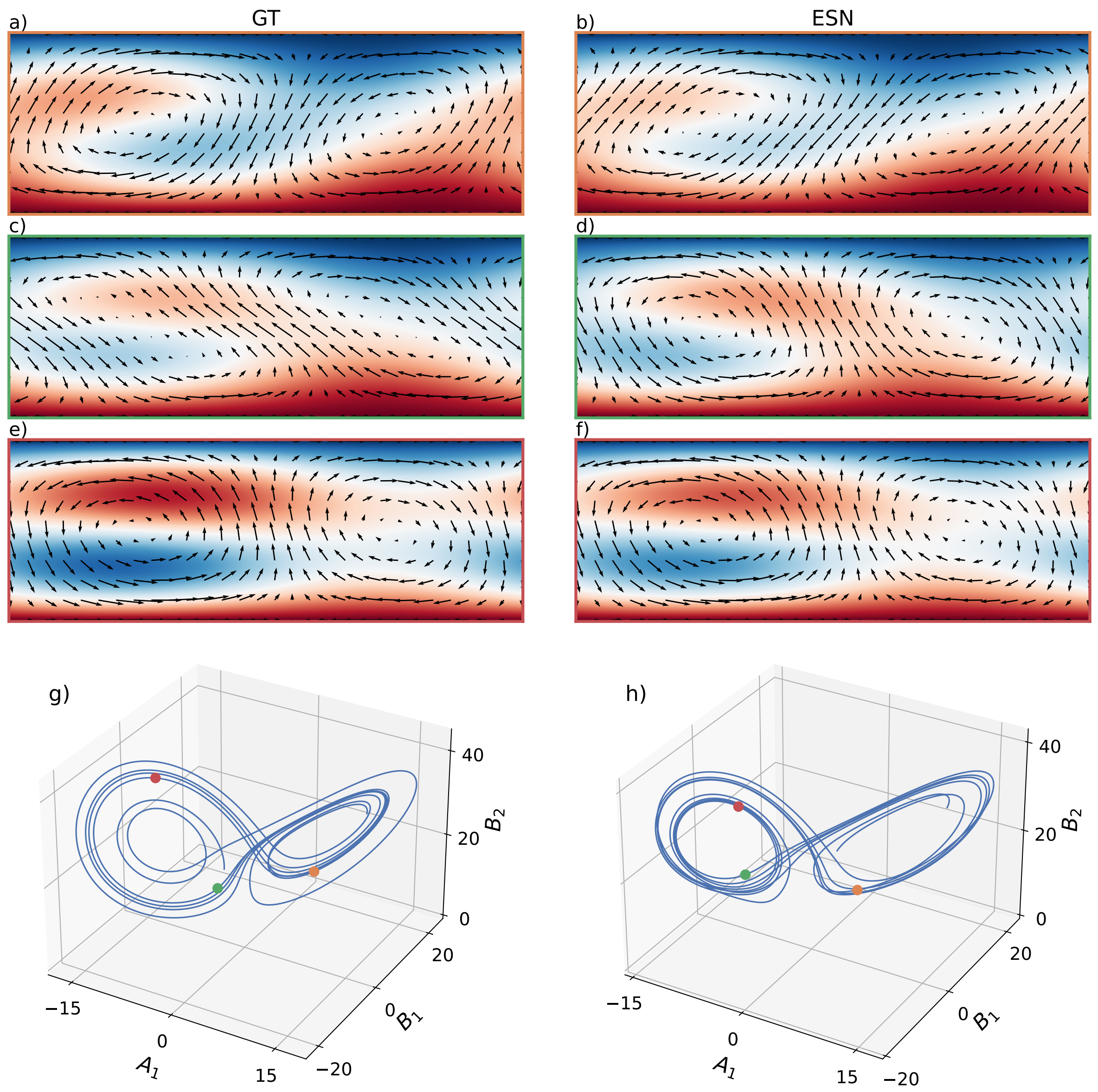

Figure 3.3: **Reconstructed temperature field** $T(x, z)$ of ground truth (a,c,e) and ESN prediction (b,d,f) in the testing phase at three instances in time. See color frame and markers in (g,h). The reconstructed velocity fields $u_x(x, z)$ and $u_z(x, z)$ are superimposed. The reconstructed chaotic attractor is shown in (h) for the ESN case. (g) shows the corresponding ground truth trajectory.

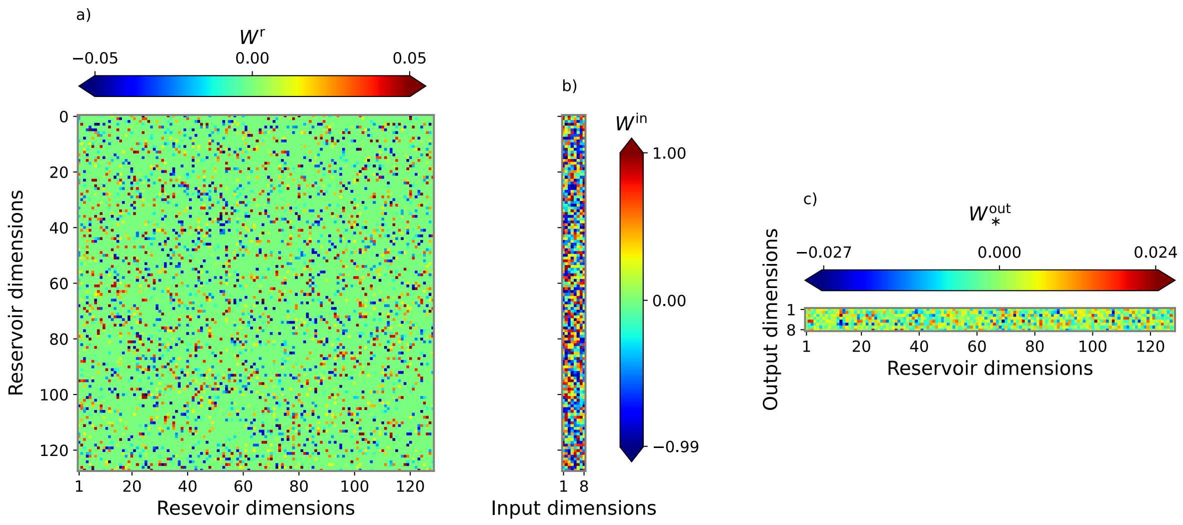

Figure 3.4: Realization of the three **reservoir matrices**: the intra-reservoir matrix W^{r} (a), the input matrix W^{in} (b) and trained output matrix W_*^{out}. The corresponding ESN setting is listed in Tab. 3.3. For visualization purposes, only the first 128 (out of $N_{\mathrm{r}} = 512$) reservoir dimensions are shown.

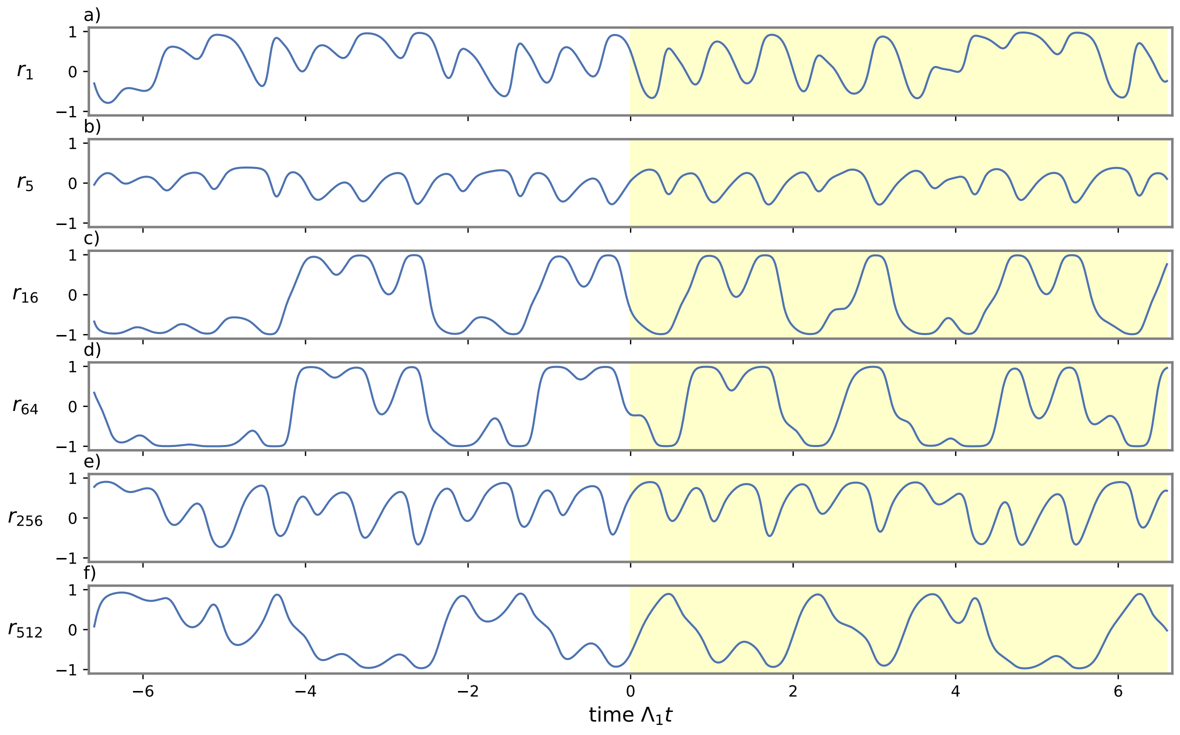

Figure 3.5: **Reservoir dynamics r**(t) to the corresponding ESN setting listed in Tab. 3.3. Shown are the dynamics during the training phase $t \leq 0$ and the testing phase $t > 0$ (highlighted in yellow). The time axis aligns with the testing outputs shown in 3.2b).

3.3 Hybrid quantum-classical Reservoir Computing of thermal convection

Now that the ESN and its closed-loop mode of operation have been introduced on the basis of a LOM of thermal convection, this section will focus on a second Reservoir Computing implementation, namely its digital gate-based quantum computing implementation. As introduced in 2.2.4, Quantum Reservoir Computing uses quantum systems to process, primarily classical, information. In a nutshell, they are recurrent machine learning algorithms for which the reservoir state is built by a highly entangled tensor product quantum state that grows exponentially in dimension with the number of qubits. However, current NISQ devices are prone to computational errors and, therefore, are not yet suited for computationally heavy and complex tasks. For this reason, dynamical systems with few degrees of freedom, as the presented Lorenz model, are ideal testing beds for the application of a quantum reservoir computer. Therefore, in 3.3.1, a QRC algorithm will be tested on the eight degrees of freedom of the above-presented LOM. In 3.3.2, a truncated POD model with 16 modes will be used to test an improved QRC model. Finally, their performance will be compared to the classical ESN implementation.

3.3.1 Quantum Reservoir Computing of an eighth order Lorenz model

The quantum circuit of Algorithm H1 is set up using the *Qiskit* library [204]. Specifically, the circuit will run the algorithm in the open-loop mode, where two of the eight Lorenz modes, namely A_4 and B_3, are continuously provided to the network. See also Section 2.2.2.3 again for the open-loop procedure. Finally, the network is tasked with reconstructing all eight modes. To this end, a seven qubit ($N_{\text{qubit}} = 7$) quantum circuit will be employed. The circuit architecture is displayed in Fig. 3.6a). The corresponding reservoir dimension is consequently $N_\text{r} = 2^{N_{\text{qubit}}} = 128$. The circuit is kept shallow to decrease the decoherence of the qubits and, hence, the error of the model. The inputs $\mathbf{x}(n) = (A_4(n), B_3(n))$ are encoded as angles of the rotational gates R_Y. Further, to reduce the cost of loading the data to the quantum register, only a subset of 14 reservoir dimensions $r_{H1,i}$ are fed back to the reservoir. Hence, the nonlinear memory of this model comprises 14 nodes. However, all 128 dimensions are employed to compute the reservoir output via eq. (2.55). The reservoir then evolves according to eq. (2.50). Remember that at each iteration, the circuit must be run several times in order to compute the probabilities $\mathbf{p}$ of all $2^{N_{\text{qubit}}}$ basis states. See again eq. (2.51). Here, 2^{17} measurements of the final quantum state were performed. This leads to a computation time of about 3s per reservoir iteration n [205]. Moreover, no error corrections were performed. The circuit was run on the quantum device *ibm_perth*, which consists of seven superconducting qubits. See Fig. 3.6b) for its qubit connections.

The Lorenz data from Tab. 3.1 is used to construct a training dataset of length $\Lambda_1 t_{\text{train}} = 7.84$ and a validation set of length $\Lambda_1 t_{\text{val}} = 7.84$. In this first proof of concept, a testing data set is neglected to keep the computation times of the QRC model at a minimum. Moreover, no hyperparameter search was conducted for the present QRC model, and only one reservoir realization is considered. Finally, the leaking rate was set to $\gamma_\text{r} = 0.3$. Fig. 3.7 shows the QRC model predictions for the case that two of the eight Lorenz modes are provided as teacher input at each reservoir iteration. To back up the results of the quantum device, two *Qiskit* simulation runs were conducted. One is the ideal *Qiskit* simulator. The other simulation was done on a noisy *Qiskit* simulator for which one can describe the probabilities of measurement errors, here 0.05, gate errors, here 0.1, and qubit resets, here 0.03. These values have been chosen such that they come close to those of an actual quantum de-

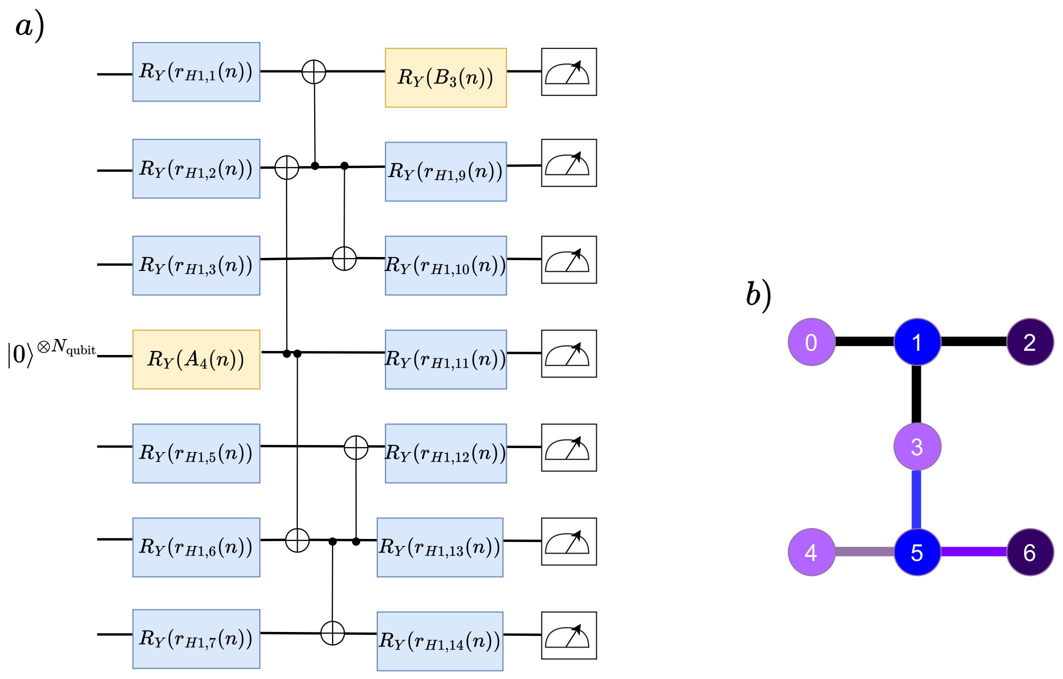

Figure 3.6: Quantum Algorithm H1's **circuit architecture** for the open-loop prediction scenario (a). The number of qubits is $N_{\text{qubit}} = 7$. The two modes A_4 and B_3 are continuously provided to the network as ground truth input. As described in Section2.2.4.2, the reservoir state $\mathbf{r}_{H1}$ and inputs are encoded as angles of rotational gates R_Y. A subset of only 14 out of the $N_r = 128$ states are fed back to the circuit to keep the circuit shallow. Moreover, $\mathbf{r}_{H1}$ evolve according to (2.50). (b) Corresponding connections of the seven qubits on the *ibm_perth* quantum computer. Note that entanglement operations, e.g., by CNOT gates, are only possible for qubits connected by the colored bars.

vice. It can be seen that the data from the noisy Qiskit simulator and the actual quantum device partly deviate from the ground truth but are found to follow the overall trend reasonably well. It can also be observed that the prediction of the actual device breaks down after about $2.1\Lambda_1^{-1}$ due to the decoherence of the superconducting qubits. Nevertheless,

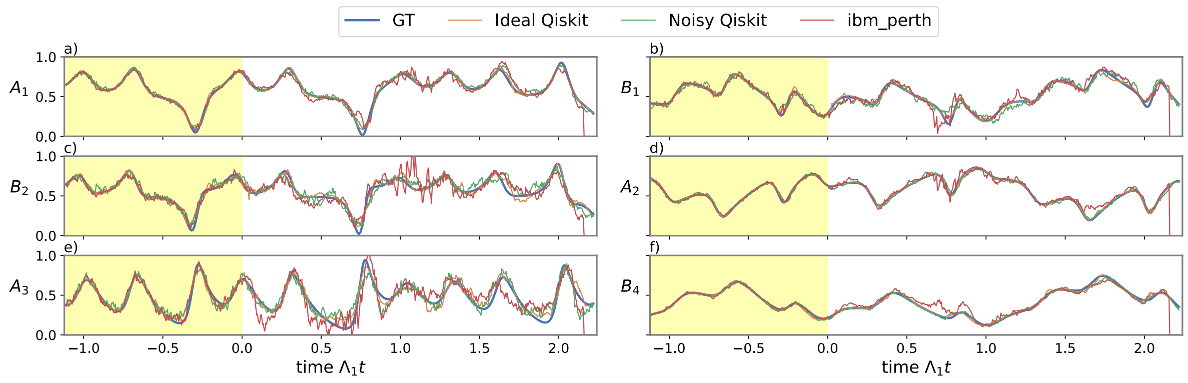

Figure 3.7: Inferred eight-order Lorenz modes of the **open-loop QRC model run** during the part of the training (yellow background) and the validation phase. The model was tasked to reconstruct the missing modes from the knowledge of only A_4 and B_3 (not shown), which were continuously provided to the reservoir. Shown are the QRC model runs performed on the actual quantum device (red), the ideal/noisy *Qiskit* simulator (orange/green), and the corresponding ground truth (blue). The leaking rate and reservoir size of all QRC were $\gamma_\mathrm{r} = 0.3$ and $N_\mathrm{r} = 128$. The runtime of a single reservoir for training and validation was about 3.3h [205].

this proves the concept of a hybrid QRCM for a classical dynamical system on a NISQ device. Finally, the performance of a classical reservoir with the same size is compared to the present QRC model. To this end, the optimal values of γ_r, ϱ_r, λ_r are identified by means of a hyperparameter search of the open-loop ESN. The details on the grid search are listed in Appendix A.2. Fig. 3.8 shows the ESN reconstructions of the Lorenz modes during the validation phase. Again, the inputs were the Lorenz modes B_3 and A_4. It can be seen that the classical RC model performs exceptionally well for this open-loop task. Despite minor inaccuracies like in mode A_3 around $1.6\Lambda_1^{-1}$, the ESN is able to reconstruct the missing information from the learned dynamics. The NRMSE of all QRC runs and the optimized ESN run, as well as their (optimal) hyperparameter set Ξ_r, are listed in Tab. 3.4. While the ESN performs best among all four runs, the two *Qiskit* simulations are still in a similar error range. The actual quantum device performs the poorest among all reservoirs. However, considering the sensitive hardware of the NISQ device, these results are a first step toward future quantum reservoir applications. In terms of hyperparameters, it can be observed that the optimal classical leaking rate value comes close to the chosen quantum value. Even though this parameter was not optimized in the quantum case, a similar value would seem natural as this hyperparameter should correlate with the characteristic time scale of the Lorenz dynamics [124]. Further, a preliminary study showed that the quantum reservoir performs best when the L2 penalty term in eq. (2.44) becomes zero, i.e., $\lambda_\mathrm{r} = 0$. This might be due to the induced noise of the quantum circuit, which acts similarly to the Tikhonov regularization. Finally, as described in Section 2.1, the spectral radius of the quantum circuit is always zero due to the unitary time evolution of the quantum state $|\psi\rangle$.

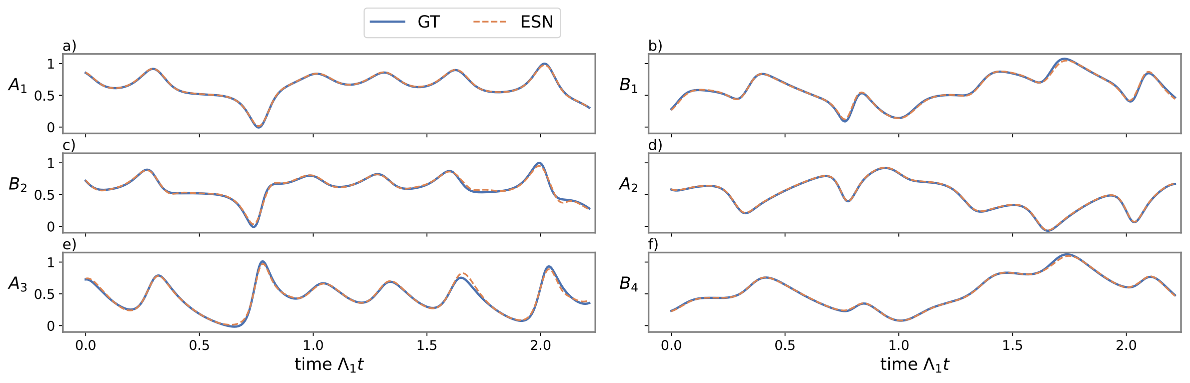

Figure 3.8: Inferred eight-order Lorenz modes of the **open-loop classical RC (ESN) run** during the validation phase. The model was tasked to reconstruct the missing modes from the knowledge of only A_4 and B_3, which were continuously provided to the reservoir. The reservoir size was the same as the QRC models $N_\mathrm{r} = 128$. The classical reservoir (orange) reproduces the GT information (blue) exceptionally well. The hyperparameters are listed in Tab. 3.3 and were optimized in a preceding grid search. See also A.2

Model	N_r	γ_r	ϱ_r	λ_r	$\mathcal{E}_\mathrm{NRMSE}$
Qiskit Ideal	128	0.3	1.0	0.0	$3.09 \cdot 10^{-2}$
Qiskit Noisy	128	0.3	1.0	0.0	$4.85 \cdot 10^{-2}$
ibm_perth	128	0.3	1.0	0.0	$1.12 \cdot 10^{-1}$
ESN	128	0.2	0.9	0.05	$1.30 \cdot 10^{-2}$

Table 3.4: **Hyperparameter** set Ξ_r and validation **normalized root-mean-square error** of the classical (ESN) and quantum Reservoir Computing model at $N_\mathrm{r} = 128$. The quantum reservoir was run with the ideal and noisy *Qiskit* simulator, as well as on the actual quantum device *ibm_perth*. Note that the hyperparameters, i.e., leaking rate and circuit architecture, of the QRC model were not optimized, while a grid search was run for the classical case. The corresponding hyperparameter landscape is shown in Appendix A.2.

3.3.2 Quantum Reservoir Computing of a two-dimensional Rayleigh-Bénard flow

The application of the presented Quantum Reservoir Computing model can be extended toward more complex classical dynamical systems. Starting with the above presented Lorenz model, the complexity of the task to be learned can be increased by proceeding to a turbulent convection flow at the same geometry and Prandtl number as in the Lorenz case but at a significantly higher Rayleigh number. To this end, direct numerical simulations of Rayleigh-Bénard convection are conducted. Since the phase space of an actual Rayleigh-Bénard system is high-dimensional, the QRC model will be combined with a data reduction method, namely the POD. The resulting model will subsequently be used to test the QRC's ability in the open-loop scenario to reproduce low-order statistics of the RBC flow while having only partial information on the reduced representation of the convective flow.

3.3.2.1 Direct numerical simulation of Rayleigh-Bénard convection

To this end, the two-dimensional Boussinesq system (2.17 - 2.19) is numerically solved using the *Nek5000* spectral element solver [180]. The aspect ratio of the two-dimensional system is $\Gamma = 2\sqrt{2}$, while the Prandtl and Rayleigh numbers are $\mathrm{Pr} = 10$ and $\mathrm{Ra} = 10^5$. Dirichlet boundary conditions are imposed for the temperature field at the top and bottom. Furthermore, free-slip boundary conditions in the vertical z-direction are applied for the velocity field. Periodic boundaries for all fields are taken in the horizontal x–direction. The chosen boundary conditions, aspect ratio and Prandtl number correspond to the Lorenz model that was used in 3.2 and 3.3.1, but at a higher Ra and thus fully turbulent in contrast to the previous sections. Table 3.5 lists the DNS parameters, and Fig. 3.9 compares the reconstructed

Γ	Ra	Pr	N_e	N	$N_x \times N_z$	t_{DNS}/t_f	$\Delta t_{\mathrm{DNS}}/t_f$	Re	τ_{eddy}/t_f	Nu
$2\sqrt{2}$	10^5	10	32×8	11	128×32	2500	$5 \cdot 10^{-4}$	19.56	5.11	8.97

Table 3.5: **Simulation parameters** of the direct numerical simulation of Rayleigh-Bénard convection. Listed are the aspect ratio Γ of the two-dimensional cell, the Rayleigh number Ra, the Prandtl number Pr, as well as the total number of spectral elements N_e and polynomial order N on each element. Finally, the DNS data is interpolated to a uniform grid of size $N_x \times N_z$. t_{DNS} and Δt_{DNS} denote the time for how long the DNS was run and the corresponding time step size, respectively. The Reynolds number Re, as defined in (2.23), largest eddy turnover time τ_{eddy}, and Nusselt number Nu, as defined in (2.24), are diagnostic parameters obtained from the DNS.

eight-order Lorenz fields to the ones obtained by the present DNS. In both cases, two convection rolls form. Further, it can be seen that the DNS case exhibits finer and more complex thermal plumes. Moreover, a thermal boundary layer at the top and bottom plate can be identified in the DNS case. For the purpose of the ML application, the DNS data is subsequently interpolated to a uniform grid of size $N_x \times N_z = 128 \times 32$ and sampled in a time interval of $0.25t_f$ in the statistically steady regime.

3.3.2.2 Dimensionality reduction via POD

As explained in Section 2.3, the utilization of interpolated snapshots of u_x, u_z, and θ as inputs for the reservoir would result in substantial computational expenses, owing to the

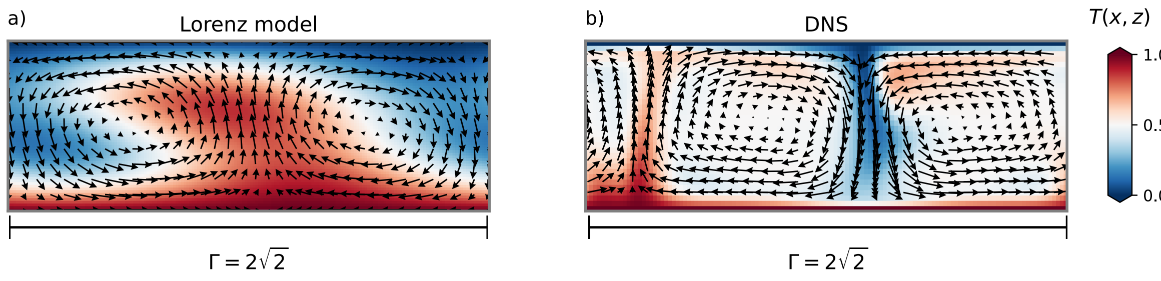

Figure 3.9: **Comparison** between a snapshot of the temperature field $T(x, z) = 1-z+\theta(x, z)$ based on (a) the reconstructed eighth order Lorenz model at Ra $= 1.84 \cdot 10^4$ ($r = 28$) (employed in the last two sections) and (b) direct numerical simulations of RBC at Ra $= 10^5$. The vector arrows indicate the superimposed velocity field $u_x(x, z)$ and $u_z(x, z)$. The aspect ratio and Prandtl number are the same in both cases. In both cases, two convection rolls develop. Note the much richer features in the DNS case.

need for a significantly enlarged reservoir dimension. To circumvent this bottleneck, a pre-processing step is introduced, in which the most prominent features of the physical fields are extracted, and the dynamics of the convection flow are translated to dynamics of temporal coefficients $\mathbf{a}(t)$. For this purpose, a Proper Orthogonal Decomposition is applied to the fluctuation fields u'_x, u'_z, and θ', which are defined by the following decomposition

$$u_x(x, z, t) = \langle u_x\rangle_t(x, z) + u'_x(x, z, t), \tag{3.12}$$
$$u_z(x, z, t) = \langle u_z\rangle_t(x, z) + u'_x(x, z, t), \tag{3.13}$$
$$\theta(x, z, t) = \langle \theta\rangle_t(x, z) + \theta'(x, z, t). \tag{3.14}$$

Fig. 3.10a) shows the cumulative spectrum of the captured variance for the present DNS case. It can be observed that already 16 modes are enough to capture more than 86% of the variance of all three physical fields. Figs. 3.10 b) and c) show the effect of the truncation to the first 16 modes. The main spatial features of the heat flux $u'_z\theta'$ are still retained, and its vertical profile, despite a decreased value in the bulk, follows the DNS trend. A subset of the N_{POD} time coefficient dynamics $\mathbf{a}(t)$ are shown in Fig. 3.11. While the first modes a_1, a_2 exhibit changes over large time scales, higher mode numbers, e.g., a_{10}, a_{11}, a_{16}, capture higher temporal frequencies. This separation of scales among the time coefficients also translates to the spatial modes Φ, shown in Fig. 3.12. It can be observed that Φ_1 captures large-scale structures like the thermal boundary layer in Fig. 3.12a) or up- and downdrafts in Fig. 3.12g). Modes of higher degree on the other hand, incorporate finer structures, as can be seen in Fig. 3.12 c, f, i). well.

In the next step, a QRC model will be trained on the dynamics of the reduced-order POD model. For this, two algorithms will be considered, namely H1, already applied in the last section, and a novel algorithm, H2. See again 2.2.4.2 for their definition. Contrary to the QRC model in 3.3.1, in the present scenario, the quantum state will be simulated using the *Qiskit Statevector Simulator* for both H1 and H2 [204]. This way, the high computational cost of approximating the necessary probabilities $\mathbf{p}$ by repeated measurements is circumvented. However, no actual quantum device will be employed in this study. Finally, for comparison, a classical reservoir computer in terms of an ESN will be considered again.

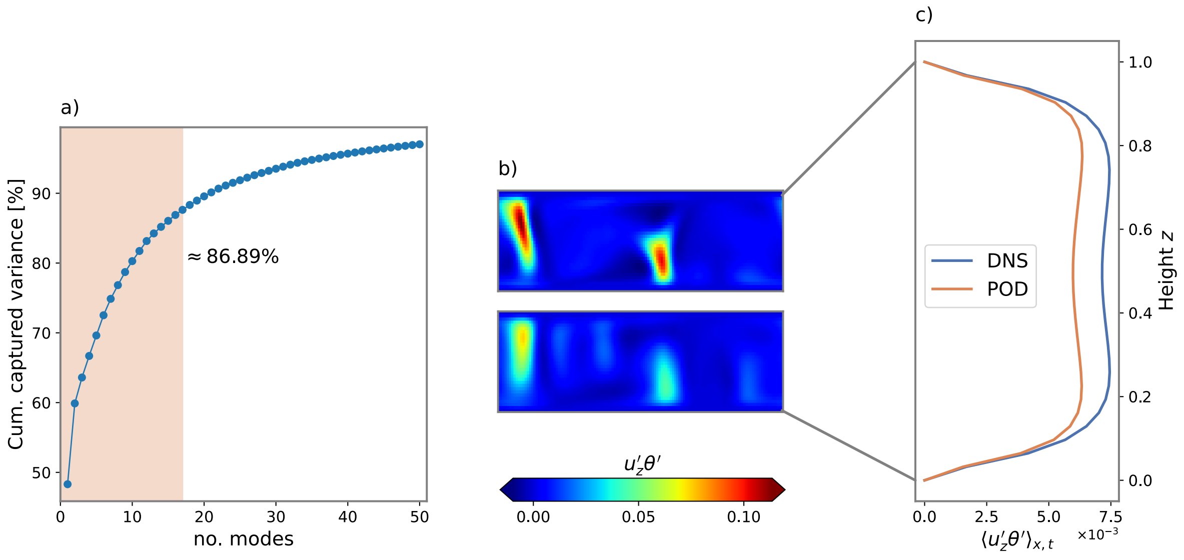

Figure 3.10: **POD model** of the Rayleigh-Bénard flow listed in Table 3.5. (a) The cumulative spectrum of captured variance. The shaded area makes the first $N_{\mathrm{POD}} = 16$ modes. (b) Comparison between DNS heat flux (top) and the reconstructed heat flux based on the first 16 modes. (c) Corresponding vertical profiles of $\langle u_z'\theta'\rangle_{x,t}$.

3.3.2.3 POD-QRC model for two-dimensional Rayleigh-Bénard convection

To this end, the RC models are again employed in the open-loop mode. They receive the first three POD coefficients as input, i.e., $\mathbf{x}(n) = (a_1(n), a_2(n), a_3(n))^T$, and are trained to output the whole 16 variables. This way, the reservoir acts similarly to a subgrid-scale parameterization, where the small-scale dynamics are computed based on the knowledge of the large-scale state. Because the quantum state $|\psi\rangle$ is being simulated, there is no need for computationally intensive approximations of probabilities through multiple measurements. This allows for a hyperparameter grid search to be conducted in order to determine the best-performing hyperparameter combination of the quantum reservoir. Details on the procedure are listed in Tab. 3.6. Note that the procedure for the quantum algorithms only considers the leaking rate γ_{r} and reservoir size N_{r}, i.e., the number of qubits N_{qubit}, as the spectral radius has no quantum counterpart and a nonzero regression parameter showed no improvement of the QRC model's performance. Moreover, the testing phase is again omitted to avoid additional computational costs. The quantum circuit architectures of both H1 and H2 can be found in Appendix A.3. To assess the performance of the reservoirs in terms of reconstructing the convective flow, the normalized average relative error $\mathcal{E}_{\mathrm{NARE}}$, as defined in eq. (2.72), is considered. Specifically, $\mathcal{E}_{\mathrm{NARE}}\left(\langle u_z'\theta'\rangle_{x,t}\right)$, i.e., the NARE of the inferred convective heat flux $u_z'\theta'$ will be used. Remember that the NARE measures the deviations between the vertical profiles $\langle\cdot\rangle_{x,t}$ of inferred and ground truth fields. As this measure is prone to the errors of two fields, it is a suited measure of the accordance of the inferred convection flow. For this, the inferred 16 POD time coefficients are used to reconstruct the three physical fields $u_x'(x,z)$, $u_z'(x,z)$, and $\theta'(x,z)$. Finally, the vertical properties of the reconstructed heat flux $u_z'\theta'$ can be compared to ground truth, here the POD model with 16 modes. The hyperparameter dependencies of $\mathcal{E}_{\mathrm{NARE}}$ w.r.t. γ_{r} and N_{r} can be seen in 3.13 for all three reservoirs. The NARE values correspond to the median $\mathcal{E}_{\mathrm{NARE}}$ value among all

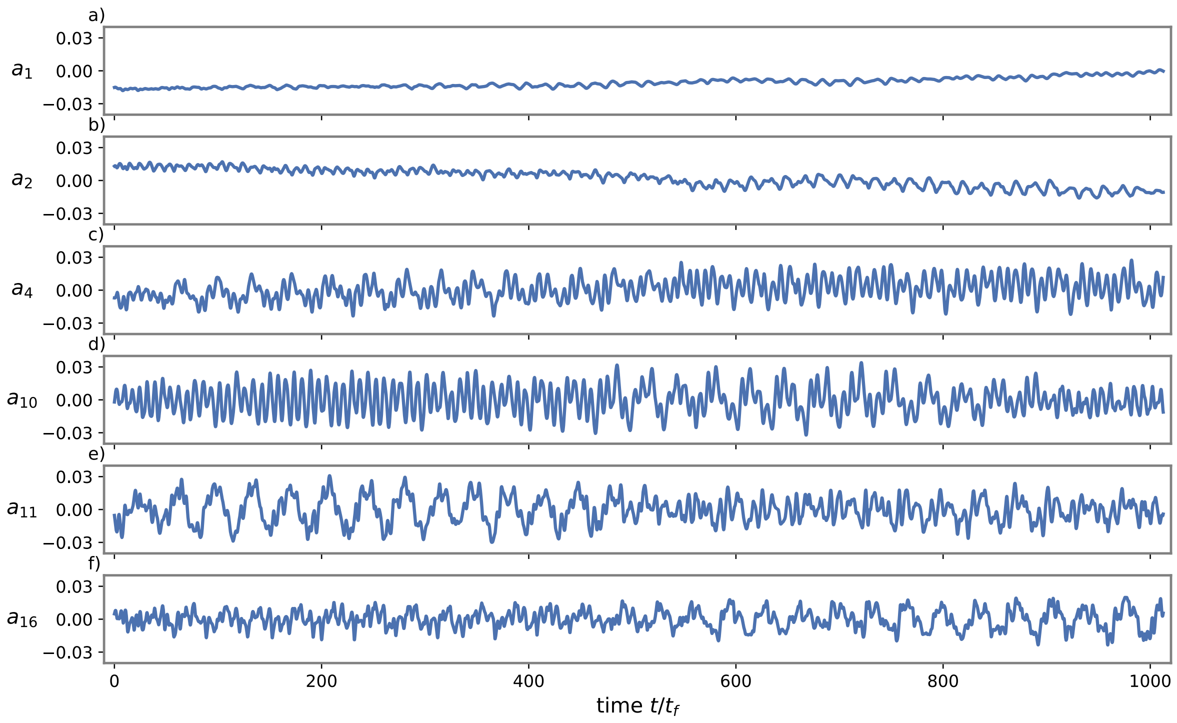

Figure 3.11: **POD time coefficients** $a_i(t)$ for modes $i = 1, 2, 4, 10, 11, 16$. The time coefficients determine the temporal behavior of the associated spatial modes $\Phi_i(x, z)$ in Fig 3.12. Note the increasing range of frequencies for higher mode numbers. This scale separation is also observed for spatial structures in the POD spatial modes. Moreover, similar temporal features can be observed in the Lorenz model. Compare e.g. $a_{10}(t)$ with mode $A_1(t)$ in Fig. 3.2c).

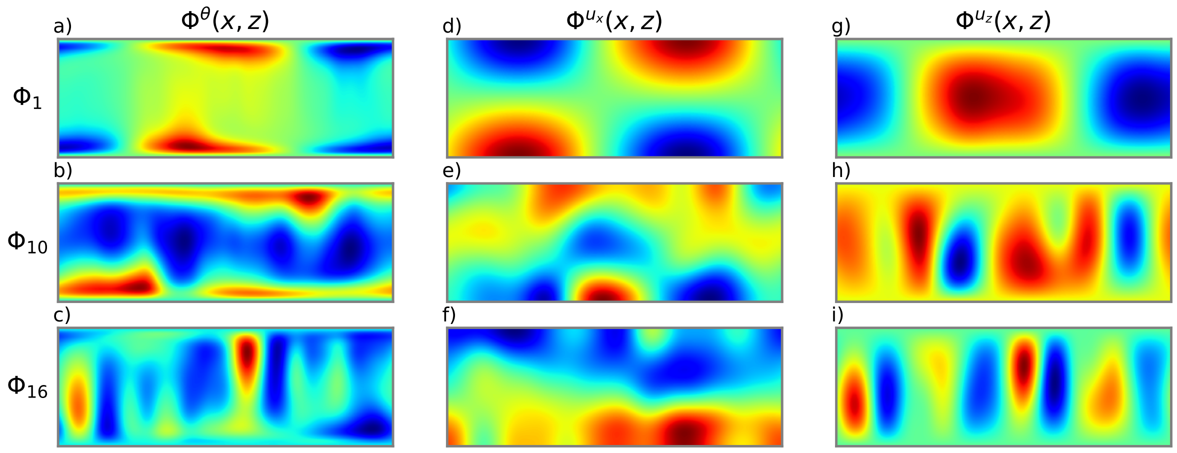

Figure 3.12: **POD spatial modes** of (a-c) the temperature fluctuations Φ^θ, (d-f) horizontal velocity Φ^{u_x}, and (g-i) vertical velocity Φ^{u_z}. Lower mode numbers are associated with large-scale features, while higher modes represent finer structures. Corresponding temporal coefficients are shown in Fig. 3.11.

	$\gamma_{\rm r}$	$\varrho_{\rm r}$	$\lambda_{\rm r}$	N_r
ESN	$[0.1, 1.0, 0.1]$	$[0.1, 1.5, 0.1]$	$\{0.05, 0.5\}$	$\{64, 128, 256, 512, 1024, 2048\}$
H1	$[0.0, 1.0, 0.05]$	–	–	$\{64, 128, 256, 512, 1024, 2048\}$
H2	$[0.0, 1.0, 0.05]$	–	–	$\{64, 128, 256, 512, 1024, 2048\}$

	Reservoir realizations	$t_{\rm train}/t_f$	$t_{\rm val}/t_f$
ESN	100	1250	125
H1	10	1250	125
H2	10	1250	125

Table 3.6: **Hyperparameter grid search** procedure for classical ESN and quantum reservoir algorithms H1, H2. Square brackets indicate the range of the searched hyperparameter: [start, end, stepsize]. Braces indicate an array of values that were used. $t_{\rm train}, t_{\rm val}, t_{\rm test}$ are the length of the training, validation, and testing interval, respectively. For the ESN, the reservoir density and input scaling were fixed at $D_{\rm r} = 0.2$ and $\sigma_{\rm r} = 1$. The total number of different ESN settings is $10 \times 15 \times 2 \times 6 = 1800$, with 100 reservoir realizations. The search for the QRC algorithms was limited to $10 \times 6 = 60$ combinations, with a total of 10 instances being run due to the still exhaustive simulation time. Also, no testing phase is used for the same reason. The grid search run times were 18.6h (ESN), 166h (H1, H2 each) [206] on the TU Ilmenau high-performance-computing cluster. The corresponding quantum circuits for H1 and H2 are shown in Appendix A.3.

reservoir realizations, while all other parameters are pre-optimized, that is, the optimal spectral radius $\varrho_{\rm r}$ and the Tikhonov parameter $\lambda_{\rm r}$ in the classical RC model case, the number of layers l (see Appendix A.3 for an explanation) in the hybrid quantum-classical cases. It can be observed that H1 and H2 seem to outperform the classical approach for qubit numbers $N_{\rm qubit} < 10$, i.e., $N_{\rm r} < 1024$. The global optimum, i.e., the minimal amplitudes of $E_{\rm NARE}\left(\langle u'_z\theta'\rangle_{x,t}\right)$, is obtained for the new architecture H2 at $N_{\rm qubit} = 9$, i.e., $N_{\rm r} = 512$, though the other RC models can perform similarly well if the reservoir is large enough. All three optimal reservoir settings $\Xi_{\rm r}$ are listed in Tab. 3.7. Finally, the reservoir outputs for

Model	$N_{\rm r}$	$\gamma_{\rm r}$	$\varrho_{\rm r}$	$\lambda_{\rm r}$	l	$\mathcal{E}_{\rm NARE}\,(u'_z\theta')$
H1	256	0.5	0.0	0.0	5	$7.46 \cdot 10^{-3}$
H2	512	0.3	0.0	0.0	5	$6.40 \cdot 10^{-3}$
ESN	2048	0.8	1.4	0.5	–	$8.07 \cdot 10^{-3}$

Table 3.7: **Hyperparameter** set $\Xi_{\rm r}$ and validation **NARE** of the classical (ESN) and quantum Reservoir Computing models H1 and H2. The quantum reservoirs were run using the ideal *Qiskit Statevector Simulator*. Hence, no multiple measurements of the final quantum state were necessary. Note that the spectral radius of the quantum reservoir is always unity due to the unitary quantum state evolution. Moreover, the quantum circuit depth l has no classical counterpart. See also the quantum circuit architecture in Appendix A.3.

the optimal setting are shown in Figure 3.14a). It can be seen that all models perform fairly well. Even though the ground truth trajectories are not fully reproduced, the overall trends are represented well. Hence, the reconstructed low-order statistics of the vertical profiles of

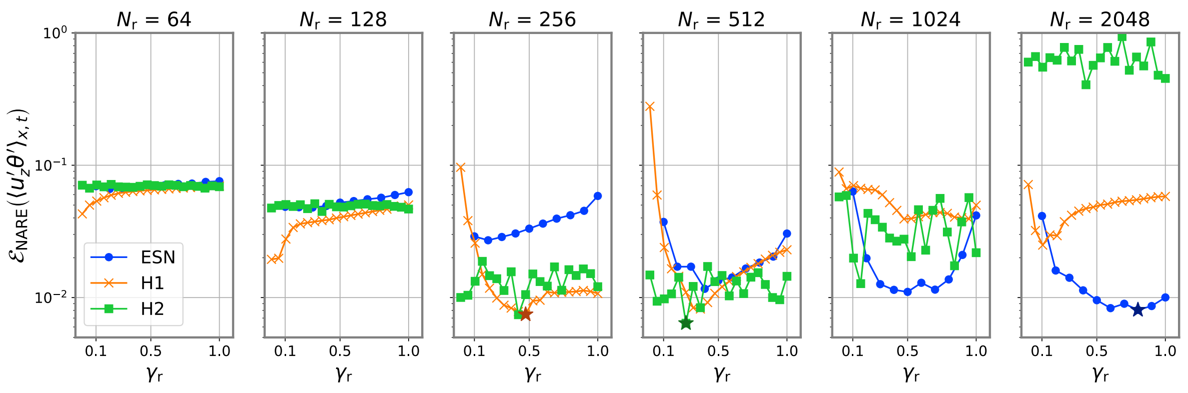

Figure 3.13: **Normalized average relative error** $\mathcal{E}_{\mathrm{NARE}}\,(\langle u_z'\theta'\rangle_{x,t})$ for the convective heat flux of quantum reservoirs H1 (orange), H2 (green) and the classical reservoir (blue) for different reservoir sizes and leaking rate values. The reservoir sizes correspond to $N_{\mathrm{qubit}} = 6, 7, 8, 9, 10, 11$. Displayed are the median values for 10 seeds with H1 and H2 and 100 seeds for the ESN. The optimum of each curve is the single-best median for the respective approach at the given reservoir size. A star marker in the respective color marks the minimum NARE value of each algorithm.

the convective heat flux and root-mean-square fluctuations of all physical fields are also in good accordance with the reference profiles. These are shown in 3.14b) - e). Even though the predictions were chosen according to the profile in 3.14e), the other lateral-time averages are also perfectly reproduced. Hence, the information on three POD time coefficients is sufficient for all reservoirs to reproduce the vertical properties of the turbulent flow.

3.4 Conclusions from this chapter

This chapter introduced a low-order model of thermal convection with eight degrees of freedom that extended the model from Lorenz from 1963 by shear and conservation of vorticity. The reduced representation made the system an ideal testing bed for understanding how an Echo State Network responds to chaotic time series data. It was shown that an ESN can autoregressively infer the dynamics of the LOM. While an exact reproduction of the chaotic time series is beyond the grasp of this machine learning model, it was found that the reservoir is able to generate similar dynamics to the ground truth signal and even reproduces the chaotic Lorenz attractor. A second operation mode, the open-loop scenario, was used to compare an ESN against two novel Quantum Reservoir Computing models. In this scenario, the reservoir receives few input signals and, based on this knowledge, is tasked to infer the missing degrees of freedom. Both QRC models show comparable performances to the classical ESN. Furthermore, a quantum reservoir was run on a real seven-qubit quantum computer, which showed impressive first results, indicating new pathways for RC models on quantum devices. Moreover, the first results on a DNS of Rayleigh-Bénard convection indicate that quantum algorithms are able to reproduce low-order statistics of RBC at Ra $= 10^5$. However, the latter results were obtained by means of a state vector simulation. A real NISQ device would have to take samples to approximate the underlying distribution of the quantum state. Moreover, in [158], Pfeffer *et al.* indicate that such a practical implementation would require sampling of the order 2^{20}, which dampens the prospects for an ap-

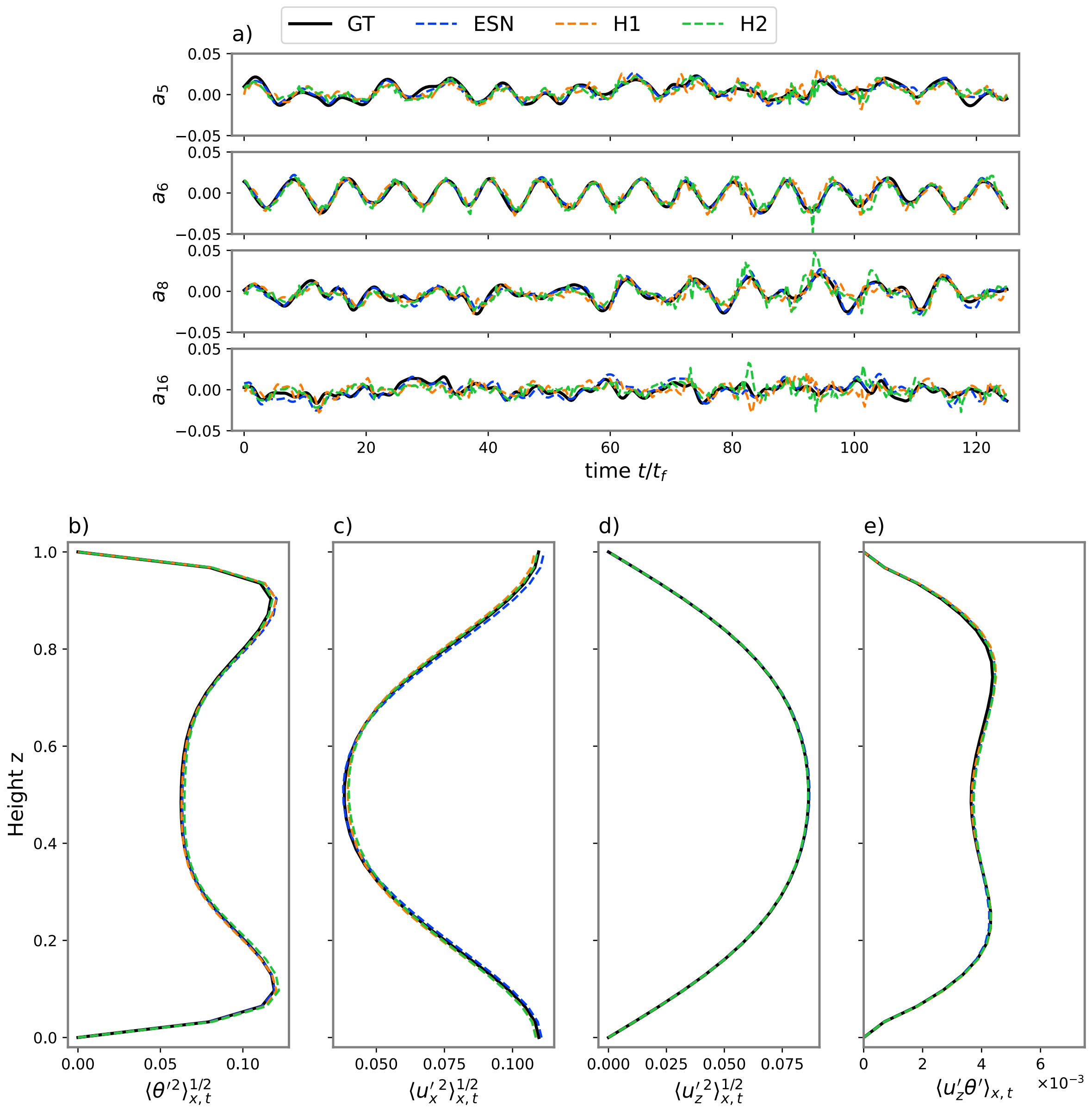

Figure 3.14: Open-loop **Reservoir Computing results** for the 16 mode POD model. (a) Inferred POD time coefficients by means of the classical ESN (blue) and quantum reservoirs H1 (orange) and H2 (green). The ground truth is shown in black. (b-e) Reconstructed vertical profiles $\langle \cdot \rangle_{x,t}$ based on the 16 POD coefficients. Shown are the root-mean-square profiles of fluctuations of temperature θ' (b), horizontal (c), and vertical velocities (d) u_x' and u_z', as well as the convective heat flux $u_z'\theta'$ (e). Note that the reservoirs received the first three coefficients as input and were tasked to reconstruct the whole 16 POD variables.

plication on current noisy intermediate-scale quantum devices. However, a repetition of the hyperparameter grid search with sample-based probabilities and the additional implementation of weak measurements might ease this problem [207]. Furthermore, it has to be evaluated if the hybrid quantum-classical Reservoir Computing approach can be further scaled up to flows at higher Rayleigh numbers and hence more turbulent flows.

The next chapter will pose a first step towards an ML-based super-parameterization, as an autoregressive ESN will be applied to two-dimensional turbulent Rayleigh-Bénard convection DNS data. Further, the spatial domain of the flow will be more than double the size considered in this chapter. Moreover, the flow will be significantly more turbulent as the Rayleigh number is increased to $\mathrm{Ra} = 10^7$ and the Prandtl number is decreased to $\mathrm{Pr} = 7$.

Chapter 4

Reservoir Computing of two-dimensional Rayleigh-Bénard convection

As discussed in the introduction, this work aims at evaluating Reservoir Computing models for use in potential super-parameterization applications. For the latter, properties like the vertical distribution of heat and moisture are essential features determining the interaction between large-scale circulation and the local two-dimensional cloud-resolving model. Hence, prior to replacing a subgrid-scale model with a trained ROM-RC model, it is essential to verify its capability to produce the spatial and statistical features that are distinctive of thermal convection. While the last chapter described how reservoir computers perform on the chaotic dynamics of eight and 16 degrees of freedom, the present chapter will evaluate the ESN performance on DNS data of two turbulent Rayleigh-Bénard systems. Naturally, as the complexity of the convection system increases, the degrees of freedom the reservoir has to learn will grow as well. In Section 4.1, the ability of the reservoir to reproduce two-dimensional RBC features will be tested. After that, in Section 4.2, a four times bigger cell at a higher state of turbulence will be considered. Furthermore, the complexity of the problem will be increased by the inclusion of water vapor that can undergo condensation, resulting in the formation of clouds. The results of this chapter were published in [117].

4.1 Two-dimensional Rayleigh-Bénard convection

This section will introduce the first application of an ESN, run in the closed-loop mode, to a DNS of Rayleigh-Bénard convection. In this way, the reservoir will autoregressively infer the turbulent flow without the need for an external input. In pursuit of this objective, the numerous degrees of freedom within the turbulent flow need to be condensed to a manageable extent. To accomplish this, the POD algorithm will be once again utilized. Owing to the much more intricate nature of the flow compared to the one examined in Section 3.3.2, the number of input features for the ESN will notably increase. It is worth noting that this section investigates a similar flow configuration to that explored by Pandey and Schumacher in ref. [116], although their work lacked an exhaustive grid search procedure and an appropriate number of reservoir realizations. This section will conduct a more thorough investigation into these matters.

4.1.1 Direct numerical simulation of Rayleigh-Bénard convection

A DNS of two-dimensional RBC at $\mathrm{Pr} = 7$ and $\mathrm{Ra} = 10^7$ was conducted. Note that the aspect ratio is more than two times larger than in Section 3.3.2, namely $\Gamma = 6$. The boundary conditions at the bottom and top lid were chosen to be no-slip for the velocity and Dirichlet for the temperature field. Again, periodic boundaries are considered in the horizontal x-direction. Further DNS parameters like the number of spectral elements and polynomial order are listed in Tab. 4.1. In a second step, the non-uniform DNS grid is interpolated to a uniform grid of size $N_x \times N_z = 256 \times 64$ for further use. An instantaneous snapshot of the temperature and velocity fields can be seen in Fig. 4.1. Two convection rolls with complex shapes characterized by the presence of finely formed thermal plumes can be observed. In the next step, this database will be reduced by means of a POD analysis.

Γ	Ra	Pr	N_e	N	$N_x \times N_z$	t_{DNS}/t_f	$\Delta t_{\text{DNS}}/t_f$	Re	τ_{eddy}/t_f	Nu
6	10^7	7	48×8	11	256×64	2500	10^{-3}	156.25	7.65	13.43

Table 4.1: **DNS parameters** of the simulation of two-dimesnional Rayleigh-Bénard convection. Listed are the aspect ratio Γ of the two-dimensional cell, the Rayleigh number Ra, the Prandtl number Pr, as well as the total number of spectral elements N_e and polynomial order N on each element. The DNS data is interpolated to the uniform grid of size $N_x \times N_z$. t_{DNS} and Δt_{DNS} denote the time for how long the DNS was run and the time stepping, respectively. The Reynolds number Re, largest eddy turnover time τ_{eddy}, and Nusselt number Nu are diagnostic parameters obtained from the DNS. The time sampling of the saved snapshots was $0.25t_f$.

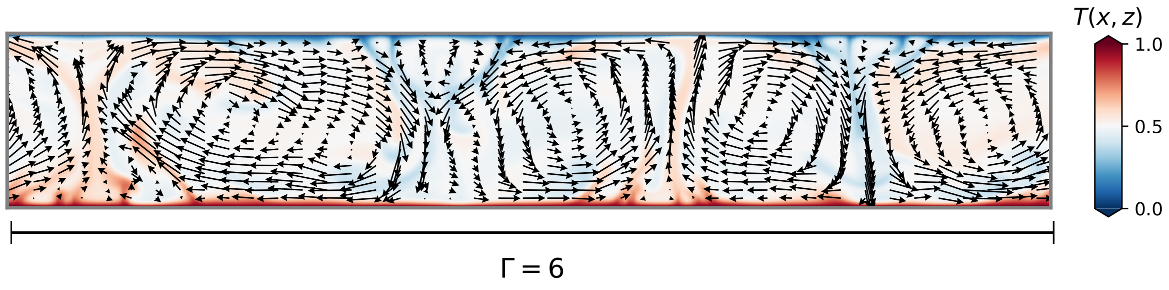

Figure 4.1: **DNS snapshot** of the temperature field $T(x, z) = 1 - z + \theta(x, z)$ of the two-dimensional Rayleigh-Bénard flow at Ra $= 10^7$ and Pr $= 7$. The aspect ratio is $\Gamma = 6$. The velocity vector fields $u_x(x, z)$, $u_z(x, z)$ are superimposed. The flow features complex characteristics like fine thermal plumes and turbulent fluid motion. The simulation parameters are listed in Table 4.1.

4.1.2 Dimensionality reduction via POD

As discussed in 2.3, this flow's number of degrees of freedom, namely $N_x \times N_z$, is too large to be used as reservoir input. At this degree of turbulence, it can be expected that more than 16 modes, as used in the previous chapter, are required to capture enough of the turbulent kinetic energy and thermal variance of the convective flow. To this end, the interpolated DNS data of $(u_x, u_z, \theta)^T$ is collected into a snapshot matrix. Then the fields are decomposed into their time mean $\langle \cdot \rangle_t$ and the fluctuations about it

$$u_x(x, z, t) = \langle u_x(x, z) \rangle_t + u'_x(x, z, t) \tag{4.1}$$

$$u_z(x, z, t) = \langle u_z(x, z) \rangle_t + u'_z(x, z, t) \tag{4.2}$$

$$\theta(x, z, t) = \langle \theta(x, z) \rangle_t + \theta'(x, z, t). \tag{4.3}$$

Here, the prime denotes the fluctuations. Therefore a POD snapshot matrix of $(u'_x, u'_z, \theta')^T$ is constructed and the eigenvalue problem (2.65) of its covariance matrix is solved. The resulting spatial modes $\Phi_i^\theta(x,z), \Phi_i^{u_x}(x,z), \Phi_i^{u_z}(x,z)$ and their temporal coefficients $a_i(t)$ are shown in figures 4.2 and 4.3 respectively.

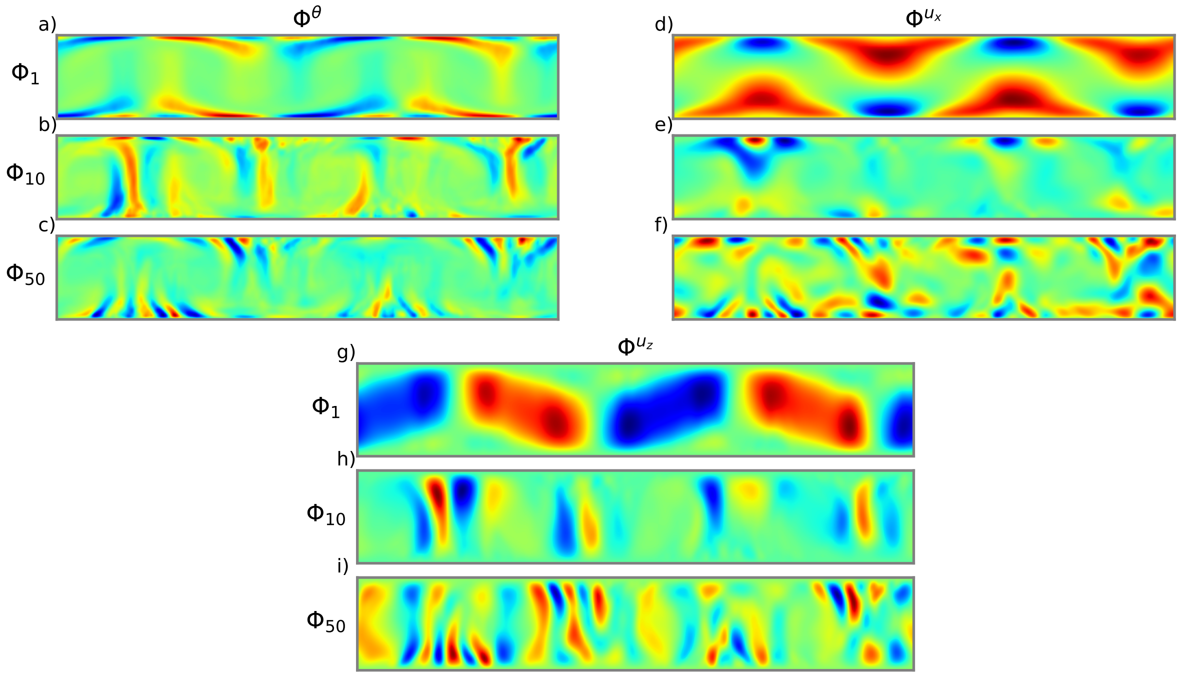

Figure 4.2: **POD spatial modes** Φ^θ of the temperature fluctuations θ' (a-c), Φ^{u_x} of the horizontal velocity fluctuations u_x' (d-f) and Φ^{u_z} of the vertical velocity u_z' (g-i). Note that lower mode numbers are associated with large-scale features, while higher modes represent finer structures. The corresponding time coefficients are illustrated in Fig. 4.3.

While the first mode Φ_1 represents large-scale features like the thermal boundary layer or main up- and downdrafts, Φ_{50} describes their finer spatial characteristics. This scale separation translates to the temporal coefficients: while coefficients with a low mode degree vary slowly over time, high mode number coefficients exhibit fast, chaotic dynamics with a wide range of frequencies. Hence, again, the POD decomposes the flow into slowly evolving large-scale and quickly varying small-scale structures. Moreover, the cumulative spectrum of the captured variance is shown in Fig. 4.4a). It can be seen that the spectrum of the turbulent flow rapidly increases, and after 150 modes, already 82% of the original variance is retained. Fig. 4.4b) compares a DNS snapshot of the heat flux with the corresponding reconstruction from the most energetic 150 POD modes. It can be observed that this reduced representation suffices in capturing the main spatial features like the locations of enhanced heat flux. However, it results in a reduced magnitude corresponding to the discarded variance of the missing POD modes. Finally, in Fig. 4.4c) the corresponding lateral-time average profiles of the convective heat flux $\langle u_z'\theta'\rangle_{x,t}$ are presented. Again, it can be seen that the ROM retains the main aspects of the heat flux profile. Therefore, this section will consider the first $N_{\mathrm{POD}} = 150$ POD time coefficients $a_1(t), ..., a_{N_{\mathrm{POD}}}(t)$ for training and evaluating the performance of an Echo State Network used in the closed-loop mode.

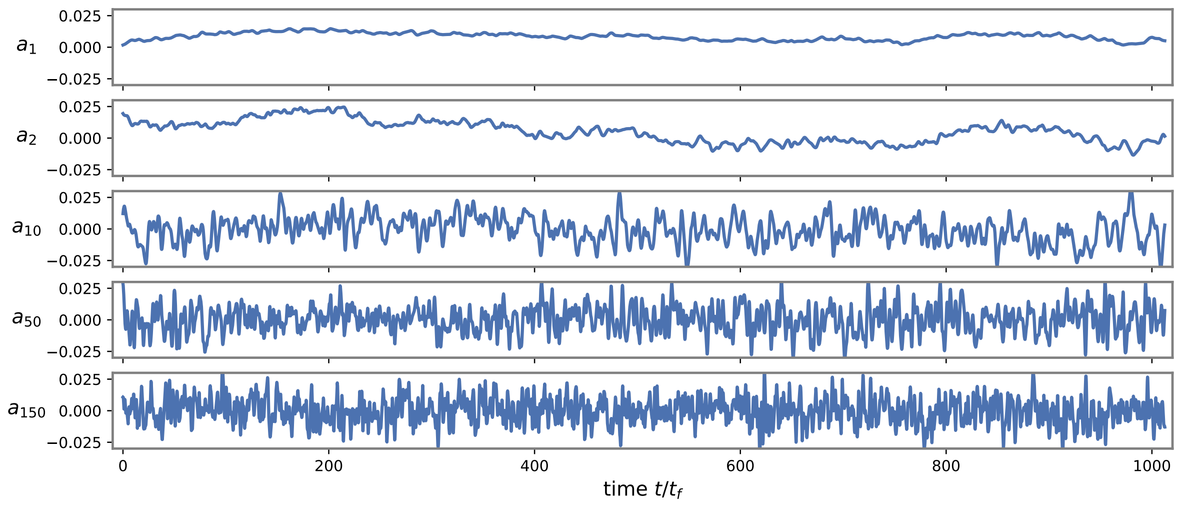

Figure 4.3: **POD time coefficients** $a_i(t)$ for modes $i = 1, 2, 3, 10, 50, 150$. The time coefficients determine the temporal behavior of the associated spatial modes $\Phi_i(x, z)$ in Fig 4.2. It is these temporal signals that the ESN is trained to reproduce. Observe the expanding spectrum of frequencies associated with higher-order modes. This scale separation is also observed for spatial structures in the POD spatial modes.

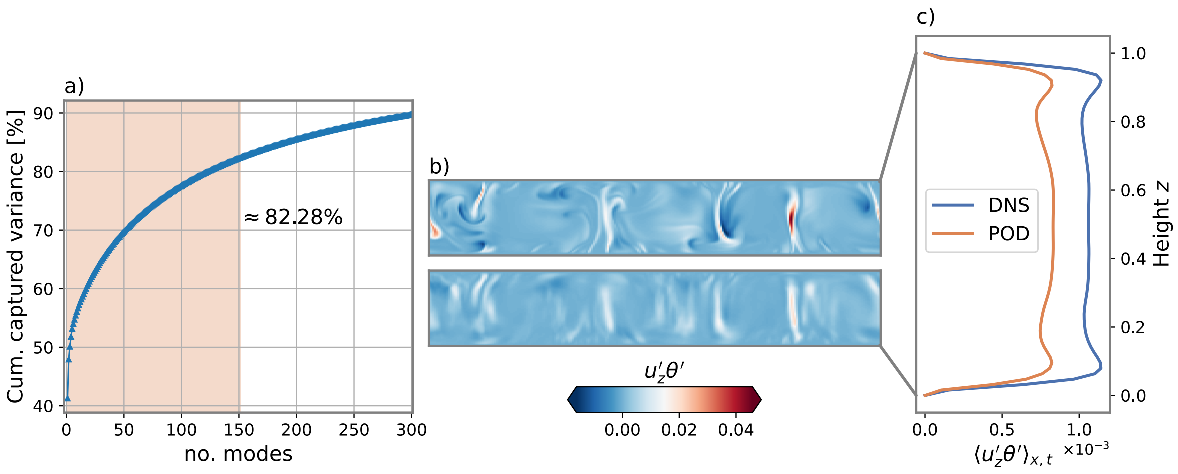

Figure 4.4: (a) **POD spectrum:** The orange-shaded region marks the first 150 modes used for further use for the Reservoir Computing model. (b) Comparison between a DNS snapshot of the heat flux $u_z'\theta'$ (top) and the reconstruction from the first 150 modes (bottom). It can be seen that the truncated POD captures the main spatial patterns. (c) Vertical profiles of the lateral-time average profiles of the heat flux $\langle u_z'\theta' \rangle_{x,t}$. The orange curve stems from the reconstructed fields of the first 150 POD modes. The full DNS profile is shown in blue for comparison. The POD model captures the trend of the profile. However, it results in an overall reduced value of the heat flux due to the loss of information by the compression algorithm.

4.1.3 POD-RC model for two-dimensional Rayleigh-Bénard convection

4.1.3.1 ESN hyperparameter search

The time-discretized coefficients $\mathbf{a}(n) = (a_1(n), a_2(n), ..., a_{N_{\mathrm{POD}}}(n))^T$ are sampled over a time interval of $\Delta t = 0.25t_f$. They are then partitioned into training, validation, and testing subsets. The training subset is employed to train the ESN, the validation subset is utilized to optimize its hyperparameter set Ξ_{r}, and the testing subset is used to evaluate the ESN's ultimate performance. The data for each of the three subsets were derived from consecutive RBC data within the statistically steady state. As a result, the dynamics across these subsets display comparable temporal characteristics. The optimization of Ξ_{r} is again done by means of a grid search procedure. The chosen hyperparameters are leaking rate, spectral radius, regression parameter, and reservoir size. The latter should ideally exceed the input dimension, here N_{POD}, by at least one order of magnitude (see Section 2.2.2 again) so that it will be restricted to the values $1024, 2048$ only. More details on the hyperparameter search can be found in Table 4.2.

γ_{r}	ϱ_{r}	λ_r	N_{r}
$[0.1, 1.0, 0.1]$	$[0, 1.0, 0.11]$	$\{3.47, 7.40, 15.79, 33.71, 71.95\}$	$\{1024, 2048\}$

Reservoir realizations	t_{train}/t_f	t_{val}/t_f	t_{test}/t_f
100	1000	500	500

Table 4.2: **ESN hyperparameter grid search** procedure for an ESN run in closed-loop mode. Square brackets indicate the range of the searched hyperparameter: [start, end, stepsize]. Braces indicate an array of values that were used. t_{train}, t_{val}, t_{test} are the length of the training, validation, and testing interval, respectively. The first $12.5t_f$ out of the training time was used for initializing the internal reservoir state. The reservoir density and input scaling were fixed at $D_{\mathrm{r}} = 0.2$ and $\sigma_{\mathrm{r}} = 1.0$. The total number of different reservoir settings is $10 \times 10 \times 5 \times 2 = 1000$, while a total of $100 \times 1000 = 10^5$ reservoirs were run. The total runtime was 38.75h using the *turbESN* library [123].

In the present closed-loop scenario, the ESN is trained to predict $\mathbf{a}(n+1)$ based on the input $\mathbf{x}(n) = \mathbf{a}(n)$. Moreover, the training length was $1000t_f$, corresponding to about 130 turnovers of the largest eddy. Unlike the Lorenz system with only eight degrees of freedom or the 16 mode POD model of RBC, the ESN must now process 150 variables with highly turbulent dynamics. Moreover, the ESN is now run in an autoregressive fashion, where no ground truth input can stabilize the ESN predictions. Similarly to the closed-loop results in Section 3.2, it is not expected that the ESN reproduces the exact time series as shown in Fig. 4.3 due to the chaotic nature of the underlying dynamics. Instead, the anticipation is for the deduced dynamics to represent realizations of the same flow configuration, ideally exhibiting consistent statistical characteristics. This has implications for how one measures the ESN performance in the post-processing of the hyperparameter search. The NRMSE is suited for tasks where one is interested in two trajectories that stay close to each other. Here, however, the ESN is supposed to mimic the statistical properties of the reduced-order RBC system. Furthermore, recall that the vertical profiles $\langle \cdot (z) \rangle_{x,t}$ are used for communication between small-scale and large-scale simulations in the super-parameterization approach. Hence, the correct reconstruction of profiles of all three fields u'_x, u'_y, θ' is crucial. Consequently, the normalized average error $\mathcal{E}_{\mathrm{NARE}}$ is a suitable measure of the ESN's performance. Here, the arithmetic average $\overline{\mathcal{E}_{\mathrm{NARE}}}$ of the NAREs of all three fields will be considered. Furthermore, to increase its significance, the NARE of the heat flux $\langle u_z'\theta'\rangle_{x,t}$ is added to the average to increase its significance. The measure then reads

$$\overline{\mathcal{E}_{\mathrm{NARE}}} = \frac{1}{4}\Big[\mathcal{E}_{\mathrm{NARE}}\left(\langle {u_x'}^2\rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle {u_z'}^2\rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle \theta'^2\rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle u_z'\theta'\rangle_{x,t}\right)\Big] \tag{4.4}$$

In this way, it is expected that the final reservoir results capture both first- and second-order statistics.

Figure 4.5 depicts the obtained relationships between the ESN output error and its corresponding hyperparameters in the validation phase. For comparison, both the NRMSE and NARE landscapes are shown. Several things can be observed. Firstly, the error landscapes of both measures differ greatly. Although particular transition regions, as can be seen in Fig. 4.5b) and e), are recognizable in both instances, their respective regions of minimum and maximum values have been swapped. That is, low NRMSE regions correspond to high NARE values and vice versa. This means reservoir outputs that reconstruct the vertical properties of the physical fields do not reproduce the exact POD time series. Once more, this outcome is within expectations, given that no ESN output can perfectly align with the validation signal. Low NRMSE values simply indicate that the average of the inferred time series is in proximity to the average of the actual ground truth signal. Based on this analysis, two optimal hyperparameter sets $\Xi_{\mathrm{r}}^{(1)}, \Xi_{\mathrm{r}}^{(2)}$ can be identified. They correspond to the minimal error values of the median $\mathcal{E}_{\mathrm{NRMSE}}$ and $\overline{\mathcal{E}_{\mathrm{NARE}}}$ respectively and are labeled in Fig. 4.5c,f) by a green star marker. It can be seen that both combinations differ in their choice of γ_{r} and ϱ_{r}, while the values for reservoir size and regression parameter coincide. The two sets are listed in Table 4.3. In the following, the results for both settings during the testing phase will be evaluated.

	Applied Error Measure	γ_{r}	ϱ_{r}	λ_{r}	N_{r}	D_{r}	σ_{r}	$\overline{\mathcal{E}_{\mathrm{NARE}}}$	$\mathcal{E}_{\mathrm{NRMSE}}$
$\Xi_{\mathrm{r}}^{(1)}$	$\overline{\mathcal{E}_{\mathrm{NARE}}}$	0.6	0.88	71.94	2048	0.2	1.0	$4.07 \cdot 10^{-3}$	0.23
$\Xi_{\mathrm{r}}^{(2)}$	$\mathcal{E}_{\mathrm{NRMSE}}$	0.1	1.00	71.94	2048	0.2	1.0	$4.34 \cdot 10^{-2}$	0.20

Table 4.3: Optimal **ESN hyperparameter settings** $\Xi_{\mathrm{r}}^{(1)}$ after analysis of the average NARE value and $\Xi_{\mathrm{r}}^{(2)}$ after analysis of the NRMSE. Both settings correspond to the lowest value of the respective measure. Both settings differ in their leaking rate and spectral radius. Both NRMSE and average NARE are listed for both settings as well. These settings are labeled by the green star marker in Fig. 4.5c,f). Note that reservoir density and input scaling were both fixed and not optimized.

4.1.3.2 Results of the combined POD-RC model

The autoregressive reservoir outputs for $\Xi_{\mathrm{r}}^{(1)}$ and $\Xi_{\mathrm{r}}^{(2)}$ during the testing phase are shown in Fig 4.6a-e). Again, one finds that both reservoirs do not reproduce the ground truth signals $a_i(t)$. Especially in the slowly evolving first modes a_1 and a_2, discrepancies between reservoir outputs and the POD coefficients can be seen. However, the individual dynamics of the coefficients are captured by both signals as the variance of the outputs increases with growing mode degree. This again suggests that the reservoir acts as a generator of the dynamical

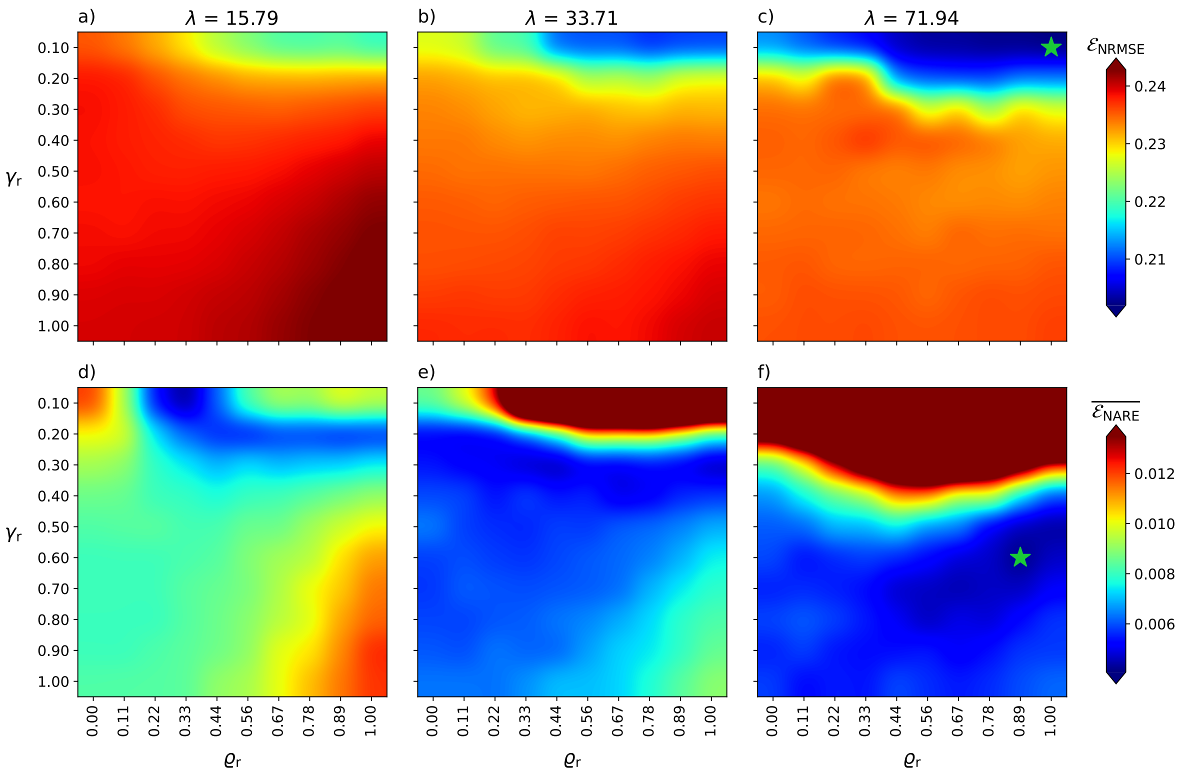

Figure 4.5: **Results of the grid search** procedure of the closed-loop ESN run with the two-dimensional RBC data. Shown is the hyperparameter landscape of leaking rate γ_r, spectral radius ϱ_r and regression parameter λ_r. The corresponding reservoir size is $N_\mathrm{r} = 2048$. For comparison, both the normalized root mean square error $\mathcal{E}_\mathrm{NRMSE}$ (a-c) and the normalized average relative error, averaged over the values for $u_x,' u_z', \theta'$ and $u_z'\theta'$, $\overline{\mathcal{E}_\mathrm{NARE}}$ (see eq. (4.4))(d-f) are shown. The error landscapes of both measures differ vastly from each other. Out of all 100 reservoir realizations, the one corresponding to the median error value is shown. The green star symbols mark the minimum error value and hence the corresponding optimal hyperparameter settings $\Xi_\mathrm{r}^{(1)}$ and $\Xi_\mathrm{r}^{(2)}$.

system rather than a predictor of a certain realization of that system. In Fig. 4.6f-j), the Fourier power spectrum $|\mathcal{F}(\cdot)|^2$, i.e. the square of the absolute value of the Fourier transform, of the three signals are shown. The hyperparameter set $\Xi_\mathrm{r}^{(1)}$, optimized for the vertical profiles of the reconstructed flow, is found to be in good accordance with the spectrum of the ground truth. The spectrum of $\Xi_\mathrm{r}^{(2)}$, on the other hand, drops off quickly and therefore fails to reproduce the high frequencies, especially in the rapidly varying time coefficients a_{50} and a_{150}.

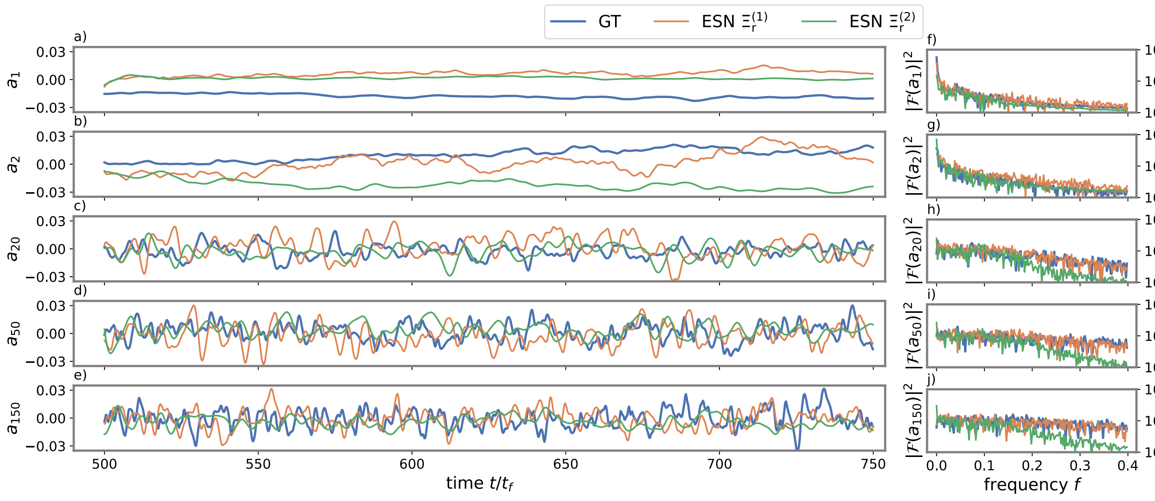

Figure 4.6: (a-e) **Reservoir testing phase outputs** for $\Xi_\mathrm{r}^{(1)}$ (orange) and $\Xi_\mathrm{r}^{(2)}$ (green) according to the lowest value of $\overline{\mathcal{E}_\mathrm{NARE}}$ and $\mathcal{E}_\mathrm{NRMSE}$ resepectively (see Tab. 4.3). The POD ground truth trajectory is shown in blue. (f-j) Their corresponding Fourier power spectrum $|\mathcal{F}(\cdot)|^2$ is shown on the right. It can be seen that the hyperparameter set optimized for the NRMSE shows poor performance in capturing the highly frequent components of the original signal.

To obtain information on the inferred convection flow, the inferred time coefficients and their corresponding spatial modes Φ_i are combined to reconstruct the physical fields. Moreover, using expressions (4.1 - 4.3), the fields $\theta(x,z)$, $u_x(x,z)$ and $u_z(x,z)$ can be derived. See Fig. 4.7 for an instantaneous snapshot of the three fields and the heat flux $u_z\theta$. Arranged from left to right, the columns display the actual field values and, using ESN outputs, the reconstructed field values linked to the lowest NARE and NRMSE scores correspondingly. Even though the time coefficients have diverged from the ground truth, the reconstructed fields show characteristical features of the underlying Rayleigh-Bénard flow. Thermal plumes, as well as up-and downdrafts, can be identified for both POD-RC models. Moreover, the two-dimensional heat flux field is reproduced nicely. As previously discussed, the ESNs can be regarded as temporal generators. Consequently, their integration with the POD enables them to not only replicate temporal aspects but also generate the distinctive spatial characteristics of Rayleigh-Bénard convection. However, for Ξ_r^2, it can be observed that the reproduced features lack the fine structures found in the original RBC flow. See e.g., Fig. 4.7l). The NARE-optimized ESN, on the other hand, shows finer features, also in the heat flux.

Reproducing vertical profiles of the aforementioned fields is crucial for potential SP applications. Hence, the lateral-time averages will be analyzed as well. Figure 4.8 shows the profiles

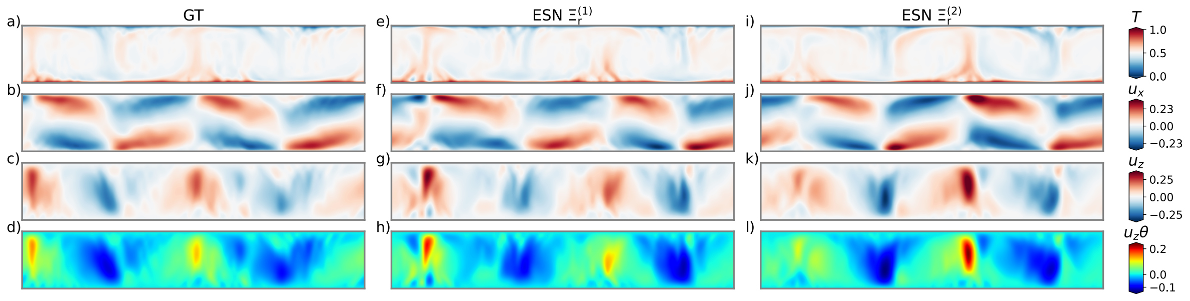

Figure 4.7: **Reconstructed physical fields** of POD (a,b,c,d) and reservoir predictions based on $\Xi_\mathrm{r}^{(1)}$ (e,f,g,h) and $\Xi_\mathrm{r}^{(2)}$(i,j,k,l). Shown is the temperature field $T(x,z) = 1 - z + \theta(x,z)$ (a,e,i), horizontal velocity field $u_x(x,z)$ (b,f,j), vertical velocity field $u_z(x,z)$ (c,g,k) and heat flux $u_z\theta(x,z)$ (d,h,l). Both reservoirs produce the typical spatial patterns of the two-dimensional RBC flow. The time instance corresponds to $t = 750t_f$ in Fig. 4.6a-e).

of the root-mean-square fluctuations of the three physical fields and the heat flux. Recall that the setting $\Xi_\mathrm{r}^{(1)}$ was chosen, s.t., the ESN reconstructions match these profiles. Therefore, good agreement is found between all profiles generated by the underlying RC model. This can not be said for the statistics associated with $\Xi_\mathrm{r}^{(2)}$. While the profiles of $\langle\theta^2\rangle_{x,t}^{1/2}$ match fairly well, the statistics of both velocities show discrepancies near the boundaries and bulk. Moreover, the minimal NRMSE setting does not retain the constant heat flux inside the RBC cell but produces an unrealistic profile. This suggests that replicating the inherent RBC flow dynamics depends not just on effective dimensionality reduction via POD, but also on the temporal features produced by the ESN. Overall, the NARE-optimized reservoir shows good performance in reproducing the temporal, spatial, and statistical features of the two-dimensional Rayleigh-Bénard flow.

These are promising first results. However, the complexity of the present RBC model is still pretty contained, as aspect ratios of mesoscale convection processes can reach up to $\Gamma \sim 10^2$ [208]. Moreover, in a real atmosphere, the presence of water vapor can lead to the formation of clouds, which are, to this day, one of the largest sources of uncertainty in the modeling process of Earth's climate system [209].

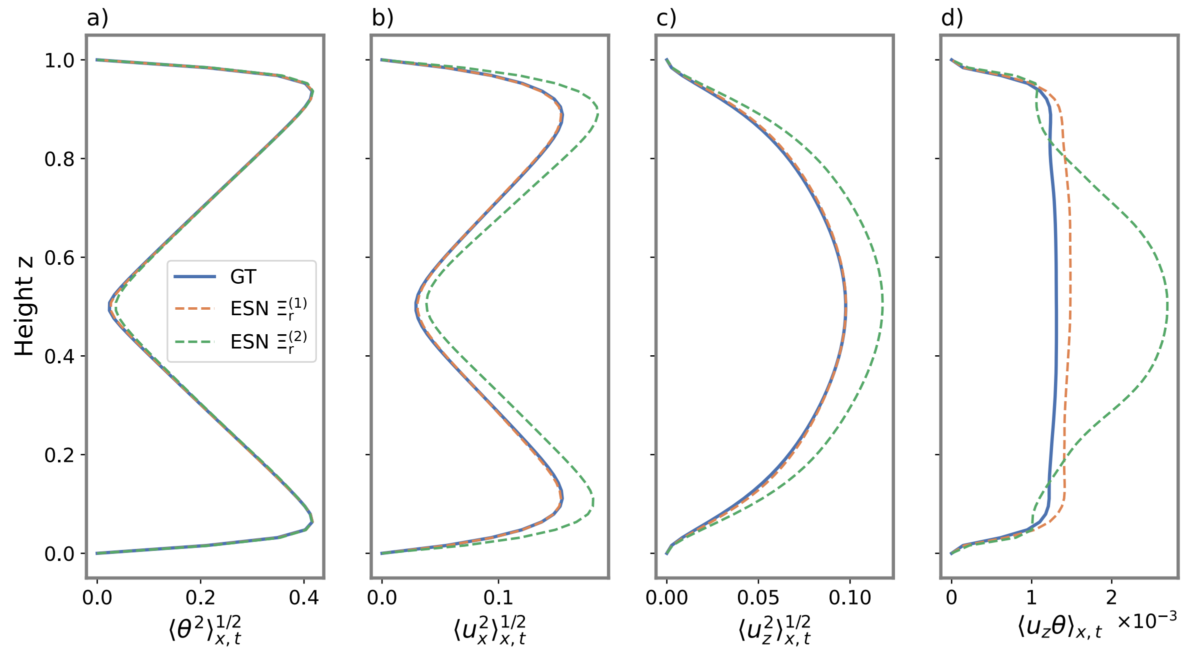

Figure 4.8: **Vertical profiles** of lateral-time averages $\langle\cdot\rangle_{x,t}$ of root-mean-square fluctuations of temperature deviations (a), horizontal vertical (b), and vertical velocity (c). The heat flux $\langle u_z\theta(z)\rangle_{x,t}$ is shown in (d). The reservoir optimized for $\mathcal{E}_{\mathrm{NRMSE}}$ (dashed green) reproduces the trends of POD reference profiles (blue) of all three fields. However, it does not capture the heat flux profile. Note that the ESN hyperparameters in $\Xi_{\mathrm{r}}^{(1)}$ were chosen to optimize $\overline{\mathcal{E}_{\mathrm{NARE}}}$. s.t. their lateral-time averages (dashed orange) match the reference curve to a high degree.

4.2 Two-dimensional moist convection in an absolutely unstable layer

To address the aforementioned topics, this section will consider moist convection in a domain with an enhanced aspect ratio. For this, the moist Rayleigh-Bénard convection (MRBC) model of shallow moist convection by Pauluis and Schumacher [210] will be employed. This system of increased physical complexity introduces an additional scalar field, the liquid water content, to the governing equations of motion. The enhanced spatial domain combined with the complex thermodynamics extends the ESN's application to RBC, presented above, to a more challenging physical system. The following section will shortly introduce the concept of the MRBC model, after which the DNS and subsequent POD-RC model will be discussed.

4.2.1 The moist Rayleigh-Bénard convection model

This subsection will briefly review the model for moist Rayleigh-Bénard convection in two spatial dimensions. A detailed derivation can be found in [210–212]. The MRBC framework is based on the mathematically equivalent formulation by Bretherton [213, 214]. Similar simplified models of moist convection with precipitation were introduced by Smith and Stechmann [215], and Vallis *et al.* [216]. For example, evaporative cooling and buoyancy reversal effects were discussed by Abma *et al.* [217].

The dimensional buoyancy in atmospheric convection is given by [92]

$$b = -g\frac{\rho(S, q_v, q_l, q_i, p) - \rho_0}{\rho_0} \tag{4.5}$$

where g is again the gravity acceleration and ρ_0 the hydrostatic mean density (see again eq. (2.4)). Moreover, the density depends on the pressure p, the entropy S, and the contents of water vapor q_v, liquid water q_l, and ice q_i. Here, only warm clouds, i.e., $q_i = 0$, will be considered. Further, local thermodynamic equilibrium is assumed. From the latter assumption, it follows that no precipitation is possible, and the number of independent variables in eq. (4.5) reduces to three. By introducing the total water content $q_T = q_v + q_l$, the buoyancy can further be expressed as $b(S, q_T, p)$. Moreover, similar to its dry counterpart, the MRBC model makes use of the Boussinesq approximation. Therefore, the pressure p loses its thermodynamic meaning (see again Section 2.1.1), s.t. the buoyancy is only dependent on the hydrostatic pressure $p_0(z)$ and the buoyancy becomes $b(S, q_T, z)$. Furthermore, the convection layer is assumed to be near the vapor-liquid phase boundary. The buoyancy can then be expressed as a piecewise linear function of S and q_T on both sides of the saturation line. This step preserves the discontinuity of the first partial derivatives of b and, therefore, the physics of a first-order phase transition. The advantage of this formulation is that, locally, the saturation state of the air can be determined. In the final stage, the linear combinations of S and q_T on both sides of the phase boundary are replaced with a dry buoyancy D and a moist buoyancy M. Consequently the buoyancy field b can be written as [210]

$$b(x, z, t) = \max\left(M(x, z, t), D(x, z, t) - N_0^2 z\right) \tag{4.6}$$

where the fixed Brunt-Väisälä frequency

$$N_0 = \sqrt{\frac{(\Gamma_u - \Gamma_s)\, g}{T_0}} \tag{4.7}$$

is determined by the lapse rate Γ_s/Γ_u of saturated/unsaturated moist air and the adiabatic reference temperature T_0. An air parcel at height z and time t is unsaturated if $M(x,z,t) < D(x,z,t) - N_0^2 z$ and saturated if $M(x,z,t) > D(x,z,t) - N_0^2 z$. Note that the newly introduced dry buoyancy field D is proportional to the liquid water static energy (or liquid water potential temperature) and the moist buoyancy field M to the moist static energy (or equivalent potential temperature). Similar to dry Rayleigh-Bénard convection with constant temperature boundary conditions, the dimensional fields D and M can be separated into static diffusive profiles $D_{\text{diff}}(z), M_{\text{diff}}(z)$ and their fluctuations θ_M, θ_D about them

$$D(x,z,t) = D_{\text{diff}}(z) + \theta_D(x,z,t) \tag{4.8}$$
$$M(x,z,t) = M_{\text{diff}}(z) + \theta_M(x,z,t) \tag{4.9}$$

with

$$D_{\text{diff}}(z) = D_{\text{bottom}} + \frac{D_{\text{top}} - D_{\text{bottom}}}{H} z \tag{4.10}$$
$$M_{\text{diff}}(z) = M_{\text{bottom}} + \frac{M_{\text{top}} - M_{\text{bottom}}}{H} z \tag{4.11}$$

where D_{bottom}, M_{bottom} and D_{top}, M_{top} are the imposed values of D, M at the bottom ($z = 0$) and top ($z = H$) of the computational domain. This section will consider, $D_{\text{bottom}} = M_{\text{bottom}}$. In order to non-dimensionalize the MRBC equations of motion, its characteristic scales have to be introduced. Here, both buoyancy fields are expressed in terms of the moist buoyancy scale $\Delta b = M_{\text{bottom}} - M_{\text{top}}$, while the spatial scale is again the height of the convection layer H. The free-fall velocity and time are then given by expressions (2.14) and (2.16) with the new definition of Δb. Four dimensionless numbers can be identified: the Prandtl number, dry Rayleigh number, and moist Rayleigh number are given by

$$\mathrm{Pr} = \frac{\nu}{\kappa} \tag{4.12}$$
$$\mathrm{Ra}_D = \frac{(D_{\text{bottom}} - D_{\text{top}})\, H^3}{\nu\kappa} \tag{4.13}$$
$$\mathrm{Ra}_M = \frac{(M_{\text{bottom}} - M_{\text{top}})\, H^3}{\nu\kappa}\,. \tag{4.14}$$

Note that in this way the moist Rayleigh number takes on the role of Ra in eq.(2.21). Additionally, both saturated and unsaturated air possess the same viscosity ν and thermal diffusivity κ, resulting in the absence of any differential diffusion between the two gas constituents. An additional parameter emerges due to the phase transition

$$\mathrm{CSA} = \frac{N_0^2 H}{M_{\text{bottom}} - M_{\text{top}}}\,. \tag{4.15}$$

The condensation in saturated ascent (CSA) controls the amount of latent heat an ascending saturated parcel can release on its way to the top. Note that both Reynolds and Nusselt number as defined in (2.23) and (5.9) can be rewritten in terms of Ra_M and M. They are, therefore, given by

$$\mathrm{Re}_M = \sqrt{\frac{\mathrm{Ra}_M}{\mathrm{Pr}}} \langle u_x^2 + u_z^2 \rangle_{V,t}^{1/2}, \tag{4.16}$$
$$\mathrm{Nu}_M = 1 + \sqrt{\mathrm{Ra}_M \mathrm{Pr}} \langle u_z M \rangle_{V,t}. \tag{4.17}$$

The non-dimensional governing equations of the moist Boussinesq system are then given by the unchanged mass and momentum balances (2.17) and (2.18). The latter incorporates the nondimensional buoyancy

$$b^* = \max\left(\theta_M^*, \theta_D^* + \left(1 - \frac{\mathrm{Ra}_D}{\mathrm{Ra}_M} - \mathrm{CSA}\right) z^*\right). \tag{4.18}$$

They are complemented by two advection-diffusion equations for the non-dimensional buoyancy fields M^* and D^*, respectively

$$\frac{\partial D^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla_*) D^* = \sqrt{\frac{1}{\mathrm{Ra}_M \mathrm{Pr}}} \nabla_*^2 D^* \tag{4.19}$$

$$\frac{\partial M^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla_*) M^* = \sqrt{\frac{1}{\mathrm{Ra}_M \mathrm{Pr}}} \nabla_*^2 M^*. \tag{4.20}$$

Again, the asterisk notation for non-dimensional variables is omitted for readability. This idealized model describes the formation of warm, non-precipitating low clouds in a shallow layer up to a depth of $\sim$ 1km. The assumptions made here hold, for example, to a good approximation over the subtropical oceans. Moreover, the saturation condition (4.18) implies that liquid water is immediately formed at a point in space and time when $\theta_M > \theta_D - (1 - \mathrm{Ra}_D/\mathrm{Ra}_M - \mathrm{CSA})z$. Therefore, there is no supersaturation considered in this model, and the liquid water content field q_l and thus the clouds are given by

$$q_l(x, z, t) = \theta_M - \theta_D - \left(1 - \frac{\mathrm{Ra}_D}{\mathrm{Ra}_M} - \mathrm{CSA}\right) z. \tag{4.21}$$

Note that in this formulation, q_l can become negative as it measures the degree of saturation. When the atmosphere is saturated, $q_l \geq 0$ and the conventional liquid water content is retained. The following sections will study the case of $\mathrm{Ra}_D, \mathrm{Ra}_M > 0$. Hence both buoyancy fields are linearly unstable. For the case of a *conditionally unstable* moist layer with $\mathrm{Ra}_D \leq 0$, the reader is referred to [213, 214] or [218]. Finally, note that dry RBC case is obtained when $b = M$, $D = 0$, $\mathrm{CSA} = 0$.

4.2.2 Direct numerical simulation of moist Rayleigh-Bénard convection

The two-dimensional Boussinesq system (2.17, 2.18) with buoyancy (4.18) together with the two diagnostic equations (4.19, 4.20) are solved numerically using the *Nek5000* code [180]. No-slip boundary conditions are considered for the velocity fields. Similar to dry RBC, both dry and moist buoyancy fields are prescribed with a constant bottom (D_{bottom}, M_{bottom}) and top value (D_{top}, M_{top}). Both Rayleigh numbers were chosen as $\mathrm{Ra}_M = 4 \cdot 10^8$ and $\mathrm{Ra}_D = 2 \cdot 10^8$, which corresponds to an absolutely unstable moist atmosphere. Moreover, the Prandlt number was set to $\mathrm{Pr} = 0.7$, which closely resembles moist air. Finally, the fourth MRBC parameter was set to $\mathrm{CSA} = 0.3$. Further parameters are listed in Table 4.4. The DNS data is subsequently interpolated to the uniform grid of size $N_x \times N_z = 1024 \times 64$ for further processing. A snapshot of the moist buoyancy and velocity fields is shown in 4.9a). Moreover, the resulting two-dimensional cloud layer field $q_l(x, z) > 0$ can be obtained by using the expression (4.21) for the liquid water content q_l. It can be seen that the turbulent fluid motion leads to the formation of complex convection cells. Further, a broken cloud cover, consisting of individual clouds, can be identified.

Γ	Ra_M	Ra_D	Pr	CSA	N_e	N	$N_x \times N_z$	t_{DNS}/t_f	$\Delta t_{\mathrm{DNS}}/t_f$
24	4×10^8	$2 \cdot 10^8$	0.7	0.3	600×32	11	1024×64	2500	10^{-3}

Re_M	τ_{eddy}/t_f	Nu_M
8335	2.87	30.98

Table 4.4: **Simulation parameters** of the direct numerical simulation of moist Rayleigh-Bénard convection. The table contains the aspect ratio Γ of the two-dimensional cell, the moist and dry Rayleigh numbers Ra_M, Ra_D, the Prandtl number Pr, the condensation in saturated ascent parameter CSA, as well as the total number of spectral elements N_e and polynomial order N on each element. The DNS was interpolated to the uniform grid of size $N_x \times N_z$. t_{DNS} denotes the period over which the DNS was run. The Reynolds number Re_M, as defined in (4.16) largest eddy turnover time τ_{eddy} and the moist Nusselt number Nu_M, as defined in (4.17), are diagnostic parameters, obtained from the DNS.

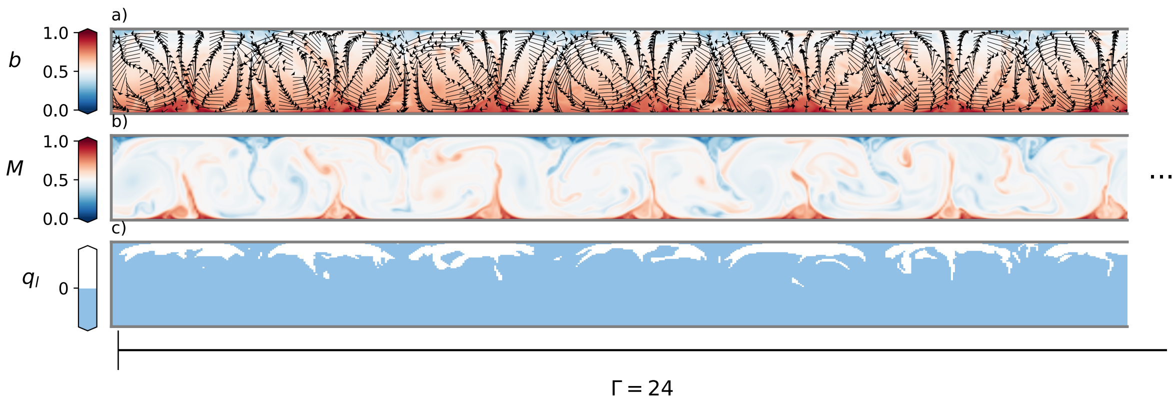

Figure 4.9: Instanteneous **DNS Snapshot** of (a) buoyancy field $b(x, z)$, (b) moist buoyancy field $M(x, z)$ and (c) the associated positive liquid water content $q_l(x, z) > 0$, i.e., clouds. The arrows in (a) indicate the two-dimensional velocity field. The simulation parameters are listed in Table 4.4. The aspect ratio is $\Gamma = 24$. For visualization purposes, only half of the horizontal extent $\Gamma/2 = 12$ is shown.

4.2.3 Dimensionality reduction via POD

The 1024×64 degrees of freedom per field have to be compressed by means of a Reduced Order Model to establish a workable input dimension for the subsequent RC model. To this end, a POD of the four fields $(u'_x, u'_z, \theta'_M, \theta'_D)^T$, based on the aforementioned DNS data, is computed. Here, the prime denotes fluctuations about the time mean, i.e., the fields are decomposed into

$$u_x(x,z,t) = \langle u_x(x,z)\rangle_t + u'_x(x,z,t), \tag{4.22}$$

$$u_z(x,z,t) = \langle u_z(x,z)\rangle_t + u'_z(x,z,t), \tag{4.23}$$

$$\theta_M(x,z,t) = \langle \theta_M(x,z)\rangle_t + \theta'_M(x,z,t), \tag{4.24}$$

$$\theta_D(x,z,t) = \langle \theta_D(x,z)\rangle_t + \theta'_D(x,z,t). \tag{4.25}$$

Note that the POD now considers one more field compared to the last section due to the additional phase transition. Figure 4.10a) shows the cumulative variance captured by the POD. While the first mode incorporates about 38% of the total variance, the individual contributions become less as the mode number increases. Again, the choice of cut-off mode is a trade-off between keeping the ESN input dimension as low as possible and retaining enough physical information about the fields. Here, a ROM of $N_{\mathrm{POD}} = 150$ is considered, as it still allows for feasible reservoir sizes and captures 79.94% of the initial information. The resulting liquid water flux $u'_z q'_l$, an important parameter for representing subgrid-scale moist processes, is shown in Fig. 4.10b). While the ROM captures the main spatial patterns of $u'_z q'_l$, its vertical properties also agree to a reasonable extent with the DNS profile. This is illustrated in Fig. 4.10c), which shows $\langle u'_z q'_l(z)\rangle_{x,t}$ of both DNS and reduced POD model. Figure 4.11 shows the resulting spatial modes for selected mode numbers. Similar characteristics to the dry case can be observed. Low mode numbers capture large-scale features, while higher modes incorporate fine-scale structures. The modes of vertical velocity show dominant patterns in the bulk of the cell. In contrast, moist and dry buoyancy modes exhibit a

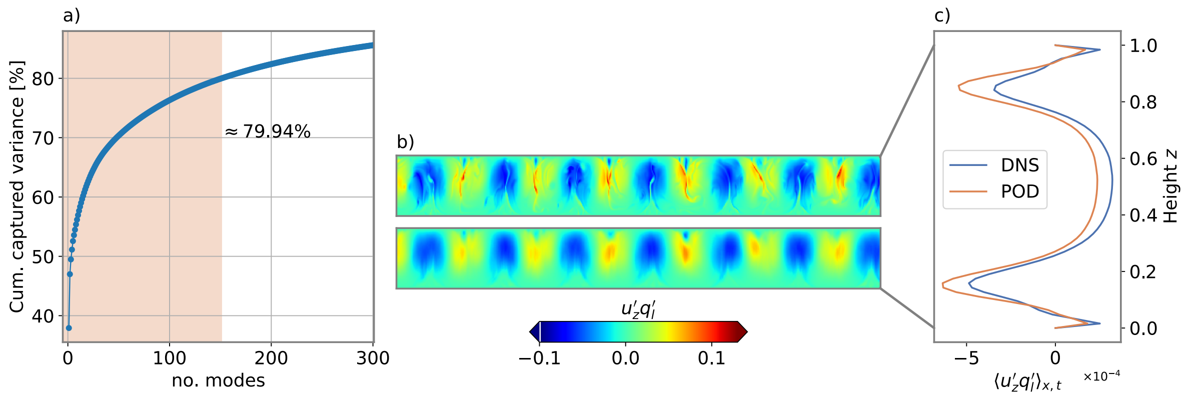

Figure 4.10: (a) **Cumulative POD spectrum:** The red-shaded region marks the first 150 modes that make up the Reduced Order Model. (b) Instantaneous snapshot of liquid water flux $u'_z q'_l$ of DNS (top) and POD model using 150 modes (bottom). Again, only half of the horizontal domain is shown. (c) Corresponding lateral time average profile $\langle u'_z q'_l \rangle_{x,t}$, where blue denotes the DNS and orange the POD profile. It can be seen that the ROM captures the main characteristics of the DNS. However, it demonstrates diminished values within the bulk region and near $z = 0.2$ and $z = 0.8$, indicating the loss of variance.

more localized behavior in the boundary layers, where plumes form and detach. The corresponding time coefficients $a_i(t)$, shared by all modes, are shown in Figure 4.12. Once more, alterations in a_1 and a_2 manifest across extended time scales, with higher frequencies becoming more prevalent as the mode number increases.

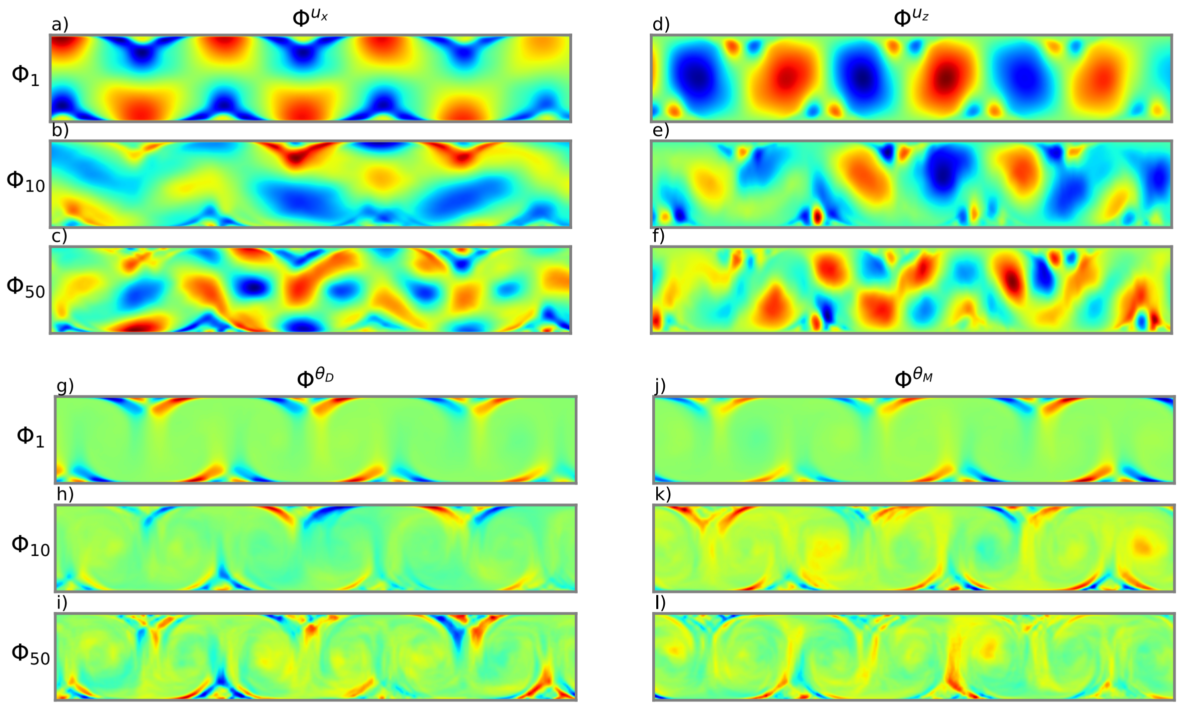

Figure 4.11: **POD spatial modes** of horizontal (a-c) and vertical velocity (d-f), as well as the ones corresponding to the dry (g-i) and moist (j-l) buoyancy deviations from their respective diffusive profiles. While the low mode numbers capture large-scale spatial features, higher modes incorporate the fine-scale structure of the physical fields. For visualization purposes, only a quarter $\Gamma/4 = 6$ of the spatial domain is shown. See also Fig 4.12 for the corresponding time coefficients.

The following section will subsequently use this Reduced Order Model of moist Rayleigh-Bénard convection as a training database for an Echo State Network. Again, the ESN will be tasked to infer the underlying POD dynamics. The assessment of the reconstructed moist convection flow will involve an evaluation of its alignment with regard to dynamic, spatial, and statistical properties.

4.2.4 POD-RC model of two-dimensional moist Rayleigh-Bénard convection

4.2.4.1 ESN hyperparameter search

In order to find the best ESN hyperparameter set Ξ_r, a grid search of its hyperparameters is performed. The details are listed in Tab. 4.5. The hyperparameters that are considered stay the same as in Section 4.1. One of the results of the last section was the choice of error measure. It was shown that the arithmetic average of the normalized average relative error of the flow's vertical profiles $\overline{\mathcal{E}_\mathrm{NARE}}$ is a meaningful measure of the ESN's performance. For the present moist case, a similar approach to the one in expression (4.4) is employed. However,

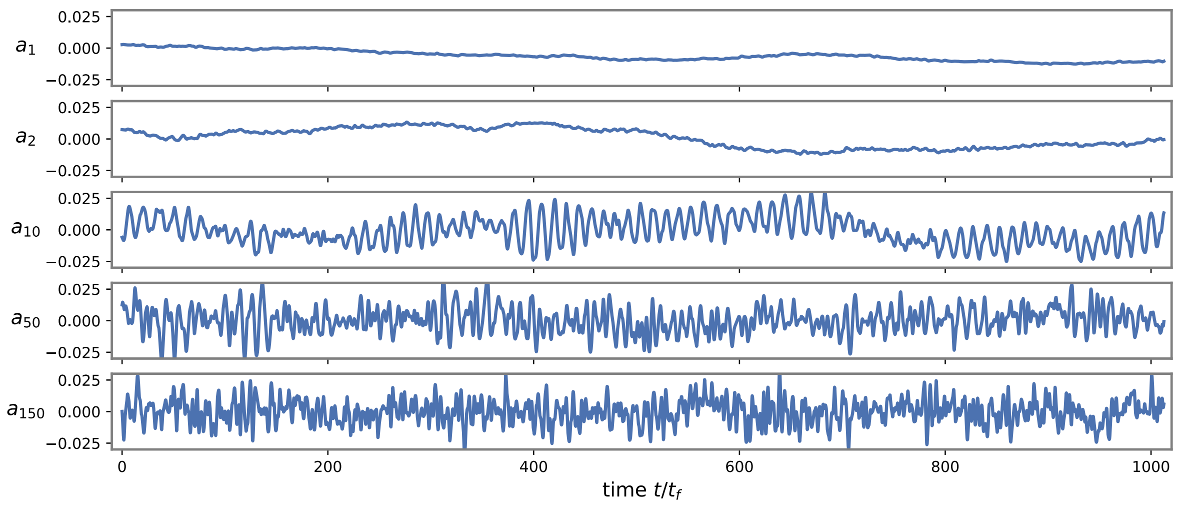

Figure 4.12: **POD time coefficients** $a_i(t)$ for modes $i = 1, 2, 10, 50, 150$. The time coefficients determine the temporal behavior of the associated spatial modes $\Phi_i(x, z)$ in Fig 4.11. Note the increasing range of frequencies for higher mode numbers.

γ_r	ϱ_r	λ_r	N_r
$[0.1, 1.0, 0.1]$	$[0, 3.0, 0.3]$	$\{0.035, 0.26, 1.87, 13.68, 100\}$	$\{1024, 2048\}$

Reservoir realizations	t_train/t_f	t_val/t_f	t_test/t_f
100	1000	500	500

Table 4.5: **ESN hyperparameter grid search** procedure for the closed-loop ESN trained on the MRBC data. Square brackets indicate the range of the searched hyperparameter: $[\mathrm{start}, \mathrm{end}, \mathrm{stepsize}]$. Braces indicate an array of values that were used. t_train, t_val, t_test are the length of the training, validation, and testing interval, respectively. The first $12.5t_f$ out of the training time was used for initializing the internal reservoir state. The reservoir density and input scaling were fixed at $D_\mathrm{r} = 0.2$ and $\sigma_\mathrm{r} = 1.0$. The total number of different reservoir settings is $10 \times 11 \times 5 \times 2 = 1100$, while a total of $100 \times 1100 = 1.1 \cdot 10^5$ reservoirs were run. The total runtime was 38.75 h.

now, this measure will additionally incorporate the second buoyancy term as well as its respective flux. Moreover, the NARE of the liquid water flux $u'_z q'_l$ will also be included. In this way, the accuracy of the convective transport of liquid water is ensured as well. The average NARE then becomes

$$\begin{aligned}\overline{\mathcal{E}_{\mathrm{NARE}}} = \frac{1}{7} \Big[&\mathcal{E}_{\mathrm{NARE}}\left(\langle {u'_x}^2 \rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle {u'_z}^2 \rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle {\theta'_M}^2 \rangle_{x,t}^{1/2}\right) \\ &+\mathcal{E}_{\mathrm{NARE}}\left(\langle {\theta'_D}^2 \rangle_{x,t}^{1/2}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle u'_z\theta'_M \rangle_{x,t}\right) + \mathcal{E}_{\mathrm{NARE}}\left(\langle u'_z\theta'_D \rangle_{x,t}\right) \\ &+ \mathcal{E}_{\mathrm{NARE}}\left(\langle u'_z q'_l \rangle_{x,t}\right)\Big] \qquad (4.26)\end{aligned}$$

Fig. 4.13 presents the median validation $\overline{\mathcal{E}_{\mathrm{NARE}}}$ error over 100 realizations of W^{in} and W^{r}. Shown is the error landscape in dependence of the leaking rate and spectral radius $(\gamma_{\mathrm{r}}, \varrho_{\mathrm{r}})$ for three values of the regression parameters λ_{r} and the two reservoir sizes $N_{\mathrm{r}} = 1024, 2048$. Contrary to the previous hyperparameter tuning procedures, the present case investigates spectral radii larger than unity. It can be seen that low error values can be found at $\varrho_{\mathrm{r}} \gtrsim 1.6$ for $\lambda_{\mathrm{r}} < 100$. This indicates that a sizeable spectral radius can be beneficial for a wide range of hyperparameter settings. Moreover, it can be observed that the low-error basin near high values of γ_{r} and ϱ_{r} (see 4.13a,d) expands at increasing λ_{r} values. Therefore, a wider range of γ_{r} and ϱ_{r} values leads to well-performing ESNs. Moreover, note that for $\lambda_{\mathrm{r}} = 100$, the smaller reservoir exhibits minor errors near low spectral radius values, while the large reservoir has a sharp low-error region at $\varrho_{\mathrm{r}} \gtrsim 0.7$. The optimal ESN setting Ξ_{r} is listed in Tab. 4.6 and can be seen in 4.13c) in the shape of a green star marker. In the following, the testing data set will be used to assess the quality of the inferred moist convection flow.

$\overline{\mathcal{E}_{\mathrm{NARE}}}$	γ_{r}	ϱ_{r}	λ_r	N_{r}	D_{r}	σ_{r}
$4.04 \cdot 10^{-3}$	0.80	3.00	100	1024	0.2	1.0

Table 4.6: **Optimal ESN hyperparameter set** Ξ_{r} according to the grid search shown in Fig. 4.13. Note the large spectral radius, which significantly exceeds unity. The corresponding average NARE $\overline{\mathcal{E}_{\mathrm{NARE}}}$, comprising the NAREs of root-mean-square fluctuations of $u'_x, u'_z, \theta'_M, \theta'_D$, as well the fluxes $u'_z\theta'_M$, $u'_z\theta'_D$ and $u'_z q'_l$ is also listed.

4.2.4.2 Results of the POD-RC model

The reservoir outputs and the ground truth signal are presented in Fig. 4.14a-e). Their corresponding Fourier power spectrum $\|\mathcal{F}(\cdot)\|^2(f)$ is shown in Fig. 4.14g-k). The large spectral radius has an impact on the inferred dynamics, which exhibit additional high-frequency components in comparison to the POD dynamics. This can be seen most notably in a_1, a_2 and a_{20}, where the power spectrum does not match the ground truth. For a_{50} and a_{150}, the frequency spectrum of the reservoir outputs becomes more similar to the one of the POD. Therefore, it seems that the reservoir has trouble reproducing the scale separation of the POD time coefficients. A snapshot of the reconstructed fields of u_x, u_z, and the liquid water flux $u'_z q'_l$ in the middle of the validation phase are shown in Fig. 4.15 (a-f). It can be seen that the autoregressive ESN accurately infers the spatial features of both velocity fields. Again, the exact position and strength of up- and downdrafts are not reproduced, as the ESN's outputs do not align with the POD signal. Nevertheless, the generated two-dimensional fields are almost indistinguishable from the ground truth. Moreover, the turbulent transport of liquid water is also reproduced to a high degree. These results are surprising as the ESN outputs show deficiencies in reproducing all temporal characteristics of the POD time coefficients. Moreover, $u'_z q'_l$ is extremely sensitive to errors, as it incorporates three fields u_z, θ_M

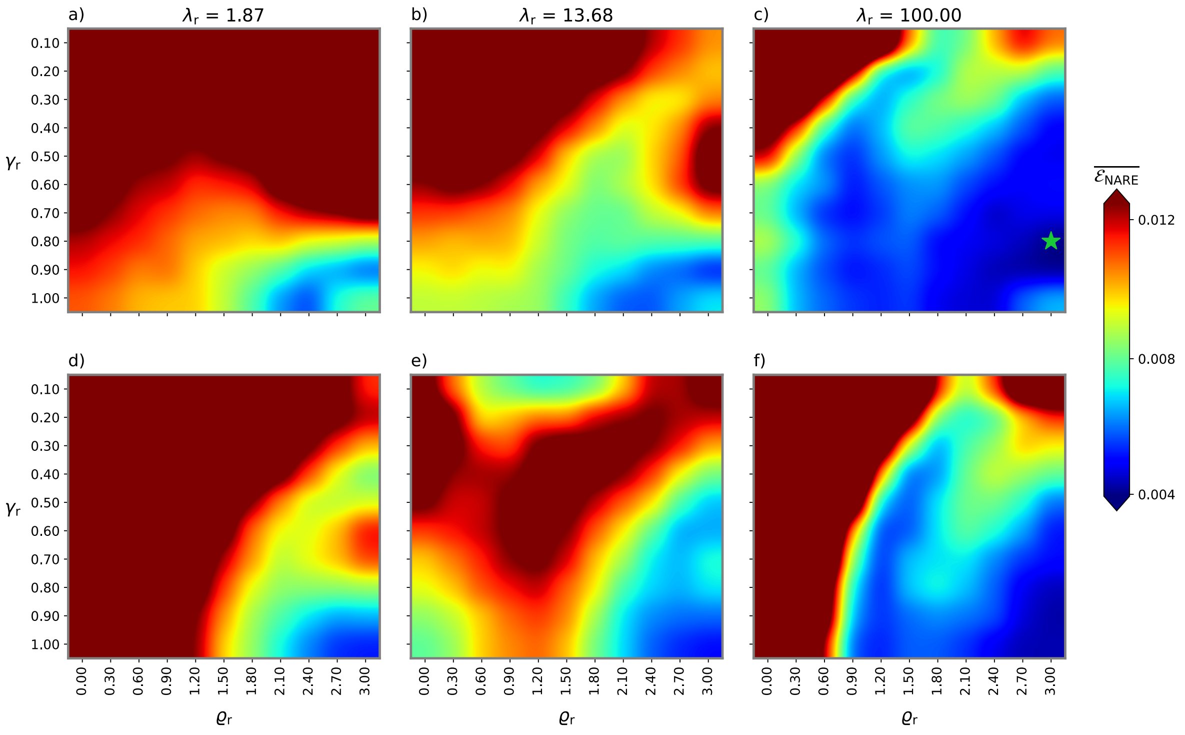

Figure 4.13: **Results of the grid search** process of the closed-loop ESN run for two-dimensional moist Rayleigh-Bénard convection. Shown is the arithmetic average normalized average relative error $\overline{\mathcal{E}_{\mathrm{NARE}}}$ involving the seven fields u'_x, u'_z, θ'_M, θ'_D, $u'_z\theta'_M$, $u'_z\theta'_D$ and $u'_zq'_l$ in dependence of the ESN hyperparameters. The hyperparameter landscape consists of leaking rate γ_r, spectral radius ϱ_r, regression parameter λ_r and reservoir size N_r. Here, (a-c) shows $N_\mathrm{r} = 1024$ and (d-f) shows $N_\mathrm{r} = 2048$. Reservoir density and input scaling were fixed to $D_\mathrm{r} = 0.2$ and $\sigma_\mathrm{r} = 1.0$. Moreover, out of the 100 reservoir realizations, the median $\overline{\mathcal{E}_{\mathrm{NARE}}}$ value is shown. The results were computed on the validation data set. The green star marker labels the lowest error, and hence the optimal hyperparameter set Ξ_r.

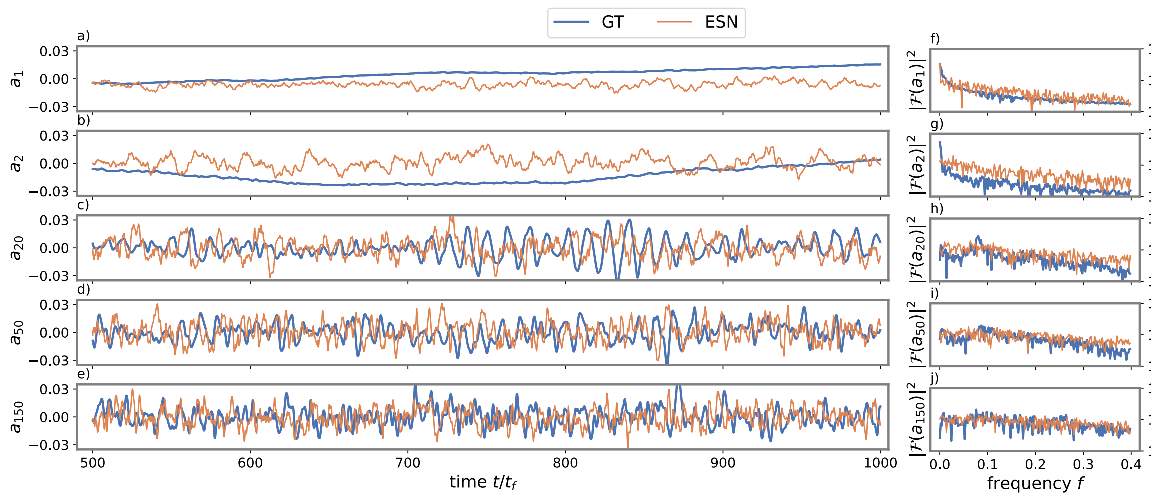

Figure 4.14: **ESN outputs** (orange) and the ground truth POD time coefficients (blue) (a-e) during the validation phase. The corresponding Fourier power spectrum of each coefficient is shown in (f-j). The ESN hyperparameters were chosen according to Ξ_r in Tab. 4.6. Note the excess high frequencies in the first modes.

and θ_D (see again (4.21)). Finally, in Fig. 4.15g,h), the resulting cloud cover, i.e., $q_l > 0$, is presented. It can be observed that the ESN produces a reasonable cloud formation that differs in cloud position and fine features like the clouds' tails. Motivated by these results, it is reasonable to assess whether the ESN is able to infer quantities such as the cloud cover CC. Here the cloud cover is defined as the ratio of the number of vertical grid lines $N_{q_l>0}$ that contain at least one mesh point with $q_l > 0$ along their vertical line of sight and the total number of vertical grid lines N_x (see again Tab. 4.4). Thus follows

$$\mathrm{CC} = \frac{N_{q_l>0}}{N_x} \times 100\%. \tag{4.27}$$

The temporal evolution of the inferred cloud cover is shown in Fig. 4.16a). It can be seen that similar to the temporal predictions of the POD time coefficients, the modeled variation of CC possesses a wide range of frequencies that are not present in the ground truth. However, the overall percentage of cloud cover is inferred to a reasonable degree. The same goes for the volume average of the positive liquid water content $\langle q_l > 0\rangle_{x,z}$, which can be seen in 4.16b). Moreover, the PDFs of the liquid water flux at the three heights $z = 0.1, 0.5, 0.9$ are plotted in 4.16c,d,e). While the ESN captures the symmetric PDF in bulk, it overestimates the tail values near the boundaries. This can be explained by the larger temporal variance of the ESN outputs, which could result in an increased presence of outlier values. However, the overall trend agrees to a reasonable extent. Finally, the POD-RC model will be evaluated in terms of the vertical properties of the inferred fields, which pose essential parameters in the super-parameterization procedure. Figure 4.17 shows the lateral-time average profiles of the root-mean-square values of both velocity fields and moist buoyancy fluctuations θ'_M, as well as of the liquid water flux $u'_z q'_l$. While certain deficiencies in the vertical velocity fields can be observed, the buoyancy profile matches fairly well. Moreover, the liquid water flux profile is captured by the ML model despite small discrepancies near the bulk and in the boundary layer. This shows that the combined POD-RC model can accurately deduce numerous characteristics of this moist convection flow, even though it possesses higher complexity when contrasted with the dry case discussed in Section 4.1.

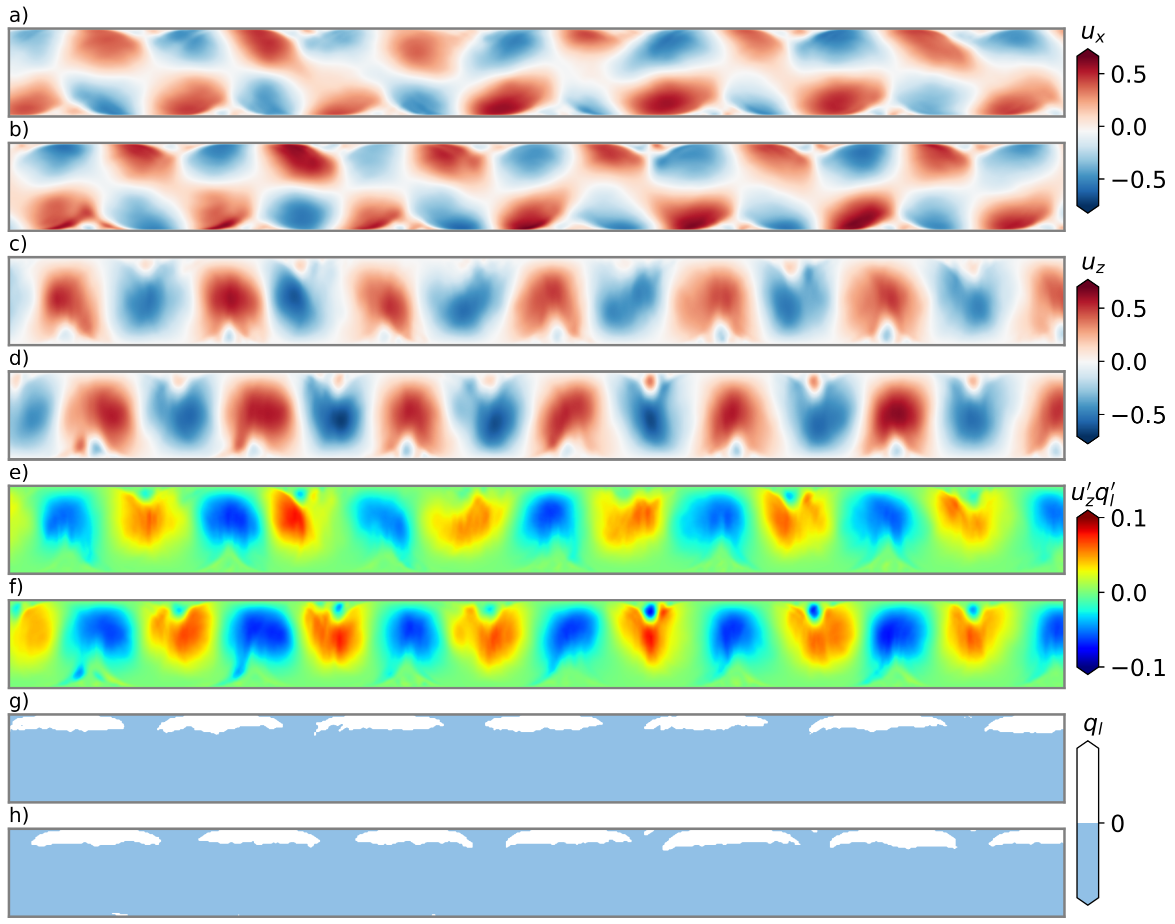

Figure 4.15: **Instantaneous snapshot** of horizontal (a,b) and vertical (c,d) velocity fields, as well as liquid water flux (e,f). The resulting cloud cover, i.e., $q_l > 0$, is shown in (g,h). Note the small differences like the cloud tails in (h). Both reconstructions from ground truth (a,c,e,g) and ESN outputs (b,d,f,h) are shown at time $t = 750t_f$ in the validation phase. The corresponding time coefficients are shown in Fig. 4.14.

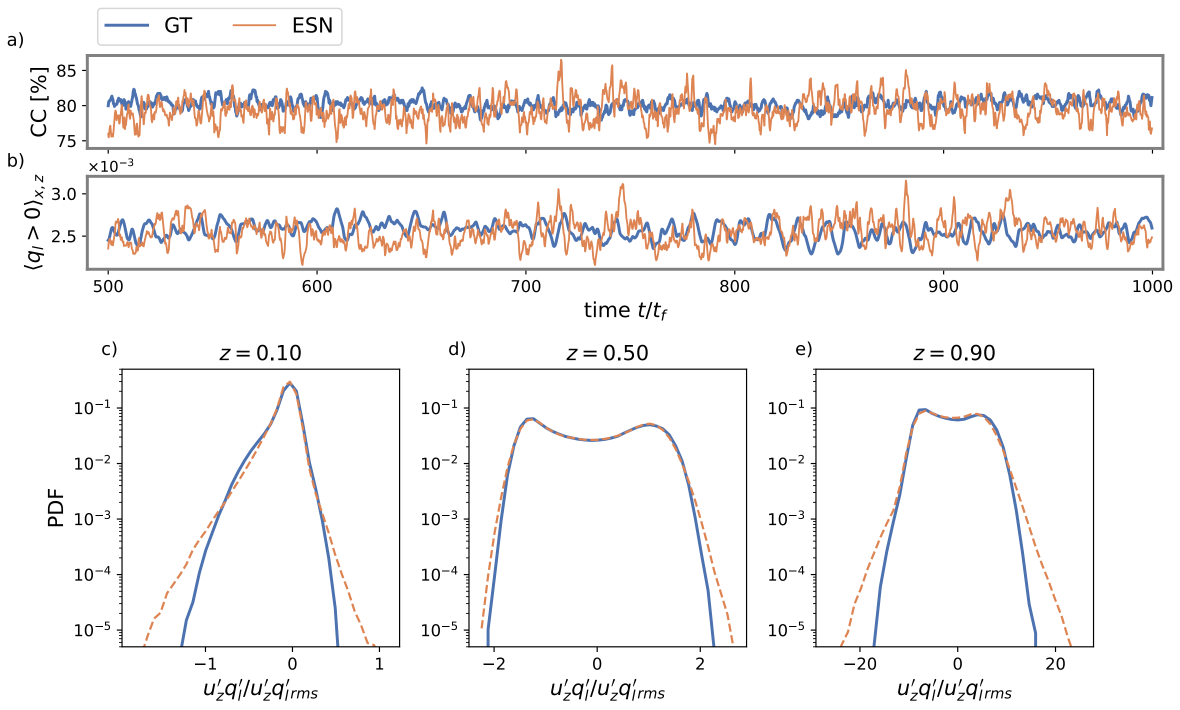

Figure 4.16: (a) Temporal evolution of the **cloud cover** CC, as defined in (4.27). Shown is the CC derived from the ESN's liquid water content (orange) and the corresponding ground truth CC of the POD model (blue). (b) Temporal evolution of the volume average of the strictly positive liquid water content $\langle q_l > 0\rangle_{x,z}$. Again, the POD-RC model produces high-frequency components not present in the ground truth. (c-e) PDFs of the liquid water flux $u'_z q'_l$, normalized by its root-mean-square value $u'_z q'_{l\,rms}$, at three different heights $z = 0.1$ (c), 0.5 (d), 0.9 (e).

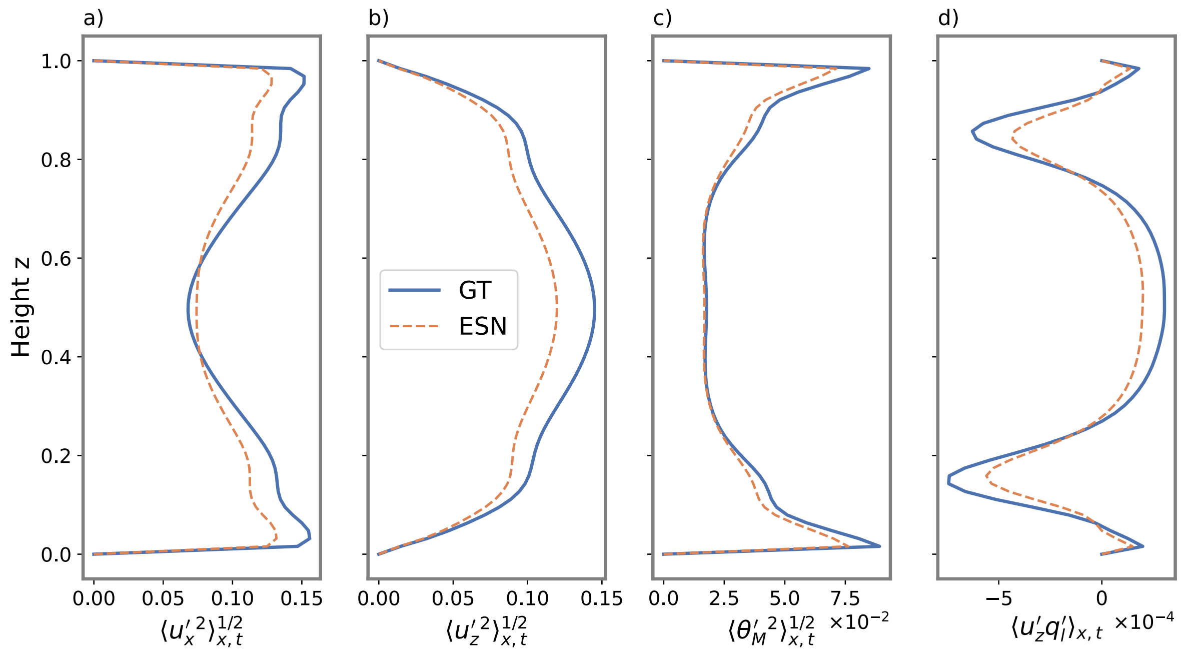

Figure 4.17: **Vertical profiles** of lateral-time averages $\langle \cdot \rangle_{x,t}$ of horizontal (a), vertical velocity (b), as well as moist buoyancy variations (c), and liquid water flux (d). The ESN profiles (dashed orange) align nicely with the ground truth profiles (blue). Deficiencies can be observed in the bulk for both velocities. The liquid water flux matches to a reasonable extent as well.

4.3 Conclusions from this chapter

This chapter studied a combined POD-RC model applied to two direct numerical simulation datasets of two-dimensional Rayleigh-Bénard convection at two levels of complexity. Moreover, the Echo State Network, the dynamical core of the model, was employed in the closed-loop mode, where it runs autoregressively and does not require an external input. This application can be seen as a first step towards an ML-based super-parameterization scheme, where the aforementioned ML model substitutes the subgrid-scale cloud-resolving model. This chapter paved the way for such an application in several ways. Firstly, it was found that the recurrent neural network is not able to reproduce the exact POD trajectory. In view of the results of Section 3.2, this is to be expected, as the now highly turbulent flow exhibits chaotic dynamics, and minor errors in the reservoir's outputs get amplified quickly. Instead, it was found that the ESN's outputs can be seen more as another realization of the chaotic POD dynamics. Hence, it acted more as a generator of temporal features than a time series predictor. Moreover, it was found that the combination of ESN and POD translates this generative behavior to spatial and statistical features. Specifically, the POD-RC model is able to represent characteristic structures like thermal plumes, up-, and downdrafts, as well as the formation of clouds. Further, it reproduces the vertical properties of both dry and moist RBC, including the turbulent transport of heat and moisture to a reasonable extent. Hence, this chapter answered two of the four research questions this doctoral dissertation posed in its introduction. The next chapter will shift the focus from Rayleigh-Bénard convection towards a convective boundary layer, an essential part of our atmosphere. Contrary to the RBC case, the buoyancy boundary conditions in this scenario are given by fixed fluxes. Further, it will introduce a generative network model for sub-grid scale feature generation.

Chapter 5

Towards a machine learning parameterization of the convective boundary layer

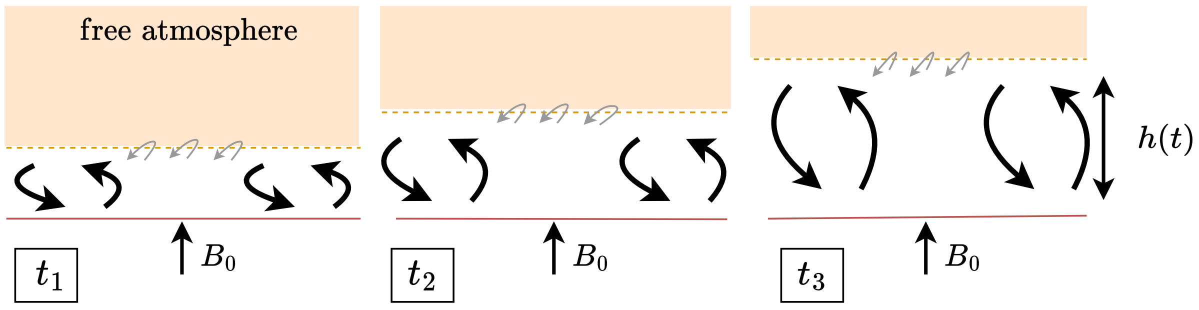

Figure 5.1: Sketch of a **convective boundary layer** at three subsequent time instances. The air above the warm ground is heated by the surface heat flux B_0, and a convective fluid motion is initiated. Contrary to RBC, there is no impermeable top boundary. Instead, warm, non-turbulent fluid imposes a capping inversion (yellow-shaded region), and strong thermals can penetrate into a thin layer above the turbulent mixed layer (gray arrows). There, they get deflected and take some of the warm fluid with them. Consequently, the CBL height $h(t)$ grows over time $t_1 < t_2 < t_3$.

The convective fluid motion considered so far in this thesis was driven by a sustained temperature or, more generally, buoyancy difference between two rigid, impenetrable plates. Convection in nature, on the other hand, often originates from a sustained heat flux rather than from a ground surface that is held at a homogenous temperature. As discussed in Section 1.1.2, the convective motion of the atmospheric boundary layer during daytime is sustained by a continuous heat flux from Earth's surface, which is itself heated by the incoming solar radiation. Hence, to depict more authentic scenarios of atmospheric convection, particularly the ABL, it is appropriate to consider a constant warming heat flux, resembling the surface flux B_0 from Earth's surface. Moreover, an impenetrable top boundary is non-existent for an ABL. Instead, the free, non-turbulent atmosphere is situated above the convective layer. It acts as a capping inversion that hinders the rising fluid from exiting the CBL. In turn, strong thermals penetrate into a small region above the turbulent layer, where they experience a buoyancy reversal and get deflected downwards. This leads to the entrainment of warm, non-turbulent air into the convective layer and, hence, the growth of the CBL. A sketch of a CBL growing over time can be seen in Fig. 5.1. As discussed in Section 1.1.3, General Circulation Models do not, or only partially, resolve CBL processes. Consequently, parameterization schemes are required to represent their characteristic features. In this chapter, the two remaining research questions of this thesis will be addressed, i.e., whether the ROM-RC approach can be used to transition between different convection regimes and whether a generative neural network can model the transient dynamics of a CBL. The former question will be approached regarding a simplified CBL model in Section 5.1. Related results were published in [120]. Finally, the latter problem is discussed in Section 5.2. A corresponding manuscript is currently under review at *Journal of Advances in Modeling Earth Systems*. A pre-print can be found in [219].

5.1 Testing the generalizability of an AE-RC model on a model of the CBL

5.1.1 A simplified model of the convective boundary layer

To test the generalizability of a combined ROM-RC model, a two-dimensional Rayleigh-Bénard system driven by buoyancy fluxes from the bottom and top is considered. See again 2.1.3 for the definition of the Neumann boundary conditions. Moreover, this initial study will focus on the scenario of dry convection. Note that this section will consider dimensional variables. Hence, non-dimensional variables will be marked by an asterisk. Again, all physical fields are periodic in the horizontal x-direction. Further, as no fixed buoyancy is prescribed at the vertical boundaries, the buoyancy difference

$$\langle \Delta b \rangle_x = \langle b(z=0,t) \rangle_x - \langle b(z=H,t) \rangle_x \tag{5.1}$$

becomes a diagnostic parameter, dependent on the incoming heat fluxes at the bottom B_0 and the top B_1. The same goes for the Dirichlet Rayleigh number (2.21) and free-fall scales (2.14,2.16), which can be defined by using $\langle \Delta b \rangle_{x,t}$. Hence, it is convenient to non-dimensionalize the Boussinesq equations (2.7 - 2.9) in terms of the surface flux B_0 and the convection cell height H. One can then define the following characteristic scales: the convective velocity

$$U_c = (B_0 H)^{1/3}, \tag{5.2}$$

the convective time

$$t_c = (H^2/B_0)^{1/3}, \tag{5.3}$$

and the convective buoyancy

$$b_c = (B_0^2/H)^{1/3}. \tag{5.4}$$

While the Prandtl number and aspect ratio, as defined in eq. (2.20) and eq. (2.22) are left unchanged, two new parameters arise: the convective Rayleigh number

$$\mathrm{Ra}_c = \frac{B_0 H^4}{\nu \kappa^2}, \tag{5.5}$$

and the buoyancy-flux ratio

$$\beta = -\frac{B_1}{B_0}. \tag{5.6}$$

Moreover, the diffusive buoyancy profile of the RBC case (2.11) changes as well. From the two-dimensional version of the Boussinesq equations (2.7 - 2.9) the diffusive buoyancy profile follows as

$$b_{\mathrm{diff}}(z,t) = \frac{B_0 H}{\kappa} \frac{(z/H-1)^2 + \beta (z/H)^2}{2} + \frac{B_0}{H}(1+\beta)t + \mathrm{const.} \tag{5.7}$$

Note that for $\beta = -1$, i.e., the flux entering at the bottom leaves at the top, the steady linear solution that corresponds to the problem with Dirichlet boundary conditions is recovered. Further, b_{diff} is quasi-steady in the sense that the shape of the parabolic profile remains constant. Integrating the diagnostic equation for the dry buoyancy b in eq. (2.19) yields that the volume-averaged buoyancy increases linearly in time with

$$\langle b(t)\rangle_{x,z} = \frac{B_0}{H}(1+\beta)t \tag{5.8}$$

Hence, in the turbulent case, the fluid warms linearly with increasing time. This model resembles a two-dimensional CBL in several terms. Firstly, it captures daytime conditions over land, where a fluid layer becomes buoyant due to heating from the warm land surface with a surface flux $B_0 > 0$. The resulting turbulent fluid motion creates a mixed layer in the bulk of the domain that resembles the vertical homogeneity of the CBL bulk. Further, the effect of warm entrained air is modeled by a $B_1 < 0$. Therefore, the case $\beta > 0$ is of main interest here. Typical atmospheric conditions correspond to β ranging approximately from 0.1 to 0.3 [16, 220]. As β increases further to a value of 0.4, the upper region of the convective layer is increasingly stabilized. Moreover, preliminary simulations indicate that at $\beta \approx 0.5$, the mean buoyancy difference $\langle \Delta b\rangle_{x,t}$ changes its sign, which is in line with different dynamics. Therefore this section will consider $\beta \in \{0.1, 0.2, 0.3, 0.4\}$, including the edge case of $\beta = 0.4$. Note that this Rayleigh-Bénard framework does not allow the convection cell height H to change over time, as would be the case for a CBL growing into a free atmosphere (see again Fig. 5.1). This case will be considered in 5.2, where a variable CBL thickness $h(t)$ will be examined. For now, this RBC model of the CBL is an ideal testing bed for the current objective, which is to test the generalizability of the ROM-RC model. Specifically, the ROM-RC model will be employed to generalize from a training dataset with $\beta = 0.3$ to the dynamics of $\beta = 0.1, 0.2, 0.4$. Further, an alternative to the POD data compression technique will be introduced: a convolutional Autoencoder network.

5.1.2 Direct numerical simulation of a two-dimensional convective boundary layer

Direct numerical simulations at four different buoyancy-flux ratios β are conducted (see Table 5.1)[1]. The Prandtl and convective Rayleigh numbers are fixed at $\text{Pr} = 1$ and $\text{Ra}_c = 3 \cdot 10^8$. Again, an extended aspect ratio of $\Gamma = 24$ is considered. The two-dimensional Boussinesq equations were 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. More details of the numerical method can be found in Mellado and Ansorge [221]. The *tlab* software used to perform the simulations is freely available at [183]. As mentioned above, the buoyancy difference across the cell $\langle \Delta b\rangle_{x,t}$ is a major dependent variable. Moreover, the buoyancy difference can be expressed in terms of a Neumann Nusselt number

$$\text{Nu}_N = \frac{\Delta b_{\text{diff}}}{\langle \Delta b\rangle_{x,t}} = \frac{1-\beta}{2\kappa}\frac{B_0 H}{\langle \Delta b\rangle_{x,t}}. \tag{5.9}$$

After an initial transient, these quantities relax into a statistically stationary state, as can be seen in Fig. 5.2a) and b) for all four cases. Moreover, based on $\langle \Delta b\rangle_{x,t}$ the free-fall scales t_f and U_f in eqs. (2.16) and (2.14) can be computed to make the results comparable to the dry and moist RBC cases in Chapter 4. A snapshot of the normalized buoyancy

$$b_{\text{norm}} = \frac{b(x,z) - \langle b\rangle_{x,t}(z=1))}{\langle \Delta b\rangle_{x,t}} \tag{5.10}$$

[1]The DNS was part of [120] and was performed by the co-author Juan Pedro Mellado.

β	Γ	Ra_c	Pr	N_{FD}	$N_x \times N_z$	$t_{\mathrm{DNS}}/\langle t_f\rangle_t$	$\Delta t_{\mathrm{DNS}}/t_c$
0.1	24	3×10^8	1	2400×150	720×30	2500	$1.2-1.6\cdot 10^{-3}$
0.2	24	3×10^8	1	2400×150	720×30	2500	$1.2-1.6\cdot 10^{-3}$
0.3	24	3×10^8	1	2400×150	720×30	2500	$1.2-1.6\cdot 10^{-3}$
0.4	24	3×10^8	1	2400×150	720×30	2500	$1.2-1.6\cdot 10^{-3}$

β	Re	$\langle\Delta b\rangle_t$	$\langle t_f\rangle_t$	$\langle\mathrm{Nu}_N\rangle_t$	$\langle\mathrm{Ra}\rangle_t$	$\tau_{\mathrm{eddy}}/\langle t_f\rangle_t$
0.1	965	19.0 ± 0.4	0.23	15.9 ± 0.3	$8.5\cdot 10^6$	3.03
0.2	894	15.4 ± 0.3	0.26	17.4 ± 0.4	$6.9\cdot 10^6$	2.92
0.3	798	10.7 ± 0.4	0.31	21.9 ± 0.9	$4.8\cdot 10^6$	2.74
0.4	700	4.1 ± 0.51	0.5	49.8 ± 6.6	$1.84\cdot 10^6$	1.94

Table 5.1: **Simulation parameters** of the direct numerical simulation of the two-dimensional RBC model of a convective boundary layer. The table contains the aspect ratio Γ of the two-dimensional cell, the convective Rayleigh number Ra_c, as defined in eq. (5.5), the Prandtl number Pr, as well as the grid size of the finite-difference scheme N_{FD} and the buoyancy flux parameter β, as defined in (5.6). N_x, N_z denote the uniform grid onto which the DNS data was interpolated. The DNS covered a period of t_{DNS} with a time-stepping of Δt_{DNS}, which approximately gives $0.25\langle t_f\rangle_t$ in each case. The buoyancy difference $\langle\Delta b\rangle_{x,t}$, free fall time $\langle t_f\rangle_t$, Neumann Nusselt number $\langle\mathrm{Nu}_N\rangle_t$ (see eq. (5.9)), Dirichlet Rayleigh number $\langle\mathrm{Ra}\rangle_t$, largest eddy turnover time τ_{eddy} and Reynolds number Re (see eq. (2.23)) are diagnostic DNS variables that reach a statistically steady state. More details can also be found in [120].

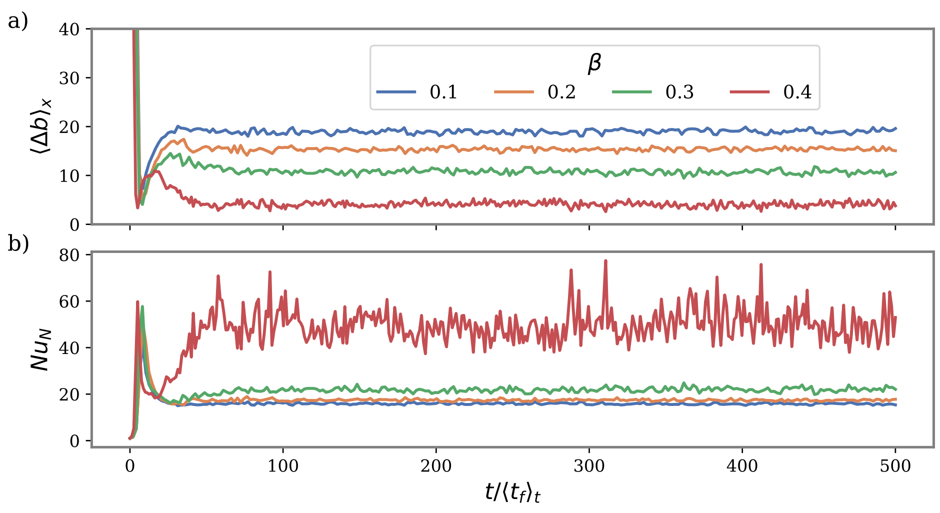

Figure 5.2: **Temporal variation** of (a) the buoyancy difference $\langle\Delta b(t)\rangle_x$, and (b) the Nusselt number Nu_N. See eqs. (5.1) and (5.9) for their definitions. Both quantities become statistically stationary after an initial transient. Note that both time axes are normalized by the time mean of the free fall time $t_f = \sqrt{H/\Delta b}$ in the statistical stationary regime.

and the vertical flux $u'_z b'(x,z)$ (definition follows) in the statistically stationary state is shown in figures (5.3) (a-d) and 5.3(e-h). It can be seen that an increasing buoyancy-flux parameter results in a growing thermal boundary layer near the top boundary. 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 in Fig 5.2b). Naturally, the structures in the buoyancy flux are also affected by the change in 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. 5.3a,e) and Figs. 5.3d,h). Finally, all three fields are decomposed into their vol-

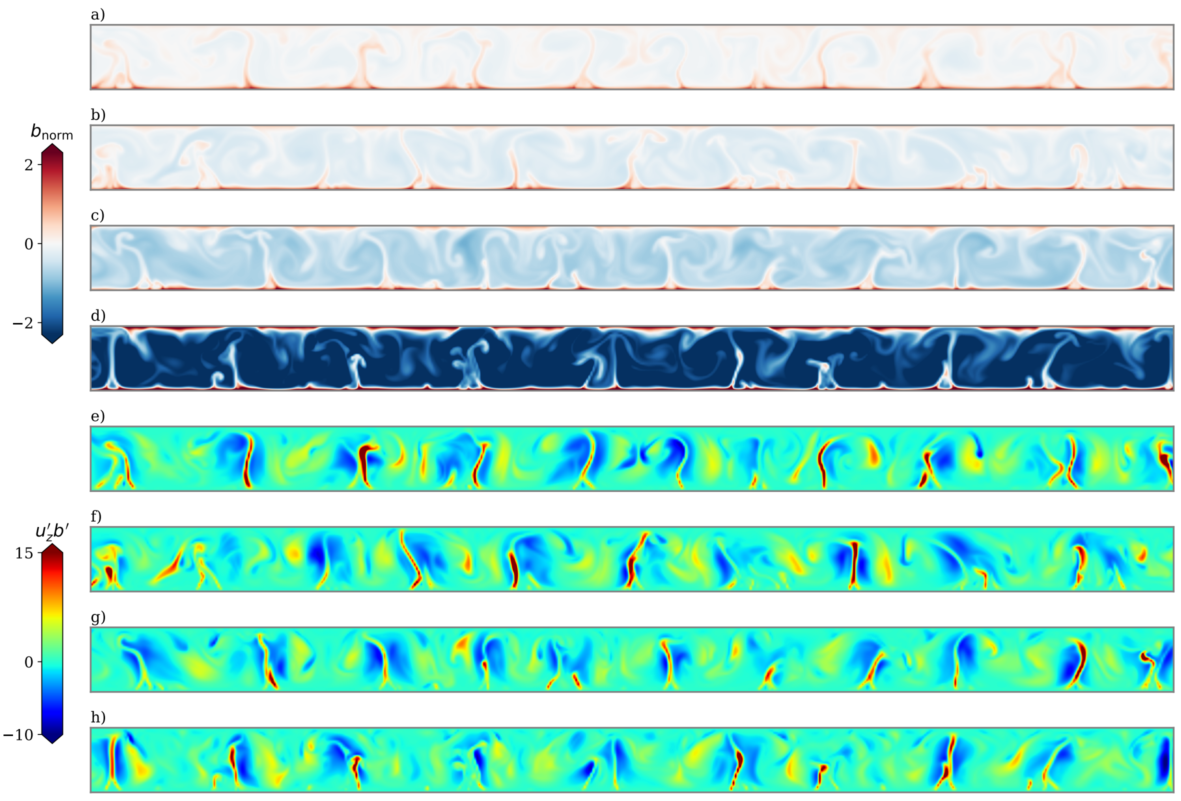

Figure 5.3: Instantaneous **DNS snapshot** of the normalized buoyancy field b_{norm} (a-d), as defined in (5.10), and turbulent buoyancy flux $u'_z b''$ (e-h) for the four β values 0.1 (a,e), 0.2 (b,f), 0.3 (c,g), 0.4 (d,h). Shown is a snapshot of the fields in the statistically stationary regime. The four different top boundary conditions ($\beta = 0.1, 0.2, 0.3, 0.4$) differ in the width of their top thermal boundary layer. Note that with increasing β, the range of b_{norm} increases.

ume mean $\langle\cdot\rangle_{x,z}$ and their fluctuations u'_x, u'_z, b'. They are given by

$$u_x(x,z,t) = \langle u_x(t)\rangle_{x,z} + u'_x(x,z,t)\,, \tag{5.11}$$

$$u_z(x,z,t) = \langle u_z(t)\rangle_{x,z} + u'_z(x,z,t)\,, \tag{5.12}$$

$$b(x,z,t) = \langle b(t)\rangle_{x,z} + b'(x,z,t)\,. \tag{5.13}$$

Note that $\langle b\rangle_{x,z}$ depends on time (see again (5.8)), 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 normalized buoyancy and the root-mean-square values of b' and u_z' are shown in 5.4a-c) for all four values of β. A reference profile of a simulation of an actual CBL, i.e., with no upper lid, is shown for reference. It can be seen that the RBC model of the CBL captures main characteristics of the CBL like the well-mixed bulk and the sharp gradients near the edges.
Additionally, the total buoyancy flux, comprising the convective and diffusive buoyancy transport,

$$F_b = \langle u_z' b' \rangle_{x,t} - \kappa \frac{\partial \langle b \rangle_{x,t}}{\partial z}, \tag{5.14}$$

normalized by its bottom value is shown in Fig. 5.4(d). For further processing steps in the next sections, the DNS data is interpolated to a uniform grid of size $N_x \times N_z = 720 \times 30$. Moreover, for the data reduction approach, the buoyancy field is further decomposed into

$$b'(x,z,t) = \langle b(x,z) \rangle_t + b''(x,z,t). \tag{5.15}$$

5.1.3 Dimensionality reduction via an Autoencoder network

In Chapter 4, the Proper Orthogonal Decomposition allowed for compression of the DNS data to a latent space of size $N_{\text{POD}} = 150$ with a loss of about 20% of the flow's variance. In the present case, however, preliminary studies show that a truncated POD model of each of the four flows would require up to $N_{\text{POD}} \simeq 700$ modes in order to represent 80% of the respective flow's variance.[2]. Hence, the linear POD approach is not able to reduce the DNS data to a sufficiently small representation without losing essential information on the flow. To circumvent this limitation, this section explores the second data reduction approach introduced in Section 2.3.2, namely a convolutional Autoencoder network.

For each of the β values, a respective neural network is trained to compress a snapshot of the turbulence fields $(u_x', u_z', b'')^T$ to a latent space $\mathbf{a} \in \mathbb{R}^{N_{\text{AE}}}$ with latent space dimension $N_{\text{AE}} = 300$. Information on the training procedure is shown in Tab. 5.2. All four networks share the same architecture, where the spatial information is processed and compressed throughout several convolutional and Max Pooling layers in the Encoder. The channel dimensions are subsequently increased from three to 16. Finally, the Decoder uses transposed convolutions and Upsampling layers that increase the spatial dimensions of the latent space and decrease the channel size to the original size of the input snapshot. More details about the Autoencoder architecture can be found in Appendix B.1.2. Table 5.2 lists the Autoencoder's hyperparameters that all four networks share. Each network was trained to minimize the mean-square error loss $\mathcal{L}_{\mathbf{w}_E,\mathbf{w}_D}$ in eq. (2.68). The optimization was done by use of the adaptive moment estimation method (Adam) proposed by Kingma *et al.* [61]. Fig. 5.5a) shows the training and validation loss of the case $\beta = 0.2$. After about 1500 epochs, the mean square loss does not improve further, and training ceases. Fig. 5.5b) compares the buoyancy flux $u_z' b''(x,z)$ of DNS to the, based on 300 features, AE reconstruction. Similar to the POD method, one finds that the loss of information manifests in a reduced magnitude of the reconstructed field. While small-scale features are neglected by the network, the overall characteristics of the flux agree well with the ground truth. Finally, the vertical profile

[2]Note that in ref. [120] the POD was taken over a time interval of $500t_f$ (2000 snapshots), for which 300 modes sufficed to capture 80% of the turbulent kinetic energy and buoyancy variance. In the preliminary study, the whole $2500t_f$ (10000 snapshots) were considered, indicating that the POD in [120] was not fully converged. See also Appendix B.1.1 for the POD spectra.

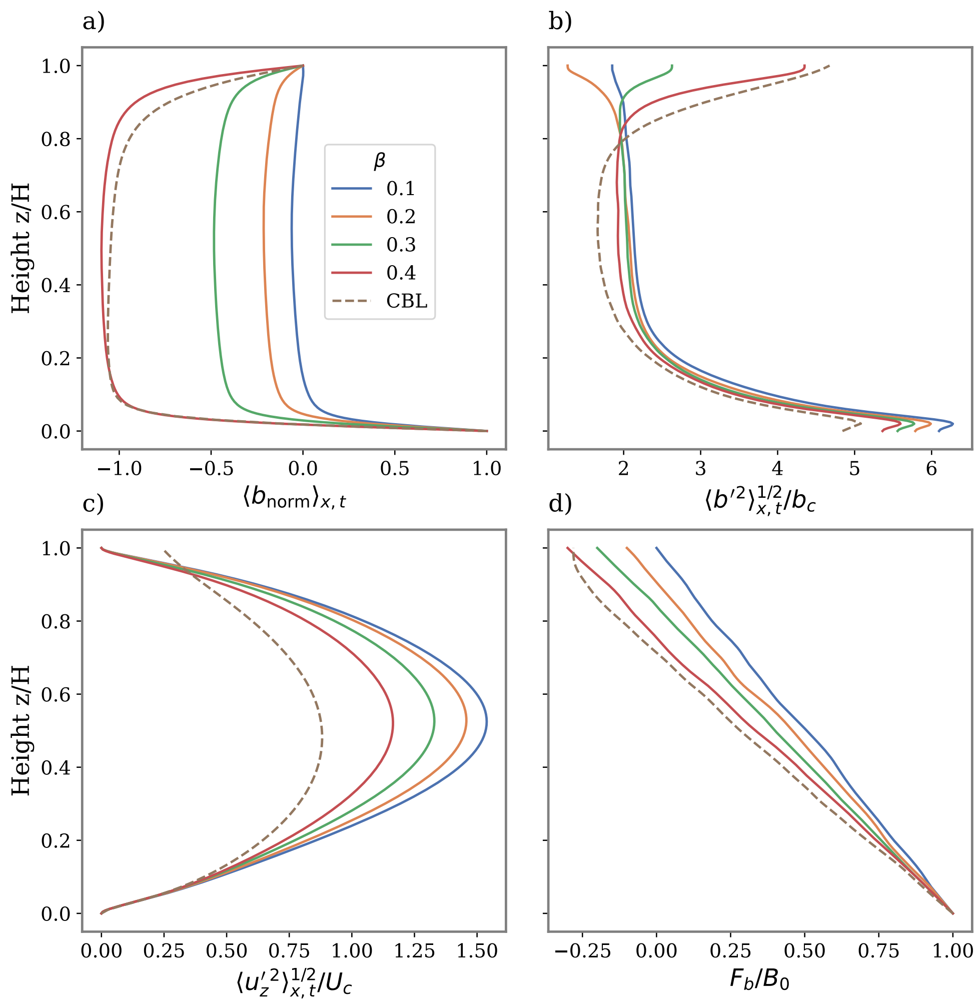

Figure 5.4: **Vertical profiles** of (a) the normalized buoyancy b_{norm}, (b) buoyancy fluctuations, (c) vertical velocity fluctuations, and (d) normalized total buoyancy flux $F_b(z)/F_b(z=0) = F_b(z)/B_0$. While the boundary conditions significantly affect the buoyancy and its fluctuations, the influence on the vertical velocity profiles is less important. The fluxes show linear variation across the cell. The brown dashed line shows the profiles of an actual CBL growing into a free atmosphere.

N_{AE}	η	batch size	$t^{\text{AE}}_{\text{train}}/\langle t_f \rangle_t$	$t^{\text{AE}}_{\text{val}}/\langle t_f \rangle_t$	$t^{\text{AE}}_{\text{test}}/\langle t_f \rangle_t$
300	10^{-5}	64	2000	250	250

Table 5.2: Details on the **Autoencoder's training procedure**. The latent dimension was N_{AE}. η denotes the learning rate, i.e., the update speed of the training procedure. See again eq. (1.7). The batch size refers to the number of training examples processed together in a single forward and backward pass. The network was trained on 8000 snapshots comprising a period of $t^{\text{AE}}_{\text{train}}$, validated on 1000 snapshots comprising $t^{\text{AE}}_{\text{val}}$ and finally tested on a testing set with the same length as the validation data. The training was stopped when the AE showed no improvement on the validation dataset after 50 epochs.

of the turbulent flux of the ROM reproduces the characteristic linear decay over the height of the convection domain. See Fig. 5.5c). However, the neural network introduces artifacts into the profile, as can be seen at $z/H \simeq 0.1$. Also, the boundary conditions are not exactly reproduced by this method. See, e.g., at the lower boundary. After the AE is trained,

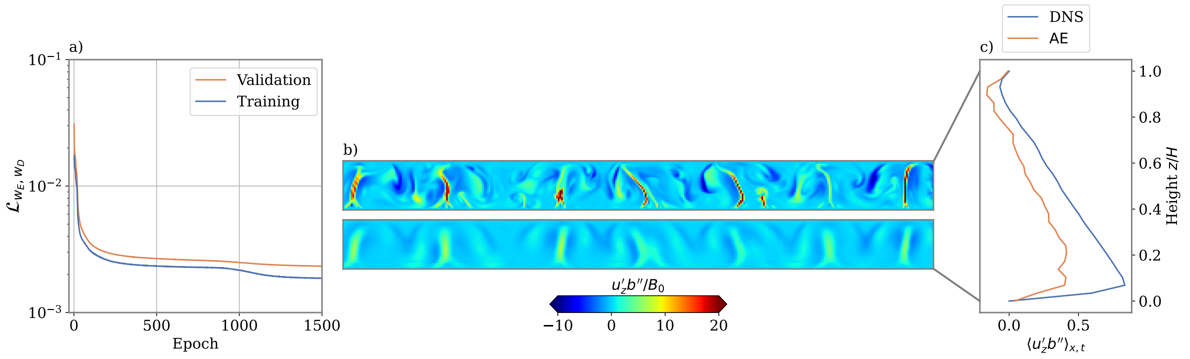

Figure 5.5: Dimensionality reduction via **convolutional Autoencoder network** for the case $\beta = 0.2$. The latent space dimension was $N_{\text{AE}} = 300$. (a) Training and validation loss throughout the training procedure. (b) Comparison between DNS (top) and the AE's reconstructed buoyancy flux $u_z'b''$ (bottom). Similar to the POD, the characteristics of the flux are reproduced, while small-scale features are not captured. For better visibility, only half of the horizontal domain is shown. (c) Lateral-time average $\langle\cdot\rangle_{x,t}$ profiles of the buoyancy flux for the DNS (blue) and Autoencoder (orange) case. Note the artifacts in the Autoencoder's profile near $z/H \simeq 0.1$.

the Encoder part can compute a low-dimensional representation of each DNS snapshot of $(u_x', u_z', b'')^T$. Therefore, the Encoder translates the convection dynamics to dynamics $\mathbf{a}(t)$ of the latent space. These are shown for $\beta = 0.2$ in Fig 5.6. Contrary to the POD time coefficients, the AE latent space does not exhibit scale separation among its N_{AE} modes. Moreover, while the POD time coefficients are related to their corresponding spatial modes, the $\mathbf{a}$ of the Autoencoder are non-trivially linked to the trained weights $\mathbf{w}_E$, $\mathbf{w}_D$ of the Encoder and Decoder network. To get an idea of how the network processes the physical fields, the reader is referred to Appendix B.1.3.

5.1.4 AE-RC model of a two-dimensional convective boundary layer model

In the following, it will be explored whether the ESN can be used 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. In this way, this procedure will probe the generalization property of the Echo State Network.

5.1.4.1 Generalization procedure

The aforementioned subject is also connected to a transfer of the learned parameters from one task to a similar one which is known as Transfer Learning [222]. 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 [198]. Here, a different approach, sketched in Fig. 5.7, will be taken. A reservoir is trained with

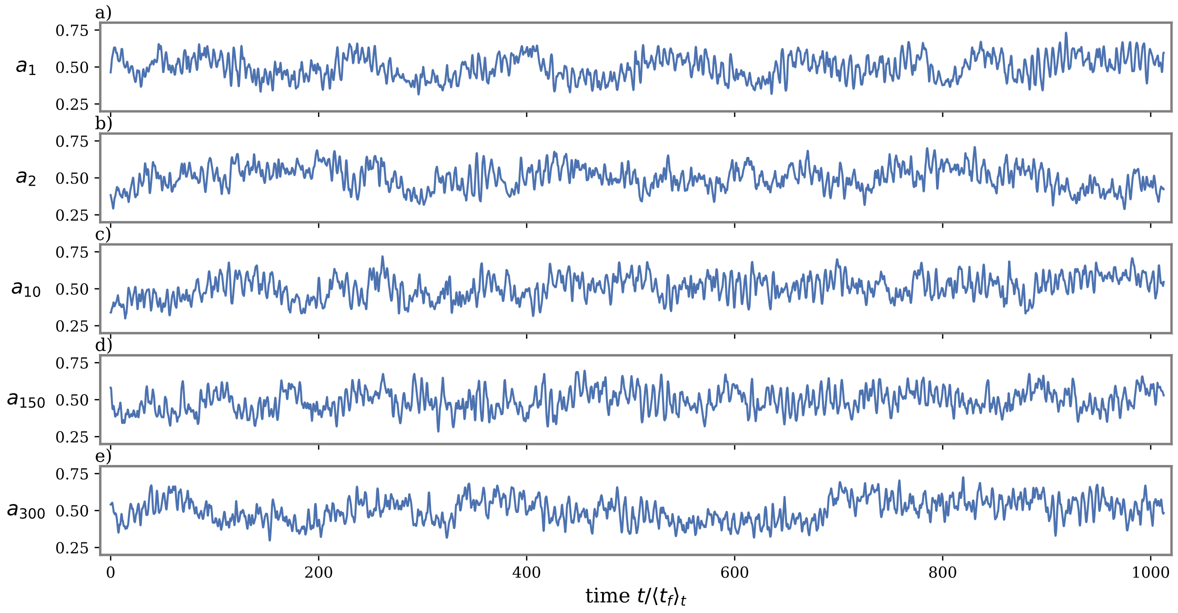

Figure 5.6: **Encoding space dynamics** $\mathbf{a}(t)$, i.e., the output of the Encoder network for a given sequence of snapshots of $(u'_x, u'_z, b'')^T$. The latent space dimension is $N_{\mathrm{AE}} = 300$. Shown is the $\beta = 0.2$ case. Note that, contrary to the POD, the reduced space does not exhibit a scale separation among the different mode numbers. See again figures 4.3 and 4.12.

the reduced data of one case of buoyancy boundary conditions at $z = H$, namely $\beta = 0.3$. An intermediate reservoir washout phase clears the reservoir memory of recent $\beta = 0.3$ information and leads to a transition of the reservoir state to one of three different and unseen convection flows with buoyancy flux parameters $\beta = 0.1$, 0.2 or 0.4. Finally, the trained network is used for the prediction of the dynamics and the statistical properties of the unseen regimes. Specifically, this means that once the ESN has learned to process the data in the latent space for the case of $\beta = 0.3$, it is exposed to unseen data of the three CBL model cases, $\beta = 0.1$, 0.2, and 0.4 without further training adjustments. For this, the reservoir state $\mathbf{r}$ is re-initialized with unseen latent space data. In detail, the reservoir input is $\mathbf{a}^{\beta=0.1}$, $\mathbf{a}^{\beta=0.2}$ or $\mathbf{a}^{\beta=0.4}$ for a period of $t_{\mathrm{trans}} = 25\langle t_f \rangle_t$ and the reservoir memory evolves according to eq. (2.42). Note that this transition time corresponds to about $8 - 12.5$ turnover times of the respective largest eddies. See also Fig. 5.8 where these crossover dynamics are displayed for one example. Note how the training and prediction dynamics of the reservoir state differ after this procedure. Therefore, with this *washout phase*, this procedure intends to transition to the run with a new β. Starting from this reservoir state, the ESN will autonomously predict future time steps with its output weights learned for $\beta = 0.3$. These predictions are validated by a direct comparison with $\mathbf{a}^{\beta=0.1}(n)$, $\mathbf{a}^{\beta=0.2}(n)$ and $\mathbf{a}^{\beta=0.4}(n)$ respectively. Moreover, their reconstructions will be examined in terms of spatial and statistical accordance. The next section will inspect the hyperparameter dependence of the error in the vertical profiles, i.e., the $\mathcal{E}_{\mathrm{NARE}}$. Further, the best ESN settings Ξ_{r} will be identified.

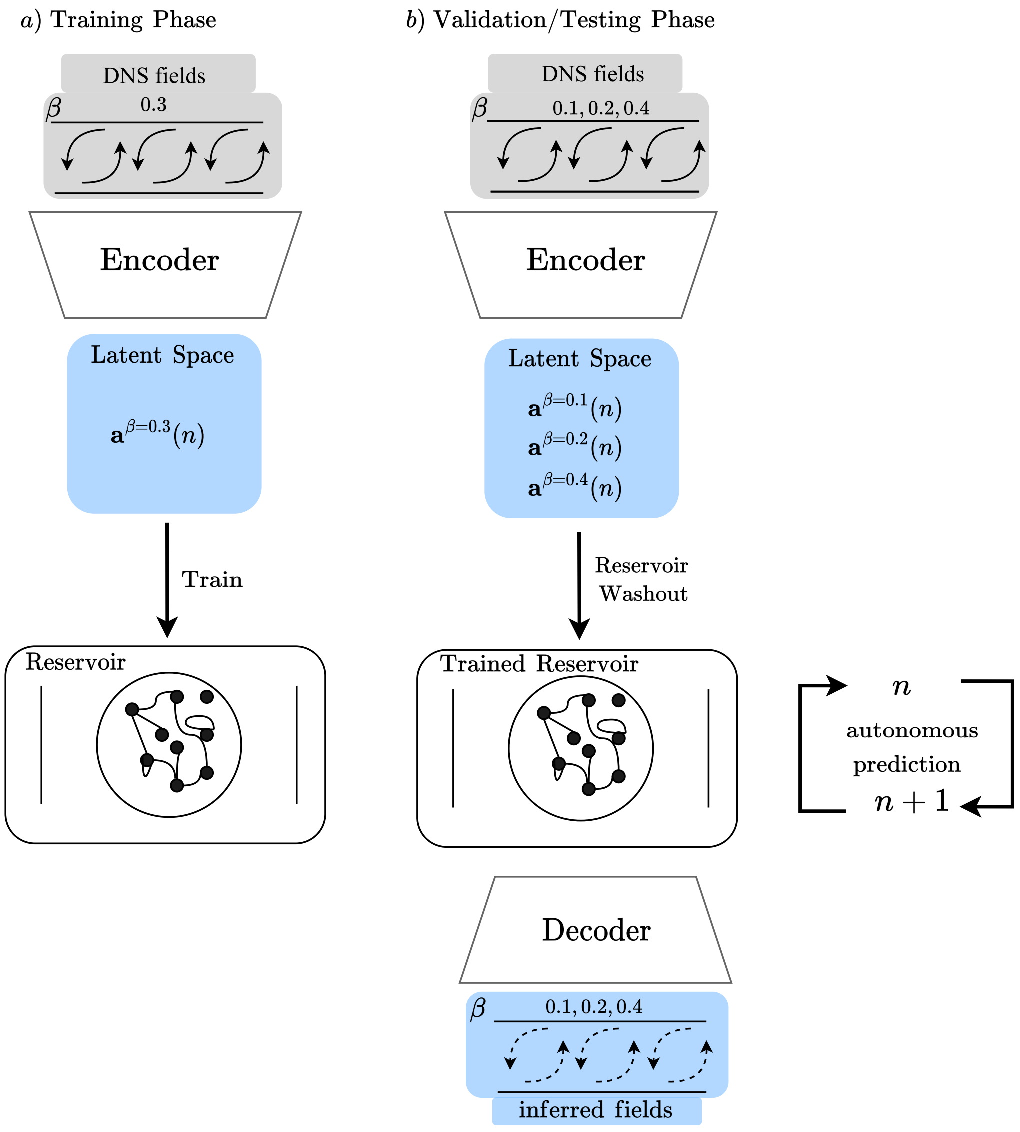

Figure 5.7: Sketch of the **ESN generalization procedure**. (a) During the training phase, snapshots of the simulation data for $\beta = 0.3$ are encoded into the latent space via the reduction by AE (denoted as $\mathbf{a}^{\beta=0.3}$). A reservoir is subsequently trained with the latent space. The network learns the dynamics; the optimal output weights are obtained. (b) In the closed-loop prediction (validation/testing) phase, the reservoir is then used to infer the dynamics of the target latent spaces at $\beta = 0.1, 0.2, 0.4$ and predicts $\mathbf{a}^{\beta\neq 0.3}$. Snapshots of the convection flow can then be reconstructed and validated by the corresponding Decoder to obtain fully resolved fields for the cases of $\beta = 0.1$, 0.2, and 0.4.

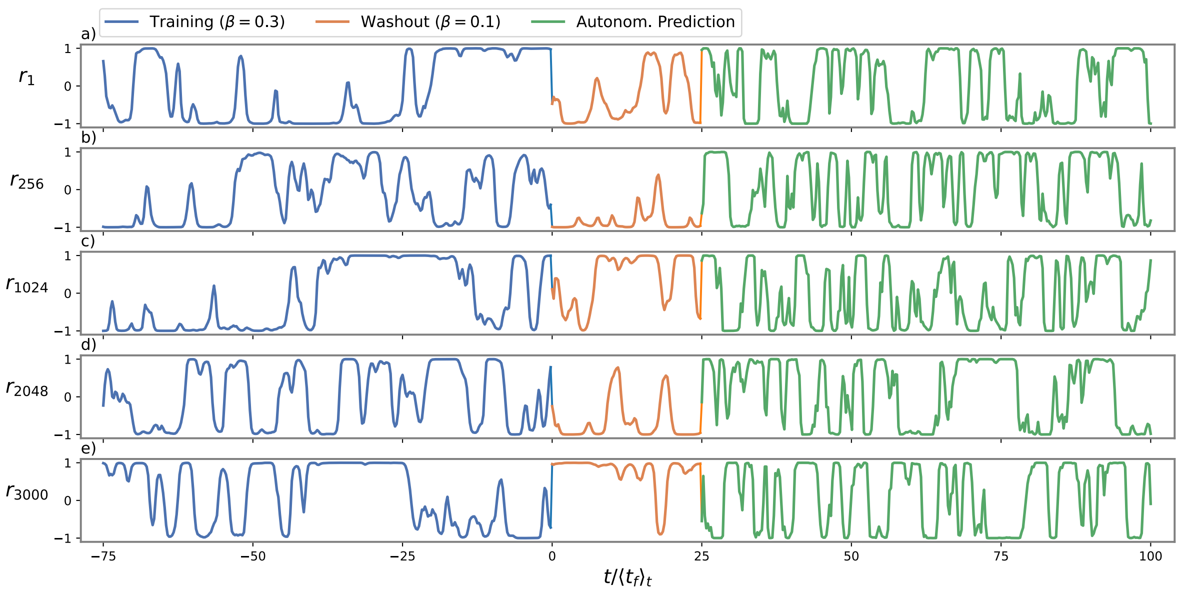

Figure 5.8: **Generalization procedure**: Time evolution of individual components of the reservoir state $\mathbf{r}$ for inference of $\beta = 0.1$ in the AE-RC algorithm. During training, the reservoir is exposed to $\mathbf{a}^{\beta=0.3}$. After the training phase, W^{out} is held fixed, and the memory on the training inputs is cleared, while inputs of $\mathbf{a}^{\beta=0.1}$ are given to the reservoir for a period of $t_{\text{trans}} = 25\langle t_f\rangle_t$. Simultaneously, the reservoir builds up a memory of the unseen $\beta = 0.1$ case. Finally, the newly initialized state and an initial input of $\mathbf{a}^{\beta=0.1}$ are used to start the autonomous prediction phase. Line styles for the different phases are given above. Note the different reservoir dynamics during training and prediction.

γ_{r}	ϱ_{r}	λ_{r}	Reservoir realizations
$[0.1, 1.0, 0.1]$	$[0, 2.0, 0.2]$	$\{0.5, 5.0\}$	50

$t_{\text{train}}/\langle t_f\rangle_t$	$t_{\text{trans}}/\langle t_f\rangle_t$	$t_{\text{val}}/\langle t_f\rangle_t$	$t_{\text{test}}/\langle t_f\rangle_t$
1000	25	250	250

Table 5.3: **ESN hyperparameter grid search** procedure for the closed-loop ESN trained on the $\beta = 0.3$ DNS data. The transition to the unseen $\beta = 0.1, 0.2, 0.4$ case is done via the procedure shown in Fig. 5.7. Square brackets indicate the range of the searched hyperparameter: [start, end, stepsize]. Braces indicate an array of values that were used. t_{train}, t_{val}, t_{test} are the length of the training, validation, and testing interval, respectively. The first $12.5t_f$ out of the training time was used for initializing the internal reservoir state. The reservoir density and input scaling were fixed at $D_{\text{r}} = 0.20$ and $\sigma_{\text{r}} = 1.00$. Moreover, the reservoir size was set to $N_{\text{r}} = 3000$. The total number of different reservoir settings is $10 \times 11 \times 2 = 220$, while a total of $50 \times 220 = 1.1 \cdot 10^4$ reservoirs were run. The total grid search runtime for one target β was 57h.

5.1.4.2 ESN hyperparameter search

The larger latent dimension, in contrast to the previous chapter, results in a reservoir with a greater input dimension. Consequently, a larger reservoir is required as well. Moreover, exhaustive grid searches, as performed in 4.1.3.1 and 4.2.4.1, will now become more computationally intensive. Hence, for the current task, the hyperparameter search space will be limited to the leaking rate γ_r, the spectral radius ϱ_r and the regression parameter λ_r in order to keep the computation time of the grid search similar to the ones of the last chapter. The reservoir size and density are fixed at $N_\mathrm{r} = 3000$, $D_\mathrm{r} = 0.2$, and a grid search for the three generalization tasks is computed. Furthermore, the input scaling is set to $\sigma_\mathrm{r} = 1.0$. In alignment with prior methodology, an average NARE will be established as a metric for measuring errors. Here, the natural choice is

$$\begin{aligned}\overline{\mathcal{E}_\mathrm{NARE}} = \frac{1}{4}\Big[&\mathcal{E}_\mathrm{NARE}\left\langle {u'_x}^2\right\rangle_{x,t}^{1/2} + \mathcal{E}_\mathrm{NARE}\left\langle {u'_z}^2\right\rangle_{x,t}^{1/2} \\ &+ \mathcal{E}_\mathrm{NARE}\left\langle b''^2\right\rangle_{x,t}^{1/2} + \mathcal{E}_\mathrm{NARE}\left\langle u'_z b''\rangle_{x,t}\right)\Big]\,, \end{aligned} \tag{5.16}$$

i.e., involving the three physical fields and the associated buoyancy flux. The resulting validation NARE in dependence of the ESN hyperparameters for each target β is shown in Fig. 5.9. The first thing that is striking is that the ESN, trained on $\beta = 0.3$, can generalize more easily towards $\beta = 0.4$, as the $\overline{\mathcal{E}_\mathrm{NARE}}$ values are all much below the ones for $\beta = 0.1$ and 0.2. This is somewhat unexpected, as the edge case $\beta = 0.4$ exhibits strongly fluctuating properties. See e.g. the Nusselt number in Fig. 5.2b). Moreover, $\beta = 0.1, 0.2$ both exhibit a similar error dependence on the spectral radius, leaking rate, and regression parameter. At sufficiently low leaking rate values, the error becomes almost independent of the spectral radius. Interestingly, for $\beta = 0.1$, the optimal ESN possesses a spectral radius $\varrho_\mathrm{r} = 0.0$, effectively deactivating its nonlinear memory. This might indicate that $\beta = 0.1$ is too hard to infer based on the learned weights for $\beta = 0.3$. The optimal hyperparameter set for each target β is listed in Tab. 5.4. The corresponding normalized average relative errors are listed in 5.5.

β	γ_r	ϱ_r	λ_r	N_r	D_r	σ_r
0.1	0.1	0.0	0.5	3000	0.20	1.0
0.2	0.1	0.2	0.5	3000	0.20	1.0
0.4	0.1	0.4	5.0	3000	0.20	1.0

Table 5.4: **Optimal ESN hyperparameter set** Ξ_r of each generalization procedure according to the ESN hyperparameter grid search. The optimal setting is marked with a green star symbol in the error landscape in Fig. 5.9. The corresponding NARE values are listed in Tab.5.5.

5.1.4.3 Results of the AE-RC model

By decoding the inferred latent spaces in the testing phase using eq. (2.67) the three physical fields are reconstructed. Figure 5.10 shows instantaneous snapshots of the local turbulent kinetic energy

$$E_\mathrm{kin}(x,z,t) = 0.5({u'_x}^2(x,z,t) + {u'_z}^2(x,z,t)) \tag{5.17}$$

turbulent flux $u'_z b''(x,z)$, and buoyancy fluctuations $b''(x,z)$ of inferred fields (ESN) and ground truth (AE). Here, the predicted and true fields are almost indistinguishable in terms

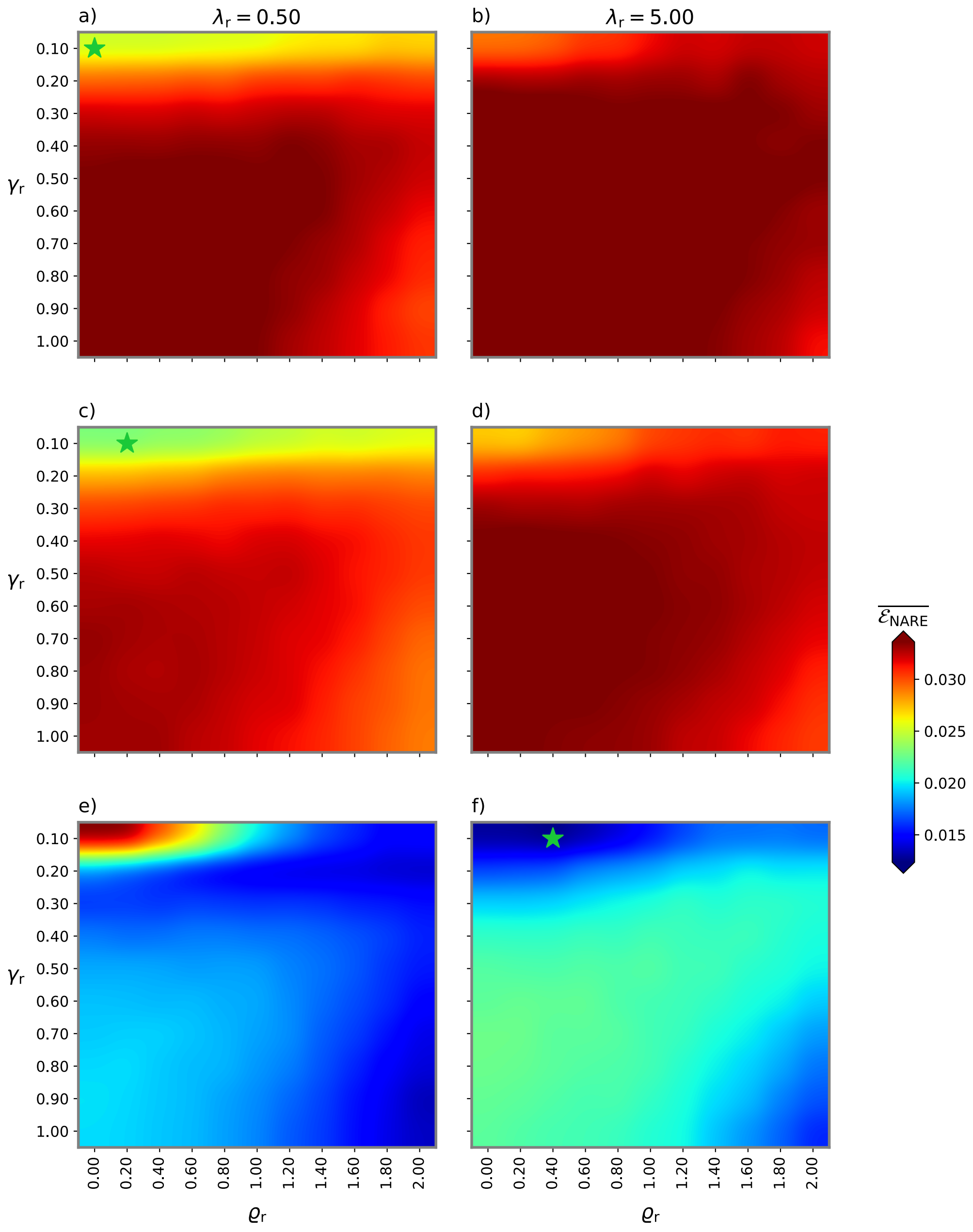

Figure 5.9: **Results of the grid search** process of the closed-loop ESN run trained for the generalization procedure described in 5.1.4.1. The ESN was trained on data from the $\beta = 0.3$ case. Shown is the arithmetic average normalized average relative error $\overline{\mathcal{E}_{\mathrm{NARE}}}$ involving the fields u'_x, u'_z, b'' and $u'_z b''$, as defined in eq. (5.16). The target β values were 0.1 (a,b), 0.2 (c,d), and 0.4 (e,f). The hyperparameter landscape consists of leaking rate γ_{r}, spectral radius ϱ_{r}, and the regression parameter λ_{r}. Reservoir size, density and input scaling were fixed at $N_{\mathrm{r}} = 3000$, $D_{\mathrm{r}} = 0.2$ and $\sigma_{\mathrm{r}} = 1.0$. Moreover, out of the 50 reservoir realizations, the median $\overline{\mathcal{E}_{\mathrm{NARE}}}$ value is shown. The results were computed on the validation data set. The green star marker indicates the lowest error, and hence the best hyperparameter set Ξ_{r} to each target β value.

β	$\mathcal{E}_{\text{NARE}}\left(\langle\langle u_x'^2\rangle_{x,t}^{1/2}\right)$	$\mathcal{E}_{\text{NARE}}\left(\langle u_z'^2\rangle_{x,t}^{1/2}\right)$	$\mathcal{E}_{\text{NARE}}\left(\langle b''^2\rangle_{x,t}^{1/2}\right)$	$\mathcal{E}_{\text{NARE}}\left(\langle u_z' b''\rangle_{x,t}\right)$
0.1	$3.60 \cdot 10^{-2}$	$3.62 \cdot 10^{-2}$	$8.20 \cdot 10^{-3}$	$1.99 \cdot 10^{-2}$
0.2	$3.34 \cdot 10^{-2}$	$2.27 \cdot 10^{-2}$	$1.11 \cdot 10^{-2}$	$2.41 \cdot 10^{-2}$
0.4	$1.63 \cdot 10^{-2}$	$7.02 \cdot 10^{-3}$	$7.15 \cdot 10^{-3}$	$1.89 \cdot 10^{-2}$

β	$\overline{\mathcal{E}_{\text{NARE}}}$
0.1	$2.50 \cdot 10^{-2}$
0.2	$2.29 \cdot 10^{-2}$
0.4	$1.23 \cdot 10^{-2}$

Table 5.5: Normalized average relative errors of the optimal ESN setting Ξ_{r} listed in Tab. 5.3. Listed are the individual NARE values of horizontal and vertical velocities, buoyancy fluctuations, and convective buoyancy flux. Moreover, the arithmetic mean of all four NAREs $\overline{\mathcal{E}_{\text{NARE}}}$ is also shown. This measure is also depicted in Fig. 5.9.

of their features. Roll patterns, as well as thermal plumes detaching from the bottom, are reproduced very naturally. Moreover, the fine features of the buoyancy flux are also inferred to a reasonable extent. The corresponding line-time averaged profiles of the physical fields are illustrated in 5.11. Different from the linear POD method, the AE is trained by a gradient descent procedure, which introduces artifacts in the statistical profiles of the ground truth (solid lines). The loss of information in the Encoder-Decoder structure thus impacts the statistical features of the reconstructed flow. This is especially the case for second-order statistics like $\langle E_{\text{kin}}(z)\rangle_{x,t}$ and $\langle u_z' b''(z)\rangle_{x,t}$. Naturally, this behavior translates to the inferred profiles. It is possible that this error can be reduced by introducing an additional term to the loss function of the AE that penalizes large deviations from the mean profiles. This is not applied here. Nevertheless, the differences are acceptable, as the asymmetry and shape of the true profiles are retained. The inferred vertical profiles of $\beta = 0.1$ and 0.2 show higher discrepancies in the turbulent kinetic energy and vertical velocity. However, the root-mean-square of buoyancy, as well as the profile of the buoyancy flux, are reproduced nicely. As mentioned above, the $\beta = 0.4$ case exhibits the lowest average NARE and, hence, agrees very well with its ground truth. However, small errors are found close to the top boundary in the root-mean-square profile of the buoyancy. Finally, Fig. 5.12 shows the inferred latent space variables $\mathbf{a}(t)$ and the corresponding Fourier power spectra for the $\beta = 0.1$ case. It can be seen that the generalization procedure does capture the low-frequency components of the target parameter dynamics. However the spectras' tails at higher frequencies is not reproduced. Similar behavior can be observed for the other two cases $\beta = 0.2$ and 0.4. See Appendix B.1.4 for the corresponding results. This suggests that the presented generalization procedure is able to reproduce spatial and statistical features of the target flow but fails to capture their temporal aspects.

While the present two-dimensional CBL model shares several characteristics with a convective boundary layer, it still lacks the transient growth behavior of an actual CBL that grows throughout the diurnal cycle. However, this behavior will become challenging to reproduce using the current combination of data compression technique and Reservoir Computing model. This is because all considered convection models so far entered a state of statistical homogeneity, where spatial and temporal patterns could easily be extracted by means of a POD or Autoencoder network. However, in reality, these patterns change dras-

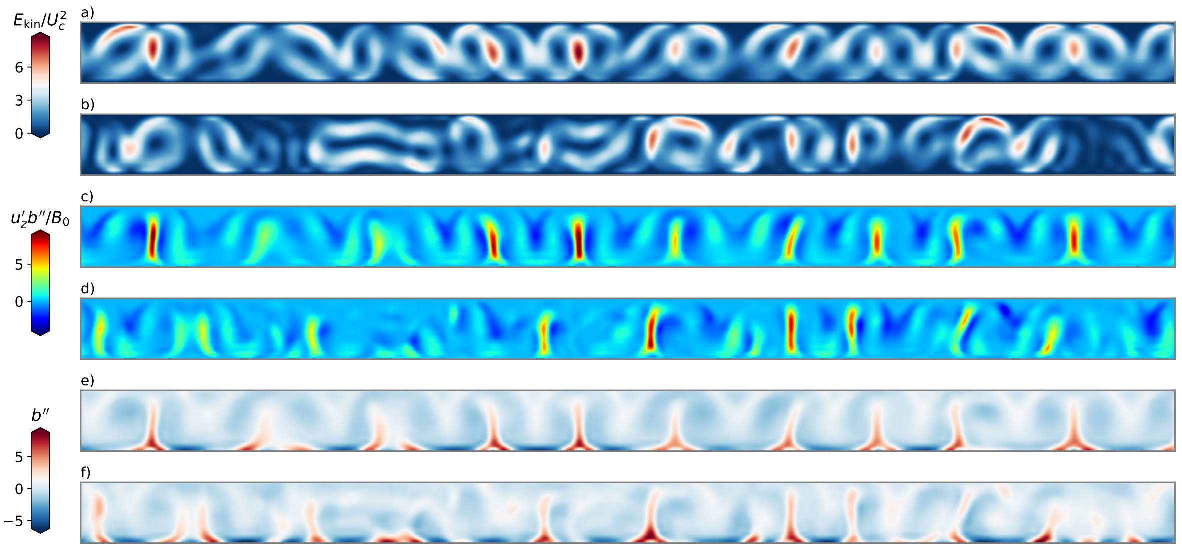

Figure 5.10: Inferred fields for $\beta = 0.1$. Instantaneous snapshots of the turbulent kinetic energy (a,b), the buoyancy flux $u_z'b''$ (c,d), and the buoyancy fluctuations b''(e,f) at $t = 375\langle t_f \rangle_t$ in the middle of the testing phase. The ground truth (AE) reconstructions are (a,c,e), and the ESN predictions are (b,d,f). For better visibility, only 75% of the horizontal domain is shown.

tically over time as the CBL thickness increases during the day. The following section will, therefore, introduce an alternative approach to capture this transient behavior. It will make use of a Generative Adversarial Network and a preceding data augmentation step that is motivated by the physical scaling laws.

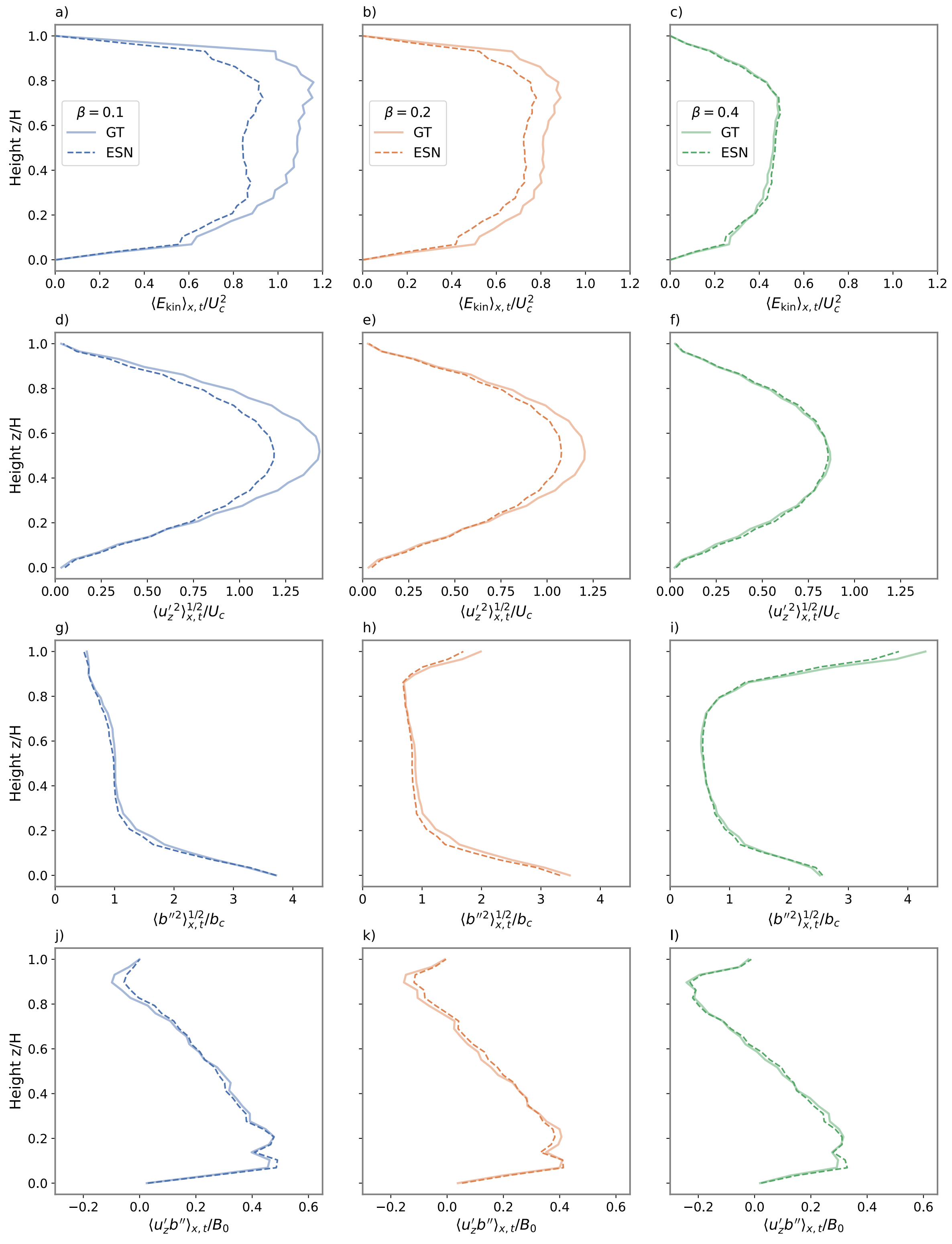

Figure 5.11: **Line-time average profiles** of the optimal ESNs listed in Tab. 5.4. All results were trained on the case $\beta = 0.3$ Shown are the profiles of (a-c) turbulent kinetic energy, (d-f) root-mean-square of the vertical velocity fluctuations, (g-i) root-mean-square buoyancy fluctuations and (j-l) convective buoyancy flux. The Autoencoder introduces artifacts in the statistical profiles (solid lines) of $\beta = 0.1$ (blue) in the first column, $\beta = 0.2$ (orange) in the second column, and $\beta = 0.4$ (green) in the third column. The dotted lines mark the ESN predictions.

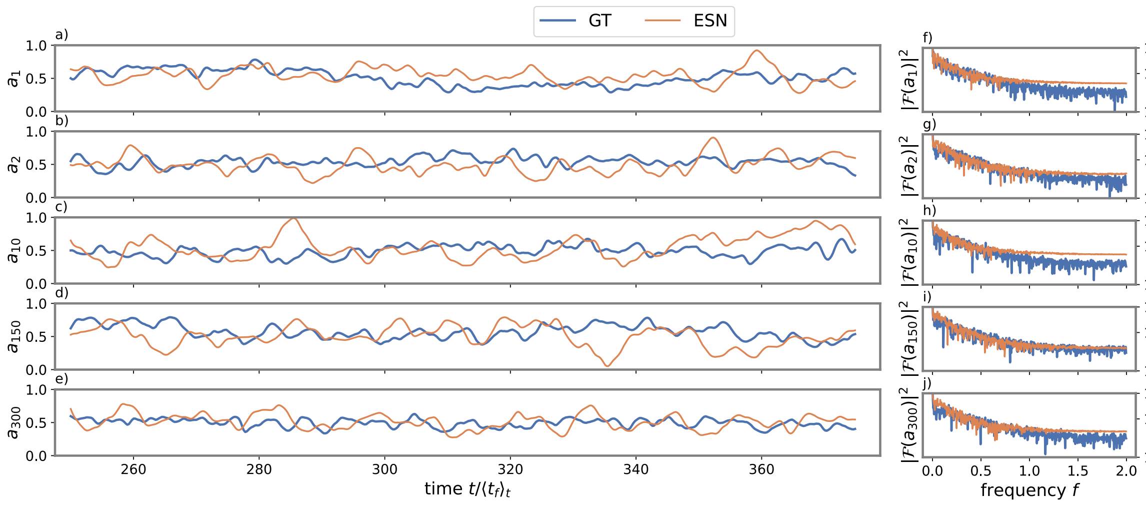

Figure 5.12: **Inferred time series** of selected AE latent space variables for the case of $\beta = 0.1$. (a-e) The ESN output (orange) is compared to the ground truth (blue). (f-j) The corresponding Fourier power spectrum is shown to the right. It can be seen that the inferred dynamics do not capture the spectrums tails at high frequencies. The inferred signals for the other two cases are shown in Appendix B.1.4

5.2 A physics-informed generative neural network approach to the parameterization of the CBL

In this section, both the ROM-RC approach and the Rayleigh-Bénard model will be abandoned. Instead, a convective boundary layer growing into a stably stratified atmosphere will be considered. Moreover, the application of a generative neural network to DNS data of the aforementioned system will be explored. The following section will provide an introduction to the specifics and the physical context of a CBL under dry and shear-free conditions. Subsequently, the findings from direct numerical simulations conducted on the CBL will be examined. Following that, information regarding the generative neural network and its augmented training dataset will be presented. Lastly, the performance of the trained machine learning model in terms of its ability to capture the transient spatial and statistical characteristics of convective flow will be evaluated in three critical zones: the surface, mixed layer, and entrainment layer. To conclude, the results are compared to a conventional eddy-diffusivity mass-flux parameterization scheme commonly used in General Circulation Models.

5.2.1 A dry convective boundary layer growing into a stratified free atmosphere

A dry convective boundary layer can be described by the dimensional form of the three-dimensional Boussinesq equations (2.7 - 2.9) for a velocity field $(u_x, u_y, u_z)^T$ and buoyancy field b. The boundary conditions are no-slip for the velocity, while a constant buoyancy flux B_0 is used at the bottom. At the top, a constant buoyancy flux is assumed as well. Contrary to the previously studied RBC setups, a CBL has no top lid that confines the fluid motion. Numerically, this can be implemented by placing the top boundary of the computational domain far enough from the bottom boundary, s.t., the boundary layer at the bottom plate is not affected by the actual top lid. In this way, the height of the computational domain becomes unimportant for a CBL simulation. Moreover, the fluid above the convective boundary layer is stably stratified due to the warm, non-turbulent atmosphere (see Fig. 5.1 again) sitting on top of the CBL. This capping inversion hinders the growth of the convective layer and leads to the formation of an entrainment layer between the mixed CBL and non-turbulent fluid. In practice, this stratification is implemented by initializing the buoyancy with a corresponding vertical background profile

$$b_{\mathrm{bg}} = N_0^2 z + f(z) \tag{5.18}$$

where N_0 is again the Brunt-Väisälä buoyancy frequency as defined in eq. (4.7) and $f(z)$ only non-zero near the bottom to assure the bottom Neumann boundary condition. In this way, convective motion is initiated near the warm surface. The thickness $h(t)$ of the CBL grows in time as strong thermals shoot into the non-turbulent atmosphere and entrain warm air into the CBL. Here, the height of the CBL is defined in terms of the encroachment height [223, 224]

$$h = \left[2N_0^{-2} \int_0^{z_\infty} (\langle b \rangle_{x,y} - b_{\mathrm{bg}}) dz \right]^{1/2}, \tag{5.19}$$

where the upper integration limit denotes the height of the computational domain[3]. Therefore, the CBL thickness is defined as the height, where the mean lateral buoyancy $\langle b \rangle_{x,y}$

[3]Note that previously z_∞ was also the height of the convection layer, denoted as H.

changes from vertically constant, well mixed inside the turbulent CBL, to the linear background profile b_{bg} in the capping free atmosphere. Another effect is that this stratification hinders the growth of the CBL thickness to the point that the characteristic time for changes of mean properties is small compared to the large eddy turnover time, and a statistically quasi-steady state is observed. It is this phase that will be of interest for the present generative ML application. Furthermore, the thickness of the entrainment layer can be characterized by the reference Ozmidov scale

$$L_0 = \left(\frac{B_0}{N_0^3}\right)^{1/2}. \tag{5.20}$$

Note that L_0 is constant in time, while h increases over time. Consider the mean buoyancy equation with statistical homogeneity in the horizontal directions. Then starting from (2.9) it can be shown that (5.19) yields

$$\frac{h(t)}{L_0} = \left[2N_0(t-t_0)\left(1+\frac{1}{\mathrm{Pr}\mathrm{Re}_0}\right)\right]^{1/2}, \tag{5.21}$$

where t_0 is an integration constant, and the buoyancy Reynolds number Re_0 is defined as

$$\mathrm{Re}_0 = \frac{B_0}{\nu N_0^2}. \tag{5.22}$$

The convective Rayleigh number (5.5), now defined via $h(t)$ instead of H, becomes time-dependent as well. As mentioned above, here, the CBL is considered without phase change. In this way, the present study can be seen as a proof of concept, which tries to ascertain the benefits and trade-offs of the generative ML approach. However, the following approach can readily be extended to the moist case. Finally, the buoyancy flux is the central transport measure. It couples buoyancy and vertical velocity fluctuations, i.e., the deviations from the respective vertical mean profiles of both fields defined as

$$u_x(x,y,z,t) = \langle u_x(z)\rangle_{x,y} + u'_x(x,y,z,t), \tag{5.23}$$
$$u_y(x,y,z,t) = \langle u_y(z)\rangle_{x,y} + u'_y(x,y,z,t), \tag{5.24}$$
$$u_z(x,y,z,t) = \langle u_z(z)\rangle_{x,y} + u'_z(x,y,z,t), \tag{5.25}$$
$$b(x,y,z,t) = \langle b(z)\rangle_{x,y} + b'(x,y,z,t), \tag{5.26}$$

where $\langle\cdot\rangle_{x,y}$ denotes the average over the horizontal directions. As touched on in the introduction, a large-scale model that does not resolve the fluctuations often employs an eddy-diffusivity mass-flux model for such quantities. This scheme models the buoyancy flux as,

$$\rho_0\langle {u_z}' b'(z)\rangle_{x,y} = -\rho_0\kappa_t\frac{\partial\langle b\rangle_{x,y}}{\partial z} + M_u\left(\langle b(z)\rangle_u - \langle b(z)\rangle_{x,y}\right), \tag{5.27}$$

where ρ_0 is the constant reference density in the convective boundary layer, and M_u is the mass flux parameter. Here, $\langle\cdot\rangle_{x,y}$ are horizontal plane averages. The first term in (5.27) is the standard Boussinesq term for a turbulent stress, which contains the turbulent diffusivity κ_t. The second term on the right-hand side stands for the mass-flux parametrization, which has to be included when the mean buoyancy gradient vanishes for the well-mixed layer, as is the case here. A detailed derivation of the mass flux parameterization can be found in Appendix B.2.1.

5.2.2 Direct numerical simulation of a convective boundary layer

A DNS of a three-dimensional CBL in a computational domain $x, y \in [-12.5, 12.5], z \in [0, 3]$ was conducted, again employing the *tlab* code [183][4]. Discretization in space is performed

[4]The DNS was part of [219] and was performed by the co-author Juan Pedro Mellado.

using sixth-order spectral-like compact finite differences on a structured Cartesian grid [221]. A low-storage fourth-order accurate Runge–Kutta scheme is used for time stepping. The discrete solenoidal constraint is satisfied to machine accuracy using a Fourier decomposition along the periodic horizontal planes in x-y directions and a factorization of the resulting set of equations along the vertical coordinate z. The corresponding simulation parameters are listed in Tab. 5.6. The final DNS snapshots are collected at three heights: $z/h = 0.2$ close to the surface layer, $z/h = 0.5$ in the mixed bulk, and the entrainment zone at $z/h = 1.0$. The horizontal slices are interpolated to a uniform grid of size $N_x \times N_y = 512 \times 512$ for further use in the following sections. Fig. 5.13 illustrates the temporal growth of the con-

L_0	N_0	B_0	t_0	ν	Pr	N_{FD}	$N_x \times N_y$	Re_0
0.025	11.69	1.0	-0.5078	$1.49 \cdot 10^{-3}$	1	$2400 \times 2400 \times 210$	512×512	4.894

Table 5.6: **Simulation parameters** of the direct numerical simulation of the three-dimensional dry convective boundary layer. Presented are the Ozmidov length L_0 (5.20), buoyancy frequency N_0, surface flux B_0, reference time t_0, see (5.21), kinematic viscosity ν, Prandtl number Pr, finite-difference grid size N_{FD}. horizontal slices of the DNS were interpolated to a uniform grid of size $N_x \times N_y$. Further, the buoyancy Reynolds number Re_0 (5.22) is listed as well. The final CBL height at the end of the DNS was $48.41 L_0$. The DNS aims to replicate the two-dimensional system of 5.1. As parameters like Γ and Ra are now time-dependent, the DNS was instructed to assume values of $\mathrm{Ra}_c = 3 \cdot 10^8$ and $\Gamma = 25$ during its runtime.

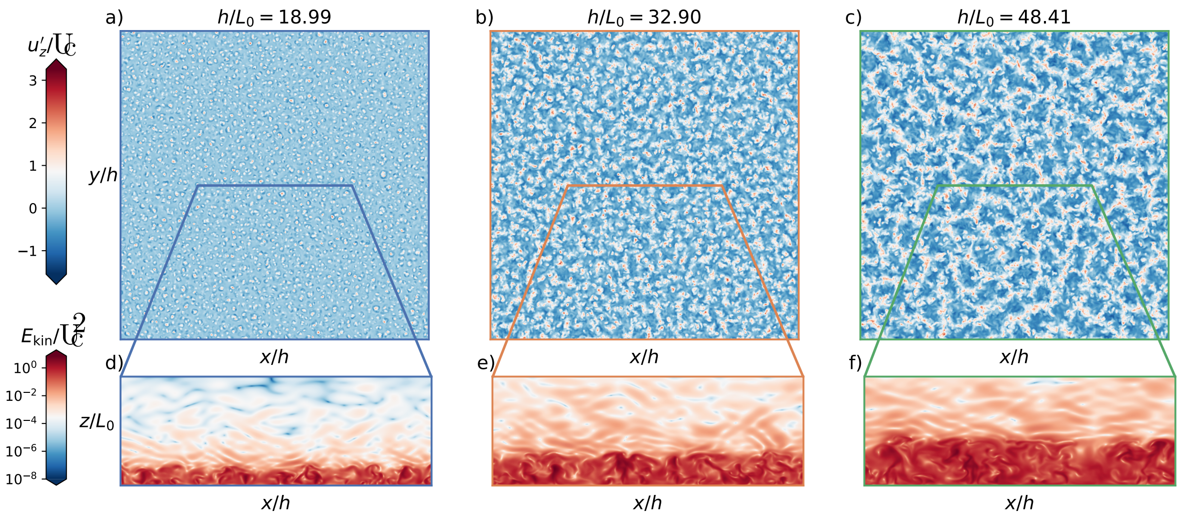

Figure 5.13: **Transient temporal growth** of the atmospheric boundary layer. (a-c) Contours of the vertical velocity fluctuations $u_z\prime$ viewed from the top onto the whole horizontal domain three different times. (d-f) Corresponding turbulent kinetic energy, $E_{\mathrm{kin}} = 0.5({u_x'}^2 + {u_y'}^2 + {u_z'}^2)$, in a vertical contour plot. The horizontal extension of the cut is indicated in (a-c). (g) panels (a-c). See also Fig. 5.14 for the corresponding vertical profiles of the buoyancy.

vective boundary layer during the DNS. While panels (a-c) provide a view from the top for the vertical velocity fluctuations with an aggregation of the convection cells, panels (d-f) dis-

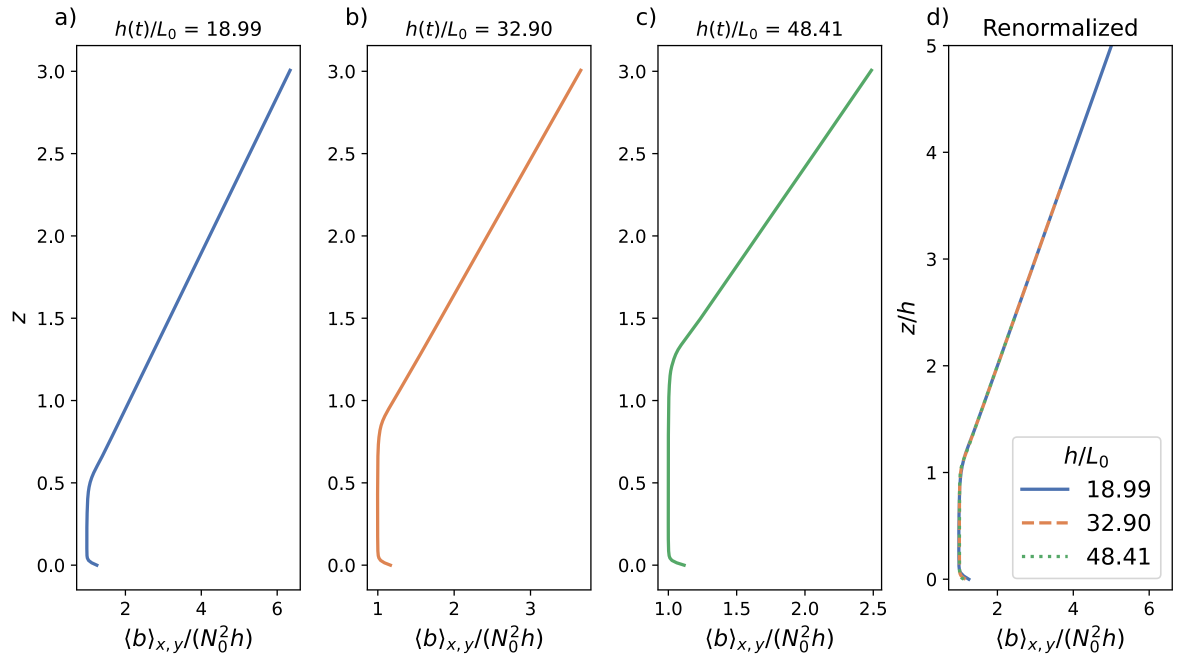

Figure 5.14: (a-c) Plane averaged **vertical profiles** of the buoyancy versus height at the three time instants from Fig. 5.13. (d) Rescaled vertical profiles of the three time instants over the rescaled height z/h. By doing so, all the profiles merge into a single unified profile. It is such a renormalization procedure that will be exploited for the training database of the generative neural network.

play the turbulent kinetic energy (TKE) in a vertical cut, illustrating the transient growth process. Moreover, Fig. 5.14 demonstrates that this transient growth follows a self-similar process and a rescaling of lengths with $h(t)$ and buoyancy with $N_0^2 h(t)$ collapses the vertical mean buoyancy profiles $\langle b(z,t)\rangle_{x,y}$ at different times to a unified profile. It is precisely this self-similar behavior given by (5.21) for the growth of the mesoscale patterns in vertical and horizontal directions, which will be used to augment the training database for the generative network.

The prefactor M_u in the second term of (5.27) incorporates the specifics of the upward transport. As detailed in Appendix B.2.1.2, the unknown mass flux M_u is estimated as a product of the updraft area fraction, a_u, and the excess of the mean vertical velocity in the updraft regions, $\langle u_z\rangle_u$, over the mean upward motion, i.e.,

$$M_u \approx \rho_0 a_u (\langle u_z\rangle_u - \langle u_z\rangle_{x,y})\,. \tag{5.28}$$

The horizontal cross-section of the layer A_{total} is decomposed into disjoint strongest updraft (u) and downdraft (d) regions, as well as the remaining intermediate motions region (i), i.e., $A_{\text{total}} = A_u \cup A_d \cup A_i$. Moreover, the environment region is further refined, typically taken as $A_e = A_i \cup A_d$, e.g., when the mass-flux models are discussed. This step is taken to analyze the asymmetry between the up- and down-welling motion of the same strength. Note that while $\langle\cdot\rangle_{x,y}$ are horizontal averages with respect to the whole cross-section A_{total}, $\langle\cdot\rangle_u$ denotes an average over the updraft regions A_u, see also eq. (5.27). Figure 5.15 underlines the observation from Fig. 5.13 more quantitatively. The buoyancy flux is arranged in narrow updraft and broad downdraft regions in the convective boundary layer. Here, the probability density functions (PDFs) for both cases at different times of the transient growth at half-height are presented in panels (a,b). A direct comparison of both panels shows that the downdraft distributions have broader tails. Consequently, downdrafts occupy slightly broader areas, which underline the top-down asymmetry of the convective motion, see also panel (c). The asymmetry becomes stronger the closer one moves to the surface (not shown). Moreover, it can be seen that the PDFs collapse increasingly better on each other in the core and parts of the tails as h/L_0 grows. The insets in Fig. 5.15(a) point out that the cell width of the updraft patterns grows with time.

5.2.3 Generative machine learning model of the CBL

5.2.3.1 Generative adversarial networks

In this section, the generative machine learning model and its architecture are introduced. Specifically, this part of the thesis will make use of a Generative Adversarial Network (GAN) [225]. DNS data will be denoted in short by $\mathbf{x}$, following the general machine learning nomenclature. Here the input will be $\mathbf{x} = (b', u_z')$. Figure 5.16 illustrates the concept of the two-component neural network. A GAN is a neural network comprising two interconnected and adversarial components: a generator G and a discriminator (or critic) D. The generator is seeded by random latent variables $\xi \sim \mathcal{N}(0,1)$[5] and generates synthetic data $\mathbf{x}_G = ({u_z'}^{\text{GAN}}(x,y),$ ${b'}^{\text{GAN}}(x,y))$, here, horizontal slices of fluctuations of vertical velocity and buoyancy. The discriminator's role is to distinguish between the synthetic data produced by the generator and a true DNS snapshot ${u_z}'(x,y)$, $b'(x,y)$. It then emits a score indicating whether the current input snapshot stems from the DNS, labeled as real, or the generator, labeled as fake. The two components are trained simultaneously in a competitive manner, with the generator striving to generate data that can fool the critic and the critic continuously improving its

[5]Note that ξ must have the same dimensions as the output of the generator.

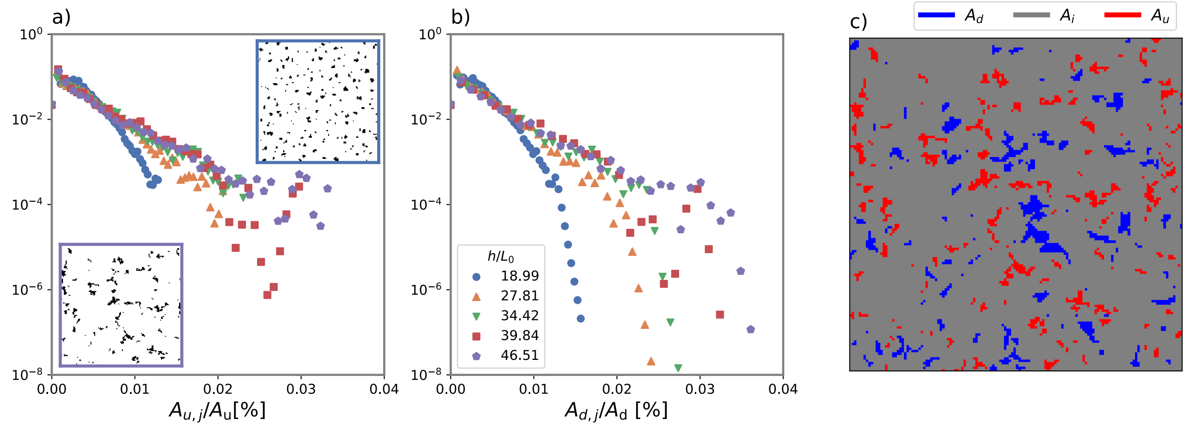

Figure 5.15: Statistics of the **up- and down-welling fluid motion** in the convective boundary layer (top 95% of u_z-values). (a) Probability density functions (PDFs) of the area elements $A_{u,j}$ of the up-welling fluid motion normalized by the horizontal updraft area $A_u = \sum_j A_{u,j}$ as a function of h/L_0 (see color legend in (b)). Note that the area fraction in the mass-flux scheme is $a_u = A_u/A_{\text{total}}$. The two insets display a cutout of the updrafts in a fixed horizontal plane $z = 0.5$ at different times. (b) PDF of the area elements $A_{d,j}$ of the down-welling fluid motion normalized by the horizontal downdraft area $A_d = \sum_j A_{d,j}$ as a function of time. (c) Horizontal cutout, which illustrates the decomposition of A_{total} into regions of strongest updrafts A_u, strongest downdrafts A_d, and the intermediate motions A_i. Note that the union of the regions of the strongest downdrafts and intermediate motions yields the environment A_e for the mass-flux formulation.

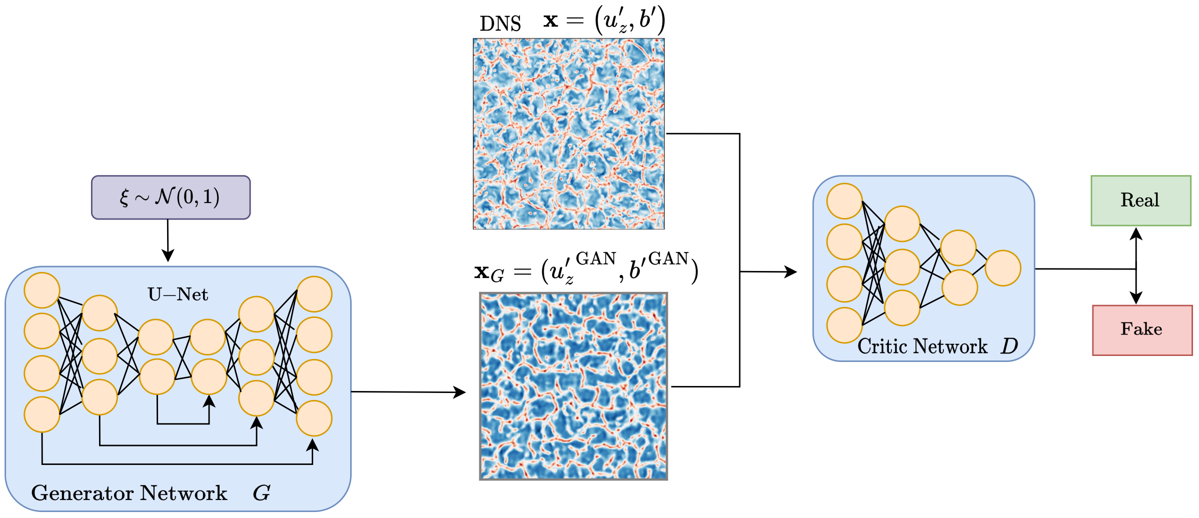
DNS $\mathbf{x} = (u_z', b')$
$\xi \sim \mathcal{N}(0,1)$
U−Net
Generator Network G
$\mathbf{x}_G = (u_z'^{\,\text{GAN}}, b'^{\,\text{GAN}})$
Critic Network D
Real
Fake

Figure 5.16: **Generative Adverserial Network** (GAN) scheme. The GAN consists of two networks: the generator G, which is trained to mimic DNS data together with its statistical properties. Secondly, the discriminator, or critic network, D is trained to classify an input image as stemming from the DNS or the generator. Both networks are trained against each other, s.t., similar to a two-player game, an equilibrium between both players is reached.

ability to classify the data correctly. This adversarial relationship drives the GAN to produce increasingly realistic, high-quality synthetic data. The training of the network is completed once the *Nash equilibrium* is reached [226] and the discriminator is unable to distinguish synthetically generated and ground truth data. Then, the generator can be decoupled from the discriminator and operate autonomously. However, GANs are notoriously hard to train [227]. Common issues include a vanishing generator gradient and a non-converging adversarial training process. To circumvent these problems, the Wasserstein–GAN (WGAN) formulation from Arjovsky *et al.* [228], which solves the vanishing gradient problem of the generator of the original GAN model, will be employed. This is done using the Wasserstein distance (or Earth-movers distance) between the synthetic and real data in the WGAN loss function. In this way, the generator network will overcome the initial phase, where the discriminator will quickly identify the generated data samples, and the GAN training ends in favor of the critic network. Furthermore, the WGAN shows greater stability and convergence during the training. Details on the specific neural network architecture that was used here can be found in Appendix B.2.2.1

In the WGAN formulation, the output of the critic corresponds to a score that grades the origin of the input. Therefore, the critic is trained to output a higher score for the DNS input $\mathbf{x}$ than for its synthetic counterpart $\mathbf{x}_G$. The generator on the other hand, is trained to maximize the critic's output for $\mathbf{x}_G$. The Wasserstein distance is incorporated into the discriminator loss function using the Kantorovich-Rubinstein duality [229]. However, this restricts the critic $D(\cdot)$ to be a 1-Lipschitz function (i.e., the Lipschitz constant is one), which can be enforced by an additional gradient penalty term [230]. The WGAN discriminator loss function then reads

$$\mathcal{L}_\mathrm{D} = \mathop{\mathbb{E}}_{\mathbf{x}_G \sim P_\mathrm{G}} [D(\mathbf{x}_G)] - \mathop{\mathbb{E}}_{\mathbf{x} \sim P_\mathrm{r}} [D(\mathbf{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}} \sim P_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2], \tag{5.29}$$

where $\mathbf{x}$ is sampled from the real, i.e., DNS data distribution P_r and $\mathbf{x}_G$ from the generator model distribution P_G. The latter is given by $G(\xi)$ with the random latent variable $\xi \sim \mathcal{N}(0,1)$ sampled from a standard normal distribution. The last term in (5.29) is the gradient penalty term, which enforces the 1-Lipschitz condition on the discriminator. The random samples $\hat{\mathbf{x}}$ are sampled from straight lines $\chi\mathbf{x} + (1-\chi)\mathbf{x}_G$ between pairs of $\mathbf{x}$ and $\mathbf{x}_G$, as proposed in by Gulrajani *et al.* [230]. Here, $\chi \sim U[0,1]$, i.e., uniformly distributed between 0 and 1. The gradient penalty weight λ is a hyperparameter. Moreover, the generator loss is given by

$$\mathcal{L}_\mathrm{G} = -\mathop{\mathbb{E}}_{\mathbf{x}_G \sim P_\mathrm{G}} [D(\mathbf{x}_G)]. \tag{5.30}$$

Finally, the Wasserstein loss function, which is given by

$$\mathcal{L}_\mathrm{WGAN} = \mathop{\mathbb{E}}_{\mathbf{x} \sim P_\mathrm{r}} [D(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}_G \sim P_\mathrm{G}} [D(\mathbf{x}_G)], \tag{5.31}$$

reflects the WGAN performance. A detailed WGAN training procedure is summarized in Algorithm 1 in Appendix B.2.2.2.

5.2.3.2 Physics-informed training data augmentation

The interpolated horizontal slices of DNS data are used as a training database for the GAN. For this, the data is augmented in a re-normalization procedure that removes the major first-order effects of the CBL growth. To this end, the DNS snapshots in the statistical quasi-steady phase of the CBL are used. Remember that, in this phase, spatial patterns of the up-

and downdrafts are not affected by the very initial fast CBL growth. See Fig. 5.15. To this end, the cutoff value of h/L_0, after which all re-normalized snapshots can be considered for the training routine, is determined by the following procedure. First, the individual up- and downdraft areas A_u and A_d, given by values $u_z > 0.95\max(u_z)$ and $u_z < 0.95\min(u_z)$ respectively, are retrieved using the algorithm introduced in ref. [231] via the *OpenCV* [232] library implementation. Finally, the *Kullback-Leibler divergence* (KLD) between the PDF p of up- or downdrafts at h/L_0 with the final PDF p_{final} at $h/L_0 = 48.41$ (i.e., the end of the recorded snapshots) is computed. For updrafts, it is then given by

$$D_{\text{KL}}(p\|p_{\text{final}}) = \sum_{A_u} p(A_{u,j}) \log\left(\frac{p(A_{u,j})}{p_{\text{final}}(A_{u,j})}\right) . \tag{5.32}$$

Figure 5.17 shows the KLD for the three planes over h/L_0. The slope of the KLD becomes less steep near $h/L_0 = 35.87$ (start of green shaded area). After this value, all snapshots are considered for the GAN training. In this statistical quasi-steady regime, a new set of

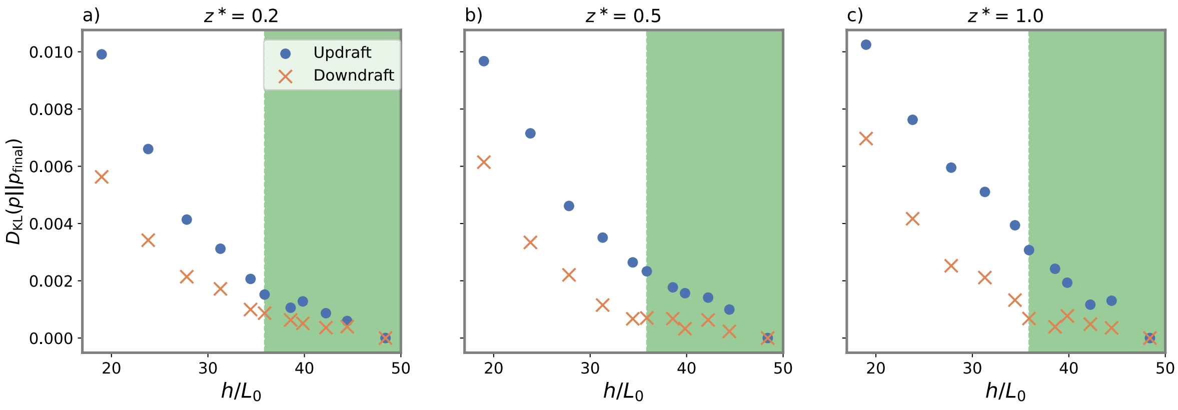

Figure 5.17: **Kullback-Leibler divergence** between the probability density functions of normalized updraft (downdraft) area elements $A_{u,j}$ ($A_{d,j}$) at h/L_0 and the corresponding final probability density function. The green shaded area marks the snapshots that were considered for the training of the GAN. It comprises the period $h/L_0 \in [35.87, 48.41]$.

variables can be introduced that embed, to leading order, the transient temporal growth of the boundary layer height. To this end, the turbulence fields are rescaled with the similarity variables, which correspond to the mixing layer theory by Deardorff [233]. These variables are given by the convective scales (5.2 - 5.4), where the constant RBC height H is substituted by the increasing CBL height $h(t)$. This gives the following transformation rules,

$$u_z{}^*(x^*, y^*, z^*, h/L_0) = \frac{u_z(x,y,z,t)}{U_c(t)} , \quad b^*(x^*, y^*, z^*, h/L_0 = \frac{b(x,y,z,t)}{b_c(t)} , \tag{5.33}$$

with rescaled time $h(t)/L_0$ and the normalized coordinates x^*, y^*, $z^* = (x/h(t), y/h(t), z/h(t))$. By doing so, the CBL height is fixed. This has implications for the horizontal extent of the normalized snapshots as the aspect ratio of the domain decreases over time. Hence, to compare two snapshots at two different h/L_0 with the same horizontal extent, one has to crop the snapshot with the smaller value of $h(t)/L_0$. This in turn results in similar-sized spatial patterns of $u_z{}^*$ and b^* and thus of $u_z{}^{*\prime} b^{*\prime}$ at different times h^*. In this way, the first-order effects of the transient CBL growth in a fixed plane $z^* = $ const are removed. This

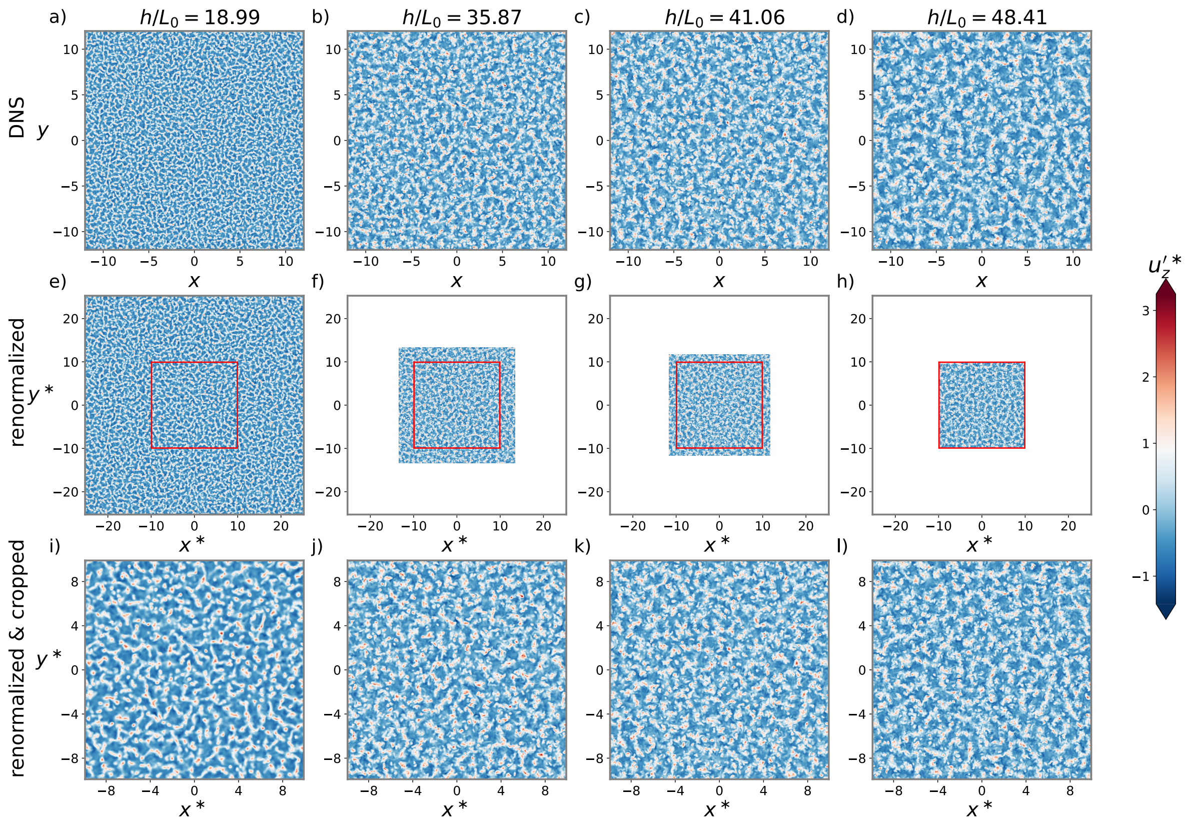

Figure 5.18: **Renormalization and cropping procedure** illustrated for the vertical velocity fluctuations ${u'_z}^*$ in plane $z^* = 0.5$ for four time instances h/L_0. (a-d) Original simulation snapshots of $u_z'(x, y)$. (e-h) Rescaled fluctuations ${u'_z}^*(x^*, y^*)$ in the new coordinate system (x^*, y^*). (i-l) ${u'_z}^*(x^*, y^*)$ cropped to the domain $x^*, y^* \in [-\tilde{L}, \tilde{L}]$ with $\tilde{L} \approx 8.5$ at time $h/L_0 = 48.41$ of panel (l). See also the red box in the middle row. The data preparation for the buoyancy fluctuations proceeds along the same lines. While the snapshots in the initial row display significant dissimilarities, the augmented snapshots in the final row demonstrate more consistent spatial characteristics.

renormalization process is illustrated in Fig. 5.18. In panels (i-l), it can be seen that for sufficiently large values of h/L_0, this procedure produces horizontal slices with comparable flow patterns. The same does hold for the buoyancy fluctuations b' (not shown). It is these flow patterns that the GAN will be trained with to generate synthetic data. The training data of the GAN is then chosen to be a horizontal slice at a plane z^* with constant horizontal extent in normalized coordinates for all snapshots. In summary, this results in an augmented training database that encodes the effects of a growing boundary layer to the first order. The similar spatial features are then easily picked up by the generator network. Finally, in order to obtain synthetic snapshots at any given rescaled time h/L_0, the re-normalization procedure is simply reversed.

5.2.3.3 GAN training

Three GANs are trained for the planes $z^* = 0.2, 0.5$ and 1.0, respectively. In this way, the ML approach enables a fast generation of spatial and statistical information at three important and representative heights: the surface and mixed layer and the entrainment zone. All GANs are trained for 2000 epochs using the Adaptive moments estimation (Adam) optimizer [61]. An early stopping routine ceased training after 300 epochs of no improvement of the Wasserstein metric $\mathcal{L}_{\mathrm{WGAN}}$. The choice of GAN hyperparameters for the three planes is listed in Tab. 5.7. Figure 5.19 shows the losses for both the generator and the discriminator, together with the Wasserstein loss for all three GANs. It can be seen that all losses converge nicely to an equilibrium state of both the generator and critic. In this converged regime, the generator can then be used for the task of subgrid-scale feature generation.

plane z^*	η	batch size	$N_{\mathrm{D}}/N_{\mathrm{G}}$	λ	N_{train}
0.2	$2 \cdot 10^{-5}$	64	12	11	550
0.5	$2 \cdot 10^{-5}$	64	11	10	550
1.0	$5 \cdot 10^{-5}$	64	90	10	550

Table 5.7: **GAN hyperparameters**. As described in Section 1.2, the learning rate η regulates the speed of the gradient descent update step. Again, the batch size refers to the number of training examples processed together in a single forward and backward pass. The parameter $N_{\mathrm{D}}/N_{\mathrm{G}}$ signifies the degree to which the critic network undergoes additional training compared to the generator network. The gradient penalty parameter λ controls the strength of the 1-Lipschitz condition in (5.29). Each GAN was trained on N_{train} DNS training samples, which covered the period $h/L_0 \in [35.87, 48.41]$. Each training run was performed on two *NVIDIA A100 Tensor-Core GPUs* and took up to 24h to finish. See also Algorithm 1 in Appendix B.2.2.2 for the role of each hyperparameter.

5.2.4 Results of the generative machine learning method

5.2.4.1 Structure and statistics of the generated sub-grid scale fields

Figure 5.20a-f) demonstrates that the GAN can produce strikingly realistic horizontal slices of ${u_z}'(x,y)$, $b'(x,y)$, and ${u_z}'b'(x,y)$ that reproduce all spatial features of all (ground truth) fields, here for half-height $z^* = 0.5$ at $h/L_0 = 48.41$. Moreover, the generated data reproduces the highly non-Gaussian, skewed PDFs of all three fields almost perfectly, see panels (g-i). Even far-tail events are captured fairly well in all three cases. This has direct implications for a machine learning-based convective parameterization, as information on the spatial

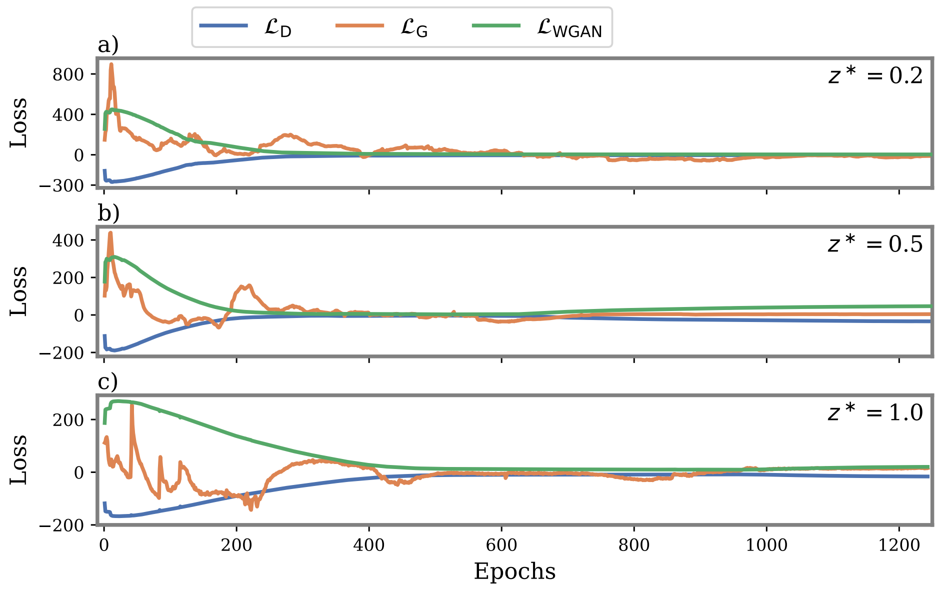

Figure 5.19: **GAN Training loss** of the three neural networks at height $z^* = 0.2$ (a), 0.5 (b) and 1.0 (c). The discriminator (blue) and generator (orange) loss $\mathcal{L}_\mathrm{D}$ and $\mathcal{L}_\mathrm{G}$ converge towards individual values. The Wasserstein loss $\mathcal{L}_\mathrm{WGAN}$ (green) measures the quality of the GAN results. Hence the parameters of the GAN are saved at the minimum Wasserstein loss value.

organization of individual plumes becomes available in contrast to the physical parameterization models based on means and Gaussian statistics. Additionally, the derived PDFs allow access to the high-order statistics of the fields, such as the strongest updrafts that shoot into the stably stratified layer above the CBL. Similar results can be observed for the distinct patterns and PDFs at $z^* = 0.2$ in Fig. 5.21, slightly above the surface layer, and for $z^* = 1.0$ in Fig. 5.22, in the entrainment zone.

5.2.4.2 Transient time evolution

Motivated by these results, the GAN's performance in reconstructing the transient statistics is evaluated. Figure 5.23 reports the turbulence statistics with respect to time. Panels (a-c) of the figure presents the PDFs of ${u_z}'/\sigma_{u_z'}$, $b'/\sigma_{b'}$, and ${u_z}'b'/\sigma_{u_z'b'}$ where the corresponding $\sigma_p^2(t) = \langle p^2 \rangle_{x,y}$ for $p = \{{u_z}', b', w'b'\}$. It is seen that the GAN generates the correct time evolution of the statistics and the corresponding moments. Panels (d-f) show the time evolution of the fluctuations $\sigma^2_{u_z'b'} = \langle ({u_z}'b')^2 \rangle_{x,y}$ and the normalized third- and fourth-order statistical moments of the buoyancy flux versus time. The latter two are the skewness $S_{u_z'b'}$ and the flatness $F_{u_z'b'}$ which are given by

$$S_{u_z'b'}(t) = \frac{\langle ({u_z}'b')^3 \rangle_{x,y}}{\sigma^3_{u_z'b'}} \quad \text{and} \quad F_{u_z'b'}(t) = \frac{\langle ({u_z}'b')^4 \rangle_{x,y}}{\sigma^4_{u_z'b'}} \,. \tag{5.34}$$

The figure shows that both normalized moments deviate strongly from the Gaussian values, which would follow to $S_{u_z'b'} = 0$ and $F_{u_z'b'} = 3$. Again, the GAN reproduces the slow growth of the moments fairly well. Note also that the fluctuations of the time series grow with time since the sampling area for taking statistics shrinks with respect to time due to the renormalization procedure discussed above. Flatness values of the order of 50 indicate a highly intermittent transport across the growing CBL.

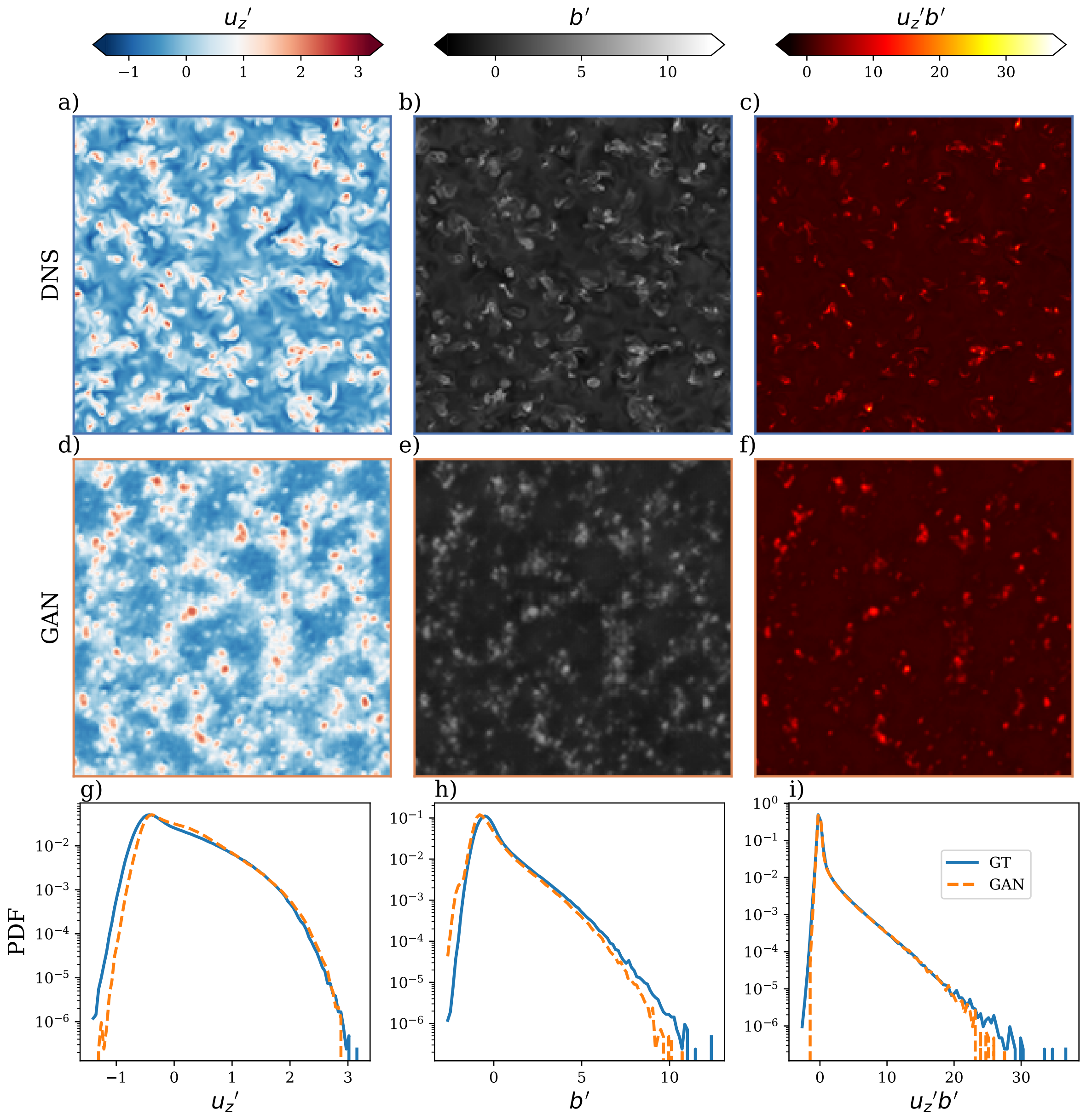

Figure 5.20: Comparison of the **generated turbulent fields** with the ground truth. (a-c) Horizontal cross sections at $z = 0.5h = 0.584$ at $h/L_0 = 48.41$ of the vertical velocity fluctuations $u_z{}'$ (a), the buoyancy fluctuations b' (b), and the local convective heat flux $u_z{}'b'$ (c) from DNS which is the ground truth (GT). (d-f) Corresponding cross-sections which have been generated by the generative adversarial network (GAN). Note that the normalization of all fields was reversed to end up with dimensional physical fields. (g-i) Comparison of the probability density functions of the fields.

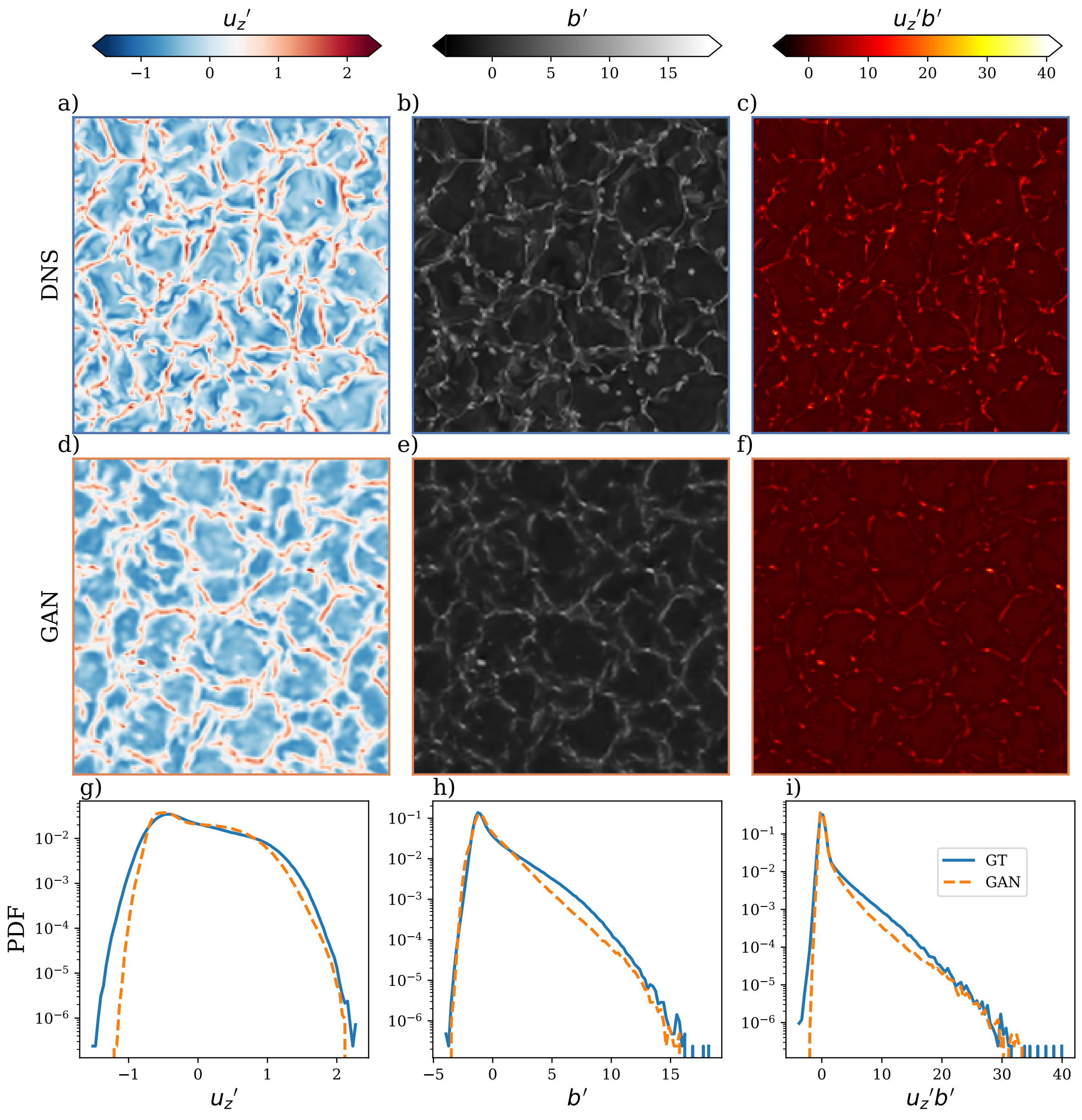

Figure 5.21: Comparison of the **generated turbulent fields** with the ground truth. Horizontal slices at $h/L_0 = 48.41$ in plane $z = 0.2h = 0.233$ of the vertical velocity fluctuations u_z' (a), the buoyancy fluctuations b' (b), and the local convective heat flux $u_z'b'$ (c) from DNS which is the ground truth (GT). (d-f) Corresponding cross-sections which have been generated by the generative adversarial network (GAN). Note that the normalization of all fields was reversed to end up with dimensional physical fields. (g-i) Comparison of the probability density functions of the fields. Note the different spatial characteristics as compared to the $z/h = 0.5$ case in Fig. 5.20

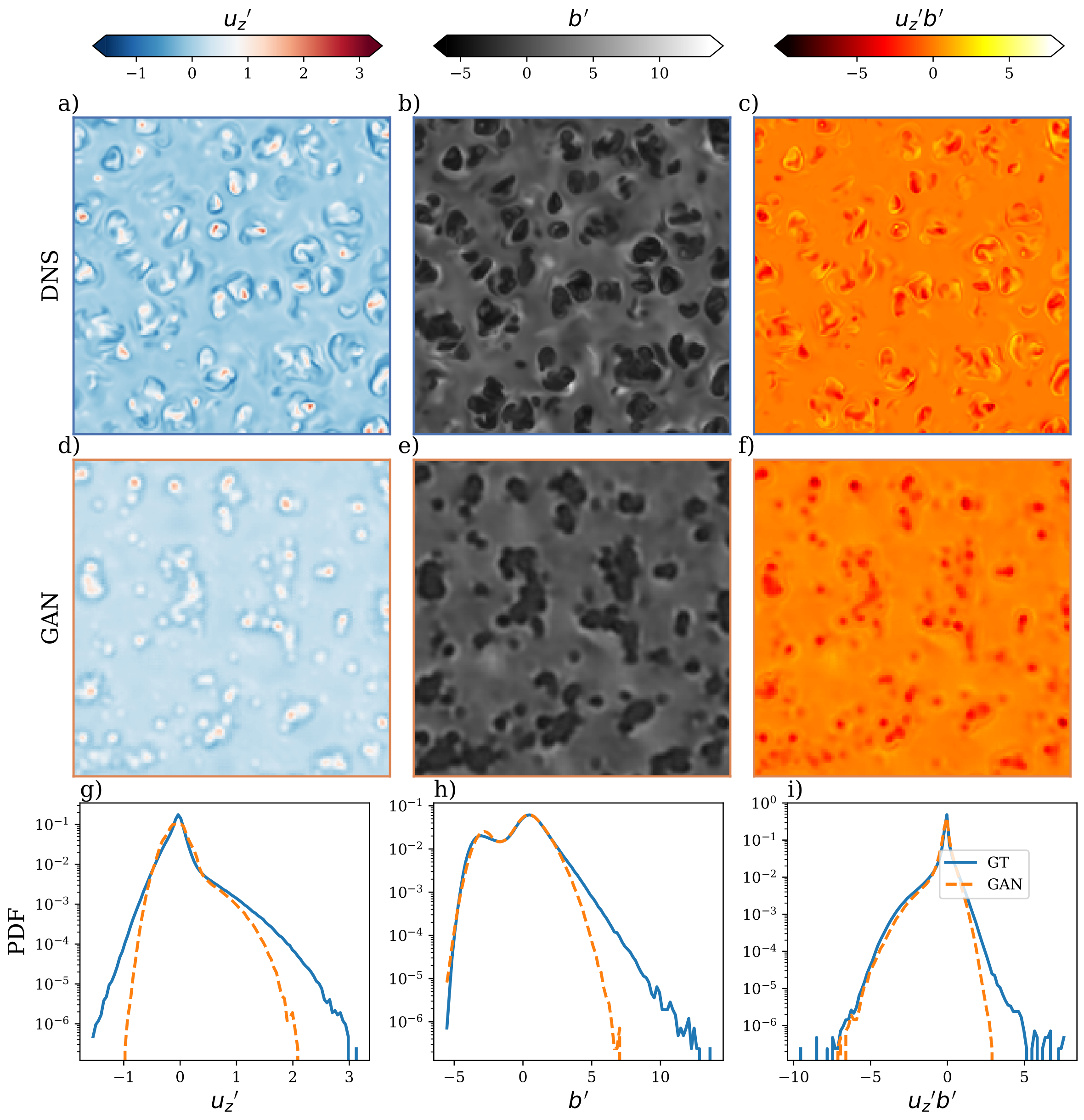

Figure 5.22: Comparison of the **generated turbulent fields** with the ground truth. Horizontal slices at $h/L_0 = 48.41$ in plane $z = 1.0h = 1.167$of the vertical velocity fluctuations u_z' (a), the buoyancy fluctuations b' (b), and the local convective heat flux $u_z'b'$ (c) from DNS which is the ground truth (GT). (d-f) Corresponding cross-sections which have been generated by the generative adversarial network (GAN). Note that the normalization of all fields was reversed to end up with dimensional physical fields. (g-i) Comparison of the probability density functions of the fields. Note the different spatial characteristics compared to the $z/h = 0.2, 0.5$ cases in Figs. 5.20, 5.21.

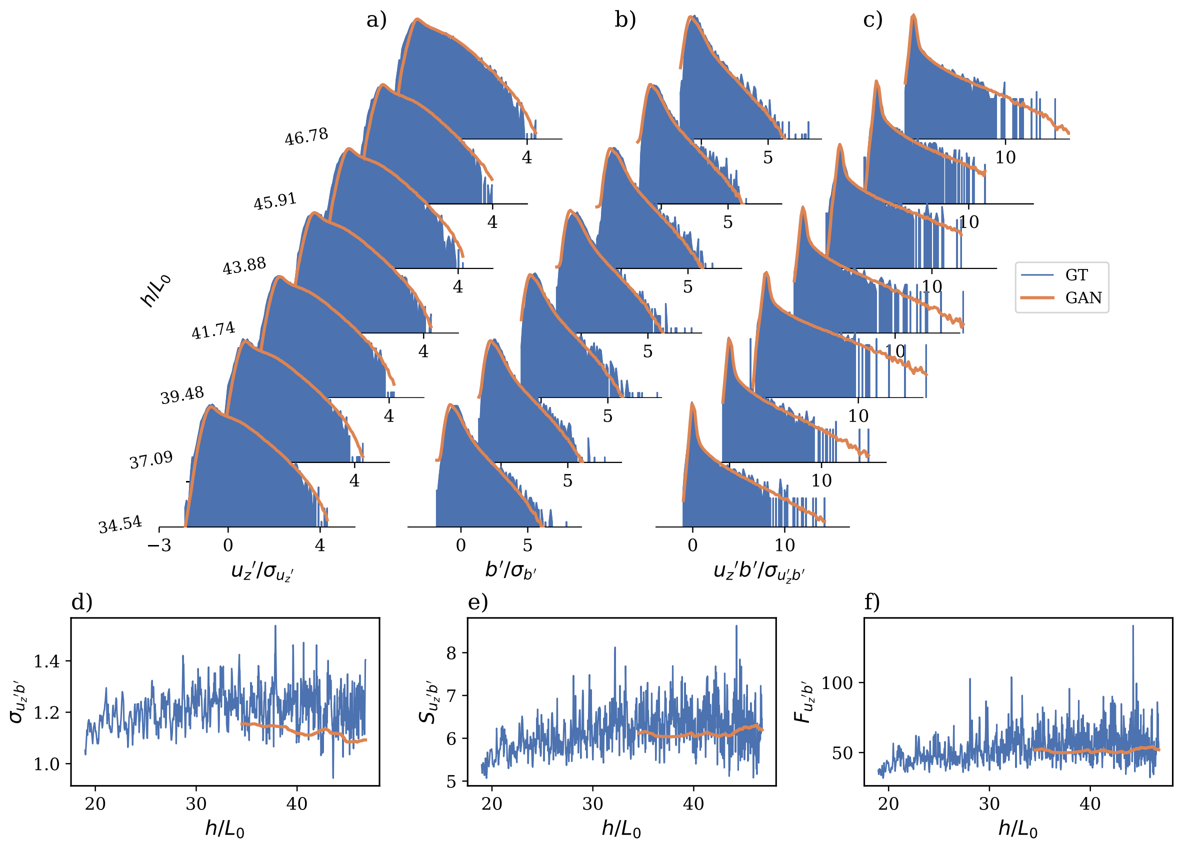

Figure 5.23: **Prediction** of the time evolution of the **turbulence statistics** by the Generative Adversarial Network (GAN). The probability density functions of the vertical velocity fluctuations (a), the buoyancy fluctuations (b), and the buoyancy flux (c) are shown for different times h/L_0. All quantities are normalized by their corresponding root-mean-square values. The ground truth (GT) is given in blue, and the GAN results in red. The bottom row panels display the time evolution of the moments of the buoyancy flux. These are the fluctuations (d), the skewness (e), and the flatness (f). The blue line is for the GT, and the orange line is for the GAN. Data are for $z^* = 0.5$.

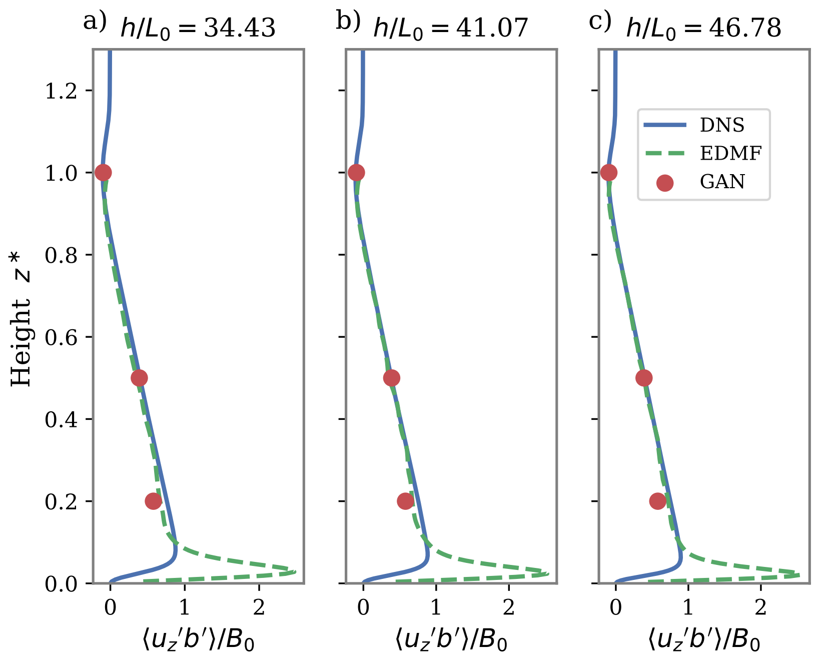

Figure 5.24: **Vertical profile of the buoyancy flux** $\langle u_z\prime b\prime \rangle / B_0$ at three values of h/L_0. The eddy diffusivity mass-flux scheme uses a mass flux that is proportional to the updraft velocity $M_u \sim \langle u_z \rangle_u$, which is obtained from a plume model (green curve), see also Appendix B.2.1. Note the discrepancy of the EDMF in the surface layer. This is due to the eddy-diffusivity component, which overestimates the maximum near $z^* = 0.1$. The generative network results at three heights $z^* = 0.2$, 0.5, and 1.0 are marked as red symbols.

5.2.4.3 Comparison with an eddy diffusivity mass-flux parameterization

Finally, the GAN networks enable computation of the mean turbulent buoyancy flux $\langle u_z' b' \rangle_{x,y}$ at reference heights inside the CBL. Their performance is now compared with the ground truth, as well as with the results from a frequently used closure, see (5.27). As mentioned before, the latter is termed eddy-diffusivity mass-flux and contains the mass-flux parametrization as a second term next to the Boussinesq term, which always connects subgrid-scale flux (or stress) with a mean gradient via an eddy (or turbulent) viscosity.
To this end, an EDMF scheme is applied, which follows from a steady plume model [33]. As detailed in Appendix B.2.1, the model comprises expression (5.27) and the following two equations

$$\frac{\partial \langle b \rangle_u}{\partial z} = \epsilon(\langle b \rangle_{x,y} - \langle b \rangle_u)\,, \tag{5.35}$$

$$\frac{1}{2}(1-2\mu)\frac{\partial \langle u_z \rangle_u^2}{\partial z} + C_1 \epsilon \langle u_z \rangle_u^2 = \langle b \rangle_u - \langle b \rangle_{x,y}\,. \tag{5.36}$$

Note that these equations still contain the adjustable model parameters μ, C_1, and ε. Typically $\mu \approx 0.15$ and $C_1 \approx 0.5$ are chosen [33]. The yet unknown entrainment rate ε is also height-dependent and modeled by the following expression with adjusted prefactor [33]

$$\varepsilon(z) \simeq 0.4\left(\frac{1}{z} + \frac{1}{h-z}\right)\,. \tag{5.37}$$

Equations (5.35)–(5.37) for the variables $\langle b(z)\rangle_u$ and $\langle u_z(z)\rangle_u$ together with eq. (5.27) are then solved numerically. The EDMF closure takes finally $M_u \simeq a_u \langle u_z\rangle_u$ [234] where $a_u = 0.05$[6]. See again Appendix B.2.1.3 where the model formula for the turbulent diffusivity κ_t is provided as well. The comparison of all models is shown in Fig. 5.24 for three different times h/L_0. It can be seen that the three flux values, predicted by the GAN, agree excellently with the DNS profiles. These results compare also well to the profiles of the EDMF parametrization. Note the discrepancy of the EDMF in the surface layer. This is due to the eddy-diffusivity component, which overestimates the maximum near $z^* = 0.1$.

5.3 Conclusions from this chapter

This chapter was concerned with the application of two major machine learning frameworks for the feature generation of turbulent convection flows. The first explored the generalization property of a combined Autoencoder-Reservoir Computing model applied to a more complex convection flow than standard RBC. To this end, buoyancy fluxes at the vertical boundaries, which can be understood as entrainment from the top and surface heating from the bottom in an atmospheric convective boundary layer, are applied to a two-dimensional convection domain. Thus the model resembles properties 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. It is thus an ideal testing bed for dynamic parametrizations of the buoyancy flux and its low-order moments by machine learning algorithms. To this end, the combined AE-RC model was trained on DNS convection data for the case where the top flux equals 0.3 of the surface flux. The performance of the machine learning algorithm to unseen data with different physical parameters was evaluated. For this purpose, three target cases where the top flux was 0.1, 0.2, or 0.4 of the bottom flux were selected. It can be concluded that for the presented 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 combination of Autoencoder and 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. However, deficiencies can be observed in their ability to reconstruct the temporal features of the unseen flow.

In the second part of this chapter, a data-driven parametrization on sub-kilometer scales for a dry convective boundary layer, the lowest part of the atmosphere, was presented. Simulations of a three-dimensional CBL growing into a stably stratified, non-turbulent atmosphere were conducted. The growing boundary layer exhibits transient dynamics as well as evolving spatial features that are not expected to be picked up by the previously employed ROM-RC approaches. Hence, the parameterization was based on a different approach using a generative adversarial network with Wasserstein loss metrics. Moreover, the network was trained on DNS data that was re-normalized using scales derived from similarity theory. The augmented DNS data resulted in an enhanced training database for the generative network. In turn, the GAN provides the horizontal spatial organization and the statistics of the vertical velocity component and the buoyancy, and thus the one of the local buoyancy flux at different heights inside the CBL. Moreover, the machine learning algorithm is also able to predict the gradual temporal change of the statistical distributions of the buoyancy flux in the course of the diurnal cycle, which is connected with a growth of the boundary layer height

[6]This corresponds to the definition of a strong updraft as $u_z > 0.95 \max(u_z)$.

$h(t)$ and increasing entrainment into the stable atmosphere on top of the CBL. The highly non-Gaussian and intermittent statistics of this process are successfully reproduced by the generative convective parametrization.

Chapter 6

Conclusions and perspectives

This thesis was concerned with the modeling of turbulent convection flows by two machine learning-based frameworks, namely a Reservoir Computing model and a Generative Adversarial Network. The purpose of the presented work was to assess their performance in regard to reproducing the spatial organization, structural features and low-order statistics of dry and moist convection flows without solving the underlying non-linear equations of motion. Taking into account the low computational expenses of a trained ML model, the motivation of this work was contextualized within the sub-grid scale parameterization of large-scale general circulation models. With this goal in mind, the two machine learning frameworks were employed to acquire spatial and temporal patterns, along with low-order statistics, from direct numerical simulation data related to turbulent convection flows.

The first and primary approach of this thesis employed a combination of a Reduced Order Model, such as a POD or an Autoencoder network, and a Reservoir Computing model, namely an Echo State Network. While the former was designed to condense the numerous degrees of freedom in a thermal convection flow into a lower-dimensional latent space, the latter served as a dynamic core trained on the dynamics of the reduced latent space to perform temporal prediction without knowing the Boussinesq equations of turbulent convection. In a subsequent autoregressive prediction phase, the model produced a time series that was subsequently decompressed to yield predictions of the turbulent convection flow. Chapter 4 showed how this framework can operate on two-dimensional dry and moist convection flows using the echo state approach together with a POD. It was observed that the autoregressive ML model generates synthetic snapshots that exhibit temporal and spatial characteristics of the corresponding convection flow. Furthermore, the model demonstrated the capability to replicate vertical profiles such as the buoyancy flux, a vital feature relevant to subgrid-scale transport within large-scale atmospheric models. In this way, **research question (i)**, as presented in Section 1.4, has been addressed.

Moreover, the ROM-RC framework was used to answer **research question (ii)**, which targeted the performance of the aforementioned model when applied to convection systems of varying complexity. To achieve this, in Section 3.2, an autoregressive Echo State Network (ESN) was trained using an eight-degree-of-freedom low-order model of thermal convection. Because the dynamical system's dimensionality was low, there was no need for a data reduction step. It was shown that even though the ESN is able to predict the Lorenz dynamics accurately for several Lyapunov times, the prediction error quickly accumulates. However, the generated data captures the typical behavior of the individual modes and even reproduces its chaotic attractor, indicating that the reservoir outputs can be seen as new realizations of the nonlinear system trajectory. This behavior was also observed in Section 4.1, where the ML model was applied to POD time coefficients of direct numerical simulations of two-dimensional Rayleigh-Bénard convection at a moderate aspect ratio and low Rayleigh number. The model not only generated matching temporal and spatial features but also robustly reproduced the low-order statistics of the convection flow. In Section 4.2, the complexity of the convection flow was increased further in two ways. First, the aspect ratio was increased by a factor of four, such that the necessary degree of data reduction is significantly higher. Secondly, the physical complexity of turbulent convection was enhanced by adding moisture and the associated effect of condensation and evaporation. After applying the POD to this new system, it was observed that the ESN is also able to match crucial moisture parameters like the fraction of clouds, cloud patterns, and liquid water flux. Hence, the present model is able to handle tasks of varying complexity. However, it should be noted that in all cases, a grid search procedure of the RC model's hyperparameters needs to be conducted in order to identify their optimal combination. This bottleneck can lead to long computation times depending on the dimensionality of the reservoir input.

In Section 5.1, the ROM-RC model was tested on its capability to generalize between convection systems with different system parameters. For this, a Rayleigh-Bénard-like model of a two-dimensional convective boundary layer was employed. Four cases differing in the ratio β of their buoyancy flux values at the vertical boundaries were considered. The systems' degrees of freedom were reduced using four convolutional Autoencoder networks that were trained on each of the convection cases. Moreover, a reservoir computer was trained on the reduced latent space data of the $\beta = 0.3$ case. It was shown that clearing the reservoir's memory of the known latent space dynamics and feeding it information of another, unseen β case can lead to a transition phase, after which the reservoir's output and the associated reconstructed flow reproduce properties of the unseen new regime. It was shown that spatial characteristics are reproduced naturally by this process. Low-order statistics like the buoyancy flux and fluctuations of velocity and buoyancy fields are also in good accordance. As a result, the third objective, **research question (iii)**, of this dissertation received a positive response.

The second machine learning model departed from the ROM-RC approach and instead ventured into a novel direction for sub-grid scale parameterization of a three-dimensional dry convective boundary layer. In Section 5.2, it was demonstrated that the transient growth of the convective boundary layer and its related pattern formation were attainable by training a Generative Adversarial Network using pre-processed simulation data of the CBL. Furthermore, the investigation revealed that despite the absence of memory in the GAN model, the dynamic pattern formation can still be captured by leveraging insights from similarity theory. This kind of physics-informed approach allowed for an enhancement of the limited amount of high-fidelity training database and, therefore, for a reproduction of the underlying transient process by the generative network. In this way, the model was able to generate snapshots of horizontal slices of buoyancy and vertical velocity fluctuations at three important CBL heights: the surface and mixed layer, as well as near the entrainment zone. Moreover, the synthetic turbulence data matched the strongly non-Gaussian statistics at the three heights to a very high degree. Further, the GAN was able to produce matching results to that of a conventional eddy-diffusivity mass-flux parameterization, however, with the advantage that the ML approach resolves individual up- and downdrafts. Finally, the presented data augmentation technique grounded in atmospheric theory reveals the potential for future machine learning applications to integrate fundamental physical principles. This answered the **final question (iv)** of this thesis.

Despite these four research questions, this thesis also touched on the subject of Quantum Reservoir Computing, where the high-dimensional reservoir was implemented in terms of a circuit of universal quantum gates. It was shown how a digital gate-based QRC model incorporates the classical RC properties of memory and input loading, by making use of the exponentially growing state space of the circuit's qubits. For this, two algorithms were proposed. Both models underwent assessment in a reconstruction assignment, wherein each model was granted access to only restricted pieces of information. Their objective was to reconstruct the complete set of information using this limited input. Notably, it was demonstrated that an actual quantum device, employing the first algorithm, can faithfully replicate the dynamics and attractor of the eight-order Lorenz model using solely the information from two of its modes. Furthermore, the QRC algorithm demonstrated significant success when executed on a real quantum NISQ device. While this algorithm still required a classical memory, a second algorithm was proposed, where the complete execution was enabled on a quantum computer. Both QRC methods exhibited encouraging *Qisiki* simulation outcomes when implemented in the ROM-RC framework for a fully-resolved simulation of Rayleigh-Bénard convection at a Rayleigh number of 10^5. It was found that three out of a total of 16 POD modes suffice to reconstruct the dynamics and low-order statistics of the chaotic flow. While these initial utilization of quantum machine learning for fluid dynamics issues remain confined in scope, they could potentially establish a pathway toward tackling more intricate challenges as we transition beyond the NISQ era.

The aforementioned results give rise to several research directions. While the ESN has shown good performance in reconstructed spatial and statistical features of the underlying RBC flow, it becomes harder for the network to capture the correct dynamics as the flow becomes more turbulent. See e.g. Section 4.2. By definition, neural networks are not designed to process data that live on a continuum of different lengths and times, a property inherent to turbulent flows. A potential approach toward achieving a scale-aware reservoir could involve manipulating the information flow within the reservoir by carefully designing the random input and reservoir weights. Recent studies have already started to pursue such directions by imposing a small-world topology onto the reservoir [235, 236]. Further, the reservoir was mostly employed in its closed-loop mode, where the error quickly accumulates, and the reservoir exhibits a generative behavior. The open-loop mode, on the other hand, showed excellent results, even for the novel quantum reservoir, indicating that the reservoir performs best when it is recalibrated with a reference signal. Hence, it is probably such tasks where the future of ESN applications lies. A potential use case could involve employing it as a turbulence model for subgrid-scale processes. In this scenario, the teacher signal would encompass data related to the larger scale, and the resulting output would pertain to the corresponding subgrid scale. A first test might be a shell model, as it was done in ref. [237] for an LSTM. Another application could be for super-resolution tasks or tasks of spatial prediction, as it was done in refs. [119] and [114].

The last chapter of this work revealed that a proper encoding of physical laws can enhance the performance of a neural network for feature generation. This pathway seems to be a favorable direction for future applied ML research, as it enforces the physical accuracy of the model's outputs. This could improve several of the methods employed in this work. First, as shown in Section 5.1, the convolutional Autoencoder introduced small artifacts into the lateral time average profiles of all physical fields. A way to circumvent this problem would be to introduce additional penalty terms into the Autoencoder loss function, s.t., the network preserves these features. In addition to such soft constraints, the physical information could be encoded directly into the architecture of the network. A similar path was taken, for example, in ref. [72], where an additional non-trainable layer enforced the incompressibility constraint. Another next step would be to make the ESN respect specific restrictions of the learned temporal data. In the case where the ESN learns POD time coefficients, which are pairwise orthonormal using the snapshot POD method, it should ideally also output an orthonormal multidimensional time series. Similarly, the AE's latent space is often restricted to a specific range due to a squashing latent activation function. Again, the ESN output should not lie outside this range so as not to generate latent space representations that the decoder network has difficulties deciphering. Additional incorporation of physical constraints could potentially be achievable through the previously mentioned structuring of the reservoir matrices.

Finally, this work was not concerned with the implementation of the presented ML models into an actual weather or climate model. The conventional methods of sub-grid scale parameterization used in these models necessitate stability in addition to physical accuracy and low computational costs. However, recent studies suggest that neural networks, while performing exceptionally well in offline tests, can lead to instabilities when coupled to a running atmospheric model [238]. Only if the neural network produces both reliable and stable outputs can they be coupled to long-term, large-scale simulations of our atmosphere. This can become a challenge for an autoregressive Echo State Network, where prediction errors can quickly accumulate and may even diverge. Further, the present work applied the echo state approach solely to Rayleigh-Bénard convection systems, which, due to their simplicity, do not represent essential effects like deep convection, precipitation, or the transient dynamics of an atmospheric flow. In such scenarios, one might question the suitability of an RC model, given the potentially significant differences in dynamics between the training and prediction phases. Such tasks exceed the ESN generalization procedure introduced in Section 5.1. Utilizing alternative neural network approaches that leverage transfer learning could be a viable direction to consider. Lastly, ML models have a broad range of potential applications in climate science besides data-driven parameterization models [239]. Other aspects comprise cloud detection and classification [240], uncertainty quantification [241] or pattern analysis in climate datasets [242]. All these fields employ a variety of machine learning models, which have to satisfy different requirements. This underlines the fact that there is no single model that performs best in all disciplines. After all, there is "no free lunch" [243]. While the last years have seen a great interest in ML applications in the fields of fluid dynamics and climate science, it is to be expected that certain ML models will show a significant advantage over others for certain tasks like subgrid-scale parameterization. It is, therefore, up to future research in this interdisciplinary field to ascertain whether a Reservoir Computing model will be among them. I hope that I have provided a fair starting point with my dissertation.

Appendix A

Appendix to Chapter 3

A.1 Low-order Lorenz-like model

A.1.1 Derivation of the low-order Lorenz model from the two-dimensional Boussinesq system

The starting point for the derivation is the two-dimensional Boussinesq system (2.7 - 2.9). As mentioned above, free slip boundary conditions are considered for the velocity fields $\mathbf{u}(x,z,t) = (u_x(x,z,t),\, u_z(x,z,t))^T$ and constant temperature boundary conditions for the temperature field $T(x,z,t)$. As is often done in RBC, one subtracts the non-dimensional linear temperature profile $T_{\text{diff}} = 1 - z$, corresponding to the purely conducting case, from the temperature field and uses only its fluctuations

$$\theta(x,z,t) = T(x,z,t) - T_{\text{diff}}(z) = T(x,z,t) - 1 + z. \tag{A.1}$$

The two-dimensionality and incompressibility of the flow allow for a more compact description of the problem. More precisely, the velocity field $\mathbf{u}(x,z,t)$ can be expressed in terms of a scalar stream function $\zeta(x,z,t)$

$$u_x = -\frac{\partial \zeta}{\partial z}, \quad u_z = \frac{\partial \zeta}{\partial x}. \tag{A.2}$$

This relation automatically satisfies the incompressibility condition (2.7), as $\mathbf{u} = \nabla \times \zeta \hat{\mathbf{e}}_z$. Substituting this definition into the Boussinesq system (2.8) and (2.9) and non-dimensionalizing with the free-fall scales (2.14) and (2.16) yields

$$\frac{\partial \nabla^2 \zeta}{\partial t} = \frac{\partial \zeta}{\partial z}\frac{\partial \nabla^2 \zeta}{\partial x} - \frac{\partial \zeta}{\partial x}\frac{\partial \nabla^2 \zeta}{\partial z} + \mathrm{Pr}\nabla^4 \zeta + \mathrm{RaPr}\frac{\partial \theta}{\partial x}, \tag{A.3}$$

$$\frac{\partial \theta}{\partial t} = \frac{\partial \zeta}{\partial z}\frac{\partial \theta}{\partial x} - \frac{\partial \zeta}{\partial x}\frac{\partial \theta}{\partial z} + \nabla^2 \theta + \frac{\partial \zeta}{\partial x}. \tag{A.4}$$

Further, the boundary conditions translate to

$$\zeta\big|_{z=0,1} = 0, \quad \left.\frac{\partial^2 \zeta}{\partial z^2}\right|_{z=0,1} = 0 \quad \text{and} \quad \theta\big|_{z=0,1} = 0. \tag{A.5}$$

Both the stream function and the temperature fluctuations are then expanded into a finite series of trigonometric Fourier modes

$$\zeta(x,z,t) = c_\zeta \sum_{i,j=1}^{N_\zeta} A_{ij}(t)\Phi(\alpha_i x)\sin(\beta_j z), \tag{A.6}$$

$$\theta(x,z,t) = c_\theta \sum_{k,l=1}^{N_\theta} B_{kl}(t)\Phi(\alpha_k x)\sin(\beta_l z), \tag{A.7}$$

where c_ζ and c_θ denote normalization constants. Here, $\Phi(x)$ can be either $\cos(x)$ or $\sin(x)$. $\{A_1(\tau), \dots, A_{N_\zeta}(\tau)\}$ are real amplitudes for the stream function and $\{B_1(\tau), \dots, B_{N_\theta}(\tau)\}$ the real amplitudes for the temperature fluctuations[1]. The horizontal and vertical wavenumbers are

$$\alpha_k = k\alpha = \frac{2\pi k}{\Gamma} \quad \text{and} \quad \beta_k = k\beta = k\pi. \tag{A.8}$$

Finally, N_ζ and N_θ are the leading truncation order of the thermal convection LOM. This Ansatz satisfies the boundary conditions for the stream function and temperature and encodes the spatial structure of the thermal convection flow. Finally, a low-order model of thermal convection is obtained by substituting Ansatz (A.6,A.7) into eqs. A.3 and A.4, while only considering quadratic nonlinearities

$$\frac{dA_i}{dt} = F_i(A_j, B_k, \mathrm{Pr}, \mathrm{Ra}, \delta), \tag{A.9}$$
$$\frac{dB_k}{dt} = G_k(B_l, A_i, \mathrm{Pr}, \mathrm{Ra}, \delta). \tag{A.10}$$

For convenience, the Rayleigh number is often expressed relative to the critical Rayleigh number $\mathrm{Ra}_{\mathrm{crit}} = \frac{(\alpha^2+\beta^2)^3}{\alpha^2}$ of the onset of convection for free-slip boundary condition [200], i.e.

$$r = \frac{\mathrm{Ra}}{\mathrm{Ra}_{\mathrm{crit}}} > 1. \tag{A.11}$$

Further, the aspect ratio Γ is encoded into a geometric parameter

$$\delta = \frac{4\Gamma^4}{4+\Gamma^2} \tag{A.12}$$

For a given number of truncation modes N_ζ, N_θ the specific LOM setting is defined by the parameter space (Pr, r, δ).

[1]The amplitudes A_{ij}, B_{kl} can be rewritten as A_i, B_k by viewing the former in row-major (lexicographic access) order.

A.1.2 Eight-mode Lorenz model

The full eight-mode Lorenz model is obtained for $N_\zeta = N_\theta = 4$

$$\begin{aligned}
\frac{dA_1}{d\tau} &= \mathrm{Pr}\,(B_1 - A_1) - \frac{3\beta^2 + \alpha^2}{\sqrt{2}(\alpha^2 + \beta^2)} A_2 A_3 \\
&+ \frac{3\alpha^2 - 15\beta^2}{\sqrt{2}(\alpha^2 + \beta^2)} A_3 A_4, && \text{(A.13)} \\
\frac{dA_2}{d\tau} &= -\frac{\mathrm{Pr}\delta}{4} A_2 - \frac{3}{2\sqrt{2}} A_1 A_3, && \text{(A.14)} \\
\frac{dA_3}{d\tau} &= -\mathrm{Pr}\frac{\alpha^2 + 4\beta^2}{\alpha^2 + \beta^2} A_3 - \mathrm{Pr}\frac{\alpha^2 + \beta^2}{\sqrt{2}(4\beta^2 + \alpha^2)} B_3 \\
&+ \frac{\alpha^2}{\sqrt{2}(\alpha^2 + \beta^2)} A_1 A_2 + \frac{24\beta^2 - 3\alpha^2}{\sqrt{2}(4\beta^2 + \alpha^2)} A_1 A_4, && \text{(A.15)} \\
\frac{dA_4}{d\tau} &= -\frac{9\mathrm{Pr}\delta}{4} A_4 - \frac{1}{2\sqrt{2}} A_1 A_3, && \text{(A.16)} \\
\frac{dB_1}{d\tau} &= -B_1 + rA_1 - A_1 B_2 + \frac{1}{2} A_2 B_3 + \frac{3}{2} A_4 B_3, && \text{(A.17)} \\
\frac{dB_2}{d\tau} &= -\delta B_2 + A_1 B_1, && \text{(A.18)} \\
\frac{dB_3}{d\tau} &= -\frac{\alpha^2 + 4\beta^2}{\alpha^2 + \beta^2} B_3 - A_2 B_3 + \sqrt{2} r A_3 + 3A_4 B_1 \\
&- 2\sqrt{2} A_3 B_4, && \text{(A.19)} \\
\frac{dB_4}{d\tau} &= -4\delta B_4 + \frac{3\sqrt{2}}{4} A_3 B_3. && \text{(A.20)}
\end{aligned}$$

Note that the time is rescaled to $\tau = (\alpha^2 + \beta^2)t$, where t is measured in the free-fall time scale t_f.

A.2 Hyperparameter search to the open-loop ESN in 3.3.1

A classical ESN is compared to the QRC model presented in 3.3.1. For comparison the reservoir size is fixed at $N_\mathrm{r} = 128$. The ESN grid search details are listed in table A.1. The search comprised the leaking rate, spectral radius, and regression parameter. The resulting NRMSE error landscape is shown in Fig. A.1. It can be seen that for $\lambda_\mathrm{r} = 0.05$ and $\lambda_\mathrm{r} = 0.5$, a broad range of leaking rate and spectral radius values exhibit good reservoir performances.

A.3 Quantum circuits for H1, H2 in 3.3.2.3

Here, the quantum circuits for the two QRC algorithms H1, see Fig. A.2 and H2, see Fig. A.3, used in Section 3.3.2.3 are reported. The circuit depth l poses a hyperparameter of the QRC model and denotes the number of repetitions of a block of quantum gates. It is proportional to the computational cost of running the circuit. More information on the circuit architecture and the hyperparameter dependence of, e.g., $\mathcal{E}_\mathrm{NARE}$ can be found in ref. [158].

γ_r	ϱ_r	λ_r
$[0.1, 1.0, 0.1]$	$[0.0, 1.5, 0.1]$	$\{0.05, 0.5, 5.0\}$

Reservoir realizations	$\Lambda_1 t_{\text{train}}$	$\Lambda_1 t_{\text{val}}$	$\Lambda_1 t_{\text{test}}$
100	7.10	5.00	7.00

Table A.1: Open-loop ESN **hyperparameter grid search** procedure for the eight-mode Lorenz system. The reservoir density was fixed at $D_r = 0.2$. The input scaling was $\sigma_r = 1$. Square brackets indicate the range of the searched hyperparameter: [start, end, stepsize]. The reservoir size was $N_r = 128$. Braces indicate an array of values that were used. $t_{\text{train}}, t_{\text{val}}, t_{\text{test}}$ are the length of the training, validation, and testing interval, respectively. The total number of different reservoir settings is $10 \times 16 \times 3 = 480$, while a total of $100 \times 480 = 4.8 \cdot 10^4$ reservoirs were run. The grid search run time was 2.44 h using the *turbESN* library [123].

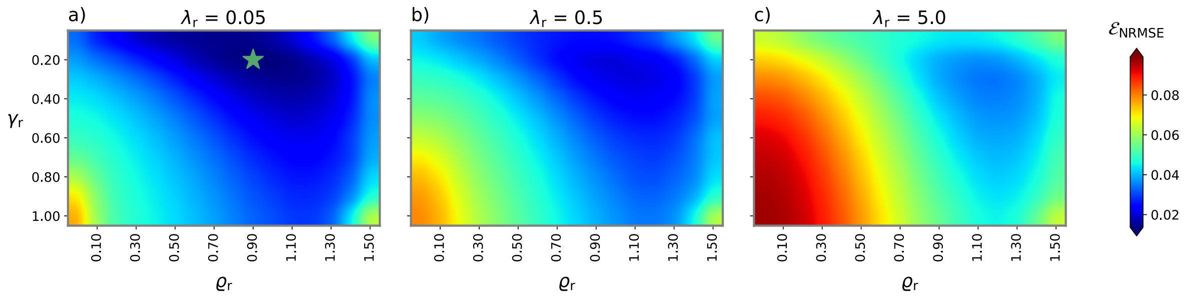

Figure A.1: Results of the **grid search** described in Table A.1. (a) Validation $\mathcal{E}_{\text{NRMSE}}$ landscape for ESN hyperparameters $(\varrho_r, \gamma_r, \lambda_r)$, $N_r = 128$. Shown is the median value of all reservoir realizations. (b) ESN outputs during the validation phase. Shown are the ground truth (blue) and the a) associated best ESN prediction (green). The corresponding optimal hyperparameter setting is marked by the green star symbol in (a).

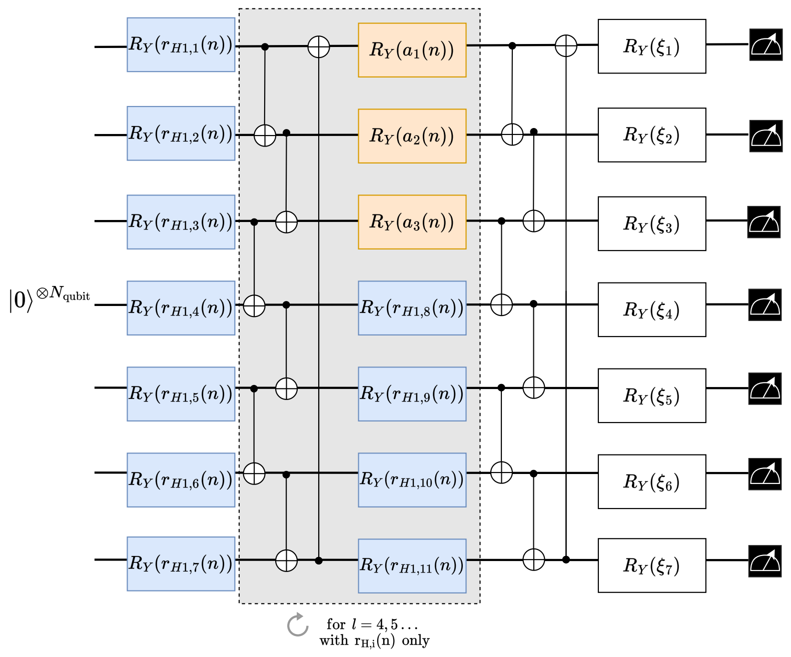

Figure A.2: **Quantum circuit** for algorithm H1 used in Section3.3.2.3. The circuit depth was $l = 5$. The variables ξ_i were chosen randomly from a uniform distribution $\mathcal{U}[0, 4\pi]$.

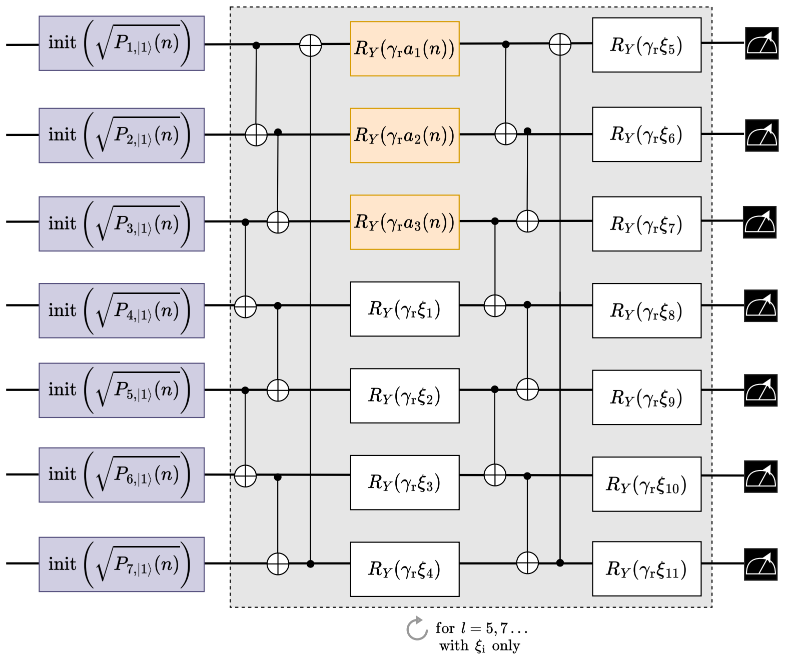

Figure A.3: **Quantum circuit** for algorithm H2 used in Section3.3.2.3. The circuit depth was $l = 5$. The variables ξ_i were chosen randomly from a uniform distribution $\mathcal{U}[0, 4\pi]$

Appendix B

Appendix to Chapter 5

B.1 Appendix to Section 5.1

B.1.1 Preliminary POD studies in Section 5.1.3

Fig. B.1 shows the cumulative POD spectrum of all four β cases. For this, all 10000 snapshots of the DNS time span $2500\langle t_f \rangle_t$ were considered. It can be seen that the POD converges very slowly and requires at least 700 modes to capture similar portions of the variance as in Sections 4.1 and 4.2. An ESN learning the reduced POD space would subsequently require a reservoir of size $N_\mathrm{r} \geq 7000$. This would impose a significant memory load, resulting in a computationally expensive hyperparameter exploration process. Hence, in Section 5.1, the POD approach was abandoned, and the Autoencoder network was applied.

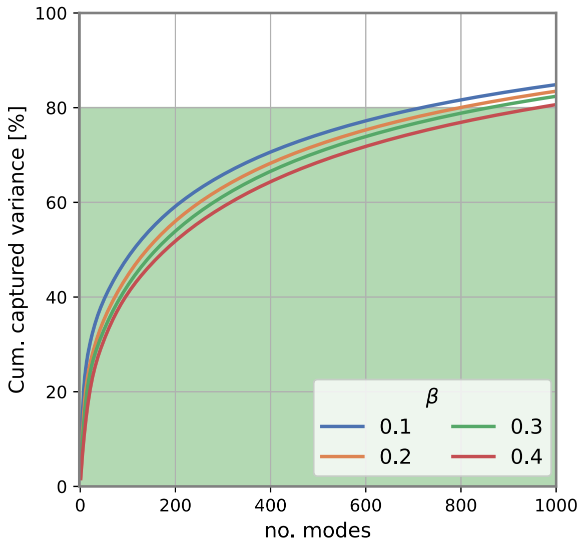

Figure B.1: Results of the preliminary POD studies for all four β values involving all 10000 snapshots in the DNS period of $2500\langle t_f \rangle_t$. It can be seen that the POD requires about $N_\mathrm{POD} \geq 700$ modes to capture 80% of the flow's variance. This, in turn, would require a reservoir dimension of at least $N_\mathrm{r} = 7000$. The green shaded area marks the captured variance of up to 80%.

B.1.2 Details on the Autoencoder implementation in Section 5.1

In Tab. B.1, two basic neural network blocks are listed. Both Encoder and Decoder consist of four such blocks. Details on the number of channels of the convolutional network, as well as the sizes of the kernels of Max-Pooling and Upsampling layers in each block, are listed in Tab. B.2. A *Python* script with the corresponding AE architecture, as well as a training script, can be found in the freely available supplementary repository to this thesis on *Zenodo* [244].

Encoder block	5× Conv2d + ELU	Max-Pooling	BatchNorm	Dropout
Decoder block	5× ConvTranspose2d + ELU	Upsample	BatchNorm	Dropout

Table B.1: Architecture of an Encoder and Decoder block. In the former two-dimensional convolutions Conv2d, the exponential linear unit (ELU) [245] activation function, as well as Max-Pooling (MP) layers, are employed. The Decoder block uses transposed convolutions ConvTranspose2d, the ELU function, and an Upsampling layer, which increases the spatial dimensions by a constant factor. Both blocks make use of batch normalization and dropout layers. The dropout probability was set to $p = 0.15$. The last Encoder block is followed by a fully connected linear layer and a Sigmoid activation. Further, the first decoding block is preceded by a linear layer as well. See also Tab. B.2 for the channel and kernel sizes.

Layer	E#1	E#2	E#3	E#4	D#1	D#2	D#3	D#4
channels	(3,8)	(8,16)	(16,8)	(8,3)	(3,8)	(8,16)	(16,8)	(8,3)
MP/Upsample kernel	(2,1)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,1)

Table B.2: Size of Max Pooling (MP)/Upsampling kernels and channels for each Encoder/Decoder block in the Autoencoder network. See also the definition of one block in Tab. B.1. The channels are given in the form (no. input channel, no. output channel), while both MP and Upsample kernel are given by (height, width). The shape of the input data was (3,30,720), where 3 denotes the three turbulence fields. Its spatial dimensions were zero-padded to the shape (3,32,1024) before entering the Encoder and cropped to the original size after passing through the Decoder. The kernel size of all convolutions and transposed convolutions was $(3, 3)$.

B.1.3 Additonal results of the Autoencoder network in Section 5.1.3

The final training and validation errors of the Autoencoder networks are listed in Tab. B.3. Moreover, the reconstructed fields u'_x, u'_z and b'' for the cases $\beta = 0.1$, 0.2, 0.3, 0.4 are compared to the corresponding DNS snapshots shown in Figs. B.2, B.3, B.4 and B.5 respectively. Furthermore, to understand what the network does to the physical fields, Fig. B.6d-g) shows four of the eight-channel outputs of Encoder block E#1. See again Table B.2 for its definition. For comparison, the input fields u'_x, u'_z, b'', in their for the AE normalized form, are shown in B.6a-c). It can be seen that the outputs possess features of all three input fields, e.g., diagonal patches in Fig.B.6a,d).

B.1.4 Additonal results of the AE-RC model in Section 5.1.4.3

This section provides additional results to the ones presented in Section 5.1.4.3. Figures B.7 and B.8 show the inferred fields of $\beta = 0.2$ and $\beta = 0.4$. Similar results to the $\beta = 0.1$ in the main text can be observed. Finally, the latent space predictions are shown in Fig. B.9 and

β	training MSE	validation MSE
0.1	$1.6 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$
0.2	$1.8 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$
0.3	$1.8 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$
0.4	$2.1 \cdot 10^{-3}$	$2.7 \cdot 10^{-3}$

Table B.3: **Mean square error loss** during the training and validation phase of the four AE networks. The cross-validation split was 8000 snapshots for training and 1000 snapshots for validation and testing each.

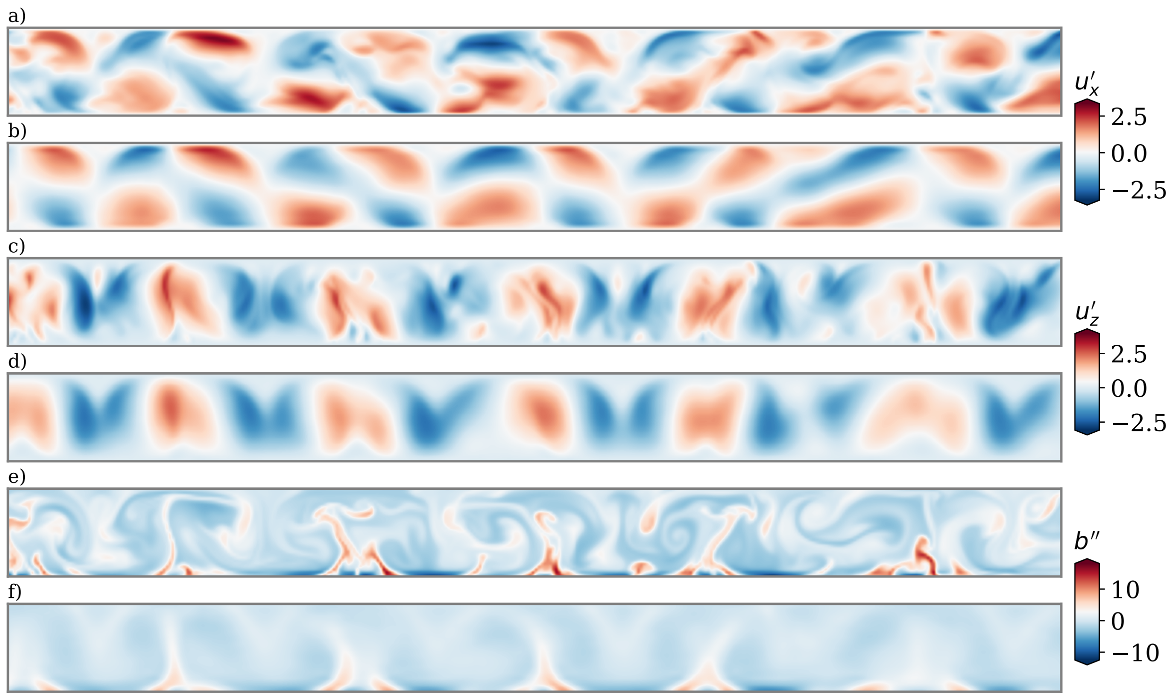

Figure B.2: Comparison between DNS (a,c,e) and reconstructed decoder fields (b,d,f) of u'_x (a,b), u'_z (c,d) and b'' (e,f). Shown is the $\beta = 0.1$ case.

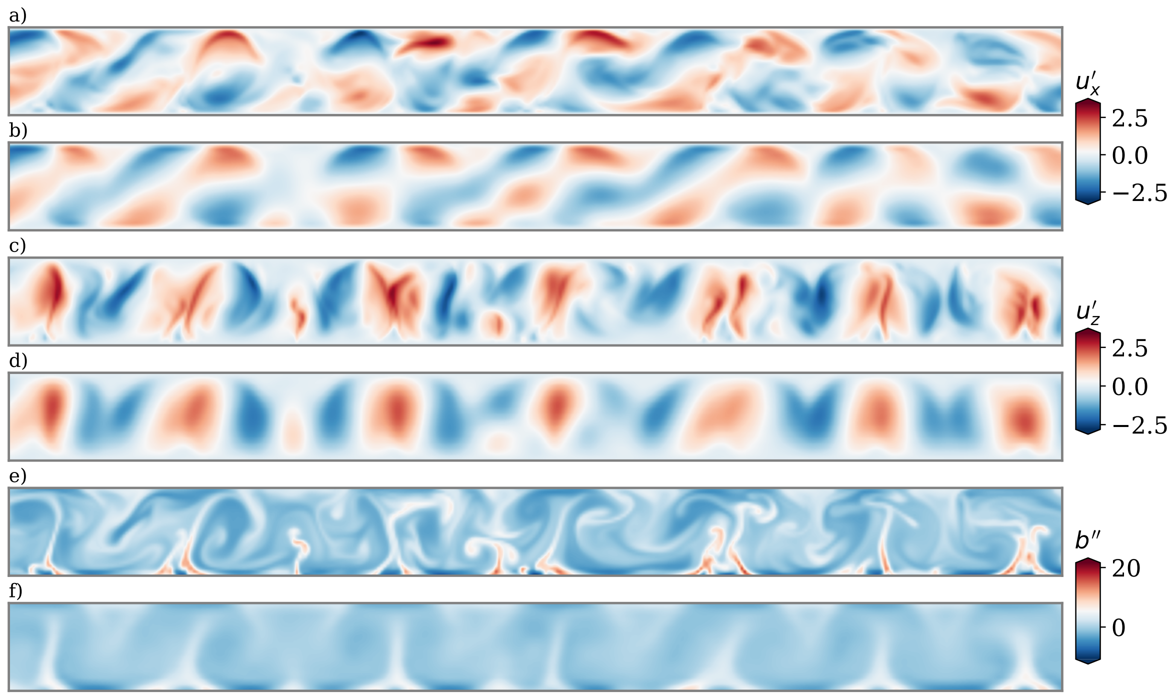

Figure B.3: Comparison between DNS (a,c,e) and reconstructed decoder fields (b,d,f) of u'_x (a,b), u'_z (c,d) and b'' (e,f). Shown is the $\beta = 0.2$ case.

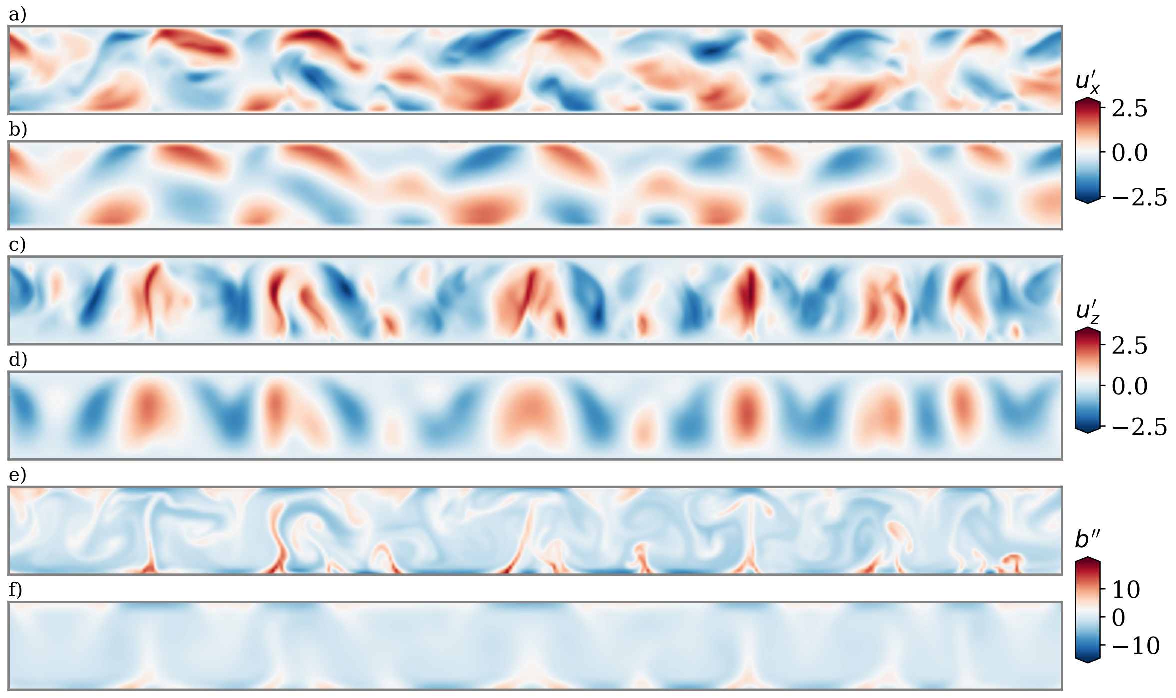

Figure B.4: Comparison between DNS (a,c,e) and reconstructed decoder fields (b,d,f) of u'_x (a,b), u'_z (c,d) and b'' (e,f). Shown is the $\beta = 0.3$ case.

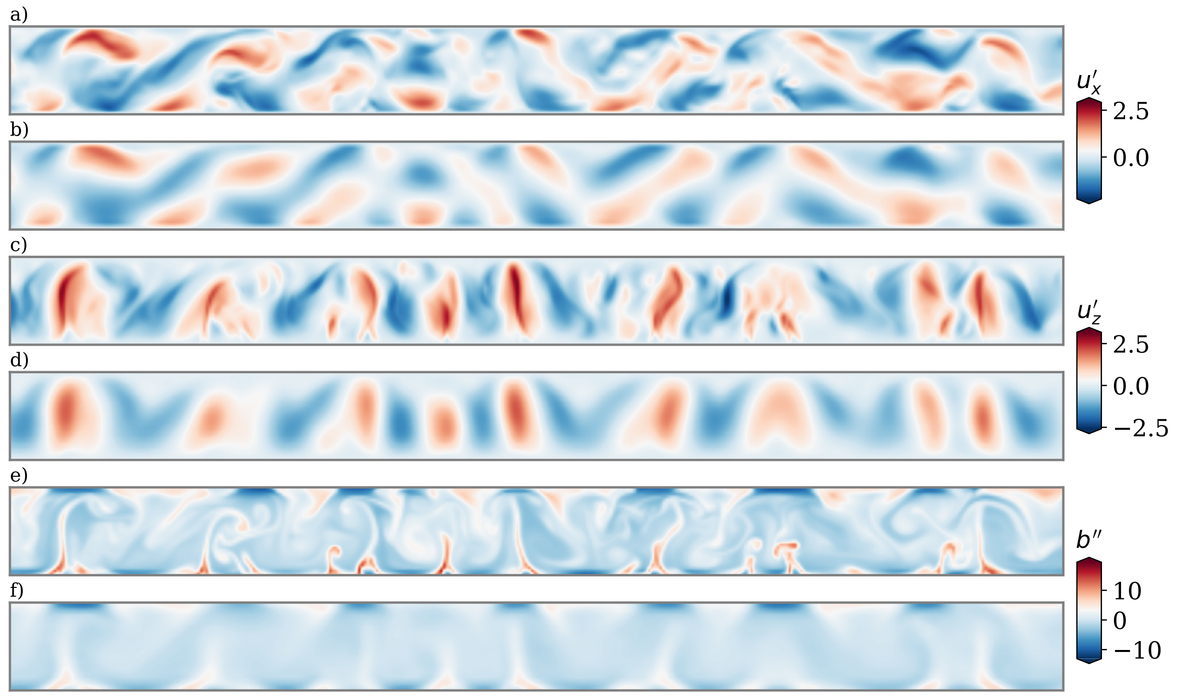

Figure B.5: Comparison between DNS (a,c,e) and reconstructed decoder fields (b,d,f) of u'_x (a,b), u'_z (c,d) and b'' (e,f). Shown is the $\beta = 0.4$ case.

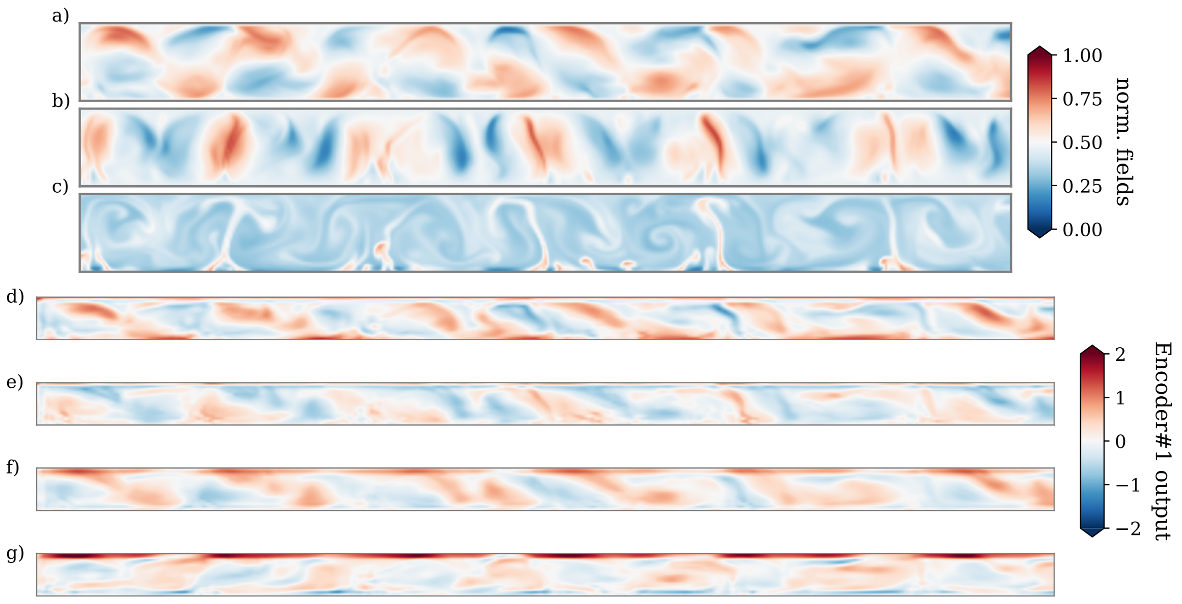

Figure B.6: Comparison between, for the AE, normalized input fields, obtained from DNS (only half the horizontal domain is shown), (a-c) and their processed form after passing through the Encoder block E#1 (d-g). See again tables B.1 and B.2 for the definition of this block. Out of the eight output channels, four are shown. Note that the vertical dimension has halved due to a pooling operation. It can be seen that the outputs possess features of the physical fields.

Fig. B.10 for the case $\beta = 0.2$ and $\beta = 0.4$, respectively. Again, the same trend as the main text can be observed for these target cases.

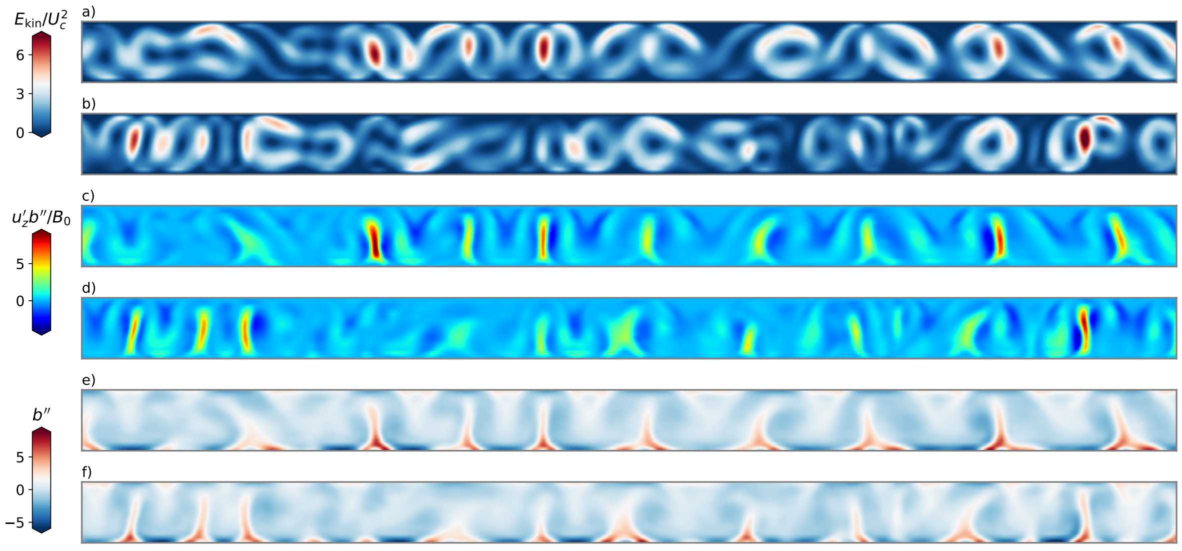

Figure B.7: Inferred fields for $\beta = 0.2$. Instantaneous snapshots of the turbulent kinetic energy (a,b), the buoyancy flux $u_z'b''$ (c,d), and the buoyancy fluctuations b''(e,f) at $t = 375\langle t_f \rangle_t$ in the middle of the testing phase. The ground truth (AE) reconstructions are (a,c,e), and the ESN predictions are (b,d,f). For visualization purposes, only 75% of the horizontal domain is shown.

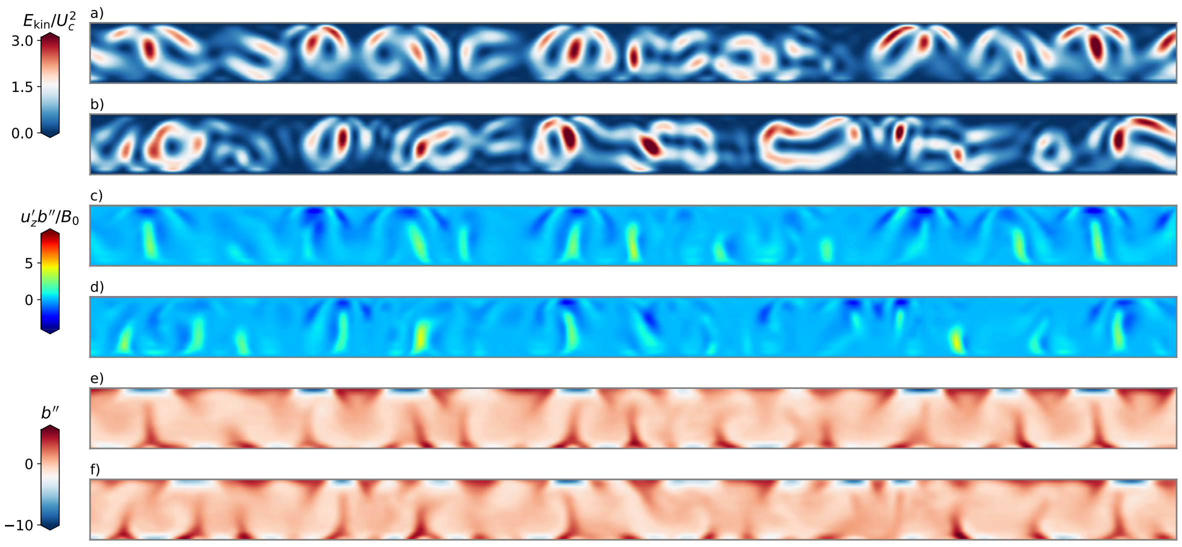

Figure B.8: Inferred fields for $\beta = 0.4$. Instantaneous snapshots of the turbulent kinetic energy (a,b), the buoyancy flux $u_z'b''$ (c,d), and the buoyancy fluctuations b''(e,f) at $t = 375\langle t_f \rangle_t$ in the middle of the testing phase. The ground truth (AE) reconstructions are (a,c,e), and the ESN predictions are (b,d,f). For visualization purposes, only 75% of the horizontal domain is shown.

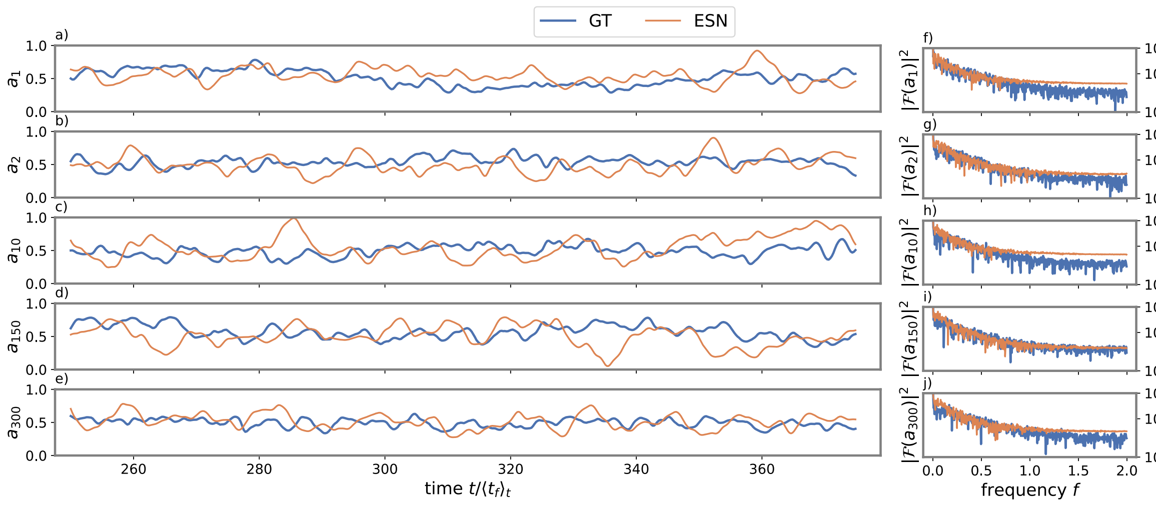

Figure B.9: Inferred time series of selected AE latent space variables for the case of $\beta = 0.2$. (a-e) The ESN output (orange) is compared to the ground truth (blue). (f-j) The corresponding Fourier power spectrum is shown to the right. It can be seen that the inferred dynamics do not capture the spectrum tails at high frequencies.

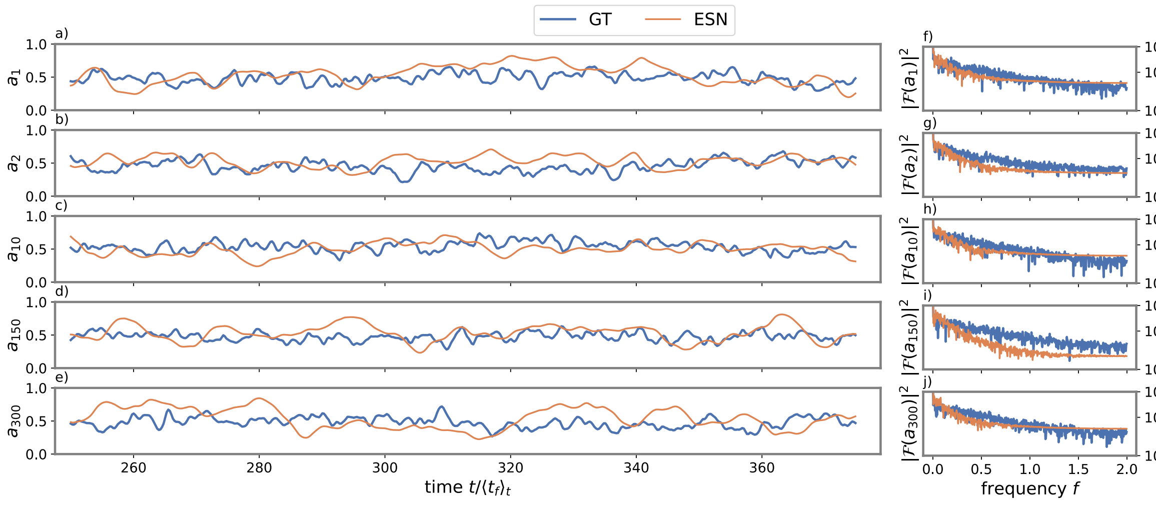

Figure B.10: Inferred time series of selected AE latent space variables for the case of $\beta = 0.4$. (a-e) The ESN output (orange) is compared to the ground truth (blue). (f-j) The corresponding Fourier power spectrum is shown to the right. Again, it can be seen that the inferred dynamics do not capture the spectrum tails at high frequencies.

B.2 Appendix to Section 5.2

B.2.1 Mass-flux parametrization

In this section, the basics of the mass-flux parametrization is described. The following formulation is used in many global numerical models of the Earth system, such as the Integrated Forecasting System of the European Centre for Medium-Range Weather Forecasts (www.ecmwf.int/en/publications/ifs-documentation). The mass-flux parametrization is one part of the eddy diffusivity mass-flux, which models the vertical profiles of the subgrid-scale fluxes of liquid-water potential temperature θ_l (convective heat flux) or total water content q_t (moisture flux) across the atmosphere.

B.2.1.1 Buoyancy flux closure by eddy diffusivity mass-flux

In the present proof-of-concept study, phase changes and cloud formation are omitted. The liquid-water potential temperature θ_l reduces then to the virtual potential temperature θ_{v} and the buoyancy b becomes

$$b(x,y,z,t) = g\frac{\theta_{\mathrm{v}}(x,y,z,t)-\theta_{\mathrm{v},0}}{\theta_{\mathrm{v},0}}\,, \tag{B.1}$$

with the acceleration due to gravity g and the virtual potential temperature at the surface $\theta_{\mathrm{v},0}$. The eddy diffusivity mass-flux closure for the subgrid-scale buoyancy flux is given by [33, 246, 247],

$$\rho_0\langle {u_z}' b'(z)\rangle_{x,y} = -\rho_0\kappa_t\frac{\partial\langle b\rangle_{x,y}}{\partial z} + M_u(\langle b(z)\rangle_u - \langle b(z)\rangle_{x,y})\,. \tag{B.2}$$

Here, ρ_0 is the constant reference density in the lower atmospheric boundary layer and M_u is the unknown mass flux. As already said in Section 5.2.2, the horizontal cross section A can be decomposed into disjoint updraft regions (u) and the remaining environment (e), i.e., $A_{\mathrm{total}} = A_u \cup A_e$. For simplicity A_{total} will simply be denoted by A to ease notation. The first term in (B.2) is the standard Boussinesq term for a turbulent stress which contains the eddy diffusivity κ_t and the mean buoyancy gradient. The additional second term comprises the mass-flux parametrization. In detail,

$$\langle {u_z}' b'(z)\rangle_{x,y} = \frac{1}{A}\int_A (u_z - \langle u_z\rangle_{x,y})(b-\langle b\rangle_{x,y})\,dA\,, \tag{B.3}$$

$$a_u\langle {u_z}' b'(z)\rangle_u = \frac{1}{A}\int_A (u_z - \langle u_z\rangle_u)(b-\langle b\rangle_u)\,dA\,, \tag{B.4}$$

$$(1-a_u)\langle {u_z}' b'(z)\rangle_e = \frac{1}{A}\int_A (u_z - \langle u_z\rangle_e)(b-\langle b\rangle_e)\,dA\,, \tag{B.5}$$

with the area fractions $a_u = A_u/A$ and $1-a_u = A_e/A$. Let us neglect the first gradient term on the right-hand side of (B.2) for a while. The second term of (B.2) can be rewritten to

$$\langle {u_z}' b'\rangle_{x,y} = a_u\langle {u_z}' b'\rangle_u + (1-a_u)\langle {u_z}' b'\rangle_e + a_u(1-a_u)[\langle u_z\rangle_u - \langle u_z\rangle_e][\langle b\rangle_u - \langle b\rangle_e]\,. \tag{B.6}$$

The first term quantified the buoyancy flux connected with A_u, the second with A_e, and the third is an exchange term due to organized turbulence, which connects both. The following approximations can be made: (i) $a_u \ll 1$, in practice mostly about 5 %; (ii) $\langle u_z\rangle_e \simeq \langle w\rangle_{x,y}$, and (iii) $\langle b\rangle_e \simeq \langle b\rangle_{x,y}$. This results in the mass-flux parametrization

$$\langle {u_z}' b'(z)\rangle_{x,y} \simeq \frac{M_u}{\rho_0}(\langle b(z)\rangle_u - \langle b(z)\rangle_{x,y}) \quad \text{with} \quad M_u = \rho_0 a_u[\langle u_z(z)\rangle_u - \langle u_z(z)\rangle_{x,y}]\,, \tag{B.7}$$

with the undetermined mass flux M_u. Recall also that the mass flux carries the physical dimension of $\mathrm{kg\,m^{-2}s^{-1}}$. A plume model is required to determine this unknown quantity which is discussed in the following.

B.2.1.2 Steady plume model

(2.9) of the Boussinesq model is the counterpart of the set of prognostic equations in a global circulation model. We apply a Reynolds decomposition and obtain the following unclosed coarse-grid equation [248]

$$\frac{\partial \langle b(z)\rangle_{x,y}}{\partial t} = -\frac{1}{\rho_0}\frac{\partial}{\partial z}[\rho_0 \langle u_z{}' b'(z)\rangle_{x,y}] + \left.\frac{\partial \langle b(z)\rangle_{x,y}}{\partial t}\right|_{\text{forcing}} . \tag{B.8}$$

All remaining terms due to horizontal mean advection and diffusion are summarized in the last term of (B.8) as a forcing term. Note that we get in this way an effective *vertical* transport model without any information on the spatial horizontal mesoscale organization of the convective turbulence which the DNS in Section5.2.2 display. The area decomposition into A_u and A_e gives

$$\langle b(z)\rangle_{x,y} = a_u \langle b(z)\rangle_u + (1-a_u)\langle b(z)\rangle_e . \tag{B.9}$$

Equations (B.6) and (B.9) are now plugged into (B.8). We use that $a_u(1-a_u) \approx a_u$ since $a_u \ll 1$ and apply additionally the approximations (ii) and (iii) from the last subsection. This leads to the following updraft budget equation (omitting the z argument),

$$\rho_0 \frac{\partial}{\partial t}[a_u \langle b\rangle_u] = -\frac{\partial}{\partial z}[\rho_0 a_u \langle u_z{}' b'\rangle_u] - \frac{\partial}{\partial z}[\underbrace{\rho_0 a_u(\langle u_z\rangle_u - \langle u_z\rangle_{x,y})}_{=M_u}\langle b\rangle_u] + E\langle b\rangle_e - D\langle b\rangle_u + \rho_0 a_u \left.\frac{\partial \langle b\rangle_{x,y}}{\partial t}\right|_{\text{forcing}} , \tag{B.10}$$

and the corresponding downdraft budget equation

$$\rho_0 \frac{\partial}{\partial t}[(1-a_u)\langle b\rangle_e] = -\frac{\partial}{\partial z}[\rho_0 (1-a_u)\langle u_z{}' b'\rangle_e] + \frac{\partial}{\partial z}[M_u \langle b\rangle_e] - E\langle b\rangle_e + D\langle b\rangle_u + \rho_0(1-a_u) \left.\frac{\partial \langle b\rangle_{x,y}}{\partial t}\right|_{\text{forcing}} . \tag{B.11}$$

Two terms in both equations have been added to express the mass exchange due to entrainment and detrainment with rates E and D, respectively. Adding both equations brings us back to (B.8). Furthermore, a continuity equation is required, which can be obtained from one of the budget equations by substituting $\langle b\rangle$ with 1. This gives

$$\rho_0 \frac{\partial a_u}{\partial t} = -\frac{\partial M_u}{\partial z} + E - D . \tag{B.12}$$

Next, a (statistically) steady state is assumed such that (B.10) and (B.12) simplify to, see also ref. [33],

$$\frac{\partial}{\partial z}[M_u \langle b\rangle_u] = -\frac{\partial}{\partial z}[\rho_0 a_u \langle u_z{}' b'\rangle_u] + M_u \varepsilon \langle b\rangle_e - M_u \delta \langle b\rangle_u , \tag{B.13}$$

$$\frac{\partial M_u}{\partial z} = M_u(\varepsilon - \delta) , \tag{B.14}$$

where fractional entrainment and detrainment rates follow from $E = M_u\varepsilon$ and $D = M_u\delta$, respectively. When the first term on the right-hand side of (B.13) is set to zero, both equations can be combined to give the following simple updraft equation

$$\frac{\partial \langle b\rangle_u}{\partial z} = \varepsilon(\langle b\rangle_{x,y} - \langle b\rangle_u) . \tag{B.15}$$

Furthermore, we need an equation for $\langle u_z \rangle_u$ such that we can close the buoyancy flux model [33, 249]. This is obtained from the momentum balance for the updrafts, assuming a steady regime again. This gives

$$-\frac{1}{2}\frac{\partial \langle u_z \rangle_u^2}{\partial z} - C_1 \varepsilon \langle u_z \rangle_u^2 - \frac{1}{\rho_0}\frac{\partial \langle p \rangle_{x,y}}{\partial z} + \langle b \rangle_u - \langle b \rangle_{x,y} = 0\,. \tag{B.16}$$

This equation went again through a number of simplification steps: (i) the first term results from the nonlinear advection of $\langle u_z \rangle_u$, (ii) the second term encloses all remaining advection terms, (iii) the effective entrainment rate which is connected with the turbulent mixing in (ii) is assumed to be directly proportional to the entrainment for $\langle b \rangle_u$ in (B.15). (iv) The pressure gradient term is approximated finally to be $\partial(\mu \langle u_z \rangle_u)/\partial z$. This is the last step, and the *mass-flux parametrization* comprises the following coupled system of equations which would have to be solved together with the mean prognostic equation for $\langle b \rangle_{x,y}$,

$$\frac{\partial \langle b \rangle_u}{\partial z} = \varepsilon(\langle b \rangle_{x,y} - \langle b \rangle_u)\,, \tag{B.17}$$

$$\frac{1}{2}(1-2\mu)\frac{\partial \langle u_z \rangle_u^2}{\partial z} + C_1 \varepsilon \langle u_z \rangle_u^2 = \langle b \rangle_u - \langle b \rangle_{x,y}\,. \tag{B.18}$$

If $\langle u_z \rangle_u$ and $\langle b \rangle_u$ are known, the second term on the right-hand side of the EDMF model in (B.2) can be computed for a given a_u. The model parameters ε, C_1, and μ are however unknown. Large eddy simulations studies by Siebesma et al. [33] suggest $C_1 \simeq 0.5$ and $\mu \simeq 0.15$. The fractional entrainment rate ε, which will be height dependent, is more recently included as a stochastic multi-plume model, i.e., for each plume i the EDMF model equations are solved. The entrainment rates ε_i are chosen from a Poisson distribution [247].

B.2.1.3 Numerical Implementation

The eddy-diffusivity κ_t is implemented by a simple profile method [250] and results to

$$\kappa_t(z^*) = k\,(39kz^*)^{1/3}\, z^*\,(1-z^*)^2\, U_c h\,, \tag{B.19}$$

where $k \approx 0.4$ is the von Kármán constant. The mass-flux contributions were obtained by integrating equations (B.17) and (B.18) using a fourth-order Runge-Kutta scheme. The starting height $z_0 > 0$ together with the boundary conditions are chosen as

$$\langle b(z_0) \rangle_u = \langle b(z_0) \rangle + \alpha B_0/\sigma_{u_z}\,, \tag{B.20}$$

$$\langle u_z(z_0) \rangle_u^2 = \sigma_{u_z}^2(z_0), \tag{B.21}$$

where an empirical standard deviation σ_{u_z} can be used, such as in ref. [251] which is given by

$$\sigma_{u_z}(z^*)/U_c \simeq 1.3\,(0.6z^*)^{1/3}\,(1-z^*)^{1/2}\,, \tag{B.22}$$

which fits well for the present direct numerical simulations, as seen in Fig. B.11. Furthermore, an entrainment rate profile as suggested in [33, 252] is used. It is given by

$$\varepsilon \simeq 0.4\left(\frac{1}{z} + \frac{1}{h-z}\right). \tag{B.23}$$

These inputs lead to the profiles in Fig. 5.24 of the main text, which was used to compare this closure with the machine-learning results.

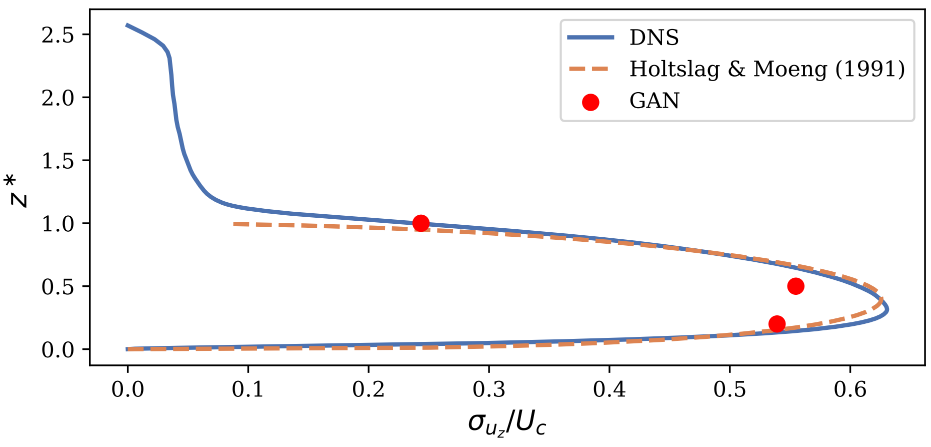

Figure B.11: Standard deviation of vertical velocity w obtained from the direct numerical simulation (blue) in comparison to the expression (B.22) (orange). The GAN values are shown in red.

B.2.2 Supplementary information about the GAN used in Section 5.2.3

B.2.2.1 Details on the GAN architecture

Here, a generator with a U-shaped deep neural network structure, in short U-Net [68, 253], is used. A U-Net applies additional residual connections between the contraction and expansion paths of the network and is used for complex segmentation tasks with few training examples. In the present case, it has seven contraction and seven subsequent expansion layers. After each contraction layer, the spatial dimensions are halved via a convolution operation, while the channel dimensions are doubled. The opposite is done in the expansion path via transposed convolutions. Both paths are connected via residual connections, allowing for a direct information flow between two layers in the contraction and expansion paths of the U-shape. The activation function was chosen as the *Parametric Rectified Linear Unit* [254] (PReLU)

$$\mathrm{PReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{otherwise} \end{cases} \tag{B.24}$$

where the slope a is an additional learnable parameter. The discriminator network consists of eight convolution layers, which reduce the input data to a scalar score. Here, the PReLU activation was used as well. Finally, to increase the variance of the network, we use Dropout layers, which set the layer output to zero with a probability of 0.3. A *Python* script with the corresponding GAN architecture can be found in the freely available supplementary repository to this thesis on *Zenodo* [244].

B.2.2.2 Details on the GAN training procedure

Algorithm 1 illustrates the procedure of training a GAN. The listed hyperparameters in the algorithm are the ones used in the present study. A training script can be found in the freely available supplementary repository to this thesis on *Zenodo* [244].

Algorithm 1 WGAN with gradient penalty. Default values: $m = 64$, $\beta_1 = 0.9$, $\beta = 0.999$.

```
Require: batch size $m$, Adam parameters $\beta_1$, $\beta_2$, learning rate $\gamma$, number of discriminator
    iterations per generator iterations $N_D/N_G$, gradient penalty parameter $\lambda$.
Require: initial discriminator and generator weights $\theta_D$, $\theta_G$, $\mathcal{L}_{\text{WGAN}} \leftarrow \infty$
 1: while $\mathcal{L}_{\text{WGAN}}$ has not converged do
 2:     for $j = 1, ..., N_D/N_G$ do
 3:         for $i = 1, ..., m$ do
 4:
 5:             Sample real data $\mathbf{x} \sim \mathbb{P}_r$, latent variable $\xi \sim \mathcal{N}(0, 1)$
 6:             $\mathbf{x}_G \leftarrow G_{\theta_G}(\xi)$
 7:             $\mathcal{L}^{(i)}_{\text{WGAN}} \leftarrow D_{\theta_D}(\mathbf{x}) - D_{\theta_D}(\mathbf{x}_G)$
 8:
 9:             Sample random number $\chi \sim U[0, 1]$.
10:             $\hat{\mathbf{x}} \leftarrow \chi\mathbf{x} + (1 - \chi)\mathbf{x}_G$
11:             $\mathcal{L}^{(i)}_D \leftarrow -\mathcal{L}^{(i)}_{\text{WGAN}} + \lambda \left(\|\nabla_{\hat{\mathbf{x}}} D_{\theta_D}(\hat{\mathbf{x}})\|_2 - 1\right)^2$
12:
13:         end for
14:         end for
15:         $\theta_D \leftarrow \text{Adam}(\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}_D, \theta_D, \gamma, \beta_1, \beta_2)$        ▷ Train Discriminator
16:     end for
17:     end for
18:
19:     Sample batch of latent variables $\{\xi^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, 1)$
20:     $\mathcal{L}_G \leftarrow \frac{1}{m} \sum_{i=1}^{m} \left[-D_{\theta_D}\left(G_{\theta_G}(\xi^{(i)})\right)\right]$
21:     $\theta_G \leftarrow \text{Adam}(\nabla_{\theta_G} \mathcal{L}_G, \theta_G, \gamma, \beta_1, \beta_2)$        ▷ Train Generator
22:
23: end while
24: end while
```

Appendix C
Technical Appendix

C.1 Direct numerical simulations

The DNS of RBC in Sections 3.3.2, 4.1 and 4.2 were conducted using the spectral element solver *Nek5000* [180]. Moreover, the spectral interpolation to the uniform grids $N_x \times N_z$ was also performed using the Nek code. The corresponding run files can be found in the supplementary *Zenodo* repository available in [244]. All DNS runs were performed at the high-performance computing cluster of TU Ilmenau. The DNS of the RBC-like system in Section 5.1 and the dry CBL in Section 5.2 were conducted by Juan Pedro Mellado (Department of Earth System Sciences, Meteorologisches Institut, Universität Hamburg) using computational resources at the SuperMUC supercomputer in Garching. For this, the *tlab* solver [183] was employed.

C.2 Echo State Network implementation

For this thesis, the ESN framework was implemented in the shape of a *Pyhton* package *turbESN*. It is available on GitHub at [123]. All computations were performed on the computing cluster of TU Ilmenau. Note that a single ESN run is easily computed on a PC with moderate computational capabilities. For grid search procedures and multiple reservoir realizations, the package makes use of Python's multiprocessing package. For larger grid search procedures, 24 to 96 CPUs are recommended. All ESN scripts for the results of this thesis can be found in the *Zenodo* repository at [244].

C.3 Quantum Reservoir Computing

All computations with the QRC models were performed by my colleague Philipp Pfeffer at TU Ilmenau. For this, he employed IBM's Python library *Qiskit* [204].

C.4 Proper Orthogonal Decomposition

The POD of a given DNS dataset is readily computed using the steps explained in Section 2.3.1. The computation of POD time coefficients and POD spatial modes was done at the computing cluster of TU Ilmenau. The *Python* POD code for obtaining the results of this thesis is available in the *Zenodo* repository at [244].

C.5 Autoencoder and GAN implementations

Autoencoder and Generative Adversarial networks were implemented using the *Pyhton* Deep-Learning library *PyTorch* [255]. Both models were run using up to four Nvidia A100 Tensor

Core GPUs at the computing cluster at TU Ilmenau. All scripts are available in the *Zenodo* repository at [244].

C.6 Post-Processing

All post-processing (e.g., running the trained neural networks, computations based on the simulated flows, figures, etc.) was done in *Python*, mainly using IPython Notebooks. All scripts are available in the *Zenodo* repository at [244].

List of peer-reviewed publications

108. Pfeffer, P., Heyder, F. & Schumacher, J. Hybrid quantum-classical reservoir computing of thermal convection flow. *Phys. Rev. Res.* **4,** 033176 (2022).

113. Ghazijahani, M. S., Heyder, F., Schumacher, J. & Cierpka, C. On the benefits and limitations of Echo State Networks for turbulent flow prediction. *Meas. Sci. Technol.* **34,** 014002 (2022).

114. Ghazijahani, M. S., Heyder, F., Schumacher, J. & Cierpka, C. Spatial prediction of the turbulent unsteady von Kármán vortex street using Echo State Networks. *Phys. Fluids* **35,** 115141 (2023).

117. Heyder, F. & Schumacher, J. Echo state network for two-dimensional turbulent moist Rayleigh-Bénard convection. *Phys. Rev. E* **103,** 053107 (2021).

120. Heyder, F., Mellado, J. P. & Schumacher, J. Generalizability of reservoir computing for flux-driven two-dimensional convection. *Phys. Rev. E* **106,** 055303 (2022).

158. Pfeffer, P., Heyder, F. & Schumacher, J. Reduced-order modeling of two-dimensional turbulent Rayleigh-Bénard flow by hybrid quantum-classical reservoir computing. *Phys. Rev. Res.* **5** (2023).

List of non-peer-reviewed publications

Besides the peer-reviewed publications listed above, the results of this thesis were presented at several international conferences. See refs. [256–259]. Moreover, parts of the presented work are documented in ref. [260].

219. Heyder, F., Mellado, J. P. & Schumacher, J. Generative convective parametrization of dry atmospheric boundary layer. *arXiv:2307.14857* (2023).

256. Heyder, F., Pandey, S. & Schumacher, J. *Reservoir Computing of Dry and Moist Turbulent Convection* 73th Annual Meeting of the APS Division of Fluid Dynamics. 2020.

257. Heyder, F., Mellado, J. P. & Schumacher, J. *Two-Dimensional Convective Boundary Layer: Numerical Analysis and Echo State Network model* 74th Annual Meeting of the APS Division of Fluid Dynamics. 2021.

258. Heyder, F. & Schumacher, J. *Reservoir computing of three-dimensional turbulent convection* 8th European Congress on Computational Methods in Applied Sciences and Engineering. 2022.

259. Heyder, F., Mellado, J. P. & Schumacher, J. *Reservoir computing of a three-dimensional turbulent atmospheric boundary layer* 18th European Turbulence Conference. 2023.

260. Vieweg, P., Heyder, F., John, J. P. & Schumacher, J. *Analysis of the Large-Scale Order in Turbulent Mesoscale Convection* in *NIC Symposium 2022 Proceedings* (eds Müller, M., Peter, C. & Trautmann, A.) (Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Jülich, 2022), 405–415.

References

1. Schumacher, J. & Sreenivasan, K. R. Colloquium: Unusual dynamics of convection in the Sun. *Rev. Mod. Phys.* **92,** 041001 (2020).
2. Christensen, U. Effects of phase transitions on mantle convection. *Annu. Rev. Earth Planet. Sci.* **23,** 65–87 (1995).
3. Pauluis, O. & Schumacher, J. Radiation Impacts on Conditionally Unstable Moist Convection. *J. Atmos. Sci.* **70,** 1187–1203 (2013).
4. Stevens, B. Atmospheric moist convection. *Annu. Rev. Earth Planet. Sci.* **33,** 605–643 (2005).
5. Normand, C., Pomeau, Y. & Velarde, M. G. Convective instability: A physicist's approach. *Rev. Mod. Phys.* **49,** 581–624 (3 1977).
6. Bénard, H. Les Tourbillons Cellulaires dans une Nappe Liquide. *Revue Générale des Sciences Pures et Appliquées,* 1261–1271 and 1309–1328 (1900).
7. Pearson, J. On convection cells induced by surface tension. *J. Fluid Mech.* **4,** 489–500 (1958).
8. Rayleigh, L. On the Convective Currents in a Horizontal Layer of Fluid when the Higher Temperature is on the Under Side. *Philosophical Magazine* **6,** 529–546 (1916).
9. Hartlep, T., Tilgner, A. & Busse, F. H. Large Scale Structures in Rayleigh-Bénard Convection at High Rayleigh Numbers. *Phys. Rev. Lett.* **91,** 064501 (6 2003).
10. Pandey, A., Scheel, J. D. & Schumacher, J. Turbulent superstructures in Rayleigh-Bénard convection. *Nat. Commun.* **9,** 2118 (2018).
11. Stevens, R. J. A. M., Blass, A., Zhu, X., Verzicco, R. & Lohse, D. Turbulent thermal superstructures in Rayleigh-Bénard convection. *Phys. Rev. Fluids* **3,** 041501 (4 2018).
12. Moller, S., Resagk, C. & Cierpka, C. Long-time experimental investigation of turbulent superstructures in Rayleigh–Bénard convection by noninvasive simultaneous measurements of temperature and velocity fields. *Exp. Fluids* **62,** 64 (2021).
13. Vieweg, P. P., Scheel, J. D. & Schumacher, J. Supergranule aggregation for constant heat flux-driven turbulent convection. *Phys. Rev. Res.* **3,** 013231 (2021).
14. Chillà, F. & Schumacher, J. New perspectives in turbulent Rayleigh-Bénard convection. *Eur. Phys. J. E.* **35,** 58 (2012).
15. Helbig, M. *et al.* Integrating continuous atmospheric boundary layer and tower-based flux measurements to advance understanding of land-atmosphere interactions. *Agric. For Meteorol.* **307,** 108509 (2021).
16. Stull, R. B. *An Introduction to Boundary Layer Meteorology* ISBN: 978-9400930278 (Springer Netherlands, 2012).
17. Ghil, M. & Lucarini, V. The physics of climate variability and climate change. *Rev. Mod. Phys.* **92,** 035002 (2020).

18. Kaspar, F. *et al.* Regional atmospheric reanalysis activities at Deutscher Wetterdienst: review of evaluation results and application examples with a focus on renewable energy. *Adv. Sci. Res.* **17,** 115–128 (2020).

19. Crueger, T. *et al.* ICON-A, The Atmosphere Component of the ICON Earth System Model: II. Model Evaluation. *J. Adv. Mod. Earth Sys.* **10,** 1638–1662 (2018).

20. Chen, D. *et al. Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* 147–286 (Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2021).

21. Honnert, R. *et al.* The Atmospheric Boundary Layer and the Gray Zone of Turbulence: A Critical Review. *J. Geophys. Res. Atmos.* **125,** e2019JD030317 (2020).

22. Boussinesq, J. Essai sur la théorie des eaux courantes, Mémoires présentés par divers savants. *Mémoires présentés par divers savants à l'Académie des Sciences,* 1–680 (1877).

23. Prandtl, L. Bericht über Untersuchungen zur ausgebildeten Turbulenz. *Z. Angew. Math. Mech.* **5,** 136–139 (1925).

24. Smagorinsky, J. General circulation experiments with the primitive equations. *Mon. Weather Rev.* **91,** 99–164 (1963).

25. Troen, I. B. & Mahrt, L. A simple model of the atmospheric boundary layer sensitivity to surface evaporation. *Boundary Layer Meteorol.* **37,** 129–148 (1986).

26. Louis, J. F. A parametric model of vertical eddy fluxes in the atmosphere. *Boundary Layer Meteorol.* **17,** 187–202 (1979).

27. Lykossov, V. N. K-theory of atmospheric turbulent planetary boundary layer and the Bussinesque generalized hypothesis. *Russ. J. Numer. Anal. Math. Model.* **5,** 221–240 (1990).

28. Yano, J. I. Subgrid-scale physical parameterization in atmospheric modeling: How can we make it consistent? *J. Phys. A Math. Theor.* **49,** 284001 (2016).

29. Edwards, J. M., Beljaars, A. C. M., Holtslag, A. A. M. & Lock, A. P. Representation of boundary-layer processes in numerical weather prediction and climate models. *Bound.-Layer Meteorol.* **177,** 511–539 (2020).

30. Manabe, S. & Strickler, R. F. Thermal equilibrium of the atmosphere with a convective adjustment. *J. Atmos. Sci.* **21,** 361–385 (1964).

31. Kuo, H. L. On formation and intensification of tropical cyclones through latent heat release by cumulus convection. *J. Atmos. Sci* **22,** 40–63 (1965).

32. Ooyama, K. A theory on parameterization of cumulus convection. *J. Meteorol. Soc. Japan* **49,** 744–756 (1971).

33. Siebesma, A. P., Soares, P. M. M. & Teixeira, J. A combined eddy-diffusivity mass-flux approach for the convective boundary layer. *J. Atmos. Sci.* **64,** 1230–1248 (2007).

34. Arakawa, A. & Schubert, W. H. Interaction of a cumulus cloud ensemble with the large-scale environment, Part I. *J. Atmos. Sci.* **31,** 674–701 (1974).

35. Emanuel, K. A. A scheme for representing cumulus convection in large-scale models. *J. Atmos. Sci.* **48,** 2313–2329 (1991).

36. Randall, D. A., Ding, P. & Pan, D.-M. in *The Physics and Parameterization of Moist Atmospheric Convection* (ed Smith, R. K.) 281–296 (Springer Netherlands, Dordrecht, 1997). ISBN: 978-94-015-8828-7.

37. Bauer, P., Thorpe, A. & Brunet, G. The quiet revolution of numerical weather prediction. *Nature* **525,** 47–55 (2015).

38. Arakawa, A. The Cumulus Parameterization Problem: Past, Present, and Future. *J. Clim.* **17,** 2493–2525 (2004).

39. Randall, D., Khairoutdinov, M., Arakawa, A. & Grabowski, W. Breaking the cloud parameterization deadlock. *Bull. Am. Meteorol. Soc.* **84,** 1547–1564 (2003).

40. Hannah, W. M. *et al.* Initial Results From the Super-Parameterized E3SM. *J. Adv. Mod. Earth Sys.* **12,** e2019MS001863 (2020).

41. Grabowski, W. W. & Smolarkiewicz, P. K. CRCP: a Cloud Resolving Convection Parameterization for modeling the tropical convecting atmosphere. *Physica D* **133,** 171–178 (1999).

42. Grabowski, W. W. Coupling cloud processes with the large-scale dynamics using the Cloud-Resolving Convection Parameterization (CRCP). *J. Atmos. Sci.* **58,** 978–997 (2001).

43. Khairoutdinov, M. F. & Randall, D. A. A cloud resolving model as a cloud parameterization in the NCAR Community Climate System Model: Preliminary results. *Geo. Phys. Lett.* **28,** 3617–3620 (2001).

44. Jung, J.-H. & Arakawa, A. Development of a Quasi-3D Multiscale Modeling Framework: Motivation, Basic Algorithm and Preliminary results. *J. Adv. M. Earth Syst.* **2,** 31 (2010).

45. Campin, J.-M., Hill, C., Jones, H. & Marshall, J. Super-parameterization in ocean modeling: Application to deep convection. *Ocean Model.* **36,** 90–101 (2011).

46. Parishani, H., Pritchard, M. S., Bretherton, C. S., Wyant, M. C. & Khairoutdinov, M. Toward low-cloud-permitting cloud superparameterization with explicit boundary layer turbulence. *J. Adv. Mod. Earth Sys.* **9,** 1542–1571 (2017).

47. Jansson, F. *et al.* Regional Superparameterization in a Global Circulation Model Using Large Eddy Simulations. *J. Adv. Mod. Earth Sys.* **11,** 2958–2979 (2019).

48. Mooers, G. *et al.* Assessing the Potential of Deep Learning for Emulating Cloud Superparameterization in Climate Models With Real-Geography Boundary Conditions. *J. Adv. Mod. Earth Sys.* **13** (2021).

49. Gomez-Uribe, C. A. & Hunt, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* **6,** 13 (2016).

50. Wu, Y. *et al.* Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144* (2016).

51. Xiong, W. *et al.* The Microsoft 2017 Conversational Speech Recognition System. *arXiv:1708.06073* (2017).

52. Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification* in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 1701–1708.

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

54. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5,** 115–133 (1943).

55. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65,** 386–408 (1958).

56. Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* (Spartan Books, Buffalo, NY, USA, 1962).

57. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **2,** 359–366 (1989).

58. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals, Syst.* **2,** 303–314 (1989).

59. Arens, T. *et al. Mathematik* (Springer Berlin Heidelberg, 2018).

60. Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701* (2012).

61. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (2017).

62. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323,** 533–536 (1986).

63. Kutz, J. N. Deep learning in fluid dynamics. *J. Fluid Mech.* **814,** 1–4 (2017).

64. Pandey, S., Schumacher, J. & Sreenivasan, K. R. A perspective on machine learning in turbulent flows. *Journal of Turbulence* **21,** 567–584 (2020).

65. Vinuesa, R. & Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science* **2,** 358–366 (2022).

66. Guan, Y., Chattopadhyay, A., Subel, A. & Hassanzadeh, P. Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning. *J. Comput. Phys.* **458,** 111090 (2022).

67. List, B., Chen, L.-W. & Thuerey, N. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *J. Fluid Mech.* **949,** A25 (2022).

68. E. Fonda, E., Pandey, A., Schumacher, J. & K. R. Sreenivasan, K. R. Deep learning in turbulent convection networks. *Proc. Natl. Acad. Sci. USA* **116,** 8667–8672 (2019).

69. Park, J. & Choi, H. Machine-learning-based feedback control for drag reduction in a turbulent channel flow. *J. Fluid Mech.* **904,** A24 (2020).

70. Ren, F., Hu, H.-b. & Tang, H. Active flow control using machine learning: A brief review. *J. Hydrodyn.* **32,** 247–253 (2020).

71. Fukami, K., Fukagata, K. & Taira, K. Super-resolution analysis via machine learning: a survey for fluid flows. *Theor. Comput. Fluid Dyn.* **37,** 421–444 (2023).

72. Mohan, A. T., Lubbers, N., Chertkov, M. & Livescu, D. Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence. *Phys. Rev. Fluids* **8,** 014604 (2023).

73. Obayashi, W., Aono, H., Tatsukawa, T. & Fujii, K. Feature extraction of fields of fluid dynamics data using sparse convolutional autoencoder. *AIP Adv.* **11,** 105211 (2021).

74. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378,** 686–707 (2019).

75. Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367,** 1026–1030 (2020).

76. Doya, K. *Bifurcations in the learning of recurrent neural networks* in *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems* **6** (1992), 2777–2780.

77. Bengio, Y., Frasconi, P. & Simard, P. *The problem of learning long-term dependencies in recurrent networks* in *IEEE International Conference on Neural Networks* **3** (IEEE, 1993), 1183–1188.

78. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* **5,** 157–166 (1994).

79. Pascanu, R., Mikolov, T. & Bengio, Y. *On the difficulty of training recurrent neural networks* in *Proceedings of the 30th International Conference on Machine Learning* **28** (PMLR, Atlanta, Georgia, USA, 2013), 1310–1318.

80. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks. *GMD-Forschungszentrum Informationstechnik Technical Report* **148** (2001).

81. Maass, W., Natschläger, T. & Markram, H. Real-Time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14,** 2531–2560 (2002).

82. Salmen, M. & Ploger, P. *Echo State Networks used for Motor Control* in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (2005), 1953–1958.

83. Buteneers, P. *et al.* Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing. *Artif. Intell. Med.* **53,** 215–223 (2011).

84. Skowronski, M. D. & Harris, J. G. Automatic speech recognition using a predictive echo state network classifier. *Neural Networks* **20,** 414–423 (2007).

85. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys. Rev. Lett.* **120,** 024102 (2018).

86. Pathak, J. *et al.* Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos* **28,** 041101 (2018).

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

88. Pershin, A., Beaume, C., Li, K. & Tobias, S. M. Training a neural network to predict dynamics it has never seen. *Phys. Rev. E* **107,** 014304 (2023).

89. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **9,** 1735–1780 (1997).

90. Cho, K., van Merrienboer, B., Bahdanau, D. & Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259* (2014).

91. Wallace, J. & Hobbs, P. *Atmospheric Science: An Introductory Survey* chap. Atmospheric Thermodynamics. ISBN: 9780127329512 (Elsevier Academic Press, 2006).

92. Emanuel, K. A. *Atmospheric Convection* 580 pp. (Oxford University Press, New York, 1994).

93. Verhoeven, J., Wiesehöfer, T. & Stellmach, S. Anelastic versus fully compressible turbulent Rayleigh Bénard convection. *Astrophys. J.* **805,** 62 (2015).

94. Batchelor, G. K. The conditions for dynamical similarity of motions of a frictionless perfect-gas atmosphere. *Q. J. R. Meteorol. Soc.* **79,** 224–235 (1953).

95. Gough, D. O. The Anelastic Approximation for Thermal Convection. *J. Atmos. Sci.* **26,** 448–456 (1969).

96. Lantz, S. R. & Fan, Y. Anelastic Magnetohydrodynamic Equations for Modeling Solar and Stellar Convection Zones. *Astrophys. J., Suppl. Ser.* **121,** 247–264 (1999).

97. Lipps, F. B. On the Anelastic Approximation for Deep Convection. *J. Atmos. Sci.* **47,** 1794–1798 (1990).

98. Oberbeck, A. Ueber die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömungen infolge von Temperaturdifferenzen. *Ann. Phys.* **243,** 271–292 (1879).

99. Boussinesq, J. *Theorie Analytique de la Chaleur, Vol. 2* (Gauthier-Villars (Paris), 1903).

100. Buckingham, E. On Physically Similar Systems; Illustrations of the Use of Dimensional Equations. *Phys. Rev.* **4,** 345–376 (1914).

101. Rapp, B. E. in *Microfluidics: Modelling, Mechanics and Mathematics* 243–263 (Elsevier, 2017).

102. Verstraeten, D., Schrauwen, B., D'Haene, M. & Stroobandt, D. An experimental unification of reservoir computing methods. *Neural Netw.* **20,** 391–403 (2007).

103. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Networks* **115,** 100–123 (2019).

104. Steil, J. J. *Backpropagation-decorrelation: online recurrent learning with O(N) complexity* in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)* **2** (2004), 843–848.

105. Fujii, K. & Nakajima, K. Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning. *Phys. Rev. Applied* **8,** 024030 (2017).

106. Nakajima, K., Fujii, K., Negoro, M., Mitarai, K. & Kitagawa, M. Boosting Computational Power through Spatial Multiplexing in Quantum Reservoir Computing. *Phys. Rev. Applied* **11,** 034021 (2019).

107. Kutvonen, A., Fujii, K. & Sagawa, T. Optimizing a quantum reservoir computer for time series prediction. *Sci. Rep.* **10,** 14687 (2020).

108. Pfeffer, P., Heyder, F. & Schumacher, J. Hybrid quantum-classical reservoir computing of thermal convection flow. *Phys. Rev. Res.* **4,** 033176 (2022).

109. Wilimitis, D. *The Kernel Trick in Support Vector Classification. Towards Data Science article available at https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f [accessed 09-May-2023]* 2018.

110. Lu, Z. *et al.* Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos* **27,** 041102 (2017).

111. Morando, S., Jemei, S., Gouriveau, R., Zerhouni, N. & Hissel, D. *Fuel Cells prognostics using echo state network* in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society* (IEEE, 2013).

112. Özdemir, A. *et al.* EchoVPR: Echo state networks for visual place recognition. *IEEE Robot. Autom. Lett.* **7,** 4520–4527 (2022).

113. Ghazijahani, M. S., Heyder, F., Schumacher, J. & Cierpka, C. On the benefits and limitations of Echo State Networks for turbulent flow prediction. *Meas. Sci. Technol.* **34,** 014002 (2022).

114. Ghazijahani, M. S., Heyder, F., Schumacher, J. & Cierpka, C. Spatial prediction of the turbulent unsteady von Kármán vortex street using Echo State Networks. *Phys. Fluids* **35,** 115141 (2023).

115. Doan, N. A. K., Polifke, W. & Magri, L. Short- and long-term predictions of chaotic flows and extreme events: a physics-constrained reservoir computing approach. *Proc. Math. Phys. Eng. Sci.* **477,** 20210135 (2021).

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

117. Heyder, F. & Schumacher, J. Echo state network for two-dimensional turbulent moist Rayleigh-Bénard convection. *Phys. Rev. E* **103,** 053107 (2021).

118. Pandey, S., Teutsch, P., Mäder, P. & Schumacher, J. Direct data-driven forecast of local turbulent heat flux in Rayleigh-Bénard convection. *Phys. Fluids* **34,** 045106 (2022).

119. Valori, V., Kräuter, R. & Schumacher, J. Extreme vorticity events in turbulent Rayleigh-Bénard convection from stereoscopic measurements and reservoir computing. *Phys. Rev. Research* **4,** 023180 (2022).

120. Heyder, F., Mellado, J. P. & Schumacher, J. Generalizability of reservoir computing for flux-driven two-dimensional convection. *Phys. Rev. E* **106,** 055303 (2022).

121. Goudarzi, A., Banda, P., Lakin, M. R., Teuscher, C. & Stefanovic, D. A Comparative Study of Reservoir Computing for Temporal Signal Processing. *arXiv:1401.2224* (2014).

122. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* 43–99 (Springer New York, 2008).

123. Heyder, F. *turbESN v.0.0.1.9.4.0. Available at https://github.com/flohey/turbESN*

124. Lukoševičius, M. A Practical Guide to Applying Echo State Networks. *LNCS* **7700,** 659–686 (2012).

125. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3,** 127–149 (2009).

126. Ozturk, M. C., Xu, D. & Príncipe, J. C. Analysis and Design of Echo State Networks. *Neural Comput.* **19,** 111–138 (2007).

127. Yildiz, I. B., Jaeger, H. & Kiebel, S. J. Re-Visiting the echo state property. *Neural Netw.* **35,** 1–9 (2012).

128. Verzelli, P., Alippi, C. & Livi, L. Learn to synchronize, synchronize to learn. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31,** 083119 (2021).

129. Viehweg, J., Worthmann, K. & Mäder, P. Parameterizing echo state networks for multi-step time series prediction. *Neurocomputing* **522,** 214–228 (2023).

130. Hinaut, X. & Trouvain, N. *Which Hype for my New Task? Hints and Random Search for Reservoir Computing Hyperparameters* in *ICANN 2021 - 30th International Conference on Artificial Neural Networks* (Bratislava, Slovakia, 2021).

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

132. Takano, K. *et al.* Compact reservoir computing with a photonic integrated circuit. *Optics Express* **26,** 29424 (2018).

133. Brunner, D. *et al.* Tutorial: Photonic neural networks in delay systems. *J. of Appl. Phys.* **124,** 152004 (2018).

134. Brunner, D., Soriano, M. C. & Van der Sande, G. *Photonic Reservoir Computing* chap. Optical Recurrent Neural Networks. ISBN: 9783110583496 (De Gruyter, Berlin, Boston, 2019).

135. Hülser, T., Köster, F., Jaurigue, L. & Lüdge, K. Role of delay-times in delay-based photonic reservoir computing. *Optical Materials Express* **12,** 1214 (2022).

136. Dion, G., Mejaouri, S. & Sylvestre, J. Reservoir computing with a single delay-coupled non-linear mechanical oscillator. *Journal of Applied Physics* **124,** 152132 (2018).

137. Appeltant, L. *et al.* Information processing using a single dynamical node as complex system. *Nat. Commun.* **2,** 468 (2011).

138. Larger, L. *et al.* High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification. *Phys. Rev. X* **7,** 011015 (2017).

139. Furuta, T. *et al.* Macromagnetic Simulation for Reservoir Computing Utilizing Spin Dynamics in Magnetic Tunnel Junctions. *Phys. Rev. Appl.* **10,** 034063 (2018).

140. Tsunegi, S. *et al.* Physical reservoir computing based on spin torque oscillator with forced synchronization. *Appl. Phys. Lett.* **114,** 164101 (2019).

141. Taniguchi, T., Ogihara, A., Utsumi, Y. & Tsunegi, S. Spintronic reservoir computing without driving current or magnetic field. *Scientific Reports* **12,** 10627 (2022).

142. Soriano, M. C. *et al.* Delay-based reservoir computing: Noise effects in a combined analog and digital implementation. *IEEE Trans. Neural Netw. Learn. Syst.* **26,** 388–393 (2015).

143. Carroll, T. L. Do reservoir computers work best at the edge of chaos? *Chaos* **30** (2020).

144. Biamonte, J. *et al.* Quantum machine learning. *Nature* **549,** 195–202 (2017).

145. Deutsch, I. H. Harnessing the Power of the Second Quantum Revolution. *PRX Quantum* **1,** 020101 (2 2020).

146. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21,** 467–488 (1982).

147. *IBM Quantum https://quantum-computing.ibm.com/* 2021.

148. Blatt, R. *The Rochester Conferences on Coherence and Quantum Optics and the Quantum Information and Measurement meeting* Th1.1 (Optica Publishing Group, 2013).

149. Bruzewicz, C. D., Chiaverini, J., McConnell, R. & Sage, J. M. Trapped-ion quantum computing: Progress and challenges. *Appl. Phys. Rev.* **6,** 021314 (2019).

150. Kjaergaard, M. *et al.* Superconducting Qubits: Current State of Play. *Annu. Rev. Condens. Matter Phys.* **11,** 369–395 (2020).

151. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2,** 79 (2018).

152. Flöther, F. F. The state of quantum computing applications in health and medicine. *arXiv:2301.09106* (2023).

153. Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **26,** 1484–1509 (1997).

154. Grover, L. K. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Phys. Rev. Lett.* **79,** 325–328 (1997).

155. Araiza Bravo, R., Khadijeh, N., Gao, X. & Yelin, S. F. Quantum reservoir computing using arrays of Rydberg atoms. *PRX Quantum* **3,** 030325 (2022).

156. Dasgupta, S., Hamilton, K. E. & Banerjee, A. Characterizing the memory capacity of transmon qubit reservoirs. *arXiv:2004.08240* (2022).

157. Suzuki, Y., Gao, Q., Pradel, K. C., Yasuoka, K. & Yamamoto, N. Natural quantum reservoir computing for temporal information processing. *Sci. Rep.* **12** (2022).

158. Pfeffer, P., Heyder, F. & Schumacher, J. Reduced-order modeling of two-dimensional turbulent Rayleigh-Bénard flow by hybrid quantum-classical reservoir computing. *Phys. Rev. Res.* **5** (2023).

159. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* 649–699 (Springer New York, 2008).

160. Taylor, G. I. Statistical theory of turbulenc. *Proc. Math. Phys. Eng. Sci.* **151,** 421–444 (1935).

161. Cantwell, B. J. Organized Motion in Turbulent Flow. *Annu. Rev. Fluid Mech.* **13,** 457–515 (1981).

162. Lumley, J. L. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation* (1967).

163. Pearson, K. LIII. iOn lines and planes of closest fit to systems of points in space/i. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **2,** 559–572 (1901).

164. Deisenroth, M. P., Faisal, A. A. & Ong, C. S. *Mathematics for Machine Learning* ISBN: 978-1-108-45514-5 (Cambridge University Press, 2020).

165. Holmes, P., Lumley, J. L., Berkooz, G. & Rowley, C. W. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry* 2nd ed. (Cambridge University Press, 2012).

166. Bailon-Cuba, J., Emra, M. S. & Schumacher, J. Aspect ratio dependence of heat transfer and large-scale flow in turbulent convection. *J. Fluid. Mech.* **655,** 152–173 (2010).

167. Bailon-Cuba, J. & Schumacher, J. Low-dimensional model of turbulent Rayleigh-Bénard convection in a Cartesian cell with square domain. *Phys. Fluids* **23,** 077101 (2011).

168. Weidauer, T. & Schumacher, J. Toward a Mode Reduction Strategy in Shallow Moist Convection. *New J. Phys.* **15,** 125025 (2013).

169. Aubry, N., Holmes, P., Lumley, J. L. & Stone, E. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *J. Fluid Mech.* **192,** 115–173 (1988).

170. Barbagallo, A., Sipp, D. & Schmid, P. J. Closed-loop control of an open cavity flow using reduced-order models. *J. Fluid Mech.* **641,** 1–50 (2009).

171. Ahuja, S. & Rowley, C. W. Feedback control of unstable steady states of flow past a flat plate using reduced-order estimators. *J. Fluid. Mech.* **645,** 447–478 (2010).

172. Brunton, S. L. & Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control* ISBN: 978-1-108-42209-3 (Cambridge University Press, Cambridge, 2019).

173. Volkwein, S. Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. *Lecture Notes, University of Konstanz* (2012).

174. Sirovich, L. Turbulence and the dynamics of coherent structures. Part I: Coherent structures. *Q. Appl. Math.* **XLV,** 561 (1987).

175. Park, H. & Sirovich, L. Turbulent thermal convection in a finite domain: Part II. Numerical results. *Phys. Fluids A: Fluid Dynamics* **2,** 1659–1668 (1990).

176. Maulik, R., Lusch, B. & Balaprakash, P. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Phys. Fluids* **33,** 037106 (2021).

177. Mohan, A., Daniel, D., Chertkov, M. & Livescu, D. Compressed Convolutional LSTM: An Efficient Deep Learning framework to Model High Fidelity 3D Turbulence (2019).

178. Behrens, G. *et al.* Non-Linear Dimensionality Reduction With a Variational Encoder Decoder to Understand Convective Processes in Climate Models. *J. Adv. Mod. Earth Sys.* **14,** e2022MS003130 (2022).

179. Mooers, G., Tuyls, J., Mandt, S., Pritchard, M. & Beucler, T. G. *Generative Modeling of Atmospheric Convection* in *Proceedings of the 10th International Conference on Climate Informatics* (ACM, 2020).

180. *NEK5000 version 17.0, Argonne National Laboratory, Illinois. Dec. 18 2017. Available at https://nek5000.mcs.anl.gov (last visit on 22.05.2023)*

181. Deville, M. O., Fischer, P. F. & Mund, E. H. *High-Order Methods for Incompressible Fluid Flow* 499 pp. ISBN: 978-0-521-45309-7 (Cambridge University Press, Cambridge, UK ; New York, 2002).

182. Scheel, J. D., Emran, M. S. & Schumacher, J. Resolving the fine-scale structure in turbulent Rayleigh–Bénard convection. *New J. Phys.* **15,** 113063 (2013).

183. Mellado, J. P., Ansorge, C., Kostelecky, J. & Fodor, K. *Tlab. Available at https://github.com/turbulencia/tlab*

184. Srinivasan, P. A., Guastoni, L., Azizpour, H., Schlatter, P. & Vinuesa, R. Predictions of turbulent shear flows using deep neural networks. *Phys. Rev. Fluids* **4,** 054603 (2019).

185. Lorenz, E. N. Low-order Models of Atmospheric Circulations. *J. Meteorol. Soc. Japan* **60,** 255–267 (1982).

186. Platzman, G. W. The spectral form of the vorticity equation. *J. Atmos. Sci.* **17,** 635–644 (1960).

187. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20,** 130–141 (1963).

188. Saltzman, B. Finite Amplitude Free Convection as an Initial Value Problem - I. *J. Atmos. Sci.* **19,** 329–341 (1962).

189. Howard, L. N. & Krishnamurti, R. Large-scale flow in turbulent convection: A mathematical model. *J. Fluid. Mech.* **170,** 385–410 (1986).

190. Grossmann, S. & Lohse, D. Scaling in hard turbulent Rayleigh-Bénard flow. *Phys. Rev. A* **46,** 903–917 (1992).

191. Gluhovsky, A. & Tong, C. The structure of energy conserving low-order models. *Phys. Fluids* **11,** 334–343 (1999).

192. Gluhovsky, A., Tong, C. & Agee, E. Selection of modes in convective low-order models. *J. Atmos. Sci.* **59,** 1383–1393 (2002).

193. Lorenz, E. N. *Predictability: A problem partly solved* in *Proc. Seminar on predictability* **1** (1996).

194. Eggers, J. & Grossmann, S. Does deterministic chaos imply intermittency in fully developed turbulence? *Phys. Fluids* **3,** 1958–1968 (1991).

195. Grossmann, S. & Lohse, D. Intermittency in the Navier-Stokes dynamics. *Z. Phys. B* **89,** 11–19 (1992).

196. Moehlis, J., Faisst, H. & Eckhardt, B. A low-dimensional model for turbulent shear flows. *New J. Phys.* **6,** 56–56 (2004).

197. Haluszczynski, A. & Räth, C. Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing. *Chaos* **29,** 103143 (2019).

198. Inubushi, M. & Goto, S. Transfer learning for nonlinear dynamics and its application to fluid turbulence. *Phys. Rev. E* **102** (2020).

199. Kim, J. Z., Lu, Z., Nozari, E., Pappas, G. J. & Bassett, D. S. Teaching recurrent neural networks to infer global temporal structure from local examples. *Nat. Mach. Intell.* **3,** 316–323 (2021).

200. Chandrasekhar, S. *Hydrodynamic and hydromagnetic stability* ISBN: 978-0486640716 (Dover Publications Inc., 1981).

201. Hermiz, K. B., Guzdar, P. N. & Finn, J. M. Improved low-order model for shear flow driven by Rayleigh-Bénard convection. *Phys. Rev. E* **51,** 325–331 (1995).

202. Sprott, J. C. *Chaos and time-series analysis* (Oxford University Press Oxford, 2003).

203. Geurts, B. J., Holm, D. D. & Luesink, E. Lyapunov exponents in two stochastic Lorenz 63 systems. *J. Stat. Phys.* **179,** 1343–1365 (2020).

204. Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing. Available at www.doi.org/10.5281/zenodo.2573505* (2023).

205. *Conversation with Philipp Pfeffer at TU Ilmenau on 19.08.2023.* (2023).

206. *Conversation with Philipp Pfeffer at TU Ilmenau on 22.08.2023.* (2023).

207. Mujal, P. *et al.* Opportunities in Quantum Reservoir Computing and Extreme Learning Machines. *Adv. Quantum Technol.* **4,** 2100027 (8 2021).

208. Atkinson, B. & Wu Zhang, J. Mesoscale shallow convection in the atmosphere. *Rev. Geophys.* **34,** 403–431 (1996).

209. IPCC. *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change* (Cambridge University Press, Cambridge, UK and New York,NY,USA, 2013).

210. Pauluis, O. & Schumacher, J. Idealized Moist Rayleigh-Benard Convection with Piecewise Linear Equation of State. *Commun Math Sci.* **8,** 295–319 (2010).

211. Weidauer, T., Pauluis, O. & Schumacher, J. Cloud Patterns and Mixing Properties in Shallow Moist Rayleigh–Bénard Convection. *New J. Phys.* **12,** 105002 (2010).

212. Schumacher, J. & Pauluis, O. Buoyancy Statistics in Moist Turbulent Rayleigh–Bénard Convection. *J. Fluid Mech.* **648,** 509–519 (2010).

213. Bretherton, C. S. A Theory for Nonprecipitating Moist Convection between Two Parallel Plates. Part I: Thermodynamics and "Linear" Solutions. *J. Atmos. Sci.* **44,** 1809–1827 (1987).

214. Bretherton, C. S. A Theory for Nonprecipitating Moist Convection between Two Parallel Plates. Part II: Nonlinear Theory And Cloud Field Organization. *J. Atmos. Sci.* **45,** 2391–2415 (1988).

215. Smith, L. M. & Stechmann, S. N. Precipitating Quasigeostrophic Equations and Potential Vorticity Inversion with Phase Changes. *J. Atmos. Sci.* **74,** 3285–3303 (2017).

216. Vallis, G. K., Parker, D. J. & Tobias, S. M. A Simple System for Moist Convection: the Rainy–Bénard Model. *J. Fluid Mech.* **862,** 162–199 (2019).

217. Abma, D., Heus, T. & Mellado, J. P. Direct Numerical Simulation of Evaporative Cooling at the Lateral Boundary of Shallow Cumulus Clouds. *J. Atmos. Sci.* **70,** 2088–2102 (2013).

218. Pauluis, O. & Schumacher, J. Self-Aggregation of Clouds in Conditionally Unstable Moist Convection. *Proc. Natl. Acad. Sci. U.S.A.* **108,** 12623–12628 (2011).

219. Heyder, F., Mellado, J. P. & Schumacher, J. Generative convective parametrization of dry atmospheric boundary layer. *arXiv:2307.14857* (2023).

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

221. Mellado, J. P. & Ansorge, C. Factorization of the Fourier transform of the pressure-Poisson equation using finite differences in colocated grids. *Z. Angew. Math. Mech.* **92,** 380–392 (2012).

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

223. Carson, D. J. & Smith, F. B. *Thermodynamic model for the development of a convectively unstable boundary layer* in *Advances in Geophysics* **18** (Elsevier, 1975), 111–124.

224. Garcia, J. R. & Mellado, J. P. The two-Layer structure of the entrainment zone in the convective boundary layer. *J. Atmos. Sci.* **71,** 1935–1955 (2014).

225. Goodfellow, I. J. *et al.* Generative Adversarial Networks. *arXiv:1406.2661* (2014).

226. Nash, J. F. Equilibrium points in n-person games. *Proc. Natl. Acad. Sci. USA* **36,** 48–49 (1950).

227. Arjovsky, M., Chintala, S. & Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862* (2017).

228. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein GAN. *arXiv:1701.07875* (2017).

229. Villani, C. *Optimal Transport: Old and New* ISBN: 9783540710509 (Springer, Berlin, 2008).

230. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. *Improved training of Wasserstein GANs* in *Advances in Neural Information Processing Systems* (eds Guyon, I. *et al.*) **30** (Curran Associates, Inc., 2017).

231. Suzuki, S. & be, K. Topological structural analysis of digitized binary images by border following. *Comput. Graph. Image Process.* **30,** 32–46 (1985).

232. Itseez. *Open Source Computer Vision Library* https://github.com/itseez/opencv. 2015.

233. Deardorff, J. W. Convective velocity and temperature scales for the unstable planetary boundary layer and for Rayleigh convection. *J. Atmos. Sci.* **27,** 1211–1213 (1970).

234. Soares, Miranda, P. M. A., Siebesma, A. P. & Teixeira, J. An eddy-diffusivity/mass-flux parameterization for dry and shallow cumulus convection. *Q. J. R. Soc. Meteorol.* **130,** 3365–3384 (2004).

235. Kawai, Y., Tokuno, T., Park, J. & Asada, M. *Echo in a small-world reservoir: Time-series prediction using an economical recurrent neural network* in *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* (2017), 126–131.

236. Kawai, Y., Park, J. & Asada, M. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks* **112,** 15–23 (2019).

237. Ortali, G., Corbetta, A., Rozza, G. & Toschi, F. Numerical proof of shell model turbulence closure. *Phys. Rev. Fluids* **7,** L082401 (2022).

238. Brenowitz, N. D. *et al.* Machine Learning Climate Model Dynamics: Offline versus Online Performance. *arXiv:2011.03081* (2020).

239. Beucler, T., Ebert-Uphoff, I., Rasp, S., Pritchard, M. & Gentine, P. Machine Learning for Clouds and Climate (Invited Chapter for the AGU Geophysical Monograph Series "Clouds and Climate"). *Earth and Space Science Open Archive* (2021).

240. Rasp, S. Combining crowdsourcing and deep learning to explore the mesoscale organization of shallow convection. *Bull. Am. Meteorol. Soc.* **101,** E1980–E1995 (2020).

241. Watson-Parris, D. *et al.* Constraining Uncertainty in Aerosol Direct Forcing. *Geophys. Res. Lett.* **47,** e2020GL087141 (2020).

242. Toms, B. A., Kashinath, K., Prabhat & Yang, D. Testing the reliability of interpretable neural networks in geoscience using the Madden–Julian oscillation. *Geosci. Model. Dev.* **14,** 4495–4508 (2021).

243. Wolpert, D. & Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1,** 67–82 (1997).

244. Heyder, F. *Numerical implementations to the doctoral thesis Reduced Order Modeling of Thermal Convection Flows: A Reservoir Computing Approach. Available at www.doi.org/10.5281/zenodo.8307591*

245. Clevert, D.-A. & Unterthiner Thomasand Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289* (2016).

246. Siebesma, P. *On the mass flux approach for atmospheric convection* in *Proc. Workshop on New Insights and Approaches to Convective Parametrization* (1996), 25–57.

247. Witte, A. *et al.* Augmenting the double-Gaussian representation of atmospheric turbulence and convection via a coupled stochastic multi-plume mass-flux scheme. *Mon. Weath. Rev.* **150,** 2339–2355 (2022).

248. Siebesma, A. P. & Cuijpers, J. W. M. Evaluation of parametric assumptions for shallow cumulus convection. *J. Atmos. Sci.* **52,** 650–666 (1995).

249. Schumann, U. & Moeng, C.-H. Plume budgets in clear and cloudy convective boundary layers. *J. Atmos. Sci.* **48,** 1758–1770 (1991).

250. Holtslag, A. & Duynkerke, P. G. *Clear and Cloudy Boundary Layers: Proceedings of the Colloquium" Clear and Cloudy Boundary Layers," Amsterdam, 26-29 August 1997* (Koninklijke Nederlandse akademie van wetenschappen, Amsterdam, 1998).

251. Holtslag, A. A. M. & Moeng, C.-H. Eddy Diffusivity and Countergradient Transport in the Convective Atmospheric Boundary Layer. *J. Atmos. Sci.* **48,** 1690–1698 (1991).

252. van Ulden, A. P. & Siebesma, A. P. A model for strong updrafts in the convective boundary layer. *12th Symp. on Boundary Layers and Turbulence, Vancouver, BC, Canada, Amer. Meteor. Soc.,* 257–259 (1997).

253. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computational Science* **9351,** 234–241 (2015).

254. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *arXiv:1502.01852* (2015).

255. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* 8024–8035 (Curran Associates, Inc., 2019).

256. Heyder, F., Pandey, S. & Schumacher, J. *Reservoir Computing of Dry and Moist Turbulent Convection* 73th Annual Meeting of the APS Division of Fluid Dynamics. 2020.

257. Heyder, F., Mellado, J. P. & Schumacher, J. *Two-Dimensional Convective Boundary Layer: Numerical Analysis and Echo State Network model* 74th Annual Meeting of the APS Division of Fluid Dynamics. 2021.

258. Heyder, F. & Schumacher, J. *Reservoir computing of three-dimensional turbulent convection* 8th European Congress on Computational Methods in Applied Sciences and Engineering. 2022.

259. Heyder, F., Mellado, J. P. & Schumacher, J. *Reservoir computing of a three-dimensional turbulent atmospheric boundary layer* 18th European Turbulence Conference. 2023.

260. Vieweg, P., Heyder, F., John, J. P. & Schumacher, J. *Analysis of the Large-Scale Order in Turbulent Mesoscale Convection* in *NIC Symposium 2022 Proceedings* (eds Müller, M., Peter, C. & Trautmann, A.) (Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Jülich, 2022), 405–415.