Symmetry Constraints Enhance Long-term Stability and Accuracy in Unsupervised Learning of Geophysical Fluid Flows

Yunfei Huang¹ and David S. Greenberg²

¹Institute of Coastal Systems, Helmholtz-Zentrum Hereon ²Helmholtz-Zentrum Hereon, Geesthacht, Germany

March 07, 2024

Abstract

Fluid dynamical systems are well described by discretized partial differential equations, but computational costs limit accuracy, duration and/or resolution in numerical integrations. Recent studies showed that deep neural networks trained on simulations or PDE-derived losses can improve cost-accuracy tradeoffs, but purely data-centric approaches discard physical and mathematical insights and require computationally costly training data. Here we draw on advances in geometric deep learning to design solver networks that respect PDE symmetries as hard constraints. We construct equivariant convolutional layers for mixed scalar-vector input fields in order to capture the symmetries inherent to specific PDEs. We demonstrate our approach on a challenging 1D semi-implicit shallow water scheme with closed boundaries, applying unsupervised learning with a physics-derived loss function. We report strong improvements in accuracy and stability of equivariant solvers compared to standard convolutional networks with the same architectures and parameter counts. Solver equivariance also improves performance on new initial conditions not encountered during training, and suppresses error accumulation in global momentum and energy. Strikingly, these benefits do not reduce loss values during training, but appear later during ML-assisted rollouts over time steps. Our results suggest that symmetry constraints could improve deep learning performance across a wide range of fluid dynamical tasks, learning algorithms and neural architectures.

Symmetry Constraints Enhance Long-term Stability and Accuracy in Unsupervised Learning of Geophysical Fluid Flows

 $_{3}$ Yunfei Huang^{1, 2} and David S. Greenberg^{1, 2}

¹⁾Institute of Coastal Systems, Helmholtz-Zentrum Hereon, Geesthacht, Germany

⁵ ²⁾Helmholtz AI

4

- (*Electronic mail: david.greenberg@hereon.de)
- 7 (*Electronic mail: Yunfei.Huang@hereon.de)
- 8 (Dated: 23 January 2024)

Fluid dynamical systems are well described by discretized partial differential equations (PDEs), but computational 9 costs limit accuracy, duration and/or resolution in numerical integrations. Recent studies showed that deep neural 10 networks trained on simulations or PDE-derived losses can improve cost-accuracy tradeoffs, but purely data-centric 11 approaches discard physical and mathematical insights and require computationally costly training data. Here we draw 12 on advances in geometric deep learning to design solver networks that respect PDE symmetries as hard constraints. 13 We construct equivariant convolutional layers for mixed scalar-vector input fields in order to capture the symmetries 14 inherent to specific PDEs. We demonstrate our approach on a challenging one-dimensional semi-implicit shallow water 15 scheme with closed boundaries, applying unsupervised learning with a physics-derived loss function. We report strong 16 improvements in accuracy and stability of equivariant solvers compared to standard convolutional networks with the 17 same architectures and parameter counts. Solver equivariance also improves performance on new initial conditions 18 not encountered during training, and suppresses error accumulation in global momentum and energy. Strikingly, these 19 benefits do not reduce loss values during training, but appear later during machine learning (ML)-assisted rollouts over 20 time steps. Our results suggest that symmetry constraints could improve deep learning performance across a wide range 21 of fluid dynamical tasks, learning algorithms and neural architectures. 22

54

55

23 I. INTRODUCTION

Partial differential equations (PDEs) are essential for understanding and simulating complex fluid dynamics. Examples include convection-diffusion¹, Euler² and Navier–Stokes equations (NS)^{3,4}. The shallow water equations (SWEs)⁵, derived by depth integration of NS, are mathematically simpler but widely employed as test cases to evaluate solution techniques for ocean, weather and climate applications^{6–11}.

PDEs describing geophysical fluid flows require numer- 63 31 ical methods, for example, finite difference^{12,13}, finite ele-64 32 ment^{14,15}, finite volume¹⁶, boundary element¹⁷, and spectral ⁶⁵ 33 element methods¹⁸. While small spatial domains admit di-⁶⁶ 34 rect numerical simulation, geophysical applications require 67 35 coarse grids with Reynolds-averaging¹⁹ or large eddy simu- 68 36 lation (LES)^{20,21} to approximate unresolved scales. Explicit ⁶⁹ 37 time stepping simplifies computations but requires small steps 70 38 for stability, while (semi)implicit schemes take larger time 71 39 steps but must iteratively solve a system of equations ^{22–25}.⁷² 40 However, these classical approaches incur heavy computa-73 41 tional costs at high spatial and temporal resolutions. 42

Recent machine learning approaches aim to transcend these ⁷⁵ 43 cost-accuracy tradeoffs by training a model to accurately 76 44 and efficiently solve PDEs on modern computational hard-77 45 ware. Supervised learning uses simulations from a classi-78 46 cal solver to train a machine learning (ML) model that uses 79 47 larger space and time steps or skips the iterative computa-80 48 tions of an implicit scheme. This approach has shown success ⁸¹ 49 in accelerating PDE solutions while maintaining accuracy, 82 50 obeying conservation laws and preserving high frequency fea- 83 51 tures²⁶⁻³², and has also been applied to mesh-free particle- 84 52 based solvers³³. 53 85

Unsupervised learning trains the model to satisfy the PDE without requiring training data. It is most effective for implicit schemes, since solving their equations iteratively is complex and costly but verifying a solution is simple and fast. Unsupervised learning avoids overfitting by training on its own outputs, but cannot avoid discretization errors for large the space or time steps. It has been used to solve several fluid dynamical PDEs^{34–43}.

Hybrid models replace only part of a classical PDE solver with an ML model, leaving other components unchanged. Early work applied this to computer graphics⁴⁴, while a later study demonstrated an approach combining a fluid solver with ML techniques to approximate NS in a Lagrangian framework using regression forests⁴⁵. More recently, authors⁴⁶ used supervised learning to compute an additive correction to low-resolution incompressible NS, so that its evolution mimics a high-resolution model coarsened at each time step. An LSTM-based hybrid approach⁴⁷ with significant practical speed-ups has been presented for predicting pressure changes for incompressible flow, while ref.⁴⁸ proposed an ML-based approach for replacing the linear projection in the Eulerian fluid implicit simulation, authors²⁷ combined two wellestablished turbulent flow simulation techniques with deep learning and the paper⁴⁹ developed an accelerated integrative ML solverto aid convergence of Reynolds Averaged Navier-Stokes simulations. Overall, hybrid methods allow us to effectively incorporate physical knowledge while simplifying the learning task, and can improve accuracy and generalization capabilities.

Major challenges remain for ML-based PDE solvers: longterm stability and accuracy are not guaranteed even for low loss values on training and testing data^{46,50–52}, and general-



FIG. 1. Schematic representation of a one-dimensional shallow water problem with C-grid staggering of discretized fields. (a) Shallow water system with domain length *L*. *d* and *h* are un-disturbed- and disturbed- water depth, ζ is fluid surface elevation and $h = \zeta + d$. *u* is the velocity along the space coordinate *x*. (b) Staggered grid for elevation and velocity. Top: fluid surface elevation ζ_j is represented at red squares while velocity $u_{j+1/2}$ is represented at black circles. Bottom: A flipped fluid surface elevation ζ_i , as well as a flipped-and-inverted velocity. These transformed fields solve the SWE with transformed initial and boundary conditions.

ization to new scenarios remains problematic^{53–56}. A partial₁₂₂ 86 explanation is that training neural networks means choosing₁₂₃ 87 from a large, high-dimensional family of functions, many of₁₂₄ 88 which are physically or mathematically implausible. Nar-125 89 rowing the search by constraining the learned function has126 90 shown great promise: for example, conservation laws im-127 91 prove process representations in climate, weather, and ocean128 92 models^{57,58}, while symmetry constraints aid image classifica-129 93 tion⁵⁹ and segmentation^{60,61}. However, the potential benefit₁₃₀ 94

for fluid dynamics remains mostly unclear. 95 In this work we construct hybrid PDE solvers using equiv-132 96 ariant neural networks that obey PDE symmetry constraints. 97 We draw on previous work in geometric deep learning^{59,62}, 98 but extend group equivariant convolutions to handle mixed133 99 scalar/vector inputs with the correct, PDE-specific transfor-100 mation rules. We demonstrate the benefit of equivariant solver 101 networks using an unsupervised learning task, in which the 102 network is trained to integrate a semi-implicit scheme for one,136 103 dimensional (1-D) shallow water equations. These equations₁₃₇ 104 exhibit challenging stiff dynamics due to closed boundaries 105 and reflecting waves. Our experiments show significant im-106 provements in long term accuracy and stability compared to 107 standard convolutional neural networks (CNNs), despite simi-108 lar loss values during training. We also observe that symmetry 109 constraints improve performance on initial conditions not en-110 countered during training, as well as representations of global 111 mass, momentum and energy. 142 112

113 II. NUMERICAL INTEGRATION OF FLUID DYNAMICS

In this section we establish concepts and notation for PDE¹⁴⁷ integration with classical numerical techniques, allowing us to¹⁴⁸ describe our task and approach in the following section. We¹⁴⁹ consider a general governing partial differential equation for¹⁵⁰ fluid dynamics: 151

¹¹⁹
$$\frac{\partial q(t,x)}{\partial t} = \mathcal{F}[q] = f(t,x,q(t,x),\frac{dq}{dx},\frac{d^2q}{dx^2},\ldots), \quad x \in \Omega \quad (1)_{15}$$

$$q(x,t) = q_{\Omega}(x), \quad x \in \partial \Omega$$
(2)

$$q(x,0) = q^0(x)$$
 (3)₁₉

x and *t* are space and time coordinates and q(t,x) is the vector of modeled variable fields at one place and time, such as velocity and pressure in NS or velocity and height in SWE. \mathcal{F} is a nonlinear operator computing time derivatives as nonlinear functions of functions of the fields and their spatial derivatives. The Dirichlet boundary conditions (BCs) $q_{\Omega}(x)$ on the boundary $\partial \Omega$ and initial conditions (ICs) $q^0(x)$ are given while q(x,t) is the unknown quantity for which we solve the PDE. Eqs. 1-3 are a common form for governing a fluid flow, though other types of BCs and constraints (such as incompressibility) can also be used.

A. Spatial Discretization

We solve our PDEs with the classical finite difference methods with uniform time step Δt and all prognostic variables defined on a regular grid with space step Δx . For 1-D fields, and denoting the *k*-th variable field in *q* by *z*, we use $z_j^n = q(j\Delta x, n\Delta t)_k$ to denote the value of *z* at the *j*-th location on the *n*-th time step. We employ staggered representations of scalar fields and velocities using Arakawa C-grids⁶³, and the notation $z_{1/2}^n, z_{3/2}^n, \ldots$ for shifted variables in Fig. 1b.

B. Time Stepping

143

144

145

146

152

Given the discretized variable fields q^n at time $t = n\Delta t$, a time stepping scheme is used to compute the next fields q^{n+1} . Here we consider the broad range of schemes in which each occurence of q in the definition of \mathcal{F} is replaced by a weighted average of q^n and q^{n+1} , and the weighting may be different for each field and each term of the PDE. Thus when q^n is used in every case we have an explicit method, while using $(q^n + q^{n+1})/2$ in every case gives a Crank-Nicholson method⁶⁴. Denoting the discretized version of \mathcal{F} by $\tilde{\mathcal{F}}$, the scheme can be written as a system of equations

$$q_j^{n+1} = q_j^n + \Delta t \widetilde{\mathcal{F}}[q^n, q^{n+1}]$$
(4)

Because q^{n+1} appears on both sides of the equation, we must solve the equations to obtain it, for example by using iterative



FIG. 2. Schematic representation of equivariant U-net architecture and 'hybrid' training procedure for a deep SWE solver. (a) Equivariant U-Net architecture with an example of channels. Grid staggering leads to input channels with different sizes, so we use equivariant input layers with different kernel sizes to obtain a uniform size across channels in the first hidden layer. All activations except for inputs and outputs are defined as real-valued functions on the infinite discrete group H containing reflections and translations (light blue area). Network outputs provide a fluid surface update $\Delta \zeta$. (b) A 'hybrid' training loop adapted from³⁴. A pool of system states is first filled with the initial conditions ζ^0, u^0 . For a randomly selected batch of system states, the U-net then generates ζ^{t+dt} at the next time step while velocity is calculated as in a numerical solver. The unsupervised loss function summed over the batch, its gradients are used to update network parameters and the new states overwrite their own inputs in the pool.

202

methods. Calculating $\widetilde{\mathcal{F}}$ for the discretized variable fields re-185 156 quires discretized versions of all spatial derivatives, which in186 157 general must be designed and tested for each PDE to ensure187 158 accuracy and stability. This class of time stepping schemes188 159 is widely used for fluid dynamical PDEs: examples include 160 incompressible Navier Stokes⁶⁵⁻⁶⁷ and certain shallow water 161 solvers with land-water boundaries⁶⁸ (see below). 189 162

UNSUPERVISED LEARNING OF PDE INTEGRATION Ш. 163

Problem Statement 164 Α.

We aim to replace an expensive semi-implicit time scheme¹⁹³ 165 (Eq. 4) with a faster, neural-network based solver. Critically, 166 we do not assume that we have access to simulation data for 167 training purposes, but must train the network using only our¹⁹⁴ 168 knowledge of the PDE, spatial discretization and time step-195 169 ping scheme. 170

Concretely, we wish to train (that is, optimize) the pa-197 171 rameters ϕ of a flexible function approximator \hat{S}_{ϕ} , such that¹⁹⁸ 172 $\widehat{S}_{\phi}(q^n) \approx S(q^n) = q^{n+1}$. Here S denotes a single step of time 173 integration using a classical numerical solver that acts as our 174 target reference solution. We aim to achieve a close approx-175 imation between \hat{S}_{ϕ} and S on PDE integrations with initial 176 distributions drawn from a specified probability distribution²⁰⁰ 177 201 $\Pi(q)$: 178

 $q^0 \sim \Pi(q)$ 179 $(5)_{203}$

$$a^n = S^{(n)}(a^0) \tag{6}^{204}$$

(6) 205 $q^{n} = S^{(n)}(q^{0})$ $\widehat{q}^{n} = \widehat{S}_{\phi}^{(n)}(q^{0})$ $\widehat{q}^{n} \approx q^{n}$ $(7)_{206}$ 181

 $(8)^{207}$ 182

Without access to simulation data, we cannot carry out super-209 183 vised training of \widehat{S}_{ϕ} using input-output pairs (q^n, q^{n+1}) . The₂₁₀ 184

motivation behind this problem formulation without access to training data is that it avoids expensive simulations, and does not require us to commit to a fixed set of simulated system states at the onset of training.

Β. **Physics-derived Loss Function**

To train $g_{\phi} \approx S$ without simulation data, we construct a physics-derived loss ^{34,41} that is zero if and only if the discretized PDE is precisely solved:

$$\mathcal{L}_{\text{PDE}}(q,\phi) = \left\| q + \Delta t \widetilde{\mathcal{F}}[q, g_{\phi}(q)] - \widehat{S}_{\phi}(q) \right\|_{2}^{2} + \mathbb{1}_{\Omega} \cdot \left\| \widehat{S}_{\phi}(q) - q_{\Omega} \right\|_{2}^{2}$$
(9)

The first loss term measures deviations from the prescribed PDE, and is clearly zero when $\hat{S}_{\phi} = S$. The second term measures violation of the Dirichlet BCs, with $\mathbb{1}_{\Omega}$ an indicator function for the boundary. If S_{ϕ} is constructed to satisfy the BCs for any ϕ the second term can be dropped.

С. **Training Algorithm**

In principle, we could minimize \mathcal{L}_{PDE} using any input fields q, but to obtain optimal results when ICs are drawn from $\Pi(q)$, we should train on fields likely to occur when timeintegrating from those ICs. We therefore train \widehat{S}_{ϕ} on fields it has itself integrated, following the strategy³⁴ in Fig. 2b.

We first initialize a pool of 5000 simulations with ICs drawn randomly from $\Pi(q)$. For each gradient step, a batch of simulations is randomly selected from the pool, and stepped forward using S_{ϕ} . The fields at the old and new time steps for this batch are used to compute \mathcal{L}_{PDE} , and the resulting gradients are used to update ϕ . The updated simulations are then



FIG. 3. Reflection-equivariant 1-D convolutional neural network (GE-CNNs). For illustrative purposes, a network with only one channel in each layers is shown. Blue arrows indicate a pair of fields that are reflections of each other, while black arrows indicate equivariant convolution layers. The 1-D input and its flipped duplicate are shown on the left side. Activations in each layer are computed by applying both a standard and flipped version of the convolutional filter to the previous layer. The final output is obtained by averaging over reflected and non-reflected version of each channel pooling on the geometrical two features.

247

248

249

250

251

stored in the simulation pool, where they overwrite their own₂₄₂

²¹² previous values. After every 50 gradient steps, a randomly se-

²¹³ lected simulation is overwritten with a new initialization from₂₄₃

 $\Pi(q)$. We use a default batch size of b = 100, 60 epochs, and

the Adam optimizer⁶⁹ with an initial learning rate of 0.001. $\frac{244}{245}$

216 D. Hybrid Solvers

For a fully data-driven architecture, the time stepping func-253 217 tion S_{ϕ} can be fully specified by a deep neural network or²⁵⁴ 218 other general function approximator. However, several stud-255 219 ies have shown that combining deep learning and numerical²⁵⁶ 220 physics in a single model can provide better results than ei^{257} ther approach alone^{35,48,49,70,71}. A particular focus of this hy-²⁵⁸ 221 222 brid approach has been semi-implicit numerical schemes that²⁵⁶ 223 require a system of equations to be solved at each time step. 224 Classical numerical solvers for these schemes often use vari-225 able substitution to reduce the number of equations and un-260 226 knowns: examples include elimination of velocity when solv-227 ing for fluid height in the SWEs⁶⁸ and the pressure projection₂₆₁ 228 step for incompressible NS. Hybrid approaches therefore train₂₆₂ 229 a deep neural network to efficiently solve the reduced set of₂₆₃ 230 equations, after which the remaining output variables are cal-264 231 culated using formulae from the original numerical solver. We₂₆₅ 232 describe this approach in detail for a SWE scheme in Fig. 2b. 266 233 This hybrid approach can offer several benefits compared²⁶⁷ 234 to a learning a fully data-driven time stepping scheme. $By^{^{268}}$ 235 replacing only expensive computations, it retains some induc-269 236

tive biases of the original scheme, and ensures that the full set
of updated fields are accurate when the learned computations
are correct. Fewer input and output channels for the trained²⁷⁰

²³⁹ are correct. Pewer input and output channels for the trained ²⁴⁰ model also reduce parameter counts and improve optimiza-

tion and data efficiencies.

E. Neural Architectures

To learn (hybrid) time stepping for spatially structured fields, we employ the U-net architecture⁷². The U-net is a convolutional encoder-decoder network. In the encoder, the number of channels increases with depth while spatial resolution decreases, while the decoder enacts the opposite transformations while receiving skip connections from the encoder at each resolution (Fig. 2a). At each resolution, the encoder and decoder employ two convolution layers with kernel size 7. For input fields with C-grid staggering, different kernel sizes are used to achieve a uniform spatial extent for output fields in the encoder's first convolution layer. The final outputs of the U-net are interpreted as updates Δq , and added to the corresponding input fields q^n to produce the time-step fields q^{n+1} . The default resolution of input is 200 for mass grids and 199 for velocity grids. The number of parameters for the network is changed by using a multiplier for the channel counts of all hidden layers (Fig. 2 shows a multiplier of 16).

IV. GEOMETRIC CONSTRAINTS

Many PDEs have symmetries: certain spatial transformations of initial and boundary conditions lead to a corresponding transformation of the system state at future time points. We aim to improve ML-based PDE solvers by endowing them with these properties as a hard constraint built into the neural architecture. This effectively narrows the class of functions through which we are searching for an effective and efficient solver, by filtering out functions inconsistent with the symmetry.

A. Equivariance

271

Suppose we have a finite group of symmetries $g \in G$ acting



FIG. 4. Empirical verification of identical PDE and GE-CNN symmetries for the shallow water equations. (a) ICs, ζ , and u for the numerical solver (top left). A flipped version is shown with inversion of u (bottom left). One step of numerical integration produces a new ζ (top, third column), from which we compute a flipped version. Integration from the flipped/inverted ICs produces a different ζ (bottom, third column), which precisely matches the flipped output from the non-flipped ICs (bottom right). (b) As in 'a,' but for a trained U-net based solver. Note however that the equivariance of the network is not learned, but exists as a hard constraint throughout the training process.

on a set of spatially extended fields q by transformations $q \rightarrow_{285}$ $\mathcal{T}_{g}(q)$, with $\mathcal{T}_{g_1g_2} = \mathcal{T}_{g_1} \circ \mathcal{T}_{g_2}$. A function $\Psi(q)$ is equivariant

when transforming its inputs is equivalent to transforming its outputs. Concretely, for each $g \in G$ the transformations $\mathcal{T}_{g}, \mathcal{T}_{g'}^{286}$

 $_{276}$ act on Ψ 's inputs and outputs respectively, and

277

$$\forall g, q : \Psi(\mathcal{T}_{g}q) = \mathcal{T}_{g}'\Psi(q) \tag{10}^{291}$$

For example, let Ψ denote time integration of the 2D heat²⁹³ equation $\frac{\partial q}{\partial t} = \kappa \left(\frac{\partial^2 q}{\partial x_1^2} + \frac{\partial^2 q}{\partial x_2^2} \right)$. This Ψ is equivariant to rota-²⁹⁴ tions, reflections, and translations of the heat field q. In this²⁹⁵ case both the input and output transformations are simply the²⁹⁶ same point-to-point mappings of the scalar heat fields, but for more complex PDEs involving vector fields the transforma-²⁹⁷ tions can be more involved (see below).

B. Equivariant Convolutions

We now describe the construction of convolutional networks with equivariance as a hard constraint, meaning that the \widehat{S}_{ϕ} is equivariant for any ϕ . Throughout this section we follow⁵⁹, but simplify notation by describing a single input and output channel, both of which are *n*-D fields of the same size. We denote by $\mathbb{Y} \subset \mathbb{Z}^n$ the regular grid of integer valued coordinates on which the input and output channels are defined.

A standard convolutional layer applies an *n*-D convolutional filter W to a spatially extended scalar input field q to produce a scalar output field $q \star W$:

$$[q \star W](x) = \sum_{y \in \mathbb{Y}} q(y)W(y-x) + b \tag{11}$$

where W(y) = 0 for y outside the spatial extent of the filter. $q_{,346}$ W, and $q \star W$ are simply real-valued functions on \mathbb{Y} , while b_{347} is a scalar. Standard convolutions are equivariant with respect³⁴⁸ to translations, but not other symmetries.

In contrast, equivariant convolutional layers produce out- $_{350}$ puts that are real-valued functions of an extended discrete $_{351}$ group *H* generated by the symmetry group *G* of interest as well as translations in \mathbb{R}^n . The first such layer takes standard scalar fields as input:

³⁰⁷
$$[q \star W](h) = \sum_{y \in \mathbb{Y}} q(y)W(h^{-1}y) + b \qquad (12)^{355}_{356}$$

Subsequent layers use functions on H as both inputs and out-³⁵⁷ puts:

310
$$[\gamma \star W](h) = \sum_{h' \in \mathbb{H}} \gamma(h') W(h^{-1}h') + b \qquad (13)_{359}$$
360 360

³¹¹ γ , W, and $\gamma \star W$ are real-valued functions of H. As shown³⁶¹ ³¹² in⁵⁹, Eqs. 12-13 satisfy equivariance (Eq. 10). The input trans-³⁶³ ³¹³ formation in Eq. 12 is simply G's action on \mathbb{Z}^n described by

G, while other transformations act on real-valued functions of

³¹⁵ *H*. For any such function $\alpha(h)$ we have

316

$$[T_h \alpha](h') = \alpha(h^{-1}h') \tag{14}_{_{366}}$$

We visualize functions on H as collections of maps over \mathbb{Y} ,

with one map (i.e. real-valued function on \mathbb{Y}) for each $g \in G_{366}$ (Fig. 3). Since the nonzero regions of q, γ and W are bounded,₃₆₇ the outputs' nonzero regions are as well. In Eq. 12, W is de-₃₆₈ fined on a patch of \mathbb{Z}^n , but in Eq. 13 W is a function on H. To₃₆₉ include multiple input output and channels, we simply sum₃₇₀ over inputs for each output in Eqs. 11-13, and note that the₃₇₁ arrays storing W acquire two additional dimensions. The bias h is then indexed by the output additional dimension or

 $_{325}$ *b* is then indexed by the output channel, but not by location or ₃₂₆ group element.

To build an equivariant convolutional network, a sequence of equivariant convolutional layers is interspersed with point-

³²⁹ wise nonlinearities. To obtain an equivariant final output on₂₇₂

³³⁰
$$\mathbb{Y}$$
 instead of *H*, a pooling operation (e.g. a mean or maxi-

mum) operates 'along the *G*-axis' of the *G*-indexed collection

 $_{332}$ of maps on $\mathbb Y.$ Concretely, we take the mean or maximum over

all elements of H that share the same translational component.³⁷⁴ 333 For a symmetry group of size |G|, a standard convolu-³⁷⁵ 334 tion with c input and output channels has as many param-³⁷⁶ 335 eters as an equivariant layer with $c/\sqrt{|G|}$ channels. When³⁷⁷ 336 \mathbb{Y} is a *D*-dimensional grid with *N* points per axis, the com-³⁷⁸ 337 putational complexity of the forward and backward passes³⁷⁹ 338 is $O(N^D K^D c^2)$ in both cases, since the equivariant network³⁸⁰ 339 has |G| times fewer input-output channel pairs but each in-³⁸¹ 340 put channel must be convolved with a transformed slice of the³⁸² 341 383 filter bank |G| times. 342 384

343 1. Reflection-equivariant 1-D Convolutions

on the concrete example of 1-D reflective symmetries. This two-element group is the only nontrivial symmetry group and the main focus of this paper. Here $\mathbb{Y} = \{-N, ..., N\}$, *G* contains the identity and a reflection, and *H* consists of either element of *G* followed by any translation. In this case, a convolutional time stepping network $\widehat{S}_{\phi} : q^n \to \widehat{q}^{n+1}$ is equivariant if $R_F(\widehat{S}_{\phi}(q)) = \widehat{S}_{\phi}(R_F(q))$, for any input field(s) *q* and where the 'mirroring' operator R_F reflects the fields on the spatial axis, i.e. $[R_F(q)](x) = q(-x)$.

A standard 1-D convolutional layer in Eq. 11 with c_{in} inputs q and c_{out} outputs a is defined as:

$$a_{j,\cdot} = \sum_{i=1}^{c_{\text{in}}} W_{j,i,\cdot} \star q_{i,\cdot} + b_j$$
(15)

where the \cdot symbol denotes all values along a given axis. *W* is an $c_{in} \times c_0 \times K$ array for filter size *K*, while *b* is a c_{out} -element vector.

In 1-D reflection-equivariant networks, the first layer is a special case with c_{in} input channels q defined on \mathbb{Y} and c_1 outputs a^1 defined on H. Eq. 12 thus becomes:

$$a_{j,0,\cdot}^{1} = \sum_{i=1}^{c_{\text{in}}} W_{j,i,\cdot}^{1} \star q_{i,\cdot} + b_{j}^{1}$$
(16)

$$a_{j,1,\cdot}^{1} = \sum_{i=1}^{c_{\text{in}}} R_{\text{F}} \left(W_{j,i,\cdot}^{1} \right) \star q_{i,\cdot} + b_{j}^{1}$$
(17)

While q, W^1, b^1 have the same size here as in standard convolutional layers, a^1 gains a third dimension that indexes the elements of G.

For subsequent layers, both the $c_{\ell-1}$ input channels $a^{\ell-1}$ and c_{ℓ} outputs a^{ℓ} are defined on *H* and are stored in 3-D arrays. Eq. 13 becomes:

$$a_{j,0,\cdot}^{\ell} = \sum_{i=1}^{c_{\ell-1}} W_{j,i,0,\cdot} \star a_{i,0,\cdot}^{\ell-1} + W_{j,i,1,\cdot} \star a_{i,1,\cdot}^{\ell-1} + b_j^{\ell}$$
(18)

$$a_{j,1,\cdot}^{\ell} = \sum_{i=1}^{c_{\ell-1}} R_{\mathrm{F}} \left(W_{j,i,1,\cdot} \right) \star a_{i,0,\cdot}^{\ell-1} + R_{\mathrm{F}} \left(W_{j,i,0,\cdot} \right) \star a_{i,1,\cdot}^{\ell-1} + b_{j}^{\ell}$$
(19)

The filter bank *W* now has four dimensions, the third of which indexes *G*. When computing results at the second index along the second, *G*-indexing dimension of each output channel (Eq. 19), the filters are flipped on the spatial axis and permuted on the *G* axis (Fig. 3). While the equivariance of these layers follows as a special case of the results in⁵⁹, we include simple proofs for the case of 1-D reflections in the Appendices A 1-A 2.

Finally, to produce a network output that is defined on a simple 1-D grid (not as a function on H), we use a mean pooling operation over the symmetry dimension

$$y_{j,.}^{\text{out}} = \left(a_{j,0,.}^{L} + a_{j,1,.}^{L}\right)/2 \tag{20}$$

Having introduced equivariant convolutions for any finite₃₈₇ symmetry group *G* acting on a discrete grid \mathbb{Y} , we next focus₃₈₈

385

386

which also obviously has the desired equivariance property. Thus, by chaining together these input, internal, and output layers, our entire network Ψ is reflection equivariant.

389 2. Extension to Mixed Scalar-Vector Inputs

Unfortunately, the equivariance defined for convolutions 390 above does not match the reflection symmetries of many 391 PDEs, since it fails to account for differences in how vector 392 and scalar fields are affected by rotation and reflection. For 393 a scalar field, the value of the transformed field (e.g. heat) is 394 simply the value of the original field at a different point. But 395 for vector fields (e.g. velocity) both a location change and a 396 reflection/rotation of the vector at the corrected location are 397 required. Simply transforming each component of the veloc-398 ity field in a PDE solution as a separate scalar would yield a 399 new field that does not solve the PDE. 400

⁴⁰¹ For reflections of 1-D vector field u, the necessary transfor-⁴³²₄₃₂ mation is

$$[R_{\rm F}u](x) = -u[-x] \tag{21}_{_{434}}^{_{433}}$$

To implement the proper transformation when q contains both scalar fields ζ and vector fields u, we define the following

⁴⁰⁶ input layer (compare to Eq. 19):

403

Since the output of this layer is a real-valued function on H_{445}^{440} subsequent equivariant layers can be used without modifica-⁴⁴⁶ tion in Eq. 13. Defining an equivariant output layer to produce⁴⁴⁷ vector fields is straightforward, but because we construct hy-⁴⁴⁸ brid solvers (see below) scalar outputs are sufficient for our⁴⁴⁹ purposes. ⁴⁵⁰

⁴¹⁵ We prove the equivariance of our input layer in A 3. For an ⁴¹⁶ empirical confirmation of this, Fig. 4 shows the equivariance ⁴¹⁷ of a classical PDE solver *S* and a trained equivariant convo-⁴⁵² ⁴¹⁸ lutional network \hat{S}_{ϕ} for the shallow water equations, which⁴⁵³ ⁴¹⁹ govern scalar height and vector velocity fields. ⁴⁵⁴

420 C. One-dimensional shallow water equations

421 We evaluated our learning strategies using the 1-D SWEs,⁴⁵⁶ 422 composed of momentum and continuity equations: 457

$$\frac{\partial u}{\partial t} = -C_D \frac{1}{h} u |u| - g \frac{\partial \zeta}{\partial x}$$
(24)⁴⁵⁹

 $\frac{\partial \zeta}{\partial t} = -\frac{\partial (hu)}{\partial x}$

with spatial coordinate $x \in [0, L]$, time *t*, velocity *u*, surface

⁴²⁶ disturbance ζ , total depth $h = d + \zeta$, bottom drag C_D , and grav-⁴⁶³

⁴²⁷ itational acceleration g. SWEs are commonly used to describe

⁴²⁸ large-scale flows in coasts, oceans, estuaries, and rivers, based₄₆₄

429 on the assumption that fluid depth is well below the length465

TABLE I. Simulation parameters used for SWEs.

Parameters	Explanation	Value
L	simulation domain	2000(Km)
d	undisturbed water depth	100(m)
C_D	bottom drag coefficient	1.0e - 3
g	acceleration due to gravity	$9.81 (m/s^2)$
Δx	space step	10(Km)
Δt	time step	300(s)
^W imp	implicit weighting	0.5

scale of horizontal motion, as illustrated in Fig. 1a. Our dry BCs, common in riverine and coastal models^{68,73,74}, mean that no fluid enters or escapes:

$$u(x=0) = u(x=L) = 0$$
 (26)

$$\zeta(x=0) = \zeta(x=L) = 0$$
(27)

By default we use 'Gaussian bell' ICs:

$$u(x,0) = 0 \tag{28}$$

$$\zeta(x,0) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/\sigma^2}$$
(29)

with μ and σ uniformly distributed on [100*Km*, 1900*Km*] and [10*Km*, 100*Km*] respectively.

Substituting $\zeta \leftarrow R_{\rm F}(\zeta), u \leftarrow -R_{\rm F}(u)$ into Eqs. 24-27 demonstrates reflection equivariance of the SWEs, which we confirmed empirically in Fig. 4a.

Since closed, wave-reflecting boundaries tend to require minuscule time steps for explicit schemes, we used a semiimplicit scheme⁶⁸ to generate reference simulations and to construct loss functions (Eq. 9) for unsupervised learning. This finite difference method stores velocities and surface elevations on staggered grids (details in the B, simulation parameters in Table I). Its computational cost is dominated by solving a tridiagonal linear system

$$A\zeta^{n+1} = b \tag{30}$$

Where *A*, *b* are functions of ζ^n and u^n . The relative costs of calculating the coefficients of *A* and *b* or computing u^{n+1} given ζ^{n+1} are negligible.

V. EVALUATION METRICS

We compare trained solvers \hat{S}_{ϕ} to a reference numerical method *S* with four error measures described previously⁷⁵. We calculate these metrics for each individual field \hat{z}^n estimating $z^n \in q^n$.

• *Normalized Root Mean Square Error* (NRMSE) describes a relative difference between estimated and reference solutions:

NRMSE =
$$\frac{\|\widehat{z}^n - z^n\|_2}{\|z^n\|_2}$$
 (31)

Note that all estimated fields \hat{z}^n are integrated *n* time steps from the ICs of the reference solution.

 $(23)^{443}$ S

455

 $(25)_{461}$

462

436



498

499

500

501

502

503

504

505

506

523

524

525

526

537

538

539 540

FIG. 5. Progress and success rate of training for standard and₅₀₇ reflection-equivariant convolutional networks. (a) Training loss₅₀₈ curves for standard CNN and GE-CNN solvers. During training the loss of GE-CNN solver is a little larger than one of CNN solver. (b) Training success rate standard and GE-CNNs. Here, high- and⁵¹⁰ small network respectively have about 148M and 1.6M parameters.⁵¹¹

466	• Time-averaged Normalized Root Mean Square Error513
467	$(\mathbb{E}_t \text{NRMSE})$ averages the NRMSE over the full dura-
468	tion of a simulation (in this work, 1200 time steps span-514
469	ning 100 simulated hours). 515

• Pearson's correlation $\rho(\hat{z}^n, z^n)$ of reference and learned solutions.

• *Training Success Rate* (TSR) is the probability that⁵¹⁹ training will converge to a \mathbb{E}_t NRMSE < 10. This mea-⁵²⁰ sure allows us to incorporate the stochastic aspect of⁵²¹ deep learning in evaluating performance (both the initial weights and the order of ICs differ across runs). In this work, we quantify TSR by using \mathbb{E}_t NRMSE – ζ for⁵²² SWEs.

479 VI. SOFTWARE IMPLEMENTATION

Classical numerical and machine learning solvers are527 480 implemented in Pytorch and Numpy. Code for equivariant₅₂₈ 481 convolutions is partly adapted from GrouPy at https: 529 482 //github.com/jornpeters/GrouPy/tree/pytorch\ 483 530 _p4_p4m_gconv/groupy/gconv, while code for training₅₃₁ 484 on evolving simulations is based partly on code published₅₃₂ 485 in³⁴. https://github.com/wandeln/Unsupervised_533 486 Deep_Learning_of_Incompressible_Fluid_Dynamics. 534 487 The code for both numerical- and ML solver are publicly at₅₃₅ 488 https://github.com/m-dml/GE-CNN_learning_SWEs. 536 489

490 VII. EXPERIMENTS

We carried out unsupervised training of a hybrid PDE in-541 tegration scheme for the SWEs, to determine whether hard542

symmetry constraints improve long-term accuracy and stability. Our reference simulations (see section IV C) used a staggered grid with 200 height and 199 velocity points shown in
Fig. 1b.

The neural network inferred surface height ζ_n from the system state at step t - 1. Following³⁴, we provide 13 input channels describing u_{n-1}, z_{n-1} and the BCs:

input =
$$(\zeta, h, m_{\zeta}^{b}, m_{\zeta}, m_{\zeta}^{b} \cdot \zeta, m_{\zeta} \cdot \zeta, m_{\zeta}^{b} \cdot h, m_{\zeta} \cdot h, u, m_{u}^{b}, m_{u}, m_{u}^{b} \cdot u, m_{u} \cdot u)$$
 (32)

Here, m_{ζ} is a mask for ζ . It is zero for boundary values and one for interior values while $m_{\zeta}^b = 1 - m_{\zeta}$, and m_u, m_u^b are the same for velocities. To deal with different spatial dimensions across input channels, we used kernel size 6 for ζ -sized inputs and kernel size 7 for *u*-sized inputs, and added the results in Fig. 2a.

In our hybrid scheme, the neural network replaced the expensive tridiagonal solve in Eq. 30 to compute ζ^{n+1} , while the numerical scheme computes coefficients of the tridiagonal system and updates u^{n+1} while imposing BCs. We can therefore drop the second term in Eq. 9 and replace the first with:

$$\mathcal{L}_{SWE} = \|A\zeta^{n+1} - b\|_2^2$$
 (33)

We compared equivariant networks (GE-CNN) to standard convolutional U-nets with the same architecture, loss and training procedure. For both network types, we adjusted the number of trainable parameters by scaling the number of output channels for all convolutions except the final layer, and for our default configuration as shown in Fig. 2a. This resulted in 148M parameters for the equivariant convolutions and 149M for standard convolutions.

A. Equivariance Improves Accuracy and Convergence

We compared default configurations of our standard convolutional neural networks (CNN) to reflection-equivariant versions (GE-CNN), see Fig. 2a. We trained on a library of Gaussian bells ICs with occasional resets, as shown in section III C. Both standard and equivariant convolutional networks achieved low loss values and accurately predicted how the next time step for the SWEs (Fig. 5a). Since the standard CNN architecture describes a less restricted function class than the GE-CNN, it achieves a slightly lower loss value during training, at the cost of failing to respect symmetry in autoregressive predictions.

To test whether respecting symmetry would improve long term-accuracy, we therefore evaluated we compared the CNN and GE-CNN after training in autoregressive rollouts. Network outputs were used to define inputs for the next time step, and the results were compared to reference numerical solutions over 1200 time steps (100 simulated hours, Fig. 6a). The reference solutions describe waves propagating outward from the initial Gaussian bell before reflecting off the domain boundary five times (Fig. 6a, Fig. 6d black).



FIG. 6. GE-CNNs produce accurate SWE rollouts from Gaussian bell ICs. (a) Reference simulation of surface elevation ζ and velocity *u*. (b) Rollouts from CNN solver for ζ and *u*, with errors (prediction – reference). (c) As in 'b,' but for GE-CNNs. (d) Snapshots of ζ and *u* obtained from the reference solver (black), CNN (blue) and GE-CNN orange. (e) Mean over ICs of NRMSE for ζ and *u* as a function of time from the start of the simulation, for CNNs (blue) and GE-CNNs. (f) As in 'e,' but for correlation. (g) Histograms of time averaged NRMSE in ζ and *u* for CNNs (blue) and GE-CNNs (orange).

Individual CNN rollouts successfully reproduced the prop-560 543 agation and reflection of waves, but exhibited gradually in-561 544 creasing errors that increased when the waves were reflected 545 by the closed boundaries (Fig. 6b). Over the CNN rollout the $_{563}$ 546 waves broadened, developed additional peaks in ζ not present₅₆₄ 547 in the reference simulation. Compared to to reference sim-548 ulations, CNN rollouts exhibited higher spatial frequencies, 566 549 a positive velocity bias and spatially asymmetric errors (Fig. 567 550 6d, blue). By the end of the rollout the magnitude of errors 551 reached the amplitude of the simulated wave heights and ve-552 locities. 553 570

The GE-CNN followed the reference solution more closely,⁵⁷¹ with errors at least one order of magnitude smaller than the⁵⁷² simulated signals, and difficult to discern visually (Fig. 6c).⁵⁷³ The shape and width of the propagating and reflecting waves⁵⁷⁴ and surrounding smaller ripples closely matched the reference⁵⁷⁵ simulation. By the end of the simulation, errors appear as⁵⁷⁶ additional undulations in ζ , while *u* continues to to follow the reference simulation closely (Fig. 6d, orange).

The training success rate, defined as the probability over multiple training runs and ICs of achieving low time-averaged error (in Section V), was 10/12 for the GE-CNN but only 2/12 for the CNN (Fig. 5b). When using networks ~ 100 times smaller, we observed 9/12 successes for the GE-CNN (1.6M parameters) and 1/12 for the CNN (1.7M).

We further measured how accuracy of CNN and GE-CNN rollouts varied over time and ICs by computing rollouts for each over 200 ICs. NRMSE (see Section V) in both ζ and u increased more quickly for the CNN, reaching average values 1-2 order of magnitude higher (Fig. 6e). Correlation coefficients between rollouts and the reference simulation followed a similar trend, with a decrease over time but clear superiority for the GE-CNN (Fig. 6f). We further examined the distribution of time-averaged NRMSE across ICs, computing histograms



FIG. 7. Reflection equivariance improves accuracy for all network sizes. (a-b) \mathbb{E}_t NRMSE- ζ and \mathbb{E}_t NRMSE-u as functions of network parameter counts in CNNs and GE-CNNs. Error bars show standard deviations which are obtained by using the predictions in 100 *h*. (c-f) Plots of NRMSE- ζ , NRMSE-u, $\rho(\hat{\zeta}, \zeta)$, and $\rho(\hat{u}, u)$ as functions of integration time for several sizes of CNN and GE-CNN.

626

for *u*- and ζ -errors on logarithmic scales (Fig. 6g). The GE-₆₁₁ CNN error distributions peaked near zero, while CNN errors₆₁₂ peaked around the targeted signals' amplitude, with a long-₆₁₃ tailed distribution.

For both CNN and GE-CNN rollouts, \mathbb{E}_t NRMSE- ζ de-615 581 creases with parameter count of the trained networks, and₆₁₆ 582 is lower for GE-CNNs (Fig. 7a-b). The largest CNNs com-617 583 puted ζ as accurately as the smallest GE-CNNs, but for u_{618} 584 even the smallest GE-CNNs outperformed CNNs of all sizes619 585 tested. The same trend was observed at individual time points₆₂₀ 586 for NRMSEs (Fig. 7c-d) and correlation values (Fig. 7e-f).621 587 Overall, these results show that the long-rollout accuracy im-622 588 provement provided by equivariance is robust to the choice of 623 589 network size and accuracy metric. 624 590

⁵⁹¹ B. Generalization Capabilities After Training

We next examined how well PDEs solvers trained using⁶²⁷ Gaussian bell ICs would perform on conditions beyond their⁶²⁸ training data.

We first measured rollout accuracy for an ICs described₆₂₉ 595 by an isosceles triangle in ζ a the domain center with 400₆₃₀ 596 km base and 0.12 m height (Fig. 8a). This system state was631 597 never encountered during training, as it contains a discontinu-632 598 ous first spatial derivative in contrast to the smooth Gaussian633 599 bell. Rollout errors we were considerably higher than for a634 600 novel Gaussian bell scenario, though the GE-CNN was again635 601 more accurate, especially for u (Fig. 8b-c). In the CNN roll-636 602 out the propagating wave dissipated into many high-frequency₆₃₇ 603 ripples, while for the GE-CNN waves propagated with the638 604 correct shape but too slowly, leading to a position mismatch639 605 with reference simulations (Fig. 8d). CNN and GE-CNN roll-640 606 out performance for triangular ICs with random height, width641 607 and position (uniform on 0.09-0.36 m, 200-300 km and 0-642 608 200 km respectively) showed similar trends to Gaussian bells.643 609 GE-CNN was uniformly superior, its errors grew more slowly₆₄₄ 610

over time, and its error distribution peaked near zero while the CNN's peaked above 4 times the estimated signals (Fig. 8g).

For a more challenging generalization task, we used a sum of 3 Gaussian bells as an initial condition. The reference simulation (Fig. 9a) shows 6 propagating and reflecting waves that form a complex interference pattern. As previously, in CNN rollouts the waves were distorted and dissipated over time with a positive bias emerging for *u*, while the GE-CNN maintained the correct shapes but introduced timing errors (Fig. 9c), and was more accurate at every time point (Fig. 9d). We also computed accuracy measures for triple-bell ICs (Fig. 9e-g) with random means (uniform on 100-1900 km) and widths (10-100 km). The GE-CNN yielded better NRMSE and correlation values for all time delays, and a distribution of time-averagd NRMSE that peaked near zero and showed little overlap with CNN results.

C. Learned Representations of Global Mass, Momentum and Energy

A challenge for ML-based PDE solvers is that their predictions do not always satisfy conservation laws, even when these laws are manifested in their unsupervised loss or training data^{71,76}. Our reference SWE solver conserves mass, conserves energy except for bottom drag, and conserves momentum except for bottom drag and boundary effects. Each reflection of a propagating wave from the closed boundaries involves a temporary conversion of kinetic to potential energy.

We investigated how well trained networks represent 4 quantities: mass $\sum h_i$, momentum $\sum h_i u_i$, kinetic energy $\frac{1}{2} \sum h_i u_i^2$, and potential energy $\frac{1}{2} \sum gh_i$. Note that we employed the disturbed water depth *h* to represent mass. We computed these for CNN and GE-CNN rollouts and compared to the reference solver. In individual held-out Gaussian bell ICs (Fig. 10a, upper row), CNN rollouts show a rapid error accumulation in all 4 quantities, while GE-CNNs exhibited a



FIG. 8. GE-CNN solver trained on Gaussian bell ICs generalizes to triangular ICs. (a) Reference simulation of surface elevation ζ and velocity u from a triangular IC. (b) Rollouts from CNN solver for ζ and u, with errors (prediction – reference). (c) As in 'b,' but for GE-CNNs. (d) Snapshots of ζ and u obtained from the reference solver (black), CNN (blue), and GE-CNN (orange). (e) Mean over ICs of NRMSE for ζ and u as a function of time from the start of the simulation, for CNNs (blue) and GE-CNNs. (f) As in 'e,' but for correlation. (g) Histograms of time averaged NRMSE in ζ and u for CNNs (blue) and GE-CNNs (orange).

669

670

671

slow drift in total mass that produced a drift in potential en-658 645 ergy, and negligible errors in momentum and kinetic energy.659 646 When averaging over many ICs, we found that average val-660 647 ues of the conserved quantities matched closely for GE-CNN₆₆₁ 648 and the reference simulation until almost 100 hours, while₆₆₂ 649 the CNN showed clear differences after 50 h (Fig. 10a, lower663 650 row). Similar results were observed for triangular (Fig. 10b)₆₆₄ 651 and multi-bell ICs (Fig. 10c); for these ICs errors grew more₆₆₅ 652 quickly but the GE-CNN matched the reference simulation666 653 more closely. 654 667

655 VIII. DISCUSSION

⁶⁵⁶ We developed reflection-equivariant 1-D convolutional net-⁶⁷⁷ works for mixed vector-scalar inputs, and trained them to solve the SWEs with an unsupervised loss. We showed how these networks can be endowed with the same symmetries and the targeted PDEs, and our experiments showed how they improve accuracy and stability over standard CNNs with similar parameter counts, over a broad range of scenarios and tests. GE-CNNs matched reference simulations more closely at all time points, performed on new IC types and more faithfully represented mass, momentum, and energy. A remarkable aspect of these equivariant networks is that their advantages first become apparent when generating and evaluating longer rollouts, with no differences from standard CNNs apparent during training. Our results show that equivariant architectures offer significant benefits for long-term accuracy and physical consistency, with no modifications to the loss function or training procedures.



FIG. 9. GE-CNN solver trained on Gaussian bell ICs generalizes to multi-bell ICs. (a) Reference simulation of surface elevation ζ and velocity u from a multi-bell IC (centers 400, 1000, 1350 km; widths 3, 5, 7 km). (b) Rollouts from CNN solver for ζ and u, with errors (prediction – reference). (c) As in 'b,' but for GE-CNNs. (d) Snapshots of ζ and u obtained from the reference solver (black), CNN (blue) and GE-CNN orange. (e) Mean over ICs of NRMSE for ζ and u as a function of time from the start of the simulation, for CNNs (blue) and GE-CNNs. (f) As in 'e,' but for correlation. (g) Histograms of time averaged NRMSE in ζ and u for CNNs (blue) and GE-CNNs (orange).

673 A. Related Work

688 mixed input types or solve PDEs.

Our work builds on and complements existing studies seek-674 ing to exploit symmetries for solving PDEs. Authors⁷⁷ use 675 PDE symmetries to design data augmentations for use dur-676 ing training, instead of making their networks equivariant. 677 Ref.⁷⁸ built steerable CNNs and the paper⁷⁹ demonstrated their 678 utility predicting the evolution of incompressible NS and⁶⁹¹ 679 an advected temperature field, but do not consider mixed⁶⁹² 680 scalar/vector inputs and examine their predictions only 10⁶⁹³ 681 time steps into the future. 682 695

In Ref.⁸⁰, the authors use network layers that solve a spe-⁶⁹⁶ cific PDE to build convolutional networks, instead of con-⁶⁹⁷ structing layers to match the symmetry groups of a specific⁶⁹⁸ PDE as we do here. Authors⁸¹ use rotation equivariant convo-⁶⁹⁹ lutional layers to operate on vector fields, but do not consider⁷⁰⁰ The equivariant convolution layers we have developed for mixed scalar/vector inputs could also be realized using steerable convolutions⁷⁹ with the correct combination of scalar and vector capsules. Instead of transforming filter banks, steerable convolutions are based on optimizing convolution weights within a pre-computed linear subspace that satisfies the desired constraints. While this approach is flexible and efficient, we believe our layers can provide considerable utility through their mathematical simplicity. Some studies have also reported successful implementation of equivariant network through filter bank transformation, but could not achieve the same results through steerable convolutions⁸².



FIG. 10. GE-CNN solver robustly predicts mass, momentum, and energy. (a) Global mass, momentum, kinetic- and potential energy from the reference solver, CNN- and GE-CNN rollouts (upper; same IC as Fig. 6). Mean and standard error of the mean for global mass, momentum, kinetic- and potential energy over 200 Gaussian bell ICs. (b) As in 'a,' but for triangular ICs. (c) As in 'a,' but for multi-bell ICs.

714

715

701 B. Future Outlook

In future work, we intend to extend our results to higher₇₁₆ 702 dimensional and more complex systems, and to combine ge_{717}^{-17} 703 ometric and physical constraints^{83,84}. We also anticipate that $\frac{11}{718}$ 704 by offering a combination of long-rollout performance and au-705 tomatic differentiability, equivariant deep PDE solvers could 706 prove useful for solving inverse problems^{85,86}. The observed 707 performance gains for long rollouts could also find useful ap-719 708 plications in climate, weather and ocean modeling, which re-709 quire stability and accuracy over far longer time intervals than₇₂₀ 710 commonly evaluated scenarios for deep PDE solvers. 721 711

712 ACKNOWLEDGMENTS

We thank Kai Logemann for providing code and valuable insights into the semi-implicit shallow water solver. We also thank Vien Minh Nguyen-Thanh, Ali Can Bekar, Tobias Schanz, Shivani Sharma, Vadim Zinchenko, and Andrey Vlasenko for useful discussions and comments on the manuscript.

Appendix A: Proof of Equivariance for Convolution Layers

For completeness, we first provide proofs for the equivariance of the original scalar-field-only convolution layers⁵⁹

in our notation. We then prove equivariance for our mixed749 Flipping outputs gives 722

scalar/vector input layers. 723

1. Proof of Equivariance for Scalar-input Convolutional Input 724 Layer 725 751

The outputs of the first (input) layer of an equivariant con-726 volutional network $a_{j,0}^1$ and $a_{j,1}^1$ are defined in Eq. (16) and⁷⁵² 727 Eq. (17). We flip the input $R_{\rm F}(q_{i,.})$. Applying this layer to a 728

fliped input using the same weights and biases gives: 729 754

73

$$\tilde{a}_{j,0,\cdot}^{1} = \sum_{i=1}^{c_{\text{in}}} W_{j,i,\cdot}^{1} \star R_{\text{F}}(q_{i,\cdot}) + b_{j}^{1}$$
(A1)

an
$$\tilde{a}_{j,1,\cdot}^{1} = \sum_{i=1}^{c_{\text{in}}} R_{\text{F}}\left(W_{j,i,\cdot}^{1}\right) \star R_{\text{F}}\left(q_{i,\cdot}\right) + b_{j}^{1}$$
 (A2)757

758 To prove the equivariance in this layer, we flip the first layer 732

of output. This flipping operator can be moved into the con-733

volution. Then, we obtain 734

$$R_{\rm F}(\tilde{a}_{j,0,\cdot}^{1}) = R_{\rm F}\left(\sum_{i=1}^{c_{\rm in}} W_{j,i,\cdot}^{1} \star R_{\rm F}(q_{i,\cdot}) + b_{j}^{1}\right)$$

$$= \sum_{i=1}^{5m} R_{\rm F} \left(W_{j,i,\cdot}^1 \right) \star q_{i,\cdot} + b_j^1 = a_{j,1,\cdot}^1 \qquad (A3)^{763}$$

737
$$R_{\rm F}(\tilde{a}_{j,1,\cdot}^{1}) = R_{\rm F}\left(\sum_{i=1}^{c_{\rm in}} R_{\rm F}\left(W_{j,i,\cdot}^{1}\right) \star R_{\rm F}\left(q_{i,\cdot}\right) + b_{j}^{1}\right)$$

$$= \sum_{i=1}^{C_{\text{in}}} W_{j,i,\cdot}^{1} \star q_{i,\cdot} + b_{j}^{1} = a_{j,0,\cdot}^{1} \qquad (A4)^{768}$$

Now, these two equations satisfy the definition of the group₇₇₀ 739 equivariance in Eq. (10). Thus, we finish the proof. An ex-740 ample plot for the group equivariance of this layer is shown in 741 Fig. (3). 742

2. Proof of Equivariance for Non-Input Convolution Layers 772 743

773 The output of subsequent layers, for which inputs and out-744 put channels are both real-valued functions on H, is given in 745 774 eq. 19. A flipped input $R_{\rm F}(x_{i,\cdot})$ gives the output 746

$$\tilde{a}_{j,0,\cdot}^{\ell} = \sum_{i=1}^{c_{\ell-1}} W_{j,i,0,\cdot} \star \tilde{a}_{i,0,\cdot}^{\ell-1} + W_{j,i,1,\cdot} \star \tilde{a}_{i,1,\cdot}^{\ell-1} + b_j^{\ell}$$
(A5)

⁷⁴⁸
$$\tilde{a}_{j,1,\cdot}^{\ell} = \sum_{i=1}^{c_{\ell-1}} R_{\mathrm{F}}(W_{j,i,0,\cdot}) \star \tilde{a}_{i,1,\cdot}^{\ell-1} + R_{\mathrm{F}}(W_{j,i,1,\cdot}) \star \tilde{a}_{i,0,\cdot}^{\ell-1} + b_{j}^{\ell}$$
(A6)⁷⁷⁶

750

755

756

759

760

761

762

765

766

771

$$R_{\rm F}(\tilde{a}_{j,0,\cdot}^{\ell}) = R_{\rm F}\left(\sum_{i=1}^{c_{\ell-1}} W_{j,i,0,\cdot} \star \tilde{a}_{i,0,\cdot}^{\ell-1} + W_{j,i,1,\cdot} \star \tilde{a}_{i,1,\cdot}^{\ell-1} + b_j^{\ell}\right)$$
(A7)

$$R_{\rm F}(\tilde{a}_{j,1,\cdot}^{\ell}) = R_{\rm F} \left(\sum_{i=1}^{c_{\ell-1}} R_{\rm F}(W_{j,i,0,\cdot}) \star \tilde{a}_{i,1,\cdot}^{\ell-1} + R_{\rm F}(W_{j,i,1,\cdot}) \star \tilde{a}_{i,0,\cdot}^{\ell-1} + b_j^{\ell} \right)$$
(A8)

Then, we move the flipping operator $R_{\rm F}$ into the conversation features. We not only need to flip the weights but also switch the non-flipped input. Thus, Eqs. (A7-A8) can be written as

$$R_{\rm F}(\tilde{a}_{j,0,\cdot}^{\ell}) = \sum_{i=1}^{c_{\ell-1}} R_{\rm F}(W_{j,i,1,\cdot}) \star a_{i,0,\cdot}^{\ell-1} + R_{\rm F}(W_{j,i,0,\cdot}) \star a_{i,1,\cdot}^{\ell-1} + b_j^{\ell}$$

$$= a_{j,1,\cdot}^{\ell}$$
(A9)

$$R_{\rm F}(\tilde{a}_{j,1,\cdot}^{\ell}) = \sum_{i=1}^{c_{\ell-1}} W_{j,i,0,\cdot} \star a_{i,0,\cdot}^{\ell-1} + W_{j,i,1,\cdot} \star a_{i,1,\cdot}^{\ell-1} + b_j^{\ell} = a_{j,0,\cdot}^{\ell}$$
(A10)

Therefore, according to the definition of equivariance, we have proven the equivariance convolution in subsequent layers. The example plot is also illustrated in Fig. (3).

3. Proof of Equivariance for Mixed Scalar-Vector **Convolution Layers**

The first layer's outputs for mixed scalar-vector inputs are shown in Eqs. (22-23). Here, we prove the equivariance in this layers. According to the symmetry of the vector field shown in Eq. (21), we transform the input as $R_{\rm F}(\zeta_{i,\cdot})$ and $-R_{\rm F}(u_{i,\cdot})$. Thus, the first layer of output using the flipping input is written as

$$\tilde{a}_{j,0,\cdot}^{1} = \sum_{i=1}^{c_{in}^{s}} W_{j,i,\cdot}^{\zeta} \star R_{\mathrm{F}}(\zeta_{i,\cdot}) + \sum_{i=1}^{c_{in}^{u}} W_{j,i,\cdot}^{u} \star - R_{\mathrm{F}}(u_{i,\cdot}) + b_{j}^{\ell}$$
(A11)

$$\tilde{a}_{j,1,\cdot}^{1} = \sum_{i=1}^{c_{in}^{\xi}} R_{\mathrm{F}}(W_{j,i,\cdot}^{\zeta}) \star R_{\mathrm{F}}(\zeta_{i,\cdot}) + \sum_{i=1}^{c_{in}^{u}} -R_{\mathrm{F}}(W_{j,i,\cdot}^{u}) \star -R_{\mathrm{F}}(u_{i,\cdot}) + b_{j}^{\ell}$$
(A12)

Next, we flip these outputs

$$R_{\rm F}(\tilde{a}_{j,0,\cdot}^{1}) = R_{\rm F}\left(\sum_{i=1}^{c_{in}^{\zeta}} W_{j,i,\cdot}^{\zeta} \star R_{\rm F}(\zeta_{i,\cdot}) + \sum_{i=1}^{c_{in}^{u}} - W_{j,i,\cdot}^{u} \star R_{\rm F}(u_{i,\cdot}) + b_{j}^{\ell}\right)$$
(A13)

$$R_{\rm F}(\tilde{a}_{j,1,\cdot}^{1}) = R_{\rm F}\left(\sum_{i=1}^{c_{in}^{u}} R_{\rm F}(W_{j,i,\cdot}^{\zeta}) \star R_{\rm F}(\zeta_{i,\cdot}) + \sum_{i=1}^{c_{in}^{u}} R_{\rm F}(W_{j,i,\cdot}^{u}) \star R_{\rm F}(u_{i,\cdot}) + b_{j}^{\ell}\right)$$
(A14)

Now, we move the flipping operator into the convolu-813 written as 777 tion operator. The flipping weight and input feature can be 778

changed as the following equations, 779

$$R_{\rm F}(\tilde{a}_{j,0,\cdot}^{1}) = \sum_{i=1}^{c_{in}^{\zeta}} R_{\rm F}(W_{j,i,\cdot}^{\zeta}) \star \zeta_{i,\cdot} + \sum_{i=1}^{c_{in}^{u}} -R_{\rm F}(W_{j,i,\cdot}^{u}) \star u_{i,\cdot} + b_{j}^{\ell} {}^{_{815}}_{_{816}}$$

$$= a_{j,1,\cdot}^{1} \qquad (A15)$$

793

$$R_{\rm F}(\tilde{a}_{j,1,\cdot}^{1}) = \sum_{i=1}^{c_{in}^{\zeta}} W_{j,i,\cdot}^{\zeta} \star \zeta_{i,\cdot} + \sum_{i=1}^{c_{in}^{u}} W_{j,i,\cdot}^{u} \star u_{i,\cdot} + b_{j}^{\ell} = a_{j,0,\cdot}^{1}$$
(A16)⁸¹⁸

Thus, according to the definition of equivariance of convolu-783

tion, we have proven the equivariance for mixed scalar-vector⁸¹⁹ 784 convolution layers. 785 820

Appendix B: Numerical Discretization of SWEs 786

Here we describe how the space- and time-discretized vari-823 787 able fields u^n and ζ^n of the SWE at the *n*-th time step are used₈₂₄ 788 to compute the (n + 1)-th time step. We describe the proce-825 789 dures used for the semiiplict non-deep-learning-based classi-826 790 cal numerical solver, which is a biased upwind scheme 68 . 791

We discretize the momentum equation (eq. 24) as follows: 792

$$u^{n+1} = u^n - \Delta t C_D \frac{1}{h} u^n |u^n| - \Delta t g (1 - w_{imp}) \frac{\partial \zeta^n}{\partial x} - \Delta t g w_{imp} \frac{\partial \zeta^{n+1}}{\partial x} \frac{\partial \zeta^{n+1}}{(B1)^{828}}$$

where w_{imp} is a fixed parameter controlling weighting be-⁸²⁹ 794 tween implicit and explicit time stepping. The mass equation⁸³⁰ 795 (eq. 25) is discretized as: 796 832

$$\zeta^{n+1} = \zeta^n - \Delta t (1 - w_{\text{imp}}) \frac{\partial h^n u^n}{\partial x} - \Delta t w_{\text{imp}} \frac{\partial h^n u^{n+1}}{\partial x} \quad (\text{B2})_{\text{B35}}^{\text{B33}}$$

Recall that $h = d + \zeta$ and d is the undisturbed water depth.⁸³⁷ 798 838 Eq. (B1) is inserted into eq. (B2) to obtain 799 839

$$\zeta^{n+1} = \zeta^n - \Delta t (1 - w_{\text{imp}}) \frac{\partial h^n u^n}{\partial x} - \Delta t w_{\text{imp}} \frac{\partial h^n u^*}{\partial x}$$

$$+\Delta t^2 w_{imp}^2 g \frac{\partial^2 h^n \zeta^{n+1}}{\partial x^2}$$
(B3)⁸⁴³
⁸⁴³
⁸⁴³

where u^* is an 'interim solution' defined by 802

$$u^{*} = u^{n} - \Delta t c_{D} \frac{1}{h} u^{n} |u^{n}| - \Delta t g (1 - w_{imp}) \frac{\partial \zeta^{n}}{\partial x} \qquad (B4)_{850}^{848}$$

When calculating the product of two variables defined on the852 804 velocity and mass points of the Arakawa C-grid (Fig. 1b), we853 805 interpolate the mass variable to velocity grid points by aver-806 aging adjacent values. For a quantity α defined on the ve-807 locity or mass gird, the first spatial derivative is discretized⁶⁵⁰ as $\frac{\partial \alpha}{\partial x_i} = \frac{\alpha_{i+1/2} - \alpha_{i-1/2}}{\Delta x}$, with outputs staggered by $\Delta x/2$ from⁸⁵⁹ inputs. The second spatial derivative is discretized using the⁸⁵⁹ 808 809 810 second order finite difference $\frac{\partial^2 \alpha}{\partial x^2}_i = \frac{\alpha_{i+1} - 2\alpha_i + \alpha_{i-1}}{\Delta x^2}$, with out-⁸⁶⁰₈₆₁ puts on the same grid as inputs. Therefore, eq. (B3) can be⁸⁶² 811 812

814

$$\zeta_i^{n+1} = \frac{1}{1 + c_E + c_W} \left[\zeta^n + \operatorname{div} + c_E \zeta_{i+1}^{n+1} + c_W \zeta_{i-1}^{n+1} \right]$$
(B5)

where div = $-\Delta t (1 - w_{imp}) \frac{\partial h^n u^n}{\partial x} - \Delta t w_{imp} \frac{\partial h^n u^*}{\partial x}$, while c_E and c_W are defined as

$$c_{E} = \begin{cases} \frac{0.5\Delta t^{2} w_{imp}^{2} g(h_{i}^{n} + h_{i+1}^{n})}{\Delta x^{2}} & if \ h_{i+1}^{n} > 0\\ 0 & \text{otherwise.} \end{cases}$$

$$c_{W} = \begin{cases} \frac{0.5\Delta t^{2} w_{imp}^{2} g(h_{i}^{n} + h_{i-1}^{n})}{\Delta x^{2}} & if \ h_{i-1}^{n} > 0\\ 0 & \text{otherwise.} \end{cases}$$

Eq. (B5) describes a linear system of equations in ζ^{n+1} that can be written in matrix-vector form

$$A\zeta^{n+1} = b \tag{B6}$$

where A is a $N \times N$ tridiagonal matrix $(N = L/\Delta x)$ with where *A* is a $N \times N$ integrating that matrix (N - D/2k) with $A_{k,k} = 1, A_{k,k-1} = -\frac{c_W}{1+c_E+c_W}, A_{k,k+1} = -\frac{c_E}{1+c_E+c_W}$ and all other elements zero. $b \in \mathbb{R}^N$ with $b = \frac{\zeta^n + div}{1+c_E+c_W}$. Following⁶⁸, we employ Gauss-Seidel iterations to solve eq. B6. Having obtained ζ^{n+1} , the new velocity u^{n+1} is calculated as

$$u^{n+1} = u^* - \Delta t g w_{\text{imp}} \frac{\partial \zeta^{n+1}}{\partial x}$$
(B7)

- ¹H. Egger and J. Schöberl, "A hybrid mixed discontinuous galerkin finiteelement method for convection-diffusion problems," SIAM J. Numer. Anal. 30, 1206-1234 (2010).
- ²L. Euler, "Principes généraux du mouvement des fluides," Mémoires de l'académie des sciences de Berlin, 274-315 (1757).

³R. Temam, Navier-Stokes equations: theory and numerical analysis, Vol. 343 (American Mathematical Soc., 2001).

- ⁴P. Constantin and C. Foias, Navier-stokes equations (University of Chicago Press, 2020).
- ⁵B. De St Venant, "Theorie du mouvement non-permanent des eaux avec application aux crues des rivers et a l'introduntion des marees dans leur lit," Academic de Sci. Comptes Redus 73, 148-154 (1871).
- ⁶P. J. Dellar and R. Salmon, "Shallow water equations with a complete coriolis force and topography," Phys. Fluids 17, 106601 (2005).
- ⁷S. Bunya, E. J. Kubatko, J. J. Westerink, and C. Dawson, "A wetting and drying treatment for the runge-kutta discontinuous galerkin solution to the shallow water equations," Comput. Methods Appl. Mech. Eng. 198, 1548-1562 (2009).
- ⁸T. Kärnä, B. De Brye, O. Gourgue, J. Lambrechts, R. Comblen, V. Legat, and E. Deleersnijder, "A fully implicit wetting-drying method for dg-fem shallow water models, with an application to the scheldt estuary," Comput. Methods Appl. Mech. Eng. 200, 509-524 (2011).
- ⁹G. Zängl, D. Reinert, P. Rípodas, and M. Baldauf, "The icon (icosahedral non-hydrostatic) modelling framework of dwd and mpi-m: Description of the non-hydrostatic dynamical core," Quarterly Journal of the Royal Meteorological Society 141, 563-579 (2015).
- ¹⁰M. Klöwer, P. Düben, and T. Palmer, "Number formats, error mitigation, and scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow water model," J. Adv. Model. Earth Syst. 12, e2020MS002246 (2020).
- ¹¹P. Korn, N. Brüggemann, J. H. Jungclaus, S. Lorenz, O. Gutjahr, H. Haak, L. Linardakis, C. Mehlmann, U. Mikolajewicz, D. Notz, et al., "Icon-o: The ocean component of the icon earth system model-global simulation characteristics and local telescoping capability," J. Adv. Model. Earth Syst. 14, e2021MS002952 (2022).

822

840

841

842

846

847

821

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

932

- ¹³V. Casulli, "Semi-implicit finite difference methods for the two-935 dimensional shallow water equations," J. Comput. Phys. **86**, 56–74 (1990).936
- ¹⁴O. Zienkiewicz and P. Ortiz, "A split-characteristic based finite element937 model for the shallow water equations," Int. J. Numer. Methods Fluids.938
 20, 1061–1080 (1995). 939
- ¹⁵F. Bassi and S. Rebay, "A high-order accurate discontinuous finite ele-940 ment method for the numerical solution of the compressible navier–stokes941 equations," J. Comput. Phys. **131**, 267–279 (1997). 942
- ¹⁶R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," Hand-943 book of numerical analysis 7, 713–1018 (2000). 944
- ¹⁷S. Grilli, T. Pedersen, and P. Stepanishen, "A hybrid boundary element⁹⁴⁵ method for shallow water acoustic propagation over an irregular bottom,"⁹⁴⁶ Eng. Anal. Bound. Elem. **21**, 131–145 (1998).
- ¹⁸M. Taylor, J. Tribbia, and M. Iskandarani, "The spectral element method₉₄₈ for the shallow water equations on the sphere," J. Comput. Phys. **130**, 92–949 108 (1997).
- ¹⁹O. Reynolds, "Iv. on the dynamical theory of incompressible viscous fluids951 and the determination of the criterion," Philos. Trans. Royal Soc. A , 123–952 164 (1895).
- ²⁰J. Smagorinsky, "General circulation experiments with the primitive equa-954 tions: I. the basic experiment," Mon. Weather Rev. **91**, 99–164 (1963).
- ²¹ J. W. Deardorff, "A numerical study of three-dimensional turbulent chan-956 nel flow at large reynolds numbers," J. Fluid Mech. 41, 453–480 (1970).
 957
- ²²Y. Saad, *Iterative methods for sparse linear systems* (SIAM, 2003).
 ²³B. T. Polyak, "The conjugate gradient method in extremal problems," 959
- USSR Comput. Math. & Math. Phys. 9, 94–112 (1969). 960 ²⁴ Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algo-961 rithm for solving nonsymmetric linear systems," SIAM J. Sci. Comput. 7,962 856–869 (1986). 963
- ²⁵H. A. Van der Vorst, "Bi-cgstab: A fast and smoothly converging variant964 of bi-cg for the solution of nonsymmetric linear systems," SIAM J. Sci.965 Comput. 13, 631–644 (1992).
- ²⁶B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler,⁹⁶⁷ "Deep fluids: A generative network for parameterized fluid simulations,"⁹⁶⁸ in *Computer graphics forum*, Vol. 38 (Wiley Online Library, 2019) pp.⁹⁶⁹ 59–70. 970
- ²⁷R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards971 physics-informed deep learning for turbulent flow prediction," in *Proceed*-972 ings of the 26th ACM SIGKDD International Conference on Knowledge973 Discovery & Data Mining (2020) pp. 1457–1466. 974
- ²⁸Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stu-975 art, and A. Anandkumar, "Fourier neural operator for parametric partial976 differential equations," arXiv preprint arXiv:2010.08895 (2020). 977
- ²⁹G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson,978
 "U-fno—an enhanced fourier neural operator-based deep-learning model979
 for multiphase flow," Adv. Water Resour. 163, 104180 (2022).
- ³⁰G. Gupta, X. Xiao, and P. Bogdan, "Multiwavelet-based operator learn-981 ing for differential equations," Adv. Neural Inf. Process Syst. 34, 24048–982 24062 (2021).
- ³¹Y. Li, L. Xu, and S. Ying, "Dwnn: Deep wavelet neural network for₉₈₄ solving partial differential equations," Mathematics **10**, 1976 (2022). 985
- ³²G. Kohl, L.-W. Chen, and N. Thuerey, "Turbulent Flow Simulation using⁹⁸⁶ Autoregressive Conditional Diffusion Models," (2023), arXiv:2309.01745⁹⁸⁷ [physics].
- ³³Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stu-989 art, and A. Anandkumar, "Neural operator: Graph kernel network for990 partial differential equations," arXiv preprint arXiv:2003.03485 (2020). 991
- ³⁴N. Wandel, M. Weinmann, and R. Klein, "Learning incompressible fluid992 dynamics from scratch-towards fast, differentiable fluid models that gen-993 eralize," arXiv preprint arXiv:2006.08762 (2020). 994
- ³⁵N. Wandel, M. Weinmann, and R. Klein, "Teaching the incompressible995 navier–stokes equations to fast neural surrogate models in three dimen-996 sions," Phys. Fluids **33**, 047117 (2021). 997
- ³⁶M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural998 networks: A deep learning framework for solving forward and inverse999 problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
 - ³⁷L. Bar and N. Sochen, "Unsupervised deep learning algorithm for pde1002

based forward and inverse problems," arXiv preprint arXiv:1904.05417 (2019).

- ³⁸Z. Cai, J. Chen, M. Liu, and X. Liu, "Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic pdes," J. Comput. Phys. **420**, 109707 (2020).
- ³⁹A. Stanziola, S. R. Arridge, B. T. Cox, and B. E. Treeby, "A helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound," J. Comput. Phys. **441**, 110430 (2021).
- ⁴⁰M. Y. Michelis and R. K. Katzschmann, "Physics-constrained unsupervised learning of partial differential equations using meshes," arXiv preprint arXiv:2203.16628 (2022).
- ⁴¹Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physicsconstrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," J. Comput. Phys. **394**, 56 – 81 (2019).
- ⁴²L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," Comput. Methods Appl. Mech. Eng. **361**, 112732 (2020).
- ⁴³N. Geneva and N. Zabaras, "Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks," J. Comput. Phys. 403, 109056 (2020).
- ⁴⁴R. Grzeszczuk, D. Terzopoulos, and G. Hinton, "NeuroAnimator: fast neural network emulation and control of physics-based models," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98 (Association for Computing Machinery, New York, NY, USA, 1998) pp. 9–20.
- ⁴⁵L. Ladický, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross, "Datadriven fluid simulations using regression forests," ACM Trans. Graph. **34**, 1–9 (2015).
- ⁴⁶K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey, "Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers," Adv. Neural Inf. Process. Syst. **33**, 6111–6122 (2020).
- ⁴⁷S. Wiewel, M. Becher, and N. Thuerey, "Latent space physics: Towards learning the temporal evolution of fluid flow," in *Computer graphics forum*, Vol. 38 (Wiley Online Library, 2019) pp. 71–82.
- ⁴⁸J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating eulerian fluid simulation with convolutional networks," in *International Conference on Machine Learning* (PMLR, 2017) pp. 3424–3433.
- ⁴⁹O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowliswharan, "Cfdnet: A deep learning-based accelerator for fluid simulations," in *Proceedings of the 34th ACM international conference on supercomputing* (2020) pp. 1–12.
- ⁵⁰M. Nonnenmacher and D. S. Greenberg, "Learning Implicit PDE Integration with Linear Implicit Layers," in *The Symbiosis of Deep Learning and Differential Equations* (2021).
- ⁵¹B. List, L.-W. Chen, and N. Thuerey, "Learned turbulence modelling with differentiable fluid solvers," arXiv preprint arXiv:2202.06988 (2022).
- ⁵²T. Wu, T. Maruyama, and J. Leskovec, "Learning to accelerate partial differential equations via latent global evolution," arXiv preprint arXiv:2206.07681 (2022).
- ⁵³N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows," AIAA Journal 58, 25–36 (2020).
- ⁵⁴K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, *et al.*, "Physics-informed machine learning: case studies for weather and climate modelling," Philos. Trans. R. Soc. A **379**, 20200093 (2021).
- ⁵⁵K. O. Lye, S. Mishra, and D. Ray, "Deep learning observables in computational fluid dynamics," J. Comput. Phys. **410**, 109339 (2020).
- ⁵⁶S. Fresca and A. Manzoni, "Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition," Comput. Methods Appl. Mech. Eng. **388**, 114181 (2022).
- ⁵⁷J. Yuval, P. A. O'Gorman, and C. N. Hill, "Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision," Geophys. Res. Lett. 48, e2020GL091363 (2021).
- ⁵⁸J. Yuval and P. A. O'Gorman, "Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions," Nat. Commun. **11**, 3295 (2020).

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1003

1004

- ⁵⁹T. Cohen and M. Welling, "Group equivariant convolutional networks," inora International conference on machine learning (PMLR, 2016) pp. 2990-4074 2999
- ⁶⁰N. Tajbakhsh, L. Jeyaseelan, Q. Li, J. N. Chiang, Z. Wu, and X. Dingi076 "Embracing imperfect datasets: A review of deep learning solutions foro77 medical image segmentation," Med. Image Anal. **63**, 101693 (2020). 1078
- ⁶¹J. Han, J. Ding, N. Xue, and G.-S. Xia, "Redet: A rotation-equivariant deto79 tector for aerial object detection," in *Proceedings of the IEEE/CVF Confer*1080 ence on Computer Vision and Pattern Recognition (2021) pp. 2786–27951081
- ⁶²J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Pe¹⁰⁸² tersson, and D. Persson, "Geometric deep learning and equivariant neuralo⁸³³ networks," Artif. Intell. Rev., 1–58 (2023). 1084
- ⁶³A. Akio and R. L. Vivian, "Computational design of the basic dynamicaloss processes of the ucla general circulation model," in *General Circulation*086 *Models of the Atmosphere*, Methods in Computational Physics: Advances087 in Research and Applications, Vol. 17, edited by J. Chang (Elsevier, 1977)088 pp. 173–265. 1089
- ⁶⁴J. Crank and P. Nicolson, "A practical method for numerical evaluation ofoso solutions of partial differential equations of the heat-conduction type," inosi *Mathematical proceedings of the Cambridge philosophical society*, Vol. 43092 (Cambridge University Press, 1947) pp. 50–67. 1093
- ⁶⁵S. Turek, "A comparative study of time-stepping techniques for the incom¹⁰⁹⁴ pressible navier-stokes equations: from fully implicit non-linear schemes⁰⁹⁵ to semi-implicit projection methods," Int. J. Numer. Methods Fluids. **22**₁₀₉₆ 987–1011 (1996). 1097</sub>
- ⁶⁶P. F. Fischer, F. Hecht, and Y. Maday, "A parareal in time semi-implicitose approximation of the navier-stokes equations," Lect. Notes Comput. Sci099 Eng. **40**, 433–440 (2005). 1100
- ⁶⁷D. Forti and L. Dedè, "Semi-implicit bdf time discretization of the navier+101 stokes equations with vms-les modeling in a high performance computing102 framework," Comput. Fluids **117**, 168–182 (2015). 1103
- ⁶⁸J. O. Backhaus, "A semi-implicit scheme for the shallow water equation tions for application to shelf sea modelling," Cont. Shelf Res. 2, 243–254105 (1983).
- ⁶⁹D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,"1107 arXiv preprint arXiv:1412.6980 (2014). 1108
- ⁷⁰Z. Long, Y. Lu, and B. Dong, "Pde-net 2.0: Learning pdes from data with¹⁰⁹ a numeric-symbolic hybrid deep network," J. Comput. Phys. **399**, 108925¹¹⁰ (2019).
- ⁷¹D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyen112
 "Machine learning–accelerated computational fluid dynamics," Proc. Natlinia
 Acad. Sci. U. S. A. 118, e2101784118 (2021).
- ⁷²O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks115 for biomedical image segmentation," in *International Conference on Med*₁₁₁₆ *ical image computing and computer-assisted intervention* (Springer, 2015)117 pp. 234–241.
- ⁷³V. Casulli and R. A. Walters, "An unstructured grid, three-dimensional¹¹⁹ model based on the shallow water equations," Int. J. Numer. Methods Flu¹²⁰ ids **32**, 331–348 (2000). 1121
- ⁷⁴M. Zijlema and G. Stelling, "Efficient computation of surf zone waves us₁₁₂₂ ing the nonlinear shallow water equations with non-hydrostatic pressure," Coast. Eng. **55**, 780–790 (2008). 1124
- ⁷⁵Y. Huang, G. Gompper, and B. Sabass, "A bayesian traction force mi+125 croscopy method with automated denoising in a user-friendly software126 package," Comput. Phys. Commun. **256**, 107313 (2020). 1127
- ⁷⁶A. T. Mohan, N. Lubbers, M. Chertkov, and D. Livescu, "Embed₁₁₂₈ ding hard physical constraints in neural network coarse-graining of three₁₁₂₉ dimensional turbulence," Phys. Rev. Fluids 8, 014604 (2023). 1130
- ⁷⁷J. Brandstetter, M. Welling, and D. E. Worrall, "Lie point symmetry data131 augmentation for neural pde solvers," in *International Conference on Ma*132 *chine Learning* (PMLR, 2022) pp. 2241–2256. 1133
- ⁷⁸R. Wang, R. Walters, and R. Yu, "Incorporating symmetry into134 deep dynamics models for improved generalization," arXiv preprint135 arXiv:2002.03061 (2020).
- ⁷⁹T. S. Cohen and M. Welling, "Steerable cnns," arXiv preprint137 arXiv:1612.08498 (2016).
- ⁸⁰B. M. Smets, J. Portegies, E. J. Bekkers, and R. Duits, "Pde-based group139
 equivariant convolutional neural networks," J. Math. Imaging Vis., 1–31140
 (2022).
- ¹⁰⁷² ⁸¹D. Marcos, M. Volpi, N. Komodakis, and D. Tuia, "Rotation equivariant142

vector field networks," in *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 5048–5057.

- ⁸²J. Helwig, X. Zhang, C. Fu, J. Kurtin, S. Wojtowytsch, and S. Ji, "Group equivariant fourier neural operators for partial differential equations," in *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, Vol. 202 (JMLR.org, Honolulu, Hawaii, USA, 2023) pp. 12907–12930.
- ⁸³Y. Guan, A. Subel, A. Chattopadhyay, and P. Hassanzadeh, "Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les," Phys. D: Nonlinear Phenom., 133568 (2022).
- ⁸⁴A. Ross, Z. Li, P. Perezhogin, C. Fernandez-Granda, and L. Zanna, "Benchmarking of machine learning ocean subgrid parameterizations in an idealized model," J. Adv. Model. Earth Syst. **15**, e2022MS003258 (2023).
- ⁸⁵M. Nonnenmacher and D. S. Greenberg, "Deep emulators for differentiation, forecasting, and parametrization in earth science simulators," J. Adv. Model. Earth Syst. **13**, e2021MS002554 (2021).
- ⁸⁶B. J. Holzschun, S. Vegetti, and N. Thuerey, "Score matching via differentiable physics," (2023), arXiv:2301.10250 [cs.LG].
- ⁸⁷K. Hu, C. G. Mingham, and D. M. Causon, "Numerical simulation of wave overtopping of coastal structures using the non-linear shallow water equations," Coast. Eng. **41**, 433–465 (2000).
- ⁸⁸H. Salman, L. Kuznetsov, C. Jones, and K. Ide, "A method for assimilating lagrangian data into a shallow-water-equation ocean model," Mon. Weather Rev. **134**, 1081–1101 (2006).
- ⁸⁹M. Heniche, Y. Secretan, P. Boudreau, and M. Leclerc, "A twodimensional finite element drying-wetting shallow water model for rivers and estuaries," Adv. Water Resour. 23, 359–372 (2000).
- ⁹⁰N. Gouta and F. Maurel, "A finite volume solver for 1d shallow-water equations applied to an actual river," Int. J. Numer. Methods Fluids. **38**, 1–19 (2002).
- ⁹¹E. Hanert, D. Y. Le Roux, V. Legat, and E. Deleersnijder, "An efficient eulerian finite element method for the shallow water equations," Ocean Model. **10**, 115–136 (2005).
- ⁹²B. CUSHMAN-ROISIN, "Exact analytical solutions for elliptical vortices of the shallow-water equations," Tellus A **39**, 235–244 (1987).
- ⁹³M. Läuter, D. Handorf, and K. Dethloff, "Unsteady analytical solutions of the spherical shallow water equations," J. Comput. Phys. **210**, 535–553 (2005).
- ⁹⁴Y. Huang, C. Schell, T. B. Huber, A. N. Şimşek, N. Hersch, R. Merkel, G. Gompper, and B. Sabass, "Traction force microscopy with optimized regularization and automated bayesian parameter selection for comparing cells," Sci. Rep. 9, 1–16 (2019).
- ⁹⁵J. Nordström and M. H. Carpenter, "Boundary and interface conditions for high-order finite-difference methods applied to the euler and navier-stokes equations," J. Comput. Phys. **148**, 621–645 (1999).
- ⁹⁶P. L. Roe, "Characteristic-based schemes for the euler equations," Annu. Rev. Fluid Mech. **18**, 337–365 (1986).
- ⁹⁷P. Constantin, "On the euler equations of incompressible fluids," Society. Bull. Am. Math. **44**, 603–621 (2007).
- ⁹⁸K. W. Morton, Numerical solution of convection-diffusion problems (CRC Press, 2019).
- ⁹⁹N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um, *Physics-based Deep Learning* (WWW, 2021).
- ¹⁰⁰Z. Janjic, "Finite difference methods for the shallow water equations on various horizontal grids," in *Seminar, ECMWF*, 1983 (1984) pp. 29–101.
- ¹⁰¹S. Wiewel, B. Kim, V. C. Azevedo, B. Solenthaler, and N. Thuerey, "Latent space subdivision: stable and controllable time predictions for fluid flow," in *Computer Graphics Forum*, Vol. 39 (Wiley Online Library, 2020) pp. 15–25.
- ¹⁰²J. Yuval, P. A. O'Gorman, and C. N. Hill, "Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision," Geophys. Res. Lett. 48, e2020GL091363 (2021).
- ¹⁰³Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, and A. Uno, "Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box," Phys. Fluids 15, L21–L24 (2003).
- ¹⁰⁴C. J. Greenshields and H. G. Weller, Note on Computational Fluid Dynamics: General Principles (CFD Direct https://cfd. direct., 2022).
- ¹⁰⁵K. Atz, F. Grisoni, and G. Schneider, "Geometric deep learning on molec-

- ular representations," Nat. Mach. Intell **3**, 1023–1032 (2021).
- ¹⁰⁶P. J. Olver, *Applications of Lie groups to differential equations*, Vol. 107167
 (Springer Science & Business Media, 1993).

- ¹⁰⁷G. W. Bluman and J. D. Cole, "The general similarity solution of the heat169 equation," J. math. mech. 18, 1025–1042 (1969).
- ¹⁰⁸ J. Perez, M. Menendez, P. Camus, F. J. Mendez, and I. J. Losada, "Statis¹¹⁷¹
 ¹¹⁴⁹ tical multi-model climate projections of surface ocean waves in europe,"¹¹⁷²
 ¹¹⁵⁰ Ocean Modelling **96**, 161–170 (2015).
- ¹⁰⁹A. Fournier and W. T. Reeves, "A simple model of ocean waves," in *Pro*¹¹⁷⁴
 *ceedings of the 13th annual conference on Computer graphics and inter*¹¹⁷⁵
 active techniques (1986) pp. 75–84.
- ¹¹⁰V. A. Squire, "Of ocean waves and sea-ice revisited," Cold Reg. Sci. Tech₁₁₇₇
 nol. 49, 110–133 (2007).
- ¹¹¹L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics¹¹⁷⁹ informed neural networks for forward and inverse pde problems with noisy¹⁸⁰ data," J. Comput. Phys. **425**, 109913 (2021).
- ¹¹² R. Khodayi-Mehr and M. Zavlanos, "Varnet: Variational neural networks182
 for the solution of partial differential equations," in *Learning for Dynamics*183
 and Control (PMLR, 2020) pp. 298–307. 1184
- 113X. Zhang and K. Garikipati, "Bayesian neural networks for weak solution185
 of pdes with uncertainty quantification," arXiv preprint arXiv:2101.04879186
 (2021).
- ¹¹⁶⁵ ¹¹⁴M. Eliasof, E. Haber, and E. Treister, "Pde-gcn: novel architectures for

graph neural networks motivated by partial differential equations," Adv. Neural Inf. Process Syst. **34**, 3836–3849 (2021).

- ¹¹⁵V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *International conference on machine learning* (PMLR, 2021) pp. 9323–9332.
- ¹¹⁶Y. Huang, Y. Mabrouk, G. Gompper, and B. Sabass, "Sparse inference and active learning of stochastic differential equations from data," Sci. Rep. **12**, 21691 (2022).
- ¹¹⁷J. Magiera, D. Ray, J. S. Hesthaven, and C. Rohde, "Constraint-aware neural networks for riemann problems," J. Comput. Phys. **409**, 109345 (2020).
- ¹¹⁸S. Qi, X. Ning, G. Yang, L. Zhang, P. Long, W. Cai, and W. Li, "Review of multi-view 3d object recognition methods based on deep learning," Displays 69, 102053 (2021).
- ¹¹⁹P. Rípodas, A. Gassmann, J. Förstner, D. Majewski, M. Giorgetta, P. Korn, L. Kornblueh, H. Wan, G. Zängl, L. Bonaventura, *et al.*, "Icosahedral shallow water model (icoswm): results of shallow water test cases and sensitivity to model parameters," Geosci. Model Dev. **2**, 231–251 (2009).
- ¹²⁰ H. Weller, H. G. Weller, and A. Fournier, "Voronoi, delaunay, and blockstructured mesh refinement for solution of the shallow-water equations on the sphere," Mon. Weather Rev. **137**, 4208–4224 (2009).