JGR | Machine Learning and Computation

RESEARCH ARTICLE
10.1029/2024JH000438

Special Collection:
Advanced machine learning in solid earth geoscience

Accelerating the Discovery of Steady-States of Planetary Interior Dynamics With Machine Learning

Siddhant Agarwal[1,2], Nicola Tosi[1], Christian Hüttig[1], David S. Greenberg[2], and Ali Can Bekar[2]

[1]Institute of Planetary Research, German Aerospace Center (DLR), Berlin, Germany, [2]Model-Driven Machine Learning, Helmholtz-Zentrum Hereon, Geesthacht, Germany

Key Points:

- Achieving a statistical steady-state in complex mantle convection simulations can be computationally challenging
- Optimal initial conditions predicted by neural networks allow reaching a steady-state 2.8 times faster than standard initial conditions
- This machine learning model can be easily shared and used to accelerate mantle convection research, such as for deriving scaling laws

Correspondence to:
S. Agarwal,
siddhant.agarwal@dlr.de

Citation:
Agarwal, S., Tosi, N., Hüttig, C., Greenberg, D. S., & Bekar, A. C. (2025). Accelerating the discovery of steady-states of planetary interior dynamics with machine learning. *Journal of Geophysical Research: Machine Learning and Computation*, *2*, e2024JH000438. https://doi.org/10.1029/2024JH000438

Received 20 SEP 2024
Accepted 20 FEB 2025

Author Contributions:
Conceptualization: Siddhant Agarwal, Nicola Tosi, Christian Hüttig, David S. Greenberg, Ali Can Bekar
Data curation: Siddhant Agarwal, Nicola Tosi, Christian Hüttig
Formal analysis: Siddhant Agarwal, Christian Hüttig
Funding acquisition: Siddhant Agarwal, Nicola Tosi, David S. Greenberg
Investigation: Siddhant Agarwal, Nicola Tosi, Christian Hüttig

Abstract Simulating mantle convection often requires reaching a computationally expensive steady-state, crucial for deriving scaling laws for thermal and dynamical flow properties and benchmarking numerical solutions. The strong temperature dependence of the rheology of mantle rocks causes viscosity variations of several orders of magnitude, leading to a slow-evolving "stagnant lid" where heat conduction dominates, overlying a rapidly evolving and strongly convecting region. Time-stepping methods, while effective for fluids with constant viscosity, are hindered by the Courant criterion, which restricts the time step based on the system's maximum velocity and grid size. Consequently, achieving steady-state requires a large number of time steps due to the disparate time scales governing the stagnant and convecting regions. We present a concept for accelerating mantle convection simulations using machine learning. We generate a data set of 128 two-dimensional simulations with mixed basal and internal heating, and pressure- and temperature-dependent viscosity. We train a feedforward neural network on 97 simulations to predict steady-state temperature profiles. These can then be used to initialize numerical time-stepping methods for different simulation parameters. For an example application, the number of time steps required to reach steady-state is reduced by a factor of 2.8, compared to typically used initializations. The benefit of this method lies in requiring very few simulations to train on, providing a steady-state solution that is numerically accurate as we initialize a numerical method, and posing minimal computational overhead at inference time. We demonstrate the effectiveness of our approach and discuss its potential for advancing mantle convection research.

Plain Language Summary Simulating the dynamics of the rocky interiors of planets can be particularly demanding when seeking a statistical steady-state. Yet, such simulations play a key role in planetary research, for instance, in benchmarking numerical codes and deriving parameterizations for convective heat transfer and mixing. We show that a neural network trained on fewer than 100 simulations can accurately predict the horizontally averaged temperature profile of the final steady-state. If these profiles are used as initial conditions for new numerical simulations, the corresponding steady-state for the simulation in two dimensions and all the solution variables is attained 2.8 times faster than when using other typical initializations. By speeding up thermal convection simulations, machine learning promises to enhance our understanding of the dynamics of planetary interiors.

1. Introduction

1.1. Challenges in Achieving Steady-State in Mantle Convection Simulations

Simulating natural convection phenomena, particularly mantle dynamics, often necessitates achieving a steady-state, a condition where parameters of interest either remain constant or oscillate around a mean value. This equilibrium phase is essential for deriving scaling laws for convective heat transfer (e.g., Christensen, 1984; Ferrick & Korenaga, 2023; Grasset & Parmentier, 1998; Korenaga, 2010; Vilella & Deschamps, 2018; Solomatov, 1995, to name just a few studies spread over three decades of research), validating numerical codes (e.g., Blankenbach et al., 1989; King et al., 2010; Tosi et al., 2015), studying chaotic mixing processes (e.g., Coltice & Schmalzl, 2006; Farnetani & Samuel, 2003; Samuel & Tosi, 2012; van Keken et al., 2014), and determining characteristic flow wavelengths (e.g., Grigné et al., 2005; Höink & Lenardic, 2010; Phillips & Coltice, 2010).

In computational fluid dynamics (CFD), simulations typically evolve non-linear terms such as advection by advancing the temperature over discrete time intervals. The time-step size is constrained by the maximum velocity within the system and by the grid size as per the Courant criterion (see e.g., Ch. 6 in Ferziger et al., 2019),

Methodology: Siddhant Agarwal, Nicola Tosi, Christian Hüttig, David S. Greenberg, Ali Can Bekar
Project administration: Nicola Tosi, David S. Greenberg, Ali Can Bekar
Resources: Nicola Tosi, Christian Hüttig
Software: Siddhant Agarwal, Christian Hüttig
Supervision: Nicola Tosi, David S. Greenberg
Validation: Siddhant Agarwal, Nicola Tosi, Christian Hüttig
Visualization: Siddhant Agarwal, Nicola Tosi
Writing – original draft: Siddhant Agarwal, Nicola Tosi, Christian Hüttig
Writing – review & editing: Siddhant Agarwal, Nicola Tosi, Christian Hüttig, David S. Greenberg, Ali Can Bekar

which is crucial for maintaining numerical stability. This approach is effective for fluids with constant viscosity, where steady-state conditions are rapidly achieved. In mantle convection, however, this criterion becomes problematic due to the extensive viscosity range (often spanning 10 orders of magnitude). This variability leads to cold upper regions that remain static forming a so-called stagnant lid—a "frozen" layer where heat transfer occurs very slowly, primarily through conduction, while deeper and hotter regions exhibit vigorous convection (e.g., Moresi & Solomatov, 1995). Below this layer, as the temperature increases, convection dominates, providing a heat flux into the stagnant lid that affects its time evolution (e.g., Breuer & Moore, 2015). The coexistence of these two distinct thermal regimes implies that the temperature distribution of such a system evolves over vastly different time scales. This often requires several hundred thousand to a few million time iterations before the system reaches a steady-state. Any reduction in the number of iterations needed to reach the final state can thus offer significant computational and time savings for mantle convection studies.

1.2. Machine Learning in Fluid Dynamics

The high computational cost of running fluid dynamics simulations has made them a prime area of research in scientific machine learning (e.g., Brunton et al., 2020; Lino et al., 2023). This is not only true for surrogate modeling strategies where trained machine learning models serve to approximate the forward problem, but also for hybrid strategies, where machine learning is used to accelerate partial differential equations solvers without compromising their numerical accuracy (e.g., Alieva et al., 2023; Illarramendi et al., 2020; Kochkov et al., 2021; Tompson et al., 2017).

Likewise, machine learning remains a promising avenue for accelerating discovery in geo- and planetary physics. Recent successes include modeling of seismic wave propagation (Xiong et al., 2022), seismic fault segmentation (Wu et al., 2019), wave inversion (Rasht-Behesht et al., 2022), predictions of pairwise planetary collisions from Smooth Particle Hydrodynamics simulations (Winter et al., 2022), trace gases prediction (Azmi et al., 2023), weather prediction (e.g., Kurth et al., 2023; Lam et al., 2023), air pollution forecasting based on a foundation model (Bodnar et al., 2024), probabilistic characterization of the interior structure of planets (e.g., Baumeister & Tosi, 2023; Haldemann et al., 2023; Zhao & Ni, 2021), prediction of surface heat flow from seismic structure for modeling ice sheet dynamics (Zhang & Ritzwoller, 2024), pressure and temperature predictions for mineral combinations in the Earth's lithosphere based on experiments (Qin et al., 2024), and gravity moments modeling of Jupiter (Ziv et al., 2024). The recent publication of a few review articles on data-driven methods in geophysics and geodynamics (e.g., Bergen et al., 2019; Morra et al., 2020; Yu & Ma, 2021) underscores the field's increasing significance.

Closer yet to the topic of convection simulations, Shahnas and Pysklywec (2020) used a feedforward neural network (NN) to predict the surface heat flux and mean temperature of steady-state simulations as a function of parameters such as the Rayleigh number (*Ra*), the core-to-planet radius ratio, and the presence of melting. *Ra*, which is a key parameter in convection, quantifies the ratio of buoyancy forces due to thermal expansion that drive convection to the stabilizing effects of thermal diffusion and viscosity. Pandey and Schumacher (2020) used reservoir computing to predict statistics of two-dimensional (2D) turbulent Rayleigh-Bénard convection simulations in form of the coefficients of the Proper Orthogonal Decomposition (POD) of the fields. Heyder et al. (2022) extended this method to study the generalization properties on varying ratios of buoyancy flux at the top and bottom boundaries, as well as compared POD and convolutional autoencoders for encoding data into a latent space. Heyder et al. (2024) used generative adversarial networks to parametrize transiently growing convective boundary layer and complex non-gaussian statistics of the heat flux. Agarwal et al. (2021) demonstrated that 2D surrogate models of thermal evolution of planetary interiors could be modeled using autoencoders for dimensionality reduction and then by using Long Short-Term Memory (LSTM) networks for advancing the compressed representations of this flow in time. This was a follow-up study to Agarwal et al. (2020), where an NN was used to predict the horizontally averaged temperature profiles from thermal evolution simulations. Akbari et al. (2023) leveraged POD and LSTM networks to build reduced-order models for data assimilation in Rayleigh–Bénard convection. Ali Boroumand et al. (2024) used deep convolutional networks to estimate the Prandtl number (the ratio of momentum diffusivity to thermal diffusivity) and *Ra* from two-dimensional snapshots of convection simulations. Sen et al. (2024) used eXtreme Gradient Boosting to predict the Nusselt number (the ratio of the total heat flux to the conductive heat flux) as a function of *Ra* and aspect ratios of the computational domain for turbulent thermal convection. The ability of machine learning methods to model high-dimensional function and parameter spaces has thus triggered tremendous interest in several areas of fluid dynamics.

1.3. Using Neural Networks to Predict Statistical Steady-State

In this paper, we exploit the findings of Agarwal et al. (2020) that an NN can predict horizontally averaged temperature profiles as a function of simulation parameters. Here, we focus on predicting temperature profiles of statistically steady-state mantle convection simulations instead of thermal evolution simulations. We demonstrate that these predicted profiles serve as excellent initial conditions for a traditional numerical solver. By using the profiles predicted by our NN, we can determine the statistical steady-state of a given system in a fraction of the time than would be needed when starting from a typical initial condition such as an arbitrary high or low temperature, or a conductive profile.

While the predictions of the temperature profile from the NN are expected to be relatively accurate, feeding these to numerical solvers allows obtaining other valuable information on the system, namely, (a) two-dimensional temperature, velocity, and pressure fields, and thereby the characteristic flow pattern and wavelength, as well as (b) numerically accurate heat fluxes at the top and bottom of the domain, which can then be used to derive suitable scaling laws for convective heat transfer (see Section 3.3). Hence, we propose a hybrid modeling approach, where NNs serve as an accelerator for our specific problem of predicting the statistical steady-state of thermal convection simulations. To the best our knowledge, this is the first time such an approach has been proposed in the context of mantle convection. The data used in this study as well as a trained NN are made available on HuggingFace. The trained model can be run to generate and download temperature profiles: https://huggingface.co/spaces/agsiddhant/steadystate-mantle. No installation is needed.

The outline of the paper is as follows. We first explain the methods used, including the mantle convection equations solved and the numerical setup of the simulations (Section 2.1). We provide details on the simulation data set (Section 2.2), on the neural network used (Section 2.3) as well as on the other regression methods used as baselines (Section 2.4) and on the calculation of the stagnant lid thickness and of some basic scaling laws (Section 2.5). In Section 3.1, we examine the accuracy of the NN predictions and compare it to the other regression baselines. In Section 3.2 we compare the number of iterations needed to reach a statistical steady-state for two qualitatively different simulations that are started from different initial conditions. In Section 3.3, we derive some basic scaling laws using simulations accelerated by NN predictions as an example application. We then conclude the paper discussing the significance of this approach, possible improvements (Section 4), and summarizing our main findings (Section 5).

2. Methods

2.1. Convection Model

We solve the conservation equations of mass, linear momentum and thermal energy in a 2D Cartesian geometry for an incompressible fluid with variable viscosity and negligible inertia, heated from below and from within. In non-dimensional form, these read respectively:

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

$$-\nabla p + \nabla \cdot \left(\eta\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{T}}\right)\right) = Ra T \boldsymbol{e}_y, \tag{2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \nabla^2 T + Q, \tag{3}$$

where $\boldsymbol{u}$ is the velocity vector, p is the dynamic pressure, η is the dynamic viscosity, which depends on temperature and depth (see Equation 5), T is the temperature, $\boldsymbol{e}_y$ is the unit vector in the vertical direction, Q is the internal heating rate (assumed to be uniformly distributed in the domain), and Ra is the Rayleigh number defined as

$$Ra = \frac{\rho^2 \, c_p \, g \, \alpha \, \delta T \, D^3}{\eta_0 \, k}, \tag{4}$$

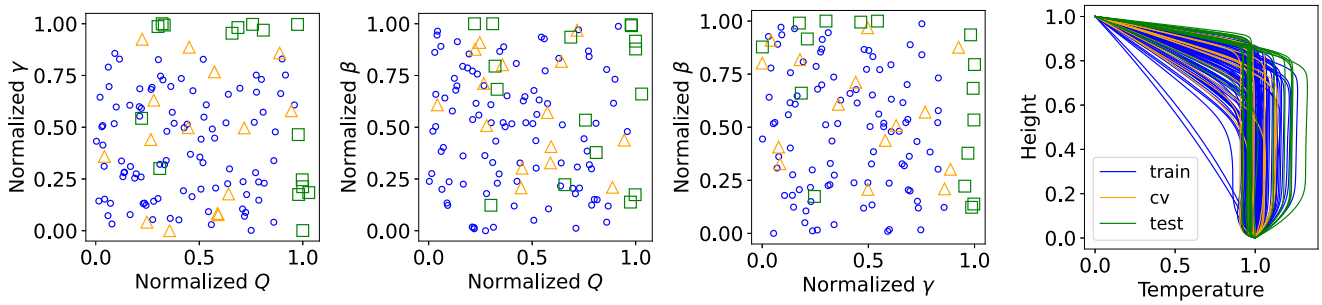

Figure 1. A visualization of the normalized simulation parameters and temperature profiles for training (blue), validation (cv, orange) and test (green) sets. We scale the parameters by their respective minimum and maximum values. For γ and β, we first take the $\log_{10}$ of the parameters and then scale them by their respective minimum and maximum values. The test set (green symbols and lines) is deliberately picked in this manner to check for NNs ability to extrapolate slightly.

where ρ is the density, c_p is the heat capacity, g is the gravitational acceleration, α is the coefficient of thermal expansion, δT is the temperature scale, D is the domain thickness, k is the thermal conductivity, and η_0 is a reference viscosity.

Due to the neglect of inertia (Equation 2), that is, under the assumption of Stokes flow, velocity and pressure fields are completely determined by the temperature distribution. In other words, for a given temperature field, and hence a given right hand side of Equation 2, one can obtain the velocity fields with a single solve of the Stokes and mass conservation equations, which, in the current setup, takes approximately 3 s. This means that it is sufficient for a neural network to learn and predict the temperature for initializing a simulation.

All quantities appearing in Equation 4 are constant (i.e. their non-dimensional reference value is 1). The (non-dimensional) viscosity in Equation 2 depends on temperature and depth according to the following rheological law:

$$\eta(T,y) = \exp(-\log(\gamma)T + \log(\beta)(1-y)), \tag{5}$$

where y is the height (0 at the bottom of the domain and 1 at the top), and γ and β are the maximum viscosity contrasts due to temperature and depth, respectively. The Rayleigh number Equation 4 is set equal to 1. By using large values of γ in Equation 5, we obtain an exponential decrease of viscosity with temperature, which can be partly compensated by an increase with depth controlled by the β term. This leads to a highly viscous, non-convecting stagnant lid occupying the upper part of the domain, which is subcritical for convection. Below the stagnant lid, the decrease in viscosity due to temperature ensures that the local Rayleigh number (i.e., Ra/η) is highly supercritical and convection is vigorous. Assuming for example, $\gamma = 10^6$ and $\beta = 1$, at the bottom of the domain we have $\eta(T = 1, y = 0) = 10^{-6}$ and a local Rayleigh number of $Ra/\eta = 10^6$.

We use the finite volume code GAIA (Hüttig et al., 2013) to solve Equations 1–3 in a 2D rectangular domain with an aspect ratio of 4. We use a regular grid consisting of 126 and 504 grid points, uniformly spaced in the vertical and horizontal directions, respectively. We treat all boundaries as impermeable, that is, with zero normal velocity, and free-slip, that is, with zero shear stress. The top and bottom boundaries are isothermal (with $T(y = 0) = 1$ and $T(y = 1) = 0$) and the sidewalls insulating, that is, with zero heat flux.

2.2. Parameters, Data Set, and Output Quantities

We generate a data set of 128 simulations of steady or statistically steady convection depending on the three parameters γ, β and Q (Agarwal et al., 2024). We vary γ between 10^6 and 10^{10}, β between 1 and 100, and Q between 0 and 10. This choice results in (maximum) local Rayleigh numbers ranging from 10^4 (for $\gamma = 10^6$ and $\beta = 100$) to 10^{10} (for $\gamma = 10^{10}$ and $\beta = 1$).

Due to the choice of rectangular geometry, the combination of high internal heating, depth dependence of the viscosity, and fixed-temperature bottom boundary conditions can lead to internal temperatures exceeding the bottom temperature, which is set to the non-dimensional value of 1 (Figure 1, last column). We note that this effect, albeit physically realistic, would be less strong upon using a spherical shell geometry. In fact, plane-layer

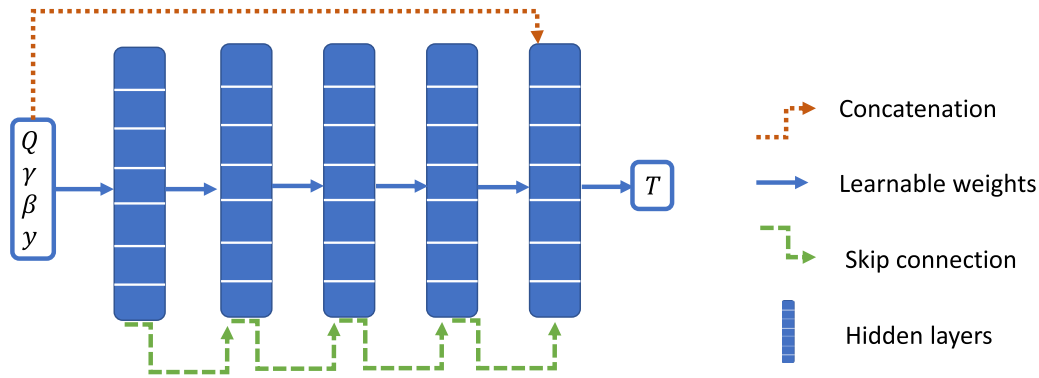

Figure 2. We use a feedforward neural network with five hidden layers and 128 hidden units per layer to predict the temperature at a given height y as a function of the simulation parameters: internal heating (Q), viscosity contrast due to temperature (γ), and viscosity contrast due to pressure (β). The network has skip connections to regularize the loss landscape. We also condition the last hidden layer with the input vector by concatenation, as this tends to improve the accuracy of the predictions. Although not shown in the figure, we apply *SELU*() activation (Klambauer et al., 2017) to each hidden layer, that is, each layer except the input and the output.

convection models may need to introduce artificial cooling in order to reproduce temperatures obtained in spherical shells (O'Farrell et al., 2013; O'Farrell & Lowman, 2010). Also, if one were to simulate the full thermal evolution of a planetary interior (i.e., transient convecting cooling rather than a statistical steady state), the bottom temperature (i.e., the temperature of the top of the core) would adapt according to the local heat flux, at least as long as the core is liquid and can be approximated as a convecting and homogeneous body of given density and heat capacity (e.g., Stevenson et al., 1983). In this situation, the core would tend to heat up, thereby reducing the difference between its temperature and the—possibly higher—internal mantle temperature. At any rate, these cases provide useful tests both because they can be representative of transient phases of mantle and core heating, and because they allow us to probe the ability of our ML predictions to reproduce non-trivial temperature profiles.

We split the data set of 128 simulations as follows: 97 simulations are used for training the network, 15 are used as validation (cv) to test different architectures, and 16 are kept as the test set. While training and cv simulations are chosen randomly from the same uniform distribution of the parameters, we intentionally chose our test set so that at least one of the simulation parameters is slightly outside the range of the training and cv data sets: if $Q > 9.5$ or if $\gamma > 5 \times 10^9$ or if $\beta > 95$. This allows us to test the ability of the network to extrapolate, albeit slightly given the limited amount of simulations at hand. The simulation parameters as well as the temperature profiles are visualized in Figure 1. As somewhat expected, the temperature profiles of some of the simulations exceed the internal temperature and the heat fluxes of the profiles of the training and cv sets.

To facilitate optimization of the NNs and other regression algorithms, we scale the parameters by their respective minimum and maximum values. For γ and β, we first take the $\log_{10}$ of the parameters before scaling them.

2.3. Neural Network Model

We use a typical feedforward neural network, such as the one used by Agarwal et al. (2020) to predict the 1D temperature profiles from thermal convection evolution simulations of a Mars-like planet. In that study, the network predicted the complete profile as a function of six parameters: time, reference viscosity of the mantle, activation energy and activation volume of diffusion creep, enrichment factor of heat-producing elements in the crust, and initial temperature of the mantle. For our purposes here, though, we introduce some changes (see Figure 2 and Agarwal et al. (2025) for a PyTorch implementation):

- Unlike the NN in Agarwal et al. (2020), the network output is a scalar prediction of temperature at a given height ($T(y)$). Such coordinate-based networks are heavily used for more memory-efficient learning on unstructured representations such as point clouds and meshes (e.g., Hines Chaves et al., 2023; Park et al., 2019). This also helps avoid wiggles in the prediction, which we see when the profile is high-dimensional (more than ≈ 60 points, as empirically observed in Agarwal et al. (2020)). This also allows the end-user to directly train and evaluate on arbitrary depth profiles that are no longer tied to the underlying discretization.

- We train on mini-batches of only 32 points. Therefore, we find it beneficial to sample comparatively more points in the upper and lower thermal boundary layers of the temperature profile as these would otherwise be underrepresented in a batch as most of the mantle tends to be isothermal due to convective heat transfer. We do so by taking 100 evenly sampled points between the top and the 15th point from the top. We then interpolate the temperature profile to these new points. We also take 50 evenly sampled points between the bottom and the 10th point from bottom and interpolate the temperature profile to these points. In the middle of the domain, we take the temperature at the points which retain their original spacing. This leads to a total of 12,416 points in the training set, 1,920 points in the validation set and 2,048 points in the test set.
- Skip connections are introduced from each hidden layer to every following hidden layer by adding these two layers element-wise before applying the activation function $SELU()$ (Klambauer et al., 2017). These additional connections act as additional pathways for the gradient information to flow during backpropagation and help mitigate vanishing gradients in the earlier layers of the network.
- We inject the input to the NN into the last hidden layer of the network by concatenation, as this has been shown to be useful for improving accuracy (e.g., Park et al., 2019).
- Finally, we apply a correction at the boundaries of the domain. We find it important to provide profiles that satisfy exactly the temperature boundary conditions to be readily used by any traditional numerical solver without the need to apply any further correction. Imperfect boundary conditions could violate the temperature scale, making δT smaller or greater than 1. As NNs are approximators optimized by an iterative process, their predictions naturally tend to be inexact and have a hard time satisfying boundary conditions. In other words, the boundary conditions tend to be no more or less accurate than any other point in the domain. In the future, it could be worth exploring some non-trivial approaches for exact enforcing of the boundary conditions (e.g., Sukumar & Srivastava, 2022; Wang et al., 2023), but here we simply overwrite the NN predictions at the top with 0 and at the bottom with 1. However, this also means that the gradient of the boundary layer can become inconsistent, especially if the grid resolution is changed. To ensure that at least the heat fluxes are consistent with respect to the discretization, we overwrite the points within a certain distance of the top (0.04) and bottom (0.985) with the same gradient between the first and last point.

We train the NNs in PyTorch (Ansel et al., 2024) using an Adam optimizer (Kingma & Ba, 2014). To keep overfitting in check, especially for such a small data set, we use a batch size of 32, a weight decay value of 0.0005 with the Adam optimizer and save the weights of the NN after an epoch only if the loss on the cv set has decreased. We train for 140 epochs and decrease the initial learning rate of 0.001 by a factor of 0.5 every 20 epochs. On a Tesla V100 GPU, training takes up to a couple of minutes. In contrast to Agarwal et al. (2020), the combination of small mini-batches, $SELU()$ activation function instead of $\tanh()$, pointwise formulation (using y as an input), and a small number of simulation profiles (128) compared to 10,000 simulations $\times 100$ time-steps makes for a significantly faster training time. We tested NN architectures by varying the number of hidden layers from 2 to 6 and the number of hidden units per layer between 32 and 256, and found 4 or 5 hidden layers with 128 or 256 units to perform the best. Here, we present the results of a NN trained with 128 units with five hidden layers as this achieved the lowest error on the cv set.

2.4. Regression Baselines

For a fair comparison, we include the following baseline algorithms for predicting the temperature profile using scikit-learn (Pedregosa et al., 2011):

- Linear regression: we test linear regression to predict the temperature profiles as a function of the normalized simulation parameters.
- Kernel ridge regression: it is a standard workhorse for regression problems and has the advantage of being able to approximate non-linear functions while being robust to outliers due to regularization. We use a radial basis function kernel with $\alpha = 0.1$, which controls the regularization strength, and keep the default value of the multiplier of the radial basis function.
- Nearest neighbor: as a simple baseline, we checked how far off the temperature profile would be for a set of parameters if we simply picked the profile at the nearest neighbor in terms of the simulation parameters in our data set. The nearest neighbor was determined using KDTree, which uses the Minkowski distance in an n-dimensional parameter space.
- Nearest neighbor interpolation: as a slightly more advanced version of nearest neighbor, we also baseline by taking (a) the temperature profiles of the three nearest neighbors in terms of simulation parameters and (b) by

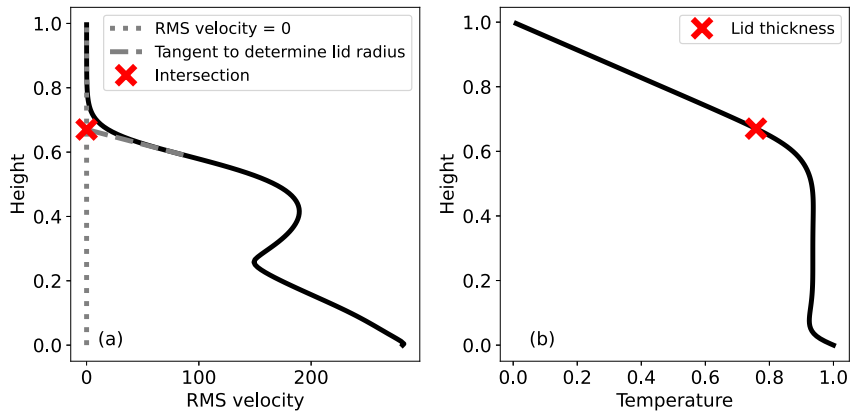

Figure 3. The "tangent method" is used for determining the thickness of the lid. (a) A tangent is drawn from the point of maximum gradient of the horizontally averaged profile of the RMS velocity to where it equals zero. This is taken as the lid thickness from top or the height of the lid in terms of the y-coordinate. (b) The temperature contrast between this point and the maximum temperature (i.e., $T = 1$ at the bottom of the domain in this example) is used to determine an effective Rayleigh number of the system with respect to which scaling laws are derived.

using inverse distance weighting to sum the three profiles resulting in an interpolated result. If the set of parameters being evaluated matches a set of parameters in the "training" data set, we do not perform the averaging and simply take the exact output from the data set.

2.5. Derivation of Basic Scaling Laws for Convection

As an example of downstream applications of the kind of simulations we aim to accelerate in this study, we derive some basic scaling laws for mixed heating with temperature- and pressure-dependent viscosity. There are three main steps for deriving the scaling laws.

First, we determine the thickness of the stagnant lid using the "tangent method" (e.g., Reese et al., 1999) as shown in Figure 3. The y-intercept of the tangent from the point of maximum gradient of the horizontally averaged root mean squared (RMS) velocity profile is taken as the height of the lid.

Second, the difference between the temperature at the base of this lid and the maximum temperature is used to calculate an effective Rayleigh number (Ra_{eff}):

$$Ra_{\text{eff}} = Ra\,\frac{y_{\text{lid}}^3 (T_{\text{max}} - T_{\text{lid}})}{\eta_{\text{eff}}}, \tag{6}$$

where, Ra is the Rayleigh number defined in Equation 4 (in this case simply $Ra = 1$), y_{lid} is the height of the lid (i.e., the distance in y from the base of the domain), T_{max} is the maximum temperature of the profile, T_{lid} is the temperature at the base of the lid, and η_{eff} is calculated as the harmonic mean of the viscosity below the lid.

Third, the simulations are used to derive a Nusselt-Rayleigh scaling. As this is not the primary goal of this study, we avoid a comprehensive survey of and comparison to the vast literature on scaling laws and instead use a basic Nusselt-Rayleigh scaling of the form proposed by Moore (2008) that considers an additive term for internal heat production:

$$Nu = a\,Ra_{\text{eff}}^{b} + c\,Q, \tag{7}$$

where Nu is the Nusselt number at either the top or the bottom of the domain, and a, b and c are coefficients that are determined from simulations accelerated by the NN predictions with the help of the curve fitting function in Scipy (Virtanen et al., 2020), which accepts arbitrary expressions such as the power law function defined in Equation 7.

Table 1
Mean Absolute Error (MAE) of All the Prediction Algorithms on the Train, Validation (cv) and Test Sets

Prediction algorithm	MAE train	MAE cv	MAE test
Linear regression	0.0385	0.0388	0.0676
Kernel ridge regression	0.0148	0.0147	0.0371
Nearest neighbor	*0.0000*	0.0282	0.0495
Interpolation	*0.0000*	0.0230	0.0448
Neural network	0.0048	*0.0053*	*0.0133*

Note. In the test set, at least one of the three simulation parameters is extrapolated. For completeness, the MAE for the training data set is also presented for nearest neighbor and nearest neighbors interpolation although it is obvious that when the simulation with the same parameters in the data set is present, the error will be exactly zero. The lowest error in each subset of the data set is italicized.

3. Results and Discussion

3.1. Neural Network Predictions Versus Regression Baselines

In this section, we present the results of predictions from an NN (see Section 2.3 for details) with five hidden layers containing 128 units each and compare these to the predictions of the regression baselines outlined in Section 2.4. Table 1 shows that the NN achieves the lowest mean absolute error (MAE) of all the algorithms on the cv and test sets. Of course, in the case of nearest neighbor and nearest neighbors interpolation, the predictions on the training set are redundant and the MAE is exactly zero. The NN seems to generalize not only for parameters within the bounds of the training data set (cv), but also generalizes better than the other algorithms when extrapolating slightly (test).

Depending on the study at hand, if the prediction error in this extrapolated range is too high, one could address it by extending the range of the training data set. Our NN architecture is able to fit on the test data set—we present a comparison of the loss curves when trained on the training set versus when trained on the training plus the test set together in Figure A1 in Appendix A. In the latter case, the NN now achieves a training MAE of 0.00437 and an MAE of 0.0037 on the profiles from the test set alone, which is significantly better than 0.0133. Nevertheless, with the exception of this one experiment, all the results are presented with the NN that was trained on the original training set only.

We plot some sample predictions of the NN and of the regression baselines against the ground truth in Figure 4 for the cv set and the test set. In both cases we leave out the nearest neighbor prediction and take the interpolated profile based on the nearest neighbors instead, which is based on the three closest neighbors and is therefore slightly more accurate. Visually, we can observe that the NN predictions seem to match the ground truth most consistently. Linear regression and kernel ridge regression sometimes introduce larger wiggles at the base of the lid. The nearest neighbors interpolation produces visually reasonable results, but ultimately suffers from low accuracy. We also note that, while we use a coordinate-based approach for the NN, we directly predict the complete profile as a function of three simulation parameters for all the other regression algorithms. These delivered lower MAEs than the coordinate-based approach: 0.0388 versus 0.1576 for linear regression, 0.0147 versus 0.0449 for kernel ridge regression and 0.0230 versus 0.0285 for interpolation. For nearest neighbors, the results were identical. Therefore, we use a coordinate-based approach for the NN and full-profile approach for the other regression algorithms in all our analyses.

We further use the temperature profiles to calculate the top and bottom Nusselt numbers, that is, the non-dimensional heat fluxes computed at the upper and lower domain boundary. We note here that we simply predict the Nusselt numbers as a function of the three simulation parameters using the NN. This differs from deriving a scaling law as a function of an effective Rayleigh number (Equation 7), which would require estimating the stagnant lid thickness from the a velocity field. However, as mentioned earlier, obtaining the velocity field is also not problematic, as it is readily available from a single Stokes solver computation. As shown in Figure 5, even though the NN predictions are not accurate enough to derive scaling laws, they outperform the baselines. The NN accuracy diminishes when extrapolating, especially at the bottom. This is because the thermal gradients at the bottom are steeper than in the top thermal boundary layer and are therefore more sensitive to slightly perturbed (inaccurate) predictions.

3.2. Accelerating Convergence of Numerical Simulations

We demonstrate the impact of different initial conditions on the time it takes to reach a statistical steady-state. For this, we consider two qualitatively different simulations by varying the three parameters Q, γ and β as follows:

- Case 1 (weak convection): $Q = 5, \gamma = 10^8, \beta = 50$
- Case 2 (strong convection): $Q = 7.5, \gamma = 10^9, \beta = 25$

With respect to Case 1, Case 2 has a higher internal heating rate, a stronger temperature dependence of the viscosity, and a weaker pressure dependence, all of which strengthen the vigor of convection. For each of these

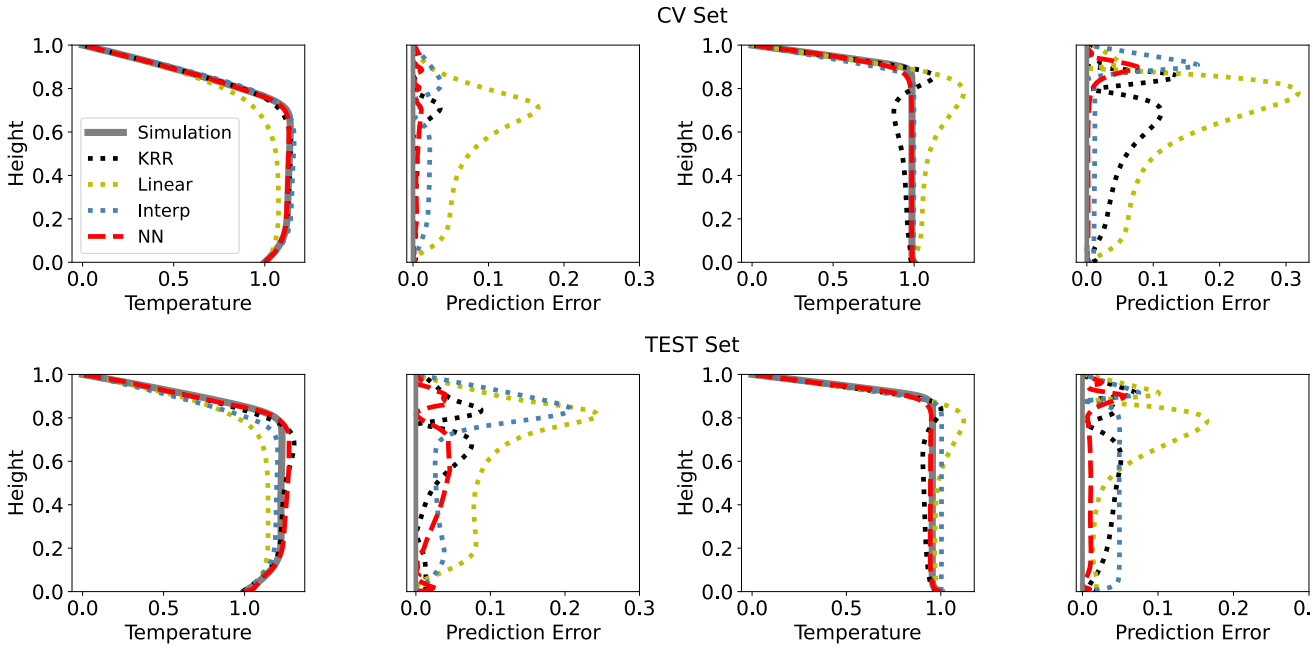

Figure 4. Comparison of some neural network predictions and those of the baseline algorithms against ground truth temperature profiles for the validation set and the test set. For each set of predictions on the left, the prediction error is plotted to their right.

two simulations, we then start from the following initial temperature profiles: (a) hot, with $T = 1$ everywhere except at the top boundary; (b) cold, with $T = 0$ everywhere (except at the bottom boundary); (c) linear, with T increasing linearly from the top, where $T = 0$, to the bottom, where $T = 1$, as in a conductive profile; (d) exact temperature profile, corresponding to the last time step of a finished numerical simulation; and (e) NN prediction. For the two convection cases, we show the restart with the exact steady-state temperature profile to get a sense of how the simulation would evolve if we used the exact steady-state temperature profile from a finished simulation.

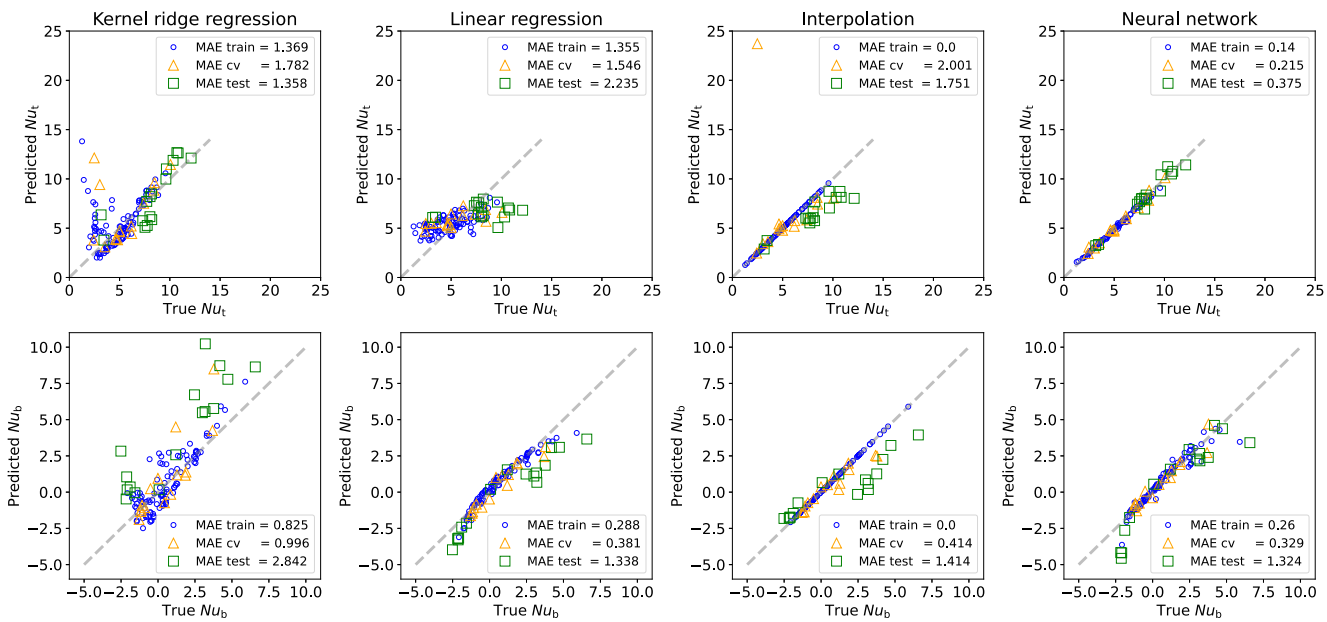

Figure 5. Comparison of neural network predictions and those of the baseline algorithms against the true non-dimensional heat flux at the top Nu_t (top row) and at the bottom Nu_b (bottom row).

10.1029/2024JH000438

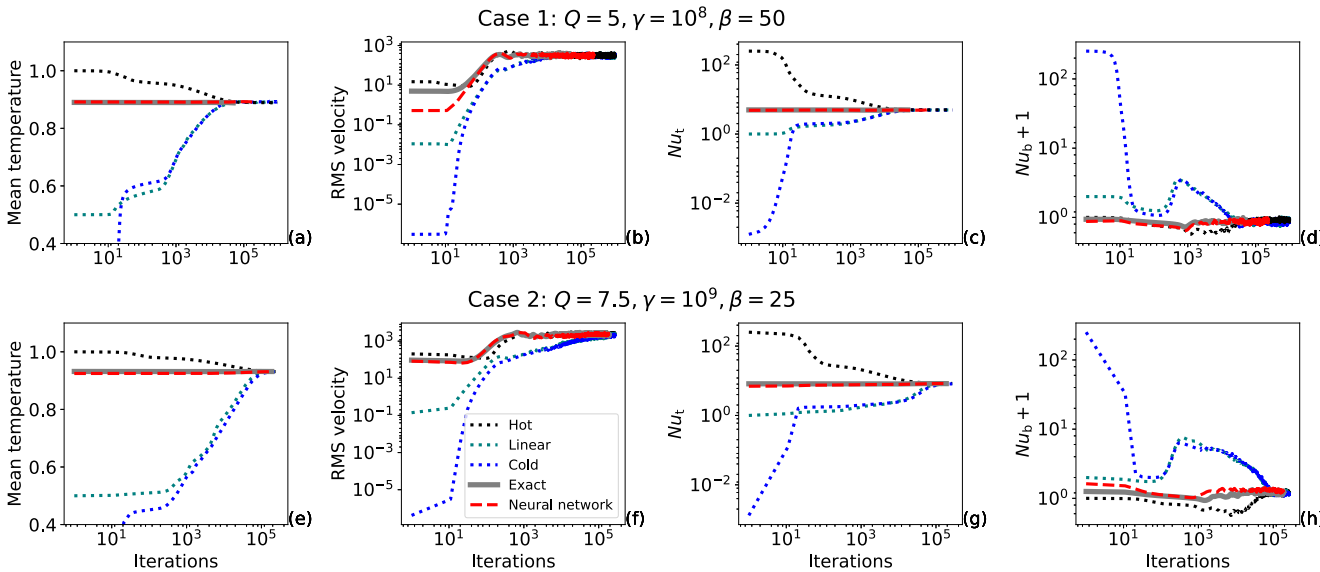

Figure 6. Time-series of mean temperature, velocity, top and bottom Nusselt number for two different simulations initialized with different temperature profiles: hot, cold, linear, exact, that is, the exact profile from a finished simulation for reference, and NN prediction. To enhance the visualization, we use a $\log_{10}$-$\log_{10}$ scale for RMS velocity and the Nusselt numbers and therefore add one to the bottom Nusselt number to ensure that the $\log_{10}$ is defined.

Figure 6 shows the mean temperature, mean velocity, Nusselt number at the top and Nusselt number at the bottom for all the simulations and initializations as a function of the number of time steps of the numerical solver (indicated as Iterations in the figure). For the two cases, the NN-initialized simulation (dashed red line) follows the simulation with the exact temperature profile initialization (solid gray line) very closely. The other initializations are naturally very far off at the beginning of the simulation and typically take on the order of 10^4 iterations to reach a statistical steady-state. In Table 2, we list the number of solver iterations needed to reach within one percent of the final mean temperature, RMS velocity and heat fluxes at top and bottom averaged over the last 10% of total iterations. For reference, we also provide the iterations needed for the restart with exact steady-state temperature profile. This serves as an upper bound of the speedup that can be achieved.

Table 2
Iterations Needed for Mean Temperature, RMS Velocity and Top and Bottom Heat Fluxes to Reach Within 1% of the Final Value Averaged Over the Last 10% of Total Iterations

Initialization	Case 1	Case 2	Sum
Hot	47,948	115,950	163,898
Cold	45,909	212,906	258,815
Linear	101,802	194,898	296,700
Linear regression	11,175	137,930	149,105
Kernel ridge regression	26,013	90,080	116,093
Interpolation	17,469	87,828	105,297
Nearest neighbor	35,974	65,450	101,424
Neural network	17,624	68,058	85,682
Exact	5,593	21,368	26,961

Note. In most cases, initializing the simulations with the profiles predicted by the NN results in the minimum number of iterations needed to reach a statistical steady state. For reference, we also provide the iterations needed for the restart with the exact steady-state temperature profile as an upper bound of the speedup that can be expected.

We further examine the acceleration of the simulations when initialized with the temperature profiles predicted by the baseline regression algorithms. We list the corresponding number of iterations in Table 2 and plot the time-series in Figure 7. All the prediction methods provide a more suitable initial condition than the hot start. In fact, for Case 1, interpolation and linear regression require fewer iterations than the neural network. It is interesting that despite being less accurate than the NN, other regression algorithms can still deliver some speedups in the two study cases.

Nevertheless, in order to gather more comprehensive statistics, it would be desirable to carry out a large number of simulation tests across the whole parameter space covered by the training set. It is difficult to establish how errors in the initial temperature profile impact the speedup of a non-linear dynamic system. So, one might advocate for using the most accurate method, as it is most likely to deliver the highest speedups. That would be the NN, as it achieved the lowest prediction error on the profiles (see Table 1).

Furthermore, deciding when a simulation has reached a statistical steady-state is not always straightforward. For example, if the goal is to derive a type of Nusselt-Rayleigh scaling law, then the stability of the mean quantities such as the Nusselt numbers plays a key role. However, if one wishes to retrieve the two-dimensional temperature field, it might be also important to consider if this field has stabilized or not. Figure 8 shows some snapshots for Cases 1 and

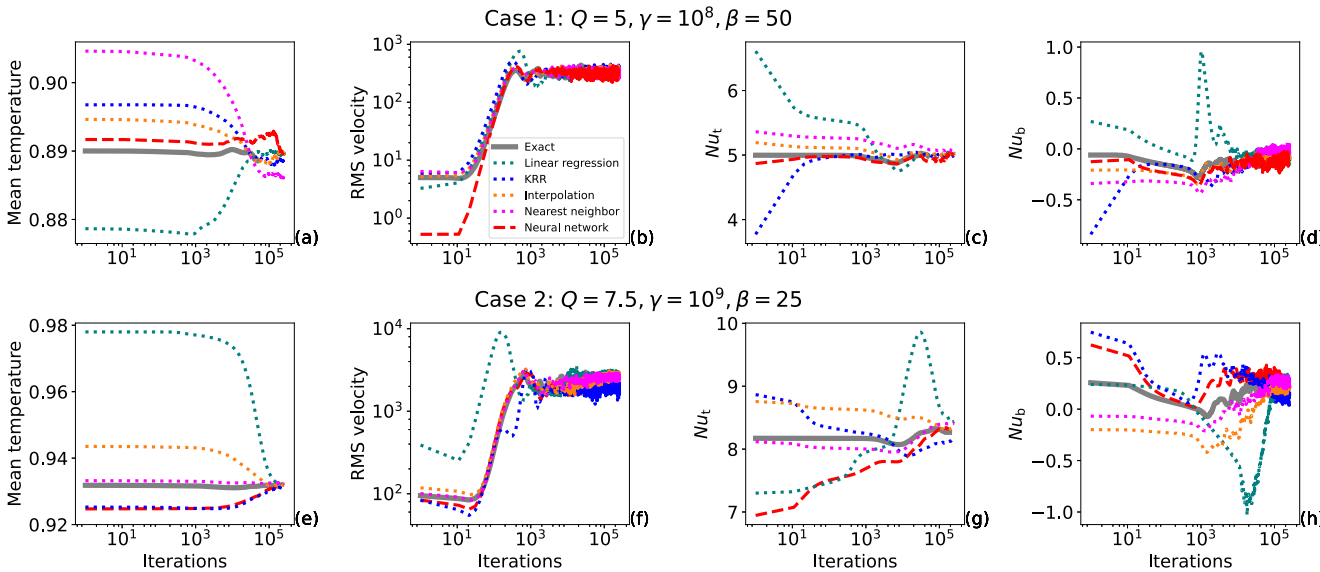

Figure 7. Time-series as in Figure 6, but for simulations initialized with profiles from the various baseline prediction algorithms: linear regression, KRR—kernel ridge regression, interpolation, nearest neighbor, in addition to the exact steady-state profile restart and neural network initialization.

2, when started from the NN initialization: 1,000 iterations after the restart (a, d), at the iteration where the statistical steady-state is reached (b, e), and at the final iteration available (c, f). Even though the top Nusselt number stabilizes for the first simulation (Figure 6c) almost immediately at the start, the horizontal distribution of the stagnant lid takes longer to develop (Figures 8a–8c). Similarly, owing to a stronger vigor of convection, case 2 maintains a relatively thin lid at iteration 1,000, but takes longer to develop the convection patterns with the longer wavelength features such as the lid stabilizing, and only the smaller higher-frequency plumes and downwellings fluctuating. Hence, deciding when steady-state has been reached is a decision best left for the informed user to make based on the purpose of the study.

3.3. Scaling Laws for Convection With Variable Viscosity and Internal Heating

As an example of an application of simulations accelerated by NN predictions, we derive some basic scaling laws for three cases, where we vary the maximum viscosity contrast due to temperature (γ), and thereby the effective Rayleigh number:

1. No internal heating and no pressure dependence of viscosity: $Q = 0$ and $\beta = 1$
2. Internal heating and no pressure dependence of viscosity: $Q = 2$ and $\beta = 1$
3. Internal heating and pressure dependence of viscosity: $Q = 1$ and $\beta = 10$

To generate the data needed to fit the coefficients of Equation 7, we run eight simulations for each of the cases listed above and vary γ between 10^6 and 10^9 in $\log_{10}$ space. This is achieved with the `logspace()` function in numpy and is equivalent to taking `linspace()` between 6 and 9 and raising it to the power of 10. We run these simulations starting from the temperature profile predicted by the trained NN. For $Q = 1$, $\beta = 10$ and $\gamma = 10^6$, the simulation was purely conductive and hence discarded, leading to a total of 23 simulations (eight for the first two cases and seven for the third). While one could attempt to use the heat fluxes predicted by the NN directly, we found these to not be sufficiently accurate. Hence, for now, to obtain reliable results, we recommend initializing a numerical simulation with the NN prediction and letting the system reach the steady-state.

Using these simulations, we derive scaling laws for each of the three scenarios described above. We then use these scaling laws to predict the Nusselt numbers at the top (Nu_t) and bottom (Nu_b), and plot them against the true values from the simulations in Figure 9. Despite some scatter, the scaling laws are able to predict the Nusselt numbers at top and bottom well. Our approach thus paves the way to efficiently generate large data sets of multi-parameter simulations that can be used to predict generalized scaling laws for complex flows with variable viscosity and heating mode, relevant for planetary interior evolution modeling.

10.1029/2024JH000438

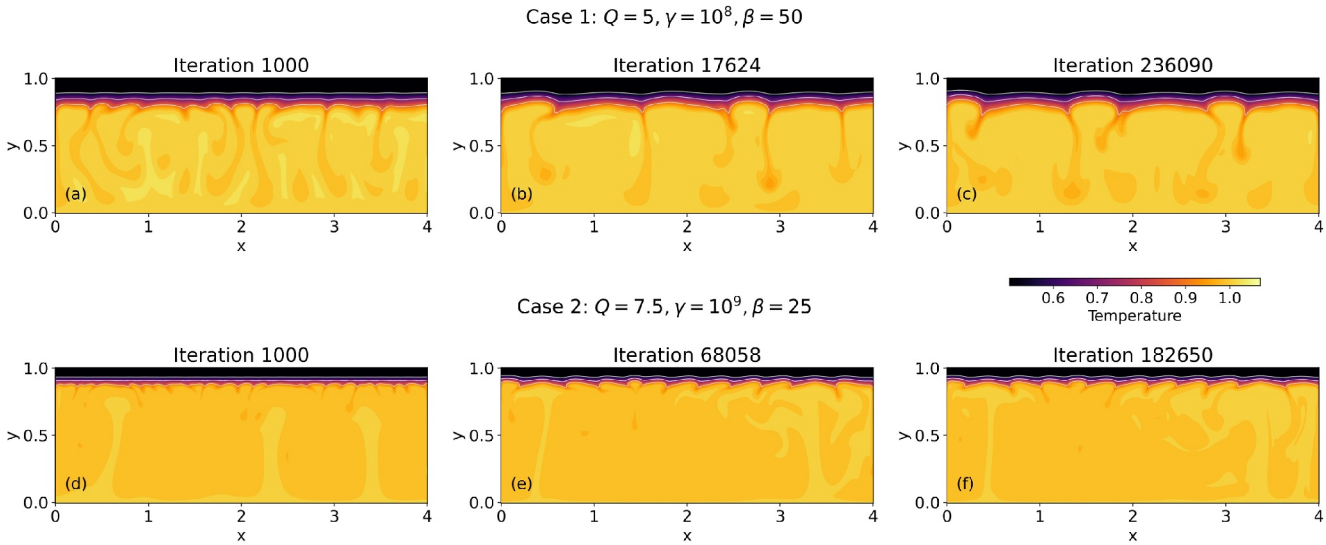

Figure 8. Snapshots of the temperature fields for the weak (a–c) and strong convection (d–f) cases at (a, d) 1000 iterations, (b, e) the iteration identified as the statistical steady-state from the mean quantities (see Figure 6 and Table 2), and (c, f) the last iteration available for the simulation.

To enrich the investigation in Section 3.2 on the speedup offered by our NN, we run the same set of 23 simulations with the so-called hot start as well as the interpolation algorithm as a simple baseline and compare the three initializations in terms of the iterations needed to reach within 1% of the final value of the mean temperature, RMS velocity and Nusselt numbers at the top and bottom. Figure 10 shows the speedup defined as the number of

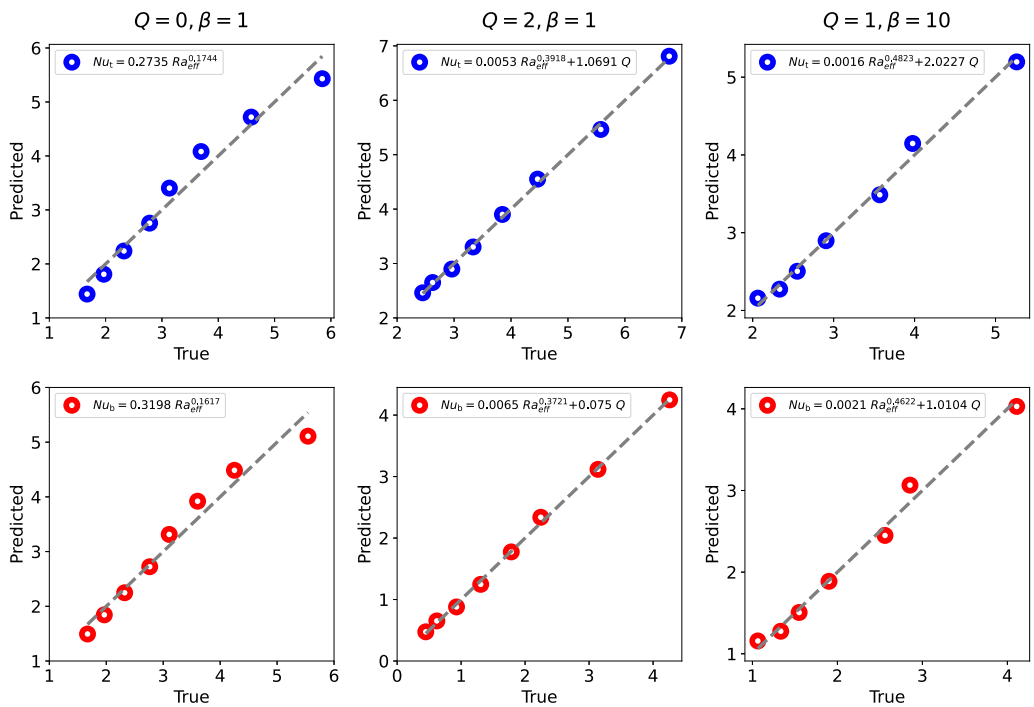

Figure 9. As an example of typical applications of such steady-state simulations, we derive basic scaling laws for three cases: no internal heating and no pressure dependence of the viscosity (left column), internal heating with $Q = 2$ and no pressure dependence (middle column), and internal heating with $Q = 1$ and pressure-dependent viscosity with $\beta = 10$ (right column). For the case $Q = 1$, $\beta = 10$, only seven data points are reported because the simulation with $\gamma = 10^6$ was purely conductive and hence was not included in the fit. The derived scaling laws are used to predict the Nusselt numbers at the top and the bottom and plotted against the true values from the simulations.

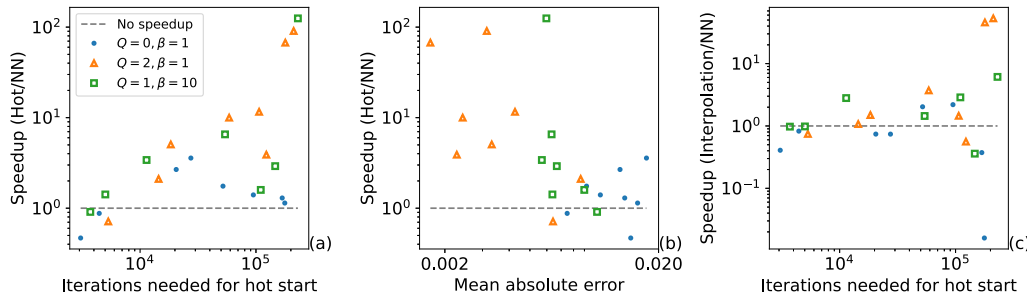

Figure 10. For the simulations used to derive scaling laws, three types of initializations are used: NN, interpolation, and hot start. The number of iterations needed by the simulations with hot starts is divided by those needed by those initialized with the NN predictions to arrive at the speedup factor. The speedup factor is plotted as a function of (a) the iterations needed by the hot starts and (b) the mean absolute error between the initial condition, that is, the NN prediction and the final temperature profile of the statistically steady-state system (obtained by averaging the temperature profiles over the last 1% of the iterations). Despite the scatter, there is a negative correlation with respect to the prediction error in the profiles, suggesting that the more accurate the initial condition of the system is, the faster it converges. (c) The speedup of the NN with respect to the interpolation is plotted against the iterations needed for the hot start. The NN is faster than the hot start by a factor of 2.8 and than the interpolation algorithm by a factor of 1.4.

iterations needed by the simulations with hot or interpolation start divided by the number of iterations needed by the NN-initialized simulations. With respect to Q, γ and β, no clear pattern is evident in the speedup factor, which is scattered across a few orders of magnitude: from being even slower (0.5 times) to being significantly faster (124 times). However, in Figure 10a, we observe a positive correlation between the speedup factor and the number of iterations needed by the hot start to reach the steady-state, indicating that the more computationally challenging simulations often benefit more from this method as the iterations needed by the NN start do not grow as significantly and seem less sensitive. In Figure 10b, we observe a negative correlation between the speedup factor and the mean absolute error between the final temperature profile of the system at statistical steady-state and the starting profile, making the case for using the most accurate regression algorithm for initializing the simulation.

In Figure 10c, we plot the iterations needed for the interpolation initialization divided by those of the NN. We summed the iterations needed to reach steady-state across the entire ensemble of simulations and found that the NN accelerates the attainment of a statistical steady-state in 83% of the cases. Including simulations without any observed speedup, the NN needs 2.8 times fewer iterations than the hot start and 1.4 times fewer than the interpolation start. This modest speedup of the NN compared to the interpolation algorithm begs the question whether an NN is worth using. For the following reasons, we believe that it is:

1. For a typical simulation in this setup that takes 5 days to run, an NN-initialized ensemble would finish 0.7 days earlier than an interpolation- one.
2. The trained NN provided on Hugging Face is extremely fast at inference time and requires no installation.
3. Most importantly, this ensemble is not representative of the full data set. It is generated by varying one parameter at a time and thereby, takes only three "lines" through a three-dimensional parameter space. In fact, we see in Figure 10c, that the NN often fails to offer a speedup over the interpolation algorithm for the case where $Q = 0, \beta = 1$. From the normalized $Q - \beta$ plot in Figure 1, we observe that the corresponding bottom left corner of the parameter space (normalized $Q = 0$ and normalized $\beta = 0$) has no data points, which makes the NN susceptible to increased prediction errors (Figure 10b) because it is extrapolating. In the future, for the derivation of scaling laws with respect to all three parameters, it would be interesting to see how the NN-initialized simulations perform in the larger parameter space.

It is also important to consider the upfront cost of generating the data set required to train the different regression algorithms in the scenario where simulations are not already available or they cannot be used because their physical/numerical setup is too different from each other. In our case, where we generated a data set of 112 simulations for training and validation and achieved a speedup of approximately 2.8 over the traditional initializations, it would only be worth using our approach for an application that required more than $112/2.8 + 112 = 152$ simulations. If more research groups open-source their data sets and trained models, this could help mitigate the upfront cost of data set generation.

4. Discussion

While sophisticated machine learning methods have demonstrated remarkable capabilities in approximating highly non-linear forward maps, they often tend to be data-intensive. This is also true for mantle convection, where surrogate models must be a function of simulation parameters and thereby learn features on vastly different spatio-temporal scales. For example, Agarwal et al. (2021) were able to learn the two-dimensional evolution of temperature fields, but they needed more than 10,000 simulations and did not predict other system variables such as flow velocities. The predictions also had errors with respect to the numerical simulations and the two-step learning procedure (dimensionality reduction with autoencoders and time-stepping with LSTMs) took almost a week to train on a single GPU.

This study provides an alternative approach where only 97 simulations were used to learn the profiles with a training time of approximately 2 min. No High Performance Computing resources were used. Also, the finally obtained steady-state is numerically accurate as the method initializes a numerical solver, whereas sometimes data-driven machine learning algorithms can be pretty far off from the numerical solution. At the same time, this study demonstrates a speedup of 2.8 times, whereas Agarwal et al. (2021) demonstrated a speedup on the order of 10,000. Although, a one-to-one comparison of the two approaches is non-trivial due to their different goals, setup of the simulations and learning tasks, it seems to be the case that the more computational effort is put into learning (data generation plus optimizing the neural network), the greater the potential speedup. We show how one can already start reaping efficiency gains from machine learning models trained on small data sets.

While methods for advancing the full system states in time remain an interesting alternative avenue of research—even for simulations aimed at reaching a statistical steady-state—the approach presented in this paper can also benefit from some improvements:

- Since higher accuracy in the prediction of temperature profiles generally led to a higher speedup, it would be worth improving the accuracy of the NN. One way to do so would be to add more data. Interestingly, these new simulations could already be initialized with the predictions of the current NN and hence potentially faster to generate. In the future one can even consider an incremental data set generation approach, where the NN trained on the previous chunks of data is used to accelerate the next batch. This approach has some interesting parallels with simulation-based inference methods where evaluations on limited data guide how later simulations are generated (e.g., Papamakarios & Murray, 2016).
- It would be remiss of us to generate larger training data sets without also extending the setup of the physical simulations. It would be interesting for mantle convection studies to incorporate more realistic geometries such as the spherical annulus (Fleury et al., 2024; Hernlund & Tackley, 2008), and to vary the ratio of core-to-surface radius in order to make the results readily relevant for different planetary bodies.
- Our initial attempts to learn a two-dimensional temperature field using the pointwise neural network were not successful, but it could be worth exploring other learning strategies to overcome the uncertainties introduced by the fluctuating components of the temperature fields. One example could be to learn a set of phase-shift features whose sole job is to help distinguish one snapshot of the temperature field in the statistical steady-state from another where only some plumes and downwellings have shifted. One could then produce any reasonable realization of this set of features as an initial condition and hopefully then attain the steady-state almost immediately with the solver. Alternatively, more recent advances in generative modeling for fluid flows could also be a promising area of research (e.g., Lienen et al., 2024; Saydemir et al., 2024).
- With this extended physical setup and possibly an even greater speedup from the improvements mentioned above, one would then be in the position to generate a large data set and derive generalized scaling laws in a spherical annulus geometry that incorporate mixed heating as well as pressure- and temperature-dependence of viscosity.

5. Conclusions

We presented a method to accelerate the computationally intensive determination of the statistical steady state of mantle convection simulations. Using a neural network (NN), it is possible to learn the final horizontally averaged temperature profile of the system. For unseen simulations, this prediction serves as an efficient initial condition for the solver which attains a statistical steady-state significantly faster than with other typically used initial conditions. While the NN prediction is already quite accurate in most cases, running the solver enables us to

obtain higher order statistics of the system such as the two-dimensional temperature fields and that too at numerical accuracy, that is, without any prediction error as the method initializes a numerical solver.

Even in the limit of the few simulations (97) used to train the NN, we found that it outperformed other regression baselines such as linear regression, kernel ridge regression and nearest neighbor interpolation. This was not only true when interpolating in the parameter space, but also when slightly extrapolating in at least one parameter. Furthermore, the results indicate that all the baseline prediction algorithms like kernel ridge regression and interpolation lead to some speedup over the typically used initializations for the two qualitatively distinct simulations we tested. It is encouraging that for the 23 simulations we ran to derive some basic scaling laws, we observed a speedup by our method in 83% of the cases. Including cases where no speedup was observed, the simulations initialized with NN predictions reached a steady-state 2.8 times faster than those initialized with hot profiles and 1.4 times faster than the interpolation algorithm.

From a machine learning for fluid dynamics perspective, this study falls on the less resource-intensive side, requiring fewer than 100 simulations with a training time of 2 min for the NN. While more sophisticated learning approaches can offer greater speedups, we have shown here that, when combined with domain expertise, machine learning can deliver actual benefits and can do so by speeding up traditional methods instead of competing with them. We would like to extend the physical setup of the simulations to a spherical annulus and also consider the core-to-surface radius ratio as an additional parameter. We further propose an incremental generation of a larger data set, where the NNs trained on the finished simulations are used to initialize the new ones. With these simulations, one can expect an even more accurate NN, allowing derivation of general scaling laws that simultaneously incorporate internal heating and temperature- and pressure-dependence of viscosity. We foresee generating a large enough data set for this application that would justify the upfront cost of generating training and validation data sets.

Regardless of where one falls on the spectrum of investment versus efficiency gains, machine learning promises to significantly accelerate mantle convection simulations and help improve our understanding of planetary interiors.

Appendix A: Training on Higher Parameter Ranges

As a demonstration of the network's capacity to fit simulations with higher parameter values than present in the original training set, we do a further training run where train on the simulations from the training and test sets and compare it to the NN used throughout the study.

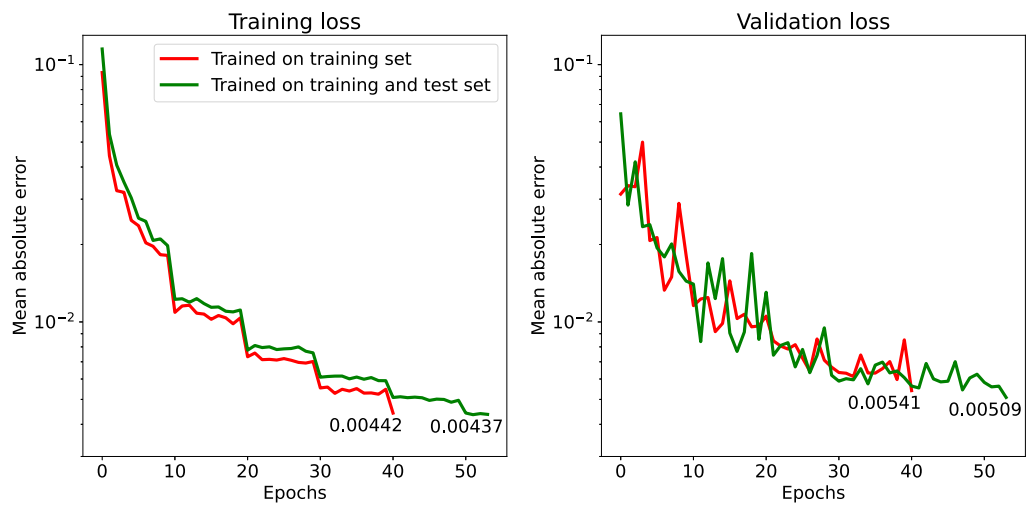

Figure A1. Training and validation loss curves for two different scenarios: when the same NN is trained on the training set alone versus when it is trained on the training set plus test set. In the latter case, the NN is able to fit these new simulations as well which now include higher parameter ranges ($Q > 9.5$ or $\gamma > 5 \times 10^9$ or $\beta > 95$). On the profiles from these extrapolate ranges alone, the NN achieves an MAE of 0.0037. Unsurprisingly, training on more simulations helps the NN achieve a better MAE on the validation set as well.

Data Availability Statement

The mantle convection simulations were run using GAIA (Hüttig et al., 2013). GAIA is proprietary code of the German Aerospace Center (DLR) and users interested in working with it should contact Nicola Tosi (nicola.tosi@dlr.de) and Christian Hüttig (christian.huettig@dlr.de). The temperature profiles (Agarwal et al., 2024) and the software (Agarwal et al., 2025) for preprocessing, training and evaluating as well as for reproducing all the figures in this paper are available via Zenodo as zip folders or more conveniently via the Hugging Face repository. We refer the readers to the README file for instructions on how to navigate the repository which is available under the MIT license: https://huggingface.co/spaces/agsiddhant/steadystate-mantle/blob/main/README.md. The trained model can be run to generate and download temperature profiles: https://huggingface.co/spaces/agsiddhant/steadystate-mantle. No installation is needed for this. Furthermore, the following open-source software libraries were used in Python (version 3.10.9): PyTorch 2.2.2 (Ansel et al., 2024), Scipy 1.13.1 (Virtanen et al., 2020), scikit-learn 1.5.0, (Pedregosa et al., 2011), and Matplotlib 3.8.4 (Hunter, 2007).

Acknowledgments
We thank Phil Livermore, Dorian Abbot and two anonymous reviewers for their constructive reviews that helped improve the initial version of the paper, and Steven Tobias and Enrico Camporeale for editorial handling. This work was carried out in the frame of the PLAGeS (Physics-based Learning Algorithms for Geophysical flow Simulations) project and funded by the German Ministry of Education and Research BMBF (project numbers 16DKWN117A and 16DKWN117B) under the "Deutschen Aufbau- und Resilienzplans" (DARP), which is part of NextGenerationEU's "Aufbau- und Resilienzfazilität" (ARF) by the European Union. Open Access funding enabled and organized by Projekt DEAL.

References

Agarwal, S., Tosi, N., Breuer, D., Padovan, S., Kessel, P., & Montavon, G. (2020). A machine-learning-based surrogate model of Mars' thermal evolution. *Geophysical Journal International*, *222*(3), 1656–1670. https://doi.org/10.1093/gji/ggaa234

Agarwal, S., Tosi, N., Hüttig, C., Greenberg, D., & Bekar, A. (2024). steadystate-mantle-dataset [Dataset]. *Zenodo*. https://doi.org/10.5281/zenodo.13771510

Agarwal, S., Tosi, N., Hüttig, C., Greenberg, D., & Bekar, A. (2025). steadystate-mantle [Software]. *Zenodo*. https://doi.org/10.5281/zenodo.14872695

Agarwal, S., Tosi, N., Kessel, P., Breuer, D., & Montavon, G. (2021). Deep learning for surrogate modeling of two-dimensional mantle convection. *Physical Review Fluids*, *6*, 113801. https://doi.org/10.1103/PhysRevFluids.6.113801

Akbari, S., Dabaghian, P. H., & San, O. (2023). Blending machine learning and sequential data assimilation over latent spaces for surrogate modeling of boussinesq systems. *Physica D: Nonlinear Phenomena*, *448*, 133711. https://doi.org/10.1016/j.physd.2023.133711

Ali Boroumand, M., Morra, G., & Mora, P. (2024). Extracting fundamental parameters of 2D natural thermal convection using convolutional neural networks. *Journal of Applied Physics*, *135*(14), 144702. https://doi.org/10.1063/5.0198004

Alieva, A., Hoyer, S., Brenner, M., Iaccarino, G., & Norgaard, P. (2023). Toward accelerated data-driven Rayleigh–Bénard convection simulations. *The European Physical Journal E*, *46*(7), 64. https://doi.org/10.1140/epje/s10189-023-00302-w

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., et al. (2024). PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Vol. 2, pp. 929–947). ACM. (asplos '24). https://doi.org/10.1145/3620665.3640366

Azmi, E., Meyer, J., Strobl, M., Weimer, M., & Streit, A. (2023). Approximation and optimization of global environmental simulations with neural networks. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. Association for Computing Machinery. https://doi.org/10.1145/3592979.3593418

Baumeister, P., & Tosi, N. (2023). ExoMDN: Rapid characterization of exoplanet interior structures with mixture density networks. *Astronomy & Astrophysics*, *676*, A106. https://doi.org/10.1051/0004-6361/202346216

Bergen, K. J., Johnson, P. A., de Hoop, M. V., & Beroza, G. C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. *Science*, *363*(6433), eaau0323. https://doi.org/10.1126/science.aau0323

Blankenbach, B., Busse, F., Christensen, U., Cserepes, L., Gunkel, D., Hansen, U., et al. (1989). A benchmark comparison for mantle convection codes. *Geophysical Journal International*, *98*(1), 23–38. https://doi.org/10.1111/j.1365-246X.1989.tb05511.x

Bodnar, C., Bruinsma, W. P., Lucic, A., Stanley, M., Brandstetter, J., Garvan, P., et al. (2024). Aurora: A foundation model of the atmosphere.

Breuer, D., & Moore, W. (2015). Dynamics and thermal history of the terrestrial planets, the Moon, and Io. In G. Schubert (Ed.), *Treatise on geophysics* (2nd ed., Vol. 10, pp. 255–305). Elsevier. https://doi.org/10.1016/B978-0-444-53802-4.00173-1

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*(1), 477–508. https://doi.org/10.1146/annurev-fluid-010719-060214

Christensen, U. (1984). Convection with pressure-and temperature-dependent non-Newtonian Rheology. *Geophysical Journal International*, *77*(2), 343–384. https://doi.org/10.1111/j.1365-246x.1984.tb01939.x

Coltice, N., & Schmalzl, J. (2006). Mixing times in the mantle of the early Earth derived from 2-d and 3-d numerical simulations of convection. *Geophysical Research Letters*, *33*(23). https://doi.org/10.1029/2006gl027707

Farnetani, C. G., & Samuel, H. (2003). Lagrangian structures and stirring in the Earth's mantle. *Earth and Planetary Science Letters*, *206*(3–4), 335–348. https://doi.org/10.1016/s0012-821x(02)01085-3

Ferrick, A. L., & Korenaga, J. (2023). Generalizing scaling laws for mantle convection with mixed heating. *Journal of Geophysical Research: Solid Earth*, *128*(5), e2023JB026398. https://doi.org/10.1029/2023JB026398

Ferziger, J. H., Perić, M., & Street, R. L. (2019). *Computational methods for fluid dynamics* (4th ed.). Springer. https://doi.org/10.1007/978-3-319-99693-6

Fleury, A., Plesa, A.-C., Hüttig, C., & Breuer, D. (2024). Assessing the accuracy of 2-d planetary evolution models against the 3-d sphere. *Geochemistry, Geophysics, Geosystems*, *25*(2), e2023GC011114. https://doi.org/10.1029/2023gc011114

Grasset, O., & Parmentier, E. (1998). Thermal convection in a volumetrically heated, infinite Prandtl number fluid with strongly temperature-dependent viscosity: Implications for planetary thermal evolution. *Journal of Geophysical Research*, *103*(B8), 18171–18181. https://doi.org/10.1029/98jb01492

Grigné, C., Labrosse, S., & Tackley, P. J. (2005). Convective heat transfer as a function of wavelength: Implications for the cooling of the Earth. *Journal of Geophysical Research*, *110*(B3). https://doi.org/10.1029/2004jb003376

Haldemann, J., Ksoll, V., Walter, D., Alibert, Y., Klessen, R. S., Benz, W., et al. (2023). Exoplanet characterization using conditional invertible neural networks. *Astronomy & Astrophysics*, *672*, A180. https://doi.org/10.1051/0004-6361/202243230

Hernlund, J. W., & Tackley, P. J. (2008). Modeling mantle convection in the spherical annulus. *Physics of the Earth and Planetary Interiors*, *171*(1–4), 48–54. (Recent Advances in Computational Geodynamics: Theory, Numerics and Applications). https://doi.org/10.1016/j.pepi.2008.07.037

Heyder, F., Mellado, J. P., & Schumacher, J. (2022). Generalizability of reservoir computing for flux-driven two-dimensional convection. *Physical Review E*, *106*(5), 055303. https://doi.org/10.1103/PhysRevE.106.055303

Heyder, F., Mellado, J. P., & Schumacher, J. (2024). Generative convective parametrization of a dry atmospheric boundary layer. *Journal of Advances in Modeling Earth Systems*, *16*(6), e2023MS004012. https://doi.org/10.1029/2023MS004012

Hines Chaves, D., Dias Ribeiro, M., & Bekemeyer, P. (2023). Physics-based regularization of neural networks for aerodynamic flow predictions. In *15th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control* (pp. 22–39). https://doi.org/10.7712/140123.10188.18859

Höink, T., & Lenardic, A. (2010). Long wavelength convection, Poiseuille–Couette flow in the low-viscosity asthenosphere and the strength of plate margins. *Geophysical Journal International*, *180*(1), 23–33. https://doi.org/10.1111/j.1365-246x.2009.04404.x

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hüttig, C., Tosi, N., & Moore, W. (2013). An improved formulation of the incompressible Navier-Stokes equations with variable viscosity. *Physics of the Earth and Planetary Interiors*, *220*, 11–18. https://doi.org/10.1016/j.pepi.2013.04.002

Illarramendi, E. A., Alguacil, A., Bauerheim, M., Misdariis, A., Cuenot, B., & Benazera, E. (2020). Towards an hybrid computational strategy based on deep learning for incompressible flows. In *AIAA Aviation 2020 Forum*. https://doi.org/10.2514/6.2020-3058

King, S. D., Lee, C., van Keken, P. E., Leng, W., Zhong, S., Tan, E., et al. (2010). A community benchmark for 2-D Cartesian compressible convection in the Earth's mantle. *Geophysical Journal International*, *180*(1), 73–87. https://doi.org/10.1111/j.1365-246X.2009.04413.x

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv e-prints*.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In U. Von Luxburg, et al (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, *118*(21), e2101784118. https://doi.org/10.1073/pnas.2101784118

Korenaga, J. (2010). Scaling of plate tectonic convection with pseudoplastic Rheology. *Journal of Geophysical Research*, *115*(B11). https://doi.org/10.1029/2010jb007670

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., et al. (2023). Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. Association for Computing Machinery. https://doi.org/10.1145/3592979.3593412

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., et al. (2023). Learning skillful medium-range global weather forecasting. *Science*, *382*(6677), 1416–1421. https://doi.org/10.1126/science.adi2336

Lienen, M., Lüdke, D., Hansen-Palmus, J., & Günnemann, S. (2024). From zero to turbulence: Generative modeling for 3D flow simulation. In *International Conference on Learning Representations*.

Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2023). Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *479*(2275), 20230058. https://doi.org/10.1098/rspa.2023.0058

Moore, W. B. (2008). Heat transport in a convecting layer heated from within and below. *Journal of Geophysical Research*, *113*(B11). https://doi.org/10.1029/2006JB004778

Moresi, L.-N., & Solomatov, V. (1995). Numerical investigation of 2d convection with extremely large viscosity variations. *Physics of Fluids*, *7*(9), 2154–2162. https://doi.org/10.1063/1.868465

Morra, G., Yuen, D. A., Tufo, H. M., & Knepley, M. G. (2020). Fresh outlook in numerical methods for geodynamics—Part 2: Big data, HPC, education. In D. Alderton & S. A. Elias (Eds.), *Encyclopedia of Geology* (pp. 841–855). Academic Press.

O'Farrell, K. A., & Lowman, J. P. (2010). Emulating the thermal structure of spherical shell convection in plane-layer geometry mantle convection models. *Physics of the Earth and Planetary Interiors*, *182*(1–2), 73–84. https://doi.org/10.1016/j.pepi.2010.06.010

O'Farrell, K. A., Lowman, J. P., & Bunge, H.-P. (2013). Comparison of spherical-shell and plane-layer mantle convection thermal structure in viscously stratified models with mixed-mode heating: Implications for the incorporation of temperature-dependent parameters. *Geophysical Journal International*, *192*(2), 456–472. https://doi.org/10.1093/gji/ggs053

Pandey, S., & Schumacher, J. (2020). Reservoir computing model of two-dimensional turbulent convection. *Physical Review Fluids*, *5*(11), 113506. https://doi.org/10.1103/PhysRevFluids.5.113506

Papamakarios, G., & Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. In *Neural Information Processing Systems Foundation, Inc. (30th Annual Conference on Neural Information Processing Systems, NIPS 2016 ; Conference date: 05-12-2016 Through 10-12-2016)* (pp. 1028–1036).

Park, J. J., Florence, P. R., Straub, J., Newcombe, R. A., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 165–174).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Phillips, B. R., & Coltice, N. (2010). Temperature beneath continents as a function of continental cover and convective wavelength. *Journal of Geophysical Research*, *115*(B4). https://doi.org/10.1029/2009jb006600

Qin, B., Ye, C., Liu, J., Huang, S., Wang, S., & ZhangZhou, J. (2024). Mapping global lithospheric mantle pressure-temperature conditions by machine-learning thermobarometry. *Geophysical Research Letters*, *51*(7), e2023GL106522. (e2023GL106522 2023GL106522). https://doi.org/10.1029/2023GL106522

Rasht-Behesht, M., Huber, C., Shukla, K., & Karniadakis, G. E. (2022). Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions. *Journal of Geophysical Research: Solid Earth*, *127*(5), e2021JB023120. https://doi.org/10.1029/2021JB023120

Reese, C., Solomatov, V., Baumgardner, J., & Yang, W.-S. (1999). Stagnant lid convection in a spherical shell. *Physics of the Earth and Planetary Interiors*, *116*(1), 1–7. https://doi.org/10.1016/S0031-9201(99)00115-6

Samuel, H., & Tosi, N. (2012). The influence of post-perovskite strength on the Earth's mantle thermal and chemical evolution. *Earth and Planetary Science Letters*, *323*, 50–59. https://doi.org/10.1016/j.epsl.2012.01.024

Saydemir, A., Lienen, M., & Günnemann, S. (2024). Unfolding time: Generative modeling for turbulent flows in 4d. In *ICML 2024 AI for Science Workshop*.

Sen, N., Pisharody, A. S., & Madanan, U. (2024). Empirical and machine learning approaches for turbulent thermal convection in rectangular enclosures tilted at acute angles. In S. Das, N. Mangadoddy, & J. Hoffmann (Eds.), *Proceedings of the 1st International Conference on Fluid, Thermal and Energy Systems* (pp. 525–536). Springer Nature Singapore.

Shahnas, M. H., & Pysklywec, R. N. (2020). Toward a unified model for the thermal state of the planetary mantle: Estimations from mean field deep learning. *Earth and Space Science*, *7*. https://doi.org/10.1029/2019EA000881

Solomatov, V. S. (1995). Scaling of temperature- and stress-dependent viscosity convection. *Physics of Fluids*, *7*(2), 266–274. https://doi.org/10.1063/1.868624

Stevenson, D. J., Spohn, T., & Schubert, G. (1983). Magnetism and thermal evolution of the terrestrial planets. *Icarus*, *54*(3), 466–489. https://doi.org/10.1016/0019-1035(83)90241-5

Sukumar, N., & Srivastava, A. (2022). Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, *389*, 114333. https://doi.org/10.1016/j.cma.2021.114333

Tompson, J., Schlachter, K., Sprechmann, P., & Perlin, K. (2017). Accelerating Eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference On Machine Learning—Volume 70* (pp. 3424–3433). JMLR.org.

Tosi, N., Stein, C., Noack, L., Hüttig, C., Maierova, P., Samuel, H., et al. (2015). A community benchmark for viscoplastic thermal convection in a 2-d square box. *Geochemistry, Geophysics, Geosystems*, *16*(7), 2175–2196. https://doi.org/10.1002/2015gc005807

van Keken, P. E., Ballentine, C. J., & Hauri, E. H. (2014). Convective mixing in the Earth's mantle. In H. D. Holland & K. K. Turekian (Eds.), *Treatise on geochemistry* (2nd ed., pp. 509–525). Elsevier. https://doi.org/10.1016/B978-0-08-095975-7.00212-6

Vilella, K., & Deschamps, F. (2018). Temperature and heat flux scaling laws for isoviscous, infinite Prandtl number mixed heating convection. *Geophysical Journal International*, *214*(1), 265–281. https://doi.org/10.1093/gji/ggy138

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, J., Mo, Y., Izzuddin, B., & Kim, C.-W. (2023). Exact dirichlet boundary physics-informed neural network epinn for solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, *414*, 116184. https://doi.org/10.1016/j.cma.2023.116184

Winter, P., Burger, C., Lehner, S., Kofler, J., Maindl, T., & Schäfer, C. (2022). Residual neural networks for the prediction of planetary collision outcomes. *Monthly Notices of the Royal Astronomical Society*, *520*(1), 1224–1242. https://doi.org/10.1093/mnras/stac2933

Wu, X., Liang, L., Shi, Y., & Fomel, S. (2019). FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation. *Geophysics*, *84*(3), IM35–IM45. https://doi.org/10.1190/geo2018-0646.1

Xiong, F., Liu, J., Guo, Z., & Liu, J. (2022). Deep-neural-networks-based approaches for biot-squirt model in rock physics. *Acta Geophysica*, *70*(2), 593–607. https://doi.org/10.1007/s11600-022-00740-8

Yu, S., & Ma, J. (2021). Deep learning for geophysics: Current and future trends. *Reviews of Geophysics*, *59*(3), e2021RG000742. https://doi.org/10.1029/2021RG000742

Zhang, S., & Ritzwoller, M. H. (2024). Applying machine learning to characterize and extrapolate the relationship between seismic structure and surface heat flow. *Geophysical Journal International*, *238*(3), 1201–1222. https://doi.org/10.1093/gji/ggae218

Zhao, Y., & Ni, D. (2021). Machine learning techniques in studies of the interior structure of rocky exoplanets. *Astronomy & Astrophysics*, *650*, A177. https://doi.org/10.1051/0004-6361/202140375

Ziv, M., Galanti, E., Sheffer, A., Howard, S., Guillot, T., & Kaspi, Y. (2024). Neuralcms: A deep learning approach to study Jupiter's interior. *Astronomy & Astrophysics*, *686*, L7. https://doi.org/10.1051/0004-6361/202450223