Deep learning for surrogate modeling of two-dimensional mantle convection

Siddhant Agarwal ●,[1,2,*] Nicola Tosi ●,[1] Pan Kessel,[2] Doris Breuer,[1] and Grégoire Montavon[2]

[1]*Planetary Physics, Institute of Planetary Research, German Aerospace Center (DLR),*
Berlin 12489, Germany
[2]*Machine Learning Group, Berlin Institute of Technology, Berlin 10587, Germany*

Mantle convection, the buoyancy-driven creeping flow of silicate rocks in the interior of terrestrial planets like Earth, Mars, Mercury, and Venus plays a fundamental role in the long-term thermal evolution of these bodies. Yet key parameters and initial conditions of the partial differential equations governing mantle convection are poorly constrained. This often requires a large sampling of the parameter space to determine which combinations can satisfy certain observational constraints. Traditionally, 1D models based on scaling laws used to parameterized convective heat transfer have been used to tackle the computational bottleneck of high-fidelity forward runs in two or three dimensions. However, these are limited in the amount of physics they can model (e.g., depth-dependent material properties) and predict only mean quantities such as the mean mantle temperature. A recent machine learning study has shown that feedforward neural networks (FNNs) trained using a large number of 2D simulations can overcome this limitation and reliably predict the evolution of entire 1D laterally averaged temperature profile in time for complex models. We now extend that approach to predict the full 2D temperature field, which contains more information in the form of convection structures such as hot plumes and cold downwellings. Using a data set of 10 525 2D simulations of the thermal evolution of the mantle of a Mars-like planet, we show that deep learning techniques can produce reliable parameterized surrogates (i.e., surrogates that predict state variables such as temperature based only on parameters) of the underlying partial differential equations. We first use convolutional autoencoders to compress the size of each temperature field by a factor of 142 and then use FNNs and long-short-term memory networks (LSTMs) to predict the compressed fields. On average, the FNN predictions are 99.30% and the LSTM predictions are 99.22% accurate with respect to unseen simulations. Proper orthogonal decomposition (POD) of the LSTM and FNN predictions shows that despite a lower mean relative accuracy, LSTMs capture the flow dynamics better than FNNs. When summed, the POD coefficients from FNN predictions and from LSTM predictions amount to 96.51% and 97.66% relative to the coefficients of the original simulations, respectively.

I. INTRODUCTION

Studying the long-term thermal evolution of terrestrial planets like Earth, Venus, Mercury, and Mars requires detailed modeling of thermal convection in their rocky mantles [1]. Similar to the flow of crystalline ice in glaciers, mantle convection is a form of subsolidus convection. Mantle rocks at high temperature and pressure, but still well below their melting point, behave like a

---

*agsiddhant@gmail.com

highly viscous fluid over geological timescales (millions to billions of years), largely in response to thermal and compositional buoyancy forces. Mantle convection is typically modelled using fluid dynamics codes [2–5] that numerically solve the nonlinear partial differential equations (PDEs) of mass, momentum, and energy conservation governing the creeping flow (i.e., with negligible inertia) of silicate rocks subject to basal heating from the metallic core and internal heating due to the decay of radioactive elements. Unfortunately, key model parameters (such as the rock viscosity or the amount of internal heating) and initial conditions (such as the initial mantle and core temperatures), which are inputs to these forward numerical models, are poorly constrained. Thus, one typically chooses and tests a large number of different parameter values (within a reasonable range) to observe how these affect the outputs of the simulations. The outputs can be processed to arrive at various quantities of interest such as surface heat flux, amount of thermal contraction or expansion, duration and timing of volcanism, or volume of produced crustal material. To some extent, these quantities are "observables" that can be inferred from geophysical and geochemical data delivered by planetary space missions. In turn, they provide fundamental constraints on the convective evolution of terrestrial bodies (see Ref. [6] for a recent review about this topic).

This approach of testing several different values of parameters, however, suffers from a bottleneck imposed by the computational cost of the 2D and 3D forward models. No matter whether one approaches this issue from the perspective of an inverse problem (inferring the model parameters given the observables) or of a forward problem (calculating the observables given the model parameters), it is often impractical to run several thousands of simulations to determine which parameters and combinations thereof can satisfy a set of given observational constraints.

A number of inverse-problem studies have attempted to overcome this computational bottleneck of expensive simulations, ranging from using modified Markov chain Monte Carlo (MCMC) methods [7,8], all the way to completely bypassing MCMC methods and directly learning the mapping between parameters and observables from simulations run prior to the inversion using mixture density networks (MDNs) [9–13].

Machine learning (ML) methods such as MDN can also be used to learn highly nonlinear forward mappings from parameters to observables. These can preserve some physical insights into the flow being modeled (such as convection patterns) in contrast to purely statistical inferences made under an inverse formulation. Traditionally, several mantle convection studies have employed "scaling laws," which parametrize the heat flow out of the mantle [quantified by the Nusselt number (Nu), or the nondimensional heat flux], in terms of the vigor of convection [quantified by the Rayleigh number (Ra)][14–17]. These scaling laws are then used in the frame of 1D, spherically symmetric models to advance the mean mantle temperature (and a number of associated quantities) in time by solving two ordinary differential equations governing the global energy balance of the mantle and core [18–25].

On the one hand, the use of scaling laws makes models of planetary evolution computationally very efficient as these require only the solution of ordinary differential equations. On the other hand, scaling laws are limited in the amount of physics they can capture. For example, solid-solid phase transitions or the pressure dependence of the viscosity and other thermal and transport properties such as thermal expansivity and conductivity can hardly be taken into account in scaling laws for heat transfer. Additionally, they allow one only to predict the evolution of global quantities such as the surface heat flux or the mean mantle temperature, but do not provide any insight into the spatial and temporal variability of mantle flow. Recently, Ref. [26] showed that some of these limitations can be overcome by using machine learning. Those authors showed that feedforward neural networks (FNNs) can be used for predicting the surface heat flux and mean temperature of steady-state simulations, given parameters such as Ra, the core-to-planet radius ratio, and mode of convection, i.e., mobile lid, characterizing Earth's plate tectonics, or stagnant lid, characterizing the other terrestrial bodies of the solar system (see also Sec. II). Reference [27] demonstrated that limitations related to taking into account complex physics and the time variability of the heat transfer can be overcome through FNNs properly trained with a large set of 2D dynamic simulations. By

using multiple parameters such as, e.g., mantle reference viscosity (related to Ra) and activation volume and activation energy of diffusion creep rheology (controlling the temperature and pressure dependence of the viscosity) as inputs to the FNN, it is possible to directly predict the thermal evolution of the entire horizontally averaged 1D temperature profile of the mantle in time with a mean accuracy of 99.7%. Another interesting study uses generative adversarial networks to reconstruct missing plate boundaries derived from horizontal divergence maps of a steady-state 3D convection model [28].

In this paper, we build upon Ref. [27]. While the full 1D temperature profile already provides much more information than simply the surface heat flux and mean temperature and is also beyond what can be expected to be retrieved for planets like Mars through future planetary missions, it still lacks the rich convection structures such as plumes and downwellings that are delivered by 2D or 3D simulations. In fact, Ref. [13] demonstrated that using the same setup for a Mars-like planet as in Ref. [27], the surface heat flux provides only a loose constraint for parameters governing mantle convection and even knowledge of the entire 1D temperature profile is insufficient for constraining certain parameters (e.g., the activation volume of diffusion creep rheology, which controls the pressure dependence of the viscosity). In other words, more information (such as horizontal variations of the temperature field) might hold the key to placing tighter constraints on certain thermal-evolution parameters. Hence, it is desirable to construct surrogate models capable of predicting more information about a convecting mantle.

Here we show how deep learning (DL) can be leveraged to directly predict surrogates of the 2D temperature fields from five key parameters. Figure 1 shows the concept and general context within which this study fits. We sample the model parameters from a flat distribution with a broad, yet reasonable range and input these to the convection simulations. The five key parameters are (1) reference mantle viscosity (η_{ref}), which is the viscosity attained at a given reference depth and reference temperature; (2) activation energy of the diffusion creep (E), which controls the temperature dependence of the viscosity and (3) activation volume of diffusion creep (V) controlling the degree to which viscosity depends on pressure; (4) enrichment factor (Λ), which determines the proportion of the heat-producing radiogenic elements extracted from the convecting mantle upon melting and enriched in the crust; and (5) the initial temperature of the mantle (T_{ini}). These five parameters typically have the largest influence on the forward model for the thermal evolution of terrestrial planets that we use [29,30].

To our best knowledge, this is the first time a parameterized 2D surrogate is proposed in the context of mantle convection (see Ref. [31] for a recent review of applications of data science methods in geodynamics). We use the term "parameterized" to stress that we are interested in predicting a variety of mantle flows based on different combinations of input parameters and not, for example, given a certain amount of time steps of a single simulation with fixed parameters, predicting the subsequent time steps. Reference [32] demonstrated that direct numerical simulation (DNS) of 2D turbulent Rayleigh-Bénard convection can be modeled using reservoir computing. In that study, the time steps of the same, single simulation are split into training and test sets. In contrast, we are interested in modeling all the time steps of different simulations, which is made possible by the relatively low computational cost of each simulation (20 to 500 CPU hours depending on the combination of parameters).

In general, machine learning for predicting flows is an active research area. Particularly worth highlighting is the seminal paper [33], which showed that the PDEs can be embedded in the loss function through automatic differentiation of state variables with respect to spatial and temporal coordinates. However, here we stick to a purely data-driven approach, reserving the application of methods such as those in Ref. [33] for future research. A notable example of such a purely data-driven approach was shown to be effective in capturing the dynamics of 3D turbulence [34]. They showed that the velocity fields can be compressed using convolutional autoencoders [35]. Convolutional autoencoders successively down-size the original field (or image) into a bottleneck, from where they are reconstructed back to the original size. In this way, the dimensionality of
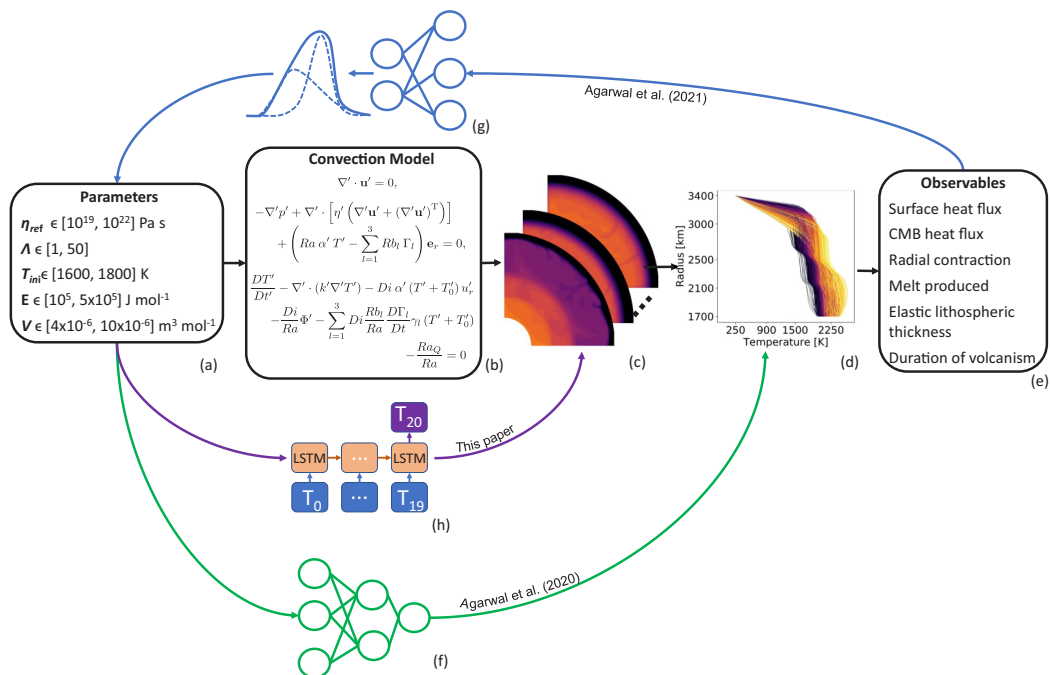
FIG. 1. The context for this study. (a) Typically a mantle convection study starts by randomly drawing input parameters from a flat distribution and then feeding them to the forward models. (b) The PDEs are solved using dedicated mantle convection codes. (c) The outputs of the simulations can be processed to arrive at certain lower-dimensional observables such as (d) the horizontally averaged 1D temperature profiles or (e) more global quantities such as the surface heat flux, radial contraction, duration of volcanism, etc. ML methods have been shown to work well for these low-dimensional observables, both (f) in a forward study [27] and (g) in an inverse study [13]. In this work, we demonstrate that (h) a surrogate can model 2D mantle convection using deep learning.

the original high-resolution fields can be decreased and made more computationally efficient to work with. Reference [34] then predicted these compressed time steps using a convolutional long-short-term memory (LSTM) network [36], which by providing a mechanism to relate the time steps of a simulation allows one to learn the attractor for the underlying dynamics. Just as Refs. [32,34] split the time steps of the same simulation into training and test sets, we show that this approach can be adapted to our purposes, so that we can predict all the time steps for any simulation in our data manifold, given just a set of five parameters. Somewhat closer to our purposes, Ref. [37] used a convolutional encoder-decoder architecture for predicting pressure and temperature fields around airfoils, given the spatial grid as well as two additional parameters (angle of attack and Reynolds number). We refer to Ref. [38] for an overview of several ML techniques that have been used for prediction, dimensionality reduction, and optimization and control in fluid dynamics.

The outline of the paper is as follows. In Sec. II we briefly outline the setup of the simulations of a Mars-like planet. Then in Sec. III we present how we compress the temperature fields obtained from the mantle convection simulations. We follow with Sec. IV, where we predict the compressed temperature fields using two different ML algorithms, FNN and LSTM. In the same section, we further delve into the differences in the predictions of the two algorithms by analyzing them from the lens of proper orthogonal decomposition (POD) or its ML equivalent, principal component analysis. We then conclude the paper by offering some potentially interesting follow-ups to this work.
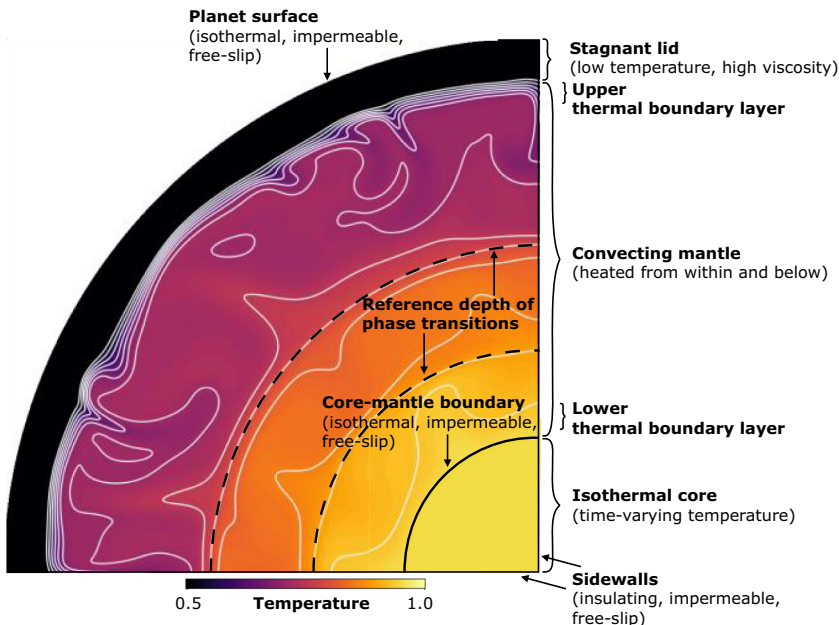
FIG. 2. Two-dimensional, quarter-cylindrical domain illustrating the main features of the employed mantle convection model. The mantle is colored according to the nondimensional temperature field of one specific simulation of the data set. The diagram depicts the smaller rescaled core (see Sec. A 8). Since a large part of the temperature variations occur across the stagnant lid, the color scale is truncated at 0.5 for ease of visualization. See text for more details and Appendix A for a complete model description.

II. DATA SET OF MANTLE CONVECTION SIMULATIONS

We employ a data set consisting of 10 525 simulations of the thermal evolution of a Mars-like planet run on a 2D quarter-cylindrical grid (Fig. 2). A detailed description of the setup is provided in Appendix A. Here we summarize the main features of the model.

We model the mantle as a viscous fluid with negligible inertia (i.e., with infinite Prandtl number or, equivalently, undergoing Stokes flow). We consider a pressure- and temperature-dependent Newtonian rheology [39], which is calculated using the Arrhenius law for diffusion creep, whose dimensional form reads

$$\eta(T, P) = \eta_{\text{ref}} \exp\left(\frac{E + PV}{T} - \frac{E + P_{\text{ref}}V}{T_{\text{ref}}}\right). \tag{1}$$

The reference viscosity η_{ref} is attained at reference temperature $T_{\text{ref}} = 1600$ K and reference pressure $P_{\text{ref}} = 3$ GPa, respectively. P is the hydrostatic pressure, E is the activation energy, and V is the activation volume.

Mars, in contrast to the Earth, but like Mercury, the Moon and, at least at present-day, Venus, operates in a so-called stagnant-lid convection mode (see e.g., Ref. [6]). The strong temperature dependence of the viscosity causes the relatively cold upper part of the mantle to develop a high-viscosity layer—the stagnant lid. Such a stiff layer remains immobile during the entire evolution of the planet (although its thickness can decrease or increase in response to heating or cooling of the mantle). The stagnant lid insulates the interior causing thermal convection to take place only in the mantle beneath it and to be largely driven by cold downwellings developing at its base (see Fig. 2). This mode of convection is remarkably different from the plate tectonic (or mobile lid) mode of convection that characterizes the Earth. Cold tectonic plates are in fact an active part of the Earth's convecting engine: by sinking into the mantle at subduction zones, they provide a strong cooling

for the mantle and core with fundamental consequences for large-scale transport of materials in the deep interior and for the generation of the Earth's magnetic field.

Within the relatively low pressure range of Mars mantle ($\sim$20 GPa), the degree of compressibility of mantle rocks is limited; the dissipation number [Eq. (A11)] is quite small for Mars ($\sim$0.13, significantly smaller than the Earth's, which amounts to $\sim$0.5). In this situation, it is appropriate to employ the so-called extended Boussinesq approximation [40]. Like the standard Boussinesq approximation, the extended one assumes constant density everywhere except in the buoyancy term of the momentum equation [Eq. (A2)]. However, it further accounts for the effects of adiabatic compression/decompression and viscous dissipation in the conservation equation for the thermal energy [Eq. (A3)].

The mantle temperature evolves in time due to the decay of radiogenic elements present in the mantle and due to the cooling of the core, which also provides a source of basal heat. Following a standard approach adopted in the mantle convection community, the core is simply considered as a homogeneous sphere of given density and heat capacity, whose mean temperature evolves at a rate imposed by the cooling rate of the mantle [see Fig. 2, Eq. (A21), and, e.g., Ref. [18]].

Geological evidence suggests that the bulk of the crust of Mars formed very early in the evolution of the planet [41]. Similar to other studies (e.g., Ref. [30]), we thus assume that a crust of a fixed thickness has been present since the beginning of the evolution. This assumption implies that, from the beginning of each simulations, the mantle is partly depleted of radiogenic elements (resulting in new depleted mantle composition C_{depleted}) with respect to their primordial concentration (C_0), which we set according to the model of Ref. [42] as follows:

$$C_{\text{depleted}} = \frac{M_{\text{m}} C_0}{M_{\text{cr}}(\Lambda - 1) + M_{\text{m}}}, \tag{2}$$

where M_{m} and M_{cr} are the mass of the mantle and crust, respectively and Λ is the crustal enrichment factor. The rationale of this assumption is that upon partial melting of the mantle—a common event during the evolution of terrestrial bodies—radiogenic elements behave as incompatible elements; i.e., they tend to be enriched in the melt phase with respect to the solid phase. As a consequence, the production of crust, which results from mantle melting, melt migration toward the surface, and solidification, causes a net depletion of radiogenic elements in the mantle. In practice, we reduce the primordial bulk abundances of radiogenic elements according to a crustal enrichment factor Λ. The mantle is further depleted in radiogenic heat-producing elements during the evolution whenever local partial melting takes place according to the model described in Ref. [43] (see Sec. A 6).

We also consider the influence on the mantle flow of two major solid-solid phase-transitions using the standard phase-function approach of Ref. [44] adopted in mantle convection (Sec. A 5). We further assume that the coefficient of thermal expansion and the thermal conductivity depend on pressure and temperature as appropriate for silicate materials (see Sec. A 4 and Ref. [45]).

The simulations are initialized with a constant mantle temperature T_{ini} combined with upper and lower 300-km-thick thermal boundary layers. For all simulations, a small random perturbation is added to the temperature field to initiate convection. The initial mantle temperature has a strong effect on the early evolution of the planet. However, this initial condition becomes less important with time (after $\sim$2 billion years) due to the "thermostat effect" [46]: the temperature dependence of the viscosity regulates the temperature of the mantle. On the one hand, when the mantle is hot, the viscosity decreases, leading to more vigorous convection and thereby, efficient cooling. On the other hand, when the mantle is cooler at later stages in the evolution, it is more viscous and cools less efficiently. While in this data set of forward models we vary only the initial mantle temperature, other parameters related to the initial conditions that can have a potential impact on the thermal evolution of the interior could also be considered. For example, the starting temperature at the core-mantle boundary has implications for the dynamics of the lower mantle and affects melt and magnetic field generation [47]. The initial thickness of the thermal boundary layers, although irrelevant for the long-term evolution, influences the heat fluxes across the

TABLE I. Mean relative accuracy of different LSTM architectures on the cross-validation (CV) and the test sets, when computed in inference mode (i.e., each input temperature field is also an LSTM prediction). For reference, mean relative accuracy for the FNN architecture is also presented.

Accuracy	wrt GAIA (%)	wrt ConvAE (%)
2×900		
Test	98.182 ± 9.853	98.237 ± 9.854
CV	96.542 ± 15.041	96.597 ± 15.049
3×900		
Test	99.222 ± 0.515	99.278 ± 0.513
CV	99.109 ± 0.602	99.164 ± 0.605
4×900		
Test	99.226 ± 0.524	99.285 ± 0.525
CV	99.082 ± 0.674	99.141 ± 0.682
5×900		
Test	99.199 ± 0.537	99.257 ± 0.537
CV	99.051 ± 0.680	99.108 ± 0.687
4×600		
Test	99.221 ± 0.495	99.281 ± 0.495
CV	99.081 ± 0.665	99.140 ± 0.671
2×1200		
Test	98.275 ± 8.313	98.328 ± 8.317
CV	97.364 ± 11.352	97.416 ± 11.356
2×1800		
Test	98.561 ± 6.446	98.616 ± 6.445
CV	98.326 ± 7.376	98.379 ± 7.380
FNN		
Test	99.297 ± 0.433	99.354 ± 0.422
CV	99.207 ± 0.482	99.262 ± 0.475

surface and CMB during the first few hundred million years. Additionally, an initial compositional stratification resulting from the crystallization of a primordial magma ocean, neglected in our isochemical models, could also have a significant influence on the evolution of the mantle and core [48–50].

As for the boundary conditions, all domain boundaries are impermeable and free-slip. The surface temperature is kept fixed at 250 K throughout the evolution. Latitudinal variations of the surface temperature, as on Mars [51], do not have a significant impact on the long-term evolution and large-scale dynamics of the planet [52] and thus can be safely neglected. The temperature of core-mantle boundary evolves as the core cools [Eq. (A21)]. There is no heat flux across the side walls of the computational domain, i.e., they are assumed to be insulating (see Fig. 2). Table II and Table III in Appendix A list all the fixed dimensional parameters that are shared by all simulations.

We ran several single-core simulations using the finite-volume code GAIA [5] with a grid resolution of 300 radial layers and 392 cells per layer. Even though 3251 out of 10 525 simulations did not reach the end time of 4.5 Gyr (i.e., time since formation of the planet until today), we use all the time steps available from all the simulations. Some combinations of parameters can make the system of PDEs too stiff to efficiently solve, as the time stepping becomes increasingly small. This means that some simulations can take more than 50 times the average run time. In total, the data set amounts to 10 TB and took approximately 2 million CPU hours. The challenging task of predicting the 2D temperature field for a variety of parameters necessitated this computational effort. The number of simulations used in this study is therefore almost 100 times that of what a

TABLE II. Values of fixed parameters shared by all simulations (part 1).

Parameter	Physical meaning	Value	Unit
$\Delta T_{t=0}$	Initial temperature difference between core and surface[a]	2000	K
T_0	Surface temperature[a]	250	K
ρ_c	Core density[a]	7000	kg m^{-3}
ρ_m	Mantle density[a]	3500	kg m^{-3}
c_{p_c}	Core specific heat capacity[a]	850	J kg^{-1} K^{-1}
c_{p_m}	Mantle specific heat capacity[a]	1200	J kg^{-1} K^{-1}
k_{ref}	Reference thermal conductivity[a]	4	W m^{-1} K^{-1}
α_{ref}	Reference thermal expansivity[a]	2.5×10^{-5}	K^{-1}
R_c	Outer radius of the core[a]	1700	km
R_p	Planetary radius[a]	3400	km
d_{cr}	Thickness of the crust	64.3	km
z_{ref}	Reference depth for viscosity	232	km
T_{ref}	Reference temperature for viscosity[a]	1600	K
$z_{\alpha\beta}^0$	Reference depth for α to β spinel[a]	1020	km
$z_{\beta\gamma}^0$	Reference depth for β to γ spinel[a]	1360	km
$\Delta\rho_{\alpha\beta}^0$	Density difference for α to β spinel[a]	250	kg m^{-3}
$\Delta\rho_{\beta\gamma}^0$	Density difference for β to γ spinel[a]	150	kg m^{-3}
$\gamma_{\alpha\beta}$	Clapeyron slope for α to β spinel[a]	3×10^6	Pa K^{-1}
$\gamma_{\beta\gamma}$	Clapeyron slope for β to γ spinel[a]	5.1×10^6	Pa K^{-1}
$T_{\alpha\beta}$	Reference temperature for α to β spinel[a]	1820	K
$T_{\beta\gamma}$	Reference temperature for β to γ spinel[a]	1900	K
d_l	Width of phase transitions[a]	20	km
$^U C_0$	Bulk abundance of uranium[b]	16×10^{-9}	kg kg^{-1}
$^{Th} C_0$	Bulk abundance of thorium[b]	56×10^{-9}	kg kg^{-1}
$^K C_0$	Bulk abundance of potassium[b]	305×10^{-6}	kg kg^{-1}

[a]Reference [29]
[b]Reference [81].

typical parameter study in mantle convection would use. The distribution of the parameters for all the simulations in the training, cross-validation, and test set is plotted in Fig. 16 in Appendix B.

III. COMPRESSION OF TEMPERATURE FIELDS

The temperature fields alone account for approximately 1 TB. While most everyday computers cannot hold that much data in memory, one could still overcome the issue using a "data generator." A data generator feeds the ML algorithm during training and/or inference by reading and storing only a limited number of examples from the disk at any given time. Nevertheless, these require careful programming to ensure that the CPU or GPU is constantly fed batches of data without having to wait for the next one. Furthermore, performing ML calculations on the full-sized fields would be slower than a compressed version of it and introduce many more trainable parameters which could increase the risk of overparameterization. Hence, we decided to compress the temperature fields first and then scan for different ML algorithms and architectures that can help us predict this latent space representation.

Traditionally, linear reduced-order modeling techniques like proper orthogonal decomposition (POD) have been used for truncating high-fidelity simulations using the most dominant modes [53]. However, the orthonormal bases (typically obtained through singular value decomposition) of one simulation do not generalize well onto those of another and often require nontrivial basis interpolation [54]. Recently, Ref. [34] demonstrated that convolutional autoencoders (ConvAE) [35]

TABLE III. Values of fixed parameters shared by all simulations (part 2).

Parameter	Physical meaning	Value	Unit
a_0	Coefficient of thermal expansivity[a]	3.15×10^{-5}	K^{-1}
a_1	Coefficient of thermal expansivity[a]	1.02×10^{-8}	K^{-2}
a_2	Coefficient of thermal expansivity[a]	-0.76	K
a_3	Coefficient of thermal expansivity[a]	3.63×10^{-2}	GPa^{-1}
c_0	Coefficient of thermal conductivity[a]	2.47	$Wm^{-1} K^{-1}$
c_1	Coefficient of thermal conductivity[a]	0.33	$Wm^{-1} K^{-1} GPa^{-1}$
c_2	Coefficient of thermal conductivity[a]	0.48	
e_0	Coefficient for solidus parametrization[b]	1400	K
e_1	Coefficient for solidus parametrization[b]	149.5	$K\, GPa^{-1}$
e_2	Coefficient for solidus parametrization[b]	-9.4	$K\, GPa^{-2}$
e_3	Coefficient for solidus parametrization[b]	0.313	$K\, GPa^{-3}$
e_4	Coefficient for solidus parametrization[b]	-0.0039	$K\, GPa^{-4}$
f_0	Coefficient for liquidus parametrization[c]	1977	K
f_1	Coefficient for liquidus parametrization[c]	64.1	$K\, GPa^{-1}$
f_2	Coefficient for liquidus parametrization[c]	-3.92	$K\, GPa^{-2}$
f_3	Coefficient for liquidus parametrization[c]	0.141	$K\, GPa^{-3}$
f_4	Coefficient for liquidus parametrization[c]	-0.0015	$K\, GPa^{-4}$

[a]Reference [45]
[b]Reference [78]
[c]Reference [79].

provide a powerful nonlinear tool for compressing flow fields, bypassing the need for calculating POD modes.

Instead of fully connected dense layers like conventional autoencoders, ConvAE uses convolutional filters. As shown in Fig. 3(a), a filter with trainable weights **w** moves across the state variable field (temperature) as specified by a hyperparameter called "stride." A stride of 2, for example, means that the filters move two units (two numerical grid cells) horizontally and then when a row is completed, two units vertically. Figure 3(a) shows a filter with height 5 and length 7 (also hyperparameters) convolving with the temperature field at strides of 2 in both the x- and y-directions. We use tanh() as activation function, which, when applied to the sum of the bias and the convolution product, introduces nonlinearity and returns the output for the next hidden layer, which can then be convolved on. This process continues until the desired latent space representation is reached [Fig. 3(b)]. Then the compressed state can be successively restored to the original size with the help of another sequence of convolution filters, called deconvolutions in this context. Once the forward graph has been set up, one can then minimize the difference between the original and reconstructed 2D temperature field by back-propagating the derivative of the error with respect to the network weights. It was pointed out in Ref. [35] that a convolutional architecture offers two main benefits over the fully connected dense layers: (1) the 2D structures of the convolutional filters retain spatial correlations which would otherwise be lost in 1D dense layers and (2) ConvAE have significantly fewer trainable parameters due to the shared weights.

We use Keras, an API built on top of Tensorflow [55] for training our ConvAE. We split the simulations into 98%, 1%, and 1% for training, validating, and testing, respectively. Given the size of the entire training set, we feed only the GPU minibatches of 16 temperature fields (i.e., time steps of any simulation) during training. We use L2 regularization and early stopping by manually monitoring the validation loss (mean-squared error), and the optimization is carried out using Adam [56].

Since the computer cannot hold the entire training set in memory, we use a data-generator. Using multiprocessing built into Keras' "fit generator" for creating multiple batches in parallel along with
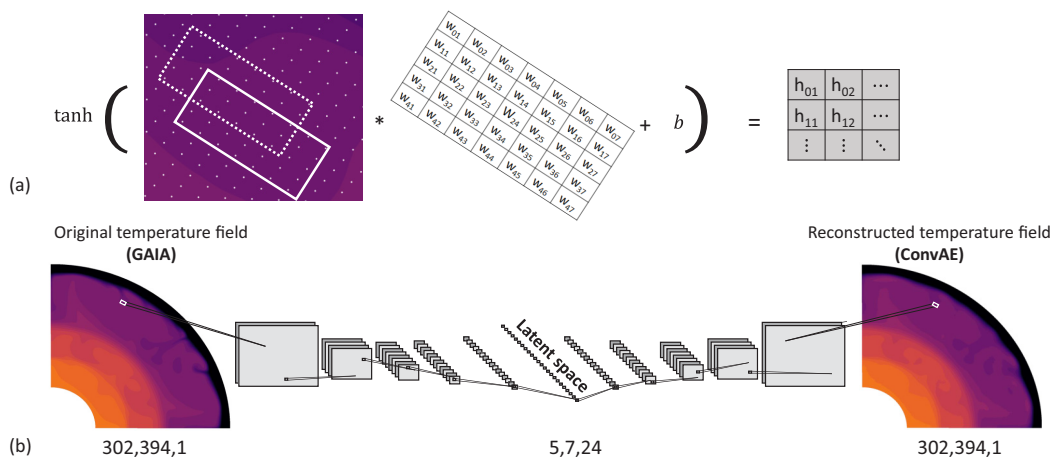
FIG. 3. Dimensionality reduction of the 2D temperature fields using convolutional autoencoders. (a) Filters with trainable parameters ($\mathbf{w}$) move across the computational domain with specified strides. After the convolution operation ($*$), the bias b is added to it before applying the activation function, resulting in the entries for the next hidden layer $\mathbf{h}$. (b) Several filters can be used to successively reduce the size of the original field ($302 \times 394 \times 1$) till a desired encoding or latent space representation is obtained ($5 \times 7 \times 24$). The convolutional autoencoder then reconstructs the compressed field back to its original dimensions ($302 \times 394 \times 1$) using deconvolution operations and optimizes the weights using the difference between the prediction and the ground truth.

multithreading for populating each batch with the help of Joblib [57], it takes around 6 h for one epoch to complete. After 5–10 epochs, acceptable results are obtained. Figures 4(a) and 4(b) show results of reconstruction for two different examples in the test set using three different ConvAE architectures, which differ in the dimensions of the latent space. As can be expected, the more the temperature fields are compressed, the less accurate the reconstruction is. In all the plots of the temperature field in this paper, we clip the color bar below 0.5 and above 1.0 to enhance the visualization of plumes and downwellings. Furthermore, we always plot the nondimensionalized temperature and the nondimensionalized radius.

We find that ConvAE with a latent size of 840 (or width $\times$ height $\times$ channels $= 5 \times 7 \times 24$) offers an excellent compression factor of 142, while being able to reconstruct the temperature fields with a mean relative accuracy of 99.80% on the test set. In comparison, ConvAEs with 1620- and 7600-dimensional latent spaces are 99.88% and 99.90% accurate, respectively. To calculate the mean relative accuracy, we dimensionalize the temperature using Eq. (A7) to avoid division by zeros.

In the subsequent sections, we discuss the training of ML algorithms to predict the temperature fields compressed to 840 numbers and then reconstruct the 302×394-sized temperature fields using the trained decoder, i.e., we can compress the data from 1 TB to approximately 7 GB. We also tested the 1620- and 7600-dimensional encodings when predicting the compressed temperature fields— as explained in the subsequent section—and found no significant improvement in the predictions, suggesting that the accuracy of the 840-dimensional compression is good enough for a data set of this size. Keeping the latent space as small as possible is especially useful when training LSTMs, because an LSTM cell has eight times as many trainable parameters as a dense FNN layer.

The results of the simple ConvAE are also encouraging in the light that the computational grid is structured, but not uniform, as is typically the case in computer vision applications. It seems that the different filters in the ConvAE are capable of capturing features at different spatial scales. For now, accounting for the curvilinear nature of the mesh and thereby potentially achieving higher compressibility and/or accuracy remains subject to future research.
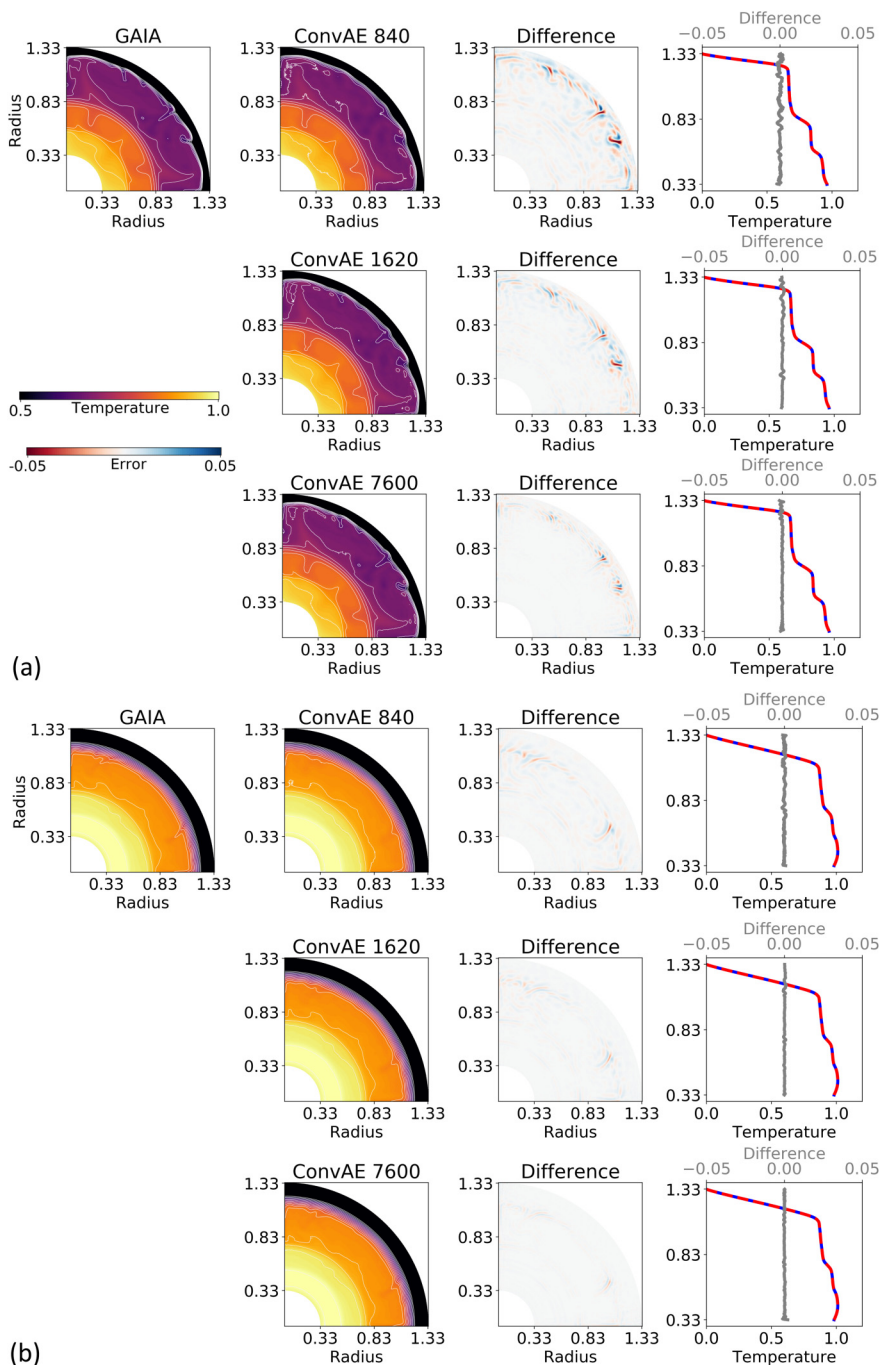
FIG. 4. (a) First example of compression and reconstruction from the test set for different architectures of convolutional autoencoders. ConvAE 840 represents a 142-fold compression of the original field (302×394), while ConvAE 1620 compresses the field by a factor of 73 and ConvAE 7600 by a factor of 16. (b) Second example with vigorous convection and small-scale structures from the test set. For each ConvAE, the error in the reconstruction is plotted in the third column, along with the horizontally averaged 1D temperature profiles in the fourth column (and the difference between the two).
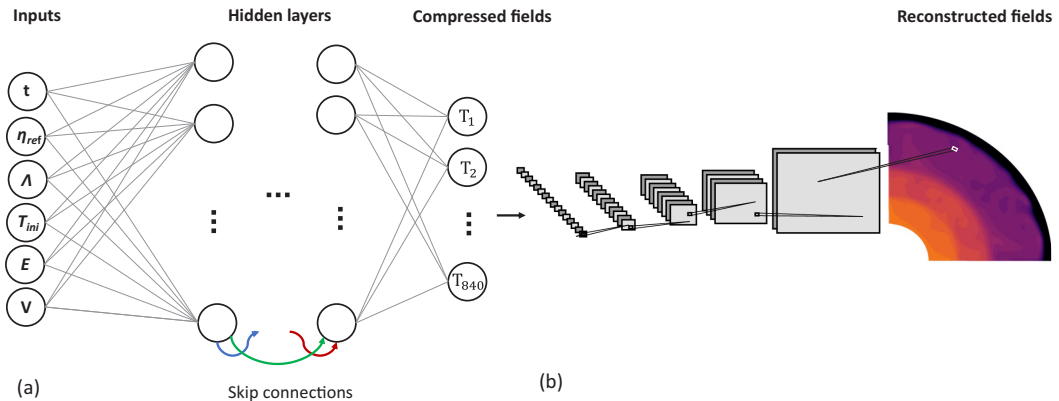
FIG. 5. (a) The five parameters governing mantle convection and time are used to predict the compressed temperature fields. Skip connections are used to add the output of each hidden layer after activation to that of each of the following hidden layers before activation. (b) After the training is complete, the trained decoder from ConvAE is used to reconstruct the temperature field back to its original dimensions.

IV. PREDICTION OF COMPRESSED TEMPERATURE FIELDS

A. Neural networks

With the temperature fields compressed, we now move on to predicting this latent space representation of 840 numbers, based on the five parameters of the simulation. Since Ref. [27] demonstrated that an FNN which takes the five parameters as inputs and time as a sixth input is capable of predicting the 1D temperature profile with a high accuracy, FNN is an obvious candidate for predicting the compressed temperature fields.

Figure 5(a) shows the five parameters and the time that are used as inputs to the FNN to predict the compressed temperature fields. For computational efficiency, we optimize the network weights using small minibatches of 16 temperature fields and Adam. Also, we used scaled exponential linear unit (SELU) as our activation function [58]:

$$\text{SELU}(x) = \lambda \begin{cases} x \\ \alpha e^x - \alpha \end{cases}, \tag{3}$$

as it seemed to deliver slightly better results than tanh. In Eq. (3), $\alpha = 1.67326324$ and $\lambda = 1.05070098$ are predefined and come from the original paper [58]. We further schedule the learning rate to decrease by a factor of 10 after 200 epochs and then again by a factor of 10 after the next 300 epochs. During the training, we save the network weights only if the validation loss drops. Furthermore, we employ a dropout of 5% after each hidden layer as a further regularization. Once the training has finished, the 2D fields are reconstructed from the predicted latent states as a postprocessing step using the already trained decoder.

We trained different FNN architectures with fully connected dense layers. The evolution of the mean-squared error (MSE) on the training and the cross-validation data is plotted in Fig. 6. The cross-validation loss over epochs converges to approximately the same value, especially for deeper networks. Since we also test some deep architectures such as five hidden layers with 800 neurons each and eight hidden layers with 400 neurons each, we use skip connections from each hidden layer to all the following hidden layers via addition. Given the challenging task of predicting a 840-dimensional vector, we use fairly deep architectures. Therefore, we employ SELU activation and skip connections to overcome the problem of vanishing gradients typically observed in deep networks.
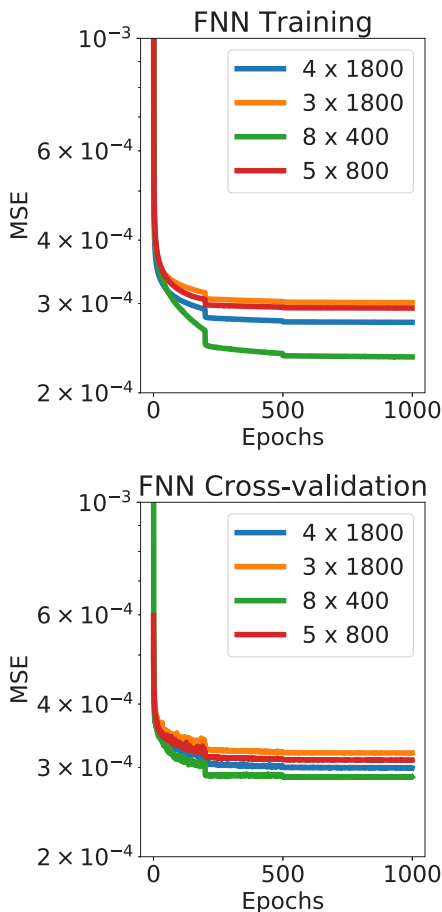
FIG. 6. The evolution of the mean-squared error (MSE) on (a) training data and on (b) cross-validation data for different FNN architectures. The legend shows the number of hidden layers as well as the number of neurons per layer of a given FNN architecture. For example, 4×1800 means the network has four hidden layers with 1800 neurons each. The steplike drop after 200 epochs is a result of the decrease in the learning rate.

For the FNN with eight hidden layers of 400 units each, we take two examples from the test set with different convection patterns and plot one characterized by a sluggish behavior in Fig. 7 and another one characterized by vigorous convection in Fig. 8. A schematic of the this particular FNN architecture is provided in Appendix C. The figures show the 2D temperature fields computed with GAIA (first column), those predicted by the FNN (second column), and the difference between the two (third column), as well as the horizontally averaged 1D temperature profiles along with their difference (fourth column). While the 1D profiles show good agreement (as previously demonstrated by Ref. [27]), the 2D predictions show more inaccuracies. In particular, cold, sublithospheric downwellings, which are a fundamental feature of the planform of stagnant-lid convection, tend to be completely lost by the FNN prediction. See the Supplemental Material [59] for animations of these two examples along with three further examples from the test set. In comparison with the GAIA simulations, the FNN predictions also fail to capture the vigor of convection. Even when the FNN captures a downwelling early on in the evolution, its lateral transport is not captured. On average, the 2D temperature fields predicted by FNN are 99.30% accurate (mean relative accuracy of dimensionalized temperature fields) with respect to GAIA and 99.35% with respect to ConvAE.
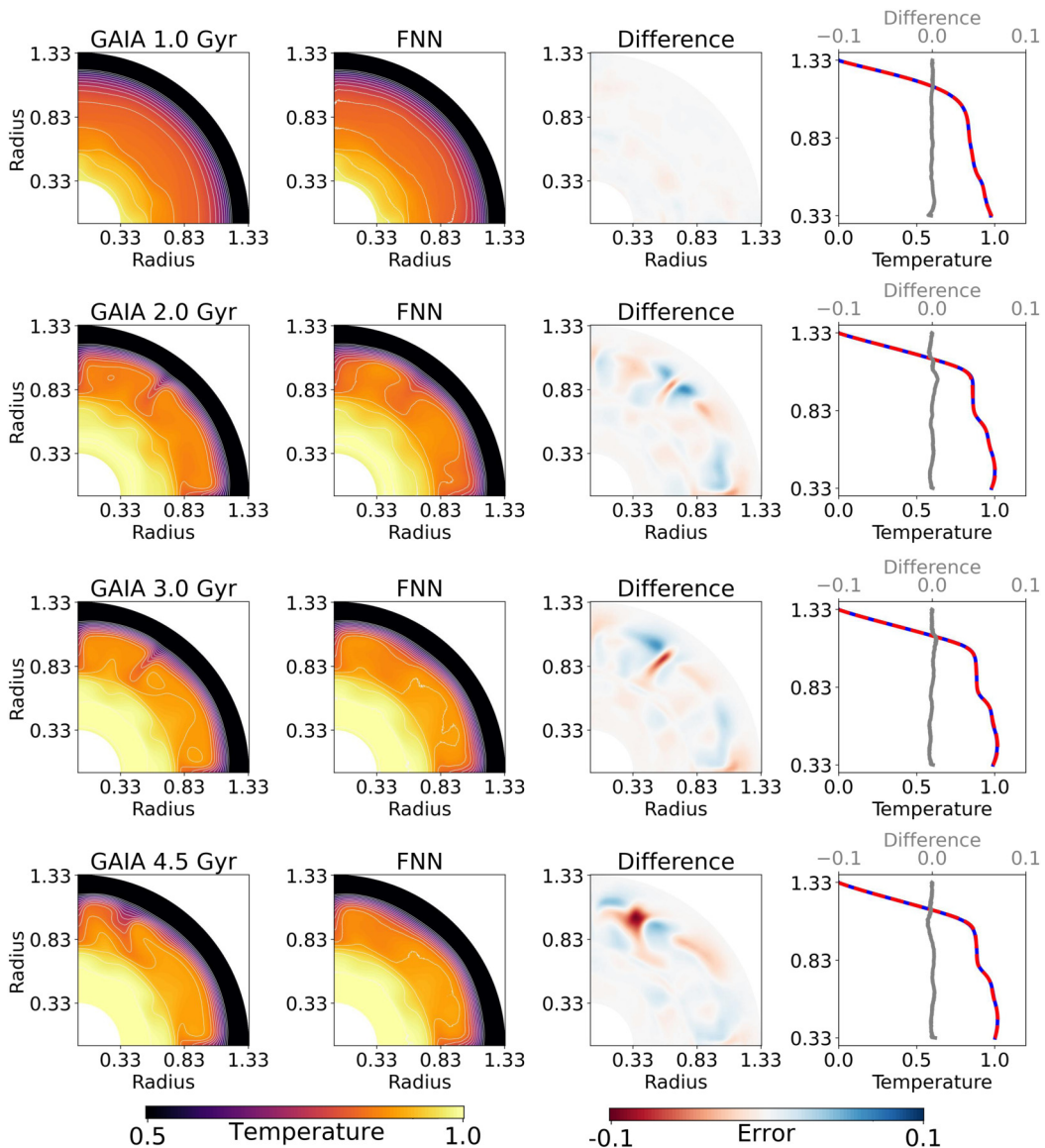
FIG. 7. Example of a sluggishly convecting mantle from the test set. The parameters are $\eta_{\mathrm{ref}} = 3.6 \times 10^{21}$ Pa S, $E = 1.6 \times 10^5$ J mol^{-1}, $V = 4.4 \times 10^{-6}$ m^3 mol^{-1}, $\Lambda = 15.3$, and $T_{\mathrm{ini}} = 1634$ K. The temperature field from GAIA and its equivalent FNN prediction are shown in columns 1 and 2, respectively. The third column shows the difference between the two. Column 4 shows the horizontally averaged 1D temperature profiles from GAIA(solid blue) and FNN (dashed red), as well as the difference between the two (gray).

B. Long short-term memory (LSTM)

The failure of the FNNs to capture lateral convection structures such as down- and upwellings can be attributed in large part to the fact that the temporal snapshots of any given simulation are disconnected. By treating time as an additional input variable and shuffling time steps of different simulations (but within the training/validation/test splits), the details of the dynamics of the flow are largely lost.
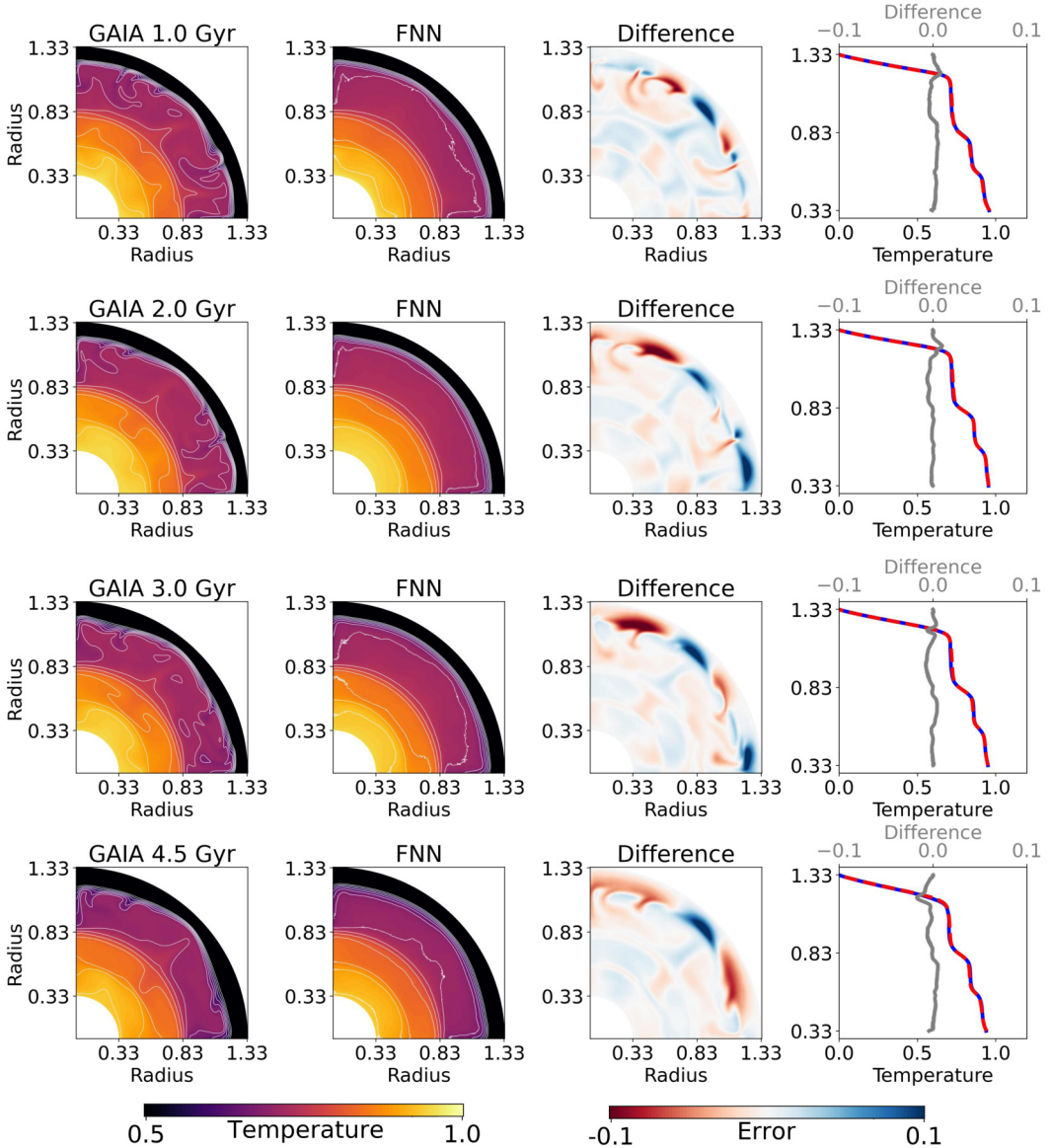
FIG. 8. Example of a vigorously convecting mantle from the test set. The parameters are $\eta_{\mathrm{ref}} = 5.0 \times 10^{19}$ Pa S, $E = 1.5 \times 10^5$ J mol^{-1}, $V = 7.6 \times 10^{-6}$ m^3 mol^{-1}, $\Lambda = 30.7$, and $T_{\mathrm{ini}} = 1705$ K. The temperature field from GAIA and its equivalent FNN prediction are shown in columns 1 and 2, respectively. The third column shows the difference between the two. Column 4 shows the horizontally averaged 1D temperature profiles from GAIA(solid blue) and FNN (dashed red), as well as the difference between the two (gray).

Hence, we turn our attention to recurrent neural networks that have been shown to be successful for a variety of Natural Language Processing tasks. Recurrent architectures such as LSTM [60] provide a back-propagation mechanism acting through a sequence of inputs (such as time steps of a simulation), thereby allowing the network to learn temporal dynamics [34,61].

As shown in Fig. 9(b), LSTMs differ from an FNN architecture [Fig. 9(a)] in that they use a set of previous time steps to predict the next time step. Furthermore, each LSTM cell [Fig. 9(c)] is more
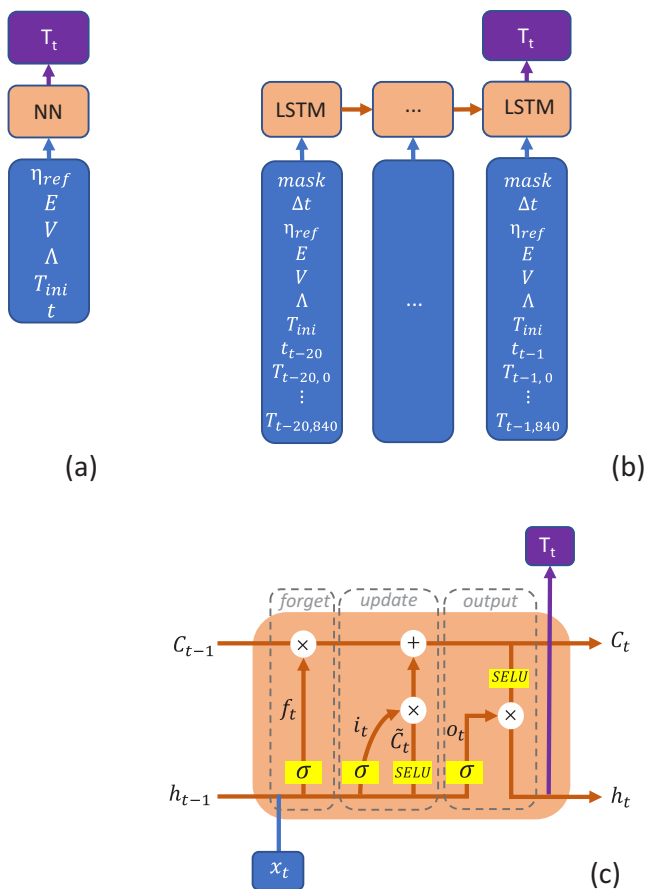
FIG. 9. Comparison of how (a) the FNNs are trained and how the (b) many-to-one LSTMs are trained for predicting compressed the temperature field T_t. (c) Each LSTM cell receives as input the compressed temperature field at time $t-1$ along with other parameters: a mask for whether the next time step exists, difference between the time step used as input and the one being predicted (Δt), the five mantle convection parameters (η_{ref}, E, V, Λ and T_{ini}), as well as the time itself t_{t-1} of the input compressed temperature field. In practice, the mask is used by the data generator to decide whether to provide the next time step as output to train the LSTM on.

elaborate than a simple neuron of a fully connected FNN. Reference [62] provides an accessible explanation of LSTMs. A brief description of an LSTM cell is provided in Appendix D.

As before, we use Keras API for setting up the forward graph [Eqs. (D1)–(D5)] and optimizing the trainable parameters by minimizing the MSE between the prediction and the target. Since the time steps for simulations were stored after a specified number of time steps of the numerical solver, as well as at every physical time interval of 100 million years of planetary evolution, we ended up with nonuniform time series. Hence, we use "masking," as done, for example, in Ref. [63], to specify if there is a time step to predict (mask = 1) or not (mask = 0). Most of the simulations have fewer than 200 time steps, although nine simulations exceed 400 time steps.

In addition to the mask, we also input the size of the time step (Δt), the five model parameters, time, and the compressed temperature field at the current time (t). After some trial and error, we found 20 time steps to serve as a rich enough input to predict the 21st. When predicting a time step when 20 previous inputs are not available, say, for time step 10, we took time steps 0 through 9

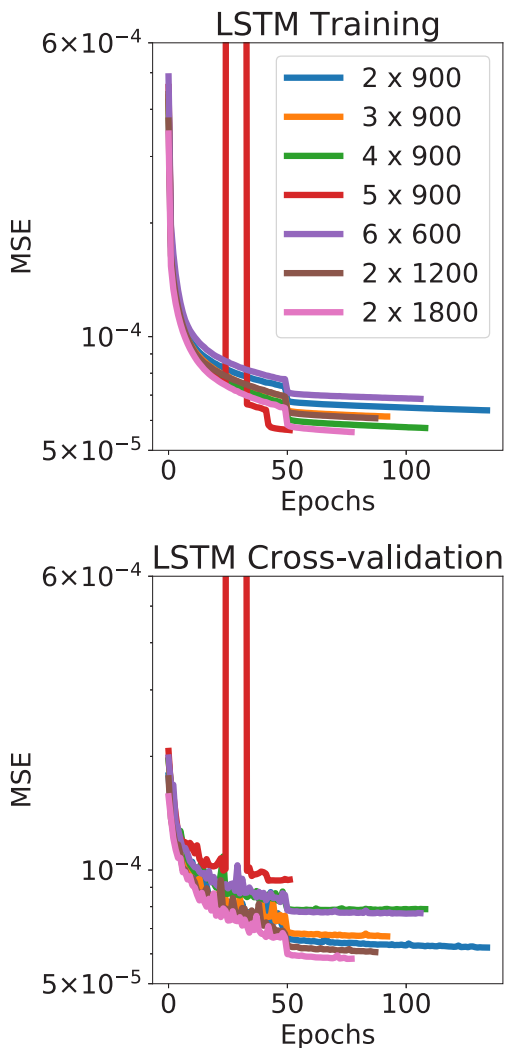## LSTM Training

LSTM Cross-validation

FIG. 10. The evolution of MSE on (top panel) training data and (bottom panel) cross-validation data for different LSTM architectures. The legend shows the number of hidden layers as well as the number of LSTM cells per layer. For example, 4×900 means the network has four hidden layers with 900 cells each. The steplike drop after 50 epochs is a result of the decrease in the learning rate.

and filled the rest with time step 0. In this way, the thermal evolution of a planet can be simulated based purely on input parameters. The input shape of each minibatch is thus (simulations $=16$, time steps $= 20$, input $= 848$), and the output shape is (simulations $= 16$, time steps $= 1$, output $= 840$).

The four different gates with two sets of matrices each in an LSTM cell mean that there are eight times as many trainable parameters per hidden layer as a regular dense layer in an FNN. Therefore, we tested a limited number of LSTM architectures as shown in Fig. 10. For computational efficiency, we use SELU again, instead of tanh as activation function [64]. Such models take about 2 wk on a Tesla V100 GPU to reach asymptotic loss values. Just as in the case of training FNNs, we use the following strategies to prevent overfitting: (1) storing the weights only if the validation loss drops, (2) using a dropout of 5% for each hidden layer, and (3) training on minibatches of 16.
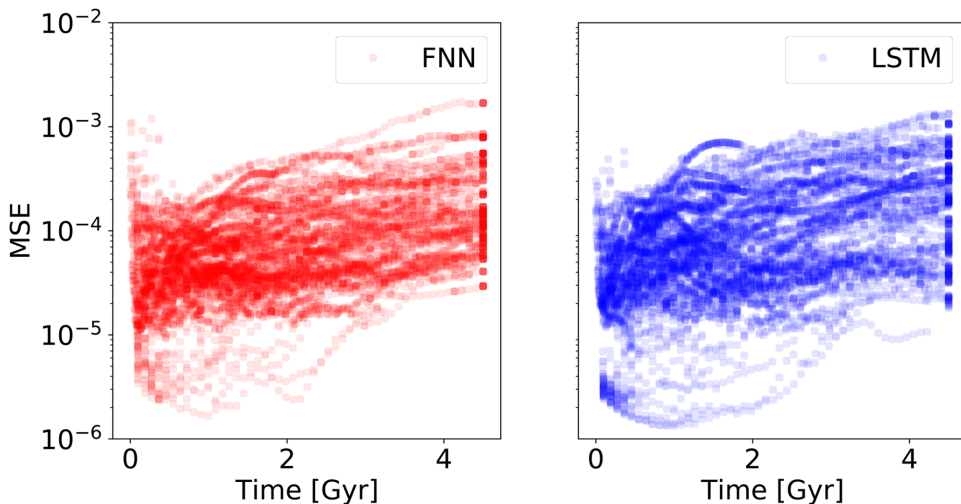
FIG. 11. Spatially averaged MSE for FNN (8×400) and LSTM (4×600) vs physical time for each simulation in the test set.

The difference between the loss curves for different architectures is small and, given the stochasticity associated with training of LSTMs, not particularly enlightening with one small exception. The loss curve for the LSTM with five hidden layers of 900 cells each diverged around epoch 20, but seems to have found its way back a few epochs later, indicating that such a deep architecture with roughly 33 million trainable parameters might be prone to exploding gradients, especially when the learning rate is too large (0.0001). Of course, the initial learning rate is divided by a factor of 10 after the first 50 epochs and then by another factor of 10 after the next 150 epochs (if reached).

LSTM architectures achieve a lower MSE loss (Fig. 10) than FNN architectures (Fig. 6). However, it is important to note that the MSE loss displayed for LSTMs during training is based on a prediction which takes the highly accurate compressed temperature fields as inputs. In inference mode, as the network iteratively predicts the next time step based on the previously predicted 20 time steps, this accuracy might decrease due to accumulation of error. Therefore, we calculated the mean relative accuracy for all the simulations in the cross-validation set using different LSTMs in purely inference mode and display them along with the mean relative accuracy on the test set in Table I. Table I seems to contradict the results of the MSE plots in Fig. 10. While during training, the [1800, 1800] LSTM with 46.5 million trainable parameters attained the lowest MSE, in inference mode it had the third lowest mean relative accuracy of all architectures. In contrast, the [600, 600, 600, 600] LSTM with 12.7 million trainable parameters achieved a higher accuracy on the cross-validation set and on the test set. We simply do not have enough simulations to fit 46.5 million weights without overfitting, which is evident from the high standard deviation of absolute relative accuracy on test and cross-validation sets. In inference mode, the errors in predicting time steps can accumulate to the point where a simulation diverges. Luckily, this behavior is not observed in the [600, 600, 600, 600] LSTM, for example. We present the evolution of the averaged MSE with time for different simulations in the test set, for both the FNN and LSTM, in Fig. 11. As the mantle typically starts cooling after some point in the thermal evolution, the convection should get slightly less vigorous. This means that the upwellings and downwellings should have a longer wavelength and, therefore, become slightly easier to predict. Nevertheless, it seems that the lack of data towards the end of the evolution (from unfinished simulations) is the reason for the increased error. In case of LSTMs it can be further exacerbated by the accumulation of error.
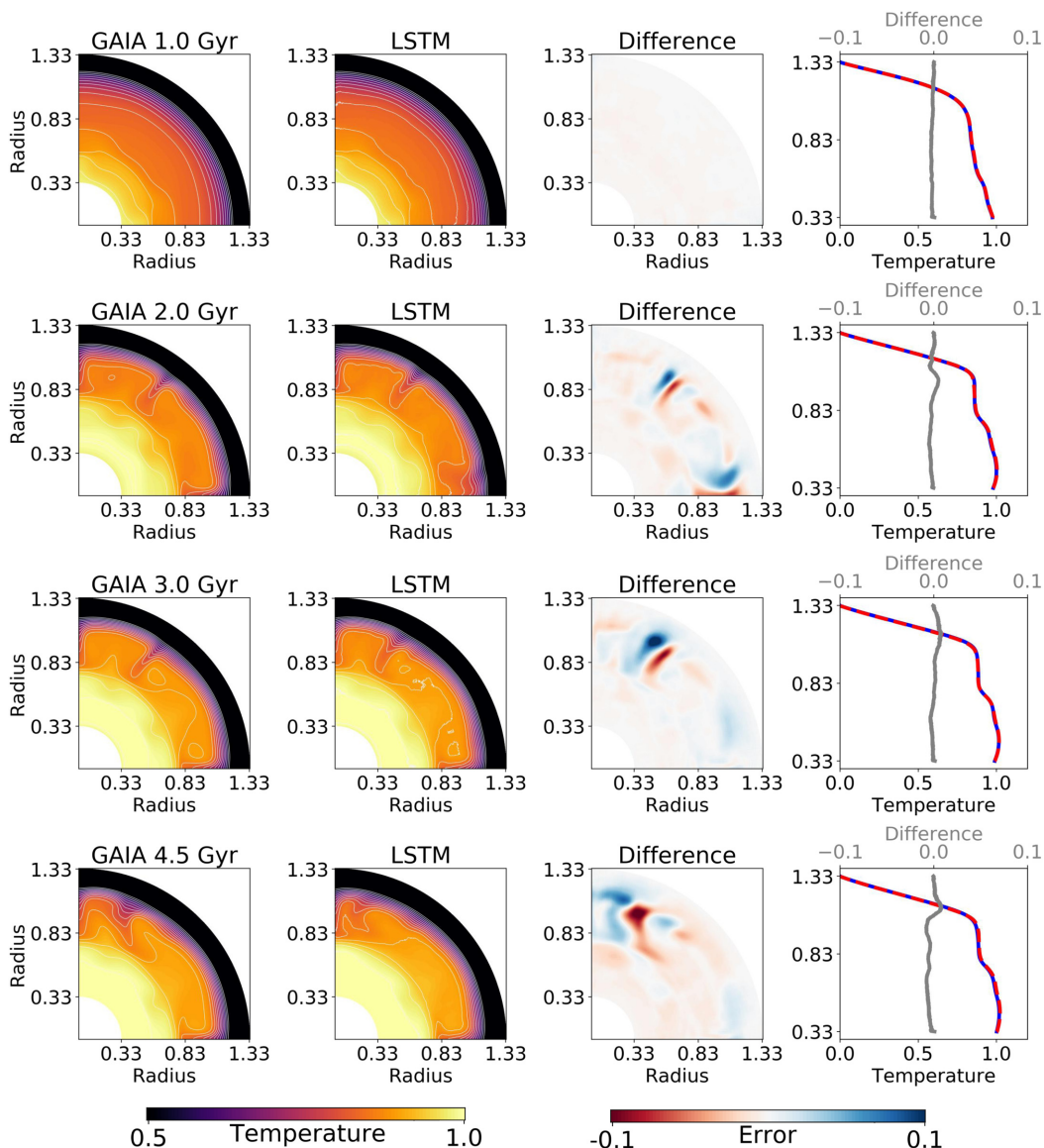
FIG. 12. Example 1 from the test set. The temperature field from GAIA and its equivalent LSTM prediction are shown in columns 1 and 2, respectively. The third column shows the difference between the two. Column 4 shows the horizontally averaged 1D temperature profiles from GAIA (solid blue) and LSTM (dashed red), as well as the difference between the two (gray).

For both the LSTM and the FNN, the error tends to increase marginally with time. This can be attributed in large part to the fact that data get sparser with increasing time as not all simulations finished. For the LSTM though, the accumulation of error can be another contributing factor.

In Fig. 12 and Fig. 13, we plot the same two examples from the test set as in Sec. IV A, but using the [600, 600, 600, 600] LSTM this time. Despite the slightly lower mean relative accuracy compared to FNN, the LSTMs do a better job of capturing the convection structures. This is especially true for the more sluggish simulations such as the one in Fig. 12, where the downwelling
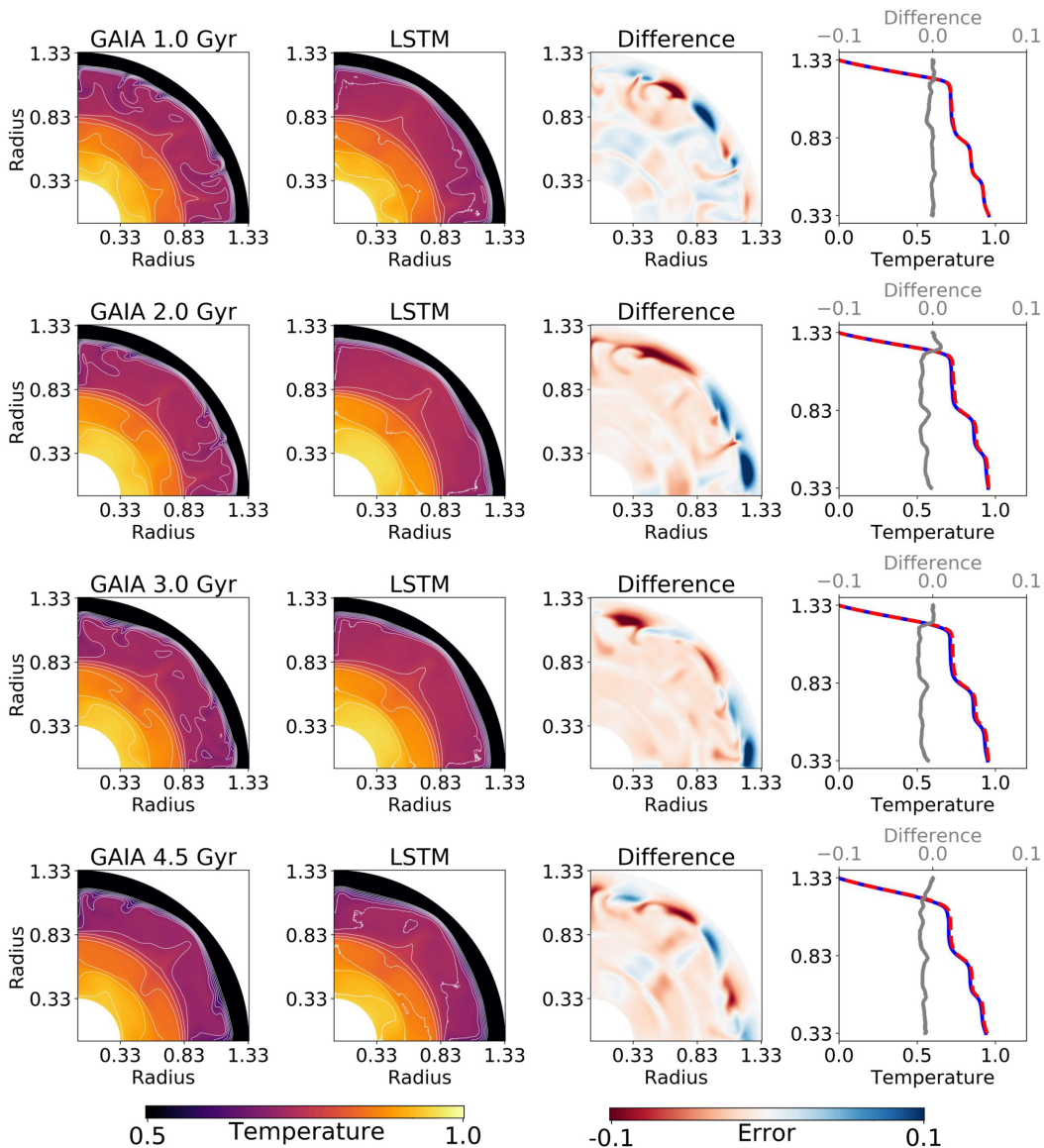
FIG. 13. Example 2 from the test set. The temperature field from GAIA and its equivalent LSTM prediction are shown in columns 1 and 2, respectively. The third column shows the difference between the two. Column 4 shows the horizontally averaged 1D temperature profiles from GAIA (solid blue) and LSTM (dashed red), as well as the difference between the two (gray).

is not only formed at 2 Gyr (second row), but also maintained and transported towards the left boundary in time, unlike the FNN.

Figure 13 is a good example of how the small-scale downwellings formed under more vigorous convection are not well captured by the LSTM. This explains the richness of structures one can see from the difference plots in column 3. Nevertheless, the big downwelling captured at 1 Gyr to the right of the domain at radius of 0.83 to 1.1, for example, or the upwelling at same radial location, but towards the middle of the domain present an improvement over the smudged-out prediction of the same simulation by an FNN in Fig. 8.

In summary, LSTMs are better at predicting sharper structures such as downwellings as well as the dynamics of their transport. A reason for the lower relative mean accuracy compared to FNNs is that the movement of plumes and downwellings, while captured, can be longitudinally off. In other words, the downwelling captured in Fig. 12 (rows 2 and 3) is slightly shifted in the angular direction. The same can be seen in all the difference plots of Fig. 13, at radial locations of 0.5–0.83 for a downwelling and 0.83–1.2 for a plume in the longitudinal center.

One could further examine the 1D temperature profiles, obtained by horizontally averaging the 2D temperature fields (column 4 in Fig. 7, Fig. 8, Fig. 12, and Fig. 13). LSTM temperature profiles have a mean relative absolute accuracy of 99.42%, while those of FNN are 99.71%. While this intuitively counters the longitudinal shift argument, the mean of smudged-out/lacking plumes and downwellings predicted by the FNN can match the mean of sharp plumes and downwellings of GAIA better than the LSTM. The fact that the LSTM temperature fields capture some, but not all, the downwellings in cases of vigorous convection (Fig. 13) can throw the horizontal mean off. In fact, finding an error metric that is invariant to, for example, longitudinal shift of a downwelling is nontrivial. Reference [65], for example, showed how modified versions of normalized root-mean-squared (NRMSE) can be computed that are invariant to certain effects such as multiplication by a constant or phase shift for image reconstruction. Since, from a planetary evolution viewpoint, the magnitude of the temperature field matters, one could attempt to find similar shift- or rotation-invariant metrics but in terms of MSE, instead of NRMSE. Furthermore, one must consider whether to use the modified MSE expression to only evaluate the error, or also to optimize the weights of the machine learning architectures. In the next subsection, we show that MSE seems capable of learning some nontrivial dynamics of mantle convection, as long as the underlying machine learning algorithm is suitable.

The mean relative accuracy for all the simulations in the test set is provided in Fig. 14. Low accuracy seems to be slightly correlated to low reference viscosity and low activation energy for the diffusion creep. A low reference viscosity generally leads to more vigorous convection, thereby inducing small-scale convection structures, which the LSTM finds difficult to predict. Similarly, a low activation energy or, equivalently, a low dependence of viscosity on temperature has the same qualitative effect of reducing viscosity. Otherwise, the method works well across the entire range of parameters.

C. POD comparison of FNN and LSTM predictions

We analyze the FNN and LSTM predictions from the lens of POD (proper orthogonal decomposition). Following e.g., Ref. [66], we compute the singular value decomposition of a "tall" simulation matrix $X \in \mathbb{R}^{p \times q}$ (spatial points $\times$ time steps),

$$X = U \Sigma V^*, \tag{4}$$

to obtain the spatial modes $U \in \mathbb{R}^{p \times r}$, complex conjugate V^* of temporal modes $V \in \mathbb{R}^{r \times q}$, and the POD coefficients (eigenvalues) $\Sigma \in \mathbb{R}^{r \times r}$, where r is determined by the minimum of p and q. In Fig. 15(a) we plot the eigenvalues for the simulation in Fig. 12 and Fig. 7, and in Fig. 15(b) we display the eigenvalues for the simulation in Fig. 13 and Fig. 8.

As can be seen in both cases in Fig. 15, the eigenvalues of an FNN-predicted temperature field evolution decay very rapidly after the first three to five modes. Hence, the cumulative distribution function (CDF) of the FNN predictions is the steepest, reaching most of its energy within the first few modes, as opposed to the other CDFs, where latter modes also carry non-negligible energy. This phenomenon can be visually observed in the animations of all the five test-case simulations. See the Supplemental Material [59] for the animations. The FNN predictions are not "energetic" enough. The POD coefficients of the simulations predicted by LSTM decay less rapidly, even when, occasionally, the decay is desired [Fig. 15(a), modes 40–76]. However, in the case of vigorous convection, Fig. 15(b) shows that LSTMs, while better than FNNs, still do not fully capture the energy characteristics of the GAIA simulation. On average, the sum of eigenvalues of the FNN
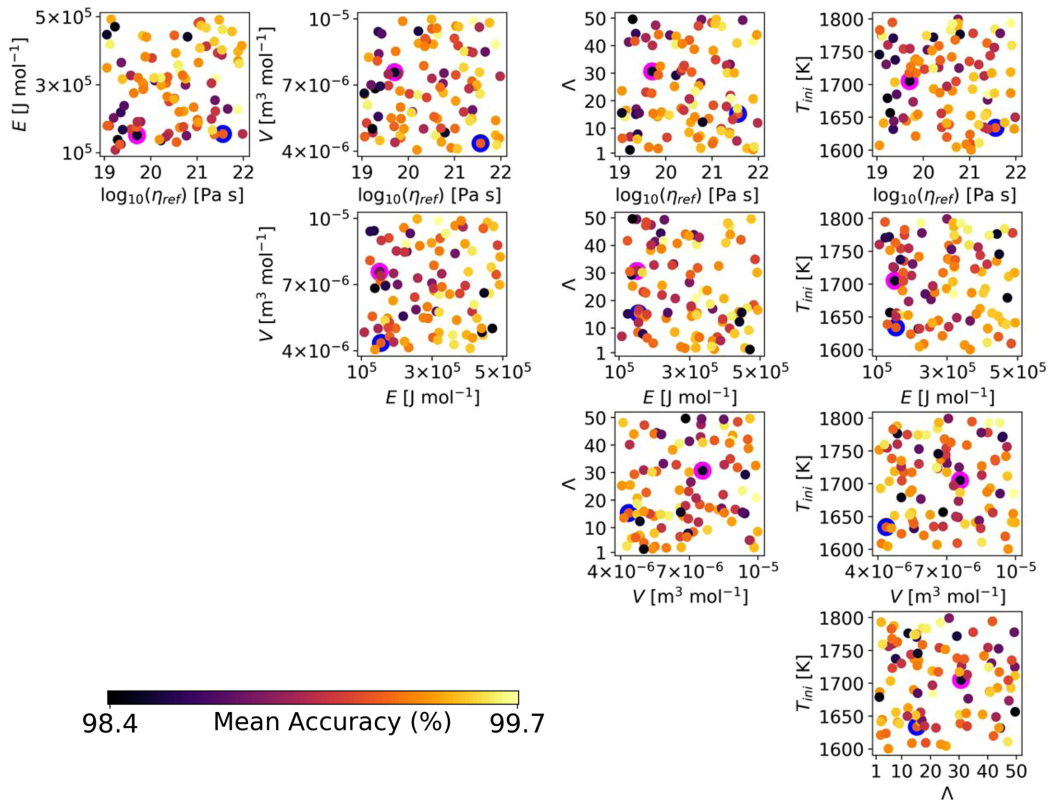
FIG. 14. Mean relative accuracy of LSTM predictions of the temperature fields for all the simulations in the test set with respect to the original GAIA simulations, expressed as a percentage. The mean relative accuracy is plotted with respect to two parameters at a time, indicated on the *x*- and *y*-axis with the units in which the parameters are measured. Example in Fig. 12 is circled in blue, and the example in Fig. 13 is circled in magenta.

predictions on the entire test set amounts to 96.51% of the the sum of eigenvalues of the GAIA predictions. For LSTM-predicted simulations, the sum of POD coefficients increases to 97.66% relative to those of GAIA simulations. Thus, LSTMs capture the dynamics of the simulations better, while FNNs provide a marginally better mean accuracy for snapshots.

V. CONCLUSION AND FUTURE WORK

We use deep learning techniques to model parameterized surrogates of mantle convection simulations. The data to train the algorithms come from 10 525 mantle convection simulations of a Mars-like planet, run on a 2D quarter-cylindrical grid. Focusing on only one state variable in this study, temperature, we first compressed the temperature fields using convolutional autoencoders by a factor of 142 (from 1 TB to 7 GB) and then tested two regression algorithms for predicting the compressed temperature fields given five key parameters: reference viscosity (linked to the Rayleigh number), activation energy and activation volume of the diffusion creep, an enrichment factor for radiogenic elements in the crust, and the initial mantle temperature [see Fig. 1(a)].

We found that while feedforward neural networks (FNNs) offer a reasonable mean accuracy, the sharper plumes and downwellings formed in the upper mantle are never transported laterally and are often lost early in the evolution. In contrast, many-to-one long-short-term memory networks (LSTM) were able to capture the sharper convection structures along with their lateral transport more often, but ultimately delivered a slightly lower mean accuracy (99.22%) in comparison to
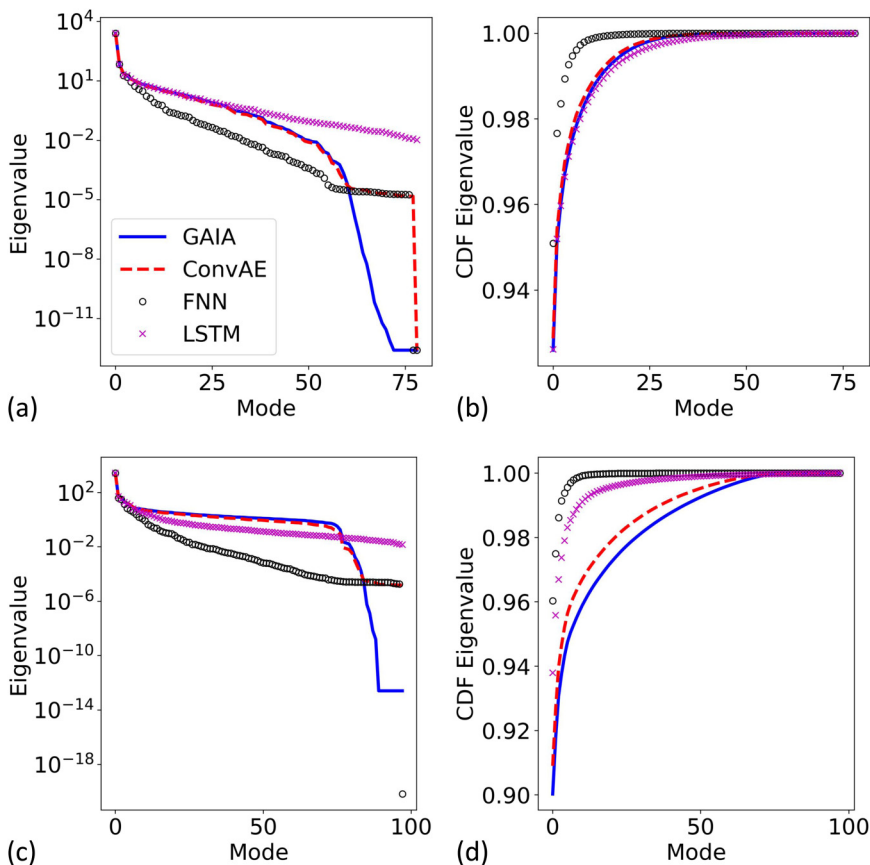
FIG. 15. (a) POD coefficients and (b) their cumulative distribution for example simulation 1 in the test set, and (c) and (d) correspond to example simulation 2.

FNNs (99.30%). Two factors are mainly responsible for the lower mean accuracy: (1) the convection structures were longitudinally shifted and (2) the prediction error can accumulate in time. Despite this, the eigenvalues obtained through proper orthogonal decomposition (POD) show that the FNN predictions decay too rapidly after the first three to five modes, while LSTM predictions decay less rapidly and hence capture the flow dynamics more accurately than the FNN, if not perfectly. When summed, the eigenvalues from FNN predictions and the eigenvalues from LSTM predictions amount to 96.51% and 97.66% relative to those obtained through POD of the original simulations, respectively.

This study serves as a first-proof that deep learning can be used to model high-dimensional parameterized surrogates of mantle convection simulations. Given five parameters, the complete spatiotemporal evolution of the temperature field can be predicted up to a reasonable accuracy, i.e., the longer wavelength structures such as the 1D temperature profile and larger plumes and downwellings as well as their lateral transport can be captured, albeit not perfectly. With respect to the thermal evolution of terrestrial planets, the 2D temperature fields can be used to calculate a number of fields of interest and relate them to various quantities that can be inferred from actual observations. Lateral variations in the heat flux are important for estimating the elastic lithospheric thickness (e.g., [52]). Spatiotemporal variations in the heat flux at the core-mantle boundary affect the generation and morphology of the magnetic field [67]. The formation of plumes and downwellings is important for calculating the amount of melt produced during the thermal

evolution and to relate this to estimates of the thickness of the crust [68]. The fact that plumes and downwellings are not accurately captured by the FNN can impact local melt production. Similarly, a LSTM's longitudinally shifted plumes compared to the true simulations can result in slightly different looking crustal distributions. It is not straightforward to predict how consequential these errors would be to constraining the parameters. Ideally, one would conduct an inverse study to test the sensitivity of uncertainties in the observables resulting from instrumentation and/or from the surrogate model [13].

Parameterized surrogates such as the ones presented in this paper or the one proposed in Ref. [37] are primarily useful for performing parameter-studies—be it placing constraints on the evolution of a planet like Mars, or optimizing an airfoil to achieve the target aerodynamic performance.

This is fundamentally different from applications where the time steps of the same simulation can be split into training and test sets [32–34]. On the one hand, the latter is an easier learning task because one can expect the dynamics of a single simulation to exist on a significantly smaller manifold than multiple simulations with a wide range of parameters. On the other hand, the simplicity of the flow in our simulations (e.g., no turbulence, limited compressibility, and 2D flow instead of 3D) begs the question if parameterized surrogates can be learned for more complex flows than the one used in this study.

Particularly important would be the computational cost of generating the data. For flows similar to those considered here, but in three dimensions, running 10 000 simulations would be intractable. Worse yet, 10 000 simulations could be an order of magnitude less than needed to learn spatiotemporal dynamics in three dimensions. This is especially important in the light of the fact that while the 2D convection models provide significantly more information than 0D [18–25] or 1D evolution models [27], they still cannot be used to constrain parameters based on localized observational constraints in three dimensions such as crustal thickness, elastic lithospheric thickness, or surface heat flux. Hence, this study serves only as a stepping stone to the ultimate goal of using high-dimensional forward surrogates for probabilistic inversion of mantle convection parameters. To reach this goal, further research is needed into data-efficient methods for parameterized surrogate modeling of 3D mantle convection. One or both of the following approaches might hold the key. (1) One could use the partial differential equations and boundary conditions governing mantle convection to place soft and/or hard constraints on the optimization problem at hand [33,69–71]. (2) Although POD bases are not appropriate for this problem because they do not generalize well among simulations with different parameters, the idea of finding a set of basis functions, that one needs only to learn the coefficients to, remains an attractive one [72–74].

Despite these challenges, the combination of scientific deep learning and modern supercomputing hardware presents an unprecedented opportunity for gaining robust insights into geophysical flows.

A JupyterNotebook to predict the entire spatiotemporal evolution of the 2D temperature field from five parameters is available at Ref. [75].

D.B., G.M.; *Visualization*: S.A., N.T.; *Supervision*: N.T., D.B., P.K., G.M.; *Funding acquisition*: N.T., D.B., G.M.

APPENDIX A: MATHEMATICAL MODEL OF MANTLE CONVECTION SIMULATIONS

The mathematical model used to run the simulations is the same as the one presented in Ref. [27]. Yet, for completeness, we describe it in detail in this Appendix. Numerical values of the model parameters that are shared by all simulations are listed at the end of the Appendix in Table II and Table III.

1. Governing equations

The equations of conservation of mass, linear momentum, and thermal energy under the extended Boussinesq approximation can be written in nondimensional form as follows [40,77]:

$$\nabla' \cdot \mathbf{u}' = 0, \tag{A1}$$

$$-\nabla' p' + \nabla' \cdot [\eta'(\nabla'\mathbf{u}' + (\nabla'\mathbf{u}')^{\mathrm{T}})] + \left(\mathrm{Ra}\,\alpha'\,T' - \sum_{l=1}^{3} \mathrm{Rb}_l\,\Gamma_l\right)\mathbf{e}_r = 0, \tag{A2}$$

$$\frac{DT'}{Dt'} - \nabla' \cdot (k'\nabla'T') - \mathrm{Di}\,\alpha'(T' + T_0')u_r' - \frac{\mathrm{Di}}{\mathrm{Ra}}\Phi'$$
$$- \sum_{l=1}^{3} \mathrm{Di}\frac{\mathrm{Rb}_l}{\mathrm{Ra}}\frac{D\Gamma_l}{Dt}\gamma_l(T' + T_0') - \frac{\mathrm{Ra}_Q}{\mathrm{Ra}} = 0. \tag{A3}$$

Primed quantities are nondimensional. $\mathbf{u}'$ is the velocity vector, p' the dynamic pressure, η' the viscosity, Ra the thermal Rayleigh number, α' the thermal expansivity, T' the temperature, Rb_l the Rayleigh number associated with the boundary of the lth phase transition, Γ_l the corresponding phase function ([44], $\mathbf{e}_r$ the unit vector in the radial direction, t' the time, k' the thermal conductivity, Di the dissipation number, T_0' the surface temperature, u_r' the radial component of the velocity, Φ' the viscous dissipation, and Ra_Q the Rayleigh number for internal heating. Viscosity, thermal expansivity, and thermal conductivity are functions of pressure and temperature [see Eq. (1) and Sec. A 4].

2. Nondimensionalization of state variables

In Eqs. (A1)–(A3), the dimensional variables are scaled as follows:

$$\mathbf{u}' = \mathbf{u}\frac{\rho_m c_{p_m} D}{k_{\mathrm{ref}}}, \tag{A4}$$

$$p' = p\frac{\rho_m c_{p_m} D^2}{\eta_{\mathrm{ref}} k_{\mathrm{ref}}}, \tag{A5}$$

$$t' = t\frac{k_{\mathrm{ref}}}{\rho_m c_{p_m} D^2}, \tag{A6}$$

$$T' = \frac{T - T_0}{\Delta T}. \tag{A7}$$

In Eqs. (A4)–(A7), ρ_m is the mantle density and c_{p_m} its heat capacity; $D = R_p - R_c$ is the mantle thickness (R_p and R_c are the planet and core radius, respectively); k_{ref} is the reference thermal conductivity; η_{ref} the reference viscosity [Eq. (1)]; and ΔT the initial temperature drop across the mantle.

3. Dimensionless quantities

The following nondimensional numbers appear in Eqs. (A1)–(A3):

$$\text{Ra} = \frac{\rho_m^2 c_{p_m} \alpha_{\text{ref}} g \Delta T D^3}{\eta_{\text{ref}} k_{\text{ref}}}, \tag{A8}$$

$$\text{Rb}_l = \frac{\rho_m c_{p_m} \Delta \rho_l g D^3}{\eta_{\text{ref}} k_{\text{ref}}}, \tag{A9}$$

$$\text{Ra}_Q = \frac{\rho_m^3 c_{p_m} \alpha_{\text{ref}} g H_0 D^5}{\eta_{\text{ref}} k_{\text{ref}}^2}, \tag{A10}$$

and

$$\text{Di} = \frac{\alpha_{\text{ref}} g D}{c_{p_m}}, \tag{A11}$$

where, α_{ref} and k_{ref} are the reference thermal expansivity and conductivity, respectively, g is the gravitational acceleration, $\Delta \rho_l$ is the density contrast across the lth phase transition, and H_0 is the initial rate of mantle heat production due to radiogenic elements.

4. Thermal expansion and conductivity

The temperature- and pressure-dependent thermal expansivity and conductivity are calculated using the parametrizations introduced in Ref. [45], which in dimensional form read

$$\alpha(T, P) = (a_0 + a_1 T + a_2 T^{-2}) \exp(-a_3 P), \tag{A12}$$

$$k(T, P) = (c_0 + c_1 P) \left(\frac{300}{T} \right)^{c_2}. \tag{A13}$$

Here $a_0, \ldots, a_3$ and $c_0, \ldots, c_2$ are coefficients based on experimental data valid for Mg-rich olivine. Numerical values of these coefficients are listed in Table III.

5. Phase transitions

We consider two solid-solid phase transitions in the olivine system, α- to β-spinel and β- to γ-spinel [44]. The temperature-dependent depth of the lth phase boundaries $z_l(T)$ is calculated as

$$z_l(T) = z_l^0 + \gamma_l (T - T_l^0). \tag{A14}$$

Here γ_l is the Clapeyron slope (positive for both phase transitions), z_l^0 is the reference transition depth, and T_l^0 the corresponding reference temperature. The phase-transition function Γ_l is expressed in terms of z_l and phase-transition width d_l as

$$\Gamma_l = \frac{1}{2} \left[1 + \tanh \left(\frac{z - z_l(T)}{d_l} \right) \right]. \tag{A15}$$

6. Depletion of heat-producing elements and partial melting

We assume that a crust of thickness d_{cr} formed early in the planet evolution [41] and that this event led to the extraction of a large amount of radiogenic elements from the mantle [30]. We use an enrichment factor Λ to modify the bulk abundance of heat-producing elements C_0 in the mantle (based on Ref. [42]) to a new depleted composition C_{depleted} as follows:

$$C_{\text{depleted}} = \frac{M_m C_0}{M_{\text{cr}}(\Lambda - 1) + M_m}, \tag{A16}$$

where M_m and M_{cr} are the mass of the mantle and crust, respectively.

We model partial melting following the approach of Ref. [43]. Whenever the mantle temperature T locally exceeds the solidus, we calculate the melt fraction φ_i by solving the following equation:

$$c_p(T_i - T_{\text{sol}}) = L_{\text{m}}\varphi_i + c_p\Delta\varphi_i\Delta T_{\text{liq-sol}}(1 - \varphi_i), \qquad (A17)$$

where T_i is the temperature in the ith cell of the computational domain, T_{sol} the local solidus temperature, L_{m} the latent heat of melting, and $\Delta T_{\text{liq-sol}}$ the local difference between the liquidus and solidus temperature. For the solidus and liquidus, we use the parametrization of Refs. [78] and [79], respectively:

$$T_{\text{sol}} = e_0 + e_1 P + e_2 P^2 + e_3 P^3 + e_4 P^4, \qquad (A18)$$

$$T_{\text{liq}} = f_0 + f_1 P + f_2 P^2 + f_3 P^3 + f_4 P^4, \qquad (A19)$$

where T_{sol} and T_{liq} are the dimensional solidus and liquidus temperatures, respectively, P is the dimensional hydrostatic pressure, and $e_0, \ldots, e_4$ and $f_0, \ldots, f_4$ are numerical coefficients listed in Table III.

Using the sum of melt produced in all cells at time step t, φ_t, we adjust the internal heating Rayleigh number to model the further extraction of heat-producing elements due to melting:

$$\text{Ra}_{Q_t} = \text{Ra}_{Q_{t-1}}(1 - \Lambda\varphi_t). \qquad (A20)$$

7. Evolution of the core-mantle boundary temperature

An isothermal boundary condition is imposed at the core-mantle boundary whose temperature T_{c} evolves according to Ref. [18]:

$$c_{p_{\text{c}}}\rho_{\text{c}}V_{\text{c}}\frac{dT_{\text{c}}}{dt} = -q_{\text{c}}A_{\text{c}}. \qquad (A21)$$

Here $c_{p_{\text{c}}}$ is the specific heat-capacity of the core, V_{c} the volume of the core, q_{c} the average heat flux at the core-mantle boundary (CMB), and A_{c} the outer area of the core.

8. Rescaling of the core

As often done in simulations of mantle convection carried out in a 2D cylindrical shell geometry, a geometric rescaling of the core radius is applied in order to better reproduce the temperature field that would be obtained in a 3D spherical shell. Following Ref. [80], the radius of the core of the cylinder (R_c^{cyl}) is rescaled in such a way that the core-to-planet radius ratio is the same as the core-to-planet surface ratio of a sphere. In formulas,

$$\left(\frac{R_{\text{c}}}{R_{\text{p}}}\right)^2 = \frac{R_{\text{c}}^{\text{cyl}}}{R_{\text{p}}^{\text{cyl}}}, \quad R_{\text{p}}^{\text{cyl}} - R_{\text{c}}^{\text{cyl}} = 1, \qquad (A22)$$

where R_{p}, R_{c}, and $R_{\text{p}}^{\text{cyl}}$ are, respectively, the radii of the spherical planet, spherical core, and cylindrical planet.

APPENDIX B: DISTRIBUTION OF PARAMETERS IN THE DATA SET

Figure 16 plots the distribution of simulation parameters in the training, cross-validation, and test sets.

APPENDIX C: FNN SCHEMATIC

Figure 17 is a schematic of one of the FNN architectures that were trained and the one we present the results for in Fig. 7 and Fig. 8. The trained FNN has eight hidden layers with 400 neurons each, and each hidden layer is connected to all the following hidden layers via skip connections. For example, hidden layer 0 is connected to hidden layers 1 through 7. For ease of visualization, we show only three hidden layers in Fig. 17.
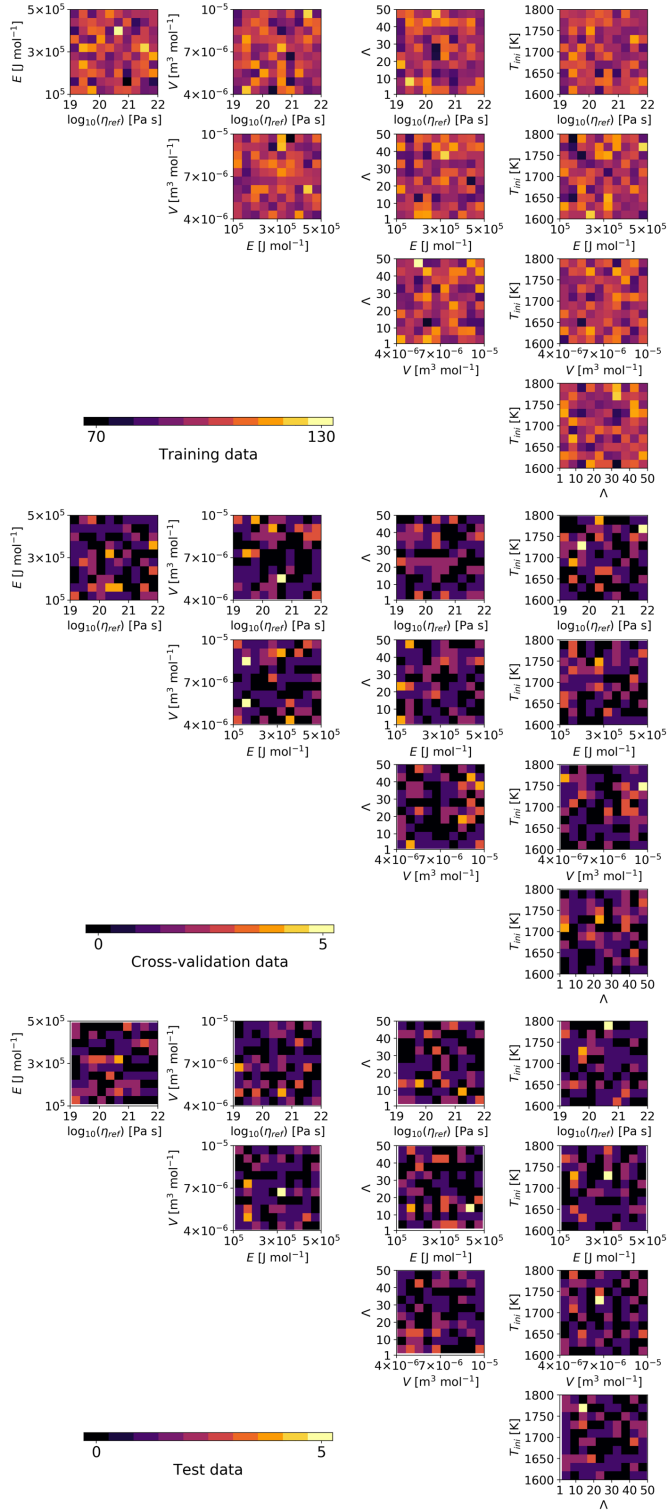
FIG. 16. Distribution of the simulation parameters in the training, cross-validation, and test sets.

input_11: InputLayer	input:	[(None, 6)]
	output:	[(None, 6)]

dense_24: Dense	input:	(None, 6)
	output:	(None, 400)

activation_13: Activation	input:	(None, 400)
	output:	(None, 400)

dropout_14: Dropout	input:	(None, 400)
	output:	(None, 400)

dense_25: Dense	input:	(None, 400)
	output:	(None, 400)

add_9: Add	input:	[(None, 400), (None, 400)]
	output:	(None, 400)

activation_14: Activation	input:	(None, 400)
	output:	(None, 400)

dropout_15: Dropout	input:	(None, 400)
	output:	(None, 400)

dense_26: Dense	input:	(None, 400)
	output:	(None, 400)

add_10: Add	input:	[(None, 400), (None, 400), (None, 400)]
	output:	(None, 400)

activation_15: Activation	input:	(None, 400)
	output:	(None, 400)

dropout_16: Dropout	input:	(None, 400)
	output:	(None, 400)

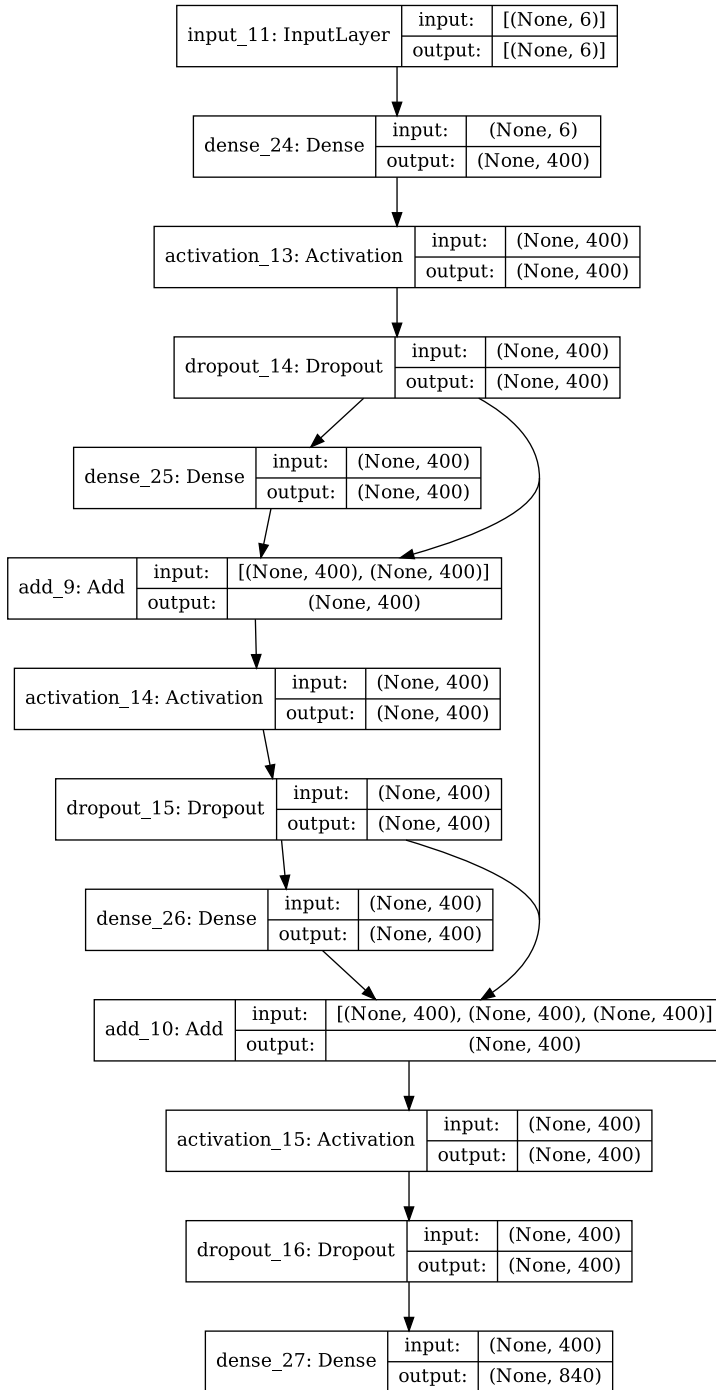| dense_27: Dense	input:	(None, 400)
	output:	(None, 840)

FIG. 17. A schematic of the FNN architecture with three hidden layers with 400 neurons each. For ease of visualization, we show only three hidden layers even though the trained FNN has eight.

APPENDIX D: EQUATIONS OF AN LSTM CELL

Referring to Fig. 9(c), an LSTM cell has three main blocs. The "forget" gate f_t determines how much information from the previous cell state C_{t-1} should be retained given the input vector x_t and the previous hidden state h_{t-1} (or equivalently, the previous output of an LSTM cell):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{D1}$$

where σ is the sigmoid activation $[\sigma(x) = (1/(1 + e^{-x}))]$, $W_f \in \mathbb{R}^{n \times m}$ is a matrix of trainable parameters, n is the number of LSTM cells, m is the size of the input vector x_t, $U_f \in \mathbb{R}^{n \times n}$ is another matrix of trainable parameters, and $b_f \in \mathbb{R}^n$ is a set of biases. The subscript f in W_f and b_f stands for the forget fate.

Then, in the "update" bloc, a sigmoid layer decides which values should be updated:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{D2}$$

while the SELU layer creates new values $\tilde{C}_t$ to be added to the state:

$$\tilde{C}_t = \text{SELU}(W_c x_t + U_c h_{t-1} + b_c). \tag{D3}$$

W_i and W_c are the weights for the input connections, where subscript i denotes the weights used to update values and subscript c denotes the weights used to create new values. Similarly, U_i and U_c are weights for the recurrent connections, and b_c and b_i are biases.

Using Eq. (D1)–(D3), we can now update the cell state C_t:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{D4}$$

where $\odot$ is an element-wise (Hadamard) product.

Finally, a last sigmoid layer decides the amount of cell state to be outputted via the dot product of output o_t with the SELU() of the cell state:

$$h_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \odot \text{SELU}(C_t), \tag{D5}$$

where W_o, U_o, and b_o are the final set of trainable input weights, recurrent weights, and output biases, respectively.

[1] D. Breuer and W. B. Moore, Dynamics and thermal history of the terrestrial planets, the moon, and Io, in *Treatise on Geophysics*, 2nd ed., edited by G. Schubert (Elsevier, Oxford, 2015), pp. 255–305.

[2] P. J. Tackley, Modelling compressible mantle convection with large viscosity contrasts in a three-dimensional spherical shell using the yin-yang grid, Phys. Earth Planet. Inter. **171**, 7 (2008).

[3] S. Zhong, A. McNamara, E. Tan, L. Moresi, and M. Gurnis, A benchmark study on mantle convection in a 3-D spherical shell using CitcomS, Geochem. Geophys. Geosys. **9**, Q10017 (2008).

[4] M. Kronbichler, T. Heister, and W. Bangerth, High accuracy mantle convection simulation through modern numerical methods, Geophys. J. Int. **191**, 12 (2012).

[5] C. Hüttig, N. Tosi, and W. Moore, An improved formulation of the incompressible navier-Stokes equations with variable viscosity, Phys. Earth Planet. Inter. **220**, 11 (2013).

[6] N. Tosi and S. Padovan, Mercury, Moon, Mars: Surface expressions of mantle convection and interior evolution on stagnant-lid bodies, in *Mantle Convection and Surface Expressions*, edited by H. Marquardt, M. Ballmer, S. Cottar, and J. Konter, AGU Monograph Series (American Geophysical Union, 2021), Chap. 17, 455–489..

[7] M. Sambridge, Geophysical inversion with a neighbourhood algorithm—I. Searching a parameter space, Geophys. J. Int. **138**, 479 (1999).

[8] M. Sambridge, Geophysical inversion with a neighbourhood algorithm—II. Appraising the ensemble, Geophys. J. Int. **138**, 727 (1999).

[9] U. Meier, A. Curtis, and J. Trampert, Global crustal thickness from neural network inversion of surface wave data, Geophys. J. Int. **169**, 706 (2007).

[10] P. Käufl, A. P. Valentine, R. W. de Wit, and J. Trampert, Solving probabilistic inverse problems rapidly with prior samples, Geophys. J. Int. **205**, 1710 (2016).

[11] R. W. L. de Wit, A. P. Valentine, and J. Trampert, Bayesian inference of Earth's radial seismic structure from body-wave travel times using neural networks, Geophys. J. Int. **195**, 408 (2013).

[12] S. Atkins, A. P. Valentine, P. J. Tackley, and J. Trampert, Using pattern recognition to infer parameters governing mantle convection, Phys. Earth Planet. Inter. **257**, 171 (2016).

[13] S. Agarwal, N. Tosi, P. Kessel, S. Padovan, D. Breuer, and G. Montavon, Towards constraining Mars' thermal evolution using machine learning, Earth Space Sci. **8**, e2020EA001484 (2021).

[14] C. Reese, V. Solomatov, and L.-N. Moresi, Heat transport efficiency for stagnant lid convection with dislocation viscosity: Application to Mars and Venus, J. Geophys. Res. Planets **103**, 13643 (1998).

[15] C. Dumoulin, M.-P. Doin, and L. Fleitout, Heat transport in stagnant lid convection with temperature- and pressure-dependent Newtonian or non-Newtonian rheology, J. Geophys. Res. **104**, 12759 (1999).

[16] V. S. Solomatov and L.-N. Moresi, Scaling of time-dependent stagnant lid convection: Application to small-scale convection on Earth and other terrestrial planets, J. Geophys. Res. **105**, 21795 (2000).

[17] F. Deschamps and C. Sotin, Thermal convection in the outer shell of large icy satellites, J. Geophys. Res. Planets **106**, 5107 (2001).

[18] D. J. Stevenson, T. Spohn, and G. Schubert, Magnetism and thermal evolution of the terrestrial planets, Icarus **54**, 466 (1983).

[19] M. Gurnis, A reassessment of the heat transport by variable viscosity convection with plates and lids, Geophys. Res. Lett. **16**, 179 (1989).

[20] G. Schubert and T. Spohn, Thermal history of Mars and the sulfur content of its core, J. Geophys. Res.: Solid Earth **95**, 14095 (1990).

[21] S. A. Hauck II, A. J. Dombard, R. J. Phillips, and S. C. Solomon, Internal and tectonic evolution of Mercury, Earth Planet. Sci. Lett. **222**, 713 (2004).

[22] J. Korenaga, Thermal evolution with a hydrating mantle and the initiation of plate tectonics in the early Earth, J. Geophys. Res. Solid Earth **116**, B12403 (2011).

[23] A. Morschhauser, M. Grott, and D. Breuer, Crustal recycling, mantle dehydration, and the thermal evolution of Mars, Icarus **212**, 541 (2011).

[24] N. Tosi, M. Grott, A. C. Plesa, and D. Breuer, Thermochemical evolution of Mercury's interior, J. Geophys. Res. Planets **118**, 2474 (2013).

[25] J. G. O'Rourke and J. Korenaga, Thermal evolution of Venus with argon degassing, Icarus **260**, 128 (2015).

[26] M. H. Shahnas and R. N. Pysklywec, Toward a unified model for the thermal state of the planetary mantle: Estimations from mean field deep learning, Earth Space Sci. **7**, e2019EA000881 (2020).

[27] S. Agarwal, N. Tosi, D. Breuer, S. Padovan, P. Kessel, and G. Montavon, A machine-learning-based surrogate model of Mars' thermal evolution, Geophys. J. Int. **222**, 1656 (2020).

[28] T. Gillooly, N. Coltice, and C. Wolf, An anticipation experiment for plate tectonics, Tectonics **38**, 3916 (2019).

[29] A.-C. Plesa, N. Tosi, M. Grott, and D. Breuer, Thermal evolution and Urey ratio of Mars, J. Geophys. Res. Planets **120**, 995 (2015).

[30] A.-C. Plesa, S. Padovan, N. Tosi, D. Breuer, M. Grott, M. A. Wieczorek, T. Spohn, S. E. Smrekar, and W. B. Banerdt, The thermal state and interior structure of Mars, Geophys. Res. Lett. **45**, 12,198 (2018).

[31] G. Morra, D. A. Yuen, H. M. Tufo, and M. G. Knepley, Fresh outlook on numerical methods for geodynamics. Part 2: Big data, HPC, education, in *Encyclopedia of Geology*, 2nd ed., edited by D. Alderton and S. A. Elias (Academic Press, 2021), pp. 841–855.

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

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

[34] A. T. Mohan, D. Tretiak, M. Chertkov, and D. Livescu, Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics, J. Turbul. **21**, 484 (2020).

[35] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in *Artificial Neural Networks and Machine Learning—ICANN 2011*, edited by T. Honkela, W. Duch, M. Girolami, and S. Kaski (Springer, Berlin, 2011), pp. 52–59.

[36] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, in *Proceedings of the 28th International Conference on Neural Information Processing Systems- Vol. 1, Montreal, Canada* (MIT Press, Cambridge, MA, 2015), pp. 802–810.

[37] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. **64**, 525 (2019).

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

[39] G. Hirth and D. Kohlstedt, Rheology of the upper mantle and the mantle wedge: A view from the experimentalists, *Inside the Subduction Factory* (American Geophysical Union, 2004), pp. 83–105.

[40] S. D. King, C. Lee, P. E. van Keken, W. Leng, S. Zhong, E. Tan, N. Tosi, and M. C. Kameyama, A community benchmark for 2-D Cartesian compressible convection in the Earth's mantle, Geophys. J. Int. **180**, 73 (2010).

[41] F. Nimmo and K. Tanaka, Early crustal evolution of Mars, Annu. Rev. Earth Planet Sci. **33**, 133 (2005).

[42] H. Wänke and G. Dreibus, Chemistry and accretion history of Mars, Philos. Trans. R. Soc. London A **349**, 285 (1994).

[43] S. Padovan, N. Tosi, A.-C. Plesa, and T. Ruedas, Impact-induced changes in source depth and volume of magmatism on Mercury and their observational signatures, Nat. Commun. **8**, 1945 (2017).

[44] U. Christensen and D. A. Yuen, Layered convection induced by phase transitions, J. Geophys. Res. Solid Earth **90**, 10291 (1985).

[45] N. Tosi, D. A. Yuen, N. de Koker, and R. M. Wentzcovitch, Mantle dynamics with pressure- and temperature-dependent thermal expansivity and conductivity, Phys. Earth Planet. Inter. **217**, 48 (2013).

[46] D. Tozer, Towards a theory of thermal convection in the mantle, in *The Earth's Mantle*, edited by T. Gaskell (Academic Press, New York, 1967), pp. 327–353.

[47] D. Breuer and T. Spohn, Early plate tectonics versus single-plate tectonics on Mars: Evidence from magnetic field history and crust evolution, J. Geophys. Res. Planets **108**, 5072 (2003).

[48] N. Tosi, A.-C. Plesa, and D. Breuer, Overturn and evolution of a crystallized magma ocean: A numerical parameter study for Mars, J. Geophys. Res. Planets **118**, 1512 (2013).

[49] A.-C. Plesa, N. Tosi, and D. Breuer, Can a fractionally crystallized magma ocean explain the thermo-chemical evolution of Mars? Earth Planet. Sci. Lett. **403**, 225 (2014).

[50] A. Scheinberg, L. T. Elkins-Tanton, and S. J. Zhong, Timescale and morphology of Martian mantle overturn immediately following magma ocean solidification, J. Geophys. Res. Planets **119**, 454 (2014).

[51] H. H. Kieffer, Thermal model for analysis of Mars infrared mapping, J. Geophys. Res. Planets **118**, 451 (2013).

[52] A.-C. Plesa, M. Grott, N. Tosi, D. Breuer, T. Spohn, and M. A. Wieczorek, How large are present-day heat flux variations across the surface of Mars? J. Geophys. Res. Planets **121**, 2386 (2016).

[53] J. M. Lumley, The structure of inhomogeneous turbulent flows, in *Atmospheric Turbulence and Radio Wave Propagation*, edited by A. M. Yaglom and V. I. Tatarski (Nauka, Moscow, 1967), pp. 166–178.

[54] O. Friderikos, E. Baranger, M. Olive, and D. Néron, On the stability of POD basis interpolation via Grassmann manifolds for parametric model order reduction in hyperelasticity, arXiv:2012.08851.

[55] F. Chollet *et al.*, Keras (2015), https://keras.io.

[56] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[57] Joblib Development Team, Joblib: Running python functions as pipeline jobs (2020), https://joblib.readthedocs.io/en/latest/.

[58] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, Self-normalizing neural networks, in *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), Vol. 30.

[59] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevFluids.6.113801 for animations of five example simulations from the test set.

[60] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[61] H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, Deep neural networks for nonlinear model order reduction of unsteady flows, Phys. Fluids **32**, 105104 (2020).

[62] C. Olah, Understanding LSTM Networks, Colah's Blog (2015), https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[63] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, Recurrent neural networks for multivariate time series with missing values, Sci. Rep. **8**, 6085 (2018).

[64] M. Phankokkruad and S. Wacharawichanant, A comparison of efficiency improvement for long short-term memory model using convolutional operations and convolutional neural network, in *2019 International Conference on Information and Communications Technology (ICOIACT)* (2019), pp. 608–613.

[65] J. R. Fienup, Invariant error metrics for image reconstruction, Appl. Opt. **36**, 8352 (1997).

[66] S. Brunton and J. Kutz, 7 Data-driven methods for reduced-order modeling, in *Snapshot-Based Methods and Algorithms Volume 2*, edited by P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. Silveira (De Gruyter, Berlin, Boston, 2020), pp. 307–344.

[67] H. Amit, G. Choblet, P. Olson, J. Monteux, F. Deschamps, B. Langlais, and G. Tobie, Towards more realistic core-mantle boundary heat flux patterns: A source of diversity in planetary dynamos, Prog. Earth Planet. Sci. **2**, 26 (2015).

[68] A. C. Plesa and D. Breuer, Partial melting in one-plate planets: Implications for thermo-chemical and atmospheric evolution, Planet. Space Sci. **98**, 50 (2014).

[69] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, arXiv:2102.04626.

[70] A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, Embedding hard physical constraints in neural network coarse-graining of 3D turbulence, ICLR 2020 Workshop on Tackling Climate Change with Machine Learning (2020), https://www.climatechange.ai/papers/iclr2020/7.html.

[71] H. Gao, L. Sun, and J.-X. Wang, Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. **428**, 110079 (2021).

[72] F. Brockherde, L. Li, K. Burke, and K.-R. Müller, By-passing the Kohn-Sham equations with machine learning, Nat. Commun. **8**, 872 (2017).

[73] B. Hamzi, R. Maulik, and H. Owhadi, Simple, low-cost and accurate data-driven geophysical forecasting with learned kernels, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **477**, 20210326 (2021).

[74] N. Margenberg, C. Lessig, and T. Richter, Structure preservation for the deep neural network multigrid solver, arXiv:2012.05290.

[75] See, https://github.com/agsiddhant/ForwardSurrogate_Mars_2D.

[76] A. Brand, L. Allen, M. Altman, M. Hlava, and J. Scott, Beyond authorship: Attribution, contribution, collaboration, and credit, Learned Publ. **28**, 151 (2015).

[77] G. Schubert, D. L. Turcotte, and P. Olson, *Mantle Convection in the Earth and Planets* (Cambridge University Press, Cambridge, 2001).

[78] C. Herzberg, P. Raterron, and J. Zhang, New experimental observations on the anhydrous solidus for peridotite KLB-1, Geochem. Geophys. Geosys. **1**, 2000GC000089 (2000).

[79] J. Zhang and C. Herzberg, Melting experiments on anhydrous peridotite KLB-1 from 5.0 to 22.5 GPa, J. Geophys. Res. Solid Earth **99**, 17729 (1994).

[80] P. Van Keken, Cylindrical scaling for dynamical cooling models of the earth, Phys. Earth Planet. Inter. **124**, 119 (2001).

[81] H. Wänke, G. Dreibus, S. K. Runcorn, G. Turner, and M. M. Woolfson, Chemical composition and accretion history of terrestrial planets, Philos. Trans. R. Soc. London A **325**, 545 (1988).