Data-Centric Engineering (2022), 3: e19
doi:10.1017/dce.2022.18

CAMBRIDGE UNIVERSITY PRESS

On the reproducibility of fully convolutional neural networks for modeling time–space-evolving physical systems

Wagner G. Pinto[1], Antonio Alguacil[1,2] and Michaël Bauerheim[1,*]

[1]Aerodynamics, Energetics and Propulsion Department, ISAE-SUPAERO, Université de Toulouse, 10 Avenue Edouard Belin, 31055 Toulouse, France
[2]Mechanical Engineering Department, Université de Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke QC J1K 2R1, Canada
*Corresponding author. E-mail: michael.bauerheim@isae-supaero.fr

Received: 30 September 2021; **Revised:** 07 March 2022; **Accepted:** 27 March 2022

Keywords: Autoregressive; convolutional neural network; deep learning; reproducibility

Abstract

Reproducibility of a deep-learning fully convolutional neural network is evaluated by training several times the same network on identical conditions (database, hyperparameters, and hardware) with nondeterministic graphics processing unit operations. The network is trained to model three typical time–space-evolving physical systems in two dimensions: heat, Burgers', and wave equations. The behavior of the networks is evaluated on both recursive and nonrecursive tasks. Significant changes in models' properties (weights and feature fields) are observed. When tested on various benchmarks, these models systematically return estimations with a high level of deviation, especially for the recurrent analysis which strongly amplifies variability due to the nondeterminism. Trainings performed with double floating-point precision provide slightly better estimations and a significant reduction of the variability of both the network parameters and its testing error range.

Impact Statement

The use of neural networks for modeling physical systems is a growing trend, a scenario where reproducibility of training can be crucial not only to reproduce others' work, but also to allow refining hyperparameters and the network's architecture. In this analysis, models trained under solely hardware nondeterminism and their responses' variability are discussed and quantified, with a distinction between single and double floating-point precision. The obtained models and inference results show a significant change, demonstrating the inherent variability of the training and that conclusions drawn with only a single training can be limited. This study reveals that a systematic use of multiple equivalent trainings should be encouraged and comparisons dealing with accuracy/performance of the neural network should be provided.

This research article was awarded an Open Materials badge for transparent practices. See the Data Availability Statement for details.

1. Introduction

Reproducibility, namely the principle that a method or experiment can be replicated, is essential in science. Obtaining reproducible results is complex in deep learning (DL) due to the intrinsic stochastic nature of the majority of the algorithms, the complex nature of the optimization search spaces, and the use of nondeterministic computations. Replicating a previous work is specially nontrivial due to the difficulty to reproduce the experiments' conditions associated with the high number of details of the settings (Dadvar and Eckert, 2018), the large size databases, and distinct hardware. As such, this topic is constantly being discussed (Peng, 2011; Ivie and Thain, 2018; Raff, 2019; Pineau et al., 2021; Renard et al., 2020).

While most sources of variability can be removed by a proper and well-documented setup (fixed seeds, well-defined hyperparameters, etc.), a computational variability remains due to the numerical nondeterminism issued from the rounding of numbers associated with the stochastic order of arithmetic operations (Whitehead and Fit-Florea, 2011; Chou et al., 2020). It is notably observed when the same code is evaluated in different hardware (Jézéquel et al., 2015) or with different floating-point precisions (Seznec et al., 2018). Currently, performance requirements make the use of graphics processing units (GPUs) mandatory in DL, their major drawback being the difficulty to perform deterministic operations. Different runs in the same GPU can also be nondeterministic due to use of algorithms that favor performance over repeatability (Iakymchuk et al., 2016; Jorda et al., 2019).

The use of neural networks for estimating partial differential equations (PDEs) is a growing trend and has been widely evaluated in the Fluid Mechanics field. Many examples are available on the review by Brunton et al. (2020); general purpose PDE solvers are also discussed on the literature (e.g., Berg and Nyström, 2018; Sirignano and Spiliopoulos, 2018; Raissi et al., 2019; Lu et al., 2021). In contrast with typical DL problems, such as classification problems, the estimation of spatial–temporal PDEs must guarantee pixelwise precision. For example, the prediction of the unsteady flow dynamics of a wake downstream a cylinder depends on the accurate modeling of the evolution of the velocity and pressure fields. Small variations at different regions of the domain, notably close to the walls and at the shear layers, may change the dynamics of small-scale structures and return a wrong flow (Lee and You, 2019a). Thus, even if determinism is not mandatory, levels of accuracy and reproducibility must be mastered, and thus further evaluated and discussed.

There is no simple and unique definition of reproducibility (Fidler and Wilcox, 2021); thus, it is important to clarify this concept in order to delimit the scope of the analysis. In this work, reproducibility is considered as the capacity to train models that would return identical responses (null variability) when obtained from the same materials (software, hardware, database, and random number seeds). This follows the definition presented by Goodman et al. (2016) and Bouthillier et al. (2019) ("methods reproducibility") and also evoked in Pineau et al. (2021).

Even if bitwise reproducibility is not fundamental for having a reliable machine learning model, reducing variability can be extremely advantageous. More reproducible trainings allow for a better understanding of the learning mechanisms, thus, to enable further improvements in the model's architectures and overall performance and, in the case of scientific applications, to have insights on the underlying physics. Moreover, for recurrent tasks as in space–time physical problems, the inherent nonlinear nature of neural networks can induce a large error at long times even for a very low initial variability, as illustrated in Figure 1, with the results obtained in the scope of current work by the same model trained twice under the same conditions but with nondeterministic algorithms, which also calls for reproducibility. Similar analysis in the context of Reinforcement Learning (Nagarajan et al., 2018) found that the GPU nondeterminism has an impact analogous to changing the starting model's parameters.

Consequently, this paper presents a detailed comparison between several DL models that estimate the evolution of three PDEs, namely the heat, Burgers', and wave equations, as benchmarks for time–space-evolving systems, trained in the same context (identical database, hardware, random number generator seeds, hyperparameters, software, and hardware) in a GPU with nondeterministic operations. Tests are performed to evaluate and compare the generalization capacity of the trained models. To the best of the authors' knowledge, no previous work has discussed the reproducibility of a neural network when

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

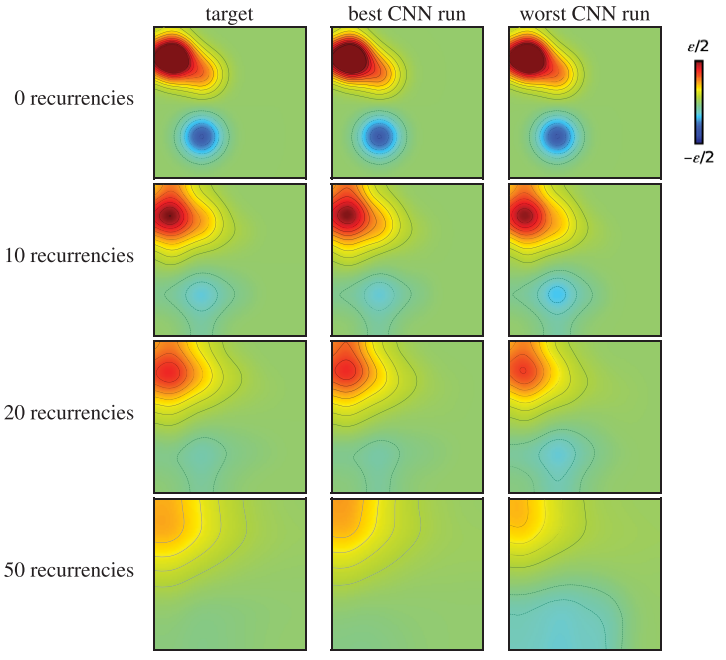

Figure 1. *Illustration of the variability induced solely by training nondeterminism with the results obtained with neural networks trained in identical conditions for the modeling of the heat equation, single precision. "Best" and "worst" refer to the runs with the smallest and the highest average loss for the test dataset (see Figure 14). Contour lines are stepped by 1/20 of the color bar limits, null isocontour is omitted, and the amplitude ε and the simulation details are available in Section 2.1.1.*

estimating time–space-evolving systems. The document is organized as follows. Methodology is presented in Section 2, with the description of the physical systems, the network architecture, and the training and testing procedures. The statistical convergence of the procedure and the models' properties (loss convergence, kernels weights, and feature fields) are discussed in Section 3. In Section 4, results for the recurrent analysis are presented for solution benchmarks and a database with randomly defined pulses. Section 5 closes the document with final remarks. Code, descriptions of the neural network, the numerical setup, and examples of the estimated fields are available in the Supplementary Material.[1]

2. Methodology

2.1. Physical systems and solvers

The analysis is performed for three different physical systems: the heat equation (parabolic PDE), the Burgers' equation (hyperbolic–parabolic PDE), and the wave equation (acoustic propagation, hyperbolic PDE). Each problem is trained independently, both in single and double precisions. The computational domain is a square of size $\ell \times \ell$, discretized by N=200 uniform cells of dimension Δx in both directions. Gaussian pulses are used to initialize the fields. A generic field F is started by a background value F_0 and n pulses:

$$F(\mathbf{x}, t=0) = F_0 + \sum_{i=1}^{i=n} \varepsilon \exp\left[-\log(2)\frac{d(\mathbf{x}, \mathbf{x}_{c,i})^2}{h_w^2}\right], \tag{1}$$

[1] The Supplementary Material is available at https://gitlab.isae-supaero.fr/daep/cnn-multiphysics-repro.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

where $\mathbf{x} = (x, y)$ are the space coordinates, t is the time, n is the total number of pulses, ε is the pulse amplitude, d is the Euclidean distance from the center of the pulse $\mathbf{x}_{c,i}$, and h_w is the half-width at half maximum of the Gaussian. When the amplitude ε value is negative, the pulse is referred to as a "negative pulse." Details on the implementation and parameters for the considered physical systems are presented next.

2.1.1. Heat equation

The heat equation, also known as diffusion equation, represents the time-space dissipation of heat:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{2}$$

where T is the temperature and α is the thermal diffusivity. Since the equation is linear, the background temperature is set to $F_0 = T_0 = 0.0$ without the loss of generality. The diffusivity is $\alpha = 2.0 \times 10^{-4}$ m^2/s, and the pulse's amplitude and size are, respectively, $\varepsilon \in [-1.0, 1.0]$—sampled by a random uniform distribution—and $h_w = 20\Delta x$ (0.1 m). Domain has $\ell = 1$ m of side, and boundaries are adiabatic (null temperature gradient).

The system is simulated using the lattice Boltzmann method (LBM), where macroscopic quantities, such as temperature, density, and velocity, are obtained by modeling the microscopic distribution of particles described by the Boltzmann equation. More details on the method can be found in Krüger et al. (2017). Simulations are implemented with the open source framework Palabos[2] (Latt et al., 2020), using the BGK (Bhatnagar–Gross–Krook) operator and a D2Q5 lattice.

2.1.2. Burgers' equation

The Burgers' equation is a nonlinear PDE used to represent a simplified dynamics of a viscous flow:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\frac{\partial \mathbf{u}}{\partial x} = \nu \frac{\partial^2 \mathbf{u}}{\partial x^2}, \tag{3}$$

where $\mathbf{u} = (u, \upsilon)$ is the flow velocity and ν is the kinematic viscosity. For each pulse, u is initialized with Gaussian pulses, as in Equation (1), over a null background $F_0 = (u_0, \upsilon_0) = (0, 0)$; the velocity in y is unchanged. The sign of the perturbations of u is selected randomly using a uniform distribution (50% of chance for each sign), and its amplitude and size are $\varepsilon = \pm 0.1$ and $h_w = 20\Delta x$ (0.1 m). Viscosity is set to $\nu = 5.0 \times 10^{-4}$ m^2/s. In order to avoid the appearance of shocks, the several initial pulses must be far from each other. Here, a minimal distance of $2h_w$ is enforced between all pulses when sampling their coordinates. The computational domain is 1×1 m with null velocity gradient imposed at the boundaries, and the simulation timestep is $\Delta t = 0.02$s. The solver is implemented using the open-source finite volumes framework OpenFOAM[3] (Weller et al., 1998), using second-order schemes in both space and time (Crank–Nicolson).

2.1.3. Wave equation

Behavior of acoustical waves can be represented by the linear wave equation:

$$\frac{\partial^2 \rho'}{\partial t^2} = a^2 \frac{\partial^2 \rho'}{\partial x^2}, \tag{4}$$

where $\rho' = \rho - \rho_0$ is the acoustic density and a is the sound speed. In the current application, the ambient density is set as $F_0 = \rho_0 = 1.0$, LBM units. The pulses are of fixed amplitude and size (0.001 and $12\Delta x$, respectively), and null velocity is imposed at boundaries (reflecting boundary condition). Each timestep

[2] Palabos solver: https://palabos.unige.ch/.
[3] OpenFOAM solver: https://www.openfoam.com/.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

represents a duration of $\Delta t = 0.0029\,\ell/a$, where $\ell = 100$ m is the domain size and $a = 343$ m/s. This means that 173 timesteps, that is, 43 recurrences (a recurrence being a jump of 4 timesteps, as explained in Section 2.3) are necessary for a wave to reach the boundaries from the center of the domain. Acoustic propagation is also simulated using the LBM with the Palabos framework, same collision operator as for the heat equation and a D2Q9 lattice.

2.2. Neural network architecture

In contrast to other time-evolving problems, such as natural language processing, most time-dependent physical problems do not require a lot of time memory. Only a few snapshots need to be remembered in order to reconstruct approximation of the partial derivatives included in the PDEs. In that context, recurrent neural network (RNN) and long short-term memory (LSTM) have been proved less effective (Fotiadis et al., 2020) for this recursive task compared with a Convolutional Neural Network (CNN) approach with a few previous snapshots as input. The latter strategy is therefore used here to output the next time frame in three unsteady physical systems.

The multiscale convolutional neural network introduced by Mathieu et al. (2016) is used. As indicated by the latter authors, the major limitation of a CNN is that it only accounts for short-range dependency. The use of multiscale network is capable of surpassing this limitation by considering a different structure size inside each scale. The current network is composed by three scales (N, $N/2$, and $N/4$), and the transitions between them (upsampling and downsampling) are performed using bilinear interpolations. Each scale follows the principle of the Laplacian pyramid used in image compression (Denton et al., 2015), similar to a U-Net (Ronneberger et al., 2015).

For the sake of simplicity, a replication padding is used at the borders in order to keep the size of the feature maps constant and to learn implicitly the physical boundary conditions of the problems (Alguacil et al., 2021b). Nonlinearity is added by applying the rectifying linear unit activation function. Due to the nature of the problem, they are not present for the totality of layers so that negative physical quantities values can be obtained. A simplified representation of the neural network is presented in Figure 2, and details of the architecture are listed in Table 1. It is composed by a total of 422,419 trainable parameters, divided in 17 convolutions. The neural network is coded using the PyTorch[4] framework (Paszke et al., 2019).

Following Alguacil et al. (2021a), the input is composed by four scalar fields, called frames, issued from the simulations, ordered in time. The output is the following frame, representing the field to be

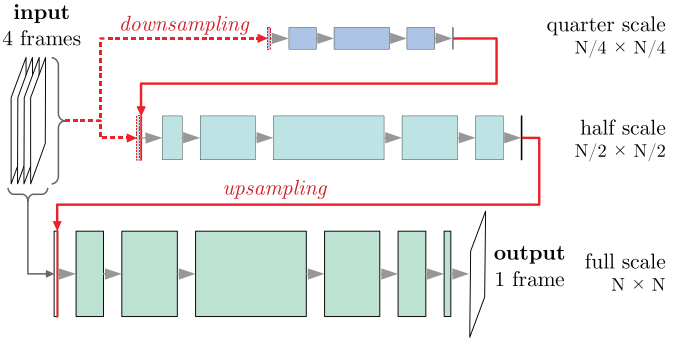

Figure 2. *Simplified diagram of the multiscale neural network. Arrows between boxes indicate a two-dimensional convolution operation, and boxes' height and breadth are proportional to the frame dimension and the number of layers, respectively; interpolation operations are presented by dashed lines for downsampling and continuous lines for upsampling.*

[4] PyTorch framework: https://pytorch.org/.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

Table 1. *Network architecture.*

Scale	$N/4 \times N/4$	$N/2 \times N/2$	$N \times N$
Number of feature maps	4[a], 32, 64, 32, 1	5, 32, 64, 128, 64, 32, 1	5, 32, 64, 128, 64, 32, 8, 1[a]
Convolution kernel size	3, 3, 3[b], 3[b]	5, 3, 3, 3, 3[b], 3[b]	5, 3, 3, 3, 3[b], 5[b], 1[b]

[a]Input and output.
[b]Not followed by an activation layer.

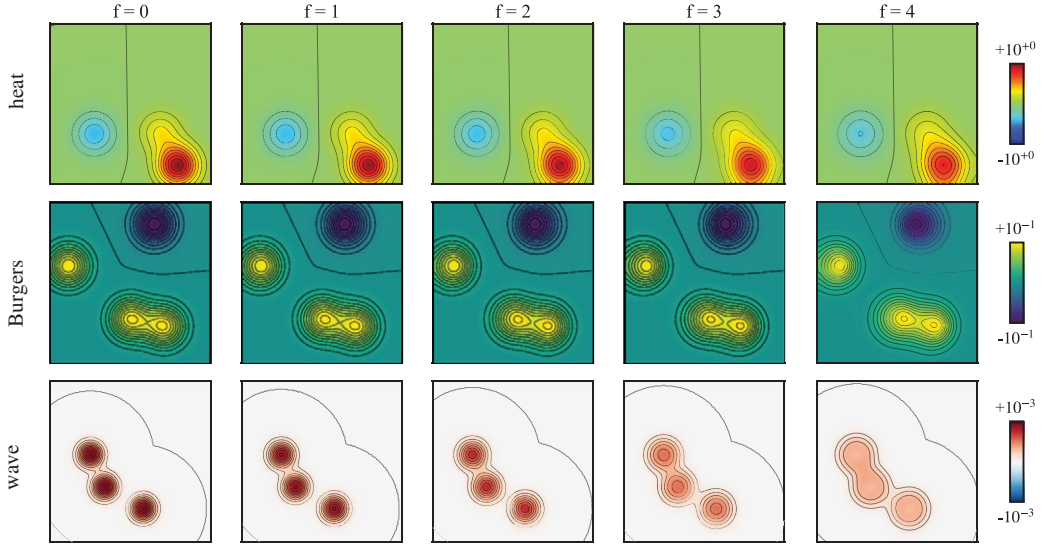

Figure 3. *Samples of a datapoint (five frames) for the three equations (more details in Section 2.1). Initial four frames are used as input, and the last is the target. Contour lines are stepped by 1/20 of the color bar limits, and null isocontour is omitted.*

estimated. A jump of four simulation timesteps is considered between each frame. A datapoint is a group of these five frames (four inputs and one output), as illustrated in Figure 3, for the three target problems. Note that the small difference between the frames is associated with the physical limitation of time marching so that the system physics is correctly captured by the neural network (Alguacil et al., 2021a), what also illustrates the importance of limiting the error and evaluating the neural network (NN) reproducibility. The simulated fields are split with no overlap, that is, every datapoint has a group of unique frames. At each forward pass, data of all frames are divided by the standard deviation of the first frame before performing the calculations.

As in Mathieu et al. (2016), the L2-norm of the scalar fields and their spatial gradients compose the loss $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{M}\sum_{k=1}^{M}(\lambda_{\mathrm{L2}}\mathcal{L}_{\mathrm{L2}} + \lambda_{\mathrm{GDL}}\mathcal{L}_{\mathrm{GDL}}), \tag{5}$$

where

$$\mathcal{L}_{\mathrm{L2}} = \left(F_k - \widetilde{F}_k\right)^2 \text{and } \mathcal{L}_{\mathrm{GDL}} = \left[\frac{\partial F_k}{\partial x} - \frac{\partial \widetilde{F}_k}{\partial x}\right]^2 + \left[\frac{\partial F_k}{\partial y} - \frac{\partial \widetilde{F}_k}{\partial y}\right]^2. \tag{6}$$

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

2.3. Nonrecurrent and recurrent analysis

The network only accounts for one future frame. Estimation of further scalar fields is done via a recurrent (autoregressive) analysis, where the output of a given model evaluation is used as input in the following estimations, as performed in previous works in the literature (Lee and You, 2019a, 2019b; Alguacil et al., 2021a):

$$\widetilde{F}^{f} = \mathcal{N}\left(\widetilde{F}^{f-1}, \widetilde{F}^{f-2}, \widetilde{F}^{f-3}, \widetilde{F}^{f-4}\right), \tag{7}$$

where $\mathcal{N}$ represents the neural network operator and the exponents the frame number. Preliminary analysis of the wave equation estimation has shown that there is an average shift of the fields when performing the recurrent analysis. This effect is contained by a posterior physics-based energy-preserving correction, as in Alguacil et al. (2021a). The specificity of autoregressive strategies is that the network was trained only to output the next frame, without knowledge on the long-time error produced. For the sake of simplicity, no long-term loss, as in Tompson et al. (2017) and Ajuria-Illarramendi et al. (2020), was added to constraint this accumulation of error in time. Consequently, small differences due to nondeterminism at the first recurrence might be amplified nonlinearly by the neural network during the autoregressive procedure: the evaluation of the long-time variability due to nondeterminism is one of the main objectives of this paper.

2.4. Training procedure and databases

The fields are initialized with one to four Gaussian pulses, as described by Equation (1), centered between $[10\%, 90\%]$ of the domain, in both directions. The number and the position of the pulses are defined randomly, following independent uniform distributions.

Training and validation databases are composed by 500 simulations (400 for training and 100 for validation) for each equation, with a total of 4,400 datapoints (3,200 + 1,200), and scripts and instructions for reproducing it are available in the Supplementary Material. During the learning phase, batches of 32 datapoints (160 frames) are considered. Data augmentation is employed to increase the diversity of the database in the case of heat and wave equations. In particular, random rotation operations by multiples of $\pi/2$rad are performed for the group of fields composing the datapoints before the training.

Reproducibility of the network is evaluated by performing several trainings with the exact same architecture, database, and hyperparameters. The seeds of random number generators are equally set at the beginning of each training. Following nomenclature is used on this document: a "model" refers to the trained neural network and a "run" refers to a given model issued from an independent training performed with the just described identical conditions. In the current framework, deterministic algorithms are not available for all the operations present in the model, such as padding, for instance (Torch Contributors, 2019). On a CPU, all the trainings can be deterministic, but it is impractical due to poor performance. Even if determinism could be achieved by using another framework, hardware, or model architecture, the use of nondeterministic algorithms is deliberate as it mimics the training of the same model on different devices or with distinct versions of libraries and drivers, where determinism may not be guaranteed (NVIDIA Corporation, 2020a, 2020b).

Since the nondeterminism is associated with the truncation error, the trainings are done considering two floating-point precisions: single (32 bits, framework's default, referenced as FP32) and double (64 bits, FP64). The model and database are exactly the same, and the framework's default type is modified at the beginning of the runs. Data are generated in double precision and converted to single precision for the single-precision trainings using the framework's default implicit to-nearest conversion. The number of runs is 10 for single and double precisions per physical system. Such a number of experiments is in accordance with the number of runs used in traditional hyperparameter optimization (Li and Talwalkar, 2020). Forty trainings are performed for the wave equation in order to assess the statistical convergence of the analysis (Section 3.2).

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

A total of 1,500 epochs are calculated on each run. In order to have a representative range of the observed behaviors, models are saved at every 125 epochs, so a total of 12 checkpoints are available at the time of the postprocessing. Training is performed on an NVIDIA V100 GPU. The average time for the completion of an epoch (training + validation) is about 30 s in single precision and 100 s in double precision, so, respectively, about 12 and 40 hr are necessary per run.

2.5. Testing procedure and databases

Two distinct testing datasets are used for the evaluation of the quality of the trained models.

Benchmarks databases: For each physical problem defined in Section 2.1, three benchmark simulations that are not present in the training database are used, aiming at evaluating the capacity of generalization of the models. The benchmarks are illustrated in Figure 4 (the pulse half-width used to generate the training database is maintained for all cases). Note that since the neural network is highly nonlinear, the principle of solution superposition for linear systems, such as the heat and wave problem, are not guaranteed, and therefore constitutes critical generalization cases. For each problem, the relevant benchmark cases are defined as:

- Heat equation:
  - Gaussian pulse: a single Gaussian pulse is initialized at the center of the domain with an amplitude $\varepsilon = 1.0$;
  - opposed Gaussians: two opposite Gaussian pulses ($\varepsilon = \pm 1.0$) are placed at the y symmetry axis with $\pm 20\Delta x$ of offset from the center of the domain; and
  - square pulse: a square pulse of side $2h_w$ is located at the center of the domain with amplitude $\varepsilon = 1.0$.
- Burgers' equation:
  - Gaussian pulse: the same initialization as for the heat equation is performed, but with an amplitude $\varepsilon = 0.01$;
  - opposed Gaussians: the same initialization as for the heat equation is performed, but with an amplitude $\varepsilon = \pm 0.01$ and offset of $40\Delta x$; and
  - sine wave: a 2D sine wave is defined by $u = \sin(2\pi x)\sin(2\pi y)/(4\pi)$ and $v = 0$.
- Wave equation:
  - Gaussian pulse: the same initialization as for the heat equation is performed, but with an amplitude $\varepsilon = 0.001$;
  - opposed Gaussians: the same initialization as for the heat equation is performed, but with an amplitude $\varepsilon = \pm 0.001$ and offset of $40\Delta x$; and
  - plane Gaussian: a plane wave of Gaussian contour in x ($\varepsilon = 0.001$), and constant shape along the y direction is investigated. Note that this case is known to be critical when evaluating the generalization capabilities of neural networks on such a wave problem (Sorteberg et al., 2018; Fotiadis et al., 2020; Alguacil et al., 2021a).

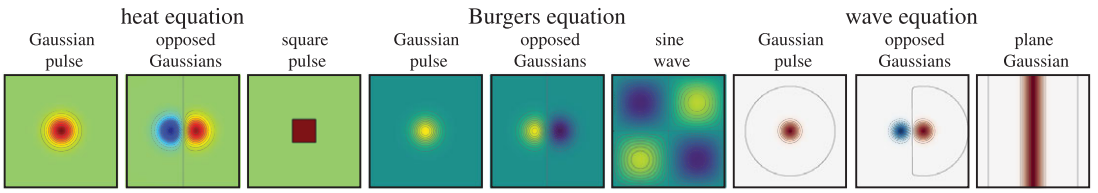

Figure 4. *Fields for the benchmark cases at the start of the simulation. Refer to Figure 3 for colormap and contour properties.*

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

Random databases: Sets of 100 simulations with random Gaussian pulses, following the procedure used for generating the training and validation datasets (see Section 2.4). This database allows the production of statistics regarding the quality and variability of the models.

3. Analysis of Models

3.1. Statistical convergence of the procedure

The stochastic nature of the nondeterminisitic trainings of DL models on GPUs requires to provide the results' statistics over multiple equivalent trainings. However, the number of trainings needed for converged statistics is still open, and highly depends on multiple factors such as the problem considered, the floating precision during training, the number of recurrences in unsteady problems, and so forth. This aspect is quantified in the current analysis on the wave equation where 40 runs are performed in both single and double precisions.

Since the analysis is focused on the accuracy of the estimations, the recurrent response of the benchmarks, further discussed on Section 4.1, is chosen as the criterion to evaluate the statistical convergence of the procedure. Figure 5 presents the convergence of the normalized average root-mean-square error (RMSE) for the three acoustic benchmarks described in Section 2.5, considering from 1 to 40 runs at recurrences 0 and 99. In order to achieve a $\pm 10\%$ deviation from the average error of the 40 trained models (red dashed lines), about 20–30 runs are necessary for single precision, whereas only 3 in double. Since there is no sorting done in the selection of the runs, that is, Run 1 or Run 40 is not the one with the lowest or highest value of error, there is no control on what will happen at the beginning of the curves. It is a simple coincidence that most curves start with values under the average. Although statistical convergence may not be achieved with only 10 runs, it is already sufficient to demonstrate the variability of the models and to observe the discrepancy between the single and double precision trainings.

Considering the high computational cost required to perform multiple trainings on several physical problems, only 10 runs are performed for the heat and Burgers' equations for both single and double precisions. Even if 40 runs are available for the wave equation, statistics in Section 3 will be drawn using the first 10 training only for consistency. Again, even if statistical convergence is not guaranteed with this limited number of runs, it is sufficient to highlight the inherent variability to nondeterminism as well as the effect of the floating precision.

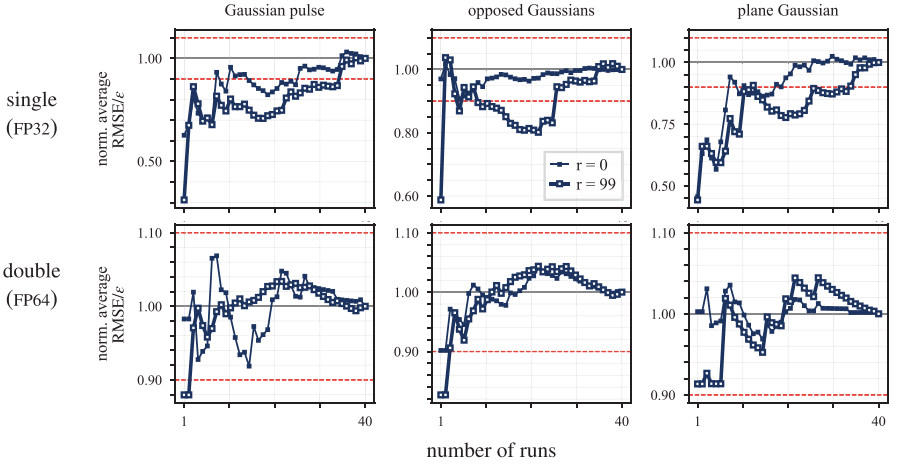

Figure 5. *Evolution of the normalized average root-mean-square error with the number of models for the wave equation benchmarks, trainings in single (top) and double (bottom) precisions. Reference is the average error at the given number of recurrences for the totality of trained models, indicated by the continuous horizontal line; the hashed lines delineate a range of $\pm 10\%$ of that value.*

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

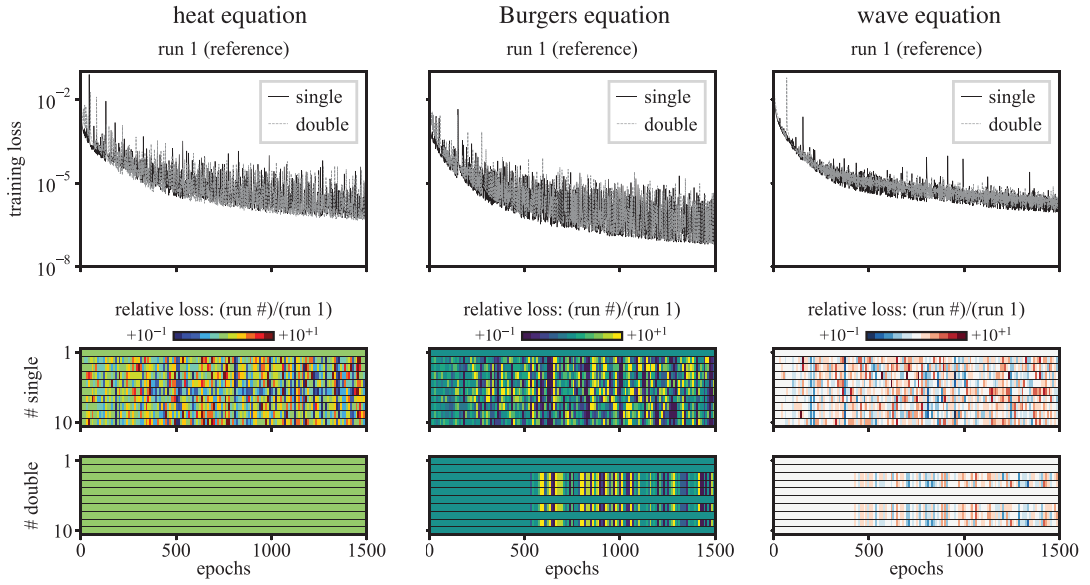

Figure 6. *Evolution of the absolute (top) and relative (ratio to first run, at bottom) training loss for the different physical systems. For visualization purposes, only 1 in every 10 values is shown for the relative loss.*

3.2. Convergence of trainings

The evolution of the training loss in the learning phase for the two groups of runs (single and double precisions) is shown in Figure 6 for the three physical equations. Trainings are limited here to 10 runs, which is sufficient to distinguish variability and floating precision effects, yet statistical convergence for FP32 is not satisfied as shown in Figure 5. In order to compare the variability among the 10 runs, a surface plot displays the evolution of the relative loss, that is, the losses of a given a run divided by the first run for each physical system and precision.

The several trainings share a similar overall behavior (value and slope of losses convergence) for both floating-point precisions. However, comparing the loss between multiple equivalent trainings reveals that the losses in single precision differ rapidly after a small number of epochs whatever the physical problem considered. Additionally, 10 different patterns are visible highlighting the high variability and nonreproducibility since all the trainings are completely different. In contrast, a different trend is observed for double precision, where trainings become dissimilar only after a larger number of epochs, For instance, all double precision heat equation models are identical, while for the Burgers' equation, dissimilarities occur after 500 epochs with only two distinct groups of results. For the wave propagation in double precision, losses evolve identically until 400 epochs. After that, four patterns can be seen (Runs 1, 2, 5, 6, and 10 returned identical losses, the same for pairs 3–8 and 4–9). This result is a first insight on how floating precision affects the training reproducibility. It suggests that reproducibility is governed by the variability in the gradient descent path leading to potential multiple different local minima depending on the physical problem considered. Intriguingly, the double-precision trainings were more reproducible for the less complex physical systems, such as for the heat equation case, implying that the optimization search space is also less complex for these cases. This aspect is tested by producing the loss landscape along the first two principal component directions using a Proper Orthogonal Decomposition (POD) as in Li et al. (2018). The analysis is illustrated in Figure 7 with the loss landscapes of the first run for the heat and wave equations, single and double precisions. The main difference observed between the two is the apparent larger gradients in the loss landscape of the former, that is, the contour lines being more concentrated in the

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

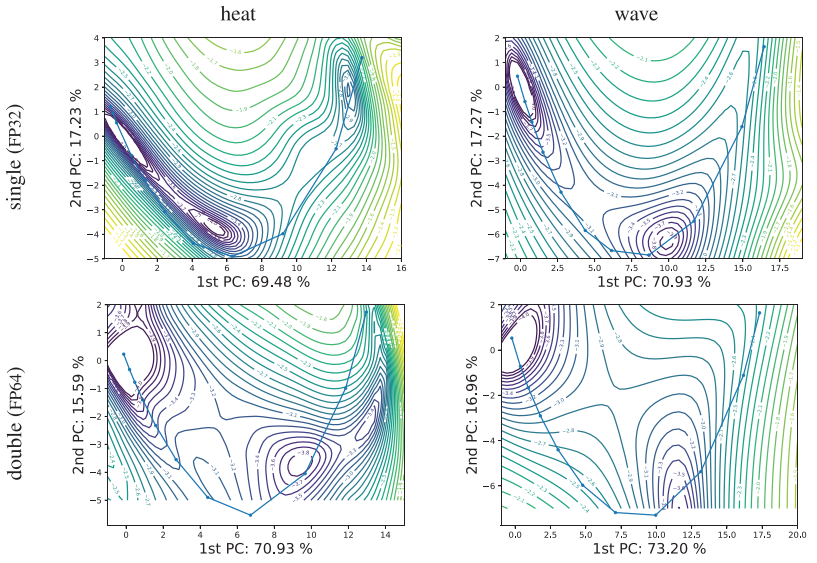

Figure 7. *Two-dimensional visualization of the loss surface for the first run at different precisions (*FP32 *and* FP64*), wave and heat equations. The optimizer trajectory is superposed as a blue line with dots, indicating the training checkpoints. Isocontours are in log scale, from* 10^{-4} *to* 10^{-1}.

Table 2. *Statistics of the final total losses considering all runs in single (*FP32*) and double (*FP64*) precisions for the best model obtained on each training.*

		Heat equation		Burgers' equation		Wave equation	
		Training	Validation	Training	Validation	Training	Validation
Single	avg, $\times 10^{-6}$	0.792	0.703	0.107	0.088	1.546	1.273
	std, $\times 10^{-7}$	2.473	0.756	0.294	0.230	4.521	2.086
	max / min	2.717	1.401	1.925	2.243	2.538	1.569
Double	avg, $\times 10^{-6}$	0.659	0.642	0.102	0.087	1.446	1.537
	std, $\times 10^{-7}$	0	0	0.140	0.021	1.795	1.537
	max / min	1	1	1.318	1.049	1.345	1.348

heat equation problem indicate larger localized gradients, which forces the optimizer toward the optimum more efficiently than flatter regions.

The statistics of losses for the trained networks are listed in Table 2. The models considered are the best of each run having the lowest training or validation losses as criteria. The training variability is significant for single-precision trainings, whereas the double-precision runs are more similar. Standard deviation of the losses is about two times smaller for the latter group.

Differences between the models are further discussed by comparing the weights and feature fields for a sample input. For these analyses and in the subsequent performance study (Section 4), the best models for each run (state associated with the lowest validation loss) are considered.

3.3. Comparison of kernels

A preliminary visual comparison of the convolution kernels is performed. As expected from the evolution of losses, weights and biases are found different among the groups of runs, especially for single precision.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

No relationship or clear tendency could be noticed in terms of which kernels and biases were the most different and their influences on each model's behavior, which suggests that each group corresponds to a distinct local minimum with different features, and not to the same minimum with various degrees of accuracy and close features. A best-matching comparison, that is, looking for groups of similar kernels from all of the available ones, could be done. However, the presence of similar filters but at different positions on the network means that they are working with different input feature fields. Simultaneously checking the similarity of kernels and incoming feature fields would become a very complex task, hard to interpret. Instead, a global quantitative analysis is proposed and presented further.

For the group of different runs at equal precision of a given physical system, a deviation criterion $\delta(w)$ calculated as the pixelwise standard deviation normalized by the maximum value among all runs in the group is proposed. For a convolution weight w, it is calculated as

$$\delta(w) = \overline{\text{deviation}(w)} = \frac{\text{std}\left(w^{\text{run}\,=\,i..n}\right)}{\max\left(\left|w^{\text{run}\,=\,i..n}\right|\right)}. \tag{8}$$

The calculation is performed for every individual weight, such that the dissimilarity between the models is estimated. An example of the application of the deviation criterion is shown in Figure 8, for the kernels that are, on average, the most contrasting among the heat and Burgers' equation models trained with single and double precisions. This result shows that for both equations, at least some filters are extremely different. It confirms the assumption that even when all runs yield similar loss values, they do not correspond to the same local minimum, and therefore are indeed different models with potential different behavior, especially when evaluated in a recurrent mode (Section 4).

It can also be observed that training with FP64 was not a guarantee of unique behavior. Considering the Burgers' equation, even if the kernels are identical among the two sets of unique networks, the two kernels are not analogous, with the upper right weight being specially distinct: negative for Runs 1, 2, 6, 8, and 10 and positive for the remaining trainings.

The distribution of the deviation criteria $\Pi_{\delta(w)}$ for all convolution weights is displayed in Figure 9, considering each set of physical system and precision. Runs with single precision have a probability mode (i.e., the location at which the peak of $\Pi_{\delta(w)}$ occurs) around 0.1–0.2 deviation units, with about 80% of the

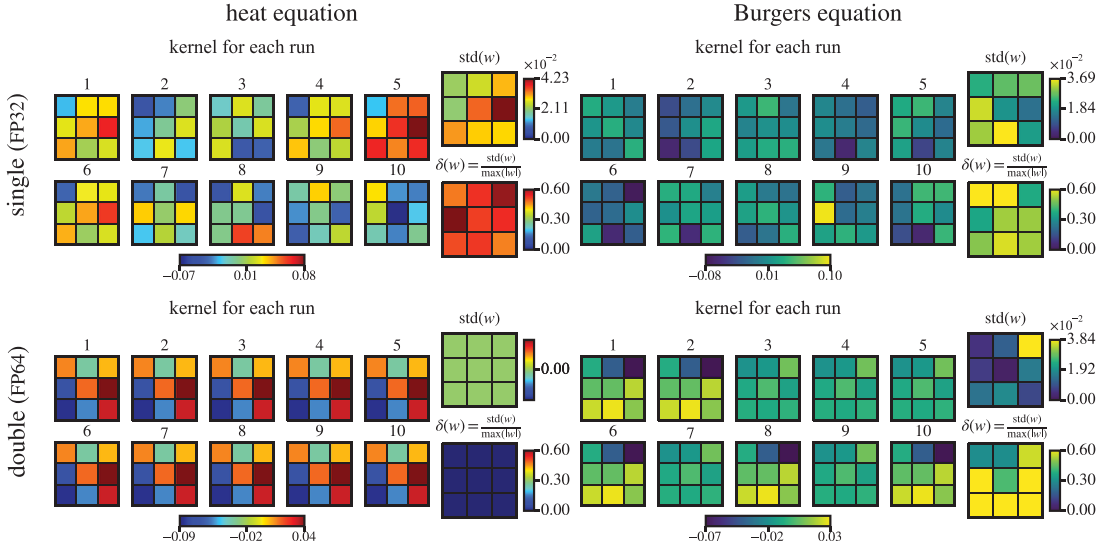

Figure 8. *Weights, standard deviation, and deviation criterion for most contrasting convolution kernel for the sets of heat (left) and Burgers' (right) equation models trained with single (top) and double (bottom) precisions.*

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

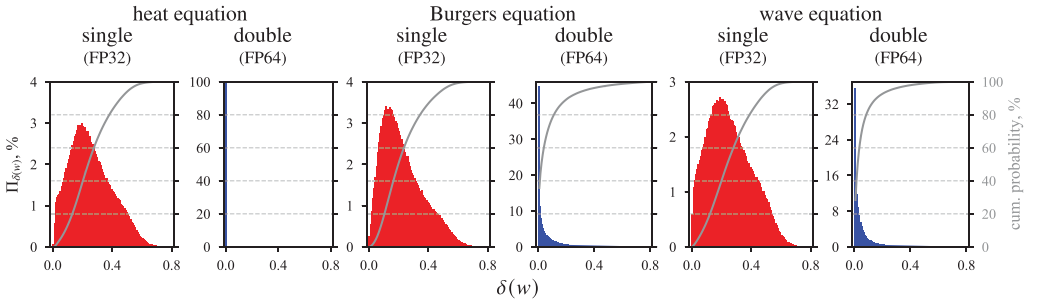

Figure 9. *Probability density of the deviation of the convolution weights for the single and double precision runs.*

weights lower than 0.4 units. For the runs with the double precision, the mode corresponds to the null deviation, representing 35% of the weights. Furthermore, 80% of the weights for the Burgers' and wave equations are contained within 0.1 of the deviation criterion. Finally, since all models were found to be identical for the heat equation, deviation is 0 for all weights for such case. These values reinforce both the significant variability of the single precision runs, probably converged to multiple local optima, and the homogeneity of the trainings with double precision.

It can be also noticed that in single precision, $\Pi_{\delta(w)}$ is similar for the three physical systems, whereas large differences are observed in double precision. For instance, the heat equation yields perfectly reproducible results, whereas Burgers' and wave equations not. This not only exemplifies that the behavior of the trainings in FP64 cannot be inferred from FP32, but also suggests that the nature of the physical system plays a role in its reproducibility.

3.4. Feature fields in nonrecurrent mode

In complement of the computation of the deviation probability function $\Pi_{\delta(w)}$ (Section 3.3) to quantify the difference between the models, the feature fields produced for a fixed input are also compared. Due to the size of the network, the analysis is limited to the outputs of each scale (quarter, half, and full), the latter being the output of the model, used as new input when evaluating the network in recurrent mode (Section 4).

The first four frames and Frames 99–103 of the Gaussian pulse benchmark are selected as input. Quantification of the deviation among different models is made using the criterion defined in Equation (8), here applied to each value that composes the fields. The analysis is split between the runs in single and double precisions, and the results are shown in Figures 10 and 11.

Even if restricted by the small size of the dataset, this analysis leads to three conclusions. First, a large variation (order of 10% of the maximum value) is obtained for both the quarter and half scales, indicating a rather high variability of the response for the hidden layers. Due to the use of nondeterministic algorithms and the stochastic aspect of the optimization, the fact that each run leads to a unique result is expected. Nevertheless, it is remarkable that every run leads to a completely different internal dynamics, being the features observed at hidden layers different in both shape and amplitude. This is represented by the uniform patches of high deviation. Second, it is clear that double-precision runs are more uniform, with deviation values lower by around one decade and regions of high deviation being sparser and smaller, indicating that the global behavior of the network is more similar despite using nondeterministic algorithms. Third, floating-point precision influence seems to be analogous at different simulation times. The distinctions between FP32 and FP64 just described are similar when having the beginning of the simulations as input (Figure 10) or at a later time (Figure 11). Note that in Figure 11, the ground truth inputs have been fed to the network, even for Frames 99–103, so that the error accumulation in time is not taken into account here, but will be studied in Section 4.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

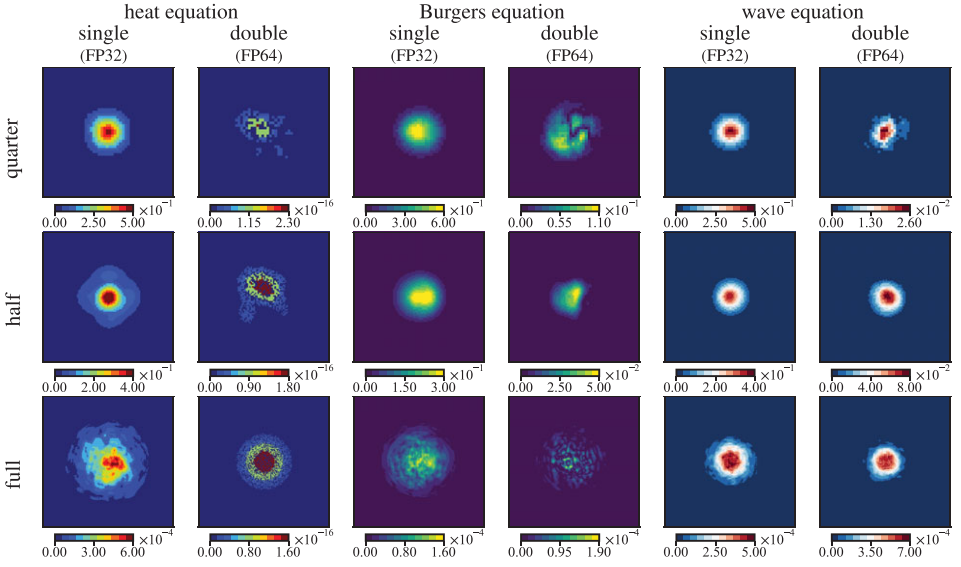

Figure 10. *Pixelwise deviation of the feature fields (last field of each scale) for a model input corresponding to the starting simulation of the Gaussian pulse benchmark, for runs with single and double precisions. Note that the magnitudes do not correspond to any physical quantity, and colormap ranges are different for each field, so the shapes of the features are visible.*

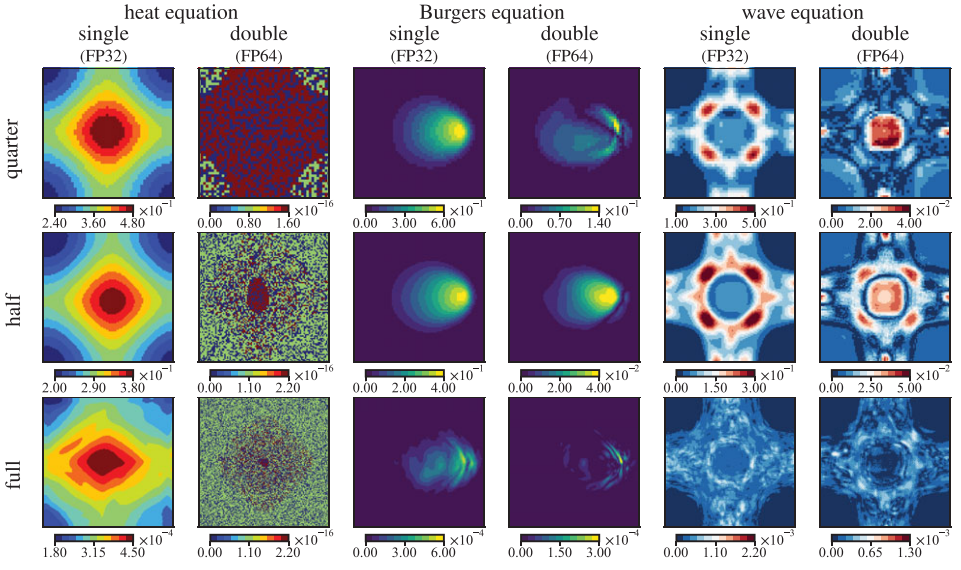

Figure 11. *Pixelwise deviation of the feature fields (last field of each scale) for a model input corresponding to the 99th frame of the Gaussian pulse benchmark, for runs with single and double precisions. Note that the magnitudes do not correspond to any physical quantity, and colormap ranges are different for each field, so the shapes of the features are visible.*

The amplitude of the deviation at the end of the full scale, corresponding to the output of the neural network, is comparable between single and double runs, with the exception of the heat equation where FP64 yields a deviation criterion close to the machine precision. For instance, the upper limit of the color

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

bar for the deviation of the Burgers' equation feature fields with start of the Gaussian pulse as input (Figure 10) is 1.60×10^{-4} for single and 1.90×10^{-4} for double precisions. A high level of similarity is expected at that stage of the CNN, since the output field is directly constrained by the loss function which had similar values (Figure 6). Nevertheless, such small differences are capable of changing the behavior of the solution when the analysis becomes recursive, aspect that is exemplified in Figure 1 and discussed and quantified in Section 4. It suggests that a potential strategy to enforce reproducibility in multiscale networks is to add a loss constraint for each scale output in order to constrain its internal dynamics. For instance, this could be achieved with a loss combining each scale $\mathcal{L} = \sum_i \mathcal{L}_i$ where $\mathcal{L}_i$ compares the scaled output with the corresponding downscaled target. However, such a technique could possibly affect the dynamics of the multiscale architecture and reduce the model's accuracy since the features do not necessarily need to reproduce the downscaled target.

4. Performance of Recurrent Estimation

Several models have been obtained from the nondeterministic training, leading to small differences of the output in single-prediction mode. This section investigates how the nondeterminism inherent to the training affects the long-time prediction in recurrent mode. This is evaluated by calculating the total loss (Equation (5)), considering the simulated and estimated fields, starting from the single-prediction mode (recurrence number $r = 0$), and reusing the output as a new input (recurrent mode, $r > 0$). Note that the recurrent fields are not used as targets when training the neural network, so the name "loss" is an extrapolation of its concept. Since the loss is calculated considering normalized fields, it is not a direct quantification of the estimation of the physical accuracy. To do so, the RMSE, issued from physical units, is also calculated:

$$\mathrm{RMSE}(r) = \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left[F_k(r) - \widetilde{F}_k(r)\right]^2}. \tag{9}$$

For convenience, the RMSE values are normalized by the pulse amplitude ε. The recursive analysis is performed for the nine benchmarks (Section 4.1) and the random pulse databases (Section 4.2).

4.1. Benchmarks test

The evolution of the total loss in the recurrent estimation of the benchmarks for the models trained with single and double precisions is presented in Figure 12. The generalization capacity of the neural network to capture the underlying dynamics is evidenced by the fact that a similar performance is obtained for features that were not present in the training database (e.g., negative and planar pulses in the case of wave propagation).

The loss (and consequently the error) remains relatively small even after 100 recurrences, showing the overall quality of the estimation through time. The curves are similar when comparing the runs with the same floating-point precision or the two groups of runs. For the wave equation, for instance, losses approach $\mathcal{L} = 0.01$ in the two graphs, that is, both single and double precisions resulted in similar overall accuracy for the tested benchmarks. The quality and similarity of the two groups of runs is also present when comparing the evolution of the error criterion RMSE, presented in Figure 13.

In spite of the general quality and similar behavior, an important spread exists among the different runs for the single-precision trainings. Table 3 presents the statistics of the maximum to minimum ratio for the 10 models available per physics and precision considering the recurrent analysis illustrated in Figure 13. Error max/min ratio peaks at 64 for single-precision trainings (heat equation and square pulse benchmark), and curves fluctuate around 5–10. For double precision, such a ratio is always lower than 12 for all the physics and benchmarks, with an average value ranging from 1 to 4.

In conclusion, both single and double precision models yield a similar behavior on the benchmarks, with a slightly better accuracy when using double precision. However, the comparison of several runs explicit that

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

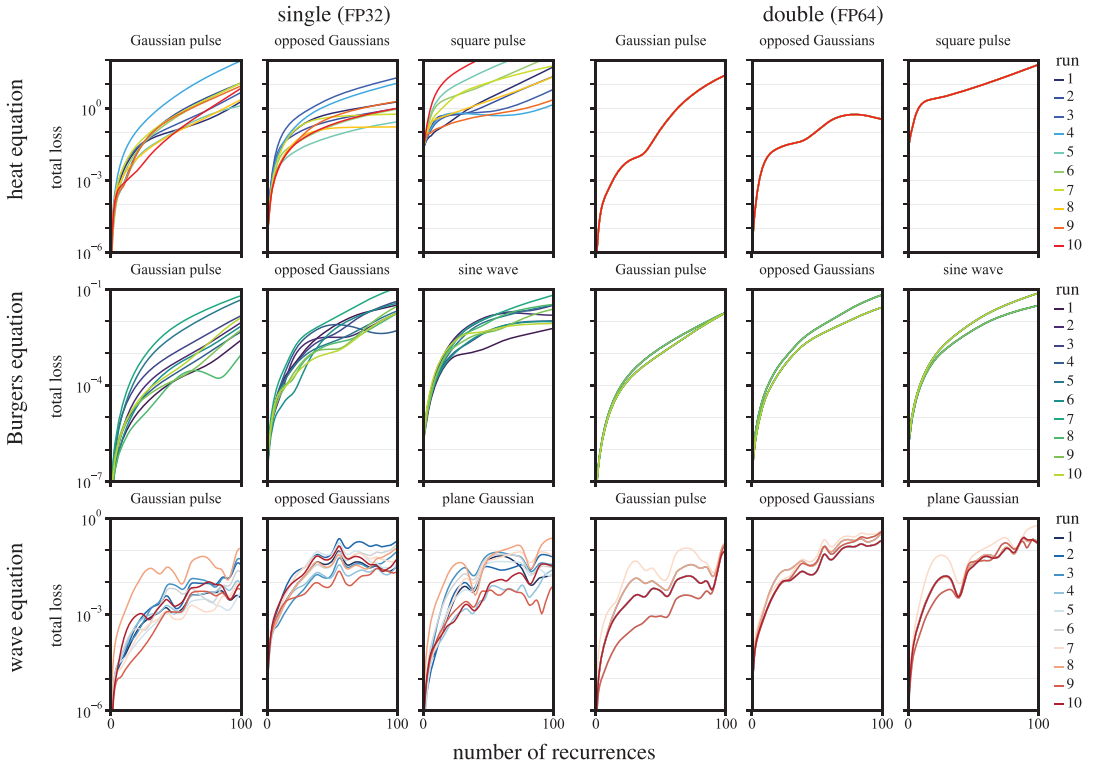

Figure 12. *Evolution of total loss for the recurrent test of benchmarks for models trained with single (left) and double (right) precisions; number of curves for double precision are reduced due to superposition of identical responses.*

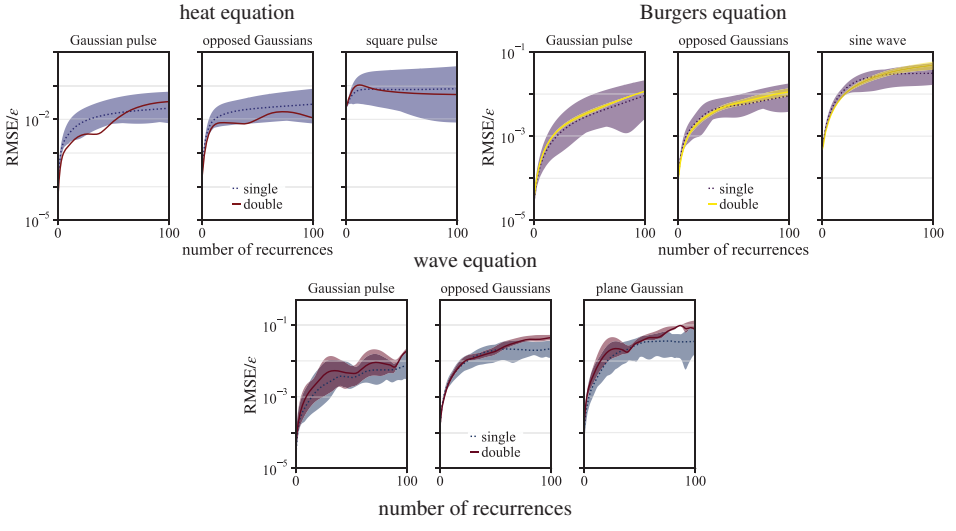

Figure 13. *Evolution of the root-mean-square error normalized by the pulse amplitude for the recurrent test of benchmarks for models trained with single (dotted line) and double (full line) precisions; central line represents the average among the runs, and band indicates the minimum to maximum range.*

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

Table 3. *Minimum, maximum, and average of the max/min ratio of the recursive root-mean-square errors considering the benchmarks from 0 to 100 recurrences.*

		Heat equation			Burgers' equation			Wave equation		
		GP	OG	SP	GP	OG	SW	GP	OG	PG
min	Single	2.705	1.657	1.135	1.985	1.809	1.300	3.154	1.320	1.897
	Double	1	1	1	1.001	1.250	1.064	1.213	1.276	1.042
max	Single	16.809	18.905	64.272	14.516	4.844	4.541	14.279	4.162	11.939
	Double	1	1	1	1.243	1.565	1.825	11.089	1.901	4.581
avg	Single	8.379	8.905	32.278	7.071	3.621	2.962	6.216	2.689	5.324
	Double	1	1	1	1.110	1.427	1.434	4.205	1.520	2.099

Abbreviations: GP, Gaussian pulse; OG, opposed Gaussian; PG, plane Gaussian; SP, square pulse; SW, sine wave.

a high variability is present, especially for models obtained with single precision. Results also highlight that the recursive strategy amplifies the differences due to the nondeterministic training, which makes reproducibility a crucial element to take into account when applying neural network to recursive tasks.

4.2. Random pulses test

A statistical approach is used to compare the recurrent test results for the random databases. For each simulation, the same recursive test used for the benchmarks database is performed. The evolution of the testing losses is represented for a subset of recurrences (0, 5, 10, 25, and 50) in Figure 14. Each boxplot describes the distribution of the loss for the 100 simulations that compose the test database, at the given number of recurrences.

For most of the models, the median loss at recurrence 0 (evaluation of the model as it was trained) is lower than the converging amplitude in training ($\mathcal{L}$ around 1.5×10^{-6}). For the heat equation, comparison of average losses at recurrence 0 returns a factor 3.58 between the worst (Run 3) and the best (Run 8) models trained with single precision and 1.00 for double precision (response is the same for all runs). For Burgers' equation, the values are 2.47 and 1.23. The wave equation is the one with the largest range, with ratios 14.42 and 6.03. Clearly, this variation is not a direct outcome of the training losses' ratios for the different runs (Table 2) and is another indication of the increased difficulty for the neural network to learn the wave equation case because of its hyperbolic nature.

The relationship between the models' training losses and their performance on recurrent mode is examined. Since the focus is on the accuracy of the estimation, RMSE is considered instead of the loss. In Figure 15, the average error and min–max bands for the 100 simulations in the database are plotted against training losses for recurrence numbers 0, 10, and 50. After several recurrences, small deviations observed during training, here quantified by the models' training losses, are significantly amplified. The recursive error variability is elevated, and no conclusions can be made in terms of a direct relationship between the variability and accuracy during the training and the inference. This means that using the validation or training loss as the sole criterion for inferring on a model's quality is not sufficient for the current cases.

Important conclusions can be drawn from the acoustic propagation results. The model trained with the biggest error both for the benchmarks (Figure 12), and the random pulse database (Figure 14) is Run 7 for both precision. This result indicates that using a few double-precision trainings does not guarantee a more precise network or lesser variability when compared to a unique or a few single-precision trainings. That is, even if statistically the models obtained with double precision will probably be less variable, mostly due to the repeatability of the models, the individual runs can result in an error range as big as in the case of single precision. As a general rule, authors recommend that, on the minimum, one must not consider a single model representative of the neural network capacity to model time-evolving systems, since only the GPU nondeterminism can produce significant changes in the models.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

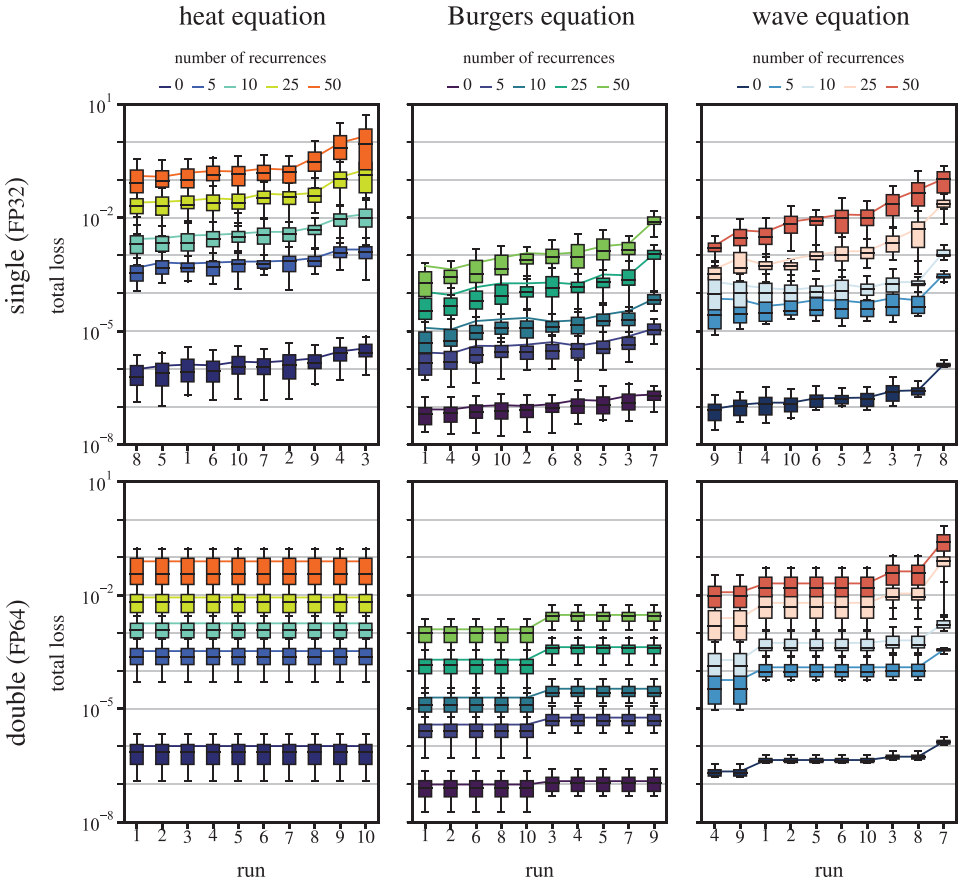

Figure 14. *Box plots of the total loss for the different models obtained with single precision (top) and double precision (bottom), considering the random pulse databases at multiple number of recurrences. The central lines indicate the median value, the box limits represent first and third quartiles, the whiskers represent the median ±1.5 times the interquartile range, and continuous lines connect the average for each run.*

In order to investigate if the precision used on training has an impact in the accumulation of error during the recursive inference, the scope of the analysis is extended by considering the complete set of models available per run (one in every 125 epochs, 12 models per run) in the wave equation case, and not only considering the best models per run as in the previous experiments. On this analysis, not shown here for conciseness, no clear distinction could be made in terms of amplification of the initial errors for models trained with single or double precision.

Further evaluation of the floating-point precision influence at the time of inference is performed by converting the models, that is, performing the inference with double precision for a model trained with single precision and vice versa. Figure 16 shows the evolution of the percentual disparity between the inference with the proper precision (the FP32 model evaluated in single precision and the FP64 model evaluated in double precision) and the flipped precision (the FP64 model evaluated in single precision—"FP64 as FP32"—and FP32 model evaluated in double precision—"FP32 as FP64") for the Burgers' and wave equations, when using the first run for each precision. Recurrent estimation until $r = 100$ of the benchmarks returned an absolute disparity in RMSE lower than 0.1% when compared to the inference performed with the proper precision. In other words, there is no apparent increase in error accumulation due to the use of single or double precision in testing. This indicates that inference may be

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

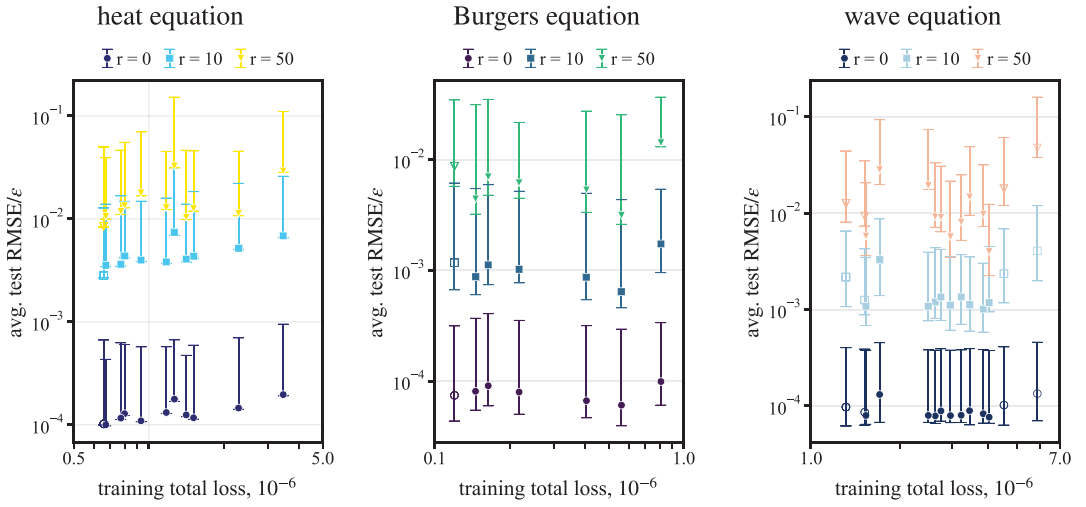

Figure 15. *Average recurrent test error versus validation loss for all simulations in random pulse databases at multiple numbers of recurrences (0, 10, and 50), filled markers for single precision and empty markers for double precision; error bars indicate minimum to maximum range.*

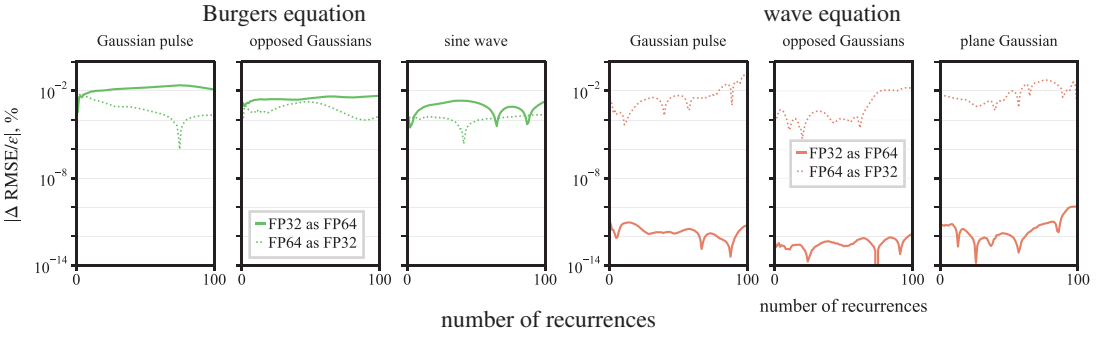

Figure 16. *Evolution of the absolute disparity of the root-mean-square error for the recurrent test of benchmarks for models when comparing inference with proper and flipped precisions (Run 1): trained in single and inference in double (FP32 as FP64, full line) and for trained in double and inference in single (FP64 as FP32, dotted line), for the Burgers' (left) equation and the wave (right) equation.*

performed with a lower precision with reduced impact in accuracy while gaining in performance (factor 1/3 in calculation time for the current application).

5. Conclusions

Reproducibility of a fully convolutional neural network used to model time-evolving physical systems (heat, Burgers', and wave equations) has been assessed by performing the same training several times under GPU nondeterminism. The use of deterministic calculations is, up to now, not always available and may also not be consistent among distinct hardware. Since the training nondeterminism is associated with the truncation error, tests are performed with single (32-bit) and double (64-bit) precision.

Runs with single precision revealed a large variation in terms of parameters and results. For the current test cases, the use of double precision was capable of limiting the variability of kernels by a factor 2, and constrained the error in the recurrent analysis, while slightly increasing the models accuracy. Compared to

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

the single-precision runs, computation cost was multiplied by 2 in terms of memory and by 3 for the training time. This important gain in cost may be prohibitive for more complex networks and equations, such as in the evaluation of three-dimensional problems, and a hybrid strategy—multi- or mixed-precision (Micikevicius et al., 2018)—could also be envisaged. In addition, it was observed that performing training with double and inference with single precision resulted in negligible loss of accuracy.

The test with a broader number of simulations showed that there is an important range of variation associated with the different models. Although the overall quality of the approximation is almost unmodified when training with single or double precision, the latter produces models that return more uniform results when tested, what is directly associated with its similarity at null recurrence. However, there is no guarantee that the variability of models will result in less variability of results, unless all models are unique.

The results presented here are for a single neural network architecture, a natural extension of this work would be performing similar analysis for different architectures. Candidate strategies to the increase of the precision that could also reduce the variability, but were not tested in current work, are: to impose hidden layer reproducibility by additional terms to the loss function, effectively constraining intermediary results with their corresponding target fields; the use of physics informed network (Meng et al., 2020); or the use of "long-term loss" (Tompson et al., 2017). Furthermore, the analysis can be extended by evaluating whether the variability and the influence of the floating-point precision are similar with an RNN, such as an LSTM architecture. The hybridization of CNN predictions with classical simulation tools (Ajuria-Illarramendi et al., 2020; Um et al., 2020; Özbay et al., 2021) could also limit the variability of results, in particular for long times, even if still submitted to model variability.

The performed analysis has shown a high variability associated with a typical fully CNN model, trained to reproduce time-evolving fields. Based on the highlighted behavior, it is recommended that analogous work should be performed whenever possible when modeling physical systems, where small deviations may lead to divergences or the onset of instabilities. This is reinforced by the apparent absence of direct relationship between learning phase metrics and the inference errors among the best models obtained on each run. Reproducibility must be given a continuous focus during the development of data-driven physical surrogates, particularly if one aims at producing robust, application-oriented models such as solver accelerators.

Acknowledgment. The authors are grateful to Stephane Moreau and Marc Jacob for the fruitful discussions.

Data Availability Statement. Code and instructions for generating the databases can be found in https://gitlab.isae-supaero.fr/daep/cnn-multiphysics-repro.

Author Contributions. Conceptualization, Data curation, Formal Analysis, Investigation, Visualization, and Writing—original draft: W.G.P.; Methodology, Software, and Validation: W.G.P and A.A.; Funding acquisition, Project administration, Resources, and Supervision: M.B.; Writing—review & editing: W.G.P, M.B., and A.A. All authors approved the final submitted draft.

Funding Statement. This research was supported by grants from the French Government Defense procurement and technology agency Directorate General of Armaments (DGA) in the project POLA3; computations have been performed in the CALcul en MIdi Pyrénées (CALMIP) calculation center of Toulouse, France, under the project P20035.

Competing Interests. The authors have no competing interests.

References

Ajuria Illarramendi E, **Alguacil A**, **Bauerheim M**, **Misdariis A**, **Cuenot B, and Benazera E** (2020) Towards an hybrid computational strategy based on Deep Learning for incompressible flows. AIAA AVIATION 2020 FORUM. https://doi.org/10.2514/6.2020-3058

Alguacil A, **Bauerheim M**, **Jacob MC and Moreau S** (2021a) Predicting the propagation of acoustic waves using deep convolutional neural networks. *Journal of Sound and Vibration 512*, 116285.

Alguacil A, **Pinto W G**, **Bauerheim M**, **Jacob MC and Moreau S** (2021b) Effects of Boundary Conditions in Fully Convolutional Networks for Learning Spatio-Temporal Dynamics. In: Dong, Y., Kourtellis, N., Hammer, B., Lozano, J.A. (eds) *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track. ECML PKDD 2021. Lecture Notes in Computer Science()*, vol *12979*. Springer, Cham. https://doi.org/10.1007/978-3-030-86517-7_7

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

Berg J and Nyström K (2018) A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing 317*, 28–41.

Bouthillier X, **Laurent C and Vincent P** (2019) Unreproducible research is reproducible. In Chaudhuri K and Salakhutdinov R (eds), *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, Proceedings of Machine Learning Research, Vol. *97*, pp. 725–734.

Brunton SL, **Noack BR and Koumoutsakos P** (2020) Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics 52*(1), 477–508.

Chou YH, **Ng C**, **Cattell S**, **Intan J**, **Sinclair MD**, **Devietti J**, **Rogers TG and Aamodt TM** (2020) Deterministic atomic buffering. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Athens, Greece. IEEE.

Dadvar M and Eckert K (2018) Cyberbullying detection in social networks using deep learning based models; a reproducibility study. arXiv preprint arXiv:*1812*.08046.

Denton EL, **Chintala S**, **Szlam A and Fergus R** (2015) Deep generative image models using a Laplacian pyramid of adversarial networks. In Cortes C, Lawrence N, Lee D, Sugiyama M and Garnett R (eds),*Advances in Neural Information Processing Systems*, Vol. *28*. Curran Associates, Inc., Red Hook, New York, United States, pp. 1486–1494.

Fidler F and Wilcox J (2021) Reproducibility of scientific results. In Zalta EN (ed.), *The Stanford Encyclopedia of Philosophy*, Summer 2021 Edn. Metaphysics Research Lab, Stanford University, Stanford, USA.

Fotiadis S, **Pignatelli E**, **Valencia ML**, **Cantwell C**, **Storkey A and Bharath AA** (2020) Comparing recurrent and convolutional neural networks for predicting wave propagation. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, virtual, April 26, 2020. ICLR.

Goodman SN, **Fanelli D and Ioannidis JPA** (2016) What does research reproducibility mean? *Science Translational Medicine 8* (341), 341ps12.

Iakymchuk R, **Defour D**, **Collange C and Graillat S** (2016). Reproducible and Accurate Matrix Multiplication. In: Nehmeier, M., Wolff von Gudenberg, J., Tucker, W. (eds) *Scientific Computing, Computer Arithmetic, and Validated Numerics. SCAN 2015. Lecture Notes in Computer Science*, vol *9553*. Springer, Cham. https://doi.org/10.1007/978-3-319-31769-4_11

Ajuria Illarramendi, E, **Alguacil, A**, **Bauerheim, M**, **Misdariis, A**, **Cuenot, B, and Benazera, E** (2020) Towards an hybrid computational strategy based on Deep Learning for incompressible flows. AIAA AVIATION 2020 FORUM. https://doi.org/10.2514/6.2020-3058

Ivie P and Thain D (2018) Reproducibility in scientific computing. *ACM Computing Surveys 51*(3), 1–36.

Jézéquel F, **Lamotte JL and Saïd I** (2015). Estimation of numerical reproducibility on CPU and GPU. In M. Ganzha, L. Maciaszek, M. Paprzycki (eds), *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*. ACSIS, Vol. *5*, pp. 675–680.

Jorda M, **Valero-Lara P and Pena AJ** (2019) Performance evaluation of cuDNN convolution algorithms on NVIDIA Volta GPUs. *IEEE Access 7*, 70461–70473.

Krüger T, **Kusumaatmaja H**, **Kuzmin A**, **Shardt O**, **Silva G and Viggen E** (2017) *The Lattice Boltzmann Method*, Graduate Texts in Physics. Springer, Cham.

Latt J, **Malaspinas O**, **Kontaxakis D**, **Parmigiani A**, **Lagrava D**, **Brogi F**, **Belgacem MB**, **Thorimbert Y**, **Leclaire S**, **Li S**, **Marson F**, **Lemus J**, **Kotsalos C**, **Conradin R**, **Coreixas C**, **Petkantchin R**, **Raynaud F**, **Beny J and Chopard B** (2020) Palabos: Parallel lattice Boltzmann solver. *Computers & Mathematics with Applications 81*, 334–350.

Lee S and You D (2019a) Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics 879*, 217–254.

Lee S and You D (2019b) Mechanisms of a convolutional neural network for learning three-dimensional unsteady wake flow. In *72nd Annual Meeting of the APS Division of Fluid Dynamics*, Seattle, Washington, November 23–26, 2019, American Physical Society.

Li H, **Xu Z**, **Taylor G**, **Studer C and Goldstein T** (2018) Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems 31*.

Li L and Talwalkar A (2020) Random search and reproducibility for neural architecture search. In Adams RP and Gogate V (eds), *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, Proceedings of Machine Learning Research, 115. Tel Aviv: PMLR, pp. 367–377.

Lu L, **Meng X**, **Mao Z and Karniadakis GE** (2021) DeepXDE: A deep learning library for solving differential equations. *SIAM Review 63*(1), 208–228.

Mathieu M, **Couprie C and LeCun Y** (2016) Deep multi-scale video prediction beyond mean square error. In BengioY and LeCunY (eds), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*.

Meng X, **Li Z**, **Zhang D and Karniadakis GE** (2020) PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering 370*, 113250.

Micikevicius P, **Narang S**, **Alben J**, **Diamos G**, **Elsen E**, **Garcia D**, **Ginsburg B**, **Houston M**, **Kuchaiev O**, **Venkatesh G and Wu H** (2018) Mixed precision training. In *International Conference on Learning Representations, ICLR 2018*, Vancouver, Canada, Apr 30 May 3, 2018. ICLR.

Nagarajan P, **Warnell G and Stone P** (2018) The Impact of Nondeterminism on Reproducibility in Deep Reinforcement Learning. In *2nd Reproducibility in Machine Learning Workshop at ICML 2018*, Stockholm, Sweden, July 15, 2018. ICML.

https://doi.org/10.1017/dce.2022.18 Published online by Cambridge University Press

NVIDIA Corporation (2020a) CUDA Toolkit Documentation v11.1.0: 2.1.4. Results reproducibility. Available at https://docs.nvidia.com/cuda/archive/11.1.0/cublas/index.html#cublasApi_reproducibility (accessed 18 January 2021).

NVIDIA Corporation (2020b). cuDNN Developer Guide: Chapter 8. Reproducibility (determinism). Available at https://docs.nvidia.com/deeplearning/cudnn/pdf/cuDNN-Developer-Guide.pdf (accessed 18 January 2021).

Özbay AG, **Hamzehloo A**, **Laizet S**, **Tzirakis P**, **Rizos G and Schuller B** (2021) Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh. *Data-Centric Engineering 2*, E6. https://doi.org/10.1017/dce.2021.7

Paszke A, **Gross S**, **Massa F**, **Lerer A**, **Bradbury J**, **Chanan G**, **Killeen T**, **Lin Z**, **Gimelshein N**, **Antiga L**, **Desmaison A**, **Kopf A**, **Yang E**, **DeVito Z**, **Raison M**, **Tejani A**, **Chilamkurthy S**, **Steiner B**, **Fang L**, **Bai J and Chintala S** (2019) PyTorch: An imperative style, high-performance deep learning library. In Wallach H, Larochelle H, Beygelzimer A, d Alché-Buc F, Fox E and Garnett R (eds), *Advances in Neural Information Processing Systems*, Vol. *32*. Curran Associates, Inc., Red Hook, New York, United States, pp. 8024–8035.

Peng RD (2011) Reproducible research in computational science. *Science 334*(6060), 1226–1227.

Pineau J, **Vincent-Lamarre P**, **Sinha K**, **Lariviére V**, **Beygelzimer A**, **d'Alché Buc F**, **Fox E, and Larochelle H** (2021) Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research*, Vol. *22*, No. 164 p. 1–20.

Raff E (2019) A step toward quantifying independently reproducible machine learning research. In Wallach H, Larochelle H, Beygelzimer A, d Alché-Buc F, Fox E and Garnett R (eds), *Advances in Neural Information Processing Systems*, Vol. *32*. Curran Associates, Inc. Red Hook, New York, United States.

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

Renard F, **Guedria S**, **Palma ND and Vuillerme N** (2020) Variability and reproducibility in deep learning for medical image segmentation. *Scientific Reports 10*, 13724.

Ronneberger O, **Fischer P and Brox T** (2015) U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, pp. 234–241.

Seznec M, **Gac N**, **Ferrari A and Orieux F** (2018) A study on convolution using half-precision floating-point numbers on GPU for radio astronomy deconvolution. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, Cape Town, South Africa. IEEE.

Sirignano J and Spiliopoulos K (2018) DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics 375*, 1339–1364.

Sorteberg W, **Garasto S**, **Pouplin A**, **Cantwell C and Bharath AA** (2018) Approximating the solution to wave propagation using deep neural networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada. NeurIPS Foundation.

Tompson J, **Schlachter K**, **Sprechmann P and Perlin K** (2017) Accelerating Eulerian fluid simulation with convolutional networks. In Precup D and Teh YW (eds), *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70. Sydney: International Convention Centre and PMLR, pp. 3424–3433.

Torch Contributors (2019) PyTorch documentation. Available at https://pytorch.org/docs/stable/generated/torch.set_deterministic.html (accessed 18 January 2021).

Um K, **Brand R**, **Fei Y**, **Holl P and Thuerey N** (2020) Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. In Larochelle H, Ranzato M, Hadsell R, Balcan MF and Lin H (eds), *Advances in Neural Information Processing Systems*, Vol. *33*. Curran Associates, Inc., Red Hook, New York, United States, pp. 6111–6122.

Weller HG, **Tabor G**, **Jasak H and Fureby C** (1998) A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics 12*(6), 620.

Whitehead N and Fit-Florea A (2011) Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs. Technical report, NVIDIA Corporation.

Cite this article: Pinto WG, Alguacil A and Bauerheim M (2022). On the reproducibility of fully convolutional neural networks for modeling time–space-evolving physical systems. *Data-Centric Engineering*, 3, e19. doi:10.1017/dce.2022.18