Computational Mechanics (2023) 71:1227–1249
https://doi.org/10.1007/s00466-023-02295-x

ORIGINAL PAPER

U^p-Net: a generic deep learning-based time stepper for parameterized spatio-temporal dynamics

Merten Stender[1,2] · Jakob Ohlsen[2] · Hendrik Geisler[3] · Amin Chabchoub[6,7,8] · Norbert Hoffmann[2,4] · Alexander Schlaefer[5]

Received: 2 September 2022 / Accepted: 17 February 2023 / Published online: 24 March 2023

Abstract
In the age of big data availability, data-driven techniques have been proposed recently to compute the time evolution of spatio-temporal dynamics. Depending on the required a priori knowledge about the underlying processes, a spectrum of black-box end-to-end learning approaches, physics-informed neural networks, and data-informed discrepancy modeling approaches can be identified. In this work, we propose a purely data-driven approach that uses fully convolutional neural networks to learn spatio-temporal dynamics directly from parameterized datasets of linear spatio-temporal processes. The parameterization allows for data fusion of field quantities, domain shapes, and boundary conditions in the proposed U^p-Net architecture. Multi-domain U^p-Net models, therefore, can generalize to different scenes, initial conditions, domain shapes, and domain sizes without requiring re-training or physical priors. Numerical experiments conducted on a universal and two-dimensional wave equation and the transient heat equation for validation purposes show that the proposed U^p-Net outperforms classical U-Net and conventional encoder–decoder architectures of the same complexity. Owing to the scene parameterization, the U^p-Net models learn to predict refraction and reflections arising from domain inhomogeneities and boundaries. Generalization properties of the model outside the physical training parameter distributions and for unseen domain shapes are analyzed. The deep learning flow map models are employed for long-term predictions in a recursive time-stepping scheme, indicating the potential for data-driven forecasting tasks. This work is accompanied by an open-sourced code.

Keywords Partial differential equations · Machine learning · Wave propagation · Representation learning · Time integration · Sensor data fusion

1 Introduction

Numerical solutions schemes for spatio-temporal problems have matured over past decades, enabling highly accurate simulations of complex dynamics and geometries, e.g. numerical wind tunnel testing, wave propagation in solids, weather predictions, and meteorology [1,2]. Major computational speedups in numerical simulations have been obtained through massive parallelization and utilization of GPUs, such as in Lattice-Boltzmann methods [3]. However, there remain complex cases that require timely predictions of the future

Merten Stender and Jakob Ohlsen have contributed equally to this work.

✉ Merten Stender
merten.stender@tu-berlin.de
https://www.tu.berlin/cpsme

✉ Jakob Ohlsen
jakob.ohlsen@tuhh.de

[1] Cyber-Physical Systems in Mechanical Engineering, Technische Universität Berlin, Berlin, Germany

[2] Hamburg University of Technology, Dynamics Group, Hamburg, Germany

[3] Institute of Continuum Mechanics, Leibniz University Hanover, Hanover, Germany

[4] Department of Mechanical Engineering, Imperial College London, London, UK

[5] Institute of Medical Technology and Intelligent Systems, Hamburg University of Technology, Hamburg, Germany

[6] Hakubi Center for Advanced Research, Kyoto University, Kyoto, Japan

[7] Disaster Prevention Research Institute, Kyoto University, Kyoto, Japan

[8] Centre for Wind, Waves and Water, The University of Sydney, Sydney, Australia

state, such as accurate weather predictions, or short-time wave field predictions for seakeeping operations at offshore wind turbines. Despite algorithmic advances, such cases can be still computationally too demanding for timely predictions [4] and thus, hinder the development of several applications and technology with direct potential for societal benefits. In different scenarios, some problem needs to be evaluated for a large number of minor parameter variations, e.g. inside optimization loops, where solving the underlying equations becomes computationally infeasible. Instead, it would be desirable to utilize methods that make implicit use of the joint structure of all the individual problems. Moreover, there are situations for which the underlying governing equations or their parameterization are uncertain or unknown, such as seismic wave propagation [5], shear waves in inhomogeneous tissues [6,7], or the spatio-temporal evolution of epidemic outbreaks [8]. Here, a solely data-driven method would allow us to learn from raw observations and make efficient future state predictions without explicitly modeling the underlying processes as done in standard data assimilation schemes.

This work proposes a purely data-driven deep learning-based time stepper for spatio-temporal dynamics inspired by recent advances in computer vision [9,10]. Domain state snapshots at preceding time instances are fed to a U-Net-type [10] convolutional network to infer the dynamics at the next time step, i.e. to build a deep learning flow map. To consider parameterized problems and multi-domain inputs, a novel architecture denoted as U^p-Net is proposed that features multiple network encoding paths for fusing multiple data sources. Thereby, domain geometries, boundary conditions, and spatial parameter distributions, such as material properties, can be considered explicitly by feeding them into the p different encoder paths. The fully convolutional architecture allows for inference at variable domain dimensions without requiring model re-training if domain sizes differ across prediction samples. Compared to recent physics-informed learning approaches [11–13], no a priori knowledge of mathematical structures underlying the problem at hand is required. This property renders the U^p-net particularly promising for experimental data, where knowledge about governing equations and their parameters may only partly be available. The robustness and performance of the U^p-Net approach are demonstrated for typical cases of unsteady heat flow and wave dynamics in complex geometries. The generalization behavior for out-of-distribution physical parameters and different domain shapes are analyzed at the example of dedicated validation cases.

This work is structured as follows: Sect. 2 gives a short review of current approaches in the field of data-driven prediction for spatio-temporal dynamical systems. The proposed U^p-Net method and the recursive time stepping scheme are introduced in Sect. 3. Several numerical experiments are presented in Sect. 4. The complete source code for the project and multimedia visualizations are made available at https://github.com/TUHH-DYN/DeepStep under the GNU General Public License v3.0. The results presented in this work are obtained by using version 1.0.0 of the source code [14].

2 State-of-the-art and related work

Classical numerical schemes for time-stepping of dynamical processes are often based on Taylor series expansions for approximating temporal and spatial derivatives of governing equations. The mathematical description of the underlying physics is completely known, and a system's state at some future time instance $t + \Delta t$ is derived from the preceding states using implicit or explicit methods such as Runge–Kutta-type integrators, for which convergence properties, stability conditions, and error estimates are available. When applied to partial differential equations (PDEs), computational costs arise (i) from the massive number of states resulting from the spatial discretization (easily reaching millions of cells for fluid systems), (ii) from highly nonlinear relations between those states, and (iii) from the wide range of time and length scales to be resolved by the simulation.

Following the advances in the fields of machine learning (ML), physics-informed neural networks (PINNs) [15–17] have been proposed to overcome some of the computational challenges in classical numerical dynamics, solve inverse problems, and enable fast inference for predicting future states. Weakly-supervised PINNs aim at strongly regularizing universal function approximators [18] to physically consistent relations between sets of input features and target quantities utilizing governing equations, hence injecting domain expertise into black-box learning techniques. Methods for inferring symbolic differential equations from data, i.e. solving the inverse problem, have been proposed using sparse regression [19,20], compressive sensing [21], and Koopman analysis [22]. Most of these methods require rather strong assumptions and a priori knowledge of the underlying mathematical structure and focus on parametric identifications [23]. Once identified, the system descriptions can be used for time-stepping a set of initial conditions into the future [24], either using ML techniques or classical numerical schemes.

Purely data-driven approaches without any physical priors span the range of solving steady-state [25–29] cases and time-stepping dynamic cases [13,20,30–33]. For example, Thuerey et al. [27] derived the associated pressure field from a velocity field around an airfoil, thereby providing a static solution for the 2D Reynolds-Averaged Navier–Stokes equations. Farimani et al. [25] proposed an adversarial training approach to infer static solutions for two-dimensional physics problems, such as steady-state heat diffusion and incompressible fluid flow, from observations without priors

on the underlying physical process and for arbitrary boundary conditions. Guo et al. [31] compute the non-uniform steady laminar flow around 2D or 3D objects using convolutional neural networks. For ordinary differential equations (ODEs), data-driven time integration schemes [13,20] often couple priorly learned flow maps into classical multi-step integration schemes. Data-driven time steppers for PDEs tend to align with computer vision techniques, such as image translation, owing to the high dimensionality of the problems [33]. For example, [34] solve temporal evolutions of 3D fluid flows through deep convolutional-recurrent neural networks and report computational speedups of two orders of magnitudes compared to conventional numerical solvers. Other approaches propose to learn multi-scale flow maps following the ideas of para-real time stepping [35], or replace compute-intensive parts of PDE solvers with convolutional neural networks [33,36].

Despite many promising results and various research directions, one may identify four conceptual limitations to most of the current data-driven time-stepping methods for spatio-temporal dynamics regarding their applicability, generality, and generalization abilities for different computational scenes:

1. *Physical priors* A class of methods [15–17] requires rather exact priors, such as a mathematical formulation of underlying principles, or knowledge of the problem parameterization in the form of boundary conditions or material models. While enabling small-data capability and consistency with physics, these priors may exclude some cases, such as waves and deformations in tissues or soils, with partly unknown governing equations and highly inhomogeneous material properties that cannot be described analytically.
2. *Scene parameterization* Other classes of methods are built for very specific scenes [25–27,37], such as laminar flow around an airfoil. Whenever the scene is altered, e.g. changing the air's density, the airfoil's shape, or location in the domain, re-training the model is required because the scene information is not explicitly considered in the model. Very limited practical generalization properties may result from these kinds of modeling approaches. Ideally, a well-generalizing model trained on airfoil flow samples would be able to predict the flow around a cylinder without being re-trained.
3. *Finite dimensions* Whenever perceptrons, recurrent neurons, or other non-convolutional building blocks define the machine learner [25,27], the geometric resolution and shape of the computational domain are fixed per definition, e.g. through the number of neurons in the input and output layers. Only fully-convolutional neural (FCN) architectures can generalize to arbitrarily sized field inputs and outputs, and can thereby encode translational or rotational invariance through representation learning. As a result, an FCN model can be trained and deployed on arrays of different sizes and aspect ratios without any complicated (or lossy) data pre-processing steps.
4. *Simplifications* Numerical computations and physics-informed learning approaches, in many cases, rely on several simplifications (simplified models, linearizations, symmetries, etc.). Their choice is a trade-off between reduced computational cost, accuracy, or solution complexity and tends to require a lot of problem-specific expertise.

The main difficulty related to purely data-driven approaches, such as the one proposed in this work, is to obtain locally highly accurate results, i.e. amplitudes, phases, and gradients, which fit into the overall bigger picture, e.g. energy conservation throughout the complete domain. These spatial and temporal multi-scale requirements call for neural network architectures that inherently encode multi-resolution prediction stages, such as a U-Net [10]. Our novel multi-path data fusion U^p-Net architecture addresses the aforementioned limitations and aims at a generic methodology that can be customized to specific cases at the cost of limited physics integration and larger data requirements. An FCN architecture ensures generalization to differently sized computational domains, translational and rotational invariance, as well as explicit consideration of spatial and temporal neighborhood relationships in direct analogy with classical numerical differentiation schemes. For a generic parameterization of the problem, we propose to adapt the U-Net architecture featuring a single encoder path, and instead connect multiple encoder paths, thereby making scene parameterization possible through data fusion. The U^p-Net can be trained to learn a flow map by advancing a dynamic field quantity one time step into the future and ensuring the arrival at time-stepping based on raw observations of parameterized spatio-temporal processes.

3 Methodology

The proposed methodology, as schematically shown in Fig. 1, is composed of two integral parts: first, a flow map is assimilated by a U^p-Net deep learning model that advances the current state of the dynamical system one time step into the future. We highlight that the model is designed for fusing multivariate data inputs at multi-scale spatial resolutions. Second, the flow mapping model is embedded into a simplistic recursive time stepping scheme allowing for making long-term predictions from an initial state. The physics of the problem at hand binds the two components together, e.g. by requiring a specific time increment given the spa-

tial resolution of the data fed into the flow mapper. However, individual aspects of the two parts can be optimized independently, thereby leaving space for future work building on the results presented here.

3.1 Encoder–decoder model architecture

The convolutional network is set up to propagate parameterized field variables one step in time. Particularly, the network architecture can be regarded as an adaptation of the U-Net architecture [10]. We will therefore shortly revisit the U-Net architecture first, and then extend it to the proposed U^p-Net architecture. The original U-Net is built on a fully convolutional [38] feedforward encoder–decoder model with skip connections between the encoder and the decoder paths $\mathcal{P}_{\text{enc}}$, $\mathcal{P}_{\text{dec}}$ respectively. The original U-Net is composed of consecutive convolution and max-pooling layers acting on the input data, i.e. $V_i = \text{conv}(V_{i-1})$, $V = \text{conv}(\text{conv}(\dots(X)))$ in the encoding path. Each convolution layer i reduces the spatial resolution of the input X, thereby extracting representative features V_i at different spatial resolutions i. The decoding path is a series of up-convolution layers, i.e. $Y_i = \text{upconv}(Y_{i-1})$, $Y = \text{upconv}(\text{upconv}(\dots(V)))$ that increase the spatial resolution of the smallest feature map V back to the output Y of original input size. The compression and inflation actions represent a simple encoder–decoder structure typically used for feature extraction, noise filtering and other applications. The central aspect of the U-Net is to fuse feature maps V_i from each stage in the encoder path into the same level Y_i of the decoder path, thereby allowing information to escape the compression action and directly link with the high-resolution stages in the decoder. Thereby, small but important features of the input can propagate into the model output without getting lost during compression. Considering only a single encoder in Fig. 1, for example only the blue-colored layers, represents a classical U-Net. The encoder path for analysis and contraction enables the creation of higher-level representations in the form of multi-scale feature maps, while the symmetric decoder path for synthesis and expansion samples up those abstractions to re-obtain the original image size resolution at the output.[1] The skip connections combine feature maps from the encoding layers with feature maps from decoding layers of the same resolution, allowing essential fine-resolution details to be propagated into the network's output layer. The encoder and decoder path are symmetric in terms of their spatial resolution and have k stages. The network is designed such that successive stages enlarge the receptive field (at lower spatial resolution) and increase the number of feature channels. The data flow through the last stage, i.e. the network's bottleneck, represents the semantic pathway, i.e. requiring the network to learn the scene on global scale. On the contrary, the data flow through skip connections represents the geometric pathway which injects geometric precision into the decoding path and thereby increases the level of details in the decoder feature maps. U-Net-based approaches to spatio-temporal processes have been studied earlier, such as by Farimani et al. [25] and Thuerey et al. [27]. Fotaidis et al. [39] employed U-Nets and convolutional recurrent networks for surface wave predictions, Sharma et al. [26] computed the steady-state solution of the heat equation using a U-Net, while de Bézenac et al. [40] applied a U-Net for sea surface temperature predictions. As this work studies linear PDEs, it is important to note that U-Net-type methods have also been used for nonlinear PDEs, such as [25,27,29] for the Navier–Stokes equations. Most existing approaches however are limited in (i) their capabilities to fusing multiple data sources and (ii) at the same time employ the final models as a flow map for time stepping applications.

3.2 U^p-Net: data fusion through multi-domain inputs

We propose to make use of multiple input encoding paths, which jointly link to a single output decoding path. The U^p-Net results from creating p different encoder paths for different input fields and connecting them through skip connections at each stage to the decoder path. Figure 1 depicts a schematic of the proposed U^p-Net architecture for $p = 3$ paths and $k = 3$ stages. A U^3-Net is built on the example of taking field variables U, the domain geometry in G, and a spatial domain material distribution C as input. Naturally, this architecture can be adapted to more encoding paths at the cost of more model parameters θ. Even though it would be possible to stack all different input fields into a single tensor and feed it through a conventional U-Net architecture, our results indicate that the generation of separate paths for different field quantities can significantly improve the model precision at the same number of model parameters, see results in Sect. 4. The core objective for taking qualitatively different input fields into account is a major increase of generalization capabilities of the U^p-Net compared to the classical U-Net. Following the review of related work presented earlier, some current U-Net approaches for parametric PDE applications tend to lack the ability to generalize different scene configurations, i.e. other domain shapes and domain parameterizations. Other related methods do not build on a fully convolutional structure, and are thereby strongly limited to a single spatial resolution. Hence, these approaches require model retraining once predictions are required for a different scene. As the U^p-Net explicitly takes that parameterization as input, our work shows that this type of fully convolutional

[1] If unpadded convolutions are utilized, the feature map dimension is smaller than the input dimension and may require adequate post-processing in the skip connections and at the output layer.

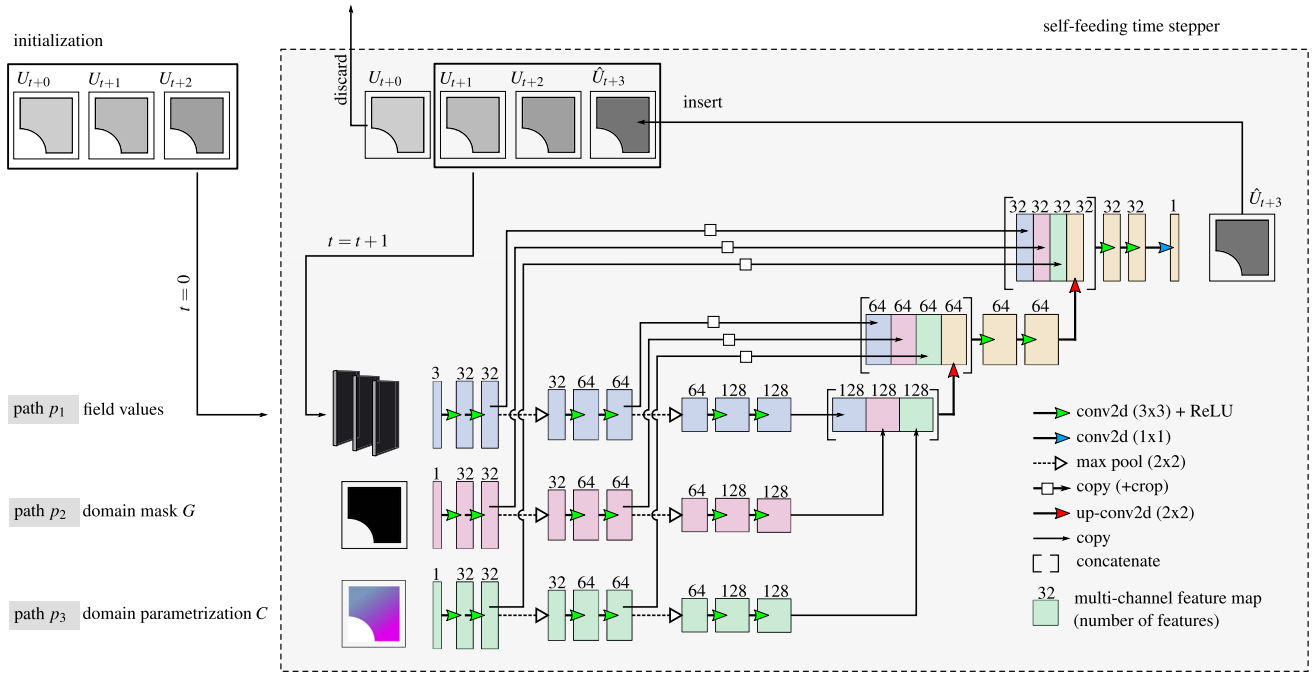

Fig. 1 Schematic of the proposed U^p-Net flow map architecture ($p = 3$ encoder paths, $k = 3$ stages) embedded into a self-feeding loop for time stepping a set of initial field states into the future. Multiple encoder paths feed different input data sources to the network for data fusion, hence enabling generalization of the time stepper to different domain geometries and domain parameterizations without requiring re-training

neural architecture can be well-generalized to new scenes. The additional cost for generalization capabilities lies in the requirement for larger training data spanning various scene parameterizations, thus resulting in longer model training.

3.3 U^p-Net design considerations

The network's prediction performance depends on a plethora of model hyperparameters and architecture choices. In the course of this work, various studies have suggested that there are four crucial ingredients to designing and training an efficient U^p-Net time stepper: the selection of the loss function, data normalization, conscious treatment of convolution operations, and consideration of boundary conditions. We use a domain-adaptive loss function, i.e. the loss is evaluated only inside the physical domain Ω based on the input domain mask G. Sample-wise data normalization, e.g. limiting field variables to a range of $[0, 1]$, is found be a crucial ingredient for accelerating the training process, at the same time resulting in weight regularization. However, as the network's kernels are enforced to learn amplitude-independent features, i.e. gradients, specific nonlinear physical process will require a different kind of normalization. In situations where a direct normalization per sample is not possible, e.g. when nonlinearities govern the dynamical process, we found that a global re-scaling of the complete training set will enhance training performance. Other works [41,42] on U-Nets suggested to use batch normalization, which we did not find as important as sample-wise normalization. Padded convolutions retain the input shape through zero-padding the input before the convolution operation. However, physical processes lack an intuitive padding option: when considering a complex wave field as input, padding the respective array with zeros would be somewhat obscure as it would introduce some physically inconsistent data. This work employs valid convolutions, hence requiring cropping of the feature maps in the skip connections as proposed originally for the U-Net [10]. Furthermore, physically consistent padding of the output field is required before re-injecting into the recursive time stepping scheme. A direct way to achieve physics-consistent padding is to request the domain Ω to have a sufficient amount of boundary pixels, such that the convolution-induced cropping will only erase features located outside the physical domain, which can then easily be padded with constants. Time-invariant Dirichlet boundary conditions are considered by enforcing $u(x,t) = h_{\text{Dirichlet}}(x)$ for all $x \in X^{\delta\Omega}$ in the input path. Accordingly, the predicted values $\hat{u}(x,t)$ are reset to $h_{\text{Dirichlet}}(x)$ for all $x \in X^{\delta\Omega}$ before the next prediction of the recursive time stepping scheme. For time-invariant no-flux (Neumann) boundary conditions we found that it is sufficient to enforce $u(x,t) = 0$ in the same manner, while also providing a domain mask G.

3.4 Recursive time stepping

To define the recursive time stepping scheme, we will establish some notations first: Let $u(t,x)$ define the state of a dynamical system at coordinate x and time t, let X be a set of coordinates x defining the grid points of a Cartesian grid and let $X^\Omega = X \cap \Omega$ be a subset of X within the domain Ω. The subset $X^{\delta\Omega}$ denotes the boundary-subset of X defined by $X^{\delta\Omega} = X \setminus X^\Omega$, which thus contains all grid points outside the domain. Furthermore we define a discrete state-representation of the dynamical system at time t as

$$U(t) = \left\{ u(t,x) \,\middle|\, x \in X,\ t \in \mathbb{R} \right\}. \tag{1}$$

where applicable, index notations $u_n(x) = u(t_0 + n\Delta t, x)$ and $U_n = U(t_0 + n\Delta t)$ are used to denote the continuous state and the respective discrete field quantity at the n-th discrete time step. A set of coordinates x in the vicinity of a coordinate p shall be denoted by

$$X^\circ(p) = \left\{ x \in X \,\middle|\, d(x,p) < r \right\} \tag{2}$$

with $d(x,p)$ as the Chebyshev distance of x and p. The extent of the vicinity is defined by r. Consequently, $U_n^\circ(p)$ shall denote the discrete state representation in the vicinity of p at the n-th discrete time step.

Following the ideas of classical explicit numerical time stepping schemes, we propose to setup a flow map model f_θ parameterized by model weights θ that advances a chronological series of m preceding states U_{t-n} ($n \in \{1, 2, \ldots, m\}$) according to

$$\begin{aligned} u_n(x) \approx \hat{u}_n(x) = f_\theta \big(& U^\circ_{n-m}(x), \ldots, U^\circ_{n-1}(x), \\ & C^\circ(x), G^\circ(x) \big) \quad \forall x \in X^\Omega \end{aligned} \tag{3}$$

respecting Dirichlet or no-flux Neumann boundary conditions:

$$\begin{aligned} & u(x,t) = \hat{u}(x,t) = h_{\text{Dirichlet}}(x) \quad \forall x \in, \\ & \frac{\partial u(x,t)}{\partial n} = \frac{\partial \hat{u}(x,t)}{\partial n} = 0 \quad \forall x \in \delta\Omega. \end{aligned} \tag{4}$$

In Eq. (3), $C^\circ(x)$ denotes the (time-invariant) spatial domain parameterizations in the vicinity of x, e.g. inhomogenous material parameters and in Eq. (4) $\delta\Omega$ denotes the domain boundary and n the unit normal at the boundary. Owing to the rectangular shape of input tensors to the ML model f_θ, the binary domain mask $G^\circ(x)$ is supplied as additional input indicating the location of the computational domain Ω in the vicinity of x. As the model can only compute a discrete approximate to the true solution $u_n(x)$, we denote the predicted state at the n-th discrete timestep as $\hat{u}_n(x)$. This leads to the discrete representation of the predicted state $\hat{U}(t,x) = \left\{ \hat{u}(t,x) \,\middle|\, x \in X^\Omega, t \in \mathbb{R} \right\}$, which the ML model is in fact predicting. For brevity x is omitted in the following specifications. In essence, the model f_θ learns a flow map for advancing the available historic field states one time step into the future, without requiring knowledge about the underlying equations governing the dynamical process, but only using data. The predicted state can be fed into the new set of inputs, and thereby form a self-feeding time stepping scheme for long term predictions: starting from m preceding states $\left\{ U^\circ_{-m}, U^\circ_{-(m-1)} \cdots, U^\circ_{-1} \right\}$, a prediction $\hat{U}^\circ_0$ is obtained, which in the next step is injected into the input while dropping the last entry of the previous input:

$$\begin{aligned} \text{step j} = 0: \quad \hat{U}_0 = \Big\{ & f_\theta \Big(U^\circ_{-m}(x), U^\circ_{-(m-1)}(x), \ldots, \\ & U^\circ_{-1}(x), C^\circ(x), G^\circ(x) \Big) \,\Big|\, x \in X \Big\} \\ \text{step j} = 1: \quad \hat{U}_1 = \Big\{ & f_\theta \Big(U^\circ_{-(m-1)}(x), U^\circ_{-(m-2)}(x) \ldots, \\ & U^\circ_{-1}(x), \hat{U}^\circ_0(x), C^\circ(x), G^\circ(x) \Big) \,\Big|\, x \in X \Big\} \\ \text{step j} = 2: \quad \hat{U}_2 = \Big\{ & f_\theta \Big(U^\circ_{-(m-2)}(x), U^\circ_{-(m-3)}(x), \ldots, \\ & \hat{U}^\circ_0(x), \hat{U}^\circ_1(x), C^\circ(x), G^\circ(x) \Big) \,\Big|\, x \in X \Big\} \end{aligned} \tag{5}$$

Note that these equations represent the point-wise prediction scheme that is performed for all x in X at each time step. Consequently, from the m-th step onwards, the model is making predictions solely based on previous predictions. This self-feeding operation of the network represents a trivial recursive time stepper which can be run on new initial conditions[2] to make predictions on future state evolution. The optimal number of previous time steps m used as input is clearly related to the physics at hand. For the two-dimensional wave equation studied in this work, $m = 3$ was found as an optimal value. Theoretically, this is also the minimal value to provide enough information for approximating first and second time derivatives. The schematic in Fig. 1 illustrates this approach on the example of the propagation of waves in a rectangular domain with a rounded edge and a spatially varying distribution of the wave speed parameter. Sections 4.1.4 and 4.2.3 present more details on the rate at which the autonomous operation of the fully trained U^p-Net accumulates prediction errors and diverges from the true solution U_n in finite time.

[2] For the described architecture, m initial states are required. If only a single initial state is available, one may build analogous models for $m = 1$, $m = 2$ etc. which will be called once to generate the initial m samples required for the self-feeding time stepping.

This behavior is inevitable for explicit schemes, nonetheless, our results indicate that the prediction horizon can be long. As the proposed time stepping scheme is rather simplistic, future work may touch on more sophisticated schemes.

4 Results

We study the performance of the proposed U^p-Net for predicting the two-dimensional oscillatory dynamics of the linear wave equation and the asymptotic long-term evolution of processes governed by the heat equation. Particular focus is put on model generalization properties for different scene parameterizations. Two baseline models are chosen from two current state-of-the-art neural architectures to compare against the performance of our U^p-Net model $\mathcal{U}^p$, namely a fully convolutional encoder–decoder model $\mathcal{U}^{\mathrm{ED}}$ and a conventional U-Net model $\mathcal{U}$. See Appendix B for more details on the specific model configurations. For the $\mathcal{U}^{\mathrm{ED}}$ and $\mathcal{U}$ models, all input matrices are concatenated into one tensor, such that field, boundary and domain parameterization information represent different channels in the input. All studies presented in this work were performed on a workstation equipped with a GPU (NVIDIA Titan RTX 24 GB), 32 GB of DDR4-RAM (2666MHz) and an Intel Xeon W-2133 CPU.

4.1 2D wave equation

The two-dimensional wave equation with variable coefficient c

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u = c^2 \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} \right) \tag{6}$$

describes a wide range of phenomena and spatio-temporal dynamics in structures, fluids, electromagnetics, cosmology, geophysics, and other domains [43]. Despite the simplicity of this second-order linear PDE, complex wave fields and pattern formations arise from the time evolution of initial states in bounded domains. Domain boundaries and spatially inhomogeneous wave numbers $c(x)$ give rise to reflection and refraction, respectively, and produce waves interacting across a large range of frequencies and amplitudes.

4.1.1 Data generation

For the generation of training and validation data, a set of prototypical cases is set up using the FEniCS computing platform [44]. These cases cover a wide combinatorial range of initial states, domain shapes, and domain parameterizations. The training data comprise a square domain (128×128) with optional rectangular boundary inclusions (creating outer corners) and wave number inhomogeneities, overall resulting in 100 different training scenes. The validation scenes are set up from rectangular domains (256×128) that feature a complicated combination of training data features and completely new features, such as rounded domain edges. By setting up validation scenes that are qualitatively very different to the training data scenes, we can evaluate the generalization properties of the proposed approach. Figure 2 depicts a schematic of the training and validation scenes. For the initial state of the field quantity, i.e. the elevation $u\,(x, t = 0)\,\forall x \in X^\Omega$, a two-dimensional Gaussian of variable amplitude is placed randomly within the domain. For each scene a direct time integration is performed up to $t = 8\,\mathrm{s}$ using a variational formulation of Eq. (6). The displacement fields U_n are computed with a spatial discretization of $\Delta x = \frac{1}{32}\,\mathrm{m}$ and captured every $\Delta t = 0.01\,\mathrm{s}$ resulting in 800 time steps per scene. Data samples are generated by rolling window processing taking $m = 3$ preceding domain states as input, and the next domain state as output. Using data augmentation (horizontal and vertical flips, amplitude inversion), a total number of 478, 200 training data samples and 3188 validation data samples are generated, see Appendix A for more details on the data generation.

4.1.2 Model setup and training methodology

A U^3-Net prediction model $\mathcal{U}^3$ is set up to consume $m = 3$ wavefield snapshots U, a binary domain mask G, and the spatial distribution of wave numbers C, to output the wavefield at the next time step. $k = 3$ stages are considered, and two variants of the model are generated by choosing different convolutional blocks: $\mathcal{U}^3_{64}$ features a static number of 64 filters per stage, which we found as good tradeoff for model complexity and model accuracy, while $\mathcal{U}^3_{29,\times 2}$ features a cascade of doubling numbers of filters per stage, as displayed in Fig. 1 and starting at 29 filters. The latter number of filters is chosen in a way that the number of trainable parameters n_θ is close to 10^6 for all model architectures in the experiments. Sample-wise normalization of the wavefield inputs is mapping the zero-symmetric range of maximum wave amplitudes to $[0, 1]$. Using a batch size of 128 and a sample-wise normalized domain-adaptive mean-squared error (nMSE, $x \in X^\Omega$) loss function, all models are trained using the `adam` optimizer for 300 epochs with a learning rate decaying from 10^{-4} to 10^{-5}, followed by another 50 epochs with a learning rate decay from 10^{-5} to 10^{-7}. We found this training strategy beneficial for the long-term prediction capabilities of the models, see Appendix C for details.

4.1.3 Single-step predictions

First, the training convergence, and the performance of the flow map models are studied by assessing the evaluation of a single prediction, i.e. mapping preceding domain snap-

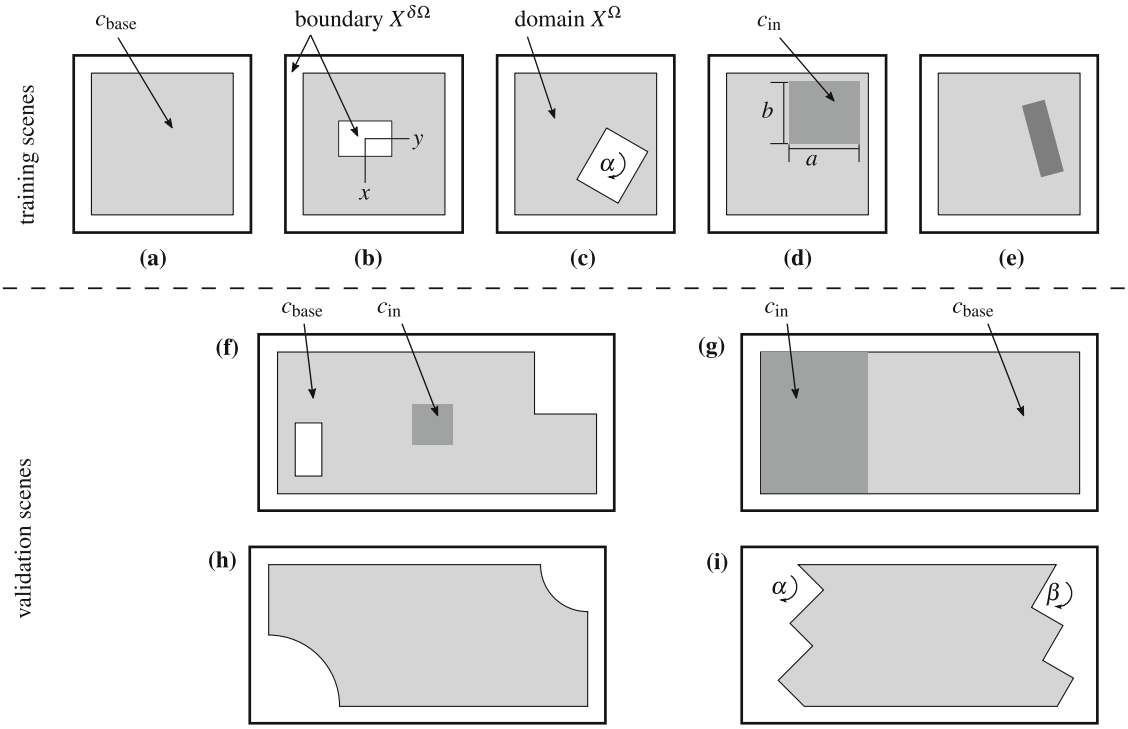

Fig. 2 Prototypical cases set up for generating the training data (top row) and validation data (bottom row) for the wave equation studies: homogeneous domain (**a**), rectangular inclusions (**b**, **c**) and domain inhomogeneities (**d**, **e**). Two variants per case are generated by randomly selecting the variation parameters x, y, a, b (for cases **b**–**e**) and α (for cases **c**, **e**). Validation cases (**f**) and (**g**) represent a combination of the training cases, while cases (**h**) and (**i**) are completely different from all training scenes

shots one time step into the future. In the following, this is denoted as single-step prediction. Figure 3 displays the training process for six models, namely the ED and U-Net baseline models and the new U^p-Net models, each in a variant with a doubling number of filters and a variant with a constant number of filters at each stage. Note that all models feature the (approximately) same amount of trainable parameters (see Appendix B), hence their overall learning capacity can be estimated to be of the same order in magnitude.

The loss evolution of all six models shows similar characteristics, but also clear differences in the overall training performance. Starting from values of magnitude nMSE $= 10^{-3}$ after the first epoch, the training loss reduces to values of approximately 2.9×10^{-7} ($\mathcal{U}^{\text{ED}}_{106}$ model), 2.2×10^{-8} ($\mathcal{U}_{96}$ model), and 1.1×10^{-8} ($\mathcal{U}^3_{64}$ model). The different filter configurations slightly effect the training and validation loss for U-Net and U^3-Net models with a clear advantage for the double-cascading filters. None of the models exhibits overfitting. The two-step learning rate decay strategy helps the U-Net and U^3-Net models particularly towards the final 100 training epochs.

The models based on the U-Net and the proposed U^p-Net architecture exhibit the best generalization capabilities to different scenes as displayed in Table 1, where the loss averaged over all four validation cases is reported in terms of different metrics. The sample-wise normalized nMSE loss as shown in Fig. 3 is accompanied by non-normalized MSE and MAE metrics. Further, the mean error of means (MEM) is reported

$$\text{MEM} = \left| \frac{\sum_{i=1}^{n} u(x_i) - \sum_{i=1}^{n} \hat{u}(x_i)}{n} \right| \tag{7}$$

that is quantifying an incorrect offset of the whole predicted field towards larger or smaller values. Analogous to the nMSE, we define the MEM as a domain-adaptive metric, where n denotes the number of field values within the domain. The U^p-Net outperforms the classical U-Net by around one order of magnitude. Therefore, we conclude that our U^p-Net exhibits superior performance over the U-Net architecture for the given problem based on the convergence behavior and the validation loss value after the same number of training epochs. The performance values reported in Table 1 were qualitatively confirmed in independent training runs.

While the overall validation loss describes a global and heavily averaged metric for the large validation data sets, Fig. 4 displays a more detailed view into the precision of the flow mapping models for each of the four validation cases. Inputs from different time points of the wave propagation are

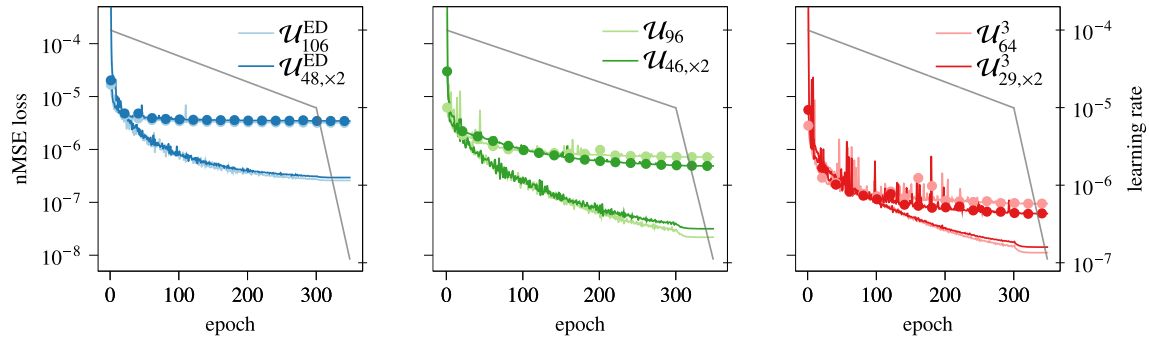

Fig. 3 Evolution of nMSE training loss (solid line) and validation loss (dot markers) of the baseline encoder–decoder models $\mathcal{U}^{\mathrm{ED}}$ (left column), the baseline U-Net models $\mathcal{U}$ (center column), and the proposed U^3-Net models $\mathcal{U}^3$ in (right column) trained on the wave equation dataset described in Sect. 4.1. (×2) indicates models with doubling cascades in the number of filters per stage. A two-stage exponential learning rate decay training strategy is applied (gray line)

Table 1 Validation error metrics of single-step predictions for baseline and proposed model configurations for training runs reported in Fig. 3 at epoch 350 for the wave equation case study

Model	Architecture	Loss (nMSE)	MSE	MAE	MEM
$\mathcal{U}^3_{64}$	U^p-Net	9.37×10^{-8}	1.44×10^{-9}	5.97×10^{-6}	5.95×10^{-7}
$\mathcal{U}^3_{29,\times 2}$	U^p-Net	6.12×10^{-8}	1.27×10^{-9}	6.28×10^{-6}	7.15×10^{-7}
$\mathcal{U}_{96}$	U-Net	7.15×10^{-7}	8.54×10^{-9}	9.28×10^{-6}	1.32×10^{-6}
$\mathcal{U}_{46,\times 2}$	U-Net	4.85×10^{-7}	7.69×10^{-9}	1.12×10^{-5}	1.72×10^{-6}
$\mathcal{U}^{\mathrm{ED}}_{106}$	ED	3.16×10^{-6}	3.91×10^{-8}	3.09×10^{-5}	4.45×10^{-6}
$\mathcal{U}^{\mathrm{ED}}_{48,\times 2}$	ED	3.43×10^{-6}	4.41×10^{-8}	3.44×10^{-5}	5.92×10^{-6}

fed to the deep learning models, and their error in predicting the next time step of the dynamic evolution is assessed. One can observe that the single-step prediction accuracy varies among the examined network architectures, and in general decreases for predictions at later time instances. The latter mainly results from the sample-wise normalization and a respective loss function operating on normalized data sets. Rather constant prediction errors are maintained, when evaluated with respect to the maximum amplitudes of the wave fields. Fluctuations in the single-step error which do not correlate with the maximum amplitude of the wave field can be the caused by a variety of phenomena (reflections, refractions, superposition of waves) that increase the complexity of the prediction task at specific time instances and thus result in slightly larger errors.

All U-Net-inspired models demonstrate good single-step prediction capabilities when evaluated on the four validation datasets. In direct comparison, the ED architectures show a significantly weaker predictive ability. Overall, the U^3-net models $\mathcal{U}^3_{64}$ and $\mathcal{U}^3_{29\times 2}$ outperform the baseline models at all times steps and validation cases. Out of the two U^3-Net models, the $\mathcal{U}^3_{64}$ with a constant number of filters exhibits slightly more accurate single-step predictions for validation cases (f) and (g) as visible from the boxplots in Fig. 4. Here, the amplitude normalization highlights the advantages in prediction quality when using the proposed U^p-Net instead of the baseline models.

4.1.4 Long-term predictions

To assess the predictive capabilities of the models we now examine multiple long-term prediction scenarios, i.e. by injecting the trained flow map model into the recursive time stepping scheme described in Sect. 3.4. Figure 5 depicts the ground truth wave propagation along with the prediction of the $\mathcal{U}^3_{64}$-Net based time stepper for validation case (f). The domain comprises a rectangular shape with an inclusion and an inhomogeneous spatial distribution of the wave speed parameter, see Fig. 2. Reflections at the domain boundaries occur after time step 50, and refraction at the material inhomogeneity is visible at time step 75 in the central domain region.

The spatial error distribution is computed as the absolute Euclidean distance between ground truth and prediction, shown in the third panel of Fig. 5. After 100 integration steps the mean absolute error (MAE) amounts to 1.33×10^{-3}, which is an error of about 2.5% in relation to the maximum amplitude of the wave field (5.36×10^{-2}). After 200 time steps the MAE is 3.79×10^{-3} which is a relative error of 6.4%. For the given time instances, the largest local error is observed in the narrow field below the left domain inclu-

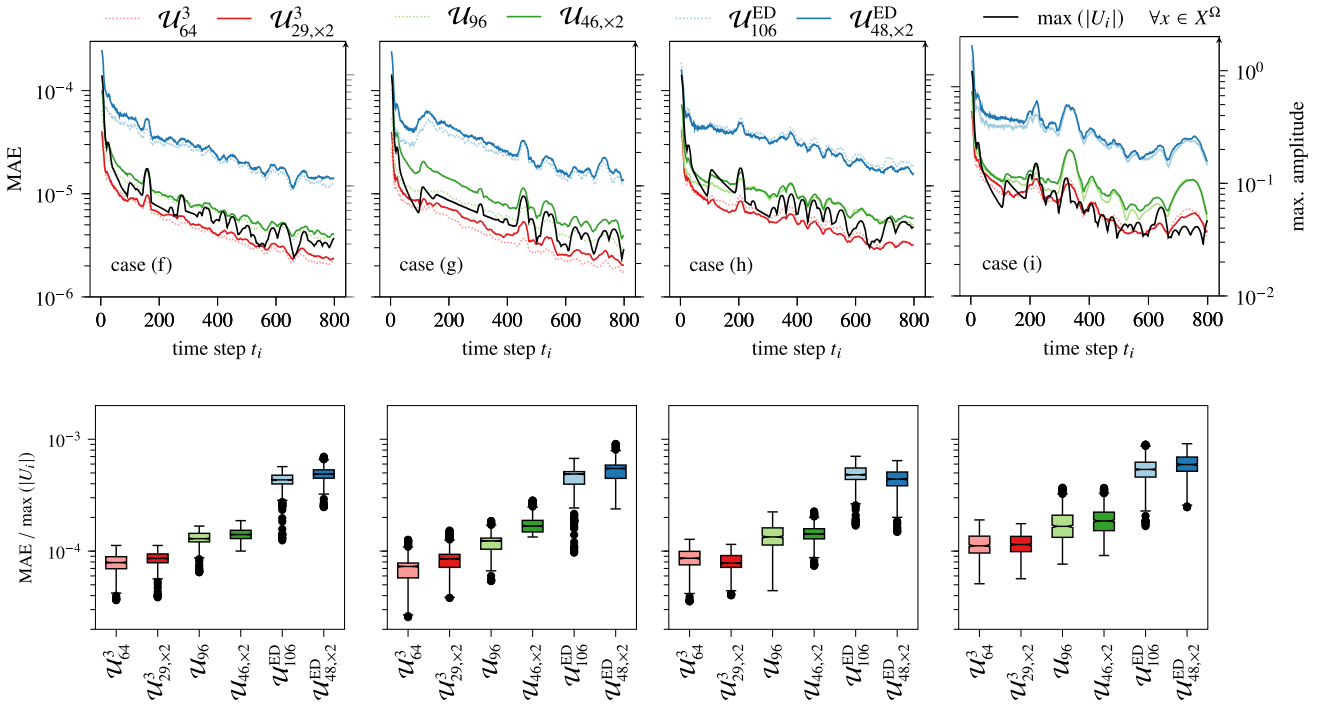

Fig. 4 The top row shows the single step prediction error of baseline and proposed models over all time steps of the validation cases (f, g, h, i) along with the evolution of the maximum wavefield amplitude over time. Note that the scale of the vertical axis are different for both quantities. The bottom row shows analogous box-plots of the single step prediction error (MAE) scaled by the maximum amplitude at each time instance

sion. After 100 prediction steps a maximum local error of 7.21×10^{-3} is obtained here, increasing to 1.9×10^{-2} after 200 steps. The point-wise error plots (clipped at 100%) in the bottom panel helps to identify local errors highlighted by the relative measure. Caution must be taken when interpreting the rather large error values displayed there: As the scene is initialized with zero amplitudes in the domain (aside from the Gaussian), very high values appear in regions where the ground truth amplitude is zero: a small prediction imperfection, even numerical rounding errors, will naturally result in extreme relative errors. As the waves propagate into the domain, large relative errors are only visible where wave amplitudes are close to zero, again causing a division by nearly zero. In most of the domain with substantial surface amplitudes low relative error values are visible. Particularly, the domain inhomogeneity is not visible in the display of the relative error, indicating an accurate prediction across the material boundary. Excluding the computational singularity in regimes of zero amplitude, the relative error analysis indicates that predictions are accurate throughout the domain up to step 100. Overall, no particularly large errors are observable in proximity of the material inhomogeneity in the center of the domain, which is a major advantage over state-of-the-art approaches. The model is capable of respecting the abrupt change of wave numbers in this regime, which is a challenging task causing refractions on the boundary of the inhomogeneity, and different wave propagation speeds inside the inhomogeneity, see time step 75. In a visual comparison it is hard to distinguish the ground truth and the prediction up to prediction step 100. After 150 and 200 steps, a slight global shift towards positive values can be observed in some regions of the predicted wave fields along with some loss of detail. However, the overall wave field topology of the ground truth is still clearly recognizable, even for long-term predictions of up to 200 steps.

A detailed investigation of the long-term prediction behavior is shown in Fig. 6. Here, the error evolution of the recursive time stepping scheme is examined using the data of validation case (f) featuring 800 time steps. To compare the predictive capabilities of the proposed U^p-Net architecture to those of the baseline models, long-term predictions are initialized at seven different time instances of the validation case using all fully trained models described in Sect. 4.1.3. Essentially, seven different cases with different initial conditions are run, to examine the impact of the different initial physics on the long-term prediction capabilities. Wave fields at early time steps comprise larger amplitudes but simpler topology, while the wave fields towards larger time steps get more complicated with many interacting traveling waves. We will solely focus on the U^3-Net predictions first. Naturally,

Fig. 5 Evolutions and predictions of wave field dynamics in a complex and heterogeneous domain using the $\mathcal{U}^3_{64}$ model. The material inhomogeneity is indicated by the hatched patch in upper left panel. The first prediction is initialized from wave fields at time steps $t = 1.47, 1.48, 1.49$ s from validation case (f). From there on, recursive time stepping (Sect. 3.4) is performed for a total of 200 time steps. The first row shows the ground truth U obtained from numerical simulations, the second row shows the predicted wave fields $\hat{U}$, the third row the absolute difference $|\hat{U} - U|$, and the last row the relative point-wise error at selected time steps. White areas indicate the domain boundary regions

the error accumulates by the number of time steps predicted. However, the analysis shows that the errors grows especially fast for simulations initialized at early time instances, such as time step 50 or 150. For example, the error grows to MAE $\approx 3 \times 10^{-3}$ within 100 time steps for the long-term predictions performed using the model $\mathcal{U}^3_{64}$. If the time stepping is initialized when the wave field is fully developed, the error accumulation is considerably slower: 100 time steps result in an MAE of 8.5×10^{-4}, 1.2×10^{-3}, 7.7×10^{-4} for simulations started from time steps 450, 550, 650. Again, in most application scenarios, this behavior is desirable and hypothesized to be linked to the normalized loss evaluation as outlined in Sect. 4.1.3.

Compared to the ED models, the U-Net-type models show significantly better long-term prediction capabilities in Fig. 6. With few exceptions, the ED models exhibit the greatest

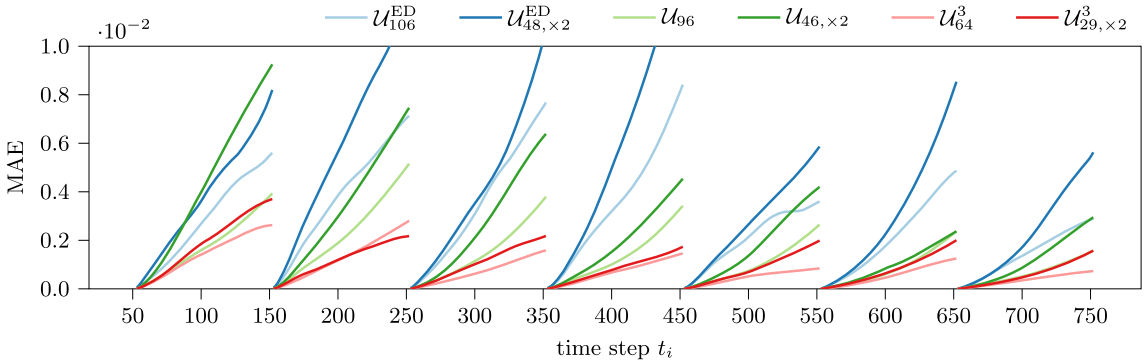

Fig. 6 Comparison of long-term predictions using the recursive time stepping scheme initialized at time steps 50, 150, 250, 350, 450, 550 and 650 of validation case (f) and evaluated 100 steps for two configurations of the ED, the classical U-Net architecture, and two model configurations of the proposed U^p-Net architecture. For scaling reasons, the displayed error evolution is limited to an MAE of 2×10^{-2}

errors of the examined network architectures. The long-term predictions exhibit four to ten times greater MAE error values compared to the respective best performing U^p-Net model. When examining the classical U-Net-type models $\mathcal{U}_{96}$ and $\mathcal{U}_{46\times 2}$, a considerable difference in the precision of long-term predictions can be observed. Regarding the scene and dynamics contained in validation case (f), the model configured with a constant number of filters ($\mathcal{U}_{96}$) shows clear advantages when compared to the model with stage-wise doubling filters ($\mathcal{U}_{46\times 2}$). A similar, although less pronounced, trend is also visible for the proposed U^p-Net models. Most importantly, both U^p-Net models, namely the $\mathcal{U}^3_{64}$ and the $\mathcal{U}^3_{29,\times 2}$ model, exhibit the smallest prediction errors of the investigated models at most time instances. Notably, the model $\mathcal{U}^3_{64}$ slightly outperforms its counterpart $\mathcal{U}^3_{29,\times 2}$. Considering the errors after 100 steps of prediction, the U^p-Net architectures yield factors between 1.9 (time step 650) and 3.14 (time step 550) performance increase compared against the best-performing U-Net model, and factors between 2.15 (time step 150) and 5.8 (time step 450) performance increase compared against the best-performing ED model.

Overall, the proposed fusion of domain information and spatial parameter distributions allows for accurate short- and long-term predictions of the investigated wave dynamics. Especially, U-Net architectures are found to be suitable for generic predictions of spatio-temporal dynamics. The proposed U^p-Net can be considered the most promising candidate among the investigated architectures. This finding is evident from the error metrics related to the previous analysis of long time-time predictions for all validation cases, as summarized in Table 2. For all 28 prediction cases, the proposed $\mathcal{U}^3_{64}$ performs the best in terms of both MAE and MSE error metric. While classical U-Net models show significantly poorer performance, some ED models even exhibit instability, i.e. exponentially growing errors.

Assessing the long-term term predictions, we found an indication for a linkage of the prediction accuracy and the single-step shift-type error MEM. The MEM measures a global offset of the mean wavefield predictions from the ground truth. This shift of the complete field is observed to out-weight the general single-step accuracy for the long-term predictions in some cases. That is, models of low MEM error tend to perform better in long term than models with a low single-step error. As outlined in Appendix C, all examined models show a fluctuating behavior of the MEM throughout the training process, i.e. the predicted field quantities experience some accumulating offset error of the mean field amplitude. By decreasing the learning rate we were able to reduce these fluctuations and thus enhance the long-term accuracy of the final models. We therefore presume that further optimizations with regard to the MEM (e.g. the incorporation of the MEM in the loss function) can lead to an even better performance of the U^p-Net architecture in terms of long-term predictions. Additional investigations of the long-term prediction behavior performed on the validation datasets (g), (h) and (i) can be found in the Appendix D.3, further supporting the findings discussed before.

4.1.5 Out-of-distribution generalization properties

So far, the generalization to different scenes are investigated on the example of the four validation cases, that significantly differ from all training examples in terms of geometry, scene composition and domain aspect ratio. The final investigation on the generalization properties is concerned with out-of-distribution physical parameters, i.e. with cases that involve wave number parameterizations that are not featured in the training datasets. The training data was generated using wave numbers from a discrete set $c_{\mathrm{base,train}} \in [0.5, 0.75, 1.0, 1.25, 1.5]$. We now study case (f) for wave numbers c_{base} picked in between the training values and

Table 2 Comparison of U^3-Net, U-Net and encoder–decoder models in terms of mean squared error (MSE) and mean absolute error (MAE) computed for predictions after 100 time steps

Model	MSE_{100}			MAE_{100}		
	Mean ± stdev	Min	Max	Mean ± stdev	Min	Max
$\mathcal{U}^3_{64}$	$9.38 \pm 9.77 \times 10^{-6}$	1.02×10^{-6}	3.99×10^{-5}	$2.08 \pm 1.12 \times 10^{-3}$	7.34×10^{-4}	4.97×10^{-3}
$\mathcal{U}^3_{29,\times 2}$	$2.01 \pm 1.93 \times 10^{-5}$	2.93×10^{-6}	7.07×10^{-5}	$2.93 \pm 1.34 \times 10^{-3}$	1.15×10^{-3}	6.22×10^{-3}
$\mathcal{U}_{96}$	$2.58 \pm 2.04 \times 10^{-5}$	3.78×10^{-6}	8.11×10^{-5}	$3.90 \pm 1.48 \times 10^{-3}$	1.57×10^{-3}	6.91×10^{-3}
$\mathcal{U}_{46,\times 2}$	$1.82 \pm 3.96 \times 10^{-2}$	6.40×10^{-6}	1.82×10^{-1}	$3.06 \pm 4.90 \times 10^{-2}$	1.52×10^{-3}	1.84×10^{-1}
$\mathcal{U}^{ED}_{106}$	$0.98 \pm 1.45 \times 10^{-3}$	1.46×10^{-5}	7.47×10^{-3}	$1.54 \pm 0.91 \times 10^{-2}$	2.89×10^{-3}	3.69×10^{-2}
$\mathcal{U}^{ED}_{48,\times 2}$	$2.51 \pm 0.13 \times 10^{+3}$	4.49×10^{-5}	$7.02 \times 10^{+4}$	$0.20 \pm 1.0 \times 10^{+1}$	5.63×10^{-3}	$5.42 \times 10^{+1}$

The self-feeding long-term predictions are performed for all four validation cases (f, g, h, i) and initialized at $t \in \{0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5\}$ s. The table shows mean and standard deviations, as well as the smallest and largest errors resulting from the 28 predictions per model

outside the maximum range of that set, i.e. for $c_{\text{base}} = [0.2, 0.25, \ldots, 1.75, 1.8]$. The data generation is performed according to Sect. 4.1.1 and again the wave fields are initialized with Gaussians. The model $\mathcal{U}^3_{64}$, which performed best in the previous sections, is applied to the new 32 cases by performing long-term predictions at seven different time-instances in analogy to Sect. 4.1.4. Note again, that no model-retraining was performed. Figure 7 shows the average MAE of all long-term predictions as a function of the domain parameterization c. It is apparent that the error is small at the wave number configurations which were used for generating the training data. However, the MAE is only slightly larger for wave numbers placed in between the values of $c_{\text{base,train}}$. Consequently, we have been able to observe good generalization capabilities in terms of interpolation for parameterizations within the training data distributions. Furthermore, the model also exhibits some capabilities to extrapolate outside the limits of $c_{\text{base,train}}$. Although the model has never seen any wave dynamics outside $c = 0.5$ and $c = 1.5$, predictions outside this interval are valid and of proper accuracy, at least for short-term and mid-term time stepping intervals. The error increases significantly for wave numbers that depart far from the training set values. The threshold for accepting a prediction may depend on the actual case and personal preferences, however we found that the predictions for edge cases $c_{\text{base}} = 0.5$ and $c_{\text{base}} = 1.8$ are in good agreement with the ground truth when compared visually for the main features of the wave field. We should also note that the wave number enters the physics in quadratic form, which explains the seemingly rapid performance degradation for values larger than $c = 1.5$. Interestingly, the error increases linearly with prediction time, as it visible from the quasi-parallel evolution of the error values reported for different numbers of prediction steps in Fig. 7.

4.2 Heat equation

The two-dimensional heat equation with variable coefficient reads

$$\frac{\partial u}{\partial t} = \nabla \cdot (c\, \nabla u) = \frac{\partial}{\partial x_1}\left(c \frac{\partial u}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(c \frac{\partial u}{\partial x_2}\right) \tag{8}$$

where $c(x) > 0$ denotes the thermal conductivity of the domain. The steady-state solution $u\,(x, t \to \infty)$ to an initial condition $u\,(x, 0)$ satisfies $\frac{\partial u}{\partial t} = 0$. We study the time evolution for Dirichlet boundary problems. Note that the constant temperatures are defined at the boundary of the domain.

4.2.1 Data generation

The generation of training and validation data for the heat equation problem is performed in analogy to Sect. 4.1.1: training data scenes comprise a square domain (128×128, $\Delta x = \frac{1}{16}$ m) featuring optional boundary inclusions and thermal conductivity inhomogeneities ($c \in \{0.5, 0.75, 1, 1.25, 1.5\}$). In addition to initial disturbances, also variations of the boundary values are considered by generating datasets with constant boundary values of variable magnitude as well as datasets with boundary values having horizontal or diagonal linear gradients. Boundary values and initial values varied between normalized values of zero and one, see also Fig. 11. This setting effectively renders the scene a singularity-like problem at the domain boundaries, where temperatures jump in the initial condition. As the temperature distributions rapidly approach a steady state, time integrations are performed up to $t = 0.6$ s, i.e. 120 steps using $\Delta t = 0.005$ s. Using data augmentation (rotation by 90°, 180° and 270°) and rolling window processing, 275, 184 data samples are generated from 588 training scenes. Four rectangular (256×128) validation datasets are generated accordingly, featuring rectangular, circular and H-shaped inclusions, conductivity inhomogeneities as well as linear temperature gradients on the boundary, see also Fig. 11. As

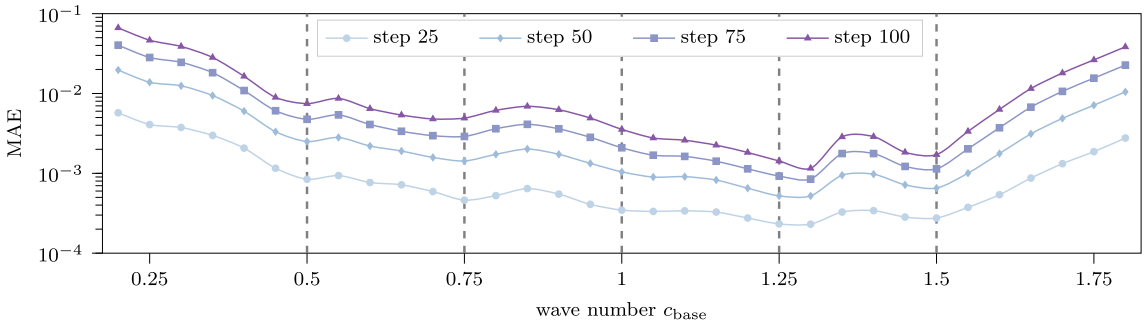

Fig. 7 Out-of-distribution study of model $\mathcal{U}_{64}^3$ on variants of validation dataset (f) with $c_{\text{base}} = c_{\text{in}}$ and $c_{\text{base}} \in [0.2, 0.25, \ldots, 1.75, 1.8]$. The MAE is computed as average over seven long-term predictions starting at different time-instants (see Fig. 6) and evaluated after 25, 50, 75 and 100 prediction steps. The vertical dashed lines indicate waves numbers c_{base} used for model training

these characteristics, if at all, are only differently featured in the training data, long-term predicting the thermal energy flow requires generally valid models that have learned a diffusion process.

4.2.2 Model setup

To predict the evolution of the temperature fields under set boundary values, a U^3-Net is utilized. The inputs of the three distinct encoder paths are (i) three preceding temperature fields separated by Δt, (ii) a binary domain mask, and (iii) a spatial distribution of the thermal conductivity parameter $c(x)$. The temperature fields (i) also include information on the Dirichlet boundary conditions by replacing the temperature field with extrapolated boundary temperatures at all $x \in X^{\delta\Omega}$. As temperatures of training and validation datasets are limited to the range [0, 1], training is performed without additional normalization. A domain adaptive MSE loss function is used, and optimizer, learning rate (decay) and number of training epochs are chosen according to Sect. 4.1.2. Following the results for the wave equation, we focus on models with constant filter sizes per model stage for the heat equation studies. For a fair comparison, the filter sizes of the U^3-Net and those of the two baseline models, the classical U-Net, and the ED architecture, are chosen such that the number of trainable parameters is approximately the same, see Table 3.

4.2.3 Predictions

A comparison of the single-step performance of the three models is shown in Table 3. The table provides the average of the $4 \times 117 = 468$ prediction errors available from the validation datasets. Out of the three models, the U^p-Net type model $\mathcal{U}_{64}^3$ exhibits the most precise single-step step predictions. In terms of the average single-step prediction error, the U^p-Net outperforms the classical U-Net by a factor of 1.8 and the ED architecture by a factor of 40. Also for the mean field values, i.e. the MEM error metric, the proposed U^p-Net outperforms the baseline models by two orders of magnitude. As all models shown here feature the same amount of trainable parameters, we conclude that the proposed U^p-Net architecture is superior to conventional neural architectures.

A closer view into the single-step error throughout the transient evolution of the temperature field is given in Fig. 8a. Here, a validation case with two rectangular inclusions and a centered rectangular inhomogeneity is investigated, see Fig. 9. In analogy to our findings from Sect. 4.1.3 the single-step predictions exhibit larger errors in the first time steps, which here coincide with the most transient phase of the temperature evolution. In terms of the single-step prediction the U^p-Net clearly outperforms the baseline architectures at all time instances. This observation holds also for the remaining validation cases. For the latter, similar characteristics are discernible (data not shown explicitly, but included in the average metrics displayed in Table 3).

To assess the long-term prediction capabilities of the proposed architecture, the evolution of the transient heat conduction is examined using the recursive time stepping scheme with models $\mathcal{U}_{64}^3$, $\mathcal{U}_{96}$ and $\mathcal{U}_{106}^{\text{ED}}$. Figure 9 displays the temperature distribution of the ground truth U, the prediction $\hat{U}$ and the spatial distribution of the $\mathcal{U}_{64}^3$ prediction error at several time steps for a validation case with rectangular domain inclusions and a conductivity inhomogeneity in the center of the domain. Notably, large gradients are evident in the thin domain regions between the inclusions and the outer domain boundaries, see upper left region of the domain in Fig. 9. Figure 8 (b) displays the respective domain-MAE errors for all three models for the same long-term prediction. Considering model $\mathcal{U}_{64}^3$, the prediction error grows until time step 20 and particularly in the regime of the inhomogeneity with abrupt changes in thermal conductivity. From time step 20 onwards, the prediction error keeps decreasing. After 100 time steps, a steady-state heat distribution is achieved. As the error does not show an accumulation behavior, the results indicate that

Table 3 Error metrics MSE, MAE and MEM of single-step predictions for baseline models and proposed model configurations evaluated within domain Ω for four heat equation validation datasets

Model	Architecture	$f_{0/1/2}$	Parameters	MSE	MAE	MEM
$\mathcal{U}^3_{64}$	U^p-Net	64	991,617	3.29×10^{-7}	2.46×10^{-4}	4.29×10^{-5}
$\mathcal{U}_{96}$	U-Net	96	991,681	3.82×10^{-5}	4.41×10^{-4}	2.18×10^{-3}
$\mathcal{U}^{\mathrm{ED}}_{106}$	ED	106	1,006,153	1.45×10^{-4}	9.89×10^{-3}	9.14×10^{-3}

f_0, f_1 and f_2 denote the number of filters at the three encoding/decoding stages of the architectures

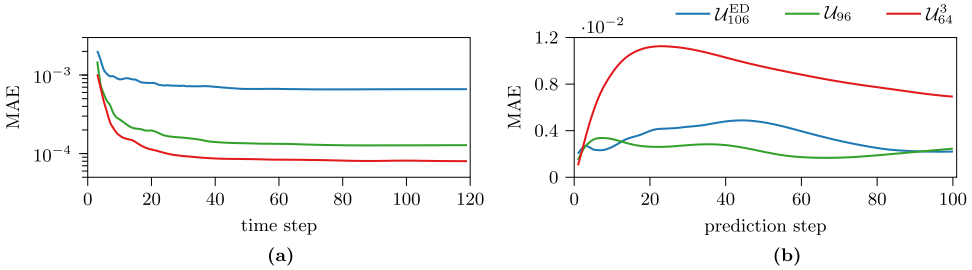

Fig. 8 Evaluation of the fully trained ED, U-Net and U^3-Net models for the heat equation case. **a** Displays the single-step prediction error, i.e. the domain adaptive MAE, for 117 time steps. **b** Shows the error evolution for a long-term prediction over 100 time steps. Results correspond to the validation case shown in Fig. 9

the $\mathcal{U}^3$ model did in fact learn a diffusion process, and hence compensates for errors made in the first prediction steps. Very similar behavior was observed in all validation cases studied in this work.

Surprisingly, the superior single-step performance of the U^p-Net model, see Table 3 and Fig. 8a, does not translate to improved long-term predictions when compared to the baseline models, see Fig. 8b. Although all models were able to generate decent long-term predictions, here the fully trained ED and U-Net models outperform the U^3-Net model in most cases. In the wave field studies, we were able to track such behavior back to the MEM single-step error, which identifies models that are prone to introducing a global field offset in long-term predictions. For the heat equation studies however, the discrepancy between single-step and long-term prediction quality can not be linked to the MEM metric, as the U^3-Net model exhibits the best results in this regard, see Table 3. However, as the models were only trained to perform single-step predictions, the discrepancy of single-step and long-term accuracy is not entirely inexplicable, as some local types of single-step prediction errors might amplify when the recursive time stepping scheme is applied. We assume that a direct integration of the recursive time stepping scheme in the training processes, i.e. the loss function, would address this problem and by design eliminate the discrepancy of training performance and long-term prediction capabilities.

4.3 Computational efforts

As computational efforts are a major factor for assessing the applicability of a numerical method, an overview is given on the efforts related to the data generation, network training and model inference stages of the presented approaches. Here, we limit ourselves to a consideration of the wave equation problem. As the model complexity for the heat equation is very similar, the timing results for the deep learning models can be transferred directly, while heat equation training times are available from Table 4. All computations reported in this work were performed on a desktop workstation equipped with a GPU (NVIDIA Titan RTX 24 GB), 32 GB of DDR4-RAM (2666MHz) and an Intel Xeon W-2133 CPU. The software, accessible at https://github.com/TUHH-DYN/DeepStep, is written in Python using the deep learning framework TensorFlow. The timings reported hereafter are subject to hardware and software configurations, hence they shall only serve for illustrative purposes.

Training and validation datasets are generated using the finite element method (FEM) provided by the open-source library FEniCS, see Sect. 4.1.1. The data generation process consists of two major parts: the FEM computations and a post-processing step to transform and save the data aligned to Cartesian grids.[3] Using our hardware, generating a 800-step training time series of 128×128 resolution required approximately 117 s. Particularly, the FEM solution efforts amount to approximately 44 s and the post-processing efforts amount to approximately 73 s on average across all scenes. The generation of a 800-step validation time series (256×128) takes approximately 654 s, of which the FEM solution consumes 248 s. The computing times vary slightly depending on the specific configuration of the scene and the resulting com-

[3] Interpolation between the computational mesh and the Cartesian grid was performed using the Python package scipy griddata, which is performing a linear barycentric interpolation for every time step.

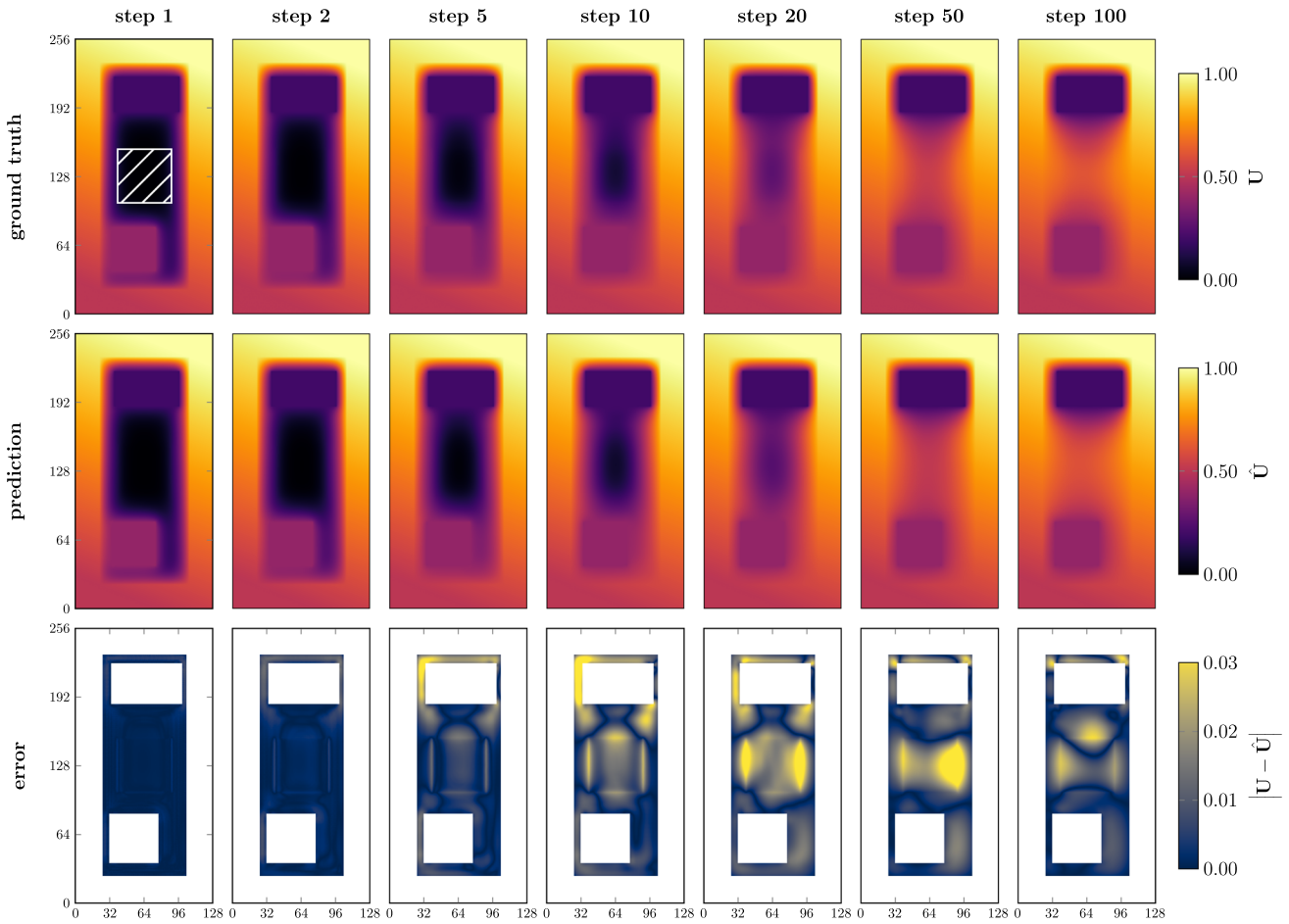

Fig. 9 Long-term predictions of heat conduction: 100 prediction steps are shown for a validation case with two rectangular inclusions and a centered rectangular inhomogeneity (hatched patch in upper-left panel). The first prediction is initialized from temperature fields at time steps $t = 0, 0.005, 0.01$. From there on, the recursive time stepping scheme is applied using the proposed the $\mathcal{U}^3{}_{64}$ model. The first row shows the ground truth obtained from numerical simulations, the second row shows the predicted fields, and the last row shows the absolute difference

plexity of the physics. The generation of the 100 training and 4 validation sets thus required approximately 239 min of pure computing. On the deep learning end, the training part is the most time-consuming part. Table 4 in Appendix B lists all training times for all models shown in this work. Considering the best-performing $\mathcal{U}^3_{64}$ network, training took 175 h. Data generation and model training hence consumed a total of 179 h of computation, and the data sets consume 12 and 11.5 GB for the wave and heat equation, respectively. A reduction of training time is in sight when using the latest data processing pipelines shipped with all major deep learning frameworks. To reduce the training efforts, additional studies will be carried out to find the minimal data set size required for achieving comparable prediction accuracy. Furthermore, more extensive neural architecture and hyperparameter search may result in models of less complexity that can be trained faster. An extension to three-dimensional cases is straight-forward from the architectural perspective. Hardware requirements will however become higher, as the size of the training batch samples will require for larger GPU storage.

Deep learning model inference times greatly depend on the GPU hardware, and on the number of concurrent predictions, i.e. the maximum batch size. The hardware used in this work allowed for a batch size of 1024 samples (256×128), hence we were able to time-step the dynamics of 1024 different scenes simultaneously. A single prediction step took on average 2.51995 s per batch, the time stepping of 1024 scenes for 800 steps thus amounts to 33.6 min. Consequently, the time integration of a single scene for 800 steps took 1.97 s, while the FEM solution without post-processing took 248 s. In contrast to physical simulations, the inference time of neural networks does not depend on the shape of the domain or the complexity of the physics. To compute more realistic scenes of larger size, parallel predictions (i.e. in the same batch) can be carried out on segments of the scene and then

need to be re-combined into one global scene. The computational efforts increase linearly with the domain size.

We would like to avoid stressing the role of computational benefits, as these clearly depend on hardware, use-case, and multiple other effects. To reduce the offline costs related with setting up the proposed U^p-Net architecture, we make the source code freely available for the interested reader and user.

5 Conclusion

The method proposed in this work sets out to overcome some limitations of existing approaches to time stepping of parameterized spatio-temporal dynamics using purely data-driven techniques. Particularly, the U^p-Net architecture is designed for fusing qualitatively different field variables, thereby allowing for large generalization potential and deep representation learning. Four distinct advantages of the proposed neural architecture were identified in this work.

Firstly, our method does not require any a priori knowledge about a mathematical description of the underlying first principles, and hence is a generic and data-centric approach. Then, the method is designed for long-term predictions through iterative self-feeding. Furthermore, domain parameterizations, such as material inhomogeneities, are considered explicitly by the proposed method, which also enforces (potentially time-varying) boundary conditions. Last but not least, the method is making use of the adaptivity property of fully convolutional networks: training can be performed on arbitrarily shaped domains of arbitrary spatial resolution (e.g. 128×128). Network re-training is not required for inference on differently shaped geometries at other resolutions (e.g. 512×1024) or at other process parameterizations. Hence, a once-trained network can be used to compute the time evolution of many different scenes (parameterization, size, shape) given that the underlying physics are the same. In future studies, a priori physical knowledge can be introduced through specialized convolutional kernels that represent temporal and spatial derivatives.

On the example of the two-dimensional universal wave and heat equations, our studies indicate superior prediction quality of the proposed U^p-Net architecture over established encoder–decoder and U-Net architectures of same model complexity, confirming the results of [29]. Exceptionally large data sets of numerical simulations are used for excessive model training. The U^p-Net's superior generalization property to significantly different scenes in terms of size, shape and boundary conditions is shown. Long-term time stepping experiments indicates how purely data-driven flow map models can make accurate predictions of time-dependent physics. In contrast to classical numerical schemes, inference time is not a function of the complexity of the observed physics for the data-driven model. The offline costs for model training can thus be compensated by very low online costs during model inference. The novel U^p-Net methodology proposed can be regarded as a starting point for the investigation of more complex wave, diffusion as well as flow processes governed by nonlinearity and turbulence [45,46]. It is therefore a scheme which complements the recent progress in the development of data-driven techniques in fluid dynamics and physical oceanography [47,48].

Note on videos and code

Additional videos of the model prediction are available, see https://github.com/TUHH-DYN/DeepStep. The source code for replicating the data generation, model training and model validation is accessible through https://github.com/TUHH-DYN/DeepStep.

Acknowledgements J. Ohlsen was supported by the Hamburg University of Technology I^3 initiative (funding ID T-LP-E01- WTM-1801-02).

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix A: Data generation

Appendix A.1: Simulation environment

Training and validation datasets are generated using the open source computing platform FEniCS [44] for numerical simulations. At first, the domain geometry is defined by a number of primitive shapes which are then combined using the set operations `union`, `intersection`, and `difference`. The domain is then discretized using a tri-mesh, while also respecting additional sub domains, utilized for defining spatial parameterization. Triangular elements with linear basis functions are employed (called *P1* elements in FEniCS). Finite element computations are performed using the build-in `GMRES` solver and incomplete LU factorization (`ilu`) for preconditioning. The implicit Euler method is used for time integration, thereby the differential equation is solved at all mesh nodes. Finally, linear interpolation is used to generate datasets defined on Cartesian grids. One crucial

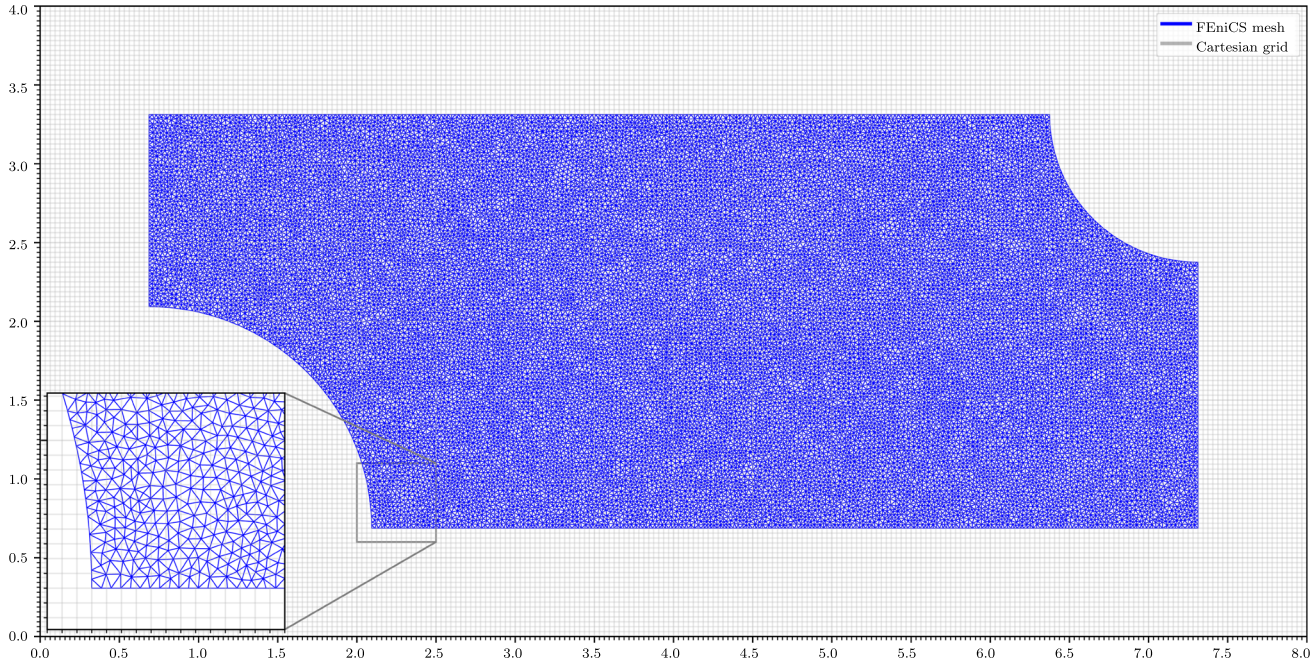

Fig. 10 Computational mesh (26,874 mesh nodes) used for the finite element computations in FEniCS and Cartesian interpolation grid (256×128) for validation scene (h) of the wave equation case

parameter when performing such numerical computations, but also when performing data-driven predictions, is the choice of time increments Δt. In numerical fluid dynamics, the Courant-Friedrichs-Lewy (CFL) number $\text{CFL} = \frac{c\Delta t}{\Delta x}$ states the number of cells of length Δx that an information of speed c can travel in one time increment Δt. While implicit Euler schemes may tolerate CFL numbers larger than 1 to ensure numerical stability, the explicit Euler integration scheme is only stable for CFL numbers smaller than 1. As the deep learning time stepper resembles an explicit Euler scheme, we chose time increments Δt, such that the CFL numbers ensure numerical stability in all configurations and application cases investigated throughout the course of this work. Future work needs to study the effect of time step size on the prediction performance in detail. Figure 10 illustrates the computational mesh (mesh convergence obtained) used for the validation scene (h) of the wave equation case.

Appendix A.2: Validation scene realizations

Figure 11 displays more examples for the validation cases used for both experiments in this work. While the domain shape is the same for all realizations of the four validation cases, the field initialization U_0 and domain parameterization c are varied. That way, many different validation cases are generated for each domain shape, out of which only four exemplary cases are displayed here for each experiment.

Appendix B: Model configurations

Two types of baseline models are considered for performance comparisons: first, fully convolutional encoder–decoder (ED) models are set up to mimic the structure of the proposed U^p-Nets but featuring only a single encoding path and no skip connections. Second, conventional U-Nets are set up that have the same structure as the respective U^p-Net, but only featuring a single encoder path. Owing to the single input layer of both baseline models, the different input tensors of the U^p-Net are stacked into one tensor along the channel direction. For example, the input to the wave equation case has five channels (three fields, the domain mask, the domain parameterization) for the baseline models. For each baseline model we chose two configurations, which solely differ in the number filters at each stage. One configuration features a constant number of filters, while the other configurations feature a cascade of doubling filters per stage, as proposed by Ronneberger et al. [10]. The general structure of encoder

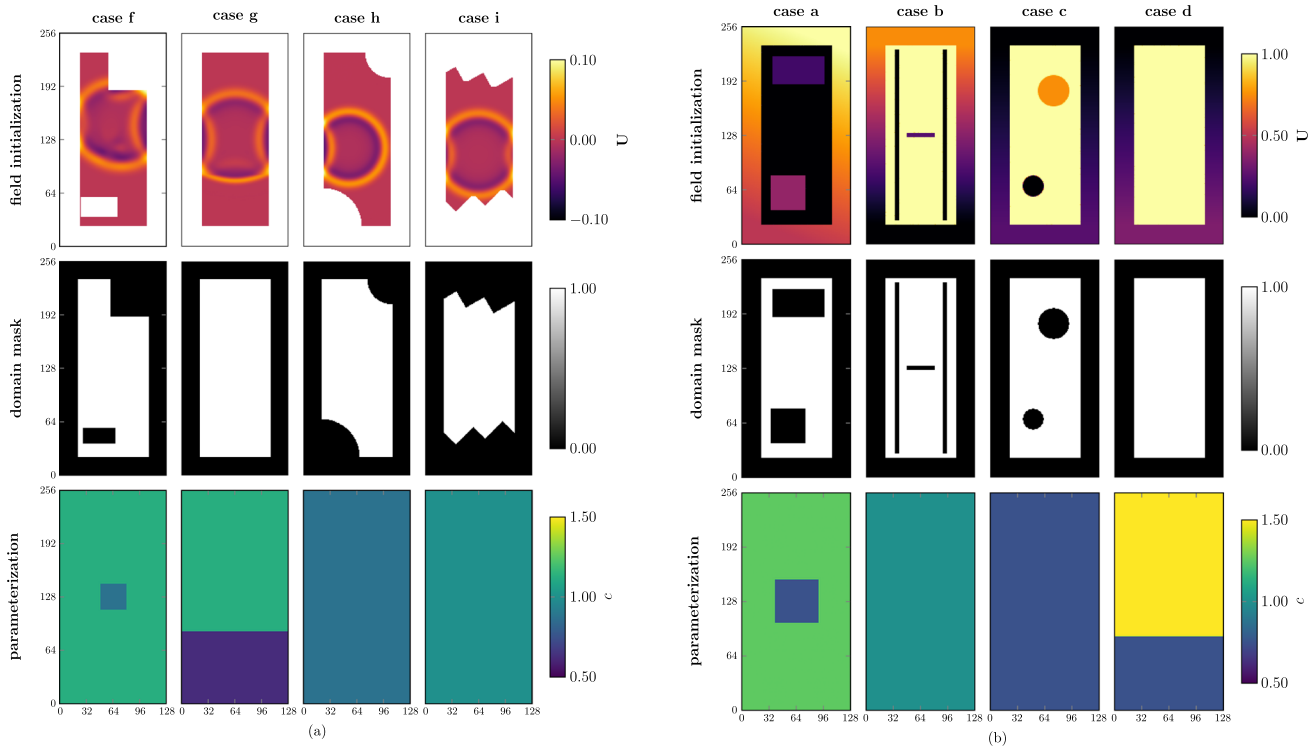

Fig. 11 Exemplary realizations of the validation cases for the wave equation studies (**a**) and the heat equation studies (**b**). The top panel shows the field initialization, the middle panel shows the domain mask, and the bottom panel shows the domain parameterization in terms of wave speed c and heat conductivity c, respectively

Table 4 Overview of baseline and proposed model configurations for the experiments presented in this work

Case	Model	Architecture	p	Stages	f_0	f_1	f_2	n_θ	Training time (h)
Wave eq	$\mathcal{U}^3_{64}$	U^p-Net	3	3	64	64	64	991, 617	175
Wave eq	$\mathcal{U}^3_{29,\times 2}$	U^p-Net	3	3	29	58	116	997, 746	115
Heat eq	$\mathcal{U}^3_{64}$	U^p-Net	3	3	64	64	64	991, 617	95
Wave eq	$\mathcal{U}_{96}$	U-Net	1	3	96	96	96	991, 681	148
Wave eq	$\mathcal{U}_{46,\times 2}$	U-Net	1	3	46	92	184	999, 857	102
Heat eq	$\mathcal{U}_{96}$	U-Net	1	3	96	96	96	991, 681	77
Wave eq	$\mathcal{U}^{\mathrm{ED}}_{106}$	ED	1	3	106	106	106	1, 006, 153	141
Wave eq	$\mathcal{U}^{\mathrm{ED}}_{48,\times 2}$	ED	1	3	48	96	192	984, 865	95
Heat eq	$\mathcal{U}^{\mathrm{ED}}_{106}$	ED	1	3	106	106	106	1, 006, 153	75

f_0, f_1 and f_2 denote the number of filters at the three encoding/decoding stages of the neural architectures, p denotes the number of encoder paths, and n_θ denotes the number of trainable parameters

and decoder paths is equivalent for all architectures considered in this work: Each stage of the encoder is comprised of two convolutional layers (3×3 kernels) followed by a max pooling layer that reduces the size of the feature map by a factor of two, while the number of filters f_k is doubled per stage k or kept constant depending on the respective configuration: 2· (conv (3×3, f_0 filters) + ReLU) + max pooling (2×2) + 2· (conv (3×3, f_1 filters) + ReLU) + max pooling (2×2) + 2· (conv (3×3, f_2 filters) + ReLU). Analogously, the decoder features 2· (conv (3×3, f_2 filters) + ReLU) + upconv(3×3, f_1 filters) + 2· (conv (3×3, 128 filters) + ReLU) + upconv(3×3, f_0 filters) + 2· (conv (3×3, f_0 filters) + ReLU) + upconv(3×3, f_0 filters) + conv(1×1, 1 filter, linear activation). We use `adam` as optimizer (parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-7}$) and an exponentially decaying learning rate $\alpha(\text{epoch}) = \alpha_{\text{base}} \cdot e^{-k \cdot \text{epoch}}$ to train the networks. Table 4 lists all model configurations used in this work.

Appendix C: Mean field quantity offsets

During our studies we found indications for a link of the mean error of the means (MEM) error metric, see Eq. (7), and the long-term prediction capabilities of the models. A large MEM indicates an incorrect shift of the whole predicted field towards larger or smaller values, i.e. a global offset. We found that this type of shifting often occurs for successive time steps, thus leading to an accumulation of offset errors in long-term predictions. Figure 12 shows the evolution of the MEM throughout the training process of the wave equation problem, as described in Sect. 4.1. Interestingly, the value of the MEM evaluated on the validation datasets oscillates quite heavily, compared to the validation loss shown in Fig. 3. We found however, that the oscillations diminish when the model is trained with very small learning rates and settle to the lower end of the oscillation range. This strategy can help to train a model in a way such that it exhibits a low MEM particularly for unseen datasets and thus additional better long-term prediction capabilities.

Appendix D: Additional results

The following paragraphs list additional results that support the findings reported in the main part of this work.

Appendix D.1: Wave equation case: single-step predictions

Table 5 reports the validation loss, MSE, MAE and MEM error metrics for a second training run for all models considered in the wave equation case study. This independent training is performed to confirm the results reported in Sect. 4.1. Again, the proposed U^p-Net architectures outperform the baseline models by a significant amount with respect to every error metric.

Appendix D.2: Wave equation case: long-term prediction studies

Figure 13 depicts the long-term prediction error accumulation of all models for the validation cases (g-i) of the wave equation scenes, that were not reported in Sect. 4.1.4. Again, long-term prediction are initialized from several time instants and run for 100 steps. It can be observed that the U^p-Net models outperform the baseline models in almost all cases

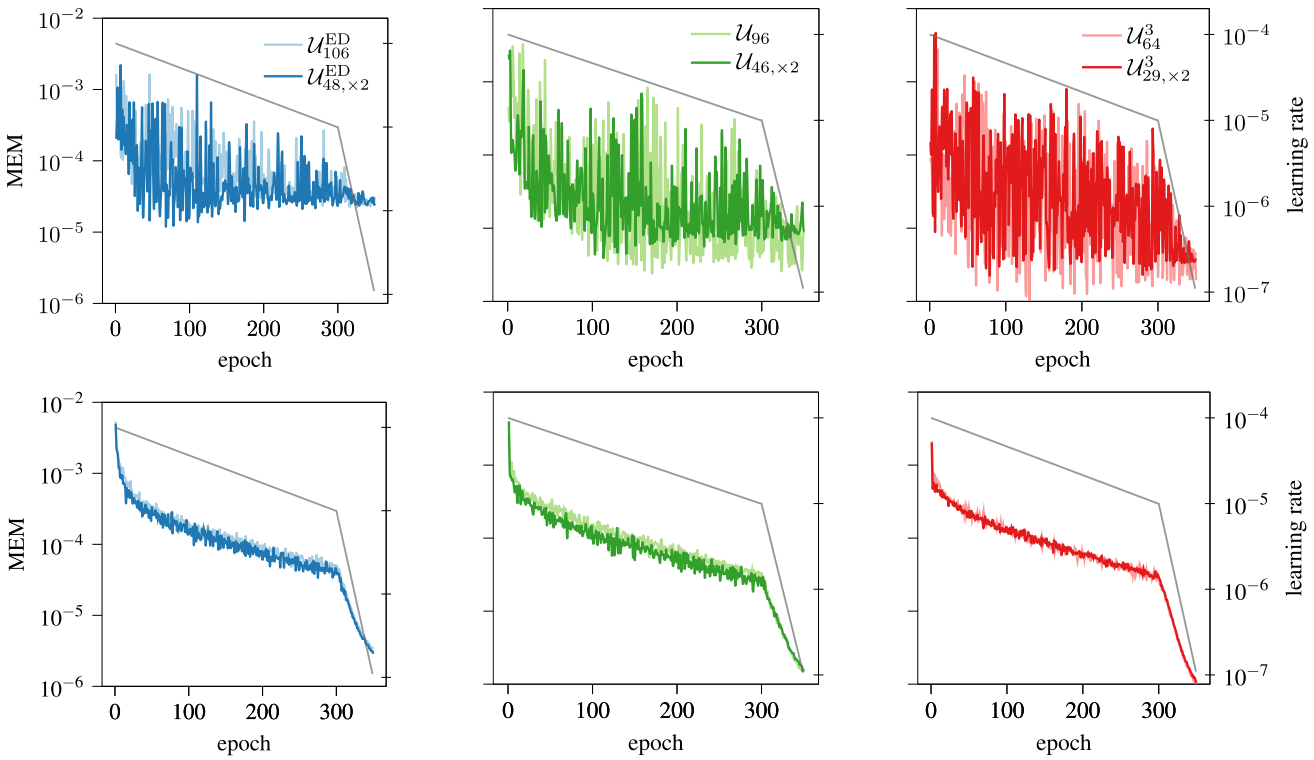

Fig. 12 Evolution of the mean error of the means (MEM) of the baseline models $\mathcal{U}^{\mathrm{ED}}$ and $\mathcal{U}$ compared to the proposed U^3-Net model $\mathcal{U}^3$ evaluated on the wave equation during training. The top panel shows the MEM evaluated on the validation data, the bottom panel shows the MEM evaluated on the training data

Table 5 Second training process: Error metrics of single-step predictions for baseline and proposed model configurations evaluated within the respective domains Ω of the four validation cases for the wave equation data

Model	Architecture	Val. loss	MSE	MAE	MEM
$\mathcal{U}^{3}_{64}$	U^{p}-Net	7.45×10^{-8}	1.30×10^{-9}	6.24×10^{-6}	8.78×10^{-7}
$\mathcal{U}^{3}_{29,\times 2}$	U^{p}-Net	7.58×10^{-8}	1.24×10^{-9}	6.42×10^{-6}	6.88×10^{-7}
$\mathcal{U}_{96}$	U-Net	4.88×10^{-7}	6.49×10^{-9}	9.04×10^{-6}	1.06×10^{-6}
$\mathcal{U}_{46,\times 2}$	U-Net	5.03×10^{-7}	7.20×10^{-9}	9.59×10^{-6}	1.42×10^{-6}
$\mathcal{U}^{\mathrm{ED}}_{106}$	ED	3.16×10^{-6}	3.88×10^{-8}	2.90×10^{-5}	2.38×10^{-6}
$\mathcal{U}^{\mathrm{ED}}_{48,\times 2}$	ED	3.49×10^{-6}	4.46×10^{-8}	3.49×10^{-5}	5.56×10^{-6}

The validation loss is evaluated on sample-wise normalized datasets and thus differs from the MSE which is evaluated on raw datasets (as are the mean absolute error MAE and the mean error of the means MEM (see 1)

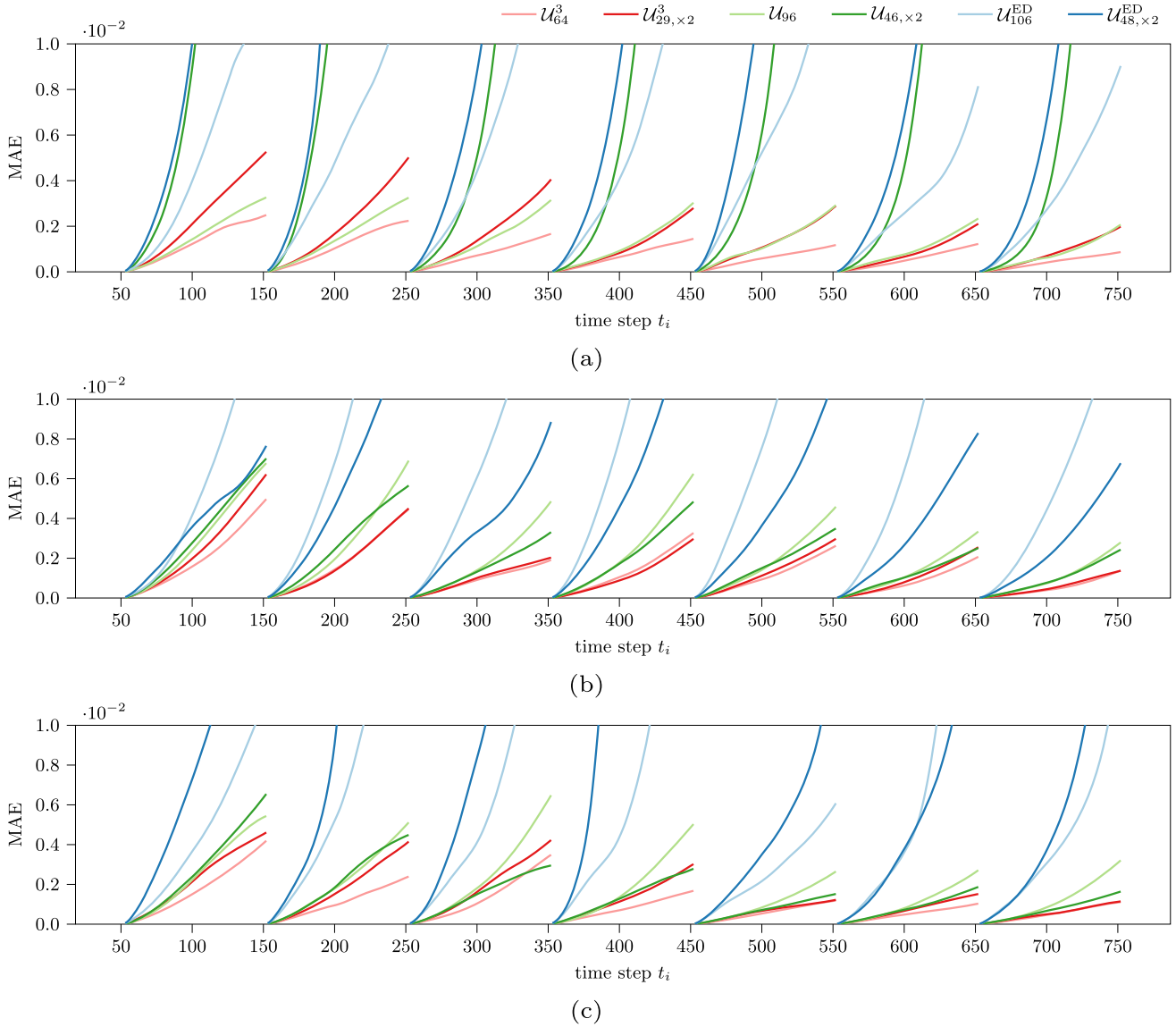

Fig. 13 Comparison of long-term predictions using the recursive time stepping scheme, starting at time steps 50, 150, 250, 350, 450, 550 and 650 of validation cases (g) to (i), shown in panels (a) to (c), and evaluated for configurations of the ED and the classical U-Net architecture each as well as two model-configurations according to the proposed U^{p}-Net architecture

Table 6 Comparison of U^3-Net, U-Net and fully convolutional encoder–decoder models in terms of the domain adaptive mean absolute error (MAE)

Model	MAE_{10}			MAE_{100}		
	Mean ± stdev	Min	Max	Mean ± stdev	Min	Max
$\mathcal{U}^3_{64}$	$8.87 \pm 4.96 \times 10^{-3}$	3.18×10^{-3}	1.82×10^{-2}	$1.34 \pm 0.90 \times 10^{-2}$	6.72×10^{-3}	3.47×10^{-2}
$\mathcal{U}_{96}$	$5.03 \pm 2.29 \times 10^{-3}$	1.49×10^{-3}	9.04×10^{-3}	$1.27 \pm 1.06 \times 10^{-2}$	2.46×10^{-3}	3.13×10^{-2}
$\mathcal{U}^{\mathrm{ED}}_{106}$	$5.37 \pm 2.44 \times 10^{-3}$	2.60×10^{-3}	9.74×10^{-3}	$6.14 \pm 3.47 \times 10^{-3}$	2.21×10^{-3}	1.14×10^{-2}

MAE_{10} and MAE_{100} denote the mean absolute error computed from self-feeding predictions after 10 and respectively 100 time steps. The self-feeding long-term predictions are performed for the four validation cases outlined in Sect. 4.2.1 and $t_{\mathrm{start}} \in \{0.0, 0.05\}$. The table shows mean and standard deviations, as well as the smallest and greatest errors

for almost all time steps. Again, the model with a constant number of feature maps $\mathcal{U}^3_{64}$ outperforms the same architecture using cascading feature maps.

Appendix D.3: Heat equation case: long-term prediction studies

Table 6 reports the prediction errors of all models for self-feeding long-term predictions. Particularly, the error after 10 prediction steps and after 100 prediction steps are reported averaged over all validation sets.

References

1. Mathews JH (1992) Numerical methods for mathematics, science and engineering, vol 10. Prentice-Hall International
2. Bezanson J, Edelman A, Karpinski S, Shah VB (2017) Julia: a fresh approach to numerical computing. SIAM Rev 59(1):65–98
3. Schönherr M, Kucher K, Geier M, Stiebler M, Freudiger S, Krafczyk M (2011) Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs. Comput Math Appl 61(12):3730–3743. https://doi.org/10.1016/j.camwa.2011.04.012. Proceedings of ICMMES-09 mesoscopic methods for engineering and science
4. Klein M, Dudek M, Clauss GF, Ehlers S, Behrendt J, Hoffmann N, Onorato M (2020) On the deterministic prediction of water waves. Fluids 5(1):9. https://doi.org/10.3390/fluids5010009
5. Virieux J, Operto S, Ben-Hadj-Ali H, Brossier R, Etienne V, Sourbier F, Giraud L, Haidar A (2009) Seismic wave modeling for seismic imaging. Lead Edge 28(5):538–544. https://doi.org/10.1190/1.3124928
6. Ravikumar N, Noble C, Cramphorn E, Taylor Z (2015) A constitutive model for ballistic gelatin at surgical strain rates. J Mech Behav Biomed Mater 47:87–94. https://doi.org/10.1016/j.jmbbm.2015.03.011
7. Sarvazyan AP, Rudenko OV, Swanson SD, Fowlkes J, Emelianov SY (1998) Shear wave elasticity imaging: a new ultrasonic technology of medical diagnostics. Ultrasound Med Biol 24(9):1419–1435. https://doi.org/10.1016/s0301-5629(98)00110-0
8. Beira MJ, Sebastião PJ (2021) A differential equations model-fitting analysis of COVID-19 epidemiological data to explain multi-wave dynamics. Sci Rep. https://doi.org/10.1038/s41598-021-95494-6
9. Krizhevsky Alex, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol 25. Curran Associates, Inc, pp 1097–1105
10. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation, In: Navab N, Hornegger J, Wells WM, Frangi AF (Eds.), Medical image computing and computer-assisted intervention—MICCAI 2015, vol 9351 of Lecture Notes in Computer Science. Springer, Cham, pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
11. Raissi M, Karniadakis GE (2018) Hidden physics models: machine learning of nonlinear partial differential equations. J Comput Phys 357:125–141. https://doi.org/10.1016/j.jcp.2017.11.039
12. Raissi M (2018) Forward–backward stochastic neural networks: deep learning of high-dimensional partial differential equations. https://arxiv.org/pdf/1804.07010
13. Raissi M, Perdikaris P, Karniadakis GE (2018) Multistep neural networks for data-driven discovery of nonlinear dynamical systems. https://arxiv.org/pdf/1801.01236
14. Stender M, Ohlsen J (2022) DeepStep: v1.0.0, Zenodo. https://doi.org/10.5281/zenodo.6244753
15. Raissi M (2018) Deep hidden physics models: deep learning of nonlinear partial differential equations. J Mach Learn Res 19(25):1–24
16. Raissi M, Perdikaris P, Karniadakis G (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045
17. Raissi M, Yazdani A, Karniadakis GE (2020) Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. Science 367(6481):1026–1030. https://doi.org/10.1126/science.aaw4741
18. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314. https://doi.org/10.1007/BF02551274
19. Brunton SL, Proctor JL, Kutz JN (2016) Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc Natl Acad Sci USA 113(15):3932–3937
20. Rudy SH, Brunton SL, Proctor JL, Kutz JN (2017) Data-driven discovery of partial differential equations. Sci Adv 3(4):e1602614. https://doi.org/10.1126/sciadv.1602614
21. Wang W-X, Yang R, Lai Y-C, Kovanis V, Grebogi C (2011) Predicting catastrophes in nonlinear dynamical systems by compressive sensing. Phys Rev Lett 106(15):154101. https://doi.org/10.1103/PhysRevLett.106.154101
22. Brunton SL, Brunton BW, Proctor JL, Kaiser E, Kutz JN (2017) Chaos as an intermittently forced linear system. Nat Commun 8(1):19. https://doi.org/10.1038/s41467-017-00030-8
23. Rudy S, Alla A, Brunton SL, Kutz JN (2019) Data-driven identification of parametric partial differential equations. SIAM J Appl Dyn Syst 18(2):643–660. https://doi.org/10.1137/18M1191944

24. Long Z, Lu Y, Dong B (2019) PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. J Comput Phys 399:108925. https://doi.org/10.1016/j.jcp.2019.108925
25. Farimani AB, Gomes J, Pande VS (2017) Deep learning the physics of transport phenomena. arXiv preprint arXiv:1709.02432
26. Sharma R, Farimani AB, Gomes J, Eastman P, Pande V (2018) Weakly-supervised deep learning of heat transport via physics informed loss. arXiv:1807.11374
27. Thuerey N, Weißenow K, Prantl L, Hu X (2020) Deep learning methods for Reynolds-averaged Navier-Stokes simulations of airfoil flows. AIAA J 58(1):25–36. https://doi.org/10.2514/1.J058291
28. Chen J, Viquerat J, Hachem E (2019) U-net architectures for fast prediction of incompressible laminar flows. arXiv preprint arXiv:1910.13532
29. Eichinger M, Heinlein A, Klawonn A (2022) Surrogate convolutional neural network models for steady computational fluid dynamics simulations. Electron Trans Numer Anal 56:235–255. https://doi.org/10.1533/etna_col56s235
30. Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia PW (2020) Learning to simulate complex physics with graph networks. arXiv:2002.09405
31. Guo X, Li W, Iorio F (2016) Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD'16, Association for Computing Machinery, New York, pp 481–490. https://doi.org/10.1145/2939672.2939738
32. Sorteberg WE, Garasto S, Cantwell CC, Bharath AA (2020) Approximating the solution of surface wave propagation using deep neural networks. In: Oneto L, Navarin N, Sperduti A, Anguita D (eds) Recent advances in big data and deep learning. Springer, Cham, pp 246–256
33. Kim B, Azevedo VC, Thuerey N, Kim T, Gross M, Solenthaler B (2019) Deep fluids: a generative network for parameterized fluid simulations. In: Computer graphics forum, vol 38. Wiley, pp 59–70
34. Wiewel S, Becher M, Thuerey N (2019) Latent space physics: towards learning the temporal evolution of fluid flow. Comput Gr Forum 38(2):71–82. https://doi.org/10.1111/cgf.13620
35. Liu Y, Kutz JN, Brunton SL (2020) Hierarchical deep learning of multiscale differential equation time-steppers. arXiv preprint arXiv:2008.09768
36. Tompson J, Schlachter K, Sprechmann P, Perlin K (2017) Accelerating Eulerian fluid simulation with convolutional networks. In: International conference on machine learning, PMLR, pp 3424–3433
37. Moseley B, Markham B, Nissen-Meyer T (2020) Solving the wave equation with physics-informed deep learning. arXiv preprint arXiv:2006.11894
38. Shelhamer E, Long J, Darrell T (2017). Fully convolutional networks for semantic segmentation. https://doi.org/10.1109/TPAMI.2016.2572683
39. Fotiadis S, Pignatelli E, Valencia ML, Cantwell C, Storkey A, Bharath AA (2020) Comparing recurrent and convolutional neural networks for predicting wave propagation. arXiv:2002.08981
40. de Bézenac E, Pajot A, Gallinari P. Deep learning for physical processes: incorporating prior scientific knowledge. CoRR abs/1711.07970. arXiv:1711.07970
41. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, PMLR, pp 448–456
42. Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O (2016) 3d U-Net: learning dense volumetric segmentation from sparse annotation, In: International conference on medical image computing and computer-assisted intervention. Springer, pp 424–432
43. Whitham GB (2011) Linear and nonlinear waves. Wiley
44. Alnæs MS, Blechta J, Hake J, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The Fenics project version 1.5, Archive of Numerical Software 3(100). https://doi.org/10.11588/ans.2015.100.20553
45. Närhi M, Salmela L, Toivonen J, Billet C, Dudley JM, Genty G (2018) Machine learning analysis of extreme events in optical fibre modulation instability. Nat Commun 9(1):1–11
46. Aksamit NO, Sapsis T, Haller G (2020) Machine-learning mesoscale and submesoscale surface dynamics from Lagrangian ocean drifter trajectories. J Phys Oceanogr 50(5):1179–1196
47. Vlachas PR, Pathak J, Hunt BR, Sapsis TP, Girvan M, Ott E, Koumoutsakos P (2020) Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. Neural Netw 126:191–217
48. Chu B, Farazmand M (2021) Data-driven prediction of multistable systems from sparse measurements. Chaos Interdiscip J Nonlinear Sci 31(6):063118