Towards Multi-spatiotemporal-scale Generalized PDE Modeling

Anonymous authors
Paper under double-blind review

Abstract

Partial differential equations (PDEs) are central to describing complex physical system simulations. Their expensive solution techniques have led to an increased interest in deep neural network based surrogates. However, the practical utility of training such surrogates is contingent on their ability to model complex multi-scale spatio-temporal phenomena. Various neural network architectures have been proposed to target such phenomena, most notably Fourier Neural Operators (FNOs), which give a natural handle over local & global spatial information via parameterization of different Fourier modes, and U-Nets which treat local and global information via downsampling and upsampling paths. However, generalizing across different equation parameters or time-scales still remains a challenge. In this work, we make a comprehensive comparison between various FNO, ResNet, and U-Net like approaches to fluid mechanics problems in both vorticity-stream and velocity function form. For U-Nets, we transfer recent architectural improvements from computer vision, most notably from object segmentation and generative modeling. We further analyze the design considerations for using FNO layers to improve performance of U-Net architectures without major degradation of computational cost. Finally, we show promising results on generalization to different PDE parameters and time-scales with a single surrogate model.

1 Introduction

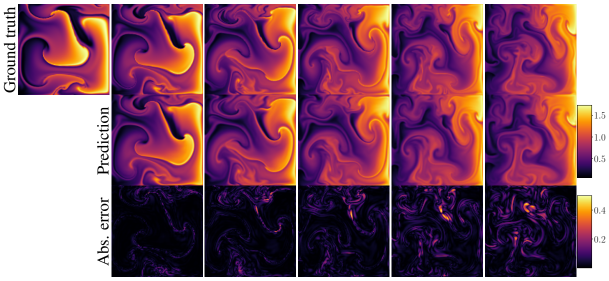


Figure 1: Example rollout trajectories of the best-performing U-Net model, which is trained to generalize across different timesteps (Δt) and different force terms.

Many mathematical models of physical phenomena are expressed in differential equation forms (Olver, 1986), generally as temporal partial differential equations (PDEs). Their expensive solution techniques have led to an increased interest in deep neural network based surrogates (Bar-Sinai et al., 2019; Raissi et al., 2019; Lu et al., 2021; Li et al., 2020a; Brandstetter et al., 2022c; Um et al., 2020); especially in the studies that relate to fluid dynamics (Guo et al., 2016; Kochkov et al., 2021; Rasp & Thuerey, 2021; Keisler, 2022; Weyn et al., 2020; Sønderby et al., 2020; Wang et al., 2020a; Pathak et al., 2022). However, generalizing across different PDE parameters, and different time-scales is a notoriously hard problem. For example, in fluid mechanics, slightly different values of a single parameter like Reynolds numbers can make all the difference for flows being laminar or turbulent. Another source of challenge stems from the fact that physical phenomena appear at different spatial and temporal scales. For example, blizzards are rather local weather phenomena, whereas heat waves are rather global ones, both resulting from the same underlying principles. For exactly these reasons, fluid mechanics in general (Munson et al., 2013), and weather forecasting in particular (Jolliffe & Stephenson, 2012) have always posed a great scientific challenge.

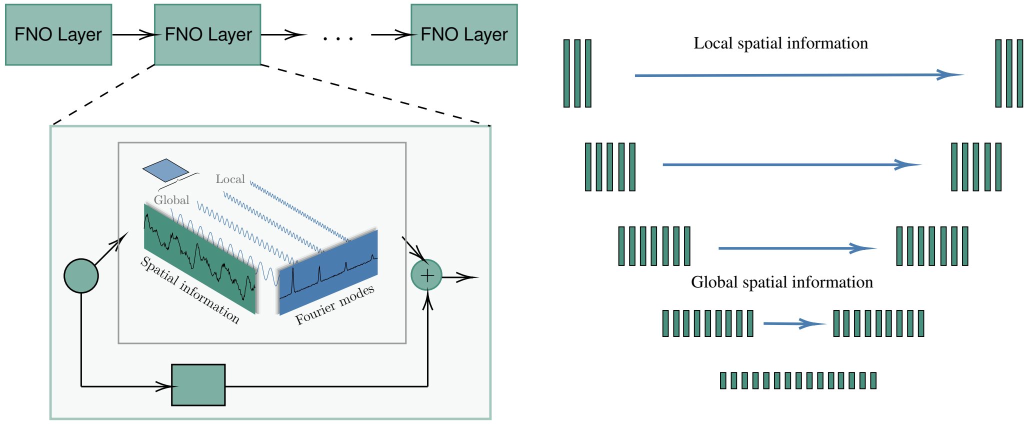


Figure 2: Information flow in Fourier based (left) and U-Net based architectures (right). FNO layers (Li et al., 2020a) consist of Fast Fourier transforms and weight multiplication in the Fourier space. Low Fourier modes provide global and high Fourier modes provide local information. U-Nets (Ronneberger et al., 2015) are constructed as a spatial downsampling pass, followed by a spatial upsampling pass, where information from the downsampling pass is added via skip-connections.

Prominent examples of neural PDE surrogates are **Fourier Neural Operators (FNOs)** (Li et al., 2020a). At its core, FNO building blocks consist of Fast Fourier transforms (FFTs) (Cooley & Tukey, 1965; Van Loan, 1992) and weight multiplication in the Fourier space, where low Fourier modes provide global information, and high Fourier modes provide local information. An FNO layer processes global and local information simultaneously via weight multiplication of the different modes. On the other hand, **U-Nets** (Ronneberger et al., 2015) are standard architectures in the context of image modeling, image segmentation, and image generation. U-Nets are constructed as a spatial downsampling pass followed by a spatial upsampling pass, with additional skip connections present between the downsampling pass activations and corresponding upsampling layers. Local and global information is therefore treated in a more distributed fashion than in FNO like architectures. Downsampling corresponds to sequentially processing information more globally, whereas upsampling corresponds to fine-graining the global information and adding local information via skip connections. Figure 2 contrasts local and global information flows in FNO and U-Net like architectures. Given the recent success of modern U-Net architectures in complex generative image modeling tasks (Ho et al., 2020; Nichol & Dhariwal, 2021; Ramesh et al., 2021) it's pertinent that these are evaluated on PDE Operator learning tasks and compared to FNO like approaches. Furthermore, given the different nature of FNO and U-Net like approaches, it is worth surveying their respective advantages and performance on different tasks, as well as investigating under which circumstances combining them might be beneficial. The third line of models are **ResNet** (He et al., 2016a) like architectures, which a priori have no natural handle on processing local and global information – in contrast to recently introduced **Dilated ResNets** (Stachenfeld et al., 2021) which adapt filter sizes at different layers via dilated convolutions.

To summarize our contributions: (1) To our knowledge, we are the first to present a side-by-side analysis of FNO, ResNet, and U-Net like architectures on their ability to model complex multi-scale spatio-temporal phenomena. In doing so, we present new architecture designs based on modern updates to U-Nets. (2) We generalize to different PDE parameters and time-scales showing promising results for single surrogate models as exemplified in Figure 1. (3) We propose a unified PyTorch based framework for enabling easy side-by-side comparisons of various PDE operator learning methods which we plan to opensource.

2 Preliminaries

Common parameterization of Fourier transform layers. The discrete Fourier transform (DFT) together with point-wise multiplication in the Fourier space is the heart of Fourier Neural Operator (FNO) layers. DFTs convert an n-dimensional complex signal $f(x) = f(x_1, \ldots, x_n) : \mathbb{R}^n \to \mathbb{C}$ at

$M_1 \times \ldots \times M_n$ grid points into its complex Fourier modes $\hat{f}(\xi_1, \ldots, \xi_n)$ via:

$$\hat{f}(\xi_1, \ldots, \xi_n) = \mathcal{F}\{f\}(\xi_1, \ldots, \xi_n) = \sum_{m_1=0}^{M_1} \ldots \sum_{m_n=0}^{M_n} f(x) \cdot e^{-2\pi i \cdot \left(\frac{m_1 \xi_1}{M_1} + \ldots + \frac{m_n \xi_n}{M_n}\right)} , \tag{1}$$

where $(\xi_1, \ldots, \xi_n) \in \mathbb{Z}_{M_1} \ldots \times \ldots \mathbb{Z}_{M_n}$. In FNO layers, discrete Fourier transforms on real-valued input fields and respective back-transforms – implemented as Fast Fourier Transforms[1] on real-valued inputs (RFFTs)[2] – are interleaved with a weight multiplication by a complex weight matrix of shape $c_{\text{in}} \times c_{\text{out}}$ for each mode, which results in a complex-valued weight tensor of the form $W \in \mathbb{C}^{c_{\text{in}} \times c_{\text{out}} \times (\xi_1^{\text{max}} \times \ldots \times \xi_n^{\text{max}})}$, where Fourier modes above cut-off frequencies $(\xi_1^{\text{max}}, \ldots, \xi_n^{\text{max}})$ are set to zero. These cut-off frequencies turn out to be important hyperparameters. Additionally, a residual connection is usually implemented as convolution layer with kernel size 1 (see Figure 2).

Common parameterizations of convolution layers. Regular convolutional neural network (CNN) (Fukushima & Miyake, 1982; LeCun et al., 1998) layers are the basic building blocks of U-Net like architectures. CNNs take as input feature maps $f : \mathbb{Z}^n \to \mathbb{R}^{c_{\text{in}}}$ and convolve[3] them with a set of c_{out} filters $\{w^i\}_{i=1}^{c_{\text{out}}}$ with $w^i : \mathbb{Z}^n \to \mathbb{R}^{c_{\text{in}}}$:

$$[f \star w^i](x) = \sum_{y \in \mathbb{Z}^n} \left\langle f(y), w^i(y - x) \right\rangle , \tag{2}$$

which can be interpreted as an inner product of input feature maps with the corresponding filters at every point $y \in \mathbb{Z}^n$. The filter size of a convolutional layer is a crucial choice in neural network design since it defines the regions from which information is obtained. Common practice is to use rather small filters (Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016a). Continuous formulations of filters were introduced to handle irregularly sampled data (Schütt et al., 2018; Simonovsky & Komodakis, 2017; Wang et al., 2018; Wu et al., 2019) and to match the resolution of the underlying data (Peng et al., 2017; Cordonnier et al., 2019; Romero et al., 2021b). A promising direction is to adapt filter sizes at different layers using dilation (Dai et al., 2017). For example, dilated convolutions in the context of PDE modeling were proposed in Stachenfeld et al. (2021). As a downside, dilations might limit the bandwidth of the filters, and thus the amount of collected detail. A rather new direction is therefore to adapt filter sizes either via learnable dilation (Pintea et al., 2021) or via flexible sized continuous convolutions (Romero et al., 2021a; 2022).

Connecting Fourier transform and convolution. Starting with the 1-dimensional case and omitting channel dimensions, we assume a signal consisting of n input points, and we further assume circular padding. We can now rewrite Equation 2 into a discrete *circular convolution* (Bamieh, 2018; Bronstein et al., 2021) of two $n-$dimensional vectors $\boldsymbol{f}, \boldsymbol{w} \in \mathbb{R}^n$:

$$[\boldsymbol{f} \star \boldsymbol{w}]_i = \sum_{j=0}^{n} \boldsymbol{w}_{(i-j) \bmod n} \boldsymbol{f}_j = \sum_{j}^{n-1} (\mathbf{C}_w)_{ij} \boldsymbol{f}_j , \tag{3}$$

The indexing $(i - j) \bmod n$ returns circular shifts, which can be combined into a circulant matrix $\mathbf{C}_w$. It is general practice to use rather small filters which only consist of k non-zero elements where usually $k << n$. The remaining elements of $\mathbf{C}_w$ are filled up with zeros. The action of $\mathbf{C}_w$ on $\boldsymbol{f}$, or equivalently the convolution of $\boldsymbol{f}$ with $\boldsymbol{w}$ can be expressed via the convolution theorem:

$$\mathbf{C}_w \boldsymbol{f} = \left(\frac{1}{\sqrt{n}} \mathbf{W}\right) \mathbf{D} \left(\frac{1}{\sqrt{n}} \mathbf{W}^*\right) \boldsymbol{f} , \tag{4}$$

where the matrix $\mathbf{W}$ consists of the eigenvectors of $\mathbf{C}_w$ and $\mathbf{W}^*$ is its complex conjugate. All circulant matrices have the same eigenvectors, which if multiplied with a signal yields the discrete Fourier transform (DFT) of the signal. That is, multiplication (from left) with $\mathbf{W}^*$ is the discrete Fourier transform (of $\boldsymbol{f}$), and multiplication by $\mathbf{W}$ is the inverse Fourier transform. The matrix $\mathbf{D}$ has the Fourier modes of the vector $\boldsymbol{w}$ on its diagonal. Thus, we can analyze a convolution by expressing its filters as vectors $\boldsymbol{w} \in \mathbb{R}^n$, which comprise the actual k filter values (k corresponds to the

[1] Fast Fourier transforms (FFTs) immensely accelerate DFT computation by factorizing Equation 1 into a product of sparse (mostly zero) factors.

[2] The FFT of a real-valued signal is Hermitian-symmetric, so the output contains only the positive frequencies below the Nyquist frequency for the last spatial dimension.

[3] In deep learning, a convolution operation in the forward pass is implemented as cross-correlation.

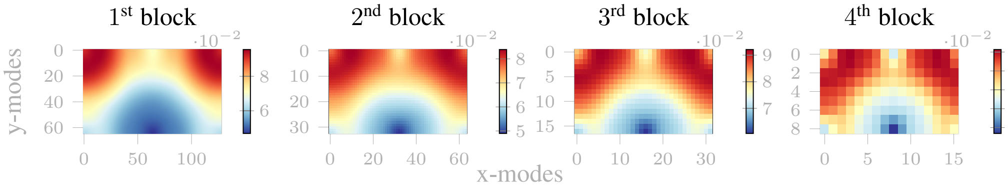


Figure 3: Analyzing filter properties of trained U-Net architectures. Absolute values of Fourier modes of the filters in each first layer of the respective down-sampling blocks are shown, where for each mode the average is taken over all filters.

kernel size) and additional $n - k$ zeros. When we extend the circular convolution approach to two dimensions, $\boldsymbol{w}$ itself becomes a matrix $\boldsymbol{w} \in \mathbb{R}^{n \times n}$. In Figure 3, we plot the Fourier modes of the two dimensional filters of a trained U-Net. We take the absolute values of the modes and average for each mode over all filters in the first convolution layer of different down-sampling blocks. Although precise statements are difficult to make, it is evident that Fourier mode averages of different blocks are downsampled versions of each other, which complies with the interpretation that the downsampling blocks of U-Nets process information at different scales. This is therefore in contrast to FNO like architectures which process different scales within each FNO layer.

Fourier transform for downsampling. Bandlimited pre-subsampling (Mallat, 1999), i.e. suppressing high-frequencies before down-sampling, is a well know technique in signal processing; for an illustrative example see e.g. Figure 1 in Worrall & Welling (2019). We hypothesize that replacing convolutions with FNO layers which set Fourier modes above cut-off frequencies to zero might be advantageous, especially in the lower parts of the downsampling blocks of U-Net architectures. The counter-hypothesis is that convolutions are all what is needed to learn an efficient downsampling.

Partial differential equations. A partial differential equation (PDE) relates solutions $\boldsymbol{u} : \mathcal{X} \rightarrow \mathbb{R}^n$ and respective derivatives for all points $\boldsymbol{x}$ in the domain $\mathcal{X} \in \mathbb{R}^m$, where $\boldsymbol{u}^0(\boldsymbol{x})$ are *initial conditions* at time $t = 0$ and $B[\boldsymbol{u}](t, \boldsymbol{x}) = 0$ are *boundary conditions* with boundary operator B when $\boldsymbol{x}$ is on the boundary $\partial\mathbb{X}$ of the domain. In this work, we investigate PDEs of fluid mechanics problems. To be more precise, we focus on the **incompressible Navier-Stokes** equations (Temam, 2001), in velocity function and vorticity stream formulation. In 2 dimensions, the Navier-Stokes equations in **vector velocity form** conserve the velocity flow fields $\boldsymbol{v} : \mathcal{X} \rightarrow \mathbb{R}^2$ where $\mathcal{X} \in \mathbb{R}^2$ via:

$$\frac{\partial \boldsymbol{v}}{\partial t} = -\boldsymbol{v} \cdot \nabla \boldsymbol{v} + \mu \nabla^2 \boldsymbol{v} - \nabla p + \boldsymbol{f} \ , \qquad \nabla \cdot \boldsymbol{v} = 0 \ , \tag{5}$$

where $\boldsymbol{v} \cdot \nabla \boldsymbol{v}$ is the convection, i.e. the rate of change of $\boldsymbol{v}$ along $\boldsymbol{v}$, $\mu \nabla^2 \boldsymbol{v}$ the viscosity, i.e. the diffusion or net movement of $\boldsymbol{v}$, ∇p the internal pressure and $\boldsymbol{f}$ an external force. An additional incompressibility constraint $\nabla \cdot \boldsymbol{v} = 0$ yields mass conservation of the Navier-Stokes equations.

By introducing the vorticity $\omega : \mathcal{X} \in \mathbb{R}$ as the curl of the flow velocity, i.e. $\omega = \nabla \times \boldsymbol{v}$, we can rewrite the incompressible 2-dimensional Navier-Stokes equations in **scalar vorticity stream function form** (Kundu et al., 2015; Guyon et al., 2001; Acheson, 1991) as:

$$\frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} + \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial x^2}\right) \ , \qquad \left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial x^2}\right) = -\omega \ , \tag{6}$$

where the streamfunction is defined via the relations $\frac{\partial \phi}{\partial y} = \boldsymbol{v}_x$ and $\frac{\partial \phi}{\partial x} = -\boldsymbol{v}_y$, and Re is the Reynolds number which is indirectly proportional to the viscosity and proportional to the absolute velocity. As a result, the 2D incompressible Navier-Stokes equations are turned into one parabolic equation, i.e. the vorticity transport equation (Equation 6 left), and one elliptic equation, i.e. the Poisson equation (Equation 6 right). Since the streamfunction is directly obtained from the vorticity via the Poisson equation one usually solves for the scalar vorticity.

The **shallow water** equations (Vreugdenhil, 1994) can be derived from integrating the incompressible Navier–Stokes equations, in cases where the horizontal length scale is much larger than the vertical length scale. As such, shallow water equations describe a thin layer of fluid of constant density in hydrostatic balance, bounded from below by the bottom topography and from above by a free surface. For simplified weather modeling, the shallow water equations express the velocity in x- direction, or zonal velocity, the velocity in the y- direction, or meridional velocity, and the vertical displacement of free surface, which subsequently is used to derive pressure fields. Since the shallow water equations are derived from the Navier-Stokes equations, a vorticity-stream function formulation exists as well. Note however that when it comes to describing flows in e.g. more complex geometries, the velocity formulation is in general easier to deal with (Kundu et al., 2015).

3 PDE SURROGATES

(Dilated) ResNets. We implement ResNet architectures using 8 residual blocks, where each block consists of two convolution layers with 3×3 kernels, shortcut connections, group normalization (Wu & He, 2018), and GeLU activation functions (Hendrycks & Gimpel, 2016). In contrast to standard ResNets for image classification, we don't use any down-projection techniques, e.g. convolution layers with strides larger than 1 or via pooling layers. In doing so, ResNets have no natural built-in handle over local and global informations, and therefore serve as important baseline to ablate effects of local and global information flow which is fundamental in e.g. FNO and U-Net like architectures. Recently, Stachenfeld et al. (2021) introduced Dilated ResNets, which adapt filter sizes at different layers using dilated convolutions, and thus are an alternative way of subsequently aggregating global information. The models consist of 4 residual blocks where each block individually consists of 7 dilated CNN layers with dilation rates of $[1, 2, 4, 8, 4, 2, 1]$. We implement Dilated ResNets with and without group normalization layers.

Fourier Neural Operators. We implement FNO architectures where the number of FNO layers, the number of channels, and the number of non-zero Fourier modes are hyperparameters. All architectures consist of two embedding and two output layers. Each FNO layer comprises a convolution path with a 1×1 kernel and a Fourier path where pointwise weight multiplication is done for the lower modes in the Fourier domain. We use GeLU activation functions, and no normalization scheme.

U-Nets. U-Nets have already been used as PDE surrogates in Ma et al. (2021); Chen & Thuerey (2021). U-Nets are constructed as a spatial downsampling followed by a spatial upsampling pass, where each down- and upsampling block consists of two convolutional layers. A particularity of U-Nets is the presence of skip connections between the downsampling pass activations and corresponding upsampling layers. Orignially, downsampling is achieved via max-pooling operations. We term the 2015 U-Net implementation as U-Net$_{2015}$, which is based on the `PDEbench` repository of Takamoto et al. (2022). Furthermore, we include a slightly different version which we term U-Net$_{\text{base}}$ which has bias weights and group normalization instead of batch normalization to be comparable with modern U-Net versions. To match the number of weights of U-Net$_{2015}$, the bottleneck layer in U-Net$_{\text{base}}$ is omitted. Modern versions of the architecture (Ho et al., 2020; Nichol & Dhariwal, 2021; Ramesh et al., 2021) often use Wide ResNet (Zagoruyko & Komodakis, 2016) style 2D convolutional blocks, each of which can be followed by a spatial attention block (Vaswani et al., 2017). Other notable changes are the substitution of max-pooling operations by downsampling layers. We term the respective implementations U-Net$_{\text{mod}}$ and U-Net$_{\text{att}}$ in our experiments.

Fourier U-Nets. Based on the insights of Section 2, we replace lower blocks both in the downsampling and in the upsampling path of U-Net architectures by Fourier blocks, where each block consists of 2 FNO layers and residual connections. We test substituting only the lowest block (U-F1Net), and the lowest two blocks (U-F2Net) of the U-Net$_{\text{mod}}$ architecture. Substituting all blocks would yield an architecture which resembles the UNO architecture (Rahman et al., 2022b), with the difference that in UNO architectures downsampling is done individually via linear layers along the x- and y- dimension, and that "mode scheduling" reduces the number of modes for higher blocks in the respective downsampling and upsampling paths. For complete comparison, we therefore also implement the UNO architecture[4].

3.1 OPERATOR LEARNING

Major practical benefits of neural PDE surrogates come from amortizing the cost of their compute-expensive training process which depends on the surrogates' ability to effectively generalize across different parameter settings as well as across different time discretizations. Operator learning is a popular term for training these neural surrogates. Theoretical grounding arises from Chen & Chen

[4] We based our implementation on `https://github.com/ashiq24/UNO`

(1995) who extend the universal approximation theorem in neural networks (Hornik et al., 1989; Cybenko, 1989) to operator approximation, forming the basis for DeepONets (Lu et al., 2019) with theoretical extensions in Lu et al. (2021), graph kernel networks (Li et al., 2020b), and FNOs. An impressive comparison of DeepONets and FNOs can be found in Lu et al. (2022).

Operator learning (Lu et al., 2019; Li et al., 2020b;a; Lu et al., 2021; 2022) relates solutions $\boldsymbol{u} : \mathcal{X} \to \mathbb{R}^n$, $\boldsymbol{u}' : \mathcal{X}' \to \mathbb{R}^{n'}$ defined on different domains $\mathcal{X} \in \mathbb{R}^m$, $\mathcal{X}' \in \mathbb{R}^{m'}$ via operators $\mathcal{G}$:

$$\mathcal{G} : (\mathcal{U} \in \boldsymbol{u} \to \boldsymbol{u}' \in \mathcal{U}') \, , \tag{7}$$

where $\mathcal{U}$ and $\mathcal{U}'$ are the spaces of solutions $\boldsymbol{u}$ and $\boldsymbol{u}'$, respectively.

Parameter conditioning. We evaluate FNO and U-Net like architectures on their generalization capabilities across PDE parameters and different time-scales. The chosen data sets to do so consist of solution pairs $\boldsymbol{u}, \boldsymbol{u}' \in \mathcal{U}$ where the pair itself is from the same solution space $\mathcal{U}$, but different pairs $\{\boldsymbol{u}, \boldsymbol{u}'\}_1$ and $\{\boldsymbol{u}, \boldsymbol{u}'\}_2$ are from different solution spaces $\mathcal{U}_1$ and $\mathcal{U}_2$ characterized by different force terms. Further, the mapping $\boldsymbol{u} \to \boldsymbol{u}'$ should generalize across different time windows Δt. We therefore train neural surrogates to generalize across different initial conditions, different PDE parameters (force terms) and different time windows. Both time windows Δt and force terms are continuous scalar parameters, and thus can be encoded into a vector representation by using sinusoidal embeddings as is common in Transformers (Vaswani et al., 2017) and various neural implicit representation learning techniques (Mildenhall et al., 2021).

4 Experiments

We establish the following set of desiderata for our benchmarks: (i) *simplicity*: the tasks should be easy to setup, while being backed by actual PDE solvers written by domain experts, (ii) *challenging*: the tasks should be difficult enough, (iii) *diverse*: the tasks should be diverse, both in their formulation as well as in their requirements, and (iv) *generalizability*: the tasks should probe generalization across different time horizons as well as different parameter settings. Following these desiderata, we assessed the described architectures in four experimental settings to probe (i) Fourier vs. U-Net based approaches, (ii) differences due to the velocity vs. vorticity formulation of the datasets, and (iii) parameter conditioning performance. Results of the main paper are complemented by comprehensive studies and various ablations in Appendix B.
All datasets contained multiple input and output fields. More precisely, one scalar and one velocity vector field in case of the velocity formulation, and two scalars in case of the vorticity formulation. Inputs to the neural PDE surrogates were respective fields at previous t timesteps, where t varies for different PDEs. The *one-step loss* is the mean-squared error at the next timestep summed over fields. The *rollout loss* (reported in Appendix B) is the mean-squared error after applying the neural PDE surrogate 5 times, summing over fields and time dimension. We alternatively test the relative MSE loss as used in Li et al. (2020a). We optimized models using the AdamW optimizer (Kingma & Ba, 2014; Loshchilov & Hutter, 2019) for 50 epochs and minimized the summed mean squared error. We used cosine annealing as learning rate scheduler (Loshchilov & Hutter, 2016) with a linear warmup. Table 1 compares parameter count, runtime and memory requirement of the tested architectures, showing that runtime and memory requirements are in the same ballpark for both architecture families if the number of parameters is kept similar.

Shallow water equations. We modified the implementation in `SpeedyWeather.jl`[5](Klöwer et al., 2022), obtaining data on a grid with spatial resolution of 192×96 ($\Delta x = 1.875°$, $\Delta y = 3.75°$), and temporal resolution of $\Delta t = 48\,\mathrm{h}$. We first evaluated the different architectures on the shallow water equations in velocity function formulation, predicting scalar pressure field and vector wind velocity field. Figure 4 (left) shows results obtained by various models. In general, all methods which have a dedicated local and global information flow, i.e. Dilated ResNet, FNO, and U-Net architectures, perform rather well. Nevertheless, across all tested models, performance differences of an order of magnitude arise, where U-Nets in general perform best. Adding FNO blocks to U-Net architectures (U-F1Net, U-F2Net) seems to be beneficial. We further evaluate on the shallow water equations in vorticity stream formulation, and predict the scalar pressure field and the scalar wind vorticity field. Figure 4 (middle) shows results obtained by various models. Performance-wise a similar pattern arises, where again the lowest losses are observed for U-Net architectures.

[5] https://github.com/milankl/SpeedyWeather.jl

Table 1: Comparison of parameter count, runtime, and memory requirement of various architectures. Subscript numbers indicate the used number of Fourier modes. For U-FNet experiments subscript numbers indicate the number of Fourier modes in the lowest and second-lowest block.

METHOD	Channels	Res.Layers/Blocks	Params.	Runtime [s]		Mem. [MB]	
				Fwd.	Fwd.+bwd.	`f32` size	Peak usage
ResNet128	128	8	2.4 M	0.084	0.180	9	4273
ResNet256	256	8	9.6 M	0.231	0.497	38	8500
DilResNet128	128	4	4.2 M	0.118	0.342	16	4849
DilResNet128-norm	128	4	4.2 M	0.183	0.423	16	6922
FNO128-8$_{\text{modes8}}$	128	8	33.7 M	0.057	0.162	134	2161
FNO128-8$_{\text{modes16}}$	128	8	134 M	0.059	0.171	537	2953
FNO128-4$_{\text{modes16}}$	128	4	67.2 M	0.031	0.089	268	1852
FNO64-4$_{\text{modes32}}$	64	4	67.1 M	0.016	0.050	268	1204
FNO96-4$_{\text{modes32}}$	96	4	151 M	0.026	0.080	604	2179
FNO128-4$_{\text{modes32}}$	128	4	268 M	0.036	0.118	1100	3420
UNO64	64	7	110 M	0.070	0.134	440	1925
UNO128	128	7	440 M	0.160	0.341	1800	5513
U-Net$_{2015}$64	64	9	31 M	0.013	0.037	124	1305
U-Net$_{2015}$128	128	9	124 M	0.042	0.117	496	3002
U-Net$_{\text{base}}$64	64	8	31.1 M	0.021	0.046	124	1277
U-Net$_{\text{base}}$128	128	8	124 M	0.056	0.132	496	3000
U-Net$_{\text{mod}}$64	64	9	144 M	0.079	0.184	577	3900
U-Net$_{\text{att}}$64	64	9	148 M	0.081	0.190	593	3975
U-F1Net$_{\text{modes8}}$	64	9	154 M	0.083	0.205	617	3936
U-F1Net$_{\text{modes16}}$	64	9	185 M	0.084	0.208	743	4037
U-F2Net$_{\text{modes8,4}}$	64	9	163 M	0.085	0.213	652	3961
U-F2Net$_{\text{modes8,8}}$	64	9	193 M	0.085	0.216	772	4046
U-F2Net$_{\text{modes16,8}}$	64	9	224 M	0.086	0.219	897	4149
U-F2Net$_{\text{modes16,16}}$	64	9	344 M	0.090	0.232	1400	4496

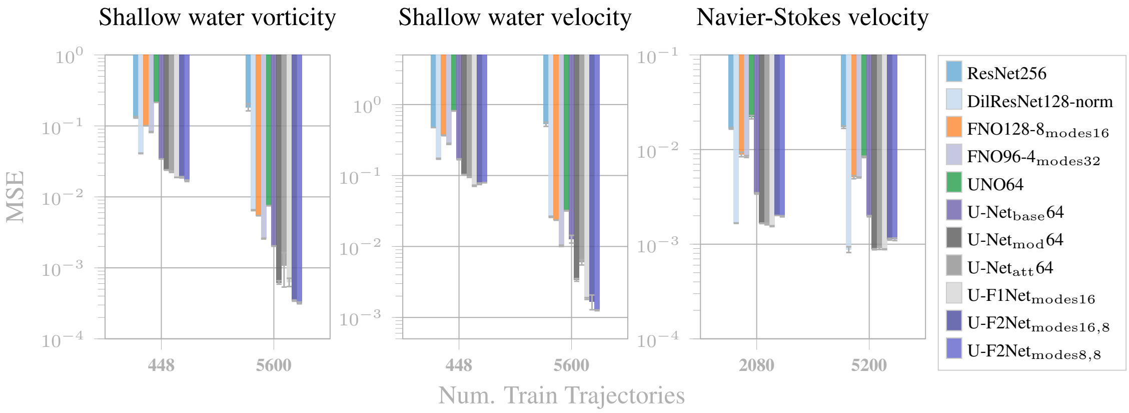


Figure 4: One-step errors for modeling different PDEs, shown for different number of training trajectories. Results are averaged over three different random seeds, and are obtained for the velocity function and vorticity stream formulation of the shallow water equations on 2-day prediction (left, middle), and for the Navier-Stokes equation (right). For better visibility only selected architectures are displayed, for full comparisons see Appendix B. Note the logarithmic scale of the y-axes.

Velocity function formulation of Navier-Stokes equations. We further tested on Navier-Stokes equations in velocity function form, which is more common in the real world than the vorticity stream function form. In addition to the velocity field $\boldsymbol{v}$ of Equation 5, we introduced a scalar field representing a scalar quantity, i.e. particle concentration, that is being transported via the velocity field. The scalar field is *advected* by the vector field, i.e. as the vector field changes, the scalar field is transported along with it. Complementary, the scalar field influences the vector field only via an external buoyancy force term in y-direction, i.e. $\boldsymbol{f} = (0, f)^T$. We obtained data on a grid with spatial resolution of 128×128 ($\Delta x = 0.25$, $\Delta y = 0.25$), and temporal resolution of

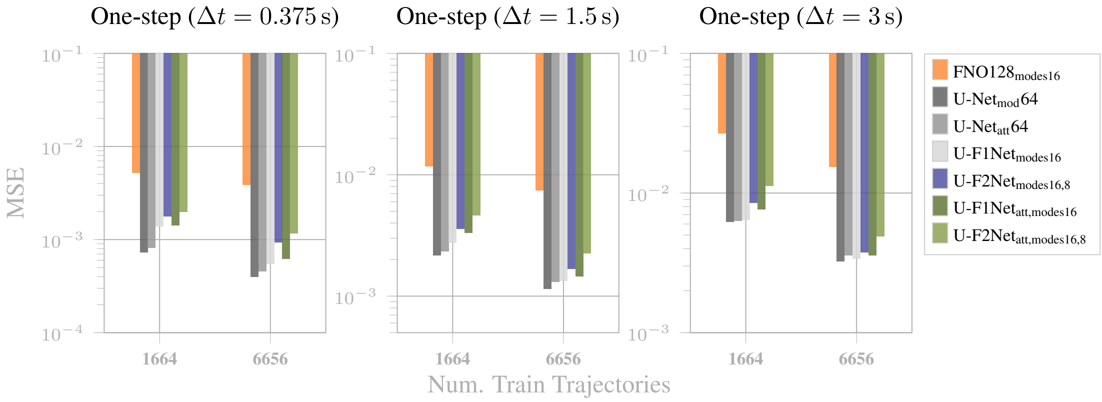


Figure 5: One-step errors obtained on the parameter conditioning experiments of the Navier-Stokes equation. Results are shown for selected architectures, different number of training trajectories, and different time windows: $\Delta t = 0.375$ s (left), $\Delta t = 1.5$ s (middle), and $\Delta t = 3$ s (right). Results are averaged over 208 different unseen evaluation buoyancy force values between 0.2 and 0.5.

$\Delta t = 1.5$ s using ΦFlow[6] (Holl et al., 2020). Figure 4 (right) shows results obtained by different architectures. Compared with shallow ater experiments, the compute-expensive Dilated ResNet architectures perform on par with the best U-Net and U-FNet architectures.

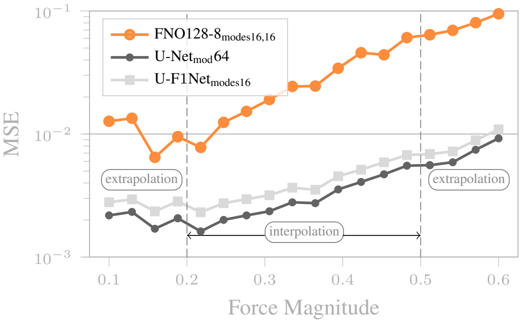
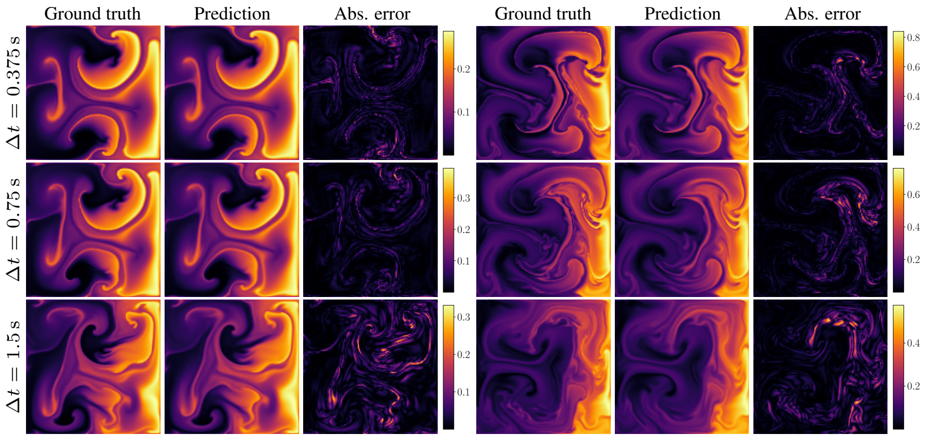


Figure 6: Inter- and extrapolation performance of different models tested on buoyancy force values in the range $0.1 \leq f \leq 0.6$ performing 5 steps rollout at $\Delta t = 0.375$ s.

Probing parameter conditioning. We probe parameter conditioning on the velocity function formulation of the Navier-Stokes equation. We test FNO and U-(F)Net variants, experiments for Dilated ResNet are too compute-expensive due to their long runtimes, see Table 1. For training, we used a dataset with higher temporal resolution of $\Delta t = 0.375$ s and get equal number of trajectories from uniformly sampling 832 different external buoyancy force values, $\boldsymbol{f} = (0, f)^T$ in Equation 5, in the range $0.2 \leq f \leq 0.5$, using input fields at one timestep. We conditioned our models to predict for different time windows in the range $0.375\,\text{s} \leq \Delta t \leq 20$ s, and different strengths of the y-component of the external buoyancy force f. Due to the unbalanced nature of the dataset size at different Δt, we reweighed the sampling frequency in our dataloader to try to maintain parity. We provided conditioning information in the form of an embedding vector which can be added to each or subset of residual blocks (Ho et al., 2020). Both, Δt and f, are continuous valued scalar parameters, and thus can be encoded into a vector representation by using sinusoidal Fourier embeddings as is common in Transformers (Vaswani et al., 2017). We added the embedding vector to the feature maps after the first convolution/FNO layer in respective down- and up-sampling blocks. To be more precise, for each feature map we replicated the respective embedding value along x- and y-coordinates. For Fourier layers, this results in adding the embedding vector, the Fourier branch, and the residual connection together. We also compare an alternative conditioning approach for U-Nets termed AdaGN (Nichol & Dhariwal, 2021), based on affine transformation of group normalization layers via projections of our embeddings in Figure 19 of Appendix B.5.

Figure 5 shows results obtained by various models averaged over 208 different unseen evaluation force values. U-Net based methods perform best. In contrast to the unconditioned experiments, substituting lower blocks by FNO blocks didn't yield better generalization capabilities. In general, we observe that conditioning is more difficult for FNO layers, most strikingly seen in the performance curves of FNOs. We however do not discard the possibility that for FNO layers, alternative parame-

[6] https://github.com/tum-pbs/PhiFlow



(a) Buoyancy force= 0.22

(b) Buoyancy force= 0.48

Figure 7: Scalar field predictions obtained for the parameter conditioning experiments using the best performing U-Net$_{\text{mod}}$ model. Predicted and ground truth fields are shown for different buoyancy force values and different time windows. Model inputs are the same for different time window prediction tasks. More conditioning experiments can be found in Appendix B.5.

ter embedding and conditioning methods might be required. Nevertheless, our results also coincide with the findings of Lu et al. (2022), which state that FNO like architectures seemed to be extremely sensitive to noise, and failed to predict solutions for even small amounts of added Gaussian noise. In Figure 6, we show performance of different models tested on buoyancy force values in the range $0.1 \leq f \leq 0.6$. The curves indicate that the difficulty of the tasks increases for larger buoyancy force values, but U-Net based PDE surrogates show better interpolation and extrapolation abilities. In Figure 7, we display example scalar fields obtained for the parameter conditioning experiments of the Navier-Stokes equations using the best performing U-Net$_{\text{mod}}$ model. Predicted and ground truth fields are shown for different values of the absolute buoyancy force and different time-scale values.

5 Conclusion

We presented a comprehensive comparison between various ResNet, FNO, and U-Net based approaches on fluid mechanic problems, paving a basis towards strong baselines for the development of neural PDE surrogates. For U-Nets, we transferred recent architectural improvements from computer vision, most notably from object segmentation and generative modeling. We found that the original U-Net architecture of Ronneberger et al. (2015), with only an additional down- and upsample layer, already functions as a powerful neural PDE surrogate, and e.g. outperforms FNOs on the presented tasks. Combined with recent architectural improvements, we achieved further significant improvements in performance. Moreover, we were able to use the best performing U-Net architectures to generalize to different PDE parameters as well as different time-scales within a single surrogate model. FNO layers during early downsampling in U-Nets further improved performance under certain circumstances, although similar to the findings of Lu et al. (2022) FNO layers seem to have negative effects when generalizing to different time-scales and PDE parameters. Finally, we hope that our codebase can be a starting point for further investigations on neural PDE surrogates.

Limitations & Future Work. This work focuses on the “image-to-image” modeling aspect of PDE surrogate modeling, more precisely on the understanding of complex multi-scale spatio-temporal phenomena. That said, in this work we did not elaborate on important aspects of neural PDE surrogates such as stability over long rollouts, preservation of invariants, or generalization over sampling regularities, over domain topologies and geometries, and over boundary conditions. We see many of these aspects as future work. Moreover, in this work we focused on modeling Navier-Stokes equations directly, rather then in the Reynolds-averaged Navier–Stokes (RANS) form (Tennekes et al., 1972), which is very common when describing turbulent flows. Finally, future work could extend towards Vision Transformers (Dosovitskiy et al., 2020), comparing their abilities to model and generalize across spatio-temporal information.

Reproducibility and ethical statement

Reproducibility statement. We have included error bars, and ablation studies wherever we found it necessary and appropriate. We have described our architectures in Section 3 and provided further implementation details in Appendix B. We see it as crucial part of this work to move towards more accessible and comprehensive comparisons for neural PDE surrogates on interesting benchmark problems. We will therefore open-source our codebase in such a way that all our models can be run on all our datasets, but also other models and other datasets can easily be plugged in.

Ethical statement. The societal impact of neural PDE surrogates is difficult to predict. Neural PDE surrogates will however play an important role in modeling physical, mechanical, chemical and other phenomena, and therefore are potential shortcuts for computationally expensive simulations. And if used as such, PDE surrogates might potentially be directly or indirectly related to reducing the carbon footprint and advancing various research directions. On the downside, relying on simulations always requires rigorous cross-checks and monitoring, especially when simulations or simulated quantities are learned.

References

David J Acheson. Elementary fluid dynamics, 1991.

Troy Arcomano, Istvan Szunyogh, Jaideep Pathak, Alexander Wikner, Brian R Hunt, and Edward Ott. A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, 47(9):e2020GL087776, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bassam Bamieh. Discovering transforms: A tutorial on circulant matrices, circular convolution, and the discrete fourier transform. *arXiv preprint arXiv:1805.05533*, 2018.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K. Gupta. Clifford neural layers for PDE modeling. *arXiv preprint arXiv:2209.04934*, 2022a.

Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. *arXiv preprint arXiv:2202.07643*, 2022b.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022c.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Gengxiang Chen, Yingguang Li, Qinglu Meng, Jing Zhou, Xiaozhong Hao, et al. Residual fourier neural operator for thermochemical curing of composites. *arXiv preprint arXiv:2111.10262*, 2021.

Li-Wei Chen and Nils Thuerey. Towards high-accuracy deep learning inference of compressible turbulent flows over aerofoils. *arXiv preprint arXiv:2109.02183*, 2021.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Thomas Frerix, Dmitrii Kochkov, Jamie Smith, Daniel Cremers, Michael Brenner, and Stephan Hoyer. Variational data assimilation with a learned inverse observation operator. In *International Conference on Machine Learning*, pp. 3449–3458. PMLR, 2021.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pp. 267–285. Springer, 1982.

Victor Garcia Satorras, Zeynep Akata, and Max Welling. Combining generative and discriminative models for hybrid inference. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 13802–13812. Curran Associates, Inc., 2019.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 2415–2423, 2019.

Steven Guan, Ko-Tsung Hsu, and Parag V Chitnis. Fourier neural operator networks: A fast and general solver for the photoacoustic wave equation. *arXiv preprint arXiv:2108.09374*, 2021.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Etienne Guyon, Jean-Pierre Hulin, Luc Petit, Catalin D Mitescu, et al. *Physical hydrodynamics*. Oxford university press, 2001.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Philipp Holl, Vladlen Koltun, Kiwon Um, and Nils Thuerey. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *NeurIPS Workshop*, volume 2, 2020.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural PDE solvers with convergence guarantees. *arXiv preprint arXiv:1906.01200*, 2019.

Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4504–4512, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Ian T Jolliffe and David B Stephenson. *Forecast verification: a practitioner's guide in atmospheric science*. John Wiley & Sons, 2012.

Ryan Keisler. Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Milan Klöwer, Tom Kimpson, Alistair White, and Mosè Giordano. milankl/SpeedyWeather.jl: v0.2.1, July 2022. URL https://doi.org/10.5281/zenodo.6788067.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Pijush K Kundu, Ira M Cohen, and David R Dowling. *Fluid mechanics*. Academic press, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Markov neural operators for learning chaotic systems. *arXiv preprint arXiv:2106.06898*, 2021a.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021b.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022b.

Marten Lienen and Stephan Günnemann. Learning the dynamics of physical systems from sparse observations with finite element networks. *arXiv preprint arXiv:2203.08852*, 2022.

Burigede Liu, Nikola Kovachki, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, Andrew M Stuart, and Kaushik Bhattacharya. A learning-based multiscale method and its application to inelastic impact problems. *Journal of the Mechanics and Physics of Solids*, 158:104668, 2022.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Winfried Lötzsch, Simon Ohler, and Johannes S Otterbach. Learning the solution operator of boundary value problems using graph neural networks. *arXiv preprint arXiv:2206.14092*, 2022.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Hao Ma, Yuxuan Zhang, Nils Thuerey, Xiangyu Hu, and Oskar J Haidn. Physics-driven learning of the steady navier-stokes equations using deep convolutional neural networks. *arXiv preprint arXiv:2106.09301*, 2021.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Romit Maulik, Vishwas Rao, Jiali Wang, Gianmarco Mengaldo, Emil Constantinescu, Bethany Lusch, Prasanna Balaprakash, Ian Foster, and Rao Kotamarthi. Efficient high-dimensional variational data assimilation with machine-learned reduced-order models. *Geoscientific Model Development*, 15(8):3433–3445, 2022.

Andreas Mayr, Sebastian Lehner, Arno Mayrhofer, Christoph Kloss, Sepp Hochreiter, and Johannes Brandstetter. Boundary graph neural networks for 3d simulations. *arXiv preprint arXiv:2106.11299*, 2021.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Bruce Roy Munson, Theodore Hisao Okiishi, Wade W Huebsch, and Alric P Rothmayer. *Fluid mechanics*. Wiley Singapore, 2013.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

P.J. Olver. Symmetry groups of differential equations. In *Applications of Lie Groups to Differential Equations*, pp. 77–185. Springer, 1986.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4353–4361, 2017.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Silvia L Pintea, Nergis Tömen, Stanley F Goes, Marco Loog, and Jan C van Gemert. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, 30:8342–8353, 2021.

Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V Butz, and Wolfgang Nowak. Finite volume neural network: Modeling subsurface contaminant transport. *arXiv preprint arXiv:2104.06010*, 2021.

Md Ashiqur Rahman, Manuel A Florez, Anima Anandkumar, Zachary E Ross, and Kamyar Azizzadenesheli. Generative adversarial neural operators. *arXiv preprint arXiv:2205.03017*, 2022a.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022b.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in Neural Information Processing Systems*, 34, 2021.

Stephan Rasp and Nils Thuerey. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, 13(2):e2020MS002405, 2021.

David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. FlexConv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021a.

David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021b.

David W Romero, David M Knigge, Albert Gu, Erik J Bekkers, Efstratios Gavves, Jakub M Tomczak, and Mark Hoogendoorn. Towards a general purpose CNN for long range dependencies in ND. *arXiv preprint arXiv:2206.03398*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.

Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. SchNet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Wenlei Shi, Xinquan Huang, Xiaotian Gao, Xinran Wei, Jia Zhang, Jiang Bian, Mao Yang, and Tie-Yan Liu. Lordnet: Learning to solve parametric partial differential equations without simulated data. *arXiv preprint arXiv:2206.09418*, 2022.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140*, 2020.

Kimberly Stachenfeld, Drummond B Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*, 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022. URL https://doi.org/10.18419/darus-2986.

Roger Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

Hendrik Tennekes, John Leask Lumley, Jonh L Lumley, et al. *A first course in turbulence*. MIT press, 1972.

Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Cornelis Boudewijn Vreugdenhil. *Numerical methods for shallow-water flow*, volume 13. Springer Science & Business Media, 1994.

Nils Wandel, Michael Weinmann, and Reinhard Klein. Learning incompressible fluid dynamics from scratch–towards fast, differentiable fluid models that generalize. *arXiv preprint arXiv:2006.08762*, 2020.

Nils Wandel, Michael Weinmann, Michael Neidlin, and Reinhard Klein. Spline-PINN: Approaching PDEs without data using fast, physics-informed hermite-spline CNNs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8529–8538, 2022.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020a.

Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint arXiv:2002.03061*, 2020b.

Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2589–2597, 2018.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Jonathan A Weyn, Dale R Durran, and Rich Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020.

Jonathan A Weyn, Dale R Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002502, 2021.

Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. *arXiv preprint arXiv:2206.07681*, 2022.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Yan Yang, Angela F Gao, Jorge C Castellanos, Zachary E Ross, Kamyar Azizzadenesheli, and Robert W Clayton. Seismic wave propagation and inversion with neural operators. *The Seismic Record*, 1(3):126–134, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, et al. Neuralpde: Automating physics-informed neural networks (PINNs) with error approximations. *arXiv preprint arXiv:2107.09443*, 2021.

CONTENTS

1 Introduction **1**

2 Preliminaries **2**

3 PDE Surrogates **5**

3.1 Operator learning 5

4 Experiments **6**

5 Conclusion **9**

A Related work **18**

B Experiments **18**

B.1 Experimental details 18

B.2 Additional model details 19

B.2.1 ResNet 19

B.2.2 Dilated ResNet 19

B.2.3 FNO 19

B.2.4 U-Net 19

B.2.5 Parameter Conditioning 20

B.2.6 Spatial-spectral parameter conditioning for Fourier layers 20

B.3 Shallow water equations. 21

B.4 Navier-Stokes equations. 30

B.5 Parameter conditioning. 36

A Related work

Neural PDE modeling is appearing in many flavors. Various works are numerical-neural hybrid approaches where the computation graph of the solver is preserved and heuristically-chosen parameters are left for the neural network to predict (Bar-Sinai et al., 2019; Kochkov et al., 2021; Greenfeld et al., 2019; Hsieh et al., 2019; Praditia et al., 2021; Um et al., 2020; Garcia Satorras et al., 2019). The works of Sanchez-Gonzalez et al. (2020); Pfaff et al. (2020); Mayr et al. (2021) are of similar flavor where neural network predictions are input to the time-update of node positions in graphs and meshes. Fully neural network based approaches can be roughly split into two parts. First, methods that focus on the approximation of the solution function of the underlying PDE Sirignano & Spiliopoulos (2018); Han et al. (2018); Raissi et al. (2019); Jin et al. (2021); Raissi et al. (2020); Zubov et al. (2021). And second, methods that focus on the surrogate learning of solution operators. CNN-base models were among the first PDE surrogates (Guo et al., 2016; Bhatnagar et al., 2019; Zhu & Zabaras, 2018). Operator learning models were popularized via Fourier Neural Operators (Li et al., 2020a) and FNO-based applications and refinements (Li et al., 2021b; Rahman et al., 2022b; Rao et al., 2021; Guibas et al., 2021; Li et al., 2021a; Rahman et al., 2022a; Pathak et al., 2022; Wen et al., 2022; Liu et al., 2022; Yang et al., 2021; Guan et al., 2021; Hwang et al., 2022; Chen et al., 2021; Li et al., 2022b), as well as via DeepONets (Lu et al., 2019; 2021; 2022). Other directions include the modeling of PDE solution operators via latent space models, transformers, and graph neural networks (GNNs) (Wu et al., 2022; Li et al., 2022a; Brandstetter et al., 2022c; Lötzsch et al., 2022; Lienen & Günnemann, 2022). The ever persisting chicken-egg problem (Brandstetter et al., 2022b; Shi et al., 2022) of how to obtain high quality ground truth training data for neural PDE surrogates is approached either via clever data augmentation (Brandstetter et al., 2022b), via equivariant neural solvers (Wang et al., 2020b), or via "data-free" learning paradigms (Geneva & Zabaras, 2020; Wandel et al., 2020; 2022; Shi et al., 2022). Pratical applications of neural PDE surrogates can especially be found in weather forecasting (Pathak et al., 2022; Guibas et al., 2021; Keisler, 2022; Rasp & Thuerey, 2021; Weyn et al., 2020; 2021; Arcomano et al., 2020; Sønderby et al., 2020; Frerix et al., 2021; Maulik et al., 2022), and modeling of fluid dynamics (Ma et al., 2021; Stachenfeld et al., 2021; Wang et al., 2020a; Brandstetter et al., 2022a).

B Experiments

This appendix supports Section 4 of the main paper.

B.1 Experimental details

Loss functions and metrics. We report the summed MSE (SMSE) loss defined as:

$$\mathcal{L}_{\text{SMSE}} = \frac{1}{N_y} \sum_{y \in \mathbb{Z}^2} \sum_{j=1}^{N_t} \sum_{i=1}^{N_{\text{fields}}} \|\boldsymbol{u}_i(y, t_j) - \hat{\boldsymbol{u}}_i(y, t_j)\|_2^2 \,, \tag{8}$$

where $\boldsymbol{u}$ is the target, $\hat{\boldsymbol{u}}$ the model output, N_{fields} comprises scalar fields as well as individual vector field components, and N_y is the total number of spatial points. Equation 8 is used for training with $N_t = 1$, and further allows us to define our two main metrics:

- *One-step* loss where $N_t = 1$ and N_{fields} comprises all scalar and vector components.
- *Rollout* loss where $N_t = 5$ and N_{fields} comprises all scalar and vector components.

Alternatively, we train with the summed relative MSE (RMSE) loss as introduced in Li et al. (2020a):

$$\mathcal{L}_{\text{SMSE}} = \frac{1}{N_y} \sum_{y \in \mathbb{Z}^2} \sum_{j=1}^{N_t} \sum_{i=1}^{N_{\text{fields}}} \frac{\|\boldsymbol{u}_i(y, t_j) - \hat{\boldsymbol{u}}_i(y, t_j)\|_2^2}{\|\hat{\boldsymbol{u}}_i(y, t_j)\|_2^2} \,. \tag{9}$$

Training and model selection. We optimized models using the AdamW optimizer (Kingma & Ba, 2014; Loshchilov & Hutter, 2019) with the best learning rates of $[10^{-4}, 2 \cdot 10^{-4}]$ and weight decay of 10^{-5} for 50 epochs and minimized the summed mean squared error (SMSE) which is outlined in Equation 8. We used cosine annealing as learning rate scheduler (Loshchilov & Hutter, 2016) with a linear warmup. For baseline ResNet models, we optimized number of layers, number of channels, and normalization procedures. For the reported results we used group normalization (Wu & He, 2018) with 1 group which is equivalent to Layer norm (Ba et al., 2016) (except for final normalization layer in U-Nets where we use 8 groups). We further tested different activation functions. For baseline FNO models, we optimized number of layers, number of channels, and number of Fourier modes. Larger numbers of layers or channels did not improve the performances for both ResNet and FNO models. For U-Net like architectures, especially for U-Net$_{\text{att}}$, we specifically needed to optimize the maximum learning rate to be lower (10^{-4}). We further optimized for different number of hidden layers, and initialization and normalization schemes. For the reported results, we used pre-activations (He et al., 2016b) and layer normalization (Ba et al., 2016). We used an effective batch size of 32 for training.

Computational resources. All experiments used 4×16 GB NVIDIA V100 machines for training. Average training times varied between 2 h and 140 h, depending on task and number of trajectories. Parameter conditioning runs were the most expensive ones.

Runtime comparison. We warmup the benchmark for 10 iterations and report average runtimes over 100 runs on a single 16 GB NVIDIA V100 machine with input batch size of 8. Note that UNO was much slower on multi-GPU training in the cloud.

B.2 Additional model details

B.2.1 ResNet

We use two embedding and two output layers with kernel sizes of 1×1.

B.2.2 Dilated ResNet

The implemented Dilated ResNet models consist of 4 residual blocks where each block individually consists of 7 dilated CNN layers with dilation rates of $[1, 2, 4, 8, 4, 2, 1]$. We implement Dilated ResNets with and without group normalization layers applied to each layer in the respective dilation blocks.

B.2.3 FNO

We use FNOs consisting of $\{4, 8\}$ FNO layers, where $\{8, 16, 32\}$ modes are multiplied in the Fourier space, and $\{64, 128\}$ channels are used. We use two embedding and two output layers with kernel sizes of 1×1 as suggested in Li et al. (2020a). The number of non-zero Fourier modes, the number of FNO layers and the number of channels are hyperparameter, where we report results for different values in each of the experiment. We use GeLU activation functions, and no normalization scheme. Normalization schemes and residual connections did not improve performance, as already reported in Brandstetter et al. (2022a).

B.2.4 U-Net

We use one embedding and one output layers with kernel sizes of 3×3. To allow a fair comparison to FNO (and ResNet) architectures, we ablated architectures also for kernel sizes of 1×1 for embedding and output layers.

U-Net$_{2015}$. We use channel multipliers of $(2, 2, 2, 2)$. The network consists overall of 4 downsampling, one bottleneck and 4 upsampling layers. We further use batch normalization (Ioffe & Szegedy, 2015), and no bias weights. The implementation is based on the `PDEbench` repository of Takamoto et al. (2022). Compared to the implementation of Takamoto et al. (2022), we use GeLU activations instead of tanh activations since we observe significant better performances. For the sake of completeness, we also report tanh results terming the models U-Net$_{\text{2015-tanh}}$.

U-Net$_{\text{base}}$. We use channel multipliers of $(2, 2, 2, 2)$. We replaced batch normalization (Ioffe & Szegedy, 2015) with group normalization (Wu & He, 2018) with number of groups equal 1 to be consistent with other architectures. Additionally, compared to the U-Net$_{2015}$ version, we use bias weights but no bottleneck layer.

U-Net$_{\text{mod}}$. We use channel multipliers of $(1, 2, 2, 4)$, and residual connections in each down- and upsampling block. We use pre-normalization and pre-activations (He et al., 2016b). Additionally, we zero-initialize the second `Conv` layer in each residual block.

U-Net$_{\text{mod,attn}}$. Adding attention to all down- and upsampling blocks made training unstable, and would have required an extensive hyperparameter search. We therefore only use attention in the middle blocks after downsampling. We further only use a single attention along with a residual connection bypassing attention.

B.2.5 Parameter Conditioning

Embedding. We use sinusoidal embedding as proposed in Vaswani et al. (2017) for positional encoding of scalar values, such as prediction time window and force strength:

$$\text{Emb}(x, d) = \left[\cos\frac{x}{10000^{2x/d_i}}, \sin\frac{x}{10000^{2x/d_i}}\right] \text{ for } 0 \leq d_i < d\,, \tag{10}$$

where x is the embedded quantity and d is the output embedding dimension.

Projection. We use a two-layer feed-forward network to project each of the embeddings to higher dimension ($4\times$ hidden channels), and add them together before passing them to each block via another linear layer.

Conditioning. We explore two different mechanisms for conditioning, originally proposed in the image modeling literature. Simple "Addition" as proposed by Ho et al. (2020) which can easily be extended to FNO layers and "AdaGN" as proposed by Nichol & Dhariwal (2021) which requires normalization layers to be applicable and was therefore restricted to U-Net based architectures in our experiments.

- **Addition:** A single `Linear` layer is used to scale the dimensions appropriately to match the dimensions of the conditioned block. Conditions are added to the first `Conv` layer's output, followed by normalization and a second `Conv` layer. This conditioning is applied to all blocks in the network.
- **AdaGN.** Instead of directly adding the `Linear` projection of the embedding to the respective blocks, the projection y is split into $[y_s, y_b]$ to scale and shift the normalized output h before passing to the second `Conv` layer in each block, similar to as is done in FiLM (Perez et al., 2018):
$$h' = y_s \odot \text{GroupNorm}(h) + y_b\,, \tag{11}$$
where $\odot$ denotes the pointwise product. The conditioning is applied to all blocks. Since FNO architectures were implemented without group norm, AdaGN was not applicable to those.

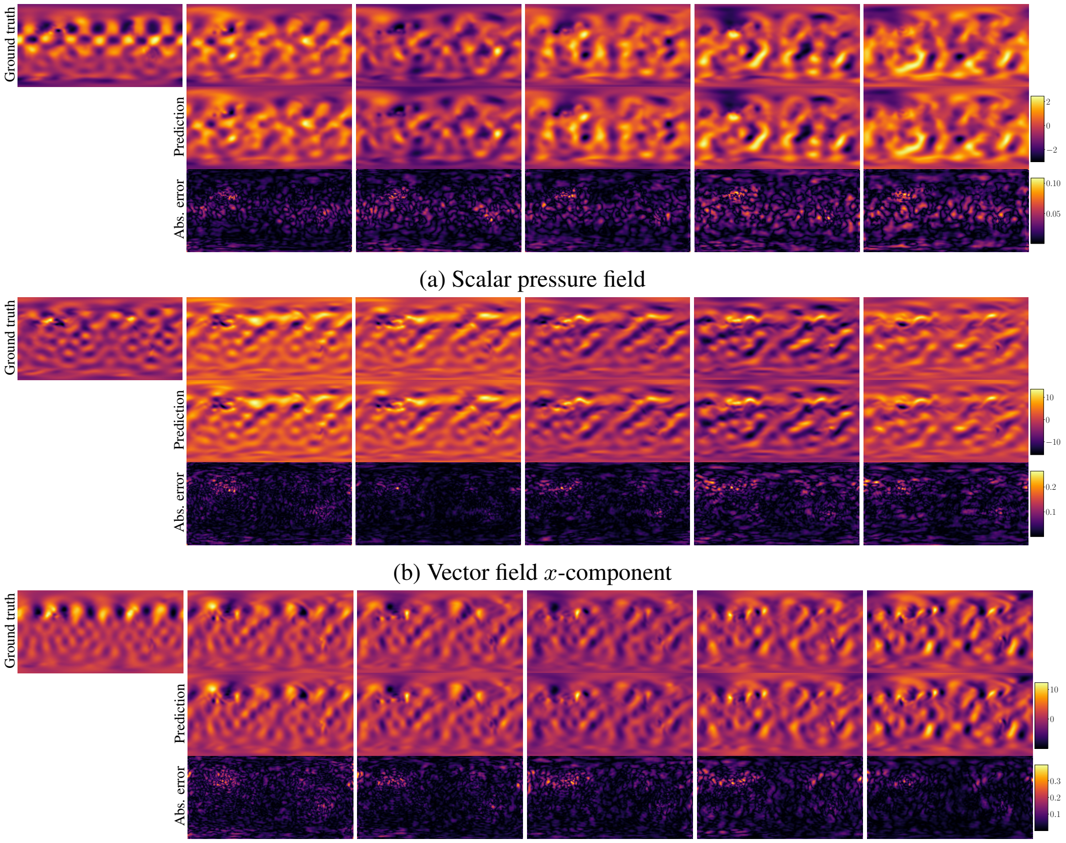
In Appendix B.5, we ablate "Addition" and "AdaGN" embeddings for U-Net architectures.

B.2.6 Spatial-spectral parameter conditioning for Fourier layers

Since FNO like architectures are usually implemented without normalization schemes, only "Addition" is applicable as conditioning strategy. Adding the conditioning at the end of each FNO layer, i.e. in the spatial domain, omits that the conditioning information is accessible in the Fourier domain too. This was somewhat unsatisfying, so we explored a straightforward mechanism to apply conditioning to the Fourier branch as well. We implement and alternative `FreqLinear` layer to project the embeddings into Fourier space too. Each Fourier mode is first multiplied with the embedding, then weights are mode-wise multiplied and the inverse Fourier transform is performed. Adding conditioning both in the Fourier and the spatial domain seems to work best. We term this alternative embedding "Spatial-Spectral" embedding. In Appendix B.5, we ablate "Addition" and "Spatial-Spectral" embeddings for FNO like architectures including UFNets with FNO blocks in the downsampling path.

B.3 Shallow water equations.

The shallow water equations are solved on a regular grid with periodic boundary conditions as described in Section 4 of the main paper. The inputs to the shallow water experiments are respective fields at the previous 2 timesteps. Pressure and vorticity fields are normalized for training. Example rollout trajectories are displayed in Figure 8 for the velocity function formulation and in Figure 9 for the vorticity stream function formulation. We outline further details on the results on the shallow water experiments in Figures 10,11, and Tables 2,3. Additionally, we show results for 1-day predictions in Figure 12 and Table 7. We further ablate different encoding/decoding choices for U-Net like architectures in Figures 13,14. Finally, we compare different the specs of different FNO, UNO, and U-FNet architectures in Table 4.

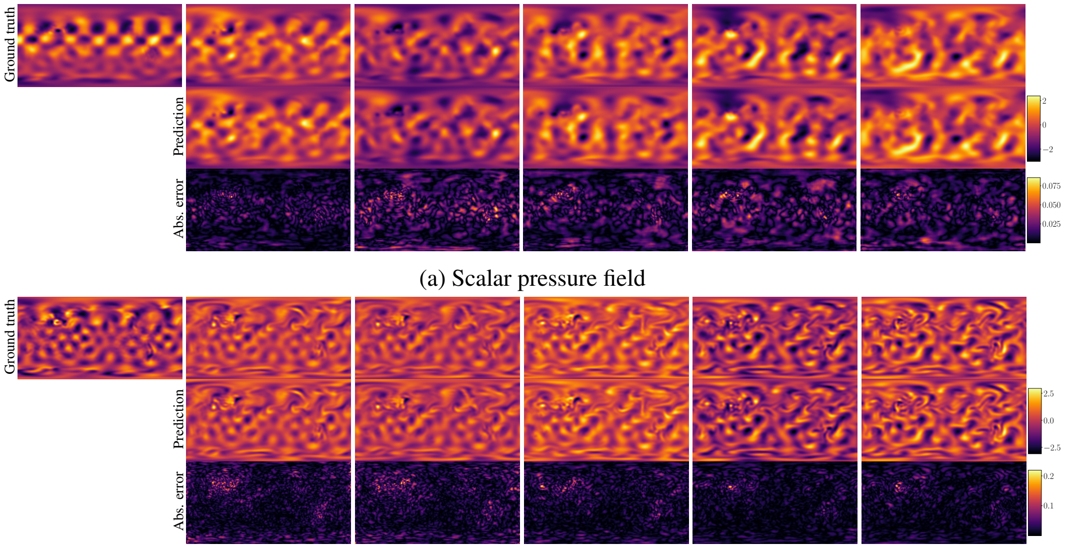


(a) Scalar pressure field

(b) Vector field x-component

(c) Vector field y-component

Figure 8: Shallow water 2-day predictions, velocity function form. Example rollouts of the scalar pressure and the vector wind field of the shallow water experiments are shown, obtained by a U-F1Net$_{\text{modes16}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained for a time window $\Delta t = 48\,\text{h}$. The respective model input fields comprise two timesteps, we only show the first of those (left-most ground truth column).



(a) Scalar pressure field

(b) Scalar vorticity field

Figure 9: Shallow water 2-day predictions, vorticity stream function form. Example rollouts of the scalar pressure and the scalar vorticity field of the shallow water experiments are shown, obtained by a U-F2Net$_{\text{modes16,16}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained for a time window $\Delta t = 48\,\text{h}$. The respective model input fields comprise two timesteps, we only show the first of those (left-most ground truth column).

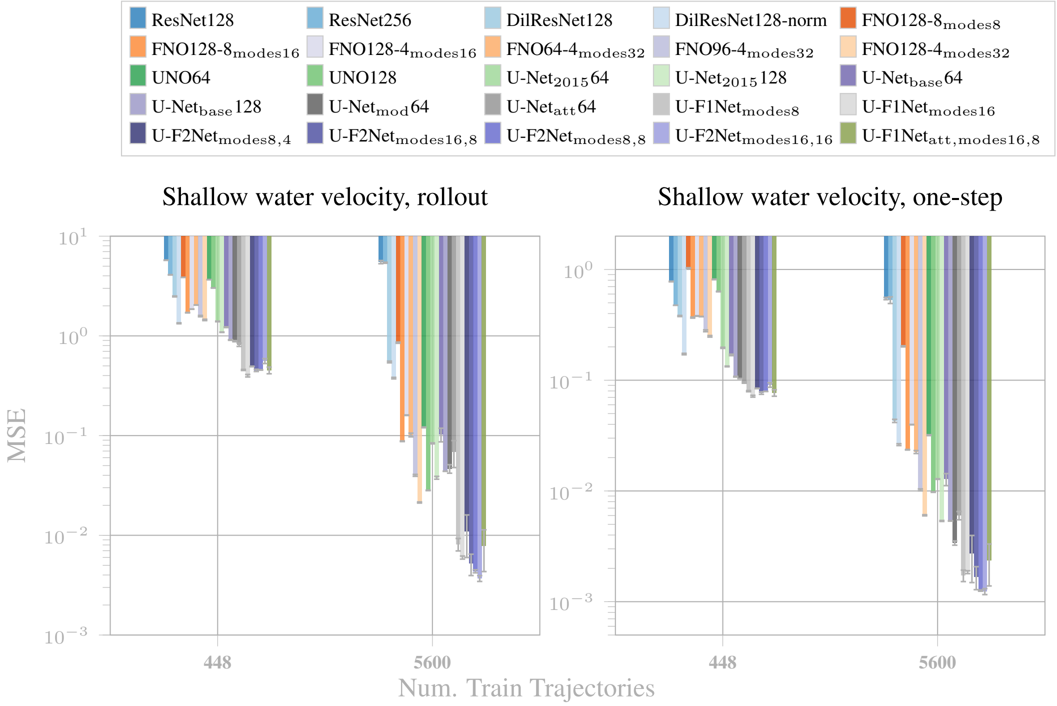


Figure 10: Shallow water 2-day predictions. Rollout and one-step errors of various architectures on the shallow water equations are reported. Results are obtained for predictions of 2-day time windows for the velocity function formulation and are averaged over three different random seeds. Note the logarithmic scale of the y-axes.

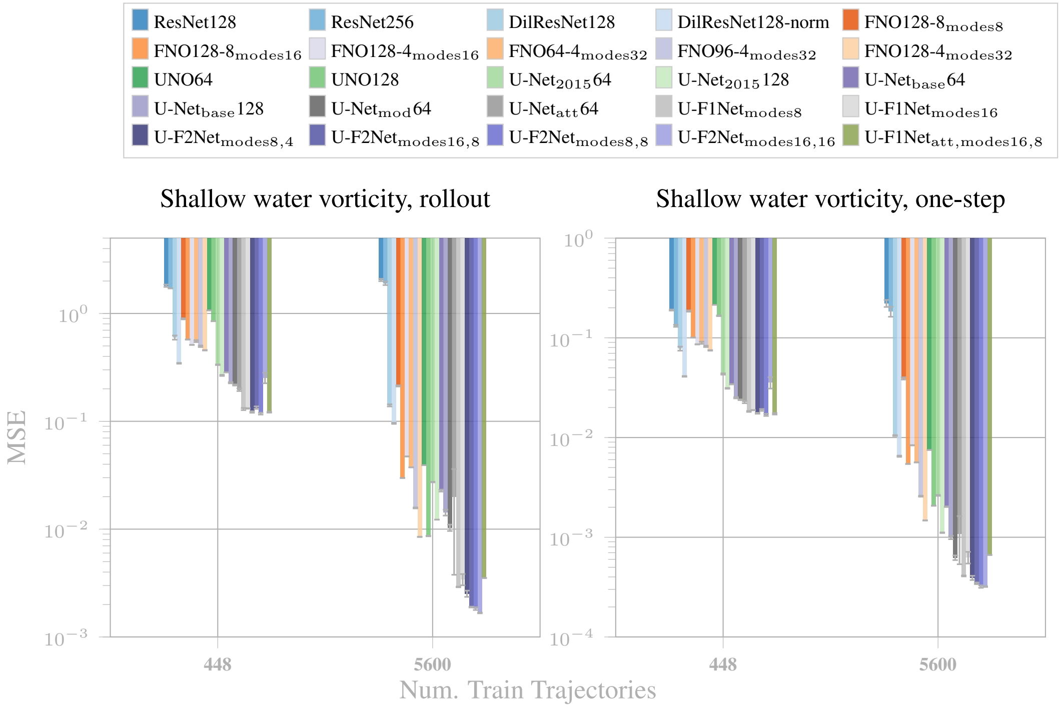


Figure 11: Shallow water 2-day predictions. Rollout and one-step errors of various architectures on the shallow water equations. Results are obtained for predictions of 2-day time windows for the vorticity stream function formulation and are averaged over three different random seeds. Note the logarithmic scale of the y-axes.

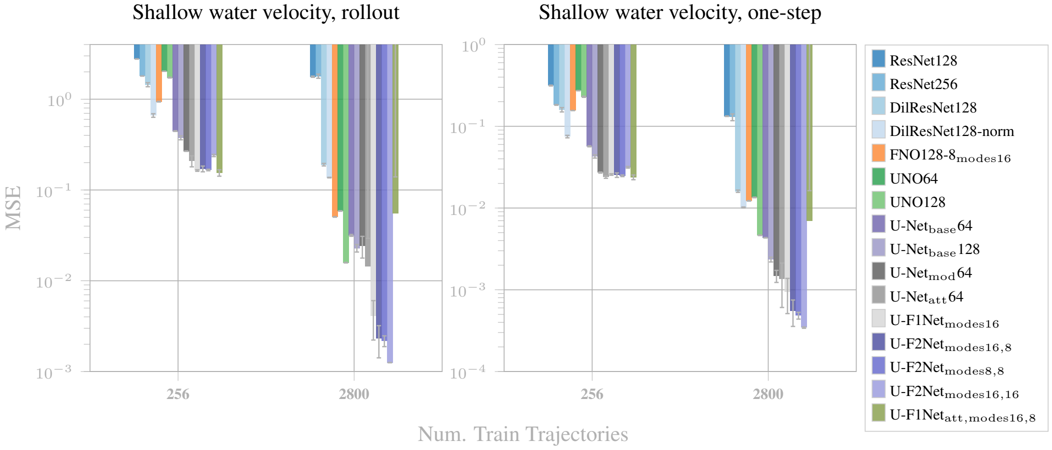


Figure 12: Shallow water 1-day predictions. Rollout and one-step errors of various architectures on the shallow water equations are reported. Results are obtained for predictions of 1-day time windows for the velocity function formulation and are averaged over three different random seeds. Note the logarithmic scale of the y-axes.

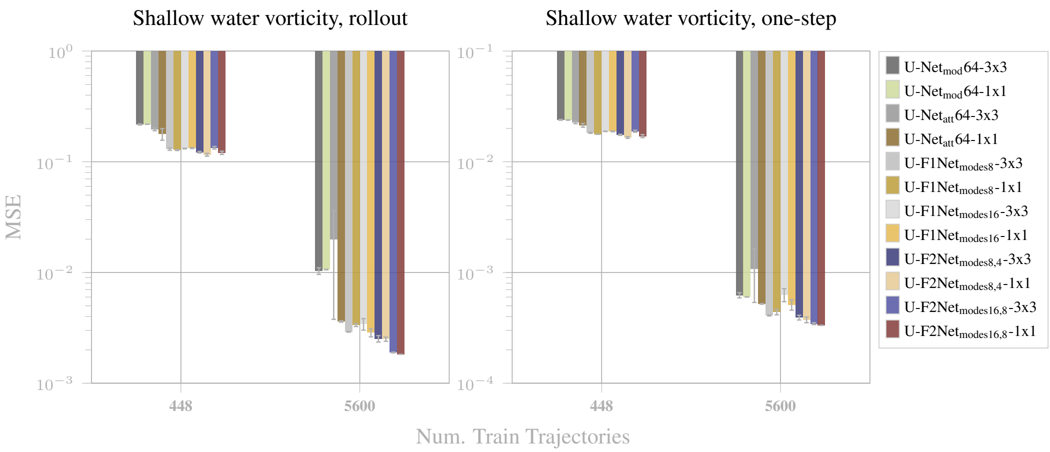


Figure 13: Shallow water 2-day predictions. Ablation results of different encoding and decoding choices for various U-Net architectures are reported. 1×1 and 3×3 kernels are compared for both encoding and decoding. Rollout and one-step errors are obtained on the shallow water equations for 2-day predictions for the vorticity stream function formulation and are averaged over three different random seeds.

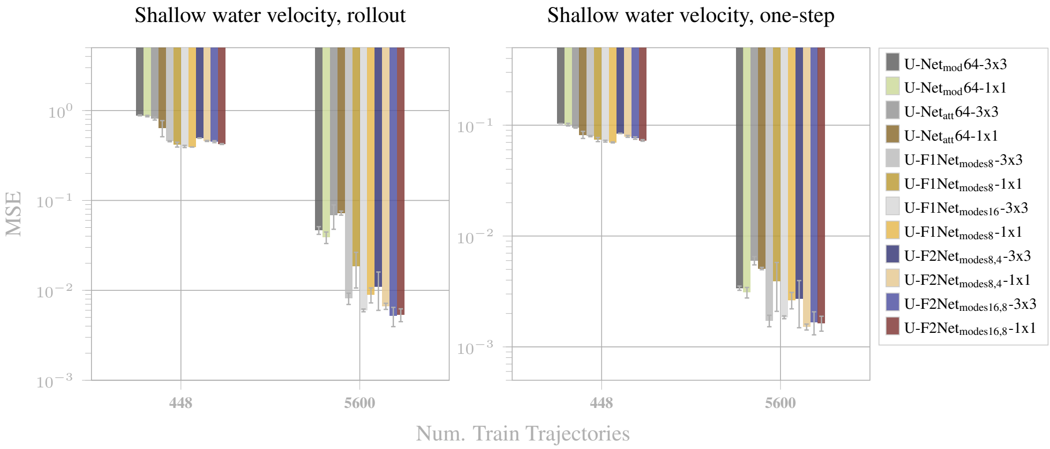


Figure 14: Shallow water 2-day predictions. Ablation results of different encoding and decoding choices for various U-Net architectures are reported. 1×1 and 3×3 kernels are compared for both encoding and decoding. Rollout and one-step errors are obtained on the shallow water equations for 2-day predictions for the velocity function formulation and are averaged over three different random seeds.

Table 2: Shallow water 2-day predictions, velocity function formulation. Rollout and one-step errors of various architectures on the shallow water equations are reported. Summed mean-squared errors (SMSE) are obtained for 2-day predictions for the velocity function formulation and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the four digit precision minimum. The best model of each model class is highlighted.

		SMSE	
METHOD	Trajs.	onestep	rollout
ResNet128	448	0.7787 ± 0.0049	5.7408 ± 0.0223
ResNet128	5600	0.5465 ± 0.0130	5.4946 ± 0.2009
ResNet256	448	0.4751 ± 0.0005	4.0954 ± 0.0181
ResNet256	5600	0.5294 ± 0.0364	5.4155 ± 0.0657
DilResNet128	448	0.3800 ± 0.0042	2.4860 ± 0.0260
DilResNet128	5600	0.0429 ± 0.0014	0.5476 ± 0.0109
DilResNet128-norm	448	0.1723 ± 0.0026	1.3377 ± 0.0144
DilResNet128-norm	5600	0.0262 ± 0.0006	0.3770 ± 0.0081
FNO128-8$_{\text{modes8}}$	448	1.0322 ± 0.0055	3.8635 ± 0.0090
FNO128-8$_{\text{modes8}}$	5600	0.2023 ± 0.0023	0.8549 ± 0.0124
FNO128-8$_{\text{modes16}}$	448	0.3681 ± 0.0042	1.7088 ± 0.0096
FNO128-8$_{\text{modes16}}$	5600	0.0236 ± 0.0001	0.0878 ± 0.0007
FNO128-4$_{\text{modes16}}$	448	0.3802 ± 0.0021	1.8542 ± 0.0056
FNO128-4$_{\text{modes16}}$	5600	0.0397 ± 0.0002	0.1601 ± 0.0010
FNO64-4$_{\text{modes32}}$	448	0.3750 ± 0.0012	2.0393 ± 0.0050
FNO64-4$_{\text{modes32}}$	5600	0.0225 ± 0.0007	0.1015 ± 0.0044
FNO96-4$_{\text{modes32}}$	448	0.2794 ± 0.0053	1.5788 ± 0.0253
FNO96-4$_{\text{modes32}}$	5600	0.0102 ± 0.0002	0.0399 ± 0.0008
FNO128-4$_{\text{modes32}}$	448	0.2492 ± 0.0040	1.4460 ± 0.0226
FNO128-4$_{\text{modes32}}$	5600	0.0060 ± 0.0001	0.0213 ± 0.0003
UNO64	448	0.8134 ± 0.0048	3.6621 ± 0.0103
UNO64	5600	0.0319 ± 0.0003	0.1208 ± 0.0010
UNO128	448	0.6328 ± 0.0041	3.0240 ± 0.0064
UNO128	5600	0.0098 ± 0.0001	0.0282 ± 0.0002
U-Net$_{\text{base}}$64	448	0.1693 ± 0.0026	1.2224 ± 0.0108
U-Net$_{\text{base}}$64	5600	0.0128 ± 0.0016	0.1026 ± 0.0161
U-Net$_{\text{base}}$128	448	0.1076 ± 0.0008	0.9096 ± 0.0067
U-Net$_{\text{base}}$128	5600	0.0054	0.0439 ± 0.0001
U-Net$_{\text{mod}}$64	448	0.1034 ± 0.0001	0.8847 ± 0.0109
U-Net$_{\text{mod}}$64	5600	0.0034 ± 0.0001	0.0465 ± 0.0045
U-Net$_{\text{mod}}$64-1x1	448	0.1013 ± 0.0028	0.8681 ± 0.0161
U-Net$_{\text{mod}}$64-1x1	5600	0.0031 ± 0.0003	0.0389 ± 0.0057
U-Net$_{\text{att}}$64	448	0.0954 ± 0.0014	0.8158 ± 0.0318
U-Net$_{\text{att}}$64	5600	0.0060 ± 0.0005	0.0684 ± 0.0206
U-Net$_{\text{att}}$64-1x1	448	0.0819 ± 0.0060	0.6419 ± 0.1307
U-Net$_{\text{att}}$64-1x1	5600	0.0051 ± 0.0001	0.0724 ± 0.0037
U-F1Net$_{\text{modes8}}$	448	0.0797 ± 0.0010	0.4553 ± 0.0057
U-F1Net$_{\text{modes8}}$	5600	0.0017 ± 0.0002	0.0081 ± 0.0012
U-F1Net$_{\text{modes16}}$	448	0.0717 ± 0.0013	0.3988 ± 0.0115
U-F1Net$_{\text{modes16}}$	5600	0.0018 ± 0.0001	0.0060 ± 0.0002
U-F1Net$_{\text{modes8}}$-1x1	448	0.0743 ± 0.0032	0.4185 ± 0.0259
U-F1Net$_{\text{modes8}}$-1x1	5600	0.0039 ± 0.0018	0.0185 ± 0.0079
U-F1Net$_{\text{modes16}}$-1x1	448	0.0699 ± 0.0007	0.3923 ± 0.0001
U-F1Net$_{\text{modes16}}$-1x1	5600	0.0027 ± 0.0004	0.0089 ± 0.0017
U-F2Net$_{\text{modes8,4}}$	448	0.0843 ± 0.0004	0.4934 ± 0.0031
U-F2Net$_{\text{modes8,4}}$	5600	0.0027 ± 0.0012	0.0110 ± 0.0050
U-F2Net$_{\text{modes16,8}}$	448	0.0765 ± 0.0019	0.4508 ± 0.0116
U-F2Net$_{\text{modes16,8}}$	5600	0.0017 ± 0.0004	0.0052 ± 0.0013
U-F2Net$_{\text{modes8,8}}$	448	0.0793 ± 0.0001	0.4553 ± 0.0021
U-F2Net$_{\text{modes8,8}}$	5600	0.0013	0.0044 ± 0.0002
U-F2Net$_{\text{modes16,16}}$	448	0.0906 ± 0.0042	0.5596 ± 0.0317
U-F2Net$_{\text{modes16,16}}$	5600	0.0012 ± 0.0001	0.0037 ± 0.0002
U-F2Net$_{\text{modes8,4}}$-1x1	448	0.0793 ± 0.0016	0.4587 ± 0.0064
U-F2Net$_{\text{modes8,4}}$-1x1	5600	0.0015 ± 0.0001	0.0067 ± 0.0005
U-F2Net$_{\text{modes16,8}}$-1x1	448	0.0728 ± 0.0012	0.4238 ± 0.0049
U-F2Net$_{\text{modes16,8}}$-1x1	5600	0.0016 ± 0.0003	0.0054 ± 0.0009
U-F2Net$_{\text{att,modes16,8}}$	448	0.0769 ± 0.0052	0.4517 ± 0.0348
U-F2Net$_{\text{att,modes16,8}}$	5600	0.0024 ± 0.0010	0.0078 ± 0.0035
U-F3Net$_{\text{modes8,4,2}}$	448	0.0920 ± 0.0011	0.5318 ± 0.0083
U-F3Net$_{\text{modes8,4,2}}$	5600	0.0018	0.0062 ± 0.0001
U-F3Net$_{\text{modes16,8,4}}$	448	0.0812	0.4823 ± 0.0021
U-F3Net$_{\text{modes16,8,4}}$	5600	0.0016	0.0048

Table 3: Shallow water 2-day predictions, vorticity stream function formulation. Rollout and one-step errors of various architectures on the shallow water equations are reported. Summed mean-squared errors (SMSE) are obtained for 2-day predictions for the vorticity stream function formulation and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the five digit precision minimum. The best model of each model class is highlighted.

		SMSE	
METHOD	Trajs.	onestep	rollout
ResNet128	448	$0.189\,93 \pm 0.002\,32$	$1.809\,30 \pm 0.046\,36$
ResNet128	5600	$0.221\,82 \pm 0.017\,93$	$2.045\,72 \pm 0.050\,28$
ResNet256	448	$0.132\,08 \pm 0.003\,25$	$1.710\,41 \pm 0.001\,42$
ResNet256	5600	$0.183\,80 \pm 0.021\,08$	$1.906\,15 \pm 0.072\,68$
DilatedResNet128	448	$0.077\,97 \pm 0.003\,60$	$0.598\,31 \pm 0.026\,61$
DilatedResNet128	5600	$0.010\,47 \pm 0.000\,11$	$0.140\,51 \pm 0.003\,20$
DilatedResNet128-norm	448	$0.041\,05 \pm 0.000\,35$	$0.345\,23 \pm 0.003\,03$
DilatedResNet128-norm	5600	$0.006\,49 \pm 0.000\,09$	$0.095\,71 \pm 0.001\,10$
FNO128-8$_{\text{modes8}}$	448	$0.185\,53 \pm 0.002\,78$	$0.892\,44 \pm 0.011\,09$
FNO128-8$_{\text{modes8}}$	5600	$0.039\,17 \pm 0.001\,01$	$0.213\,04 \pm 0.002\,81$
FNO128-8$_{\text{modes16}}$	448	$0.100\,54 \pm 0.000\,26$	$0.574\,04 \pm 0.002\,40$
FNO128-8$_{\text{modes16}}$	5600	$0.005\,44 \pm 0.000\,01$	$0.029\,81 \pm 0.000\,05$
FNO128-4$_{\text{modes16}}$	448	$0.086\,01 \pm 0.000\,28$	$0.511\,27 \pm 0.000\,41$
FNO128-4$_{\text{modes16}}$	5600	$0.008\,36 \pm 0.000\,03$	$0.047\,23 \pm 0.000\,05$
FNO64-4$_{\text{modes32}}$	448	$0.089\,23 \pm 0.002\,33$	$0.553\,83 \pm 0.011\,40$
FNO64-4$_{\text{modes32}}$	5600	$0.005\,64 \pm 0.000\,02$	$0.037\,44$
FNO96-4$_{\text{modes32}}$	448	$0.082\,08 \pm 0.001\,32$	$0.496\,44 \pm 0.008\,51$
FNO96-4$_{\text{modes32}}$	5600	$0.002\,58 \pm 0.000\,03$	$0.015\,73 \pm 0.000\,16$
FNO128-4$_{\text{modes32}}$	448	$0.074\,83 \pm 0.000\,49$	$0.455\,85 \pm 0.002\,24$
FNO128-4$_{\text{modes32}}$	5600	$0.001\,47$	$0.008\,47 \pm 0.000\,03$
UNO64	448	$0.214\,37$	$1.068\,14$
UNO64	5600	$0.007\,51 \pm 0.000\,01$	$0.039\,18 \pm 0.000\,02$
UNO128	448	$0.166\,32 \pm 0.001\,40$	$0.848\,57 \pm 0.001\,90$
UNO128	5600	$0.002\,07$	$0.008\,58$
U-Net$_{\text{base}}$64	448	$0.034\,22 \pm 0.000\,14$	$0.284\,32 \pm 0.000\,16$
U-Net$_{\text{base}}$64	5600	$0.002\,03 \pm 0.000\,02$	$0.022\,77 \pm 0.000\,47$
U-Net$_{\text{base}}$128	448	$0.025\,05 \pm 0.000\,35$	$0.227\,93 \pm 0.002\,97$
U-Net$_{\text{base}}$128	5600	$0.000\,99 \pm 0.000\,04$	$0.014\,25 \pm 0.000\,89$
U-Net$_{\text{mod}}$64	448	$0.023\,93 \pm 0.000\,20$	$0.217\,13 \pm 0.002\,54$
U-Net$_{\text{mod}}$64	5600	$0.000\,62 \pm 0.000\,03$	$0.010\,31 \pm 0.000\,70$
U-Net$_{\text{mod}}$64-1x1	448	$0.023\,83 \pm 0.000\,11$	$0.218\,11 \pm 0.000\,45$
U-Net$_{\text{mod}}$64-1x1	5600	$0.000\,60$	$0.010\,63 \pm 0.000\,06$
U-Net$_{\text{att}}$64	448	$0.022\,69 \pm 0.000\,60$	$0.196\,43 \pm 0.005\,66$
U-Net$_{\text{att}}$64	5600	$0.001\,08 \pm 0.000\,55$	$0.019\,94 \pm 0.016\,17$
U-Net$_{\text{att}}$64-1x1	448	$0.021\,33 \pm 0.000\,83$	$0.177\,91 \pm 0.021\,44$
U-Net$_{\text{att}}$64-1x1	5600	$0.000\,52$	$0.003\,61 \pm 0.000\,05$
U-F1Net$_{\text{modes8}}$	448	$0.018\,27 \pm 0.000\,20$	$0.129\,95 \pm 0.002\,97$
U-F1Net$_{\text{modes8}}$	5600	$0.000\,41$	$0.002\,91 \pm 0.000\,02$
U-F1Net$_{\text{modes16}}$	448	$0.018\,82 \pm 0.000\,07$	$0.131\,56 \pm 0.000\,74$
U-F1Net$_{\text{modes16}}$	5600	$0.000\,63 \pm 0.000\,08$	$0.003\,42 \pm 0.000\,41$
U-F1Net$_{\text{modes8}}$-1x1	448	$0.017\,74 \pm 0.000\,04$	$0.127\,51 \pm 0.001\,20$
U-F1Net$_{\text{modes8}}$-1x1	5600	$0.000\,44 \pm 0.000\,02$	$0.003\,36 \pm 0.000\,11$
U-F1Net$_{\text{modes16}}$-1x1	448	$0.018\,82 \pm 0.000\,14$	$0.132\,87 \pm 0.001\,27$
U-F1Net$_{\text{modes16}}$-1x1	5600	$0.000\,51 \pm 0.000\,05$	$0.002\,86 \pm 0.000\,24$
U-F2Net$_{\text{modes8,4}}$	448	$0.017\,51 \pm 0.000\,19$	$0.122\,03 \pm 0.001\,99$
U-F2Net$_{\text{modes8,4}}$	5600	$0.000\,39 \pm 0.000\,02$	$0.002\,52 \pm 0.000\,17$
U-F2Net$_{\text{modes8,4}}$-1x1	448	$0.016\,56 \pm 0.000\,30$	$0.115\,67 \pm 0.003\,36$
U-F2Net$_{\text{modes8,4}}$-1x1	5600	$0.000\,37 \pm 0.000\,02$	$0.002\,50 \pm 0.000\,11$
U-F2Net$_{\text{modes16,8}}$	448	$0.018\,90 \pm 0.000\,42$	$0.133\,86 \pm 0.003\,63$
U-F2Net$_{\text{modes16,8}}$	5600	$0.000\,35 \pm 0.000\,01$	$0.001\,89 \pm 0.000\,02$
U-F2Net$_{\text{modes16,8}}$-1x1	448	$0.017\,05 \pm 0.000\,52$	$0.120\,52 \pm 0.004\,04$
U-F2Net$_{\text{modes16,8}}$-1x1	5600	$0.000\,33$	$0.001\,83$
U-F2Net$_{\text{modes8,8}}$	448	$0.016\,89 \pm 0.000\,43$	$0.117\,81 \pm 0.002\,35$
U-F2Net$_{\text{modes8,8}}$	5600	$0.000\,32 \pm 0.000\,01$	$0.001\,83 \pm 0.000\,05$
U-F2Net$_{\text{modes16,16}}$	448	$0.035\,66 \pm 0.004\,67$	$0.254\,00 \pm 0.028\,69$
U-F2Net$_{\text{modes16,16}}$	5600	$0.000\,32$	$0.001\,67 \pm 0.000\,01$
U-F2Net$_{\text{att,modes16,8}}$	448	$0.017\,31 \pm 0.000\,31$	$0.121\,50 \pm 0.001\,19$
U-F2Net$_{\text{att,modes16,8}}$	5600	$0.000\,66$	$0.003\,52$
U-F3Net$_{\text{modes8,4,2}}$	448	$0.020\,58 \pm 0.000\,13$	$0.147\,38 \pm 0.001\,31$
U-F3Net$_{\text{modes8,4,2}}$	5600	$0.000\,40$	$0.002\,37 \pm 0.000\,01$
U-F3Net$_{\text{modes16,8,4}}$	448	$0.022\,03 \pm 0.000\,26$	$0.158\,55 \pm 0.000\,55$
U-F3Net$_{\text{modes16,8,4}}$	5600	$0.000\,38 \pm 0.000\,01$	$0.002\,08 \pm 0.000\,05$

Table 4: Comparison of parameter count, runtime, and memory requirement of various FNO, UNO, and U-FNet architectures. Subscript numbers indicate the used number of Fourier modes. For U-FNet experiments subscript numbers indicate the number of Fourier modes in the lowest, second-lowest, and third lowest block.

METHOD	Channels	Res.Layers/Blocks	Params.	Runtime [s]		Mem. [MB]	
				Fwd.	Fwd.+bwd.	`f32` size	Peak usage
FNO128-8$_{\text{modes8}}$	128	8	33.7 M	0.057	0.162	134	2161
FNO128-8$_{\text{modes16}}$	128	8	134 M	0.059	0.171	537	2953
FNO128-4$_{\text{modes16}}$	128	4	67.2 M	0.031	0.089	268	1852
FNO64-4$_{\text{modes32}}$	64	4	67.1 M	0.016	0.050	268	1204
FNO96-4$_{\text{modes32}}$	96	4	151 M	0.026	0.080	604	2179
FNO128-4$_{\text{modes32}}$	128	4	268 M	0.036	0.118	1100	3420
UNO64	64	7	110 M	0.070	0.134	440	1925
UNO128	128	7	440 M	0.160	0.341	1800	5513
U-F1Net$_{\text{modes8}}$	64	9	154 M	0.083	0.205	617	3936
U-F1Net$_{\text{modes16}}$	64	9	185 M	0.084	0.208	743	4037
U-F2Net$_{\text{modes8,4}}$	64	9	163 M	0.085	0.213	652	3961
U-F2Net$_{\text{modes8,8}}$	64	9	193 M	0.085	0.216	772	4046
U-F2Net$_{\text{modes16,8}}$	64	9	224 M	0.086	0.219	897	4149
U-F2Net$_{\text{modes16,16}}$	64	9	344 M	0.090	0.232	1400	4496
U-F3Net$_{\text{modes8,4,2}}$	64	9	198 M	0.086	0.221	658	4332
U-F3Net$_{\text{modes16,8,4}}$	64	9	259 M	0.088	0.226	752	4808

Table 5: Shallow water 2-day predictions, velocity function formulation. Rollout and one-step errors of various architectures on the shallow water equations are reported. L2 training objective of Li et al. (2020a) is used. Summed mean-squared errors (SMSE) are obtained for 2-day predictions for the vorticity stream function formulation and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the five digit precision minimum. The best model of each model class is highlighted.

METHOD	Trajs.	SMSE	
		onestep	rollout
DilatedResNet128	448	0.312 43 ± 0.004 90	2.115 97 ± 0.036 31
DilatedResNet128	5600	0.042 30 ± 0.000 75	0.542 99 ± 0.012 92
DilatedResNet128-norm	448	0.148 49 ± 0.002 38	1.197 04 ± 0.008 19
DilatedResNet128-norm	5600	0.026 07 ± 0.000 73	0.382 68 ± 0.007 60
FNO128-4$_{\text{modes16}}$	448	0.339 75 ± 0.000 37	1.673 92 ± 0.003 99
FNO128-4$_{\text{modes16}}$	5600	0.040 21 ± 0.000 49	0.163 17 ± 0.002 58
FNO64-4$_{\text{modes32}}$	448	0.302 18 ± 0.002 98	1.678 19 ± 0.015 23
FNO64-4$_{\text{modes32}}$	5600	0.023 08 ± 0.001 02	0.107 56 ± 0.006 50
FNO96-4$_{\text{modes32}}$	448	0.234 39 ± 0.005 60	1.351 64 ± 0.025 51
FNO96-4$_{\text{modes32}}$	5600	0.010 16 ± 0.000 31	0.040 49 ± 0.001 50
U-Net$_{\text{base}}$64	448	0.147 26 ± 0.002 30	1.115 39 ± 0.005 14
U-Net$_{\text{base}}$64	5600	0.009 88 ± 0.000 11	0.067 11 ± 0.001 57
U-Net$_{\text{base}}$128	448	0.095 07 ± 0.000 22	0.844 98 ± 0.002 95
U-Net$_{\text{base}}$128	5600	0.004 35 ± 0.000 08	0.032 08 ± 0.000 44
U-Net$_{\text{2015}}$64	448	0.169 45 ± 0.001 13	1.264 75 ± 0.003 96
U-Net$_{\text{2015}}$64	5600	0.012 79 ± 0.000 09	0.084 20 ± 0.001 29
U-Net$_{\text{2015}}$128	448	0.114 96 ± 0.000 23	0.985 53 ± 0.001 19
U-Net$_{\text{2015}}$128	5600	0.005 41 ± 0.000 06	0.039 10 ± 0.001 39
U-Net$_{\text{2015-tanh}}$64	448	0.475 99 ± 0.002 45	2.760 14 ± 0.004 88
U-Net$_{\text{2015-tanh}}$64	5600	0.029 97 ± 0.001 37	0.190 94 ± 0.006 18
U-Net$_{\text{2015-tanh}}$128	448	0.333 82 ± 0.004 30	2.187 05 ± 0.025 79
U-Net$_{\text{2015-tanh}}$128	5600	0.015 05 ± 0.000 32	0.097 31 ± 0.001 29
U-Net$_{\text{mod}}$64	448	0.088 51 ± 0.001 42	0.808 42 ± 0.011 27
U-Net$_{\text{mod}}$64	5600	0.002 25 ± 0.000 07	0.021 95 ± 0.001 11
U-F2Net$_{\text{modes16,8}}$	448	0.061 98 ± 0.000 23	0.371 33 ± 0.001 92
U-F2Net$_{\text{modes16,8}}$	5600	0.001 23 ± 0.000 02	0.003 76 ± 0.000 04

Table 6: Shallow water 2-day predictions, vorticity stream function formulation. Rollout and one-step errors of various architectures on the shallow water equations are reported. L2 training objective of Li et al. (2020a) is used. Summed mean-squared errors (SMSE) are obtained for 2-day predictions for the vorticity stream function formulation and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the five digit precision minimum. The best model of each model class is highlighted.

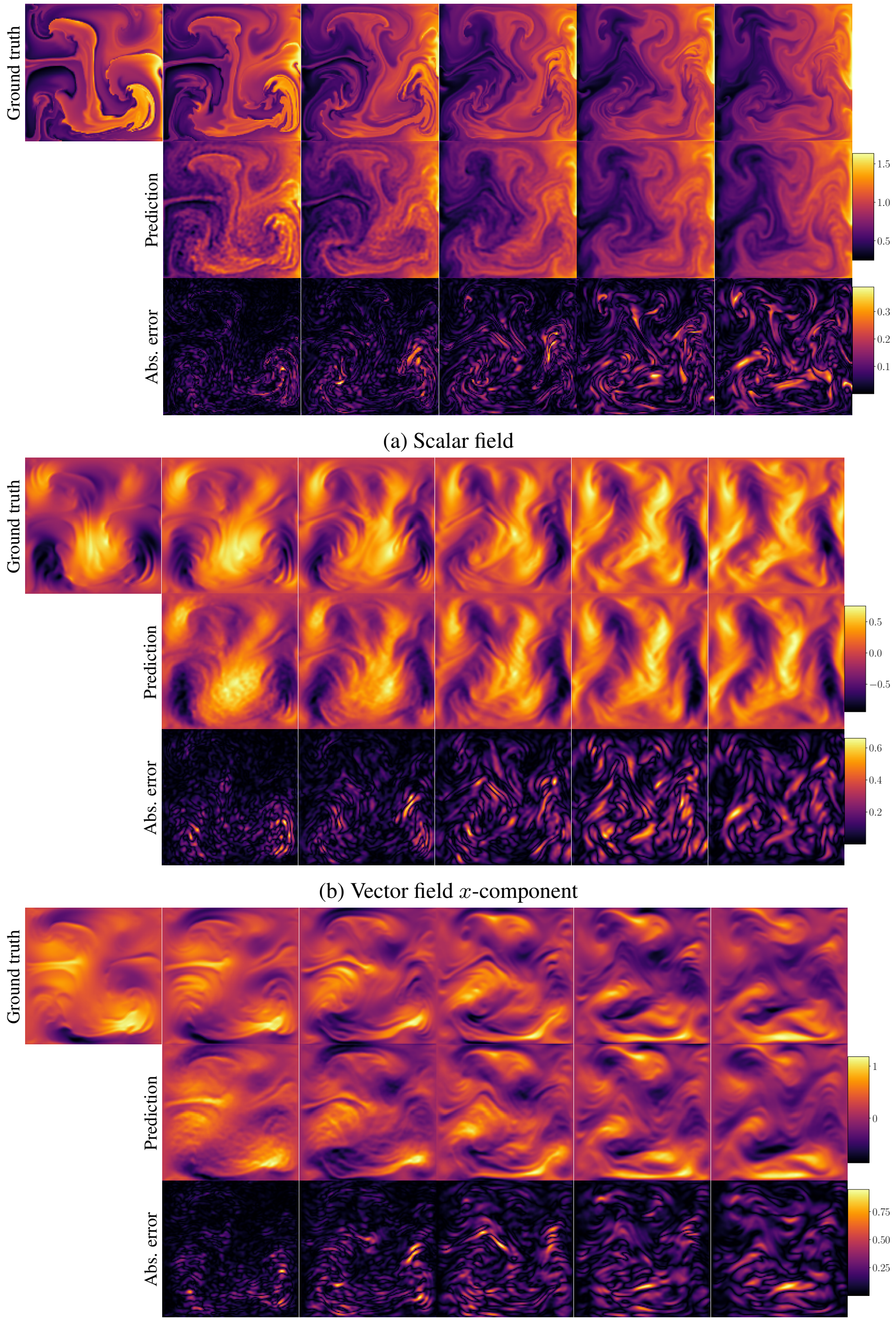
		SMSE	
METHOD	Trajs.	onestep	rollout
DilatedResNet128	448	0.069 14 ± 0.004 30	0.541 08 ± 0.034 62
DilatedResNet128	5600	0.010 49 ± 0.000 19	0.141 19 ± 0.005 32
DilatedResNet128-norm	448	0.036 62 ± 0.000 21	0.314 18 ± 0.002 97
DilatedResNet128-norm	5600	0.006 59 ± 0.000 11	0.095 90 ± 0.000 95
FNO128-8$_{\text{modes8}}$	448	0.178 17 ± 0.004 62	0.864 18 ± 0.019 15
FNO128-8$_{\text{modes8}}$	5600	0.039 73 ± 0.000 99	0.213 97 ± 0.002 21
FNO128-8$_{\text{modes16}}$	448	0.093 02 ± 0.003 25	0.543 09 ± 0.016 52
FNO128-8$_{\text{modes16}}$	5600	0.005 10 ± 0.000 02	0.027 47 ± 0.000 04
FNO64-4$_{\text{modes32}}$	448	0.080 56 ± 0.001 65	0.505 15 ± 0.008 60
FNO64-4$_{\text{modes32}}$	5600	0.005 63 ± 0.000 01	0.037 19 ± 0.000 09
FNO96-4$_{\text{modes32}}$	448	0.074 53 ± 0.001 33	0.458 38 ± 0.007 61
FNO96-4$_{\text{modes32}}$	5600	0.002 48 ± 0.000 04	0.014 81 ± 0.000 23
FNO128-4$_{\text{modes32}}$	448	0.067 41 ± 0.000 71	0.418 39 ± 0.003 05
FNO128-4$_{\text{modes32}}$	5600	0.001 33 ± 0.000 01	0.007 42 ± 0.000 01
UNO64	448	0.192 35 ± 0.000 10	0.973 45 ± 0.001 86
UNO64	5600	0.007 29 ± 0.000 01	0.037 24 ± 0.000 10
U-Net$_{\text{base}}$64	448	0.030 15 ± 0.000 12	0.258 04 ± 0.001 29
U-Net$_{\text{base}}$64	5600	0.001 99 ± 0.000 02	0.021 40 ± 0.000 17
U-Net$_{\text{base}}$128	448	0.021 85 ± 0.000 20	0.206 57 ± 0.001 45
U-Net$_{\text{base}}$128	5600	0.000 84 ± 0.000 02	0.010 85 ± 0.000 49
U-Net$_{\text{2015}}$64	448	0.038 41 ± 0.000 52	0.307 50 ± 0.003 27
U-Net$_{\text{2015}}$64	5600	0.002 66 ± 0.000 02	0.027 44 ± 0.000 14
U-Net$_{\text{2015}}$128	448	0.026 89 ± 0.000 08	0.239 41 ± 0.001 32
U-Net$_{\text{2015}}$128	5600	0.001 11 ± 0.000 01	0.012 36 ± 0.000 01
U-Net$_{\text{2015-tanh}}$64	448	0.086 29 ± 0.003 04	0.581 40 ± 0.010 45
U-Net$_{\text{2015-tanh}}$64	5600	0.005 49 ± 0.000 02	0.049 60 ± 0.000 40
U-Net$_{\text{2015-tanh}}$128	448	0.059 54 ± 0.000 36	0.449 91 ± 0.001 58
U-Net$_{\text{2015-tanh}}$128	5600	0.002 65	0.025 40 ± 0.000 23
U-Net$_{\text{mod}}$64	448	0.021 55 ± 0.000 44	0.201 22 ± 0.004 35
U-Net$_{\text{mod}}$64	5600	0.000 61 ± 0.000 01	0.010 22 ± 0.000 04
U-F2Net$_{\text{modes16,8}}$	448	0.017 78	0.128 92 ± 0.000 23
U-F2Net$_{\text{modes16,8}}$	5600	0.000 36 ± 0.000 01	0.001 92 ± 0.000 02

Table 7: Shallow water 1-day predictions, velocity function formulation. Rollout and one-step errors of various architectures on the shallow water equations are reported. Summed mean-squared errors (SMSE) are obtained for 1-day predictions for the velocity function formulation and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the four digit precision minimum. The best model of each model class is highlighted.

		SMSE	
METHOD	Trajs.	onestep	rollout
ResNet128	256	0.3152 ± 0.0039	2.7775 ± 0.0122
ResNet128	2800	0.1329 ± 0.0010	1.7711 ± 0.0273
ResNet256	256	0.1827 ± 0.0018	1.8015 ± 0.0037
ResNet256	2800	0.1294 ± 0.0122	1.7957 ± 0.0947
DilatedResNet-128	256	0.1596 ± 0.0097	1.4477 ± 0.0754
DilatedResNet-128	2800	0.0161 ± 0.0005	0.1900 ± 0.0061
DilatedResNet-128-norm	256	0.0749 ± 0.0026	0.6621 ± 0.0301
DilatedResNet-128-norm	2800	0.0102 ± 0.0002	0.1369 ± 0.0009
FNO128-8$_{\text{modes16}}$	256	0.1561	0.9394 ± 0.0081
FNO128-8$_{\text{modes16}}$	2800	0.0122 ± 0.0001	0.0505 ± 0.0004
UNO64	256	0.2732 ± 0.0009	2.0338 ± 0.0071
UNO64	2800	0.0135 ± 0.0001	0.0587 ± 0.0004
UNO128	256	0.2300 ± 0.0039	1.7308 ± 0.0173
UNO128	2800	0.0046	0.0158 ± 0.0001
U-Net$_{\text{base}}$64	256	0.0569 ± 0.0008	0.4457 ± 0.0038
U-Net$_{\text{base}}$64	2800	0.0043	0.0316 ± 0.0007
U-Net$_{\text{base}}$128	256	0.0428 ± 0.0020	0.3761 ± 0.0189
U-Net$_{\text{base}}$128	2800	0.0023 ± 0.0002	0.0229 ± 0.0022
U-Net$_{\text{mod}}$64	256	0.0272 ± 0.0004	0.2678 ± 0.0017
U-Net$_{\text{mod}}$64	2800	0.0015 ± 0.0002	0.0243 ± 0.0065
U-Net$_{\text{mod}}$64	256	0.0239 ± 0.0008	0.2089 ± 0.0291
U-Net$_{\text{mod}}$64	2800	0.0014 ± 0.0008	0.0144 ± 0.0149
U-F1Net$_{\text{modes16}}$	256	0.0257 ± 0.0005	0.1638 ± 0.0032
U-F1Net$_{\text{modes8}}$	2800	0.0009 ± 0.0004	0.0041 ± 0.0019
U-F2Net$_{\text{modes16,8}}$	256	0.0253 ± 0.0016	0.1710 ± 0.0124
U-F2Net$_{\text{modes16,8}}$	2800	0.0006 ± 0.0002	0.0023 ± 0.0009
U-F2Net$_{\text{modes8,8}}$	256	0.0248 ± 0.0006	0.1659 ± 0.0031
U-F2Net$_{\text{modes8,8}}$	2800	0.0005	0.0022 ± 0.0003
U-F2Net$_{\text{modes16,16}}$	256	0.0315 ± 0.0007	0.2398 ± 0.0068
U-F2Net$_{\text{modes16,16}}$	2800	0.0003	0.0013
U-F2Net$_{\text{att,modes16,8}}$	256	0.0238 ± 0.0016	0.1545 ± 0.0126
U-F2Net$_{\text{att,modes16,8}}$	2800	0.0070 ± 0.0093	0.0553 ± 0.0841

B.4 Navier-Stokes equations.

2D Navier-Stokes data is obtained on a grid with spatial resolution of 128×128 ($\Delta x = 0.25$, $\Delta y = 0.25$), and temporal resolution of $\Delta t = 1.5\,\mathrm{s}$, a viscosity parameter of $\nu = 0.01$, and a buoyancy factor of $(0, 0.5)^T$. The equation is solved on a closed domain with Dirichlet boundary conditions ($v = 0$) for the velocity, and Neumann boundaries $\frac{\partial s}{\partial x} = 0$ for the scalar field. We run the simulation for $21\,\mathrm{s}$ and sample every $1.5\,\mathrm{s}$. Trajectories contain scalar and vector fields at 14 different time points. The inputs to the Navier-Stokes experiments are respective fields at the previous 4 timesteps. Exemplary rollout trajectories are displayed in Figure 16. We outline further details on the results in Figure 17 and Table 8. Additionally, we ablate different encoding/decoding choices for U-Net based architectures in Figures 18.

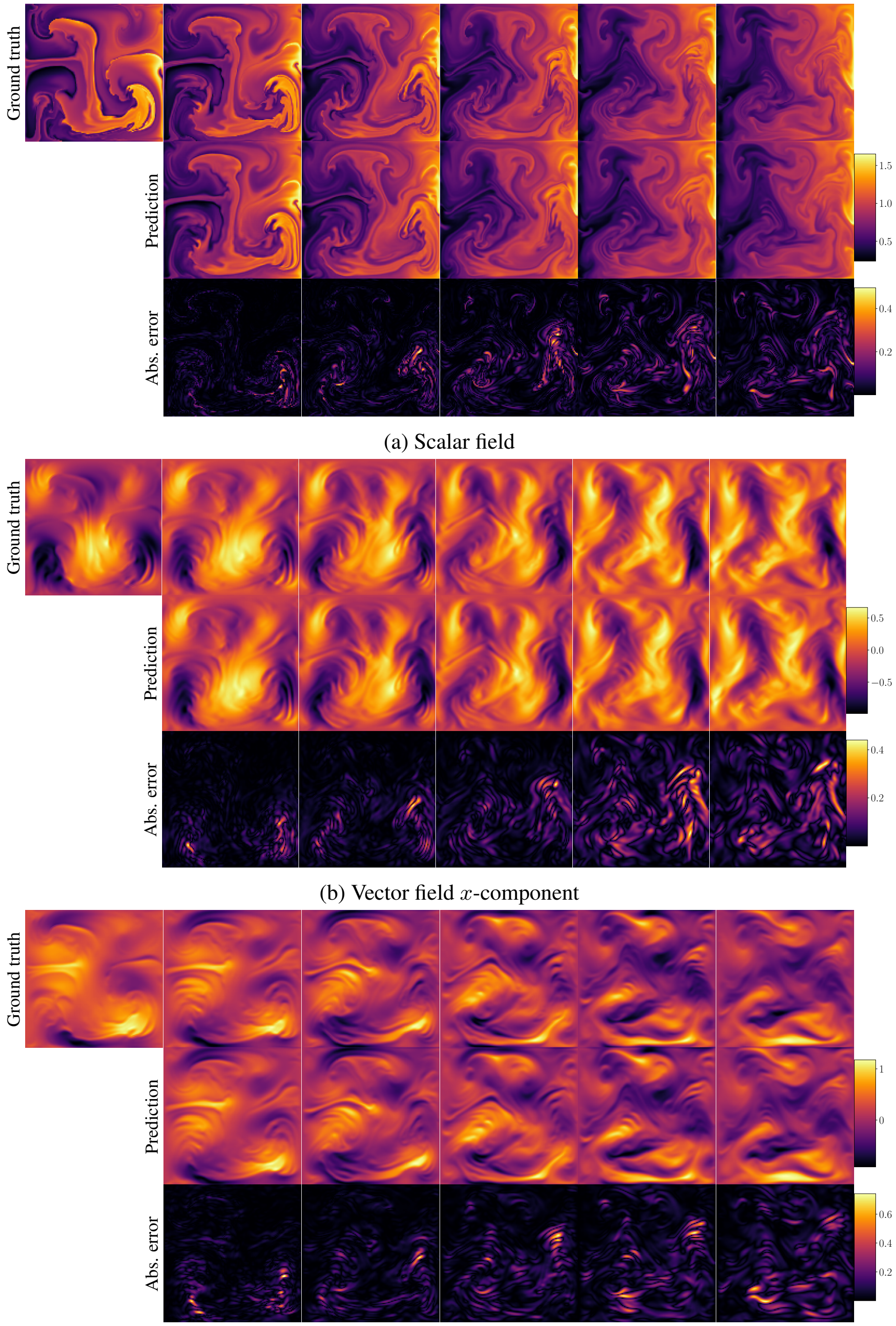


(a) Scalar field

(b) Vector field x-component

(c) Vector field y-component

Figure 15: Navier-Stokes, velocity function form. Example rollouts of the scalar and vector velocity field of the Navier-Stokes experiments are shown, obtained by a FNO96-4$_{\text{modes32,32}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained for a time window $\Delta t = 1.5\,\text{s}$. The respective model input fields comprise four timesteps, we only show the last of those (left-most ground truth column).



(a) Scalar field

(b) Vector field x-component

(c) Vector field y-component

Figure 16: Navier-Stokes, velocity function form. Example rollouts of the scalar and vector velocity field of the Navier-Stokes experiments are shown, obtained by a U-F1Net$_{\text{modes16}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained for a time window $\Delta t = 1.5\,\text{s}$. The respective model input fields comprise four timesteps, we only show the last of those (left-most ground truth column).

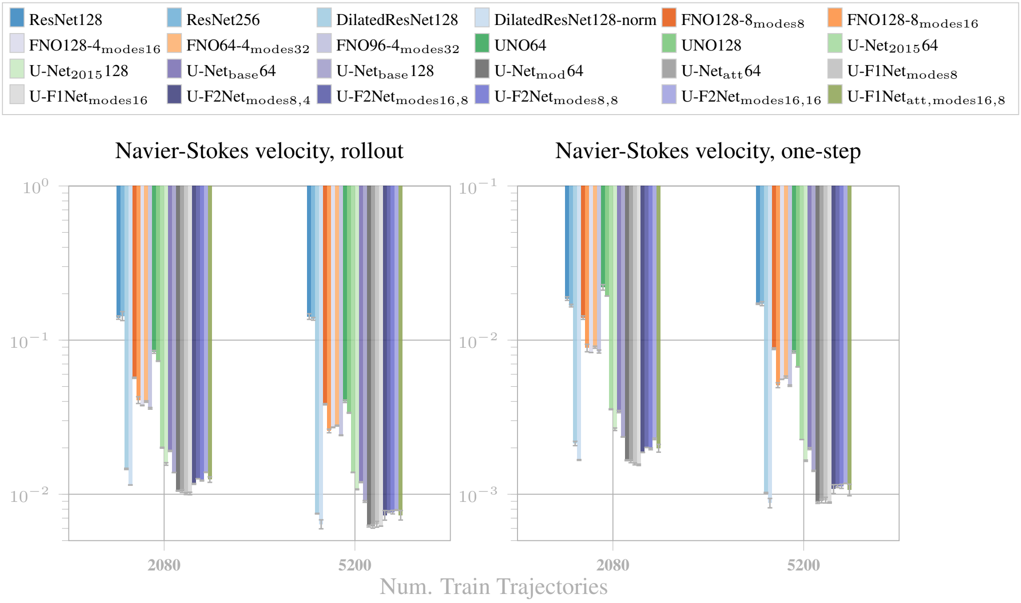


Figure 17: Navier-Stokes velocity function form. Rollout and one-step errors of various architectures on the Navier-Stokes equations are reported, obtained for predictions of 1.5 s, and are averaged over three different random seeds. Note the logarithmic scale of the y-axes.

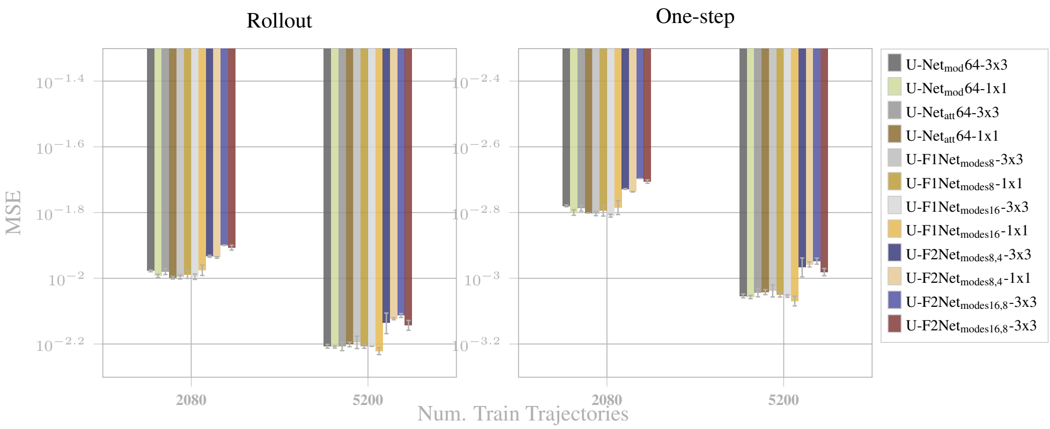


Figure 18: Navier-Stokes velocity function form. Ablation results of different encoding and decoding choices for various U-Net architectures are reported. 1×1 and 3×3 kernels are compared for both encoding and decoding. Rollout and one-step errors are obtained on the Navier-Stokes equations in velocity function form and are averaged over three different random seeds.

Table 8: Navier-Stokes, velocity function formulation. Rollout and one-step errors of various architectures on the Navier-Stokes equations are reported. Summed mean-squared errors (SMSE) are obtained and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the five digit precision minimum. The best model of each model class is highlighted.

		SMSE	
METHOD	Trajs.	onestep	rollout
ResNet128	2080	0.018 60 ± 0.000 49	0.140 02 ± 0.003 44
ResNet128	5200	0.017 22 ± 0.000 17	0.142 30 ± 0.005 98
ResNet256	2080	0.016 75 ± 0.000 32	0.143 44 ± 0.009 61
ResNet256	5200	0.017 25 ± 0.000 52	0.137 47 ± 0.002 84
DilResNet-128	2080	0.002 14 ± 0.000 07	0.014 60 ± 0.000 14
DilResNet-128	5200	0.001 02 ± 0.000 01	0.007 48 ± 0.000 04
DilResNet-128-norm	2080	0.001 67 ± 0.000 01	0.011 48 ± 0.000 01
DilResNet-128-norm	5200	0.000 88 ± 0.000 06	0.006 39 ± 0.000 43
FNO128-8$_{\text{modes8}}$	2080	0.014 00 ± 0.000 36	0.056 96 ± 0.000 51
FNO128-8$_{\text{modes8}}$	5200	0.008 79 ± 0.000 12	0.038 36 ± 0.000 37
FNO128-8$_{\text{modes16}}$	2080	0.008 90 ± 0.000 50	0.040 85 ± 0.002 05
FNO128-8$_{\text{modes16}}$	5200	0.005 10 ± 0.000 19	0.025 76 ± 0.000 71
FNO128-4$_{\text{modes16}}$	2080	0.008 31	0.037 69 ± 0.000 17
FNO128-4$_{\text{modes16}}$	5200	0.005 57	0.027 17 ± 0.000 13
FNO64-4$_{\text{modes32}}$	2080	0.008 99 ± 0.000 14	0.039 94 ± 0.000 45
FNO64-4$_{\text{modes32}}$	5200	0.005 76 ± 0.000 09	0.027 87 ± 0.000 19
FNO96-4$_{\text{modes32}}$	2080	0.008 43 ± 0.000 19	0.036 11 ± 0.000 44
FNO96-4$_{\text{modes32}}$	5200	0.005 07 ± 0.000 06	0.024 14 ± 0.000 13
UNO64	2080	0.022 00 ± 0.000 91	0.083 91 ± 0.001 79
UNO64	5200	0.008 37 ± 0.000 14	0.040 10 ± 0.000 74
UNO128	2080	0.019 33 ± 0.000 03	0.073 14 ± 0.000 35
UNO128	5200	0.006 71 ± 0.000 02	0.033 75 ± 0.000 21
U-Net$_{\text{2015-tanh}}$64	2080	0.006 51 ± 0.000 01	0.033 27 ± 0.000 01
U-Net$_{\text{2015-tanh}}$64	5200	0.003 59 ± 0.000 01	0.020 51 ± 0.000 07
U-Net$_{\text{2015-tanh}}$128	2080	0.004 56 ± 0.000 07	0.025 27 ± 0.000 43
U-Net$_{\text{2015-tanh}}$128	5200	0.002 75	0.016 71 ± 0.000 09
U-Net$_{\text{2015}}$64	2080	0.003 56 ± 0.000 02	0.020 04 ± 0.000 13
U-Net$_{\text{2015}}$64	5200	0.002 26	0.013 86 ± 0.000 04
U-Net$_{\text{2015}}$128	2080	0.002 64 ± 0.000 05	0.015 72 ± 0.000 30
U-Net$_{\text{2015}}$128	5200	0.001 65 ± 0.000 02	0.010 76 ± 0.000 07
U-Net$_{\text{base}}$64	2080	0.003 44 ± 0.000 06	0.019 10 ± 0.000 17
U-Net$_{\text{base}}$64	5200	0.001 97 ± 0.000 03	0.011 97 ± 0.000 08
U-Net$_{\text{base}}$128	2080	0.002 35 ± 0.000 01	0.013 83 ± 0.000 03
U-Net$_{\text{base}}$128	5200	0.001 42 ± 0.000 01	0.008 98 ± 0.000 13
U-Net$_{\text{mod}}$64	2080	0.001 66 ± 0.000 01	0.010 53 ± 0.000 06
U-Net$_{\text{mod}}$64	5200	0.000 88 ± 0.000 01	0.006 21 ± 0.000 08
U-Net$_{\text{att}}$64	2080	0.001 63 ± 0.000 03	0.010 48 ± 0.000 22
U-Net$_{\text{att}}$64	5200	0.000 90 ± 0.000 03	0.006 20 ± 0.000 18
U-F1Net$_{\text{modes8}}$	2080	0.001 57 ± 0.000 02	0.010 10 ± 0.000 15
U-F1Net$_{\text{modes8}}$	5200	0.000 92 ± 0.000 04	0.006 39 ± 0.000 28
U-F1Net$_{\text{modes16}}$	2080	0.001 55 ± 0.000 02	0.010 12 ± 0.000 19
U-F1Net$_{\text{modes16}}$	5200	0.000 88 ± 0.000 01	0.006 21 ± 0.000 01
U-F2Net$_{\text{modes8,4}}$	2080	0.001 87 ± 0.000 01	0.011 67 ± 0.000 08
U-F2Net$_{\text{modes8,4}}$	5200	0.001 08 ± 0.000 07	0.007 32 ± 0.000 53
U-F2Net$_{\text{modes16,8}}$	2080	0.002 01	0.012 61 ± 0.000 03
U-F2Net$_{\text{modes16,8}}$	5200	0.001 13 ± 0.000 02	0.007 71 ± 0.000 11
U-F2Net$_{\text{modes8,8}}$	2080	0.001 97 ± 0.000 03	0.012 27 ± 0.000 08
U-F2Net$_{\text{modes8,8}}$	5200	0.001 13 ± 0.000 03	0.007 66 ± 0.000 20
U-F2Net$_{\text{modes16,16}}$	2080	0.002 29 ± 0.000 03	0.013 88 ± 0.000 04
U-F2Net$_{\text{modes16,16}}$	5200	0.001 18	0.007 90 ± 0.000 04
U-F2Net$_{\text{att,modes16,8}}$	2080	0.001 99 ± 0.000 12	0.012 45 ± 0.000 51
U-F2Net$_{\text{att,modes16,8}}$	5200	0.001 07 ± 0.000 09	0.007 32 ± 0.000 52

Table 9: Navier-Stokes, velocity function formulation. L2 training objective of Li et al. (2020a) is used. Rollout and one-step errors of various architectures on the Navier-Stokes equations are reported. Summed mean-squared errors (SMSE) are obtained and are averaged over three different random seeds. If results are displayed without standard deviation, the obtained standard deviation is lower than the five digit precision minimum. The best model of each model class is highlighted.

		SMSE	
METHOD	Trajs.	onestep	rollout
DilResNet128	2080	0.002 06 ± 0.000 01	0.013 96 ± 0.000 05
DilResNet128	5200	0.001 02 ± 0.000 03	0.007 49 ± 0.000 12
DilResNet128-norm	2080	0.001 47 ± 0.000 01	0.010 11 ± 0.000 01
DilResNet128-norm	5200	0.000 82	0.006 04
FNO128-8$_{\text{modes8}}$	2080	0.012 49	0.051 87
FNO128-8$_{\text{modes8}}$	5200	0.008 05	0.035 91
FNO128-8$_{\text{modes16}}$	2080	0.008 23	0.038 65
FNO128-8$_{\text{modes16}}$	5200	0.004 84	0.024 85
FNO64-4$_{\text{modes32}}$	2080	0.007 81 ± 0.000 14	0.035 94 ± 0.000 47
FNO64-4$_{\text{modes32}}$	5200	0.005 17 ± 0.000 07	0.025 56 ± 0.000 13
FNO96-4$_{\text{modes32}}$	2080	0.007 45 ± 0.000 15	0.032 42 ± 0.000 38
FNO96-4$_{\text{modes32}}$	5200	0.004 70 ± 0.000 06	0.022 27 ± 0.000 11
FNO128-4$_{\text{modes32}}$	2080	0.007 10 ± 0.000 04	0.030 80
FNO128-4$_{\text{modes32}}$	5200	0.004 43	0.020 76
UNO64	2080	0.017 07	0.070 89
UNO64	5200	0.007 25	0.035 96
U-Net$_{\text{base}}$6	2080	0.003 11 ± 0.000 01	0.017 74 ± 0.000 05
U-Net$_{\text{base}}$64	5200	0.002 02 ± 0.000 01	0.012 22
U-Net$_{\text{base}}$128	2080	0.002 12 ± 0.000 02	0.012 61 ± 0.000 05
U-Net$_{\text{base}}$128	5200	0.001 37	0.008 68 ± 0.000 04
U-Net$_{2015}$64	2080	0.003 24 ± 0.000 07	0.018 61 ± 0.000 44
U-Net$_{2015}$64	5200	0.002 23 ± 0.000 01	0.013 76 ± 0.000 15
U-Net$_{2015}$128	2080	0.002 38 ± 0.000 02	0.014 46 ± 0.000 02
U-Net$_{2015}$128	5200	0.001 61 ± 0.000 02	0.010 47 ± 0.000 06
U-Net$_{\text{mod}}$64	2080	0.001 51	0.009 75 ± 0.000 05
U-Net$_{\text{mod}}$64	5200	0.000 91 ± 0.000 01	0.006 37 ± 0.000 02

B.5 Parameter conditioning.

We use the same spatial resolutions and boundary conditions as described in Section B.4. The inputs to the Navier-Stokes parameter conditioning experiments are respective fields at the previous timestep. Exemplary rollout trajectories predicted by a single surrogate model for different buoyancy force values are displayed in Figures 20,21,22. We outline further details on the results in Table 10. Additionally, we ablate different parameter conditioning choices in Figure 19, namely "Addition" versus "AdaGN" for U-Net blocks, and "Addition" vs "Spatial-Spectral" for Fourier blocks. The default choice is "Addition".

Table 10: Parameter conditioning on the Navier-Stokes equation, velocity function formulation. Summed mean-squared errors of various architectures are reported for different number of training trajectories, and different time windows. Conditioning results at different time windows are averaged over 208 unseen values of the buoyancy force term. The best model of each model class is highlighted. Different parameter conditioning choices are ablated, namely "Addition" versus "AdaGN" for U-Net blocks, and "Addition" vs "Spatial-Spectral" for Fourier blocks. The default choice is "Addition".

		SMSE				
METHOD	Trajs.	0.375 s	0.75 s	1.5 s	3.0 s	6.0 s
FNO128$_{\text{modes16}}$	1664	0.005 17	0.006 93	0.011 73	0.026 66	0.064 23
FNO128$_{\text{modes16}}$	6656	0.003 88	0.004 83	0.007 40	0.015 44	0.041 77
FNO128$_{\text{modes16}}$-SpaSpec	1664	0.004 62	0.006 67	0.014 04	0.038 34	0.076 48
FNO128$_{\text{modes16}}$-SpaSpec	6656	0.003 48	0.004 55	0.008 01	0.019 73	0.052 08
U-Net$_{\text{mod}}$64	1664	0.000 72	0.001 11	0.002 16	0.006 22	0.028 05
U-Net$_{\text{mod}}$64	6656	0.000 40	0.000 61	0.001 15	0.003 25	0.015 60
U-Net$_{\text{mod}}$64-AdaGN	1664	0.000 59	0.000 90	0.001 77	0.005 68	0.029 89
U-Net$_{\text{mod}}$64-AdaGN	6656	0.000 31	0.000 50	0.001 00	0.003 00	0.016 32
U-Net$_{\text{att}}$64	1664	0.000 82	0.001 22	0.002 34	0.006 30	0.028 46
U-Net$_{\text{att}}$64	6656	0.000 46	0.000 69	0.001 30	0.003 56	0.016 75
U-Net$_{\text{att}}$64-AdaGN	1664	0.000 65	0.001 01	0.001 95	0.005 97	0.031 60
U-Net$_{\text{att}}$64-AdaGN	6656	0.000 35	0.000 55	0.001 06	0.003 10	0.017 03
U-F1Net$_{\text{modes16}}$	1664	0.001 38	0.001 80	0.002 75	0.006 45	0.024 97
U-F1Net$_{\text{modes16}}$	6656	0.000 54	0.000 78	0.001 33	0.003 38	0.015 04
U-F1Net$_{\text{modes16}}$-SpaSpec	1664	0.000 76	0.001 16	0.002 42	0.007 70	0.034 95
U-F1Net$_{\text{modes16}}$-SpaSpec	6656	0.000 40	0.000 61	0.001 16	0.003 39	0.017 54
U-F1Net$_{\text{att,modes16}}$	1664	0.001 42	0.002 05	0.003 33	0.007 62	0.027 95
U-F1Net$_{\text{att,modes16}}$	6656	0.000 62	0.000 86	0.001 46	0.003 58	0.015 79
U-F1Net$_{\text{att,modes16}}$-SpaSpec	1664	0.000 95	0.001 51	0.002 74	0.007 67	0.031 93
U-F1Net$_{\text{att,modes16}}$-SpaSpec	6656	0.000 46	0.000 72	0.001 32	0.003 72	0.017 86
U-F1Net$_{\text{att,modes16}}$-AdaGN	1664	0.001 51	0.002 00	0.003 11	0.007 48	0.028 79
U-F1Net$_{\text{att,modes16}}$-AdaGN	6656	0.000 61	0.000 86	0.001 37	0.003 47	0.015 96
U-F1Net$_{\text{modes16}}$-AdaGN	1664	0.001 61	0.002 12	0.003 30	0.007 58	0.027 69
U-F1Net$_{\text{modes16}}$-AdaGN	6656	0.000 55	0.000 77	0.001 32	0.003 38	0.015 64
U-F2Net$_{\text{modes16,8}}$	1664	0.001 78	0.002 40	0.003 57	0.008 50	0.033 46
U-F2Net$_{\text{modes16,8}}$	6656	0.000 93	0.001 16	0.001 68	0.003 76	0.015 64
U-F2Net$_{\text{modes16,8}}$-SpaSpec	1664	0.000 71	0.001 18	0.002 33	0.007 26	0.032 87
U-F2Net$_{\text{modes16,8}}$-SpaSpec	6656	0.000 39	0.000 64	0.001 22	0.003 68	0.018 62
U-F2Net$_{\text{modes16,8}}$-AdaGN	1664	0.001 86	0.002 89	0.004 82	0.013 07	0.047 67
U-F2Net$_{\text{modes16,8}}$-AdaGN	6656	0.000 98	0.001 24	0.001 85	0.004 18	0.018 09
U-F2Net$_{\text{att,modes16,8}}$	1664	0.001 97	0.002 91	0.004 64	0.011 29	0.038 10
U-F2Net$_{\text{att,modes16,8}}$	6656	0.001 16	0.001 53	0.002 25	0.004 90	0.018 88
U-F2Net$_{\text{att,modes16,8}}$-SpaSpec	1664	0.000 82	0.001 35	0.002 78	0.008 43	0.034 90
U-F2Net$_{\text{att,modes16,8}}$-SpaSpec	6656	0.000 44	0.000 71	0.001 37	0.004 04	0.018 96
U-F2Net$_{\text{att,modes16,8}}$-AdaGN	1664	0.001 96	0.002 81	0.004 29	0.010 19	0.036 73
U-F2Net$_{\text{att,modes16,8}}$-AdaGN	6656	0.000 91	0.001 21	0.001 79	0.004 14	0.017 05

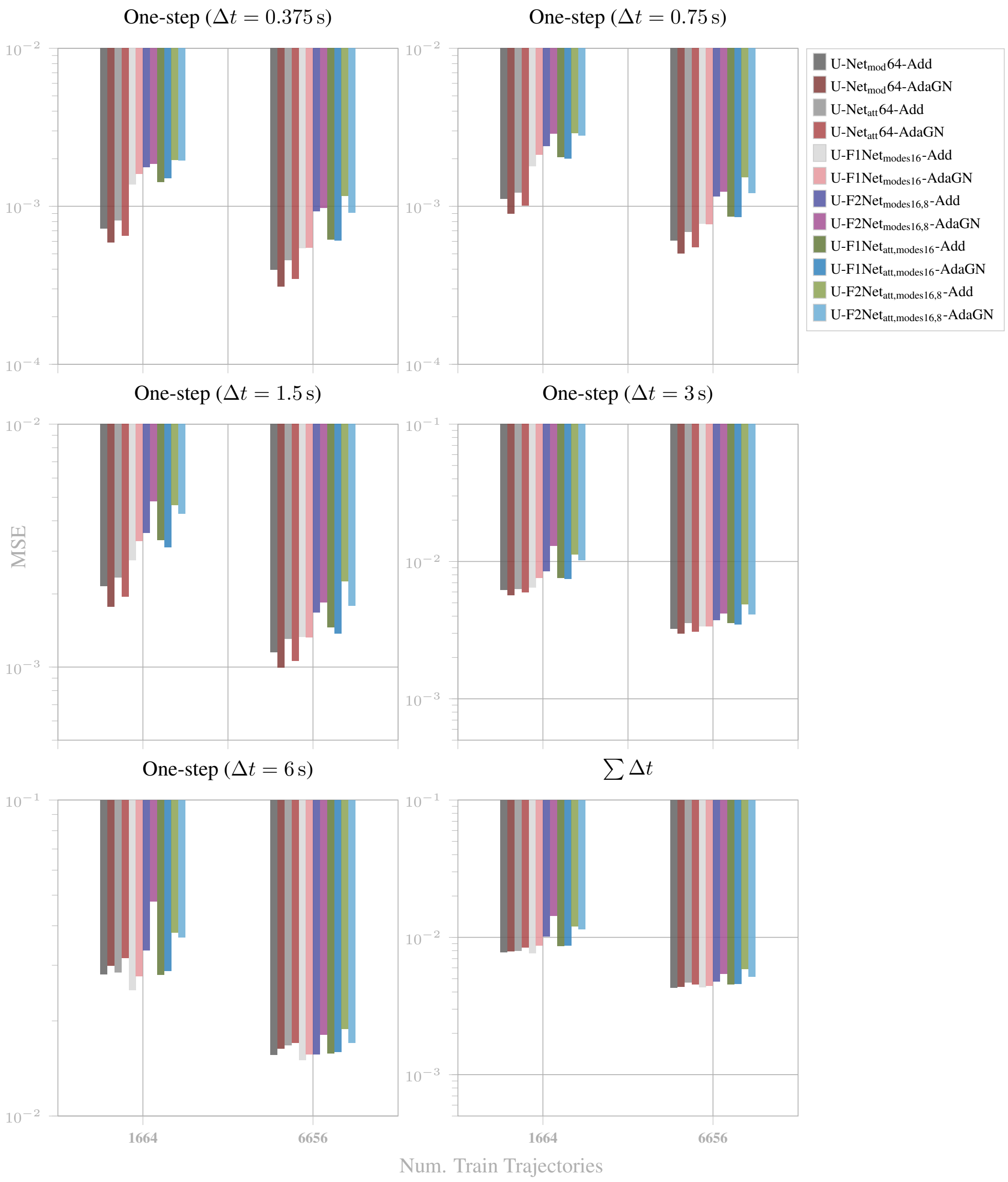
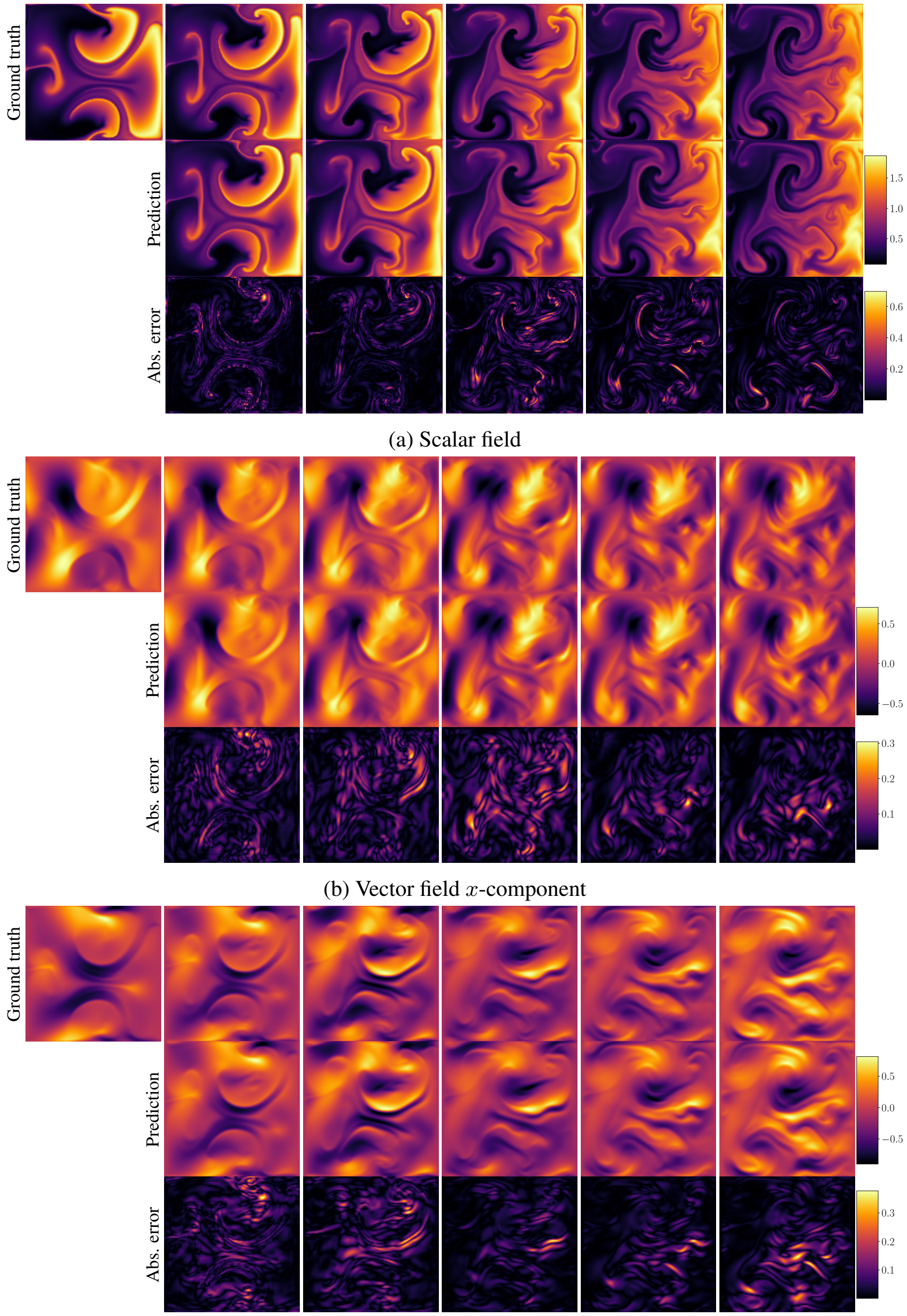


Figure 19: Navier-Stokes parameter conditioning experiments. Ablation results for different parameter conditioning methods. "Addition"(Add) and "AdaGN" are compared on one-step errors, reported for different time windows and averaged over 208 unseen values of the buoyancy force term. For low time windows, AdaGN seems to be beneficial.

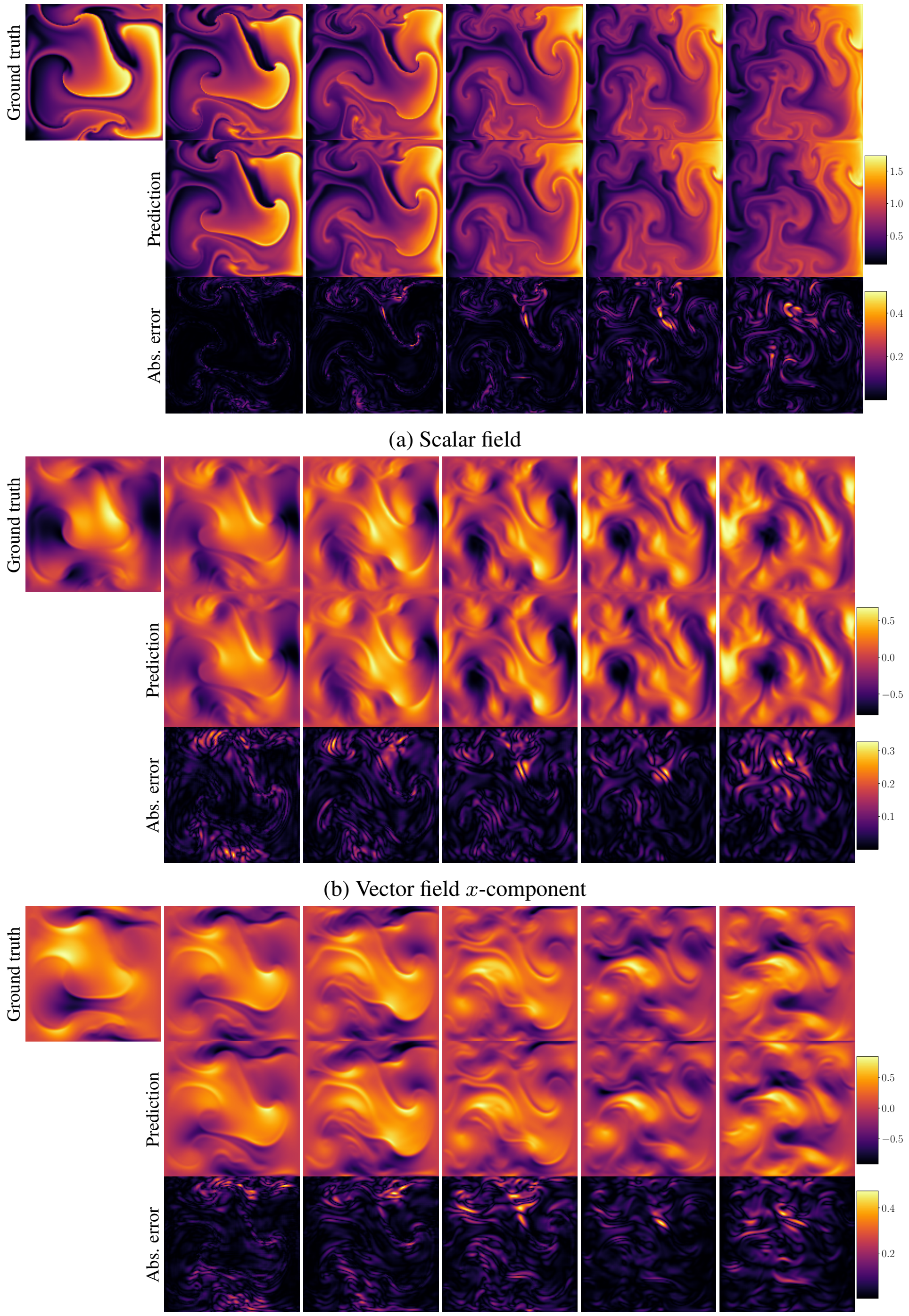


(a) Scalar field

(b) Vector field x-component

(c) Vector field y-component

Figure 20: Parameter conditioning for Navier-Stokes equations, $f = 0.21$. Example rollouts of the scalar and vector velocity field of the Navier-Stokes experiments are shown, obtained by a U-Net$_{\text{mod}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained for a time window $\Delta t = 1.5\,\text{s}$ and a buoyancy force term of $f = 0.21$. Model inputs are respective fields at the last timestep (left-most ground truth column).

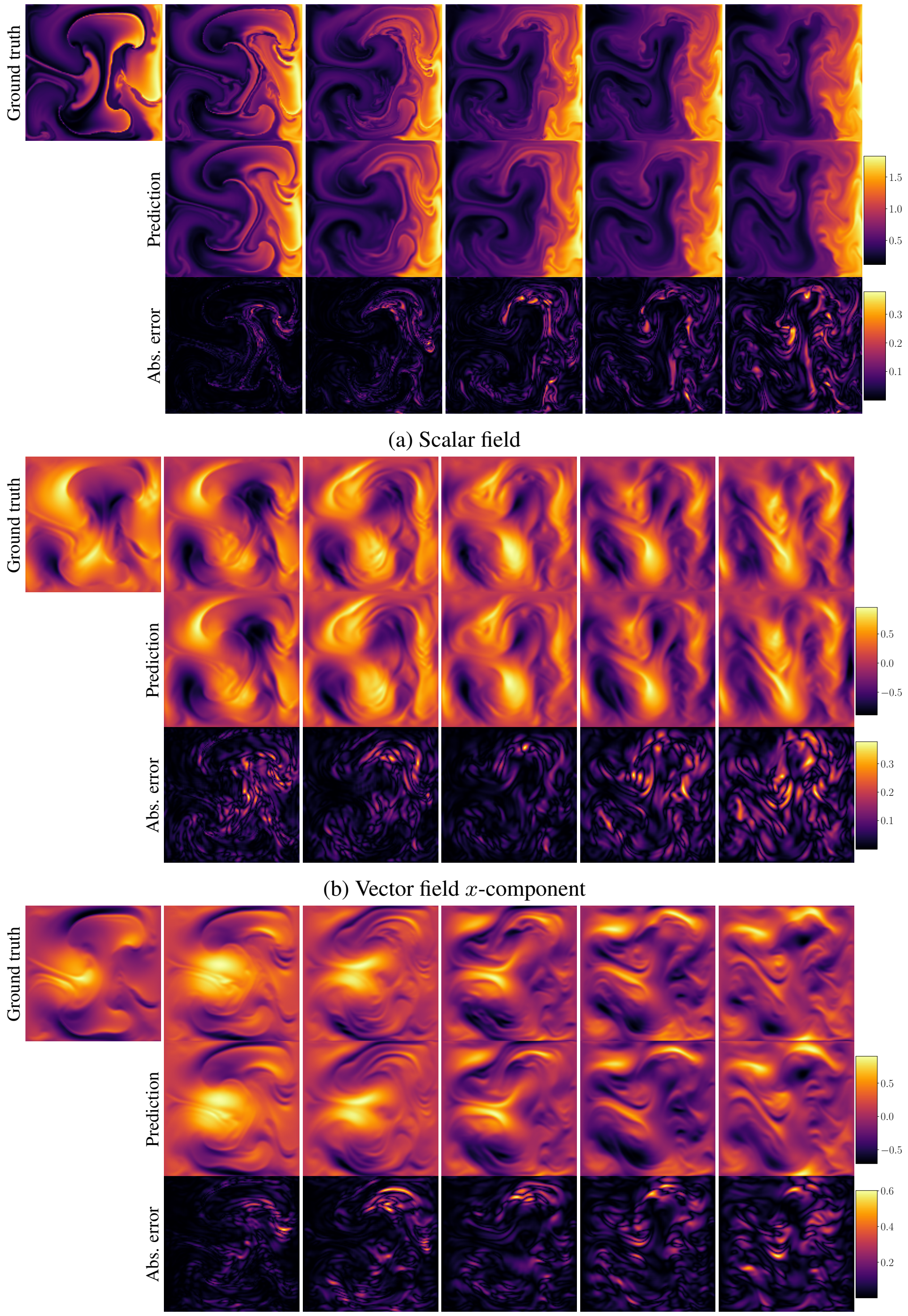


(a) Scalar field

(b) Vector field x-component

(c) Vector field y-component

Figure 21: Parameter conditioning for Navier-Stokes equations, $f = 0.33$. Example rollouts of the scalar and vector velocity field of the Navier-Stokes experiments are shown, obtained by a U-Net$_{\text{mod}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained at a time window $\Delta t = 1.5\,\text{s}$ and a buoyancy force term of $f = 0.33$. Model inputs are respective fields at the last timestep (left-most ground truth column).



(a) Scalar field

(b) Vector field x-component

(c) Vector field y-component

Figure 22: Parameter conditioning for Navier-Stokes equations, $f = 0.48$. Example rollouts of the scalar and vector velocity field of the Navier-Stokes experiments are shown, obtained by a U-Net$_{\text{mod}}$ PDE surrogate model (top), and compared to the ground truth (bottom). Predictions are obtained at a time window $\Delta t = 1.5\,\text{s}$ and a buoyancy force term of $f = 0.48$. Model inputs are respective fields at the last timestep (left-most ground truth column).