A. Appendix

A.1. PDEs and Data

We train our neural networks on datasets that contain solutions $\boldsymbol{u}$ for different boundary and initial conditions so that it is able to generalize across these conditions, without the need for retraining. Datasets are obtained as follows:

- Generate a pair of initial conditions $\boldsymbol{u}^0(\boldsymbol{x})$ and boundary conditions $\mathcal{B}[\boldsymbol{u}](t, \boldsymbol{x}) = 0$ and evaluate these values on the relevant subsets of our grid $\mathcal{T} \times \mathcal{X}$.
- Use a conventional high-accuracy numerical solver to obtain $\boldsymbol{u}(t, \boldsymbol{x})$ for all $(\boldsymbol{x}, t) \in \mathcal{T} \times \mathcal{X}$.
- Pick a series of input indices, and subsequent target indices, from the time interval $\mathcal{T}$, starting from a possibly randomly chosen location. The values of $\boldsymbol{u}(\cdot, \boldsymbol{x})$ at these indices will form the inputs and targets in our training scheme.

Burger's equation The Burger's equation is a common PDE that arises in fluid dynamics and nonlinear wave phenomena. In 1D the PDE, given the domain that we use, is given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{\nu}{\pi}\frac{\partial^2 u}{\partial x^2}, \quad x \in (0,1)\ t \in (0,2] \tag{9}$$

where u represents the speed of the fluid at a certain place and time, and ν is the viscosity coefficient. The Burger's equation describes the conservation of mass and momentum in a one-dimensional fluid flow, taking into account both convection effects ($u\frac{\partial u}{\partial x}$) and diffusion effects ($\nu\frac{\partial^2 u}{\partial x^2}$).

We use the 1D Burger's equation dataset from [27]. It is defined with a spatial resolution of 1024, with periodic boundary conditions, and temporal resolution of 200. The dataset consists of 9000 train and 1000 test trajectories started from samples of different initial conditions that are formed using a superposition of randomly chosen sinusoidal waves. A viscosity coefficient of $\nu = 0.001$ is used.

Darcy's Law The steady state 2D Darcy flow equation is a partial differential equation (PDE) that describes the flow of fluid through a porous medium. We use the PDE and domain expressed as

$$\begin{aligned} -\nabla\left(a(x)\nabla u(x)\right) = f(x), \quad x \in (0,1)^2, \\ u(x) = 0, \ x \in \partial(0,1)^2, \end{aligned} \tag{10}$$

where $a(x)$ is a diffusion coefficient based on the permeability of the porous medium and the dynamic viscosity of the fluid, $u(x)$ represents the pressure of the fluid, and f represents any external sources or sinks of fluid within the domain. We set f to constant 1 and train an operator that maps $a(x)$ to the solution $u(x)$.

We use the Darcy flow dataset from [17]. It is defined on a spatial grid of 421×421. We use 1024 train elements ($(a(x), u(x))$ pairs) and 100 validation elements. Details for how $a(x)$ is randomly generated for each data element can be found in [6].

Navier-Stokes equation The 2D Navier-Stokes Equation and domain that we use for our experiments is given by

$$\begin{aligned} \frac{\partial \boldsymbol{v}(\boldsymbol{x}, t)}{\partial t} &= -\boldsymbol{v}(\boldsymbol{x}, t) \cdot \nabla \boldsymbol{v}(\boldsymbol{x}, t) + \nu \nabla^2 \boldsymbol{v}(\boldsymbol{x}, t) \\ &\quad - \nabla p(\boldsymbol{x}, t) + \boldsymbol{f}(\boldsymbol{x}), \quad x \in (0, 32)^2, \ t \in (0, 21]. \end{aligned} \tag{11}$$

It describes the flow of a fluid in terms of its velocity components $\boldsymbol{v}$, the viscosity ν, and a buoyancy term $\boldsymbol{f}$. We assume incompressibility, so $\nabla \cdot \boldsymbol{v} = 0$, and Dirichlet boundary conditions ($\boldsymbol{v} = 0$).

The dataset is taken from [9]. A viscosity of $\nu = 0.01$ is used, and a buoyancy factor of $\boldsymbol{f} = (0, 0.5)^T$. While generating the data, the pressure field p is solved first, before subtracting its spatial gradients. In addition to the two velocity field components a scalar field $s(\boldsymbol{x})$ is introduced that is being transported through the velocity field. Its evolution is determined by

$$\frac{\partial s}{\partial t} = -\boldsymbol{v}(\boldsymbol{x}, t)\nabla s, \tag{12}$$

with Neumann boundaries $\frac{\partial s}{\partial \boldsymbol{x}} = 0$ on the edge of the domain. For more details, see [9, 4]. The full dataset consists of 2080 train samples and 1088 test samples.

Diffusion-Sorption Equation The diffusion-sorption equation models a diffusion process that is retarded by a sorption process. The 1D PDE is given by:

$$\frac{\partial u}{\partial t} = D/R(u)\frac{\partial^2 u}{\partial x^2} \quad x \in (0,1)\ t \in (0, 500], \tag{13}$$

where $D = 0.0005$ is the effective diffusion coefficient, and $R(u) = 1 + 2.16u^{-0.126}$ is the retardation factor hindering the diffusion process. This equation is applicable to, for example, groundwater contaminant transport.

The boundary conditions are $u(t, 0) = 1$ and $u(t, 1) = D\frac{\partial u}{\partial x}(t, 1)$ The dataset, taken from [27], is discretized into 1024 spatial steps and 501 time steps. There are 9000 train trajectories and 1000 test trajectories, each based on different randomly generated initial conditions using $u(0, x) \sim U(0, 0.2)$ for $x \in (0, 1)$.

Hyperparams	DiffSorp	Burgers'	N.S.	Darcy
Epochs	200	200	100	400
QAT Epochs	100	50	50	100
Batch size	50	50	16	4
Learning rate	1e-3	1e-3	1e-3	5e-4
Weight decay	1e-6	1e-6	1e-6	1e-6
QAT learn. rate	1e-4	1e-4	1e-4	1e-4
Input steps	5	5	4	1
Output steps	5	5	1	1
Train steps	10	20	1	1
Test steps	10	20	1	1
Subsample t	2	5	1	1
Subsample x	32	16	2	8

Table 3: Dataset-related hyperparameters for all models per experiment. The steps refer to consecutive time steps for the time-dependent PDEs, while the Darcy PDE can optionally be interpreted as having inputs at $t = 0$ and outputs at $t = 1$.

A.2. Hyperparameter specifications

We summarize the hyperparameters used per dataset in Table 3.

In the second session of experiments we did not subsample the spatial grid, except for the Darcy dataset for which we subsampled every 2 grid points. The first three scaling levels applied in figure 4 correspond (from left to right) to $0.01, 0.02, 0.05$ for the DiffSorp data, $0.02, 0.05, 0.1$ for the Burgers data, $0.1, 0.2, 0.5$ for the Navier-Stokes data and $0.05, 0.1, 0.2$ for the Darcy data. The loss measure used in all datasets is the MSE as described in equation 5. However, for the Darcy dataset, we also normalize each element in the sum by dividing by the squared targets, and take the squared root of the resulting sum.

The UNet is taken from [9], but in order to make its size comparable to the other models we use 16 hidden channels for the Navier-Stokes dataset and 8 hidden channels for the other datasets. The FNO model is taken from [17], using 4 layers, a width of 128, 32 modes for the 2D datasets, and 16 modes for the 1D datasets. The Transformer is taken from [6]. It uses 6 encoder layers, 128 hidden channels and a Galerkin attention type for the 2D datasets, and 4 encoder layers, 32 hidden channels and Fourier attention type for the 1D datasets.

A.3. Inference Cost Calculation Details

We describe a few differences compared to the regular deepspeed library [22]. Most standard deep learning operations rely on big matrix multiplications and as such deepspeed outputs the number of MACs used in their corresponding modules. On the other hand, there are some operations that have no MACs, and deepspeed simply outputs the number of FLOPs. However, because we care to differentiate addition and multiplication operations for our proxy measure of inference cost, we add some manual changes to deepspeed so that multiplications are properly accounted for.

- We assume bilinear interpolation to be three times the cost of linear interpolation, and for linear interpolation we assume 2 multiplications and 4 additions per output point.
- The FNO model uses Fast Fourier Transforms, which are not encountered for in deepspeed. To be able to take these into account in our proxy for model inference cost we assume a complexity of $N\lceil\log_2 N\rceil$ (additions and multiplications), which we divide by 2 when the real-valued FFT is used.
- In deepspeed no MACs are assigned to the einsum operator. Although it can in theory represent various different types of computations, in our code we only use it for basic matrix multiplications (in the FNO model). We thus change the deepspeed output accordingly.