Beyond Regular Grids: Fourier-Based Neural Operators on Arbitrary Domains

Levi Lingsch [1] **Mike Y. Michelis** [2 3] **Emmanuel de Bézenac** [1] **Sirani M. Perera** [4] **Robert K. Katzschmann** [2 3]
Siddhartha Mishra [1 2]

Abstract

The computational efficiency of many neural operators, widely used for learning solutions of PDEs, relies on the fast Fourier transform (FFT) for performing spectral computations. As the FFT is limited to equispaced (rectangular) grids, this limits the efficiency of such neural operators when applied to problems where the input and output functions need to be processed on general non-equispaced point distributions. Leveraging the observation that a limited set of Fourier (Spectral) modes suffice to provide the required expressivity of a neural operator, we propose a simple method, based on the efficient direct evaluation of the underlying spectral transformation, to extend neural operators to arbitrary domains. An efficient implementation* of such *direct spectral evaluations* is coupled with existing neural operator models to allow the processing of data on arbitrary non-equispaced distributions of points. With extensive empirical evaluation, we demonstrate that the proposed method allows us to extend neural operators to arbitrary point distributions with significant gains in training speed over baselines while retaining or improving the accuracy of Fourier neural operators (FNOs) and related neural operators.

1. Introduction

Partial Differential Equations (PDEs) are extensively used to mathematically model interesting phenomena in science and engineering (Evans, 2010). As closed-form or analytical solutions to solve PDEs are not available or practical, traditional numerical methods such as finite difference, finite element, and spectral methods (Quarteroni & Valli, 1994) are used to solve PDEs. Despite their tremendous success, the prohibitively high computational cost of these methods makes them infeasible for a variety of contexts in PDEs ranging from high-dimensional problems to the so-called *many query* scenarios (Karniadakis et al., 2021). This high computational cost also provides the rationale for the development of alternative *data driven* methods for the fast and accurate simulation of PDEs. Hence, a wide variety of machine learning algorithms have been proposed recently in this context. These include physics-informed neural networks (PINNs) (Raissi et al., 2019), MLPs, and CNNs for simulating parametric PDEs (Zhu & Zabaras, 2018; Lye et al., 2020; 2021; Wandel et al., 2020) as well as graph-based algorithms (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020; Brandstetter et al., 2022; Equer et al., 2023), to name a few.

However, as solutions of PDEs are expressed in terms of the so-called *solution operators*, which map input functions (initial and boundary data, coefficients, source terms) to the PDE solution, *operator learning*, i.e., learning the underlying operators from data, has emerged as a dominant framework for applying machine learning to PDEs. Existing operator learning algorithms include, but are not limited to, operator networks (Chen & Chen, 1995), DeepONets (Lu et al., 2019; Mao et al., 2020; Cai et al., 2021), attention-based methods such as (Kissas et al., 2022; Cao, 2021; Prasthofer et al., 2022), and neural operators (Kovachki et al., 2021b; Li et al., 2020b;c; Raonić et al., 2023). More recently, transformer-based approach for learning operators have been explored (Wu et al., 2023; Li et al., 2023), yet operator learning in the spectral domain retains an advantage (Hao et al., 2024).

Within this large class of operator learning algorithms, Neural Operators based on *non-local* spectral transformations, such as the Fourier Neural operator (FNO) (Li et al., 2020a) and its variants (Li et al., 2021; Pathak et al., 2022) have gained much traction and are widely applied. Apart from favorable theoretical approximation properties (Kovachki et al., 2021a; Lanthaler et al., 2023b), FNOs are attractive due to their expressivity, simplicity, and computational efficiency. A key element in the computational effi-

[1]Seminar for Applied Mathematics, ETH Zurich, Switzerland [2]ETH AI Center, ETH Zurich, Switzerland [3]Soft Robotics Lab, ETH Zurich, Switzerland [4]Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA. Correspondence to: Levi Lingsch <llingsch@student.ethz.ch>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

*Source code available on GitHub: `https://github.com/camlab-ethz/DSE-for-NeuralOperators`

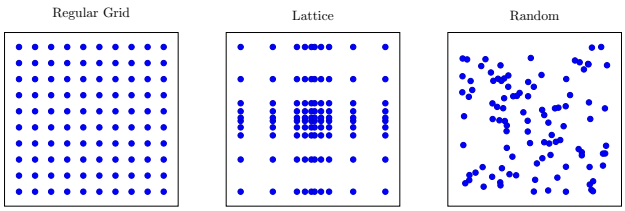


Figure 1: Distributions discussed in this paper. The vanilla FNO is restricted to the regular grid, but may be applied to a general lattice distribution, random distribution, or structured point cloud via the approach outlined in Section 2.

ciency of the FNO and its variants lies in the fact that its underlying convolution operation is efficiently carried out in Fourier space with the *fast Fourier transform* (FFT) algorithm. It is well-known that the FFT is only (log-)linear in computational complexity with respect to the number of points at which the underlying input functions are sampled. However, this computational efficiency comes at a cost as the recursive structure of the FFT limits its applications to inputs sampled on the so-called *Regular* or *equispaced Cartesian (Rectangular) grids*, see Figure 1 left for an illustration. This is a major limitation in practice. In real-world applications, where information on the input and output signals is measured by sensors, it is not always possible to place sensors only on an equispaced grid. Similarly, when data is obtained through numerical simulations, often it is essential to discretize PDEs on irregular grids, such as those adapted to capture relevant spatially localized features of the underlying PDE solution or on unstructured grids that fit the complex geometry of the underlying domain. See Figure 1 for examples of such non-equispaced distributions of sample points or *point clouds*.

Given this context, it is *imperative to design neural operators that can process input and output functions sampled on arbitrary point distributions.* Consequently, several methods have been proposed in the literature to address this limitation of FNOs and modify/enhance it to handle data on non-equispaced points. A straightforward fix would be to interpolate data from non-equispaced point distributions to equispaced grids. However, as shown in (Li et al., 2022), the resulting procedure can be computationally expensive and/or inaccurate. Li et al. (2022) propose a geometry-aware FNO (Geo-FNO) that appends a neural network to the FNO to learn a deformation from the underlying domain to a regular grid. Then, the standard FFT can be applied to the latent space of equispaced grid points. This learned diffeomorphism corresponds to an adaptive moving mesh (Huang & Russell, 2010). Factorized-FNO (F-FNO) builds upon the Geo-FNO, introducing an additional bias term in the Fourier layer and performing the Fourier transform over each dimension separately (Tran et al., 2023). The non-equispaced Fourier PDE solver (NFS) uses a vision mixer (Dosovitskiy et al., 2020) to interpolate from a non-equispaced signal onto a regular grid, again applying the standard FNO subsequently (Lin et al., 2022). All these methods share the same design principle, i.e., given inputs on non-equispaced points, *interpolate* or transform this data into a regular grid and then apply the FNO, leading to a natural question: *Is there an alternative approach where truncated spectral transformations, such as the discrete Fourier transform (DFT) inside FNO, can be performed efficiently on arbitrary point distributions?*

The main goal of this paper is to propose such an alternative approach. Our starting point is the recent observation, both theoretical as well as empirical, in (Lanthaler et al., 2023b;a) and references therein, that in practice, only a relatively small (fixed) number of Fourier modes suffice to provide the required expressivity of FNOs and the superior performance of FNO can be attributed to other factors such as residual connections, high-dimensional lifting operators and nonlinear activations. Moreover, the number of *effective* Fourier modes is significantly smaller than (and independent of) the number of points at which the input and output functions are sampled. This motivates us to design a simple method based on an *efficient direct evaluation of truncated spectral transformations within neural operators to extend them to arbitrary point clouds.* More concretely, our contributions to this paper are

- We present a simple, efficient method to formulate truncated *spectral transformations*, such as the Fourier transform or spherical harmonic transform, on arbitrary point distributions. A *novel* PyTorch implementation of these direct spectral evaluations is also presented.
- We *replace* the standard truncated spectral transformations that underpin many widely-used neural operators by the presented direct spectral evaluation, to obtain neural operators that can efficiently handle input and output data on arbitrary sample points over domains with arbitrary geometries.
- We present a suite of extensive numerical experiments to demonstrate that a variety of neural operators, based on truncated spectral transformations computed using the proposed direct approach, outperform baselines in terms of both accuracy and efficiency (training speed) in various scenarios involving input/output data on arbitrary point distributions.

2. Methods

Transforms, such as the Fourier transform and spherical harmonics, form the foundations of many widely-used neu-

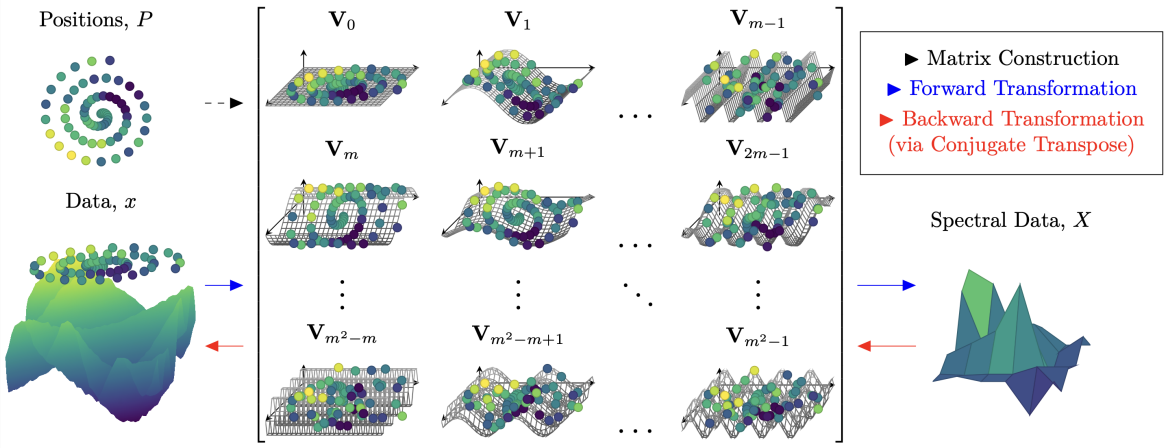


Figure 2: Illustration of the truncated Fourier transform on a point cloud. We construct a matrix using the values of a set of Fourier basis functions sampled at the positions of the sample points. The transformation between spatial and spectral domains is directly evaluated via a matrix-vector product, where the vector contains the training data. The subscript $\mathbf{V_j}$ refers to the j^{th} row of the matrix from Equation (6).

ral operators such as FNO (Li et al., 2020a), UFNO (Wen et al., 2022), F-FNO (Tran et al., 2023) and SFNO (Bonev et al., 2023). In this section, we describe the calculation of discrete forms of these transforms over nonuniform data and the construction of suitable matrices to compute these transforms efficiently within neural operators. To this end, we start with a short description of this matrix construction in the 1D case.

Forward Transformations. Any discretization of the Fourier transform will transform a sequence of N complex numbers $\mathbf{x} = [x_0, \ldots, x_{N-1}]^T$ into another sequence of complex numbers $\mathbf{X} = [X_0, \ldots, X_{N-1}]^T$ via

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i p_n f_k}, \quad 0 \leq k \leq N-1. \tag{1}$$

Here, $p_0, \ldots, p_{N-1} \in [0, 1]$ are sample point positions, $f_0, \ldots, f_{N-1} \in [0, N]$ are frequencies, and $i = \sqrt{-1}$. In this work, we focus on using nonuniform sample points with uniform (integer) frequencies; this is commonly referred to the type II NUDFT (Greengard & Lee, 2004).

Modern GPUs and machine learning libraries are optimized for handling matrices. We will construct a matrix of the form,

$$\mathbf{V}_{j,k} = \frac{1}{\sqrt{N}} \left[e^{-2\pi i (j p_k)} \right]_{j,k=0}^{m-1,N-1}, \tag{2}$$

where m the number of modes, truncated such that $m << N$. Then, we can computationally realize the transform (1) by a matrix-vector product

$$\mathbf{X} = \mathbf{V}\mathbf{x}, \tag{3}$$

relying on the optimized matrix operations provided by popular machine learning libraries, such as PyTorch. Furthermore, support for batched matrix multiplications is already present, allowing this approach to be used for batched data.

Extension to D-dimensional arbitrary point clouds. The multidimensional NUDFT may be presented as

$$X_{\boldsymbol{k}} = \sum_{\boldsymbol{n}=0}^{\boldsymbol{N}-1} x_{\boldsymbol{n}} e^{-2\pi i \boldsymbol{p_n} \cdot \boldsymbol{f_k}}, \tag{4}$$

transforming a D-dimensional array of complex numbers $x_{\boldsymbol{n}}$ into another complex D-dimensional array $X_{\boldsymbol{k}}$. In this case, $\boldsymbol{p_n} \in [0, 1]^D$ are sample points, $[0, 1]^D := [0, 1] \times [0, 1] \times \cdots \times [0, 1]$ (D times), $\boldsymbol{f_k} \in [0, N_1] \times [0, N_2] \times \cdots \times [0, N_D]$ are frequencies, and $\boldsymbol{n} = (n_1, n_2, \ldots, n_D)$ and $\boldsymbol{k} = (k_1, k_2, \ldots, k_D)$ are D-dimensional vectors of indices from 0 to $\boldsymbol{N} - 1 := (N_1 - 1, N_2 - 1, \ldots, N_D - 1)$.

For the purposes of neural operator learning, we again rearrange this summation into a matrix. We store the sample points as a matrix $P = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{N-1})^T \in [0, 1]^{N \times D}$. Additionally, the number of frequencies, or modes, is truncated at m along each spatial dimension, while higher modes are ignored. This results in the following matrix,

$$\mathbf{V}_{j,k} = \sqrt{\frac{D}{N}} \left[e^{-2\pi i \left(\sum_{l=0}^{D-1} \left(\left\lfloor \frac{j}{m^l} \right\rfloor \mod m \right) P_{k,l} \right)} \right]_{j,k=0}^{m^D-1,N-1}. \tag{5}$$

The order of the exponents is equivalent to a flattened tensor product. To further illustrate the general layout of this

matrix, we present $\mathbf{V}$ for the 2D case over m modes,

$$\mathbf{V} = \sqrt{\frac{2}{N}} \begin{bmatrix} e^{-2\pi i(0\mathbf{p_0}^T + 0\mathbf{p_1}^T)} \\ e^{-2\pi i(1\mathbf{p_0}^T + 0\mathbf{p_1}^T)} \\ \vdots \\ e^{-2\pi i((m-1)\mathbf{p_0}^T + 0\mathbf{p_1}^T)} \\ e^{-2\pi i(0\mathbf{p_0}^T + 1\mathbf{p_1}^T)} \\ e^{-2\pi i(1\mathbf{p_0}^T + 1\mathbf{p_1}^T)} \\ \vdots \\ e^{-2\pi i((m-1)\mathbf{p_0}^T + 1\mathbf{p_1}^T)} \\ \vdots \\ e^{-2\pi i(0\mathbf{p_0}^T + (m-1)\mathbf{p_1}^T)} \\ e^{-2\pi i(1\mathbf{p_0}^T + (m-1)\mathbf{p_1}^T)} \\ \vdots \\ e^{-2\pi i((m-1)\mathbf{p_0}^T + (m-1)\mathbf{p_1}^T)} \end{bmatrix}. \quad (6)$$

This construction is readily implemented in a single shot using tensorized methods, eliminating the need for less efficient loop constructs. Likewise, it leverages methods which have been highly optimized for 3D or 4D tensors, such as *torch.bmm()* and *torch.matmul()* for PyTorch.

Data from a general point cloud is stored in a vector, thus the transform relating $\mathbf{x}$ and $\mathbf{X}$ in the higher-dimensional cases is as readily calculated by the matrix-vector product (3), as in the 1D case.

In the special case that sample points lie on a lattice, it is possible to use a more memory efficient approach which is similar to the 2D FFT. We refer the interested reader to **SM** A.2 for more details.

Extension to Spherical Harmonics. The above method for constructing these matrices is not just limited to Fourier transforms and can thus be integrated in other neural operators. Simply using basis functions of other spectral transformations leads to a formulation that is adapted for the underlying spectral transform algorithms, such as wavelets or Laplace transforms. We exemplify such an extension to the spherical harmonics below.

The spherical harmonics are derived as the eigenfunctions of the Laplacian on the sphere. Given the maximum degree l_{max}, the associated harmonics can be explicitly calculated and arranged into a matrix as,

$$\begin{aligned} \mathbf{V}_{j,k} &= \left[Ce^{im\phi_k}\mathcal{P}_l^m(\cos\theta_k)\right]_{j,k=0}^{l_{max}^2-1,N-1} \\ m &= j - (\lfloor\sqrt{j}\rfloor^2 + \lfloor\sqrt{j}\rfloor), \quad l = \lfloor\sqrt{j}\rfloor, \end{aligned} \quad (7)$$

for any point k with polar angle θ_k and azimuth ϕ_k, where C is a normalization constant and $\mathcal{P}_l^m$ is the associated Legendre polynomial. The total number of modes is equal to l_{max}^2, as each degree l has $2l+1$ orders m, with $-m \leq l \leq m$. Once again, the transform is computed as a matrix-vector product as in (3).

Backward Transformations. The transformation from the spectral domain back to the physical space is immediately calculated by multiplying the data in the frequency domain by the conjugate transpose of the forward transformation matrix,

$$\mathbf{x} = \bar{\mathbf{V}}^T\mathbf{X}. \quad (8)$$

Since this approach avoids constructing a new matrix at run time, simply taking the adjoint is also computationally efficient. See Figure 2 for a visual summary of our algorithm for the realization of the *discrete spectral evaluations*.

The conjugate transpose converts from the spectral to the physical domain. However, it only serves as an inverse for orthogonal forward transforms. This is easily achieved by Euclidean Fourier transforms, when considering a subsampled grid where the number of sample points along each dimension is equal to the the number of Fourier modes taken along each respective dimension. In the case of spherical harmonics, however, orthogonality is preserved only in continuous cases. Nonetheless, orthogonal transformations on the sphere exist for discrete transforms under certain restrictions. Driscoll & Healy (1994) present sampling theorems for Fourier expansions and convolutions on a sphere which preserve orthogonality, as well as an efficient algorithm to compute such transforms. McEwen & Wiaux (2011), likewise, present sampling theorems and fast *spin* spherical harmonics algorithms for equiangular sampling points on the sphere by associating the sphere with a torus through periodic extension.

Computational Complexity. The most notable feature of FFT is its computational efficiency. Calculating the Fourier coefficients of a 1D signal, sampled at N points, by using the brute force DFT, costs $O(N^2)$. In contrast, the FFT algorithm computes these coefficients with $O(N \log N)$ complexity.

Hence, it is natural to wonder why one should reconsider matrix multiplication techniques in our setting. The maximum performance gain with FFT occurs when the FFT computes all the Fourier coefficients, or modes, of an underlying signal. Furthermore, peak efficiency is reached for points on a dyadic interval. While the number of modes to compute may be truncated, the interconnected nature of the self-recursive radix-2 FFT algorithm makes it difficult in practice to attain peak efficiency. We refer the reader to **SM** Figure 4 for a visual representation of the FFT algorithm. Thus, in the case of truncated modes, matrix multiplication techniques could be competitive vis a vis computational cost. As observed in Barnett et al. (2019), a direct

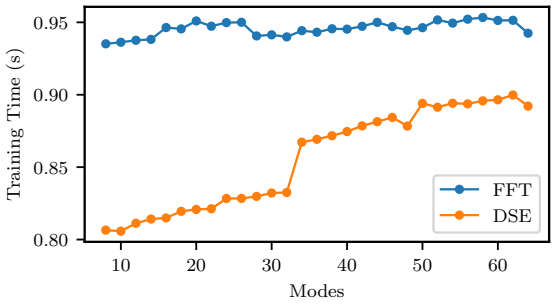


Figure 3: Ablation study to compare the FNO training time using the FFT and the direct spectral evaluations (DSE) for the 1D Burgers' equation on equispaced data.

evaluation is competitive or more efficient when the number of modes is on the order of 10^1 or fewer.

Moreover, for neural operators, only a small subset of nonzero modes are required to approximate the operator (Li et al., 2020a; Lanthaler et al., 2023b;a), independent of the number of points. Therefore, the computational complexity of the proposed approach cost $O(mN)$ as the matrix structure can be fully determined in $O(mN)$ as opposed to $O(N^2)$ (Gohberg & Olshevsky, 1994b; Pan, 2001; Gohberg & Olshevsky, 1994a). We also present an ablation study in Figure 3, varying the number of modes and observing the computation time for the 1D Burgers experiment, described in Section 3. Results for computation time as the number of modes is varied for 2D equispaced grids, 2D point clouds, and spherical geometries are presented **SM** Table 4. Directly evaluating the Fourier transform is clearly more efficient within the typical range of 12 to 32 modes required for by FNO (Li et al., 2020a), and it remains more efficient even up to 64 modes. This figure suggests that direct spectral evaluations will be faster to run in practice.

A detailed discussion on the computational complexity of the matrix method presented here, both for Fourier transforms and Spherical harmonics is provided in **SM** A.3.

3. Experimental Results

In this section, our aim is to investigate the performance of the presented *Direct Spectral Evaluations* (DSE) within various neural operator architectures on a challenging suite of diverse PDE tasks.

Implementation, Training Details and Baselines. A key contribution of this paper is a new implementation of the presented matrix multiplications in *PyTorch*, which enables us to efficiently compute Fourier transforms. Within a neural network, an efficient $O(mN)$ algorithm must also be parallelizable to handle batches, as this massively speeds up the training process. Batches of data with the same or different point distributions are easily handled by the *torch.matmul()* and *torch.bmm()* functions.

In all experiments, we use a simple grid search to select the hyperparameters. We train all models until convergence. We also use the L1-loss function, which produced both a lower L1-error and L2-error than the L2-loss. The test error was measured in all experiments as the relative L1 error.

As baselines, we use the geometric diffeomorphism (Geometric Layer), described by Li et al. (2022) in experiments where the underlying domain has a complicated, non-equispaced geometry, when applicable. For the one dimensional experiment, we use a cubic interpolation scheme on the nonequispaced data, as well as two Nonuniform FFT approaches. For the experiments on a lattice, we take the model's performance over the original grid as a baseline. To apply the SFNO in the spherical example, we explore radial-basis function interpolation schemes using both a Gaussian kernel with variance 0.1 and a linear kernel.

To show the effectiveness as well as the generality of DSE, we implement it within several prominent neural operators, namely the FNO, UFNO (Wen et al., 2022), FFNO (Tran et al., 2023), and SFNO (Bonev et al., 2023) (for data on the sphere). Model sizes are chosen to be as close as possible when comparing the DSE and the baselines. All experiments are performed on the Nvidia GeForce RTX 3090 with 24GB memory.

Benchmark 1: Burgers' Equation. The one-dimensional viscous Burgers' equation is a widely considered model problem for fluid flow given by

$$\begin{aligned} \partial_t u(x,t) + \partial_x \left(\frac{1}{2}u^2(x,t)\right) &= \nu \partial_{xx} u(x,t), \\ u(x,0) &= u_0(x), \end{aligned} \tag{9}$$

where $x \in (0,1), t \in (0,1]$, u denotes the fluid velocity and ν the viscosity. We follow (Li et al., 2020a) in fixing $\nu = 0.1$ and considering the operator that maps the initial data u_0 to the solution $u(\cdot, T)$ at final time $T = 1$. The training and test data, presented in (Li et al., 2020a) for this problem, is used. We start by comparing the standard version of FNO (with FFT) to FNO with the DSE for data sampled on an equispaced grid. The difference between the resulting test errors is negligible, because the underlying algorithm is the same in this case modulo small numerical errors, as reported in Table 1. Moreover, the training times per epoch were also comparable. In contrast, for data sampled from points drawn from a contracting-expanding distribution, illustrated in **SM** Figure 5, the proposed method was 5% more accurate on average when compared to FNO with a cubic interpolation, interpolating data from the contracting-expanding distribution to an equispaced grid. However, the training time with the DSE

Table 1: Performance results for all experiments, comparing the DSE approach to various baselines. Directly evaluating the Fourier transform or spherical harmonics over the given domain offers clear advantages in speeding up training time, improving the testing error, or both across a variety of experiments with both equispaced grids and unusual geometries.

Model	Method	Training Time (per epoch)	L1 Test Error
1D: Burgers' Equation			
Equispaced Distribution:			
FNO	**DSE**	**0.72**s	**0.0551**%
	FFT	0.78s	**0.0575**%
Contracting-Expanding Distribution:			
FNO	**DSE**	**0.11**s	**0.184**%
	FFT + Cubic Interpolation	1.00s	0.195%
	KB-NUFFT	30.0s	0.346%
	Toeplitz-NUFFT	0.85s	0.740%
2D Nonequispaced Lattice: Shear Layer			
FNO	**DSE** on Noneq. Lattice	**57**s	**5.53**%
	FFT on Equispaced Grid	251s	6.76%
UFNO	**DSE** on Noneq. Lattice	68s	5.61%
	FFT on Equispaced Grid	287s	6.61%
FFNO	**DSE** on Noneq. Lattice	108s	12.4%
	FFT on Equispaced Grid	380s	12.4%
2D Nonequispaced Lattice: Specific Humidity			
FNO	**DSE** on Noneq. Lattice	**1.5**s	4.65%
	FFT on Equispaced Grid	15s	5.09%
UFNO	**DSE** on Noneq. Lattice	2.5s	**3.91**%
	FFT on Equispaced Grid	23s	4.34%
FFNO	**DSE** on Noneq. Lattice	2.3s	4.41%
	FFT on Equispaced Grid	26s	5.02%
2D Point Cloud: Flow Past Airfoil			
FNO	**DSE**	2.8s	**0.220**%
	Geometric Layer	6.2s	1.20%
UFNO	**DSE**	2.6s	0.380%
	Geometric Layer	7.1s	0.679%
FFNO	**DSE**	**2.1**s	0.650%
	Geometric Layer	6.9s	2.10%
2D Point Cloud: Elasticity			
FNO	**DSE**	0.41s	1.96%
	Geometric Layer	0.71s	2.39%
UFNO	**DSE**	0.60s	2.05%
	Geometric Layer	1.0s	2.16%
FFNO	**DSE**	**0.28**s	**1.73**%
	Geometric Layer	0.44s	2.20%
Random Spherical Point Cloud: Shallow Water Equations			
SFNO	**DSE**	**15**s	**3.88**%
	Gaussian Interpolation	86s	7.29%
	Linear Interpolation	71s	12.7%
FNO	**DSE**	16s	5.39%
	Gaussian Interpolation	92s	8.41%
	Linear Interpolation	83s	15.2%

was notably improved, by almost a factor of 4, when compared to the interpolation baseline. Note that the cubic interpolation was computed beforehand and its cost not taken into account in reporting the training time. In addition, we draw comparisons with relevant Nonuniform FFT (NUFFT) algorithms with PyTorch implementations, namely the Kaiser-Bessel and Toeplitz NUFFT (Muckley et al., 2020). The performance of these approaches is relatively poor, as they fundamentally rely on an interpolation+FFT scheme.

Benchmark 2: Shear Layer. We follow a recent work on convolutional neural operators (Raonić et al., 2023) in considering the incompressible Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \nu \Delta \mathbf{u}, \quad \nabla \cdot \mathbf{u} = 0. \tag{10}$$

Here, $\mathbf{u} \in \mathcal{R}^2$ is the fluid velocity and p is the pressure. The underlying domain is the unit square with periodic boundary conditions and the viscosity $\nu = 4 \times 10^{-4}$, only applied to high-enough Fourier modes (those with amplitude ≥ 12) to model fluid flow at *very high Reynolds-number*. The solution operator maps the initial conditions $\mathbf{u}(t = 0)$ to the solution at final time $T = 1$. We consider initial conditions representing the well-known *thin shear layer* problem (Bell et al., 1989; Lanthaler et al., 2021) (see (Raonić et al., 2023) for details), where the shear layer evolves via vortex shedding to a complex distribution of vortices (see **SM** Figure 10a for an example of the flow). The training and test samples are generated, with a spectral viscosity method (Lanthaler et al., 2021) of a fine resolution of 1024^2 points, from an initial sinusoidal perturbation of the shear layer (Lanthaler et al., 2021), with layer thickness of 0.1 and 10 perturbation modes, the amplitude of each sampled uniformly from $[-1, 1]$ as suggested in (Raonić et al., 2023). As seen from **SM** Figure 10a, the flow shows interesting behavior with sharp gradients in two mixing regions, which are in the vicinity of the initial interfaces. However, the flow is nearly constant further away from this mixing region. Hence, we will consider input functions being sampled on a lattice shown in **SM** Figure 6 which is adapted to resolve regions with large gradients of the flow. On the other hand, the FFT based FNO, UFNO and FFNO baselines are tested on the equispaced point distribution. From Table 1, we observe that the proposed method is marginally more accurate while consistently being 4 times faster per training epoch across all models, demonstrating a significant computational advantage on this benchmark.

Benchmark 3: Surface-Level Specific Humidity. Next, we focus on a *real world* data set and learning task where the objective is to predict the surface-level specific humidity over South America at a later time (6 hours into the future), given inputs such as wind speeds, precipitation, evaporation, and heat exchange at a given time. The exact list of inputs is given in **SM** Table 2. The physics of this problem are intriguingly complex, necessitating a *data-driven approach* to learn the underlying operator. To this end, we use the (MERRA-2) satellite data to forecast the surface-level specific humidity (EarthData, 2021 - 2023). Moreover, we are interested in a more accurate regional prediction, namely over the Amazon rainforest. Hence, for models with the DSE, we will sample data on points on a lattice that is more dense over this rainforest, while being sparse (with smooth transitions) over the rest of the globe, see **SM** Figure 7 for visualization of this lattice. Test error is calculated over the region, shown in **SM** Figure 10b. The results, presented in Table 1, show that the localization capabilities of the proposed method not only offer more accurate results, but DSE based neural operators are also one order of magnitude faster to train than the baselines. The greater accuracy is also clearly observed in **SM** Figure 10b, where we observe that the FNO-DSE is able to capture elements such as the formation of vortices visible in the lower right hand corner and the mixing of airstreams over the Pacific Ocean.

Benchmark 4: Flow Past Airfoil. We also investigate transonic flow over an airfoil, as governed by the compressible Euler equations, with the problem setup considered in (Li et al., 2022). The underlying operator maps the airfoil shape to the pressure field. In this case, the underlying distribution of sample points changes between each input (airfoil shape). Directly formulating the Fourier transform within neural operators, one can readily handle this situation. As baselines, we augment FNO, UFNO and FFNO with the geometric layer of Geo-FNO as proposed in (Li et al., 2022). To have a fair comparison with DSE based models, we allow Geometric layer based models to learn the underlying diffeomorphism online. The test errors, presented in Table 1 show that DSE-based models are both much more accurate and much faster to train than the geometric layer based baselines.

Benchmark 5: Elasticity-P. Again, we follow (Li et al., 2022), to investigate the performance of this method on hyper-elastic materials. We take the data, exactly as outlined in their work, predicting the stress as output. Again, we train all models (FNO, UFNO, FFNO) on the point cloud data with DSE and with the geometric layer (as proposed in (Li et al., 2022)) as a baseline. For each underlying model, the DSE is more accurate than the geometric layer while being significantly faster to train.

Benchmark 6: Spherical Shallow Water Equations. We follow the recent work on spherical neural operators (Bonev et al., 2023) by considering the shallow water equations on a rotating sphere. In this experiment, training data

is generated on the fly such that new data is used for training and evaluation in every epoch, as described in (Bonev et al., 2023). For each sample, we draw points from a random uniform distribution over the sphere to create a point cloud, as opposed to a grid, see **SM** Figure 9 for an illustration of this point cloud. These may correspond to sensors over the globe in real-world applications. The training data for both models each have approximately 5000 points. As baselines, we considered SFNO but with data interpolated back to a regular grid on the sphere using either linear interpolation or radial basis function interpolation with a Gaussian kernel. We also compare the FNO on the interpolated data with the DSE approach on the point cloud data. The results, presented in Table 1, clearly show that SFNO using DSE readily outperforms baselines on the interpolated data in terms of both accuracy as well as training speed. This difference is very pronounced for the baselines based on interpolation, which are both expensive as well as inaccurate. As expected the FNO-DSE model performs significantly worse than the SFNO-DSE model which takes into account the underlying spherical geometry. Additionally, we provide the baseline comparison of the SFNO and FNO on the original (not from interpolation) grid in **SM** Table 6 as a reference to their relative performance. Likewise, we test a trained SFNO model on samples with varying numbers of collocation points. These results, presented in Table 7 show that this method is capable of retaining performance, even when trained on grids at different resolutions.

Summarizing, the results of the numerical experiments clearly demonstrate that DSE is able to handle data sampled on arbitrary point distributions and to accurately approximate the underlying operators. It readily outperforms baselines, based either on interpolation or geometric transformations, both on accuracy as well as on training speed. This is particularly evident on problems where the data is sampled on point clouds and where other operator learning methods such as U-Nets or convolutional neural operators cannot be readily applied.

4. Discussion

Summary. Widely used neural operators such as FNO and its variants are limited to input/output data sampled on equispaced grids as the underlying spectral transformations can only be efficiently evaluated on this data structure. To expand the range of such neural operators, we propose replacing the standard computations of these spectral transformations within neural operators with a simple, general method that leverages the low number of (truncated) modes to efficiently compute the spectral transform even on arbitrary sampling point distributions (point clouds). By design, our method is able to process data on arbitrary point distributions with low computational cost. A novel PyTorch implementation is also presented to realize the proposed approach that we term as the Direct Spectral Evaluation (DSE).

DSE is flexible enough to be embedded within any neural operator that uses non-local spectral transformations. We test with a variety of neural operators such as FNO, UFNO, FFNO, and recently SFNO for data on a sphere. Moreover, DSE is general enough to handle point distributions ranging from equispaced grids through lattices to randomly distributed point clouds. We investigate the efficiency of the proposed DSE by testing it in conjunction with a variety of available neural operators on a suite of benchmarks that correspond to a variety of PDEs as well as sampling point distributions. In all the considered benchmarks, DSE outperformed the baselines (based either on interpolation or learnable geometric differomorphisms as proposed in (Li et al., 2022)) both in terms of accuracy as well as training speed, even showing order of magnitude speedups in some cases. Additionally, we present in **SM** Table 7, the generalization capabilities of the DSE approach to be applied to point clouds not encountered during training time, even when the number of points varies greatly. Thus, we present a novel yet simple method that can be used readily within neural operators to handle input/output data sampled on arbitrary point distributions and learn the underlying operators accurately and efficiently.

Related Work. Nonequispaced FFT (NUFFT) algorithms have already been developed (Dutt & Rokhlin, 1993; Beylkin, 1995; Dutt & Rokhlin, 1995; Liu & Nguyen, 1998; Kircheis & Potts, 2023) as well as libraries for efficient GPU implementations (Shih et al., 2021) and PyTorch implementations (Muckley et al., 2020). The techniques frequently used in these NUFFTs or approximated inverse NUFFT include interpolation, windowing, and/or sampling along with FFT to achieve $\mathcal{O}(n \log n)$ complexity algorithms (Selva, 2018; Heinig & Rost, 1984; Gelb & Song, 2014; Kunis & Potts, 2007; Ruiz-Antolin & Townsend, 2018; Averbuch et al., 2016; Kircheis & Potts, 2019; 2018; Kunis, 2006; Perera et al., 2022). Essentially, the NUFFT involves interpolating to the equispaced grid followed by the FFT. However, interpolation based-approaches has already been shown to be sub-optimal for the case of neural operator learning on arbitrary (Li et al., 2022). Furthermore, interpolation within the operator structure increases the computation time, making it more efficient to interpolate data to a grid before training or testing and simply apply the standard FNO. The NUDFT is rarely used, as many applications of Fourier transforms require all Fourier coefficients, resulting in $O(n^2)$ cost (Bagchi & Mitra, 1999). However, this is not the case for the neural operators that we consider, and thus the proposed DSE avoids the rapidly growing computational costs associated with NUDFT. In some implementations, the Geo-FNO also uses a nonequis-

paced Fourier transform in conjunction with the diffeomorphism. As opposed to constructing a standard matrix, (Li et al., 2022) construct an $N+1$ tensor, N being the number of spatial dimensions. This geometric-Fourier layer then transforms the problem to a set of points on a regular grid, where the FFT is used. While the nonequispaced transformation does have some similarities to that proposed here, the proposed method differs fundamentally in its use, leading to notable differences in the results. The construction and multiplication we offer in our implementation are fast and efficient, allowing us to use the nonequispaced transformation within each Fourier layer, as opposed to transforming the data to an equispaced grid through some slower transformation process and then apply the FFT. Likewise, this method allows us to avoid the need for any sort of diffeomorphism. In practice, we found the diffeomorphism layer difficult to train in tandem with the Fourier layer of the FNO. This process often requires careful tuning of the associated diffeomorphism loss, as well as the need to freeze this layer after some point of time and to retrain the FNO with the diffeomorphism model fixed. In contrast, the FNO-DSE converges just as reliably as the basic FNO and we see the same behavior with variants of FNO such as UFNO and FFNO.

Limitations and Future Work. The elements of the matrix are directly related to the positions of the data points for a given problem. Thus, if all training samples have different point clouds, a different matrix must be constructed for each sample. Constructing the matrices at run time, *i.e.*, during training, hinders performance; however, this is mitigated by the tensorized matrix construction methods available in the implementation. As outlined by the results, the run-time matrix construction is still able to outperform other techniques for handling point cloud data. Finally, we would like to highlight the potential of the DSE for dealing with very general spectral transformations for data sampled on arbitrary point distributions. We have considered both Fourier transforms and Spherical harmonics in this paper but other relevant spectral transforms such as Wavelets or Laplace transforms will also be considered in future work.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically for applications in scientific computing. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgement

Sirani M. Perera's work was partially supported by the NSF award number 2229473.

References

Averbuch, A., Shabat, G., and Shkolnisky, Y. Direct inversion of the three-dimensional pseudo-polar fourier transform. *SIAM Journal on Scientific Computing*, 38(2): A1100–A1120, 2016.

Bagchi, S. and Mitra, S. K. *The nonuniform discrete Fourier transform and its applications in signal processing*. Springer Science and Business Media, 1999. doi: 10.1007/978-1-4615-4925-3.

Barnett, A. H., Magland, J. F., and af Klinteberg, L. A parallel non-uniform fast Fourier transform library based on an "exponential of semicircle" kernel. *SIAM J. Sci. Comput.*, 41(5):C479–C504, 2019.

Bell, J., Collela, P., and Glaz, H. M. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.

Beylkin, G. On the fast fourier transform of functions with singularities. *Applied and Computational Harmonic Analysis*, 2(4):363–381, 1995.

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., and Anandkumar, A. Spherical fourier neural operators: Learning stable dynamics on the sphere, 2023.

Brandstetter, J., Worrall, D. E., and Welling, M. Message Passsing Neural PDE solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Cai, S., Wang, Z., Lu, L., Zaki, T. A., and Karniadakis, G. E. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

Cao, S. Choose a transformer: Fourier or galerkin. In *35th conference on neural information processing systems*, 2021.

Chen, T. and Chen, H. 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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Driscoll, J. R. and Healy, D. M. Computing fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15(2):202–250, 1994. ISSN 0196-8858. doi: 10.1006/aama.1994.1008.

Dutt, A. and Rokhlin, V. Fast fourier transforms for nonequispaced data. *SIAM Journal on Scientific Computing*, 14(6):1368–1393, 1993.

Dutt, A. and Rokhlin, V. Fast fourier transforms for nonequispaced data ii. *Applied and Computational Harmonic Analysis*, 2:85–100, 1995.

EarthData, N. Modern-era restrospective analysis for research and applications (merra-2) 2-dimensional data collection., 2021 - 2023.

Equer, L., Rusch, T., and Mishra, S. Multi-scale message passing neural pde solvers. *arXiv preprint arXiv:2302.03580*, 2023.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

Gelb, A. and Song, G. A frame theoretic approach to the nonuniform fast fourier transform. *SIAM Journal on Numerical Analysis*, 52(3):1222–1242, 2014.

Gohberg, I. and Olshevsky, V. Fast algorithms with preprocessing for matrix-vector multiplication problems. *Journal of Complexity*, 10(4):411–427, 1994a. ISSN 0885-064X. doi: https://doi.org/10.1006/jcom.1994.1021. URL https://www.sciencedirect.com/science/article/pii/S0885064X84710211.

Gohberg, I. and Olshevsky, V. Complexity of multiplication with vectors for structured matrices. *Linear Algebra and its Applications*, 202:163–192, 1994b. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(94)90189-9. URL https://www.sciencedirect.com/science/article/pii/0024379594901899.

Greengard, L. and Lee, J.-Y. Accelerating the nonuniform fast fourier transform. *SIAM Review*, 46(3):443–454, 2004. doi: 10.1137/S003614450343200X.

Hao, Z., Su, C., Liu, S., Berner, J., Ying, C., Su, H., Anandkumar, A., Song, J., and Zhu, J. Dpot: Auto-regressive denoising operator transformer for large-scale pde pretraining, 2024.

Heinig, G. and Rost, K. *Algebraic Methods for Toeplitz-Like Matrices and Operators*. Akademie-Verlag, Berlin, 1984.

Huang, W. and Russell, R. D. *Adaptive Moving Mesh Methods*, volume 174. Springer Science and Business Media, 2010.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics informed machine learning. *Nature Reviews Physics*, pp. 1–19, may 2021. doi: 10.1038/s42254-021-00314-5. URL www.nature.com/natrevphys.

Kircheis, M. and Potts, D. Direct inversion of the nonequispaced fast fourier transform. 11 2018.

Kircheis, M. and Potts, D. Direct inversion of the nonequispaced fast fourier transform. *Linear Algebra and its Applications*, 575:106–140, 2019.

Kircheis, M. and Potts, D. Fast and direct inverion methods for the multivariate nonequispaced fast fourier transform. 2 2023.

Kissas, G., Seidman, J. H., Guilhoto, L. F., Preciado, V. M., Pappas, G. J., and Perdikaris, P. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

Kovachki, N., Lanthaler, S., and Mishra, S. On universal approximation and error bounds for fourier neural operators. 07 2021a.

Kovachki, N., Li, Z., Liu, B., Azizzadensheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481v3*, 2021b.

Kunis, S. Nonequispaced fft generalisation and inversion. *PhD. dissertation, Inst. Math., Univ. Lübeck, Lübeck*, 2006.

Kunis, S. and Potts, D. Stability results for scattered data interpolation by trigonometric polynomials. *SIAM Journal on Scientific Computing*, 29:1403–1419, 2007.

Lanthaler, S., Mishra, S., and Parés-Pulido, C. Statistical solutions of the incompressible euler equations. *Mathematical Models and Methods in Applied Sciences*, 31 (02):223–292, Feb 2021. ISSN 1793-6314. doi: 10.1142/s0218202521500068. URL http://dx.doi.org/10.1142/s0218202521500068.

Lanthaler, S., Li, Z., and Stuart, A. M. The nonlocal neural operator: Universal approximation, 2023a.

Lanthaler, S., Molinaro, R., Hadorn, P., and Mishra, S. Nonlinear reconstruction for operator learning of pdes with discontinuities. In *International Conference on Learning Representations*, 2023b.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020a. URL https://arxiv.org/abs/2010.08895.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Stuart, A. M., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6755–6766. Curran Associates, Inc., 2020b.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Stuart, A. M., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6755–6766. Curran Associates, Inc., 2020c.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries, 2022.

Li, Z., Shu, D., and Farimani, A. B. Scalable transformer for pde surrogate modeling, 2023.

Lin, H., Wu, L., Huang, Y., Li, S., Zhao, G., and Li, S. Z. Non-equispaced fourier neural solvers for pdes, 2022.

Liu, Q. and Nguyen, N. An accurate algorithm for nonuniform fast fourier transforms (NUFFTs). *IEEE Microwave and Guided Wave Letters*, 8(1):18–20, 1998. doi: 10.1109/75.650975.

Lu, L., Jin, P., and Karniadakis, G. E. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, abs/1910.03193, 2019. URL http://arxiv.org/abs/1910.03193.

Lye, K. O., Mishra, S., and Ray, D. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, 410:109339, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109339. URL https://www.sciencedirect.com/science/article/pii/S0021999120301133.

Lye, K. O., Mishra, S., Ray, D., and Chandrashekar, P. Iterative surrogate model optimization (ISMO): An active learning algorithm for PDE constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113575. URL https://www.sciencedirect.com/science/article/pii/S004578252030760X.

Mao, Z., Lu, L., Marxen, O., Zaki, T., and Karniadakis, G. E. DeepMandMnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Preprint, available from arXiv:2011.03349v1, 2020.

McEwen, J. D. and Wiaux, Y. A novel sampling theorem on the sphere. *IEEE Transactions on Signal Processing*, 59 (12):5876–5887, December 2011. ISSN 1941-0476. doi: 10.1109/tsp.2011.2166394. URL http://dx.doi.org/10.1109/TSP.2011.2166394.

Muckley, M. J., Stern, R., Murrell, T., and Knoll, F. TorchKbNufft: A high-level, hardware-agnostic non-uniform fast Fourier transform. In *ISMRM Workshop on Data Sampling & Image Reconstruction*, 2020. Source code available at https://github.com/mmuckley/torchkbnufft.

Pan, V. Y. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Birkhauser/Springer, Boston/New York, 2001.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, p., Kashinath, K., and Anandkumar, A. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Perera, S. M., Lingsch, L., Madanayake, A., Mandal, S., and Mastronardi, N. A fast dvm algorithm for wideband time-delay multi-beam beamformers. *the IEEE Transactions on Signal Processing*, 70:5913–5925, 2022. doi: 10.1109/TSP.2022.3231182.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *CoRR*, abs/2010.03409, 2020. URL https://arxiv.org/abs/2010.03409.

Prasthofer, M., De Ryck, T., and Mishra, S. Variable input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022.

Quarteroni, A. and Valli, A. *Numerical approximation of Partial differential equations*, volume 23. Springer, 1994.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Raonić, B., Molinaro, R., Rohner, T., Mishra, S., and de Bezenac, E. Convolutional neural operators. *arXiv preprint arXiv:2302.01178*, 2023.

Ruiz-Antolin, D. and Townsend, A. A nonuniform fast fourier transform based on low rank approximation. *SIAM Journal on Scientific Computing*, 40(1):A529–A547, 2018.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks. *CoRR*, abs/2002.09405, 2020. URL https://arxiv.org/abs/2002.09405.

Selva, J. Efficient type-4 and type-5 non-uniform fft methods in the one-dimensional case. *IET Signal Processing*, 12(1):74–81, 2018.

Shih, Y., Wright, G., Andén, J., Blaschke, J., and Barnett, A. H. cufinufft: a load-balanced gpu library for general-purpose nonuniform ffts, 2021.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tmIiMPl4IPa.

Wandel, N., Weinmann, M., and Klein, R. Unsupervised deep learning of incompressible fluid dynamics. *CoRR*, abs/2006.08762, 2020. URL https://arxiv.org/abs/2006.08762.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno – an enhanced fourier neural operator-based deep-learning model for multiphase flow, 2022.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models, 2023.

Zhu, Y. and Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 336:415–447, 2018.

Supplementary Material for:
Fourier-Based Neural Operators on Arbitrary Domains

A. Technical Details

A.1. The FFT Signal Flow Graph.

The signal flow graph provides a graphical representation of the FFT algorithm. Radix-2, recursive algorithms such as the FFT are highly dependent on the positions of the data. Violating this principle makes using similar approaches difficult, as the processes can not be easily "untangled" in some sense.

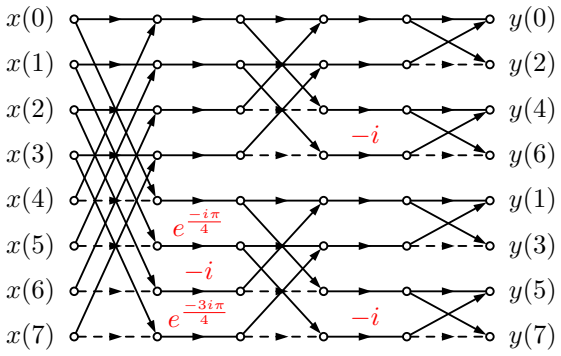


Figure 4: The 8-point fast Fourier transform signal flow graph. x and y represent the signal in the physical and Fourier domain, respectively. Dashed lines represent a multiplication by -1, red elements denote a multiplication by that factor, and converging arrows represent a sum.

A.2. Transformations on the 2D Lattice

The 2D FFT is calculated as two 1D FFTs along each axis, as

$$X = \mathcal{F}x\mathcal{F}^T, \tag{11}$$

where $\mathcal{F}$ is the FFT, transforming an array $x \in \mathbb{C}^{N_0 \times N_1}$ to an array $X \in \mathbb{C}^{N_0 \times N_1}$. N_0 and N_1 correspond to the number of points along the first and second spatial axes, respectively.

We may mirror this transformation on any 2D lattice, i.e, the tensor product of 1D point distributions along each axis (see **SM** Figure 1). To do this, we construct two matrices, $\mathbf{V}_0 \in \mathbb{C}^{m \times N_0}$ and $\mathbf{V}_1 \in \mathbb{C}^{N_1 \times m}$, corresponding to the sample points of data points along the first and second axis, respectively.

$$X = \mathbf{V}_1 x \mathbf{V}_2^T. \tag{12}$$

The operational complexity of this approach is equivalent to that of Section 2; however, the matrices contain only $m\times$

$(N_0 + N_1)$ values, while those in the general case contain $m^2 \times (N_0 \times N_1)$ values. Therefore, this approach may be more efficient depending on the distribution of sample points.

A.3. On Computational Complexity of the Direct Evaluation of Fourier Transforms.

The matrix-vector product, as described in Section 2 of the main text, is broken down into a vector with N components via inner products of vectors, requiring $2N$ real multiplications and $2(N-1)$ real additions having a real-valued input, a total of $4N^2 - 2n$ flops. In the implementation, the number of rows is reduced from N to a constant m s.t. $m << N$, as determined by the number of modes chosen for a given problem. Thus, for the 1D case, the total number of flops is $4Nm - 4N$, therefore only growing linearly with the problem size. For the D-dimensional nonequispaced data, we construct a matrix with size N columns but the number of rows $m_{total} = (m_1)(m_2)\dots(m_D)$, where m_j is the number of modes taken along the j^{th} spatial dimension. Thus, the complexity of the transform using the direct evaluation is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(m_{total}N)$. Finally, we note here that the existing non-uniform FFT has the complexity of order $\mathcal{O}(RN\,log(N))$, where R is based on diagonally scaled FFT. Thus, applying the existing non-uniform FFT on FNO won't reduce the complexity of the direct evaluation on the neural operator with the complexity reduction of order $\mathcal{O}(m_{total}N)$.

For the extension to the spherical harmonics, we pre-compute the Legendre polynomials over N points as an evaluation problem with $\mathcal{O}(N)$ complexity. Once these are pre-computed, we call these polynomials to calculate the spherical harmonics with complexity $\mathcal{O}(l^2)$, e.g. the order l is calculated using an odd number of harmonics from 1 to the order l. Thus, the associated spherical harmonics can be computed using $\mathcal{O}(N\,l^2)$ complexity, where $l << N$. Finally, arranging harmonics into the truncated matrix form followed by the matrix-vector products into vector forms (as described above) reduces the complexity to $\mathcal{O}(Nl^2)$ as opposed to $\mathcal{O}(N^2\,l^3)$.

The question may arise; *Can the number of modes really be said to be constant with respect to the number of points, as more points may be used for more complex problems, necessitating more modes?* A priori, it is not possible to say how many modes must be selected for a problem of given complexity or resolution - these must be determined through model selection, varying the number of modes, and choosing the model that minimizes the error over the validation set. This model selection process has revealed that there is often a point where increasing the number of modes results in worse performance, i.e. increasing the number of modes will not automatically result in increased performance. With regard to the sharp features and large gradients that may arise in complex sets of PDEs or high-resolution data, the lifting layer and the skip connections of the model architecture are much more capable of resolving such features than the Fourier layers. The Fourier layers serve to propagate information throughout the domain. Thus, we maintain the position that the number of modes is fixed with respect to the number of points.

B. Experimental Details

B.1. Point Distributions

We investigate both a uniform and a *contracting-expanding* distribution to help lay the foundation for the proposed method. These distributions are visualized in Figure 5.

For the two dimensional experiments, we investigate lattices as well as random or structured data on a point cloud, simplified illustrations of which are provided in Figure 1. A lattice with a nonuniform distribution along one axis (*Shear Layer*) and a lattice with a nonuniform distribution along both axes (*Surface-Level Specific Humidity*) are investigated and visualized in Figures 6 and 7. Point cloud data with varying geometries are investigated in the *Airfoil* and *Elasticity* experiments. We provide visualizations of several airfoil point clouds in Figure 8.

Finally, we investigate the performance of spherical harmonics models on a random distribution over the sphere. The original point cloud, the random points, and the grid generated by interpolating from the random points are shown in Figure 9.

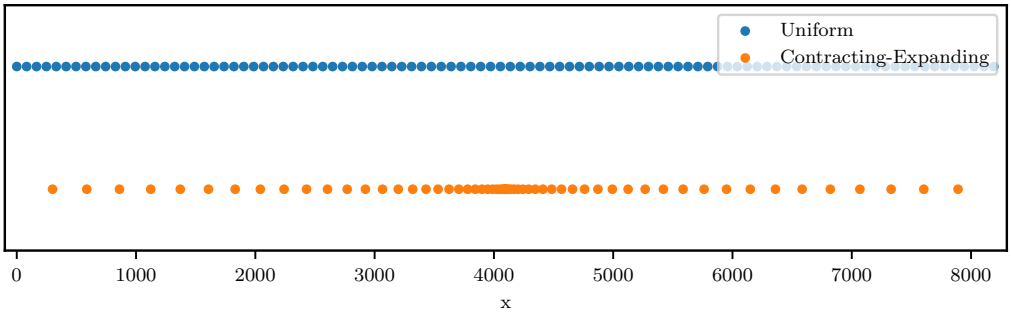


Figure 5: Point distributions used in the Burgers' equation experiments. Data is selected from the uniform distribution to construct the contracting-expanding distribution and random distribution. The space between points in the contracting-expanding distribution grows from a point in both directions according to a geometric distribution.

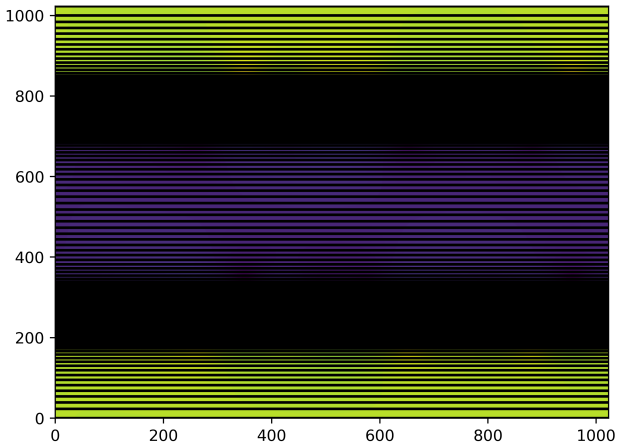
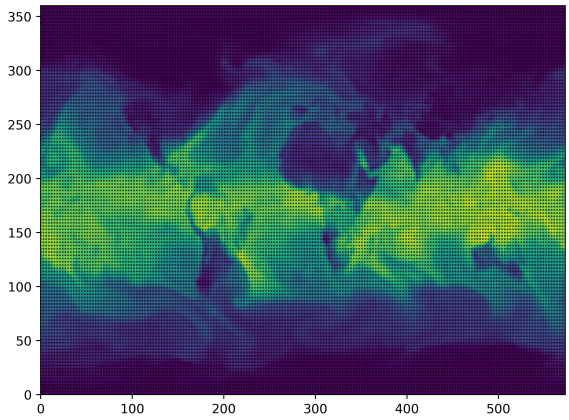
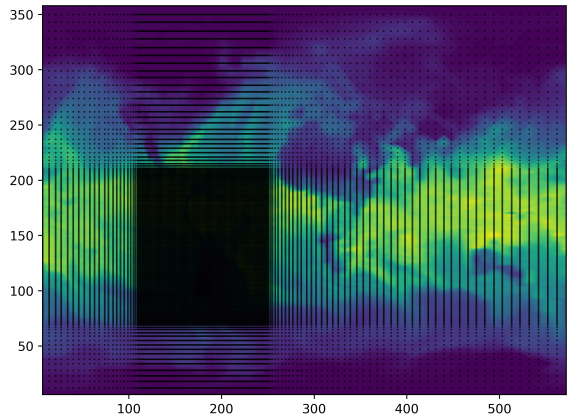


Figure 6: Nonequispaced lattice for the shear layer problem. Sampling is dense close the interface region, smoothly becoming sparse further from this region.

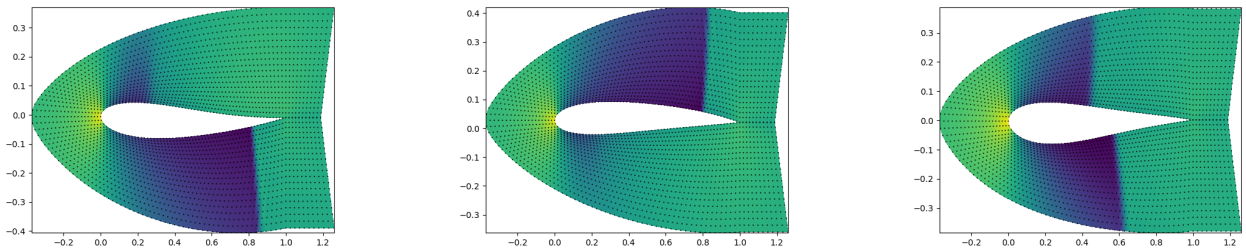


(a) FNO point distribution. The points displayed in this image have been subsampled from the original distribution to maintain clarity in the figure.



(b) Nonequispaced lattice point distribution. A densely sampled region is located over South America and the lattice becomes more sparse further from this region.

Figure 7: Distributions used within the surface-level specific humidity experiment.



(a) (b) (c)

Figure 8: Several point distributions around airfoils and their associated pressure distributions.

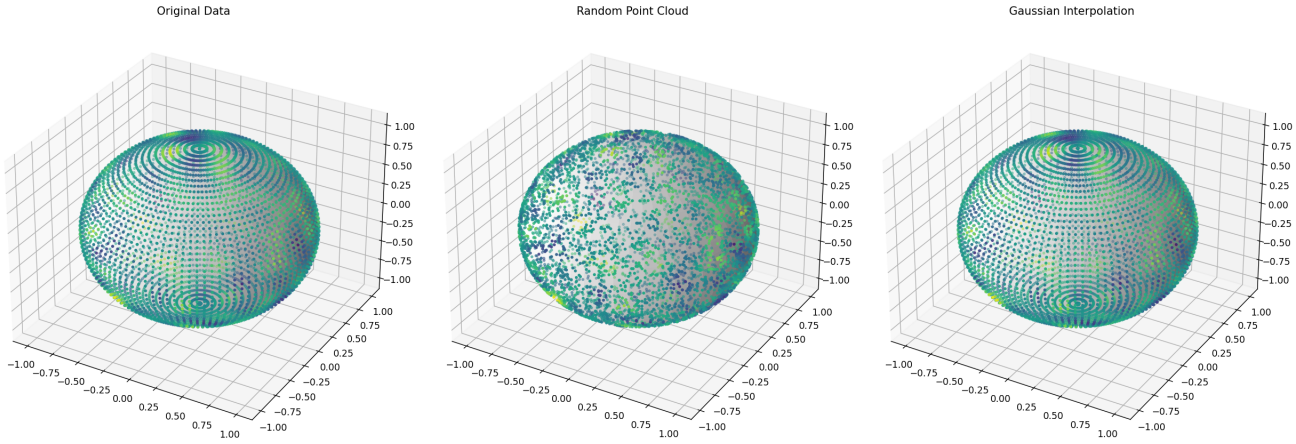


Figure 9: The original distribution of points (left) is sampled from at locations chosen randomly along the surface of a sphere (middle). To compare with baselines, which require a uniform distribution, we use a radial-basis function interpolation scheme with a Gaussian kernel and a variance of 0.1 to interpolate from the random points back up to a grid (right) as well as a linear interpolation schemer.

B.2. Additional Training Details and Results.

We use 16 different parameters to make predictions for the *Surface-Level Specific Humidity* experiments. These are listed in this subsection in Table 2. We also show the predictions of several models for several experiments in Figure 10. Table 3 provides information on the distributions of the minimum median relative L1 test error given different initial seeds. We also provide histograms showing the error for all test samples in Figures 11 and 12. We present additional results for the training times as a function of the number of modes for the Burgers, 2D Specific Humidity, 2D Airfoil, and Spherical Shallow Water experiments in Table 4. Finally, we include Table 5 to show the sizes of the various models.

B.3. Generalization Capabilities for the SWE Experiments.

First, we provide the errors of the FNO and SFNO on their respective grids for the Shallow Water Equations experiments on the Sphere in Table 6. We find that the training times are approximately 30 seconds, twice that of the DSE approaches on the unstructured data, and the errors over the collocation points are equivalent to those of the DSE approach.

We also present the results for the SFNO-DSE on new sets of points which are unseen at training time. We vary the number of points, providing a sensitivity analysis. The results, in Table 7, show that the error remains constant.

Table 2: Inputs for the surface level specific humidity predictions.

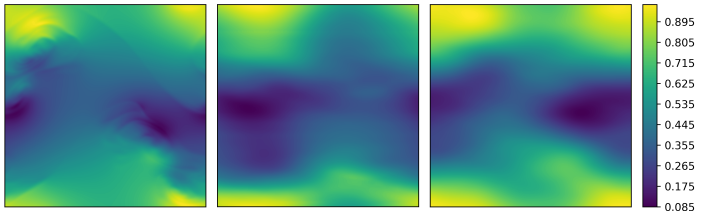
Acronym in MERRA-2	**Input**	**Units**
CDH	heat exchange coefficient	$\frac{kg}{m^2 s}$
CDQ	moisture exchange coefficient	$\frac{kg}{m^2 s}$
EFLUX	total latent energy flux	$\frac{W}{m^2}$
EVAP	evaporation from turbulence	$\frac{kg}{m^2 s}$
FRCAN	areal fraction of anvil showers	1
FRCCN	areal fraction of convective showe	1
FRCLS	areal fraction of large scale show	1
HLML	surface level height	m
QLML	surface level specific humidity	1
QSTAR	surface moisture scale	$\frac{kg}{kg}$
SPEED	surface wind speed	$\frac{m}{s^2}$
TAUX	eastward surface stress	$\frac{N}{m^2}$
TAUY	northward surface stress	$\frac{N}{m^2}$
TLML	surface air temperature	K
ULML	surface eastward wind	$\frac{m}{s}$
VLML	surface northward wind	$\frac{m}{s}$

Table 3: The mean and standard deviation *of the median test errors* among 10 models trained with different random seeds at initialization of the structured proposed method for the *Burgers'* experiments.

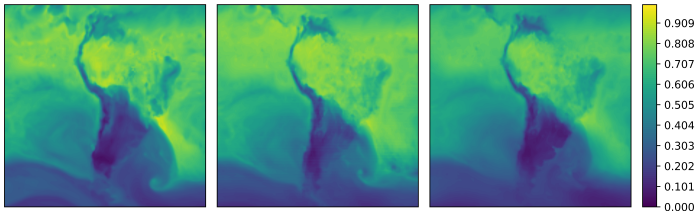
Data Distribution	Model	Method	**Mean±Std**
Uniform	FNO	FFT	$0.0650 \pm 0.0051\%$
Uniform	FNO	DSE	$0.0653 \pm 0.0074\%$
Contracting-Expanding	FNO	FFT+Interpolation	$0.2076 \pm 0.0087\%$
Contracting-Expanding	FNO	DSE	$0.1973 \pm 0.0069\%$

Table 4: Average training times for several experiments as the number of modes is varied.

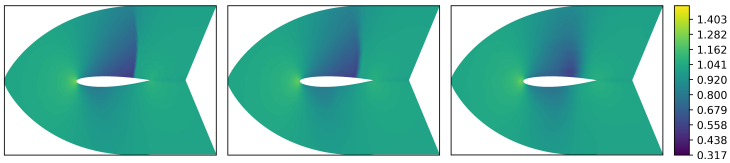
Modes	Burgers		Humidity		Airfoil		SWE	
	FFT	DSE	FFT	DSE	Geo-FNO	Direct	SFNO	DSE
8	0.93	0.81	15.3	12.3	3.5	1.9	72.6	17.1
10	0.94	0.81	15.3	12.1	4.4	2.5	72.3	17.1
12	0.94	0.81	15.0	12.6	5.6	3.3	72.0	17.6
14	0.94	0.81	15.2	12.6	6.9	4.0	72.1	17.3
16	0.95	0.82	15.2	12.6	8.5	5.2	72.1	17.3
18	0.95	0.82	15.2	12.6	10.4	7.0	72.3	18.1
20	0.95	0.82	15.4	12.8	12.5	9.1	72.4	17.4
22	0.95	0.82	15.5	13.1	14.8	10.3	72.8	18.6
24	0.95	0.83	15.3	12.9	17.2	11.8	72.1	17.8
26	0.95	0.83	15.2	13.2	20.1	14.1	72.6	18.1
28	0.94	0.83	15.0	12.9	–	16.0	73.0	19.5
30	0.94	0.83	15.4	13.1	–	18.0	72.5	18.5
32	0.94	0.83	15.2	13.1	–	20.7	72.3	20.71



(a) Horizontal velocity for the Shear Layer experiment

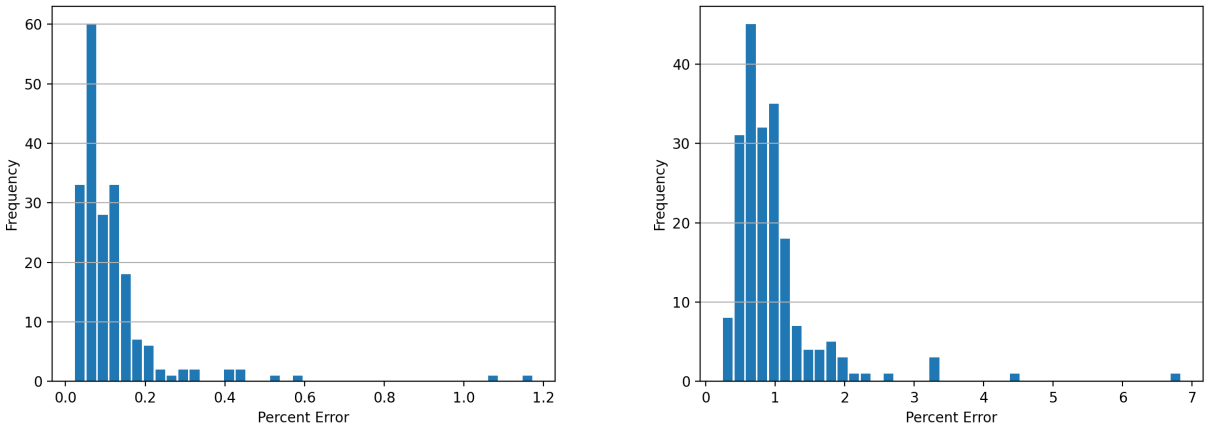


(b) Surface-level specific humidity over South America.



(c) Pressure distributions around an airfoil.

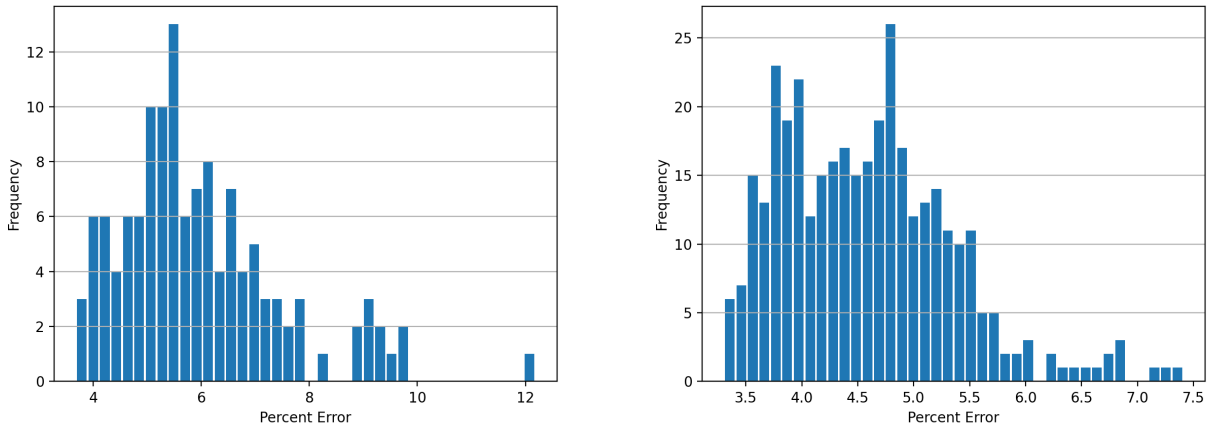
Figure 10: These figures display examples of the ground truth, the target which the FNO-DSE, FNO, or Geo-FNO attempt to match. Left: Ground Truth. Center: FNO-DSE Right: FNO for (a) and (b) and Geo-FNO for (c).



(a) *Uniform*.

(b) *Contracting-Expanding*.

Figure 11: Histograms displaying the distribution of test errors for the different point distributions of the *Burgers'* numerical experiments using the proposed method.



(a) *Shear Layer*.

(b) *Surface-Level Specific Humidity*.

Figure 12: Histograms displaying the distribution of test errors for the two-dimensional numerical experiments on the lattice using the proposed method with the FNO model.

Table 5: Model sizes in terms of the number of parameters. Within an experiment, the models are constructed to have similarly sized convolution and spectral convolution layers. This results in the FFNO often having an order of magnitude fewer parameters, a key contribution outlined by (Tran et al., 2023). The UFNO models have the most parameters due to the extra U-Net layers, and the Geometric Layer results in a slightly higher number of parameters as well.

Model	Method	No. Parameters	No. Modes
1D: Burgers' Equation			
Equispaced Distribution:			
FNO	DSE	287425	16
	FFT	287425	16
Contracting-Expanding Distribution:			
FNO	DSE	287425	16
	Interpolation	287425	16
2D Lattice: Shear Layer			
FNO	DSE	3162881	20
	Full Grid	3162881	20
UFNO	DSE	3750401	20
	Full Grid	3750401	20
FFNO	DSE	423041	20
	Full Grid	423041	20
2D Lattice: Specific Humidity			
FNO	DSE	8398049	32
	Full Grid	8398049	32
UFNO	DSE	8691553	32
	Full Grid	8691553	32
FFNO	DSE	537697	32
	Full Grid	537697	32
2D Point Cloud: Flow Past Airfoil			
FNO	DSE	1188577	12
	Geometric Layer	1250339	12
UFNO	DSE	1482081	12
	Geometric Layer	1840163	12
FFNO	DSE	500673	14
	Geometric Layer	526787	14
2D Point Cloud: Elasticity			
FNO	DSE	1484289	12
	Geometric Layer	1546403	12
UFNO	DSE	1778049	12
	Geometric Layer	1840163	12
FFNO	DSE	526787	14
	Geometric Layer	533441	14
Random Spherical Point Cloud: Shallow Water Equations			
SFNO	DSE	39749251	$l = 22$
	Gaussian Interpolation	39201536	$l = 22$
	Linear Interpolation	39201536	$l = 22$
FNO	DSE	31978563	20
	Gaussian Interpolation	39201536	20
	Linear Interpolation	39201536	20

Method	Training Time (per epoch)	Error over Collocation Points
SFNO	30s	3.60%
FNO	32s	5.74%

Table 6: Accuracy of the SFNO and FNO for the Spherical Shallow Water Equations when trained on the original 256×512 grid which has been sub-sampled to a 51×102 grid. At inference time, we calculate the error over the collocation points, i.e. those used in the random distribution. These models achieve similar results to the proposed approach when the data has not undergone interpolation.

Approximate No. Points	Error
4000	6.36%
8000	5.64%
15000	5.44%
29000	5.39%
50000	5.43%

Table 7: Accuracy of the SFNO-DSE on test sets of the Spherical Shallow Water Equations experiment with new random distributions of varying size. Each test set consists of 64 samples. The model was trained on data sets of approximately 5000 points. All point distributions and test samples used in this evaluation were not seen during training. The results show that the proposed method is able to generalize to new point distributions of varying resolution.