Spherical Fourier Neural Operators: Learning Stable Dynamics on the Sphere

Boris Bonev [1] **Thorsten Kurth** [1] **Christian Hundt** [1] **Jaideep Pathak** [1] **Maximilian Baust** [1] **Karthik Kashinath** [1] **Anima Anandkumar** [1 2]

Abstract

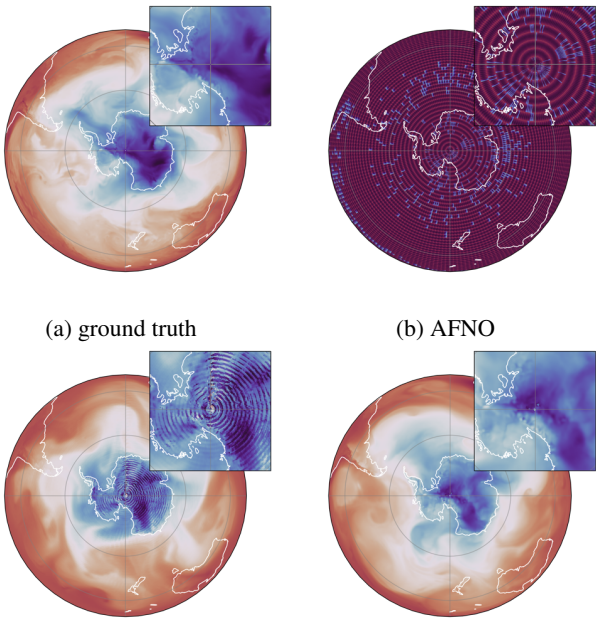
Fourier Neural Operators (FNOs) have proven to be an efficient and effective method for resolution-independent operator learning in a broad variety of application areas across scientific machine learning. A key reason for their success is their ability to accurately model long-range dependencies in spatio-temporal data by learning global convolutions in a computationally efficient manner. To this end, FNOs rely on the discrete Fourier transform (DFT), however, DFTs cause visual and spectral artifacts as well as pronounced dissipation when learning operators in spherical coordinates by incorrectly assuming flat geometry. To overcome this limitation, we generalize FNOs on the sphere, introducing Spherical FNOs (SFNOs) for learning operators on spherical geometries. We apply SFNOs to forecasting atmospheric dynamics, and demonstrate stable autoregressive rollouts for a year of simulated time (1,460 steps), while retaining physically plausible dynamics. The SFNO has important implications for machine learning-based simulation of climate dynamics that could eventually help accelerate our response to climate change.

1. Introduction

Climate change is arguably one of the greatest challenges facing humanity today. Modeling Earth's complex weather and climate accurately, and in a computationally efficient manner, has massive implications for science and society across the enterprise of climate prediction, mitigation, and adaptation.

[1]NVIDIA Corp., Santa Clara, USA [2]Caltech, Pasadena, USA. Correspondence to: Boris Bonev <bbonev@nvidia.com>, Thorsten Kurth <tkurth@nvidia.com>, Christian Hundt <chundt@nvidia.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).



(a) ground truth (b) AFNO

(c) FNO, non-linear (d) SFNO, linear

Figure 1. Qualitative comparison of temperature predictions (`t850`) over Antarctica at 4380h (730 autoregressive steps). The SFNO shows no visible artifacts even after six-month-long rollouts. Models which use FFT and do not incorporate spherical geometry are not stable for long rollouts compared to SFNO. The AFNO model breaks down early and shows large visible artifacts everywhere. In the non-linear FNO model, artifacts are less pronounced but increase in magnitude with time and towards the poles.

Weather and climate modeling has traditionally relied on principled physics- and process-based numerical simulations that solve the partial differential equations (PDEs) governing the fluid dynamics, thermodynamics, and other physics of the Earth system. These equations are discretized and solved on a grid, but the wide range of spatial and temporal scales, as well as complex nonlinear interactions across these scales, necessitate fine grids and high resolution making these computations extremely expensive.

Machine learning (ML) provides alternative approaches to

modeling weather and climate, and more generally, spatio-temporal dynamics, by describing the time evolution of the system as a learned transition map between states of the time-discretized physical system exclusively from raw data. While this enables a unified treatment of the full system, the physics is deduced from data alone without imposing the strong inductive bias of the aforementioned physics-based models. Hence, purely data-driven ML-based methods have struggled to faithfully represent the dynamics of spatio-temporal physical systems, especially those with long-range correlations in space and time.

Fourier Neural Operators (Li et al., 2020) and their variants (Guibas et al., 2021; Wen et al., 2022; Rahman et al., 2022; Kovachki et al., 2023; Kossaifi et al., 2023) possess the advantage of learning mappings between function spaces, which act globally on the entire domain. In contrast, standard neural networks, such as convolutional neural networks and vision transformers, learn on a fixed discretized grid and fail to capture the fine scales of multi-scale systems. Further, the highly optimized Fast Fourier Transform (FFT) in FNO allows for modeling global, long-range interactions in quasi-linear time. Thus, in addition to being discretization invariant, FNO is also computationally more efficient compared to standard vision transformers, which have quadratic time complexity, and more effective in capturing global dependencies compared to convolutional neural networks, graph neural networks, and other local models (McCormick, 1987; Falk et al., 2019). However, a drawback of FNOs is that FFT is defined on an Euclidean domain leading to incorrect identification of the north and south pole as well as incorrect longitudinal periodicity on the two-sphere S^2.

Our approach: We extend the FNO approach to respect the geometry of S^2 and its associated symmetries. To do so, we utilize the generalized Fourier transform, which projects functions defined on compact Riemannian manifolds $\mathcal{M}$ onto eigenfunctions of the corresponding Laplace-Beltrami operator, which form an orthogonal basis of the Hilbert space $L^2(\mathcal{M})$. In the spherical setting $\mathcal{M} = S^2$, these are the Spherical Harmonics, and the Spherical Harmonic Transform (SHT) generalizes the Fourier transform (Driscoll & Healy, 1994). We formulate the SFNO in a manner consistent with the convolution theorem on the sphere. The resulting operator satisfies equivariance properties, such that rotating the input to the operator commutes with the operator itself. Translational or rotational equivariance also motivates the formulation of physical theories, as we do not expect the physical laws to change with a changed frame of reference. This rotational equivariance is, therefore, a strong inductive bias to the learned operator.

In addition, the SFNO retains the favorable properties of FNOs. The learned operators are grid-invariant, since the operations outside of the Fourier/Spherical Harmonic layers act point-wise on the spatial domain. This allows the model to be re-trained at different configurations and resolutions, and even to change the resolution at inference time (i.e., zero-shot super-resolution or interpolation).

The proposed method is applied to the Earth Reanalysis 5 dataset (ERA5) (Hersbach et al., 2020), one of the best estimates of Earth's historical weather and climate over the period 1950-present, and to the rotating Spherical Shallow Water Equations (SWE), which are commonly used to model geophysical fluid dynamical phenomena (Nair et al., 2005; Bonev et al., 2018). In addition to accurate predictions, our method leads to greatly increased long-term stability, with autoregressive inference remaining stable for over one year (1,460 steps) as opposed to 25 days (100 steps) with a comparable, FFT-based method (FNO). As each autoregressive step takes around 500ms on a NVIDIA A6000 GPU, i.e., less than 13 minutes for a year-long simulation, these developments open the door to long-range ensemble inference and uncertainty quantification, well beyond weather timescales to subseasonal-to-seasonal (S2S) prediction and potentially, towards climate prediction.

Key contributions:

- **A novel SFNO equivariant architecture** for modeling nonlinear chaotic dynamical systems on the sphere.
- Theoretical extension of FNOs to spherical geometry, with the desirable properties of **equivariance** while retaining **grid-invariance**.
- A demonstration of exceptional long-term **stability** of the auto-regressive map F, on year-long rollouts, while observing plausible dynamics of the predicted physical quantities.
- `torch-harmonics`, an efficient and differentiable implementation of the spherical harmonics transform in `PyTorch`[1], enabling **scalable model parallelism**s.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 presents the theoretical foundations used during SFNO design. The network topology design is then laid out in Section 4.2. Section 5 documents the experimental setup and training infrastructure. Section 6 concludes the paper.

2. Related Work and Broader Context

ML-based weather and climate modeling is less than five years old, yet has seen a massive surge in capability thanks to advancements in deep learning (Yuval & O'Gorman,

[1] `torch-harmonics` and our implementation of the SFNO are available to the public at https://github.com/NVIDIA/torch-harmonics

2020; Pathak et al., 2022; Bi et al., 2022; Lam et al., 2022). Relatively simple architectures like CNN-based encoder-decoder architectures (Scher & Messori, 2019) have been applied to two-dimensional latitude-longitude grids. Weyn et al. (2020; 2021) noticed the limitations of this representation when treating the poles and proposed an extension based on the equiangular gnomonic cubed sphere. Pathak et al. (2022) combined a Fourier-transform-based token-mixing scheme (Guibas et al., 2021) with a vision transformer (ViT) backbone (Dosovitskiy et al., 2020) and developed FourCastNet (FCN) to model long-range dependencies. More recently, Bi et al. (2022) developed another ViT-based approach they call Pangu-Weather, which relies on an encoder-decoder architecture and demonstrated that ML-based methods are able to compete with and even outperform classical physics-based numerical methods for numerical weather prediction (NWP). Most recently, GraphCast, another approach based on graph neural networks (Lam et al., 2022) developed the first approach to work on a spherical geometry via a multi-scale mesh representation and demonstrated superior predictive performance on weather forecasting. Except Weyn et al. (2021), none of the above-mentioned models were stable beyond a few weeks of the autoregressive rollout. Weyn et al. (2021), however, were limited in their predictive performance and operated on a resolution that was too low to provide useful information on the regional and local scales at which extreme weather impacts society. Therefore, there remains a strong need for ML-based high-resolution stable long-term forecasts of atmospheric phenomena.

Concepts such as equivariance, symmetry, and learned representations in spectral domains exist in many different contexts. An exhaustive review of these is beyond the scope of this paper and we refer the reader to the excellent overview on geometric deep learning by Bronstein et al. (2021).

Introducing inductive biases to regularize machine learning problems have a long history in computer vision and machine learning. The inductive bias of convolutional layers to achieve translation invariance resulted in a decade of the predominance of convolutional neural networks (CNNs) (Lecun & Bengio, 1995). Nowadays, ViTs employ self-attention as the mechanism to exploit the Symmetric group $G = S_n$ of permutations acting on n input tokens as a less restrictive, yet, effective symmetry to relax the inductive bias of translational equivariance in CNNs (Bertasius et al., 2021). **Equivariance** has gained significant attention in the past few years, including group equivariant CNNs (Cohen & Welling, 2016) and their spherical counterparts (Cohen et al., 2018; Ocampo et al., 2022).

We provide an equivariant extension of Fourier Neural Operators to the sphere, based on a convolution theorem on the sphere.

3. Background

3.1. Problem Setting

We model the Earth's atmosphere as a dynamical system, where its state at time t and position x is represented by a N-dimensional vector $u(x,t) : S^2 \times [0,T) \to \mathbb{R}^N$. Each of these variables represents a physical quantity of interest, such as pressure, wind velocity, humidity, etc. The system is observed only partially, meaning that there might be hidden variables that influence the dynamics, which are not contained in this description. Moreover, the system is observed only at discrete times $t_n \in \{t_0, t_1, \dots\}$, and at discrete positions $x_i \in \{x_1, x_2, \dots, x_N\}$. For simplicity, we assume that there exists a deterministic mapping

$$u_{n+1} = F[u_n, t_n], \tag{1}$$

which maps the discrete state vector u_n at time t_n to the state u_{n+1}. Our aim is to learn this map from data. This is in contrast to the classical approach using PDEs, where this map is typically inferred from first principles by discretization and numerical integration. Carefully crafted numerical schemes often conserve certain symmetry properties of the underlying PDE, which motivate the goal of formulating an equivariant ML method.

3.2. Fourier Neural Operators

Our construction of equivariant mappings between function spaces on the sphere is an extension of the FNO framework. FNO learns a resolution-independent representation using a global convolution kernel κ

$$\mathcal{K}\big[u\big](x) = \int_{\mathcal{M}} \kappa(x-y) \cdot u(y)\,\mathrm{d}y \qquad \forall x \in \mathcal{M}\,, \tag{2}$$

which can be rewritten in terms of the convolution theorem

$$\mathcal{K}\big[u\big](x) = \mathcal{F}^{-1}\big[\mathcal{F}[\kappa] \cdot \mathcal{F}[u]\big](x) \qquad \forall x \in \mathcal{M}\,. \tag{3}$$

The continuous Fourier transform $\mathcal{F}$ can be expressed as the DFT when sampling the finite domain $\mathcal{M}$ using a uniform grid. Hence, FNOs allow for expressing long-range dependencies using global convolutions while employing an efficient implementation of the FFT, guaranteeing log-linear asymptotic time complexity.

The Spherical FNO (SFNO) layer extends this approach to adapt the Fourier transform while respecting the symmetry of the underlying manifold.

3.3. Incorporating Symmetries on General Manifolds

Symmetries are of crucial importance for the analysis of physical systems and historically played a seminal role in the theoretical formulation of natural phenomena long before the first experimental evidence was gathered for their

existence. Group theory-driven breakthroughs include the systematic classification of lattices in crystallography (Conway & Sloane, 1988), the postulation of novel particles such as quarks (Gell-Mann, 1961), and the standard model of particle physics (Workman & Others, 2022). Respecting the intrinsic geometry by means of Lie group actions $\Phi : G \times \mathcal{M} \to \mathcal{M}$, $(g, x) \mapsto \Phi(g, x)$ acting on the configuration space manifold $\mathcal{M}$ of spatial coordinates allows for: (i) the identification of conserved quantities by Noether's theorem; and (ii) the elimination of redundant variables, which usually leads to a decoupling or simplification of the associated equations of motion.

In ML, symmetries act as hard constraints limiting the search space to physically meaningful submanifolds/orbifolds, which may speed up optimization as well as improve the robustness of the solution through explicit regularization. A prominent example is convolutional layers (Fukushima, 1980), which constrain the space of trainable parameters to the inner product of a filter k being translated along the orbit parametrized by the group action $\Phi(x, y) := y - x$ of the translation group $G = (\mathbb{R}, +)$. This leads to the well-known expression for convolution.[2]

$$\begin{aligned}\big(\kappa \star u\big)(x) &= \int \kappa\big(\Phi(x, y)\big) \cdot u(y)\, \mathrm{d}y \\ &= \int \kappa(y - x) \cdot u(y)\, \mathrm{d}y \quad \forall x \in \mathbb{R}. \end{aligned} \tag{4}$$

While the Abelian translation group $G = (\mathbb{R}^n, +)$ naturally acts on flat $\mathbb{R}^n$ via plain vector addition, we cannot globally define a convolution operation on curved manifolds by means of axis-oriented translations.

Hence, for physical systems on general manifolds $\mathcal{M}$, we extend the Fourier transform as well as the operations acting on the feature representations to respect their geometric constraints. On compact Riemannian manifolds, the generalized Fourier transform can be understood as the map that decomposes functions in terms of the eigenfunctions of the Laplace-Beltrami associated with the underlying manifold $\mathcal{M}$. In general, this view neglects the group structure if $\mathcal{M}$ is a Lie group.

For all practical purposes, we are interested in dynamical systems formulated on the sphere $\mathcal{M} = S^2 \subset \mathbb{R}^3$. The canonical choice on the sphere is rotational group actions stemming from the set of orientation-preserving isometries of the scalar product in $\mathbb{R}^3$, namely the special orthogonal group $SO(3)$. We consider the Hilbert space of square-integrable functions with respect to the Lebesgue measure defined on the sphere $L^2(S^2)$. We define the (partial map of a) group action for an arbitrary but fixed rotation $\Phi_R : L^2(S^2) \mapsto L^2(S^2)$ onto a square-integrable function $u \in L^2(S^2)$ as the result of transforming the spatial coordinates with the corresponding passive rotation $R^{-1} \in SO(3)$.

$$\Phi_R[u](x) := u(R^{-1}x) \quad \forall R \in SO(3) \tag{5}$$

In other words, in order to rotate an image on the sphere, we rotate its pre-image in the opposite direction. This is analogous to the planar case where a function is shifted along the coordinate axes using the translation group $G = (\mathbb{R}^n, +)$. In the following, we are interested in mappings $F : L^2(S^2) \to L^2(S^2)$, that are compatible with the aforementioned group actions on the Hilbert space. We call a transformation F equivariant with respect to two $SO(3)$ group actions Φ_R, Ψ_R both acting onto $L^2(S^2)$, iff

$$\Psi_R\big[F[u]\big] = F\big[\Phi_R[u]\big] \quad \forall R \in SO(3) \tag{6}$$

Simply put, it does not matter whether one initially rotates the coordinate system followed by a subsequent transformation on the function space or vice versa. For simplicity, we stretch the notation and drop the explicit group actions Φ, Ψ and identify them with the multiplication of group elements. Equivariance then reads: $F \circ R = R \circ F$ for all $R \in SO(3)$.

3.4. Fourier Transform on the Sphere

In flat geometry, the Fourier transformation is a change of basis by projecting a function $u \in L^2(\mathbb{R}^n)$ onto planar waves $b_k(x) = \exp\big(i\langle k, x\rangle\big)$ effectively encoding translation equivariance. In a general setting, the set of basis functions can be obtained from the eigenfunctions of the Laplace-Beltrami operator. In the following, we will provide an explicit construction for spherical geometry.

On S^2, these are the spherical harmonics (Abramowitz et al., 1964) defined as

$$Y_l^m(\theta, \varphi) := (-1)^m c_l^m P_l^m(\cos\theta) e^{\mathrm{i}m\varphi}, \tag{7}$$

which form an orthogonal basis of $L^2(S^2)$. P_l^m and c_l^m denote the associated Legendre polynomials and normalization factors[3]. More importantly, among all the possible bases of $L^2(S^2)$, the spherical harmonics uniquely exploit the symmetries of the sphere. Applying rotations Ψ_R to Y_l^m of degree l will result in a linear combination of Y_l^m with the same degree l and $-l \leq m \leq l$.

On the sphere, the corresponding Fourier transformation is the decomposition of the function space $L^2(S^2)$ into minimal subspaces invariant under all the possible rotations

[2] While convolutional layers typically implement cross-correlation, we choose to discuss convolutions for the sake of simplicity.

[3] For a detailed introduction of the spherical harmonics see Appendix A.2.

in $SO(3)$. The decomposition onto the basis then reads

$$u(\theta,\phi) = \sum_{l\in\mathbb{N}} \sum_{|m|\leq l} \hat{u}(l,m) Y_l^m(\theta,\phi)$$
$$\hat{u}(l,m) = \int_{S^2} \overline{Y_l^m} \cdot u \, \mathrm{d}\Omega \,, \tag{8}$$

where $\mathrm{d}\Omega = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$ is the volume form of the sphere. $\mathcal{F} : u \to \hat{u}$, maps functions in $u \in L^2(S^2)$ to the harmonic coefficients $\hat{u}(l,m)$. We call this map Spherical Harmonic Transform (SHT), or alternatively, generalized Fourier transform (Driscoll & Healy, 1994).

The Fourier transform decomposes a Hilbert space into basis functions associated to symmetry operations on the underlying manifold. For the translation group $G = (\mathbb{R}^2, +)$, this basis is given by tensor products of plane waves $b_{kl}(x,y) = \exp(ikx) \otimes \exp(ily)$ since translations along the axes commute with each other. For the sphere, the situation is more complicated as the generators of rotations do not commute in general. This is reflected by the fact that the associated spherical basis functions do not factor into a tensor product structure.

4. Spherical Fourier Neural Operator

4.1. Convolutions on the Sphere

We can now generalize the FNO naively by replacing the Fourier transforms in (3) with the SHT. While the SHT generalizes the Fourier transform on S^2, this ignores that (3) is derived from a convolution theorem, however, and ignores the symmetries inherent to the sphere. Instead, we seek a formulation motivated by a convolution operator to obtain an equivariant formulation. We introduce the spherical convolution

$$(\kappa \star u)(x) = \int_{R\in SO(3)} \kappa(Rn) \cdot u(R^{-1}x) \mathrm{d}R, \; \forall x \in S^2. \tag{9}$$

This definition applies the rotation R to the northpole $n = (0,0,1)^T$, which makes the function $\kappa \in L^2(S^2)$ compatible with rotations $R \in SO(3)$. This formulation generalizes the usual definition of convolutions, which can be recovered by replacing the north pole n with the origin 0 and rotations R with translations $T \in (\mathbb{R}^n, +)$.

The spherical convolution (9) admits a convolution theorem of the form

$$\mathcal{F}[\kappa \star u](l,m) = 2\pi \sqrt{\frac{4\pi}{2l+1}} \, \mathcal{F}[u](l,m) \cdot \mathcal{F}[\kappa](l,0), \tag{10}$$

where $\mathcal{F}$ refers to the SHT[4]. By replacing the filter weights $\mathcal{F}[\kappa](l,0)$ with the learned weights $\tilde{\kappa}_\vartheta(l)$, we obtain the Spherical Fourier Neural Layer

$$\mathcal{K}_\vartheta[u] = \mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \mathcal{F}[u]], \tag{11}$$

which forms the core of the SFNO. More precisely, we have

$$\mathcal{F}[\mathcal{K}_\vartheta[u]](l,m) = \tilde{\kappa}_\vartheta(l) \cdot \mathcal{F}[u](l,m),$$

as this approach requires only one filter weight per l to be learned. If L is the maximum zonal mode ($0 \leq l \leq L$), this approach requires L learned filters, as opposed to $L(L+1)/2$ for the naive extension of the FNO.

4.2. SFNO Network Topology Design

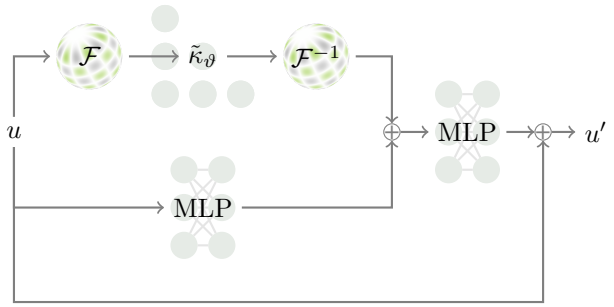


Figure 2. The structure of a single SFNO block. Multi-layer perceptrons (MLPs) act point-wise in the spatial domain and allow for channel mixing. The generalized Fourier transform $\mathcal{F}$ and its inverse $\mathcal{F}^{-1}$ allow for the learning of long-range spatial dependencies. κ is a learned filter, which is applied linearly to the frequency components.

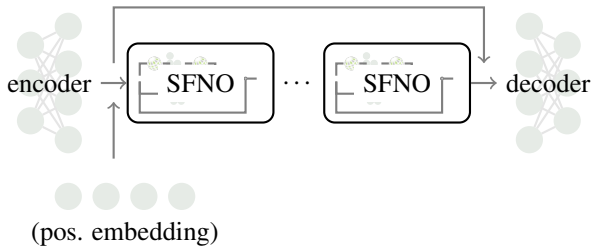


Figure 3. Diagram of the overall SFNO architecure. Encoder and decoder MLPs inflate the channel dimension. A learned position embedding is added in cases where position-dependent information should be learned by the network. At the core lie N SFNO blocks, where the first and last blocks perform up- or down-scaling. A skip connection is added for autoregressive maps close to the identity.

SFNO block: Figure 2 depicts the layout of a single SFNO block. At the core lies the Fourier layer (11), which allows the network to efficiently learn global correlations. The formulation in terms of a spherical convolution makes the Fourier layer linear and equivariant w.r.t. $SO(3)$. In case

[4]For a detailed discussion of Fourier transforms and convolutions on the sphere, we refer the reader to (Driscoll & Healy, 1994).

of vector-valued inputs u, $\tilde{\kappa}_\vartheta(l)$ is replaced by a complex-valued, learned matrix for each l and the multiplication in (11) is replaced by a matrix-vector product.

Alternatively, we propose the use of a complex-valued neural network, which acts "frequency-wise" in the Fourier domain. This approach is an extension of the AFNO architecture (Guibas et al., 2021). The non-linear approach is not equivariant w.r.t. $SO(3)$, however. The study of complex-valued neural networks is still in its infancy, and they are not as well understood as dense, ReLU networks (Voigtlaender, 2020). For this reason, we chose one possible extension of the ReLU activation, which applies it only to its real component: $x + iy \to \mathrm{ReLU}(x) + iy$.

Point-wise nonlinearities: The remaining components of our networks are equally chosen with equivariance in mind. Any operation acting point-wise on u is trivially equivariant in the continuous setting. This is not the case in the discrete setting, however, as non-linear functions may introduce arbitrary frequency components, which makes them only approximately equivariant (Karras et al., 2021). In principle, this can be remedied by applying non-linearities in the frequency domain (Poulenard & Guibas, 2021). We refrain from doing so as this leads to large performance overheads and limit ourselves to equivariance in the continuous limit.

Equivariant rescaling: To reduce the memory footprint of our models, we employ down- and up-scaling and keep a lower-resolution hidden state in our models. Adaptive FNOs, as used in FCN, realize this via patching in the encoding layer and pixel-shuffling in the decoding network (Guibas et al., 2021; Pathak et al., 2022), which is inherently not equivariant on the sphere. We propose using the SFNO block directly to perform up- and down-scaling. This is achieved by truncating the frequencies in the forward transform $\mathcal{F}$ and evaluating the inverse $\mathcal{F}^{-1}$ at a higher resolution when up-scaling. This is also applied to the skip connections to obtain the residual in the desired up- or down-scaled resolution.

Figure 3 depicts the structure of the overall network $F_\vartheta : u_n \to u_{n+1}$. It consists of three main parts: An encoder network, multiple spherical FNO blocks, and a decoder network. To maintain equivariance properties, the encoder and decoder networks are also point-wise MLPs with a single hidden layer and GELU activations (Hendrycks & Gimpel, 2016). These layers inflate and deflate the channel dimension to the embedding dimension, which remains constant throughout the network. As the autoregressive map F is close to identity, we add a large skip connection that feeds the output of the encoder to the decoder, skipping the SFNO blocks. To model spatial dependencies, a learned position embedding is added after the decoder layer. This can be regarded as an additional, learned input, and is, therefore, equivariant as well. As previously noted, the first and last SFNO blocks are used to perform up- and down-sampling. Except for the last layer, instance norm (Ulyanov et al., 2016) is applied after each MLP to normalize the inputs. We choose instance norm over other normalization methods, as it applies the same operation globally, making it equally equivariant.

We remark that the obtained model is grid-invariant, as all learned operations save for the positional embedding do not depend on the grid of the input data. This allows the model to be applied on arbitrary grids and resolutions as long as the SHT $\mathcal{F}$ and its inverse $\mathcal{F}^{-1}$ can be computed on it. Moreover, the position embedding can be parametrized in terms of spherical harmonics to facilitate the grid-invariance of the architecture.

Differentiable Spherical Harmonic Transform: To enable our method, we implement `torch-harmonics`, a library for differentiable Spherical Harmonics written in `PyTorch` (Paszke et al., 2019). We choose the "direct" algorithm (Schaeffer, 2013), as it has advantages over algorithms for the implementation on GPUs. Details on the implementation are provided in Section B.1.

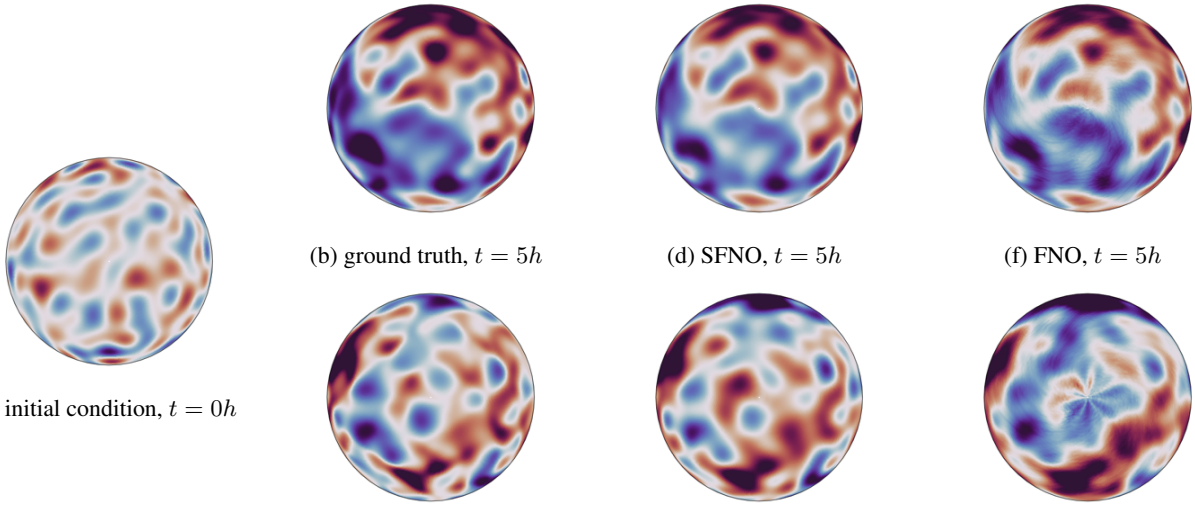
5. Numerical Experiments

5.1. Spherical Shallow Water Equations

The SWE on the rotating sphere (34) are a system of non-linear hyperbolic PDEs modeling the dynamics of a thin fluid layer in the limit where the fluid depth becomes negligible w.r.t. the characteristic wavelength. Thus, they are well-suited to model planetary fluid phenomena such as atmospheric dynamics, tsunami propagation, and tidal flows (Nair et al., 2005; Bonev et al., 2018).

We train our models on the SWE by generating random data on the fly using a classical, spectral solver (Giraldo, 2001). We use the data to train four models: two SHT-based and two FFT-based FNOs with either non-linear or linear maps in the frequency domain. Models are trained with the following hyperparameters: 4 (S)FNO blocks, a down-scaling factor of 3, and embedding dimensions of 256. The latter was chosen to make model sizes roughly equal in terms of the number of trainable parameters. The models are trained for 20 epochs using a single autoregressive step. Each time step corresponds to a single hour in the system, which requires the classical solver 150 time steps to compute. In the second stage, two autoregressive steps are used during training to fine-tune the model and improve stability. Details regarding the data generation and training of these models are outlined in Section C.1 in the Appendix.

Figure 4 shows a comparison of the (linear) SFNO model to ground truth data computed by the classical solver. Detailed numerical results and an overview of the trained models are



(a) initial condition, $t = 0h$ (b) ground truth, $t = 5h$ (d) SFNO, $t = 5h$ (f) FNO, $t = 5h$

(c) ground truth, $t = 10h$ (e) SFNO, $t = 10h$ (g) FNO, $t = 10h$

Figure 4. Solutions to the Shallow Water Equations on the rotating Sphere predicted by SFNO and FNO architectures in comparison to the ground truth solution computed using a classical spectral solver. Plots depict the geopotential height at 5 and 10 hours, corresponding to 5 and 10 autoregressive steps respectively. The view is centered on the south pole to highlight artifacts arising due to the non-geometrical treatment of the sphere.

reported in Table 1.

Discussion While both FFT and SHT-based models achieve roughly similar accuracies, it is worth noting that the SFNO models have fewer parameters as the SHT has half as many frequencies as the FFT at the same resolution. Moreover, Figure 4 depicts autoregressive rollouts of 5 and 10 hours for the linear FFT and SHT-based models. It is evident that the FFT-based approach leads to artifacts visible at the poles, which increase in severity for longer rollouts.

Training the individual models takes 240s on a single NVIDIA A6000. This is mainly bottlenecked by the numerical solver, which requires 150 time steps in order to generate a single sample 1h in advance.

5.2. Weather Prediction / Atmospheric Dynamics

We demonstrate the utility of the proposed method for the task of medium-range weather forecasting (up to two weeks) and long-timescale rollouts (up to 1 year). We train a handful of models on the ERA5 dataset (Hersbach et al., 2020) on a subset of atmospheric variables sub-sampled at a temporal frequency of 6 hours and at the native spatial resolution of the ERA5 dataset (0.25 degrees lat-long). The atmospheric variables used are listed in appendix C.2.

We use 40 years of ERA5 (1979-2018): 1979-2015 is used for training, 2016 and 2017 are used for validation, hyperparameter tuning, and model selection, and 2018 is held out as out-of-sample test set. Models are trained following a protocol similar to that outlined in Pathak et al. (2022): an initial training stage, using a single autoregressive step and a second, fine-tuning stage, in which two autoregressive steps are used. Scalable model parallelism and gradient checkpointing are used to reduce the large memory footprint encountered during autoregressive training. In both training stages, models are trained for 4 hours on 8 NVIDIA DGX machines. On average, this amounts to 40 epochs in the first stage and 5 epochs in the finetuning stage.

Table 2 reports model parameters and results for the best checkpoint. For all models, thresholding is employed and frequencies above half the sampling frequency are dropped. The embedding dimension is adapted to facilitate multi-step training with 2 autoregressive steps without gradient checkpointing on a single NVIDIA A100. The performance is evaluated using the relative L^2 and L^1 losses (lower is better), as well as the ACC score (higher is better). For a definition of the performance metrics, we refer the reader to Section B.3 in the Appendix.

To measure the predictive skill, we compare autoregressive inference results with a linear SFNO model, with predictions obtained from the state-of-the-art model in NWP – the Integrated Forecasting System (IFS) (ECMWF, 2021). To do so, we compute the ACC scores on a sample of 730 2-week-long forecasts on the out-of-sample data from 2018 shown in Figure 6.

Discussion Our SFNO architecture has predictive skill comparable to IFS on weather timescales (up to two weeks, as

Table 1. Auto-regressive results applying the Shallow Water Equations on the rotating sphere at a spatial resolution of 256×512 and a temporal resolution of 1 hour. Losses are relative and reported at 1h and 10h respectively, which correspond to 1 and 10 auto-regressive steps. Losses for the ML models are computed w.r.t. the validation data generated by the classical solver. To put them into perspective, the error of the classical method, and therefore the training data is listed in the bottom row. These errors are estimated using a high-fidelity solution computed at four times the resolution of the training data.

Model	Parameters			L^2 Loss		Eval time
	Layers	Embed. Dimension	Parameter Count	at 1h (1 step)	at 10h (10 steps)	
U-Net	20	-	$3.104 \cdot 10^7$	$2.961 \cdot 10^{-3}$	$1.462 \cdot 10^{-1}$	0.011s
FNO, linear	4	256	$4.998 \cdot 10^7$	$8.280 \cdot 10^{-4}$	$9.958 \cdot 10^{-3}$	0.156s
FNO, non-linear	4	256	$3.920 \cdot 10^7$	$8.298 \cdot 10^{-4}$	$9.784 \cdot 10^{-3}$	0.212s
SFNO, linear	4	256	$3.518 \cdot 10^7$	$7.741 \cdot 10^{-4}$	$7.239 \cdot 10^{-3}$	0.218s
SFNO, non-linear	4	256	$3.920 \cdot 10^7$	$7.673 \cdot 10^{-4}$	$1.558 \cdot 10^{-2}$	0.321s
Classical Solver	-	-	-	$1.891 \cdot 10^{-2}$	$3.570 \cdot 10^{-2}$	1.299s

Table 2. Auto-regressive results obtained on the ERA5 weather data. Relative validation losses (lower is better) are averaged over the finetuning steps at 6 and 12h (1 and 2 autoregressive steps). The Anomaly Correlation Coefficient (ACC, higher is better) is reported at 120h (20 autoregressive steps).

Model	Parameters				Loss		ACC at 120h (20 steps)			
	Layers	Embed. Dim.	Param. Count	Steps	L^2	L^1	10u	2t	z500	u500
FNO, linear	8	64	$2.190 \cdot 10^9$	1	0.1172	0.0917	0.4896	0.7829	0.7454	0.5631
				2	0.1176	0.0888	0.5140	0.8661	0.8039	0.6223
FNO, non-linear	8	384	$4.332 \cdot 10^8$	1	0.0908	0.0648	0.6284	0.8939	0.8533	0.6912
				2	0.0935	0.0669	0.6611	0.9072	0.8677	0.7156
SFNO, linear	8	384	$8.296 \cdot 10^8$	1	0.0930	0.0658	0.6271	0.8965	0.8589	0.6924
				2	0.0933	0.0660	0.6471	0.9079	0.8728	0.7141
SFNO, non-linear	8	384	$4.332 \cdot 10^8$	1	0.0942	0.0658	0.6079	0.8943	0.8356	0.6616
				2	0.0977	0.0700	0.6489	0.9010	0.8598	0.7012

observed in Figure 6), while showing unprecedented long-term stability for a year-long rollout. More importantly, a one year-long rollout of the SFNO is computed in 12.8 minutes on a single NVIDIA A6000 GPU, compared to one hour (wall-clock time) for a year-long simulation of IFS on 1000 dual-socket CPU nodes (Bauer et al., 2020). With the caveat of differing hardware, this corresponds to a speedup of close to 5,000x.

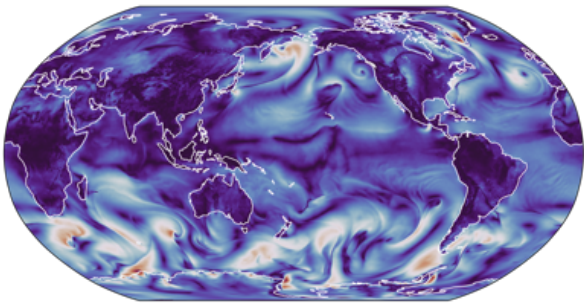
While both FFT- and SHT-based approaches achieve similar accuracy for medium-range forecasts (5- to 10-day, Table 2), the advantage of respecting the underlying spherical geometry becomes evident at longer rollouts as illustrated in Figures 5 and 1. Figure 1 illustrates the importance of equivariance especially well. Many of our architectural choices were motivated by equivariance and are also present in the non-linear FNO model. While the AFNO model shows artifacts early on, these are not as pronounced in the FNO models and absent in the SFNO model.

6. Conclusion

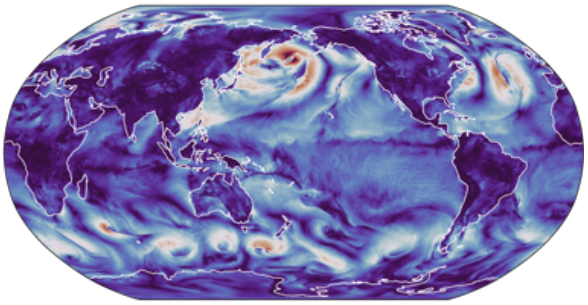
This paper demonstrates how FNOs can be extended via the generalized Fourier transform to learn operators that act on Riemannian manifolds and are equivariant with respect to operations of a symmetry group. We apply this concept to $SO(3)$ and present a novel and fully data-driven network architecture called the SFNO, which facilitates the generation of stable long-range forecasts of Earth's complex atmospheric dynamics of a whole year, in 13 minutes on a single GPU.

The proposed method pushes the frontier of data-driven deep learning for weather and climate prediction because of the following key properties:

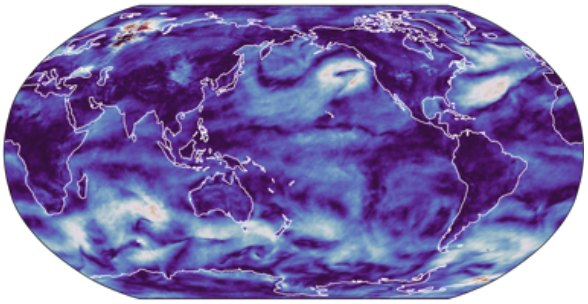
Respecting spherical geometry is essential to ensure that topological boundary conditions are realized correctly. It leads to stable, long-range roll-outs without the respective predictions degenerating to implausible distributions of the respective quantities, such as wind speed. This is an essential property for enabling the creation of ML-based digital



(a) Spherical FNO using the SHT



(b) Ground Truth



(c) FFT-based FNO architecture for reference

Figure 5. Year-long rollout (1,450 autoregressive steps) of absolute wind speed 10m above the surface depicting stable behavior over exceptionally long timescales for an ML model, which has important implications for ML-based climate modeling. In contrast, the FFT-based architecture has spurious waves and artifacts and excessive diffusion. SFNO faithfully captures the dynamics of weather within the predictability horizon of two weeks and shows physically- and statistically-consistent behaviour over longer timescales up to a year.

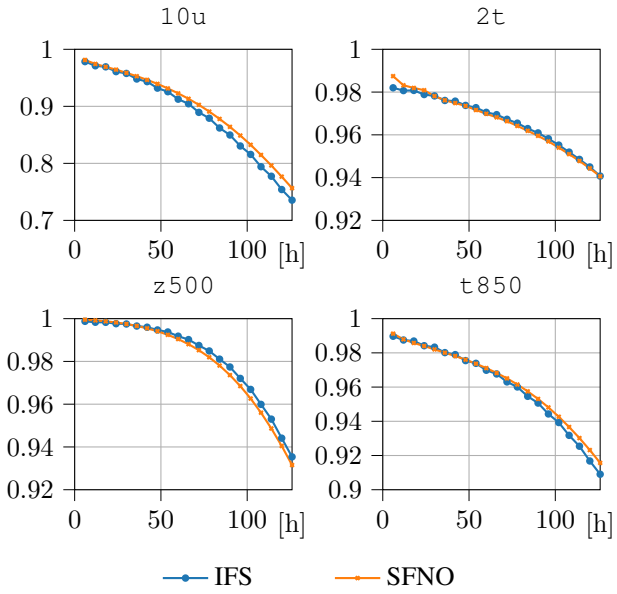


Figure 6. Comparison of forecast skill (ACC score) as a function of prediction lead time (in hours) between IFS (blue) and the SFNO (orange) for four physical variables: 10u (surface winds in the latitudinal direction); 2t (temperature 2m above the surface); z500 (geopotential height at 500 hPa); and t850 (the temperature at 850 hPa). All curves are computed by averaging the skill over 730 autoregressive rollouts starting at different initial conditions. SFNO closely matches the forecast skill of IFS in the predicted variables.

twins.

Grid-invariance of the architecture allows the model to be applied on arbitrary grids as long as the SHT and its inverse can be formulated. This allows the method to be quickly fine-tuned on new grids and resolutions, enabling training ML models on diverse weather and climate data that exist on different grids and resolutions.

Computational efficiency is tightly connected to an efficient implementation of the respective generalized Fourier transform. This is the case for spherical topologies via the spherical harmonics transform. As a consequence, the proposed method makes long-term weather forecasting significantly more accessible and is hence enabling its democratization: Predictions for an entire year can be computed within 13 minutes on a single NVIDIA RTX A6000 GPU.

The high accuracy, long-term stability, and immense speedup over classical methods bear promises for the application of Spherical Fourier Neural Operators in the holy grail of prediction: sub-seasonal-to-seasonal forecasting. It is foreseeable that such methods could one day lead to ML-based climate prediction.

Acknowledgements

We thank ECMWF for enabling this line of research by providing publicly available datasets (Hersbach et al., 2020). Moreover, we extend our gratitude to the anonymous reviewers for taking their time and providing us with valuable comments that improved the quality of the paper. Finally, we are grateful to our colleagues Kamyar Azizzadenesheli, Noah Brenowitz, Yair Cohen, Jean Kossaifi, Nikola Kovachki, Thomas Müller and Mike Pritchard for fruitful discussions and proof-reading the manuscript.

References

Abramowitz, M., Stegun, I. A., et al. *Handbook of mathematical functions*, volume 55. Dover New York, 1964.

Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R., and Ott, E. A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, 47(9):e2020GL087776, 2020.

Bauer, P., Quintino, T., Wedi, N., Bonanni, A., Chrust, M., Deconinck, W., Diamantakis, M., Düben, P., English, S., Flemming, J., Gillies, P., Hadade, I., Hawkes, J., Hawkins, M., Iffrig, O., Kühnlein, C., Lange, M., Lean, P., Marsden, O., Müller, A., Saarinen, S., Sarmany, D., Sleigh, M., Smart, S., Smolarkiewicz, P., Thiemert, D., Tumolo, G., Weihrauch, C., Zanna, C., and Maciel, P. The ecmwf scalability programme: Progress and plans, 02/2020 2020. URL https://www.ecmwf.int/node/19380.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 813–824. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/bertasius21a.html.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. 11 2022. URL http://arxiv.org/abs/2211.02556.

Bonev, B., Hesthaven, J. S., Giraldo, F. X., and Kopera, M. A. Discontinuous galerkin scheme for the spherical shallow water equations with applications to tsunami modeling and prediction. *Journal of Computational Physics*, 2018. ISSN 10902716. doi: 10.1016/j.jcp.2018.02.008.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 4 2021. URL http://arxiv.org/abs/2104.13478.

Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

Cohen, T. S. and Welling, M. Group equivariant convolutional networks. 2 2016. URL http://arxiv.org/abs/1602.07576.

Cohen, T. S., Geiger, M., Koehler, J., and Welling, M. Spherical cnns. 1 2018. URL http://arxiv.org/abs/1801.10130.

Conway, J. and Sloane, N. *Sphere Packings, Lattices and Groups*, volume 290. 01 1988. ISBN 978-1-4757-2018-1. doi: 10.1007/978-1-4757-2016-7.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Driscoll, J. and Healy, D. Computing fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15:202–250, 6 1994. ISSN 01968858. doi: 10.1006/aama.1994.1008. URL https://linkinghub.elsevier.com/retrieve/pii/S0196885884710086.

ECMWF. Ifs documentation cy47r3 - part iii- dynamics and numerical procedures, 1 2021. URL https://www.ecmwf.int/node/20202.

Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., Böhm, A., Deubner, J., Jäckel, Z., Seiwald, K., et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67–70, 2019.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

Gell-Mann, M. The Eightfold Way: A Theory of strong interaction symmetry. 3 1961. doi: 10.2172/4008239.

Giraldo, F. X. A spectral element shallow water model on spherical geodesic grids. *Int. J. Numer. Meth. Fluids*, 35: 869–901, 2001. doi: 10.1002/1097-0363(20010430).

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., and Catanzaro, B. Adaptive fourier neural operators: Efficient token mixers for transformers, 2021. URL https://arxiv.org/abs/2111.13587.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). 2016.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., Chiara, G. D., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. ISSN 1477-870X.

Kalnay, E. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, 2003.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. 6 2021. URL http://arxiv.org/abs/2106.12423.

Kossaifi, J., Kovachki, N. B., Azizzadenesheli, K., and Anandkumar, A. Multi-grid tensorized fourier neural operator for high resolution PDEs, 2023. URL https://openreview.net/forum?id=po-oqRst4Xm.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces. 8 2021. URL http://arxiv.org/abs/2108.08481.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Stott, J., Vinyals, O., Mohamed, S., and Battaglia, P. Graphcast: Learning skillful medium-range global weather forecasting. 12 2022. URL http://arxiv.org/abs/2212.12794.

Lecun, Y. and Bengio, Y. *Convolutional Networks for Images, Speech and Time Series*, pp. 255–258. The MIT Press, 1995.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations, 2020.

McCormick, S. F. *Multigrid methods*. SIAM, 1987.

McEwen, J. D. and Wiaux, Y. A novel sampling theorem on the sphere. 10 2011. doi: 10.1109/TSP.2011.2166394. URL http://arxiv.org/abs/1110.6298http://dx.doi.org/10.1109/TSP.2011.2166394.

Nair, R. D., Thomas, S. J., and Loft, R. D. A discontinuous galerkin global shallow water model. *Monthly Weather Review*, 133:876–888, 4 2005. ISSN 0027-0644. doi: 10.1175/MWR2903.1. URL https://journals.ametsoc.org/doi/10.1175/MWR2903.1.

NVIDIA, Vingelmann, P., and Fitzek, F. H. Cuda, release: 10.2.89, 2020. URL https://developer.nvidia.com/cuda-toolkit.

Ocampo, J., Price, M. A., and McEwen, J. D. Scalable and equivariant spherical cnns by discrete-continuous (disco) convolutions. 9 2022. URL http://arxiv.org/abs/2209.13603.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. 12 2019. URL http://arxiv.org/abs/1912.01703.

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. 2 2022. URL http://arxiv.org/abs/2202.11214.

Poulenard, A. and Guibas, L. J. A functional approach to rotation equivariant non-linearities for tensor field networks. pp. 13169–13178. IEEE, 6 2021. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.01297. URL https://ieeexplore.ieee.org/document/9578769/.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators. 4 2022. URL http://arxiv.org/abs/2204.11127.

Rasp, S. and Thuerey, N. 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.

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thuerey, N. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

Schaeffer, N. Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations. *Geochemistry, Geophysics, Geosystems*, 14:751–758, 3 2013. ISSN 15252027. doi: 10.1002/ggge.20071.

Scher, S. and Messori, G. Weather and climate forecasting with neural networks: using general circulation models (gcms) with different complexity as a study ground. *Geoscientific Model Development*, 12(7):2797–2809, 2019.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Voigtlaender, F. The universal approximation theorem for complex-valued neural networks. 12 2020. URL http://arxiv.org/abs/2012.03351.

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. *Advances in Water Resources*, 163:104180, 2022.

Weyn, J., Durran, D., Caruana, R., and Cresswell-Clay, N. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13:e2021MS002502, 2021.

Weyn, J. A., Durran, D. R., and Caruana, R. 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.

Workman, R. L. and Others. Review of Particle Physics. *PTEP*, 2022:083C01, 2022. doi: 10.1093/ptep/ptac097.

Yuval, J. and O'Gorman, P. A. Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature communications*, 11(1): 3295, 2020.

A. Preliminaries

A.1. Rotations on the 2-Sphere

We discuss some basic notions regarding the two-dimensional sphere S^2 and the rotation group $SO(3)$.

To define functions on the sphere, we require coordinates. A familiar choice is the parametrization of points $x \in S^2$ in terms of colatitude $\theta \in [0, \pi]$ and longitude $\varphi \in [0, 2\pi]$. The unit vector x can then be parametrized as $(\cos(\varphi)\sin(\theta), \sin(\varphi)\sin(\theta), \cos(\theta))^T$.

To discuss symmetries, we require the special orthogonal group in three variables $SO(3)$. These are proper rotations in $\mathbb{R}^3$, characterized by three-by-three matrices of determinant one, whose inverses are their transpose. Any rotation $R \in SO(3)$ can be written in terms of the Eulerian angles $\varphi \in [0, 2\pi], \theta \in [0, \pi], \psi \in [0, 2\pi]$, such that

$$R = R_z(\varphi) R_y(\theta) R_z(\psi), \tag{12}$$

where R_z and R_y are rotations around the z- and y-axes:

$$R_z(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix}. \tag{13}$$

Unlike translations in the plane, rotations do not commute in general, making $SO(3)$ non-abelian. The rotations $R \in SO(3)$ sweep the entire sphere. We can see this by applying the rotation R to the north pole $n = (0, 0, 1)^T$, which yields

$$Rn = R_z(\varphi) R_y(\theta) R_z(\psi) n = (\cos(\varphi)\sin(\theta), \sin(\varphi)\sin(\theta), \cos(\theta))^T. \tag{14}$$

We observe that the last rotation angle ψ is dropped, illustrating that S^2 can be obtained as the quotient of $SO(3)$ and $SO(2)$.

A.2. Spherical Harmonics

We define the inner product on the 2-Sphere S^2

$$\langle u, v \rangle_{L^2(S^2)} = \int_{S^2} \overline{u}\, v \, \mathrm{d}\Omega = \int_0^{2\pi} \int_0^{\pi} \overline{u(\theta, \varphi)}\, v(\theta, \varphi) \, \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\varphi, \tag{15}$$

where $\mathrm{d}\Omega$ is the associated Lebesgue measure on the sphere[5]. This induces the L^2 norm

$$\|u\|_{L^2(S^2)} = \sqrt{\langle u, u \rangle_{L^2(S^2)}}, \tag{16}$$

as well as the Hilbert space $L^2(S^2)$ of square integrable functions on S^2. We introduce the spherical harmonics defined as

$$Y_l^m(\theta, \varphi) = (-1)^m c_l^m P_l^m(\cos\theta) e^{\mathrm{i}m\varphi} = \widehat{P}_l^m(\cos\theta) e^{\mathrm{i}m\varphi}, \quad c_l^m := \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}}, \tag{7}$$

where $P_l^m(\cos\theta)$ are the associated Legendre polynomials. The normalization factor c_l^m normalizes the spherical harmonics w.r.t. the $L^2(S^2)$ inner product, s.t.

$$\langle Y_l^m, Y_{l'}^{m'} \rangle_{L^2(S^2)} = \delta_{ll'} \delta_{mm'}. \tag{17}$$

In other words, the spherical harmonics (7) form an orthogonal basis of $L^2(S^2)$.

The spherical harmonics (7) have many useful properties, induced by properties of the trigonometric functions and the associated Legendre polynomials (Abramowitz et al., 1964). One such useful property is the symmetry relation

$$\overline{Y_l^m}(\theta, \phi) = (-1)^m \, Y_l^{-m}(\theta, \phi), \tag{18}$$

which is particularly useful for real-to-complex spherical harmonic transforms. Using this property, we can recover the negative m components from the positive ones[6].

[5] The measure $\mathrm{d}\Omega$ is invariant under rotations in $SO(3)$. The same is true for the inner product.

[6] Analogous to the Hermitian symmetry of the Fourier coefficients of a real-valued signal where negative frequency contributions can be inferred from the positive ones.

A.3. Fourier Neural Operators

On the doubly periodic domain $S^1 \times S^1$, the Fourier transform and it's inverse can be expressed as

$$\mathcal{F}[u](k,l) = \tilde{u}(k,l) = \int_0^1 \int_0^1 u(x,y) \exp(-i2\pi kx) \mathrm{d}x \exp(-i2\pi ly) \mathrm{d}y \tag{19}$$

and

$$\mathcal{F}^{-1}[\tilde{u}](x) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \tilde{u}(k,l) \exp(i2\pi kx) \exp(i2\pi ly) \tag{20}$$

respectively.

At the core of the Fourier Neural Operator lies the Fourier layer, which can be understood as a global convolution

$$u \to \mathcal{K}[u] = \mathcal{F}^{-1}[\mathcal{F}[\kappa] \cdot \mathcal{F}[u]]. \tag{2}$$

By replacing the filter weights $\mathcal{F}[\kappa]$ with learned weights $\tilde{\kappa}_\vartheta$ in Fourier space, we obtain the Fourier Neural Operator

$$u \to \mathcal{K}_\vartheta[u] = \mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \mathcal{F}[u]]. \tag{21}$$

Here, $\tilde{\kappa}_\vartheta$ represents a parametrization of the filter weights in Fourier space with the weight vector ϑ. As such, a filter weight $\tilde{\kappa}_\vartheta(k,l)$ is learned for each frequency k, l, such that

$$\mathcal{K}_\vartheta[u] = \mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \tilde{u}](x) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \tilde{\kappa}_\vartheta(k,l) \cdot \mathcal{F}[u](k,l) \ \exp(i2\pi kx) \exp(i2\pi ly). \tag{22}$$

In practice, we truncate the Fourier series as only a finite number of filter parameters is learned. Moreover, if u is vector-valued, we replace $\tilde{\kappa}_\vartheta(k,l)$ with a learned matrix and $\tilde{\kappa}_\vartheta(k,l) \cdot \mathcal{F}[u]$ becomes a matrix-vector product with summation over the embedding dimension (Li et al., 2020).

The above formulation admits efficient learning of non-local operators, which is a desireable property for PDE applications (Kovachki et al., 2021). We remark that the FNO is trivially translation-equivariant w.r.t. translations $T(a,b) : (x,y) \to (x+a, y+b)$. Applying the passive translation $\Phi_T[u](x) = u(T^{-1}x)$ to the input of the FNO yields

$$\begin{aligned} \mathcal{K}_\vartheta[\Phi_T[u]] &= \mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \mathcal{F}[\Phi_T[u]]] = \mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \exp(i2\pi ka) \exp(i2\pi lb) \, \mathcal{F}[u]] \\ &= \Phi_T[\mathcal{F}^{-1}[\tilde{\kappa}_\vartheta \cdot \mathcal{F}[u]]] = \Phi_T[\mathcal{K}_\vartheta[u]], \end{aligned} \tag{23}$$

where we have used $\Phi_T[\mathcal{F}[u]] = \exp(i2\pi ka) \exp(i2\pi lb) \, \mathcal{F}[u]$.

B. Implementation Details

B.1. Differentiable Spherical Harmonics Transforms

In order to facilitate differentiable computation we require a differentiable implementation of the Spherical Harmonic Transforms (SHT). To do so we implemented our own differentiable Spherical Harmonics Transform library for `PyTorch` (`https://pytorch.org/`).

While it is possible to compute the SHT in $\mathcal{O}(L^2 \log^2 L)$ time (Driscoll & Healy, 1994; Schaeffer, 2013; McEwen & Wiaux, 2011), these algorithms typically suffer from numerical-intabilities and tend to have large constants in their algorithmic complexity. As such, the 'semi-naive' algorithm, which computes the projection onto the associated Legendre polynomials via quadrature and the projection onto the harmonic functions via the FFT (Schaeffer, 2013). In our experience, this algorithm tends to outperform the asymptotically optimal algorithms, and moreover, is better-suited to the execution on GPUs and distributed GPU-systems due to the availability of highly optimized primitives for these operations (NVIDIA et al., 2020). As such, `torch-harmonics` implements the 'semi-naive' algorithms and distributes the projection of the associated Legendre polynomials onto multiple GPUs in the distributed case.

We implement the forward transformation in equation (8) by performing a 1D real to complex DFT over the azimuth degrees of freedom, followed by a matrix multiplication of the Legendre polynomials $P_l^m(\cos\theta)$ and re-scaled by the normalization

factors c_l^m. Therefore, we can write

$$\hat{u}[\ldots, l, m] = \sum_{j=0}^{H-1} P[l, m, j]\, \mathrm{FT}\big[u[\ldots, j, k], k\big][j, m] \tag{24}$$

Where $0 \leq j < H$, $0 \leq k < W$, where H and W are the number of discrete θ and ϕ angles respectively. Furthermore, $0 \leq l \leq L$ and $0 \leq m \leq M$. Note that we do not need to store the modes for negative m since we are only considering real-to-complex transforms and thus those components can be retrieved using the symmetry relation (18). Lastly, the ellipsis in (24) denotes all tensor modes which are not contracted in the transform and thus can be vectorized over. Typical modes include the batch as well as the feature dimension of tensor u.

We define the real-to-complex DFT as

$$\mathrm{FT}\big[u[\ldots, j, k], k\big][j, m] \equiv \frac{1}{\sqrt{W}} \sum_{k=0}^{W-1} \exp\left(-i\frac{2\pi m k}{W}\right) u[\ldots, j, k] \tag{25}$$

Note that the negative phase arises from the fact that the forward transform involves the complex conjugate of Y_l^m. We define the Legendre weights as

$$P[l, m, j] \equiv (-1)^m\, c_l^m\, P_l^m(\cos\theta_j)\, w(\theta_j), \tag{26}$$

where the weight vector

$$w(\theta_j) \simeq \sin\theta \cdot \mathrm{d}\theta, \tag{27}$$

is chosen such that the sum over j in (18) approximates the integral over θ in (8). These weight factors typically depend on the chosen grid of the spatial discretization. Since we are working with static input and output grids, the Legendre weight matrices P can be pre-computed and stored. Furthermore, effective downsampling can be achieved by reducing the maximum wave number L.

The inverse transform can be defined in a similar fashion:

$$u[\ldots, j, k] = \mathrm{FT}^{-1}\left[\sum_{l=0}^{L} \hat{P}[l, m, j]\, \hat{u}[\ldots, l, m], m\right][j, k] \tag{28}$$

where $\mathrm{FT}^{-1}\big[\ldots, m\big][j, k]$ denotes the inverse complex-to-real inverse DFT with respect to m. The Legendre weights in this case are given by:

$$\hat{P}[l, m, j] \equiv (-1)^m\, c_l^m\, P_l^m(\cos\theta_j). \tag{29}$$

Note that we do not need to include an additional weight vector for the backward transform since the sum over l in (28) does not approximate an integral and thus does not receive an additional term from an integration measure.

The above implementation of discrete SHT can be parallelized using the pencil decomposition technique which is also used in higher-dimensional distributed Fourier transforms: first, the user needs to initializes a GPU communication grid for H, L and W, M directions respectively and pass this to the distributed initialization routine of `torch-harmonics`. We further assume that the input to the distributed transform is split evenly among those communication dimensions (including padding if required).

For example, in case of the forward transform, the input to the distributed transform can be a spatially decomposed field in H and W but with local feature data. The idea is to perform a global transposition using `torch.distributed.all_to_all` of the data in feature and W domain, so that after the transposition the W domain is fully local and the feature domain is distributed. After this, the one dimensional FFT along W can be performed locally and is embarrassingly parallel with respect to the now split feature dimension. After this first transform, the data is again globally transposed such that the H domain local and the the feature domain as well as the m domain are both distributed. In This case, the Legendre transformation can be performed locally and embarrassingly parallel with respect to the distributed feature dimension. Afterwards, we perform a third global transposition to achieve a decomposed L, M domain but local feature data. For the inverse SHT, we invert this whole process. Output dimensions which cannot be evenly split among GPUs, are zero-padded automatically on the largest rank in the corresponding communication dimension. Information about the padding and input and output sizes are stored in the corresponding SHT instance and can be queried by the user.

The advantage of this approach is that input and output tensors have similar spatial decompositions and thus this approach simplifies the end-to-end spatial parallelization of SFNO models. Additionally, since all transforms are performed locally, the result of the distributed spherical transform is bit-wise identical to the result of a serial transform. This is not the case for fork-join approaches as discussed below.

Alternatively, one could also follow a fork-join approach in forward and inverse SHT respectively. This is can be achieved by splitting the l degrees-of-freedom of the Legendre weights $P, \hat{P}$ into evenly sized chunks $l_c \doteq (L{+}1)/R$ and distributing those across all ranks R. This means, that every rank $0 \leq r < R$ owns a sub-tensor $P_r[l, m, j]$ with $r \cdot l_c \leq l < (r{+}1) \cdot l_c$ but full m and j (and equivalently for $\hat{P}$). Therefore, the distributed forward transform variant of (24) transforms a shared input tensor $u[\ldots, j, k]$ onto a rank-local transformed tensor $u_r[\ldots, l, m]$ with the above ranges for l, m (fork). The distributed inverse transform in turn transforms the rank-local tensor $\hat{u}$ back into a shared tensor u (join). The communication primitive for both operations is `torch.distributed.all_reduce`, where it has to be applied on the input gradient in the backward pass of the forward routine as well as on the output tensor in the forward pass of the inverse routine. The forward pass of the forward and the backward pass of the inverse transformation do not need additional communication. Since PyTorch cannot back-propagate through communication collectives natively, we use `torch.autograd.Function` to implement collective primitives with fully defined forward and backward pass.

B.2. Training

To find the parameter vector ϑ of the learned map F_ϑ, an objective function is minimized. We choose a geometric loss function, which is obtained by approximating the L^p norm $\|F_\vartheta[u_n] - u_{n+1}\|_{L^p(S^2)}$ on the sphere:

$$\mathcal{L}[F_\vartheta[u_n], u_{n+1}] = \sum_{c \in \text{channels}} \left(\frac{\sum_{i \in \text{grid}} w_i \left| F_\vartheta[u_n](x_i) - u_{n+1}(x_i) \right|^p}{\sum_{i \in \text{grid}} w_i \left| u_{n+1}(x_i) \right|^p} \right)^{\frac{1}{p}}. \tag{30}$$

To compute the loss, the absolute difference is summed over the gridpoints x_i and weighted with w_i, which are the products of the Jacobian $\sin\theta_i$ and the quadrature weights. As quadrature rule we picked the simple Riemann sum[7] The loss (30) is then computed for each channel $c \in \{\texttt{u10m}, \texttt{v10m}, \texttt{t2m}, \ldots\}$ separately, and then normalized by the norm of the target. It is then averaged over all predicted channels to obtain the final loss. For training we set $p = 2$.

Training is performed in two stages. In the first stage, the model is trained to obtain the best possible single-step performance. To this end, the loss (30) is minimized after a single prediction step. The learning rate is scheduled to follow a cosine pattern, starting with a learning rate of 10^{-3}

The second stage is a finetuning stage where the model is optimized for autoregressive performance. To this end, n_{steps} autoregressive steps are performed and the loss is accumulated at each step of the forecast, i.e.

$$\mathcal{L}_{\text{autoregressive}} = \frac{1}{n_{\text{steps}}} \sum_{s=1}^{n_{\text{steps}}} \mathcal{L}[F_\vartheta^s[u_n], u_{n+m}], \quad \text{where} \quad F_\vartheta^s = \underbrace{F_\vartheta \circ \ldots \circ F_\vartheta}_{s \text{ times}}. \tag{31}$$

Gradients are then backpropagated through the entire unrolled sequence, to get the weight updates. The finetuning is performed with with increasing n_{steps}, starting at $n_{\text{steps}} = 2$. For each n_{steps}, the model is trained for 5 epochs with a constant learning rate of 10^{-5}.

B.3. Performance evaluation

To assess the performance of our models, the we use the relative L^2 and L^1 losses as defined in equation (30). Another common metric used in weather prediction is the anomaly correlation coefficient (ACC). The latitude weighted ACC for a forecast variable v at forecast time-step l is defined following Rasp et al. (2020) as follows:

$$\text{ACC}(v, l) = \frac{\sum_{m,n} w_{\text{Lat}}(m) \tilde{X}_{\text{pred}}(l)\,[v, m, n]\, \tilde{X}_{\text{true}}(l)\,[v, m, n]}{\sqrt{\sum_{m,n} w_{\text{Lat}}(m) \left(\tilde{X}_{\text{pred}}(l)\,[v, m, n] \right)^2 \sum_{m,n} w_{\text{Lat}}(m) \left(\tilde{X}_{\text{true}}(l)\,[v, m, n] \right)^2}}, \tag{32}$$

[7] On equiangular grids on the sphere, Clenshaw-Curtiss quadrature is often preferable (Schaeffer, 2013). This did not improve performance over the Riemann sum however, and we chose to use the simpler Riemann sum instead.

where $\tilde{X}_{\text{pred/true}}(l)\,[v, m, n]$ represents the long-term-mean-subtracted value of predicted (/true) variable v at the location denoted by the grid co-ordinates (m, n) at the forecast time-step l. The long-term mean of a variable is simply the mean value of that variable over a large number of historical samples in the training dataset. The long-term mean-subtracted variables $\tilde{X}_{\text{pred/true}}$ represent the anomalies of those variables that are not captured by the long term mean values. $w_{\text{Lat}}(m)$ is the latitude weighting factor at the co-ordinate m. The latitude weighting is defined by Equation 33 as

$$w_{\text{Lat}}(j) = \frac{\cos(\text{Lat}(m))}{\frac{1}{N_{\text{Lat}}}\sum_{j}^{N_{\text{lat}}}\cos(\text{Lat}(m))}. \tag{33}$$

We report the mean ACC over all computed forecasts from different initial conditions and report the variability in the ACC over the different initial conditions by showing the first and third quartile value of the ACC in all the ACC plots that follow unless stated otherwise.

C. Datasets

C.1. Shallow Water Equations on the Rotating Sphere

The shallow water equations on the rotating 2-sphere model a thin layer of fluid covering a rotating sphere. They are typically derived from the three-dimensional Navier-Stokes equations, assuming incompressibility and integrating over the depth of the fluid layer. They are formulated as a system of hyperbolic partial differential equations

$$\begin{cases} \partial_t \varphi + \nabla \cdot (\varphi u) = 0 & \text{in } S^2 \times (0, \infty) \\ \partial_t (\varphi u) + \nabla \cdot T = S & \text{in } S^2 \times (0, \infty) \\ \varphi = \varphi_0 & \text{on } S^2 \times \{t = 0\}, \\ u = u_0 & \text{on } S^2 \times \{t = 0\}. \end{cases} \tag{34}$$

The state vector $(\varphi, \varphi u^T)^T$ contains the geopotential layer depth φ (mass) and the tangential momentum vector φu (discharge). In curvilinear coordinates, the flux tensor T can be written written with the outer product as $\varphi u \otimes u$. The right-hand side contains flux terms such as the Coriolis force. A detailed treatment of the SWE equations can be found in e.g. (Giraldo, 2001; Bonev et al., 2018; Nair et al., 2005).

Training data for the SWE is generated by randomly generating initial conditions and advancing them in time using a classical numerical solver. The initial geopotential height and velocity fields are realized as Gaussian random fields on the sphere. The initial layer depth has an average of $\varphi_{\text{avg}} = 10^3 \cdot g$ with a standard deviation of $120 \cdot g$. The initial velocity components have a zero mean and a standard deviation of $0.2 * \sqrt{\varphi_{\text{avg}}}$. The parameters of the PDE, such as gravity, radius of the sphere and angular velocity, we choose the parameters of the Earth. Training data is generated on the fly by using a spectral method to numerically solve the PDE on an equiangular grid with a spatial resolution of 256×512 and timesteps of 150 seconds. Time-stapping is performed using the third-order Adams-Bashford scheme. The numerical method then computes geopotential height, vorticity and divergence as output.

This data is z-score normalized and the modes are trained using epochs containing 256 samples each. To optimize the weights, we use the popular Adam optimizer with a learning rate of $2 \cdot 10^{-3}$.

C.2. Weather prediction/ERA5 Data

It is a multi-decadal, high-frequency estimate of the state of the Earth's atmosphere. It is the result of reanalysis, a process that uses data-assimilation (Carrassi et al., 2018; Kalnay, 2003) to combine modern numerical weather forecasting models with historical observational records to produce an estimate of the historical ocean-atmosphere system. A reanalysis dataset such as ERA5 spans multiple decades using dynamics from an unchanging modern numerical model. This is in contrast to an operational analysis dataset where the numerical model gets periodically updated due to advances in computational techniques, numerical methods and and improvements in the understanding of fundamental geophysics. Thus a reanalysis dataset maintains temporal consistency. Furthermore, raw observations of the earth's ocean-atmosphere system are sparsely distributed in time and space, multimodal, and of variable quality. A reanalysis dataset assimilates various observational sources informed by the uncertainty estimates of those observation sources. A reanalysis dataset provides a consistent picture of the history of the earth's atmosphere making it very useful as a training dataset for a machine learning model. Consequently, a large and growing number of researchers (Rasp & Thuerey, 2021; Weyn et al., 2020; Scher & Messori,

Table 3. Atmospheric and Surface variables predicted by our model. Two subsets of 26 and 73 variables each are used to train models of various sizes. Detailed desciptions can be accessed on the ECMWF website https://apps.ecmwf.int/codes/grib/param-db.

DESN	DESCRIPTION	ECMWF ID
10U	10 METRE u-WIND COMPONENT	165
10V	10 METRE v-WIND COMPONENT	166
2T	2 METRE TEMPERATURE	167
SP	SURFACE PRESSURE	135
MSL	MEAN SEA LEVEL PRESSURE	151
TCWV	TOTAL COLUMN VERTICALLY-INTEGRATED WATER VAPOUR	137
100U	100 METRE u-WIND COMPONENT	228246
100V	100 METRE v-WIND COMPONENT	228247
Z---	GEOPOTENTIAL (AT PRESSURE LEVEL ---)	129
T---	TEMPERATURE (AT PRESSURE LEVEL ---)	130
U---	u COMPONENT OF THE WIND (AT PRESSURE LEVEL ---)	131
V---	v COMPONENT OF THE WIND (AT PRESSURE LEVEL ---)	132
R---	RELATIVE HUMIDITY (AT PRESSURE LEVEL ---)	157

2019; Arcomano et al., 2020; Pathak et al., 2022; Bi et al., 2022; Lam et al., 2022) have used the ERA5 dataset for training data-driven numerical weather models.

Models are trained on two subsets of the variables: a 26 channel dataset for evaluating and comparing models to each other, and a 73 channel dataset used to train a larger model for comparison with IFS. The following set of 26 variables is used in the 26 variable dataset: 10u, 10v, 2t, sp, msl, tcwv, 100u, 100v, z50, z250, z500, z850, z1000, u250, u500, u850, u1000, v250, v500, v850, v1000, t100, t250, t500, t850, r500. For the 73 variable dataset, we add z---, t---, u---, v--- and r--- at pressure levels 50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 850, 925, 1000 hPa to the already existing variables. Table 3 lists an overview of variables used during training and their meaning. For a complete overview, we refer to the ECMWF website https://apps.ecmwf.int/codes/grib/param-db.

D. Supplementary material

Our differentiable implementation of the SHT and a reference implementation of SFNO can be found at https://github.com/NVIDIA/torch-harmonics.

Short videos depicting long rollouts of SFNO and polar artifacts can be found at https://youtu.be/OM3JZZN5uE4 and https://youtu.be/LPVejeU8YDE, respectively.