
Attention on the Sphere

Boris Bonev^{†*}, Max Rietmann^{†*}, Andrea Paris, Alberto Carpentieri, Thorsten Kurth^{†*}
NVIDIA Corporation, 95051 Santa Clara, CA, USA
{bbonev, mrietmann, a paris, acarpentieri, tkurth}@nvidia.com

Abstract

We introduce a generalized attention mechanism for spherical domains, enabling Transformer architectures to natively process data defined on the two-dimensional sphere - a critical need in fields such as atmospheric physics, cosmology, and robotics, where preserving spherical symmetries and topology is essential for physical accuracy. By integrating numerical quadrature weights into the attention mechanism, we obtain a geometrically faithful spherical attention that is approximately rotationally equivariant, providing strong inductive biases and leading to better performance than Cartesian approaches. To further enhance both scalability and model performance, we propose neighborhood attention on the sphere, which confines interactions to geodesic neighborhoods. This approach reduces computational complexity and introduces the additional inductive bias for locality, while retaining the symmetry properties of our method. We provide optimized CUDA kernels and memory-efficient implementations to ensure practical applicability. The method is validated on three diverse tasks: simulating shallow water equations on the rotating sphere, spherical image segmentation, and spherical depth estimation. Across all tasks, our spherical Transformers consistently outperform their planar counterparts, highlighting the advantage of geometric priors for learning on spherical domains.

1 Introduction

The two-dimensional sphere embedded in three dimensions (S^2) plays a crucial role in a variety of scientific and engineering domains such as geophysics, cosmology, chemistry, and computer graphics. All of these domains require the processing of functions defined on the surface of the sphere. With the advent and increased success of machine-learning in these domains, there is a growing demand to generalize state-of-the-art architectures to spherical data [13, 18, 19, 34, 6]. This is particularly important in scientific applications which require the solution of time-dependent partial differential equations which are often approximated with auto-regressive models. These models are particularly sensitive to spurious artifacts which arise from a non-geometrical treatment as these tend to build up and compromise stability [6].

While spherical convolutional neural networks (CNNs) and other geometric deep learning models have advanced the state of the art for learning on the sphere by respecting its symmetries and topology, these architectures are primarily designed for capturing local interactions and equivariant representations [13, 34, 30]. However, many scientific and engineering problems on the sphere, such as weather and climate modeling, 360° perception, and cortical surface analysis, require modeling complex, long-range dependencies that are not efficiently handled by convolutional approaches [7, 3, 9]. Transformers, with their global attention mechanism, have revolutionized learning in Euclidean domains by enabling flexible, data-driven modeling of both local and global relationships

^{*}Corresponding author.

[†]Equal contribution.

[15, 38, 4, 24]. Extending Transformers to spherical domains promises to combine the geometric fidelity of spherical models with the expressive power of attention, enabling new capabilities for spherical data analysis.

Recent works have begun to explore this direction, demonstrating that Transformer-based models tailored to spherical geometry can outperform traditional methods on tasks such as depth estimation or semantic segmentation by directly leveraging spherical geometry in their design [3, 10]. However, due to their reliance on permutation equivariance on icosahedral grids, they fail to capture the full rotational equivariance associated with the sphere. Other works have successfully applied Euclidean Transformer architectures to weather forecasting on the sphere [4], however, they show deterioration in longer auto-regressive rollouts [26]. This motivates the development of spherical attention mechanisms that operate natively on the sphere, unlocking the full potential of Transformer architectures for scientific and engineering applications involving spherical data.

Our contribution We generalize the attention mechanism to the spherical domain. To respect rotational symmetry, we derive a continuous formulation on the sphere for both global attention and neighborhood attention mechanisms. A discrete formulation is then derived using quadrature rules to approximate kernel integrals, thus ensuring approximate equivariance w.r.t. three-dimensional rotations. Both spherical attention variants are implemented in PyTorch, including custom CUDA extensions of the spherical neighborhood attention. We demonstrate the effectiveness of our approach by generalizing standard Transformer architectures to the spherical domain using our formulation and scoring them against their Euclidean counterparts in three different spherical benchmark problems.

Our implementation and code to reproduce experiments are available in the open-source library `torch-harmonics`.

2 Related work

Transformers Attention mechanism and Transformer architectures were initially developed for sequence modeling applications such as language models [37] and later applied to vision tasks in the Euclidean domain [15]. This has proven to be an effective method despite the lack of inductive biases such as locality [4, 24]. The attention mechanism is equivariant with respect to permutations of the input tokens and therefore lacks any notion of the underlying topology of the domain. This makes position embeddings necessary, which introduce the notion of locality. A notable advantage of the attention mechanism is its ability to capture long-range interactions. However, this makes it prohibitively expensive for large resolutions due to its $\mathcal{O}(N^2)$ asymptotic complexity, especially for multi-dimensional data where the number of pixels N grows exponentially with dimension limiting the resolution.

Neighborhood Transformers Hassani et al. [22] address this by simultaneously introducing the notion and inductive bias of locality into the architecture. This is achieved by using the neighborhood attention mechanism. In this approach, the attention layer is constrained to a neighborhood of grid points around the query point. This has great computational benefits as it lowers the asymptotic complexity of the attention mechanism to $\mathcal{O}(kN)$ where k is the kernel size of the neighborhood. While this method is highly efficient [23], it is generally only formulated for equidistant Cartesian grids. For spherical problems, this leads to distorted neighborhoods on spherical geometries.

Equivariant architectures Several convolutional architectures address this problem and capture the geometric properties alongside $SO(3)$ rotational symmetry [12, 17, 13, 11, 34, 6]. It has not been addressed by Transformer architectures yet. Finally, Assaad et al. [2] introduce a rotationally equivariant Transformer architecture in the context of vector neurons using the Frobenius norm.

Graph equivariant Transformers Various equivariant transformer architectures have been proposed in the literature. Notable examples are the $SE(3)$ -transformer [20] and the Equiformer architecture [29], and related architectures [39, 8]. However, these architectures focus on vector-valued features as inputs, rather than entire functions defined over a subdomain. Examples are positions of atoms in molecules and LiDAR point clouds. As such, the attention mechanisms in these architectures aim to preserve both distances of positions with each other and the relative orientation of the vectors.

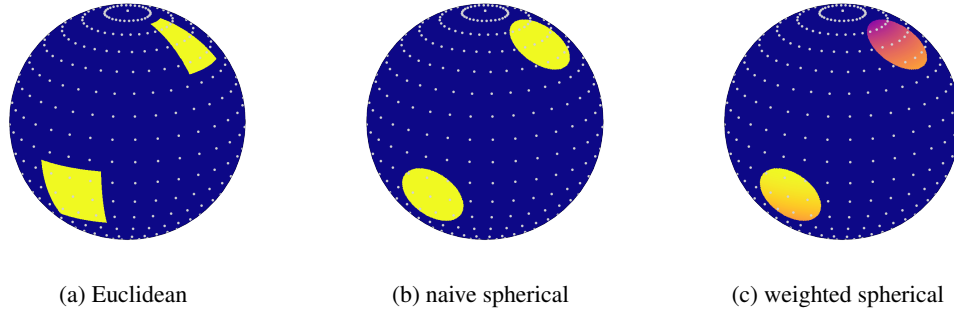


Figure 1: Neighborhood attention on the sphere: (a) Euclidean neighborhoods use planar distances, distorting receptive fields near poles. (b) Naive spherical attention uses geodesic neighborhoods but equally weights points, oversampling points and breaking $SO(3)$ symmetry. (c) Our method introduces quadrature weights to account for non-uniform sampling density, preserving rotational equivariance regardless of grid alignment through proper discretization of the continuous formulation.

Graph transformers applied to spherical functions Several graph-transformer models have been introduced that operate on icosahedral discretizations of the sphere [10, 9, 3], enabling the application of transformer architectures to function-valued data on spherical domains. While this approach has shown strong performance for spherical signals, it relies on the standard attention mechanism, which ensures only permutation equivariance with respect to the underlying graph. Consequently, these architectures exploit discrete graph symmetries rather than the continuous rotational symmetry of the sphere itself.

3 Method

3.1 Attention mechanism and kernel regression

We revisit the motivation behind the original attention mechanism. For two one-dimensional signals $q, k : \mathbb{R}^+ \rightarrow \mathbb{R}$, we can compute the correlation function $C[q, k](x, x') = \text{corr}(q(x), k(x'))$ using the Pearson correlation $\text{corr}(\cdot, \cdot)$, which measures the statistical dependence between signal pairs as a function of the point-pair (x, x') . Many stochastic processes, images and other spatial signals have distinct auto-correlation functions, which can be modeled using a kernel function $\alpha(x, x')$.

This analogy between correlation functions and kernel functions is exploited in nonparametric regression techniques such as kernel regression. For query/key signals $q, k : \mathbb{R}^+ \rightarrow \mathbb{R}^d$, and value signal $v : \mathbb{R}^+ \rightarrow \mathbb{R}^e$, the kernel regressor is given by

$$r(q(x)) = \int_{\mathbb{R}^+} \frac{\alpha(q(x), k(x'))}{\int_{\mathbb{R}^+} \alpha(q(x), k(x')) dx'} v(x') dx', \quad (1)$$

where $\alpha(\cdot, \cdot)$ is a suitable kernel function. For instance, by choosing $q(x) = x$, $k(x') = x'$ and the Gaussian kernel $\alpha(x, x') = \exp(-(x - x')^2/h^2)$ of width h , we recover the radial-basis regressor.

The attention mechanism [37] generalizes this concept further through learnable kernel transformations. The continuous-domain attention mechanism

$$\text{Attn}[q, k, v](x) = \int_{\mathbb{R}^+} A[q, k](x, x') v(x') dx', \quad (2a)$$

$$A[q, k](x, x') = \frac{\exp(q^T(x) \cdot k(x')/\sqrt{d})}{\int_{\mathbb{R}^+} \exp(q^T(x) \cdot k(x')/\sqrt{d}) dx'}, \quad (2b)$$

corresponds to exponential kernel smoothing, where the exponential kernel $\alpha(q, k) = \exp(-q^T k/\sqrt{d})$ measures similarity between query-key pairs while the denominator normalizes attention weights as probability distributions. The linearized variant [36] omits softmax normalization,

analogous to using unnormalized kernel weights in regression numerators. Modern attention mechanisms extend this kernel method by learning adaptive correlation patterns which are parameterized by the query and key weights, rather than relying on predefined kernel functions.

By discretizing the continuous attention mechanism (2) on the sequence $\{x_i\}_{i=1}^{N_{\text{seq}}}$, we recover the standard attention mechanism

$$\text{Attn}[q, k, v](x) = \sum_{j=1}^{N_{\text{seq}}} \frac{\exp\left(q^T(x_i) \cdot k(x_j)/\sqrt{d}\right)}{\sum_{l=1}^{N_{\text{seq}}} \exp\left(q^T(x_i) \cdot k(x_l)/\sqrt{d}\right)} v(x_j), \quad (3)$$

commonly found in sequence modeling [37] and other applications. This formulation is permutation-equivariant, in the sense that a permutation of the tokens $q(x_i)$, $k(x_i)$ and $v(x_i)$ will result in the same output but permuted.

3.2 Attention on the sphere

The generalization of the attention mechanisms for spherical data requires a proper treatment of spherical geometry and topology. To do so, we change the domain in the continuous formulation (2) to the sphere and obtain

$$\text{Attn}_{S^2}[q, k, v](x) = \int_{S^2} A_{S^2}[q, k](x, x') v(x') d\mu(x'), \quad (4a)$$

$$A_{S^2}[q, k](x, x') = \frac{\exp\left(q^T(x) \cdot k(x')/\sqrt{d}\right)}{\int_{S^2} \exp\left(q^T(x) \cdot k(x'')/\sqrt{d}\right) d\mu(x'')}, \quad (4b)$$

for inputs $q, k : S^2 \rightarrow \mathbb{R}^d$, $v : S^2 \rightarrow \mathbb{R}^e$. $\mu : S^2 \rightarrow \mathbb{R}_0^+$ denotes the invariant Haar measure on the sphere and ensures that the integrals in (4) remain unchanged under three-dimensional rotations $R \in \text{SO}(3)$ of the variables of integration. This makes the attention mechanism (4) equivariant, i.e., $\text{Attn}_{S^2}[q, k, v](R^{-1}x) = \text{Attn}_{S^2}[q', k', v'](x)$ for rotated signals $q'(x) = q(R^{-1}x)$, $k'(x) = k(R^{-1}x)$, $v'(x) = v(R^{-1}x)$.

A local, neighborhood attention variant of (4) can be derived by choosing a compactly supported kernel α . We define

$$N_{S^2}[q, k](x, x') = \frac{\mathbb{1}_{D(x)}(x') \exp\left(q^T(x) \cdot k(x')/\sqrt{d}\right)}{\int_{S^2} \mathbb{1}_{D(x)}(x'') \exp\left(q^T(x) \cdot k(x'')/\sqrt{d}\right) d\mu(x'')}, \quad (5)$$

where $\mathbb{1}_{D(x)}(x')$ is the indicator function of the spherical disk $D(x) = \{x' \in S^2 | d(x, x')_{S^2} \leq \theta_{\text{cutoff}}\}$ centered at x . This disk is determined by the geodesic distance (great circle/Haversine distance) $d(\cdot, \cdot)_{S^2}$ and the hyperparameter θ_{cutoff} which controls the size of the neighborhood. By replacing the attention operator A_{S^2} with N_{S^2} , we obtain the spherical neighborhood attention

$$\text{NAttn}_{S^2}[q, k, v](x) = \int_{S^2} N_{S^2}[q, k](x, x') v(x') d\mu(x'), \quad (6)$$

which retains the equivariance property of (4).

To obtain discrete attention mechanisms, we discretize the spherical domain with a set of grid points $\{x_i \in S^2 \text{ for } i = 1, 2, \dots, N_{\text{grid}}\}$ and apply a suitable quadrature formula $\omega_i := \omega(x_i)$, such that

$$\int_{S^2} u(x) dx \approx \sum_{i=1}^{N_{\text{grid}}} u(x_i) \omega_i. \quad (7)$$

This allows a numerical evaluation of the integrands in (4) and (6) giving us the discrete full and neighborhood attention (see Appendix B.3 for additional detail).

Figure 1 visualizes our approach to neighborhood attention on spherical data. By deriving the continuous formulation through quadrature rules and rigorously incorporating geodesic distances, we establish our architecture as a neural operator. This formulation enables consistent evaluation

across arbitrary discretizations while preserving geometric structure. Crucially, the quadrature-based implementation ensures approximate $SO(3)$ equivariance, maintaining transformation consistency under 3D rotations. To validate the equivariance of our implementation, we conduct a convergence test in Appendix D that demonstrates convergence of the approximate equivariance error in relation to the interpolation error as resolution is increased.

3.3 Implementation

We implement both spherical attention mechanisms for equiangular and Gaussian grids and make them publicly available in torch-harmonics, a library for machine learning and signal processing on the sphere. The global spherical attention may be efficiently implemented using the PyTorch scaled dot product attention implementation by leveraging the attention mask to incorporate the quadrature weights. The neighborhood variant, on the other hand, requires a custom CUDA implementation due to the non-uniform memory access patterns implied by the sparse attention support. Details regarding the implementation and expressions of input gradients are provided in Appendix B.4.

3.4 Spherical Transformer architecture

We apply the spherical attention mechanisms derived in Section 3 in the context of popular Transformer architectures. To this end, we modify the Vision Transformer (ViT) [15] to obtain a fully spherical architecture that attains equivariance properties. To do so, we replace non-spherical operations with spherical counterparts and derive the S^2 Transformer and S^2 neighborhood Transformer architectures (see Figure 2). A similar approach can be taken to obtain spherical variants of the SegFormer architecture [38]; i.e. the S^2 SegFormer and S^2 neighborhood SegFormers, respectively. Due to their continuous formulations, all of these architectures are neural operators [28], allowing them to be realized on arbitrary discretizations of the sphere, that permit a suitable quadrature rule.

In the following, we outline the methodology for deriving the spherical Transformers.

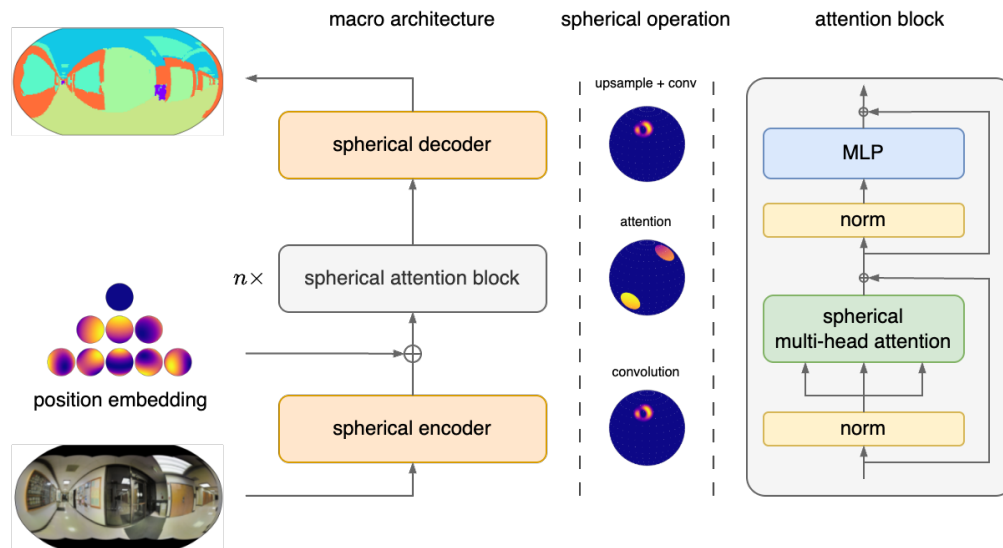


Figure 2: Illustration of the spherical Transformer architecture.

Encoding Vision Transformers typically use patch embeddings which can be regarded as convolutions with a special choice of pre-set, sparse weights. As such, we employ a convolutional embedding scheme based on the discrete-continuous spherical convolutions [34]. Unlike standard patch embeddings, our approach leverages spherical convolutions to ensure that the resulting representation remains a function defined on S^2 preserving equivariance properties of the model with respect to rotations. Moreover, the discrete-continuous formulation makes these convolutions neural operators [30], enabling flexible discretization.

Spherical attention blocks The latent representation is refined by a sequence of n spherical attention blocks (Fig. 2, right). Each block applies spherical multi-head attention followed by a multilayer perceptron (MLP), both with residual connections and preceded by instance normalization, forming a pre-norm architecture. Positional embeddings (defined in Section 3.4) are added before the multi-head attention step. The spherical attention can operate either globally across the entire sphere, as defined in Eq. (4), or locally, as shown in Eq. (6).

Decoding The final latent is passed through a decoder, which mirrors the encoding step. Bilinear spherical interpolation is performed to upsample the signal to the full resolution. This is followed by a spherical discrete-continuous convolution, which produces the output.

Position embeddings Transformer architectures typically incorporate positional embeddings to provide information about the location of each input token. This is necessary because the attention mechanism (3) is inherently permutation-invariant and thus unaware of spatial structure. We experimented with several possibilities including learnable [15], sequence-based [37], and spectral position embeddings [35]. In the end, we chose the spectral embedding, where the k -th channel is determined by the real-valued spherical harmonics $Y_\ell^m(x)$, with $\ell = \lfloor \sqrt{k} \rfloor$ and $m = k - \ell(\ell + 1)$.

The resulting spherical Transformer architecture is naturally a neural operator, as all spatial operations can be formulated in the continuous domain, as well as for arbitrary grid-resolutions which permit the accurate numerical computation of integrals over the sphere. Finally, the same design modifications are applied to the SegFormer architecture [38] to obtain the spherical SegFormer and spherical neighborhood SegFormer. These architectures are made available as part of torch-harmonics.

4 Experiments

We evaluate our method by adapting the popular Vision Transformer (ViT) and SegFormer architecture to equirectangular grids on the spherical domain. We follow the procedure described in Section 3.4, replacing the attention and convolution layers with their spherical counterparts while maintaining equivalent architectural configurations, such as depth and embedding size. In particular, receptive fields of neighborhood attention mechanisms and convolutions are carefully chosen to match the covered areas. This allows for a fair comparison of models, as models retain similar parameter counts, as listed in Table 5 in the appendix. A comprehensive study comparing spherical Transformers to their corresponding Euclidean baseline is carried out for both global and local (neighborhood) attention variants. Implementation details are provided in Appendix C.1.

4.1 Segmentation on the sphere

The Stanford 2D3DS Dataset [1] provides spherical images taken indoors in a university setting. It provides a total of 1621 RGB images along with a number of output pairs including segmentation data containing 14 classes, such as *clutter*, *ceiling*, *window*, etc. For the purpose of training, we downsample the data to a resolution of 128x256 and split the dataset into 95% train, 2.5% test and 2.5% validation parts. Training is carried using uniform weighted cross entropy loss for 200 epochs using the ADAMW optimizer [31], a learning rate of 0.5×10^{-4} and a cosine scheduler. Furthermore, to reduce overfitting, weight decay and dropout path rate are both set to 0.1. On a single NVIDIA RTX 6000 Ada, training took between 12 and 215 minutes depending on the respective models. The results are validated using the Accuracy (Acc) and Intersection over Union (IoU) metrics [25] on the validation dataset which contains 35 samples. For a detailed discussion refer to Appendix A.2.

Table 1 reports the results of our experiments. For both transformer and SegFormer models, we see the spherical variants outperforming their Euclidean counterparts, where the Transformer models have compatible hyperparameters across spherical and euclidean variants. We train EGFormer [40] and SphericalCNN [14] to serve as additional geometric baselines. We note that the EGFormer seems to overfit and may require different hyperparameters than the other architectures. Moreover, we report results for the SphereUFormer [3] alongside high-resolution variants of our neighborhood SegFormers with double the embedding dimension. We observe that our S^2 SegFormer with a locally supported spherical attention mechanism outperforms the other models in the validation metrics.

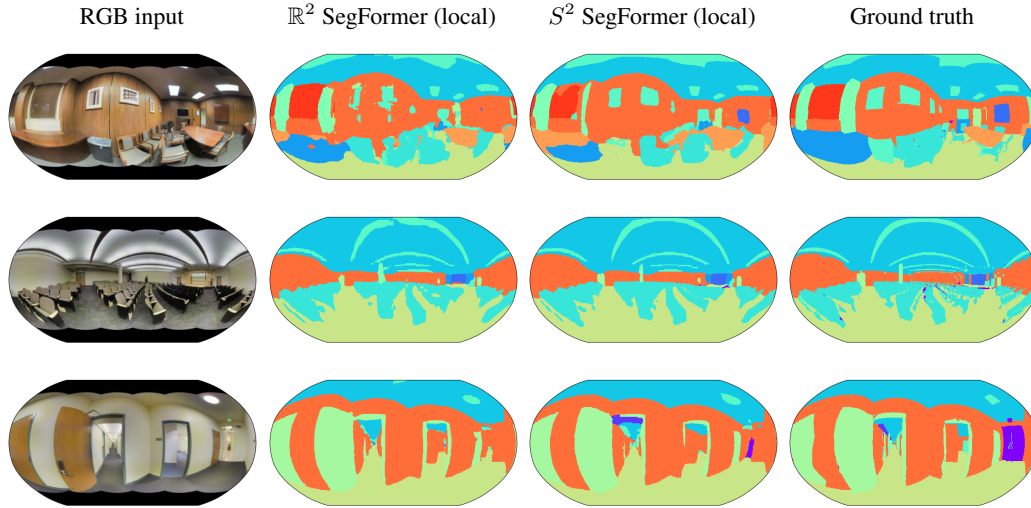


Figure 3: Samples from the segmentation dataset with Euclidean and S^2 neighborhood SegFormers.

Table 1: Numerical results on the segmentation dataset. Cross entropy training and validation loss alongside Intersection-over-Union (IoU) and Accuracy (Acc) metrics are reported.

Model	Training loss ↓	Validation loss ↓	IoU ↑	Acc ↑
Results on 128×256 data				
EGFormer [40]	0.203	1.29 ± 0.487	0.586 ± 0.106	0.962 ± 0.012
SphericalCNN [14]	0.382	0.775 ± 0.300	0.622 ± 0.099	0.966 ± 0.011
$\mathbb{R}^2$ Transformer	0.325	1.041 ± 0.812	0.593 ± 0.151	0.962 ± 0.018
S^2 Transformer	0.312	1.016 ± 0.499	0.588 ± 0.111	0.962 ± 0.013
$\mathbb{R}^2$ Transformer (local)	0.344	0.854 ± 0.352	0.619 ± 0.103	0.966 ± 0.011
S^2 Transformer (local)	0.298	0.852 ± 0.369	0.622 ± 0.106	0.966 ± 0.012
$\mathbb{R}^2$ SegFormer	0.296	0.796 ± 0.363	0.636 ± 0.109	0.967 ± 0.012
S^2 SegFormer	0.307	0.693 ± 0.303	0.657 ± 0.106	0.970 ± 0.011
$\mathbb{R}^2$ SegFormer (local)	0.288	0.782 ± 0.334	0.637 ± 0.101	0.968 ± 0.011
S^2 SegFormer (local)	0.320	0.667 ± 0.309	0.667 ± 0.110	0.971 ± 0.012
Results on 256×512 data				
SphereUFormer [3]	-	-	0.722	0.886
$\mathbb{R}^2$ Segformer (local, large)	0.184	0.740 ± 0.858	0.712 ± 0.153	0.974 ± 0.017
S^2 SegFormer (local, large)	0.189	0.554 ± 0.550	0.748 ± 0.126	0.979 ± 0.013

Figure 3 depicts results from the best performing local S^2 SegFormer architecture alongside its Euclidean counterpart. It demonstrates an improvement in the quality of segmentation masks, when compared to the ground truth target.

4.2 Depth estimation on the sphere

The Stanford 2D3DS dataset provides depth maps alongside segmentation data, which we use to train our models on the depth estimation task. Models are trained for 100 epochs using the $\mathcal{L}_{\text{depth}}$ objective functions (16) which combines L_1 and Sobolev $W^{1,1}$ losses, to better preserve image sharpness. The top and bottom 15% are masked out as there is no valid target data in this area. Training times vary between 11 and 220 minutes on a single NVIDIA RTX6000 Ada GPU depending on the architecture.

The numerical results alongside individual samples from the Transformer architectures are provided in Table 2 and Figure 4. We observe improved validation losses and $W^{1,1}$ error for spherical architectures (see Appendix A.2 for detail on the losses and norms).

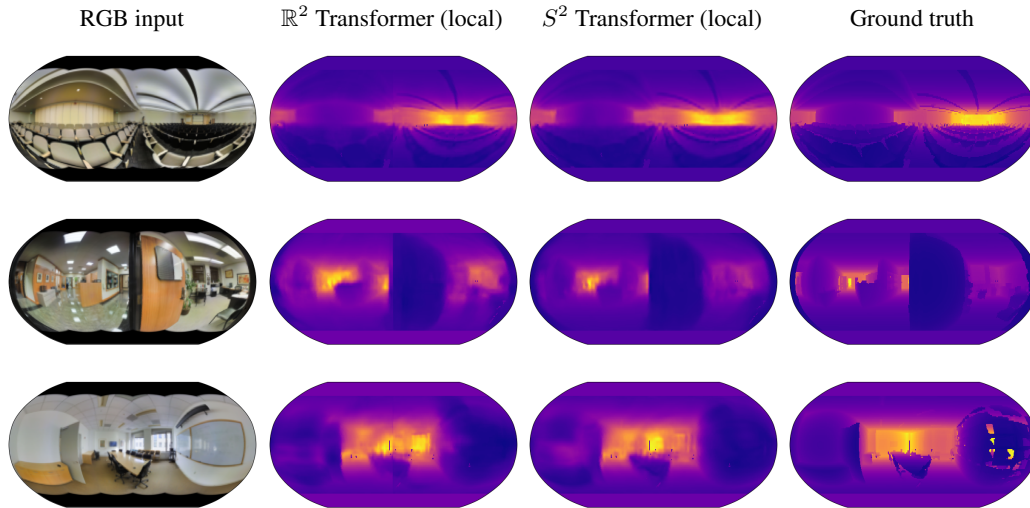


Figure 4: Sample predictions from our transformers alongside ground truth data from the spherical depth estimation task.

Table 2: Numerical results on the depth estimation dataset. Training and validation errors, alongside errors for various cartesian Transformer variants and their spherical counterparts.

Model	Training loss ↓	Validation loss ↓	L_1 error ↓	$W^{1,1}$ error ↓
$\mathbb{R}^2$ Transformer	0.954	0.982 ± 0.386	0.215 ± 0.084	7.67 ± 3.22
S^2 Transformer	0.978	0.993 ± 0.381	0.234 ± 0.086	7.59 ± 3.23
$\mathbb{R}^2$ Transformer (local)	1.014	1.023 ± 0.370	0.237 ± 0.079	7.86 ± 3.17
S^2 Transformer (local)	0.978	0.976 ± 0.354	0.209 ± 0.068	7.68 ± 3.12
$\mathbb{R}^2$ SegFormer	1.037	0.980 ± 0.371	0.169 ± 0.058	8.11 ± 3.22
S^2 SegFormer	1.005	0.935 ± 0.369	0.165 ± 0.057	7.70 ± 3.23
$\mathbb{R}^2$ SegFormer (local)	1.055	1.002 ± 0.374	0.174 ± 0.061	8.27 ± 3.25
S^2 SegFormer (local)	1.014	0.948 ± 0.368	0.174 ± 0.061	7.75 ± 3.21
additional baselines				
Spherical CNN [14]	1.019	1.098 ± 0.412	0.315 ± 0.123	7.83 ± 3.21
LSNO [30]	0.994	1.020 ± 0.387	0.248 ± 0.095	7.72 ± 3.19
EGFormer [40]	0.831	0.945 ± 0.386	0.182 ± 0.084	7.63 ± 3.26

4.3 Shallow water equations on the rotating sphere

The shallow water equations are a system of partial differential equations that describe the dynamics of a thin fluid layer on a rotating sphere. As such, this system is a useful model for geophysical fluid flows such as atmospheric dynamics or ocean dynamics [21, 5]. We use the dataset as presented in [6]. Initial conditions are sampled from a Gaussian random process and reference solutions are generated on the fly using a spectral solver. The solutions are computed at a resolution of 128×256 on a Gaussian grid, on a sphere which matches the Earth’s radius and angular velocity. The target solution is set at a lead time of 30 minutes, which requires 12 time-steps of the numerical solver to compute the target due to time-stepping restrictions. A detailed explanation is provided in Appendix C.2.

All architectures are trained using solutions output by a traditional spectral solver, with 12 classical time-steps steps to provide the single prediction step at a learning rate of 10^{-3} . All of the above is carried out using the ADAM optimizer [27] and using a ReduceLROnPlateau learning rate scheduler. On an NVIDIA A6000 GPU, training took approximately between 24 and 34 minutes depending on the model. To evaluate the models, we compared the classically-solved solution with model predictions after a single prediction step and five autoregressive prediction steps.

Table 3: Numerical results on the shallow water equations on the rotating sphere. Validation errors are reported after a single and after five autoregressive steps.

Model	Training loss ↓	Validation loss ↓	L_1 error ↓	L_2 error ↓
1 step				
$\mathbb{R}^2$ Transformer	0.051	0.051 ± 0.002	0.148 ± 0.003	0.196 ± 0.005
S^2 Transformer	0.003	0.003 ± 0.000	0.042 ± 0.001	0.054 ± 0.001
$\mathbb{R}^2$ Transformer (local)	0.015	0.016 ± 0.001	0.075 ± 0.002	0.115 ± 0.004
S^2 Transformer (local)	0.003	0.003 ± 0.000	0.040 ± 0.001	0.050 ± 0.001
$\mathbb{R}^2$ SegFormer	0.087	0.085 ± 0.003	0.214 ± 0.004	0.281 ± 0.005
S^2 SegFormer	0.039	0.038 ± 0.001	0.147 ± 0.003	0.193 ± 0.004
$\mathbb{R}^2$ SegFormer (local)	0.052	0.050 ± 0.002	0.163 ± 0.003	0.219 ± 0.005
S^2 SegFormer (local)	0.034	0.033 ± 0.001	0.138 ± 0.002	0.181 ± 0.004
5 steps				
$\mathbb{R}^2$ Transformer	-	36.986 ± 9.390	3.282 ± 0.367	5.237 ± 0.605
S^2 Transformer	-	0.062 ± 0.006	0.190 ± 0.009	0.239 ± 0.011
$\mathbb{R}^2$ Transformer (local)	-	5.667 ± 0.566	0.933 ± 0.044	1.987 ± 0.088
S^2 Transformer (local)	-	0.033 ± 0.005	0.137 ± 0.010	0.172 ± 0.012
$\mathbb{R}^2$ SegFormer	-	0.680 ± 0.025	0.618 ± 0.012	0.781 ± 0.015
S^2 SegFormer	-	0.154 ± 0.013	0.304 ± 0.013	0.386 ± 0.017
$\mathbb{R}^2$ SegFormer (local)	-	0.546 ± 0.024	0.544 ± 0.013	0.692 ± 0.016
S^2 SegFormer (local)	-	0.104 ± 0.008	0.250 ± 0.010	0.319 ± 0.013

Based on the outcomes of this experiment as reported in Table 3, we highlight the significantly reduced validation loss of the spherical architectures compared to their Euclidean counterparts. This is reflected in the single- and five-step results in the table and visualized in Figure 5, which depicts a drastic reduction in distortion errors towards the poles for the S^2 neighborhood Transformer over its Euclidean counterparts. We note that training on larger time-steps caused all transformer models (including the S^2 models) to be unstable on autoregressive roll-outs, which non-transformer S^2 -optimized convolutional, neural-operator models reported in [6, 30] do not seem to suffer from.

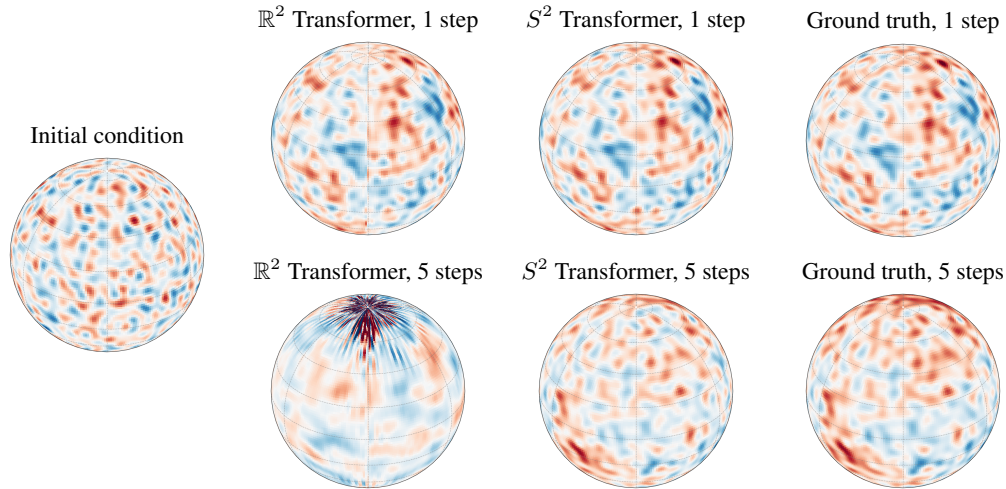


Figure 5: Comparison of single step and five-step autoregressive predictions of both local $\mathbb{R}^2$ and S^2 Transformers to a ground truth solution of the shallow water equations.

4.4 Ablation study

We conduct an ablation study on the shallow water equations setting to assess the contribution of individual architectural choices. Starting from the general-purpose vision transformer [15], we incrementally introduce spherical convolution, spherical attention, and local attention to evaluate

their individual and combined effects. Training losses and validation metrics after 5 autoregressive steps are reported in Table 4. The results imply that incorporating the inductive biases of spherical geometry and locality into encoding and attention layers is highly effective in obtaining competitive performance on the given task and other geophysical settings. We remark that the changed from learned position embeddings to spectral embeddings decreases the skill as it constitutes a significant reduction in trainable parameters. Moreover, some improvements are only notable as you after multiple autoregressive steps, such as the changes from global $\mathbb{R}^2$ to local S^2 attention.

Table 4: Ablation results on the shallow water equations on the rotating sphere. Validation reports are reported for 5 autoregressive steps.

Model	Training loss ↓	Validation loss ↓	L_1 error ↓	L_2 error ↓
ViT [15]	0.463	1.202 ± 0.043	0.805 ± 0.014	1.013 ± 0.017
– patch embeddings				
+ $\mathbb{R}^2$ conv. encoder	0.042	6.753 ± 0.948	1.384 ± 0.060	2.210 ± 0.137
– learned pos. embed.				
+ spectral pos. embed.	0.053	40.361 ± 8.379	3.275 ± 0.310	5.285 ± 0.529
– $\mathbb{R}^2$ conv. encoder				
+ S^2 conv. encoder	0.003	0.079 ± 0.011	0.208 ± 0.013	0.261 ± 0.016
– $\mathbb{R}^2$ attention				
+ S^2 attention	0.003	0.062 ± 0.007	0.193 ± 0.011	0.242 ± 0.013
– S^2 global attention				
+ S^2 local attention	0.003	0.034 ± 0.006	0.141 ± 0.012	0.177 ± 0.014

5 Limitations and future work

The unstructured nature of the neighborhoods in our S^2 method requires custom CUDA kernels that are well optimized, and show increased performance over the full attention methods using heavily optimized SDPA kernels available within PyTorch. The NATTEN [23] package used in our baseline comparisons provides optimized neighborhood attention kernels (for a fixed, rectangular neighborhood size), including support for lower precision data types, such as Float16 and BFloat16. Our custom kernel employs vectorization over features and optimized memory access but utilizes 32 bit floating point type. We will investigate if our kernel can benefit from lower precision data types as well. Furthermore, we aim at implementing distributed memory support for both global and neighborhood spherical attention layers, similar to other layers implemented in torch-harmonics.

6 Conclusion

We have generalized the attention mechanism in its continuous formulation to the two-dimensional sphere, obtaining an equivariant formulation. Moreover, a spherical variant of neighborhood attention using geodesic distance has been derived. These approaches are implemented as neural operators via quadrature rules for practical spherical discretizations.

In conjunction with other spherical signal-processing techniques, we obtain spherical counterparts of popular Vision Transformer architectures such as the ViT, Neighborhood Transformer, and SegFormer. The effectiveness of our method has been demonstrated for three distinct learning tasks. The new spherical Transformers consistently outperform their Euclidean counterparts, demonstrating the importance of geometrically faithful treatment of such problems. Considering the growing importance of transformer architectures in scientific and engineering domains, we foresee this method becoming a valuable approach for researchers and practitioners seeking to advance the state of the art.

Acknowledgments

We are especially grateful to our colleague Mauro Bisson for helping us write and optimize CUDA kernels. Moreover, we are grateful to Alexander Keller, J.P. Lewis, and Vishal Mehta for fruitful discussions and proofreading the manuscript.

References

- [1] Iro Armeni, Sasha Sax, Amir R. Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.
- [2] Serge Assaad, Carlton Downey, Rami Al-Rfou, Nigamaa Nayakanti, and Ben Sapp. Vn-transformer: Rotation-equivariant attention for vector neurons. 6 2022. URL <http://arxiv.org/abs/2206.04176>.
- [3] Yaniv Benny and Lior Wolf. SphereUFormer: A u-shaped transformer for spherical 360 perception. 12 2024. URL <http://arxiv.org/abs/2412.06968>.
- [4] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619:533–538, 7 2023. ISSN 14764687. doi: 10.1038/s41586-023-06185-3.
- [5] Boris Bonev, Jan S. Hesthaven, Francis X. Giraldo, and Michal A. Kopera. 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.
- [6] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. *Proceedings of the 40th International Conference on Machine Learning*, 202:2806–2823, 6 2023. URL <http://arxiv.org/abs/2306.03838>.
- [7] Salva Rühling Cachay, Peetak Mitra, Haruki Hirasawa, Sookyoung Kim, Subhashis Hazarika, Dipti Hingmire, Phil Rasch, Hansi Singh, and Kalai Ramea. Climformer-a spherical transformer model for long-term climate projections. In *Proceedings of the Machine Learning and the Physical Sciences Workshop, NeurIPS 2022*, 2022.
- [8] Evangelos Chatzipantazis, Stefanos Pertigkiozoglou, Edgar Dobriban, and Kostas Daniilidis. Se(3)-equivariant attention networks for shape reconstruction in function space. 2 2023. URL <http://arxiv.org/abs/2204.02394>.
- [9] Jiale Cheng, Xin Zhang, Fenqiang Zhao, Zhengwang Wu, Xinrui Yuan, John H. Gilmore, Li Wang, Weili Lin, and Gang Li. Spherical transformer on cortical surfaces. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 13583 LNCS, pages 406–415. Springer Science and Business Media Deutschland GmbH, 2022. ISBN 9783031210136. doi: 10.1007/978-3-031-21014-3_42.
- [10] Sungmin Cho, Raehyuk Jung, and Junseok Kwon. Spherical transformer. 2 2022. URL <http://arxiv.org/abs/2202.04942>.
- [11] Oliver J. Cobb, Christopher G. R. Wallis, Augustine N. Mavor-Parker, Augustin Marignier, Matthew A. Price, Mayeul d’Avezac, and Jason D. McEwen. Efficient generalized spherical CNNs. 10 2020. URL <http://arxiv.org/abs/2010.11661>.
- [12] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. 2 2016. URL <http://arxiv.org/abs/1602.07576>.
- [13] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns. *International Conference on Learning Representations*, 1 2018. URL <http://arxiv.org/abs/1801.10130>.
- [14] Michaël Defferrard, Martino Milani, Frédéric Gusset, and Nathanaël Perraudin. DeepSphere: a graph-based spherical CNN. In *International Conference on Learning Representations (ICLR)*, 2020. URL <https://openreview.net/forum?id=B1e301StPB>.
- [15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020. URL <http://arxiv.org/abs/2010.11929>.

- [16] J.R. Driscoll and D.M. Healy. 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>.
- [17] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning $so(3)$ equivariant representations with spherical cnns. 11 2017. URL <http://arxiv.org/abs/1711.06721>.
- [18] Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. Spin-weighted spherical cnns. *Advances in Neural Information Processing Systems*, pages 8614–8625, 2020. URL <http://arxiv.org/abs/2006.10731>.
- [19] Carlos Esteves, Jean-Jacques Slotine, and Ameesh Makadia. Scaling spherical cnns. *Proceedings of the 40th International Conference on Machine Learning*, pages 9396–9411, 6 2023. URL <http://arxiv.org/abs/2306.05420>.
- [20] Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. 11 2020. URL <http://arxiv.org/abs/2006.10503>.
- [21] Francis X Giraldo. A spectral element shallow water model on spherical geodesic grids. Technical report, 2001.
- [22] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6185–6194, 2023. URL <http://arxiv.org/abs/2204.07143>.
- [23] Ali Hassani, Wen-Mei Hwu, and Humphrey Shi. Faster neighborhood attention: Reducing the $o(n^2)$ cost of self attention at the threadblock level. *Neural Information Processing Systems*, 2024. URL <http://arxiv.org/abs/2403.04690>.
- [24] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. 5 2024. URL <http://arxiv.org/abs/2405.19101>.
- [25] Pavel Iakubovskii. Segmentation models. https://github.com/qubvel/segmentation_models.pytorch, 2019.
- [26] Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R. Durran, Raul A. Moreno, Thorsten Kurth, Boris Bonev, Noah Brenowitz, and Martin V. Butz. Advancing parsimonious deep learning weather prediction using the healpix mesh. *Journal of Advances in Modeling Earth Systems*, 16, 8 2024. ISSN 19422466. doi: 10.1029/2023MS004021.
- [27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 12 2014. URL <http://arxiv.org/abs/1412.6980>.
- [28] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. 8 2021. URL <http://arxiv.org/abs/2108.08481>.
- [29] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. 2 2023. URL <http://arxiv.org/abs/2206.11990>.
- [30] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. 2 2024. URL <http://arxiv.org/abs/2402.16845>.
- [31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *In International Conference on Learning Representations*, 2019. URL <http://arxiv.org/abs/1711.05101>.
- [32] Jason. D. McEwen and Yves Wiaux. A novel sampling theorem on the sphere. 10 2011. doi: 10.1109/TSP.2011.2166394. URL <http://arxiv.org/abs/1110.6298><http://dx.doi.org/10.1109/TSP.2011.2166394>.

- [33] Ramachandran D. Nair, Stephen J. Thomas, and Richard D. Loft. 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>.
- [34] Jeremy Ocampo, Matthew A. Price, and Jason D. McEwen. Scalable and equivariant spherical cnns by discrete-continuous (disco) convolutions. 9 2022. URL <http://arxiv.org/abs/2209.13603>.
- [35] Marc Rußwurm, Konstantin Klemmer, Esther Rolf, Robin Zbinden, and Devis Tuia. Geographic location encoding with spherical harmonics and sinusoidal representation networks. *International Conference on Learning Representations*, 10 2023. URL <http://arxiv.org/abs/2310.06743>.
- [36] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 6 2017. URL <http://arxiv.org/abs/1706.03762>.
- [38] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. 5 2021. URL <http://arxiv.org/abs/2105.15203>.
- [39] Yinshuang Xu, Dian Chen, Katherine Liu, Sergey Zakharov, Rares Ambrus, Kostas Daniilidis, and Vitor Guizilini. $se(3)$ equivariant ray embeddings for implicit multi-view depth estimation. 11 2024. URL <http://arxiv.org/abs/2411.07326>.
- [40] Zhihui Yang, Qingyong Zhang, Wanfeng Chang, Peng Xiao, and Minglong Li. Egformer: an enhanced transformer model with efficient attention mechanism for traffic flow forecasting. *Vehicles*, 6(1):120–139, 2024.

A Signals on the sphere

A.1 Grids and quadrature rules

Our method deals with real-valued signals $u : S^2 \rightarrow \mathbb{R}^n$ defined on the unit sphere

$$x(\vartheta, \varphi) = \begin{bmatrix} \sin \vartheta \cos \varphi \\ \sin \vartheta \sin \varphi \\ \cos \vartheta \end{bmatrix}. \quad (8)$$

Processing these signals requires discretizations by means of a sampling scheme. While there are various sampling theorems on the sphere [16, 32], we restrict our discussion to sampling schemes motivated by quadrature rules. Many of the operations discussed in this chapter require evaluating integrals of the form

$$\int_{S^2} u(x) \, d\mu(x) = \int_0^{2\pi} \int_0^\pi u(x(\vartheta, \varphi)) \sin \vartheta \, d\vartheta \, d\varphi, \quad (9)$$

where $u \in L^1(S^2)$ is an integrable function on the sphere and $\mu(x)$ denotes the Haar measure on the sphere. We approximate this integral using a quadrature rule

$$\int_{S^2} u(x) \, d\mu(x) \approx \sum_{i=1}^{n_{\text{grid}}} u(x_i) q_i, \quad (10)$$

which is characterized by a choice of grid points $\{x_i\}_{i=1}^{n_{\text{grid}}}$ and corresponding quadrature weights $\{q_i\}_{i=1}^{n_{\text{grid}}}$. While our methods are applicable to many types of grids, we typically use the equiangular grid $\{x(\vartheta, \varphi) \mid \forall \vartheta \in \{\vartheta_i\}, \varphi \in \{\varphi_j\}\}$ defined by

$$\vartheta_i = \pi i / n_{\text{lat}} \quad \text{for } i = 0, 1, \dots, n_{\text{lat}} - 1, \quad (11a)$$

$$\varphi_j = 2\pi j / n_{\text{lon}} \quad \text{for } j = 0, 1, \dots, n_{\text{lon}} - 1, \quad (11b)$$

where n_{lat} and n_{lon} denote the number of grid points in latitude and longitude. The associated quadrature weights are

$$q_{ij} = \frac{2\pi^2}{n_{\text{lat}} n_{\text{lon}}} \sin \vartheta_i, \quad (12)$$

and approximately sums to 4π . This choice of quadrature weights correspond to trapezoidal quadrature weights in spherical coordinates.

A.2 Loss functions and metrics

For two real-valued signals on the sphere $u, u^* : S^2 \rightarrow \mathbb{R}$, we define loss functions for the purpose of regression and classification tasks. All of the losses are integrated over the sphere which is numerically evaluated using a quadrature rule.

L_p norms The L_1 distance on the sphere is defined via the $L_1(S^2)$ norm as follows:

$$\mathcal{L}_1[u, u^*] = \|u - u^*\|_{L_1(S^2)} = \frac{1}{4\pi} \int_{S^2} |u - u^*| \, d\mu(x). \quad (13)$$

In the same fashion, the squared L_2 distance is defined as

$$\mathcal{L}_2^2[u, u^*] = \|u - u^*\|_{L_2(S^2)}^2 = \frac{1}{4\pi} \int_{S^2} (u - u^*)^2 \, d\mu(x). \quad (14)$$

Sobolev norm The spherical Sobolev $W^{1,1}$ semi-norm is defined as

$$\|u - u^*\|_{W^{1,1}(S^2)} = \frac{1}{4\pi} \int_{S^2} \left| \frac{1}{\sin \theta} \partial_\phi (u - u^*) \right| + |\partial_\theta (u - u^*)| \, d\mu(x). \quad (15)$$

The loss $\mathcal{L}_{\text{depth}}$ can be composed as a combination of the Sobolev $W^{1,1}$ semi-norm and the L_1 distance:

$$\mathcal{L}_{\text{depth}} = \|u - u^*\|_{L_1(S^2)} + \lambda \|u - u^*\|_{W^{1,1}(S^2)}, \quad (16)$$

with $\lambda = 0.1$ to approximately match the magnitudes of both components in the depth estimation task.

Cross entropy loss The cross entropy loss $\mathcal{L}_{\text{CE}}$ can be defined as follows

$$\mathcal{L}_{\text{CE}} = -\frac{1}{4\pi} \int_{S^2} \text{CrossEntropy}(u, u^*), \quad (17)$$

for C classes, and the predicted and ground truth functions u and u^* . The point-wise cross-entropy is given by

$$\text{CrossEntropy}(u, u^*) = -\sum_{c=1}^C u_c^* \log \left(\frac{\exp(\text{logits}(u_c))}{\sum_{i=1}^C \exp(\text{logits}(u_i))} \right), \quad (18)$$

and the logits operator is given by

$$\text{logits}(u) = \log \left(\frac{u}{1-u} \right). \quad (19)$$

Intersection over Union (IoU) IoU is a commonly used metric for segmentation quality that captures the correctly guessed segmentation classes normalized by the sum of the correct predictions and all false predictions. The IoU is given by

$$\text{IoU} = \frac{\sum_{c=1}^C T_p^{(c)}}{\sum_{c=1}^C T_p^{(c)} + F_p^{(c)} + F_n^{(c)}}, \quad (20)$$

where for a given class c , $T_p^{(c)}$, $T_n^{(c)}$, $F_p^{(c)}$, $F_n^{(c)}$ denote area fraction associated with true positives, true negatives, false positives, and false negatives, respectively.¹ Those can be computed from the predictions u_c and ground-truth values u_c^* via

$$T_p^{(c)} = \frac{1}{4\pi} \int_{S^2} u_c \cap u_c^* d\mu(x), \quad (21a)$$

$$F_p^{(c)} = \frac{1}{4\pi} \int_{S^2} u_c \cap u_{\neg c}^* d\mu(x), \quad (21b)$$

$$F_n^{(c)} = \frac{1}{4\pi} \int_{S^2} u_{\neg c} \cap u_c^* d\mu(x), \quad (21c)$$

$$T_n^{(c)} = \frac{1}{4\pi} \int_{S^2} u_{\neg c} \cap u_{\neg c}^* d\mu(x). \quad (21d)$$

$$(21e)$$

For each class c , those quantities satisfy

$$T_p^{(c)} + F_p^{(c)} + F_n^{(c)} + T_n^{(c)} = 1. \quad (22)$$

Accuracy (Acc) For segmentation evaluation, the segmentation accuracy

$$\text{Acc} = \frac{1}{C} \sum_{c=1}^C T_p^{(c)} + T_n^{(c)} \quad (23)$$

is the fraction of true predictions in the overall domain, averaged over all classes c .

A.3 Spherical harmonics

The spherical harmonic functions $Y_l^m(\vartheta, \varphi)$ define a set of orthogonal polynomials on the sphere. They are defined as

$$Y_l^m(\vartheta, \varphi) = (-1)^m c_l^m P_l^m(\cos \vartheta) e^{im\varphi} = \hat{P}_l^m(\cos \vartheta) e^{im\varphi}, \quad (24a)$$

$$c_l^m := \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}}, \quad (24b)$$

¹Note that (20) is sometimes referred to as micro-average, where the class averages are computed over numerator and denominator separately. This definition is more stable than it's macro-averaged counterpart, where the average is computed over the ratio instead. For this work, we use the micro-averaged expression.

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

$$\int_{S^2} Y_l^m(x) \overline{Y_{l'}^{m'}(x)} d\mu(x) = \delta_{ll'} \delta_{mm'}. \quad (25)$$

B Implementation details

B.1 Discrete realization

With an appropriate quadrature rule applied to the integrands of the continuous formulation, we get the discrete attention

$$\text{Attn}_{S^2}[q, k, v](x) = \sum_{j=1}^{N_{\text{grid}}} \frac{\exp\left(q^T(x) \cdot k(x_j)/\sqrt{d}\right)}{\sum_{l=1}^{N_{\text{grid}}} \exp\left(q^T(x) \cdot k(x_l)/\sqrt{d}\right) \omega_l} v(x_j) \omega_j, \quad (26)$$

followed by the discrete neighborhood attention

$$\text{Attn}_{S^2}[q, k, v](x) = \sum_{j=1}^{N_{\text{grid}}} \frac{\mathbb{1}_{D(x)}(x_j) \exp\left(q^T(x) \cdot k(x_j)/\sqrt{d}\right)}{\sum_{l=1}^{N_{\text{grid}}} \mathbb{1}_{D(x)}(x_l) \exp\left(q^T(x) \cdot k(x_l)/\sqrt{d}\right) \omega_l} v(x_j) \omega_j, \quad (27)$$

B.2 Global spherical attention implementation

Our global spherical attention mechanism leverages PyTorch's native scaled dot product attention (SDPA), where quadrature weights ω_i are incorporated via the framework's additive attention mask. SDPA applies additive attention masking, i.e. by adding the mask tensor to the $q \cdot k$ logits prior to computing the softmax. We can translate this into a multiplicative weighting of the exponential functions by encoding $\log(\omega_i)$ directly into the mask—a valid approach for strictly positive weights ($\omega_i > 0$).

B.3 Spherical neighborhood attention implementation

Our implementation for spherical neighborhood attention follows the ideas used to derived discrete continuous convolutions on the sphere (DISCO) by [34]. In this work, a spherical convolution can be described by a contraction of sparse matrix ψ , which encodes the geometry of the problem, with a dense input tensor x of shape $B \times C \times H \times W$:

$$y[b, k, c, h, w] = \sum_{h'=0}^{\text{nlat}-1} \sum_{w'=0}^{\text{nlon}-1} \psi[k, h, h', w'] x[b, c, h', w' + w] \quad (28)$$

Here, $0 \leq k < K$ and K is the dimension of a chosen set of basis functions which support as well as coefficient values are encoded in ψ . The final result of the convolution is obtained by contracting the channels c and basis indices k with a learnable weight tensor.

Note that ψ only depends on the input and output grids (via it's sparsity structure and quadrature weights for the input grid $\omega_{h'}$) as well as the number and type of basis functions. However, ψ does not depend on the number of channels c or any learnable parameters. Therefore, ψ can typically be re-used across many different layers in contemporary neural networks. Additionally, ψ does not depend on w because all grids considered in this work are translationally invariant in longitude direction. This again reduces the memory footprint of DISCO convolution layers.

In order to implement spherical neighborhood attention, we modify equation (28). In this specific case, we can choose $K = 1$ and do not need to store any basis coefficient values, as our attention kernel (5) computes the appropriate neighbor weights dynamically. In this case, ψ is basically just an indicator function $\mathbb{1}[h, h', w']$, which is equal to one for all $|x_i(h, 0) - x_j(h', w')|_{S^2} \leq \theta_{\text{cutoff}}$ and zero otherwise. We can write:

$$y[b, c, h, w] = \sum_{h'=0}^{\text{nlat}-1} \sum_{w'=0}^{\text{nlon}-1} v[b, c, h', w' + w] \times \frac{\mathbb{1}[h, h', w'] \exp\left(\sum_{l=0}^{d-1} q[b, l, h, w] \cdot k[b, l, h', w' + w] / \sqrt{d}\right) \omega[h']}{\sum_{h''=0}^{\text{nlat}-1} \sum_{w''=0}^{\text{nlon}-1} \mathbb{1}[h, h'', w''] \exp\left(\sum_{l=0}^{d-1} q[b, l, h, w] \cdot k[b, l, h'', w'' + w] / \sqrt{d}\right) \omega[h'']}. \quad (29)$$

As noted, the desire for a weighted spherical neighborhood of our attention model (6) comes with the downside that we cannot leverage pre-existing optimized attention implementations, which do not parameterize quadrature weights or the shape of the neighborhood.

B.4 Spherical neighborhood attention gradients

Beyond the CUDA implementation of forward attention A , we have to derive and implement the backward passes of the model. If we let $q_i := q(x_i)$, $k_j := k(x_j)$ and $\alpha_{ij} := \exp(q_i \cdot k_j) \omega_j$, starting with the Jacobian w.r.t. q_i , which requires use of the quotient rule, we define the terms

$$\begin{aligned} f &= \sum_j \alpha_{ij} v_j & f' &= \sum_j \alpha_{ij} k_j \\ g &= \sum_j \alpha_{ij} & g' &= \sum_j \alpha_{ij} k_j, \end{aligned}$$

which combine to form the the Jacobian

$$\begin{aligned} \frac{\partial A}{\partial q_i} dy_i &= (f'(dy_i \cdot v_j)g - f(dy_i \cdot v_j)g') g^{-2} \\ &= \left(\sum_j (\alpha_{ij} k_j (dy_i \cdot v_j)) \sum_j \alpha_{ij} - \sum_j (\alpha_{ij} (dy_i \cdot v_j)) \sum_j (\alpha_{ij} k_j) \right) \\ &\quad \left(\sum_j (\alpha_{ij}) \right)^{-2} \end{aligned} \quad (30)$$

Next we will show the Jacobian for k_j , which uses the softmax derivative identity

$$\sigma(p)_i = \frac{\exp(p_i)}{\sum_j \exp(p_j)}, \quad \frac{d\sigma(p)_i}{dp_j} = p_i \delta_{ij} - p_i p_j.$$

With this identity, we can derive the Jacobian for k_j

$$\frac{\partial A}{\partial k_j} dy_i = \frac{\sum_j q_i \alpha_{ij}}{\sum_j \alpha_{ij}} \left(dy_i \cdot v_j - \frac{\sum_j (\alpha_{ij} (dy_i \cdot v_j))}{\sum_j \alpha_{ij}} \right) \quad (31)$$

The Jacobian for v is simply

$$\frac{\partial A}{\partial v_j} dy_i = \frac{\sum_j \alpha_{ij} dy_i}{\sum_j \alpha_{ij}} \quad (32)$$

B.5 Spherical neighborhood attention CUDA implementation

With the discrete neighborhood attention formulation, we implement the necessary forward and backward routines in a custom CUDA extension to ensure reasonable performance for our S^2 neighborhood experiments. The implementation is thread parallel over the neighborhood points k_j , with output points (q_i) parallelized over thread blocks. Due to this choice of parallelism, the softmax is computed in two steps, computing the necessary $\max(q \cdot k)$ required to avoid overflow in the exponential first, and finalizing the softmax sum per query point in the second step. This pattern is repeated for the gradient kernel, which additionally fuses q, k, v gradients into a single kernel. This parallelization strategy diverges from traditional optimized attention kernels, and we leave further optimizations and restructuring of this kernel to future work.

C Experiments

C.1 Experimental setup

Standard convolutional networks and Vision Transformers are designed for planar data and struggle with spherical inputs due to distortions introduced by equirectangular projections. These distortions, particularly near the poles, break translation equivariance and lead to inconsistent feature extraction, limiting performance on tasks requiring spherical understanding. To address these limitations, we implement spherical generalizations of these architectures.

Baselines We compare against standard Euclidean models applied to equirectangular projections, including the Vision Transformer (ViT) [15] and SegFormer [38]. These models operate on regular 2D grids and perform attention or convolution uniformly.

ViT employs convolutional patch embedding with kernel and stride size 2×2 as well as global attention. To allow flexibility with input sizes not divisible by the patch stride, we also evaluate a modified Transformer that uses overlapping 3×3 patch embeddings and replaces the decoder's reshaping with bilinear interpolation. Both Transformers use 4 attention layers, an embedding dimension of 128, GELU activation, and either layer or instance normalization. The Transformer neighborhood variant uses localized attention with a 7×7 kernel. ViT uses learnable positional embeddings, while Transformer makes use of spectral positional embeddings. The standard Transformer has 531,968 parameters, while the neighborhood variant has 532,480 parameters.

SegFormer follows a hierarchical architecture with four stages. It uses embedding dimensions [16, 32, 64, 128], attention heads [1, 2, 4, 8], and layer depths [3, 4, 6, 3]. Convolutions use 4×4 kernels, and models include MLP ratios of 4.0, dropout rates of 0.5, and drop path rates of 0.1. Neighborhood versions of SegFormer apply local attention using a 7×7 attention kernel. The resulting standard SegFormer has 688,321 parameters, and its neighborhood variant has 689,265 parameters.

Spherical counterparts To evaluate the effectiveness of the spherical counterparts, we compare against spherical models that mirror their respective Euclidean baselines, with matching embedding dimensions and parameter counts. The Spherical Transformer follows the ViT structure but replaces patch embeddings and global attention with discrete-continuous convolutions and attention on the sphere. The localized variant restricts the attention window to neighborhoods with a geodesic cutoff radius of $7\pi/(\sqrt{\pi} n_{\text{lat}})$ radians, where n_{lat} is the latitudinal dimension, matching the effective receptive field of the planar 7×7 attention window at the equator. Both the global and localized spherical Transformer models have 531,968 parameters.

Spherical SegFormer models replicate the hierarchical encoder-decoder structure of the Euclidean SegFormer, with identical stage-wise dimensions, heads, and depths. They employ discrete-continuous convolutions for the encoding-decoding steps. Down-sampling and up-sampling use overlap-based patch merging and spherical convolutions to preserve structure without distortion. The spherical SegFormer has 606,113 parameters, consistent across both standard and neighborhood versions.

All spherical models employ GELU activation, instance normalization, and spectral positional embeddings. For the spherical discrete-continuous convolutions, we set the number of piecewise linear basis functions equivalent to the parameter count of their Euclidean counterparts kernels (e.g., 9 basis functions for a 3×3 kernel).

C.2 Datasets

Segmentation of spherical data To facilitate future usage of the Stanford 2D3DS Dataset [1], we report additional details regarding its usage. The semantic labels is provided as a json file in the GitHub repository, while the image data itself is hosted at https://cvg-data.inf.ethz.ch/2d3ds/no_xyz/. Each picture's segmentation information `chair_9_conferenceRoom_1_4` is labeled to account for both the type of object (e.g., *chair*, *ceiling*) and the room information (e.g., `conferenceRoom_1_4`), where we drop the additional room information leaving us with only 14 classes. We additionally note that the semantic labels were built such that one label 855309 overflowed to 3341 (due to uint8 handling of integers), which is ignored when using NumPy 1.x,

Table 5: Overview of spherical models and Euclidean counterparts.

Model	Encoder/Decoder type	Attention type	Parameter count
$\mathbb{R}^2$ Transformer	2D convolution	Global	531,968
$\mathbb{R}^2$ Transformer (local)	2D convolution	7×7 window	532,480
S^2 Transformer	S^2 convolution	Global	531,968
S^2 Transformer (local)	S^2 convolution	Geodesic window	531,968
$\mathbb{R}^2$ SegFormer	2D convolution	Hierarchical	688,321
$\mathbb{R}^2$ SegFormer (local)	2D convolution	7×7 window	689,265
S^2 SegFormer	S^2 convolution	Hierarchical	606,113
S^2 SegFormer (local)	S^2 convolution	Geodesic window	606,113

but as of NumPy 2.x, this overflow is caught, which we fixed in our implementation, but needed to account for the assumed overflow in the semantic labels json file.

Depth estimation on spherical data We adapt our approach to predict per-pixel depth values from single-view panoramic images. We leverage the Stanford 2D3DS dataset’s panoramic depth maps, which are stored as 16-bit PNGs with a maximum measurable distance of 128 m and a sensitivity of 1/512 m. Each depth value represents the Euclidean distance from the camera center to a point in the scene. To recover the metric depth from the RGB channels of the PNG images, we use the following formula:

$$d = \frac{R + 256G + 256^2B}{512} \quad (33)$$

where R , G , and B denote the 8-bit values of the red, green, and blue channels, respectively. This linear transformation preserves the original 16-bit precision while accommodating PNG storage constraints. To address invalid or distorted regions inherent in the equirectangular projection we apply a mask covering 15% of the latitude at both poles to eliminate areas with severe distortion. The remaining invalid pixels, indicated by the value $2^{16} - 1$, are filtered out. Additionally, the data is standardized to have zero-mean, unit-variance distributions across all batches.

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

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

The state $(\varphi, \varphi u^T)^T$ contains the geopotential layer depth φ (mass) and the tangential momentum vector φu (discharge). In curvilinear coordinates, the flux tensor T can be written using 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. [21, 5, 33].

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 $0.2 \cdot \sqrt{\varphi_{\text{avg}}}$. 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 $2 \cdot 10^{-3}$.

Table 6: Numerical results for the equivariance test. The test is performed on equiangular grids of various resolutions.

Resolution	global S^2 attention		local S^2 attention	
	Interpolation error	Equivariance error	Interpolation error	Equivariance error
13×24	0.321 ± 0.096	1.097 ± 0.658	0.360 ± 0.040	0.879 ± 0.119
25×48	0.190 ± 0.047	0.674 ± 0.372	0.212 ± 0.018	0.541 ± 0.063
49×96	0.105 ± 0.040	0.220 ± 0.118	0.106 ± 0.010	0.222 ± 0.030
97×192	0.052 ± 0.028	0.070 ± 0.038	0.045 ± 0.007	0.086 ± 0.014
193×384	0.018 ± 0.015	0.021 ± 0.015	0.014 ± 0.003	0.028 ± 0.005
385×768	-	-	0.003 ± 0.001	0.008 ± 0.002

D Equivariance test

To verify the approximate equivariance of the spherical attention formulation we study the equivariance error

$$\frac{\|\Psi_{R^{-1}} \text{Attn}_{S^2}[\Psi_R q, \Psi_R k, \Psi_R v](x) - \text{Attn}_{S^2}[q, k, v](x)\|_{L^2(S^2)}}{\|\text{Attn}_{S^2}[q, k, v](x)\|_{L^2(S^2)}}, \quad (35)$$

where Ψ_R is the representation corresponding to the rotation $R \in SO(3)$, passively rotating a function $f \in L^2(S^2)$, such that $\Psi_R f(x) = f(R^{-1}x)$. Equation (35) measures the approximate equivariance by comparing the result of applying the attention mechanism to a rotated signal and rotating it back to the result of directly applying it to the original signal. In the perfectly equivariant case, this error is exactly 0.

To perform this test numerically, we sample a Gaussian process on the sphere, which is defined by random, normally distributed spectral harmonic coefficients up to degrees $l = m = 12$. This process is repeated three times to generate random inputs $q(x)$, $k(x)$ and $v(x)$. The resulting functions are in turn rotated numerically by applying the rotation R to the grid on which the functions are represented and subsequently interpolating the result onto the original grid. We use cubic interpolation, while making sure that the periodicity at the boundary is respected. The spherical attention mechanism is applied and the result is rotated back using the same method. The outcome is then compared to the attention mechanism directly applied to the original signals. As this procedure incurs an interpolation error due to the numerical application of the rotation, we compare the results to the interpolation error $\|\Psi_{R^{-1}} \Psi_R a(x) - a(x)\|_{L^2(S^2)} / \|a(x)\|_{L^2(S^2)}$, where the same rotation and its inverse are numerically applied to the output of the attention mechanism $a(x) = \text{Attn}_{S^2}[q, k, v](x)$.

We perform the test for the rotation R defined by the Euler angles $\alpha = \pi/3, \beta = \pi/5, \gamma = \pi/2$, but find these results remain qualitatively the same for other choices of R . The results are averaged over 32 different random initializations of q , k and v and both mean and standard deviation are reported in Table 6. We observe that the equivariance error decays roughly at the same rate as the interpolation error incurred when rotating the signals, which indicates that the method is indeed approximately equivariant. Moreover, for the global S^2 attention, we find that the equivariance error is only slightly larger than the interpolation error and roughly twice as large as the interpolation error in the case of the local S^2 attention. While it is not possible to clearly separate the contribution of the interpolation error to the equivariance error, this indicates that the equivariance error is at worst twice that of the interpolation error.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction clearly articulate the main claims and contributions of the paper. These sections accurately summarize both the theoretical and experimental results, and appropriately outline the scope, assumptions, and limitations of our work. The claims presented are consistent with the findings reported in the main body of the paper and do not overstate the generalizability of the results.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The limitations of our work are explicitly discussed in the paper, in the context of presenting individual results. We address the main assumptions underlying our approach, discuss the robustness of our results to potential violations of these assumptions, and outline the scope and generalizability of our claims. Additionally, we comment on factors that may influence performance, and computational efficiency.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This work focuses on empirical and methodological contributions rather than theoretical analysis. While the paper includes algebraic derivations to explain its approach, it does not present formal theorems, lemmas, or proofs that would fall under the category of theoretical results. All claims are substantiated through experimental validation rather than mathematical proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides a detailed description of the experimental setup, including datasets, model architectures, hyperparameters, and evaluation protocols, ensuring that all key aspects necessary for reproducibility are covered. Additionally, the code used to conduct the experiments is publicly available on through an anonymized repo (and eventually GitHub), further facilitating independent verification and replication of the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

- (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
- (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code necessary to reproduce all main experimental results is openly available on an anonymous library (and eventually GitHub), accompanied by detailed installation and usage instructions. These instructions specify the required environment, dependencies, and exact commands to run the experiments. Additionally, we provide guidance on accessing and preparing the datasets used in our study.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides a thorough description of the experimental setup, including details on data splits, selected hyperparameters, optimization methods, and the procedures used for hyperparameter selection. These details are presented in the main text and supplemented with additional information in the appendix and the publicly available code repository.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All evaluation metrics and their corresponding error measurements are presented with appropriate explanations and referenced in the relevant tables and figures.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper specifies the hardware used for all experiments, including details on the type of compute resources, and execution times for key experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification: The research fully adheres to the NeurIPS Code of Ethics. The authors have carefully reviewed the guidelines and ensured compliance regarding data privacy, consent, copyright, fairness, and transparency. All datasets were used in accordance with their licenses. No aspect of the work was found to violate the ethical standards set by NeurIPS.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This paper presents a methodological improvement to attention mechanisms for machine learning tasks on spherical data. As the contribution is foundational and not tied to any specific application or deployment scenario, we do not foresee direct negative societal impacts arising from this work. The method is broadly applicable and does not introduce any new risks or harms beyond those already associated with general-purpose machine learning algorithms.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The methods and resources presented in this paper do not involve the release of data or models that carry a high risk of misuse. As such, no specific safeguards are necessary. The work focuses on foundational algorithmic improvements and does not introduce new risks beyond those already present in standard machine learning research.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: All external assets used in this paper, including the Stanford 2D3Ds dataset, are properly credited and cited in the relevant sections. We explicitly mention the dataset version, provide appropriate references, and acknowledge the original creators. The license and terms of use for all utilized assets are clearly stated and strictly respected throughout the work.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[NA\]](#)

Justification: This paper does not introduce new assets. The code used is an extension of the publicly available anonymous library, which is published with clear documentation and instructions for reproducing the experiments. All datasets used are publicly available and properly cited, with appropriate credit given to their original creators.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were not used as an important, original, or non-standard component in the core methodology of this research. Their use was limited to writing, editing, or formatting the manuscript.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.