Deep Learning on the Sphere

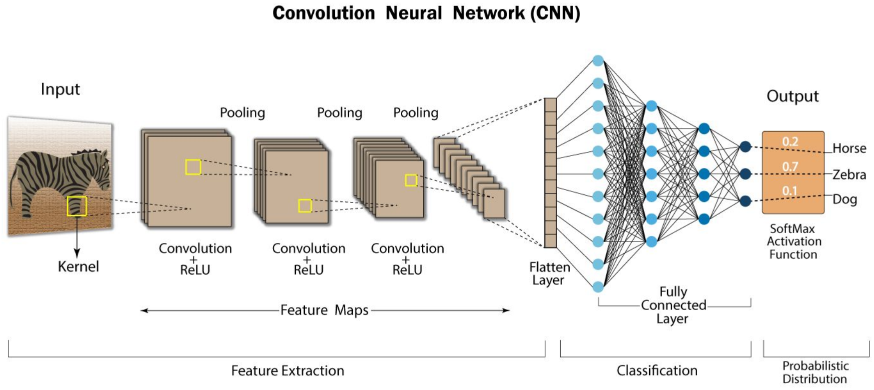
William Yik

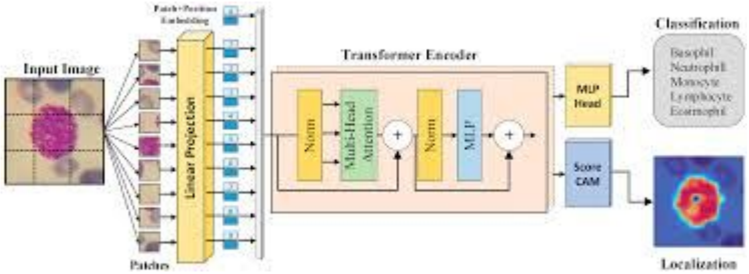
https://github.com/yikwill/mljc-spherical-ml-workshop

Why do we need special consideration for the sphere?

Deep learning for computer vision

Decades of research has optimized deep learning methods for image recognition and natural language processing

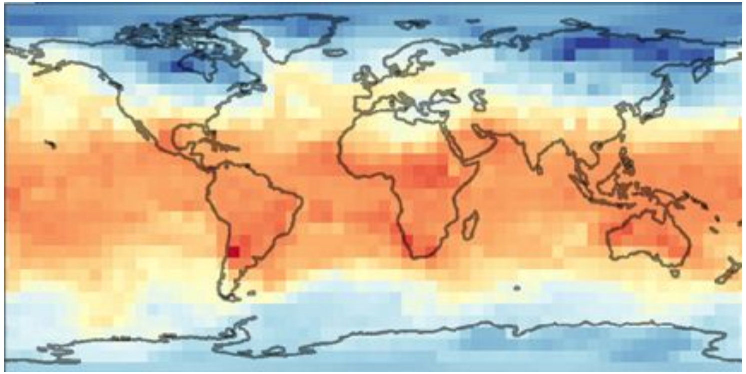




ERA5 as an image dataset?

Early attempts at global weather forecasting with ML treated global atmospheric data as images

Weather forecasting → next frame/token prediction

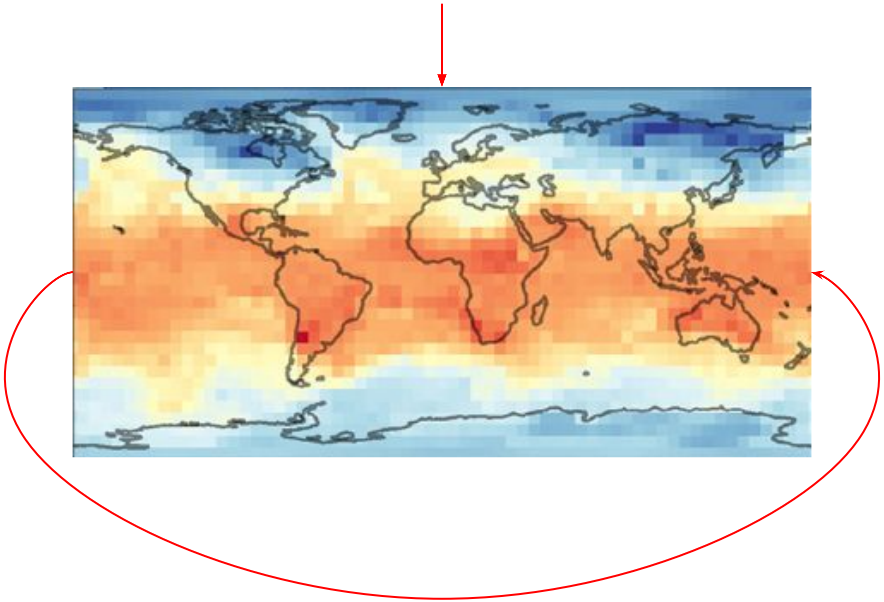


ERA5 as an image dataset?

Early attempts at global weather forecasting with ML treated global atmospheric data as images

Weather forecasting → next frame/token prediction

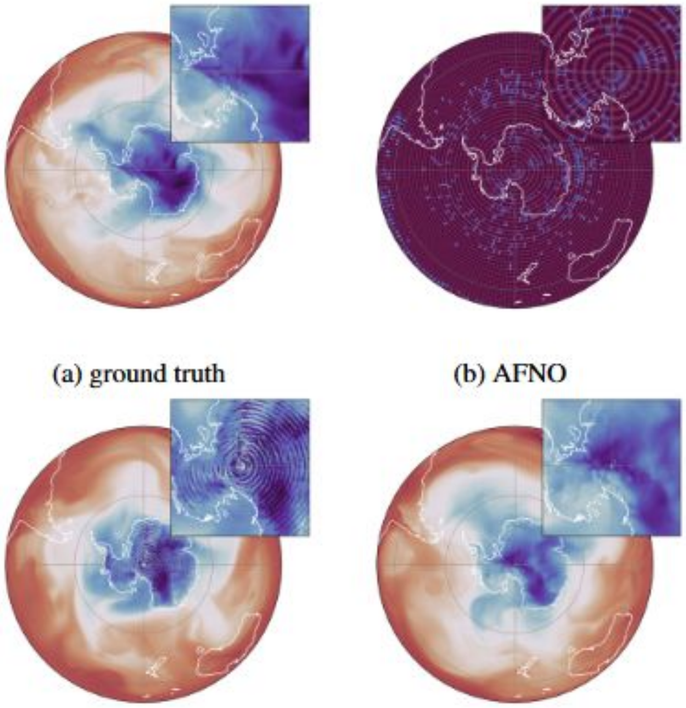
Obvious flaws with 2D representation



Spherical geometry matters!

Methods which don’t account for spherical geometry

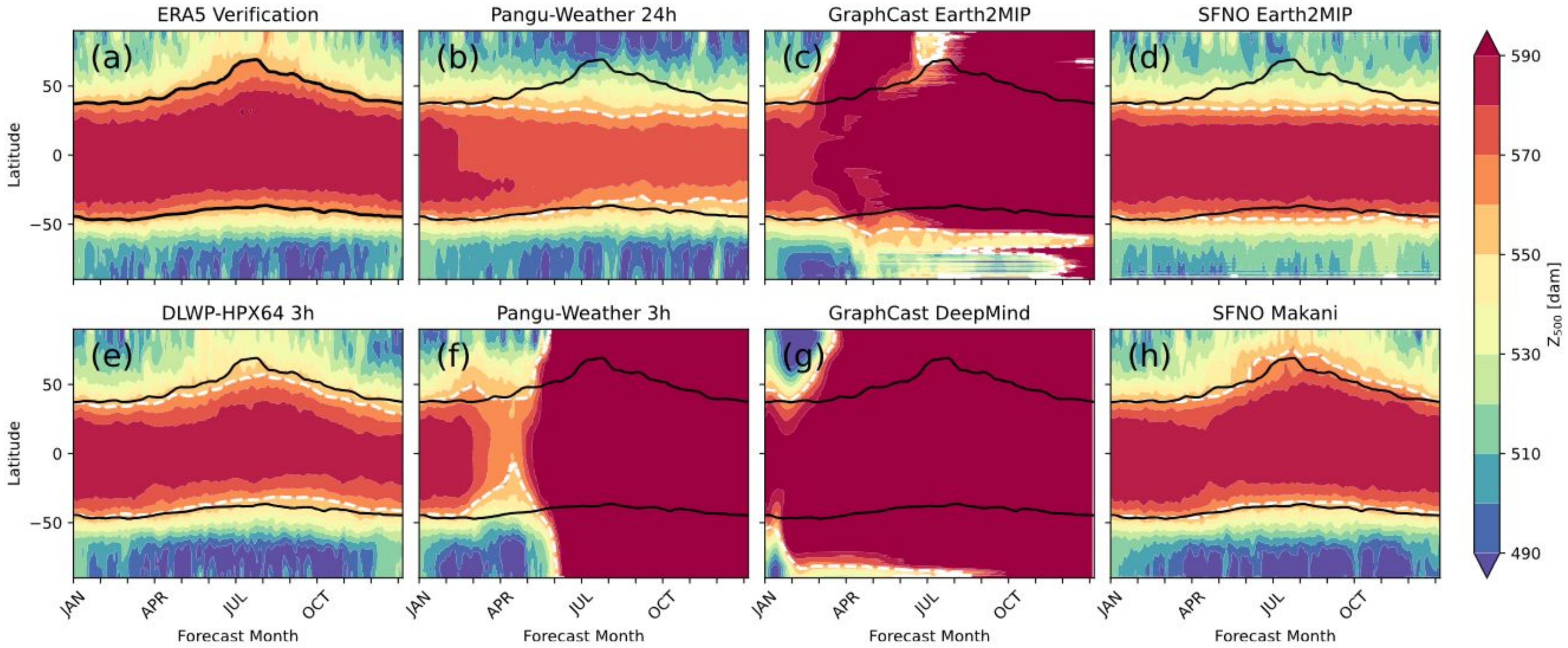
- Have distortions towards the poles
- Exhibit unrealistic behavior
- Are unstable in long rollouts



(a) ground truth (b) AFNO

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

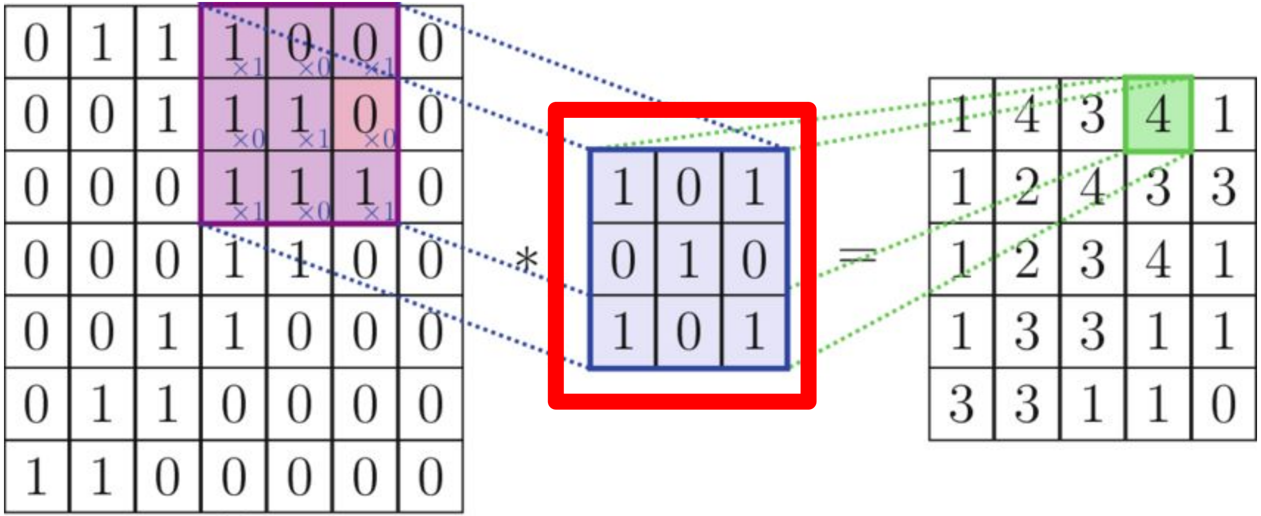
But geometry isn't the only thing that matters…



Deep learning methods for spherical data

0. Traditional Convolutional Neural Networks

Convolutional kernels



U

Data

K

Kernel

CNNs learn this

U * K

Convolution of U and K

1. Latitude/longitude padding

Padding in traditional convolutional neural networks

0	0	0	0	0	0	0	0
0	**3**	**3**	**4**	**4**	**7**	**0**	0
0	**9**	**7**	**6**	**5**	**8**	**2**	0
0	**6**	**5**	**5**	**6**	**9**	**2**	0
0	**7**	**1**	**3**	**2**	**7**	**8**	0
0	**0**	**3**	**7**	**1**	**8**	**3**	0
0	**4**	**0**	**4**	**3**	**2**	**2**	0
0	0	0	0	0	0	0	0

$6 \times 6 \;\rightarrow\; 8 \times 8$

*

1	0	-1
1	0	-1
1	0	-1

3×3

=

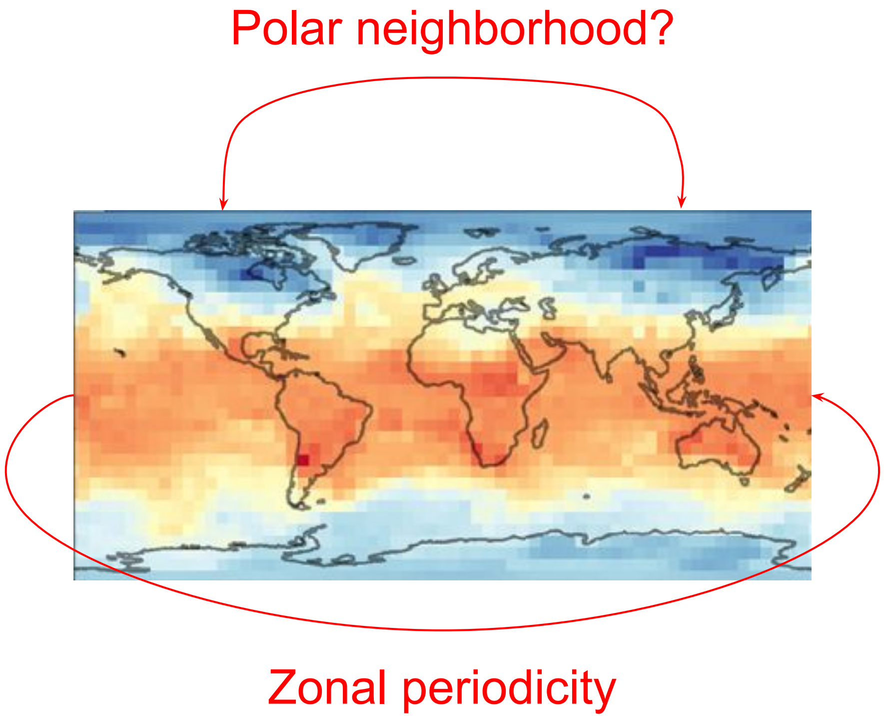
-10	-13	1			
-9	3	0			

6×6

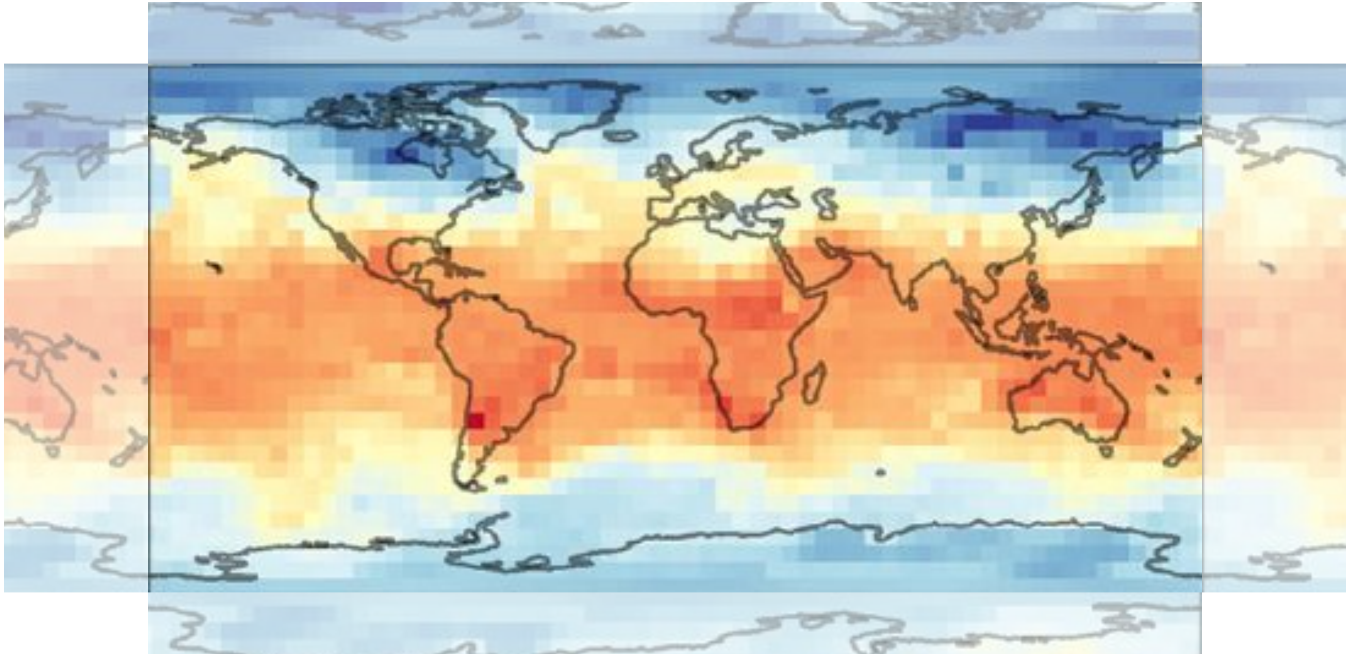
How to pad with spherical data?

Pad your “images” such that you have

- Periodicity in longitude
- Correct orientation of polar neighborhoods



Proposed lat/lon padding scheme



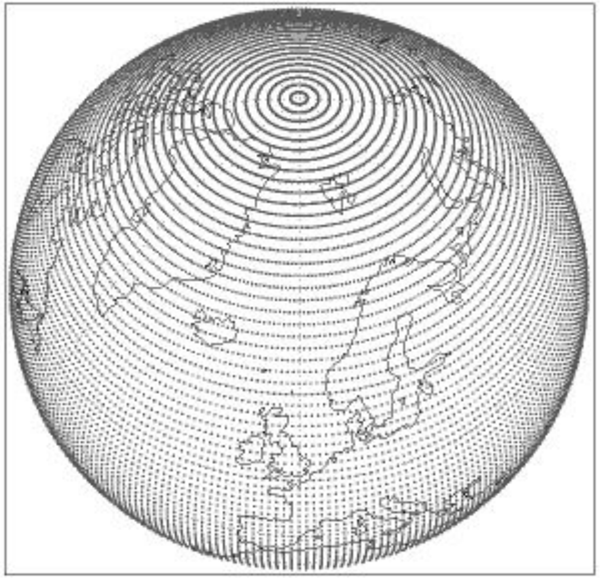
Schreck, J., Sha, Y., Chapman, W., Kimpara, D., Berner, J., McGinnis, S., ... & Gagne II, D. J. (2024). Community Research Earth Digital Intelligence Twin (CREDIT). arXiv preprint arXiv:2411.07814.

Let’s implement it!

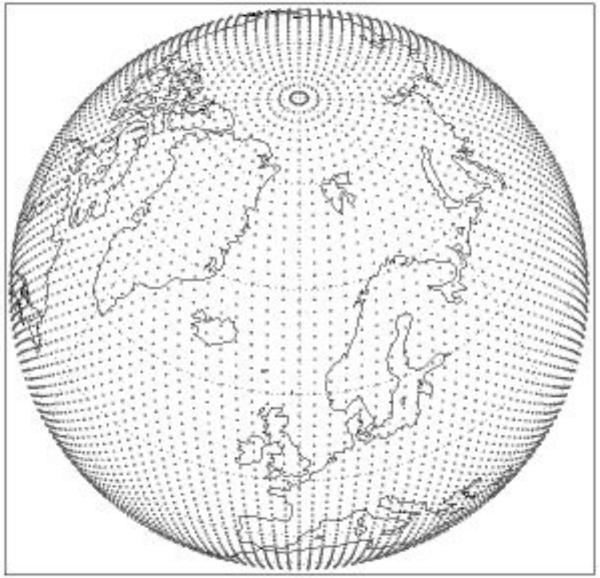
2. Grid discretization

ERA5’s grid

F80 Gaussian grid



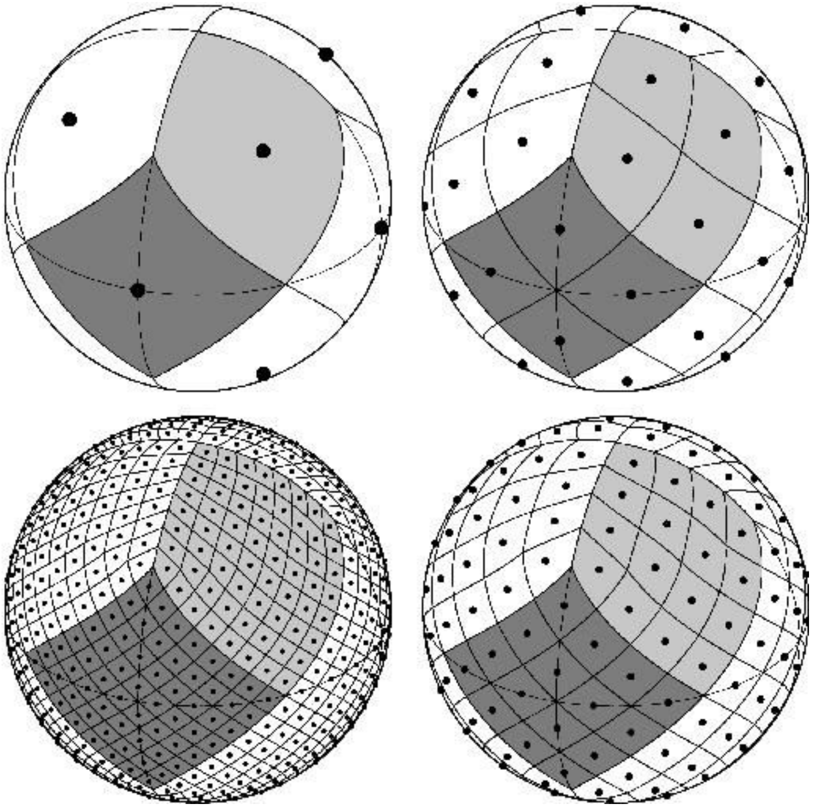
O80 octahedral reduced Gaussian grid



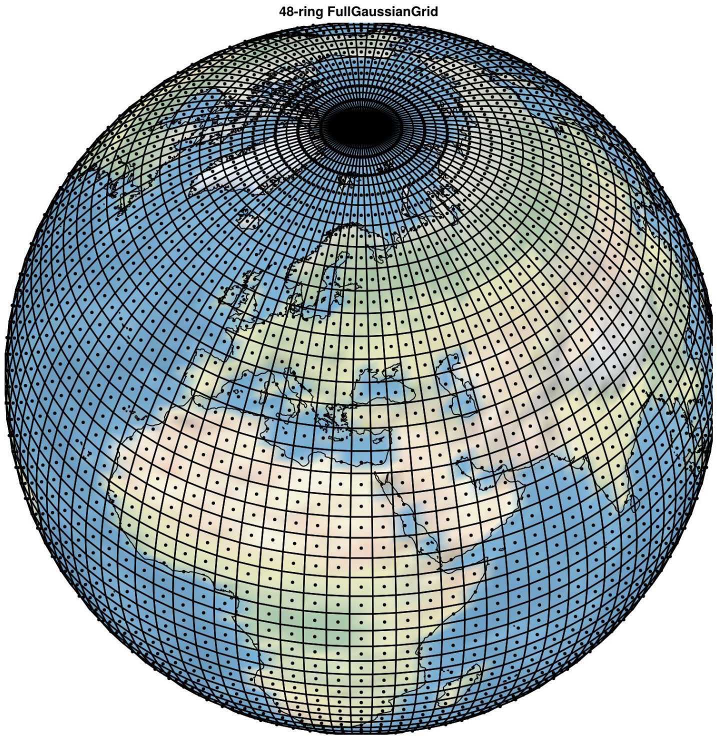
HEALPix grid

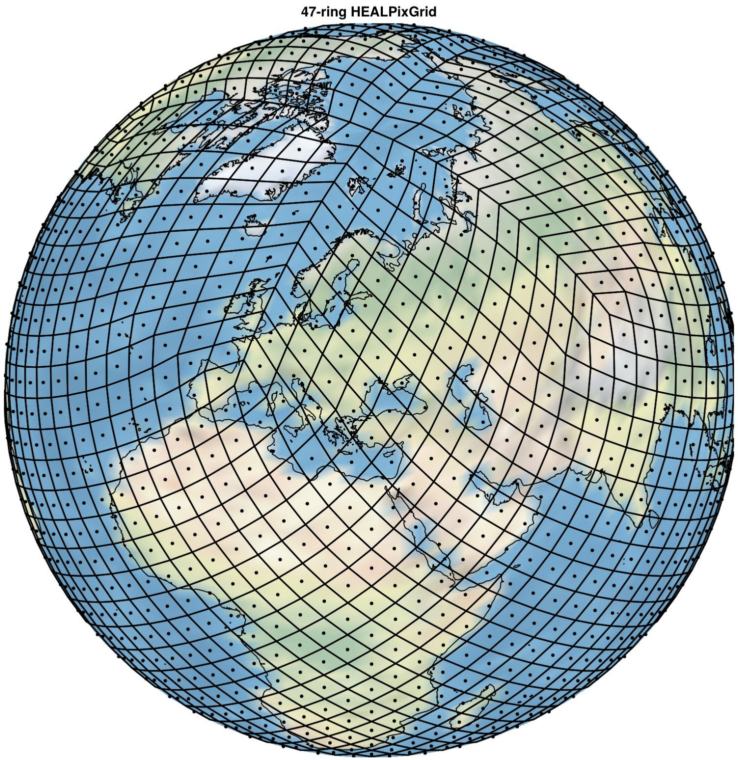
Hierarchical Equal Area isoLatitude Pixelation

- Subdivisions of 12 diamonds
- All grid cells have equal area
- Grid cells distributed on lines of constant latitude



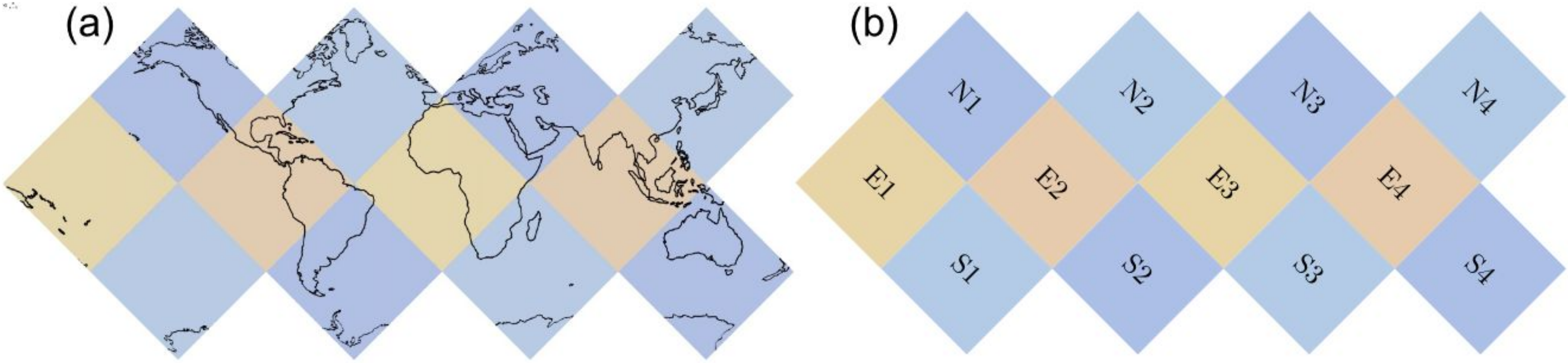
Equal area grid





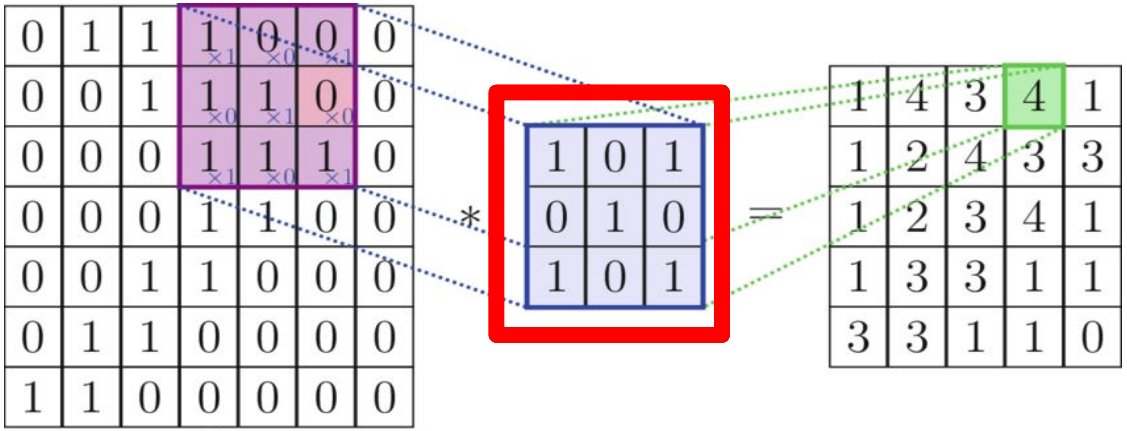
Using HEALPix for CNNs

Treat each of the 12 faces as a distinct image, pad using neighboring faces



Let’s implement it!

Local Spherical CNNs



u	k	u * k
Data	Kernel	Convolution of u and k

CNNs learn this

$$\mathcal{K}[u](x) = \int_{\mathcal{M}} \kappa(x-y)\cdot u(y)\,\mathrm{d}y$$

Convolution of u and k evaluated at x

Sum over all points **on the plane**

Product of kernel and data

Spherical Convolutions

$$\mathcal{K}[u](x) = \int_{\mathcal{M}} \underbrace{\kappa(x-y)\cdot u(y)}\,\mathrm{d}y$$

Convolution of u and k evaluated at x

Sum over all points **on the plane**

Product of kernel and data

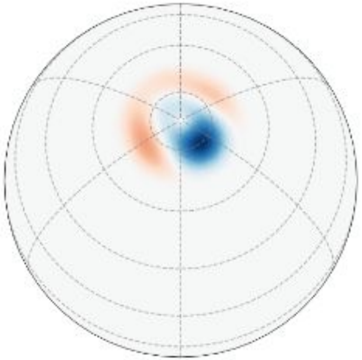
$$(\kappa \star u)(x) = \int_{R\in SO(3)} \underbrace{\kappa(Rn)\cdot u(R^{-1}x)}\mathrm{d}R$$

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

Product of kernel and data

Spherical Convolutions



(b) local convolution filter

$$(\kappa \star u)(x) = \int_{R \in SO(3)} \underbrace{\kappa(Rn) \cdot u(R^{-1}x)}_{} \mathrm{d}R$$

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

Product of kernel and data

Accounting for spherical geometry with quadrature weights

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

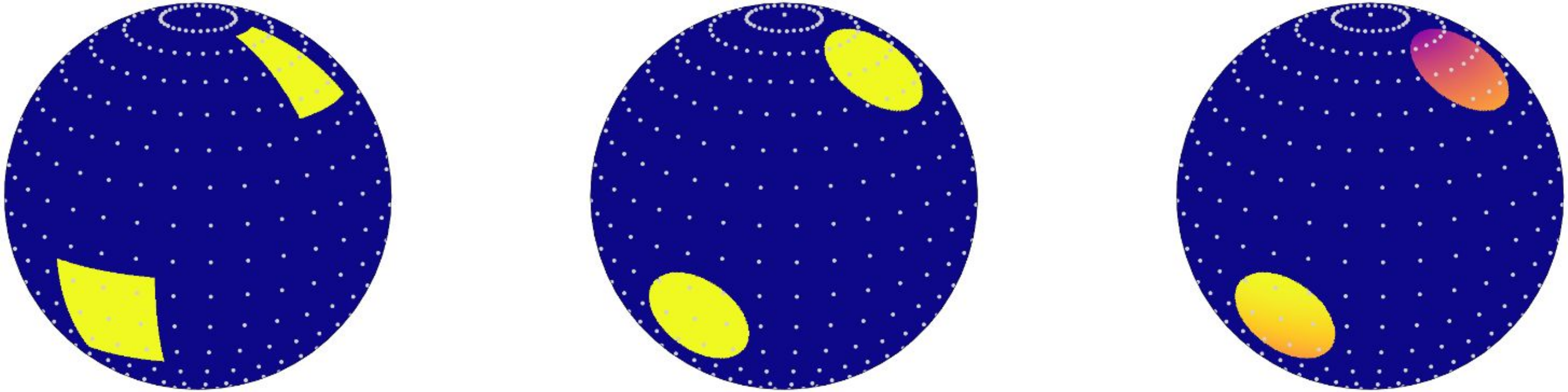
Product of kernel and data

$$(u \circledast k)(x) = \int_{S^2} u(x)\,\overline{k(R_x^{-1}x')}\,\mathrm{d}\mu(x')$$

$$\approx \sum_{j=1}^{n_{\mathrm{lat}} n_{\mathrm{lon}}} \overline{k(R_x^{-1}x_j)}\,u(x_j)\,\omega_j,$$

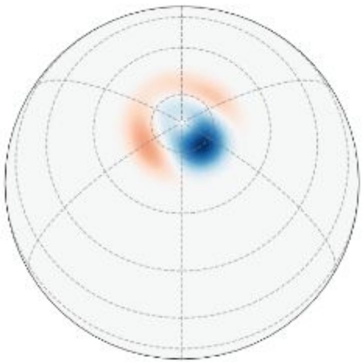
Quadrature weights

Accounting for spherical geometry with quadrature weights

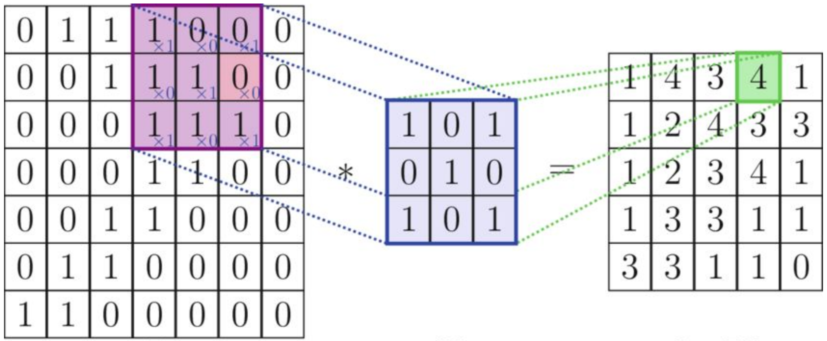


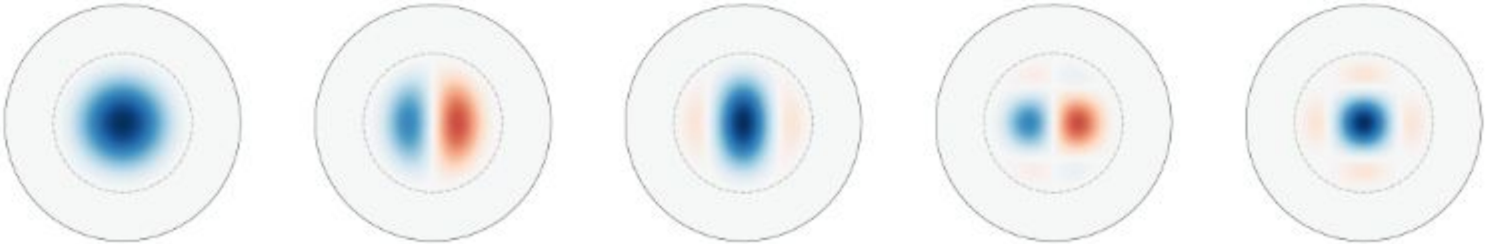
(a) Euclidean (b) naive spherical (c) weighted spherical

Defining our spherical kernel



(b) local convolution filter





(a) $\ell = 0, m = 0$ (b) $\ell = 0, m = 1$ (c) $\ell = 0, m = 2$ (d) $\ell = 2, m = 1$ (e) $\ell = 2, m = 2$

Let’s visualize it!

Global Spherical CNNs

Spherical Convolutions

$$\mathcal{K}[u](x) = \int_{\mathcal{M}} \underbrace{\kappa(x - y) \cdot u(y)}\, \mathrm{d}y$$

Convolution of u and k evaluated at x

Sum over all points **on the plane**

Product of kernel and data

$$(\kappa \star u)(x) = \int_{R \in SO(3)} \underbrace{\kappa(Rn) \cdot u(R^{-1}x)}\mathrm{d}R$$

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

Product of kernel and data

The Convolution Theorem

$$(\kappa \star u)(x) = \int_{R \in SO(3)} \underbrace{\kappa(Rn) \cdot u(R^{-1}x)}\mathrm{d}R$$

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

Product of kernel and data

Equivalent

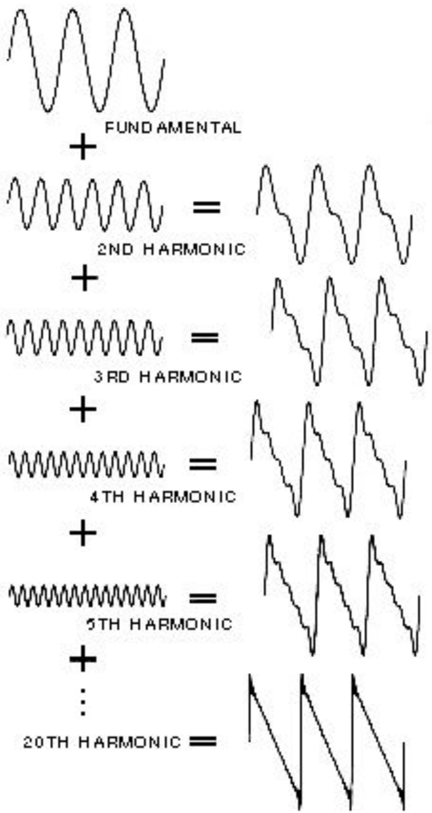
SHT = spherical harmonic transform

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

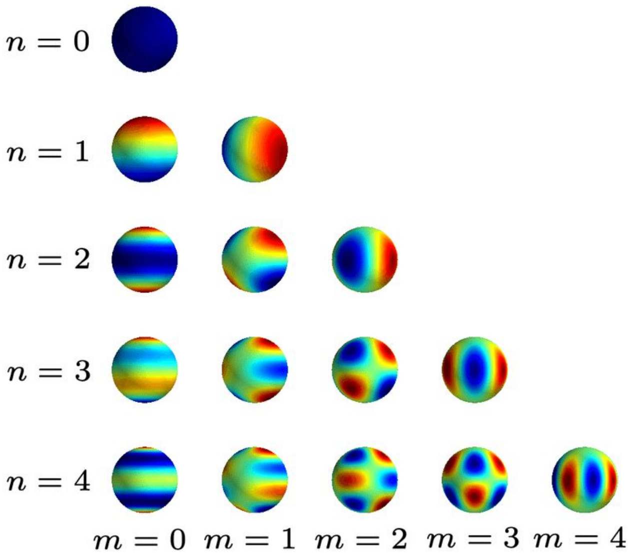
SHT of the convolution of u and k

Product of SHT(data) and SHT(kernel)

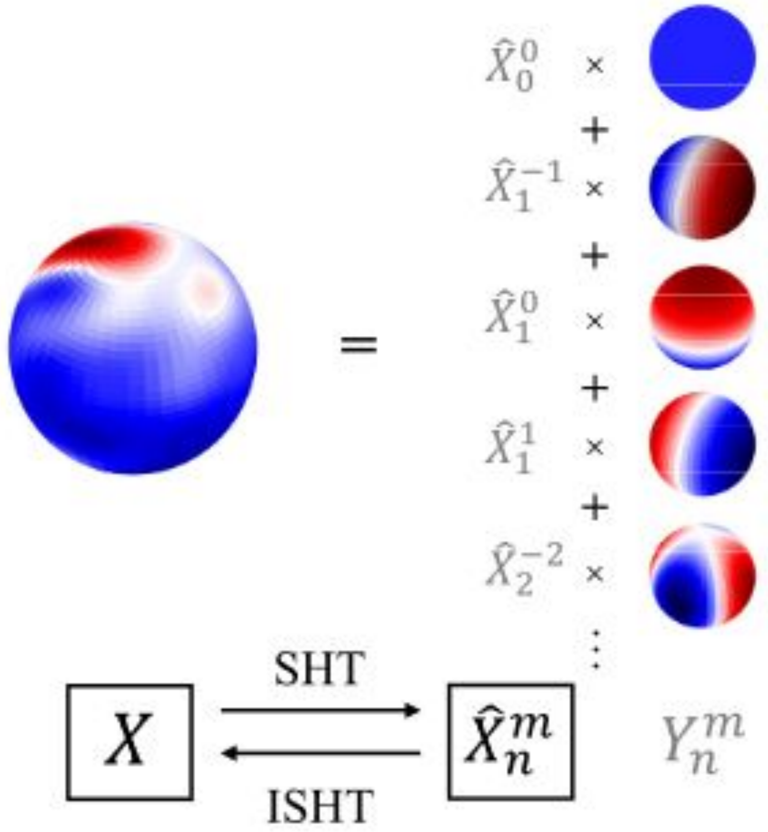
Fourier harmonics



Spherical harmonics



SHT transforms data into a linear combination of spherical harmonics



The Convolution Theorem

$$(\kappa \star u)(x) = \int_{R \in SO(3)} \underbrace{\boxed{\kappa(Rn)} \cdot u(R^{-1}x)}\mathrm{d}R$$

Convolution of u and k evaluated at x

Sum over all points **on the sphere**

Product of kernel and data

Equivalent

SHT = spherical harmonic transform

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

SHT of the convolution of u and k

Product of SHT(data) and SHT(kernel)

The Convolution Theorem

$$(\kappa \star u)(x) = \int_{R \in SO(3)} \underbrace{\boxed{\kappa(Rn)} \cdot u(R^{-1}x)}\,\mathrm{d}R$$

Conv

and k

at x

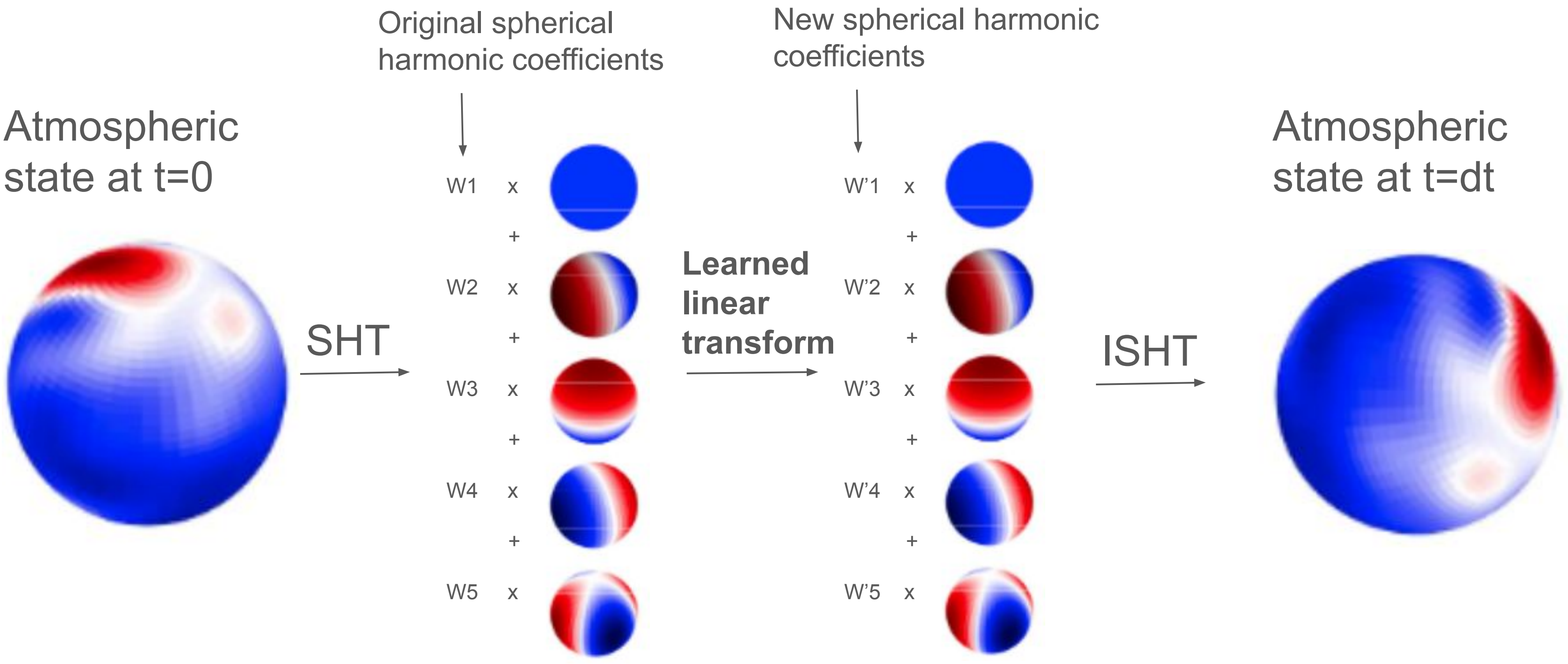
c transform

CNN learns a kernel in the spatial domain

SFNO learns a kernel in the spherical harmonics domain

SHT of the convolution of u and k

Product of SHT(data) and SHT(kernel)


Atmospheric state at t=0
SHT
Original spherical harmonic coefficients
W1 x
+
W2 x
+
W3 x
+
W4 x
+
W5 x
Learned linear transform
New spherical harmonic coefficients
W'1 x
+
W'2 x
+
W'3 x
+
W'4 x
+
W'5 x
ISHT
Atmospheric state at t=dt