Using the FEM Laplacian in Deepsphere

Goal

- Ability to use Convolutional Neural Networks on spherical data (weather/space maps…)
- Have it be rotation-equivariant by design
- Have it be efficient !

What has been done

- Deepsphere uses a graph to approximate the sphere,
- It uses a convolutional NN,
- It uses the Graph Laplacian to approximate the sphere’s Laplace-Beltrami operator.

Explanation next slide !

Without approximations

- The sphere is a manifold, with a defined “Laplace-Beltrami operator” : Δ.
- We want to do convolutions.
  These are easy in Fourier space.
- Fourier transform = a decomposition relative to Δ’s eigenfunctions.

Graph approximation

- Let’s approximate the sphere with a graph,
- Sample nodes on the sphere with HEALPix,
- Define weighted edges somehow* → Graph!
- We can define a Laplace Operator for it!

* For the HKGL : $w_{ij} = \exp(-\frac{||v_i - v_j||^2}{4t})$

Graph approximation

- The (discrete) Graph Laplacian L :

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

- Matrix representation of Δ,
- Graph sparsely connected $\Rightarrow$ L sparse,
- Graph undirected $\Rightarrow$ L symmetric.

Graph convolutions

- L is eigendecomposable : $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$
- Define Fourier Transform : $\hat{\mathbf{x}} = \mathbf{U}^{\mathrm{T}}\mathbf{x}$
- Define “a filter” : $h : \lambda \mapsto h(\lambda)$
- Define convolution :

$$\mathbf{y} = \mathcal{F}_G^{-1}(\mathbf{K}\hat{\mathbf{f}}) = \mathbf{U}\mathbf{K}\hat{\mathbf{f}} = \mathbf{U}\mathbf{K}\mathbf{U}^{\mathrm{T}}\mathbf{f} = \mathbf{U}h(\boldsymbol{\Lambda})\mathbf{U}^{\mathrm{T}}\mathbf{f}$$

(Finally, use polynomial filters for efficiency)

Graph Laplacian's problems

- How to get convergence of L's eigenvectors to the real Δ's eigenfunctions?
- HKGL converges if the graph…
  - fully connected (⇒not sparse)
  - nodes equi-area-sampled* (⇒not easy) & filter width *t* well-tuned.

* It still ~works if the sampling is regular enough

FEM approximation

- Approximate sphere with a triangulation T
- Discretize signals on that
- Turns out that FEM filtering ≅ Graph filtering
- FEM Laplacian still converges with more irregular samplings !

FEM explanation (see board)

- We want Δ’s eigenfunctions, find f :

$$\Delta f = -\lambda f$$

- Insert test function ν + integrate by parts :

$$\int_{\mathbb{S}_2} \nabla f(x) \cdot \nabla v(x) \mathrm{d}x = \lambda \int_{\mathbb{S}_2} f(x) \cdot v(x) \mathrm{d}x$$

(find f such that this holds for all ν in our function space)

FEM explanation (see board)

- ν ∈ *continuous piecewise-linear* funcs on T
- T has a basis :

$$\phi_i(x_j) = \delta_{ij} \quad \forall x_j \in \mathcal{T} \quad \forall i \in [0, n-1]$$

- Just prove for all bases : $\mathbf{A}\mathbf{f} = \lambda \mathbf{B}\mathbf{f}$

  with $$\begin{cases} (\mathbf{A})_{ij} & = & \int_{\mathbb{S}_2} \nabla\phi_i(\mathbf{x}) \cdot \nabla\phi_j(\mathbf{x})\mathrm{d}\mathbf{x} \\ (\mathbf{B})_{ij} & = & \int_{\mathbb{S}_2} \phi_i(\mathbf{x})\phi_j(\mathbf{x})\mathrm{d}\mathbf{x} \\ (\mathbf{f})_i & = & f_i : f(\mathbf{x}) = \sum_{k=0}^{n-1} f_k\phi_k(\mathbf{x}) \end{cases}$$

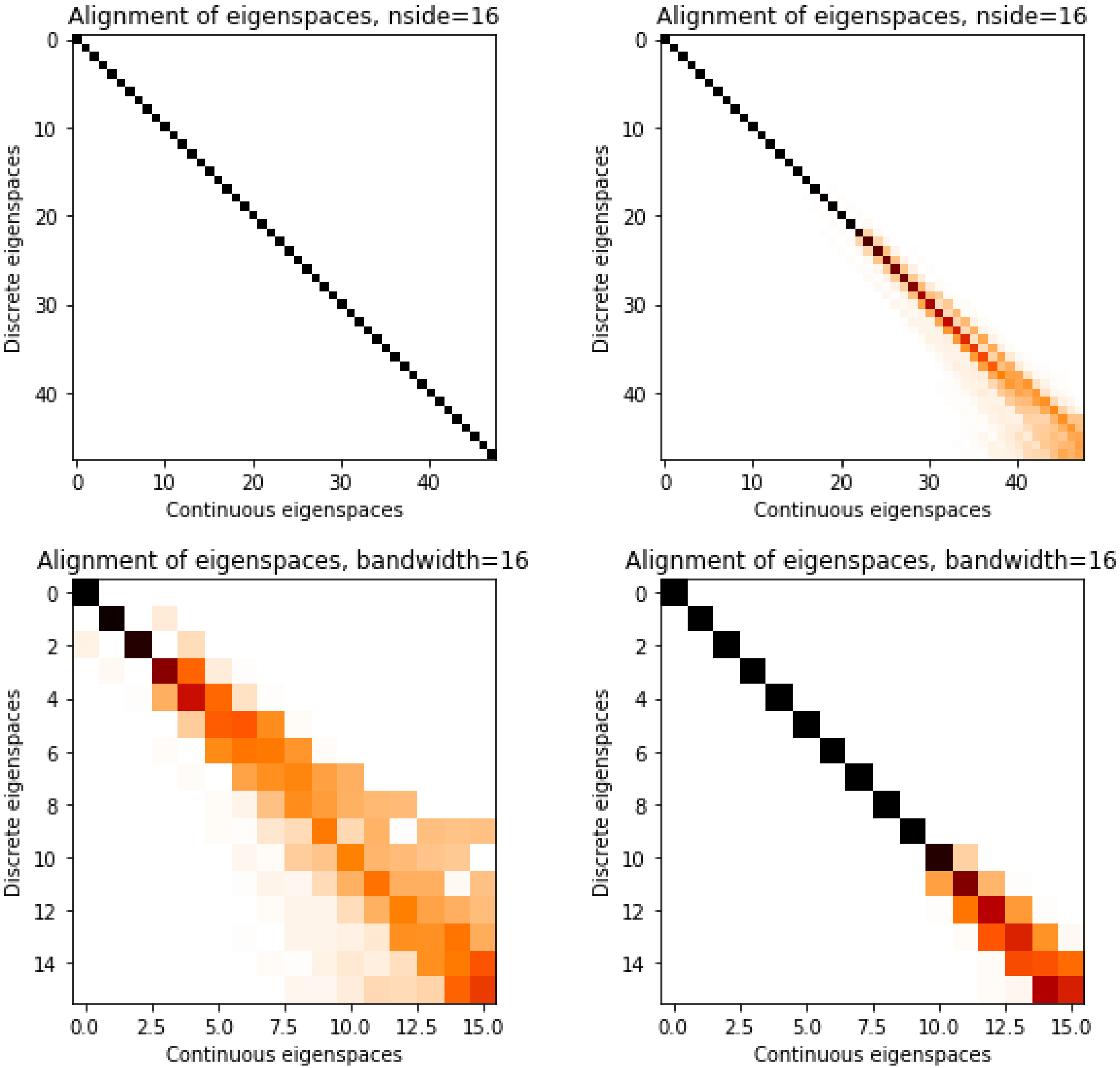
FEM explanation (see board)

- Define FEM Laplacian : $\mathbf{L} = \mathbf{B}^{-1}\mathbf{A}$
- A and B as full as triangles share vertices
- A and B symmetric
- L not sparse, nor symmetric!
  $\Rightarrow$ We will use tricks

HKGL vs FEM L

HEALPix

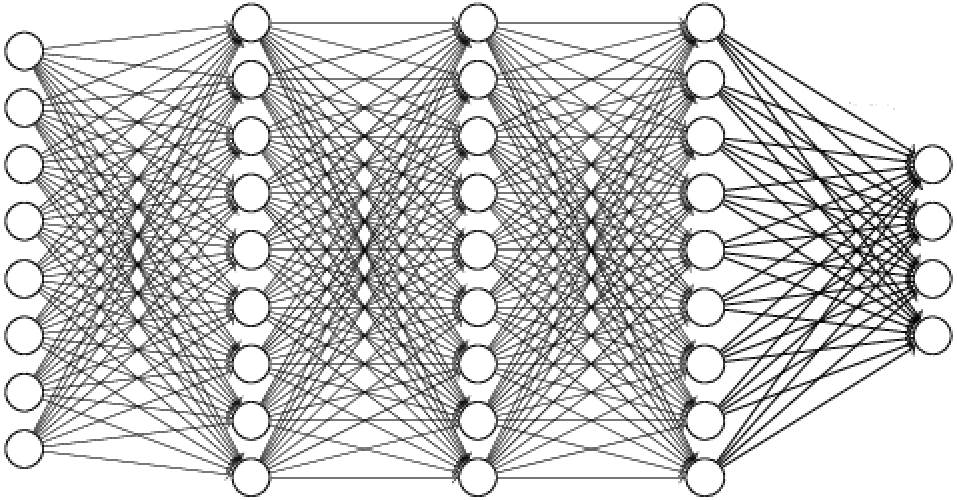
Equiangular



HKGL

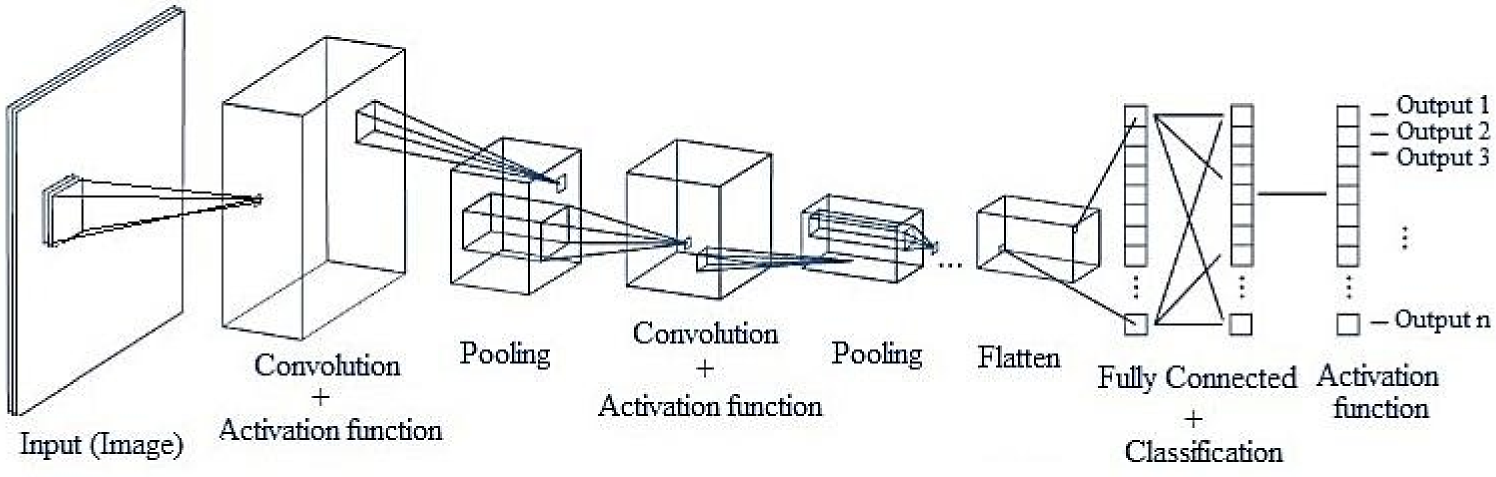
FEM L

Neural network



- Computes any function, given enough nodes
- Can be “trained” by descending gradients
- This one is fully connected ⇒ heavyweight

Convolutional neural network



- Translation-equivariant feature extraction
- Less connected + shared parameters ⇒ lightweight

Deepsphere's conv layers

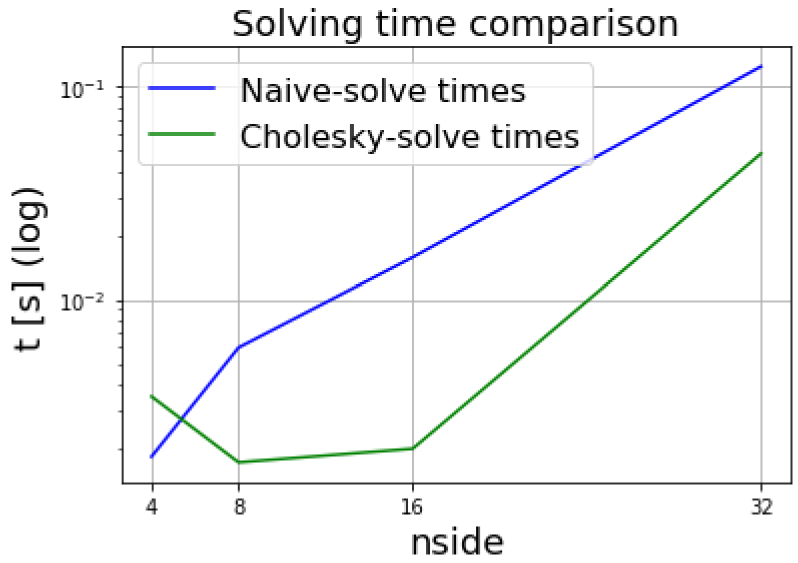
- Use polynomial filters : $h(\lambda) = \sum_{k=0}^{K-1} \theta_k \lambda^k$
- Simpler convolution : $\mathbf{y} = \sum_{k=0}^{K-1} \theta_k \mathbf{L}^k \mathbf{f}$
- Compute powers of L iteratively :

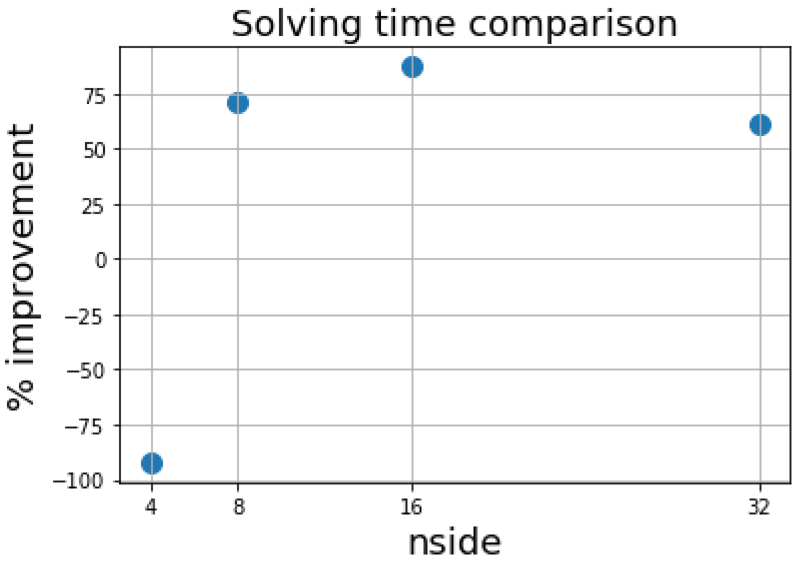
$$\begin{cases} \mathbf{L}^0\mathbf{f} & = & \mathbf{f} \\ \mathbf{L}^{k+1}\mathbf{f} & = & \mathbf{L} \cdot \mathbf{L}^k\mathbf{f} \end{cases}$$

- Convolution : a full, but small K×K matmul

Our FEM modifications

- B can be Cholesky-decomposed : $\mathbf{B} = \mathbf{C}\mathbf{C}^{\mathrm{T}}$
- C triangular + sparse like B ⇒ faster solving!





Our FEM modifications

- The FEM Laplacian isn’t sparse nor symmetric
- Powers of L must be computed another way :

$$\begin{cases} \mathbf{L}^0\mathbf{f} & = & \mathbf{f} \\ \mathbf{L}^{k+1}\mathbf{f} & = & \mathbf{L} \cdot \mathbf{L}^k\mathbf{f} \end{cases} \Rightarrow \quad \mathbf{B} \cdot (\mathbf{L}^{k+1}\mathbf{f}) = (\mathbf{A}\mathbf{L}^k\mathbf{f})$$

- Solving can be optimized because B sparse!
- But B is symmetric and positive-definite too!

Our FEM modifications

- Sadly, *tf* has no “*sparse_cholesky_solve*“
- This forces C to be full in every conv layer…
  $\Rightarrow$ *tf* limits us to 2GB/tensor $\Rightarrow$ nside$\leq$32
- That was for powers of L. Convolution is still just a K$\times$K matmul.

Recapitulation

- Defined a more robust FEM Laplacian
- Computed A, B, C for it
- Added them in Deepsphere's models
- Made it work, except for some loss problems
- Learned more about *differential geometry, discrete signal processing, FEM, deep learning*.