



## LECTURE 6

# GEOSPATIAL EMBEDDINGS

Explicit and Implicit Representations of Geographic  
Space

PRESS → OR SPACE. →

© Marc Rußwurm

Licensed under [Creative Commons Attribution-NonCommercial 4.0 International \(CC BY-NC 4.0\)](#).

You may share and adapt these slides for teaching, research, and other non-commercial educational purposes with attribution.

Commercial training, paid workshops, consulting seminars, or incorporation into commercial course products require prior permission.

# Learning Outcomes

- 1 Explain how **self-supervised learning** can produce reusable geospatial representations without task-specific labels.
- 2 Explain how **geospatial foundation models** produce explicit embeddings from observations such as images or time series.
- 3 Explain how **geographic location itself** can be mapped to an implicit embedding through coordinate-based neural models and location encoders.

# Where We Are in the Course

LECTURES 2–3

## Geospatial Data

Images, maps, time series

LECTURE 3

## Classical ML

ML on representations

LECTURE 4

## Deep Learning

Models & optimization

LECTURE 5

## Generalization

Beyond the training set

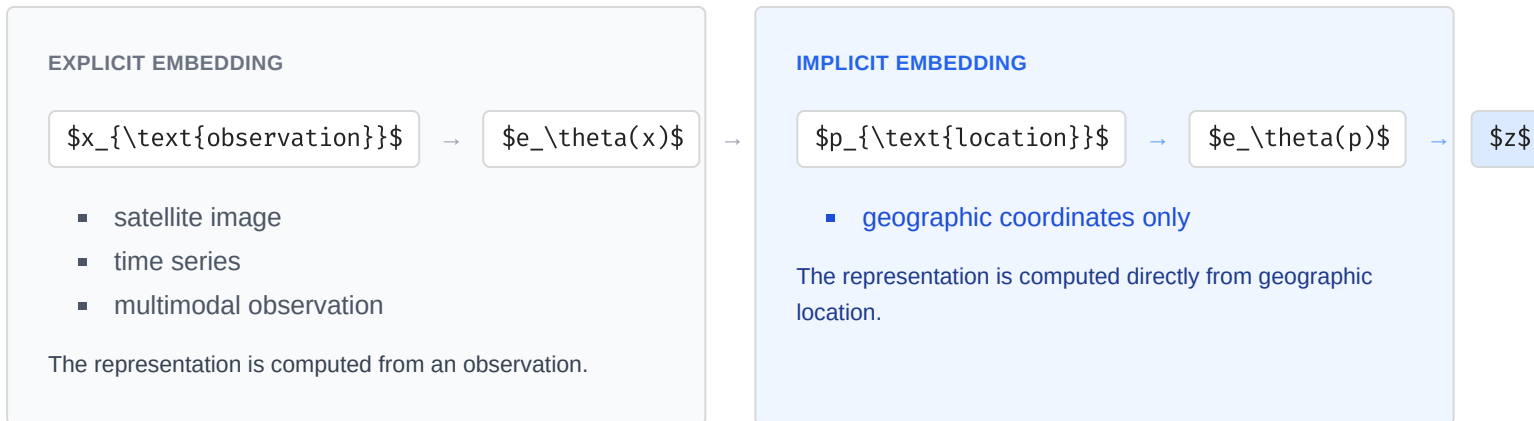
LECTURE 6 — TODAY

## Reusable Embeddings

Explicit & implicit

Main question: How can we learn representations that are useful beyond one specific supervised task?

# Two Ways to Obtain a Geospatial Embedding



This distinction structures the entire lecture.

# Explicit Geospatial Embeddings

# The Limitation of Supervised Learning

Supervised learning requires labeled data — but labels are:

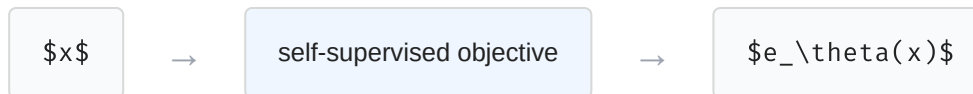
- **Expensive** — manual annotation is slow and costly
- **Task-specific** — a land-cover label is useless for species prediction
- **Specializing** — supervised representations may only encode what was asked

Can we learn useful representations from the data itself?

[ supervised pipeline vs. unlabeled data ]

# Self-Supervised Learning

Create a learning signal from the **structure of the data** rather than from manually annotated labels.



The objective is constructed entirely from data — no human annotation needed. The network must learn meaningful internal structure to solve it.

# Pretext Tasks

A pretext task is designed so that solving it forces the model to learn useful internal representations.

## PREDICT MISSING CONTENT

Mask part of the input, predict the masked part

## PREDICT TRANSFORMATIONS

Predict rotation, colorization, jigsaw order

## PREDICT CONTEXT

Predict spatial or temporal neighbours

## DISTINGUISH PAIRS

Distinguish related from unrelated samples

The pretext task is discarded after training — only the learned encoder is kept.

# Contrastive Learning

Pull **positive pairs** together in embedding space; push **negative pairs** apart.

**Positive pair:** two views of the same instance

**Negative pair:** views from different instances

Similar observations → nearby embeddings; different observations  
→ separated embeddings.

[ contrastive embedding space illustration ]

# Learning Invariances

The augmentations and positive-pair definition determine what the embedding should **ignore** and what it should **preserve**.

## GENERIC AUGMENTATIONS

- crop
- color / spectral changes
- rotation / flip

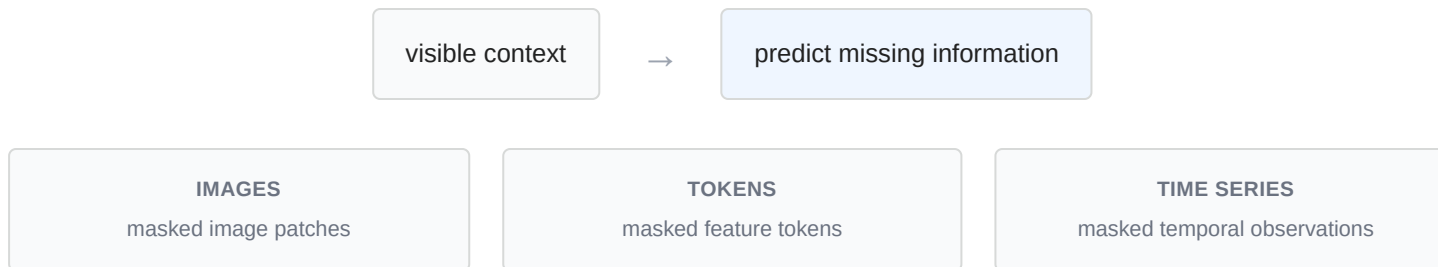
## GEOSPATIAL-SPECIFIC

- temporal changes
- sensor differences
- seasonal variation
- viewing angle

Choosing the right positive-pair definition is a core design decision in SSL for Earth observation.

# Masked Prediction

A second major self-supervised paradigm: mask part of the input, predict the missing content.



# Self-Supervision for Earth Observation

Earth observation provides natural structure for self-supervised objectives:

- Multispectral observations across bands
- Spatial context — neighbouring patches
- Temporal observations — same place, different times
- Multiple sensors observing the same area
- Repeated coverage of the same location

[ same location, different observation → positive pair ]

Main question: What should count as two views of the same place?

# From Encoder to Embedding

After pretraining, the encoder maps any new observation to a reusable feature vector.



**$e_{\theta}$** : the encoder, fixed after pretraining

**$z$** : the embedding — a compact learned representation

# What Makes an Embedding Useful?

## SEMANTIC SIMILARITY

Similar scenes → similar \$z\$

## INVARIANCE

Robust to irrelevant variation

## TRANSFERABILITY

Useful across downstream tasks

## COMPACTNESS

High information density

## LINEAR SEPARABILITY

Simple probes work

## MULTI-TASK UTILITY

Useful for many downstream tasks

# Exploring an Embedding Space

Once computed, embedding spaces can be visualized, probed, and queried.

Nearby observations share visual or semantic properties

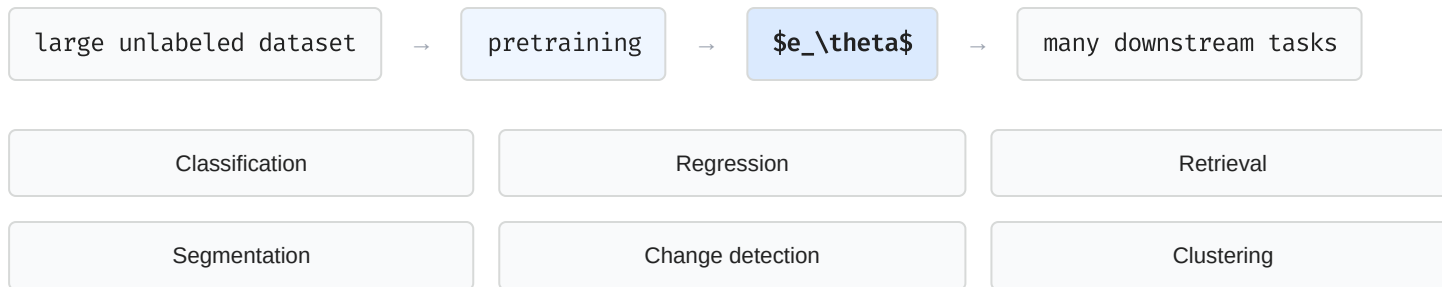
Clusters correspond to semantic categories

Trajectories trace temporal change

Nearest-neighbour retrieval finds similar observations

[ PCA / UMAP of geospatial embeddings ]

# From Pretraining to Transfer



# Scaling Up Representation Learning

## SMALL ENCODER

Pretrained on a limited dataset; transferable within domain

↓ more data, larger model, broader objective

## LARGE ENCODER

Pretrained on many scenes and sensors; transfers broadly

↓ more data, larger model, broader objective

## FOUNDATION MODEL

Large pretrained model intended as a general-purpose geospatial encoder

# Geospatial Foundation Models

Large pretrained models intended to provide reusable representations across datasets, regions, sensors, or tasks.

Pretrained on large-scale satellite or aerial data

Encoder output used directly as a feature

Fine-tuned on small labeled datasets for downstream tasks

[ selected geospatial foundation model examples ]

# What Does a Foundation Model Embedding Encode?

The embedding  $z$  may contain much more information than the downstream task explicitly requests.

Land cover

Scene structure

Season

Climate zone

Geographic context

Sensor characteristics

The representation may carry much more than the downstream task needs — or may omit what it needs.

# Explicit Geospatial Embeddings

$x_{\{\text{observation}\}}$

→

$e_{\theta(x)}$

→

$z$

**Image embedding:** one satellite patch → one vector

**Patch embedding:** sub-image patch → local descriptor

**Temporal embedding:** time series → sequence representation

**Multimodal embedding:** multiple sensors → joint representation

The observation must be available when the embedding is computed.

# Using Explicit Embeddings

## CLASSIFICATION

assign a category to each embedding

## CLUSTERING

group similar scenes without labels

## RETRIEVAL

find similar scenes in large archives

## REGRESSION

estimate continuous quantities

## CHANGE ANALYSIS

compare embeddings across time

## DIVERSITY ANALYSIS

measure coverage and variety

# Block 1 Takeaway

$$x_{\text{observation}} \rightarrow e_{\theta}(x) \rightarrow z$$

**Explicit geospatial embeddings** summarize observed data into reusable learned representations. Self-supervised pretraining removes the need for labels; scaling up yields geospatial foundation models.

# Implicit Geospatial Embeddings

# Must an Embedding Start From an Observation?

## OBSERVATION-BASED

$x_{\text{image}} \rightarrow z$

Requires an image or observation at query time.

## LOCATION-BASED

$p_{\text{location}} \rightarrow z$

Requires only geographic coordinates.

Can we learn a representation of a place without observing an image at query time?

# Discrete or Continuous Geographic Information?

## DISCRETE

- raster cells
- vector geometries
- point observations

Store values at explicit spatial supports

## CONTINUOUS

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

A continuous model stores parameters of a function instead of explicitly storing every spatial sample.

# Functional Models of the Earth

Large-scale Earth fields have long been represented as continuous functions — not stored pixel by pixel.

## GRAVITY

EGM2008: degree 2190 spherical  
harmonic model

## MAGNETIC FIELD

IGRF: global spherical harmonic  
coefficients

## GEOID

Physical sea-level surface as a  
functional model

The field is stored through *coefficients of a continuous function*, evaluated wherever needed.

# Basis Functions

A general functional model represents a field as a linear combination of basis functions:

$$f(x) = \sum_k a_k \phi_k(x)$$

**$\phi_k(x)$** : basis functions — predefined shapes that span the function space

**$a_k$** : coefficients — the learned or fitted parameters. The coefficients *are* the representation.

# Fourier Features

Sine and cosine are the simplest basis functions on a periodic domain:

$$\phi_k(x) = \sin(kx), \cos(kx)$$

**Low  $k$**  — broad spatial variation

**High  $k$**  — fine spatial detail

[ sinusoidal basis functions at  $k = 1, 2, 3, \dots$  ]

→ Connects directly to positional encoding in neural networks.

# From Fourier Modes to Spherical Harmonics

Fourier functions work on periodic or Euclidean domains. The Earth is a sphere.

$\sin, \cos$   
 $\rightarrow \text{sphere}; Y_{\ell m}(\theta, \lambda)$

Spherical harmonics are the natural generalization of sinusoidal basis functions to  $S^2$ .

[ circle  $\rightarrow$  sphere transition ]

The geometry of the domain matters.

# Geometry-Aware Basis Functions

LINE / PERIODIC

$$\sin(kx), \cos(kx)$$

Fourier modes

SPHERE  $S^2$

$$Y_{\ell m}(\theta, \lambda)$$

Spherical harmonics

GENERAL DOMAIN

$$\Delta \phi_k = \lambda_k \phi_k$$

Laplacian eigenfunctions

Each domain determines its own natural frequency basis through the geometry.

# Neural Networks for Continuous Signal Modeling

Instead of fixed basis functions, a neural network can parameterize the continuous function directly.



The network parameters  $\theta$  are the representation of the continuous function.

# Coordinate Networks

The simplest formulation: the input is the coordinate; the output is the signal at that coordinate.



# An Image as a Function

## DISCRETE: PIXEL GRID

[ pixel grid ]

Store  $H \times W \times 3$  values explicitly

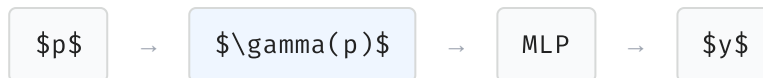
## CONTINUOUS: NEURAL FUNCTION

$$f_{\theta}(x, y) \rightarrow \mathrm{RGB}$$

Store network weights; evaluate at any  $(x,y)$

# Positional Encoding

Map coordinates to sinusoidal features before the MLP to represent fine spatial variation more easily.



The encoding  $\gamma(p) = [\sin(2\pi B p), \cos(2\pi B p)]$  injects multiple frequencies explicitly. The MLP only needs to combine them.

# SIREN

## FOURIER FEATURES

$$p \rightarrow \gamma(p) \rightarrow \mathrm{MLP}$$

Sinusoids *before* the network

## SIREN

$$\sigma(x) = \sin(\omega_0 x)$$

Sinusoidal activations *inside* the network

Both address spectral bias; SIREN propagates sinusoidal structure through all layers.

# Neural Fields Beyond Geography

Coordinate-based neural functions appear across many domains — not only geospatial.

**Images:**  $(x, y) \rightarrow \text{RGB}$

**Signed-distance functions:**  $\mathbf{x} \rightarrow \text{SDF}$   
(3D shape)

**Occupancy:**  $\mathbf{x} \rightarrow \text{occupied / free}$

**Radiance fields (NeRF):**  $(\mathbf{x}, \mathbf{d}) \rightarrow \text{colour + density}$

The same pattern — coordinate in, signal out — recurs across domains.

# NeRF as an Example

$(\mathbf{x}, \mathbf{d})$

→

$f_{\theta}$

→

$(\sigma, \mathbf{c})$

$\mathbf{x}$ : 3D position;  $\mathbf{d}$ : view direction

$\sigma$ : density;  $\mathbf{c}$ : colour

Complex signals can be represented as continuous neural functions over coordinates.

# Bring the Coordinate Network Back to Earth

Using coordinate networks for geographic space introduces domain-specific challenges.

**Longitude discontinuity:**  $-180^\circ$  and  $+180^\circ$  are the same location

**Poles:** longitude becomes degenerate at  $\pm 90^\circ$  latitude

**Spherical geometry:** Euclidean distances are not geographic distances

**Global scale:** fine local and broad global structure must coexist

# Geographic Coordinate Encodings

How we encode the location input matters.

**Raw  $(\phi, \lambda)$ :** discontinuity at  $\pm 180^\circ$ ; poles degenerate

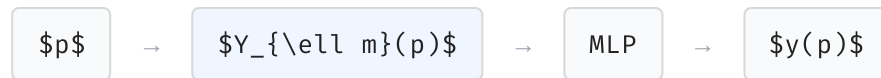
**3D Cartesian  $(x, y, z)$ :** continuous on the sphere; no discontinuity

**Fourier features:**  $[\sin(k\phi), \cos(k\phi), \dots]$  — injects frequency

**Spherical harmonics:** geometry-aware, no discontinuity

**Geometry-aware spectral:** designed for the sphere from the start

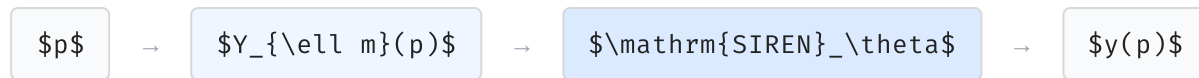
# Spherical Harmonics + Neural Network



The classical spherical harmonic basis becomes an input representation for a neural model.

The MLP learns nonlinear combinations of spherical harmonic features — more expressive than a purely linear model.

# Spherical Harmonics + SIREN



**Spherical harmonics** → geometry-aware input encoding

**SIREN** → nonlinear high-frequency modeling

# Residual and Multi-Network Variants

## RESIDUAL SIREN

Skip connections through sinusoidal layers; better gradient flow

## MULTIPLE LOCAL MLPs

Different networks for different spatial regions

## HIERARCHICAL MODELS

Coarse-to-fine scale decomposition

## MIXTURE APPROACHES

Soft partition of space into locally specialized experts

# Geographic Neural Fields

$$(p_i, y_i) \rightarrow f_{\theta}(p) \rightarrow y$$

Target signals over geographic coordinates:

Climate

Elevation

Land properties

Population

Biodiversity

Vegetation

# Species Observations as a Spatial Signal



Georeferenced occurrence records form a spatial signal — presence at location  $p$ .

A coordinate network can learn to predict species occurrence from location — effectively learning the spatial distribution of a taxon without explicit rasterization.

# From Signal Prediction to Location Representation

## SIGNAL MODEL

$$p \rightarrow y$$

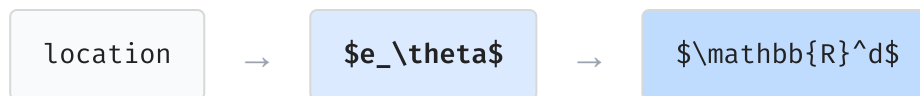
The output is a signal value.

## REPRESENTATION MODEL

$$p \rightarrow \underline{e} \rightarrow z(p) \rightarrow y$$

The hidden representation itself is the object of interest.

# Location Encoders



A location encoder maps geographic coordinates directly to a reusable  $d$ -dimensional embedding.

The embedding is implicit: spatial information is encoded in model parameters, not retrieved from local observations.

# What Can a Location Embedding Encode?

Climate

Ecosystems

Land use

Biogeography

Human activity

Visual appearance

The representation is **implicit**: information is encoded in model parameters, not retrieved from a local observation at query time.

# SatCLIP / GeoCLIP-Like Models

Train image and location encoders jointly so corresponding pairs become similar in embedding space.

## IMAGE BRANCH

image  $\rightarrow z_{\text{image}}$

## LOCATION BRANCH

location  $\rightarrow z_{\text{location}}$

Contrastive training aligns corresponding image–location pairs. The location encoder inherits visual semantics without needing an image at query time. [Links to Block 1 contrastive learning.](#)

# The Two Blocks Meet

## EXPLICIT BRANCH

$$x(p) \rightarrow e_{\text{obs}} \rightarrow z_{\text{explicit}}$$

From observing the place

## IMPLICIT BRANCH

$$p \rightarrow e_{\text{loc}} \rightarrow z_{\text{implicit}}$$

From knowing where the place is

Contrastive learning can align them:  $z_{\text{explicit}} \rightarrow z_{\text{implicit}}$

# Explicit vs Implicit Geospatial Embeddings

|                           | Explicit                      | Implicit                     |
|---------------------------|-------------------------------|------------------------------|
| Input                     | observation ( $x$ )           | location ( $p$ )             |
| Encoder                   | $e(x)$                        | $e(p)$                       |
| Observation at query time | required                      | not required                 |
| Information source        | local observation             | learned spatial associations |
| Output                    | $z$                           | $z$                          |
| Example                   | EO foundation-model embedding | SatCLIP location encoder     |

# Two Views of the Same Place

OBSERVATION → EMBEDDING

$$x(p) \rightarrow z_{\text{explicit}}$$

Requires sensing the place

LOCATION → EMBEDDING

$$p \rightarrow z_{\text{implicit}}$$

Only requires coordinates

One embedding comes from observing the place; the other comes from knowing where it is.

# Block 2 Takeaway

$$p_{\text{location}} \rightarrow e_{\theta}(p) \rightarrow z$$

**Implicit geospatial embeddings** generate representations directly from location using learned spatial structure. No observation is required at query time.

# Final Course Synthesis

# What Can Be a Representation?

## ENGINEERED REPRESENTATION

$$x \rightarrow \phi(x)$$

Human design — hand-crafted features, spectral indices, kernels

## EXPLICIT LEARNED REPRESENTATION

$$x \rightarrow e_{\theta}(x) \rightarrow z$$

Learned from observations — deep learning, self-supervised, foundation models

## IMPLICIT LEARNED REPRESENTATION

$$p \rightarrow e_{\theta}(p) \rightarrow z$$

Learned from spatial supervision — coordinate networks, location encoders

# Where Does the Information Come From?

## ENGINEERED

Human design

Domain knowledge drives feature  
construction

## EXPLICIT LEARNED

Observations + learning objective

The data itself drives what is encoded

## IMPLICIT LEARNED

Spatial supervision + learned  
associations

Spatial structure in data drives  
geographic embedding

# Geospatial Representation Learning



Geospatial representation learning is about deciding **what information should be encoded, where that information comes from, and how reusable the resulting representation should be.**

# Course Takeaway

**Good representations turn complex geospatial information into forms that make useful tasks easier.**



## LECTURE 6

# IMPLICIT NEURAL REPRESENTATIONS

Representing geospatial information as a continuous  
learned function

PRESS – OR SPACE. →

# Learning Outcomes

- 1 Explain how geospatial information can be represented as **discrete data or continuous functions**.
- 2 Explain how **coordinate-based neural networks** model continuous signals using positional encodings and sinusoidal features.
- 3 Explain how **implicit neural representations** can model geographic space and produce either spatial signals or location embeddings.

# Where We Are in the Course

## LECTURE 4

### Model & Optimization

How does a neural network learn from labeled data?

## LECTURE 5

### Generalization

How does learned structure transfer beyond the training data?

## LECTURE 6 — TODAY

### Representing Data as Functions

Can geospatial information be represented as a continuous learned function instead of discrete samples?

Main question: Can geospatial information be represented as a continuous learned function instead of only as discrete samples?

# Discrete or Continuous?

## DISCRETE

Store observations explicitly at discrete spatial supports.

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

Examples: raster cells · vector geometries · point samples

## CONTINUOUS

Store the parameters of a function; evaluate wherever needed.

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Examples: Fourier representations · spherical harmonic models · interpolation functions · **neural fields / implicit neural representations**

Neural fields are one type of continuous functional model — not a separate category.

Geospatial analysis commonly discretizes space — but global physical fields have always been modeled as continuous functions.

# Functional Models of the Earth

# The Earth as Discrete Data

## RASTER

Regular grid of cells, each holding a value

## VECTOR

Points, lines, polygons with explicit geometry

## POINT OBSERVATIONS

Irregular samples at measured locations

Most geospatial tools discretize: store values at explicit supports, then process those values.

# What Happens at Global Scale?

Some geophysical quantities vary continuously over the entire Earth:

- Gravity field
- Magnetic field
- Geoid (sea-level surface)
- Atmospheric pressure
- Temperature and precipitation

[ global field image ]

Must a global field be stored as billions of individual pixels?

# From Samples to Functions

General functional model:

$$f(x) = \sum_k a_k \phi_k(x)$$

- $\phi_k$ : basis functions — predefined shapes that span the space
- $a_k$ : coefficients — the learned or fitted parameters
- The coefficients *are* the stored representation of the field

## DISCRETE

Store  $N$  values at  $N$  locations

## FUNCTIONAL

Store  $K$  coefficients; evaluate  $f(x)$  anywhere

# Sine and Cosine as Basis Functions

The simplest basis functions on a periodic domain are sinusoids:

$$\phi_k(x) = \sin(kx), \quad \cos(kx)$$

- Each basis function oscillates at a different frequency  $k$
- Together they can represent any square-integrable periodic signal

[ sinusoidal basis functions at  $k = 1, 2, 3, \dots$  ]

This prepares us for Fourier representations and, later, positional encoding in neural networks.

# Fourier Representations

Any signal can be decomposed into components at different spatial frequencies:

$$f(x) = \sum_k (a_k \cos(kx) + b_k \sin(kx))$$

**Low frequencies** — broad, smooth variation

**High frequencies** — fine spatial detail

[ Fourier decomposition figure ]

→ This connects directly to **spectral bias** and **positional encoding** in Block 2.

# From the Circle to the Sphere

Fourier functions work on periodic or Euclidean domains. The Earth is a sphere.

$$\begin{aligned} & \sin, \cos \\ & \quad \rightarrow \text{sphere} \\ & Y_{\ell m}(\theta, \lambda) \end{aligned}$$

Spherical harmonics are the natural generalization of sinusoidal basis functions to the sphere  $S^2$ .

[ circle → sphere transition figure ]

# Spherical Harmonics

A field on the sphere is modeled as:

$$f(\theta, \lambda) = \sum_{\ell, m} a_{\ell m} Y_{\ell m}(\theta, \lambda)$$

- $\ell$ : degree — controls the spatial scale
- $m$ : order — controls the longitudinal variation
- Increasing  $\ell$  adds finer spatial detail

[ spherical harmonic visualization  $Y_{\ell m}$  ]

# Modeling Global Earth Fields

Spherical harmonics are the standard representation in physical geodesy:

- Gravity field (e.g. EGM2008: degree 2190)
- Geomagnetic field (IGRF model)
- Earth's topography

The field is stored through coefficients of a continuous functional model — not as billions of pixels.

[ global field represented via spherical harmonics ]

# Where Do the Basis Functions Come From?

So far we chose basis functions by hand (sinusoids, spherical harmonics).

Can the **geometry of the domain itself** determine useful basis functions?

The answer is yes — through the eigenfunctions of the Laplacian.

# The Laplacian

The Laplacian  $\Delta$  measures local variation of a function on a domain.

- On  $\mathbb{R}^n$ :  $\Delta f = \sum_i \partial^2 f / \partial x_i^2$
- Captures local smoothness
- Generalizes naturally to manifolds and graphs
- Encodes the geometry of the domain

[ Laplacian as local variation ]

# Eigenfunctions of the Laplacian

Eigenfunctions  $\phi_k$  satisfy:

$$\Delta \phi_k = \lambda_k \phi_k$$

- $\lambda_k$ : eigenvalue — controls the spatial frequency
- $\phi_k$ : eigenfunction — a basis shape adapted to the domain
- Together they form a geometry-aware basis for representing any function on the domain

[ Laplacian eigenfunctions on a domain ]

# Geometry Determines the Basis

LINE / PERIODIC

$$\sin(kx), \cos(kx)$$

Fourier modes

SPHERE  $S^2$

$$Y_{\ell m}(\theta, \lambda)$$

Spherical harmonics

GENERAL MANIFOLD / GRAPH

$$\Delta \phi_k = \lambda_k \phi_k$$

Laplacian eigenfunctions

Each domain determines its own natural frequency basis through the geometry.

# Block 1 Takeaway

**A spatial field can be represented by the *parameters of a continuous function* rather than by explicitly storing every sample.**

Classical geodesy and signal processing have exploited this for decades: spherical harmonic models of gravity, magnetic, and other global fields store coefficients, not pixels.

Next: what if the continuous function is parameterized by a neural network?

Fixed functional models choose basis functions by hand. Neural networks learn the function directly from data.

# Neural Networks for Continuous Signal Modeling

# From Fixed Functional Models to Neural Networks

## CLASSICAL: FIXED BASIS

$$f(x) = \sum_k a_k \phi_k(x)$$

Basis functions  $\phi_k$  are fixed; only coefficients  $a_k$  are learned.

## NEURAL: LEARNED FUNCTION

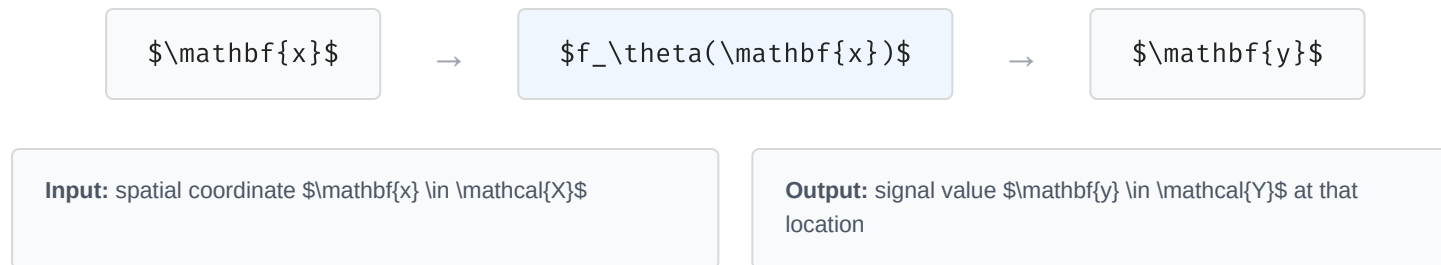
$$f_{\theta}(x) = \mathrm{NN}_{\theta}(x)$$

The entire function is parameterized by network weights  $\theta$ .

What if the continuous function itself is parameterized by a neural network?

# Coordinate-Based Neural Networks

The most direct formulation: the input is a coordinate; the output is the signal at that coordinate.



# An Image as a Continuous Function

## DISCRETE: PIXEL GRID

[ pixel grid ]

Store  $H \times W \times 3$  values explicitly

## CONTINUOUS: NEURAL FUNCTION

$$f_{\theta}(x, y) \rightarrow \mathrm{RGB}$$

Store network weights  $\theta$ ; evaluate at any  $(x,y)$

# Fit the Samples, Query the Function

## TRAINING

$$\min_{\theta} \sum_i \|f_{\theta}(\mathbf{x}_i) - y_i\|^2$$

Observed coordinates and values supervise the network.

## INFERENCE

$$y^* = f_{\theta}(\mathbf{x}^*)$$

Evaluate the function at any arbitrary coordinate.

The network learns a continuous function from discrete observations — much like classical interpolation, but with a learned parametric form.

# Spectral Bias

Standard MLPs with ReLU activations preferentially fit **low-frequency** variation first.

- Smooth, broad patterns are learned quickly
- Fine high-frequency structure is learned slowly or not at all

Recall from Block 1: Fourier decomposition separates signals by frequency. Spectral bias is a neural analogue of truncating the Fourier series.

[ spectral bias: MLP learns low frequencies first ]

# Positional Encoding and Fourier Features

Idea: map coordinates to explicit sinusoidal features before the MLP.

$$\phi(\mathbf{x}) = \left[ \sin(2\pi B_1 \mathbf{x}), \cos(2\pi B_1 \mathbf{x}), \sin(2\pi B_2 \mathbf{x}), \cos(2\pi B_2 \mathbf{x}), \dots \right]$$

followed by an MLP on  $\phi(\mathbf{x})$ .

- $B$ : frequency matrix (random or learned)
- Directly injects multiple frequency scales
- Overcomes spectral bias for fine detail

[ Fourier feature encoding diagram ]

Tancik et al. 2020

# Why Fourier Features Help

## RAW COORDINATES

$$(x, y) \rightarrow \mathrm{MLP}$$

MLP must learn all frequency content from scratch. Spectral bias limits high-frequency capacity.

## FOURIER-ENCODED COORDINATES

$$\gamma(x, y) \rightarrow \mathrm{MLP}$$

Multiple frequencies are already explicit in the input. MLP only needs to combine them.

Explicit frequency features make fine spatial variation much easier for the network to represent.

# SIREN

Alternative approach: use sinusoidal *activation functions* inside the network.

$$h_{l+1} = \sin(W_l h_l + b_l)$$

**Key distinction:**

Fourier features — sinusoidal *input encoding*

SIREN — sinusoidal *activation functions inside the network*

[ SIREN network diagram ]  
Sitzmann et al. 2020

# Fourier Basis, Fourier Features, SIREN

## CLASSICAL SIGNAL PROCESSING

$$f(x) = \sum_k a_k \phi_k(x)$$

Fourier basis: fixed sinusoidal functions

## NEURAL INPUT ENCODING

$$\gamma(\mathbf{x}) = [\sin(B\mathbf{x}), \cos(B\mathbf{x})]$$

Fourier features: sinusoidal input to MLP

## SIREN ACTIVATIONS

$$h_{l+1} = \sin(W_l h_l + b_l)$$

Sinusoidal activations inside the network

The same idea — spatial frequency decomposition — appears at every level of the representation.

# Continuous Neural Fields

The coordinate-network idea generalizes across many signal types:

Images

$(x, y) \rightarrow \mathrm{RGB}$

Audio

$t \rightarrow \text{amplitude}$

Signed-distance functions

$(x, y, z) \rightarrow \mathrm{SDF}$

Occupancy fields

$(x, y, z) \rightarrow \{0, 1\}$

Physical fields

$\mathbf{x} \rightarrow \text{temperature}$

Radiance fields (NeRF)

$(\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathrm{RGB})$

# NeRF

NeRF (Neural Radiance Field) models a 3D scene as a continuous function:

$$(\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$$

- $\mathbf{x}$ : 3D position
- $\mathbf{d}$ : viewing direction
- $\sigma$ : volume density
- $\mathbf{c}$ : RGB color

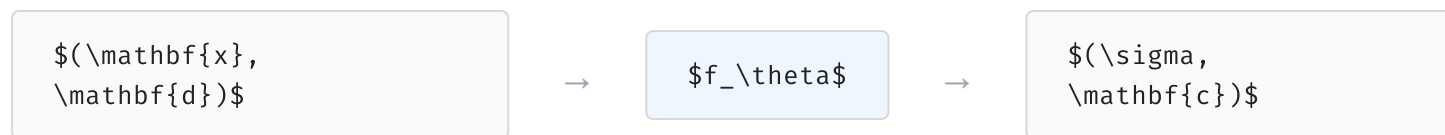
Novel views are rendered by ray-marching through the neural field.

[ NeRF scene representation ]

Mildenhall et al. 2020

# A Scene as a Continuous Neural Field

The conceptual key: a 3D scene is not stored as explicit geometry or a point cloud — it is the network itself.



Any point in the scene can be queried through the neural function — no explicit mesh, voxel grid, or point set required.

# The Same Pattern Across Domains

Common pattern:



# Block 2 Takeaway

**Neural networks can parameterize continuous functions and model signals directly from coordinates.**

Spectral bias limits plain MLPs. Positional encodings (Fourier features) and sinusoidal activations (SIREN) overcome this by explicitly representing spatial frequencies — echoing Block 1's functional models.

Next: what changes when the domain is the Earth?

Synthesizing Blocks 1 and 2: geometry-aware coordinate encodings meet neural continuous functions.

# Implicit Neural Representations of Geographic Space

# What Changes When the Domain Is the Earth?

## EUCLIDEAN IMAGE PLANE

- Flat, periodic domain
- Uniform distance metric
- $(x, y)$  coordinates well-defined
- Fourier features work directly

## THE SPHERE

- Longitude is discontinuous ( $-180^\circ/+180^\circ$ )
- Poles have degenerate coordinates
- Spherical geometry, not Euclidean
- Distance in lat/lon is non-uniform

Coordinate encodings designed for flat space may fail or distort when applied naively to geographic coordinates.

# Geographic Coordinates Are Not Just x and y

## LATITUDE / LONGITUDE

Simple but discontinuous; non-uniform distances

## 3D CARTESIAN ON SPHERE

Continuous and uniform; constrains  $x^2+y^2+z^2=1$

## FOURIER FEATURES

Fast; may distort at poles and date line

## SPHERICAL HARMONICS / LAPLACIAN

Geometry-aware; respects spherical topology

The choice of coordinate encoding determines how well the model respects the geometry of geographic space.

# Geometry-Aware Coordinate Encodings

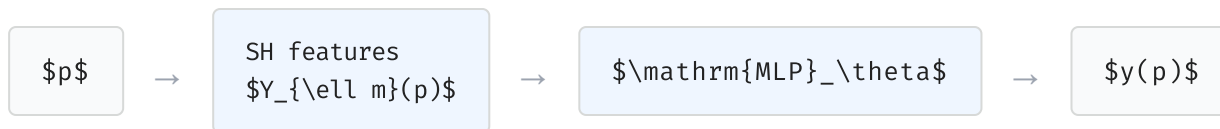
Reconnecting to Block 1: spherical harmonics can serve not only as a complete field model, but also as a **coordinate encoding** for a neural network.

$$p \mapsto Y_{\ell m}(p)$$

Map geographic location  $p = (\theta, \lambda)$  to a vector of spherical harmonic values as features.

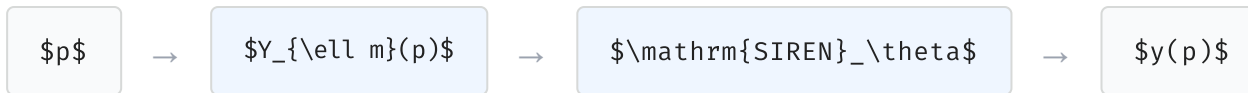
[ geographic coord → SH encoding → MLP ]

# Spherical Harmonics + Neural Network



The spherical harmonic encoding provides a **geometry-aware** positional representation that respects spherical topology before the neural network processes it.

# Spherical Harmonics + SIREN



## Spherical harmonics

Geometry-aware positional features; respect spherical topology

## SIREN

Nonlinear high-frequency modeling capacity beyond the SH basis

# Residual Implicit Networks

Residual coordinate network variants (e.g. ResIREN) extend the basic SIREN with skip connections:



# One Global Network or Many Local Networks?

## ONE GLOBAL MLP

Single network for the entire Earth. Simple but may lack local resolution.

## MULTIPLE LOCAL MLPs

One network per tile or region. High resolution; costly to scale.

## HIERARCHICAL / MULTI-SCALE

Coarse global + fine local network. Balanced capacity and resolution.

## MIXTURE-OF-EXPERTS

Dynamic routing to specialized sub-networks per location or scale.

Scaling continuous neural fields to large geographic domains is an active research direction.

# Geographic Neural Fields

General supervised formulation:

$$(p_i, y_i) \rightarrow f_{\theta}(p) \rightarrow y$$

Possible output signals:

- Climate variables
- Elevation
- Land cover / land use
- Population density
- Ecological observations

[ predicted geographic signal map ]

# Species Observations as a Spatial Signal

Biodiversity observations (e.g. iNaturalist) provide sparse, presence-only spatial data:

$$\text{location} \rightarrow P(\text{species} \mid \text{location})$$

A geographic neural field can estimate species distribution continuously from irregular observations.

[ species observation points → predicted distribution ]

# From Signal Prediction to Location Representation

## SIGNAL MODEL

$$f_{\theta}(p) \rightarrow y$$

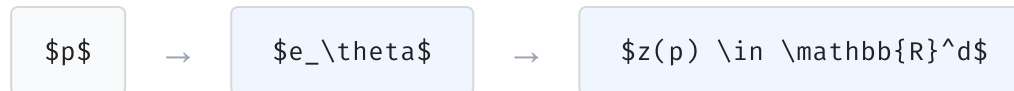
Predict one specific output (temperature, species, elevation) at any location.

## LOCATION ENCODER

$$p \mapsto e_{\theta}; z(p) \rightarrow y$$

Learn a *reusable* representation  $z(p)$  that can support many downstream tasks.

# Location Encoders

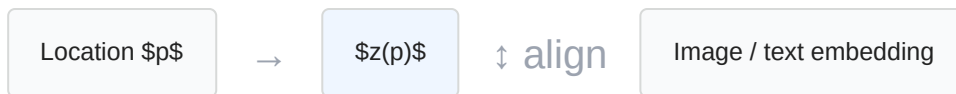


The output is a **reusable representation** of geographic location, rather than a prediction for one particular signal.

This is analogous to how image encoders learn general visual features rather than predicting a fixed label. The downstream task uses  $z(p)$  as input rather than the raw coordinate.

# GeoCLIP-Like Location Representations

Multimodal and contrastive learning approaches train location encoders by aligning geographic coordinates with other modalities (images, text, remote sensing).



Contrastive learning details are covered in a later lecture. Here the key point is that the location encoder output is a general-purpose geographic embedding.

# Signal Model or Representation Model?

## IMPLICIT SIGNAL MODEL

$$p \rightarrow y$$

One network, one output type. Train and use for a specific signal.

## LOCATION ENCODER

$$p \rightarrow z$$

Reusable embedding of location.  
Downstream task uses  $z$ .

## DOWNSTREAM TASK

$$z \rightarrow y$$

Lightweight head applied to the fixed location representation.

# Design Space of Geographic INRs

## DOMAIN

plane · sphere · manifold

## COORDINATE ENCODING

raw lat/lon · Fourier · spherical  
harmonics · Laplacian

## NEURAL MODEL

MLP · SIREN · residual network

## SCALE

global · local · hierarchical

## OUTPUT

signal · probability · embedding

## TRAINING

supervised · multimodal · self-  
supervised

# Neural Fields of the Earth

Earth coordinate → geometry-aware encoding → neural function →

geospatial signal  
– or –  
location embedding

# Lecture Synthesis

## BLOCK 1

### Functional Models of the Earth

$$p \mapsto \phi(p) \\ \mapsto y$$

Fixed basis; fitted coefficients

## BLOCK 2

### Neural Networks for Continuous Signals

$$p \mapsto \gamma(p) \\ \mapsto f_{\theta} \\ \mapsto y$$

Learned encoding; learned function

## BLOCK 3

### Implicit Neural Representations of Geographic Space

$$p \mapsto \text{geo encoding} \mapsto f_{\theta} \mapsto y \\ \text{ or } z$$

Geometry-aware; signal or embedding

Geospatial information can be stored **explicitly** at discrete locations, or **implicitly** through the parameters of a continuous function that can be queried throughout space.

# Location Encoders & Spatial Embeddings

Representing space continuously

# In this lecture (original outline)

- Coordinates as model inputs
- Positional and coordinate encodings
- Location encoders
- Spatial embeddings
- Implicit neural representations and continuous spatial functions