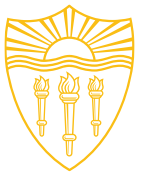


Location Encoding

-- building wheels of GeoAI

CSCI 587 Lecture 25

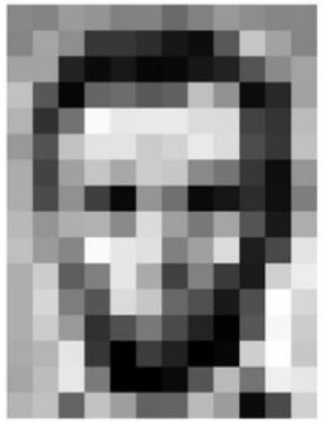
2026 Spring

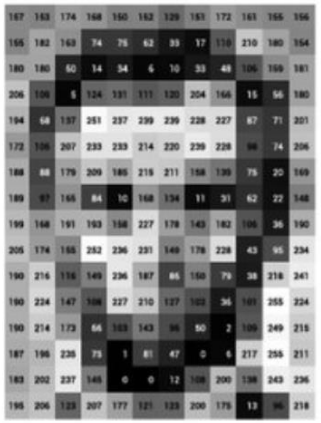
Thomas Lord
Department of Computer Science
USC Viterbi

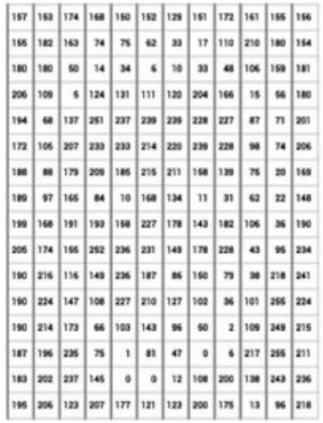
What is/Why representation learning

- Data structure.
  - Arrays
  - Strings

Abstract. We present a method that achieves state-of-the-art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views. Our algorithm represents a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location. We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis. View synthesis results are best viewed as videos, so we urge readers to view our supplementary video for convincing comparisons.







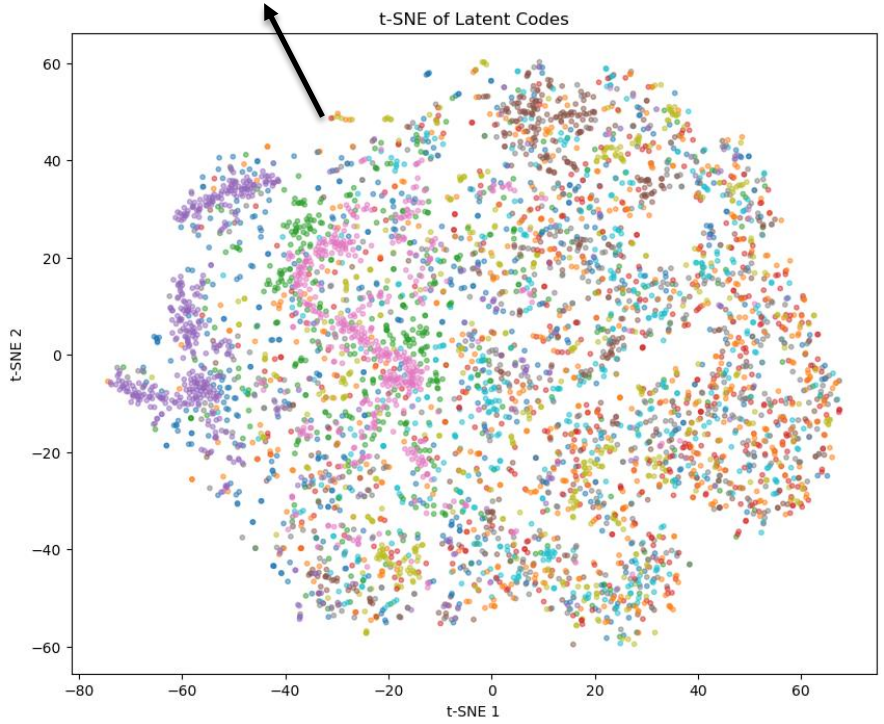
What is/Why representation learning

- Feature space
  - n = dimension
  - m = samples

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

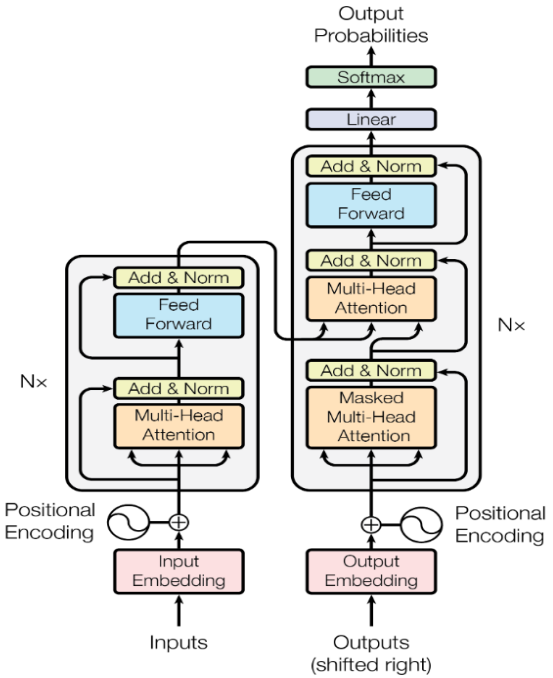
$$\begin{bmatrix} 0.124 \\ \dots\ \dots \\ -0.940 \end{bmatrix}$$

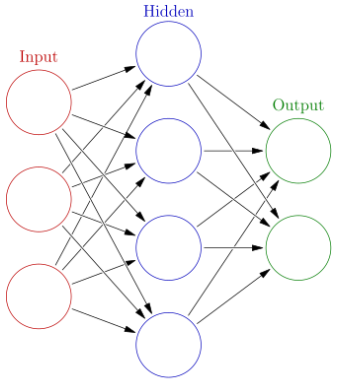
$\boldsymbol{v} \in \mathrm{R}^{1\times n}$ $\boldsymbol{V} \in \mathrm{R}^{m\times n}$



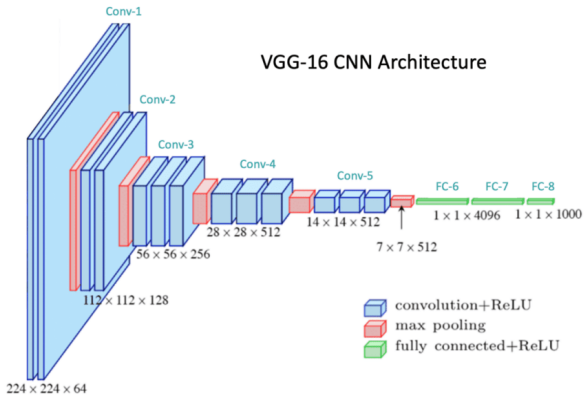
What is/Why representation learning

- Feature space
  - n = dimension
  - m = samples



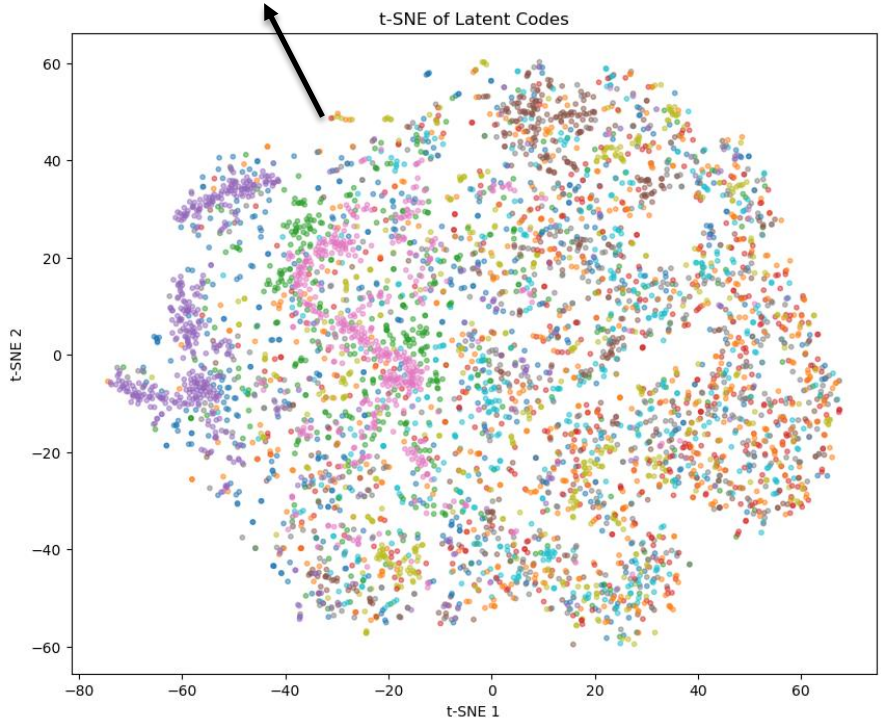


$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



$$\begin{bmatrix} 0.124 \\ \dots \dots \\ -0.940 \end{bmatrix}$$

$\boldsymbol{v} \in \mathrm{R}^{1\times n}$ $\quad$ $\boldsymbol{V} \in \mathrm{R}^{m\times n}$

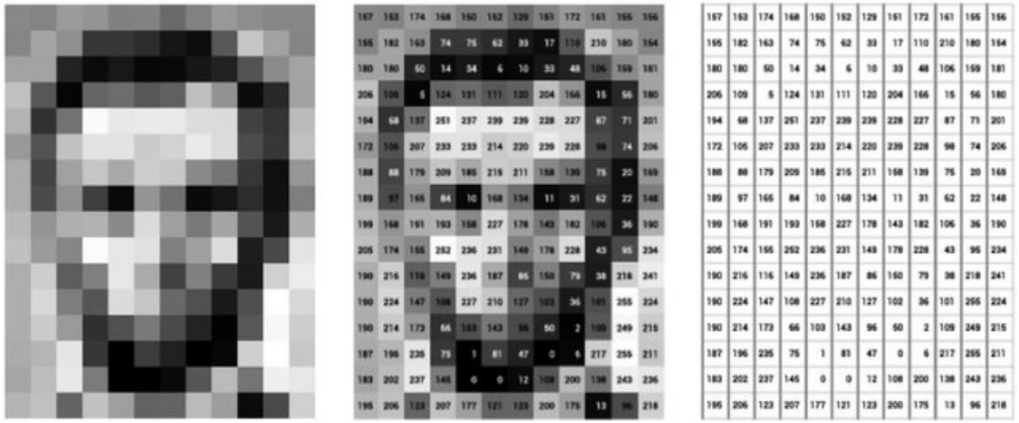


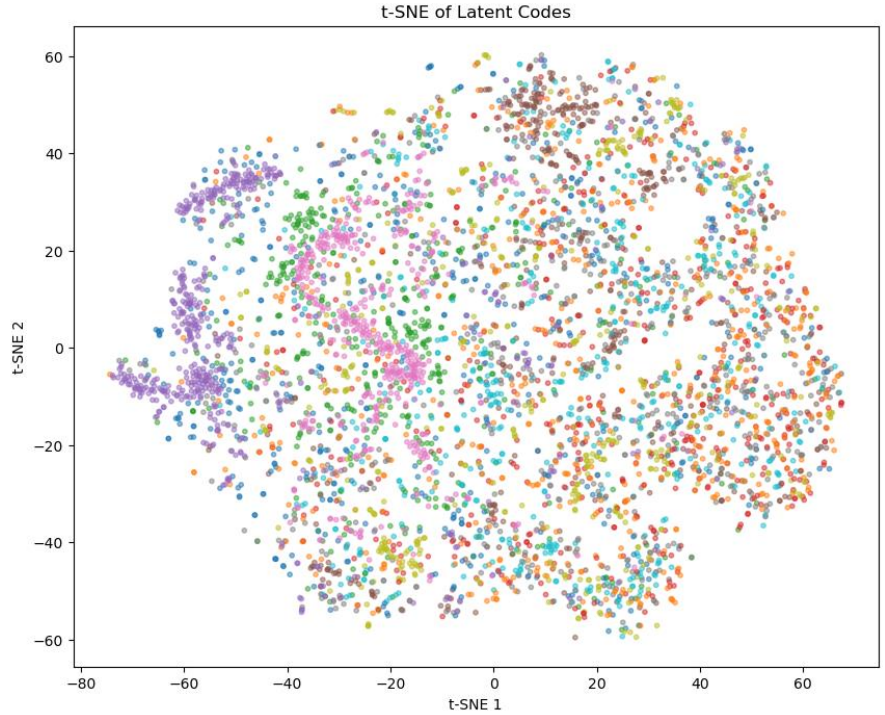
What is/Why representation learning

- Representation learning.

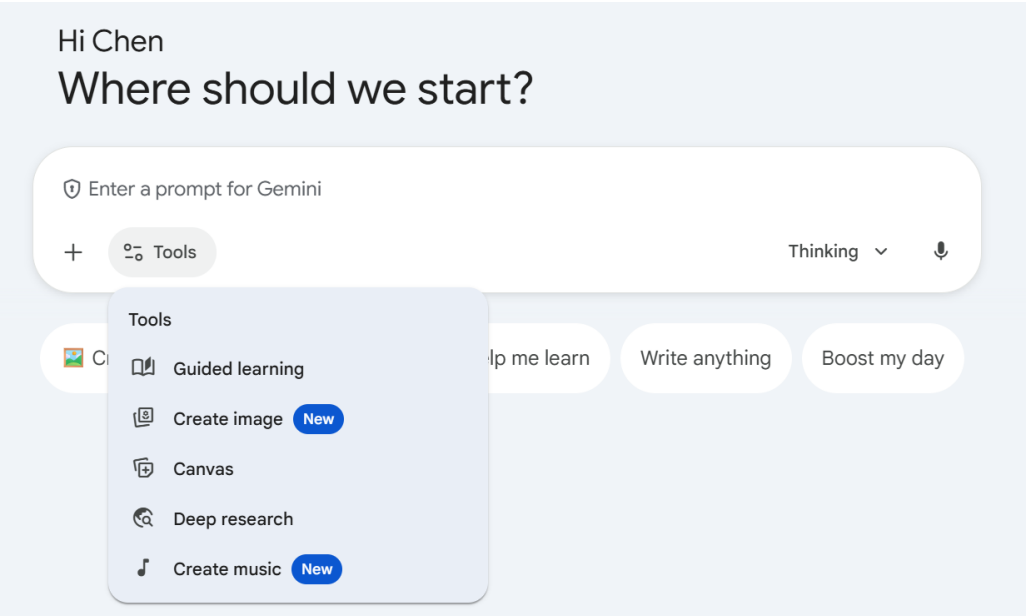
$$\theta : x \rightarrow \boldsymbol{v}$$

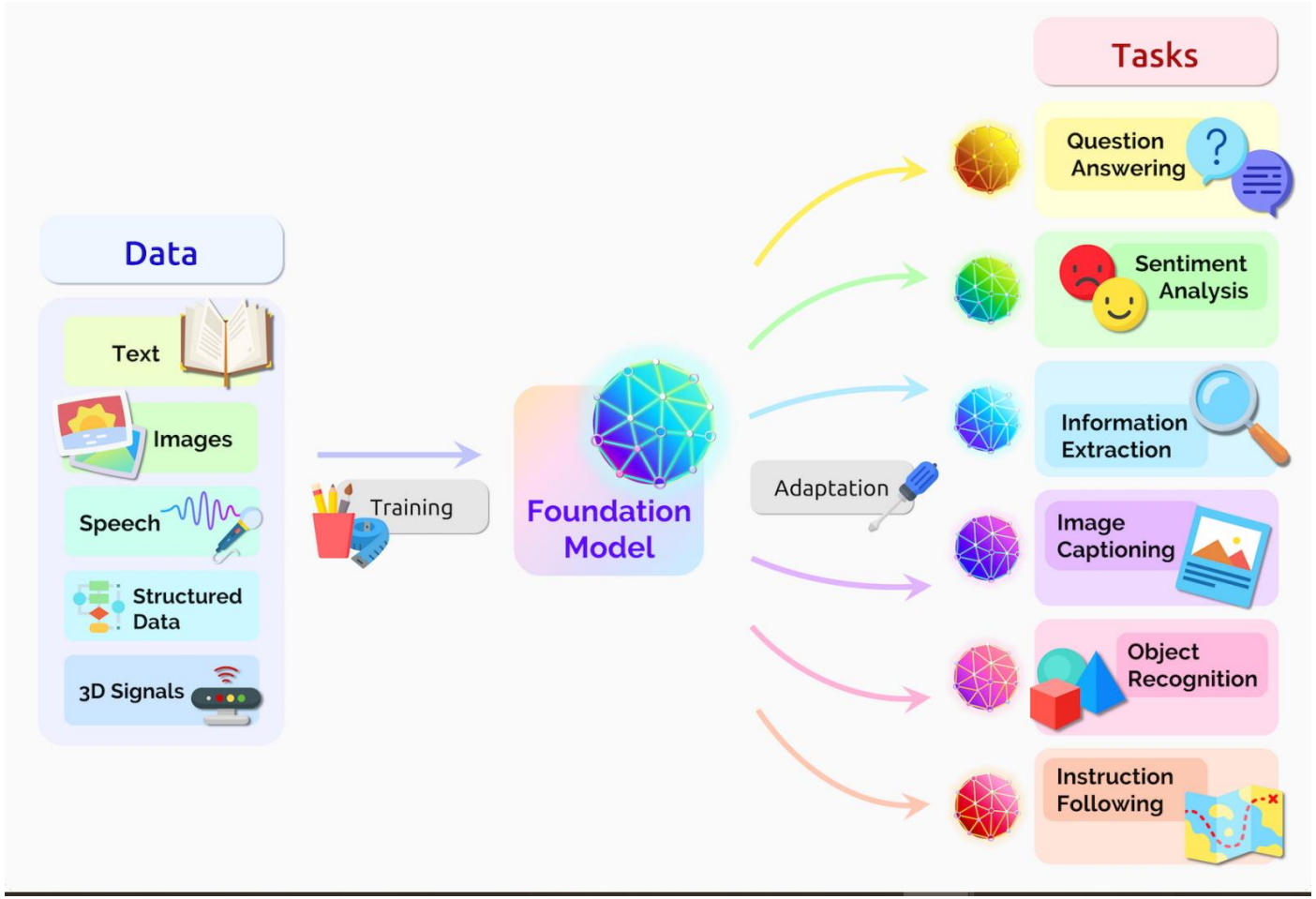
Abstract. We present a method that achieves state-of-the-art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views. Our algorithm represents a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location. We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis. View synthesis results are best viewed as videos, so we urge readers to view our supplementary video for convincing comparisons.





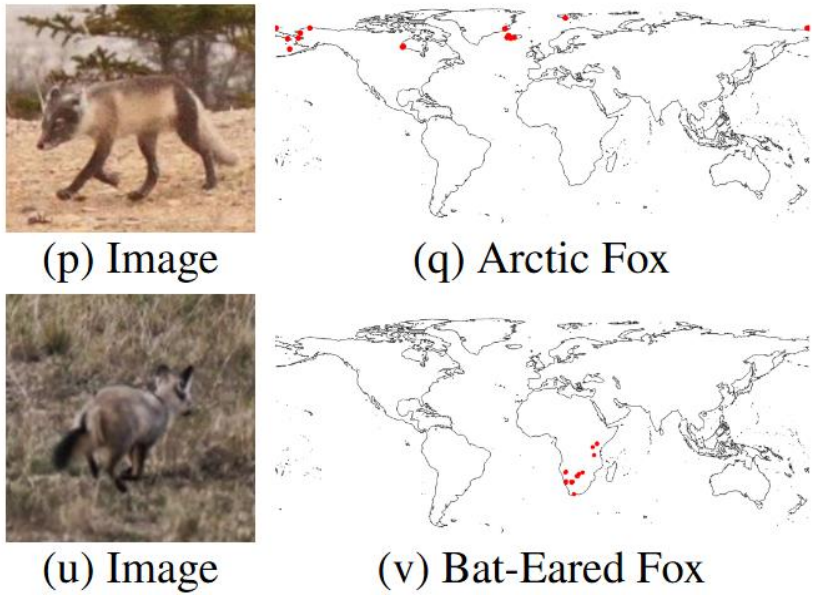
What is/Why representation learning

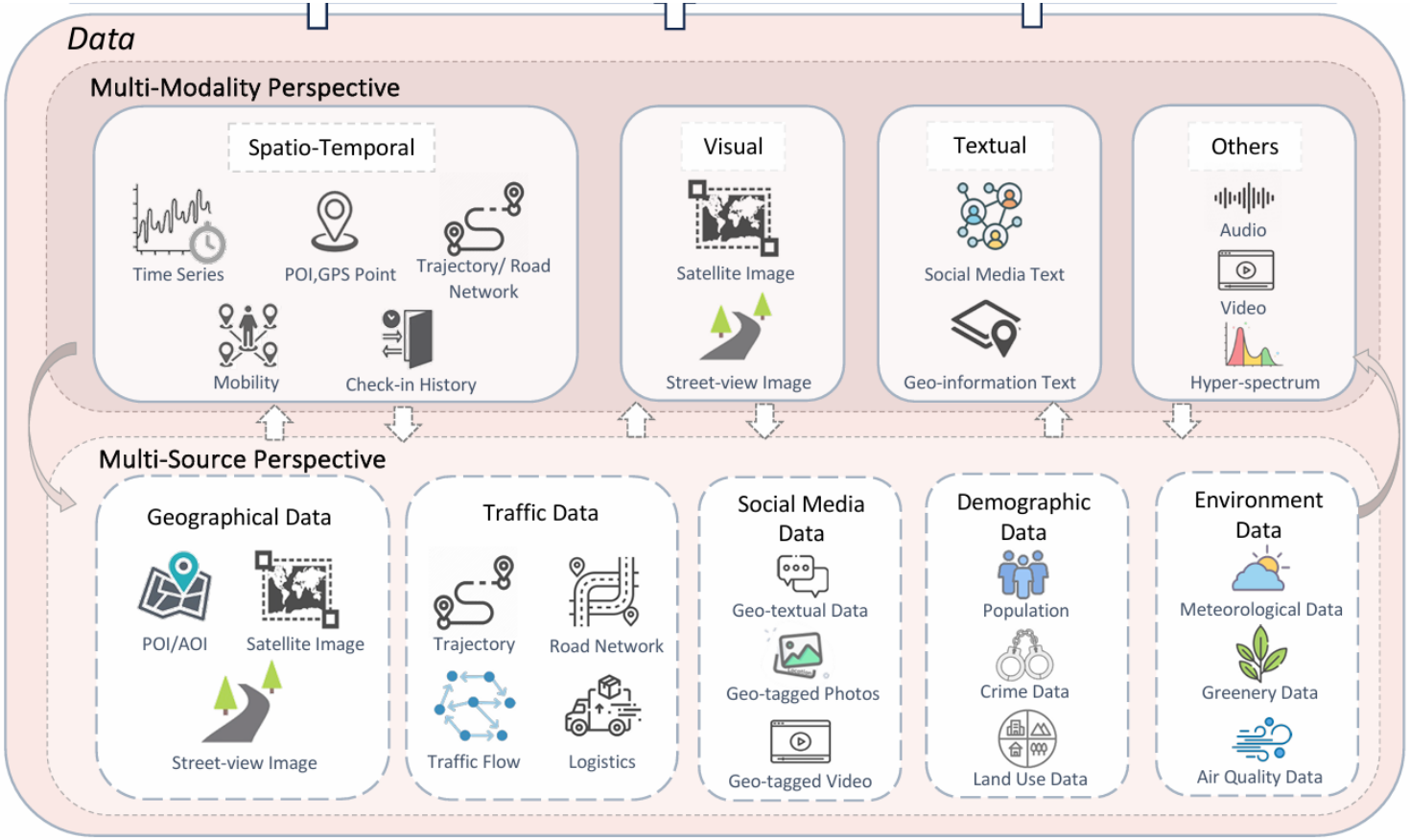




Why geospatial representation learning

80% of all data is spatial





Why geospatial representation learning

Adding location information to downstream models can boost the performance.

(p) Image

(q) Arctic Fox

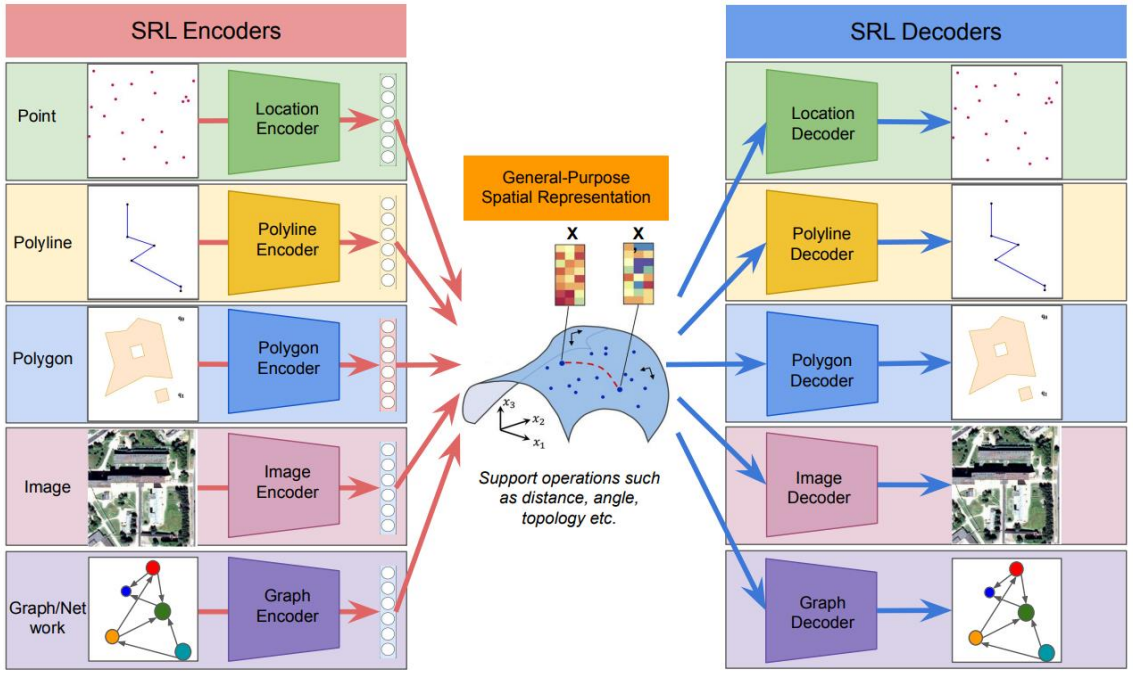
(u) Image

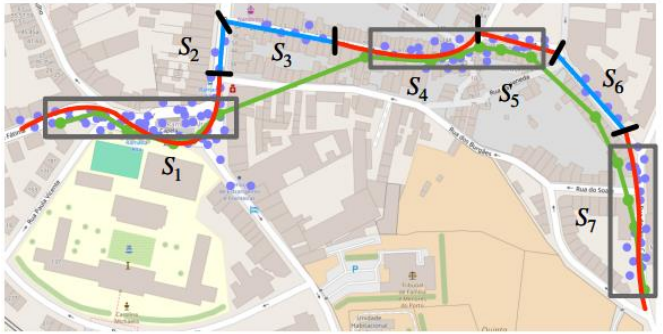
(v) Bat-Eared Fox

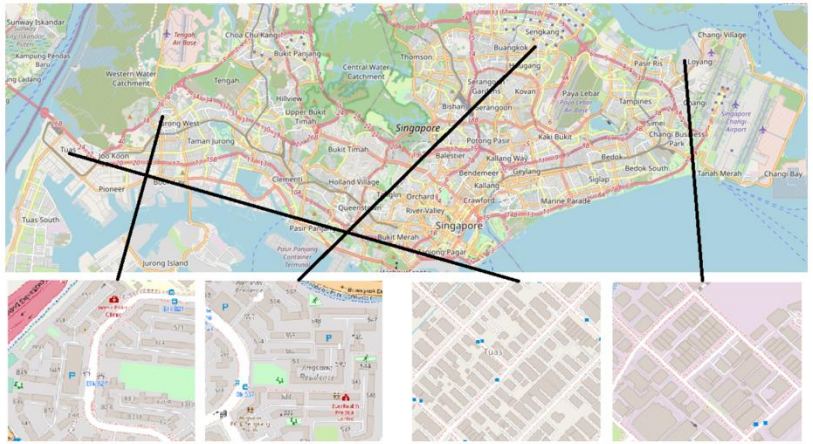
Task	Species Recognition						Flickr	RS
Dataset	BirdSnap	BirdSnap†	NABirds†	iNat2017	iNat2018	Avg	YFCC	fMOW
P(y|x) - Prior Type	Test	Test	Test	Val	Val	-	Test	Val
No Prior (i.e. image model)	70.07	70.07	76.08	63.27	60.20	67.94	50.15	69.84
tile (Tang et al., 2015)	70.16	72.33	77.34	66.15	65.61	70.32	50.43	-
xyz	71.85	78.97	81.20	69.39	71.75	74.63	50.75	70.18
wrap ∗ (Mac Aodha et al., 2019)	71.66	78.65	81.15	69.34	72.41	74.64	50.70	-
wrap	71.87	79.06	**81.62**	69.22	72.92	74.94	50.90	70.29
wrap + *ffn*	**71.99**	79.21	81.36	69.40	71.95	74.78	50.76	70.28
rbf (Mai et al., 2020b)	71.78	79.40	81.32	68.52	71.35	74.47	51.09	70.65
rff (Rahimi et al., 2007)	71.92	79.16	81.30	69.36	71.80	74.71	50.67	70.27
Space2Vec-grid (Mai et al., 2020b)	71.70	79.72	81.24	69.46	73.02	75.03	**51.18**	**70.80**
Space2Vec-theory (Mai et al., 2020b)	71.88	**79.75**	81.30	**69.47**	**73.03**	**75.09**	**51.16**	**70.81**
NeRF (Mildenhall et al., 2020)	71.66	79.66	81.32	69.45	73.00	75.02	50.97	70.64
Sphere2Vec-sphereC	72.11	79.80	81.88	69.68	73.29	75.35	**51.34**	71.00
Sphere2Vec-sphereC+	**72.41**	80.11	**81.97**	**69.75**	73.31	75.51	51.28	71.03
Sphere2Vec-sphereM	72.06	79.84	81.94	69.72	73.25	75.36	**51.35**	70.99
Sphere2Vec-sphereM+	72.24	**80.57**	81.94	69.67	**73.80**	**75.64**	51.24	71.10
Sphere2Vec-dfs	71.75	79.18	81.39	69.65	73.24	75.04	51.15	**71.46**

Why geospatial representation learning

- **Spatial Representation Learning** is the first step for almost all GeoAI tasks.





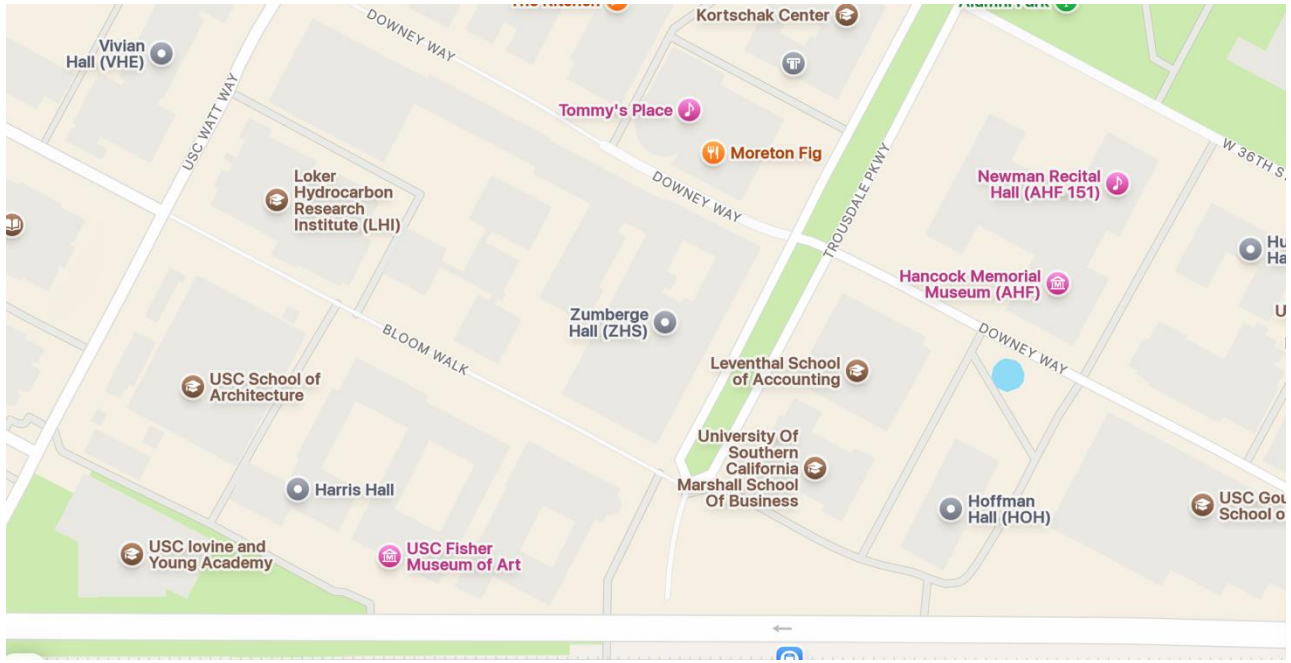


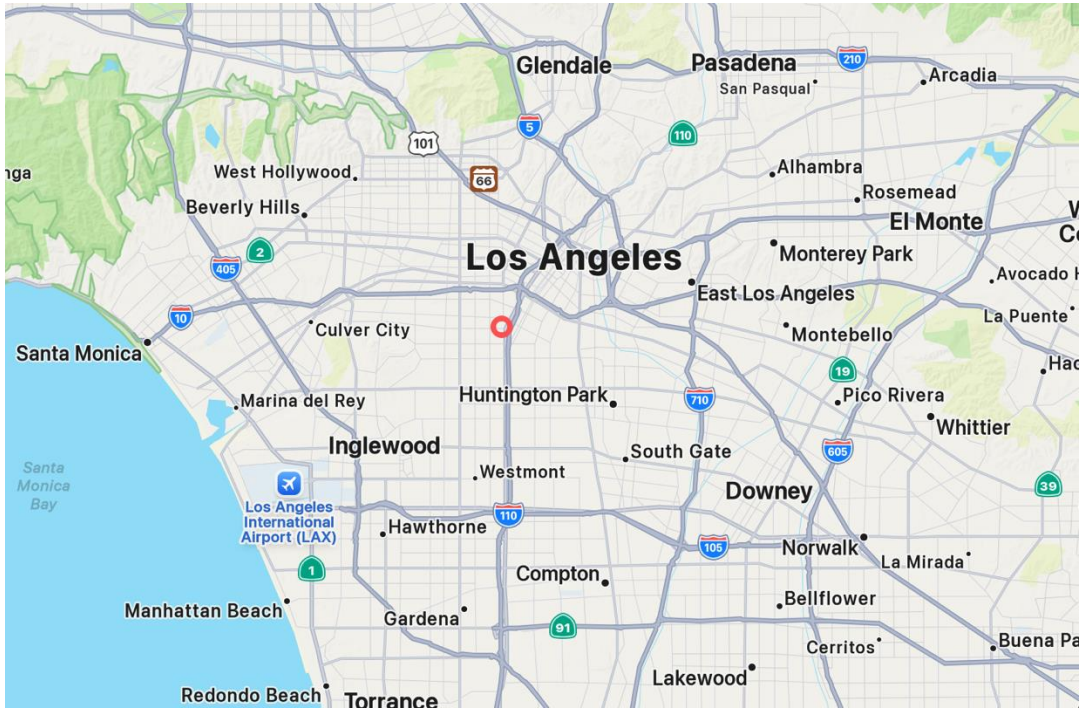
Residential Areas Industrial Areas

Gengchen Mai, Xiaobai Yao, Yiqun Xie, Jinmeng Rao, Hao Li, Qing Zhu, Ziyuan Li, and Ni Lao. 2024. SRL: Towards a General-Purpose Framework for Spatial Representation Learning. In Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24).

Point Representation

- Scaling up

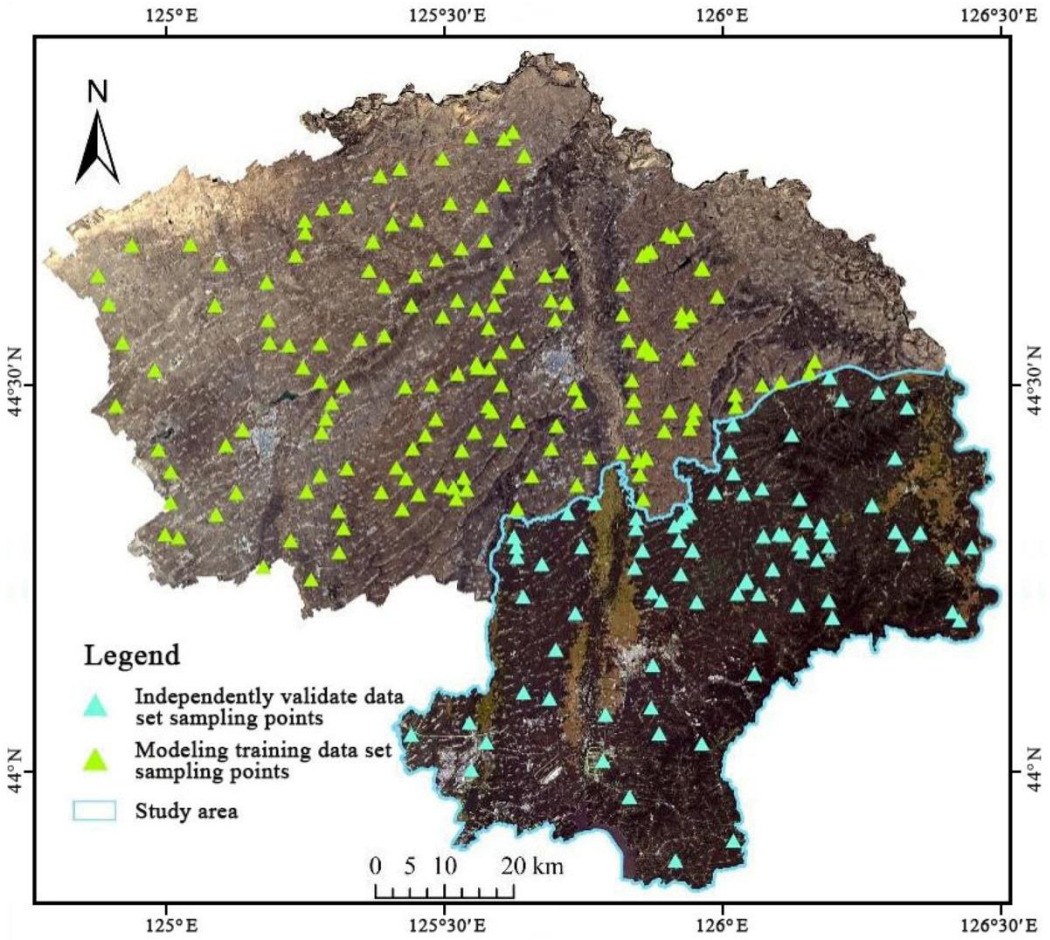





NORTH AMERICA
ROCKY MOUNTAINS
GREAT PLAINS
Vancouver
Montréal
Toronto
Chicago
New York
Washington
San Francisco
Los Angeles
Dallas
Houston
Mexico City
North Pacific Ocean

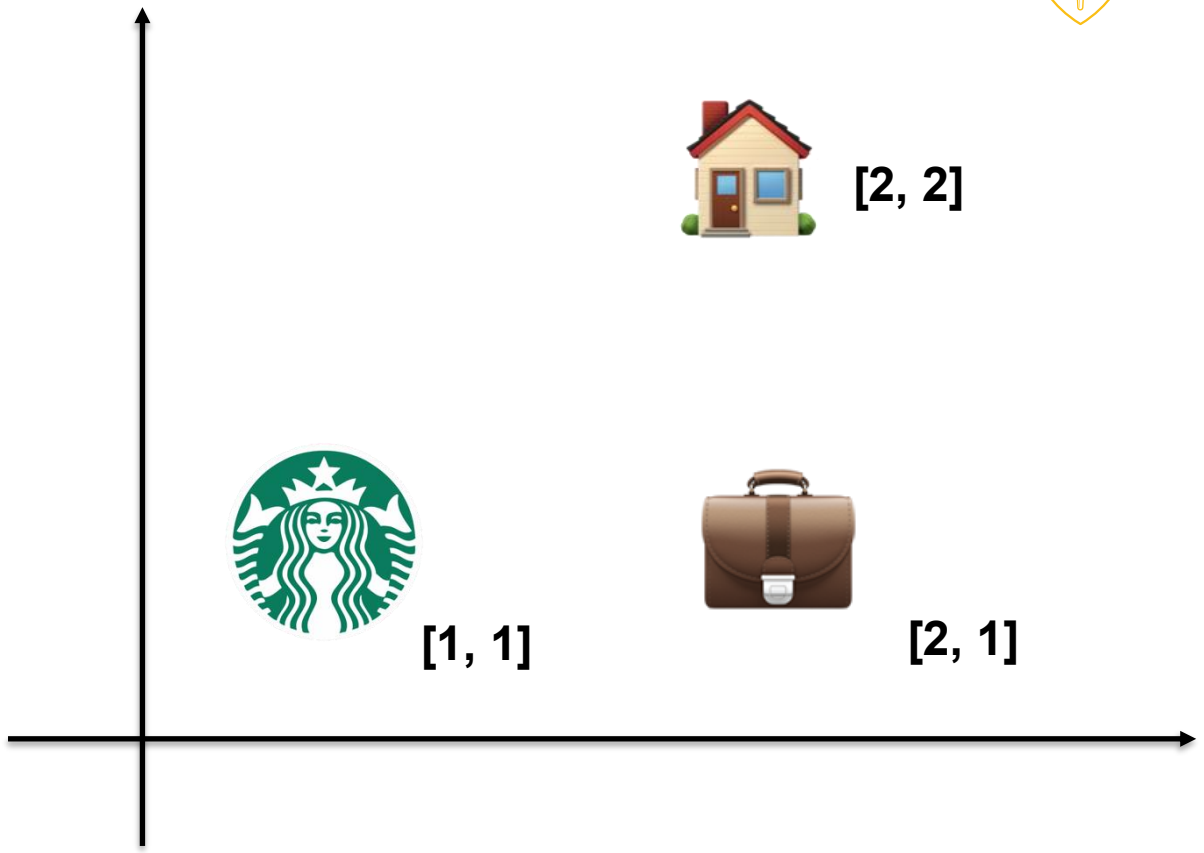
Point Representation

- Data is collected point by point.



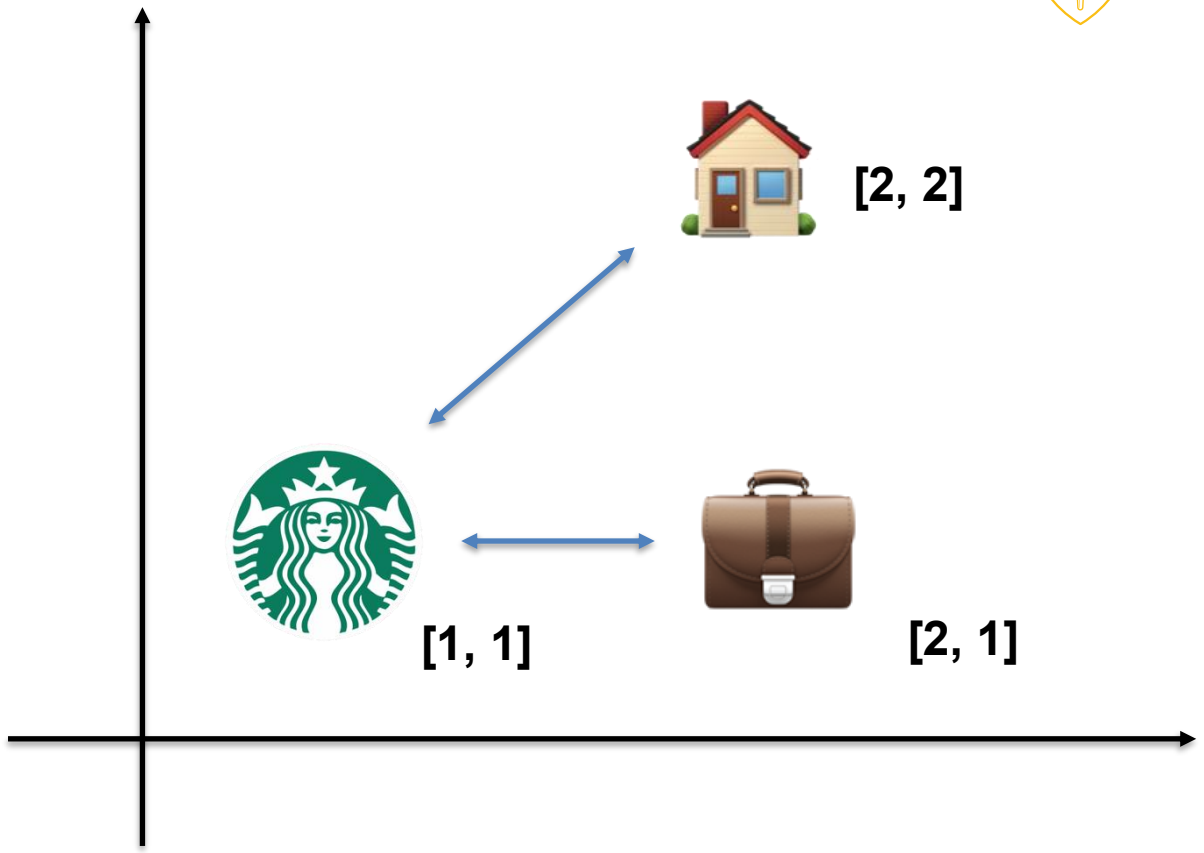
Why point encoding

- What do we need from location:
  - Absolute position
  - Relative position
  - Positional Relationship



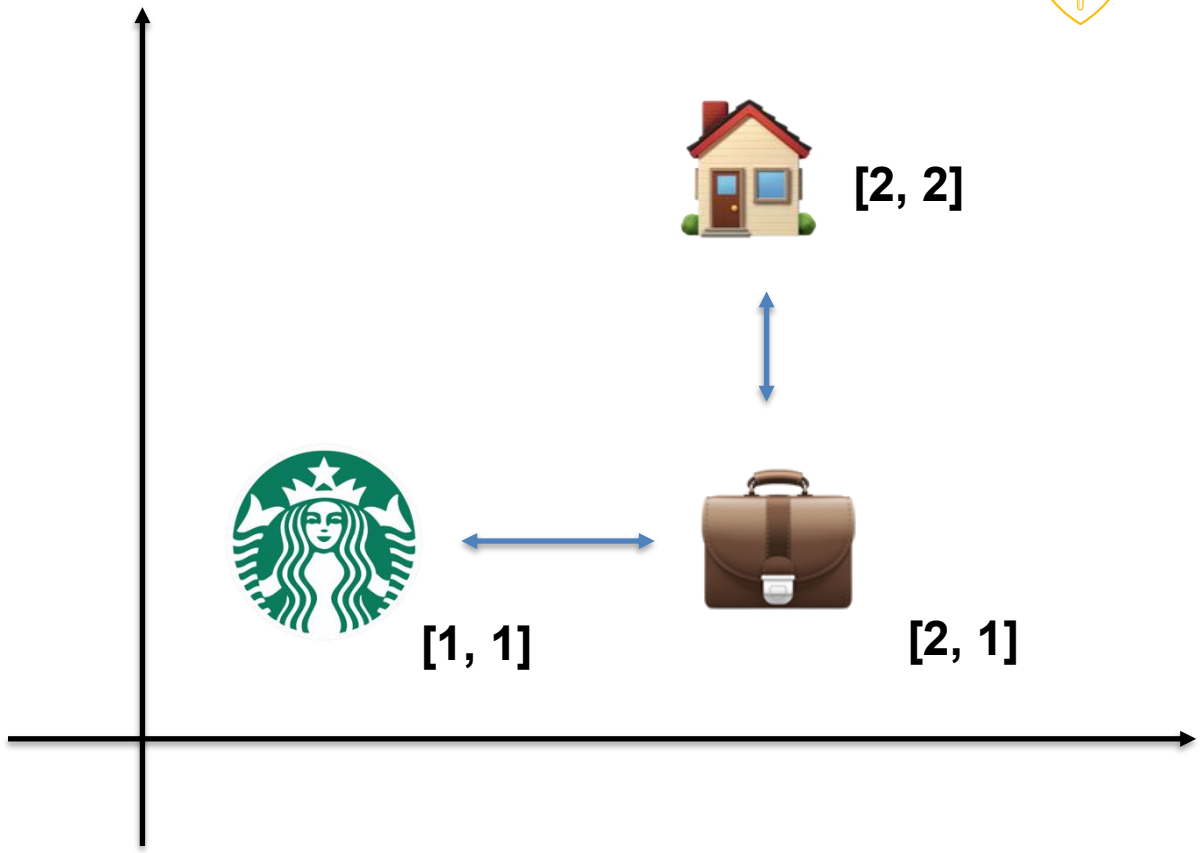
Why point encoding

- What do we need from location:
  - Absolute position
  - Relative position
  - Positional Relationship



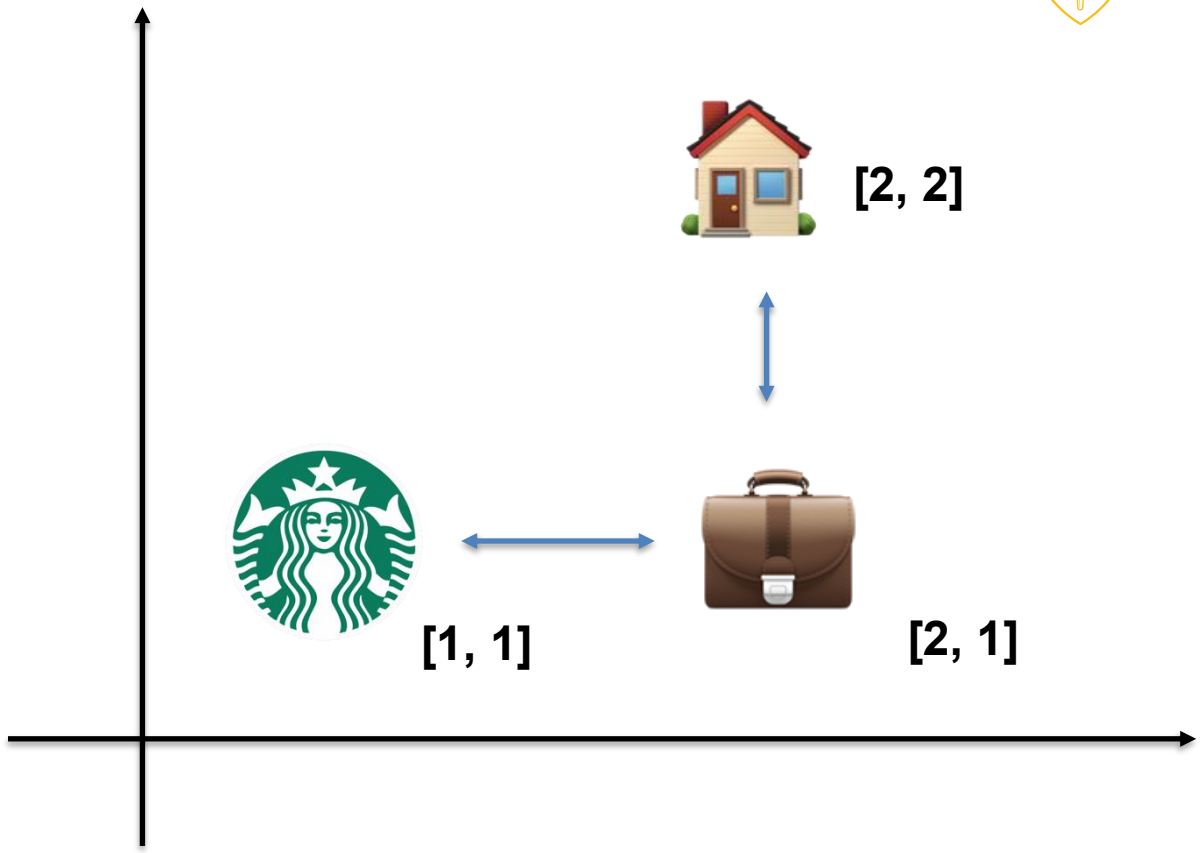
Why point encoding

- What do we need from location:
  - Absolute position
  - Relative position
  - Positional Relationship



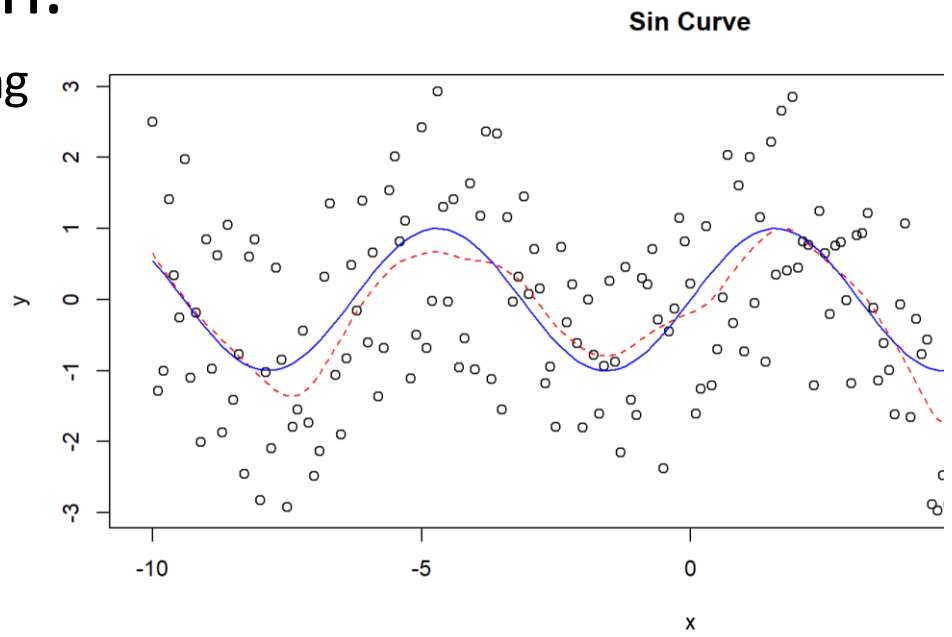
Why point encoding

- Why not (x, y)
- What do we need from location:
  - Absolute position
  - Relative position
  - Positional Relationship



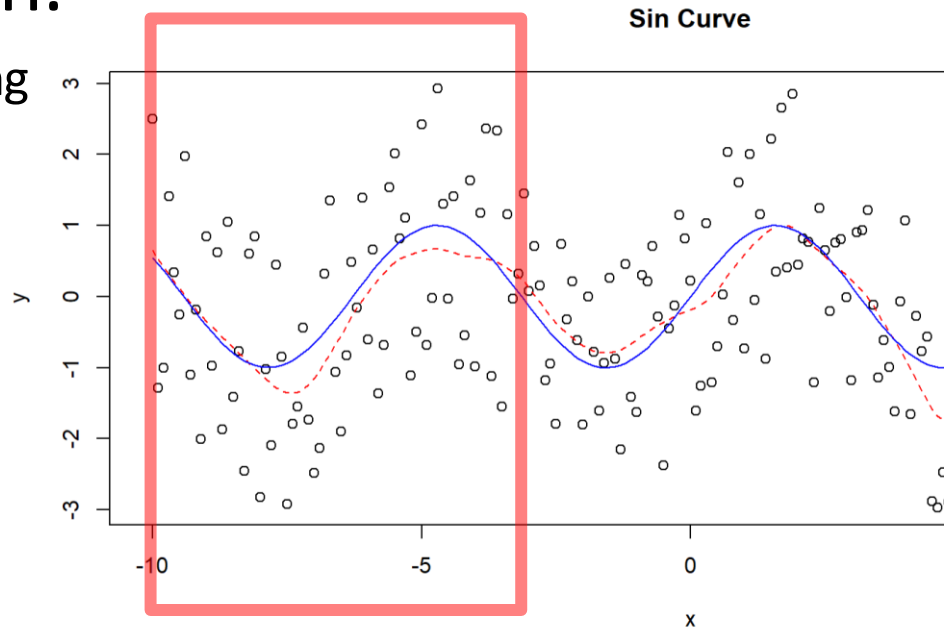
Why point encoding

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship



Why point encoding

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- Why not (x) (x , y)
  - No periodicity bias
  - No frequency awareness

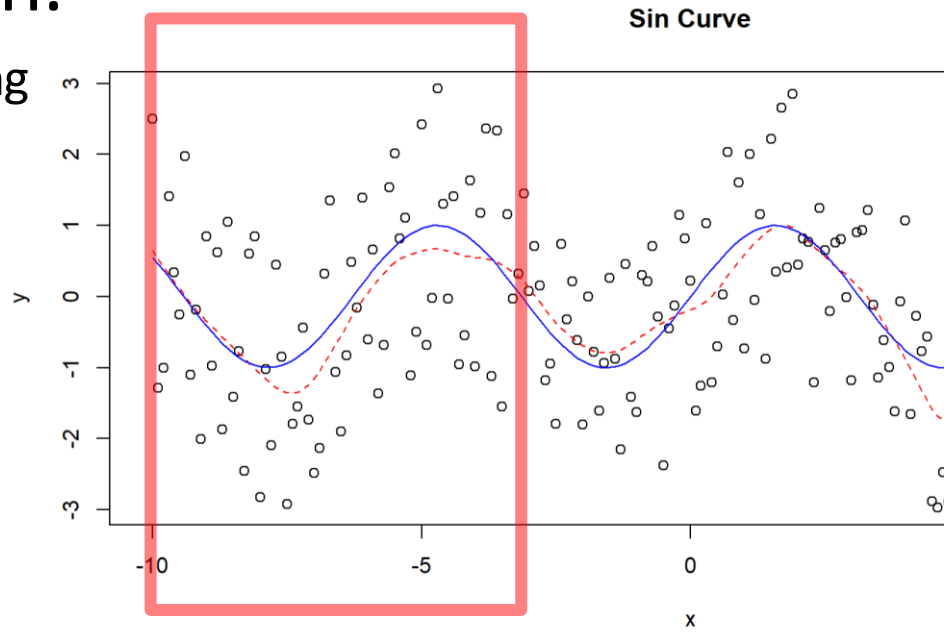


"this function should repeat" f(-10)->1 f(-5)->1 f(0)->?

"whether variation is slow or fast" f(-10)-f(-7.5)=2 f(0.12)-f(7.62)=?

Why point encoding

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- Why not (x , y)

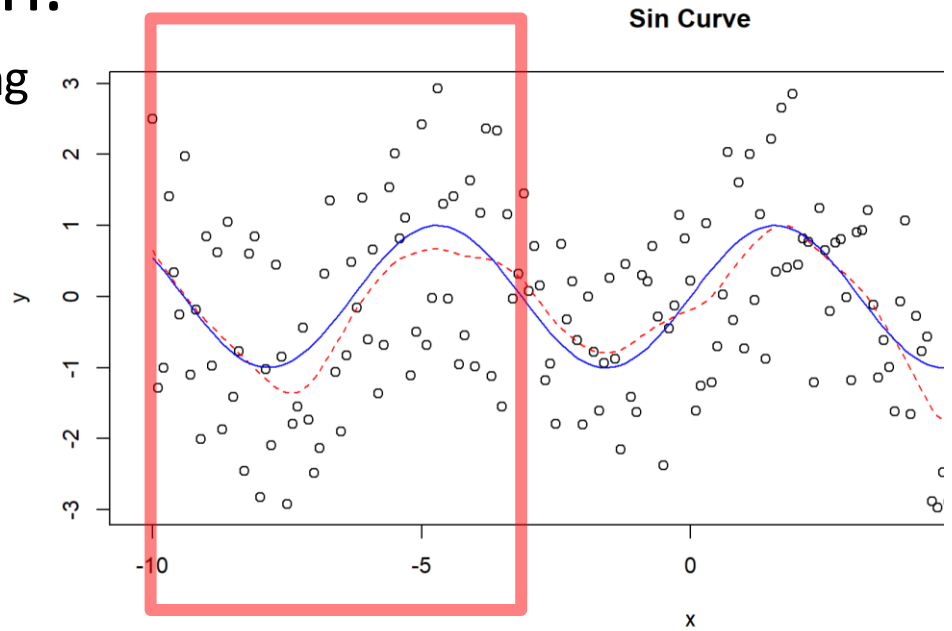


- Lack of inductive bias ⇒ High-variance model
  - *Memorize coordinates instead of learning generalizable patterns.*
- Poor generalization across scales/regions *e.g. NYC vs LA*

Why point encoding

Adding inductive bias

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- What would be good?



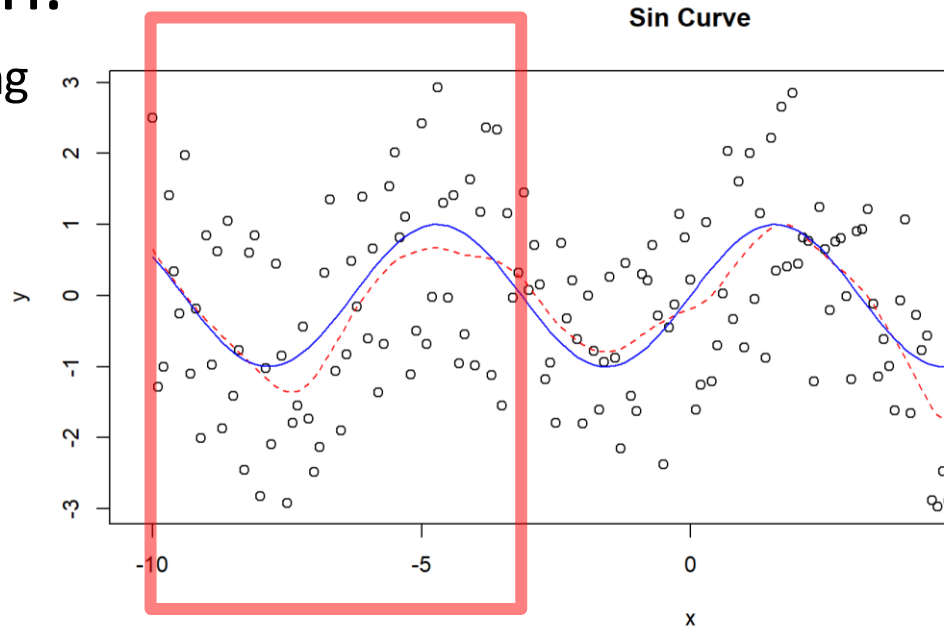
f(x) →f(sin(x)) "this function should repeat"

y = w sin(x) "whether variation is slow or fast"

Why point encoding

Adding inductive bias

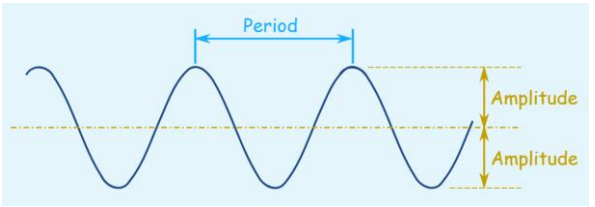
- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- What would be good?



f(x) →f(sin(x)) "this function should repeat"

y = w sin(**a**x) "whether variation is slow or fast"

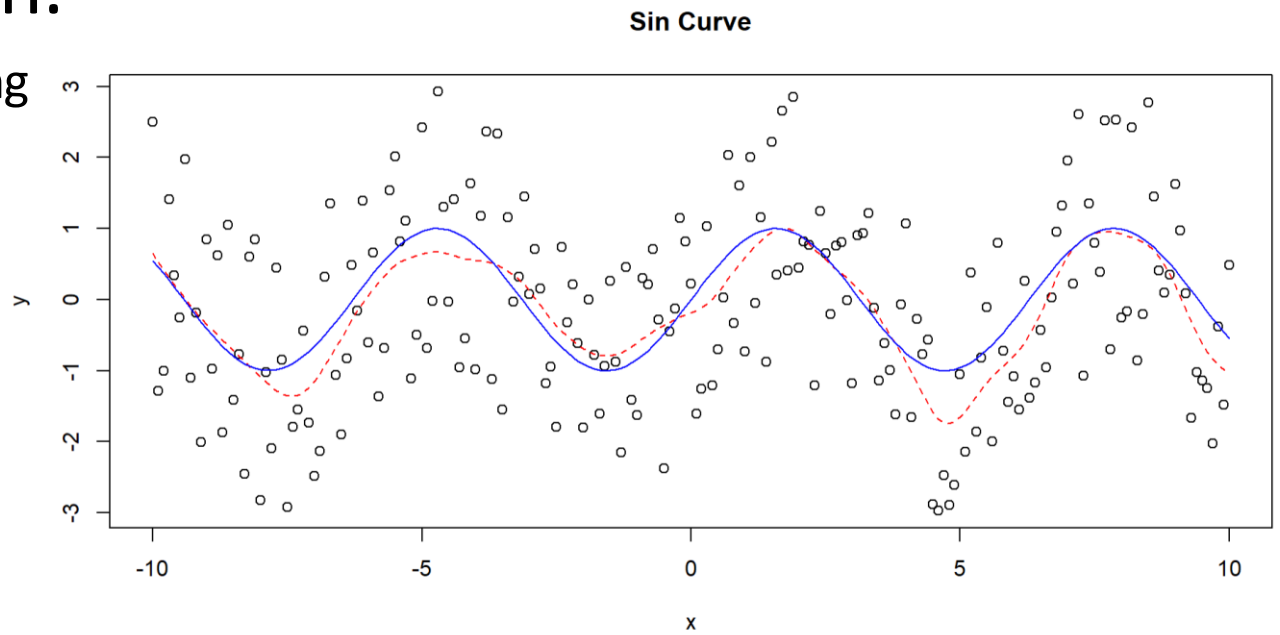
$$\text{Period} = \frac{2\pi}{a}$$



Why point encoding

Adding inductive bias

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- What would be good?



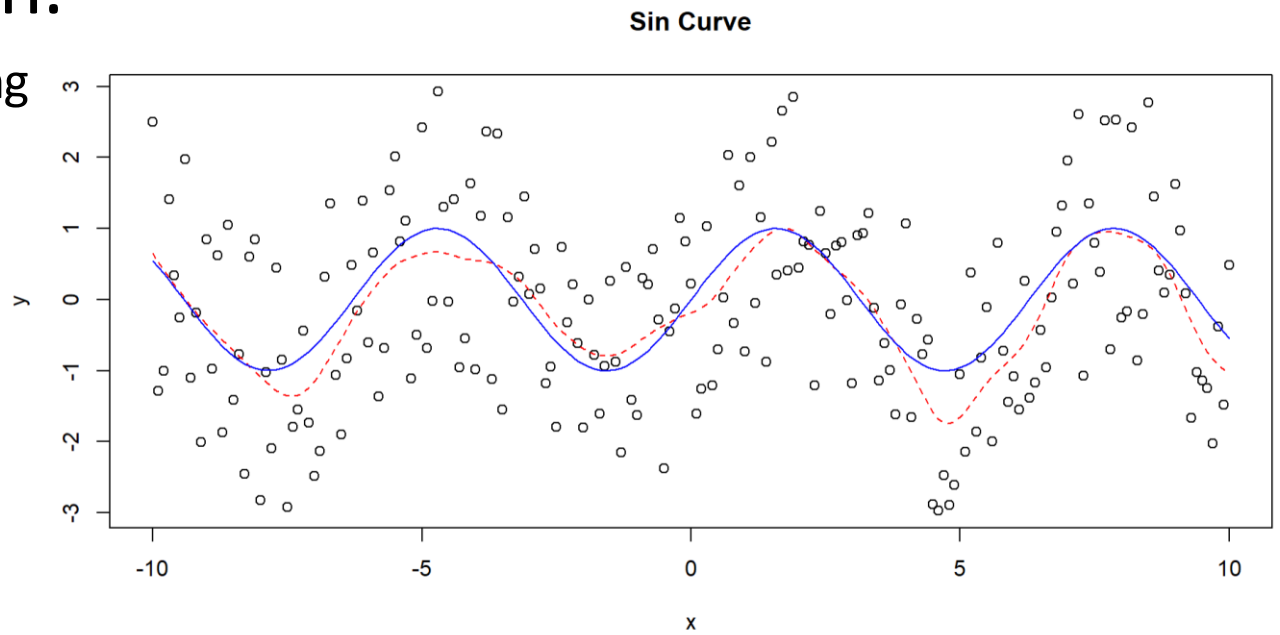
f(x) →f(sin(x)) "this function should repeat"

y = w sin(**a**x) "whether variation is slow or fast"

Why point encoding

Adding a good inductive bias

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- How do we know period **a**?



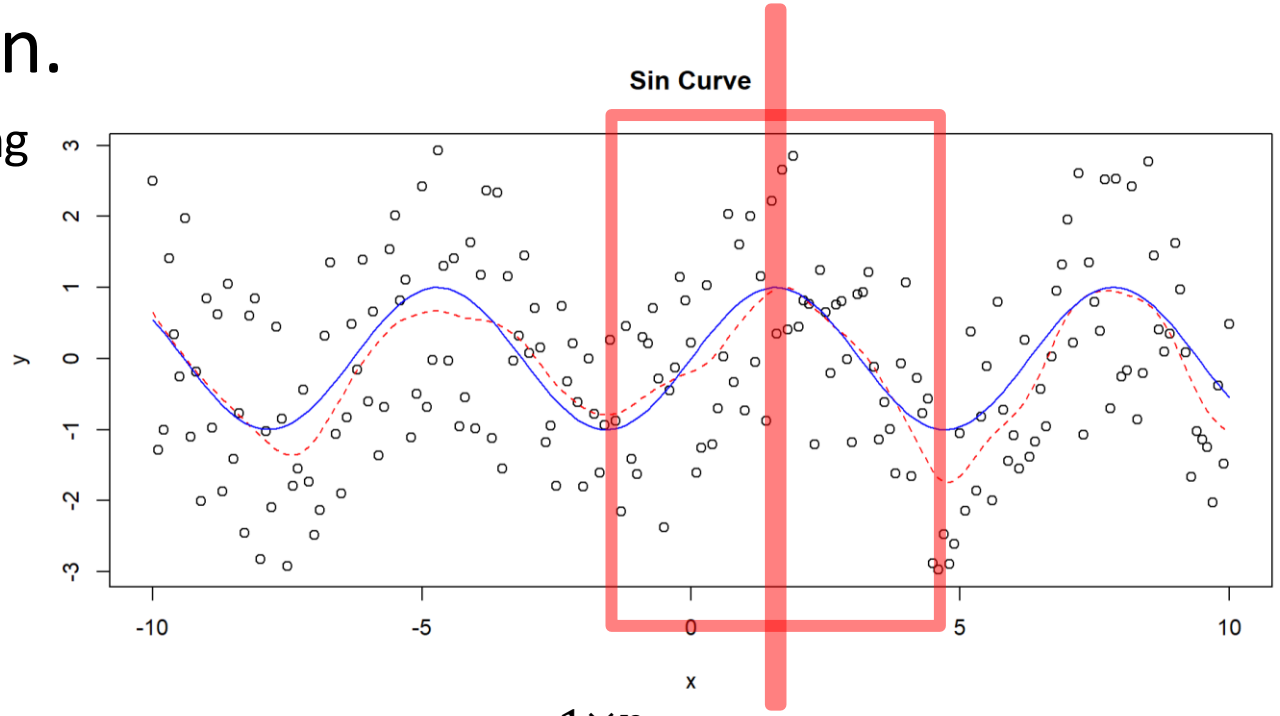
$$f(x) \rightarrow f(\sin(x))$$

$$y = w \sin(ax)$$

Why point encoding

Adding the prior that a geospatial phenomenon could have multiple variation patterns in different **scales**.

- Let's start from one dimension.
  - Reduce to transformer's positional encoding
  - Absolute position
  - Relative position
  - Positional Relationship
- How do we know period **a**?



$x \in R^{1\times n}$

f(x) →f(sin(x))

y = w sin(**a**x)

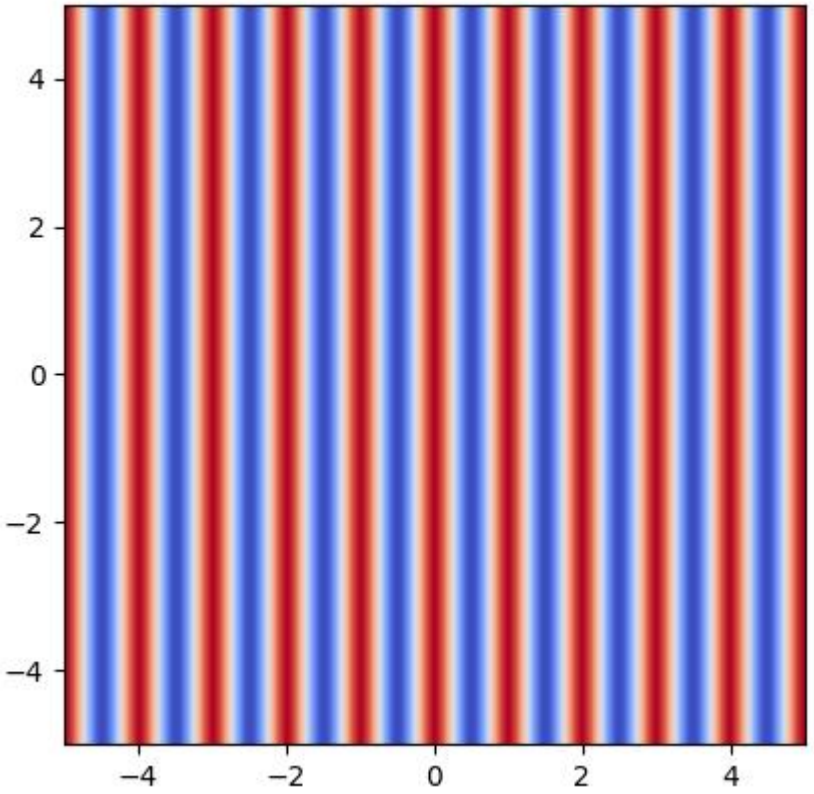
$x \rightarrow [\sin(a_1 x), \sin(a_2 x), \sin(a_3 x), \ldots, \sin(a_n x)]$

Fourier series theorem shows that a series of sine and cosine functions form a **complete basis for representing a wide class of functions.**

2D space

Encoding function: *f(x, y) = [cos(ω x), cos(ω y)]*

$$\text{where } \vec{x} = (x, y),\ \vec{a} = \text{unit vector}$$

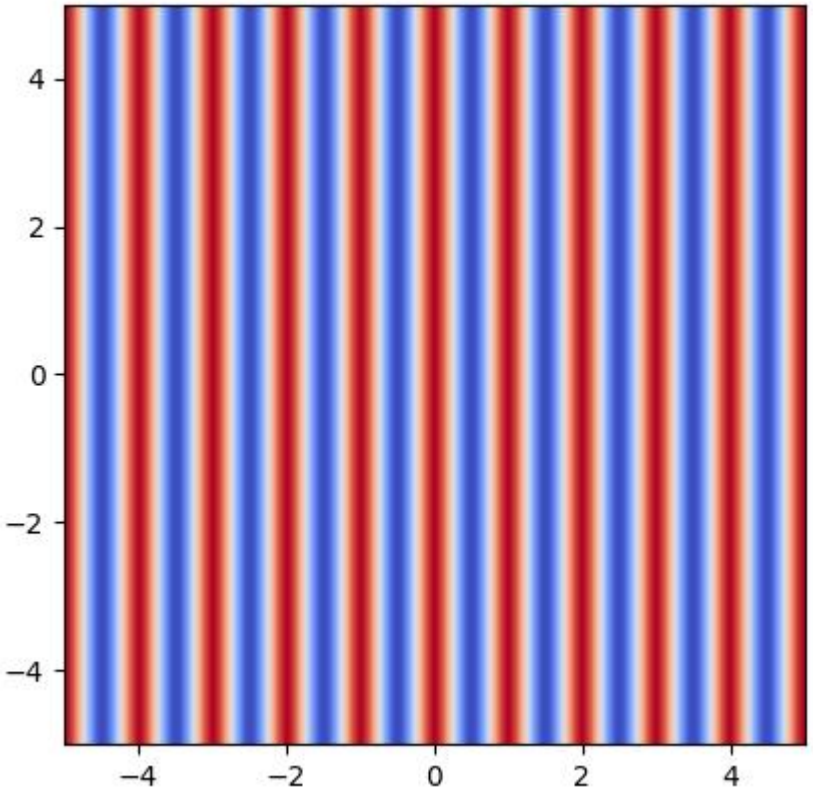


$\cos(\omega \cdot a_1 \cdot x)$

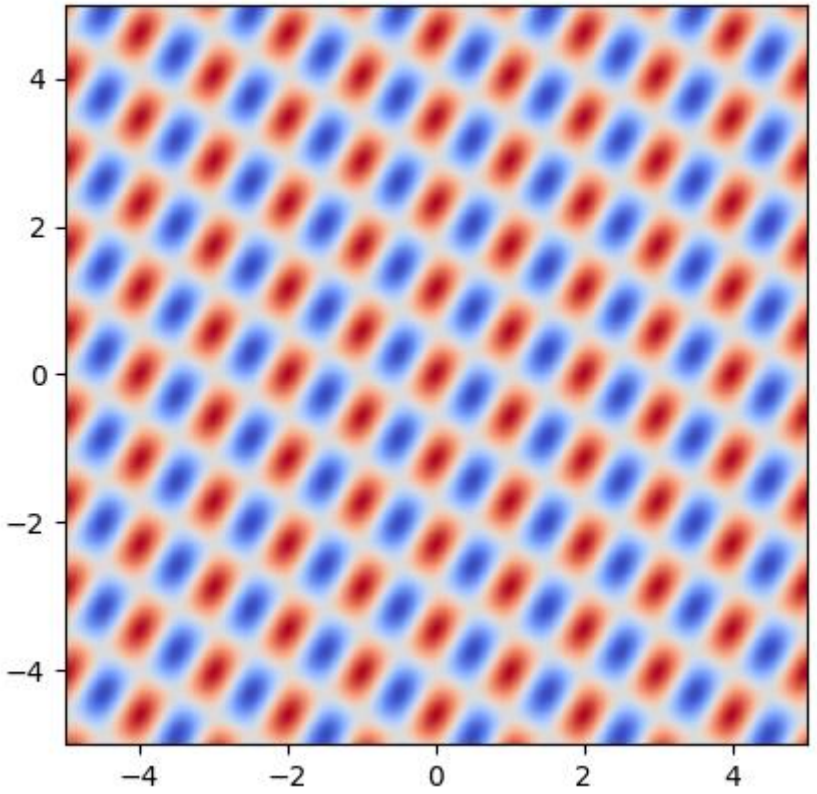
2D space

Encoding function: *f(x, y) = [cos(ω x), cos(ω y)]*

$$\text{where } \vec{x} = (x, y),\ \vec{a} = \text{unit vector}$$



$\cos(\omega \cdot a_1 \cdot x)$



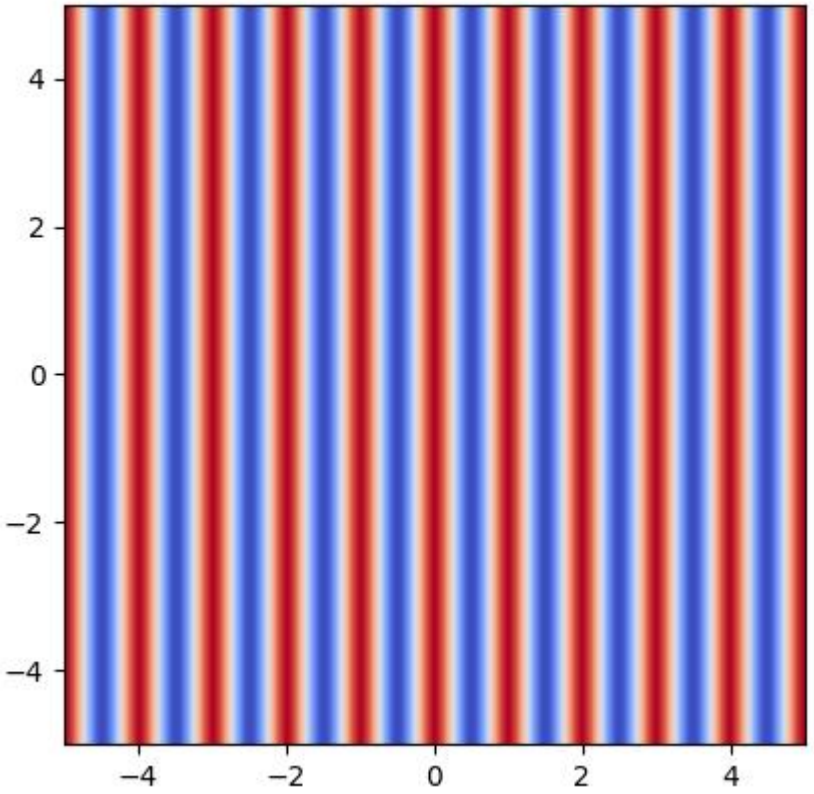
$\cos(\omega \cdot a_1 \cdot x) + \cos(\omega \cdot a_2 \cdot x)$

2D space

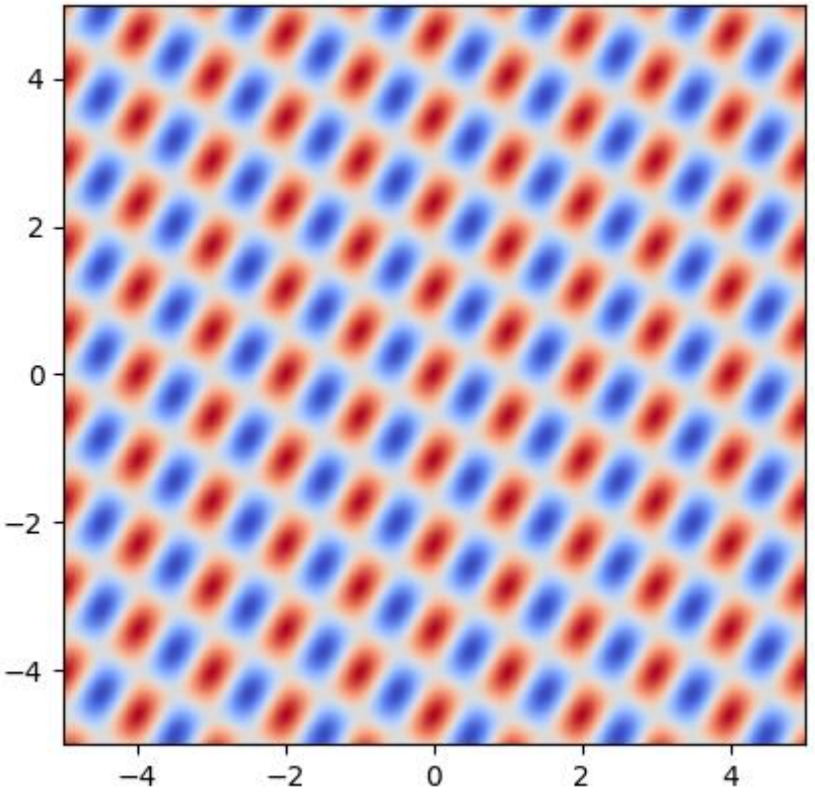
Adding the prior that a geospatial phenomenon could have multiple variation patterns in different **directions**.

Encoding function: *f(x, y) = [cos(ω x), cos(ω y)]*

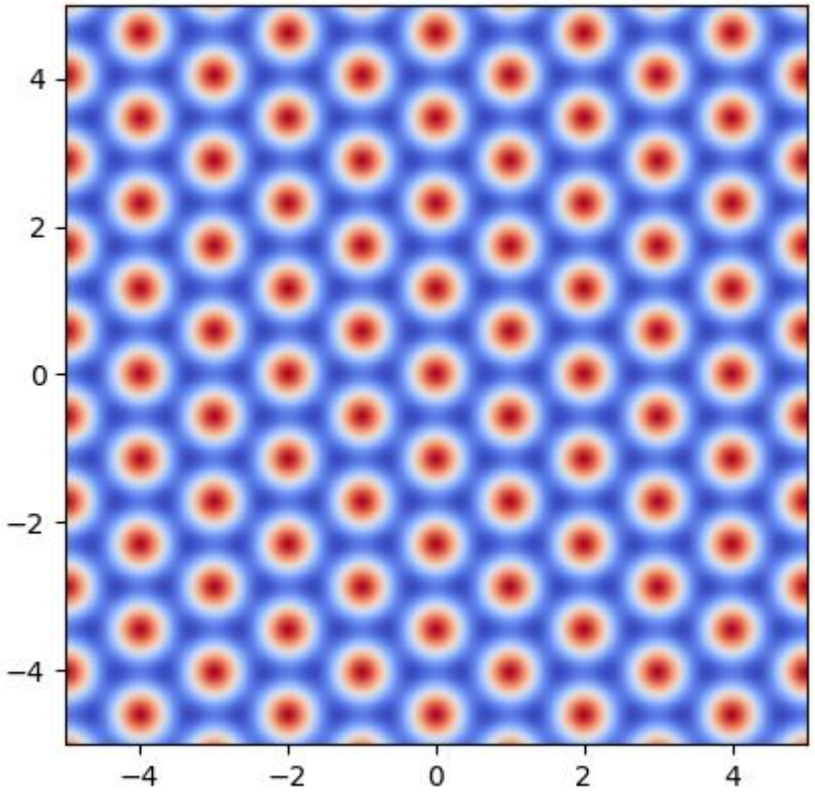
$$\text{where } \vec{x} = (x, y),\ \vec{a} = \text{unit vector}$$



$\cos(\omega \cdot a_1 \cdot x)$



$\cos(\omega \cdot a_1 \cdot x) + \cos(\omega \cdot a_2 \cdot x)$



$\cos(\omega \cdot a_1 \cdot x) + \cos(\omega \cdot a_2 \cdot x) + \cos(\omega \cdot a_3 \cdot x)$

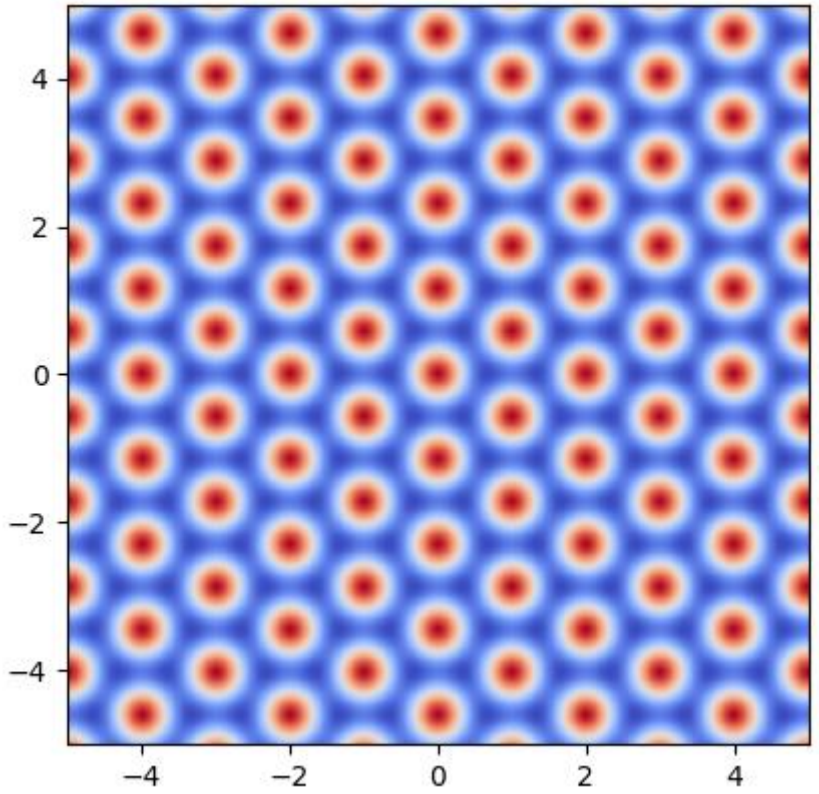
2D space

Encoding function: *f(x, y) = [cos(ω x), cos(ω y)]*

$$\text{where } \vec{x} = (x, y),\ \vec{a} = \text{unit vector}$$

$$PE^{(t)}(\mathbf{x}) = [PE_0^{(t)}(\mathbf{x}); ...; PE_s^{(t)}(\mathbf{x}); ...; PE_{S-1}^{(t)}(\mathbf{x})]$$

$$PE_{s,j}^{(t)}(\mathbf{x}) = [\cos(\frac{\langle \mathbf{x}, \mathbf{a}_j \rangle}{\lambda_{min} \cdot g^{s/(S-1)}}); \sin(\frac{\langle \mathbf{x}, \mathbf{a}_j \rangle}{\lambda_{min} \cdot g^{s/(S-1)}})] \forall j = 1, 2, 3;$$



2D space

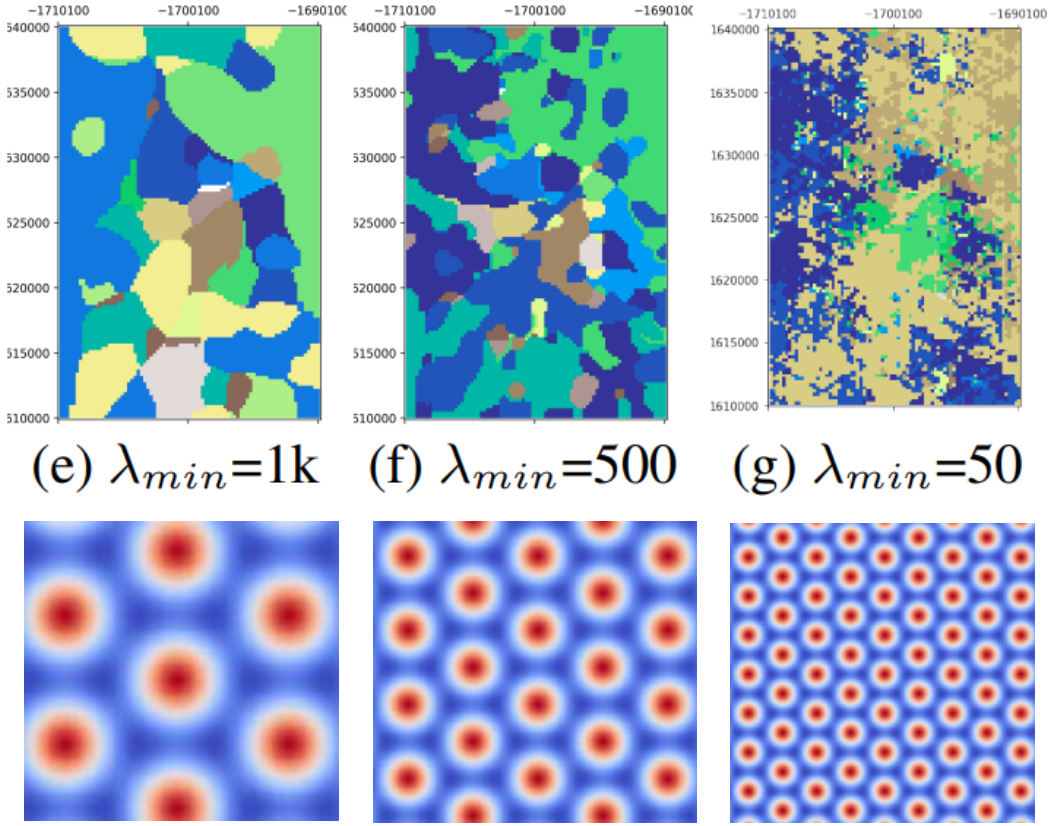
Adding the prior that a geospatial phenomenon could have multiple variation patterns in different **scales**.

Encoding function: *f(x, y) = [cos(ω x), cos(ω y)]*

$$\text{where } \vec{x} = (x, y),\ \vec{a} = \text{unit vector}$$

$$PE^{(t)}(\mathbf{x}) = [PE_0^{(t)}(\mathbf{x}); ...; PE_s^{(t)}(\mathbf{x}); ...; PE_{S-1}^{(t)}(\mathbf{x})]$$

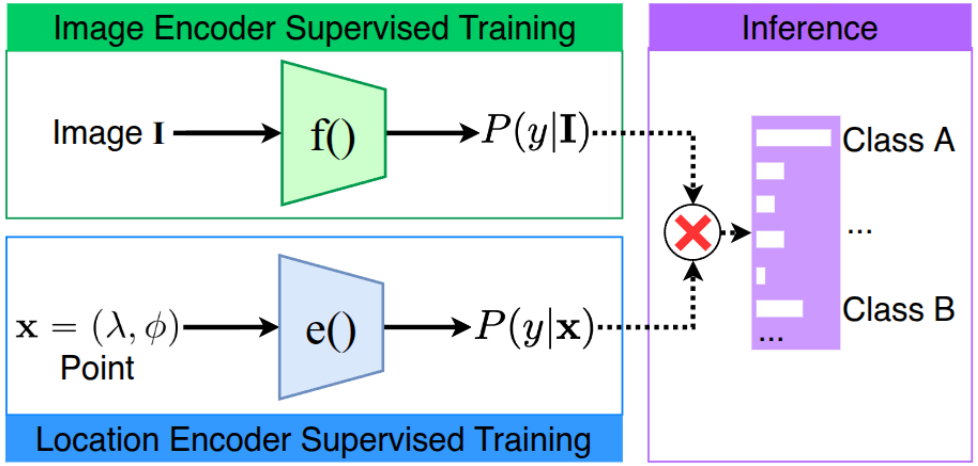
$$PE_{s,j}^{(t)}(\mathbf{x}) = [\cos(\frac{\langle \mathbf{x}, \mathbf{a}_j \rangle}{\lambda_{min} \cdot g^{s/(S-1)}}); \sin(\frac{\langle \mathbf{x}, \mathbf{a}_j \rangle}{\lambda_{min} \cdot g^{s/(S-1)}})] \forall j = 1, 2, 3;$$

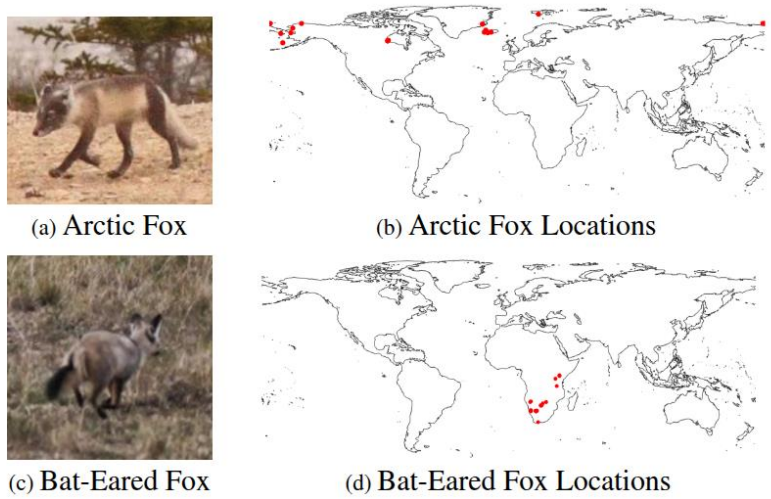


Multi-scale representation learning for spatial feature distributions using **grid cells**, ICLR 2020.

Results

Space2Vec: Location Aware Image Classification





(a) Arctic Fox

(b) Arctic Fox Locations

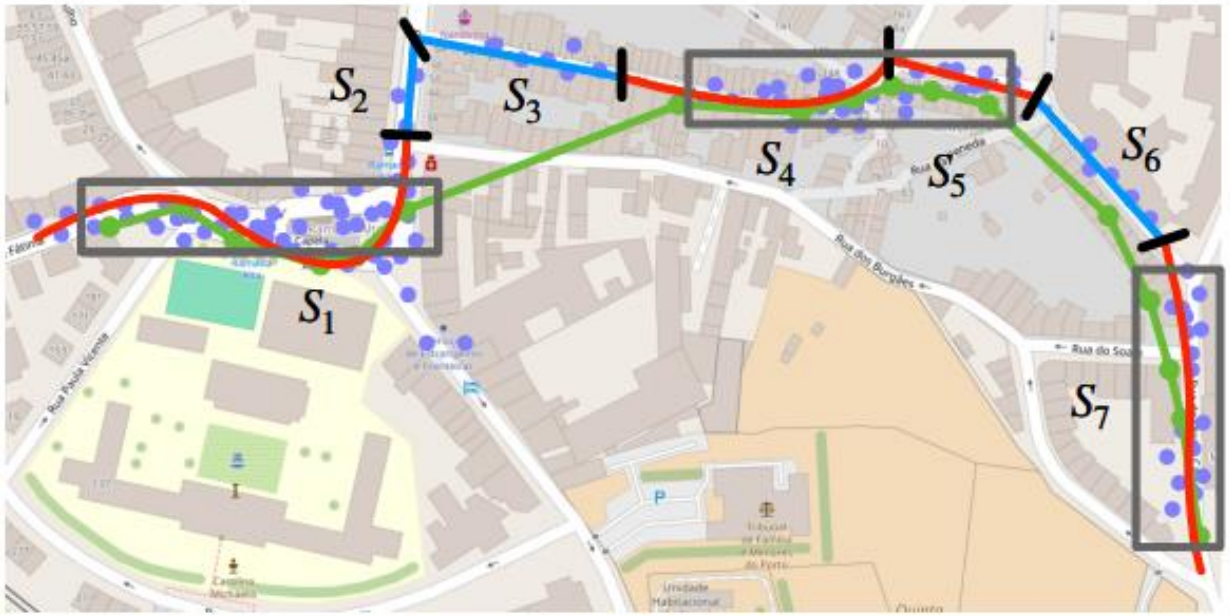
(c) Bat-Eared Fox

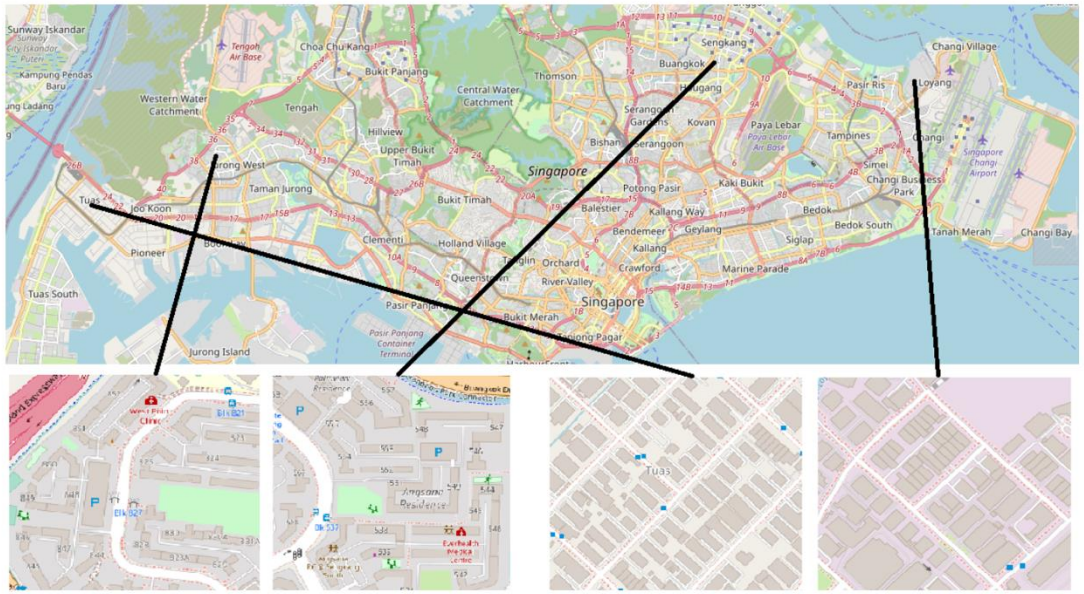
(d) Bat-Eared Fox Locations

	BirdSnap†	NABirds†
No Prior (i.e. uniform)	70.07	76.08
Nearest Neighbor (num)	77.76	79.99
Nearest Neighbor (spatial)	77.98	80.79
Adaptive Kernel (Berg et al., 2014)	78.65	81.11
tile (Tang et al., 2015) (location only)	77.19	79.58
wrap (Mac Aodha et al., 2019) (location only)	78.65	81.15
rbf (σ=1k)	78.56	81.13
grid (λ_{min}=0.0001, λ_{max}=360, $S = 64$)	**79.44**	81.28
theory (λ_{min}=0.0001, λ_{max}=360, $S = 64$)	79.35	**81.59**

What's next

- We need a unified representation of different types of geospatial data.



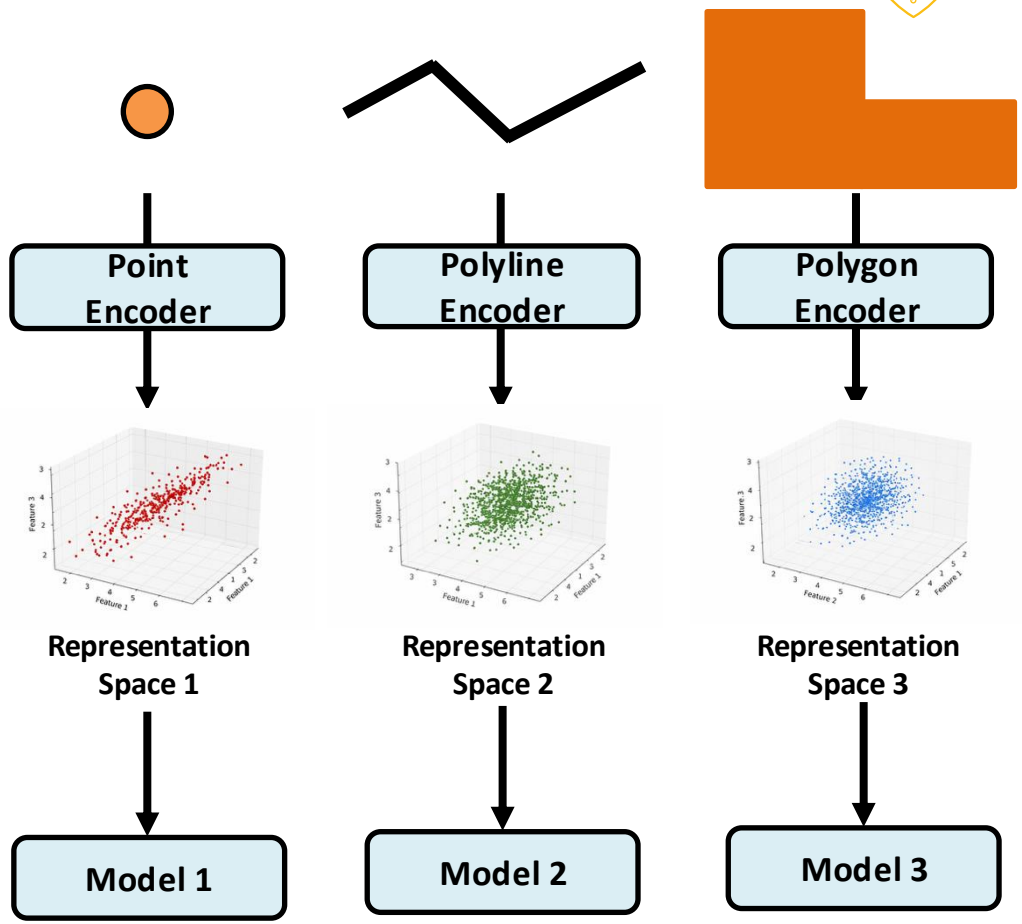


Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Existing spatial entity representation methods are largely influenced by node–edge data storage paradigms. Sequential models are used to learn line-like entities, while node-edge models are applied to model polygon entities.

```
a = LineString([[0, 0], [1, 0], [1, 1]])

coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```

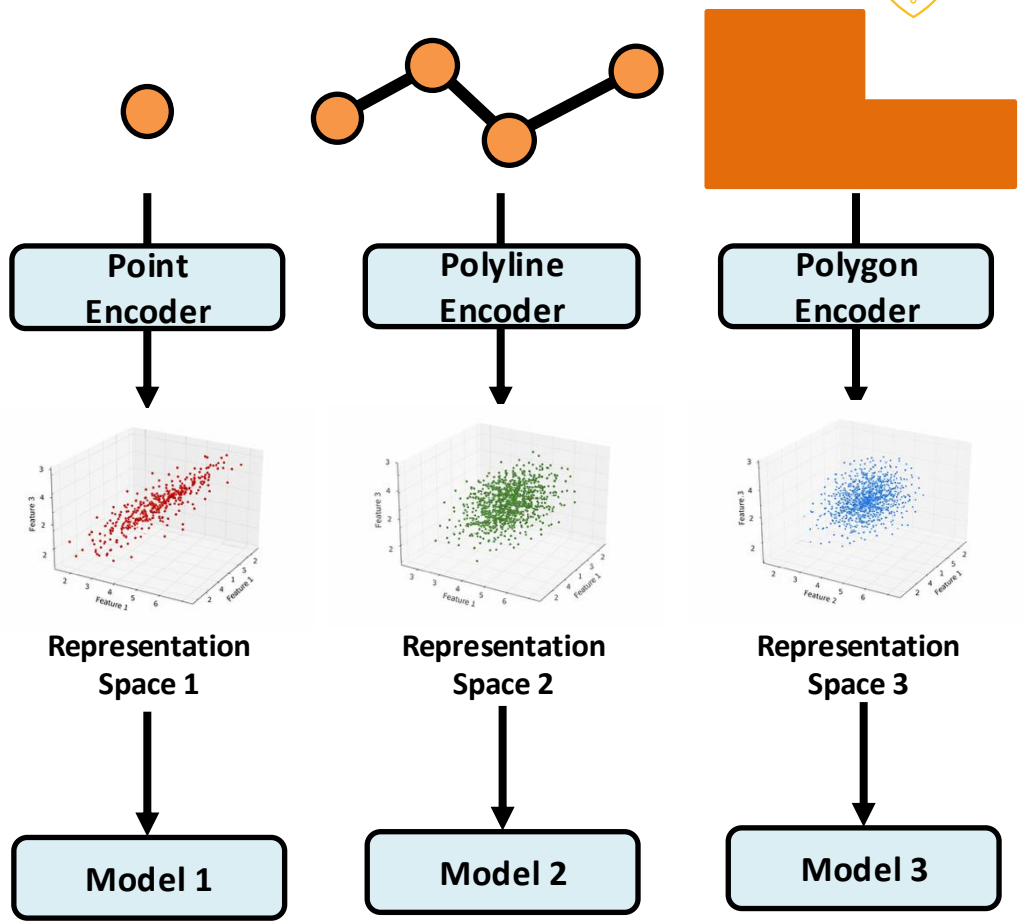


Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Existing spatial entity representation methods are largely influenced by node–edge data storage paradigms. Sequential models are used to learn line-like entities, while node-edge models are applied to model polygon entities.

```
a = LineString([[0, 0], [1, 0], [1, 1]])
```

```
coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```

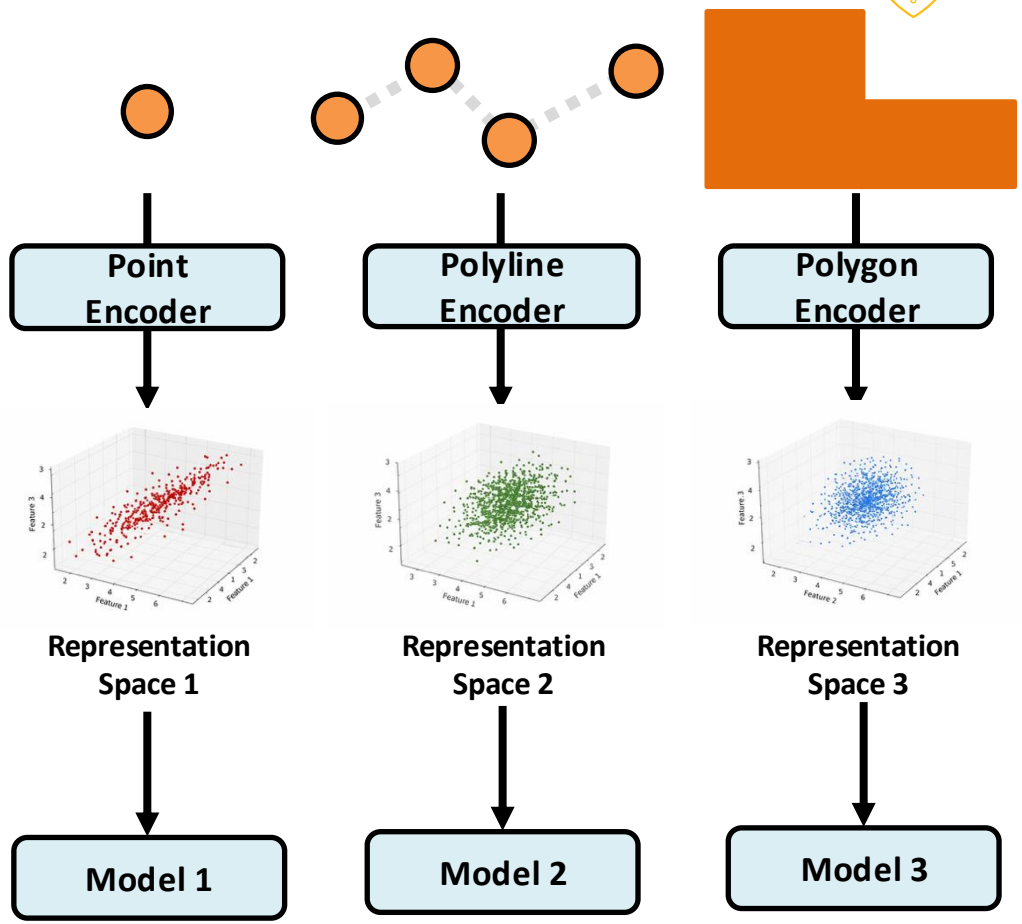


Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Existing spatial entity representation methods are largely influenced by node–edge data storage paradigms. Sequential models are used to learn line-like entities, while node-edge models are applied to model polygon entities.

```
a = LineString([[0, 0], [1, 0], [1, 1]])

coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```

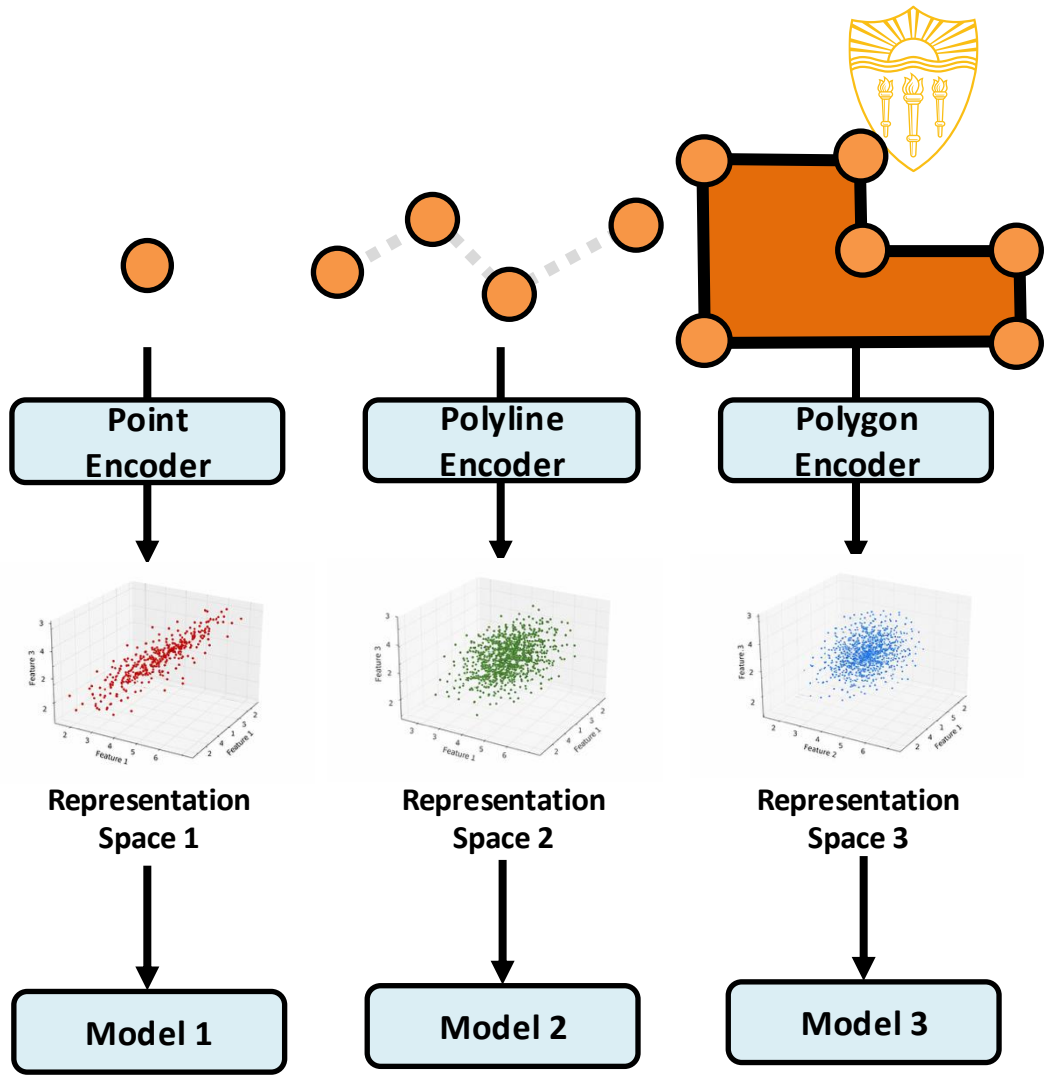


Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Existing spatial entity representation methods are largely influenced by node–edge data storage paradigms. Sequential models are used to learn line-like entities, while node-edge models are applied to model polygon entities.

```
a = LineString([[0, 0], [1, 0], [1, 1]])

coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```

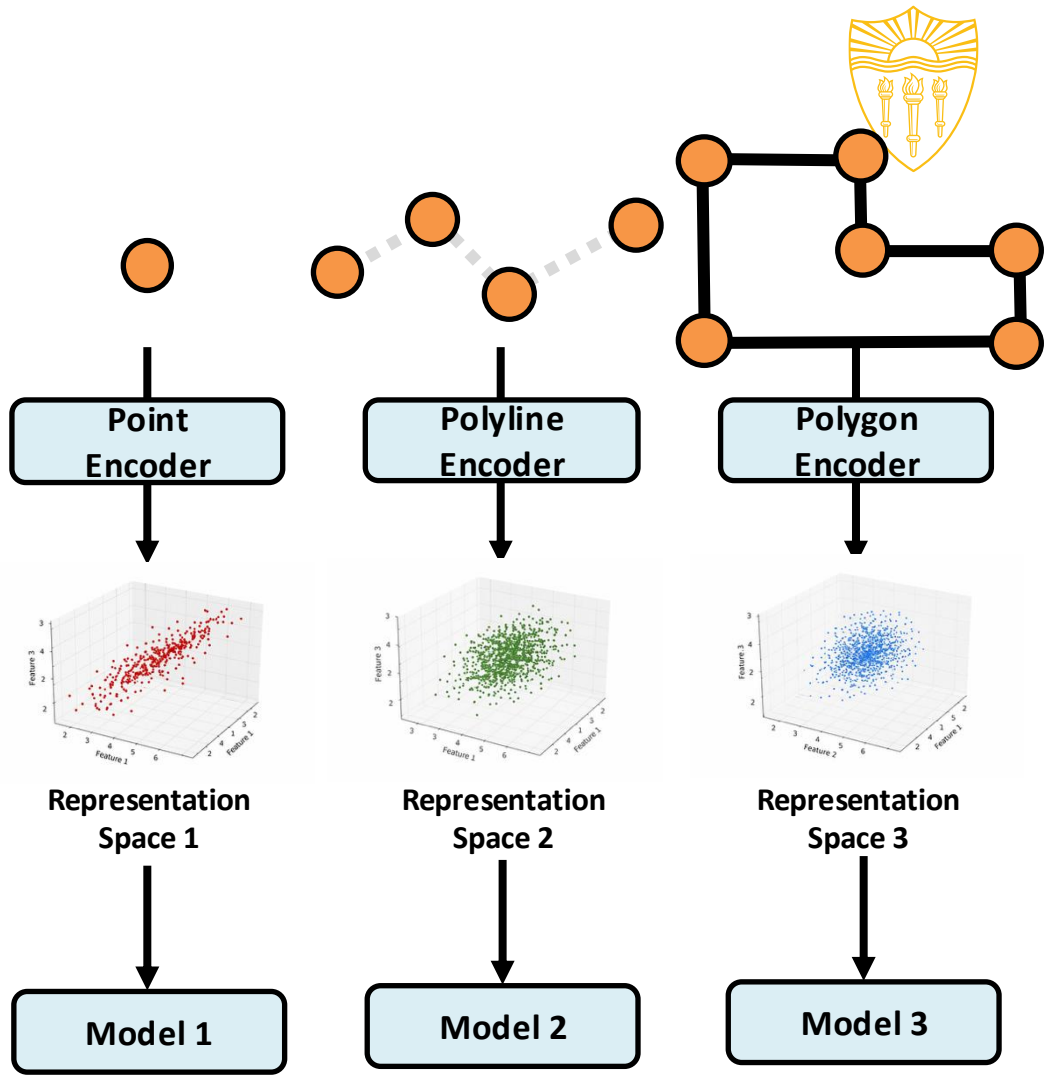


Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Existing spatial entity representation methods are largely influenced by node–edge data storage paradigms. Sequential models are used to learn line-like entities, while node-edge models are applied to model polygon entities.

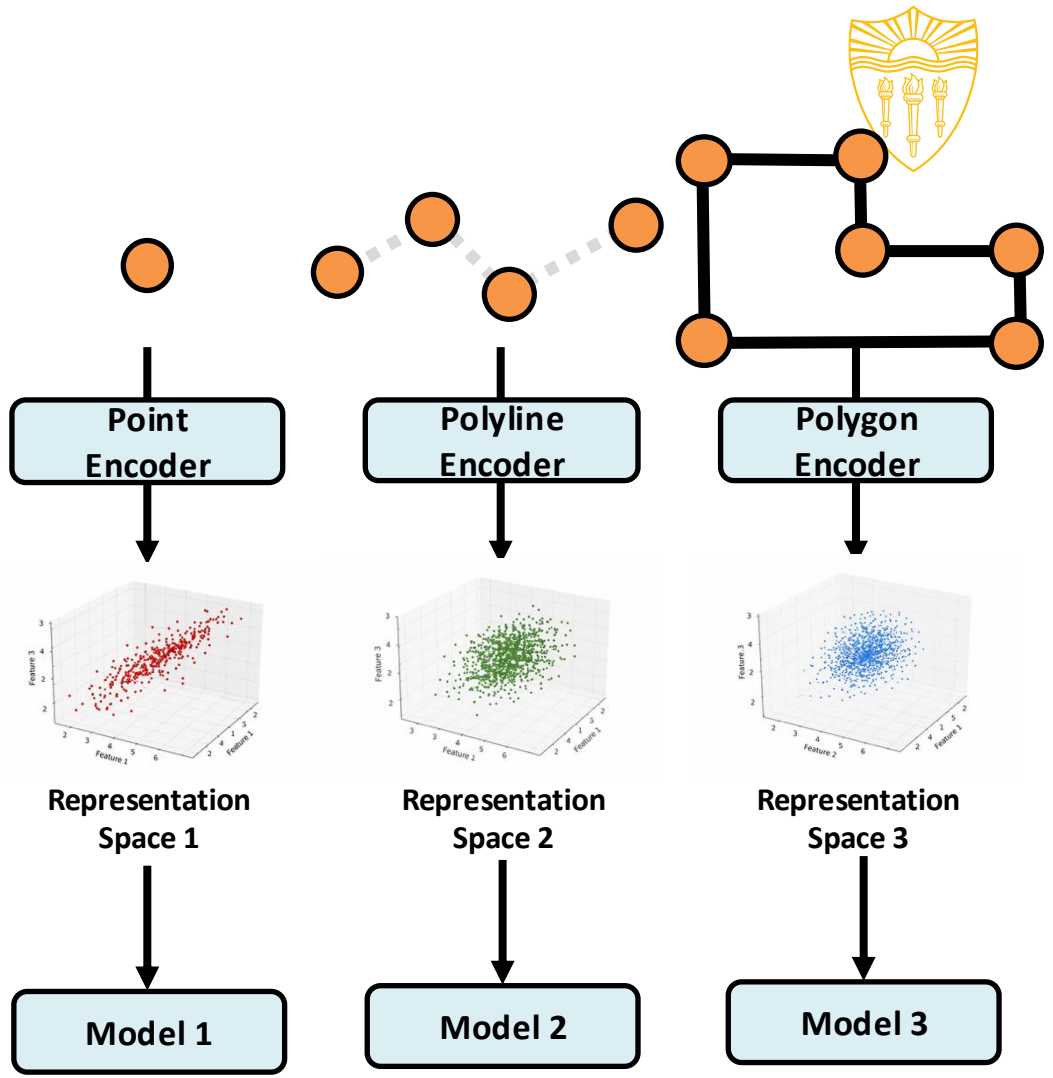
```
a = LineString([[0, 0], [1, 0], [1, 1]])

coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```



Introduction

- Geospatial entities include various data structures.
  Point → POI, Location.
  Polyline → Road, Trajectory.
  Polygon → Building Footprint, Boundary.
- **Representation Learning** of geospatial entities is the initial step for GeoAI tasks.
- Limitation: Discrete data structures are not well-suited for geospatial entity representation.



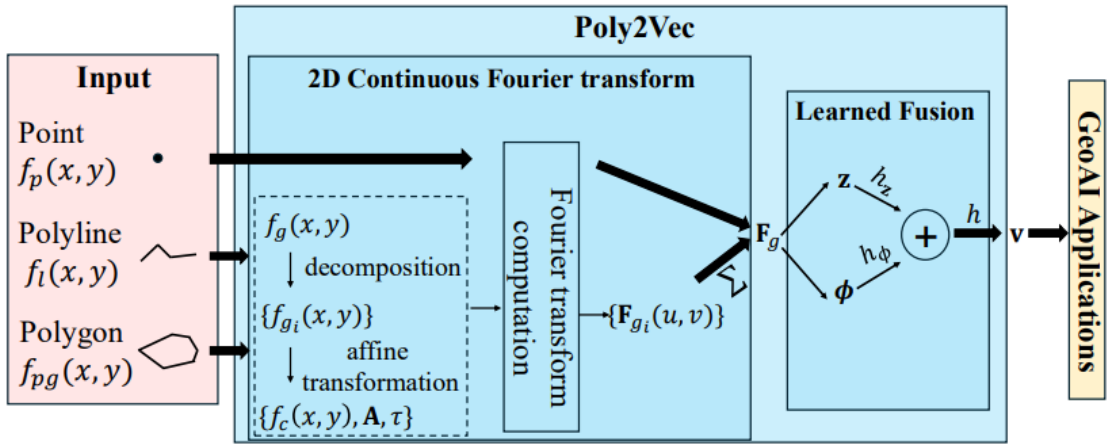

```
a = LineString([[0, 0], [1, 0], [1, 1]])

coords = ((0., 0.), (0., 1.), (1., 1.), (1., 0.), (0., 0.))
polygon = Polygon(coords)
```

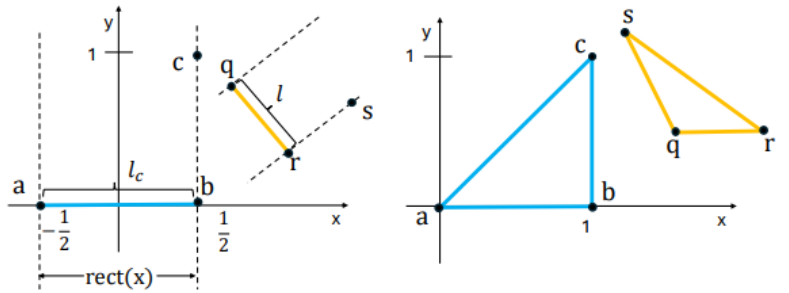
Poly2Vec

- Poly2Vec:
  - Decompose geo-entities to triangles.
  - Define a uniform line/triangle, then define the **transform function** from the uniform triangle to triangles.
  - Use Fourier transform to representation each **function**. Fourier transform is a linear operation, and addable.

$$\widehat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi\xi x}\, dx, \quad \forall \xi \in \mathbb{R}. \qquad \mathcal{F}\{x(t) + y(t)\} = \mathcal{F}\{x(t)\} + \mathcal{F}\{y(t)\} = X(\omega) + Y(\omega)$$

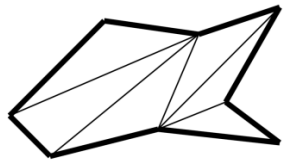


(a) The workflow of POLY2VEC.



(i) Affine transform from the arbitrary line segment l to the canonical line segment l_c.

(ii) Affine transform from the triangle $\triangle$ with vertices **qrs** to the canonical triangle $\vartriangleleft$ with vertices **abc**.



(b) Affine transform arbitrary geometry to its corresponding canonical geometry.

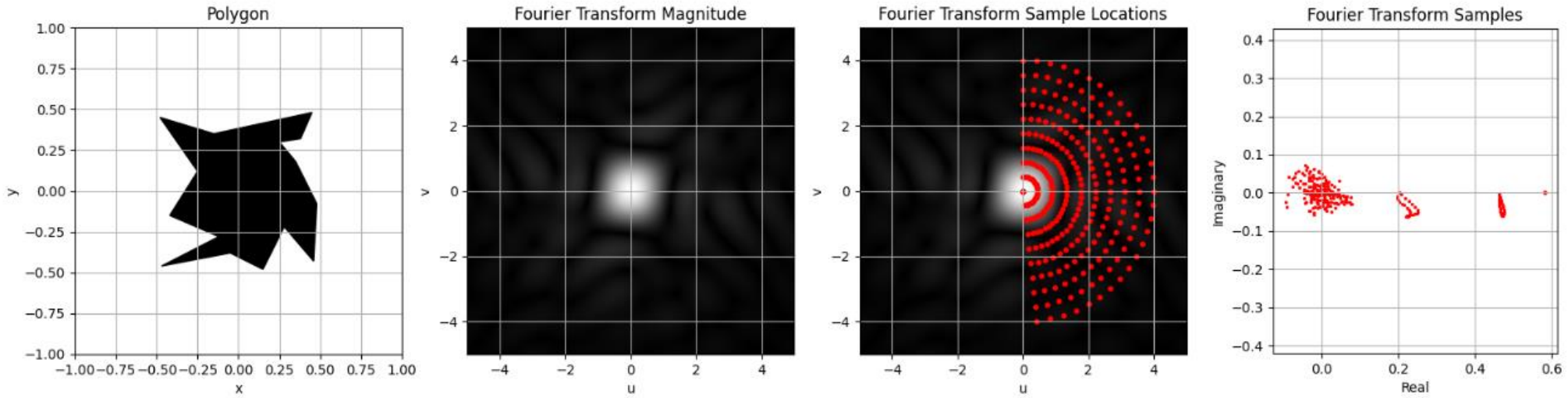
Figure 2: Overview of POLY2VEC.

Maria Despoina Siampou, Jialiang Li, John Krumm, Cyrus Shahabi, and Hua Lu. Poly2Vec: Polymorphic Fourier-Based Encoding of Geospatial Objectsfor GeoAI Applications. ICML 2025.

Poly2Vec

- Poly2Vec:
  - Decompose geo-entities to triangles.
  - Define a uniform line/triangle, then define the **transform function** from the uniform triangle to triangles.
  - Use Fourier transform to representation each **function**. Fourier transform is a linear operation, and addable.

$$\widehat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi\xi x}\, dx, \quad \forall \xi \in \mathbb{R}. \qquad \mathcal{F}\{x(t) + y(t)\} = \mathcal{F}\{x(t)\} + \mathcal{F}\{y(t)\} = X(\omega) + Y(\omega)$$



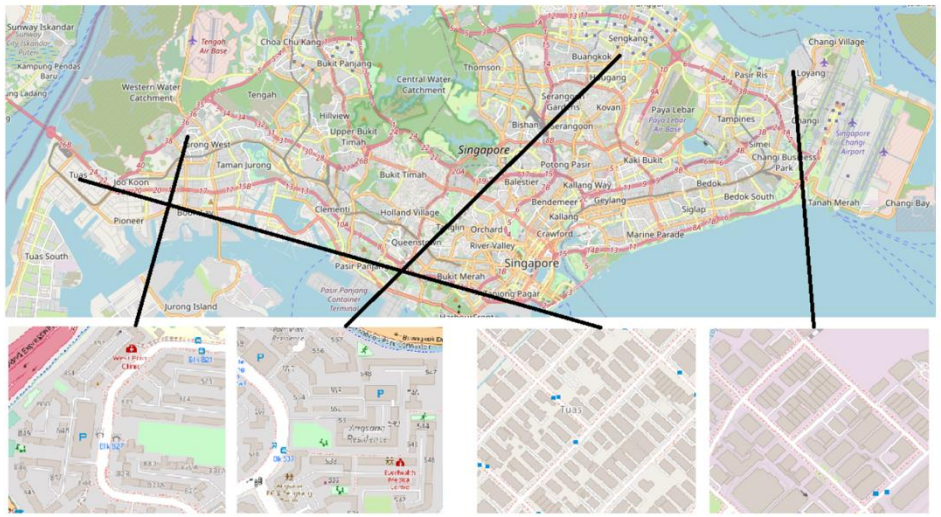
Maria Despoina Siampou, Jialiang Li, John Krumm, Cyrus Shahabi, and Hua Lu. Poly2Vec: Polymorphic Fourier-Based Encoding of Geospatial Objectsfor GeoAI Applications. ICML 2025.

Poly2Vec

- Poly2Vec:
  - Decompose geo-entities to triangles.
  - Define a uniform line/triangle, then define the **transform function** from the uniform triangle to triangles.
  - Use Fourier transform to representation each **function**. Fourier transform is a linear operation, and addable.

Table 4: Comparison of methods for Land Use Classification and Population Prediction. **Best** values are highlighted.

Land Use Classification						
Methods	**Singapore**			**New York**		
	L1 ↓	KL ↓	Cosine ↑	L1 ↓	KL ↓	Cosine ↑
RegionDCL	0.498 ± 0.038	0.294 ± 0.047	0.879 ± 0.021	0.418 ± 0.012	0.229 ± 0.013	0.912 ± 0.006
RegionDCL w/o distance-bias	0.558 ± 0.043	0.369 ± 0.067	0.844 ± 0.023	0.439 ± 0.012	0.244 ± 0.012	0.904 ± 0.005
RegionDCL w/ Poly2Vec	**0.484** ± 0.021	**0.278** ± 0.025	**0.881** ± 0.012	**0.397** ± 0.010	**0.212** ± 0.011	**0.923** ± 0.007
Population Prediction						
Methods	**Singapore**			**New York**		
	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑
RegionDCL	5807.54 ± 522.74	7942.74 ± 779.44	0.427 ± 0.108	5020.20 ± 216.63	6960.51 ± 282.35	0.575 ± 0.039
RegionDCL w/o distance-bias	6018.94 ± 641.71	8214.58 ± 931.11	0.385 ± 0.087	5293.04 ± 277.31	7348.86 ± 374.62	0.532 ± 0.030
RegionDCL w/ Poly2Vec	**4957.58** ± 506.02	**6874.47** ± 851.73	**0.561** ± 0.117	**4602.75** ± 179.66	**6393.38** ± 279.70	**0.621** ± 0.037

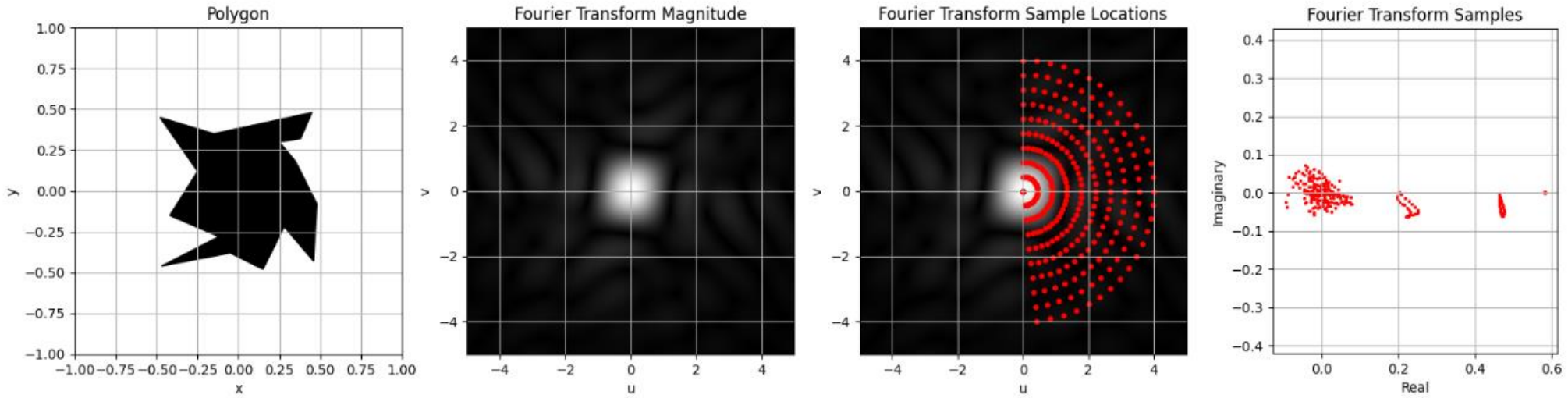


Residential Areas Industrial Areas

Maria Despoina Siampou, Jialiang Li, John Krumm, Cyrus Shahabi, and Hua Lu. Poly2Vec: Polymorphic Fourier-Based Encoding of Geospatial Objectsfor GeoAI Applications. ICML 2025.

Poly2Vec

- Poly2Vec:
  - Decompose geo-entities to triangles.
  - Define a uniform line/triangle, then define the **transform function** from the uniform triangle to triangles.
  - Use Fourier transform to representation each **function**. Fourier transform is a linear operation, and addable.

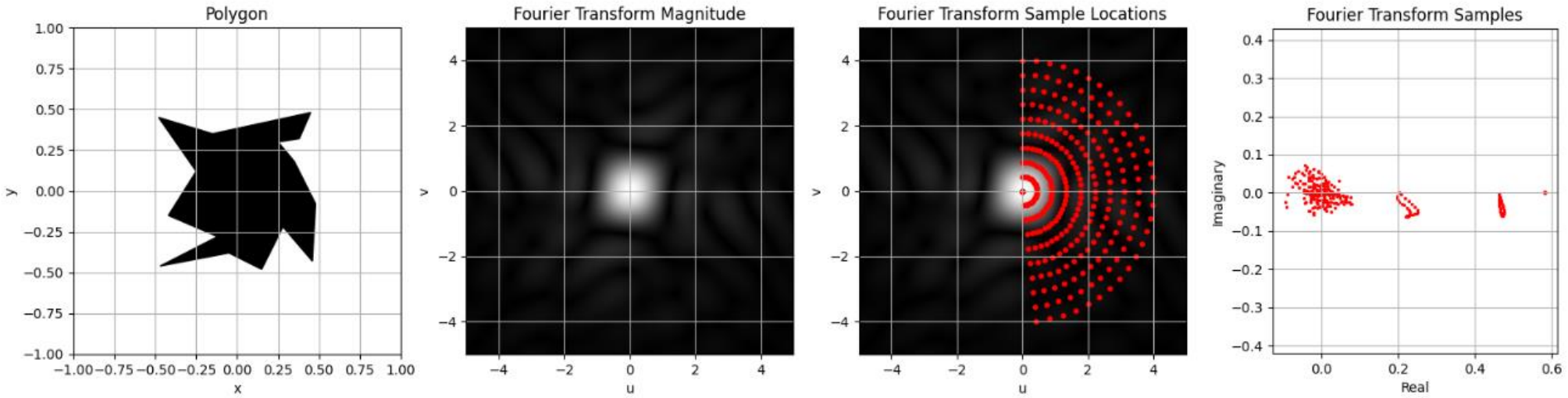


Maria Despoina Siampou, Jialiang Li, John Krumm, Cyrus Shahabi, and Hua Lu. Poly2Vec: Polymorphic Fourier-Based Encoding of Geospatial Objectsfor GeoAI Applications. ICML 2025.

Poly2Vec

- Poly2Vec:
  - Decompose geo-entities to triangles.
  - Define a uniform line/triangle, then define the **transform function** from the uniform triangle to triangles.
  - Use Fourier transform to representation each **function**. Fourier transform is a linear operation, and addable.

Limitation: Sampling in the Fourier space is **uninterpretable**.



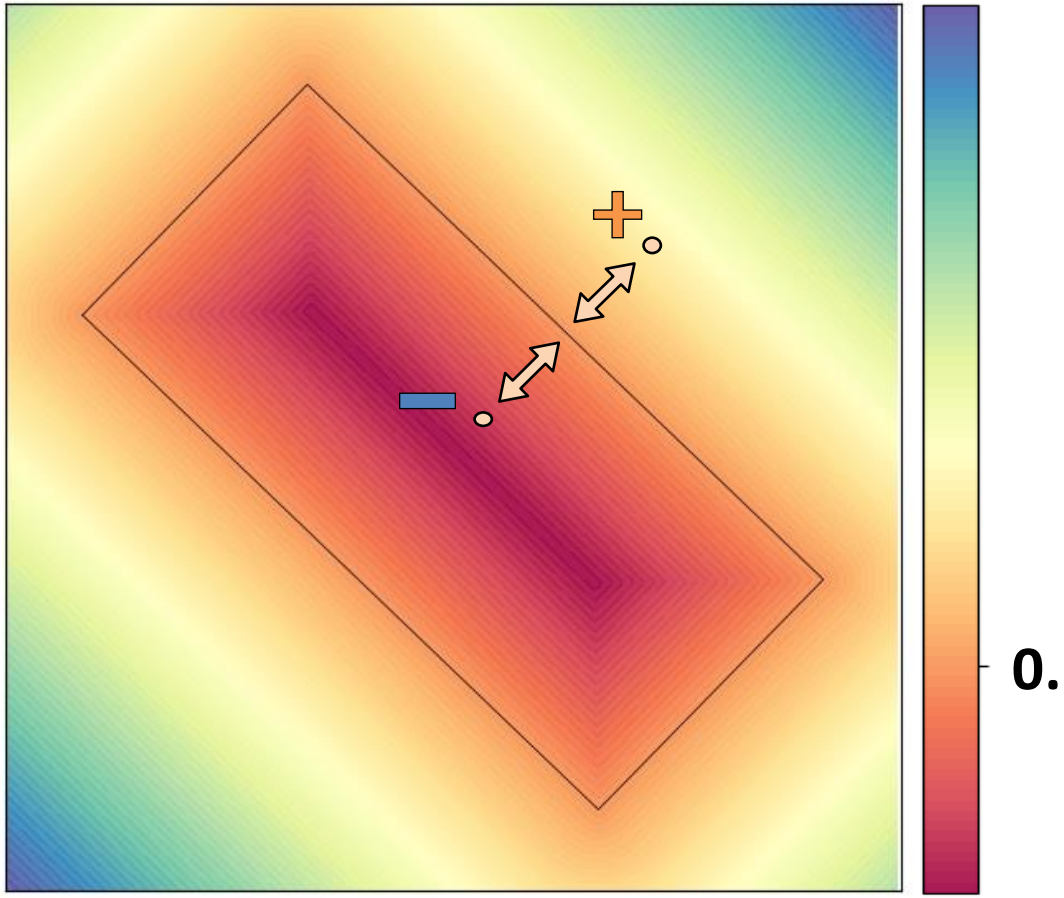
Maria Despoina Siampou, Jialiang Li, John Krumm, Cyrus Shahabi, and Hua Lu. Poly2Vec: Polymorphic Fourier-Based Encoding of Geospatial Objectsfor GeoAI Applications. ICML 2025.

Geo2Vec: Continuous Representation of Geo-entities

- Using **Signed Distance Field** (SDF) to represent each entity in the continuous space.
- Definition:

  Given a geo-entity E, the Signed Distance Field is a scalar field defined over a continuous spatial domain $\Omega_E \subseteq R^2$, in which each point $x \in \Omega_E$ is assigned a scalar value representing its signed distance to E.

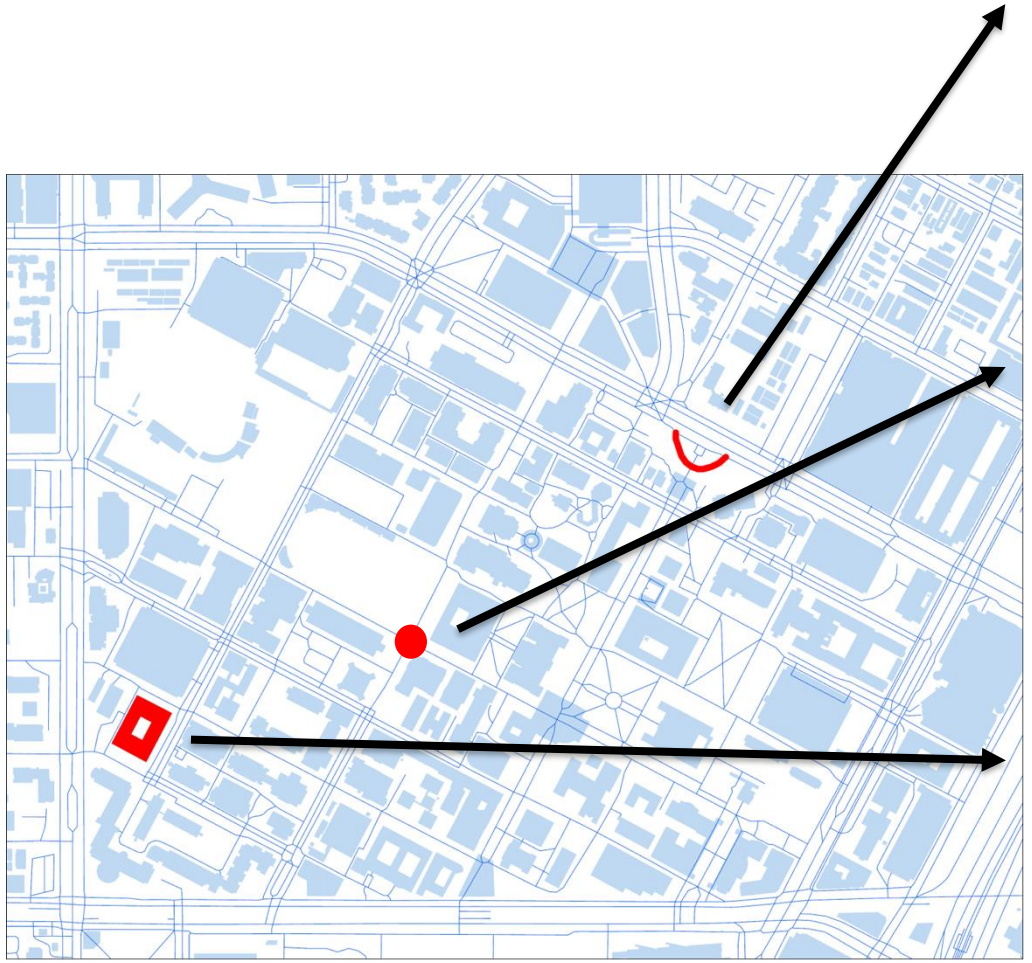
- Continuous representation

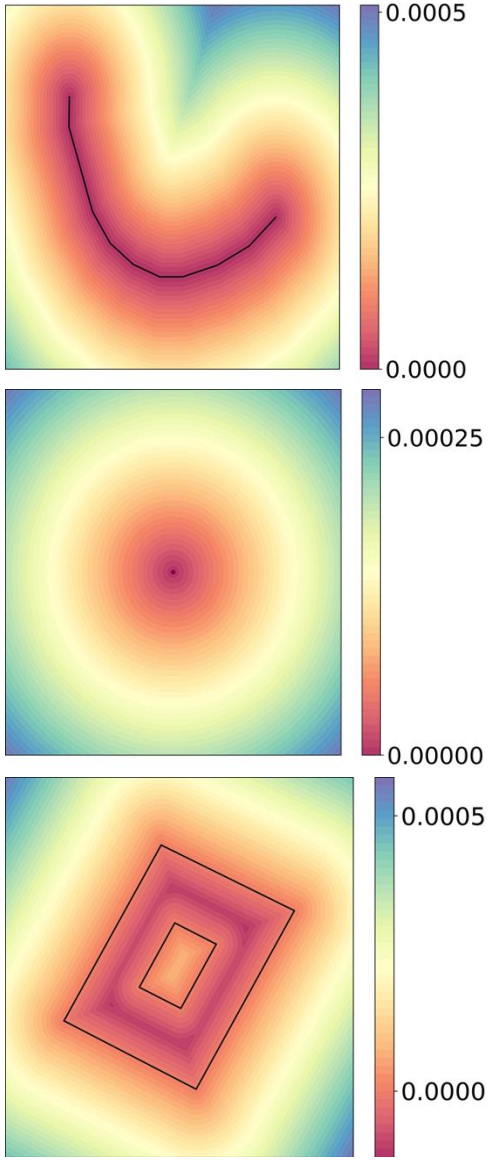


Chu, Chen, and Cyrus Shahabi. "Geo2vec: Shape-and distance-aware neural representation of geospatial entities." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. No. 23. 2026.

Unified Representation of Geo-entities

- Point, LineString, MultiLineString
  - All Positive
- Polygon, MultiPolygon:
  - Positive outside
  - Negative Inside

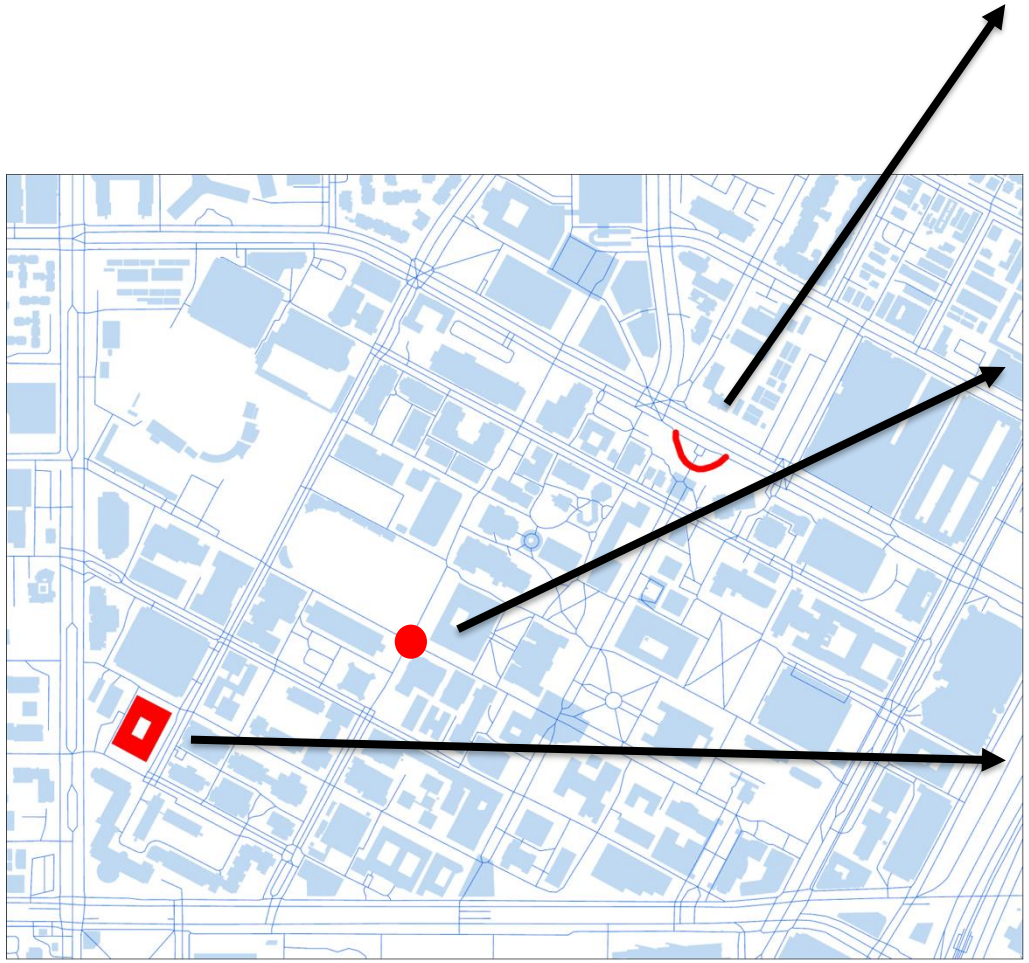


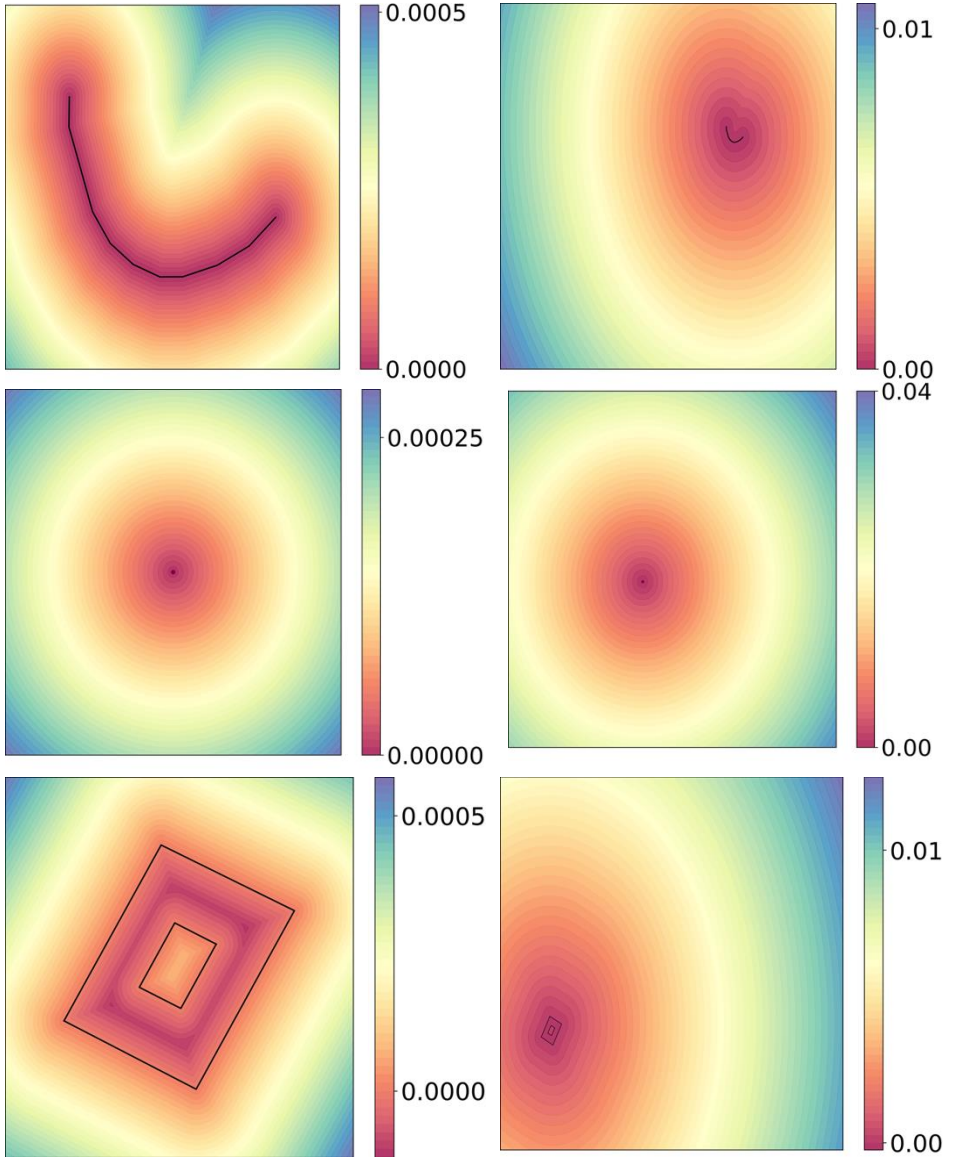


Chu, Chen, and Cyrus Shahabi. "Geo2vec: Shape-and distance-aware neural representation of geospatial entities." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. No. 23. 2026.

Location-Shape coherent representation

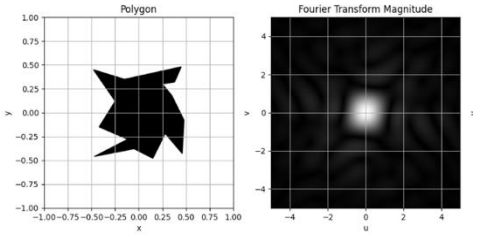
- Point, LineString, MultiLineString
  - All Positive
- Polygon, MultiPolygon:
  - Positive outside
  - Negative Inside

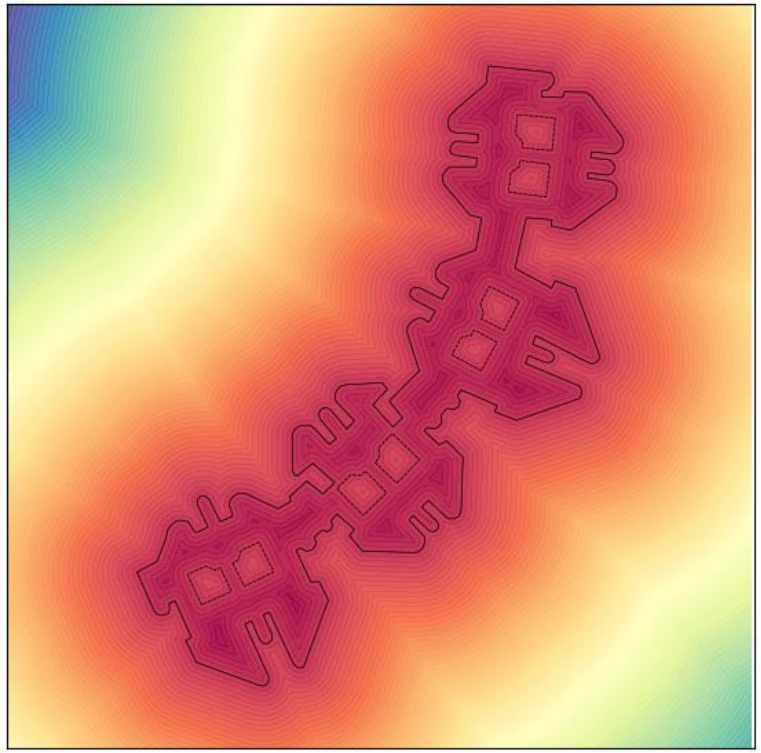


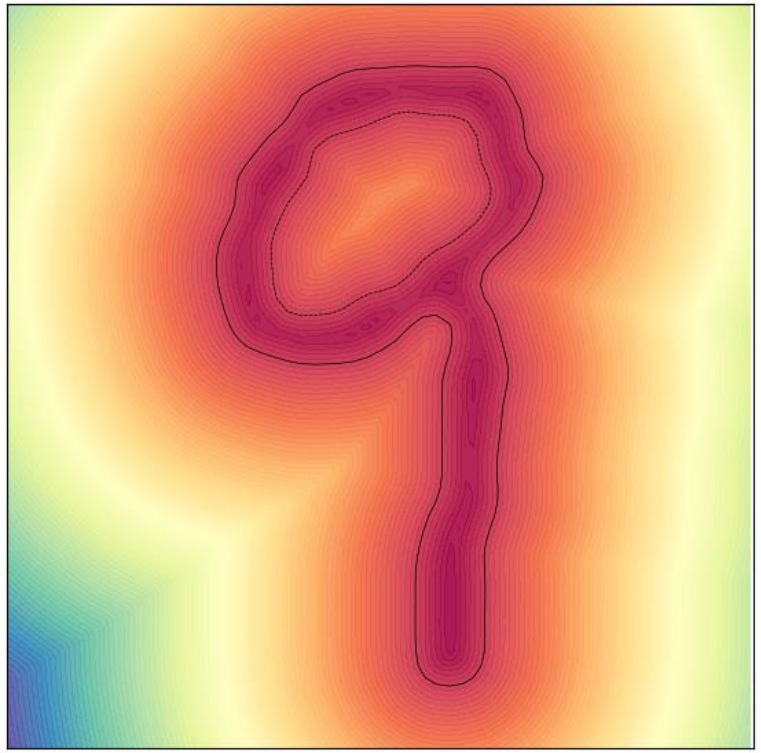


Chu, Chen, and Cyrus Shahabi. "Geo2vec: Shape-and distance-aware neural representation of geospatial entities." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. No. 23. 2026.

Interpretable representation

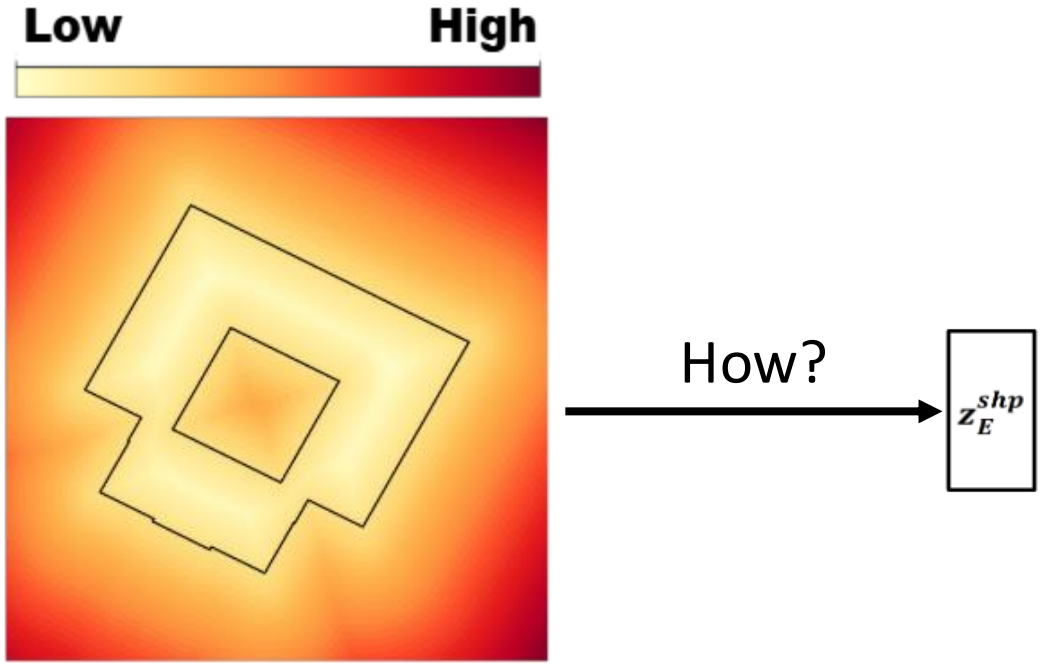






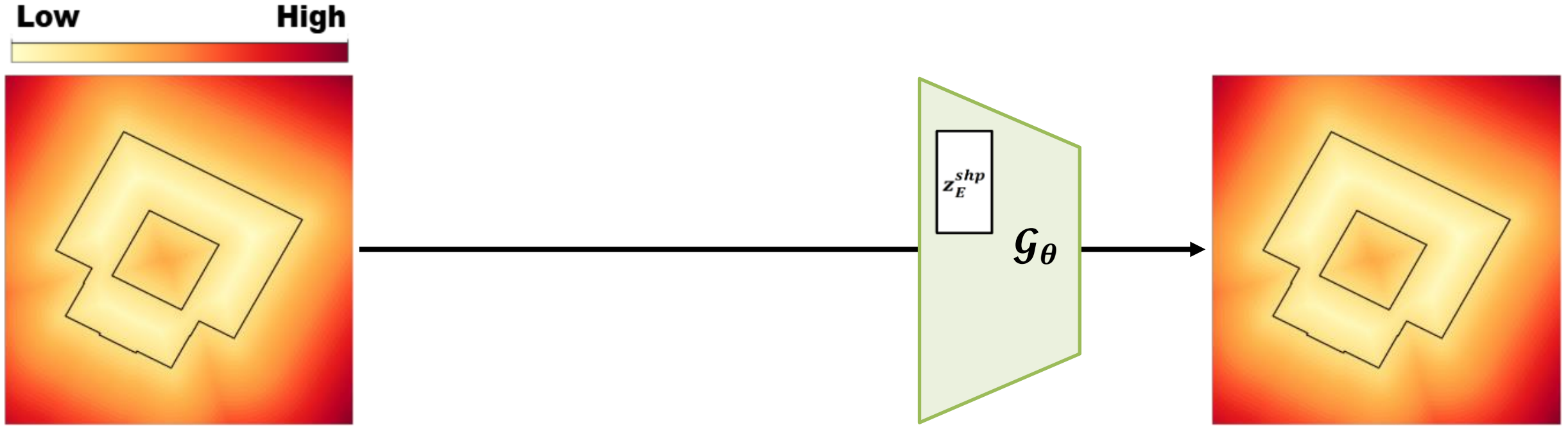
Chu, Chen, and Cyrus Shahabi. "Geo2vec: Shape-and distance-aware neural representation of geospatial entities." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 40. No. 23. 2026.

Geo2Vec model



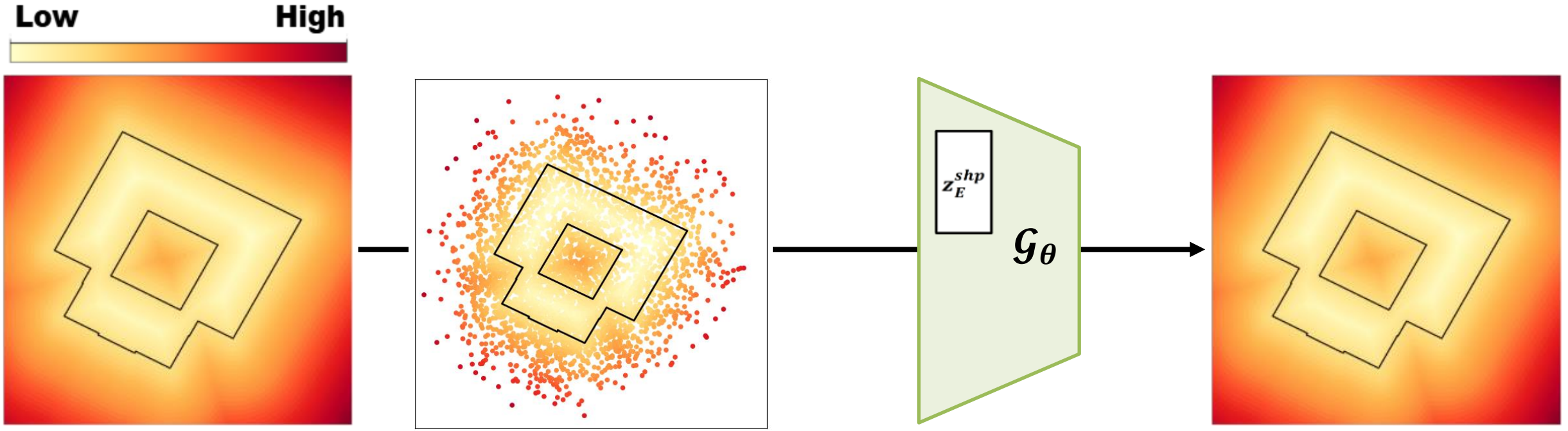
Geo2Vec model

- Approximate the SDF of a geo-entity with a neural network $\mathcal{G}_\theta$.



Geo2Vec model

- Approximate the SDF through sampling in the coordinate space.

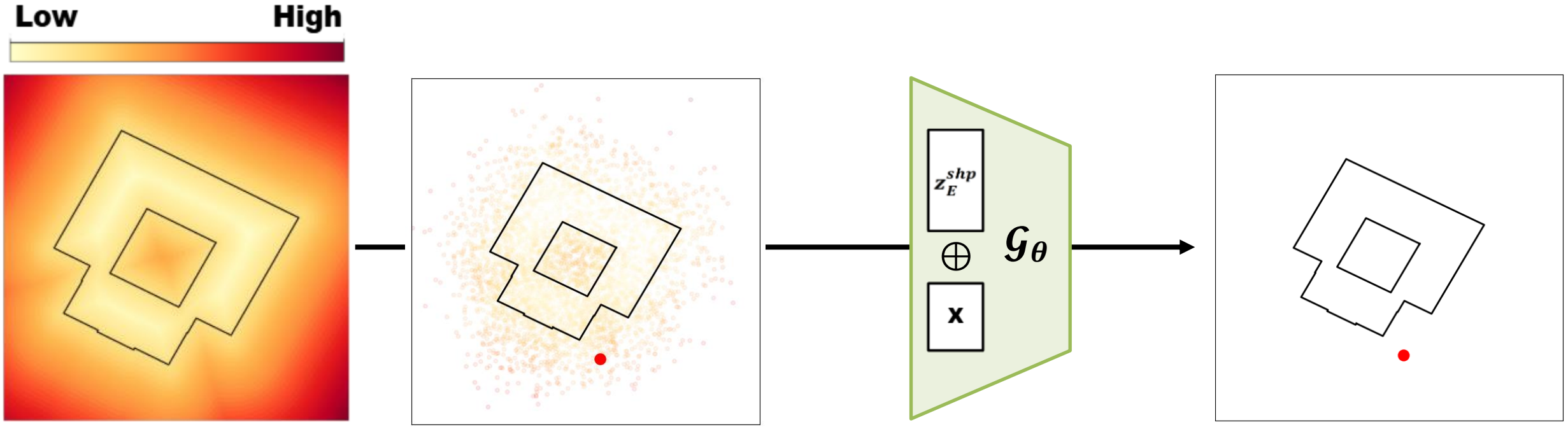


Ground Truth SDF

Learned SDF

Geo2Vec model

- Approximate the SDF through sampling in the coordinate space.

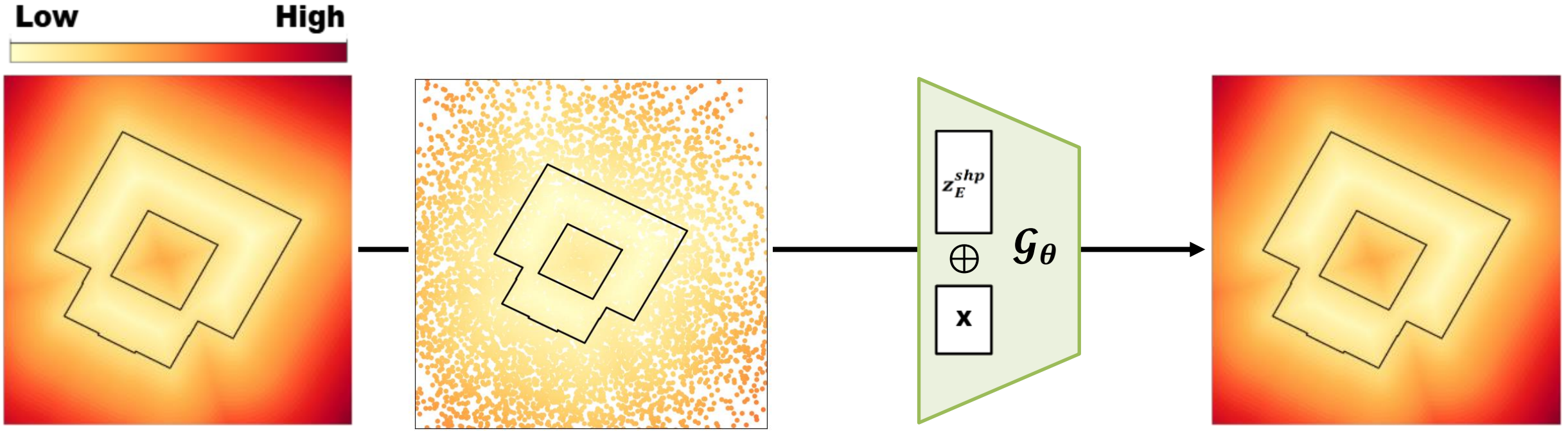


Ground Truth SDF

Learned SDF

Geo2Vec model

- Approximate the SDF of a geo-entity with a neural network $\mathcal{G}_\theta$.

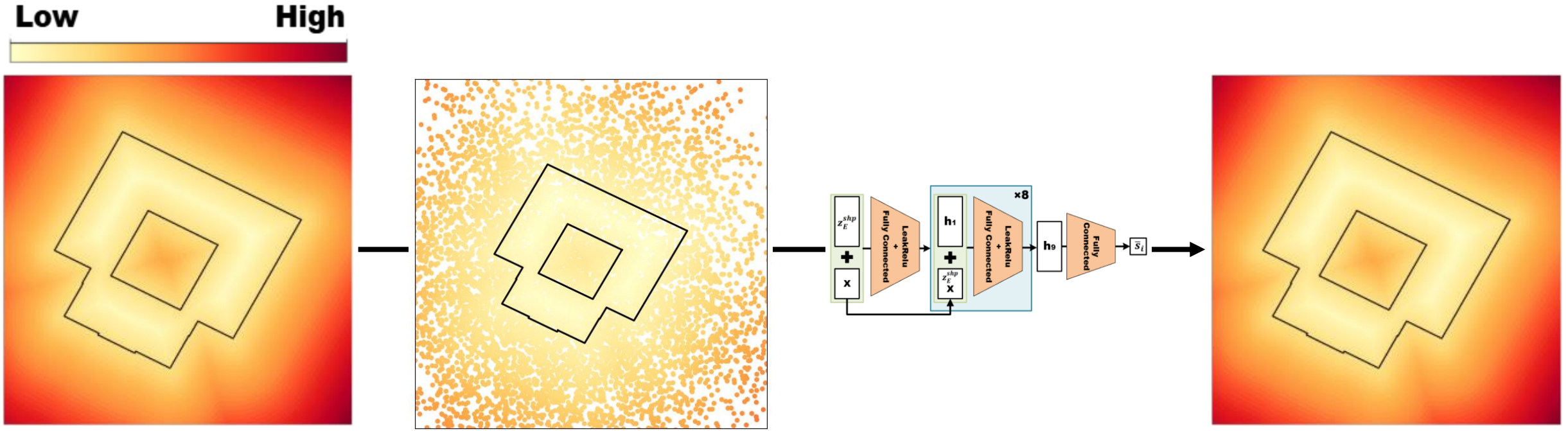


Ground Truth SDF

Learned SDF

Geo2Vec model

- Approximate the SDF of a geo-entity with a neural network $\mathcal{G}_\theta$.



Ground Truth SDF

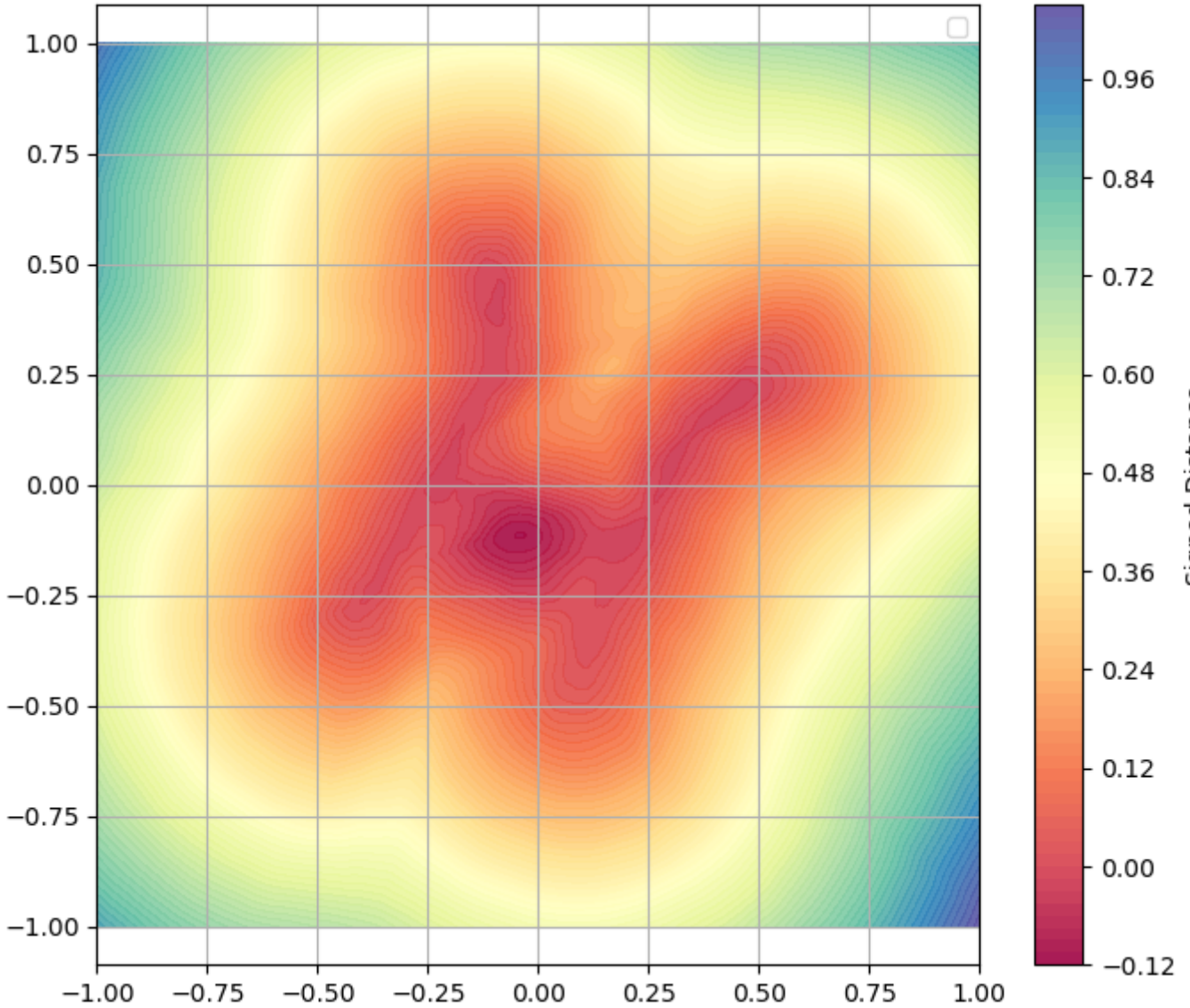
Learned SDF

Details

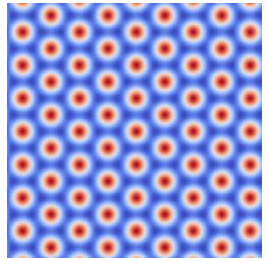
- Adaptive & Rotation invariant **Positional Encoder**:
  Learn shape in detail, learn location more effectively.

$$PE(\mathbf{x}) = (\sin(2^{L_{\min}}\pi\mathbf{x}), \cos(2^{L_{\min}}\pi\mathbf{x}), \ldots, \sin(2^{L_{\max}}\pi\mathbf{x}), \cos(2^{L_{\max}}\pi\mathbf{x})),$$

$$\Delta_x = \max_{E\in G}(E.x) - \min_{E\in G}(E.x), \Delta_y = \max_{E\in G}(E.y) - \min_{E\in G}(E.y),$$
$$\Delta_{\min} = \min(\Delta_x, \Delta_y), \Delta_{\max} = \max(\Delta_x, \Delta_y),$$
$$L^{\text{loc}}_{\max} \leq \log_2\left(\frac{2}{\Delta_{\min}}\right), \quad L^{\text{shp}}_{\max} \geq \log_2\left(\frac{2}{\Delta_{\min}}\right),$$
$$L^{\text{loc}}_{\min}, L^{\text{shp}}_{\min} \leq 1 - \log_2(\Delta_{\max}). \qquad (7)$$



Details

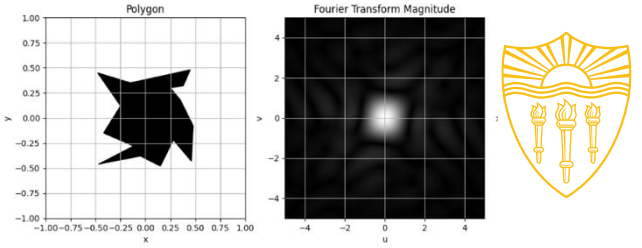


- Adaptive & Rotation invariant **Positional Encoder**:
  Learn shape in detail, learn location more effectively.

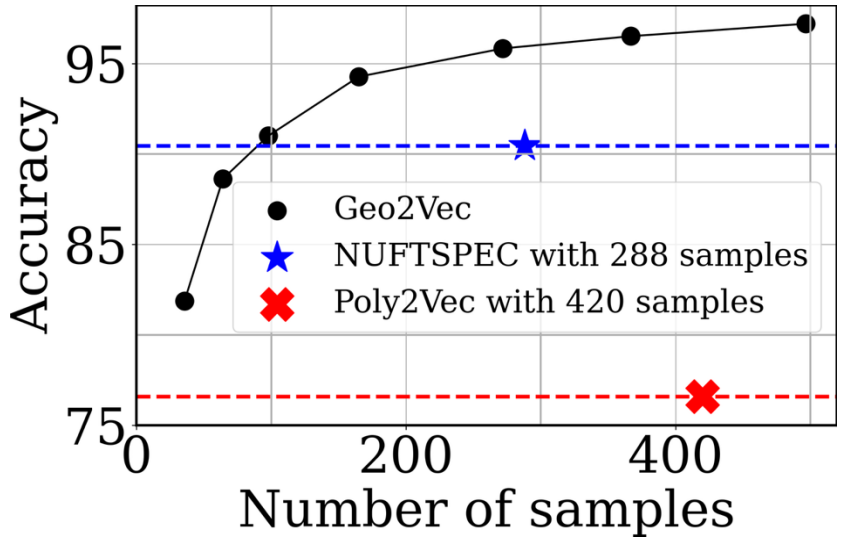
$$PE(\mathbf{x}) = (\sin(2^{L_{\min}}\pi\mathbf{x}), \cos(2^{L_{\min}}\pi\mathbf{x}), \ldots, \sin(2^{L_{\max}}\pi\mathbf{x}), \cos(2^{L_{\max}}\pi\mathbf{x})),$$

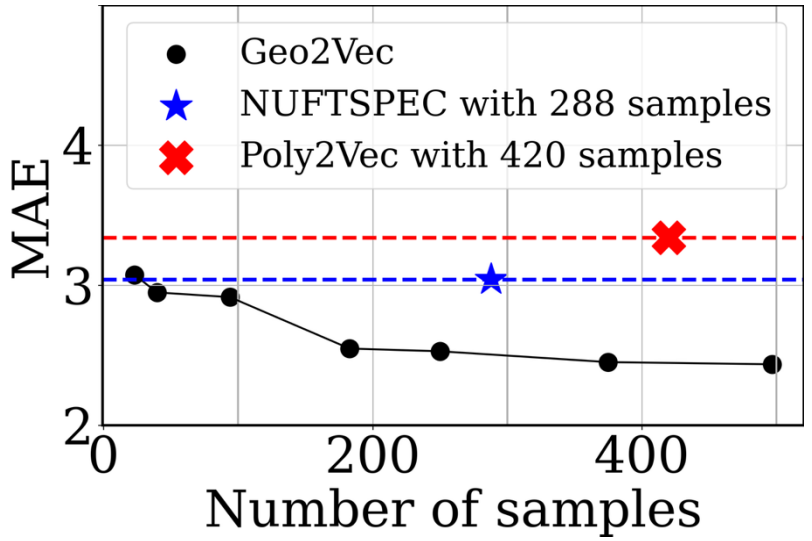
$$\Delta_x = \max_{E\in G}(E.x) - \min_{E\in G}(E.x), \Delta_y = \max_{E\in G}(E.y) - \min_{E\in G}(E.y),$$
$$\Delta_{\min} = \min(\Delta_x, \Delta_y), \Delta_{\max} = \max(\Delta_x, \Delta_y),$$
$$L^{\text{loc}}_{\max} \leq \log_2\left(\frac{2}{\Delta_{\min}}\right), \quad L^{\text{shp}}_{\max} \geq \log_2\left(\frac{2}{\Delta_{\min}}\right),$$
$$L^{\text{loc}}_{\min}, L^{\text{shp}}_{\min} \leq 1 - \log_2(\Delta_{\max}). \qquad (7)$$

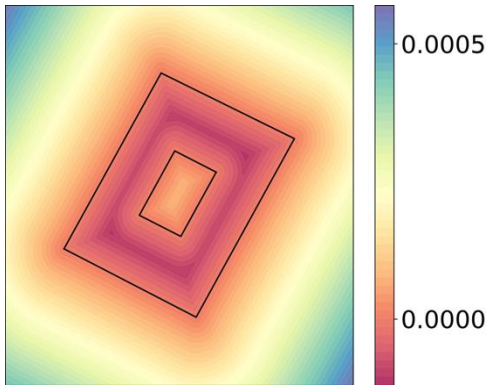
Sampling quality



- More efficient sampling in the coordinate space.







Shape Representation

- Up to 61.95% improvement on number of edges prediction of Polygon.
- At least 44.1% improvement in Polyline length representation.

Polyline Representation: MAE performance on Length of Line prediction.

	Singapore	NYC
	Line Length↓	Line Length↓
T2Vec	10.38±0.45	13.20±0.42
T-JEPA	10.25±0.54	12.65±0.36
Poly2Vec	13.55±0.79	21.11±0.48
Geo2Vec	**5.75±0.26**	**7.07±0.16**

Polygon Representation: Accuracy on Shape Classification and MAE on the Number of Edges prediction

	Building		MNIST		Singapore	NYC
	Shape↑	Edge↓	Shape↑	Edge↓	Edge↓	Edge↓
PolyGNN	87.84±0.005	–	7.77±0.013	–	–	–
NUFTSPEC	90.46±0.730	3.04±0.125	96.90±0.116	16.76±0.588	3.66±0.023	1.45±0.020
Poly2Vec	76.59±1.403	3.34±0.127	92.52±0.265	29.29±0.807	3.68±0.109	1.21±0.045
Geo2Vec	**97.34±0.310**	**2.22±0.050**	**97.58±0.097**	**9.45±0.124**	**1.40±0.011**	**0.72±0.002**

Location Representation

- Up to 54.3% improvement on Topological Relationship Classification.
- More **uniformed** performance when predicting the topology and distance between **different types of entities**.

Overall model performance on distance estimation.

	Building	MNIST	Singapore			NYC		
	Pg-Pg↓	Pg-Pg↓	Pt-Pg↓	Pl-Pg↓	Pg-Pg↓	Pt-Pg↓	Pl-Pg↓	Pg-Pg↓
TILE	217.1±1.6	223.8±1.0	99.9±1.7	115.5±1.5	114.3±1.4	127.6±0.7	154.3±2.1	167.4±2.0
THEORY	7.3±4.3	34.2±1.0	24.3±0.9	25.0±0.4	25.0±1.1	26.2±1.1	26.6±0.5	27.4±0.7
Poly2Vec	13.1±1.0	21.0±0.4	15.9±0.6	19.9±1.7	22.0±0.6	28.7±1.4	28.5±0.4	52.7±0.8
Geo2Vec	**6.4±0.9**	**13.0±0.8**	**5.4±0.5**	**5.0±0.1**	**5.5±0.4**	**10.2±0.1**	**13.0±0.9**	**12.9±0.6**

Model accuracy on Topological Relationship Classification.

	Singapore					NYC				
	Pt-Pl↑	Pt-Pg↑	Pl-Pl↑	Pl-Pg↑	Pg-Pg↑	Pt-Pl↑	Pt-Pg↑	Pl-Pl↑	Pl-Pg↑	Pg-Pg↑
NUFTSPEC	–	–	–	–	60.2±0.9	–	–	–	–	58.5±0.8
T2VEC	–	–	72.8±2.3	–	–	–	–	80.7±12.1	–	–
T-JEPA	–	–	75.4±1.8	–	–	–	–	79.8±8.6	–	–
DIRECT	82.3±1.3	84.3±0.5	73.3±0.7	36.8±1.0	35.7±1.8	84.6±1.1	90.9±1.8	74.5±0.8	49.5±0.9	44.6±2.3
TILE	79.0±2.1	70.0±1.0	50.5±0.5	45.9±1.3	41.1±1.3	65.9±1.3	78.3±0.7	50.2±0.9	49.4±3.8	40.5±0.5
WRAP	88.6±0.3	88.0±0.8	71.6±1.1	47.6±1.0	47.6±1.0	88.6±0.6	88.0±1.7	73.3±0.9	55.0±1.1	38.1±0.7
GRID	84.6±0.4	84.4±0.4	69.7±3.1	45.8±0.4	45.8±0.4	82.2±3.9	89.1±0.4	73.9±0.9	51.6±0.8	38.1±3.1
THEORY	89.2±0.3	90.0±0.5	71.9±0.8	45.0±1.0	45.0±1.0	89.7±0.8	90.9±0.8	73.4±0.8	59.1±0.6	45.5±4.1
Poly2Vec	95.5±0.7	94.9±0.2	81.2±1.0	50.9±0.8	70.2±0.6	95.3±0.3	98.0±0.2	83.0±0.4	64.1±6.2	68.4±0.8
Geo2Vec	**98.5±0.3**	**96.1±0.2**	**96.4±0.5**	**61.2±0.4**	**75.6±0.4**	**98.7±0.4**	**99.1±0.3**	**98.9±0.3**	**67.5±0.8**	**70.0±0.4**

Conclusions

- Spatial Representation Learning is the first step for building a Geospatial AI model.
- SRL introduces geospatial inductive bias, making the model generalizable to unseen regions.
- Poly2Vec and Geo2Vec are unified representation methods for geospatial entities.