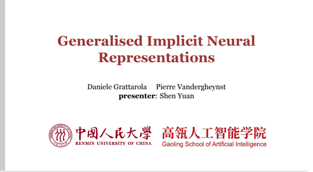
2023-01-05

1.5Pre


Generalised Implicit Neural Representations

Daniele Grattarola Pierre Vandergheynst
presenter: Shen Yuan

Generalised Implicit Neural Representations

Daniele Grattarola Pierre Vandergheynst
presenter: Shen Yuan

2023-01-05

1.5Pre

└─Outline

Outline

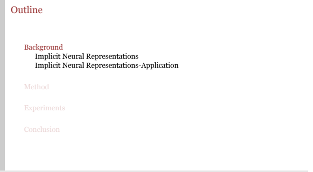
Background

Method

Experiments

Conclusion

Outline


Outline

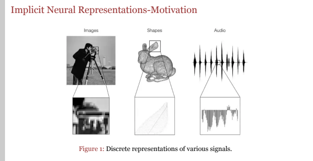
Background
Implicit Neural Representations
Implicit Neural Representations-Application

Method

Experiments

Conclusion

Outline



The world around us is not discrete. But we choose to represent real-world signals such as images or sound in a discrete manner. For example, we represent an image as a grid of pixels, we represent shapes as point clouds, and we use discrete samples for the audio.

Implicit Neural Representations-Motivation

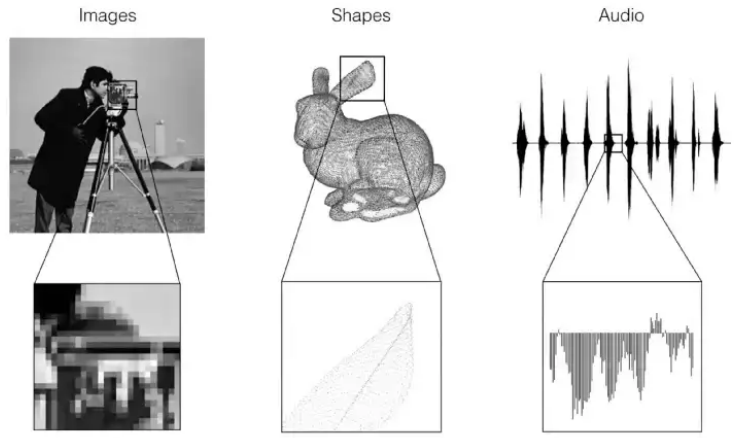
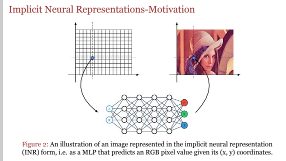


Figure 1: Discrete representations of various signals.



Now imagine we have some continuous function f for example a MLP that can accurately represents the image signal. That is, if we pass f a pixel coordinate (x,y) as input, f outputs the correct RGB value for that pixel. We could sample pixel grids at any resolution from f! And This applies to other signals such as sound!

Implicit Neural Representations-Motivation

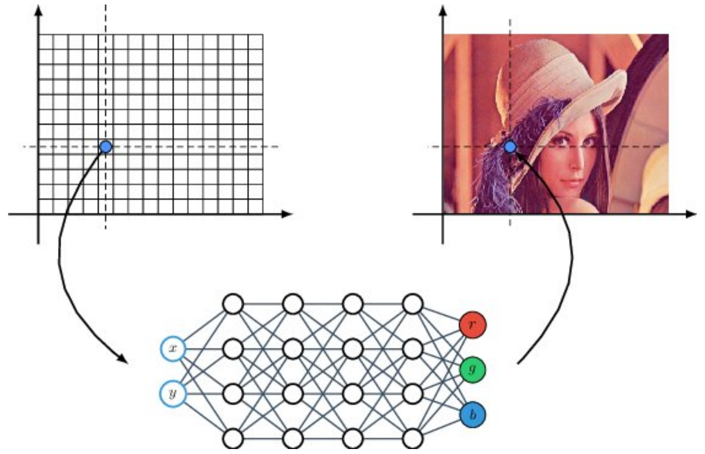
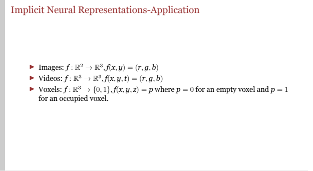


Figure 2: An illustration of an image represented in the implicit neural representation (INR) form, i.e. as a MLP that predicts an RGB pixel value given its (x, y) coordinates.

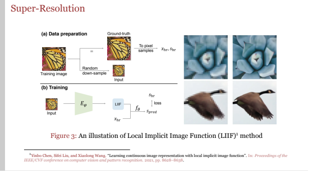

Implicit Neural Representations-Application

- Images: $f : \mathbb{R}^2 \to \mathbb{R}^3, f(x, y) = (r, g, b)$
- Videos: $f : \mathbb{R}^3 \to \mathbb{R}^3, f(x, y, t) = (r, g, b)$
- Voxels: $f : \mathbb{R}^3 \to \{0, 1\}, f(x, y, z) = p$ where $p = 0$ for an empty voxel and $p = 1$ for an occupied voxel.

Then let's see, the function f parameterizes the signal as a mathematical formula in different situation. Image Video. For voxels, voxels mean 3-dimensional objects. its output is a scalar. The p indicates whether the location in 3-dimensional space is inside or outside the object.

Implicit Neural Representations-Application

- Images: $f : \mathbb{R}^2 \to \mathbb{R}^3, f(x, y) = (r, g, b)$
- Videos: $f : \mathbb{R}^3 \to \mathbb{R}^3, f(x, y, t) = (r, g, b)$
- Voxels: $f : \mathbb{R}^3 \to \{0, 1\}, f(x, y, z) = p$ where $p = 0$ for an empty voxel and $p = 1$ for an occupied voxel.



Then, Let's see some recent developments in this area. This paper use INR to get Super resolution images. And The network is very simple. Firstly, the image is preprocessed into high-resolution and low-resolution images, and the low-resolution images are passed through an encoder to get feature map. Then, high-resolution image coordinates (x, y) and low-resolution feature map are input, and an MLP is used to predict high-resolution rgb values. Finally, we can get images in arbitrary resolution.

Super-Resolution

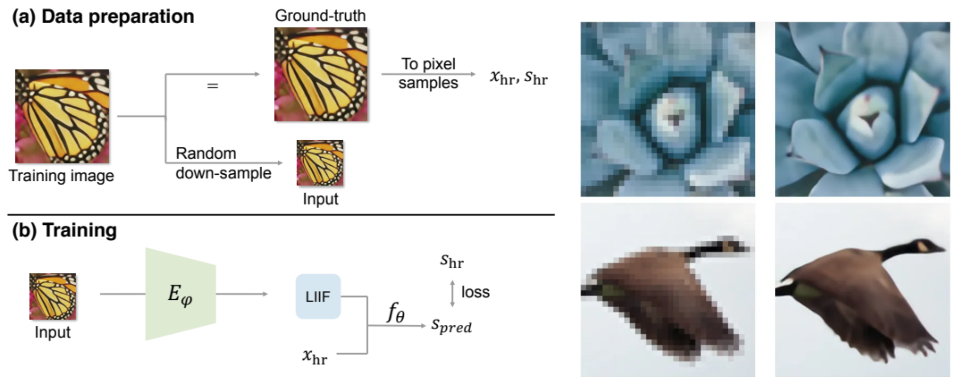
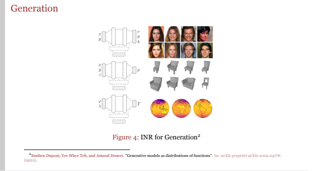


Figure 3: An illustation of Local Implicit Image Function (LIIF)[1] method

[1] Yinbo Chen, Sifei Liu, and Xiaolong Wang. "Learning continuous image representation with local implicit image function". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2021, pp. 8628–8638.



By representing data as continuous functions, this paper use the same model to generate distributions of images, 3D shapes and climate data.

Generation

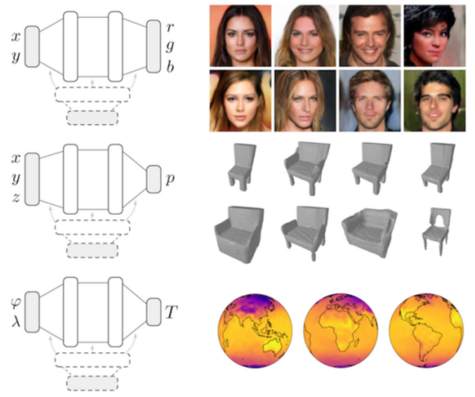


Figure 4: INR for Generation[2]

[2] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. "Generative models as distributions of functions". In: *arXiv preprint arXiv:2102.04776* (2021).

2023-01-05

1.5Pre
└─Method
└─Outline

Outline

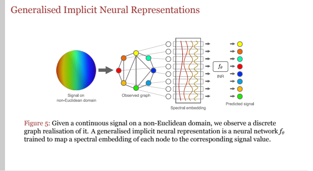
Background
Implicit Neural Representations
Implicit Neural Representations-Application

Method

Experiments

Conclusion

Outline



Then, Let's see the method. Given a continuous signal on a non-Euclidean domain, we observe a discrete graph realisation of it. Here, in this paper, the non-Euclidean domain can be seen as a huge graph. we can sample some nodes from the huge graph. Then, we can get the corresponding spectral embedding of each node by some method. Finally, train a neural network f_θ to map a spectral embedding of each node to the corresponding signal value. So How to get the spectral embedding.

Generalised Implicit Neural Representations

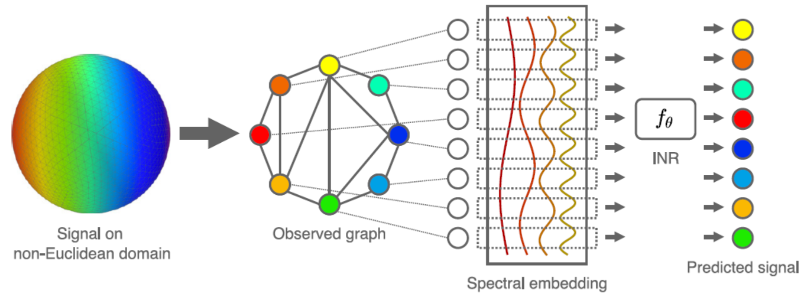
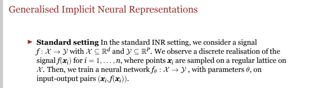


Figure 5: Given a continuous signal on a non-Euclidean domain, we observe a discrete graph realisation of it. A generalised implicit neural representation is a neural network f_θ trained to map a spectral embedding of each node to the corresponding signal value.

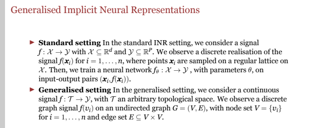

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.

before introducing the method, let's see the preliminary and the difference between Standard setting and Generalised setting. Standard setting Generalised setting. So What is the input In the generalised setting in other words How to represent the nodes? the authors use The eigenvectors of the graph Laplacian.

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.

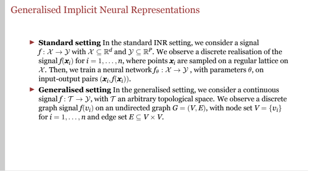

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \dots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \dots, n$ and edge set $E \subseteq V \times V$.

before introducing the method, let's see the preliminary and the difference between Standard setting and Generalised setting. Standard setting Generalised setting. So What is the input In the generalised setting in other words How to represent the nodes? the authors use The eigenvectors of the graph Laplacian.

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \dots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \dots, n$ and edge set $E \subseteq V \times V$.

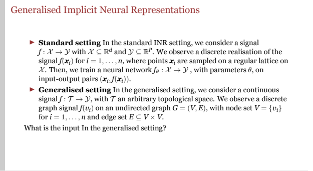

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.

before introducing the method, let's see the preliminary and the difference between Standard setting and Generalised setting. Standard setting Generalised setting. So What is the input In the generalised setting in other words How to represent the nodes? the authors use The eigenvectors of the graph Laplacian.

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.


Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.

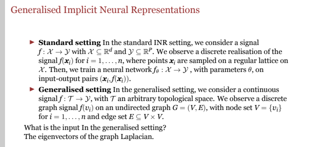
What is the input In the generalised setting?

before introducing the method, let's see the preliminary and the difference between Standard setting and Generalised setting. Standard setting Generalised setting. So What is the input In the generalised setting in other words How to represent the nodes? the authors use The eigenvectors of the graph Laplacian.

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.

What is the input In the generalised setting?


Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.

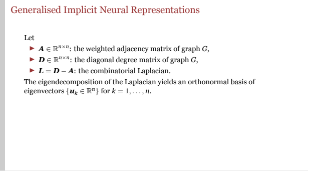
What is the input In the generalised setting?
The eigenvectors of the graph Laplacian.

before introducing the method, let's see the preliminary and the difference between Standard setting and Generalised setting. Standard setting Generalised setting. So What is the input In the generalised setting in other words How to represent the nodes? the authors use The eigenvectors of the graph Laplacian.

Generalised Implicit Neural Representations

- **Standard setting** In the standard INR setting, we consider a signal $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. We observe a discrete realisation of the signal $f(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, where points $\boldsymbol{x}_i$ are sampled on a regular lattice on $\mathcal{X}$. Then, we train a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, with parameters θ, on input-output pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$.
- **Generalised setting** In the generalised setting, we consider a continuous signal $f : \mathcal{T} \to \mathcal{Y}$, with $\mathcal{T}$ an arbitrary topological space. We observe a discrete graph signal $f(v_i)$ on an undirected graph $G = (V, E)$, with node set $V = \{v_i\}$ for $i = 1, \ldots, n$ and edge set $E \subseteq V \times V$.

What is the input In the generalised setting?
The eigenvectors of the graph Laplacian.


Generalised Implicit Neural Representations

Let

- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$: the weighted adjacency matrix of graph G,
- $\boldsymbol{D} \in \mathbb{R}^{n \times n}$: the diagonal degree matrix of graph G,
- $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$: the combinatorial Laplacian.

The eigendecomposition of the Laplacian yields an orthonormal basis of eigenvectors $\{\boldsymbol{u}_k \in \mathbb{R}^n\}$ for $k = 1, \ldots, n$.

So the authors solve the generalised INR problem through a spectral embedding obtained from the graph Laplacian. Because A and D are both real symmetric matrix. It can be seen that Laplacian matrix is also a real symmetric matrix of order n. And Real symmetric matrix has the some properties: its eigenvalues are all real numbers, and there are n linearly independent eigenvectors, and the eigenvectors are all real vectors. and its eigenvectors are orthogonal;
Then, the author define a generalised spectral embedding of size k for node v_i as the equation 1. The root n is to rescale the eigenvectors by ensuring that the embeddings for graphs of different sizes will have the same range and that similar nodes will have similar embeddings regardless of the graph size.
Finally, train a neural network to learn the map. and I checked the code on GitHub, the neural network is very simple, just a 6-layer MLP with sin activation function instead of ReLU. Trigonometric function sin()

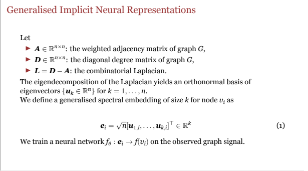
Here we focus on the Laplacian since it is well known, sparse, and easy to compute, and its eigendecomposition is stable to graph perturbations.

Generalised Implicit Neural Representations

Let

- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$: the weighted adjacency matrix of graph G,
- $\boldsymbol{D} \in \mathbb{R}^{n \times n}$: the diagonal degree matrix of graph G,
- $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$: the combinatorial Laplacian.

The eigendecomposition of the Laplacian yields an orthonormal basis of eigenvectors $\{\boldsymbol{u}_k \in \mathbb{R}^n\}$ for $k = 1, \ldots, n$.


Generalised Implicit Neural Representations

Let
- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$: the weighted adjacency matrix of graph G,
- $\boldsymbol{D} \in \mathbb{R}^{n\times n}$: the diagonal degree matrix of graph G,
- $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$: the combinatorial Laplacian.

The eigendecomposition of the Laplacian yields an orthonormal basis of eigenvectors $\{\boldsymbol{u}_k \in \mathbb{R}^n\}$ for $k = 1, \dots, n$.
We define a generalised spectral embedding of size k for node v_i as

$$\boldsymbol{e}_i = \sqrt{n}[\boldsymbol{u}_{1,i}, \dots, \boldsymbol{u}_{k,i}]^\top \in \mathbb{R}^k \quad (1)$$

We train a neural network $f_\theta : \boldsymbol{e}_i \to f(v_i)$ on the observed graph signal.

So the authors solve the generalised INR problem through a spectral embedding obtained from the graph Laplacian. Because A and D are both real symmetric matrix. It can be seen that Laplacian matrix is also a real symmetric matrix of order n. And Real symmetric matrix has the some properties: its eigenvalues are all real numbers, and there are n linearly independent eigenvectors, and the eigenvectors are all real vectors. and its eigenvectors are orthogonal;
Then, the author define a generalised spectral embedding of size k for node v_i as the equation 1. The root n is to rescale the eigenvectors by ensuring that the embeddings for graphs of different sizes will have the same range and that similar nodes will have similar embeddings regardless of the graph size.
Finally, train a neural network to learn the map. and I checked the code on GitHub, the neural network is very simple, just a 6-layer MLP with sin activation function instead of ReLU. Trigonometric function sin()

Here we focus on the Laplacian since it is well known, sparse, and easy to compute, and its eigendecomposition is stable to graph perturbations.

Generalised Implicit Neural Representations

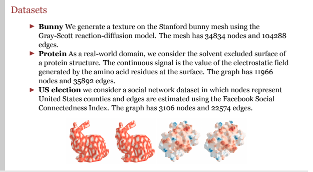
Let
- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$: the weighted adjacency matrix of graph G,
- $\boldsymbol{D} \in \mathbb{R}^{n\times n}$: the diagonal degree matrix of graph G,
- $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$: the combinatorial Laplacian.

The eigendecomposition of the Laplacian yields an orthonormal basis of eigenvectors $\{\boldsymbol{u}_k \in \mathbb{R}^n\}$ for $k = 1, \dots, n$.
We define a generalised spectral embedding of size k for node v_i as

$$\boldsymbol{e}_i = \sqrt{n}[\boldsymbol{u}_{1,i}, \dots, \boldsymbol{u}_{k,i}]^\top \in \mathbb{R}^k \quad (1)$$

We train a neural network $f_\theta : \boldsymbol{e}_i \to f(v_i)$ on the observed graph signal.

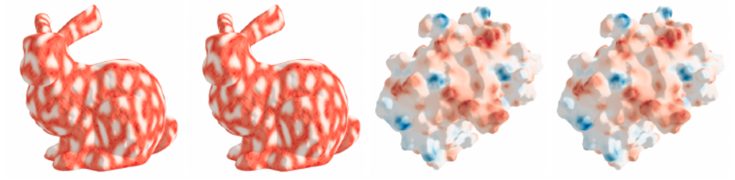
Outline



In the expriments, the authors consider three datasets as test cases. The bunny is a 3D Scanning model from The Stanford 3D Scanning Repository. and the authors generate a texture using the Gray-Scott reaction-diffusion model. The task is to predict the texture.
The Protein is to predict the electrostatic field generated by the amino acid residues at the protein surface.
The US election is to predict the county-wise outcome of the US presidential election.

Datasets

- **Bunny** We generate a texture on the Stanford bunny mesh using the Gray-Scott reaction-diffusion model. The mesh has 34834 nodes and 104288 edges.
- **Protein** As a real-world domain, we consider the solvent excluded surface of a protein structure. The continuous signal is the value of the electrostatic field generated by the amino acid residues at the surface. The graph has 11966 nodes and 35892 edges.
- **US election** we consider a social network dataset in which nodes represent United States counties and edges are estimated using the Facebook Social Connectedness Index. The graph has 3106 nodes and 22574 edges.



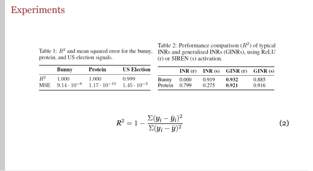

Experiments

Table 1: R^2 and mean squared error for the bunny, protein, and US election signals.

	Bunny	Protein	US Election
R^2	1.000	1.000	0.999
MSE	$9.14 \cdot 10^{-8}$	$1.17 \cdot 10^{-10}$	$1.45 \cdot 10^{-3}$

Table 2: Performance comparison (R^2) of typical INRs and generalised INRs (GINRs), using ReLU (r) or SIREN (s) activation.

	INR (r)	INR (s)	GINR (r)	GINR (s)
Bunny	0.000	0.919	**0.932**	0.885
Protein	0.799	0.275	**0.921**	0.916

$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y})^2} \quad (2)$$

Then, It's the result. The R-square is The coefficient of determination as shown in equation 2. In table 1 We can see that the GINR model accurately predict the three signals.

In Table 2 the typical INR means that the input is the node coordinates. s represents the sin() activation function. r means the ReLU function. The result is obviously bad, because the typical INR is not an equivalent model.

Experiments

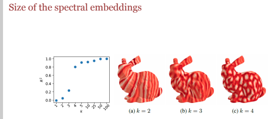
Table 1: R^2 and mean squared error for the bunny, protein, and US election signals.

	Bunny	Protein	US Election
R^2	1.000	1.000	0.999
MSE	$9.14 \cdot 10^{-8}$	$1.17 \cdot 10^{-10}$	$1.45 \cdot 10^{-3}$

Table 2: Performance comparison (R^2) of typical INRs and generalised INRs (GINRs), using ReLU (r) or SIREN (s) activation.

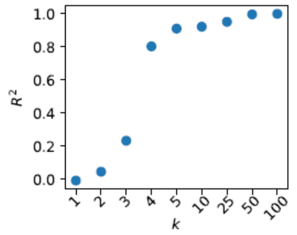
	INR (r)	INR (s)	GINR (r)	GINR (s)
Bunny	0.000	0.919	**0.932**	0.885
Protein	0.799	0.275	**0.921**	0.916

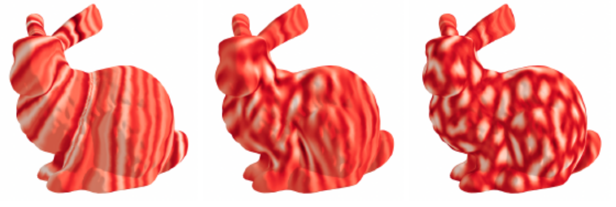
$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y})^2} \quad (2)$$



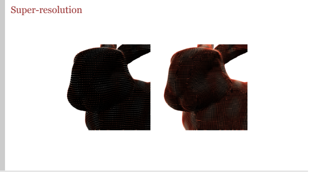
the dimension k of the spectral embeddings is a hyperparameter of the method, So the authors conduct this experiment to evaluate the impact of k on the performance of the method. we can see that the model fails to learn a meaningful representation with k less or epual to 3. This is because at least 3 non-trivial eigenvectors are needed to distinguish the main structural features of the bunny for example ears, tail

Size of the spectral embeddings



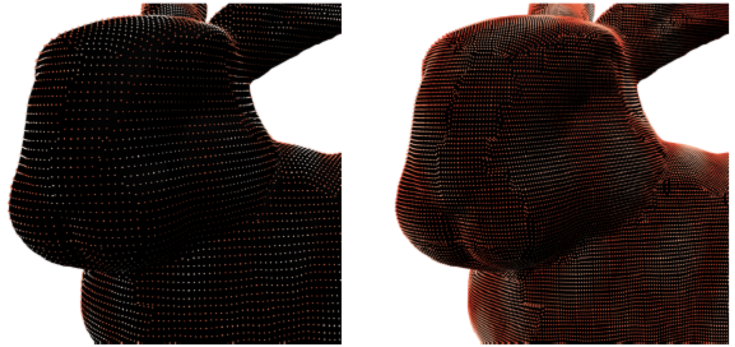


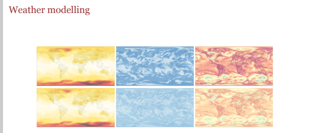
(a) $k = 2$ (b) $k = 3$ (c) $k = 4$



Next, the authors consider the super-resolution of a signal. this figure shows a comparison between the training and super-resolved signals, the left one is training graph and right one is its super-resolved version. we see that the GINR can predict the Super-resolution of the graph signal correctlly.

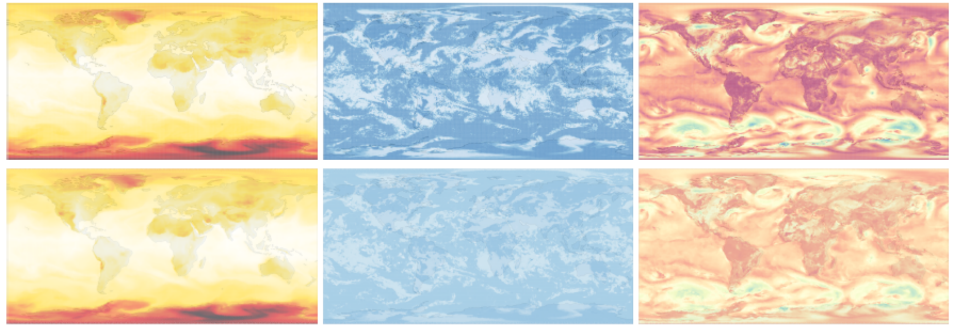
Super-resolution





the next one is final experiment that shows the application of generalised INRs in a real-world setting. Given spatial and temporal resolution, the task is to super-resolve the signal over space and time. top is the original resolution, and bottom is the super-resolved versions. There is an animated version with more details on github.

Weather modelling

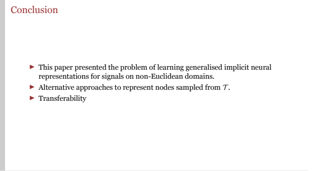


2023-01-05

1.5Pre
└─Conclusion

└─Outline

Outline


Conclusion

- This paper presented the problem of learning generalised implicit neural representations for signals on non-Euclidean domains.
- Alternative approaches to represent nodes sampled from $\mathcal{T}$.
- Transferability

the GINR method learns to map a spectral embedding of the domain to the value of the signal, without relying on node coordinate.
The authors think that locally linear embeddings, Isomap and diffusion maps can be used for node embedding besides Laplacian.Here they focus on the Laplacian because it is well known, sparse, and easy to compute, and its eigendecomposition is stable to graph perturbations.

I think the Transferability of this method is poor. Take the super resolution as an example, for each different 2d image, the model needs to learn it from scratch to obtain a super resolution version of that image, which is time consuming. right

Conclusion

- This paper presented the problem of learning generalised implicit neural representations for signals on non-Euclidean domains.
- Alternative approaches to represent nodes sampled from $\mathcal{T}$.
- Transferability