

Representation Learning of Geometric Trees

Zheng Zhang
Emory University
Atlanta, GA, USA
zheng.zhang@emory.edu

Allen Zhang
Georgia Institute of Technology
Atlanta, GA, USA
azhang490@gatech.edu

Ruth Nelson
Yale University
New Haven, CT, USA
ruth.nelson@yale.edu

Giorgio Ascoli
George Mason University
Fairfax, VA, USA
ascoli@gmu.edu

Liang Zhao*
Emory University
Atlanta, GA, USA
liang.zhao@emory.edu

ABSTRACT

Geometric trees are characterized by their tree-structured layout and spatially constrained nodes and edges, which significantly impacts their topological attributes. This inherent hierarchical structure plays a crucial role in domains such as neuron morphology and river geomorphology, but traditional graph representation methods often overlook these specific characteristics of tree structures. To address this, we introduce a new representation learning framework tailored for geometric trees. It first features a unique message passing neural network, which is both provably geometrical structure-recoverable and rotation-translation invariant. To address the data label scarcity issue, our approach also includes two innovative training targets that reflect the hierarchical ordering and geometric structure of these geometric trees. This enables fully self-supervised learning without explicit labels. We validate our method's effectiveness on eight real-world datasets, demonstrating its capability to represent geometric trees.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks; Unsupervised learning.**

KEYWORDS

Geometric Deep Learning, Self Supervised Learning, Graph Representation Learning

ACM Reference Format:

Zheng Zhang, Allen Zhang, Ruth Nelson, Giorgio Ascoli, and Liang Zhao. 2024. Representation Learning of Geometric Trees. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, August 25–29, 2024, Barcelona, Spain. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3637528.3671688>

*Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

KDD '24, August 25–29, 2024, Barcelona, Spain

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0490-1/24/08

<https://doi.org/10.1145/3637528.3671688>

1 INTRODUCTION

A geometric tree is a hierarchically arranged, tree-structured graph with nodes and edges that are spatially constrained, influencing their connectivity patterns. It is a particularly important data structure that is ubiquitous in different domains such as river geomorphology [5, 46], neuron morphology [2, 33], and vascular vessels [13]. Geometric trees are highly complex, non-linear structures and thus cannot be processed directly by common math and statistical tools. This makes representation learning on geometric trees a fundamental necessity in order to further apply them to downstream tasks such as classification, clustering, and generation. Although tree-structure representation learning has been extensively researched by techniques including sequence-based models, such as Tree-LSTM [39] and graph neural networks [6, 21, 22, 27], they are not able to jointly consider the geometric information that is coupled with the hierarchy and topology that are core properties of geometric trees. The importance of these attributes can be illustrated through real-life examples: as shown in Fig. 1(a), for a pyramidal neuronal cell, the closer to the cell body a branch is, the more curvature it exhibits. Similarly, as shown in Fig. 1(b), the node degree within a watershed's tree structure is indicative of the breadth of its corresponding subtree's expansion. On a theoretical level, as shown in Fig. 1(c), three geometric trees can be isomorphic if geometric information and hierarchy information about the levels from the root are not jointly considered.

Although recently some progress has been made with spatial graphs [28, 37, 50], they cannot be used to directly handle geometric trees as they overlook the hierarchical ordering of nodes and edges, which, as mentioned above, is crucial. Therefore, this paper focuses on developing a method that can learn the representations of geometric trees by preserving their geometric, topological, and hierarchical ordering, as well as their interplay. To achieve this, three aspects need to be addressed: **1. How do we represent the node and its geometric and topological context in geometric trees?** To address this, our study introduces a novel representation learning model named Geometric Tree Branch Message Passing (GTMP) that is specifically designed for geometric trees. At the core of our model is a unique message passing neural network that operates on the tree branches. The network is provably capable of preserving the entire geometric structure and ordering layout of trees, thereby ensuring an *information-lossless* representation. Additionally, GTMP inherently supports crucial symmetry invariant properties, including rotation and translation invariance, enhancing

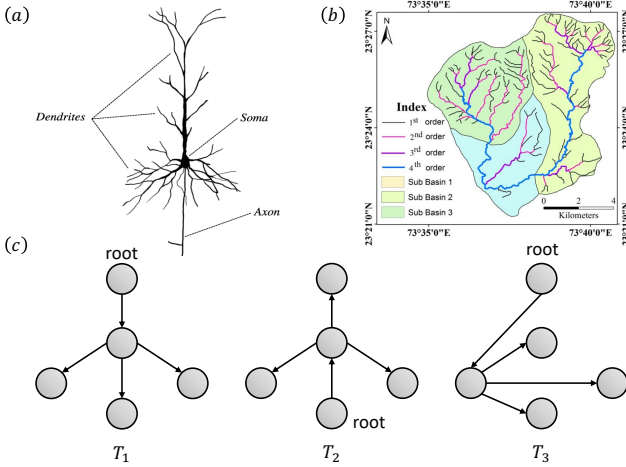


Figure 1: (a) Illustration of a neuron’s geometric tree-like structure; (b) Representation of a river network exhibiting a tree structure embedded within a geometric landscape; (c) Three different geometric trees with isomorphic network connectivity and identical spatial coordinates. Distinguishing these geometric trees requires jointly considering spatial, topology, and hierarchical layout information.

its applicability to a wide range of geometric tree analyses. **2. How do we reflect the hierarchical patterns of geometric trees in the learned representations?** Existing methods tend to be either fully permutable or non-permutable; however, the hierarchical nature of geometric tree structures requires a partially permutable pattern. This pattern requires the preservation of parent-child ordering, whereas the ordering among siblings, for instance, does not need to be as rigidly preserved. To tackle this issue, we introduce a partial ordering constraint module that enforces a strict directional relationship in the embedding space for parent-child pairs to reflect their hierarchical order. **3. How do we learn the representations with label scarcity?** Unsupervised or self-supervised learning is often used nowadays for geometric data (e.g., graphs, images, spatial, etc.) representation learning. However, the inductive bias for self-supervised learning needs to be customized to specific geometric data types, and those pertaining to geometric trees are significantly underexplored. To address this issue, a novel Geometric Tree Self-Supervised Learning (GT-SSL) framework is introduced through an innovative subtree-growing-guided objective, which aims to align the observed subtree and the expected subtree by its root.

To validate the effectiveness of our approach, comprehensive evaluations across eight real-world datasets have been conducted to demonstrate that our GTMP surpasses current leading geometric representation learning approaches by an average of over 15.6% in AUC scores. Furthermore, our proposed self-supervised learning strategy, GT-SSL, further improves the prediction performance by an additional 13.9% improvement on average. This underscores the effectiveness of our self-supervised learning approach in further refining the representations of geometric trees.

The remainder of this work is organized as follows: We begin by discussing existing relevant studies in the Section 2. This is followed by a formal problem definition in the Section 3. Subsequently, in the

Section 4, the proposed GTMP framework is described. We conclude with comprehensive experimental results, assessing aspects such as effectiveness and efficiency in Section 5.

2 RELATED WORK

2.1 Deep Learning for Tree-Structured Data

In recent years, there have been several methods proposed to more specifically handle the hierarchical nature of trees. For instance, the Tree-LSTM (Long Short-Term Memory) is an extension of the traditional LSTM architecture designed to handle hierarchical tree structures [39]. Each node in the tree has its own memory cell and gate mechanisms that control the flow of information, enabling the model to retain context and information over longer hierarchical distances. On the other hand, tree transformers integrate tree structures with more typical transformer-based architectures, usually by restricting attention heads to focus on specific constituents of the tree [45]. Tree-LSTM and tree transformers have been successfully applied to tasks involving tree-structured data, especially those in the NLP space [7, 23, 38]. However, these models only focus on the hierarchical nature of tree-structured data and are not explicitly designed for geometric trees that also include significant geometric and spatial data.

More generally, Graph Neural Networks (GNN) have also been adapted for use with tree-structured data. For instance, Qiao *et al.* [36] proposes T-GNN, a tree structure-aware graph neural network composed of two primary parts: an integrated hierarchical aggregation module, combining GNN with gated recurrent units, and a relational metric learning module to transform multi-hop neighborhood information. Hyperbolic GNNs (HGNN) leverage hyperbolic space to effectively model hierarchical and tree-like structures and capture long-range dependencies [31].

2.2 Spatial Networks

Research into spatial networks has been ongoing for decades [4], dating back almost 50 years to Haggett and Chorley’s work developing some of the first models to characterize spatial networks [19]. As improvements have been made with more complex networks [14], more modern quantitative advancements have emerged in fields such as transportation networks [3], urban mobility [8], biology [13], spatial social dynamics [26, 43], and computational chemistry [18].

As deep learning has progressed, numerous works have attempted to extend traditional deep learning methods to geometric data such as graphs [6, 12, 21, 22, 27, 30, 44, 49, 50]. Many existing graph representation methods, however, tend to overlook hierarchies and the interplay between spatial and topological properties. Deep learning has also been applied to spatial data separately, such as PointNet for 3D point clouds or 3D Convolutional Neural Networks (CNN) for 3D voxel grids [32, 35, 47], but there is limited work conjoining graph and spatial representations, especially when applied to geometric trees where both hierarchical and spatial information are crucial.

2.3 Self-Supervised Learning for Graph Data

Self-supervised learning has emerged as a powerful paradigm for learning from graph-structured data, enabling models to leverage

the rich, relational information inherent in graphs without requiring explicit labels. This approach has shown promise in extracting meaningful patterns and representations from graph data, pivotal across various domains such as social network analysis, chemical compound identification, and biological network interpretation [20, 42]. Central to self-supervised learning on graphs is the design of pretext tasks that encourage the model to uncover and exploit the underlying structure and properties of the graph. Techniques such as predicting the presence of edges, estimating node properties based on local neighborhood structures, or reconstructing graph segments serve as effective learning signals [24, 25].

However, existing self-supervised learning methods for graph-structured data often fall short in addressing the unique challenges of geometric tree self-supervised learning due to their limited focus on topology and neglect of geometric and hierarchical specificity. The unique hierarchical structures and the requirement of spatial invariance in geometric trees necessitate specialized approaches.

3 PRELIMINARIES AND PROBLEM FORMULATION

In this section, we first formalize geometric trees and the problem of representation learning on geometric trees, then we introduce the challenges with solving this problem.

Geometric trees (also known as spatial trees [4]) are tree-structured graphs for which the nodes and edges of the tree occupy positions in a Euclidean space. The geometric constraints may significantly affect their tree topological patterns. Geometric trees are ubiquitous in the real world, such as river systems [46], neurons [2], human blood vessels [13], and mobility networks [8], where geometric and tree-structured network properties are intricately linked. For example, the structure of nerve endings is often jointly determined by both their upstream neurons and the local chemical environment [9].

A geometric tree can be formally represented as $S = (T, P)$, where $T = (V, E)$ symbolizes the tree-structured graph. In this representation, V is the set of N nodes, and E , a subset of $V \times V$ with $|E| = |V| - 1$, represents the $N - 1$ edges. Each edge $e_{ij} \in E$ connects a *parent* node v_i to a *child* node v_j in V , establishing a hierarchical relationship where v_i is the parent of v_j . In this structure, starting from any node $v_i \in V$, it is impossible to traverse a *path* (e.g., $v_i \rightarrow v_{i1} \rightarrow v_{i2} \rightarrow v_i$) that forms a loop, ensuring the acyclic nature of the tree. A rooted tree, denoted as T_i , originates from any node v_i . If a node v_j is a *descendant* of node v_i , then its rooted tree T_j forms a *subtree* within the larger rooted tree T_i . In this hierarchical arrangement, node v_i is recognized as the *ancestor* of node v_j . The set P represents the spatial coordinates, defined as $P = \{(x_i, y_i, z_i) | x_i, y_i, z_i \in \mathbb{R}\}$, within the Cartesian coordinate system. For each node $v_i \in V$, its spatial position is denoted by the coordinate tuple $(x_i, y_i, z_i) \in P$.

The primary objective of this paper is to learn the representation $f(S)$ for geometric trees $S = (T, P)$, aiming to achieve a strong discriminative capability for unique geometric tree structures and to capture significant symmetry properties. This goal presents several unique challenges:

1) **It is difficult to jointly preserve both geometric and tree topological properties in geometric trees.** As shown in the

example Figure 1, the synergy between geometric attributes and tree topology is essential for generating distinguishable representations.

2) **It is difficult to embed geometric hierarchy patterns.** Existing works either consider or ignore node permutation. However, geometric tree hierarchy requires capturing them in some patterns (e.g., parent-child) but not for others (e.g., siblings).

3) **It is difficult to label sufficient geometric tree data.** The scarcity of labeled data in specialized areas, such as neuron data, necessitates alternative approaches such as self-supervised learning. However, specialized self-supervised learning strategies for geometric trees are underexplored.

4 METHOD

In order to develop a novel geometric tree representation learning method by addressing the challenges outlined above, we propose a new representation learning model named Geometric Tree Branch Message Passing (GTMP) to fully exploit the interplay between geometric and tree-topological structures. In addition, a novel Geometric Tree Self-Supervised Learning (GT-SSL) framework is introduced to extract the customized geometric tree properties without any supervision labels. Specifically, to discriminate geometric trees, especially for the spatial tree joint patterns, we propose a new message passing scenario that aggregates the geometric information via tree branches, which is elaborated on in Section 4.1. This scenario preserves the geometric structure of tree information with theoretical guarantees on the invariance to SE(3)-symmetric transformations and *spatial-information-lossless*. To address the issue of insufficient labels, we developed two self-supervised learning objectives that are tailored for intrinsic geometric tree structures. Specifically, to incorporate the underlying hierarchical relationships, a partial ordering constraint over the parent-child pair embeddings is introduced in Section 4.2. This implies that a node's embedding should maintain a clear directional relationship with its subtree nodes to accurately represent the hierarchical structure. To introduce geometric tree-specific inductive bias as a self-supervised learning target, we further propose a top-down subtree growth learning process. As discussed in Section 4.3, our goal is to align the observed geometric structure of the subtree with the structure anticipated by its root.

4.1 Tree Branch Geometric-Topology Information Representation Learning

To effectively tackle the complexities of geometric tree representation learning, we have first developed an innovative message-passing technique known as Geometric Tree Branch Message Passing (GTMP). As illustrated in Figure 2, this approach first harnesses the interplay between geometric properties and topological structures, enabling the computation of a comprehensive geometric-topology information representation for all branches originating from a tree node. Subsequently, we employ a neural network designed in a message-passing fashion, tailored specifically to account for the hierarchical ordering inherent in the tree branch structure. Our method systematically aggregates spatial information along an ordered tree branch. This strategy not only maintains the integrity of the tree's geometric structure but also assures robustness

against SE(3)-symmetric transformations, thereby enhancing the discriminative power of the process.

Formally, the spatial information of a geometric tree with N nodes can be expressed as a set of Cartesian coordinates $P = \{(x_i, y_i, z_i) | x_i, y_i, z_i \in \mathbb{R}\}_{i=1}^N$. It can also be represented as $P \in \mathbb{R}^{N \times 3}$ in a matrix form. The set of all *length n tree paths* starting from node v_i to its descendant nodes can be represented as π_n^i . In particular, a *length three branch* $v_i \rightarrow v_j \rightarrow v_k \rightarrow v_p$ can be expressed as $\pi_{ijkp} \in \Pi_3^i$, where v_i is the parent node of v_j , v_j is the parent node of v_k , and v_k is the parent node of v_p . Given a *length three branch* $\pi_{ijkp} \in \Pi_3^i$, the proposed spatial information representation can be expressed as

$$(d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}), \quad (1)$$

where

$$\begin{aligned} d_{ij} &= \|\mathbf{P}_{ij}\|_2, \quad d_{jk} = \|\mathbf{P}_{jk}\|_2, \quad d_{jp} = \|\mathbf{P}_{jp}\|_2, \\ \theta_{ijk} &= \arccos\left(\left\langle \frac{\mathbf{P}_{ij}}{d_{ij}}, \frac{\mathbf{P}_{jk}}{d_{jk}} \right\rangle\right), \quad \theta_{ijp} = \arccos\left(\left\langle \frac{\mathbf{P}_{ij}}{d_{ij}}, \frac{\mathbf{P}_{jp}}{d_{jp}} \right\rangle\right), \\ \varphi_{ijkp} &= \delta(\mathbf{n}_{ijk}, \mathbf{n}_{ijp}, \mathbf{P}_{ij}) \cdot \arccos\left(\left\langle \mathbf{n}_{ijk}, \mathbf{n}_{ijp} \right\rangle\right), \\ \mathbf{n}_{ijk} &= \frac{\mathbf{P}_{ij} \times \mathbf{P}_{jk}}{\|\mathbf{P}_{ij} \times \mathbf{P}_{jk}\|_2}, \quad \mathbf{n}_{ijp} = \frac{\mathbf{P}_{ij} \times \mathbf{P}_{jp}}{\|\mathbf{P}_{ij} \times \mathbf{P}_{jp}\|_2}, \end{aligned} \quad (2)$$

where $\delta(\mathbf{n}_{ijk}, \mathbf{n}_{ijp}, \mathbf{P}_{ij}) = \left\langle \frac{\mathbf{n}_{ijk} \times \mathbf{n}_{ijp}}{\|\mathbf{n}_{ijk} \times \mathbf{n}_{ijp}\|_2}, \frac{\mathbf{P}_{ij}}{\|\mathbf{P}_{ij}\|_2} \right\rangle$.

Theorem 1. The defined distances $d_{ij} \in [0, \infty)$, angles $\theta_{ijk} \in [0, \pi)$, and torsions $\varphi_{ijkp} \in [-\pi, \pi)$ exhibit rigorous invariance under any rotation and translation transformations $\mathcal{T} \in \text{SE}(3)$.

The proof of Theorem 1 is straightforward and provided in Appendix A. Notably, the representation formulated in Equation 1 not only achieves invariance to rotation and translation, but also simultaneously preserves essential information for reconstructing the original geometric tree structure under weak conditions.

Theorem 2. Given a geometric tree $S = (T, P)$, where T denotes a tree structure with a minimum depth of $\zeta \geq 3$, if the Cartesian coordinates for any set of three non-collinear, connected nodes (v_j, v_k, v_p) within a length three branch π_{ijkp} starting from node v_i are known, then the Cartesian coordinates P of the entire tree can be accurately determined. This determination is based on the representation outlined in Equation 1.

The proof to this theorem is a consequence of the following theorem, which is proved in Appendix A.

Theorem 3. Given Cartesian coordinates of three non-collinear connected nodes (v_j, v_k, v_p) in a length three branch π_{ijkp} of one node v_i , the Cartesian coordinate P_i of node v_i can be fully determined by the representation defined in Equation 1.

Proof of Theorem 2. As stated in Lemma 1, the Cartesian coordinate of node v_i can be determined by its connected neighbors v_j, v_k, v_p in the path of π_{ijkp} . Due to the property of strong connectivity of graph $G = (V, E)$, we can repeatedly solve the coordinate of a connected node to the set of nodes with known coordinates. Thus, starting from an arbitrary length three path, the Cartesian coordinates P of the whole spatial network can be determined. \square

Upon extracting geometric features as outlined in Equation 1, the next step involves creating a convolutional strategy. This strategy aims to merge tree-topology and spatial information derived from

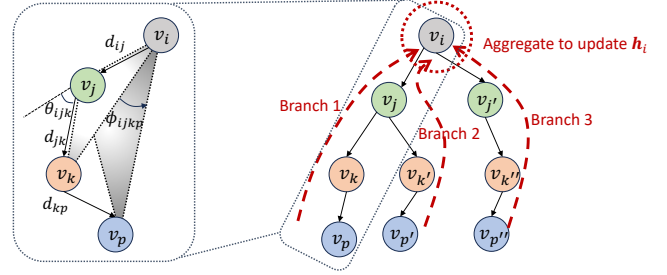


Figure 2: Illustration of the GTMP model. The geometric information is first extracted on each length-three branch starting from node v_i , then they are aggregated with other node information to update the node embedding h_i .

both the geometric representation of branches and their structural layouts into a comprehensive tree node representation. The essential challenge lies in preserving the model's ability to discriminate while also maintaining the integrity of the tree's structure and geometric details during the aggregation process.

To achieve this, we propose the following operation to update the hidden state embeddings h_i^ℓ by aggregating the message passing along all length three branches Π_3^i originating from node v_i :

$$h_i^{\ell+1} = \sigma^{(\ell)}\left(\text{AGG}\left(\{m^{(\ell)}(\pi_{ijkp}) | \pi_{ijkp} \in \Pi_3^i\}\right)\right), \quad (3)$$

where $\sigma^{(\ell)}$ is an arbitrary nonlinear transformation function (e.g. multilayer perceptron) and AGG denotes a set aggregation function.

In our model, the representation of a branch π_{ijkp} at layer ℓ integrates both topological and geometric information to produce a comprehensive message. The foundation of this approach lies in the aggregation of node features and the geometric configuration of the branch.

The integration of node features with geometric data is accomplished through a function $\phi^{(\ell)}$, which combines the aggregated node features $\tilde{m}^{(\ell)}$ with a transformation of the geometric information $\psi^{(\ell)}(\hat{m}(\pi_{ijkp}))$. The final message for the branch, $m^{(\ell)}(\pi_{ijkp})$, is thus given by:

$$\begin{aligned} m^{(\ell)}(\pi_{ijkp}) &= \phi^{(\ell)}\left(\tilde{m}^{(\ell)}(\pi_{ijkp}), \psi^{(\ell)}(\hat{m}(\pi_{ijkp}))\right), \\ \tilde{m}^{(\ell)}(\pi_{ijkp}) &= (h_i^{(\ell)}, \alpha_1 h_j^{(\ell)}, \alpha_2 h_k^{(\ell)}, \alpha_3 h_p^{(\ell)}), \\ \hat{m}(\pi_{ijkp}) &= (d_{ij}, d_{jk}, d_{jp}, \theta_{ijk}, \theta_{ijp}, \varphi_{ijkp}), \end{aligned} \quad (4)$$

where $\phi^{(\ell)}$ and $\psi^{(\ell)}$ are two nonlinear functions to extract the complicated coupling relationship between geometric and tree topology information. The aggregated node features, $\tilde{m}^{(\ell)}(\pi_{ijkp})$, are computed as a weighted combination of the features from node v_i and its descendants in order: v_j, v_k , and v_p . Here α_1, α_2 , and α_3 adjust the influence of each ancestor's features at layer ℓ .

Time Complexity. It is worth highlighting that while our specially designed branch message passing network architecture incorporates the aggregation of higher-order neighborhood information, it diverges significantly from other higher-order message passing neural networks in terms of computational complexity. Typically, these other networks exhibit a $O(|N|^Q)$ time complexity, where Q

represents the neighborhood order. In contrast, our message passing mechanism achieves a more efficient performance by maintaining linear time complexity $O(|N|)$ relative to the number of nodes. This key distinction underscores the efficiency and scalability of our approach, making it uniquely suited for scaling to large-size data without the exponential increase in computational demand often associated with higher-order processing.

4.2 Hierarchical Relationship Modeling through Partial Ordering Objective Function

To accurately represent the inherent hierarchical relationships among nodes in tree structures, we introduce a partial ordering constraint module. The fundamental concept behind this function is to constrain embeddings in such a way that the embedding of a node in the tree not only represents itself but also maintains a structured relationship over its subtree nodes in the embedding space.

To formally define the ordering constraint between node embeddings, we introduce the concept of partial ordering in the embedding space, ensuring that hierarchical relationships are accurately represented: as shown in Fig. 3, if T_j is a subtree of T_i , then the embedding h_j of node j has to be within the "lower-left" region of node i ' embedding h_i :

$$h_j[b] \leq h_i[b], \forall_{b=1}^D \quad \text{iff} \quad T_j \subseteq T_i, \quad (5)$$

where D is the dimension of hidden embeddings and $[b]$ denotes the b -th dimension of hidden embeddings.

To operationalize the above constraint into a function that can be optimized, we accordingly define the objective function for generating embeddings to utilize the max margin loss:

$$\mathcal{L}_{order} = \sum_{(h_i, h_j) \in \mathcal{P}} \max(0, h_j - h_i) + \sum_{(h_i, h_j) \in \mathcal{N}} \max(0, \delta - \|h_i - h_j\|^2), \quad (6)$$

where \mathcal{P} and \mathcal{N} denote the set of positive pairs and negative pairs in the minibatch where tree T_j is a subtree of tree T_i . The term δ represents a margin that enforces a minimum distance between the embeddings of negative pairs compared to positive pairs, ensuring that h_j (the embedding of the lower hierarchical node) is within the lower-left space to h_i (the embedding of the higher hierarchical node), and is at least a margin distance δ apart for negative pairs.

4.3 Self-Supervised Learning via Subtree Growth Learning

To tackle the challenge of insufficient training labels in real-world scenarios, we introduce an innovative Geometric Tree Self-Supervised Learning (GT-SSL) framework. While there are existing self-supervised learning objectives aimed at general graph data, they are insufficient when applied to geometric trees. The primary limitation is their inability to incorporate both intrinsic hierarchical relationships and coupled geometric-tree topology information into the self-supervised learning objectives. Our GT-SSL framework, by contrast, is specifically tailored to address these complexities, ensuring that the resulting representations fully reflect the unique structural and spatial characteristics of geometric trees.

Our approach focuses on growing the geometric tree's information from the root node. We introduce a unique subtree-growth

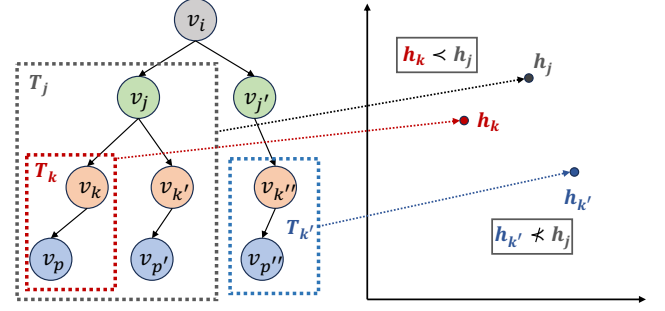


Figure 3: Illustration of the hierarchical relationship between tree nodes through partial ordering. In this scenario, T_k is a subtree of T_j , establishing a partial ordering relationship between their respective subtree embeddings. Conversely, since $T_{k'}$ does not constitute a subtree of T_j , there is no requirement for a partial ordering relationship between the embeddings of these two subtrees.

learning goal, which generates the entire geometric structure from top to bottom. This process unfolds iteratively, with each step predicting the geometric configuration of a node's subtree based on the geometric structure of its ancestors. This approach is inspired by the observed natural growth patterns in geometric trees, where evolution occurs in a hierarchical manner, cascading from higher-level nodes down to lower-level ones. An illustrative example can be found in river systems, where the configuration of a tributary is largely influenced by the main river's structure and the characteristics of the surrounding geometric environment.

To formalize our approach, for a node v_i within a geometric tree T , we denote $\mathcal{C}(v_i)$ as the set of its child nodes and $\mathcal{A}(v_i)$ as the set of its ancestor nodes within the tree's hierarchy. The objective of our subtree growing process is to accurately predict the geometric structure of the child nodes, $\hat{\mathcal{G}}(\mathcal{C}(v_i))$, given both the geometric structures of the node itself and that of its ancestors. This can be mathematically expressed as:

$$\hat{\mathcal{G}}(\mathcal{C}(v_i)) = g(\mathcal{G}(v_i \cup \mathcal{A}(v_i))), \quad (7)$$

where g represents a learnable function designed to synthesize the geometric details of the child nodes based on the aggregated geometric information of v_i and its ancestors. In our study, we prioritize predicting essential geometric features such as the distances between a node and its child nodes as well as the angles between the parent node, the current node, and its child nodes.

Unfortunately, to feasibly represent the geometric structure information as \mathcal{G} , a significant challenge arises from the variable number of child nodes associated with any given node in a tree-structured graph, complicating the prediction of these geometric features. Simple aggregated indicators, such as averaging the distances, could be employed, but they risk obscuring the comprehensive geometric structure of the child nodes.

To overcome this limitation, we introduce an approach that involves converting the geometric features into the frequency domain. This transformation enables us to focus on predicting the frequency distribution of the geometric features across the child nodes, rather than attempting to directly predict specific values of the geometric features, thus avoiding the issue posed by the uncertain number of child nodes.

Formally, to represent the geometric information in a form that is amenable to pattern recognition and prediction, we expand these geometric features into the frequency space with radial basis functions. Mathematically, for a given node v_i with child nodes $C(v_i) = \{v_{i1}, v_{i2}, \dots, v_{in}\}$, the distances $\{d_{i1}, d_{i2}, \dots, d_{in}\}$ are expanded as follows:

$$e_k(v_i) = \sum_{v_{ij} \in C(v_i)} \exp(-\gamma \|d_{ij} - \mu_k\|^2), \quad (8)$$

where e_k denotes the k -th radial basis and μ_k is the corresponding distances. To obtain the distribution of geometric features over the radial basis, the ground truth distribution can be denoted as:

$$\mathcal{G}(C(v_i)) = \left[\sum_{v_{ij} \in C(v_i)} e_k(v_i) \right]_{k=1}^K, \quad (9)$$

where K is the total number of radial basis functions employed. Thus, the estimated distribution can be written as:

$$\hat{\mathcal{G}}(C(v_i)) = g \left(\left[\sum_{v_{ij} \in \mathcal{A}(v_i) \cup v_i} e_k(v_i) \right]_{k=1}^K \right). \quad (10)$$

Subsequently, we formulate the objective function using Earth Mover's Distance (EMD) to measure the discrepancy between the estimated distribution $\hat{\mathcal{G}}(C(v_i))$ and the ground truth distribution $\mathcal{G}(C(v_i))$:

$$\mathcal{L}_{generative} = \sum_{v_i \in V} \text{EMD} \left(\hat{\mathcal{G}}(C(v_i)), \mathcal{G}(C(v_i)) \right), \quad (11)$$

where EMD denotes the Earth Mover's Distance to quantify the cost of transforming the estimated distribution into the ground truth distribution.

Finally, the overall self-supervised learning objective function can be written as the combination of the subtree generative objective and the partial ordering function:

$$\mathcal{L}_{\text{GT-SSL}} = \mathcal{L}_{generative} + \mathcal{L}_{order} \quad (12)$$

5 EXPERIMENTS

In this section, the experimental settings are introduced first in Section 5.1, then the performance of the proposed method on the eight real-world datasets are presented in Section 5.2. We further present the robustness test on our CL method against topological structure noise in Section ?? We verify the transfer ability of framework through different datasets in Section 5.3. The effectiveness of proposed components are measured through ablation studies in Section 5.4. In addition, we measure the parameter sensitivity in Section 5.5 and running time analysis in Section 5.7. All experiments are conducted on a 64-bit machine with four NVIDIA A4000 GPUs (16 GB GDDR5). The proposed method is implemented with PyTorch [34] and the PyTorch-Geometric [15] deep learning framework. The code to our work can be found in the Github repository: https://github.com/rollingstonezz/KDD24_geometric_trees.

5.1 Experimental Settings

Datasets. To evaluate the performance of our proposed GTMP and comparison methods in real-world scenarios, eight geometric tree datasets are used in our experiments, which includes five neuron morphology datasets and three river flow network datasets across multiple tasks.

1) Neuron morphology: We conducted classification experiments using neuronal morphology data from NeuroMorpho.org, the largest online collection of 3D neural cell reconstructions contributed by hundreds of laboratories around the world [1]. Specifically, we constructed binary classification tasks between control cells and cells from two experimental conditions, 5xFAD and lipopolysaccharide injection (lps), where experimental condition was the target for prediction. All cells were mouse neural cells but tasks were split across three cell types: glia, interneurons (inter), and principal cells (pc). In total, we constructed five tasks: mouse 5xFAD glia versus mouse control glia; 5xFAD primary cells versus control primary cells; lps glia versus control glia; lps interneurons versus control interneurons; and lps primary cells versus control primary cells. A statistical description of the datasets is shown in Table 2.

2) River flow networks: We also conducted regression experiments using publicly available river flow network data from the United States Geological Survey's (USGS) National Hydrography Dataset (NHD) [17, 40]. We incorporated 2,231 river flow network tree samples with an average node count of 11,141 per tree. To facilitate prediction, we established three key geometric topology-coupled metrics as targets: the clustering coefficient, spatial diameter, and spatial radius.

Comparison Methods. To the best of our knowledge, there has been little previous work directly handling geometric trees. Several advanced spatial graph networks have been developed to address generic spatial networks; among these, SchNet [37], DimeNet [28], and SGMP [50] are some of the closest related works to our approach and were selected as comparison methods. Additionally, we benchmarked our approach against three prominent graph neural network (GNN) methods— GCN [27], GAT [41], and GIN [48]— and two spatial neural network (SNN) techniques, PointNet [35] and SpatialNet [10]. For GNN methods, Cartesian coordinates are provided as node attributes, while for SNN methods both node attribute and graph connectivity information are incorporated to ensure an equitable comparison. Further details on the benchmark models and comparison methodology are available in Appendix B.

Implementation Details. In the supervised learning configuration, all models utilize an identical architecture comprising three convolutional layers with the hidden dimension size set to 64. During the self-supervised representation learning pretraining phase, this architecture is maintained with three convolutional layers leading to final embeddings of 64 dimensions. For subsequent fine-tuning on specific tasks, a three-layer Multilayer Perceptron (MLP) is appended to the convolutional base, utilizing ReLU activation functions to enhance non-linear processing capabilities. To ensure a balanced evaluation across our proposed message passing mechanism and other GNN methods under comparison, we standardized the hyperparameter selection process. Detailed specifications of this process are delineated in Appendix B, providing transparency and facilitating reproducibility in our experimental setup.

| | | Neuron - Classification (\uparrow) | | | | | River - Regression (\downarrow) | | |
|------------|------------|--|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|---|--|
| | | lps-glia | lps-inter | lps-pc | 5xfad-glia | 5xfad-pc | μ | D | r |
| GCN | Supervised | 0.6063 \pm 0.0235 | 0.5000 \pm 0.0000 | 0.5000 \pm 0.0000 | 0.7934 \pm 0.0218 | 0.7543 \pm 0.0611 | 0.1134 \pm 0.0244 | 172.1041 \pm 2.8023 | 87.6032 \pm 1.2830 |
| | GT-SSL | 0.9798 \pm 0.0197 | 0.9568 \pm 0.0134 | 0.9546 \pm 0.0122 | 0.8131 \pm 0.0111 | 0.8938 \pm 0.0134 | 0.0941 \pm 0.0321 | 95.5785 \pm 2.3104 | 76.4080 \pm 1.5892 |
| | diff (+/-) | 37.35% | 45.68% | 45.46% | 1.97% | 13.95% | 17.01% | 44.46% | 12.77% |
| GIN | Supervised | 0.6060 \pm 0.0326 | 0.5173 \pm 0.0388 | 0.5351 \pm 0.0355 | 0.8204 \pm 0.0083 | 0.7903 \pm 0.0407 | 0.1872 \pm 0.0150 | 172.5895 \pm 3.6765 | 87.4989 \pm 1.0998 |
| | GT-SSL | 0.9784 \pm 0.0147 | 0.9181 \pm 0.0237 | 0.7943 \pm 0.1235 | 0.8046 \pm 0.0258 | 0.9343 \pm 0.0109 | 0.1135 \pm 0.0819 | 102.9059 \pm 2.5516 | 73.3257 \pm 1.3291 |
| | diff (+/-) | 37.24% | 40.08% | 25.92% | -1.58% | 14.4% | 39.37% | 40.38% | 16.20% |
| GAT | Supervised | 0.5878 \pm 0.0411 | 0.5221 \pm 0.0231 | 0.5000 \pm 0.0000 | 0.8101 \pm 0.0232 | 0.6576 \pm 0.0981 | 0.2335 \pm 0.0731 | 171.1645 \pm 2.2507 | 89.0507 \pm 2.5466 |
| | GT-SSL | 0.5032 \pm 0.0048 | 0.5000 \pm 0.0000 | 0.5000 \pm 0.0000 | 0.8318 \pm 0.0177 | 0.8726 \pm 0.0390 | 0.2752 \pm 0.1201 | 141.8387 \pm 5.1076 | 76.5700 \pm 1.9842 |
| | diff (+/-) | -8.46% | -2.21% | 0% | 2.17% | 21.5% | -17.86% | 17.13% | 14.02% |
| PointNet | Supervised | 0.6308 \pm 0.0835 | 0.7295 \pm 0.0310 | 0.5508 \pm 0.0431 | 0.8108 \pm 0.0078 | 0.8960 \pm 0.0140 | 0.1244 \pm 0.0102 | 169.0530 \pm 2.0981 | 85.9079 \pm 2.4306 |
| | GT-SSL | 0.9733 \pm 0.0102 | 0.9543 \pm 0.0140 | 0.6432 \pm 0.0741 | 0.7707 \pm 0.0316 | 0.9771 \pm 0.0111 | 0.0754 \pm 0.0107 | 94.3706 \pm 2.7810 | 71.4937 \pm 1.8721 |
| | diff (+/-) | 34.25% | 22.48% | 9.24% | -4.01% | 8.11% | 39.39% | 44.18% | 16.78% |
| SpatialNet | Supervised | 0.7300 \pm 0.0432 | 0.7445 \pm 0.0419 | 0.5354 \pm 0.0796 | 0.8614 \pm 0.0113 | 0.8243 \pm 0.0353 | 0.2335 \pm 0.0418 | 151.0569 \pm 2.5466 | 86.2461 \pm 1.2461 |
| | GT-SSL | 0.5010 \pm 0.0025 | 0.5003 \pm 0.0007 | 0.9232 \pm 0.0460 | 0.8972 \pm 0.0150 | 0.9764 \pm 0.0164 | 0.0353 \pm 0.0381 | 91.6590 \pm 3.2311 | 66.9319 \pm 3.1098 |
| | diff (+/-) | -22.9% | -24.42% | 38.78% | 3.58% | 15.21% | 84.88% | 39.32% | 22.39% |
| SchNet | Supervised | 0.7561 \pm 0.0319 | 0.7597 \pm 0.0286 | 0.6178 \pm 0.0509 | 0.8994 \pm 0.0105 | 0.9502 \pm 0.0162 | 0.0342 \pm 0.0072 | 154.4101 \pm 3.2536 | 82.6100 \pm 1.2830 |
| | GT-SSL | 0.9698 \pm 0.0120 | 0.9393 \pm 0.0231 | 0.9612 \pm 0.0202 | 0.9046 \pm 0.0092 | 0.9893 \pm 0.0083 | 0.0112 \pm 0.0050 | 85.0109 \pm 2.3832 | 37.8552 \pm 1.4879 |
| | diff (+/-) | 21.37% | 17.96% | 34.34% | 0.52% | 3.91% | 67.25% | 44.94% | 54.18% |
| DimeNet | Supervised | 0.7049 \pm 0.0620 | 0.8338 \pm 0.0226 | 0.5601 \pm 0.1020 | 0.9123 \pm 0.0089 | 0.9544 \pm 0.0120 | 0.0196 \pm 0.0053 | 134.4048 \pm 2.4952 | 80.9102 \pm 1.2827 |
| | GT-SSL | 0.9351 \pm 0.0194 | 0.9627 \pm 0.0176 | 0.9345 \pm 0.0131 | 0.9540 \pm 0.0059 | 0.9902 \pm 0.0015 | 0.0077 \pm 0.0033 | 80.2417 \pm 2.0114 | 35.3140 \pm 0.9910 |
| | diff (+/-) | 23.02% | 12.89% | 37.44% | 4.17% | 3.58% | 60.71% | 40.30% | 56.35% |
| SGMP | Supervised | 0.7599 \pm 0.0442 | 0.8078 \pm 0.0382 | 0.6231 \pm 0.0512 | 0.8917 \pm 0.0123 | 0.9839 \pm 0.0034 | 0.0087 \pm 0.0031 | 123.3789 \pm 3.8385 | 68.0974 \pm 2.1300 |
| | GT-SSL | 0.9568 \pm 0.0170 | 0.9709 \pm 0.0142 | 0.9952 \pm 0.0009 | 0.9333 \pm 0.0085 | 0.9814 \pm 0.0072 | 0.0033 \pm 0.0012 | 76.7912 \pm 1.4728 | 36.0287 \pm 1.3890 |
| | diff (+/-) | 19.69% | 16.31% | 37.21% | 4.16% | -0.25% | 62.07% | 37.76% | 47.09% |
| GTMP | Supervised | 0.7996 \pm 0.0392 | 0.8529 \pm 0.0370 | 0.6560 \pm 0.0621 | 0.9417 \pm 0.0123 | 0.9887 \pm 0.0063 | 0.0052 \pm 0.0024 | 125.4951 \pm 2.8295 | 61.2353 \pm 1.3177 |
| | GT-SSL | 0.9836 \pm 0.0096 | 0.9996 \pm 0.0106 | 0.9872 \pm 0.0013 | 0.9011 \pm 0.0192 | 0.9992 \pm 0.0004 | 0.0041 \pm 0.0019 | 76.7699 \pm 1.8740 | 33.0287 \pm 0.7680 |
| | diff (+/-) | 18.4% | 11.87% | 33.12% | -4.06% | 1.05% | 21.15% | 38.83% | 46.06% |

Table 1: The main experimental results on neuron morphology and river flow network datasets. Here we present the performance of our GTMP method alongside other comparative methods within both supervised learning and our GT-SSL training approach. For each dataset, we highlight in bold both the best performance in the supervised learning context across all methods, and also the top performer within our GT-SSL training framework. Additionally, we show the percentage improvement in performance of all methods when leveraging our GT-SSL over traditional supervised learning settings. Specifically for the river flow network dataset, we use μ to represent the clustering coefficient, D for spatial diameter, and r for spatial radius.

We executed each experiment five times, subsequently averaging the results and computing the standard deviation. We adopted AUC score as the evaluation metric for the classification tasks on the neuron datasets due to the imbalanced distributions of all classes. On all runs in all tasks, the datasets were randomly divided into training, validation, and test sets with an 80:10:10 ratio, and identical hyperparameters were employed across all tasks for each dataset, except for the random seed that was responsible for the data split.

5.2 Effectiveness Results

In this section, we first assess the performance of our GTMP method against competing approaches across the real-world datasets within a supervised learning manner. Additionally, we explore the efficacy of our GT-SSL framework to evaluate the quality of the generated representations in a pretrain-finetune manner over the same dataset. Given that the GT-SSL framework is designed to be a general approach applicable to various representation learning methods, we present the outcomes for both our GTMP approach and all other methods being compared. In this section, we pretrain and fine-tune the same datasets for implementing the GT-SSL framework, ensuring a fair comparison with the supervised learning results.

The comparison of AUC scores for the neuron morphology datasets and MAE results for the river datasets is provided in Table 1. We summarize our observations on the effectiveness of the GTMP model and the GT-SSL training framework below:

- (1) **Strength of GTMP model in learning effective geometric tree representations.** The supervised learning results demonstrate the strength of our proposed method, which consistently achieved the best results in seven out of eight datasets and securing the second-best result in the only dataset where the best performance was not attained. Specifically, our results outperformed the other comparison models by over 12.7% on average for the neuron morphology datasets and 22.1% on average for the river flow network datasets. The outcomes demonstrate that our GTMP model successfully leverages the specially designed branch message passing mechanism to generate representations, which significantly enhances performance on downstream supervised learning tasks.
- (2) **Benefits of utilizing GT-SSL framework to enhance the quality of geometric tree representations.** Table 1 reveals that the GT-SSL’s pretrain-finetune approach consistently enhances the performance of all representation learning models on 62 of the 72 total prediction tasks when compared to traditional supervised learning settings. Notably, the GT-SSL method surpasses supervised learning by an average margin of over 23.02% across all tasks. This substantial improvement underscores the effectiveness of the GT-SSL pretraining framework in significantly enhancing the quality of learned representations via self-supervised learning objectives tailored to geometric tree structures.
- (3) **Integrating the GTMP model with the GT-SSL framework results in superior overall performance.** It is worth noting

| | | Source | | | | |
|--------------------|------------|---------------|---------------|---------------|---------------|---------------|
| | | lps-glia | lps-inter | lps-pc | 5xfad-glia | 5xfad-pc |
| # of Trees | | 28,687 | 8,092 | 28,224 | 33,757 | 28,086 |
| Average # of Nodes | | 2,025 | 4,488 | 3,146 | 1,994 | 3,152 |
| Target | lps-glia | <u>0.9836</u> | 0.9983 | 0.9987 | 0.9231 | 0.9999 |
| | lps-inter | 0.9111 | <u>0.9996</u> | 0.9936 | 0.8627 | 0.9999 |
| | lps-pc | 0.9995 | 0.9964 | <u>0.9872</u> | 0.9705 | 0.9969 |
| | 5xfad-glia | 0.9357 | 0.9981 | 0.9399 | <u>0.9011</u> | 0.9740 |
| | 5xfad-pc | 0.9798 | 0.9969 | 0.9959 | 0.8993 | 0.9992 |

Table 2: Transfer learning results. We underline the results where the source and target datasets are the same. Additionally, we highlight the best results for each target dataset.

that the combination of our proposed GTMP model and GT-SSL framework shows a more competitive performance than any other combination of methods by achieving the best performance in six out of eight datasets. Specifically, our results outperformed the other comparison models by over 10.0% on average for the neuron morphology datasets and 29.3% on average for the river flow network datasets. The results demonstrate that integrating the GTMP model with the GT-SSL framework can successfully lead to state-of-the-art representation learning performance on geometric tree datasets.

(4) Advantage of specialized spatial network representation learning methods over conventional graph and spatial neural networks. It is also worth noting that the specialized spatial network representation learning methods (SchNet, DimeNet, SGMP, and GTMP) show a more competitive performance than both the vanilla graph neural network-based methods (GCN, GIN, and GAT) and the spatial network-based methods (PointNet and SpatialNet). Specifically, these specialized methods surpass traditional graph neural networks by an average of over 19.3% in supervised learning contexts and 22.3% when integrated with the GT-SSL framework; against spatial neural network approaches, they demonstrate an average improvement of 11.1% in supervised settings and 23.7% with GT-SSL. These results indicate that standard graph neural network and spatial neural network methods have limited capability to effectively discriminate patterns that require joint consideration of geometric information and tree topological information.

5.3 Transfer Ability Analysis

We further investigate the transferability of our GT-SSL framework. In practical settings, the ability to deploy a model trained on one dataset to a new, unseen dataset without requiring retraining is highly beneficial. This strategy aims to address two main goals: (1) overcome the obstacle posed by insufficient data in the new dataset, which might hinder effective model training, and (2) save computational resources, as developing a model from the ground up demands significant time and resources. To assess how well our model adapts to new datasets, we initially pretrain the GTMP model using self-supervised learning objectives on source datasets. Subsequently, we finetune this pretrained model on target datasets to evaluate its performance in downstream task predictions.

The experimental results and sizes of datasets are shown in Table 2. (1) **Strong transfer ability across different source and target datasets.** It is evident that the transfer model, when applied from the source to the target dataset, can achieve performance on par with, or in some instances even surpassing, the model directly

| | Neuron (\uparrow) | | | River (\downarrow) | |
|---------------|-----------------------|---------------|---------------|------------------------|----------------|
| | lps-glia | lps-inter | 5xfad-pc | Diameter | Radius |
| Supervised | 0.7996 | 0.8529 | 0.9887 | 125.4951 | 61.2353 |
| GT-SSL | 0.9836 | 0.9996 | 0.9992 | 76.7699 | 33.0287 |
| No Ordering | 0.9769 | 0.9922 | 0.9981 | 78.8122 | 33.8345 |
| No Generative | 0.8235 | 0.8834 | 0.9847 | 108.5359 | 60.5332 |

Table 3: Ablation study results. NO Ordering refers to a variant that removes the partial ordering constraint module. NO Generative refers to another variant that removes the subtree growth learning module. The best result of each dataset is highlighted in bold.

trained on the source dataset. Specifically, the discrepancy in average performance between scenarios where the source and target datasets are identical and those where they differ is a mere 0.91%. (2) **Correlation between tree sizes and transfer performance.** More importantly, our findings reveal that pretraining on datasets with larger average tree sizes can significantly enhance transfer performance on target datasets. In particular, the average performance when pretraining on datasets characterized by relatively larger tree sizes (such as lps-inter, lps-pc, and 5xfad-pc) surpassed that of datasets with smaller tree sizes (such as lps-glia and 5xfad-glia) by an average of 5.51%. Notably, the dataset with the largest average tree size, lps-inter, achieved an impressive average AUC score of 0.9979 across all datasets. These results underscore the ability of our proposed model to leverage larger dataset sizes for improving performance on unseen, relatively smaller-sized datasets. This capability presents a strategic advantage in addressing challenges related to data scarcity and computational constraints.

5.4 Ablation Studies

This paper primarily concentrates on exploring the fundamental question of how effectively representation learning can leverage uniquely designed properties of geometric trees. Here, we investigate the impact of the proposed two self-supervised learning components of GT-SSL framework. We first consider a variant **No Ordering** that removes the *Partial Ordering* module. To study the effectiveness of the proposed *subtree growth learning* module, we further construct a variant **No Generative** that removes the corresponding module. Due to length constraints, we only present the results of five real-world datasets in Table 3.

(1) Our full GT-SSL framework achieved the best performance on all five datasets. Specifically, the full model outperforms the variants **No Ordering** and **No Generative** by 10.3% on average. In turn, these two variants exceeded the performance of the supervised learning model by an average of 13.3%. These outcomes confirm that incorporating *partial ordering* and *subtree growth learning* modules significantly enhances geometric tree representation learning tasks. (2) The performance drops significantly when we remove the *subtree growth learning* module, in comparison to removing the partial ordering module, which may indicate that this module plays a more critical role in understanding the joint geometric and tree topological properties towards learning powerful representations.

5.5 Parameter Sensitivity Analysis

In this section we present the parameter sensitivity analysis for the GTMP model, focusing on the impact of three types of hyperparameters across eight datasets. Initially, we explore the effect of the

| | Neuron - Classification (\uparrow) | | | | | River - Regression (\downarrow) | | |
|-----------------------|--|-----------|--------|------------|----------|-------------------------------------|---------|---------|
| | lps-glia | lps-inter | lps-pc | 5xfad-glia | 5xfad-pc | μ | D | r |
| $\{\alpha\}_1$ | 0.9836 | 0.9996 | 0.9872 | 0.9011 | 0.9992 | 0.0041 | 76.7699 | 33.0287 |
| $\{\alpha\}_2$ | 0.9779 | 0.9999 | 0.9764 | 0.8890 | 0.9985 | 0.0038 | 77.3823 | 32.0424 |
| $lr=1 \times 10^{-3}$ | 0.9836 | 0.9996 | 0.9804 | 0.8984 | 0.9991 | 0.0042 | 76.7699 | 33.0287 |
| $lr=5 \times 10^{-4}$ | 0.9810 | 0.9958 | 0.9872 | 0.9011 | 0.9992 | 0.0041 | 76.9832 | 33.8315 |
| bs=16 | 0.9836 | 0.9996 | 0.9804 | 0.9003 | 0.9991 | 0.0045 | 76.7699 | 33.0287 |
| bs=64 | 0.9849 | 0.9996 | 0.9751 | 0.9011 | 0.9991 | 0.0041 | 77.5353 | 34.1242 |

Table 4: Parameter sensitivity analysis. Here $\{\alpha\} = \{\alpha_1, \alpha_2, \alpha_3\}$ denotes the hyperparameters to control the influence of each ancestor, lr is short for learning rate, and bs is short for batch size.

| # of Nodes | 1,000 | 2,000 | 3,000 | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 | 10,000 |
|----------------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|
| Execution Time | 1.905 | 3.209 | 4.850 | 6.272 | 8.102 | 9.826 | 11.371 | 13.248 | 15.462 | 17.084 |

Table 5: The efficiency analysis of the GTMP model, measured in seconds.

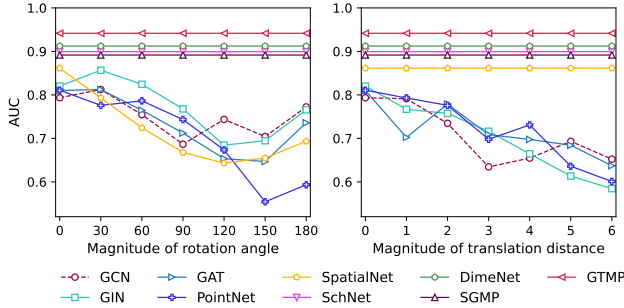


Figure 4: Robustness test on rotation and translation invariance by augmenting the data on the test set. The x-axis corresponds to the magnitude of the rotation angle (left) and translation distance (right) while y-axis shows the AUC scores. We can observe that our proposed GTMP model stays invariant to both translation and rotation transformations.

hyperparameter set $\{\alpha\} = \{\alpha_1, \alpha_2, \alpha_3\}$, which modulates the influence of each ancestor in Equation 4. The notation $\{\alpha\}_1$ signifies uniform weighting, achieved by setting each α value to 1, thereby distributing equal influence among ancestors. Conversely, $\{\alpha\}_2$ represents a decreasing weighting scheme, with $\alpha_1 = 1$ maintaining full influence, $\alpha_2 = 0.8$ indicating moderately reduced influence, and $\alpha_3 = 0.5$ depicting significantly diminished influence. Subsequently, we evaluate the model’s sensitivity to the initial learning rate, comparing the effects of setting it to 1×10^{-3} and 5×10^{-4} . Lastly, we assess the impact of selecting different batch sizes, as 16 and 64, on the model’s performance.

5.6 Rotation and Translation Invariant Test

Similar to prior research [16, 50], our study evaluates the robustness against rotation and translation by uniformly applying these transformations to the input Cartesian coordinates. Due to length constraints, we present accuracy results for the classification task solely on the neuron dataset lps-inter, although the findings are consistent across all datasets. As depicted in Figure 4, our model demonstrates remarkable invariance to both translation and rotation, maintaining stable performance under these transformations.

Comparable robustness is observed in models such as our proposed GTMP and comparison methods SchNet, DimeNet, and SGMP, attributable to their reliance on rotation and translation-invariant spatial features. In contrast, SpatialNet exhibits invariance to translation but not to rotation, as it employs only relative coordinates. This discrepancy underscores the critical advantage of incorporating rotation and translation-invariant features into models, as evidenced by the substantial performance decline of models lacking theoretical invariance when subjected to both rotation and translation transformations. This experiment highlights the significance of adopting models with built-in invariance to ensure consistent performance across varied spatial orientations and positions.

5.7 Efficiency Analysis

In this section, we further examine the efficiency of GTMP by measuring the relationship between the number of nodes in the trees, ranging from 1,000 to 10,000, and the execution time. The results are shown in Table 5. The latter reflects the training time for one epoch of 1,000 trees, averaged over 100 runs. The Pearson correlation coefficient between the number of nodes and the execution time is 0.999, and the p-value is 1.76×10^{-11} . These results indicate a strong linear time complexity of our proposed method with respect to the number of nodes, demonstrating its efficiency and scalability.

6 CONCLUSION

In this paper, we introduce the Geometric Tree Message Passing (GTMP) model, designed to efficiently learn coupled spatial-topology representations from geometric tree-structured data. Theoretical guarantees are given to assure its ability to preserve essential geometric structure information. To overcome the challenge of insufficient labeled data and to enhance transferability, we also introduce the Self-Supervised Learning Framework for Geometric Trees (GT-SSL). This framework significantly improves geometric tree representations by leveraging their inherent hierarchies and tree-oriented geometric structures. The integration of the GTMP model with the GT-SSL framework further accentuates its effectiveness, leading to state-of-the-art performance on various datasets.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation (NSF) Grant No. 1755850, No. 1841520, No. 2007716, No. 2007976, No. 1942594, No. 1907805, a Jeffress Memorial Trust Award, Amazon Research Award, NVIDIA GPU Grant, and Design Knowledge Company (subcontract number: 10827.002.120.04). The authors acknowledge Emory Computer Science department for providing computational resources and technical support that have contributed to the experimental results reported within this paper.

REFERENCES

- [1] Masood A Akram, Sumit Nanda, Patricia Maraver, Rubén Armañanzas, and Giorgio A Ascoli. 2018. An open repository for single-cell reconstructions of the brain forest. *Scientific data* 5, 1 (2018), 1–12.
- [2] Giorgio A Ascoli, Duncan E Donohue, and Maryam Halavi. 2007. NeuroMorpho. Org: a central resource for neuronal morphologies. *Journal of Neuroscience* 27, 35 (2007), 9247–9251.
- [3] Jayanth R Banavar, Amos Maritan, and Andrea Rinaldo. 1999. Size and form in efficient transportation networks. *Nature* 399, 6732 (1999), 130–132.
- [4] Marc Barthélemy. 2011. Spatial networks. *Physics Reports* 499, 1-3 (2011), 1–101.
- [5] Gary J Brierley and Kirstie A Fryirs. 2013. *Geomorphology and river management: applications of the river styles framework*. John Wiley & Sons.
- [6] Wenming Cao, Zhiyue Yan, Zhiqian He, and Zhihai He. 2020. A comprehensive survey on geometric deep learning. *IEEE Access* 8 (2020), 35929–35949.
- [7] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038* (2016).
- [8] Gerardo Chowell, James M Hyman, Stephen Eubank, and Carlos Castillo-Chavez. 2003. Scaling laws for the movement of people between locations in a large city. *Physical Review E* 68, 6 (2003), 066102.
- [9] Won-Suk Chung, Nicola J Allen, and Cagla Eroglu. 2015. Astrocytes control synapse formation, function, and elimination. *Cold Spring Harbor perspectives in biology* 7, 9 (2015), a020370.
- [10] Tomasz Danel, Przemysław Spurek, Jacek Tabor, Marek Śmieja, Łukasz Struski, Agnieszka Słowik, and Łukasz Maziarka. 2020. Spatial graph convolutional networks. In *International Conference on Neural Information Processing*. Springer, 668–675.
- [11] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 2018. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 195–205.
- [12] Yuanqi Du, Shiyu Wang, Xiaojie Guo, Hengning Cao, Shujie Hu, Junji Jiang, Aishwarya Varala, Abhinav Angirekula, and Liang Zhao. 2021. GraphGT: Machine Learning Datasets for Deep Graph Generation and Transformation. (2021).
- [13] Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. 2005. Scale-free brain functional networks. *Physical review letters* 94, 1 (2005), 018102.
- [14] Paul Erdos, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.
- [15] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
- [16] Fabian B Fuchs, Daniel E Worrall, Volker Fischer, and Max Welling. 2020. SE (3)-transformers: 3D roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503* (2020).
- [17] Geological Survey (U.S.). 2004. *National Hydrography Dataset*. U.S. Dept. of the Interior, U.S. Geological Survey, Reston, Va.
- [18] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 1263–1272.
- [19] Peter Haggett and Richard J Chorley. 1969. *Network analysis in geography*. Vol. 1. Hodder Education.
- [20] William L Hamilton. 2020. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, 3 (2020), 1–159.
- [21] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).
- [22] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).
- [23] Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449* (2019).
- [24] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. *International Conference on Learning Representations* (2020).
- [25] Longlong Jing and Yingli Tian. 2020. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE.
- [26] Cathleen Johnson and Robert P Gilles. 2003. *Spatial social networks*. Springer.
- [27] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [28] Johannes Klicpera, Janek Groß, and Stephan Günnemann. 2020. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123* (2020).
- [29] Christof Koch and Allan Jones. 2016. Big science, team science, and open science for neuroscience. *Neuron* 92, 3 (2016), 612–616.
- [30] Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. 2023. Deep graph representation learning and optimization for influence maximization. In *International Conference on Machine Learning*. PMLR, 21350–21361.
- [31] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. *Advances in neural information processing systems* 32 (2019).
- [32] Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 922–928.
- [33] Ruchi Parekh and Giorgio A Ascoli. 2013. Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron* 77, 6 (2013), 1017–1038.
- [34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).
- [35] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.
- [36] Ziyue Qiao, Pengyang Wang, Yanjie Fu, Yi Du, Pengfei Wang, and Yuanchun Zhou. 2020. Tree structure-aware graph representation learning via integrated hierarchical aggregation and relational metric learning. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 432–441.
- [37] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. 2017. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566* (2017).
- [38] Vighnesh Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems* 32 (2019).
- [39] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).
- [40] U.S. Geological Survey. 2019. USGS TNM Hydrography (NHD). <https://apps.nationalmap.gov/services/>. Accessed: June 7, 2019.
- [41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [42] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. *International Conference on Learning Representations* (2019).
- [43] Junxiang Wang, Junji Jiang, and Liang Zhao. 2022. An invertible graph diffusion neural network for source localization. In *Proceedings of the ACM Web Conference 2022*. 1058–1069.
- [44] Shiyu Wang, Xiaojie Guo, and Liang Zhao. 2022. Deep generative model for periodic graphs. *Advances in Neural Information Processing Systems* 35 (2022).
- [45] Yau-Shian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. *arXiv preprint arXiv:1909.06639* (2019).
- [46] Kelin X Whipple. 2004. Bedrock rivers and the geomorphology of active orogens. *Annu. Rev. Earth Planet. Sci.* 32 (2004), 151–185.
- [47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.
- [48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanije Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- [49] Zheng Zhang, Junxiang Wang, and Liang Zhao. 2023. Curriculum Learning for Graph Neural Networks: Which Edges Should We Learn First. *arXiv preprint arXiv:2310.18735* (2023).
- [50] Zheng Zhang and Liang Zhao. 2021. Representation learning on spatial networks. *Advances in Neural Information Processing Systems* 34 (2021), 2303–2318.

A MATHEMATICAL PROOFS

Theorem 1. *The defined distances $d_{ij} \in [0, \infty)$, angles $\theta_{ijk} \in [0, \pi)$, and torsions $\phi_{ijkp} \in [-\pi, \pi)$ exhibit rigorous invariance under any rotation and translation transformations $\mathcal{T} \in \text{SE}(3)$.*

Proof of Theorem 1. To demonstrate the invariance of distances, angles, and torsions with respect to translation transformations in Theorem 1, we note that these properties depend solely on relative coordinates, making them inherently immune to such transformations.

For rotation transformations characterized by R in $\text{SO}(3)$, the 3D rotation group, we begin by establishing two fundamental identities:

$$\begin{aligned} \langle Rx, Ry \rangle &= \langle x, y \rangle, \\ (Rx) \times (Ry) &= R(x \times y). \end{aligned}$$

These identities allow us to confirm the invariance under rotations as follows:

$$\begin{aligned} d_{ij} &= \|\mathbf{P}_{ij}\|_2 = \sqrt{\langle \mathbf{P}_{ij}, \mathbf{P}_{ij} \rangle} = \sqrt{\langle R\mathbf{P}_{ij}, R\mathbf{P}_{ij} \rangle}, \\ \theta_{ijk} &= \arccos \left(\frac{\langle \mathbf{P}_{ij}, \mathbf{P}_{jk} \rangle}{d_{ij}d_{jk}} \right) \\ &= \arccos \left(\frac{\langle R\mathbf{P}_{ij}, R\mathbf{P}_{jk} \rangle}{d_{ij}d_{jk}} \right), \\ \bar{\phi}_{ijkp} &= \arccos \left(\frac{\langle \mathbf{n}_{ijk}, \mathbf{n}_{jkp} \rangle}{\|\mathbf{n}_{ijk}\|_2 \|\mathbf{n}_{jkp}\|_2} \right) \\ &= \arccos \left(\frac{\langle R\mathbf{n}_{ijk}, R\mathbf{n}_{jkp} \rangle}{\|R\mathbf{n}_{ijk}\|_2 \|R\mathbf{n}_{jkp}\|_2} \right), \\ \delta(\mathbf{n}_{ijk}, \mathbf{n}_{ijp}, \mathbf{P}_{ij}) &= \frac{\langle \mathbf{n}_{ijk} \times \mathbf{n}_{ijp}, \mathbf{P}_{ij} \rangle}{\|\mathbf{n}_{ijk} \times \mathbf{n}_{ijp}\|_2 \|\mathbf{P}_{ij}\|_2} \\ &= \frac{\langle (R\mathbf{n}_{ijk}) \times (R\mathbf{n}_{ijp}), R\mathbf{P}_{ij} \rangle}{\|(R\mathbf{n}_{ijk}) \times (R\mathbf{n}_{ijp})\|_2 \|R\mathbf{P}_{ij}\|_2}. \end{aligned}$$

This proof confirms that all considered elements retain their values under both rotation and translation, thus proving the invariance as stated in Theorem 1. \square

Theorem 3. *Given Cartesian coordinates of three non-collinear connected nodes (v_j, v_k, v_p) in a length three branch π_{ijkp} of one node v_i , the Cartesian coordinate P_i of node v_i can be fully determined by the representation defined in Equation 1.*

Proof of Theorem 3. Let \mathbf{P}_j , \mathbf{P}_k , and \mathbf{P}_p denote the Cartesian coordinates of the nodes v_j , v_k , and v_p , respectively. Let \mathbf{P}_i denote the Cartesian coordinate of node v_i , which we seek to determine.

Given the distance d_{ij} between nodes v_i and v_j ; the angle θ_{ijk} between the vectors $\mathbf{P}_i - \mathbf{P}_j$ and $\mathbf{P}_k - \mathbf{P}_j$; the torsion angle $\bar{\phi}_{ijkp}$ defined by the plane containing \mathbf{P}_i , \mathbf{P}_j , \mathbf{P}_k ; and the plane containing \mathbf{P}_j , \mathbf{P}_k , \mathbf{P}_p :

1. The vector $\mathbf{v}_{ij} = \mathbf{P}_i - \mathbf{P}_j$ is such that $\|\mathbf{v}_{ij}\| = d_{ij}$.
2. The angle θ_{ijk} satisfies the equation:

$$\cos(\theta_{ijk}) = \frac{(\mathbf{P}_i - \mathbf{P}_j) \cdot (\mathbf{P}_k - \mathbf{P}_j)}{\|\mathbf{P}_i - \mathbf{P}_j\| \|\mathbf{P}_k - \mathbf{P}_j\|}.$$

3. The torsion angle $\bar{\phi}_{ijkp}$ involves the cross product of the normals to the planes $(\mathbf{P}_i, \mathbf{P}_j, \mathbf{P}_k)$ and $(\mathbf{P}_j, \mathbf{P}_k, \mathbf{P}_p)$:

$$\sin(\bar{\phi}_{ijkp}) = \frac{(\mathbf{n}_{ijk} \times \mathbf{n}_{jkp}) \cdot (\mathbf{P}_k - \mathbf{P}_j)}{\|\mathbf{n}_{ijk} \times \mathbf{n}_{jkp}\| \|\mathbf{P}_k - \mathbf{P}_j\|},$$

where $\mathbf{n}_{ijk} = (\mathbf{P}_i - \mathbf{P}_j) \times (\mathbf{P}_k - \mathbf{P}_j)$ and $\mathbf{n}_{jkp} = (\mathbf{P}_k - \mathbf{P}_j) \times (\mathbf{P}_p - \mathbf{P}_j)$.

To determine \mathbf{P}_i , we reverse-engineer these relationships starting from known quantities \mathbf{P}_j , \mathbf{P}_k , and \mathbf{P}_p . Given d_{ij} , we find all possible \mathbf{P}_i that satisfy the distance constraint. Among these, θ_{ijk} narrows the possibilities to a circle in the plane defined by \mathbf{P}_i , \mathbf{P}_j , and \mathbf{P}_k . Finally, $\bar{\phi}_{ijkp}$ selects a unique \mathbf{P}_i by specifying its orientation relative to \mathbf{P}_p .

Therefore, the coordinate \mathbf{P}_i can be uniquely determined by solving these geometric constraints simultaneously, leveraging the non-collinearity of \mathbf{P}_j , \mathbf{P}_k , and \mathbf{P}_p to ensure a unique solution exists. \square

Theorem 2. *Given a geometric tree $S = (T, P)$, where T denotes a tree structure with a minimum depth of $\zeta \geq 3$, if the Cartesian coordinates for any set of three non-collinear, connected nodes (v_j, v_k, v_p) within a length three branch π_{ijkp} starting from node v_i are known, then the Cartesian coordinates P of the entire tree can be accurately determined. This determination is based on the representation outlined in Equation 1.*

Proof of Theorem 2. As stated in Lemma 1, the Cartesian coordinate of node v_i can be determined by its connected neighbors v_j, v_k, v_p in the path of π_{ijkp} . Due to the property of strong connectivity of graph $G = (V, E)$, we can repeatedly solve the coordinate of a connected node to the set of nodes with known coordinates. Thus, starting from an arbitrary length three path, the Cartesian coordinates P of the whole spatial network can be determined. \square

B ADDITIONAL EXPERIMENTAL SETTINGS

B.1 Comparison Methods

This section provides details on the comparison models used in our study.

Graph Neural Networks (GNNs).

(i) **GIN.** Graph Isomorphism Networks (GIN), proposed by Xu et al. [48], is a type of GNN known for its powerful capability to discriminate graph structures within the class of 1-order GNNs;

(ii) **GAT.** Graph Attention Networks (GAT) [41] leverages multi-head attention layers to facilitate information propagation across the graph;

(iii) **GCN.** GCN [27] is a commonly used GNN model by using a localized first-order approximation of spectral graph convolutions;

Spatial Neural Networks (SNNs).

(i) **PointNet.** PointNet [35] processes pointwise features independently using multiple MLP layers and aggregates them into global features with a max-pooling layer;

(ii) **SpatialNet.** SpatialNet [11] is a spatial deep learning framework designed to learn globally aware 3D descriptors;

Spatial Networks.

(i) **SchNet.** SchNet [37] is a model tailored for spatial networks predictions. It applies a continuous filter function to model distances between nodes and their nearest neighbors;

(ii) **DimeNet.** DimeNet [28] is another model focused on spatial networks. It incorporates directional information by aggregating

messages from “length two paths”, using a physically informed representation of distances and angles;

(iii) **SGMP**. SGMP [50] is a deep learning approach tailored for spatial networks within Euclidean spaces. It enhances the model’s interpretability and accuracy by incorporating directional information. This is achieved through the aggregation of messages from paths of length three, utilizing a physically-grounded representation that includes distances, angles, and torsions to encode spatial relationships.

B.2 Searching Space for Hyper Parameters

Number of epochs trained: {50, 100, 200};

Size of each batch: {16, 32, 64};

Initial learning rate for model: { $1e-3$, $5e-3$, $1e-4$ };

Number of convolution layers: {3};

Dimension of hidden state: {64};

Weight decay ratio: {0.9} (if loss on validation set does not decrease for 10 epochs).

B.3 Example Visualization of Experimental Data

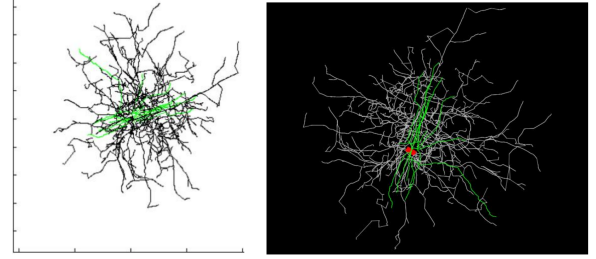


Figure 5: Images of example neural cell reconstruction from NeuroMorpho.org, which depict the reconstruction for the cell with NeuroMorpho.org ID NMO_86952 [2, 29]. The left image is a screenshot of the cell as viewed in the “Animation” feature on the cell’s corresponding NeuroMorpho.org page, and the right image is the cell’s representative image in the database.