Efficient and Effective Spatial Graph Representation Learning via Canonical Location Transformation

***Abstract*—Spatial graphs are graphs that are embedded in a (Euclidean) geometric space. As such, they represent twofold information, namely graph topology and nodes' geometric locations. Spatial graph representation learning (SGRL) assigns nodes of a spatial graph numerical vector representations (embeddings) so as to preserve nodes' similarity both in terms of graph topology and within the underlying geometric space. A major challenge in SGRL is to produce embeddings invariant to rotation and translation of the nodes within the geometric space. Existing approaches to translation-rotation-invariant SGRL suffer from effectiveness and efficiency issues. Also, they are end-to-end methods where the steps of translation-rotation-invariant location transformation and representation learning are strictly tied. This importantly limits their generalizability and extendability.**

In this paper, we advance the state of the art in SGRL by devising a methodology that transforms nodes' locations of a spatial graph into canonical edge features which are rotation-translation-invariant. Our methodology is principled, fast (it takes linear time in the size of the input graph), easy-to-implement, and backed by theoretical guarantees. It is fully general too, as it can be coupled with any (non-spatial) GRL method.

Extensive experiments conducted on a variety of real and synthetic datasets, and on node classification and regression tasks, attest high effectiveness and efficiency of the proposed approach, and its superiority over the state-of-the-art methods.

Reproducibility. ***Code will be made available upon acceptance.***

I. Introduction

Graphs, i.e., sets of entities (*nodes*) linked to one other (via *edges*), have become a ubiquitous model for representing real-world data from a plethora of domains. *Graph representation learning* (GRL), or *graph embedding*, automates the task of assigning elements of a graph (e.g., nodes, edges, subgraphs, entire graphs) numerical vectors – termed *embeddings* or *representations* – such that the *similarity* between those elements in the graph corresponds, as much as possible, to the *similarity* between their embeddings [25].

Spatial graphs are graphs that are embedded in a (Euclidean) geometric space [3]. As such, they represent twofold information, namely graph topology information, as described by nodes and edges, and spatial information, as represented by nodes' locations within the geometric space. *Spatial graph representation learning* (SGRL) assigns nodes of a spatial graph numerical vector representations (embeddings) so as to preserve nodes' similarity in terms of both graph topology and geometric locations. Spatial graphs and SGRL have been employed in a plethora of applications on a variety of real-world network data, including molecular-structure networks, biological networks, transportation networks, mobility networks [3], [4], [10], [14], [17], [21], [23]. The produced embeddings in SGRL are required to reflect similarity among nodes in terms of the twofold information encoded in the input spatial graph. Specifically, similarity among embeddings needs to reflect similarity among the corresponding nodes in terms of both graph-topology patterns and closeness between nodes' locations in the underlying geometric space. A major challenge in SGRL consists in guaranteeing identical embeddings for any two graphs which exhibit the same graph topology and such that their node locations are equal too, but just rotated and/or translated. This property is commonly referred to as rotation-translation invariance [23] (more formal definitions and details on this property in Sect. III).

Motivation. To date, the prominent state-of-the-art approach to SGRL is the *Spatial Graph Message Passing neural network* (SGMP) method, devised by Zhang *et al.* [23]. SGMP aggregates node-spatial information using higher-order edges to discern spatial graph patterns, effectively preserving graph and spatial information. SGMP ensures structural pattern preservation and rotation-translation invariance by utilizing length-p paths to represent node spatial information.

Although SGMP is principled and well-suited for SGRL, it comes with two main downsides. First, it is an end-to-end approach, where the phase of rotation-translation-invariance preservation is tied with the phase of representation learning (i.e., the phase of generating the embeddings). This makes it limited in terms of generalizability and extendability. As a second downside, SGMP has room for improvement in terms of tradeoff between effectiveness and efficiency. In fact, the exact methodology of SGMP has a time complexity of $\mathcal{O}(|V|^p)$, where V is the node set of the input spatial graph and p is the length of the paths considered. Even restricting $p = 3$ (as SGMP actually does), the resulting time complexity would be cubic in the number of nodes, thus prohibitive to handle moderate-to-large graphs. To overcome this, a path-sampling-based acceleration strategy is employed. Although the resulting accelerated method takes linear time in $|V|$, it exhibits limited effectiveness (as observed in our experiments in Sect. IV).

Contributions. In this paper, we advance the state of the art in SGRL by devising a novel approach that overcomes the above limitations. Our main intuition consists in decoupling the phases of rotation-translation-invariance preservation and representation learning, and devising a general rotation-translation-invariant methodology to handle the former only, so that it can be plugged into any existing non-spatial GRL to ultimately produce the embeddings. Specifically, we transform the original node locations of the input spatial graph into new canonical locations that are invariant to rotations and

translations. Rotation-translation invariance is ensured in such a way that the transformed node locations also preserve, as much as possible, the pairwise distances among the original node locations. This is important to keep the original geometric patterns unaltered and avoid trivial yet pointless transformations (such as, e.g., just set all the transformed location to a constant).

As our transformation is rotation-translation-invariant, applying any non-spatial GRL method on the transformed graph suffices to guarantee rotation-translation invariance for the resulting overall SGRL as well. This way, we eliminate the dependency on a specific GRL technique, which is beneficial as it makes the resulting overall approach to SGRL general, extendable, and prone to being up-to-date with the present and future advances in (GRL. This overcomes the first one of the downsides of the state of the art in SGRL discussed above. As for the second downside, the overall SGRL approach resulting from the upstream application of the proposed location-transformation methodology achieves the best tradeoff in terms of effectiveness and efficiency, consistently outperforming the SGRL state-of-the-art methods in this regard (cf. Sect. IV).

Technically speaking, our methodology transforms an input undirected spatial graph with geometric locations on nodes into a directed graph with edge features. The idea is to translate the original node locations into new canonical, rotation-translation-invariant locations that are placed on the edges of the transformed directed graph so that any directed edge (u, v) pertains to the spatial relationship between v and $\{u\} \cup N_u$ (where N_u is u's neighborhood). Such canonical edge features are computed inspired by a well-established linear algebra concept, i.e., deriving a new linear basis for the vectors representing the locations of $\{u\} \cup N_u$ so that the new basis vectors correspond to new dimensions that maximize the variance of the projected locations. However, this approach alone is insufficient, as identifying the dimensions that maximize variance is an optimization problem with multiple solutions. Thus, we need a sound "disambiguation" strategy that assigns always the same solution regardless of rotations/translations of the original locations. In this respect, we devise two disambiguation strategies which tradeoff differently among simplicity, effectiveness, and efficiency.

Summary. This paper provides the following contributions:

- We advance the state of the art in SGRL (Sect. II) by devising a general methodology to compute canonical rotation-translation-invariant locations (Sect. III), which can be coupled with any non-spatial GRL method, and allow for overcoming the limitations of the prominent existing SGRL methods in terms of generalizability, extendability and effectiveness-efficiency tradeoff.
- We provide theoretical properties of the proposed methodology, namely rotation-translation invariance, distance preservation, and linear execution time (Sect. III-A).
- We conduct extensive experiments whose results attest that the proposed approach achieves the best effectiveness-efficiency tradeoff when compared to the state-of-the-art GMP method [23] and other competitors (Sect. IV).

II. Preliminaries, Background, and Related Work

Problem statement. A *spatial (undirected) graph* is a graph whose nodes are assigned coordinates that make them located in a Euclidean geometric space [3]. Formally, a spatial graph is a triple $G = (V, E, \ell)$, where V is a set of nodes, $E \subseteq V \times V$ is a set of edges, and $\ell \colon V \to \mathbb{R}^d$ is a function that assigns each node a location in the d-D Euclidean space. We denote by $N_u = \{v \in V \mid (u, v) \in E\}$ the neighborhood of node $u \in V$. The problem tackled in this work is stated next.

Problem 1 (Spatial Graph Representation Learning (SGRL))**.** *Given a spatial graph $G = (V, E, \ell)$, and a natural number $q \in \mathbb{N}^+$, compute a real-valued matrix $R \in \mathbb{R}^{|V| \times q}$, where every row $R[u]$ corresponds to the* embedding *(or* representation*) of node u, for all $u \in V$. Each $R[u]$ encodes the role of u in G in terms of both graph topology as modeled by the edges E and spatial representation of the nodes V in the underlying Euclidean space.*

Problem 1's desiderata in encoding graph topology. In Problem 1, the encoding of graph topology is aimed at expressing the traditional broad objective GRL for non-spatial graphs, i.e., guaranteeing as much as possible that the "*similarity*" between nodes in the graph corresponds to the *similarity* between their embeddings [25]. The notion of similarity may capture node proximity or structural properties. Proximity-based approaches [6], [12], [15], [24] assign similar representations for nodes "physically" close in the graph. Conversely, structural similarity considers nodes' neighborhood structure (e.g., isomorphism among (k-hop) neighborhoods), regardless of their physical distance in the graph [8], [13]

Problem 1's desiderata in encoding spatial information. Encoding the spatial information means that the lower the (Euclidean) distance among node locations is, the more similar the embeddings should be. In ensuring this, a further critical requirement is to guarantee *invariance of the embeddings under rotation and translation* [23]. Yet, this invariance should be accomplished by also *preserving, as much as possible, the original pairwise distances among node locations*. This is essential to keep the original geometric patterns unaltered.

State of the art in SGRL and limitations. The state-of-the-art approach to SGRL is the *Spatial Graph Message Passing neural network* (SGMP) [23] method, which ensures rotation-translation invariance using *length-p* paths with distance, angle, and torsion features. However, even with $p = 3$, its time complexity is $\mathcal{O}(|V|^3)$. A random-spanning-tree-based sampling strategy reduces this to $\mathcal{O}(|V|)$, sacrificing optimality and overall effectiveness. SGMP has two key limitations: ($L1$) It is an end-to-end approach, coupling rotation-translation-invariance preservation with representation learning, limiting generalizability and extendability. ($L2$) Its $\mathcal{O}(|V|^3)$ time complexity is impractical for most real graphs. Acceleration reduces runtime but affects effectiveness (cf. Sect. IV).

We propose a general methodology that transforms node locations into a canonical edge features, ensuring rotation-translation invariance while mostly preserving node distances.

Algorithm 1 Canonical Spatial Graph Transformation

Input: Spatial graph $G = (V, E, \ell)$
Output: Directed graph $G' = (V, E', \ell_e)$ with features on edges

1: $E' \leftarrow \emptyset$; $\ell_e \leftarrow$ empty function
2: **for all** $u \in V$ **do**
3: $\quad \mathbf{L} \leftarrow$ empty matrix $\in \mathbb{R}^{(|N(a)|+1)\times d}$
4: $\quad$ Fill the first low $\mathbf{L}_1$ of matrix $\mathbf{L}$ with $\ell(u)$
5: $\quad$ Let $v_1, v_2, \ldots, v_{|N(u)|}$ be the neighbors of u sorted in some order
6: $\quad$ **for all** $i = 1, \ldots, |N(u)|$ **do**
7: $\quad\quad$ Fill the $(i+1)$-th row $\mathbf{L}_{i+1}$ of matrix $\mathbf{L}$ with $\ell(v_i)$
8: $\quad$ **end for**
9: $\quad \mathbf{L}' \leftarrow$ Canonical Spatial Mapping$(u, \mathbf{L})$ ▷ Alg. 2
10: $\quad$ **for all** $i = 1, \ldots, |N(u)|$ **do**
11: $\quad\quad E' \leftarrow E' \cup \{(u, v_i)\}$
12: $\quad\quad \ell_e(u, v_i) \leftarrow i$-th row of $\mathbf{L}'$
13: $\quad$ **end for**
14: **end for**
15: $G' \leftarrow (V, E', \ell_e)$

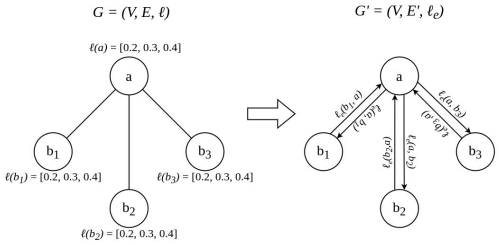

Fig. 1. A simple example of how an undirected spatial graph $G = (V, E, \ell)$ (left) is converted into a directed graph $G' = (V, E', \ell_e)$ with edge features (right) according to the proposed Canonical Spatial Graph Transformation algorithm (Alg. 1). In graph G (left), each node is labeled with its original coordinates, as specified by function ℓ ($d = 3$ in this example). In the produced graph G', each directed edge (u, v) from node u to node v is is assigned a 3-D feature vector corresponding to the transformed coordinates $\ell(u, v)$ of v with respect to the locations of $\{u\} \cup N_u$, based on the Canonical Spatial Mapping algorithm (Alg. 2).

Our approach overcomes SGMP's limitations. In fact, $(L1)$ it decouples invariance preservation from representation learning, allowing any (non-spatial) GRL method (that supports edge features) to be applied, ensuring generality, flexibility, and adaptability to future GRL advancements. $(L2)$, our (linear-time) methodology achieves the best tradeoff between effectiveness and efficiency, consistently outperforming SGMP.

Other (marginally) related work. Early approaches to SGRL treat spatial graphs as point clouds ignoring the graph structure [10], [14], or utilize off-the-shelf neural networks with Cartesian coordinates as inputs and a large amount of rotation-and translation-augmented data [4], [16], which is computationally expensive and lacks guarantee of rotation-translation invariance. Other existing SGRL approaches are domain-specific [2], [10], [5], [14], [19], preserve rotation-invariance only [18], or handle non-Euclidean spaces [22].

In this work, we deal with general-purpose SGRL that considers both graph structure and spatial node coordinates, and is translation- and rotation-invariant in a Euclidean space. Thus, all those SGRL approaches are marginally related to ours, as they are restrictive in at least one of these aspects.

III. Algorithms

Existing approaches to SGRL yields the embeddings of a spatial graph performing the phases of rotation-translation invariance (Phase 1) and representation learning (Phase 2) in an end-to-end, tied fashion. Here, we provide a novel perspective in tackling SGRL: we propose to decouple Phase 1 and Phase 2, and focus on designing a principled, efficient, and effective approach for Phase 1, so that any method for (non-spatial) GRL can be utilized for Phase 2. In the following, we provide the details of the proposed methodology for Phase 1.

From node coordinates to features on (directed) edges (Alg. 1). Our methodology transforms an input undirected spatial graph $G = (V, E, \ell)$ into a directed graph $G' = (V, E', \ell_e)$ with edge features, where E' is the set of directed edges that result from expanding any edge $(u, v) \in E$ into the two corresponding directed edges (u, v) and (v, u), i.e., $E' = E \cup \{(v, u) \mid (u, v) \in E\}$, and $\ell_e \colon E' \to \mathbb{R}^d$ is a function that assigns a d-dimensional feature vector to each edge in E'.

The idea is to have the edge features $\ell_e(u, v)$ of any directed edge $(u, v) \in E'$ represent the new location of node v transformed with respect to u and its neighborhood N_u. For any $u \in V$, the location $\ell_e(u, v)$ on each edge (u, v) outgoing from u represents the location of neighbor v of u transformed by taking the locations of $\{u\} \cup N_u$ as a reference. Equivalently, from an opposite point of view, our transformation assigns a set $\{\ell_e(u_1, v), \ldots, \ell_e(u_{|N_v|}, v))\}$ of new locations to any node $v \in V$, one per neighbor u_i of v, where every $\ell(u_i, v)$ corresponds to the new location computed in terms of u_i and N_{u_i}. This process is illustrated in Fig. 1 and outlined in Alg. 1. The algorithm first stacks the d-dimensional coordinate vectors of all the nodes $\{u\} \cup N_u$ as rows of a matrix $\mathbf{L}$ (Lines 3–8). Then, the vectors in $\mathbf{L}$ are transformed into the new coordinate vectors $\mathbf{L}'$ through the Canonical Spatial Mapping algorithm (Line 9). Finally, the directed edges of the newly constructed graph G' are assigned the transformed coordinates in $\mathbf{L}'$ (Lines 10–13). The way how the transformed locations $\mathbf{L}'$ are computed (Alg. 2) is described next.

Rotation-translation-invariant edge features: desiderata. Our key requirement to compute transformed locations $\mathbf{L}'$ in Alg. 1 (Line 9) is to guarantee invariance w.r.t. rotation and translation of the input graph. The meaning of this is informally explained next, while Sect. III-A discusses it more formally.

Let node u and the nodes of N_u be assigned spatial coordinates $\ell(u)$ and $\{\ell(v)\}_{v \in N_u}$, respectively. Let also x and N_x be replicas of u and N_u, respectively, whose assigned spatial coordinates $\ell(x)$ and $\{\ell(y)\}_{y \in N_x}$ result from applying arbitrary rotations and translations to the coordinates of u and u's neighbors. More precisely, let $t \colon \mathbb{R}^d \to \mathbb{R}^d$ a function that applies arbitrary rotations and translations to d-dimensional locations. It holds that $\ell(x) = t(\ell(u))$ and, for each $v \in N_u$, there exists one and only one $y \in N_x$ such that $\ell(y) = t(\ell(v))$. Now, for each $v \in N_u$, let $\ell_e(u, v)$ be the edge features corresponding to the location of v transformed according to u. Similarly, for each $y \in N_x$ let $\ell_e(x, y)$ be the edge features corresponding to the location of y transformed according to x. Rotation-translation invariance means that $\ell_e(u, v) = \ell_e(x, y)$, for all $v \in N_u$ and its counterpart $y \in N_x$, for all (compositions of) rotations and translations $t(\cdot)$, and for all u and N_u and their rotated-translated counterparts x and N_x.

Note that rotation-translation invariance may in principle be ensured with naïve and pointless methodologies: e.g., just assigning constant locations, regardless of the original ones. Hence, an additional important requirement in guaranteeing rotation-translation invariance is the preservation, as much as possible, of the distances between the original node locations.

Rotation-translation-invariant edge features: intuition. To yield rotation-translation-invariant edge features, we borrow a well-established idea in linear algebra, i.e., finding a new linear basis for the vectors representing the locations of $\{u\} \cup N_u$ such that the basis vectors of this new basis correspond to new dimensions that maximize the variance of the projected locations (principal components). This is the idea employed in the well-known Principal Component Analysis (PCA) technique. However, while PCA leverages principal components for dimensionality reduction, retaining only a subset of them, our objective is fundamentally different: we identify dimensions that remain consistent regardless of translation and rotation of the original locations. This ensures that the data are projected onto the same locations even when the spatial graph undergoes rotation-translation transformations. We refer to this process as Canonical Spatial Mapping. The key idea is that, given the location of a node and of its neighbors, computing the principal directions of maximum variance (i.e., the eigenvectors of the covariance matrix) yields dimensions that project the data onto a consistent spatial configuration.

However, as illustrated in Fig. 2 (commented below, while we describe the proposed disambiguation strategies), this approach alone is insufficient. The main challenge is that identifying the dimensions that maximize variance is an optimization problem with multiple solutions. PCA guarantees that these dimensions are the eigenvectors of the covariance matrix, but in our case, we must also consider the multiplicity of the associated eigenvalues. If an eigenvalue has a multiplicity of one, the optimal dimension is uniquely defined up to sign ambiguity–meaning that there are only two possible solutions: the eigenvector itself and its negation (i.e., the eigenvector multiplied by -1). In this case, a disambiguation is necessary to ensure a consistent selection. When an eigenvalue has a multiplicity more than one, there exist infinitely many valid dimensions, as any linear combination of the corresponding eigenvectors remains an optimal solution. Unlike PCA, where choosing any dimension within this space is irrelevant due to its focus on compression, our method requires selecting a single, consistent dimension per eigenvalue to ensure invariance. To address this, we introduce a disambiguation algorithm that systematically ensure unique and stable projections.

Rotation-translation-invariant edge features: algorithm (Alg. 2). The intuition just described are formalized in Alg. 2 (Canonical Spatial Mapping). At the beginning, each column of the input matrix $\mathbf{L}$ is subtracted the column mean, so as to get matrix $\mathbf{L}_0$ with all zero-mean columns (Line 1). This is a common practice in the operation of change of basis. Then, the eigenvalues λ_i of the covariance matrix $\mathbf{A}$ of $\mathbf{L}_0$ are computed (Lines 2–3), along with the corresponding eigenvectors $\mathbf{v}_{\lambda_i}$

Algorithm 2 Canonical Spatial Mapping

Input: Node u, real-valued matrix $\mathbf{L} \in \mathbb{R}^{(|N_u|+1)\times d}$
Output: Real-valued matrix $\mathbf{L}' \in \mathbb{R}^{|N_u|\times d}$

1: $\mathbf{L}_0 \leftarrow \mathbf{L} - columnMean(\mathbf{L})$
2: $\mathbf{A} \leftarrow$ covariance matrix of $\mathbf{L}_0$
3: $\lambda_{list} = [\lambda_1, \ldots, \lambda_{d'}] \leftarrow$ eigenvalues of $\mathbf{A}$ sorted in non-increasing order
4: $\mathbf{L}' \leftarrow$ empty matrix $\in \mathbb{R}^{|N_u|\times d'}$
5: **for all** $i = 1, \ldots, d'$ **do**
6: $\quad \mathbf{v}_{\lambda_i} \leftarrow$ set of eigenvectors of $\mathbf{A}$ with eigenvalue λ_i
7: $\quad \mathbf{c}_i \leftarrow$ Disambiguation$(u, \mathbf{L}_0, \mathbf{v}_{\lambda_i})$ or Disambiguation$^*(u, \mathbf{L}_0, \mathbf{v}_{\lambda_i})$ ▷Algs. 3, 4
8: $\quad$ Fill the i-th column of matrix $\mathbf{L}'$ with $\mathbf{c}_i$
9: **end for**

(Line 6). The multiplicity of the eigenvalue λ_i determines the number of the corresponding $\mathbf{v}_{\lambda_i}$ eigenvectors. However, even when λ_i has multiplicity 1, two valid eigenvectors exist, namely $\mathbf{v}$ and $-\mathbf{v}$. Thus, regardless of the eigenvector multiplicity, we need to "disambiguate" within multiple eigenvectors, i.e., a method to pick always the same eigenvector from possibly multiple ones. To accomplish this, we devise two strategies (described below): a simpler yet faster one (Disambiguation, Alg. 3), and an enhanced yet more computationally-demanding one (Disambiguation*, Alg. 4). One of these disambiguation strategies is employed in Canonical Spatial Mapping, so as to get the ultimate transformed vector $\mathbf{c}_i \in \mathbb{R}^{|N_u|}$ (Line 7) to be put as i-th column of the output matrix $\mathbf{L}'$ (Line 8).

Disambiguation (Alg. 3). The key idea of Alg. 3 is to sequentially try two different strategies to disambiguate the projected data $\mathbf{c} = \mathbf{L}_0 \cdot \mathbf{v}$ (Line 4), given only the eigenvector $\mathbf{v}$ (Line 3). If both strategies fail, to preserve translation-rotation invariance, all the locations are set to to $\mathbf{0}^{|N_u|}$ (Line 16).

The first strategy ensures that the projected location of u on the new dimension always results in positive coordinates. Specifically, if the projected location $\mathbf{c}[1]$ of the first neighbor of u on $\mathbf{v}$ is positive, the algorithm returns the projected locations $[\mathbf{c}[2], \ldots, \mathbf{c}[|N_u|+1]]^\top$ of the neighbor nodes N_u as they are (Line 6). Otherwise, if the projected location $\mathbf{c}[1]$ of node u is negative, the algorithm flips the vector of projected locations, returning $[-\mathbf{c}[2], \ldots, -\mathbf{c}[|N_u|+1]]^\top$ (Line 8). This guarantees that, if the locations undergo rotation or translation, the projected locations remain in the same relative position.

If $\mathbf{c}[1]$ is zero, there is still ambiguity. Thus, the second strategy is adopted, which enforces that the maximum absolute value among the projected neighbor locations of u on $\mathbf{v}$ is given by a positive value. To achieve this, the algorithm computes the maximum M and the minimum m of the projected neighbor locations (Line 10). The maximum absolute value of the projected neighbor locations on $\mathbf{v}$ corresponds to $\max\{|m|, M\}$. If $M > -m$ (Line 11), then $M > 0$, making the maximum absolute value equal to M, which is indeed positive. In this case, the algorithm returns the projected locations $[\mathbf{c}[2], \ldots, \mathbf{c}[|N_u|+1]]^\top$ of the neighbor nodes N_u as they are (Line 12). Conversely, if $M < -m$ (Line 13), then the maximum absolute value is $-m$, which is provided by a negative projected value. In this case, the algorithm flips the signs of the projected neighbor locations and returns $[-\mathbf{c}[2], \ldots, -\mathbf{c}[|N_u|+1]]^\top$. If $M = -m$, then the maximum

Algorithm 3 Disambiguation

Input: Node u, real-valued matrix $\mathbf{L}_0 \in \mathbb{R}^{(|N_u|+1)\times d}$ with zero-mean columns, eigenvectors $\mathbf{v}_\lambda$ of $\mathbf{L}_0$'s covariance matrix corresponding to eigenvalue λ
Output: Column vector $\mathbf{c} \in \mathbb{R}^{|N_u|}$

$\mathbf{c} \leftarrow \mathbf{0}^{|N_u|}$
if $|\mathbf{v}_\lambda| = 1$ **then**
  $\mathbf{v} \leftarrow$ the only eigenvector in $\mathbf{v}_\lambda$
  $\mathbf{c} \leftarrow \mathbf{L}_0 \cdot \mathbf{v}$
  if $\mathbf{c}[1] > 0$ **then**
    $\mathbf{c} \leftarrow [\mathbf{c}[2], \ldots, \mathbf{c}[|N_u|+1]]^\top$
  else if $\mathbf{c}[1] < 0$ **then**
    $\mathbf{c} \leftarrow [-\mathbf{c}[2], \ldots, -\mathbf{c}[|N_u|+1]]^\top$
  else
    $M \leftarrow \max_{i=2,\ldots,|N_u|+1} \mathbf{c}[i]$; $m \leftarrow \min_{i=2,\ldots,|N_u|+1} \mathbf{c}[i]$
    if $M > -m$ **then**
      $\mathbf{c} \leftarrow [\mathbf{c}[2], \ldots, \mathbf{c}[|N_u|+1]]^\top$
    else if $M < -m$ **then**
      $\mathbf{c} \leftarrow [-\mathbf{c}[2], \ldots, -\mathbf{c}[|N_u|+1]]^\top$
    else
      $\mathbf{c} \leftarrow \mathbf{0}^{|N_u|}$
    end if
  end if
end if

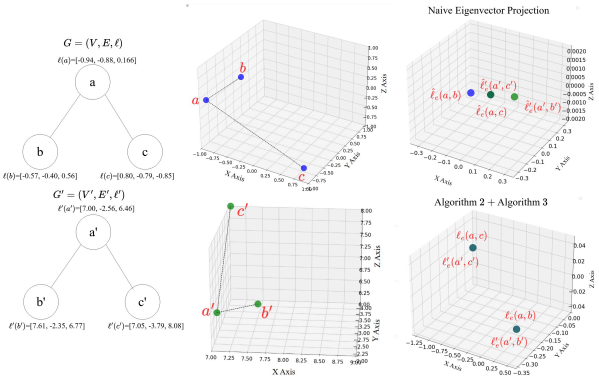

Fig. 2. Comparing the proposed methodology to compute rotation-translation-invariant locations (Alg. 2, equipped with the basic disambiguation strategy (Alg. 3) to the naïve eigenvector projection methodology where no disambiguation is employed. This comparison demonstrates that for a spatial graph G and its transformed version G' (which is identical to G but with added random rotations and translations), simply projecting the locations onto the eigenvectors (top-right figure) does not ensure invariance to these transformations, while this is accomplished by our methodology (bottom-right figure).

absolute value is provided by both a positive and a negative projected value. This still results in ambiguity, meaning that the second strategy fails too, and leading to outputting $\mathbf{0}^{|N_u|}$.

Fig. 2 illustrates the key role of our disambiguation. Graph $G = (V, E, \ell)$ has 3 nodes (i.e., $V = \{a, b, c\}$) and 3-D node locations. $G' = (V', E', \ell')$ is obtained by applying random translations and rotations to G, depicted in the second-column plots. V and V' are the same except for a renaming, e.g., from a to a'. As G and G' differ only in the rotation and translation of node locations, the resulting directed graphs with canonical edge features should be identical. As shown in the bottom-right plot, this holds when Alg. 2 and Alg. 3 are employed. However, without Alg. 3, when the data are projected directly onto the eigenvector without disambiguation, inconsistencies arise, as shown in the top-right plot, where $\hat{l}_e(a,b) \neq \hat{l}'_e(a',b')$.

Enhanced disambiguation (Alg. 4). If the eigenvalue multiplicity is more than one, Alg. 3 places all the projected neighbor locations at position zero. While this preserves the translation-rotation invariance property, it reduces the ability of any GRL

Algorithm 4 Disambiguation*

Input: Node u, real-valued matrix $\mathbf{L}_0 \in \mathbb{R}^{(|N_u|+1)\times d}$ with zero-mean columns, eigenvectors $\mathbf{v}_\lambda$ of $\mathbf{L}_0$'s covariance matrix corresponding to eigenvalue λ
Output: Column vector $\mathbf{c} \in \mathbb{R}^{|N_u|}$

1: $\mathbf{c} \leftarrow \mathbf{0}^{|N_u|}$
2: **if** $|\mathbf{v}_\lambda| = 1$ **then**
3: $\mathbf{c} \leftarrow$ Disambiguation$(u, \mathbf{L}_0, \mathbf{v}_\lambda)$ ▷ Alg. 3
4: **else**
5: $best = -\infty$; $l \leftarrow [\,]$
6: **for all** $i = 2, \ldots, |N_u|+1$ **do**
7: $\mathbf{c}[i-1] \leftarrow$ solution to Problem 2 with input $\langle \mathbf{L}_0, \mathbf{v}_\lambda, i \rangle$
8: **if** $\mathbf{c}[i-1] > best$ **then**
9: $best \leftarrow \mathbf{c}[i-1]$; $l \leftarrow [i]$
10: **else if** $\mathbf{c}[i-1] = best$ **then**
11: append i to l
12: **end if**
13: **end for**
14: $\mathbf{w}^* \leftarrow \frac{1}{|l|} \sum_{i \in l} \mathbf{L}_0[i]$ ▷ $\mathbf{L}_0[i] :=$ i-th row of $\mathbf{L}_0$
15: **for all** $i = 2, \ldots, |N_u|+1$ **do**
16: **if** solution to Prob. 3 with input $\langle \mathbf{L}_0, \mathbf{v}_\lambda, i, \mathbf{w}^*, best \rangle$ exists **then**
17: $\mathbf{c}[i-1] \leftarrow$ solution to Prob. 3 with input $\langle \mathbf{L}_0, \mathbf{v}_\lambda, i, \mathbf{w}^*, best \rangle$
18: **end if**
19: **end for**
20: **end if**

technique to distinguish different spatial structural patterns. To address this limitation, we propose an enhanced disambiguation procedure, described in Alg. 4, which is specifically designed for cases where the eigenvalue multiplicity is >1 (i.e., $|\mathbf{v}_\lambda| > 1$). In this scenario, the dimension that maximizes the variance of the projected locations can be any linear combination of the eigenvectors in $\mathbf{v}_\lambda$, provided that the resulting vector has a norm equal to 1. There exist infinitely many directions that satisfy this, making disambiguation way more complex.

The key idea of our enhanced disambiguation is to determine, for each neighbor location i of node u, the maximum projected value of i on any linear combination of the eigenvectors in $\mathbf{v}_\lambda$, subject to the following constraints: (i) the linear combination is a unit-norm vector, (ii) the projected location of u results in a value ≥ 0, and (iii) if feasible, the projection of the mean of the neighbor locations that achieve the maximum projected value, on the linear combination dimension produces the same maximum projected value. The maximum projection obtained for each neighbor location represents the disambiguated projections. This means that each location is projected onto a dimension that maximizes its projected value, which may differ from the dimensions where other locations are projected. This transformation maintains the invariance property and is much more informative than a zero vector.

More specifically, Alg. 4, starting from Line 5, computes the solution for the Auxiliary problem I (Line 7) defined as follows for each neighbor location (Line 6).

Problem 2 (Auxiliary problem I for Alg. 4)**.** *Given a matrix* $\mathbf{L}_0 \in \mathbb{R}^{(|N_u|+1)\times d}$*, a list* $\mathbf{v}_\lambda = [\mathbf{v}_j \in \mathbb{R}^d]_{j=1,\ldots,k}$ *of* k *eigenvectors, and an integer* $i \in [2, \ldots, |N_u|+1]$*, find*

$$h = \underset{\alpha_1,\ldots,\alpha_k \in \mathbb{R}}{\text{maximize}} \;\; \mathbf{L}_0[i] \cdot \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j$$

$$\textit{subject to} \qquad \mathbf{L}_0[1] \cdot \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j \geq 0; \;\; \left\| \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j \right\|_2 \leq 1.$$

Given a location i, the Auxiliary problem I for Alg. 4 computes the maximum projection value of i among all possible linear combinations of the eigenvectors in $\mathbf{v}_\lambda$, subject to the constraint that the norm of the linear combination is at most one, and that the projection of the location of node u is ≥ 0 on the selected dimension. This value is stored in $\mathbf{c}[i-1]$. Importantly, although it is required that the norm is at most one, the maximization of the projected location value will naturally result in the norm being *exactly one*. This optimization problem is a quadratic convex optimization one, which can be efficiently solved using the interior point algorithm in polynomial time in k and in linear time in d. Continuing with the description of Alg. 4, list l collects each id i of the locations with the maximum value of h from the Auxiliary problem I and stores in $best$ the highest value of h among all the locations (Lines 8—12). Then, in Line 14, the algorithm computes the mean of the locations contained in l that have the highest value of h in Line 7. For each neighboring location, if feasible, the algorithm formulates and solves a new optimization problem, defined as Auxiliary problem II for Alg. 4 (Lines 16—20).

Problem 3 (Auxiliary problem II for Alg. 4)**.** *Given a matrix* $\mathbf{L}_0 \in \mathbb{R}^{(|N_u|+1)\times d}$*, a list* $\mathbf{v}_\lambda = [\mathbf{v}_j \in \mathbb{R}^d]_{j=1,\ldots,k}$ *of* k *eigenvectors, an integer* $i \in [2,\ldots,|N_u|+1]$*, a real-valued vector* $\mathbf{w}_{best} \in \mathbb{R}^d$*, and a real number* $best \in \mathbb{R}$*, find*

$$h = \underset{\alpha_1,\ldots,\alpha_k \in \mathbb{R}}{\text{maximize}} \;\; \mathbf{L}_0[i] \cdot \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j$$

$$\textit{subject to} \qquad \mathbf{L}_0[1] \cdot \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j \geq 0;$$

$$\mathbf{w}_{best} \cdot \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j \geq best; \;\; \left\| \textstyle\sum_{j=1}^{k} \alpha_j \; \mathbf{v}_j \right\|_2 \leq 1.$$

Auxiliary problem II is the same as Auxiliary problem I with the additional constraint that the projection of the centroid $\mathbf{w}_{best}$ on the linear combination of the eigenvectors must produce a value $\geq best$. This additional constraint can render the problem either feasible or infeasible. Regardless of the specific location i, Auxiliary problem II always maintains the same constraints, with only the objective function changing. Consequently, either all such problems are feasible, or none of them is. If Auxiliary problem II is not feasible, Alg. 4 returns the projection computed using Auxiliary problem I (Line 8). Otherwise, the projected locations are recomputed using Auxiliary problem II. Auxiliary problem II remains convex quadratic, which can be efficiently solved using the interior-point algorithm in polynomial time with respect to k and in linear time with respect to d. The advantage of using Auxiliary problem II is that it imposes more constraints, leading to projected locations that are more interrelated with each other.

Ultimate proposed SGRLvCLT algorithm (Alg. 5). The ultimate algorithm we propose for SGRL (Problem 1) is termed SGRL via Canonical Location Transformation (for short, SGRLvCLT) and outlined in Alg. 5. As anticipated above, SGRLvCLT applies the proposed canonical rotation-translation-invariant location transformation on the input spatial graph G so as to produce the transformed directed graph G' with features on edges, and then it let an existing (non-spatial) GRL method $\mathcal{A}_{\text{GRL}}$ run on G' to produce the desired embeddings.

Algorithm 5 SGRLvCLT

Input: Spatial graph $G = (V, E, \ell)$, natural number $q \in \mathbb{N}^+$, GRL algorithm $\mathcal{A}_{\text{GRL}}$
Output: Real-valued matrix $R \in \mathbb{R}^{|V|\times q}$, where every row $R[u]$ corresponds to the embedding of $u \in V$

1: $G' \leftarrow$ Canonical Spatial Graph Transformation(G) ▷ Alg. 1
2: $R \leftarrow \mathcal{A}_{\text{GRL}}(G', q)$

Note that this proposed approach is general enough to produce embeddings also for elements of the input graph other than nodes (e.g., edges or subgraphs), or for entire graphs. Achieving this is as simple as just employing a different GRL method $\mathcal{A}_{\text{GRL}}$, which is suited for embedding any other desired elements of the graph or entire graphs.

A. Theoretical results

In the following, we show some theoretical results of the proposed canonical location transformation methodology (Alg. 1). All the proofs are reported in the **Appendix**.

Theorem III.1 (Rotation-translation invariance)**.** *Given two nodes* $a, b \in V$ *of a spatial graph* $G = (V, E, \ell)$*, let* a*'s neighborhood* N_a *"rotation-translation aligned" to* b*'s neighborhood* N_b*, i.e., there exist a bijective function* $fi\colon N_a \to N_b$ *and a function* tr *representing a composition of arbitrary rotations and translations in the* d*-D Euclidean space such that* $\ell(a) = tr(\ell(b))$ *and* $\ell(c) = tr(\ell(fi(c)))$*, for all* $c \in N_a$*. It holds that running Alg. 1 on* G *yields* $G' = (V, E', \ell_e)$ *such that* $\ell_e(a, c) = \ell_e(b, fi(c))$*, for all* $c \in N_a$*.*

Theorem III.2 (Pairwise distance preservation in the canonical spatial mapping)**.** *If in Alg. 2, all eigenvalues have multiplicity 1 and Alg. 3 successfully applies at least one disambiguation strategy for each eigenvalue (i.e., does not return the zero vector), then for all* $i, j \in 2,\ldots,|N_u|+2$*, the Euclidean distance between* $L[i]$ *and* $L[j]$ *remains identical to that between* $L'[i-1]$ *and* $L'[j-1]$*.*

Theorem III.3 (Linear execution time)**.** *Given a graph* $G = (V, E, l)$*, Alg. 1 runs in linear time with respect to* $|V| + |E|$*, i.e., its time complexity is* $O(|V| + |E|)$*.*

IV. Experiments

This section evaluates our approach on datasets and tasks (namely, graph classification and regression) that were recognized as benchmarks in the state of the art on SGRL [23].

Datasets (Table I). We use real-world spatial-graph datasets which have been employed in the state of the art in SGRL [23] (BACE, BBBP, LIPO, ESOL, QM9, HCP), along with one synthetic dataset created by us. In the **extended technical report** [1], we also experiment with a further molecular dataset which was employed in a well-established challenge (OC22).

Real-world molecular datasets. We involve five benchmark datasets from MoleculeNet [17], encompassing both graph classification and regression tasks. The Lipophilicity (LIPO) dataset contains experimental measurements of octanol/water distribution coefficient, which reflects the permeability and solubility of compounds. ESOL contains water solubility values,

TABLE I

MAIN CHARACTERISTICS OF THE DATASETS USED IN THIS WORK. EACH DATASET CONTAINS A NUMBER G OF SPATIAL GRAPHS EMBEDDED IN A 3-D EUCLIDEAN SPACE, WITH N NODES AND M EDGES IN TOTAL (I.E., SUMMING THE NUMBER OF NODES AND EDGES IN ALL THE GRAPHS). L DENOTE THE NUMBER OF CLASSES (MISSING FOR DATASETS USED FOR THE REGRESSION TASK ONLY).

Dataset	*Domain*	*Task*	*N*	*M*	*G*	*L*
BACE	molecular	classification	51 577	111 536	1 513	2
BBBP	molecular	classification	49 068	105 842	2 050	2
LIPO	molecular	regression	113 568	247 798	4 200	—
ESOL	molecular	regression	14 991	30 856	1 128	—
QM9	molecular	regression	2 359 210	4 883 516	133 885	—
HCP	brain	regression	8 152	9 464	823	—
Synthetic	—	classification	19 500	53 400	2 400	24

used to train models to predicting solubility from molecular structures. QM9 is a multi-task regression benchmark featuring 12 quantum mechanical properties. For graph classification, BACE provides quantitative and qualitative binding results for β-secretase, while BBBP focuses on predicting blood-brain barrier permeability. All these datasets were obtained from the `PyTorch Geometric` library.

Real-world brain dataset. The HCP dataset consists of structural connectivity graphs derived from magnetic resonance imaging (MRI) data obtained through the Human Connectome Project (HCP). A regression task is used to predict the age of the patient based on brain structural connectivity. To preserve comparability with other models, we followed the same data processing procedure as outlined in [23].

Synthetic graph dataset. We also created a synthetic dataset consisting of 2 400 graphs with spatial coordinates, where graph labels are dependent on structure and location.[1]

Proposed method(s). We test the proposed SGRLvCLT method (Alg. 5), equipped with two state-of-the art GRL algorithms that provide embeddings of entire graphs, namely SIR-GN [9] and Graph Isomorphism Network (GIN) [20]. SIR-GN and GIN were used as algorithm $\mathcal{A}_{\text{GRL}}$ in Alg. 5. As for GIN, we actually use its adaptation that incorporates edge features into the aggregation procedure, dubbed GINE [7], to utilize the output of Alg. 1. We selected SIR-GN and GIN(E) as two prototypical structural GRL methods, well-representative of the unsupervised (SIR-GN) and supervised (GINE) category, respectively. We remark (again) that our SGRLvCLT method can be coupled with *any* GRL algorithm. By equipping it with SIR-GN and GINE, we believe we can provide an informative picture of its performance. A more exhaustive experimentation on the various GRL methods is beyond the scope of this work.

The resulting two versions of our method are referred to as SGRLvCLT-SIRGN and SGRLvCLT-GINE, respectively.

Competitors. As a main competitor, we involved the prominent state-of-the-art method for SGRL, namely SGMP by Zhang *et al.* [23] (described in Sect. II). We also selected two further methods, namely SGCN [4] and Gated-GCN [11], which were among the top-performing methods reported in Zhang *et al.*'s work [23]. For all these competitors, we relied on the official source code provided by the authors on their `GitHub`.

[1]A detailed description of how we created our synthetic dataset, along with the code used for its creation and the dataset itself are available at [1].

TABLE II

ABLATION STUDY ON OUR SGRLvCLT EQUIPPED WITH SIR-GN OR GINE, AND USING DIFFERENT SPATIAL NODE AND EDGE FEATURES. FOR CLASSIFICATION, ACCURACY (ACC) RESULTS ARE REPORTED. FOR REGRESSION, RMSE IS ROOT MEAN SQUARED ERROR, AND MAE IS MEAN ABSOLUTE ERROR. SUBSCRIPT OF *nf* DENOTES METHODS RUN WHERE ONLY SPATIAL FEATURES ARE PRESENT. THE BEST PERFORMING NON-BASELINE VALUES ARE IN BOLD.

	Classification			Regression					
	BACE	BBBP	Synth	LIPO		ESOL		HCP	
SGRLvCLT +SIRGN	ACC	ACC	ACC	RMSE	MAE	RMSE	MAE	RMSE	MAE
Distance in Edges_{nf}	.733	.837	.333	1.06	.816	1.34	1.05	.791	.632
Position in Nodes_{nf}	.757	.801	.525	1.15	.884	1.41	1.11	.798	.644
Canonical Position in Edges_{nf}	.789	.867	**.883**	1.02	.789	1.37	1.08	**.788**	**.627**
Canonical Position in Edges & Position in Nodes_{nf}	.784	.866	.871	1.01	.782	1.36	1.08	.791	.635
Canonical Position & Distance in Edges_{nf}	**.8**	**.875**	.881	**.92**	**.7**	**.943**	**.7**	.795	.634
SGRLvCLT-GINE									
Distance in Edges_{nf}	.755	.969	.298	.862	.688	.417	.332	.263	.21
Position in Nodes_{nf}	.74	.953	.333	.862	.688	.415	.331	.264	.21
Canonical Position in Edges_{nf}	.76	.973	.803	.862	.688	.415	.331	.265	.212
Canonical Position in Edges & Position in Nodes_{nf}	**.793**	**.996**	.564	.862	.688	**.414**	**.33**	.263	.21
Canonical Position & Distance in Edges_{nf}	.745	.983	**.858**	**.862**	**.687**	.415	.331	**.262**	**.209**

Assessment. We evaluate the various methods in both graph classification and regression tasks. For the classification task, we trained a classifier (*Extra Trees*) using the embeddings as feature vectors and the node labels as a target variable to be predicted. We measure *accuracy* (ACC) ($\in [0, 1]$, higher values meaning better performance). As for regression, we trained a regressor (*Extra Trees* again) using the embeddings as feature vectors, and the to-be-predicted score as a target variable. We assess the performance in terms of *root mean squared error* (*RMSE*) and *mean average error* (*MAE*) (both $\in [0, +\infty)$, lower values corresponding to better performance).

Settings. We experiment both with and without (denoted with subscript "***nf***") the use of additional, non-spatial node and edge features which could be available with the various datasets. Such "*nf*" versions of the various methods are tested to isolate the contribution of spatial locations to performance on the downstream tasks. In fact, some non-spatial features may carry some overlap with location information (e.g. the "is_aromatic" node feature and the the "bond_type" edge feature) and thus blur the impact of spatial features.

All the experiments are run with three to five different seeds for random shuffling of training/test sets. All the reported results are averaged over all these runs.

Testing environment. All experiments were conducted on a 64-bit workstation equipped with a NVIDIA RTX 2080-ti 11GB, an Intel core-i7 CPU, and 32 GB RAM.

A. Results

Ablation study (Table II). We tested our method by varying the spatial information provided with it, in the form of d-dimensional positions as node features, Euclidean distance between nodes as edge features, our canonical spatial features (as computed by Alg. 5) as edge features, along with combinations of each. The results in Table II reveal that incorporating

TABLE III

COMPARISON OF OUR SGRLvCLT (EQUIPPED WITH SIR-GN OR GINE) TO STATE-OF-THE-ART METHODS. FOR CLASSIFICATION, ACCURACY (ACC) RESULTS ARE REPORTED. FOR REGRESSION, RMSE IS ROOT MEAN SQUARED ERROR, AND MAE IS MEAN ABSOLUTE ERROR. THE BEST-PERFORMING VALUES ARE HIGHLIGHTED IN BOLD. SUBSCRIPT *nf* DENOTES METHODS RUN WITHOUT FEATURES ADDITIONAL TO SPATIAL INFORMATION.

	Classification			Regression					
	BACE	BBBP	Synth	LIPO		ESOL		HCP	
	ACC	ACC	ACC	RMSE	MAE	RMSE	MAE	RMSE	MAE
SGRLvCLT-SIRGN	.792	.87	**.881**	.882	.666	.981	.682	.852	.699
SGRLvCLT-GINE	.755	**.973**	.858	.862	.688	**.416**	**.332**	**.268**	**.214**
SGMP	**.825**	.859	.033	.984	.778	.979	.695	.757	.576
SGCN	.629	.819	.6	**.856**	**.657**	.788	.626	.745	.591
Gated-GCN	.704	.833	.538	.886	.666	.975	.624	.744	.578
SGRLvCLT-SIRGN $_{nf}$	**.8**	.875	**.881**	.92	.7	.943	.7	.795	.634
SGRLvCLT-GINE $_{nf}$	.745	**.983**	.858	**.862**	**.687**	**.415**	**.331**	**.262**	**.209**
SGMP$_{nf}$	.771	.852	.033	.945	.735	.974	.765	.754	.572
SGCN$_{nf}$	.57	.737	.6	1.19	.956	1.4	1.07	.745	.591
Gated-GCN$_{nf}$	.628	.799	.538	1.14	.915	1.8	.915	.744	.579

TABLE IV

COMPARISON OF OUR SGRLvCLT (EQUIPPED WITH SIR-GN OR GINE) TO STATE-OF-THE-ART METHODS ON THE REGRESSION TASK ON THE QM9 DATASET [17]. THE TASK CONSISTS IN PREDICTING A SERIES OF 12 QUANTUM MECHANICS PROPERTIES ASSOCIATED WITH THE DATASET (EACH COLUMN REFERS TO ONE OF SUCH QUANTUM MECHANICS PROPERTIES). RESULTS ARE IN TERMS OF MAE (MEAN ABSOLUTE ERROR). THE BEST-PERFORMING VALUES ARE HIGHLIGHTED IN BOLD. VALUES OF THE COMPETITORS ON THIS DATASET ARE THOSE REPORTED IN [23]

	μ	α	ϵ_H	ϵ_L	Δ_ϵ	$\langle R^2 \rangle$	$ZPVE$	U_0	U	H	G	c_v
SGRLvCLT-SIRGN	.613	.659	**.144**	**.16**	**.196**	66.7	.01	**.756**	**.774**	**.775**	**.76**	.306
SGRLvCLT-GINE	.585	.538	251.1	256.1	258.5	5.05	15.1	26.9	263.4	265.5	251.1	.52
SGMP	**.13**	**.113**	64.7	44.7	83.7	**5.9**	2.3	26.1	25.2	27.5	24.6	**.043**
SGCN	.503	.531	193.8	141.7	275.5	34.9	7.4	201.3	21.1	199.2	207.8	.277
Gated-GCN	.543	.609	206.2	135.4	314.4	63.1	12.1	222.5	244.7	239.2	221.1	.283

canonical spatial locations, either alone or in combination with other features, consistently enhance classification and regression performance. This outperforms using only the Euclidean distance between nodes as edge features or raw node positions (as seen in rows 1-2 vs. 3-5 of the table). In particular, we observe a 3-5% performance increase over the methods that do not incorporate spatial features on the BACE and BBBP datasets, and 50% on Synthetic. For regression tasks such as LIPO and ESOL, our approach, when combined with SIR-GN, yields a 7-8% improvement. These results suggest that incorporating our canonical spatial features significantly enhance performance. We found that encoding the canonical edge features as well as the inter-node distance provided maximal performance. Thus, subsequent performance experiments use this combination.

Comparison to the state of the art – Classification (Table III, first three columns). On the classification task, our approach outperforms all other methods on BBBP, regardless of whether non-spatial features are included, demonstrating high performance when coupled with both SIR-GN and GINE. On BACE, our method (using SIR-GN) achieves the highest accuracy when leveraging only spatial information, while also outperforming SGCN and Gated-GCN in all cases. The results on the Synthetic dataset reveal that, when equipped with SIR-GN, our method significantly outperforms all the competitors. This remains true even when using only spatial features.

Interestingly, the incorporation of non-spatial features impacts the performance of our method much less than the competitors. This suggests that our canonical spatial features are often sufficient to capture and extrapolate the underlying structure and dependencies in the data. This is a further important benefit of our approach.

TABLE V

PERFORMANCE OF OUR SGRLvCLT EQUIPPED WITH SIR-GN OR GINE, COMPARED TO THEIR NON-OPTIMIZED VERSIONS (I.E., VERSIONS WHERE THE BASIC DISAMBIGUATION STRATEGY DISAMBIGUATION (ALG. 3) IS USED INSTEAD OF THE ENHANCED DISAMBIGUATION* ONE (ALG. 4)). FOR CLASSIFICATION, ACCURACY (ACC) RESULTS ARE REPORTED. FOR REGRESSION, RMSE IS ROOT MEAN SQUARED ERROR, AND MAE IS MEAN ABSOLUTE ERROR. SUBSCRIPT *nf* DENOTES METHODS RUN WITHOUT FEATURES ADDITIONAL TO SPATIAL INFORMATION. SUPERSCRIPT *wo* DENOTES METHODS RUN WITHOUT THE ENHANCED STRATEGY (I.E., USING DISAMBIGUATION AND NOT DISAMBIGUATION*).

	Classification			Regression					
	BACE	BBBP	Synth	LIPO		ESOL		HCP	
	ACC	ACC	ACC	RMSE	MAE	RMSE	MAE	RMSE	MAE
SGRLvCLT-SIRGN	0.792	0.87	0.881	0.882	0.6656	0.981	0.682	0.852	0.699
SGRLvCLT-SIRGN $_{nf}$	0.8	0.875	0.881	0.92	0.7	0.943	0.7	0.795	0.634
SGRLvCLT-GINE	0.755	0.973	0.858	0.862	0.688	0.416	0.332	0.268	0.214
SGRLvCLT-GINE $_{nf}$	0.745	0.983	0.858	0.862	0.687	0.415	0.331	0.262	0.209
SGRLvCLT-SIRGN ^{wo}	0.789	0.864	0.884	0.885	0.668	0.976	0.678	0.852	0.699
SGRLvCLT-SIRGN $^{wo}_{nf}$	0.801	0.868	0.884	0.92	0.708	0.939	0.695	0.795	0.634
SGRLvCLT-GINE ^{wo}	0.745	0.926	0.875	0.862	0.688	0.416	0.332	0.264	0.211
SGRLvCLT-GINE $^{wo}_{nf}$	0.7	0.973	0.875	0.862	0.688	0.415	0.331	0.262	0.209

Comparison to the state of the art – Regression (Table III, last six columns; Table IV). On the regression task, our method, when coupled with GINE, consistently outperforms the competitors on the ESOL and LIPO datasets. On ESOL, our method achieves an RMSE of 0.4162 compared to 0.7875 of the second best performer. A similar result holds on HCP, where the second highest performer has RMSE and MAE values nearly triple that of our SGRLvCLT. On LIPO, our method shows superior performance in terms RMSE using only spatial features, and performs nearly as well as the best performer when non-spatial features are included. On QM9, our method achieves a lower MAE in 8 out of 12 of the regression targets, with average improvements of 98% over SGMP and 99% over Gated-GCN. In the case of the QM9 dataset, we observe that SIR-GN can outperform GINE in certain settings due to its unsupervised nature, which may help mitigate overfitting. However, this same property can also act as a limitation in task-specific scenarios, where the lack of supervision hinders model's ability to produce embeddings finely tuned to the downstream objective–potentially explaining performance differences between QM9 and other datasets.

Impact of Disambiguation* (Table V). We also provide a comparison of the enhanced disambiguation strategy Disambiguation* (Alg. 4) to the basic strategy Disambiguation (Alg. 3). Table V shows that Disambiguation* makes performance improve, even consistently in certain cases–for instance, classification accuracy increases by approximately 4% for SGRLvCLT-GINE $_{nf}$ on the BACE dataset, and all variations show improved accuracy on the BBBP dataset. Additionally, we observe a slight performance boost in regression for the LIPO dataset. However, for the Synthetic, ESOL, and HCP datasets, the optimization has negligible impact, suggesting that these tasks benefit less from its application.

Execution time (Table VI). We analyze runtimes of SGRLvCLT alone, when coupled with SIR-GN, and when coupled with GINE, and compare them to the best-performing competitor of our method, i.e., the state-of-the-art SGMP. Runtimes of SGMP and SGRLvCLT-GINE are more directly comparable, as both make use of the GPU, while SIR-GN does not. From

TABLE VI
EXECUTION TIMES (SECONDS) OF OUR SGRLvCLT, IN ISOLATION AND WHEN EQUIPPED WITH SIR-GN OR GINE, COMPARED TO THE BEST-PERFORMING STATE-OF-THE-ART METHOD (I.E., SGMP).

	Classification			Regression		
	BACE	BBBP	Synth	LIPO	ESOL	HCP
SGRLvCLT Only	25	19	165	59	7	2
SGRLvCLT-SIRGN	154	142	349	687	80	65
SGRLvCLT-GINE	122	172	298	775	76	33
SGMP	180	243	360	487	133	108

these results, it is clear that SGRLvCLT in isolation takes only a small portion of the overall runtime, even without the use of parallelization. The combinations of SGRLvCLT and SIR-GN or GINE are shorter runtime than SGMP in all but one dataset.

V. CONCLUSION

In this paper, we advance the state of the art in spatial graph representation learning (SGRL) by devising a novel approach that decouple the main phases of rotation-translation-invariance preservation and representation learning, and devises a novel general methodology for the first one of these phase. The proposed methodology can be coupled with any existing non-spatial graph representation learning (GRL) method, so as to ultimately solve SGRL in a novel way which allows for overcoming the main limitations of the state of the art in terms of generalizability, extendability, and effectiveness-efficiency tradeoff. The proposed approach comes with important theoretical properties in terms of rotation-translation invariance and distance preservation, and is shown, by extensive experiments on a variety of datasets and tasks, to outperform the state-of-the-art methods in terms of effectiveness-efficiency tradeoff.

In the future, we plan to investigate different disambiguation strategies and to experiment with other scenarios and tasks.

REFERENCES

[1] Anonymous Authors, "Efficient and effective spatial graph representation learning via canonical location transformation – TECHNICAL REPORT," 2025. [Online]. Available: https://anonymous.4open.science/??????

[2] S. Aykent and T. Xia, "GBPNet: Universal geometric representation learning on protein structures," in *Proc. of KDD Conf.*, 2022, pp. 4–14.

[3] M. R. Dale, *Spatial Graphs*. Cambridge University Press, 2017, p. 191–221.

[4] T. Danel, P. Spurek, J. Tabor, M. Smieja, L. Struski, A. Slowik, and L. Maziarka, "Spatial graph convolutional networks," in *Proc. of ICONIP Conf.*, 2020, pp. 668–675.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. of ICML Conf.*, 2017, pp. 1263–1272.

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of KDD Conf.*, 2016, pp. 855–864.

[7] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *Proc. of ICLR Conf.*, 2020.

[8] J. Jin, M. Heimann, D. Jin, and D. Koutra, "Toward understanding and evaluating structural node embeddings," *TKDD*, vol. 16, no. 3, pp. 58:1–58:32, 2022.

[9] M. Joaristi and E. Serra, "SIR-GN: A fast structural iterative representation learning approach for graph nodes," *TKDD*, vol. 15, no. 6, pp. 100:1–100:39, 2021.

[10] J. Klicpera, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," in *Proc. of ICLR Conf.*, 2020.

[11] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in *Proc. of ICLR Conf.*, 2016.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *Proc. of KDD Conf.*, 2014, pp. 701–710.

[13] R. A. Rossi, D. Jin, S. Kim, N. K. Ahmed, D. Koutra, and J. B. Lee, "On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications," *TKDD*, vol. 14, no. 5, pp. 63:1–63:37, 2020.

[14] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Proc. of NeurIPS Conf.*, 2017, pp. 991–1001.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *Proc. of WWW Conf.*, 2015, pp. 1067–1077.

[16] Z. Wang, C. Ju, G. Cong, and C. Long, "Representation learning for spatial graphs," *CoRR*, vol. abs/1812.06668, 2018.

[17] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "MoleculeNet: a benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018.

[18] T. Xia and S. Aykent, "SO(3) equivariant framework for spatial networks," in *Proc. of IEEE Big Data Conf.*, 2024, pp. 1883–1891.

[19] T. Xia and W. Ku, "Geometric graph representation learning on protein structure prediction," in *Proc. of KDD Conf.*, 2021, pp. 1873–1883.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. of ICLR Conf.*, 2019.

[21] D. Yu, G. Bai, Y. Li, and L. Zhao, "Deep spatial domain generalization," in *Proc. of IEEE ICDM Conf.*, 2022, pp. 1293–1298.

[22] Z. Zhang, S. Li, J. Zhou, J. Wang, A. Angirekula, A. Zhang, and L. Zhao, "Non-euclidean spatial graph neural network," in *Proc. of SIAM SDM Conf.*, 2024, pp. 154–162.

[23] Z. Zhang and L. Zhao, "Representation learning on spatial networks," in *Proc. of NeurIPS Conf.*, 2021, pp. 2303–2318.

[24] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, "Arbitrary-order proximity preserved network embedding," in *Proc. of KDD Conf.*, 2018, pp. 2778–2786.

[25] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *TKDE*, vol. 34, no. 1, pp. 249–270, 2022.

APPENDIX

Proof of Theorem III.1. Assume that the locations of node a and its neighbors are stored in order in the matrix $\mathbf{L}$, which, we recall, it has $m = |N_a| + 1$ rows and d columns. The locations of node b and its neighbors are just translated and rotated together w.r.t. the locations of node a and its neighbors. Then, $\mathbf{L}_b$ (storing the locations of b and its aligned neighbors to N_a) is a composition of rotations and translations of $\mathbf{L}_a$.

Any rotation-translation transformation $(\mathbf{A}, \mathbf{b})$ of a vector $\mathbf{x}$ can be defined as $\mathbf{A}\mathbf{x}+\mathbf{b}$, where $\mathbf{A}$ is a rotation matrix and $\mathbf{b}$ is a translation vector. The composition $\mathbf{A}_2(\mathbf{A}_1\mathbf{x}+\mathbf{b}_1)+\mathbf{b}_2$ of any two transformations $(\mathbf{A}_1, \mathbf{b}_2)$ and $(\mathbf{A}_2, \mathbf{b}_2)$ can be rewritten as $\mathbf{A}_3 x + \mathbf{b}_3$, where $\mathbf{A}_3 = \mathbf{A}_2\mathbf{A}_1$ and $\mathbf{b}_3 = \mathbf{A}_2\mathbf{b}_1 + \mathbf{b}_2$. This means that, without any loss of generality, a composition of an arbitrary number of rotation-translation transformations can be represented as a single pair $(\mathbf{A}, \mathbf{b})$. Thus, we can safely assume that $\mathbf{L}_b^\top = \mathbf{A}\mathbf{L}_a^\top + |\underbrace{\mathbf{b}, \ldots, \mathbf{b}}_{m}|$ where, $|\underbrace{\mathbf{b}, \ldots, \mathbf{b}}_{m}|$ is a matrix with, $m = |N_a| + 1$ columns, each one equal to $\mathbf{b}$. By executing Line 1 of Alg. 2, it holds that

$$\mathbf{L}_{a,0}^\top = \mathbf{L}_a^\top - |\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|, \text{and}$$

$$\mathbf{L}_{b,0}^\top = \mathbf{A}\mathbf{L}_a^\top + |\underbrace{\mathbf{b}, \ldots, \mathbf{b}}_{n}| - \underbrace{|RM(\mathbf{A}\mathbf{L}_a^\top + |\underbrace{\mathbf{b}, \ldots, \mathbf{b}}_{n}|), \ldots, RM(\mathbf{A}\mathbf{L}_a^\top + |\underbrace{\mathbf{b}, \ldots, \mathbf{b}}_{n}|)|}_{m},$$

where $RM(\mathbf{A})$ is a vector corresponding to the mean of each row of $\mathbf{A}$. Then, $\mathbf{L}_{b,0}^\top$ can be simplified as $\mathbf{L}_{b,0}^\top = \mathbf{A}\mathbf{L}_a^\top + |\underbrace{RM(\mathbf{A}\mathbf{L}_a^\top), \ldots, RM(\mathbf{A}\mathbf{L}_a^\top)}_{m}| = \mathbf{A}\mathbf{L}_a^\top + \mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|$. This means that $\mathbf{L}_{b,0}^\top$ just depends on the compositions of rotations $\mathbf{A}$. Lines 3 and 6 compute the eigenvectors of the covariance matrix of $\mathbf{L}_0$. Such eigenvectors represent new dimensions with maximum variance of the projected locations each one perpendicular to each other. Such new dimensions will represent the new canonical locations. Let us assume $\mathbf{v}$ is the first eigenvector of the covariance matrix of $\mathbf{L}_{a,0}$, then $\mathbf{v}$ is, according to the principal component analysis, the first dimension that maximizes the variance of the projected location. Since $\mathbf{L}_{b,0}^\top = \mathbf{A}\mathbf{L}_a^\top + |\underbrace{RM(\mathbf{A}\mathbf{L}_a^\top), \ldots, RM(\mathbf{A}\mathbf{L}_a^\top)}_{m}|$, it is easy to see that the rotated vector $\mathbf{A}\mathbf{v}$ is the dimension that maximizes the variance of the projected data $\mathbf{L}_{b,0}$.

Then, we will compute the projected data $\mathbf{L}_{a,0}$ on $\mathbf{v}$ as $\mathbf{v}^\top(\mathbf{L}_a^\top - |\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|) = \mathbf{v}^\top\mathbf{L}_a^\top - \mathbf{v}^\top|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|$ and the projected data $\mathbf{L}_{a,0}$ on $\mathbf{A}\mathbf{v}$ as $(\mathbf{A}\mathbf{v})^\top(\mathbf{A}\mathbf{L}_a^\top + \mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|) = (\mathbf{A}\mathbf{v})^\top\mathbf{A}\mathbf{L}_a^\top + (\mathbf{A}\mathbf{v})^\top\mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|$.

Focusing on $(\mathbf{A}\mathbf{v})^\top\mathbf{L}_{b,0}^\top = (\mathbf{A}\mathbf{v})^\top\mathbf{A}\mathbf{L}_a^\top + (\mathbf{A}\mathbf{v})^\top\mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}| = \mathbf{v}^\top\mathbf{A}^\top\mathbf{A}\mathbf{L}_a^\top + \mathbf{v}^\top\mathbf{A}^\top\mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|$, since $\mathbf{A}$ is a product of rotation matrices and for each rotation matrices the transpose is equivalent to the inverse, then $\mathbf{A}^\top = \mathbf{A}^{-1}$. This implies that $(\mathbf{A}\mathbf{v})^\top\mathbf{L}_{b,0}^\top = \mathbf{v}^\top\mathbf{A}^\top\mathbf{A}\mathbf{L}_a^\top + \mathbf{A}\mathbf{v}^\top\mathbf{A}^\top\mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}| = \mathbf{v}^\top\mathbf{A}^{-1}\mathbf{A}\mathbf{L}_a^\top + \mathbf{v}^\top\mathbf{A}^{-1}\mathbf{A}|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}|$ which is equal to $\mathbf{v}\mathbf{L}_a^\top - \mathbf{v}^\top|\underbrace{RM(\mathbf{L}_a^\top), \ldots, RM(\mathbf{L}_a^\top)}_{m}| = \mathbf{L}_{a,0}^\top$.

We can generalize such a result to all the eigenvectors $\mathbf{v}$ of the covariance matrix of $\mathbf{L}_{a,0}$ and $\mathbf{A}\mathbf{v}$ of the covariance matrix of $\mathbf{L}_{b,0}$, respectively. This means that the projected locations on $\mathbf{v}$ and $\mathbf{A}\mathbf{v}$ of $\mathbf{L}_{a,0}$ and $\mathbf{L}_{b,0}$, respectively, will be the same.

Now, considering Disambiguation in Alg. 3 (a similar reasoning holds for Disambiguation*, Alg. 4), the eigenvector directions considered are only the ones with eigenvalue with multiplicity 1, in all the other cases it returns a column with all the elements 0 (which makes the theorem trivially true). Now, for each eigenvalue with multiplicity 1, there exist only two eigenvectors $\mathbf{v}$ and $-\mathbf{v}$. In this case, to make sure that for both $\mathbf{L}_a$ and $\mathbf{L}_b$ the algorithm returns the same canonical locations, it is required that in both the computations the choice of such eigenvectors is done properly. The algorithm first verifies if the projected location of the node a is positive when projects on the eigenvector; otherwise, if negative, it considers the negated eigenvector. In this way, the projected locations are the same regardless of the fact that they are rotated or translated. If the projected location of a is zero, this is still an undetermined situation, which the algorithm handles by looking at the projected locations of a's neighbors, and uniquely disambiguating by properly considering the maximum and the minimum of such neighbors' projected values as described in Lines 10–17. The theorem follows. □

Proof of Theorem III.2. In Alg. 2, if all the eigenvalues have multiplicity one and Alg. 3 successfully applies at least one disambiguation strategy for each eigenvalue (i.e., it does not return the zero vector), then the eigenvectors associated with these eigenvalues forms a basis for distinct eigenspaces of rank d, which is the maximum rank in d-D space. In fact, the success of one disambiguation strategy implies that, for each eigenvalue, the data are projected using either an eigenvector or its complement.Regardless of which eigenvector is selected, the chosen set of eigenvectors constitutes a basis for distinct eigenspaces, which is then used to transform each location $\mathbf{L}[i]$, for $i \in 2, \ldots, |N_u| + 2$, into a new location $\mathbf{L}'[i-1]$ within the eigenspaces. Since the basis of these eigenspaces forms an orthogonal matrix (i.e., $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$), the transformation preserves Euclidean distances. Consequently, for all $i, j \in 2, \ldots, |N_u| + 2$, the Euclidean distance between $\mathbf{L}[i]$ and $\mathbf{L}[j]$ remains identical to that between $\mathbf{L}'[i-1]$ and $\mathbf{L}'[j-1]$. □

Proof of Theorem III.3. In the analysis, we assume d constant as it is much smaller than $|V|$ and $|E|$, thus being negligible.

Alg. 1 iterates over each node $u \in V$ and each u's neighbor in N_u to construct matrix $\mathbf{L} \in \mathbb{R}^{(|N_u|+1)\times d}$. This operation takes $O(|V| + |E|)$ time. Next, function Canonical Spatial Mapping (Alg. 2) is run on $\mathbf{L}$, and the directed graph with canonical edge features is constructed afterward. This construction takes $O(|V|+|E|)$ time too. Thus, if Alg. 2 runs in time linear to $O(|N_u|)$, the entire Alg. 1 would have a time complexity of $O(|V| + |E|)$. Alg. 2 takes as input matrix $\mathbf{L} \in \mathbb{R}^{(|N_u|+1)}$. Both Line 1 and Line 2 run in $O(|N_u|)$ (assuming again d constant). The construction of the $d \times d$ covariance matrix takes constant time. Line 4 takes $O(|N_u|)$ time. Since $d' \leq d$, the loop in Line 5 is executes for a constant number of iterations, with its dominant cost occurring in Line 7, where either Alg. 3 or Alg. 4 is called. Alg. 3 consists only of linear operations in $|N_u|$, including the maximum and minimum computations in Line 10, making its complexity trivially $O(|N_u|)$. Similarly, Alg. 4 either calls Alg. 3 (which takes $O(|N_u|)$ time) or iterates over N_u, solving one of the two auxiliary optimization problems at each step. Both optimization problems are polynomial in the eigenvalue multiplicity k and linear in d. Since k is at most d, solving these optimization problems has a constant cost. Hence, Alg. 4 takes $O(|N_u|)$ time too, completing the proof. □

Efficient and Effective Spatial Graph Representation Learning via Canonical Location Transformation

Supplementary Material

SI. Performance on Open Catalyst and Comparison with Physically Informed Deep Neural Networks

Open Catalyst project. The Open Catalyst Project [ZCD+20] is a joint initiative between Meta's Fundamental AI Research (FAIR) team and Carnegie Mellon University's Department of Chemical Engineering. Its goal is to leverage AI for modeling and discovering novel catalysts to advance renewable energy storage and combat climate change.

A key challenge in catalyst research is to identify low-cost materials that can drive reactions at high rates. Quantum mechanical simulations, such as Desnity Functional Theory (DFT), allow for testing and evaluating new catalyst structures. The Open Catalyst Project has released the Open Catalyst 2020 (OC20) [CDG+21] and Open Catalyst 2022 (OC22) [TLS+23] datasets for training machine learning models. Together, these datasets include 1.3 million molecular relaxations and results from over 260 million DFT calculations.

Structure to Total Energy and Forces. We included in our experiments the Structure to Total Energy and Forces (S2EF) task of the Open Catalyst 2022 (OC22) dataset. This task involves predicting the total energy and per-atom forces based on atomic positions as computed by DFT. In this context, energy refers to adsorption energy, which is defined as the difference between the total energy of the combined surface-adsorbate system and the sum total of the relaxed slab and gas-phase adsorbate molecule energies. Forces are determined as the negative gradient of the energy with respect to atomic positions. The S2EF task serves as a general benchmark for catalyst research due to its large-scale dataset and inclusion of inorganic and organic materials. The OC22 dataset specifically consists of $62,331$ DFT relaxations spanning various oxide materials, coverages, and adsorbates. It provides precomputed LMDB files for training, validation, and testing, which incudes input structures from relaxation trajectories along with their energy and atomic forces. The validation and test sets are further divided into in-distribution (ID) and out-of-distribution (OOD) splits based on material similarity to the training set.

While this task has been explored in the context of Physically Informed Neural Networks (PINNs), it has not been a primary benchmark for the other methods we compare against. Therefore, we include it in the appendix as an additional evaluation to assess the generalization of our approach beyond the main experiments.

1) Experimental Results on Open Catalyst.: Testing Metrics To assess model performance and practical applicability, we use evaluation metrics based on DFT calculations. The four key metrics for this task include:

- Energy MAE (lower better): Measures the Mean Absolute Error (MAE) between the computed and ground truth energy values.
- Force MAE (lower better): MAE between the predicted and ground truth per-atom forces, computed only for free catalyst and adsorbate atoms.
- Force Cosine Similarity (higher better: Measures the alignment between predicted and ground truth force vectors by calculating the mean cosine of their angles, considering only free atoms.
- Energy and Forces within Threshold (EFwT) (higher better): A practical benchmark that evaluates whether both energy and forces are close to the ground truth. A prediction is considered "correct" if the energy error is within 0.2 eV and the maximum per-atom force error is below 0.3 $eV/\mathring{A}$. EFwT represents the percentage of structures meeting this criteria.

TABLE SI
Results for the S2EF task of the OC22 dataset for our SGRLvCLT coupled with SIR-GN, compared to state-of-the-art physically informed neural networks. Values for SchNet, DimeNet++, and GemNet, are literature produced [TLS+23]. The best performing values are highlighted in bold.

S2EF	Energy MAE ⇓		Force MAE ⇓		Force Cosine ⇑		EFwT ⇑	
	ID	OOD	ID	OOD	ID	OOD	ID	OOD
SchNet	7.92	7.93	0.06	.082	.363	.22	0%	0%
DimeNet++	2.08	2.48	.043	.059	.61	.44	0%	0%
GemNet	.37	**.83**	.03	**.04**	.69	.55	**.02%**	0%
SGRLvCLT-SIRGN	**.37**	.86	**.03**	.04	**.7**	**.57**	.01%	0%

Comparison to the state of the art – PINN. In our experiment, we compare our approach to three state-of-the-art physically informed neural networks (PINNs) [KGMG20], [GSS+22], [SKF+17], which incorporate specialized loss functions to enforce specific physical constraints. Performance results for these models were obtained from literature values [TLS+23] to provide a direct comparison.

As shown in Table SI, our method, when combined with the SIR-GN model, achieves performance comparable to or exceeding that of specialized PINN-based approaches such as GemNet. Notably, our approach remains general and independent of any explicit physical constraints, demonstrating its effectiveness without requiring domain-specific modifications.

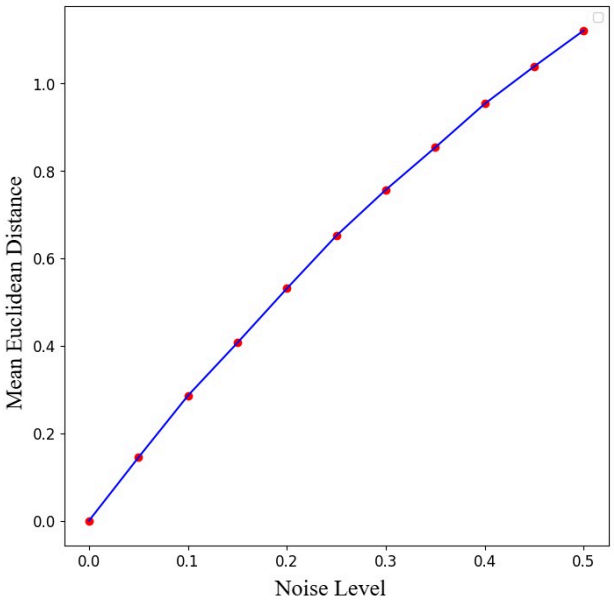

Fig. SI. Mean Euclidean distance between the canonical locations of a perturbed graph, and its unperturbed counterpart. The perturbation applies random rotations and translations, with random noise in the range $\gamma \in (0, 0.5)$ with increments of 0.05.

SII. Robustness to noise of our rotation-translation invariant transformation

To evaluate the robustness of the proposed SGRLvCLT method and confirm its invariance to rotation and translation, Figure SI presents the mean Euclidean distance between the canonical locations of a perturbed graph, $\tilde{G}$, and its unperturbed counterpart, G. The perturbation applies random rotations and translations, with random noise on top, which ranges from $\gamma \in (0, 0.5)$ with increments of 0.05. When $\gamma = 0$, the mean distance is zero, verifying invariance to rotation and translation. As γ increases, the distance increases and follows a generally linear trend. This confirms that our approach successfully preserves spatial relationships under these transformations, validating its effectiveness in ensuring geometric invariance.

References

[CDG+21] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 11(10):6059–6072, 2021.

[GSS+22] Johannes Gasteiger, Muhammed Shuaibi, Anuroop Sriram, Stephan Günnemann, Zachary W. Ulissi, C. Lawrence Zitnick, and Abhishek Das. GemNet-OC: Developing graph neural networks for large and diverse molecular simulation datasets. *TMLR*, 2022.

[KGMG20] Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *CoRR*, abs/2011.14115, 2020.

[SKF+17] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Proc. of NeurIPS Conf.*, pages 991–1001, 2017.

[TLS+23] Richard Tran, Janice Lan, Muhammed Shuaibi, Brandon Wood*, Siddharth Goyal, Abhishek Das, Javier Heras-Domingo, Adeesh Kolluru, Ammar Rizvi, Nima Shoghi, Anuroop Sriram, Zachary Ulissi, and C. Lawrence Zitnick. The open catalyst 2022 (oc22) dataset and challenges for oxide electrocatalysts. *ACS Catalysis*, 2023.

[ZCD+20] C. Lawrence Zitnick, Lowik Chanussot, Abhishek Das, Siddharth Goyal, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Thibaut Lavril, Aini Palizhati, Morgane Riviere, Muhammed Shuaibi, Anuroop Sriram, Kevin Tran, Brandon M. Wood, Junwoong Yoon, Devi Parikh, and Zachary W. Ulissi. An introduction to electrocatalyst design using machine learning for renewable energy storage. *CoRR*, abs/2010.09435, 2020.