Is Distance Matrix Enough for Geometric Deep Learning?

Anonymous Authors[1]

Abstract

Graph Neural Networks (GNNs) are often used for tasks involving the geometry of a given graph, such as molecular dynamics simulation. Although the distance matrix of a geometric graph contains complete geometric information, it has been demonstrated that Message Passing Neural Networks (MPNNs) are insufficient for learning this geometry. In this work, we expand on the families of counterexamples that MPNNs are unable to distinguish from their distance matrices, by constructing families of novel and symmetric geometric graphs. We then propose k-DisGNNs, which can effectively exploit the rich geometry contained in the distance matrix. We demonstrate the high expressive power of our models and prove that some existing well-designed geometric models can be unified by k-DisGNNs as special cases. Most importantly, we establish a connection between geometric deep learning and traditional graph representation learning, showing that those highly expressive GNN models originally designed for graph structure learning can also be applied to geometric deep learning problems with impressive performance, and that existing complex, equivariant models are not the only solution. Experimental results verify our theory.

1. Introduction

Many real-world tasks are relevant to learning the geometric structure of a given graph, such as molecular dynamics simulation, physical simulation, drug designing and protein structure prediction. (Schmitz et al., 2019; Sanchez-Gonzalez et al., 2020; Jumper et al., 2021; Guo et al., 2020). Usually in these tasks, the coordinates of nodes and their individual properties, such as atomic numbers, are given. The goal is to accurately predict both invariant properties, like the energy of the molecule, and equivariant properties, such as the force acting on each atom. This kind of graphs is also referred to as *geometric graphs* by researchers (Bronstein et al., 2021; Han et al., 2022).

In recent years, Graph Neural Networks (GNNs) have achieved outstanding performance on such tasks, as they can learn a representation for each graph or node in an end-to-end fashion, rather than relying on handcrafted features. It has led to the development of various GNNs which go beyond simply using *distance*, the most basic invariant geometric feature, as input. Instead, these models use complex irreducible representations, low-order equivariant representations, or manually-designed complex invariant geometric features such as angles or dihedral angles to learn better representations from geometric graphs. It is generally believed that pure distance information is insufficient for complex geometric deep learning (GDL) tasks (Klicpera et al., 2020b; Schütt et al., 2021).

On the other hand, it is well known that the *distance matrix*, which contains the distances between all pairs of nodes in a geometric graph, holds all of the geometric structure information (Satorras et al., 2021). This suggests that it is possible to obtain all of the desired geometric structure information from *complete distance graphs*, i.e., complete weighted graphs with distance as edge weight. Therefore, the complex geometric deep learning task with node coordinates as input may be transformed into an equivalent graph representation learning task with the complete distance graph as input.

Lots of previous works focus on explaining that MPNNs (Gilmer et al., 2017) are *insufficient* for learning the geometry from *incomplete* distance graphs (where a cutoff distance is used to remove long edges). They provide pairs of non-isomorphic incomplete distance graphs that cannot be distinguished by MPNNs (Schütt et al., 2021; Zhang et al., 2021; Garg et al., 2020). However, as the cutoff of the counterexamples becomes larger, the graphs can once again be distinguished by MPNNs. Note that Pozdnyakov & Ceriotti (2022) constructed both finite-size and infinite-size (periodic) counterexamples with infinite cutoff, and demonstrated that the counterexamples include chemically-plausible configurations, which essentially gave the first families of valid counterexamples and thus proved

[1] Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

that MPNNs are incomplete even when taking the entire distance matrix as input. In our effort to enrich the set of counterexamples, we make a twofold contribution. Firstly, we exhaustively sample nodes from regular polyhedrons to construct a large number of counterexamples. These counterexamples exhibit high symmetry and diversity and are easy to understand. Secondly, we introduce a novel method to construct families of counterexamples based on these basic units, significantly enriching the current set of counterexample families.

Given the limitations of MPNNs in learning geometry, we propose k-DisGNN and its variants k-F-DisGNN/k-E-DisGNN which take complete distance graphs as input, mainly based on the well-known k-(F)WL test. We theoretically demonstrate their superior ability to learn high-order geometric information. We also show that two state-of-the-art models, DimeNet (Klicpera et al., 2020b) and GemNet (Gasteiger et al., 2021), can be implemented by our models and thus are **special cases** of k-DisGNNs. A key insight is that these two models are augmented with manually-designed high-order geometric features including angles (three-node features) and dihedral angles (four-node features), which correspond to the k-tuples in k-(F)WL test. However, our models can learn more than these handcrafted features in an end-to-end fashion. We conduct experiments on benchmark datasets. Our models achieve state-of-the-art results on a wide range of the targets in the MD17 dataset.

To summarize, our main contributions are:

1. We enrich the counterexample families that MPNNs cannot distinguish even taking the whole distance matrix as input, by proposing diverse families of novel and symmetric counterexamples.
2. Inspired by k-(F)WL, we propose **k-DisGNNs** with provably higher expressive power than MPNNs for learning from distance graphs.
3. We **unify** two classical GDL models, DimeNet and GemNet, with k-DisGNNs.
4. Our models outperform previous **state-of-the-art** models on a wide range of targets of the MD17 dataset.
5. We establish a **connection** between geometry deep learning and traditional graph representation learning through distance graphs.

2. Related Work

Expressiveness of GNNs. It has been proven that the expressiveness of MPNNs is limited by the Weisfeiler-Leman test (Weisfeiler & Leman, 1968; Xu et al., 2018; Morris et al., 2019), a classical algorithm for graph isomorphism test. While MPNNs can distinguish most graphs (Babai & Kucera, 1979), they are unable to count rings, triangles, or distinguish regular graphs, which are common in real-world data such as molecules (Huang et al., 2023). To increase the expressiveness and design space of GNNs, Morris et al. (2019; 2020) proposed high-order GNNs based on k-WL. Maron et al. (2019) designed GNNs based on the folklore WL (FWL) test (Cai et al., 1992) and Azizian & Lelarge (2020) showed that these GNNs are the most powerful GNNs for a given tensor order. These works mainly focus on unweighted graphs. In the context of distinguishing distance graphs, it has also been demonstrated that MPNNs are incomplete for distance graphs with finite cutoffs (Zhang et al., 2021; Garg et al., 2020; Schütt et al., 2021). Pozdnyakov & Ceriotti (2022) first proved MPNNs are incomplete even when the cutoff is infinite by giving both finite-size and infinite-size (periodic) counterexamples, and demonstrated the counterexamples include real chemical structures. They for the first time demonstrated the inherit limitations of MPNNs on distance matrices.

Equivariant Neural Networks. Symmetry is a rather important design principle for geometric deep learning (Bronstein et al., 2021). In the context of geometric graphs, it is desirable for models to be equivariant or invariant under the E(3) or SO(3) transformation of the input graphs. These models include those using group irreducible representations (Thomas et al., 2018; Batzner et al., 2022; Anderson et al., 2019; Fuchs et al., 2020), invariant geometric features (Schütt et al., 2018; Klicpera et al., 2020b; Gasteiger et al., 2021) and low-order equivariant representations (Satorras et al., 2021; Schütt et al., 2021; Thölke & De Fabritiis, 2021). Particularly, Dym & Maron (2020) proved that both Thomas et al. (2018) and Fuchs et al. (2020) are universal for SO(3)-equivariant functions. Besides, Villar et al. (2021) showed that one could construct universal equivariant outputs by leveraging invariances, highlighting the potential of invariant models to learn equivariant targets.

More related work can be referred to Appendix F.

3. Preliminaries

In this paper, we use the following notations. We denote multiset with $\{\!\{\}\!\}$. We use $[n]$ to represent the set $\{1, 2, ..., n\}$. A complete weighted graph with n nodes is denoted by $G = (V, \boldsymbol{E})$, where $V = [n]$ and $\boldsymbol{E} = [e_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$. The neighborhoods of node i are denoted by $N(i)$. For two tensors $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n^k \times d}$, if there exists some permutation π such that $\forall a_i \in [n], i \in [k], j \in [d]$, $\mathbf{A}_{\pi(a_1),\pi(a_2),...,\pi(a_k),j} = \mathbf{B}_{a_1,a_2,...,a_k,j}$, we will write $\mathbf{A} =_p \mathbf{B}$.

Distance graph. In many tasks, we need to deal with geometric graphs (Han et al., 2022; Bronstein et al., 2021), where each node i is attached with its 3D coordinates $\mathbf{x}_i \in \mathbb{R}^3$ in addition to other invariant features. The geometric part of geometric graphs can be represented by $\mathbf{X}^{n\times 3}$, i.e., stacking n nodes' coordinates. Geometric graphs contain rich geometric information useful for learning chemical or physical properties. Corresponding to geometric graphs are distance graphs, which we define as follows:

Definition 3.1. Distance graph G is a complete weighted graph where $e_{ij} = \|\mathbf{x_i} - \mathbf{x_j}\|$. Here, $\|\|$ is the L^2 norm.

Note that distance graphs **do not have coordinates** attached to each node. Distance graphs can also be used to represent real-world graphs like molecules, but usually people consider a graph with **cutoff** r, meaning that only edges with $e_{ij} \leq r$ are considered. When $r \to +\infty$, all-pair distances are considered, and the adjacency matrix of such a distance graph is thus the distance matrix (denoted by $\mathrm{DM}(G)$). We also call these graphs complete distance graphs, in contrast to incomplete ones using a cutoff. *We mainly focus on complete distance graphs in this paper*. It is important to note that a complete distance graph contains all the **geometric structure information**, as stated in the following theorem:

Theorem 3.2. *(Satorras et al., 2021) Let G_1, G_2 be two n-size geometric graphs with nodes' coordinates denoted by $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times 3}$, then $\mathrm{DM}(G_1) =_p \mathrm{DM}(G_2) \iff$ there exists some $\mathbf{Q} \in SO(3)$ and $\mathbf{T} \in \mathbb{R}^3$,* s.t. $\mathbf{XQ^T} + \mathbf{T} =_p \mathbf{Y}$.

The above theorem indicates that we can recover the full geometry (node coordinates) of a system from its distance matrix up to rotations and translations. This result is important, as it provides possibilities to learn all E(3)-invariant features merely from the distance matrix, and combining with the system orientation we may be able to learn all equivariant features too (Villar et al., 2021).

Taking distance graphs as input instead of geometric graphs mainly has three advantages:

1. Distance graphs are just a special kind of weighted graphs, thus **traditional GNNs can be applied to them without any modification**.
2. All features in distance graphs are invariant to E(3)-transformation, thus models taking distance graphs as input are **naturally invariant models**.
3. Complete distance graph contains all the geometric structure information, thus sufficiently expressive models can in principle exploit all the geometric information solely from it.

Weisfeiler-Leman Algorithms. Weisfeiler-Lehman test (also called as 1-WL) (Weisfeiler & Leman, 1968) iteratively updates the labels of nodes according to nodes' own labels and their neighbors' labels, and compares the histograms of the labels to distinguish two graphs. Specifically, we use l_i^t to denote the label of node i at iteration t, then 1-WL updates the node label by

$$l_i^{t+1} = \mathrm{hash}(l_i^t, \{\!\{l_j^t \mid j \in N(i)\}\!\}), \tag{1}$$

where $N(i)$ denotes the set of neighbors of node i, and hash is an injective function that maps different inputs to different labels. However, 1-WL cannot distinguish all the graphs, and thus k-dimensional WL (k-WL) and k-dimensional Folklore WL (k-FWL) $k \geq 2$ are proposed to boost the expressiveness of WL test.

Instead of updating the label of nodes, k-WL and k-FWL update the label of k-tuples $\boldsymbol{v} := (v_1, v_2, ..., v_k) \in V^k$, denoted by $l_{\boldsymbol{v}}$. Both methods initialize k-tuples' labels according to their isomorphic types and update the labels iteratively according to their j-neighbors $N_j(\boldsymbol{v})$. The difference between k-WL and k-FWL mainly lies in the definition of $N_j(\boldsymbol{v})$. To be specific, tuple $\boldsymbol{v}$'s j-neighbors in k-WL and k-FWL are defined as follows respectively

$$N_j(\boldsymbol{v}) = \{\!\{(v_1, ..., v_{j-1}, w, v_{j+1}, ..., v_k) \mid w \in V\}\!\}, \tag{2}$$

$$N_j^F(\boldsymbol{v}) = \big((j, v_2, ..., v_k), ..., (v_1, v_2, ..., j)\big). \tag{3}$$

To update the label of each tuple, k-WL iterates as follows

$$l_{\boldsymbol{v}}^{t+1} = \mathrm{hash}\Big(l_{\boldsymbol{v}}^t, \big(L_1^t(\boldsymbol{v}), L_2^t(\boldsymbol{v}), ..., L_k^t(\boldsymbol{v})\big)\Big), \tag{4}$$

$$\text{where } L_j^t(\boldsymbol{v}) = \{\!\{l_{\boldsymbol{w}}^t \mid \boldsymbol{w} \in N_j(\boldsymbol{v})\}\!\}. \tag{5}$$

And k-FWL iterates as follows

$$l_{\boldsymbol{v}}^{t+1} = \mathrm{hash}\Big(l_{\boldsymbol{v}}^t, \{\!\{L_j^{\mathrm{F},t}(\boldsymbol{v}) \mid j \in [|V|]\}\!\}\Big), \tag{6}$$

$$\text{where } L_j^{\mathrm{F},t}(\boldsymbol{v}) = \big(l_{\boldsymbol{w}}^t \mid \boldsymbol{w} \in N_j^F(\boldsymbol{v})\big). \tag{7}$$

Note that in Equation (7), the order of the tuple $L_j^{\mathrm{F},t}(\boldsymbol{v})$ is the same as that of $N_j^F(\boldsymbol{v})$.

It is worth mentioning that k-(F)WL methods were originally proposed to distinguish unweighted graphs, where edge information is binary (exist or not). However, they can still be applied to weighted graphs with only a slight modification to the initial step: we can initialize k-tuples according to not only the connectivity among nodes but also the edge weights. However, if we want to take edge weight into consideration in 1-WL, we need to modify its message passing function, which is discussed in Section 4.

4. Incompleteness of MPNNs on Distance Graphs

To adapt 1-WL to distance graphs, a natural way is to consider both node label and edge weight when aggregating neighbors, which is called edge-enhanced 1-WL (denoted by 1-WL-E) (Pozdnyakov & Ceriotti, 2022):

$$l_i^{t+1} = \mathrm{hash}(l_i^t, \{\!\{(l_j^t, e_{ij}) \mid j \in N(i)\}\!\}). \tag{8}$$

In this context, previous research (Zhang et al., 2021; Garg et al., 2020; Schütt et al., 2021) has attempted to construct counterexamples that cannot be distinguished by 1-WL-E. However, these constructed distance graphs are all incomplete and can be once again distinguished by increasing the

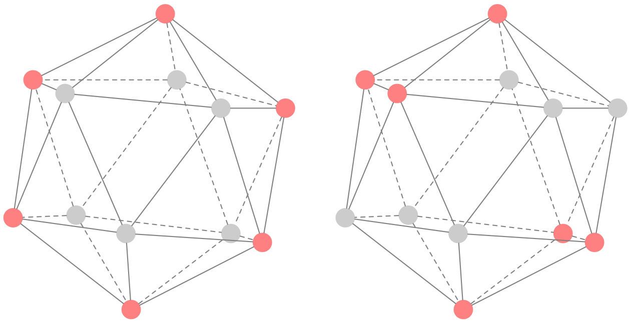


Figure 1. A pair of complete distance graphs that are non-isomorphic but cannot be distinguished by MPNNs/1-WL-E. Only red nodes belong to the two distance graphs. The grey nodes and the "edges" are for visualization purpose only. Note that the nodes of the distance graphs are sampled from regular icosahedrons.

cutoff, suggesting that *higher cutoffs lead to greater discriminatory power*. To provide valid counterexamples with infinite cutoff, Pozdnyakov & Ceriotti (2022) proposed both finite-size and infinite-size (periodic) counterexamples and showed that they included real chemical structures. This work demonstrated for the first time the inherent limitations of MPNNs on distance matrices. However, it should be noted that the essential change between these counterexamples is merely the distortion of $C^{\pm}$ atoms, which may lack diversity despite having high manifold dimensions. In this regard, constructing new families of counterexamples can not only enrich the diversity of existing families, but also demonstrate MPNNs' limitations from different angles.

Increasing the diversity of counterexamples has high theoretical values, too. If only a few carefully designed and complex counterexamples exist, then these counterexamples may not significantly impact the performance of MPNNs on real-world graphs. However, if one can show that even some simple and common distance graphs (not restricted to molecules, but can be point clouds) are also indistinguishable by MPNNs, then we may say MPNNs are not enough for distance graphs with more confidence. Additionally, insights gained from novel counterexamples can inspire the development of more powerful geometric GNNs, as commonly seen in traditional graph representation learning (Zhang & Li, 2021; Huang et al., 2023).

In this paper, we give a simple valid counterexample which MPNNs/1-WL-E cannot distinguish even with an infinite cutoff, as shown in Figure 1. In both graphs, all nodes have exactly the same unordered list of distances (infinite cutoff considered), which means that 1-WL-E will always label them identically. Nevertheless, the two distance graphs are obviously non-isomorphic, since there are two equilateral triangles on the right, and zero on the left.

Beyond this, we construct three kinds of counterexamples, including several individual counterexamples and several families of counterexamples by arbitrarily combining some basic units. This essentially gives an infinite number of diverse and symmetric counterexamples, which can significantly enriches the counterexamples found by Pozdnyakov & Ceriotti (2022). The three kinds are

1. Several individual counterexamples sampled from regular icosahedron and regular dodecahedron that can be directly verified (Appendix A.1).
2. Two small families of counterexamples that can be transformed in one dimension (Appendix A.2).
3. Several families of counterexamples constructed by arbitrary combinations of some basic units (Appendix A.3).

These counterexamples further demonstrate the incompleteness of MPNNs/1-WL-E in learning the geometry of a distance graph: even quite simple graphs as shown in Figure 1 cannot be distinguished by them. Although there may not be real molecules corresponding to our constructed counterexamples, such symmetric structures are commonly seen in other geometry-relevant tasks, such as physical simulations and point clouds. In summary, we construct novel families of counterexamples to reveal the inherent limitation of MPNNs for distance graphs, which not only enrich the existing families proposed by Pozdnyakov & Ceriotti (2022), but also are easier to grasp and help understand MPNNs' limitation in a more intuitive way.

5. k-DisGNNs

Since vanilla MPNNs are unable to fully utilize the geometric information contained in distance graphs, is it possible to design more powerful GNNs to do so (without explicitly calculating high-order geometric features such as angles)? In this section, we propose the framework of k-DisGNNs, which are provably more expressive models to extract geometric information from distance graphs. We first introduce the framework, and will delve deeper into the expressiveness of k-DisGNNs in Section 6. To contrast the order of geometric features leveraged, we also call MPNNs *vanilla DisGNNs*, and call our proposed models *high-order DisGNNs*. Note that we only show the basic forms of functions in this section, and the detailed implementation is referred to Appendix C.

k-DisGNNs consist of three versions: k-DisGNN, k-F-DisGNN, and k-E-DisGNN. These models initialize the embedding $h_{\boldsymbol{v}}^0$ of k-tuple $\boldsymbol{v}$ in a distance graph and update them iteratively in a message passing way. All versions of high-order DisGNNs share the same initialization and output block but differ in their message passing blocks.

Note that high-order DisGNNs differ from traditional k-(F)WL algorithms and their neural counterparts (Morris et al., 2019; Maron et al., 2019) in that they take complete distance graphs as input and aim to learn their **geometric representations**. In this context, all the k-tuples are actually **geometric tuples** because they also contain information about all-pair distances within the nodes of the tuple.

Initialization Block. Given a geometric k-tuple $\boldsymbol{v}$, high-order DisGNNs initialize it with an injective function by $h_{\boldsymbol{v}}^0 = f_{\text{init}}\Big(\big(z_{v_i} \mid i \in [k]\big), \big(e_{v_i v_j} \mid i,j \in [k], i < j\big)\Big)$. This function can injectively embed the *ordered* distance matrix along with the z type of each node within the k-tuple, thus preserving all the geometric information within it.

Message Passing Block. The key difference between the three versions of high-order DisGNNs lies in their message passing blocks. The message passing blocks of k-DisGNN and k-F-DisGNN are based on the paradigms of k-WL and k-FWL, respectively. Their core message passing function f_{MP}^t and $f_{\text{MP}}^{\text{F},t}$ are formulated simply by replacing the tuple labels $l_{\boldsymbol{v}}$ in k-(F)WL with geometric tuple representation $h_{\boldsymbol{v}}$, giving

$$h_{\boldsymbol{v}}^{t+1} = f_{\text{MP}}^t\Big(h_{\boldsymbol{v}}^t, \big(H_1^t(\boldsymbol{v}), H_2^t(\boldsymbol{v}), ..., H_k^t(\boldsymbol{v})\big)\Big), \quad (9)$$

$$\text{where } H_j^t(\boldsymbol{v}) = \{\!\{h_{\boldsymbol{w}}^t \mid \boldsymbol{w} \in N_j(\boldsymbol{v})\}\!\}. \quad (10)$$

$$h_{\boldsymbol{v}}^{t+1} = f_{\text{MP}}^{\text{F},t}\Big(h_{\boldsymbol{v}}^t, \{\!\{H_j^{\text{F},t}(\boldsymbol{v}) \mid j \in [|V|]\}\!\}\Big), \quad (11)$$

$$\text{where } H_j^{\text{F},t}(\boldsymbol{v}) = \big(h_{\boldsymbol{w}}^t \mid \boldsymbol{w} \in N_j^F(\boldsymbol{v})\big). \quad (12)$$

Tuples containing local geometric information interact with each other in message passing blocks, thus allowing models to learn considerable global geometric information.

However, during message passing in k-DisGNN, the information about distance is not explicitly used but is embedded implicitly in the initialization block when each geometric tuple is embedded according to its distance matrix and node types. This means that k-DisGNN is **unable to capture the distance** between a k-tuple $\boldsymbol{v}$ and its neighbor $\boldsymbol{w}$ during message passing, which could be very helpful for learning the geometric structural information of the graph. For example, in a physical system, a tuple $\boldsymbol{w}$ far from $\boldsymbol{v}$ should have much less influence on $\boldsymbol{v}$ than another tuple $\boldsymbol{w}'$ near $\boldsymbol{v}$.

Based on this observation, we propose k**-E-DisGNN**, which maintains a representation e_{ij}^t for each edge e_{ij} at every time step and explicitly incorporates it into the message passing procedure.

The edge representation e_{ij}^t is updated by

$$e_{ij}^t = f_{\text{e}}^t\Big(e_{ij}, \big(H_{uv}^t(i,j) \mid u,v \in [k], u < v\big)\Big), \quad (13)$$

$$\text{where } H_{uv}^t(i,j) = \{\!\{h_{\Phi_{uv}(\boldsymbol{w},i,j)}^t \mid \boldsymbol{w} \in V^k\}\!\}. \quad (14)$$

Here, the function $\Phi_{uv}(\boldsymbol{w}, i, j)$ replaces the u^{th} and v^{th} elements in vector $\boldsymbol{w}$ with values i and j, respectively. Note that e_{ij}^t not only contains the distance between nodes i and j, but also pools all the tuples related to edge ij, making it informative and general. For example, in the special case $k = 2$, Equation (13) is equivalent to $e_{ij}^t = f_{\text{e}}^t(e_{ij}, h_{ij}^t)$.

We realize the message passing function $f_{\text{MP}}^{\text{E},t}$ by replacing $H_j^t(\boldsymbol{v})$ in Equation (9) with $H_j^{\text{E},t}(\boldsymbol{v})$ defined as follows

$$H_j^{\text{E},t}(\boldsymbol{v}) = \{\!\{(h_{\boldsymbol{w}}^t, e_{\boldsymbol{v}\backslash\boldsymbol{w},\boldsymbol{w}\backslash\boldsymbol{v}}^t) \mid \boldsymbol{w} \in N_j(\boldsymbol{v})\}\!\}, \quad (15)$$

where $\boldsymbol{v}\backslash\boldsymbol{w}$ gives the only element in $\boldsymbol{v}$ but not in $\boldsymbol{w}$. In other words, $e_{\boldsymbol{v}\backslash\boldsymbol{w},\boldsymbol{w}\backslash\boldsymbol{v}}^t$ gives the representation of the edge connecting $\boldsymbol{v}, \boldsymbol{w}$. By this means, a tuple can be aware of **how far** it is to its neighbors and by what kind of edges each neighbor is connected. This can boost the ability of geometric structure learning. Note that the calculation of e_{ij}^t does not increase the time complexity of k-DisGNN, which is detailed in Appendix C.

Output Block. The output function $t = f_{\text{out}}\big(\{\!\{h_{\boldsymbol{v}}^T \mid \boldsymbol{v} \in V^k\}\!\}\big)$, where T is the final iteration, is a multiset function. It pools all the tuple representations injectively, and generates the invariant geometric target t.

Expressive Power. Here we discuss the expressive power of the three variants through analyzing the functions these models can approximate (Chen et al., 2019). To formalize this, given two neural networks $\mathcal{A}$ and $\mathcal{B}$, if all the functions model $\mathcal{A}$ can approximate can also be approximated by model $\mathcal{B}$, then we denote it by $\mathcal{A} \sqsubseteq \mathcal{B}$, and say that $\mathcal{B}$ is not less powerful/expressive than $\mathcal{A}$. And if the relation holds strictly, we denote it by $\mathcal{A} \sqsubset \mathcal{B}$, and say that $\mathcal{B}$ is more powerful/expressive than $\mathcal{A}$.

We characterize the expressiveness of k-DisGNNs with

Theorem 5.1. *k-DisGNN $\sqsubseteq$ k-E-DisGNN $\sqsubseteq$ k-F-DisGNN for any $k \geq 2$.*

In addition, like the conclusion about k-(F)WL for unweighted graphs, *the expressiveness of DisGNNs does not decrease as k increases*, i.e., k-DisGNNs $\sqsubseteq$ $(k+1)$-DisGNNs (hold for all three versions). This is simply because all k-tuples are contained within some $(k+1)$-tuples, and by designing message passing that ignores the last index we can realize k-DisGNNs with $(k+1)$-DisGNNs.

At the end of this section, we note that whether $\sqsubseteq$ discussed above can be replaced by $\sqsubset$ is unknown as of now. This is because constructing a pair of complete distance graphs that cannot be distinguished by some k-DisGNNs but can be distinguished by another k-DisGNNs or $(k+1)$-DisGNNs is challenging. The reason has been discussed in Section 4, and previous counterexamples (Cai et al., 1992) demonstrating that k-(F)WL is strictly less powerful than $(k+1)$-(F)WL are not valid anymore. We leave this topic for future work.

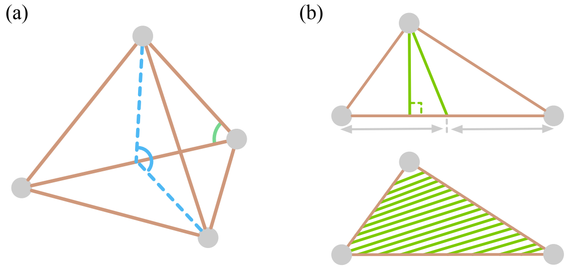


Figure 2. High-order geometric information contained in the distance matrix of k-tuples. We mark different orders with different colors, with brown for 2-order, green for 3-order, and blue for 4-order. (a) The high-order geometric information contained in 4-tuples, including distances, angles and dihedral angles. (b) More 3-order geometric features, such as vertical lines and middle lines of triangles, as well as the area of triangles.

6. Rich geometric information in k-DisGNNs

In this section, we will discuss more about the geometry learning ability of k-DisGNNs from different perspectives. In Subsection 6.1, we will discuss what **high-order geometric features** the models can extract from distance graphs. In Subsection 6.2, we will show that DimeNet and GemNet, two classical GNNs for GDL using invariant geometric representations, are just **special cases** of our k-DisGNNs, thus showcasing the high expressiveness of our models.

6.1. Ability of learning high-order geometry

We first give the concept of *high-order geometric information* for better understanding.

Definition 6.1. k-order geometric information is the **E(3)-invariant** features calculable from k nodes' 3D coordinates.

For example, in a 4-tuple, one can find various high-order geometric information, including distance (2-order), angles (3-order) and dihedral angles (4-order), as shown in Figure 2a. Note that high-order information is not limited to the common features listed above: we show some of other possible 3-order geometric features in Figure 2b.

We first note that k-DisGNNs can learn and process at least k-order geometric information. Theorem 3.2 states that we can reconstruct the whole geometry of the k-tuple from the embedding of its distance matrix. In other words, we can calculate all the desired k-order geometric information solely from the embedding of the k-tuple's distance matrix with a learnable function.

In contrast, these high-order geometric information are hardly learnable by vanilla DisGNNs (MPNNs). For example, the two graphs in Figure 1 can be easily distinguished by 3-DisGNNs, because they can count how many *equilateral triangles* each graph has at the initial step, which cannot be done by vanilla DisGNNs even infinite message passing layers are stacked up. This example also demonstrates **high-order DisGNNs are strictly more powerful than vanilla DisGNNs**.

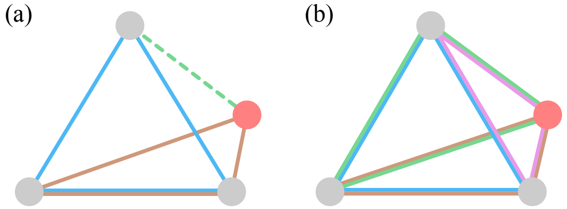


Figure 3. Examples explaining that two neighboring 3-tuples can form a 4-tuple. Blue represents the center tuple, the other colors represent the neighbor tuples, and the red node is the one node of that neighbor tuple which is not in the center tuple. (a) Example for 3-E-DisGNN. With the green edge, the two 3-tuples can form a 4-tuple. (b) Example for 3-F-DisGNN. Four 3-tuples form a 4-tuple.

Furthermore, both k-E-DisGNN and k-F-DisGNN can actually **learn $(k+1)$-order geometric information** in their message passing layers.

For k-E-DisGNN, information about $(h_{\boldsymbol{v}}, h_{\boldsymbol{w}}, e_{\boldsymbol{v}\backslash\boldsymbol{w},\boldsymbol{w}\backslash\boldsymbol{v}})$, where $\boldsymbol{w}$ is some j-neighbor of $\boldsymbol{v}$, is included in the input of its update function. Since the distance matrices of tuples $\boldsymbol{v}$ and $\boldsymbol{w}$ can be reconstructed from $h_{\boldsymbol{v}}$ and $h_{\boldsymbol{w}}$, the all-pair distances of $(v_1, v_2, ..., v_k, \boldsymbol{w}\backslash\boldsymbol{v})$ can be reconstructed from $(h_{\boldsymbol{v}}, h_{\boldsymbol{w}}, e_{\boldsymbol{v}\backslash\boldsymbol{w},\boldsymbol{w}\backslash\boldsymbol{v}})$, as shown in Figure 3a. This means that k-E-DisGNN can construct the distance matrix of such $(k+1)$-tuple. Similarly, the distance matrix of $(v_1, v_2, ..., v_k, \boldsymbol{w}\backslash\boldsymbol{v})$ can also be reconstructed from $H_j^{\mathrm{F},t}$ in Equation (12) as shown in Figure 3b. This enables k-E-DisGNN and k-F-DisGNN to learn all the $(k+1)$-order geometric information contained in the graph.

The ability to learn and process high-order geometric information makes k-DisGNNs effective and powerful for geometry learning. As the value of k increases, the high-order geometric information they can learn becomes more extensive, and when k reaches n (the size of the graph), the entire distance matrix of the graph is embedded at the initial step, resulting in the achievement of universality.

6.2. Unifying existing geometric models with DisGNNs

There have been many attempts to improve GDL models by *manually designing and incorporating high-order geometric features* such as angles (3-order) and diherdal angles (4-order). These features are all invariant geometric features that can be learned by some k-DisGNNs. It is therefore natural to ask whether the approximating function space of k-DisGNNs can incorporate these models. In this subsection, we show that two classical models, DimeNet (Klicpera et al., 2020b) and GemNet (Gasteiger et al., 2021), are special cases of k-DisGNNs, thus unifying existing methods based on hand-crafted features with a learning paradigm that learns **arbitrary** high-order features from distance matrix.

DimeNet proposed a message passing scheme analogous to belief propagation which incorporates angle information into it. To be specific, DimeNet embeds the atom i with a set of incoming messages m_{ji}, i.e., $h_i = \sum_{j \in N_i} m_{ji}$, and updates the message m_{ji} by

$$m_{ji}^{(l+1)} = f_{\text{MP}}^{\text{D}}\big(m_{ji}^{l}, \sum_{k} f_{\text{int}}^{\text{D}}(m_{kj}^{(l)}, d_{ji}, \phi_{kji})\big), \qquad (16)$$

where d_{ji} is the distance between node j and node i, and ϕ_{kji} is the angle kji. We simplify the subscript of $\sum$, same for that in Equation (17). The detailed ones are referred to Appendix B.2, B.1.

DimeNet is one early approach that uses geometric features to improve the geometry learning ability of GNNs, especially useful for learning the angle-relevant energy or other chemical properties. Note that the angle information that DimeNet explicitly incorporates into its message passing function is actually a kind of 3-order geometric information, which can be exploited from distance matrices by our 2-F-DisGNN with a learning paradigm. This gives the key insight for the following proposition.

Proposition 6.2. *DimeNet $\sqsubseteq$ 2-F-DisGNN.*

GemNet is also a graph neural network designed to process graphs embedded in Euclidean space. Unlike DimeNet, GemNet incorporates both angle and dihedral angle information into the message passing functions, with a 2-step message passing scheme. The core procedure is as follows:

$$m_{ca}^{(l+1)} = f_{\text{MP}}^{\text{G}}\big(m_{ca}^{l}, \sum_{b,d} f_{\text{int}}^{\text{G}}(m_{db}^{(l)}, d_{db}, \phi_{cab}, \phi_{abd}, \theta_{cabd})\big), \qquad (17)$$

where θ_{cabd} is the dihedral angle of planes cab and abd.

The use of dihedral angle information allows GemNet to learn at least 4-order geometric information, making it much more powerful than models that only consider angle information. In the following proposition, we prove that GemNet is just a special case of 3-E-DisGNN.

Proposition 6.3. *GemNet $\sqsubseteq$ 3-E-DisGNN.*

It is worth noting that while both our models and existing models can learn some k-order geometric information, our models have the advantage of **learning arbitrary k-order geometric information in an end-to-end fashion**. This means that we can learn different high-order geometric features according to specific tasks, including but not limited to angles and dihedral angles.

7. Experiments

In this section, we evaluate the performance of k-DisGNNs. Our main objectives are to answer the following questions:

1. Does 2-F-DisGNN outperform DimeNet in experiments? (Corresponding to Proposition 6.2)
2. Does 3-E-DisGNN outperform GemNet in experiments? (Corresponding to Proposition 6.3)
3. Does incorporating well-designed edge representations in the k-E-DisGNN result in improved performance?

The best and the second best results are shown in bold and underline respectively in tables. The specific experimental settings can be found in Appendix D. Supplementary explanation and experiments can be found in Appendix E.

MD17. MD17 (Chmiela et al., 2017) is a dataset commonly used to evaluate the performance of machine learning models in the field of molecular dynamics. It contains a collection of molecular dynamics simulations of small organic molecules such as aspirin and ethanol. Given the atomic numbers and coordinates, the task is to predict the energy of the molecule and the atomic forces. We mainly focus on the comparison between 2-F-DisGNN/3-E-DisGNN and DimeNet/GemNet. At the same time, we also compare our models with the state-of-the-art models: FCHL (Christensen et al., 2020), PaiNN (Schütt et al., 2021), NequIP (Batzner et al., 2022), TorchMD (Thölke & De Fabritiis, 2021), GNN-LF (Wang & Zhang, 2022). The results are shown in Table 1. 2-F-DisGNN and 3-E-DisGNN outperform their counterparts on 16/16 and 4/8 targets, respectively, with an average improvement of 44.2% and 6.9%, suggesting that data-driven models can subsume carefully designed manual features given high expressiveness. In addition, our models also achieve the best performance on 9/16 targets, outperforming the best baselines GNN-LF and TorchMD by 6.9% and 12.4%, respectively. Note that GNN-LF and TorchMD are all complex equivariant models. Our results reveal the potential of pure distance-based methods for GDL.

QM9. QM9 (Ramakrishnan et al., 2014; Wu et al., 2018) consists of 134k stable small organic molecules with 19 regression targets. The task is like that in MD17, but this time we want to predict the molecule's properties such as the dipole moment. We mainly compare 2-F-DisGNN with DimeNet on this dataset. The results are shown in Table 2. 2-F-DisGNN outperforms DimeNet on 9/12 targets, by 6.88% on average, which verifies our theory. A full comparison to other state-of-the-art models is included in Appendix E.

Effectiveness of edge representations. To answer the third question, we split the edge representation e_{ij}^{t} in Equation (13) into two parts, namely the edge weight (the first element) and the tuple representations (the second element), and explore whether incorporating the two elements is beneficial. We conduct experiments on MD17 with three versions of 2-DisGNNs, including 2-DisGNN (no edge representation), 2-e-DisGNN (add only edge weight) and 2-E-DisGNN (add the full edge representation). The results are shown in Table 3. 2-E-DisGNN outperforms 2-DisGNN and 2-e-

Table 1. MAE loss on MD17. Energy (E) in kcal/mol, force (F) in kcal/mol/Å. The improvement ratio is calculated relative to DimeNet. Our models rank **top 2** on force prediction (which determines the accuracy of molecular dynamics (Gasteiger et al., 2021)) on average.

Target		FCHL	PaiNN	NequIP	TorchMD	GNN-LF	DimeNet	2F-Dis.	GemNet	3E-Dis.
aspirin	E	0.182	0.167	-	**0.124**	0.1342	0.204	0.1294	-	0.1691
	F	0.478	0.338	0.348	0.255	0.2018	0.499	**0.1408**	0.2168	0.2230
benzene	E	-	-	-	**0.056**	0.0686	0.078	0.0676	-	0.0702
	F	-	-	0.187	0.201	0.1506	0.187	**0.1431**	0.1453	0.1470
ethanol	E	0.054	0.064	-	0.054	0.0520	0.064	**0.0501**	-	0.0518
	F	0.136	0.224	0.208	0.116	0.0814	0.230	**0.0479**	0.0853	0.0576
malonal.	E	0.081	0.091	-	0.079	0.0764	0.104	**0.0736**	-	0.0767
	F	0.245	0.319	0.337	0.176	0.1259	0.383	**0.0872**	0.1545	0.1020
napthal.	E	0.117	0.166	-	**0.085**	0.1136	0.122	0.1133	-	0.1312
	F	0.151	0.077	0.097	0.06	0.0550	0.215	**0.0459**	0.0553	0.0558
salicyl.	E	0.114	0.166	-	**0.094**	0.1081	0.134	0.1073	-	0.1235
	F	0.221	0.195	0.238	0.135	0.1005	0.374	**0.0862**	0.1048	0.1333
toluene	E	0.098	0.095	-	**0.074**	0.0930	0.102	0.0924	-	0.1009
	F	0.203	0.094	0.101	0.066	0.0543	0.216	**0.0416**	0.0600	0.0484
uracil	E	0.104	0.106	-	**0.096**	0.1037	0.115	0.1034	-	0.1055
	F	0.105	0.139	0.173	0.094	**0.0751**	0.301	0.0839	0.0969	0.0952
Avg improv.	E	11.58%	-2.09%	-	26.40%	17.05%	0.00%	18.42%	-	10.25%
Rank		4	7	-	**1**	3	6	2	-	5
Avg improv.	F	31.85%	39.13%	29.86%	52.40%	63.53%	0.00%	70.02%	60.96%	63.68%
Rank		7	6	8	5	3	9	**1**	4	2

Table 2. Comparison of 2-F-DisGNN and DimeNet on QM9.

Target	Unit	DimeNet	2F-Dis.
μ	D	0.0286	**0.0221**
α	a_0^3	0.0469	**0.0455**
ϵ_{HOMO}	meV	27.8	**25.44**
ϵ_{LUMO}	meV	**19.7**	22.05
$\Delta\epsilon$	meV	**34.8**	43.70
$<R^2>$	a_0^2	0.331	**0.1065**
ZPVE	meV	**1.29**	1.3353
U_0	meV	8.02	**7.53**
U	meV	7.89	**7.78**
H	meV	8.11	**7.85**
G	meV	8.98	**8.56**
c_v	cal/mol/K	0.0249	**0.0233**

DisGNN on 14/16 and 11/16 targets, respectively, with an average improvement of 32.4% and 3.20%. This verifies our theory that capturing the *edge feature* connecting two tuples can boost the representation power.

Table 3. Effectiveness of edge representations on MD17. Energy (E) in kcal/mol, force (F) in kcal/mol/Å.

Target		2-Dis.	2e-Dis.	2E-Dis.
aspirin	E	0.2070	**0.1398**	0.1452
	F	0.4428	0.1679	**0.1596**
benzene	E	0.0897	0.0896	**0.0718**
	F	0.1928	**0.1480**	0.1499
ethanol	E	**0.0550**	0.0565	0.0561
	F	0.0852	0.0594	**0.0466**
malonal.	E	0.0995	0.0802	**0.0780**
	F	0.2137	0.1090	**0.0916**
napthal.	E	0.1283	0.1181	**0.1146**
	F	0.1645	**0.0455**	0.0585
salicyl.	E	0.1480	**0.1338**	0.1456
	F	0.2758	0.1101	**0.0980**
toluene	E	0.0982	**0.0958**	0.1157
	F	0.0949	0.0549	**0.0402**
uracil	E	0.1220	0.1105	**0.1086**
	F	0.2612	0.0889	**0.0831**

8. Conclusions and Limitations

In this work we have thoroughly studied the ability of GNNs to learn the geometry of a graph solely from its distance matrix. We expand on the families of counterexamples that MPNNs are unable to distinguish from their distance matrices by constructing families of symmetric and diverse geometric graphs. To better leverage the geometric structure information contained in real-world graphs, we proposed k-DisGNNs with provably higher expressive power than MPNNs for learning from distance graphs. We demonstrated the ability of k-DisGNNs to learn high-order geometric features, and unified two classical models, DimeNet and GemNet, into our framework. In experiments, k-DisGNNs outperformed previous state-of-the-art models on a wide range of targets of the MD17 dataset. Our work reveals the potential of using expressive GNN models, which were originally designed for graph structure learning, for the geometric deep learning tasks, and opens up new opportunities for this domain.

Limitations. Although k-DisGNNs have theoretical assurance for learning high-order geometry, they are difficult to train and scale for $k \geq 3$, just like the case for unweighted graphs (Morris et al., 2019). As a result, 3-E-DisGNN demonstrates slightly weaker performance compared to 2-F-DisGNN, and can hardly be applied to QM9 due to the long training time. This raises the concern on the exponentially increased cost for learning higher-order geometric features with our framework. Nevertheless, we expect that the most important geometric features can be covered by a relatively small k. Recent works on simplifying and accelerating k-WL (Morris et al., 2022; Zhao et al., 2022) when k is large may also be applied, which is left for future work.

References

Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

Azizian, W. and Lelarge, M. Expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*, 2020.

Babai, L. and Kucera, L. Canonical labelling of graphs in linear average time. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 39–46. IEEE, 1979.

Bartók, A. P., Kondor, R., and Csányi, G. On representing chemical environments. *Physical Review B*, 87(18): 184115, 2013.

Batatia, I., Kovacs, D. P., Simm, G., Ortner, C., and Csányi, G. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35: 11423–11436, 2022.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):1–11, 2022.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Cai, J.-Y., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., and Müller, K.-R. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.

Christensen, A. S. and Von Lilienfeld, O. A. On the role of gradients for machine learning of molecular energies and forces. *Machine Learning: Science and Technology*, 1(4): 045018, 2020.

Christensen, A. S., Bratholm, L. A., Faber, F. A., and Anatole von Lilienfeld, O. Fchl revisited: Faster and more accurate quantum machine learning. *The Journal of chemical physics*, 152(4):044107, 2020.

Drautz, R. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1): 014104, 2019.

Dym, N. and Maron, H. On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*, 2020.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

Garg, V., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020.

Gasteiger, J., Becker, F., and Günnemann, S. Gemnet: Universal directional graph neural networks for molecules. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 6790–6802. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/35cf8659cfcb13224cbd47863a34fc58-Paper.pdf.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020.

Han, J., Rong, Y., Xu, T., and Huang, W. Geometrically equivariant graph neural networks: A survey. 2022.

Huang, Y., Peng, X., Ma, J., and Zhang, M. Boosting the cycle counting power of graph neural networks with i^2-gnns. 2023.

Joshi, C. K., Bodnar, C., Mathis, S. V., Cohen, T., and Liò, P. On the expressive power of geometric graph neural networks. *arXiv preprint arXiv:2301.09308*, 2023.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klicpera, J., Giri, S., Margraf, J. T., and Günnemann, S. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020a.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020b.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Morris, C., Rattan, G., and Mutzel, P. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33:21824–21840, 2020.

Morris, C., Rattan, G., Kiefer, S., and Ravanbakhsh, S. Speqnets: Sparsity-aware permutation-equivariant graph networks. *arXiv preprint arXiv:2203.13913*, 2022.

Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C. J., Kornbluth, M., and Kozinsky, B. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, 2023.

Pozdnyakov, S. N. and Ceriotti, M. Incompleteness of graph convolutional neural networks for points clouds in three dimensions. *arXiv preprint arXiv:2201.07136*, 2022.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Schmitz, G., Godtliebsen, I. H., and Christiansen, O. Machine learning for potential energy surfaces: An extensive database and assessment of methods. *The Journal of chemical physics*, 150(24):244113, 2019.

Schütt, K., Unke, O., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pp. 9377–9388. PMLR, 2021.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Shapeev, A. V. Moment tensor potentials: A class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016.

Thölke, P. and De Fabritiis, G. Equivariant transformers for neural network based molecular potentials. In *International Conference on Learning Representations*, 2021.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

Wang, X. and Zhang, M. Graph neural network with local frame for molecular potential energy surface. *arXiv preprint arXiv:2208.00716*, 2022.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Zhang, M. and Li, P. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34: 15734–15747, 2021.

Zhang, Y., Xia, J., and Jiang, B. Physically motivated recursively embedded atom neural networks: incorporating local completeness and nonlocality. *Physical Review Letters*, 127(15):156002, 2021.

Zhao, L., Härtel, L., Shah, N., and Akoglu, L. A practical, progressively-expressive gnn. *arXiv preprint arXiv:2210.09521*, 2022.

A. Counterexamples for Vanilla DisGNNs

In this section, we will give some counterexamples, i.e., a pair of complete distance graphs that are not isomorphic but cannot be distinguished by vanilla DisGNNs. We will organize the section as follows: First, we will give some isolated counterexamples that can be directly verified; Then, we will give some counterexamples that can hold the property (i.e., to be counterexamples) after some simple continuous transformation; Finally, we will give a family of counterexamples that can be obtained by the combination of some basic units. Since the latter two cases cannot be directly verified, we will also give detailed proof.

In order to have a clear cognition about the component of each complete distance graphs, we say there are M *kinds* of nodes if there are M different labels after infinite iterations of vanilla DisGNNs. Note that in this section, all the grey nodes and the "edges" are just for **visualization purpose**. Only colored nodes belong to the distance graph.

Note that we provide **verification programs** in our code for all the counterexamples. The first type of counterexamples (Appendix A.1) is limited in quantity and can be directly verified. The remaining two types (Appendix A.2 and Appendix A.3) are families of counterexamples and have an infinite number of cases. Our code can verify them to some extent by randomly selecting some parameters. Theoretical proofs for these two types are also provided in Appendix A.2 and Appendix A.3, respectively.

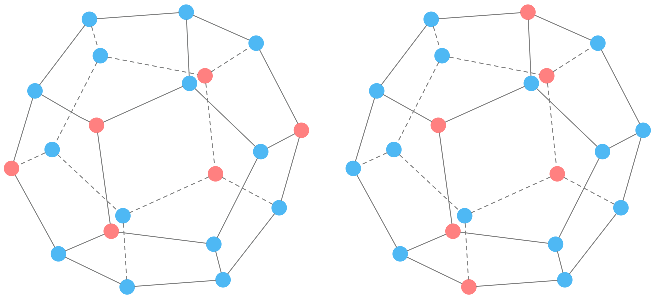
A.1. Counterexamples That Can Be Directly Verified

Since the distance graph is a kind of complete graph and contains lots of geometric constraints, if we want to get a pair of non-isomorphic graphs that cannot be distinguished by DisGNNs, they must both exhibit great *symmetry* and have just *minor difference*. Inspired by this, we can try to sample nodes from regular polyhedrons, which themselves exhibit high symmetry.

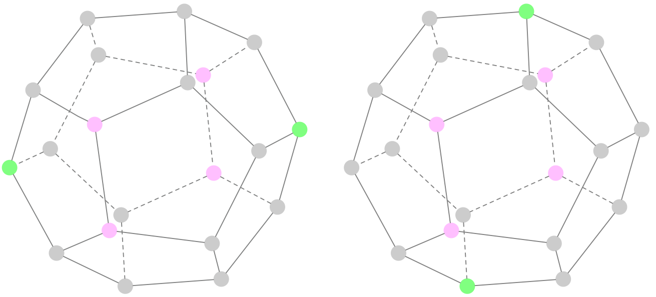
The first pair of complete distance graphs is sampled from regular icosahedrons, which is referred to the main body of our paper, see Figure 1. Since all the nodes from both graphs have the same unordered distance list, there is only **1** kind of nodes for both graphs. We note that it is the only counterexample we can sample from just regular icosahedron.

The following pairs of distance graphs are all sampled from regular dodecahedrons. We will distinguish them by the graph size.

6-size and 14-size counterexamples.



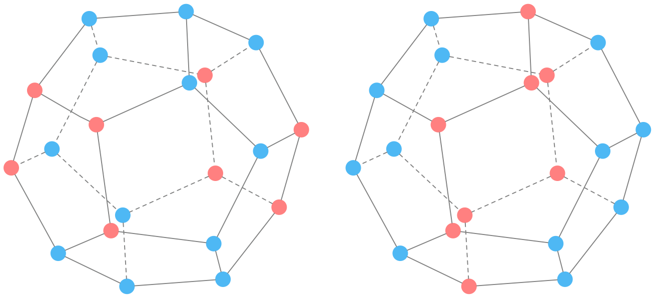
A pair of 6-size distance graphs (red nodes) and a pair of 14-size distance graphs (blue nodes) that are counterexamples can be seen in the figure above. They are actually complementary in terms of forming all vertices of a regular dodecahedron.



For the 6-size counterexample, the stable label distribution is shown in the picture above, and the nodes are colored differently to represent different labels. It can be directly verified that there are **2** kinds of nodes. Note that the two graphs are non-isomorphic: The isosceles triangle of the two distance graphs formed by one green point and two pink nodes have the same leg length but different base lengths.

For the 14-size counterexample, we give the conclusion that there are **4** kinds of nodes and the number of nodes of each kind is 2, 4, 4, 4 respectively. Since the counterexamples here and in the following are complex and not so intuitionistic, we do not give a corresponding figure to illustrate and recommend readers to verify it with our programs.

8-size and 12-size counterexamples.

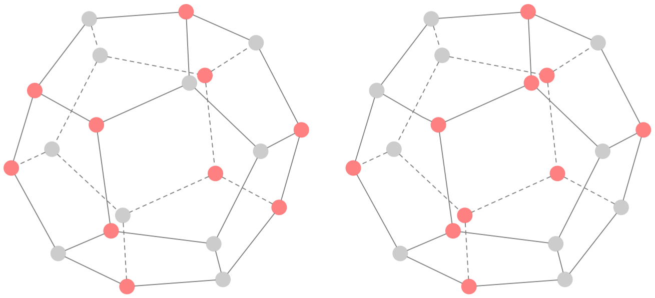


A pair of 8-size distance graphs (red nodes) and a pair of 12-size distance graphs (blue nodes) that are counterexamples can be seen in the figure above.

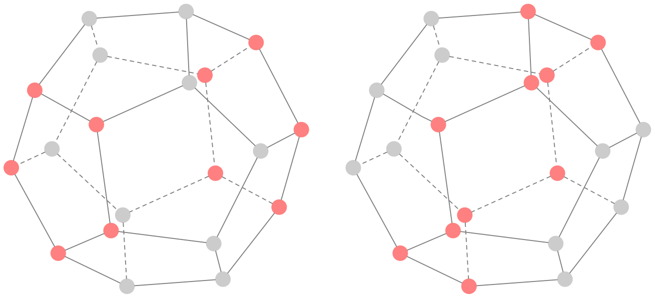
For the 8-size counterexample, note that it is actually obtained by carefully inserting two nodes into the 6-size counterexample mentioned just before. We note that there are **2** kinds of nodes in this counterexample, and the numbers of nodes of each kind are 4, 4 respectively.

For the 12-size counterexample, it is actually obtained by carefully deleting two nodes from the 14-size counterexample (corresponds to the way to obtain the 8-size counterexample) mentioned just before. There are **4** kinds of nodes in this counterexample, and the numbers of nodes of each kind are 4, 4, 2, 2 respectively.

Two pairs of 10-size counterexamples.



The above figure shows the first pair of 10-size counterexamples. There are **3** kinds of nodes and the numbers of nodes of each kind are 4, 4, 2 respectively.



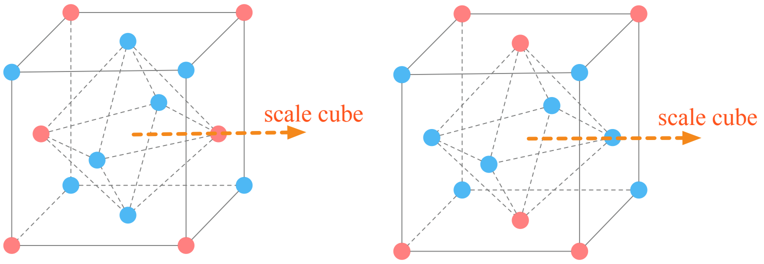
The above figure shows the second pair of 10-size counterexamples. There is actually only **1** kind of nodes, all the nodes of both graphs are isomorphic (i.e., have the same unordered distance list). It is interesting because there are two regular pentagons in the left graph and a ring in the right graph, which means that the two graphs are non-isomorphic, and it is quite similar to the case shown in Figure 1, where there are two equilateral triangles in the left graph and also a ring in the right graph.

In the end, we mention that the above counterexamples are probably all the counterexamples one can sample from a single regular polyhedron.

A.2. Counterexamples That Hold After Some Continuous Transformation

Inspired by the intuition mentioned in Appendix A.1, we can also sample nodes from the combination of regular polyhedrons, which also have great symmetry but with more nodes.

Cube + regular octahedron.



The first counterexample is the combination of a cube and a regular octahedron. We combine them in the following way: Let the center of the two polyhedrons coincide, and make the vertex of the regular octahedron, the center of the cube's face and the center of the polyhedrons co-linear, see the figure above. We don't limit the **relative size** of the two polyhedrons: one can scale the cube to any size as long as the constraints are not broken. In this way, a *small* family of counterexamples is given (we say small here because the dimension of the transformation space is small).

Proof of the red case. Let the side length of the cube be $2a$, the diagonal of the regular octahedron be $2b$ and the initial label of all the nodes be l_0. Now, let us simulate the vanilla DisGNNs on the two graphs.

At the first iteration, there are actually two kinds of unordered distance lists for both of the graphs: For the vertexes on the regular octahedron, the list is $\left\{\left((2b, l_0), 1\right), \left((\sqrt{3a^2+b^2+2ab}, l_0), 2\right), \left((\sqrt{3a^2+b^2-2ab}, l_0), 2\right)\right\}$. For the vertexes on the cube, the list is $\left\{\left((2a, l_0), 1\right), \left((2\sqrt{2}a, l_0), 1\right), \left((2\sqrt{3}a, l_0), 1\right), \left((\sqrt{3a^2+b^2+2ab}, l_0), 1\right), \left((\sqrt{3a^2+b^2-2ab}, l_0), 1\right)\right\}$. Thus they'll be labeled differently at this step, let the labels be l_1 and l_2 respectively.

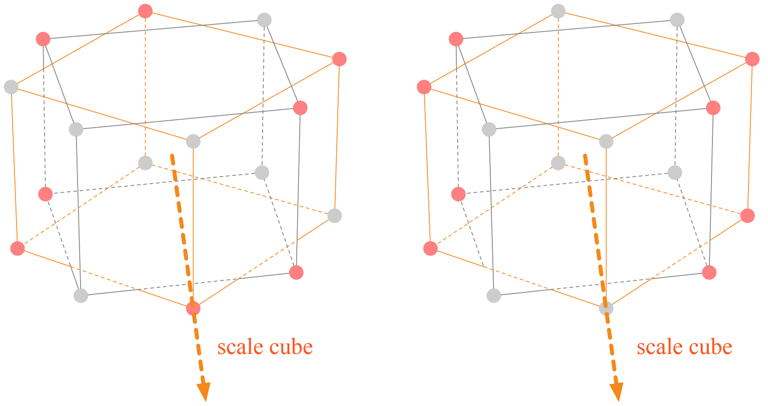
At the second iteration, the lists of all the vertexes on the regular octahedron from both graphs are still the same, i.e. $\left\{\left((2b, l_1), 1\right), \left((\sqrt{3a^2+b^2+2ab}, l_2), 2\right), \left((\sqrt{3a^2+b^2-2ab}, l_2), 2\right)\right\}$, as well as the lists of all the vertexes on the cube from both graphs, i.e.

$$\Big\{\big((2a,l_2),1\big),\big((2\sqrt{2}a,l_2),1\big),\big((2\sqrt{3}a,l_2),1\big),\big((\sqrt{3a^2+b^2+2ab},l_1),1\big),\big((\sqrt{3a^2+b^2-2ab},l_1),1\big)\Big\}.$$

Since vanilla DisGNNs cannot subdivide the vertexes at the second step, they can still not at the following steps. So it cannot distinguish the two graphs. But they are non-isomorphic: on the left graph, the nodes on the regular octahedron can only form isosceles triangles with the nodes on the face diagonal of the cube. And on the right graph, they can only form isosceles triangles with nodes on the cube's nodes that are on the same side. We note that the counterexample in Figure 1 is not a special case of this, because on the left graph of this case all the nodes are on the same surface, but none of graphs in Figure 1 have all nodes on the same surface.

Proof of the blue case. The proof of the blue case is quite similar to that of the red case, thus we omit the proof here. Note that the labels will stabilize after the first iteration. At the end of 1-WL-E, there are **2** kinds of nodes, each kind has 4 nodes. And the two graphs are non-isomorphic because the plane formed by four nodes on the regular octahedron is perpendicular to the plane formed by four nodes on the cube in the left graph, and is not in the right graph.

2 cubes.



The second is the combination of 2 cubes. We combine them in the following way: Let the center of the two cubes coincide as well as the axis formed by the centers of two opposite faces. Also, we don't limit the **relative size** of the two cubes: one can scale the cube to any size as long as the constraints are not broken. The proof of this case is quite similar to that in *Cube + regular octahedron*, and we omit the proof here too. Note that the labels will stabilize after the first iteration. If the sizes of the two cubes are the same, then at the end of 1-WL-E, there is only **1** kind of node. If not, there are **2** kinds of nodes, each kind has 4 nodes.

A.3. Families of Counterexamples

In this section, we will prove that any one of the counterexamples given in Appendix A.1 can be augmented to a family. In order to do this, we need to introduce some notations first.

Notation. Let (G^L_{ori}, G^R_{ori}) be any pair of graphs that forms a counterexample given in Appendix A.1. Notice that for each graph, since the nodes are sampled from a regular polyhedron, they are on the same **spherical surface**. Let the radius of the sphere is r, then we can use $(G^L_{ori,r}, G^R_{ori,r})$ to uniquely denote the two graphs. Since the nodes of the two graphs are sampled from the same regular polyhedron (denoted by $G_{all,r}$), there must be nodes that aren't sampled on the same polyhedron, which we call as complementary nodes. We denote the distance graphs formed by these nodes $(G^L_{com,r}, G^R_{com,r})$. Note that the node set of G_{all} can be split into the two node sets by definition, i.e. $V_{all} = V_{com} \cup V_{ori}$. Note that we use the subscript ***ori***, ***com***, ***all*** to represent the original graph, complementary graph and all-node graph respectively. Given two graphs G_{Q_1,r_1} and G_{Q_2,r_2}, $Q_i \in \{ori, com, all\}, i \in [2]$, we say that we put them to **the same center and direction** if we make sure that the spherical centers of their nodes' circumscribed sphere coincide and one can get G_{Q_2,r_2} by just scaling all the nodes of G_{Q_2,r_1} along spherical center at the same time. We use $\mathcal{L}$ to represent a **label distribution** for some graph G, i.e. $\mathcal{L} = \big\{(v, l_v) \mid v \in V\big\}$, and call it a **stable state** if 1-WL-E cannot further refine the labels of the graph G with initial label $\mathcal{L}$. We say that $\mathcal{L}$ is the **final state** of G if it is the label distribution obtained by performing 1-WL-E on G with no initial labels. Note that final state is a special stable state. We denote the final state of G_{ori} and G_{com} by $\mathcal{L}_{ori}$ and $\mathcal{L}_{com}$ respectively.

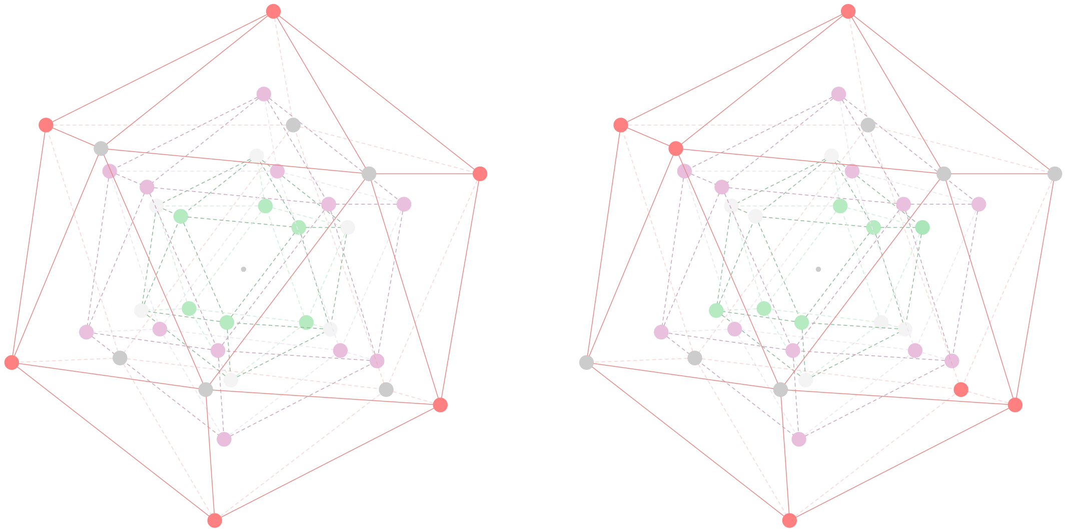


Figure 4. An example of the augmentation. In this example, (G^L_{ori}, G^R_{ori}) are from Figure 1, and $\mathcal{QR}_3 = \big\{(ori, r_r), (all, r_p), (com, r_g)\big\}$ (with r, p, g representing red, purple and green respectively). Nodes with different colors indicate that they are from different $G^L_{Q,r}$ generated. The node sets of $(G^L, G^R) = \mathrm{AUG}_{\mathcal{QR}_3}\big((G^L_{ori}, G^R_{ori})\big)$ are all the colored nodes from the left graph and the right graph respectively.

Now, let us introduce an augmentation method. Again, let (G^L_{ori}, G^R_{ori}) be any pair of graphs that forms a counterexample given in Appendix A.1. For a given set $\mathcal{QR}_k = \big\{(Q_1, r_1), (Q_2, r_2), ..., (Q_k, r_k)\big\}$ where $\forall i, j \in [k], r_i > 0, Q_i \in \{ori, com, all\}$ and if $i \neq j$ then $r_i \neq r_j$, we get the pair of graphs $(G^L, G^R) = \mathrm{AUG}_{\mathcal{QR}_k}\big((G^L_{ori}, G^R_{ori})\big)$ by the following steps (take G^L for example):

1. Generate k left graphs $G^L_{Q_1,r_1}, G^L_{Q_2,r_2}, ..., G^L_{Q_k,r_k}$ according to $\mathcal{QR}_k$.
2. Put the k graphs together to make sure they have the same center and direction.
3. Combine all the nodes of $G^L_{Q_i,r_i}, i \in [k]$ to be the node set of G^L, and generate the complete distance graph G^L according to the node set.

We give an example of the augmentation in Figure 4. Now we give our theorem.

Theorem A.1. *Let (G^L_{ori}, G^R_{ori}) be any pair of graphs that forms a counterexample given in Appendix A.1. For $\forall k \in \mathbb{Z}^+$ and arbitrary elements in $\mathcal{QR}_k = \big\{(Q_1, r_1), (Q_2, r_2), ..., (Q_k, r_k)\big\}$ where $Q_i \in \{ori, com, all\}$ and $r_i \neq r_j$ when $i \neq j$, if $\exists i \in [k], Q_i \neq all$, then $(G^L, G^R) = \mathrm{AUG}_{\mathcal{QR}_k}\big((G^L_{ori}, G^R_{ori})\big)$ is a counterexample.*

Before we prove the theorem, we need to introduce some lemmas first.

Lemma A.2. *If 1-WL-E cannot distinguish (G^L, G^R) with initial label $(\mathcal{L}^L, \mathcal{L}^R)$, then 1-WL-E cannot distinguish (G^L, G^R) without initial label (identical labels).*

Proof of Lemma A.2. We use l^t_u to represent the label of u after t iterations of WL without initial labels, and use $l^{\mathcal{L},t}_u$ to represent the label of u after t iterations of WL with initial label $\mathcal{L}$. We use l^∞_u and $l^{\mathcal{L},\infty}_u$ to represent the corresponding stable label.

In order to prove this lemma, we first prove a conclusion: for arbitrary graphs $G = (V, E)$ and arbitrary initial label $\mathcal{L}$, let u, v be two arbitrary nodes in V, then we have $\forall t \in \mathbb{Z}^+$, $l^t_u \neq l^t_v \rightarrow l^{\mathcal{L},t}_u \neq l^{\mathcal{L},t}_v$. We'll prove the conclusion by induction.

- $t = 1$ holds. If $\big(l^0_u, \{\!\{(d_{us}, l^0_s) \mid s \in N(u)\}\!\}\big) \neq \big(l^0_v, \{\!\{(d_{vs}, l^0_s) \mid s \in N(v)\}\!\}\big)$, then we have $\{\!\{d_{us} \mid s \in N(u)\}\!\} \neq \{\!\{d_{vs} \mid s \in N(v)\}\!\}$, because $l^0_s, s \in V$ are all identical. Then it is obvious that $\big(l^{\mathcal{L},0}_u, \{\!\{(d_{us}, l^{\mathcal{L},0}_s) \mid s \in N(u)\}\!\}\big) \neq \big(l^{\mathcal{L},0}_v, \{\!\{(d_{vs}, l^{\mathcal{L},0}_s) \mid s \in N(v)\}\!\}\big)$.

- For $\forall k \in \mathbb{Z}^+$, $t = k$ holds $\Rightarrow t = k + 1$ holds.

Assume $t = k+1$ doesn't hold, i.e., $\exists u, v \in V$, $l_u^{(k+1)} \neq l_v^{(k+1)}$ but $l_u^{\mathcal{L},(k+1)} = l_v^{\mathcal{L},(k+1)}$. We can derive from $l_u^{\mathcal{L},(k+1)} = l_v^{\mathcal{L},(k+1)}$ that

$$\begin{aligned}
& l_u^{\mathcal{L},(k+1)} = l_v^{\mathcal{L},(k+1)} && (18)\\
\Rightarrow\ & \left(l_u^{\mathcal{L},k}, \{\!\{(d_{us}, l_s^{\mathcal{L},k}) \mid s \in N(u)\}\!\}\right) = \left(l_v^{\mathcal{L},k}, \{\!\{(d_{vs}, l_s^{\mathcal{L},k}) \mid s \in N(v)\}\!\}\right) && (19)\\
\Rightarrow\ & l_u^{\mathcal{L},k} = l_v^{\mathcal{L},k} \text{ and } \{\!\{(d_{us}, l_s^{\mathcal{L},k}) \mid s \in N(u)\}\!\} = \{\!\{(d_{vs}, l_s^{\mathcal{L},k}) \mid s \in N(v)\}\!\} && (20)
\end{aligned}$$

Since $t = k$ holds, we have

$$\begin{aligned}
& l_u^{\mathcal{L},k} = l_v^{\mathcal{L},k} && (21)\\
\Rightarrow\ & l_u^k = l_v^k. && (22)
\end{aligned}$$

Together with Equation (20), we have $l_u^k = l_v^k$ and $\{\!\{(d_{us}, l_s^k) \mid s \in N(u)\}\!\} = \{\!\{(d_{vs}, l_s^k) \mid s \in N(v)\}\!\}$. Then we have $l_u^{(k+1)} = l_v^{(k+1)}$, which is contradictory to assumptions.

This means that the conclusion holds for $\forall t \in \mathbb{Z}^+$, as well as $t = \infty$ (stable time). In other words, specifying an initial label $\mathcal{L}$ will only result in the subdivision of the stable labels.

Back to Lemma A.2, if 1-WL-E cannot distinguish (G^L, G^R) even when the stable labels are furthermore subdivided, it can still not when the stable labels are not subdivided, i.e., cannot distinguish (G^L, G^R) without initial labels. □

Lemma A.3. *Let (G^L_{ori}, G^R_{ori}) be any pair of graphs that forms a counterexample given in Appendix A.1. Then $(\mathcal{L}^L_{ori} \cup \mathcal{L}^L_{com}, \mathcal{L}^R_{ori} \cup \mathcal{L}^R_{com})$ is a pair of stable states of (G^L_{all}, G^R_{all}). We denote the labels by $\mathcal{L}^L_{all}$ and $\mathcal{L}^R_{all}$ respectively. What's more, 1-WL-E cannot distinguish (G^L_{all}, G^R_{all}) with initial labels $\mathcal{L}^L_{all}$ and $\mathcal{L}^R_{all}$.*

Proof of Lemma A.3. Since the number of situations are finite, this lemma is directly verified by a computer program. It is worth noting that we have also included the **verification program** in our code. □

This lemma allows us to draw several useful conclusions. In the graph G_{all} (same for both graphs so superscripts are omitted), the node set V_{all} can be split into two subsets by definition, namely V_{ori} and V_{com}. Now for arbitrary node u from G^L_{all} with initial label $\mathcal{L}^L_{all}$ and v from G^R_{all} with initial label $\mathcal{L}^R_{all}$, if the initial labels of node u and node v are the same, they must be in the same subset (either V_{ori} or V_{com}) of both G^L_{all} and G^R_{all}. Without loss of generality, let us assume that both u and v are in V_{ori}. Since $(\mathcal{L}^L_{all}, \mathcal{L}^R_{all})$ is a stable state of (G^L_{all}, G^R_{all}), we can obtain the following equation:

$$\left(l_u, \{\!\{(d_{us}, l_s) \mid s \in V^L_{all}\}\!\}\right) = \left(l_v, \{\!\{(d_{vs}, l_s) \mid s \in V^R_{all}\}\!\}\right). \tag{23}$$

Since $(\mathcal{L}^L_{ori}, \mathcal{L}^R_{ori})$ is a stable state of the induced subgraphs (G^L_{ori}, G^R_{ori}) (where the node sets are (V^L_{ori}, V^R_{ori})), we can obtain the following equation:

$$\left(l_u, \{\!\{(d_{us}, l_s) \mid s \in V^L_{ori}\}\!\}\right) = \left(l_v, \{\!\{(d_{vs}, l_s) \mid s \in V^R_{ori}\}\!\}\right). \tag{24}$$

Notice that in Equation (23, 24), we can obtain a new equation by replacing V^L_{ori} or V^L_{all} with $(V^L_{ori} - u)$ or $(V^L_{all} - u)$ and doing the same for the v side, while keeping the equation valid, since $l_u = l_v$. By subtracting Equation (24) from Equation (23), we obtain the following equation:

$$\left(l_u, \{\!\{(d_{us}, l_s) \mid s \in V^L_{com}\}\!\}\right) = \left(l_v, \{\!\{(d_{vs}, l_s) \mid s \in V^R_{com}\}\!\}\right). \tag{25}$$

It is important to note that the same conclusion holds if both u and v are from type *com*. These equations provide valuable prior knowledge that we can use to prove Theorem A.1.

Now, let us prove Theorem A.1. Our proof is divided into four steps:

1. Construct an initial state $(\mathcal{L}^L, \mathcal{L}^R)$ for the augmented graphs (G^L, G^R).
2. Prove that $(\mathcal{L}^L, \mathcal{L}^R)$ is a pair of stable states for (G^L, G^R).
3. Explain that 1-WL-E cannot distinguish (G^L, G^R) with initial labels $(\mathcal{L}^L, \mathcal{L}^R)$.
4. Explain that (G^L, G^R) are non-isomorphic.

By applying Lemma A.2, we can get the conclusion from step 2 and 3 that 1-WL-E cannot distinguish (G^L, G^R). Since (G^L, G^R) are non-isomorphic (as established in step 4), it follows that (G^L, G^R) is a counterexample.

Proof of Theorem A.1.

Step 1. We first construct an initial state for the augmented graphs (G^L, G^R).

For simplicity, we omit the superscripts, as the rules are the same for both graphs. For each layer (i.e., the sphere of some radius) of the graph G, we label the nodes as follows: if the layer is of type *ori*, we label the nodes with labels from the set $\mathcal{L}_{ori}$; if it is of type *com*, we label the nodes with labels from the set $\mathcal{L}_{com}$; if it is of type *all*, we label the nodes with labels from the set $\mathcal{L}_{all}$. Notice that we always ensure that the labels of nodes in different layers are distinct.

Step 2. Now we prove that $(\mathcal{L}^L, \mathcal{L}^R)$ is a stable state for (G^L, G^R). This is equivalent to proving that for arbitrary node u in $(G^L, \mathcal{L}^L)$ and node v in $(G^R, \mathcal{L}^R)$, if $l_u^0 = l_v^0$, then $l_u^1 = l_v^1$.

Since G^L is a complete distance graph, the neighbors of node u are all the nodes in G^L except itself. We denote the neighbor nodes from the layer with radius r_i by $N_{r_i}(u)$. This is similar for node v. Now we need to prove that $\big(l_u^0, \{\!\!\{(d_{us}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(u)\}\!\!\}\big) = \big(l_v^0, \{\!\!\{(d_{vs}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(v)\}\!\!\}\big)$. Since $l_u^0 = l_v^0$, we only need to prove that $\{\!\!\{(d_{us}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(u)\}\!\!\} = \{\!\!\{(d_{vs}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(v)\}\!\!\}$.

We split the multiset $\{\!\!\{(d_{us}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(u)\}\!\!\}$ into k multisets, namely $\{\!\!\{(d_{us}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\}, i \in [k]$, and we do the same for the multiset of node v. Our goal is to prove that $\{\!\!\{(d_{us}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\} = \{\!\!\{(d_{vs}, l_s^0) \mid s \in N_{r_i}(v)\}\!\!\}$ for each $i \in [k]$.

Let the coordinates of node s in a spherical coordinate system be (r_s, θ_s, ϕ_s). Since nodes u and v have the same initial label, they must be from the same layer, meaning that $r_u = r_v$. Additionally, we always realign the coordinate systems of G^L and G^R to ensure that the direction of the polyhedra is the same. The distance between node u and node s in a spherical coordinate system is given by $\sqrt{r_u^2 + r_s^2 - 2r_u r_s\big(\sin\theta_u \sin\theta_s \cos(\phi_u - \phi_s) + \cos\theta_u \cos\theta_s\big)}$. We denote the angle term $\big(\sin\theta_u \sin\theta_s \cos(\phi_u - \phi_s) + \cos\theta_u \cos\theta_s\big)$ by Θ_{us} for simplicty.

For each value of r_i, it can be one of three types: *ori*, *com*, or *all*. Similarly, nodes u and v can belong to one of three categories: *ori*, *com*, or the same node subset of *all*. Regardless of the combination, these situations can always be found in the Equations (23, 24, 25) derived from Lemma A.3 and can be unified as follows:

$$\big(l_u^0, \{\!\!\{(\sqrt{r_u^2 + r_u^2 - 2r_u^2\Theta_{us}}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\}\big) = \big(l_v^0, \{\!\!\{(\sqrt{r_v^2 + r_v^2 - 2r_v^2\Theta_{vs}}, l_s^0) \mid s \in N_{r_i}(v)\}\!\!\}\big), \tag{26}$$

i.e. we pull the nodes from layer r_i to the same layer r_u and r_v but keep the directions Θ_{us} and Θ_{vs}. Since $r_u = r_v$, Equation (26) can be simplified as

$$\big(l_u^0, \{\!\!\{(\Theta_{us}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\}\big) = \big(l_v^0, \{\!\!\{(\Theta_{vs}, l_s^0) \mid s \in N_{r_i}(v)\}\!\!\}\big). \tag{27}$$

Therefore, we can transform the corresponding element in both multisets of u and v in Equation (26) in the same way, leading to the following equation:

$$\big(l_u^0, \{\!\!\{(\sqrt{r_u^2 + r_i^2 - 2r_u r_i\Theta_{us}}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\}\big) = \big(l_v^0, \{\!\!\{(\sqrt{r_v^2 + r_i^2 - 2r_v r_i\Theta_{vs}}, l_s^0) \mid s \in N_{r_i}(v)\}\!\!\}\big).$$

This means that for all values of r_i, $\{\!\!\{(d_{us}, l_s^0) \mid s \in N_{r_i}(u)\}\!\!\} = \{\!\!\{(d_{vs}, l_s^0) \mid s \in N_{r_i}(v)\}\!\!\}$. By merging all the multisets of radius r_i, we can get $\{\!\!\{(d_{us}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(u)\}\!\!\} = \{\!\!\{(d_{vs}, l_s^0) \mid s \in \bigcup_{i\in[k]} N_{r_i}(v)\}\!\!\}$, which concludes the proof.

Step 3. Since the stable states $(\mathcal{L}^L, \mathcal{L}^R)$ are obtained by assigning the stable states of each layer, i.e. $\mathcal{L}_{com}$, $\mathcal{L}_{ori}$ or $\mathcal{L}_{all}$, that cannot be distinguished by 1-WL-E (just because $\mathcal{L}_{com}$ and $\mathcal{L}_{ori}$ are final states of G_{com} and G_{ori}, and $\mathcal{L}_{all}$ is composed of the two), the histogram of both graphs are exactly the same, which means that 1-WL-E cannot distinguish the two graphs.

Step 4. Notice that the construction of an isomorphic mapping requires that the radius of the layer of node u in G^L and node v in G^R must be the same. This is just because the radius represents the distance between the node and the center of the graph, and if they are not equal, the two nodes are not isomorphic. So if there exists a pair of layers in the two graphs that are non-isomorphic, then the two graphs are non-isomorphic. According to the definition of (G^L, G^R), such layers always exist, therefore the two graphs are non-isomorphic. □

B. Proofs

B.1. Proof of Proposition 6.3

The main proof of our proposition is to construct the key procedures in GemNet (Gasteiger et al., 2021) using basic message passing layers of 3-E-DisGNN, and the whole model can be constructed by stacking these message passing layers up.

Basic methods. Assume that we want to learn $\mathcal{O}_{\text{tar}} = f_{\text{tar}}(\mathcal{I}_{\text{tar}})$ with our function $\mathcal{O}_{\text{fit}} = f_{\text{fit}}(\mathcal{I}_{\text{fit}})$. In our proof, the form of $\mathcal{O}_{\text{tar}}$ and $\mathcal{O}_{\text{fit}}$, as well as the form of $\mathcal{I}_{\text{tar}}$ and $\mathcal{I}_{\text{fit}}$, are quite different. For example, consider the case where $\mathcal{O}_{\text{fit}}$ is an embedding h_{abc} for a 3-tuple abc and $\mathcal{I}_{\text{fit}} = \mathcal{I}_{\text{fit}}^{(abc)}$ contains the information of all the neighbors of abc, while $\mathcal{O}_{\text{tar}}$ is an embedding m_{ab} for a 2-tuple ab and $\mathcal{I}_{\text{tar}} = \mathcal{I}_{\text{tar}}^{(ab)}$ contains the information of all the neighbors of ab. Therefore, directly learning functions by f_{fit} that produces exactly the same output as f_{tar} is inappropriate. Instead, we will learn functions that can calculate several different outputs in a way f_{tar} does and appropriately embeds them into the output of f_{fit}. For example, we still consider the case mentioned before, and we want to learn a function f_{fit} that can extract $\mathcal{I}_{\text{tar}}^{(ab)}, \mathcal{I}_{\text{tar}}^{(ac)}, \mathcal{I}_{\text{tar}}^{(bc)}, \mathcal{I}_{\text{tar}}^{(ba)}, \mathcal{I}_{\text{tar}}^{(ca)}, \mathcal{I}_{\text{tar}}^{(cb)}$ from $\mathcal{I}_{\text{fit}}^{(abc)}$ respectively, and calculates $m_{ab}, m_{ac}, m_{bc}, m_{ba}, m_{ca}, m_{cb}$ with these information like f_{tar}, and embed them into the output h_{abc} in an injective way.

Since we realize f_{fit} as a universal function approximator (such as MLPs and deep multisets) given the input and output space, f_{fit} can always learn the function we want. So the only concern is that **whether we can extract exact $\mathcal{I}_{\text{tar}}$ from $\mathcal{I}_{\text{fit}}$**, i.e., whether there exists an injective function $(\mathcal{I}_{\text{tar}}^{(1)}, \mathcal{I}_{\text{tar}}^{(2)}, ..., \mathcal{I}_{\text{tar}}^{(n)}) = f_{\text{ext}}(\mathcal{I}_{\text{fit}})$. We will mainly discuss about this in our proof.

If we can calculate and update the variables as GemNet does, then all the information needed for GemNet's output block is also preserved in our implementation until our output block, and we can calculate the output as GemNet does.

Notations. We use the superscript *G* to represent functions in GemNet and *3E* to represent functions in 3-E-DisGNN, and use $\mathcal{I}$ with the same superscript and subscript to represent the input of a function. We use the superscript $(a_1 a_2..a_k)$ to represent some input $\mathcal{I}$ if it 's for tuple $(a_1 a_2..a_k)$. If there exists an injective function $\mathcal{I}_{\text{tar}} = f_{\text{ext}}(\mathcal{I}_{\text{fit}})$, then we denote it by $\mathcal{I}_{\text{fit}} \to \mathcal{I}_{\text{tar}}$, meaning that $\mathcal{I}_{\text{tar}}$ can be derived from $\mathcal{I}_{\text{fit}}$. If some geometric information $\mathcal{I}_{\text{geo}}$ (such as distance and angles) is contained in the distance matrix of a tuple $a_1 a_2..a_k$, then we denote it by $\mathcal{I}_{\text{geo}} \in a_1 a_2..a_k$. For simplicity, We omit all the time superscript t if the context is clear.

B.1.1. Construction of embedding block

Initialization of directional embeddings. GemNet initialize all the two-tuples m (also called directional embeddings) at the embedding block by the following paradigm

$$m_{ab} = f_{\text{init}}^{\text{G}}(z_a, z_b, d_{ab}). \tag{28}$$

What 3-E-DisGNN does at the initial step is according to the following paradigm

$$h_{abc} = f_{\text{init}}^{\text{3E}}(z_a, z_b, z_c, d_{ab}, d_{ac}, d_{bc}). \tag{29}$$

Then we have

$$\mathcal{I}_{\text{init}}^{\text{3E},(abc)} \to (\mathcal{I}_{\text{init}}^{\text{G},(ab)}, \mathcal{I}_{\text{init}}^{\text{G},(ac)}, \mathcal{I}_{\text{init}}^{\text{G},(bc)}, \mathcal{I}_{\text{init}}^{\text{G},(ba)}, \mathcal{I}_{\text{init}}^{\text{G},(ca)}, \mathcal{I}_{\text{init}}^{\text{G},(cb)}),$$

meaning that we can extract all the information to calculate $m_{ab}, m_{ac}, m_{bc}, m_{ba}, m_{ca}, m_{cb}$ from the input of $f_{\text{init}}^{\text{3E}}$. And thanks to the universal property of $f_{\text{init}}^{\text{3E}}$, we can approximate a function that accurately calculate these variables and injectively embed them into h_{abc}, ensuring that no information about GemNet's variables is lost in our implementation.

Initialization of atom embeddings. GemNet initialize all the one-tuples u_i (also called atom embeddings) at the embedding block simply by passing atomic number z_i through an embedding layer.

Note that since we can learn all the m_{ij} and embed them into h_{abc} by $f_{\text{init}}^{\text{3E}}$, it is also possible to learn all the $u_i, i \in \{a, b, c\}$ and embed them into h_{abc} at the same time simply because it also has all the input needed for calculating $u_i, i \in \{a, b, c\}$ (namely $z_i, i \in \{a, b, c\}$). In this way h_{abc} can also contain all the atom embedding information we need.

Initialization of geometric information. It is an important observation that since $f_{\text{init}}^{\text{3E}}$ take all the pair-wise distance within 3-tuple abc as input, all the geometric information within the tuple can be included in h_{abc}. This makes the 3-tuple embedding h_{abc} rich in geometric information.

To remind the readers, now we prove that there exists a function $f^{\mathrm{3E}}_{\mathrm{init}}$ which can correctly calculate m_{ij} $(i \neq j, i, j \in \{a, b, c\})$, u_i $(i \in \{a, b, c\})$ and all the geometric information $\mathcal{I}_{\mathrm{geo}}$ within the triangle abc, and can injectively embed them into h_{abc}. This means that we can also learn an injective function which can restore these information, formulated as

$$h_{abc} \to \Big(\big(m_{ij} \mid i \neq j, i, j \in \{a, b, c\}\big), \big(u_i \mid (i \in \{a, b, c\})\big), \big(\mathcal{I}_{\mathrm{geo}} \mid \mathcal{I}_{\mathrm{geo}} \in abc\big)\Big).$$

B.1.2. Construction of Atom Emb Block

Atom emb block. GemNet updates the 1-tuple embeddings u by summing up all the relevant 2-tuple embeddings m in the atom emb block. The process of it can be formulated as

$$u_a = f^{\mathrm{G}}_{\mathrm{atom}}\big(\{\!\!\{(m_{ka}, e^{(ka)}_{\mathrm{RBF}}) \mid k \in [N]\}\!\!\}\big). \tag{30}$$

Since this function involves the process of pooling, it cannot be learned by our $f^{\mathrm{3E}}_{\mathrm{init}}$. However, it can be easily learned by the basic message passing layers of 3-E-DisGNN $f^{\mathrm{3E}}_{\mathrm{MP}}$, which is formulated as

$$h_{abc} = f^{\mathrm{3E}}_{\mathrm{MP}}\big(h_{abc}, \{\!\!\{(h_{kbc}, e_{ak}) \mid k \in [N]\}\!\!\}, \{\!\!\{(h_{akc}, e_{bk}) \mid k \in [N]\}\!\!\}, \{\!\!\{(h_{abk}, e_{ck}) \mid k \in [N]\}\!\!\}\big). \tag{31}$$

$$\text{where } e_{ab} = f^{\mathrm{3E}}_{\mathrm{e}}\big(d_{ab}, \{\!\!\{h_{kab} \mid k \in [N]\}\!\!\}, \{\!\!\{h_{akb} \mid k \in [N]\}\!\!\}, \{\!\!\{h_{abk} \mid k \in [N]\}\!\!\}\big), \tag{32}$$

Note that the edge representations are calculated from the tuple representations from previous time step. We now want to learn a function $f^{\mathrm{3E}}_{\mathrm{MP}}$ that updates the u_a, u_b, u_c embedded in h_{abc} like what $f^{\mathrm{G}}_{\mathrm{atom}}$ does and keep the other variables and information unchanging. Note that h_{abc} is the first input of $f^{\mathrm{3E}}_{\mathrm{MP}}$, so all the old information is maintained. As what we talked about earlier, the main focus is to check whether the information to update u is contained in $f^{\mathrm{3E}}_{\mathrm{MP}}$'s input. In fact, the following derivation holds

$$\begin{aligned}\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} &\to \{\!\!\{(h_{akc}, e_{bk}) \mid k \in [N]\}\!\!\} \to \{\!\!\{h_{akc} \mid k \in [N]\}\!\!\} \\ &\to \{\!\!\{(m_{ka}, d_{ka}) \mid k \in [N]\}\!\!\} \to \{\!\!\{(m_{ka}, e^{(ka)}_{\mathrm{RBF}}) \mid k \in [N]\}\!\!\} = \mathcal{I}^{\mathrm{G},(a)}_{\mathrm{atom}}.\end{aligned}$$

Note that $d_{ka} \in abc$ and $e^{(ka)}_{\mathrm{RBF}}$ can be calculated from d_{ka}. Similarly, we can derive that $\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} \to \mathcal{I}^{\mathrm{G},(b)}_{\mathrm{atom}}$ and $\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} \to \mathcal{I}^{\mathrm{G},(c)}_{\mathrm{atom}}$. This means we can update u in h by a basic message passing layer of 3-WL-E.

B.1.3. Construction of Interaction block

Message passing. There are two key procedures in GemNet's message passing block, namely two-hop geometric message passing (Q-MP) and one-hop geometric message passing (T-MP), which can be abstracted as follows

$$\mathrm{T-MP}: \quad m_{ab} = f^{\mathrm{G}}_{\mathrm{TMP}}\big(\{\!\!\{(m_{kb}, e^{(kb)}_{\mathrm{RBF}}, e^{(abk)}_{\mathrm{CBF}}) \mid k \in [N], k \neq a\}\!\!\}\big) \tag{33}$$

$$\mathrm{Q-MP}: \quad m_{ab} = f^{\mathrm{G}}_{\mathrm{QMP}}\big(\{\!\!\{(m_{k_2k_1}, e^{(k_2k_1)}_{\mathrm{RBF}}, e^{(bk_1k_2)}_{\mathrm{CBF}}, e^{(abk_1k_2)}_{\mathrm{SBF}}) \mid k_1, k_2 \in [N], k_1 \neq a, k_2 \neq b, a\}\!\!\}\big) \tag{34}$$

Note that what we need to construct is a function $f^{\mathrm{3E}}_{\mathrm{MP}}$ that can update the information about $m_{ij}, i, j \in \{a, b, c\}, i \neq j$ embedded in h_{abc} just like what $f^{\mathrm{G}}_{\mathrm{TMP}}$ and $f^{\mathrm{G}}_{\mathrm{QMP}}$ do, and keep the other variables and information unchanging. Since it is quite similar among different m_{ij}, we will just take the update process of m_{ab} for example.

First, T-MP. For this procedure, the following derivation holds

$$\begin{aligned}\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} &\to \big(h_{abc}, \{\!\!\{(h_{kbc}, e_{ak}) \mid k \in [N]\}\!\!\}\big) \to \{\!\!\{(h_{abc}, h_{kbc}, e_{ak}) \mid k \in [N]\}\!\!\} \\ &\to \{\!\!\{(m_{kb}, d_{kb}, d_{ab}, \phi_{abk}) \mid k \in [N]\}\!\!\} \to \{\!\!\{(m_{kb}, e^{(kb)}_{\mathrm{RBF}}, e^{(abk)}_{\mathrm{CBF}}) \mid k \in [N], k \neq a\}\!\!\} = \mathcal{I}^{\mathrm{3E},(ab)}_{\mathrm{MP}}.\end{aligned}$$

Note that in the derivation above, there is an important conclusion implicitly used: the tuple $(h_{abc}, h_{kbc}, e_{ak})$ actually contains all the geometric information in the 4-tuple $abck$, because the distance matrix of the four nodes can be obtained from it. Thus d_{kb}, d_{ab}, ϕ_{abk} can be obtained from it, and since $e^{(kb)}_{\mathrm{RBF}}$ and $e^{(abk)}_{\mathrm{CBF}}$ are just calculated from these geometric variables, the derivation holds.

And we can in principle exclude the element in the multiset where the index $k = a$, simply because these tuples have different patterns with others.

Second, Q-MP. In Q-MP, the pooling objects consists of two indices (which we call as two-order pooling), namely k_1 and k_2 in Equation (34). Another way to do this is two-step pooling, i.e. pool two times and once a index. For example we can pool index k_1 before we pool index k_2 like the following

$$m_{ab} = f^{\mathrm{G}}_{\mathrm{QMP}'}\big(\{\!\!\{\{\!\!\{(m_{k_2k_1}, e^{(k_2k_1)}_{\mathrm{RBF}}, e^{(bk_1k_2)}_{\mathrm{CBF}}, e^{(abk_1k_2)}_{\mathrm{SBF}}) \mid k_1 \in [N], k_1 \neq a\}\!\!\} \mid k_2 \in [N], k_2 \neq b, a\}\!\!\}\big). \tag{35}$$

Note that the expressiveness of two-step pooling is not less than double pooling, i.e.

$$\begin{aligned}&\{\!\!\{\{\!\!\{(m_{k_2k_1}, e^{(k_2k_1)}_{\mathrm{RBF}}, e^{(bk_1k_2)}_{\mathrm{CBF}}, e^{(abk_1k_2)}_{\mathrm{SBF}}) \mid k_1 \in [N], k_1 \neq a\}\!\!\} \mid k_2 \in [N], k_2 \neq b, a\}\!\!\}\\ \to&\{\!\!\{(m_{k_2k_1}, e^{(k_2k_1)}_{\mathrm{RBF}}, e^{(bk_1k_2)}_{\mathrm{CBF}}, e^{(abk_1k_2)}_{\mathrm{SBF}}) \mid k_1, k_2 \in [N], k_1 \neq a, k_2 \neq b, a\}\!\!\}.\end{aligned}$$

Thus if we use two-step pooling to implement Q-MP, it is not worse in terms of expressiveness. Inspired by this, in order to update m_{ab} in h_{abc} like what $f^{\mathrm{G}}_{\mathrm{QMP}}$ does, we first learn a function by $f^{\mathrm{3E}}_{\mathrm{MP}}$ that calculates an intermediate variable $w_c = f^{\mathrm{3E}}_{\mathrm{inter1}}\big(\{\!\!\{(m_{ck}, e^{(ck)}_{\mathrm{RBF}}, e^{(bkc)}_{\mathrm{CBF}}, e^{(abkc)}_{\mathrm{SBF}}) \mid k \in [N], k \neq a\}\!\!\}\big)$ by pooling all the m_{ck} at index k, which is feasible because the following derivation holds

$$\begin{aligned}\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} &\to \big(h_{abc}, \{\!\!\{(h_{abk}, e_{ck}) \mid k \in [N]\}\!\!\}\big) \to \{\!\!\{(h_{abc}, h_{abk}, e_{ck}) \mid k \in [N]\}\!\!\}\\ &\to \{\!\!\{(m_{ck}, d_{ck}, d_{bk}, \phi_{abk}, \theta_{abkc}) \mid k \in [N]\}\!\!\} \to \{\!\!\{(m_{ck}, e^{(ck)}_{\mathrm{RBF}}, e^{(bkc)}_{\mathrm{CBF}}, e^{(abkc)}_{\mathrm{SBF}}) \mid k \in [N], k \neq a\}\!\!\} = \mathcal{I}^{\mathrm{3E},(c)}_{\mathrm{inter}}.\end{aligned}$$

Note that in the derivation process above, m_{ck} is directly derived from e_{ck} because it contains the information by definition 32. Then we apply another message passing layer $f^{\mathrm{3E}}_{\mathrm{MP}}$ but this time we learn a function that just pools all the w_c in h_{abc} and finally updates m_{ab}:

$$\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} \to \{\!\!\{(h_{abk}, e_{ck}) \mid k \in [N]\}\!\!\} \to \{\!\!\{h_{abk} \mid k \in [N]\}\!\!\} \to \{\!\!\{w_k \mid k \in [N], k \neq b, a\}\!\!\}.$$

This means we can realize $f^{\mathrm{G}}_{\mathrm{QMP}'}$ by stacking two message passing layers up.

Atom self-interaction. This sub-block actually involves two procedures: First, update atom embeddings u according to the updated directional embeddings m. Second, update the directional embeddings m according to the updated atom embeddings u. The first step is actually an atom emb block, which is already realized by our $f^{\mathrm{3E}}_{\mathrm{MP}}$ in Appendix B.1.1. The second procedure can be formulated as

$$m_{ab} = f^{\mathrm{G}}_{\mathrm{self-inter}}(u_a, u_b, m_{ab}). \tag{36}$$

It is obvious that we can update m_{ab} in h_{abc} by $f^{\mathrm{3E}}_{\mathrm{MP}}$ according to this procedure because the following derivation holds

$$\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{MP}} \to h_{abc} \to (u_a, u_b, m_{ab}) = \mathcal{I}^{\mathrm{G},(ab)}_{\mathrm{self-inter}}.$$

B.1.4. Construction of Output block

In GemNet, the final output t is obtained by summing up all the sub-outputs from each interaction block. While it is possible to add additional sub-output blocks to our model, in our proof we only consider the case where the output is obtained solely from the final interaction block.

The output of GemNet is obtained by the following function

$$t = \sum_{a \in [N]} W_{\mathrm{out}}\big(f^{\mathrm{G}}_{\mathrm{atom}}(\{\!\!\{(m_{ka}, e^{(ka)}_{\mathrm{RBF}}) \mid k \in [N]\}\!\!\})\big), \tag{37}$$

where W_{out} is a learnable matrix.

And our output function is

$$t = f^{\mathrm{3E}}_{\mathrm{output}}\big(\{\!\!\{h_{abc} \mid a, b, c \in [N]\}\!\!\}\big). \tag{38}$$

We can realize Equation (37) this by stacking a message passing layer and an output block: First the message passing layer update all the atom embeddings u in h_{abc}, and then the output block extract u from h as follows, and calculates t like GemNet does.

$$\mathcal{I}^{\mathrm{3E},(abc)}_{\mathrm{output}} \to \{\!\!\{h_{aaa} \mid a \in [N]\}\!\!\} \to \{\!\!\{u_a \mid a \in [N]\}\!\!\}.$$

Note that the first → holds simply because h with different equality pattern is distinguishable.

B.2. Proof of Proposition 6.2

In this section, we will follow the basic method and notations in Appendix B.1. We will use the superscript *D* to represent functions in DimeNet and *2F* to represent functions in 2-F-DisGNN.

B.2.1. Construction of Embedding block

DimeNet initializes all the two-tuples in the embedding block just like how GemNet does, which can be formulated as

$$m_{ab} = f^{\mathrm{D}}_{\mathrm{init}}(z_a, z_b, d_{ab}). \tag{39}$$

What 2-F-DisGNN does at initialization step is

$$h_{ab} = f^{\mathrm{2F}}_{\mathrm{init}}(z_a, z_b, d_{ab}). \tag{40}$$

Now assume we want to learn such a function which can learn both m_{ab} and m_{ba}, then embed it into h_{ab}. Just like what we talked about in Appendix B.1, it is possible simply because the following derivation holds

$$\mathcal{I}^{\mathrm{2F},(ab)}_{\mathrm{init}} \to (\mathcal{I}^{\mathrm{D},(ab)}_{\mathrm{init}}, \mathcal{I}^{\mathrm{D},(ba)}_{\mathrm{init}}).$$

Note that h_{ab} also contains the geometric information, but in this case, just the distance. Different from GemNet, DimeNet doesn't "track" the 1-tuple embeddings: it doesn't initialize and update the atom embeddings in the model, and only pool the 2-tuples into 1-tuples in the output block. So there is no need to embed the embeddings of atom a and atom b into h_{ab}.

B.2.2. Construction of Interaction block

The message passing framework of DimeNet (so called directional message passing) can be formulated as

$$m_{ab} = f^{\mathrm{D}}_{\mathrm{MP}}\big(\{\!\{(m_{ka}, e^{(ab)}_{\mathrm{RBF}}, e^{(kab)}_{\mathrm{CBF}}) \mid k \in [N], k \neq b\}\!\}\big). \tag{41}$$

And the message passing framework of 2-F-DisGNN can be formulated as

$$h_{ab} = f^{\mathrm{2F}}_{\mathrm{MP}}\big(h_{ab}, \{\!\{(h_{kb}, h_{ak}) \mid k \in [N], k \neq b\}\!\}\big). \tag{42}$$

Now we want to learn a function $f^{\mathrm{2F}}_{\mathrm{MP}}$ that can updates the m_{ab} and m_{ba} embedded in h_{ab} like what $f^{\mathrm{D}}_{\mathrm{MP}}$ does. We need to check if $f^{\mathrm{2F}}_{\mathrm{MP}}$ have sufficient information to update m_{ab} and m_{ba} embedded in its output h_{ab}. In fact, the derivation holds respectively

$$\begin{aligned}\mathcal{I}^{\mathrm{2F},(ab)}_{\mathrm{MP}} &\to \{\!\{(h_{ab}, h_{kb}, h_{ak}) \mid k \in [N]\}\!\} \to \{\!\{(m_{ka}, d_{ab}, d_{ka}, \phi_{kab}) \mid k \in [N], k \neq b\}\!\} \\ &\to \{\!\{(m_{ka}, e^{(ab)}_{\mathrm{RBF}}, e^{(kab)}_{\mathrm{CBF}}) \mid k \in [N], k \neq b\}\!\} = \mathcal{I}^{\mathrm{D},(ab)}_{\mathrm{MP}}.\end{aligned}$$

Note that we again used the observation that tuple (h_{ab}, h_{kb}, h_{ak}) contains all the geometric information of 3-tuple abk. Similarly we can get $\mathcal{I}^{\mathrm{2F},(ab)}_{\mathrm{MP}} \to \mathcal{I}^{\mathrm{D},(ba)}_{\mathrm{MP}}$.

B.2.3. Construction of output block

The output block of DimeNet is quite similar to that of GemNet (Appendix B.1.4), which can be formulated as

$$u_a = f^{\mathrm{D}}_{\mathrm{atom}}\big(\{\!\{(m_{ka}, e^{(ka)}_{\mathrm{RBF}}) \mid k \in [N]\}\!\}\big), \tag{43}$$

$$t = \sum_{a \in [N]} u_a. \tag{44}$$

This can be realized by stacking a message passing layer of 2-F-DisGNN and its output block up.

The message passing layer of 2-F-DisGNN can learn u_a and u_b and embed it into the output h_{ab} because the following derivation holds

$$\begin{aligned}\mathcal{I}^{\mathrm{2F},(ab)}_{\mathrm{MP}} &\to \{\!\{(h_{kb}, h_{ak}) \mid k \in [N]\}\!\} \to \{\!\{h_{ak} \mid k \in [N]\}\!\} \\ &\to \{\!\{(m_{ka}, d_{ka}) \mid k \in [N]\}\!\} \to \{\!\{(m_{ka}, e^{(ka)}_{\mathrm{RBF}}) \mid k \in [N]\}\!\} = \mathcal{I}^{\mathrm{D},(a)}_{\mathrm{atom}}.\end{aligned}$$

Similarly we can get $\mathcal{I}_{\mathrm{MP}}^{2\mathrm{F},(ab)} \to \mathcal{I}_{\mathrm{atom}}^{\mathrm{D},(b)}$.

And the output block of 2-F-DisGNN is formulated as follows

$$t = f_{\mathrm{output}}^{2\mathrm{F}}\big(\{\!\{h_{ab} \mid a, b \in [N]\}\!\}\big). \tag{45}$$

Note that the following derivation holds

$$\mathcal{I}_{\mathrm{output}}^{2\mathrm{F},(ab)} \to \{\!\{h_{aa} \mid a \in [N]\}\!\} \to \{\!\{u_a \mid a \in [N]\}\!\}.$$

This means that the output block of 2-F-DisGNN can implement the sum operation in Equation 44.

B.3. Proof of Theorem 5.1

In the theorem, k-DisGNN $\sqsubseteq$ k-E-DisGNN is obvious. That is because the two models share the same initialization and output block. And for the message passing block, as long as the message passing function of k-E-DisGNN, $f_{\mathrm{MP}}^{\mathrm{E}}$, ignores the edge representations in $H_j^{\mathrm{E},t}$ defined in Equation (15), it degenerates into the message passing function of k-DisGNN. In the specific implementation, we use Hadamard product to represent the tuple $(h_{\boldsymbol{w}}^t, e^t_{\boldsymbol{v}\backslash\boldsymbol{w},\boldsymbol{w}\backslash\boldsymbol{v}})$ in Equation (15), and as long as the previous functions calculate an all-one vector for all the edge representations, then k-E-DisGNN degenerates into k-DisGNN. The non-trivial case in the theorem is k-E-DisGNN $\sqsubseteq$ k-F-DisGNN. To prove this, we will introduce some notations first.

Notations. Let $P(s)$ denote the set of all partitions of set s. For two partition $p, p' \in P(s)$, if p' is finer than p, then we say $p' \leq p$. If $p \leq p'$ and $p \neq p'$, $p < p'$. For two elements $i, j \in s$, we use $i \sim_p j$ to denote that i, j are in the same part of p. Let $\mathbf{1}(s)$ be a filter function, which outputs an all-one vector if statement s is true and an all-zero vector otherwise. Let function $\Phi_{j_1,j_2,...,j_s}(\boldsymbol{v}, a_1, a_2, ..., a_s)$ produce a tuple $\boldsymbol{u}$ by replacing the j_k^{th} element in $\boldsymbol{v}$ with a_k for all $k \in [s]$.

Note that we also use the notations and **basic methods** defined in Appendix B.1 for simplicity.

The proof is divided into three steps:

1. Prove k-DisGNN $\sqsubseteq$ k-F-DisGNN.
2. Introduce some basic operations that can be learned by the message passing functions of k-DisGNN and k-F-DisGNN, which are useful for proving k-E-DisGNN $\sqsubseteq$ k-F-DisGNN.
3. Prove k-E-DisGNN $\sqsubseteq$ k-F-DisGNN with the conclusions proved in privious steps.

B.3.1. k-DisGNN $\sqsubseteq$ k-F-DisGNN

k-DisGNN shares the same initialization block and output block as k-F-DisGNN, so we just need to prove that $f_{\mathrm{MP}}^{\mathrm{F},t}$ can implement f_{MP}^t. The two functions are defined as follows

$$h_{\boldsymbol{v}}^{\mathrm{F},(t+1)} = f_{\mathrm{MP}}^{\mathrm{F},t}\big(h_{\boldsymbol{v}}^{\mathrm{F},t}, \{\!\!\{(h^{\mathrm{F},t}_{\Phi_j(\boldsymbol{v},a)} \mid j \in [k]) \mid a \in V\}\!\!\}\big), \tag{46}$$

$$h_{\boldsymbol{v}}^{(t+1)} = f_{\mathrm{MP}}^{t}\big(h_{\boldsymbol{v}}^{t}, (\{\!\!\{h^{t}_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\!\} \mid j \in [k])\big), \tag{47}$$

where $h_v^{\mathrm{F},t}, h_v^t$ are k-F-DisGNN's and k-DisGNN's representations of k-tuple $\boldsymbol{v}$ at iteration t, respectively. Just as the basic method we use in Appendix B.1, as long as we can prove that the input of $f_{\mathrm{MP}}^{\mathrm{F},t}$, denoted as $\mathcal{I}_{\mathrm{MP}}^{\mathrm{F},t}$, can "produce" that of f_{MP}^t, denoted as $\mathcal{I}_{\mathrm{MP}}^t$, we can prove that $f_{\mathrm{MP}}^{\mathrm{F},t}$ can implement f_{MP}^t. We will prove $\mathcal{I}_{\mathrm{MP}}^{\mathrm{F},t} \to \mathcal{I}_{\mathrm{MP}}^t$ by induction.

- $t = 0$ holds.

$$\mathcal{I}_{\mathrm{MP}}^{\mathrm{F},0} = \big(h_{\boldsymbol{v}}^{\mathrm{F},0}, \{\!\!\{(h^{\mathrm{F},0}_{\Phi_j(\boldsymbol{v},a)} \mid j \in [k]) \mid a \in V\}\!\!\}\big) \tag{48}$$

$$\to \big(h_{\boldsymbol{v}}^{\mathrm{F},0}, (\{\!\!\{h^{\mathrm{F},0}_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\!\} \mid j \in [k])\big) \tag{49}$$

$$\to \big(h_{\boldsymbol{v}}^{0}, (\{\!\!\{h^{0}_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\!\} \mid j \in [k])\big) = \mathcal{I}_{\mathrm{MP}}^0. \tag{50}$$

Note that we used a conclusion: As the two models share the same initialization functions, we can have that for arbitrary $\boldsymbol{u} \in V^k$, $h_{\boldsymbol{u}}^{\mathrm{F},0} = h_{\boldsymbol{u}}^0$.

- $\forall T \in \mathbb{Z}^+ \cup \{0\}$, $t = T$ holds $\Rightarrow$ $t = T + 1$ holds.

$$\mathcal{I}_{\mathrm{MP}}^{\mathrm{F},T+1} = (h_{\boldsymbol{v}}^{\mathrm{F},T+1}, \{\!\!\{(h^{\mathrm{F},T+1}_{\Phi_j(\boldsymbol{v},a)} \mid j \in [k]) \mid a \in V\}\!\!\}) \tag{51}$$

$$\to (h_{\boldsymbol{v}}^{\mathrm{F},T+1}, (\{\!\!\{h^{\mathrm{F},T+1}_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\!\} \mid j \in [k])) \tag{52}$$

$$\to (h_{\boldsymbol{v}}^{T+1}, (\{\!\!\{h^{T+1}_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\!\} \mid j \in [k])) = \mathcal{I}_{\mathrm{MP}}^{T+1}. \tag{53}$$

Note that we use a conclusion: As $t = T$ holds, we have $\mathcal{I}_{\mathrm{MP}}^{\mathrm{F},T} \to \mathcal{I}_{\mathrm{MP}}^{T}$, then for arbitrary $\boldsymbol{u} \in V^k$, $h_{\boldsymbol{u}}^{\mathrm{F},T+1} \to h_{\boldsymbol{u}}^{T+1}$.

B.3.2. Basic Operations of k-DisGNN and k-F-DisGNN

All the k-tuples' embeddings form a tensor $h \in \mathbb{R}^{n^k \times d}$. In this subsection, we will state some operations on the tensor that can be expressed by the message passing function of k-DisGNN, f^t_{MP}. As the message passing function of k-DisGNN can be implemented by that of k-F-DisGNN, k-F-DisGNN can also implement these operations.

Tuple-wise transformation. Given an arbitrary tuple-wise transformation $f = f(h_{\boldsymbol{v}})$, it can be expressed by f^t_{MP} as follows

$$h^{t+1}_{\boldsymbol{v}} = f^t_{\mathrm{MP}}\Big(h^t_{\boldsymbol{v}}, \big(H^t_1(\boldsymbol{v}), H^t_2(\boldsymbol{v}), ..., H^t_k(\boldsymbol{v})\big)\Big) \to f(h^t_{\boldsymbol{v}}), \tag{54}$$

where f^t_{MP} produces $f(h^t_{\boldsymbol{v}})$ simply by ignoring $\big(H^t_1(\boldsymbol{v}), H^t_2(\boldsymbol{v}), ..., H^t_k(\boldsymbol{v})\big)$.

Isomorphism Type Filter. We define the symbol $\delta(p), p \in P([k])$ as follows

$$\delta(p) = \{(a_1, a_2, ..., a_k) \mid \forall i, j \in [k], a_i = a_j \Leftrightarrow i \sim_p j\}. \tag{55}$$

Note that each partition induces a subset of V^k, and $\{\delta(p) \mid p \in P([k])\}$ forms a partition of V^k. Given a tuple $\boldsymbol{u} \in V^k$, the part of the $\{\delta(p) \mid p \in P([k])\}$ that contains $\boldsymbol{u}$ is called $\boldsymbol{u}$'s isomorphism type. Abusing the notation, we denote $\boldsymbol{u}$'s isomorphism type by $\delta(\boldsymbol{u})$. Note that by definition, embeddings $h^0_{\boldsymbol{v}}$ with different isomorphism types $\delta(\boldsymbol{v})$ are initialized differently. Given a set of partitions $\{p_1, p_2, ..., p_s\}$, we can implement the following isomorphism type filter by f^t_{MP}

$$h^{t+1}_{\boldsymbol{v}} = f^t_{\mathrm{MP}}\Big(h^t_{\boldsymbol{v}}, \big(H^t_1(\boldsymbol{v}), H^t_2(\boldsymbol{v}), ..., H^t_k(\boldsymbol{v})\big)\Big) \to \mathbf{1}(\boldsymbol{u} \in \bigcup_{j=1}^{s} \delta(p_j)) \odot h^t_{\boldsymbol{v}}. \tag{56}$$

where f^t_{MP} produces output solely according to $h^t_{\boldsymbol{v}}$'s isomorphism type.

To sum neighbors' embeddings. f^t_{MP} can sum the representations of a tuple $\boldsymbol{v}$'s j-neighbors as follows

$$h^{t+1}_{\boldsymbol{v}} = f^t_{\mathrm{MP}}\Big(h^t_{\boldsymbol{v}}, \big(H^t_1(\boldsymbol{v}), H^t_2(\boldsymbol{v}), ..., H^t_k(\boldsymbol{v})\big)\Big) \to \sum_{h \in H^t_j(\boldsymbol{v})} h = \sum_{\boldsymbol{u} \in N_j(\boldsymbol{v})} h^t_{\boldsymbol{u}}. \tag{57}$$

To sum the representations of all the representations of $\{\Phi_{(j_1, j_2, ..., j_s)}(\boldsymbol{v}, a, b, c....) \mid a, b, c... \in V\}$ (we call as $\boldsymbol{v}$'s $(j_1, j_2, ..., j_s)$-neighbors), we can stack s message passing layers, where each layer sums only one index as follows

$$\begin{aligned} h^{t+s}_{\boldsymbol{v}} &\to \sum_{\boldsymbol{v_1} \in N_{j_1}(\boldsymbol{u})} h^{t+s-1}_{\boldsymbol{v_1}} \qquad (58) \\ &\to \sum_{\boldsymbol{v_1} \in N_{j_1}(\boldsymbol{u})} \sum_{\boldsymbol{v_2} \in N_{j_2}(\boldsymbol{v_1})} h^{t+s-2}_{\boldsymbol{v_2}} \qquad (59) \\ &\to \sum_{a,b,c... \in V} h^t_{\Phi_{(j_1, j_2, ..., j_s)}(\boldsymbol{v}, a, b, c....)}, \qquad (60) \end{aligned}$$

To pool neighbors' embeddings. By stacking a "tuple-wise transform" layer and a "sum" layer described above, k-DisGNN can implement deepset function and thus pools neighbors.

$$\begin{aligned} h^{t+s+1}_{\boldsymbol{v}} &\to \sum_{a,b,c... \in V} h^{t+1}_{\Phi_{(j_1, j_2, ..., j_s)}(\boldsymbol{u}, a, b, c....)} \qquad (61) \\ &= \sum_{a,b,c... \in V} f^t\big(h^t_{\Phi_{(j_1, j_2, ..., j_s)}(\boldsymbol{u}, a, b, c....)}\big) \qquad (62) \\ &\to \{\!\!\{ h^t_{\Phi_{(j_1, j_2, ..., j_s)}(\boldsymbol{u}, a, b, c....)} \mid a, b, c... \in V \}\!\!\}. \qquad (63) \end{aligned}$$

Arbitrary Permutation-Equivariant Linear Transformation. Each partition $p \in P([2k])$ induces two subsets of V^{2k}.

$$\gamma(p) = \{\boldsymbol{u} \in V^{2k} \mid \forall i_1, i_2 \in [2k], i_1 \sim_p i_2 \Leftrightarrow v_{i_1} = v_{i_2}\}, \tag{64}$$

$$\gamma'(p) = \{\boldsymbol{u} \in V^{2k} \mid \forall i_1, i_2 \in [2k], i_1 \sim_p i_2 \Rightarrow v_{i_1} = v_{i_2}\}, \tag{65}$$

As shown in (Maron et al., 2018), $\gamma(p) \bigcap \gamma(p') \neq \emptyset \Leftrightarrow p = p'$. Obviously, $\gamma'(p) = \bigcup_{p \leq p'} \gamma(p')$.

Maron et al. (2018) also show that, to express arbitrary permutation equivariant linear transformation, we only need to implement

$$h_{\boldsymbol{v}} = \sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma(p)} h_{\boldsymbol{u}}, \tag{66}$$

for all $p \in P([2k])$, where $\boldsymbol{vu}$ produces a $2k$-tuple $(v_1, v_2, ..., v_k, u_1, u_2, ..., u_k)$, and $\boldsymbol{u} \mid \boldsymbol{vu} \in \gamma(p)$ means summation over $\boldsymbol{u}$ satisfying $\boldsymbol{vu} \in \gamma(p)$.

Lemma B.1. *Implementing Equation (66) is equivalent to implementing the following operations.*

$$h_{\boldsymbol{v}} = \sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma'(p)} h_{\boldsymbol{u}}, \tag{67}$$

for all $p \in P([2k])$.

Proof. First, we perform a topological sort on $P([2k])$. Let $(p_1, p_2, ..., p_b)$ be the sorted tuple, where $b = |P([2k])|$ and $\forall i, j \in [b], p_i > p_j \Rightarrow i < j$.

Second, since $\gamma'(p) = \bigcup_{p \leq p'} \gamma(p')$ holds, we have

$$\sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma'(p)} h_{\boldsymbol{u}} = \sum_{p \leq p'} \sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma(p')} h_{\boldsymbol{u}} = \sum_{\boldsymbol{vu} \in \gamma(p)} h_{\boldsymbol{u}} + \sum_{p < p'} \sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma(p')} h_{\boldsymbol{u}}. \tag{68}$$

Let s_i denote the operator $\sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma(p_i)}$ and s_i' denote the operator $\sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma'(p_i)}$, then we have

$$s_i' = s_i + \sum_{j < i} \mathbf{1}(p_i < p_j) \odot s_j, \tag{69}$$

where $+$ of s results in the sum of the results produced by s.

Equation (69) can be expressed in a matrix form:

$$\begin{bmatrix} s_1' \\ s_2' \\ \vdots \\ s_b' \end{bmatrix} = A \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_b \end{bmatrix}, \tag{70}$$

where A is an lower triangular matrix, whose diagonal elements are all 1. Note that A is invertible, giving

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_b \end{bmatrix} = A^{-1} \begin{bmatrix} s_1' \\ s_2' \\ \vdots \\ s_b' \end{bmatrix}. \tag{71}$$

Therefore, the original form can be expressed by the new form. □

Then we prove: $\forall p \in P([2k])$,

$$h_{\boldsymbol{v}}^{L+t} \to \sum_{\boldsymbol{u} | \boldsymbol{vu} \in \gamma(p)} h_{\boldsymbol{u}}^t$$

can be expressed by L message passing layers of k-DisGNN, f_{MP}^t, for some $L \in \mathbb{Z}^+$.

Proof. Given $p \in P([2k]), \boldsymbol{u} \in V^k$, let $\{i_1, i_2, ..., i_s\}$ denote the position in $\boldsymbol{u}$ that is "fixed" by $\boldsymbol{v}$. Formally, $\{i_1, i_2, ..., i_s\} = \{i \mid i \in [k], \exists j \in [k] \text{ s.t. } (i+k) \sim_p j\}$. There exists a function $g : [k] \to [k]$ s.t. $\forall m \in [s], g(i_m) \sim_p (i_m + k)$. Note that there can be multiple indices $g(i_m)$ satisfying the relation, and we can choose arbitrary one, since it does not influence our results. For the $\boldsymbol{u}$ that satisfies $\boldsymbol{vu} \in \gamma(p)$, there are two cases.

- $s = k$, meaning that all elements in $\boldsymbol{u}$ are fixed by $\boldsymbol{v}$.

  We want to do the update so that $h_{\boldsymbol{v}} \to h_{\boldsymbol{u}}$ holds at all time steps, for arbitrary $\boldsymbol{u}, \boldsymbol{v} \in V^k$ where $\boldsymbol{u}$ is fixed by $\boldsymbol{v}$. We are going to prove it by induction.

  - Initialization utilizes isomorphism type and adjacency slice of $\boldsymbol{u}$, which are contained in $\boldsymbol{v}$'s isomorphism type and adjacency slice. So $h^0_{\boldsymbol{v}} \to h^0_{\boldsymbol{u}}$.
  - Assume that $h^t_{\boldsymbol{v}} \to h^t_{\boldsymbol{u}}$ holds for arbitrary $\boldsymbol{u}, \boldsymbol{v} \in V^k$ where $\boldsymbol{u}$ is fixed by $\boldsymbol{v}$, we are going to prove that $h^{t+1}_{\boldsymbol{v}} \to h^{t+1}_{\boldsymbol{u}}$ for arbitrary $\boldsymbol{u}, \boldsymbol{v} \in V^k$ where $\boldsymbol{u}$ is fixed by $\boldsymbol{v}$.

    The message passing functions of k-DisGNN at time step t for $\boldsymbol{v}$ and $\boldsymbol{u}$ are

$$h^{t+1}_{\boldsymbol{v}} = f^t_{\text{MP}}\big(h^t_{\boldsymbol{v}}, (\{\!\{h^t_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\} \mid j \in [k])\big), \tag{72}$$

$$h^{t+1}_{\boldsymbol{u}} = f^t_{\text{MP}}\big(h^t_{\boldsymbol{u}}, (\{\!\{h^t_{\Phi_j(\boldsymbol{u},a)} \mid a \in V\}\!\} \mid j \in [k])\big). \tag{73}$$

  Given $\Phi_j(\boldsymbol{u}, a)$ and $\Phi_{g(j)}(\boldsymbol{v}, a)$, note that all elements in $\Phi_j(\boldsymbol{u}, a)$ still exists in $\Phi_{g(j)}(\boldsymbol{v}, a)$. In other words, $\Phi_j(\boldsymbol{u}, a)$ is still fixed by $\Phi_{g(j)}(\boldsymbol{v}, a)$. This means that $h^t_{\Phi_{g(j)}(\boldsymbol{v},a)} \to h^t_{\Phi_j(\boldsymbol{u},a)}$ holds. Therefore,

$$\begin{aligned} \mathcal{I}^t_{\text{MP},\boldsymbol{v}} &= \big(h^t_{\boldsymbol{v}}, (\{\!\{h^t_{\Phi_j(\boldsymbol{v},a)} \mid a \in V\}\!\} \mid j \in [k])\big) && (74)\\ &\to \big(h^t_{\boldsymbol{v}}, (\{\!\{h^t_{\Phi_{g(j)}(\boldsymbol{v},a)} \mid a \in V\}\!\} \mid j \in [k])\big) && (75)\\ &\to \big(h^t_{\boldsymbol{u}}, (\{\!\{h^t_{\Phi_j(\boldsymbol{u},a)} \mid a \in V\}\!\} \mid j \in [k])\big) && (76)\\ &= \mathcal{I}^t_{\text{MP},\boldsymbol{u}}, && (77) \end{aligned}$$

  meaning that $h^{t+1}_{\boldsymbol{v}} \to h^{t+1}_{\boldsymbol{u}}$.

- If $s < k$, not all the elements in $\boldsymbol{u}$ are fixed. Let $\boldsymbol{v'} \in V^k$ denote a tuple satisfying $\forall m \in [s], \boldsymbol{v'}_{i_m} = \boldsymbol{v}_{g(i_m)}$, and other elements in $\boldsymbol{v'}$ simply equal to $\boldsymbol{v}_1$. Let p' denote a partition of $[2k]$ in the following form:

$$p' = \big\{s \mid s \in p, s \cap [k] = \emptyset\big\} \bigcup \big\{([k] - \{i_1, i_2, ..., i_s\}) \cup \{1\}\big\} \bigcup \big\{\{i_j, i_j + k\} \mid j \in [s], i_j \neq 1\big\}. \tag{78}$$

  Therefore, we have

$$\big\{\boldsymbol{u} \mid \boldsymbol{vu} \in \gamma'(p)\big\} = \big\{\boldsymbol{u} \mid \boldsymbol{v'u} \in \gamma'(p')\big\}. \tag{79}$$

  So we can first aggregate representations of tuples in $\big\{\boldsymbol{u} \mid \boldsymbol{vu} \in \gamma'(p)\big\}$ to tuple $\boldsymbol{v'}$, then pass messages from tuple $\boldsymbol{v'}$ to tuple $\boldsymbol{v}$.

  First, by summing multi-hop neighbors (sum $[k] - \{i_1, i_2, ..., i_s\}$ neighbors), we have

$$h^{t+k-s}_{\boldsymbol{v'}} \to \sum_{\boldsymbol{u} \mid \boldsymbol{v'u} \in \gamma'(p')} h^t_{\boldsymbol{u}} = \sum_{\boldsymbol{u} \mid \boldsymbol{vu} \in \gamma'(p)} h^t_{\boldsymbol{u}} \tag{80}$$

  Since $\boldsymbol{v'}$ is fixed by $\boldsymbol{v}$, from the conclusion in $s = k$ case, we can get that

$$h^{t+k-s}_{\boldsymbol{v}} \to h^{t+k-s}_{\boldsymbol{v'}} = \sum_{\boldsymbol{u} \mid \boldsymbol{vu} \in \gamma'(p)} h^t_{\boldsymbol{u}}. \tag{81}$$

Therefore, arbitrary permutation equivariant linear transformation can be implemented with a number of k-DisGNN or k-F-DisGNN layers. We use L_k to denote the maximum number of layers needed. □

B.3.3. k-E-DisGNN $\sqsubseteq$ k-F-DisGNN

We now show that k-E-DisGNN's message passing iterations can be implemented by $(L_k + 2)$ k-F-DisGNN's message passing iterations, thus demonstrating that k-E-DisGNN can be implemented by k-F-DisGNN.

Since k-E-DisGNN and k-F-DisGNN share the same initialization block, they can produce the same initial tuple representations, i.e., for arbitrary $\boldsymbol{v} \in V^k$, $h_{\boldsymbol{v}}^{\mathrm{F},0} = h_{\boldsymbol{v}}^{\mathrm{E},0}$.

At t^{th} iteration, the message passing function of k-E-DisGNN is

$$h_{\boldsymbol{v}}^{\mathrm{E},(t+1)} = f_{\mathrm{MP}}^{\mathrm{E},t}\Big(h_{\boldsymbol{v}}^{\mathrm{E},t}, \big(\{\!\!\{(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},t}, e_{\boldsymbol{v}_j a}^{t}) \mid a \in V\}\!\!\} \mid j \in [k]\big)\Big), \tag{82}$$

$$e_{ij}^{t} = f_{e}^{\mathrm{E},t}\Big(e_{ij}, \big(\{\!\!\{h_{\boldsymbol{w}}^{\mathrm{E},t} \mid \boldsymbol{w} \in V^k, \boldsymbol{w}_c = i, \boldsymbol{w}_d = j\}\!\!\} \mid c, d \in [k], c < d\big)\Big). \tag{83}$$

And that of k-F-DisGNN is

$$h_{\boldsymbol{v}}^{\mathrm{F},(t+1)} = f_{\mathrm{MP}}^{\mathrm{F},t}\big(h_{\boldsymbol{v}}^{\mathrm{F},t}, \{\!\!\{(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{F},t} \mid j \in [k]) \mid a \in V\}\!\!\}\big). \tag{84}$$

Then we prove that, $\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},(L_k+2)(t-1)} \to \mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{E},(t-1)}$ and $\forall a, b \in [k], \mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},(L_k+2)t} \to \mathcal{I}_{e,\boldsymbol{v}_a,\boldsymbol{v}_b}^{\mathrm{E},t}$ by induction. Note that by proving the above-mentioned two derivations, we can get $h_{\boldsymbol{v}}^{\mathrm{F},(L_k+2)(t-1)+1} \to h_{\boldsymbol{v}}^{\mathrm{E},t}$ and $\forall a, b \in [k], h_{\boldsymbol{v}}^{\mathrm{F},(L_k+2)t} \to e_{\boldsymbol{v}_a,\boldsymbol{v}_b}^{t}$ directly, thanks to the universal approximation theorem.

- $t = 1$ holds.

  We first prove that $\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},0} \to \mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{E},0}$. By definition we have

  $$\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},0} = \big(h_{\boldsymbol{v}}^{\mathrm{F},0}, \{\!\!\{(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{F},0} \mid j \in [k]) \mid a \in V\}\!\!\}\big). \tag{85}$$

  Since the following two derivations hold

  $$h_{\boldsymbol{v}}^{\mathrm{F},0} \to h_{\boldsymbol{v}}^{\mathrm{E},0}, \tag{86}$$

  $$\forall a \in V, j \in [k], \quad h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{F},0} \to (h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_{(j-1) \bmod k} a}), \tag{87}$$

  together we have

  $$\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},0} \to \big(h_{\boldsymbol{v}}^{\mathrm{E},0}, \{\!\!\{(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_{(j-1) \bmod k} a} \mid j \in [k]) \mid a \in V\}\!\!\}\big). \tag{88}$$

  By shuffling sequence, we have

  $$\forall a \in V, j \in [k], \quad \Big(\big(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_{(j-1) \bmod k} a}\big) \mid j \in [k]\Big) \to \Big(\big(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_j a}\big) \mid j \in [k]\Big). \tag{89}$$

  Therefore,

  $$\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},0} \to \big(h_{\boldsymbol{v}}^{\mathrm{E},0}, \{\!\!\{(h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_j a} \mid j \in [k]) \mid a \in V\}\!\!\}\big) \tag{90}$$

  $$\to \big(h_{\boldsymbol{v}}^{\mathrm{E},0}, (\{\!\!\{h_{\Phi_j(\boldsymbol{v},a)}^{\mathrm{E},0}, e_{\boldsymbol{v}_j a} \mid a \in V\}\!\!\} \mid j \in [k])\big) = \mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{E},0}. \tag{91}$$

  Then we are going to prove that $\forall a, b \in [k], \mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},L_k+2} \to \mathcal{I}_{e,\boldsymbol{v_a},\boldsymbol{v_b}}^{\mathrm{E},1}$. As k-F-DisGNN can express arbitrary permutation-equivariant linear transformation, we have

  $$\mathcal{I}_{\mathrm{MP},\boldsymbol{v}}^{\mathrm{F},L_k+2} \to h_{\boldsymbol{v}}^{\mathrm{F},L_k+2} \tag{92}$$

  $$\to \Big(h_{\boldsymbol{v}}^{\mathrm{F},L_k+2}, \big(h_{\boldsymbol{v}}^{\mathrm{F},L_k+2} \mid u, v \in [k]\big)\Big) \tag{93}$$

  $$\to \Big(h_{\boldsymbol{v}}^{\mathrm{F},L_k+2}, \big(\sum_{\boldsymbol{w} \in V^k \mid \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b} h_{\boldsymbol{w}}^{\mathrm{F},2} \mid u, v \in [k]\big)\Big) \tag{94}$$

  $$\to \Big(h_{\boldsymbol{v}}^{\mathrm{F},L_k+2}, \big(\sum_{\boldsymbol{w} \in V^k \mid \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b} f(h_{\boldsymbol{w}}^{\mathrm{F},1}) \mid u, v \in [k]\big)\Big), \tag{95}$$

where f, a tuple-wise operation, implements deepset function with sum. Note that we use L_k message passing layers of k-F-DisGNN to implement the arbitrary permutation equivariant linear transformation. Therefore,

$$\mathcal{I}^{\mathrm{F},L_k+2}_{\mathrm{MP},\boldsymbol{v}} \to \Big(h^{\mathrm{F},L_k+2}_{\boldsymbol{v}}, \big(\{\!\{h^{\mathrm{F},1}_{\boldsymbol{w}} \mid \boldsymbol{w} \in V^k, \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b\}\!\} \mid u, v \in [k]\big)\Big). \tag{96}$$

As the following derivations hold

$$\forall \boldsymbol{w} \in V^k,\ \ h^{\mathrm{F},1}_{\boldsymbol{w}} \to h^{\mathrm{E},1}_{\boldsymbol{w}}, \tag{97}$$

$$h^{\mathrm{F},L_k+2}_{\boldsymbol{v}} \to h^{\mathrm{F},0}_{\boldsymbol{v}} \to e_{\boldsymbol{v}_a\boldsymbol{v}_b}, \tag{98}$$

we have

$$\mathcal{I}^{\mathrm{F},L_k+2}_{\mathrm{MP},\boldsymbol{v}} \to \big(e_{\boldsymbol{v}_a\boldsymbol{v}_b}, (\{\!\{h^{\mathrm{E},1}_{\boldsymbol{w}} \mid \boldsymbol{w} \in V^k \mid \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b\}\!\} \mid u, v \in [k])\big) = \mathcal{I}^{\mathrm{E},1}_{\mathrm{MP},\boldsymbol{v}_{\mathrm{a}}\boldsymbol{v}_{\mathrm{b}}}. \tag{99}$$

- If $\forall t < T$, $\mathcal{I}^{\mathrm{F},(L_k+2)(t-1)}_{\mathrm{MP},\boldsymbol{v}} \to \mathcal{I}^{\mathrm{E},(t-1)}_{\mathrm{MP},\boldsymbol{v}}$, and $\forall a, b \in [k], \mathcal{I}^{\mathrm{F},(L_k+2)t}_{\mathrm{MP},\boldsymbol{v}} \to \mathcal{I}^{\mathrm{E},t}_{e,\boldsymbol{v}_a,\boldsymbol{v}_a}$. We want to prove that the two derivations still hold for $t = T$.

  First we prove that $\mathcal{I}^{\mathrm{F},(L_k+2)(T-1)}_{\mathrm{MP},\boldsymbol{v}} \to \mathcal{I}^{\mathrm{E},(T-1)}_{\mathrm{MP},\boldsymbol{v}}$. From the known conditions, we can get

$$h^{\mathrm{F},(L_k+2)(T-1)}_{\boldsymbol{v}} \to h^{\mathrm{F},(L_k+2)(T-2)+1}_{\boldsymbol{v}} \to h^{\mathrm{E},(T-1)}_{\boldsymbol{v}}, \tag{100}$$

$$\forall a \in V, j \in [k],\ \ h^{\mathrm{F},(L_k+2)(T-1)}_{\Phi_j(\boldsymbol{v},a)} \to (h^{\mathrm{F},(L_k+2)(T-2)+1}_{\Phi_j(\boldsymbol{v},a)}, h^{\mathrm{F},(L_k+2)(T-1)}_{\Phi_j(\boldsymbol{v},a)}) \to (h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_{(j-1) \bmod k}a}). \tag{101}$$

Together with Equation (84), we have

$$\mathcal{I}^{\mathrm{F},(L_k+2)(T-1)}_{\mathrm{MP},\boldsymbol{v}} \to \big(h^{\mathrm{E},(T-1)}_{\boldsymbol{v}}, \{\!\{(h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_{(j-1) \bmod k}a} \mid j \in [k]) \mid a \in V\}\!\}\big). \tag{102}$$

By shuffling sequence, we have

$$\forall a \in V, j \in [k],\ \ \big((h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_{(j-1) \bmod k}a}) \mid j \in [k]\big) \to \big((h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_ja}) \mid j \in [k]\big). \tag{103}$$

Therefore,

$$\mathcal{I}^{\mathrm{F},(L_k+2)(T-1)}_{\mathrm{MP},\boldsymbol{v}} \to \big(h^{\mathrm{E},(T-1)}_{\boldsymbol{v}}, \{\!\{(h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_ja} \mid j \in [k]) \mid a \in V\}\!\}\big) \tag{104}$$

$$\to \Big(h^{\mathrm{E},(T-1)}_{\boldsymbol{v}}, \big(\{\!\{h^{\mathrm{E},(T-1)}_{\Phi_j(\boldsymbol{v},a)}, e^{(T-1)}_{\boldsymbol{v}_ja} \mid a \in V\}\!\} \mid j \in [k]\big)\Big) = \mathcal{I}^{\mathrm{E},(T-1)}_{\mathrm{MP},\boldsymbol{v}}. \tag{105}$$

Then we are going to prove that $\forall a, b \in [k], \mathcal{I}^{\mathrm{F},(L_k+2)T}_{\mathrm{MP},\boldsymbol{v}} \to \mathcal{I}^{\mathrm{E},T}_{e,\boldsymbol{v}_a,\boldsymbol{v}_b}$. As k-F-DisGNN can express arbitrary permutation-equivariant linear transformation, we have

$$\mathcal{I}^{\mathrm{F},(L_k+2)T}_{\mathrm{MP},\boldsymbol{v}} \to \Big(h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}}, \big(h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}} \mid u, v \in [k]\big)\Big) \tag{106}$$

$$\to \Big(h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}}, \big(\sum_{\boldsymbol{w} \in V^k \mid \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b} h^{\mathrm{F},(L_k+2)(T-1)+2}_{\boldsymbol{w}} \mid u, v \in [k]\big)\Big) \tag{107}$$

$$\to \Big(h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}}, \big(\sum_{\boldsymbol{w} \in V^k \mid \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b} f(h^{\mathrm{F},(L_k+2)(T-1)+1}_{\boldsymbol{w}}) \mid u, v \in [k]\big)\Big), \tag{108}$$

where f, a tuple-wise operation, implements deepset function with sum, so

$$\mathcal{I}^{\mathrm{F},(L_k+2)T}_{\mathrm{MP},\boldsymbol{v}} \to \Big(h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}}, \big(\{\!\{h^{\mathrm{F},(L_k+2)(T-1)+1}_{\boldsymbol{w}} \mid \boldsymbol{w} \in V^k, \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b\}\!\} \mid u, v \in [k]\big)\Big). \tag{109}$$

As

$$h^{\mathrm{F},(L_k+2)(T-1)+1}_{\boldsymbol{w}} \to h^{\mathrm{E},T}_{\boldsymbol{w}}, \tag{110}$$

$$h^{\mathrm{F},(L_k+2)T}_{\boldsymbol{v}} \to h^{\mathrm{F},0}_{\boldsymbol{v}} \to e_{\boldsymbol{v}_a\boldsymbol{v}_b}, \tag{111}$$

we have

$$\mathcal{I}^{\mathrm{F},(L_k+2)T}_{\mathrm{MP},\boldsymbol{v}} \to \big(e_{\boldsymbol{v}_a\boldsymbol{v}_b}, (\{\!\{h^{\mathrm{E},T}_{\boldsymbol{w}} \mid \boldsymbol{w} \in V^k, \boldsymbol{w}_u = \boldsymbol{v}_a, \boldsymbol{w}_v = \boldsymbol{v}_b\}\!\} \mid u, v \in [k])\big) = \mathcal{I}^{\mathrm{E},T}_{\mathrm{MP},\boldsymbol{v}_{\mathrm{a}}\boldsymbol{v}_{\mathrm{b}}}. \tag{112}$$

C. Detailed Model Design and Analysis

Radial basis function. In k-DisGNNs, we use radial basis functions (RBF) $f_{\mathrm{e}}^{\mathrm{rbf}}: \mathbb{R} \to \mathbb{R}^{H_e}$ to expand the distance between two nodes into an H_e-dimension vector. This can reduce the number of learnable parameters and additionally provides a helpful inductive bias (Klicpera et al., 2020b). The appropriate choice of RBF is beneficial, and we use nexpnorm RBF defined as

$$f_e^{\mathrm{rbf}}(e_{ij})[k] = e^{-\beta_k(\exp(-e_{ij})-\mu_k)^2}, \tag{113}$$

where β_k, μ_k are coefficients of the k^{th} basis. Experiments show that this RBF performs better than others, such as the Bessel function used in Klicpera et al. (2020b); Gasteiger et al. (2021).

Initialization function. We realize f_{init} of k-DisGNNs' initialization block in Section 5 as follows

$$f_{\mathrm{init}}(\boldsymbol{v}) = \bigodot_{i\in[k]} f_z^i\big(f_z^{\mathrm{emb}}(z_{v_i})\big) \odot \bigodot_{i,j\in[k],i\neq j} f_e^{ij}\big(f_{\mathrm{e}}^{\mathrm{rbf}}(e_{ij})\big), \tag{114}$$

where $\odot$ represents Hadamard product and $\bigodot$ represents Hadamard product over all the elements in the subscript. $f_z^{\mathrm{emb}}: \mathbb{Z}^+ \to \mathbb{R}^{H_z}$ is a learnable embedding function, $f_{\mathrm{e}}^{\mathrm{rbf}}: \mathbb{R}^+ \to \mathbb{R}^{H_e}$ is a radial basis function, and f_z^i, f_e^{ij} are neural networks such as MLPs that maps the embeddings of z and e to the common vector space $\mathbb{R}^H$.

Note that f_{init} can learn an injective representation for $\boldsymbol{v}$ as long as the embedding dimensions are high enough. For example, $f_{\mathrm{init}}(\boldsymbol{v})$ is an k^2-length embedding where the first k positions represent the k atomic numbers of the tuple, and the remaining (k^2-k) positions represent the (k^2-k) distances. Experiments show that this function performs better than just concatenating z_i and e_{ij} as a vector and passing it through MLPs.

By this means, tuples with different **equality patterns** (Maron et al., 2018) can also be distinguished, i.e., get different representations, without explicitly incorporating the representation of equality pattern. This is because in the context of distance graphs, tuples with different equality patterns will have quite different **distance matrices** (elements are zero at the positions where two nodes are the same).

Message passing and output functions. As mentioned in Section 5, we realize the message passing functions as injective functions to ensure expressiveness. To be specific, we embed all the multisets in Equation (10, 14, 15) and output function with the injective multiset function proposed in Xu et al. (2018), and use the matrix multiplication methods proposed in Maron et al. (2019) to implement the message passing functions of k-F-DisGNN (Equation 11, 12).

Inductive bias. Although in theory, there is no need to distinguish tuples with different equality patterns explicitly, we still do so to incorporate inductive bias into the model for better learning. Specifically, we modify the initialization function and the output function as follows: 1. At the initialization step, we learn embeddings for different equality patterns and incorporate them into the results of Equation (114) through a Hadamard product. 2. At the output step, we separate tuples where all sub-nodes are the same and the others into different multisets. These modifications are both beneficial for training and generalization.

Complexity analysis of k-E-DisGNN. At the message passing step, k-DisGNN and k-F-DisGNN share the same time complexity of $O(kn^{k+1})$, as there are n^k k-tuples and each tuple needs to aggregate kn neighbors. k-E-DisGNN, on the other hand, is designed by incorporating edge features e_{ij}^t in Equation (13) to k-DisGNN. The time complexity of calculating all the edge features e_{ij}^t is $O(C_k^2 n^k)$, because the number of $H_{uv}^t(i,j)$ in Equation (13) is C_k^2, and each of $H_{uv}^t(i,j)$ needs to incorporate $n^{(k-2)}$ tuples and there are n^2 edge features that need to be calculated. It is worth noting that $O(C_k^2 n^k) = O(n^k) < O(n^{k+1}) = O(kn^{k+1})$, thus incorporating edge representations does not increase the time complexity of k-DisGNN.

D. Experiment Configuration

Table 4. Training settings.

	MD17	QM9
TRAIN SET SIZE	1000	110000
VAL. SET SIZE	1000	10000
BATCH SIZE	4	16
WARM-UP EPOCHS	25	3
MAX EPOCHS	4000	500
INITIAL LEARNING RATE	0.001	0.0003
DECAY ON PLATEAU PATIENCE (EPOCHS)	50	5
DECAY ON PLATEAU COOLDOWN (EPOCHS)	50	5
DECAY ON PLATEAU THRESHOLD	0.001	
DECAY ON PLATEAU FACTOR	0.05	
EXPONENTIAL DECAY RATE	0.99	

Training Setting. For QM9, we use the mean squared error (MSE) loss for training. For MD17, we use the weighted loss function

$$\mathcal{L}(\boldsymbol{X},\boldsymbol{z}) = (1-\rho)|f_\theta(\boldsymbol{X},\boldsymbol{z}) - \hat{t}(\boldsymbol{X},\boldsymbol{z})| + \frac{\rho}{N}\sum_{i=1}^{N}\sqrt{\sum_{\alpha=1}^{3}(-\frac{\partial f_\theta(\boldsymbol{X},\boldsymbol{z})}{\partial \boldsymbol{x}_{i\alpha}} - \hat{F}_{i\alpha}(\boldsymbol{X},\boldsymbol{z}))^2}, \tag{115}$$

where the force ratio ρ is coarsely tuned from $\{0.999, 0.99\}$.

We follow the same dataset split as GemNet (Gasteiger et al., 2021). We optimize all models using Adam (Kingma & Ba, 2014) with exponential decay and plateau decay learning rate schedulers, and also a linear learning rate warm-up. To prevent overfitting, we use early stopping on validation loss and an exponential moving average (EMA) with decay rate 0.999 for model parameters during validation and test. Detailed training setting can be referred to Table 4.

Model hyperparameters. The key model hyperparameters we coarsely tune are the rbf dimension, atom embedding dimension, tuple embedding dimension, and number of message passing blocks. For rbf dimension, we use 16 for MD17 and 32 for QM9. We choose the number of message passing blocks from $\{4, 5\}$. For 2-DisGNNs, we choose atom embedding dimension from $\{256, 512\}$, and use a fixed 512 for tuple embedding dimension. For 3-DisGNNs, we use 256 for atom embedding dimension and 320 for tuple embedding dimension. The detailed hyperparameters can be found in our codes.

E. Supplementary Experiment Information

Supplementary Explanation on MD17. In our experiments on MD17, most of the state-of-the-art performance we achieve is on force targets, and the loss on energy targets is relatively higher, see Table 1. On force prediction tasks, our models rank top 2 on force prediction tasks on average, and outperform the best results by a significant margin on several molecules, such as aspirin and malonaldehyde. However, the results on the energy prediction tasks are relatively lower, with 2-F-DisGNN ranking 3^{rd} and 3-E-DisGNN ranking 5^{th}.

This is due to the fact that we have assigned a quite **high weight** (0.99 or 0.999) to the **force loss** during training, similar to what GemNet(Gasteiger et al., 2021) does. In comparison, the TorchMD model (Thölke & De Fabritiis, 2021) assigned a weight of 0.8 to the force loss, resulting in better results on energy targets, but not as good results on force targets.

Actually, in molecular simulations, force prediction is a more challenging task. It determines the **accuracy** of molecular simulations and reflects the performance of a model better (Gasteiger et al., 2021). Therefore, it makes more sense to focus on force prediction. Furthermore, previous researches (Batzner et al., 2022; Christensen & Von Lilienfeld, 2020) have found that models can achieve significantly lower energy loss on the revised MD17 dataset than on the original MD17 dataset, while holding similar force accuracy on the two datasets. As analyzed in Batzner et al. (2022), this suggests that the **noise floor** on the original MD17 dataset is higher on the energies, indicating that better force prediction results are more meaningful than energy prediction results on the original MD17 datasets.

For a comprehensive comparison, we also conducted experiments on the revised MD17 dataset, and compared with two SOTA models, MACE (Batatia et al., 2022) and Allegro (Musaelian et al., 2023). The results are shown in table 5. Note that we achieved many state-of-the-art results on rMD17 without significantly tuning the hyperparameters. We believe that further fine-tuning could unlock the full potential of our models on this dataset.

Table 5. MAE loss on revised MD17. Energy (E) in kcal/mol, force (F) in kcal/mol/Å.

Target		MACE	Allegro	2F-Dis.
ASPIRIN	E	0.0507	0.0530	**0.0450**
	F	0.1522	0.1683	**0.1519**
AZOBENZENE	E	**0.0277**	**0.0277**	0.0343
	F	0.0692	**0.0600**	0.1228
BENZENE	E	0.0092	0.0069	**0.0045**
	F	0.0069	**0.0046**	0.0065
ETHANOL	E	0.0092	0.0092	**0.0086**
	F	0.0484	0.0484	**0.0386**
MALONAL.	E	0.0184	0.0138	**0.0131**
	F	0.0945	**0.0830**	0.0863
NAPHTHALENE	E	0.0115	**0.0046**	0.0075
	F	0.0369	**0.0208**	0.0361
PARACETAMOL	E	**0.0300**	0.0346	0.0368
	F	**0.1107**	0.1130	0.1397
SALICYLIC ACID	E	0.0208	0.0208	**0.0159**
	F	0.0715	**0.0669**	0.0821
TOLUENE	E	0.0115	0.0092	**0.0064**
	F	0.0346	0.0415	**0.0303**
URACIL	E	**0.0115**	0.0138	0.0144
	F	0.0484	**0.0415**	0.0922

Table 6. MAE loss on QM9.

Target	Unit	SchNet	Cormor.	DimeNet	DimeNet++	TorchMD	PaiNN	GNN-LF	2F-Dis.
μ	D	0.033	0.038	0.0286	0.0297	0.002	0.012	0.013	0.0221
α	a_0^3	0.235	0.085	0.0469	0.0435	0.01	0.045	0.0353	0.0455
ϵ_{HOMO}	meV	41	34	27.8	24.6	21.2	27.6	23.5	25.44
ϵ_{LUMO}	meV	34	38	19.7	19.5	17.8	20.4	17	22.05
$\Delta\epsilon$	meV	63	61	34.8	32.6	38	45.7	37.1	43.70
$\langle R^2 \rangle$	a_0^2	0.073	0.961	0.331	0.331	0.015	0.066	0.037	0.1065
ZPVE	meV	1.7	2.027	1.29	1.21	2.12	1.28	1.19	1.3353
U_0	meV	14	22	8.02	6.32	6.24	5.85	5.3	7.53
U	meV	19	21	7.89	6.28	6.3	5.83	5.24	7.78
H	meV	14	21	8.11	6.53	6.48	5.98	5.48	7.85
G	meV	14	20	8.98	7.56	7.64	7.35	6.84	8.56
c_v	cal/mol/K	0.033	0.026	0.0249	0.023	0.026	0.024	0.022	0.0233
AVERAGE IMPROVEMENT		-78.99%	-98.22%	0.00%	9.42%	25.00%	17.50%	27.82%	6.88%
RANK		7	8	6	4	2	3	1	5

Supplementary Results on QM9. We present the full results on QM9 in Table 6. We compare our model with 7 other models, including those that use invariant geometric features: SchNet (Schütt et al., 2018), DimeNet (Klicpera et al., 2020b), DimeNet++(Klicpera et al., 2020a), a model that uses irreducible representations: Cormorant(Anderson et al., 2019), those that use low-order equivariant representations: TorchMD (Thölke & De Fabritiis, 2021), PaiNN (Schütt et al., 2021) and a model that uses local frame methods: GNN-LF (Wang & Zhang, 2022). We calculate the average improvements of all the models relative to DimeNet and list them in the table.

We can see that although 2-F-DisGNN outperforms DimeNet on most of the targets, it performs much worse than some of the other models, such as TorchMD and GNN-LF. This may be because although 2-F-DisGNN can exploit and process all the 3-order geometric features (as stated in Section 6.1), it is sufficient for the conventional tasks like predicting energies or atomic forces on MD17, but is still not enough for learning the complex chemical attributes such as dipole moment and isotropic polarizability. These attributes may involve higher-order geometric information, such as dihedral angle information (4-order).

Note that in Gasteiger et al. (2021), researchers also found that GemNet-T, which is a simplified version of GemNet and includes only 3-tuples, performs quite well on MD17 but not on COLL (Klicpera et al., 2020a) (a dataset similar to QM9 but more challenging). In fact, since GemNet-T includes only 3-tuples, it is quite similar to 2-F-DisGNN and can mainly learn and process 3-order geometric information. This may explain to some extent why these two models have similar experiment results. Moreover, SchNet makes use of continuous-filter convolutional layers, and can actually be unified by vanilla DisGNNs, which can only learn 2-order geometric features. Thus it shows much worse performance on QM9.

These all indicate that **higher order geometric information** is needed for more **complex tasks**, inspiring us to use k-DisGNNs with a higher k value. However, this is challenging to achieve as of now: We have discussed the reasons and possible solutions in the limitation part in Section 8. We will explore more about it in our future work.

F. Supplenmentary Related Work

Interatomic potentials. There is a wealth of research in the field of interatomic potentials. Bartók et al. (2013) designed a similarity measure between neighborhood environments by expanding the density field of atoms into a spherical harmonics basis, which allows for kernel methods to be applied. Shapeev (2016); Drautz (2019) expand the potential function with a basis and learn the coefficients from data. Shapeev (2016) use spherical harmonics and radial basis functions, while Drautz (2019) use equivariant polynomials. Building on Drautz (2019), Batatia et al. (2022) proposed a GNN variant that can effectively exploit the rich geometric information in atoms' local environment. Additionally, Joshi et al. (2023) proposed geometric variants of the WL test to characterize the expressiveness of invariant and equivariant geometric GNNs for the general graph setting. All of Shapeev (2016); Drautz (2019); Batatia et al. (2022); Joshi et al. (2023) leverage the many-body expansion and can include k-tuples during embedding atom's neighbors, allowing atoms to obtain k-order geometric information from their local environments like k-DisGNNs.

However, there are key differences between these works on interatomic potentials and k-DisGNNs in terms of their message passing units and use of representations. The former assumes that the total energy can be approximated by the sum of energies of atomic environments of individual atoms and therefore use atoms as their message passing units, while k-DisGNNs pass messages among k-tuples. In addition, while interatomic potential methods leverage equivariant representations to enhance their expressiveness, k-DisGNNs completely decouple E(3)-symmetry and use invariant representations to achieve universality.