Cycle Invariant Positional Encoding for Graph Representation Learning
– Supplementary Material –

Anonymous Author(s)
Anonymous Affiliation
Anonymous Email

In this supplementary material, we provide:

1. the proof of theorems in Section 4.3;
2. the experimental details that include the assets we used and the limitations of the paper;
3. additional experiments.

1 Proof of Theorem 4.3 in the main paper

We begin by introducing the *graph isomorphism*. For a pair of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, if there exists a bijective mapping $f : V_1 \to V_2$, so that for any edge $(u_1, v_1) \in E_1$, it satisfies that $(f(u_1), f(v_1)) = (u_2, v_2) \in E_2$, then G_1 is isomorphic to G_2, otherwise they are not isomorphic. Up to now, there is no polynomial algorithm for solving the graph isomorphism problem. One popular method is to use the k-order Weisfeiler-Leman [17] algorithm (k-WL). It is known that 1-WL is as powerful as 2-WL, and for $k \geq 2$, $(k+1)$-WL is more powerful than k-WL.

We then provide the theoretical results below:

Theorem 1.1. *CycleNet is strictly more powerful than 2-WL, and can distinguish graphs that are not distinguished by 3-WL.*

Proof. **The pair of graphs that 3-WL cannot distinguish while CycleNet can.** It is shown in [1] that 3-WL cannot differentiate the 4×4 Rook Graph and the Shrikhande Graph shown in Figure 1. We then compute the orthogonal projector of the cycle space of the Hodge Laplacian for each graph and denote them as O_{rook} and O_{sh}. We observe that each column of O_{rook} contains 22 zeros, whereas each column of O_{sh} contains 16 zeros. To differentiate between the two graphs, we can use the function $|O_{rook} - O_{sh}|$, which can be approximated using an invariant graph network (IGN) followed by a multilayer perceptron (MLP). Specifically, the 2-2 layer of the IGN can obtain the O_{rook} and O_{sh}, and the MLP can approximate the absolute function.

More powerful than the 2-WL. Using models such as [18] to be the backbone GNNs can distinguish any pair of non-isomorphic graphs that 2-WL can distinguish. Since there exist graphs such as the 4x4 Rook Graph and the Shrikhande graph that 2-WL cannot distinguish, while CycleNet can. Therefore, CycleNet is more powerful than 2-WL. □

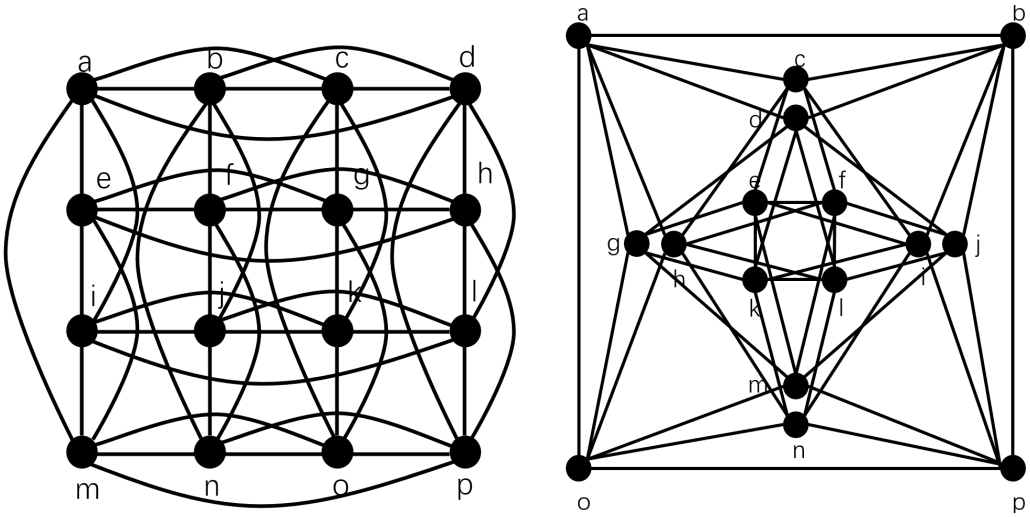
2 Proof of Theorem 4.6 in the main paper

We restate the theorem as follows:

Theorem 2.1. *Using the length of shortest cycle basis as the edge structural embedding can distinguish certain pair of graphs that are not distinguished by 3-WL, as well as pair of graphs that are not distinguished by 4-WL.*

Proof. **The pair of graphs that 4-WL cannot distinguish.** Consider the set of graphs called the Cai-Fürer-Immerman (CFI) graphs [4]. The sequence of graphs $G_k^{(\ell)}, \ell = 0, 1, \ldots, k+1$ is defined

Submitted to the Second Learning on Graphs Conference (LoG 2023). Do not distribute.



(a) (b)

Figure 1: (a) the 4x4 Rook Graph and (b) the Shrikhande Graph

as following,

$$V_{G_k^{(\ell)}} = \Big\{ u_{a,\vec{v}} \Big| a \in [k+1], \vec{v} \in \{0,1\}^k \text{ and } \vec{v} \text{ contains} \quad \begin{array}{ll} \text{an even number of 1's,} & \text{if } a = 1, 2, \ldots, k-\ell+1, \\ \text{an odd number of 1's,} & \text{if } a = k-\ell+2, \ldots, k+1. \end{array} \Big\} \tag{1}$$

Edges exists between two nodes $u_{a,\vec{v}}$ and $u_{a',\vec{v}'}$ of $G_k^{(\ell)}$ if and only if there exists $m \in [k]$ such that $a' \mod (k+1) = (a+m) \mod (k+1)$ and $v_m = v'_{k-m+1}$.

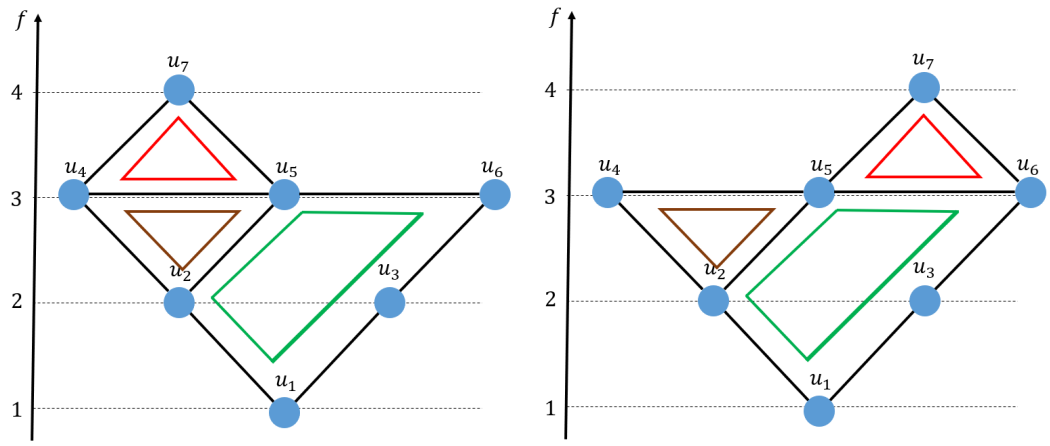
Denote the two graphs $G = G_4^{(0)}$ and $H = G_4^{(1)}$. It is shown in [21] that 4-WL cannot differentiate the pair of graphs.

The SCB can distinguish them. We begin by presenting the computation of the shortest cycle basis. Let $C_T \in R^{m \times l}$ denote the set of all tight cycles, where m is the number of edges and l is the number of tight cycles. The definition of tight cycles is described in Section 3.3 of the main paper. For a given cycle j, $C_T[i][j]$ is equal to 1 if edge i is in cycle j, and 0 otherwise. We define $low_{C_T}(j)$ as the maximum row index i such that $C_T[i][j] = 1$. To compute the shortest cycle basis, we use the matrix reduction algorithm, which is shown in Algorithm 1.

Algorithm 1 Matrix Reduction

Input: the set of tight cycles C_T
the shortest cycle basis $SCB = \{\}$
$C_T = \text{SORT}(C_T)$
for $j = 1$ **to** l **do**
  while $\exists k < j$ with $low_{C_T}(k) = low_{C_T}(j)$ **do**
    add column k to column j and
  end while
  if column j is not a zero vector **then**
    add the original column j to SCB
  end if
end for
Output: the shortest cycle basis SCB

In the given algorithm, the symbol "add" represents the modulo-2 sum of two binary vectors. It should be noted that Algorithm 1 may not be the fastest algorithm for computing the SCB, but most acceleration methods are based on it. The algorithm processes the cycles in C_T in order of increasing length, with shorter cycles added to the shortest cycle basis before longer cycles. If any cycle can



(a) (b)

Figure 2: Graphs that the PEOI encoding of the cycle incidence matrix can differentiate, while the number of cycles and the extended persistence diagrams cannot.

be represented as a sum of multiple cycles whose lengths are no more than k, then the length of the longest cycle in the shortest cycle basis will be k. We denote a cycle with length k as a k-cycle.

We obtain a total of 40 nodes for G and H by traversing a from 1 to 5 according to Equation 1. For example, in G, node 1 denotes $u_{1,\{0,0,0,0\}}$, and node 2 denotes $u_{1,\{0,0,1,1\}}$. We then traverse these nodes to obtain the edges. For example, edge 1 denotes $(1, 9)$ in G, which corresponds to node $u_{1,\{0,0,0,0\}}$ and node $u_{2,\{0,0,0,0\}}$. It is observed that in H, a 4-cycle exists between edges $\{8, 9, 24, 25\}$. These edges correspond to four nodes: $u_{1,\{0,0,0,0\}}$, $u_{1,\{0,0,1,1\}}$, $u_{4,\{0,0,0,0\}}$, and $u_{4,\{0,1,0,1\}}$. The 4-cycle cannot be represented by the modulo-2 sum of 3-cycles since there is no 3-cycle whose edge with the maximum index after matrix reduction borns earlier than edge 25, that is $(u_{1,\{0,0,1,1\}}, u_{4,\{0,1,0,1\}})$. Therefore the SCB of H contains 4-cycle.

The same 4-cycle also exists in G, and it can be represented by 38 3-cycles: $\{12, 288, 8\}$, $\{89, 94, 296\}$, $\{12, 148, 1\}$, $\{23, 215, 19\}$, $\{105, 108, 301\}$, $\{23, 282, 27\}$, $\{218, 282, 215\}$, $\{195, 318, 199\}$, $\{195, 316, 198\}$, $\{103, 105, 267\}$, $\{9, 144, 1\}$, $\{115, 121, 217\}$, $\{115, 124, 220\}$, $\{218, 313, 220\}$, $\{234, 314, 236\}$, $\{121, 124, 301\}$, $\{147, 318, 151\}$, $\{147, 316, 150\}$, $\{146, 314, 149\}$, $\{146, 312, 148\}$, $\{170, 234, 165\}$, $\{170, 313, 172\}$, $\{192, 198, 296\}$, $\{192, 199, 298\}$, $\{99, 108, 172\}$, $\{213, 267, 217\}$, $\{99, 101, 165\}$, $\{97, 103, 141\}$, $\{97, 109, 149\}$, $\{213, 266, 216\}$, $\{81, 89, 144\}$, $\{81, 94, 150\}$, $\{19, 216, 25\}$, $\{266, 270, 298\}$, $\{101, 109, 236\}$, $\{141, 270, 151\}$, $\{28, 312, 27\}$, $\{28, 288, 24\}$. The same situations exist for all other 4-cycles or cycles longer than 4. We also observe that there have been 281 3-cycles in the SCB. Considering that it is equal to the Betti number of G, the SCB does not contain any 4-cycle.

The pair of graphs that 3-WL cannot distinguish while SCB can. There are 24 3-cycles and 9 4-cycles in the SCB of the 4×4 Rook Graph, while there are 31 3-cycles and 2 4-cycles in the SCB of the Shrikhande Graph. Therefore the SCB can differentiate them.

□

3 Proof of 4.4 in the main paper

We restate the theorem as follows:

Theorem 3.1. *If choosing the same set of cycles. The PEOI encoding of the cycle incidence matrix is more powerful than using its number in terms of distinguishing non-isomorphic graphs.*

Proof. **PEOI can extract the number of cycles.** In Proposition 4.2 in the main paper, if we set ρ_1 as a function that consistently produces "1", ρ_2 as a function that ignores the $X[i][k]$ element while being an identity function for the rest elements, and ρ_3 as an identity function, we can obtain the number of cycles. Therefore, the PEOI encoding of the cycle incidence matrix is at least as powerful as the number of cycles.

Then we use the pair of graphs shown in Figure 2 as an example.

The number of cycles cannot differentiate the pair of graphs. In these two graphs the number of cycles will remain the same. For example, if using all the cycles, there are both 3 cycles in Figure 2(a) and Figure 2(b). If using cycles of a certain length, there are both 2 3-cycles and 1 5-cycle in Figure 2(a) and Figure 2(b). Therefore, only using the number of cycles cannot differentiate the pair of graphs.

The PEOI encoding of the cycle incidence matrix can differentiate the pair of graphs. The cycle incidence matrix of these two graphs is listed as follows:

$$\begin{bmatrix} & \gamma_g & \gamma_b & \gamma_r \\ (u_1,u_2) & 1 & 0 & 0 \\ (u_1,u_3) & 1 & 0 & 0 \\ (u_2,u_4) & 0 & 1 & 0 \\ (u_2,u_5) & 1 & 1 & 0 \\ (u_3,u_6) & 1 & 0 & 0 \\ (u_4,u_5) & 0 & 1 & 1 \\ (u_5,u_6) & 1 & 0 & 0 \\ (u_4,u_7) & 0 & 0 & 1 \\ (u_5,u_7) & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} & \gamma_g & \gamma_b & \gamma_r \\ (u_1,u_2) & 1 & 0 & 0 \\ (u_1,u_3) & 1 & 0 & 0 \\ (u_2,u_4) & 0 & 1 & 0 \\ (u_2,u_5) & 1 & 1 & 0 \\ (u_3,u_6) & 1 & 0 & 0 \\ (u_4,u_5) & 0 & 1 & 0 \\ (u_5,u_6) & 1 & 0 & 1 \\ (u_5,u_7) & 0 & 0 & 1 \\ (u_6,u_7) & 0 & 0 & 1 \end{bmatrix}$$

For Proposition 4.2 in the main paper, we can define $\rho_1(X[i][k], X[j][k]) = 2X[i][k] + X[j][k]$, $\rho_2(X[i][k], Y) = RELU(Y - 16)$, and ρ_3 to be an identity function. Therefore, for the graph shown in Figure 2(a), the PEOI encoding is $\{4, 4, 2, 6, 4, 4, 4, 2, 2\}$; for the graph shown in Figure 2(b), the PEOI encoding is $\{4, 4, 2, 6, 4, 2, 6, 2, 2\}$. According to Proposition 4.1 in the main paper, we can differentiate the pair of graphs using CycleNet-PEOI.

Therefore, the PEOI encoding of the cycle incidence matrix is more powerful than the number of cycles.

□

4 Proof of Theorem 4.5 in the main paper

The classic EPDs [5] can be used to measure the saliency of connected components and high-order topological structures such as voids. However, recent works [19, 20, 22] have mainly used the one-dimensional (1D) EPD as augmented topological features, particularly the features corresponding to cycles. Therefore, in this section, we mainly focus on comparing our encoding with the **1D EPDs corresponding to cycles**. For ease of complexity, we will omit the terms “1D" and “that correspond to cycles" in the rest of this section, and only use “EPDs".

Theorem 4.1. *If choosing the same set of cycles. The PEOI encoding of the cycle incidence matrix can differentiate graphs that cannot be differentiated by the extended persistence diagram. If adding the filter function to the cycle incidence matrix, the PEOI encoding of the cycle incidence matrix is more powerful than using its extended persistence diagram in terms of distinguishing non-isomorphic graphs.*

Proof. **The extended persistence diagram (EPD).** Persistent homology [7, 8] captures topological structures such as connected components and cycles, and summarizes them in a point set called the persistence diagram (PD). It is found that the extended persistence diagrams (EPD) [5] is a variant of PD that encodes richer cycle information. Specifically, an EPD is a set of points in which every point represents the significance of a topological structure in terms of a scalar function known as the filter function. Recent studies have shown that the extended persistence point of a cycle is the combination of the maximum and minimum filter values of the point in the cycle [20]. Note that in this paper, we focus on the EPDs of cycles, and do not consider the EPDs of other structures.

We illustrate the computation of the EPD for the graph in Figure 2(a), where the filter function is defined as the shortest path distance from a selected root node u_1 to other nodes. This is a common filter function used in previous models [23, 19]. We plus one to the filter value in case of the zero value. Using this definition, we have: $f(u_1) = 1$, $f(u_2) = f(u_3) = 2$, $f(u_4) = f(u_5) = f(u_6) = 3$, and $f(u_7) = 4$. The extended persistence point of the red cycle, the brown cycle, and the green cycle are $(3, 1)$, $(3, 2)$, and $(4, 3)$, respectively. Therefore the EPD of Figure 2(a) is $\{(3, 1), (3, 2), (4, 3)\}$.

We can define similarly the filter value for Figure 2(b), and the extended persistence points of the three cycles are the same as the cycles in Figure 2(a). Therefore, the EPD of Figure 2(b) is also $\{(3,1),(3,2),(4,3)\}$, and the EPD cannot differentiate the pair of graphs. Note that the PEOI encodings for these two graphs are different, as shown in the proof of Theorem 3.1.

Add the filter function to CycleNet-PEOI. It is worth noting that the filter function, which plays a crucial role in constructing the EPD, is not explicitly contained in the cycle incidence matrix. As a result, encoding the original cycle incidence matrix using the proposed PEOI method is not sufficient to extract the EPD.

However, we can incorporate the filter function into the proposed model by adding the filter function to the cycle incidence matrix. For example, we define the filter value of an edge as the minimum value of the nodes in the edge, and obtain the filter values of the edges in Figure 2(a) as $\{1,1,2,2,2,3,3,3,3\}$. Next, we compute the dot product between the filter values of edges and the cycle incidence matrix, which results in the so-called filter-enhanced cycle incidence matrix. Similarly, we can obtain the filter-enhanced cycle incidence matrix for Figure 2(b). The two matrices are listed below:

$$\begin{bmatrix} & \gamma_g & \gamma_b & \gamma_r \\ (u_1,u_2) & 1 & 0 & 0 \\ (u_1,u_3) & 1 & 0 & 0 \\ (u_2,u_4) & 0 & 2 & 0 \\ (u_2,u_5) & 2 & 2 & 0 \\ (u_3,u_6) & 2 & 0 & 0 \\ (u_4,u_5) & 0 & 3 & 3 \\ (u_5,u_6) & 3 & 0 & 0 \\ (u_4,u_7) & 0 & 0 & 3 \\ (u_5,u_7) & 0 & 0 & 3 \end{bmatrix} \qquad \begin{bmatrix} & \gamma_g & \gamma_b & \gamma_r \\ (u_1,u_2) & 1 & 0 & 0 \\ (u_1,u_3) & 1 & 0 & 0 \\ (u_2,u_4) & 0 & 2 & 0 \\ (u_2,u_5) & 2 & 2 & 0 \\ (u_3,u_6) & 2 & 0 & 0 \\ (u_4,u_5) & 0 & 3 & 0 \\ (u_5,u_6) & 3 & 0 & 3 \\ (u_5,u_7) & 0 & 0 & 3 \\ (u_6,u_7) & 0 & 0 & 3 \end{bmatrix}$$

Define the PEOI encoding. We can use a 2-layer MLP to approximate the minimum function between two elements. The hidden layer contains 4 nodes, and the ReLU activation function is used. The weights from the input layer to the hidden layer are $(1,1)$, $(1,-1)$, $(-1,1)$, and $(-1,-1)$, respectively, and the biases are set to 0 for all nodes. The weights from the hidden layer to the output layer are 0.5, -0.5, -0.5, -0.5, respectively.

In Proposition 4.2 in the main paper, We set ρ_1 as the minimum function, which is approximated by the 2-layer MLP. ρ_2 is defined as a function that ignores the $X[i][k]$ element while being an identity function for another element. ρ_3 is set as an identity function.

The defined encoding can differentiate the graphs that EPD can also differentiate. Assume that there exists a pair of graphs G_1 and G_2 whose EPDs are different, we can assume that there exist a pair of cycles whose lowest filter values are different (the highest filter values can be treated similarly). Under this assumption, we can define the filter value for edges as the minimum value of the nodes in the edge. Using the PEOI encoding defined above, we can extract the lowest filter value of these two cycles. We then use an injective function on the multiset of cycle embeddings to produce different outputs for these two graphs. Therefore the defined encoding can differentiate G_1 and G_2.

In conclusion, by incorporating the filter function, CycleNet-PEOI can differentiate all pairs of graphs that the EPDs can differentiate, and can distinguish graphs that EPDs cannot. Therefore, it is more powerful than EPDs in terms of distinguishing non-isomorphic graphs.

□

5 Implementation details

Encoding of CycleNet-PEOI. Based on Proposition 4.2 in the main paper, we provide a pytorch-like pseudo-code for the PEOI encoding in Figure 3.

In certain situations where graphs are dense and large, the original PEOI encoding may bring extra computational and memory costs. In these situations, we can ignore the final $X[i][k]$ element in Proposition 4.2 in the main paper, and then the memory cost will be no larger than $O(m \times g)$.

Encoding of CycleNet. The full approximation power requires high-order tensors to be used for the IGN [15, 16, 10]. In practice, we follow the settings of [14] and restrict the tensor dimensions

PyTorch-like pseudo-code for PEOI encoding

```
class PEOI(nn.Module):

    def __init__(self, D1, D2, D3):
        self.rho1 = MLP(2, D1) # in dim=2, out dim=D1
        self.rho2 = MLP(1+D1, D2) # in dim=1+D1, out dim=D2
        self.rho3 = MLP(D2, D3) # in dim=D2, out dim=D3

    def forward(self, x):
        # x shape: m x g

        m, g = x.shape
        x1 = x.reshape(m, 1, 1, g).expand(-1, m, 1, -1) # m x m x 1 x g
        x2 = x.reshape(1, m, 1, g).expand(m, -1, 1, -1) # m x m x 1 x g
        w = self.rho1(cat((x1; x2), dim = 2)).sum(dim = 1) # m x m x 2 x g -> m x D1 x g
        w = self.rho2(cat((x.reshape(m, 1, g); w), dim = 1)).sum(dim = 2) # m x (1+D1) x g -> m
             x D2
        w = self.rho3(w) # m x D2 -> m x D3

        return w
```

Figure 3: PyTorch-like pseudo-code for PEOI encoding. Here "cat((x, y), dim = c)" denotes the concatenation of two matrices on the c-th dimension. "sum(dim=c)" denotes the sum operation over the c-th dimension.

for efficiency. This encoding, although losing certain theoretical power, shows strong empirical performance in [14].

Experimental details. In the synthetic experiments in the main paper, we use a 5-layer GIN [18] as the backbone model. We set the hidden dimension to 128, batch size to 16, and learning rate to 1e-3 with Adam as the optimizer. We use a ReduceLROnPlateau scheduler with the reduction factor set to 0.7, the patience set to 10, and the minimum learning rate set to 1e-6. In the synthetic experiments related to cycles, we use a point cloud dataset sampled on small cycles whose centers are on a big cycle. The diameters of the large cycle and small cycle are set to 20 and 1, respectively. We randomly sample 20 points from the large cycle and 60 points from the small cycle. After obtaining the node set, we generate a k-nearest-neighbor graph with the parameter k set to 3. There is no input feature for the prediction of the Betti Number. As for the prediction of EPD, we use the position of the node as the filtration function of the EPD. The input node feature is therefore the coordinates of the nodes.

For real-world benchmarks, we use SignNet or CWN as the backbone model on ZINC. Our settings follow exactly the settings of SignNet or CWN. For the superpixel classification and the trajectory classification benchmarks, we use SAT as the backbone model. Our settings follow exactly the settings of SAT. For the homology localization benchmark, we use Dist2cycle as the backbone model. Our settings follow exactly the settings of Dist2cycle. Notice that for backbone models that fill the cycles with 2-cells, the kernel space of the Hodge Laplacian may not contain any information. Therefore, we replace the kernel space encoding with the encoding based on the original Laplacians. All the experiments are implemented with two Intel Xeon Gold 5128 processors,192GB RAM, and 10 NVIDIA 2080TI graphics cards.

The assets we used. Our model is experimented on benchmarks from [6, 9, 11, 13, 3, 2] under the MIT license.

Limitations of the paper. First, we have shown that the representation power of our model is bounded by high-order WLs in terms of distinguishing non-isomorphic graphs.

Second, the proposed model may not perform well on benchmarks where cycle information is not relevant. For example, in high-order graphs where cycles are replaced by high-order structures like triangles or cells, the proposed CycleNet-PEOI model may not be suitable.

6 Additional experiments

We present additional evaluations on (1) the memory cost in terms of the number of trainable parameters; (2) the effectiveness of the introduced cycle-related embedding on a wider range of

Table 1: Additional experiments on ZINC

Framework	test MAE	Params
GIN	0.220	497394
+ CycleNet-Hodge	0.165	543876
+ CycleNet	0.153	543876
+ CycleNet-PEOI	0.153	512812
+ BasisNet	0.169	751810
GatedGCN	0.259	491597
+ CycleNet-Hodge	0.142	510539
+ CycleNet	0.137	510539
+ CycleNet-PEOI	0.188	504090
+BasisNet	0.139	716793
PNA	0.145	473681
+ CycleNet-Hodge	0.128	479081
+ CycleNet	0.089	479081
+ CycleNet-PEOI	0.111	483769
+ BasisNet	0.094	556323
SignNet	0.084	487082
+ CycleNet-Hodge	0.081	492482
+ CycleNet	**0.077**	492482
+ CycleNet-PEOI	0.082	497170
CWN	0.079	2435785

settings; and (3) the comparison between the original Hodge Laplacian and the cycle space of the Hodge Laplacian.

To conduct the evaluation, we follow the settings of [14] and report the results in Table 1. Specifically, we name the framework **CycleNet-Hodge**, which replaces the orthogonal projector of the cycle space of the Hodge Laplacian with the original Hodge Laplacian. Notably, we follow the implementation of IGN in [14], which restricts the tensor dimensions for efficiency, leading to a slight theoretical limitation but strong empirical performance.

We find from the table that the proposed cycle-related information improves the performance of all backbones while only adding a few extra learnable parameters. This provides empirical evidence that the proposed structural embedding is robust across different backbone models. Additionally, CycleNet outperforms CycleNet-Hodge across all backbones, indicating that the basis-invariant encoding of the cycle space is better at extracting useful cycle-related information. We also observe that BasisNet introduces too many additional parameters and performs worse than our model, demonstrating the trade-off between computational efficiency and theoretical representation power when generating a basis-invariant encoding for all eigenspaces. Furthermore, comparing CWN to CycleNet, CWN achieves comparable results with CycleNet and CycleNet-PEOI, indicating its strong representation power. However, CWN introduces too many trainable parameters, leading to high memory and computational costs.

References

[1] Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

[2] Muhammet Balcilar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pages 599–608. PMLR, 2021.

[3] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural*

Information Processing Systems, 34:2625–2640, 2021.

[4] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[5] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using poincaré and lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

[6] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[7] Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.

[8] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

[9] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*.

[10] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Alexandros D Keros, Vidit Nanda, and Kartic Subr. Dist2cycle: A simplicial neural network for homology localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7133–7142, 2022.

[12] P. Langley. Crafting papers on machine learning. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

[13] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[14] Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[15] Takanori Maehara and Hoang NT. A simple proof of the universality of invariant/equivariant graph neural networks. *arXiv preprint arXiv:1910.03802*, 2019.

[16] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

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

[18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[19] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In *International Conference on Machine Learning*, pages 11659–11669. PMLR, 2021.

[20] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, Yusu Wang, and Chao Chen. Neural approximation of graph topological features. In *Advances in Neural Information Processing Systems*, 2022.

[21] Zuoyu Yan, Junru Zhou, Liangcai Gao, Zhi Tang, and Muhan Zhang. Efficiently counting substructures by subgraph gnns without running gnn on subgraphs. *arXiv preprint arXiv:2303.10576*, 2023.

[22] Simon Zhang, Soham Mukherjee, and Tamal K Dey. Gefl: Extended filtration learning for graph classification. In *Learning on Graphs Conference*, pages 16–1. PMLR, 2022.

[23] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 2896–2906. PMLR, 2020.