REDUCING LEARNING ON CELL COMPLEXES TO GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Message passing graph neural networks (GNNs) are known to have a limited expressiveness in distinguishing graphs. A recent approach tackles this problem by transforming graphs to regular cell complexes. This makes it possible to model higher-order structures and yields algorithms that are more powerful than the Weisfeiler Lehman test (WL) or GNNs. However, this approach cannot easily be combined with previous graph algorithms and implementations due to their fundamental differences. We develop *Cell Encoding*, a novel approach of transforming regular cell complexes to graphs. We show that cell encoding combined with WL or a suitably expressive GNN is at least as expressive as Cellular Weisfeiler Lehman (CWL) in distinguishing cell complexes. This means that with a simple preprocessing one can use any GNN for learning tasks on cell complexes. Additionally, we show that this approach can make GNNs more expressive and give better results on graph classification datasets.

1 INTRODUCTION

Xu et al. (2019) and Morris et al. (2019) showed that message passing graph neural networks (GNNs) have limited expressiveness in distinguishing graphs. They showed that any GNN can only distinguish two graphs if the Weisfeiler Lehman test (WL) can distinguish them. This motivated the research into provably more expressive GNNs. One approach for this is to extend the vertex features. This can be done by adding random features (Dasoulas et al., 2020; Abboud et al., 2021; Sato et al., 2021), subgraph counts (Bouritsas et al., 2020), or rooted subgraph homomorphism counts (Barceló et al., 2021). Other methods change the way message passing is performed, such as higher order GNNs (Morris et al., 2019), equivariant subgraph aggregation networks (Bevilacqua et al., 2021), structural message-passing neural networks (Vignac et al., 2020), and CW Networks (Bodnar et al., 2021a). The latter method is especially interesting as it changes the message passing scheme by transforming a graph to a topological construct called a regular cell complex. They define Cellular Weisfeiler Lehman (CWL) a variant of WL that works on regular cell complexes, and an equivalent of graph neural networks on regular cell complexes called CW Networks. Depending on the transformation from graph to cell complex, these methods can have higher expressiveness and better connectivity between nodes than WL or GNNs.

In this work, we present *Cell Encoding* an algorithm that can transform any regular cell complex to a graph. We prove that this transformation combined with WL or a suitably expressive GNN is at least as expressive as CWL and CW Networks in distinguishing regular cell complexes. This shows that it is possible to perform message passing on graphs instead of the corresponding cell complexes. We also use this approach to encode structural features as additional nodes, which corresponds to lifting the graph to a cell complex and then transforming it back. Message passing in this modified graph corresponds to message passing on the cell complex and can improve the expressiveness of GNNs. We show empirically that this approach improves the results of GNNs on graph classification tasks.

2 CELLULAR WEISFEILER LEHMAN

In this section, we introduce the concept of *expressiveness* and the necessary concepts of Bodnar et al. (2021a) to understand message passing on regular cell complexes. For details we defer to to Appendix A or Bodnar et al. (2021a). We say that algorithm A is at least as *expressive* as algorithm

B if A can distinguish every pair of graphs or cell complexes that B can distinguish. A is *more expressive* than B if A is at least as expressive as B and can distinguish more pairs of graphs or cell complexes than B.

Bodnar et al. (2021a) generalized the message passing paradigm from graphs to regular cell complexes. Regular cell complexes generalize the simplical complexes used by Bodnar et al. (2021b). A regular cell complex X is a topological space consisting of subspaces $\{X_\sigma\}_{\sigma \in P_X}$ called cells together with an indexing set P_X. This indexing set encodes all topological information about X and can be used to define a boundary relation $\prec$ between cells. This boundary relation can then be leveraged to define adjacencies between cells. Cellular Weisfeiler Lehman (CWL) performs message passing on cells. In each iteration of CWL the algorithm computes a colouring for each cell depending on the colours of neighbouring cells in the previous iteration. Similar to WL two regular cell complexes are not isomorphic if at some iteration the colour histograms of all cells are different for the two complexes.

To apply the concept of regular cell complexes to graphs, Bodnar et al. (2021a) define the concept of a *cellular lifting map*, a function f that transforms a graph to a regular cell complex such that two graphs G_1, G_2 are isomorphic if and only if $f(G_1), f(G_2)$ are isomorphic. They prove that a class of lifting maps called *skeleton preserving lifting maps* together with CWL are at least as expressive as WL. Typically, such lifting maps create cells out of vertices, together with cells that encode other structures such as induced cycles or cliques. They define CW Networks which combine neural networks with cellular message passing, similar to graph neural networks with message passing. CW Networks can be made equally expressive as CWL. Thus, by lifting graphs to cell complexes and then using a CW Network one can obtain algorithms that are strictly more expressive than WL.

3 Cell Encoding

We propose *Cell Encoding*, a novel algorithm that transforms a regular cell complex X to a graph G_X. We show that cell encoding combined with WL is at least as expressive as CWL in distinguishing regular cell complexes. With this, we can perform message passing on graphs instead of cell complexes while keeping the expressiveness guarantees from CWL. However, this approach is not limited to cell complexes obtained with a cellular lifting map. Indeed, any cell complex can be transformed into a graph while ensuring that WL is as expressive least as CWL.

Definition 1 (Cell Encoding)**.** *Given a regular cell complex X with a finite indexing set P_X, cell encoding transforms P_X into a graph $G_X = (V_X, G_X)$ with vertex features. Where*

$$V_X = P_X,$$

$$E_X = \{\{\tau, \delta\} \mid \tau, \delta \in P_X,\ \tau \prec \delta \text{ or } \delta \prec \tau\} \cup \{\{\tau, \delta\} \mid \exists \sigma \in P_X,\ \tau \prec \sigma,\ \delta \prec \sigma\},$$

and the features of a vertex σ encode the dimension of cell σ.

Encoding the dimension of a cell in vertex features can be done via one-hot encoding and we use it to distinguish between cells of different dimensions.

Theorem 2. *Cell encoding together with WL is as least as expressive as CWL.*

Proof Sketch. (Full proof in Appendix B). We prove that every pair of vertices assigned the same colour by WL imply that the underlying cells will be assigned the same colour by CWL. We show this by induction on the iterations of CWL. The base case directly follows from the fact that graphs obtained by applying CRE to a regular cell complex have the same number of vertices as the underlying cell complex has cells. In the induction step, the properties of a stable WL colouring together with the vertex features encoding the dimension of cells means we can distinguish between vertices that correspond to cells of different dimensions. This allows us to show that if cell encoding together with WL cannot distinguish a pair of regular cell complexes then neither can CWL. □

4 Cellular Ring Encoding

In this section, we show that cell encoding can be used to build more expressive GNNs. Bodnar et al. (2021a) present cellular lifting maps such as k-IC that when combined with CWL yield algorithms

strictly more expressive than WL. We focus on the lifting map k-IC that transforms every vertex, edge and induced cycle of length up to k into a cell. Note that $k \geq 3$ is a hyperparamter that needs to be set separately. Combining k-IC with cell encoding gives us a transformation we call *Cellular Ring Encoding* (CRE). CRE transforms a graph into another graph with vertex features. An example of CRE can be seen in Figure 1.

Proposition 3. *CRE together with WL is more expressive than WL.*

Proof. k-IC has been shown to be strictly more expressive than WL when combined with CWL (Bodnar et al., 2021a). By Theorem 2 it follows that combining CRE with WL is strictly more expressive than just WL. □

Furthermore, since the graph neural network GIN (Xu et al., 2019) can be made as expressive as WL it follows that combining CRE with GIN is more expressive than WL.

5 Experiments

In this section, we investigate the performance of Cellular Ring Encoding on graph classification datasets. The experiments[1] are designed to determine whether Cellular Ring Encoding is on par with the model CIN introduced in Bodnar et al. (2021a) and current GNNs. Furthermore, we also investigate whether Cellular Ring Encoding generally improves the empirical performance of graph classification methods.

Similar to CIN (Bodnar et al., 2021a), we use GIN (Xu et al., 2019) with Jumping Knowledge (Xu et al., 2018) as our model. CIN performs message passing on cell complexes constructed with the cellular lifting map k-IC. Analogously, we use Cellular Ring Encoding. The size k of the largest induced cycle to lift will be tuned to the given datasets. For details about the models we defer to Appendix C.1 and for details about hyperparameters we defer to Appendix C.4.

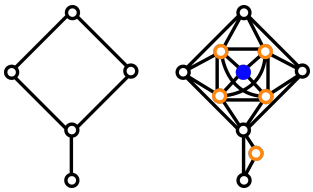

Figure 1: A graph (left) and the result of applying Cellular Ring Encoding on it (right). Orange vertices correspond to cells that were originally edges and the blue filled vertex to a cells that was a cycle.

TUDataset. We perform two sets of experiments on datasets from the TUDataset collection (Morris et al., 2020), the first follows Xu et al. (2019) and the second follows Errica et al. (2020). In the first set of experiments we evaluate GIN + CRE on `NCI1`, `NCI109`, and `PROTEINS`. These datasets consist of 1000 - 4000 graphs belonging to one of two classes. These datasets were chosen as they are the largest commonly used molecular datasets in the collection. We perform stratified 10-fold cross-validation and report the average and standard deviation of the epoch with the highest validation accuracy. We use Bayesian optimization to quickly find suitable parameters within less than 20 parameter combinations. Following Xu et al. (2019) we report the result of the parameter configuration with the best validation accuracy. The results can be found in Table 1. As expected, CRE improves the accuracy of GIN. Interestingly, GIN + CRE achieves a higher accuracy than CIN on all three datasets.

This evaluation method can overestimate the performance of models. To get a more realistic understanding of the performance of Cellular Ring Encoding we do an ablation study by adapting an experiment setup introduced by Errica et al. (2020). For this we perform stratified 10-fold cross validation on the `NCI1` dataset and ensure that hyperparameters are only selected on the training set. More details can be found in Appendix C.2. We investigate three neural networks: GIN, GIN with Cellular Ring Encoding and CIN. We also investigate the Weisfeiler-Lehman Shortest Path (WL SP) and Subtree Kernels (WL ST) using an SVM as the learning algorithm, with and without CRE.

The results can be found in Table 2. Combining GIN with with CRE improves the accuracy over GIN. Interestingly, CRE does not improve WL ST but improves WL SP when restricting it to a

[1]Code can be found at `https://anonymous.4open.science/r/ICLR22_Workshop_Code-2F50`

Table 1: Accuracy on TUDataset graph classification tasks. Citations signify the source of the result. N/A means that the authors did not evaluate their algorithm on the given dataset.

Method	PROTEINS	NCI1	NCI109
WL Kernel (Shervashidze et al., 2011)	75.0 ± 3.1	84.6 ± 1.8	N/A
GIN (Xu et al., 2019)	76.2 ± 2.8	82.7 ± 1.7	N/A
PPGNs (Maron et al., 2019)	77.2 ± 4.7	83.2 ± 1.1	82.2 ± 1.4
GSN (Bouritsas et al., 2020)	76.6 ± 5.0	83.5 ± 2.0	N/A
CIN (Bodnar et al., 2021a)	77.0 ± 4.3	83.6 ± 1.4	84.0 ± 1.6
GIN + CRE	77.5 ± 3.9	84.0 ± 1.3	84.3 ± 1.5

Table 2: Ablation on `NCI1`

Method	Accuracy
WL SP (1 iter)	76.6 ± 2.8
WL SP (2 iter)	78.9 ± 2.3
WL SP (1 iter) + CRE	78.8 ± 2.5
WL SP (2 iter) + CRE	Out of RAM
WL ST	82.3 ± 1.5
WL ST + CRE	82.5 ± 1.5
GIN	81.5 ± 2.2
CIN	81.4 ± 2.2
GIN + CRE	82.4 ± 1.8

Table 3: ROC-AUC on `ogb-molhiv`. Citations signify the source of the result.

Method	ROC-AUC
GIN + VN (Hu et al., 2020)	77.07 ± 1.49
GSN + GIN + VN (Bouritsas et al., 2020)	77.99 ± 1.00
GSN + DGN (Bouritsas et al., 2020)	80.39 ± 0.90
CIN (Bodnar et al., 2021a)	80.94 ± 0.57
GIN + VN + CRE	78.98 ± 1.53

single iteration of the WL algorithm. It is surprising that in this setting CIN performs similarly to GIN even though we would expect it to outperform GIN and achieve similar results as GIN + CRE.

ogb-molhiv. We evaluate GIN + CRE on the `ogb-molhiv` dataset (Hu et al., 2020). Results are evaluated with the ROC-AUC score, according to Hu et al. (2020). `ogb-molhiv` provides a train, validation and test split, allowing fair comparisons between different methods. Similar to Hu et al. (2020) we extend our setup with a virtual node (VN), that is, a node that is connected to all nodes in the graph. We train GIN + VN + CRE to see how much we can improve upon GIN + VN. We tune the hypereparameters via Bayesian optimization on the validation set, and train a model with the best parameters 10 times without setting a random seed. We report the mean and standard deviation of the test ROC-AUC score in the epoch with the highest validation score in Table 3. Cell encoding substantially improves over just GIN + VN, but does not manage to beat CIN.

6 Conclusion

In this paper, we have shown that transforming any regular cell complex into a graph and applying WL is at least as expressive as CWL on the cell complex. We can adapt any GNN to operate on regular cell complexes with a single line of code. Similarly, a simple preprocessing can make any GNN with WL expressiveness (such as GIN) at least as expressive as cellular message passing. We have demonstrated empirically that this approach can improve the performance on graph classification datasets. A downside of the original cellular message passing and also our cell encoding is the potentially large number of cells in a lifted graph. The reason is that cellular lifting maps create at least a cell for every vertex and edge in the graph. This can lead to increased runtimes on larger graphs. For a computational analysis of computing the cell complex we refer to Bodnar et al. (2021a).

We conclude by proposing a generalization of our ideas as future work. The idea behind cell encoding is creating vertices corresponding to structures in graphs or cells together with a feature that gives more information about this structure. We have already shown that this approach can be used to perform cellular message passing on graphs instead of regular cell complexes. A similar approach could be used to encode subgraph (Bouritsas et al., 2020) or homomorphism patterns (Barceló et al., 2021) as additional vertices. Finally, it is seems possible to emulate k-dimensional WL (Immerman & Lander, 1990) using 1-WL on a transformed graph.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *NeurIPS*. 2021.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR*. 2021.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and Lehman go cellular: CW networks. In *NeurIPS*. 2021a.

Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Liò, and Michael Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *ICML*, 2021b.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252, earlier version appeared in Graph Representation Learning and Beyond (GRL+) Workshop at ICML*, 2020.

George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. In *IJCAI*, 2020.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3:315–358, 2019.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In Alan L. Selman (ed.), *Complexity Theory Retrospective*, pp. 59–81. Springer New York, 1990.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William Hamilton, Jan Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *SDM*, 2021.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman graph kernels. *JMLR*, 12:2539–2561, 2011.

Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. GraKeL: A graph kernel library in python. *JMLR*, 21(54):1–5, 2020.

Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with Jumping Knowledge networks. In *ICML*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

A More Details About Cellular Weisfeiler Lehman

In this section we give an explanation of cellular lifting maps and the Cellular Weisfeiler Lehman algorithm. Note that all definitions are take from the cited sources with some additional details. We use $\{\!\{\cdot\}\!\}$ to denote a multiset.

Definition 4 (Hansen & Ghrist, 2019, Bodnar et al., 2021a)**.** *A* regular cell complex *is a topological space X together with an indexing set P_X that defines a partition $\{X_\sigma\}_{\sigma \in P_X}$ of subspaces X_σ of X called* cells*, such that*

1. *For each $x \in X$, every sufficiently small neighborhood of x intersects finitely many cells.*
2. *For all $\sigma, \tau \in P_X$ we have that $X_\tau \cap \overline{X_\sigma} \neq \emptyset$ only if $X_\tau \subseteq \overline{X_\sigma}$, where $\overline{X_\sigma}$ is the closure of a cell.*
3. *Every cell X_σ is homeomorphic to $\mathbb{R}^{n_\sigma}$ for some $n_\sigma \in \mathbb{N}$.*
4. *(*Regularity*) For every $\sigma \in P_X$ there is a homeomorphism ϕ of a closed ball in $\mathbb{R}^{n_\sigma}$ to $\overline{X_\sigma}$ such that the restriction of ϕ to the interior of the ball is a homeomorphism onto X_σ.*

For the purpose of this paper we will assume that P_X is finite. For every cell X_σ we call n_σ the dimension of the cell. This definition implies that the indexing set P_X has poset structure defined by $\tau \leq \sigma$ iff $X_\tau \subseteq \overline{X_\sigma}$ that encodes all topological information about X. From this Bodnar et al. (2021a) define a boundary relation:

Definition 5 (Bodnar et al., 2021a)**.** *The **boundary relation** $\sigma \prec \tau$ holds iff $\sigma < \tau$ and there is no cell δ such that $\sigma < \delta < \tau$.*

This boundary relation can then be leveraged to define adjacencies in regular cell complexes. Note, that we simplify the definitions of Bodnar et al. (2021a) by applying their Theorem 7 to remove adjacencies that do not improve the expressiveness of CWL.

Definition 6 (Bodnar et al., 2021a)**.** *For a regular cell complex X and a cell $\sigma \in P_X$, we define:*

1. *The boundary adjacent cells $\mathcal{B}(\sigma) = \{\tau \mid \tau \prec \sigma\}$. These are the lower-dimensional cells on the boundary of σ. For instance, the boundary cells of an edge are its vertices.*
2. *The co-boundary adjacent cell $\mathcal{C}(\sigma) = \{\tau \mid \sigma \prec \tau\}$. These are the higher-dimensional cells with σ on their boundary. For instance, the co-boundary cells of a vertex are the edges it is part of.*
3. *The upper adjacent cells $\mathcal{N}_\uparrow(\sigma) = \{\tau \mid \exists \delta$ such that $\sigma \prec \delta$ and $\tau \prec \delta\}$. These are the cells of the same dimension as σ that are on the boundary of the same higher-dimensional cell as σ. The typical graph adjacencies between vertices are the canonical example here.*

Definition 7 (Bodnar et al., 2021a)**.** *For any cells $\sigma, \tau \in P_X$ we define $\mathcal{C}(\sigma, \tau) = \mathcal{C}(\sigma) \cap C(\tau)$.*

From this one can define how the adjacencies influence the colouring of a cell.

Definition 8 (Bodnar et al., 2021a)**.** *A* cellular colouring *is a map c that maps every cell of a regular cell complex to a colour from a fixed colour palette.*

Definition 9 (Bodnar et al., 2021a)**.** *Let c be a celulular colouring of X with c_σ denoting the colour assigned to cell $\sigma \in P_X$. We define the following multi-sets of colours:*

1. *The colours of the boundary cells of σ : $c_{\mathcal{B}}(\sigma) = \{\!\{c_\tau \mid \tau \in \mathcal{B}(\sigma)\}\!\}$.*
2. *The upper adjacent colours of σ : $c_\uparrow(\sigma) = \{\!\{(c_\tau, c_\delta) \mid \tau \in \mathcal{N}_\uparrow(\sigma)$ and $\delta \in \mathcal{C}(\sigma, \tau)\}\!\}$.*

Finally, we can define CWL a colour refinement scheme for regular cell comple analogously to Bodnar et al. (2021a).

1. Given a regular cell complex X, all cells are initialised with the same colour.
2. Given the colour c_σ^t of cell σ at iteration t, we compute the colour of cell σ at the next iteration c_σ^{t+1} by injectively mapping the multi-sets of colours belonging to the adjacent cells of σ using a perfect HASH function: $c_\sigma^{t+1} = \text{HASH}\left(c_\sigma^t, c_{\mathcal{B}}^t(\sigma), c_\uparrow^t(\sigma)\right)$.

3. The algorithm stops when a stable colouring is reached. Two cell complexes are considered non-isomorphic if their colour histograms are different. Otherwise, the test is inconclusive.

To apply this algorithm to graphs one needs to lift a graph to a regular cell complex. To ensure that the expressiveness of CWL is comparable with that of WL such a lifting operation needs to respect isomorphisms.

Definition 10 (Bodnar et al., 2021a)**.** *A* cellular lifting map *is a function $f : \mathcal{G} \to \mathcal{X}$ from the space of graphs $\mathcal{G}$ to the space of regular cell complexes $\mathcal{X}$ with the property that two graphs G_1, G_2 are isomorphic iff the cell complexes $f(G_1), f(G_2)$ are isomorphic.*

Finally, Bodnar et al. show that a class of cellular lifting maps called *skeleton preserving lifting maps* together with CWL are at least as expressive as WL. For this one first lifts the graph to a regular cell complex with a skeleton preserving lifting map and then performs CWL on it. Then graphs can be distinguished via their cellular colouring. There also exist skeleton preserving lifting maps that when combined with CWL are strictly more expressive than WL.

B Proof of Theorem 2

We give a proof of Theorem 2. We use $\{\!\{\cdot\}\!\}$ to denote a multiset and $\mathcal{N}_G(v)$ to denote the neighbors of vertex v in graph G.

Proof. Let P_X, P_Y be the indexing sets of two regular cell complexes. Let G_X and H_Y be the graphs obtained by applying cell encoding to P_X and P_Y. We use π to denote the stable colouring obtained by WL and c^t to denote the colour obtained by CWL after iteration t. Thus π_σ denotes the colours assigned to *vertex* σ by WL and c^t_σ denotes the colour assigned to *cell* σ by CWL. We assume that WL with cell encoding cannot distinguish G_X and H_Y. From this we show that for any iteration $t \geq 0$ of CWL it holds that:

For all $\tau \in V(G_X), \sigma \in V(H_Y)$ with $\pi_\tau = \pi_\sigma$ it holds that $c^t_\tau = c^t_\sigma$.

Note that this is equivalent to showing that if WL with cell encoding cannot distinguish G_X and H_Y then CWL cannot distinguish P_X and P_Y. This is because when WL with cell encoding cannot distinguish G_X and H_Y, then we know that for every vertex in G_X there is a vertex in H_Y that are assigned the same color by WL. The statement then implies that there is a bijective mapping from cells of P_X to P_Y such that they are assigned the same colour by CWL which means that the histogram of colours is the same for both graphs. We show that this statement holds by induction on the iteration t of CWL.

During the proof we will make use of the fact that π is a stable colouring. This means that if two vertices $p \in V(G_X)$ and $q \in V(H_Y)$ are assigned the same colour $\pi_p = \pi_q$, then if WL is run for another iteration they will still have the same colour. This implies that the multiset of colours of neighbors of p is equivalent to the multiset of colours of neighbors of q. Thus, there exists a bijective function $\alpha : \mathcal{N}_{G_X}(p) \to \mathcal{N}_{H_Y}(q)$ such that for any $x \in \mathcal{N}_{G_X}(p)$ it holds that $\pi_x = \pi_{\alpha(x)}$.

Base case: We show that the statements hold for $t = 0$. CWL initializes all of its cells to the same colour. Thus, all we need to show is that P_X and P_Y have the same number of cells. The number of vertices in G_X and H_Y is equal to the number of cells in P_X and P_Y, respectively. Since WL cannot distinguish G_X and H_Y we know that they must have the same number of vertices.

Induction hypothesis: We assume that the statements holds for $t = n$.

Induction step: We show that the statements hold for $t = n + 1$. Let $\tau \in V(G_X), \sigma \in V(H_Y)$ be arbitrary vertices with $\pi_\tau = \pi_\sigma$. We need to show that $c^{n+1}_\tau = c^{n+1}_\sigma$. By the definition of CWL we know that $c^{n+1}_\tau = \text{HASH}\left(c^n_\tau, c^n_{\mathcal{B}}(\tau), c^n_{\uparrow}(\tau)\right)$ and $c^{n+1}_\sigma = \text{HASH}\left(c^n_\sigma, c^n_{\mathcal{B}}(\sigma), c^n_{\uparrow}(\sigma)\right)$. We will show that the inputs into the two hash functions are equal for c^{n+1}_τ and c^{n+1}_σ.

First, we show that $c^n_\tau = c^n_\sigma$. This immediately follows from the assumption $\pi_\tau = \pi_\sigma$ and the induction hypothesis.

Next, we want to show that $c^n_{\mathcal{B}}(\tau) = c^n_{\mathcal{B}}(\sigma)$. The assumption $\pi_\tau = \pi_\sigma$ implies that there exists a bijective function $\alpha : \mathcal{N}_{G_X}(\tau) \to \mathcal{N}_{H_Y}(\sigma)$ such that for any $x \in N_{G_X}(\tau)$ it holds that $\pi_x = \pi_{\alpha(x)}$.

Since the initial colours encode the dimensions of the cell, this function mus also respect the dimension of the cell meaning it only maps vertices to vertices whose cells have the same dimensions. This implies that for every cell μ with $\mu \prec \tau$ we know that there exists a cell $\nu = \alpha(\mu)$ with $\nu \prec \sigma$ such that $\pi_\mu = \pi_\nu$. With the induction hypothesis it follows that $c^n_\mu = c^n_\nu$. Observe, that $\mathcal{B}(\tau)$ contains only cells μ with $\mu \prec \tau$. Analogously, $\mathcal{B}(\sigma)$ contains cells ν with $\nu \prec \sigma$. Thus $c^n_{\mathcal{B}}(\tau) = c^n_{\mathcal{B}}(\sigma)$.

Finally, we need to show that $c^n_\uparrow(\tau) = c^n_\uparrow(\sigma)$. By definition we know that $c^n_\uparrow(\tau) = \{\!\{(c^n_\mu, c^n_\delta) \mid \mu \in \mathcal{N}_\uparrow(\tau) \text{ and } \delta \in \mathcal{C}(\tau, \mu)\}\!\}$. We can rewrite this as $c^n_\uparrow(\tau) = \{\!\{(c^n_\mu, c^n_\delta) \mid \tau \prec \delta \text{ and } \mu \prec \delta\}\!\}$. We will make use of the bijective function α defined in the paragraph above. We know that for any cell δ with $\tau \prec \delta$ there exists a vertex δ adjacent to τ such that $\pi_\delta = \pi_{\alpha(\delta)}$. This implies two things: first by using the induction hypothesis we know that $c^n_\delta = c^n_{\alpha(\delta)}$. Secondly, there exist a bijective function $\beta_\delta : \mathcal{N}_{G_X}(\delta) \to \mathcal{N}_{H_Y}(\alpha(\delta))$ that has the same properties as α. That is, for any $x \in \mathcal{N}_{G_X}(\delta)$ it holds that $\pi_x = \pi_{\beta_\delta(x)}$.

We can now put all of this together. The existence of α means that for any cell δ with $\tau \prec \delta$ there is a cell $\alpha(\delta) \in P_Y$ such that $c^n_\delta = c^n_{\alpha(\delta)}$. Next, with the existence of β_δ it follows that for each cell μ with $\mu \prec \delta$ there exists a cell $\beta_\delta(\mu) \in P_Y$ such that $c^n_\mu = c^n_{\beta_\delta(\mu)}$. With the fact that α and $\beta_\delta(\mu)$ are bijective, it follows that $c^n_\uparrow(\tau) = c^n_\uparrow(\sigma)$. This proves the induction step and concludes the proof of Theorem 2. □

C Experimental Details

The neural network models are implemented in Python with PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). For kernel methods we used the GraKeL library (Siglidis et al., 2020). The code to compute Cellular Ring Encoding is based on Bodnar et al. (2021a) and uses graph-tool[2] to compute induced cycles in the graphs. We used the sweep feature from Weights & Biases[3] to perform Bayesian optimization to tune the hyperparameters in all but the ablation experiments.

All models except CIN were trained on systems with an NVIDIA GeForce RTX 3080 GPU, 64 GB of RAM and an Intel i7-10700KF CPU or an Intel i9-11900KF CPU. CIN was trained on a system with an NVIDIA GeForce GTX TITAN X GPU, 94 GB of RAM and an Intel Xeon X5680 CPU.

C.1 Model Implementations

Similar to Bodnar et al. (2021a) we implement a graph readout operation that makes better use of CRE. Instead of summing up the representation of each node, we keep track whether a node came from a node, an edge or a cycle in the original graph. Then we separately sum the representations for these three types of nodes and apply a multilayer perceptron with a non linear activation function to each of the three resulting representations. Finally, we sum up all three representations to obtain a single vector that encodes an entire graph.

For experiments on the TUDatasets we use the implementation of GIN with Jumping Knowledge from Bodnar et al. (2021a) that is mostly equivalent to the benchmark implentation from PyTorch Geometric. Additionally, we add dimensional pooling to this model.

For experiments on `ogb-molhiv` we take the experimental setup from Hu et al. (2020) including their implementation of GIN with Jumping Knowledge and a virtual node. We extend this setup by adding Cellular Ring Encoding and extending the models with dimensional pooling.

C.2 Details on the Ablation Setup

We perform stratified 10-fold cross validation on the `NCI1` dataset to obtain tuples of training and test sets. For the GNN methods, we split off 10% of the trainig set as a validation set, while ensuring an equal class distribution. For each fold we tune the parameters on the training and validation set. In total we test 20 randomly selected parameters per fold. Then we train a model with the selected

[2] `https://graph-tool.skewed.de/`

[3] `https://wandb.ai/`

Table 4: Hyperparameters of GIN based methods

Hyperparameters	Gin+CRE PROTEINS, NCI1, NCI109	Gin+CRE Ablation NCI1	Gin Ablation NCI1	Gin+CR ogb-molhiv
Epochs	350	350	350	100
Batch size	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128	32, 64, 128
Learning rate	1e-3, 1e-4, 5e-4, 1e-5	1e-3, 1e-4, 5e-4, 1e-5	1e-3, 1e-4, 5e-4, 1e-5	1e-2, 1e-3, 1-e4, 5e-4, 1e-5
Drop out rate	0, 0.1, 0.2, 0.3, 0.4, 0.5	0, 0.1, 0.2, 0.3, 0.4, 0.5	0, 0.1, 0.2, 0.3, 0.4, 0.5	0, 0.5
Number of layers	2,3,4,5	2,3,4,5	2,3,4,5	2, 3, 4, 5
LR decay steps	5, 10, 20, 30, 40, 50	5, 10, 20, 30, 40, 50	5, 10, 20, 30, 40, 50	50
LR decay rate	0.25, 0.5, 0.9, 0.99	0.25, 0.5, 0.9, 0.99	0.25, 0.5, 0.9, 0.99	0.5
Embedding dimension	64	32, 64	32, 64	32, 64, 128, 300, 512, 1024
Max Ring Size	6, 8, 10	6, 8, 10	N/A	6, 8, 18
Aggr. edge features	N/A	N/A	N/A	True, False
Aggr. vertex features	True, False	True, False	N/A	True, False
Explicit pattern encoding (EPE)	True, False	True, False	N/A	True, False
Edge features in vertices	N/A	N/A	N/A	True, False
Dimensional pooling	Coupled to EPE	Coupled to EPE	N/A	True, False

parameters on the training set with early stopping on the validation set and evaluate it on the test set, we do this three times for each fold to smooth out non-deterministic behaviour in the training process.

For the kernel methods we do not split off a validation set. Instead we train on the entire training set and select the parameter configuration with the highest training accuracy, in total we try up to 20 random parameter configurations per fold depending on the size of the parameter grid. We evaluate on the test set and report the average accuracy and standard deviation over all folds.

C.3 Details on Cellular Ring Encoding

Cellular Ring Encoding One-Hot encodes the dimension of cells. This means for every vertex that corresponds to a 0-dimensional cell the vector $(1, 0, 0)$ will be added to the vertex features. For 1-dimensional and 2-dimensional the vectors $(0, 1, 0)$ and $(0, 0, 1)$ will be added to the vertex features, respectively. We call this *explicit pattern encoding*. While this is necessary for the expressiveness guarantees of Theorem 2 it can also be turned of.

Additionally, we implement the option to collect vertex features into newly created vertices by Cellular Ring Encoding. Intuitively, if CRE adds a new vertex that corresponds to an edge $\{p, q\}$ in the graph than the newly created vertex will have the average of p and q. This feature is called *aggr. vertex features* in the following section. Finally, we developed two different ways for CRE to interact with edge features. *Edge features in vertices* appends the average edge feature of a vertex to its features. This allows GNNs that normally cannot use edge features to use them. *Aggregate edge features* sets newly created edges by CRE to have the average edge features of the edge that created that edge. For example, if $\{p, q\}$ is an edge then CRE will create a vertex that corresponds to this edge. This new vertex will be adjacent to both p and q. Then the *aggregate edge features* will ensure that these newly created edges have the same features as $\{p, q\}$.

C.4 Hyperparameter

We present all used hyperparameter configurations in Tables 4, 5 and 6. The top part of the table contains model specific hyperparamters and the bottom part contains CRE specific hyperparameters. More information about the CRE specific hyperparamters can be found in Appendix C.3. As the used TUDatasets do not provide edge features the parameters "aggr. edge features" and "edge features in vertices" are irrelevant to those experiments. Additionally, for GIN + CRE on TUDatasets we couple the use of dimensional pooling to the "Explicit pattern encoding" parameter meaning dimensional pooling will be used if "Explicit pattern encoding" is set to true. For GIN + CRE on `ogb-molhiv` dimensional pooling is a separately tuneable parameter.

Table 5: Hyperparameters of kernel methods in the ablation on NCI1

Hyperparameters	WL SP (1 iter)	WL SP (2 iter)	WL SP + CRE (1 iter)	WL SP + CRE (2 iter)	WL ST + CRE	WL ST + CRE
WL Iterations	1	1, 2	1	1, 2	1, 2, 3, 4, 5, 10	1, 2, 3, 4, 5, 10
Max Ring Size	N/A	N/A	6, 8, 10	6, 8, 10	N/A	6, 8, 10
Aggr. vertex features	N/A	N/A	True, False	True, False	N/A	True, False
Explicit pattern encoding	N/A	N/A	True, False	True, False	N/A	True, False

Table 6: Hyperparameters of CIN in the ablation on NCI1

Hyperparameters	CIN
Epochs	150
Batch size	32, 128
Drop position	lin2, final readout, lin1
Drop rate	0.0, 0.5
Embedding dim.	16, 32, 64
Init method	sum, mean
Learning rate	5e-4, 1e-3, 3e-3, 1e-2
LR decay rate	0.5, 0.9
LR decay steps	50, 20
Use coboundaries	True, False
Number of layers	3 ,4