Cayley Graph Propagation

JJ Wilson
Independent Researcher
josephjwilson74@gmail.com

Maya Bechler-Speicher
Tel-Aviv University
mayab4@mail.tau.ac.il

Petar Veličković
Google DeepMind
petarv@google.com

Abstract

In spite of the plethora of success stories with graph neural networks (GNNs) on modelling graph-structured data, they are notoriously vulnerable to over-squashing, whereby tasks necessitate the mixing of information between distance pairs of nodes. To address this problem, prior work suggests rewiring the graph structure to improve information flow. Alternatively, a significant body of research has dedicated itself to discovering and precomputing bottleneck-free graph structures to ameliorate over-squashing. One well regarded family of bottleneck-free graphs within the mathematical community are *expander graphs*, with prior work—Expander Graph Propagation (EGP)—proposing the use of a well-known expander graph family—the Cayley graphs of the $\mathrm{SL}(2, \mathbb{Z}_n)$ special linear group—as a computational template for GNNs. However, in EGP the computational graphs used are truncated to align with a given input graph. In this work, we show that truncation is detrimental to the coveted expansion properties. Instead, we propose CGP, a method to propagate information over a complete Cayley graph structure, thereby ensuring it is bottleneck-free to better alleviate over-squashing. Our empirical evidence across several real-world datasets not only shows that CGP recovers significant improvements as compared to EGP, but it is also akin to or outperforms computationally complex graph rewiring techniques.

Graph neural networks (GNNs) have emerged as a cornerstone for processing graph-structured data [1] with significant contributions in various domains and real-world applications [2, 3]. The majority of GNNs are dependent on propagating information between neighbouring nodes in the graph [4], known as Message Passing Neural Networks (MPNNs) [5]. This *message-passing* paradigm involves the iterative exchange of messages, with nodes aggregating and updating their representations based on received information from neighbours. Thus, a sufficient number of layers are required to be able to capture long-range interactions. However, increasing the number of layers results in an exponential growth of the receptive field and subsequently leading to a large volume of messages to be aggregated into fixed-sized vectors. This phenomenon is known as *over-squashing* [6] and hinders the *expressivity* of GNNs [7]. The underlying graph topology has been proven to be a key contributing factor to the over-squashing problem [8].

The *over-squashing* phenomenon is an active area of research with several techniques proposed to alter the topological properties of an input graph to mitigate over-squashing. Several recent works have analysed it through varying lenses, including curvature [9], spectral expansion properties [10, 11] and effective resistance [12, 13]. Most of the solutions to this problem fall into the category of *graph rewiring*, in which the graph topology is directly modified based on an optimization target. However, this imposes the computational complexity of having to surgically analyse the graph structure. Notably, Expander Graph Propagation (EGP) [14] propose a unique solution of constructing an independent desirable graph structure to propagate information over.

To this end, Deac et al. [14] identified four desirable criteria to mitigate over-squashing and effectively handle global context in graph representation learning: *global information propagation* (i), *no bottlenecks* (ii), *subquadratic time and space complexity* (iii) and *no dedicated preprocessing* (iv). The authors surveyed prior approaches, including traditional GNNs (iii, iv), master-node methods (i,

JJ Wilson et al., Cayley Graph Propagation. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

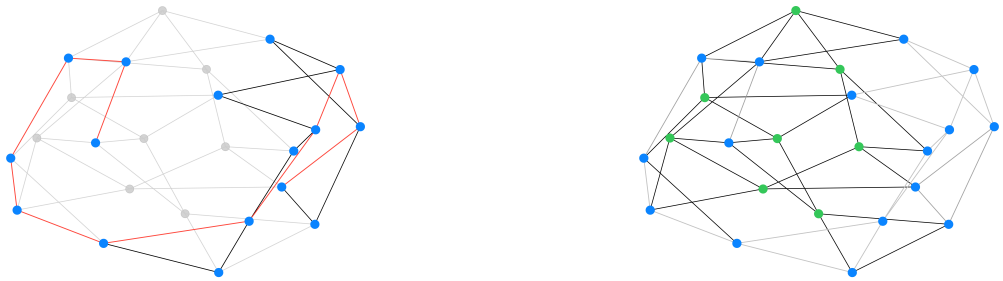


Figure 1: Both Cayley graphs represent $\mathrm{SL}(2, \mathbb{Z}_3)$ with $|V| = 24$ nodes using the same construction. **Left**: A truncated Cayley graph (*spectral gap*: 0.0751, *diameter*: 10) aligned to a given input graph. **Right**: The *complete* Cayley graph (*spectral gap*: 1.2679, *diameter*: 4) structure indicating the additional *virtual nodes* (in green).

iii, iv) [5, 15], fully connected graphs (i, ii, iv) [6] and the aforementioned graph rewiring techniques (i-iii). Ultimately, Deac et al. [14] recognised the efficacy of *expander graphs* [16, 17] as the desirable graph structure for bottleneck-free information propagation, due to their favourable topological properties.

A family of expander graphs in Deac et al. [14] has been constructed leveraging the well-known theoretical results of *special linear groups*, for which a family of corresponding Cayley graphs can be derived. This family of Cayley graphs is guaranteed to have the derisible topological properties to mitigate over-squashing, as well as being efficiently precomputable (i-iv). Although the constructed Cayley graphs are scalable, the number of nodes are in order of $O(|V|^3)$. This consideration is addressed within the paper by identifying the smallest n that yields a graph larger or equal to the desired number of nodes, and then subsequently *truncating* the Cayley graph to match the input graph's number of nodes in breadth-first order. Consequently, this truncation procedure results in a *subgraph* derived from the expander graph being used as the computational template.

Motivated by the promising research direction of Deac et al. [14] in which the authors use an independent graph structure that is theoretically known to exhibit desirable properties we propose a different approach of using the Cayley graph. We propose a more optimal approach of embracing the complete Cayley graph structure, whilst still fulfilling the desirable criteria (i-iv).

Main contributions and outline. In this paper, we present Cayley Graph Propagation (**CGP**)[1], a novel model that uses the complete Cayley graph structure to mitigate over-squashing. Our contributions are as follows:

- In Section 3, we highlight the optimal topological properties Cayley graphs for message propagation. We show that the truncation procedure performed in EGP to align the Cayley graph with the input graph may be detrimental to the coveted theoretical benefits.
- In Section 4, we introduce **CGP**, a method to propagate information over the complete Cayley graph structure, thereby ensuring it is bottleneck-free and alleviates over-squashing. CGP modifies EGP by avoiding the truncation step used to align the input graph with a Cayley graph, utilising the additional nodes as virtual nodes.
- In Section 5, we provide an empirical evaluation across several real-world datasets to show that CGP recovers significant improvements as compared to EGP. Additionally, our model is akin to or outperforms the computationally complex graph rewiring techniques.

1 Background

Graph preliminaries. Given an undirected graph denoted as $G = (V, E)$, where V and E denote its nodes and edges respectively. The topology of the graph is encoded in the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where the number of nodes $n = |V|$. Let $\mathbf{D} = \mathbf{D}(G)$ denote the diagonal matrix

[1] Our source code is available at: https://github.com/josephjwilson/cayley_graph_propagation

of degrees as given by $\mathbf{D}_{vv} = d_v$. The normalised graph Laplacian $\mathbf{L} = \mathbf{L}(G)$ is defined by $\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2}$. From the normalised Laplacian L the eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-2} \leq \lambda_{n-1}$. Importantly, from this derivation the *spectral gap* of graph G is $\lambda_1 - \lambda_0 = \lambda_1$; the *Cheeger inequality* then defines that a larger spectral gap of graph G is an indicator of good spectral expansion properties. Accordingly, a graph with desirable expansion properties (or a larger spectral gap) defines that it has strong connectivity, or alternatively it is globally lacking bottlenecks [14].

Expander graphs. An expander graph is categorised by its unique properties of being both sparse and highly connected with the number of edges scaling linearly with the number of nodes ($|E| = O(|V|)$). One such expansion property that an expander graph satisfies is derived from the aforementioned Cheeger inequality. As a result, in essence, expander graphs do not have any bottlenecks [11]; in Section 3 we further define and explore this link.

Due to the definition of an expander graph, there are consequently several known construction approaches [18, 19]. We will focus on the *deterministic* algebraic approach as introduced by Deac et al. [14]. A family of expander graphs have been precomputed leveraging the well-known theoretical results of *special linear groups*, $\mathrm{SL}(2, \mathbb{Z}_n)$, for which a family of corresponding Cayley graphs, $\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n)$, can be derived. Here, S_n ([14], Definition 8) denotes a particular generating set for $\mathrm{SL}(2, \mathbb{Z}_n)$. For appropriate choices of S_n, the corresponding Cayley graphs are guaranteed to have expansion properties. Moreover, from Figure 1 we see that the constructed graph are 4-regular, ($|E| = 2|V|$). Importantly, although Cayley graphs are scalable, achieving a specific number of nodes is not always feasible; for instance, the node count of $\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n)$ is given as per:

$$|V(\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n))| = n^3 \prod_{\text{prime } p|n} \left(1 - \frac{1}{p^2}\right). \tag{1}$$

Over-squashing. The over-squashing problem was first identified by Alon and Yahav [6], whereby the information in a MPNN is aggregated from too many neighbours, meaning as a consequence they are squashed into fixed-size vectors. This can result in a loss of information [20]. This phenomenon was then formalised [7, 9, 12], showing that the Jacobian of the node features is affected by topological properties of the graph, such as curvature and effective resistance. Furthermore, Di Giovanni et al. [8] analysed how over-squashing impacts the expressive power of GNNs. In the following section, we will address the literature and how current approaches aim to mitigate over-squashing.

2 Existing approaches to mitigate over-squashing

In this section, we explore the current landscape of several novel techniques that try to alleviate the over-squashing phenomenon [6]. In essence, the main principle behind many of these techniques is to decouple the input graph G from the computational one, such that it has structurally fewer bottlenecks. Alon and Yahav [6] simply proposed a rewiring technique that does not require the analysis of the input graph by making the last layer of the GNN fully adjacent (FA), allowing all nodes to interact with each other. The effectiveness of such an approach can be shown by (dense) Graph Transformers [21, 22], where every layer is fully-connected. However, such an approach is limited by even modest graph sizes due to it imposing $O(|V|^2)$ edges. An alternative approach is a master node [5]; here a new node is introduced, which is connected to all of the nodes within the graph. This approach is effective as it reduces the graph's diameter to 2 by only adding one new node with $O(|V|)$ edges. However, the master node itself becomes the bottleneck. Notably, both of the aforementioned approaches are independent in relation to the input graph topology, therefore satisfying (iv) of *no dedicated preprocessing*.

Graph rewiring. An alternative promising approach line of research is to *rewire* the input graph G to optimise the *spectral* or *spatial* properties of the graph. To this end, an abundance of graph rewiring techniques have stemmed to modify the graph connectivity to try and mitigate bottlenecks. A popular class of approaches are based on a *spectral* quantity of a graph [10] or to reduce the effective resistance [11–13]. These approaches have provided promising insights and have empirically reduced over-squashing, but they impose a computational complexity of having to examine the input graph structure.

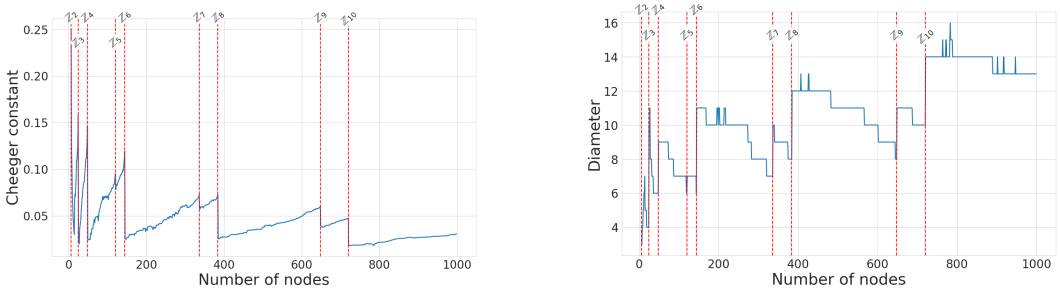


Figure 2: Illustrates the correlation between the Cheeger constant (left) and diameter (right) between extracting a truncated Cayley graphs from $\mathrm{SL}(2, \mathbb{Z}_n)$. The properties exhibit the derisible properties at each given complete Cayley graph interval (as denoted by the dotted red line) for the groups of $\mathbb{Z}_n$. The trend shows a detrimental drop in the derisible property occurring once the truncated Cayley graph aligns with the next complete Cayley graph structure.

Expander graph based rewiring. The existing approaches of quantitative analysis leads to an alternative approach being the understanding of the desirable graph structure known as an expander graph. An expander graph exhibits the desirable properties associated with *spectral gap* and *effective resistance*. For this reason, Banerjee et al. [11] proposes a construction inspired by an expander graph to randomly locally rewire a given input graph, whilst Shirzad et al. [23] use both *virtual global nodes* and expander graphs as a powerful primitive to design a more scalable graph transformer architecture. As previously examined, the work of Deac et al. [14] proposes a different schema of precomputing a bank of expander graphs, which are then interwoven by alternating layers on the input graph and then the auxiliary expander layer. This schema has proven to also be successful in high-order structures [24] and in the first rewiring approach on temporal graphs [25].

3 Benefits of Cayley graphs

In this section, we formalise the topological properties of expander graphs and why they have been used as a conduit in a number of over-squashing approaches [11, 14, 23]. In particular, we focus on the Cayley graph expander family as constructed by Deac et al. [14]. Furthermore, we importantly anaylse the impact on the topological properties of the truncated Cayley graphs as used by Deac et al. [14]. In contrast to this, in the appendix, we examine an alternative benefit to Cayley graphs being regular graphs through the lens of the recent work of Bechler-Speicher et al. [26].

3.1 Topological properties of Cayley graphs

The Cayley graph's topology serves as ideal conduit for the propagation of information due to its sparsity and being highly connected. In particular, we recall that the family of Cayley graphs derived from $\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n)$ are in the magnitude of $O(|V|^3)$, and are well-known to exhibit a high spectral gap, low diameter and optimal commute-time.

Connectivity. One metric to measure a graph's connectivity is the coveted *Cheeger constant* [27]. It provides a measurement of the narrowest bottleneck in a graph; a higher Cheeger constant indicates that a graph is globally lacking bottlenecks. Alternatively, it effectively describes how difficult it is to separate a graph G into two subgraphs by removing edges. The exact Cheeger constant $h(G)$ is known to be a computationally challenging problem. To address this difficulty, we recall the *spectral* gap of a graph G (defined in Section 1), which provides a bound for the Cheeger constant from discrete Cheeger inequality [28, 29]. As per Chung and Graham [27], this bound is:

$$\frac{\lambda_1}{2} \leq h(G) \leq \sqrt{2\lambda_1}, \tag{2}$$

where λ_1 is the second-smallest eigenvalue or spectral gap of the normalised graph Laplacian $\mathbf{L}(G)$. In Figure 2, we use the lower bound from the Cheeger constant; the figure illustrates that the complete Cayley graph structure exhibits the most desirable Cheeger constant (or higher spectral gap). In turn,

it is observed that the most unfavourable scenario occurs for a truncated Cayley graph just beyond the range of the proceeding graph, as derived from the *special linear group* $\mathrm{SL}(2, \mathbb{Z}_n)$.

Diameter. The diameter of a graph influences the effectiveness of traversal between nodes. Intuitively, a lower diameter facilitates a more efficient graph structure, enabling nodes to reach each other in a shorter number of hops. Our constructed Cayley graph with $|V|$ nodes has a low diameter, requiring only $O(log(|V(G_i)|)$ ([14], Theorem 5) steps to globally propagate information. The results in Figure 2 in relation to the diameter correlate with those shown for the Cheeger constant; the lower diameter is at each complete Cayley graph structure.

Commute time. The recent work of Di Giovanni et al. [7, 8] validates that the topology of the graph is the primary factor affecting over-squashing. In Theorem 5.5 from Di Giovanni et al. [7], it is proven that between a pair of nodes u and v over-squashing could be bounded by the commute time $T(u, v)$. The commute time is defined as the expected number of steps in a random walk from u to v and back to u. Consequently, over-squashing occurs between nodes with a large commute time. Notably, the family of Cayley graphs as derived from [14] are directly mentioned in Di Giovanni et al. [7, 8] as being the *optimal* computational template for message-passing due to the commute time scaling linearly with the number of edges. We note that the commute-time $T(u, v)$ between a pair of nodes is analogous to the *effective resistance* [12]. In Appendix (Section A), we extend our analysis of the impact of truncating a Cayley graph to align with an input graph through the perspective of effective resistance.

4 Cayley Graph Propagation

In the previous sections, we provided theoretical motivations for our proposed method of utilising the complete Cayley graph structure. The setup for CGP closely aligns with that of EGP in most aspects. We consider the input to a GNN as a node feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$.

The construction of the Cayley graph $\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n)$ is done by choosing the smallest n such that $|V(\mathrm{Cay}(\mathrm{SL}(2, \mathbb{Z}_n); S_n))| \geq |V|$. However, we no longer truncate the Cayley graph such that a subgraph $\mathbf{A}^{\mathrm{Cay}(n)}_{1:|V|,1:|V|}$ is extracted – instead, we opt for a different approach of retaining all of the nodes of the Cayley graph, and its corresponding adjacency matrix $\mathbf{A}^{\mathrm{Cay}(n)}$.

This construction requires us to add new nodes into the graph; hence, we need to modify the feature matrix into an extended version, $\mathbf{X}^{\mathrm{Cay}(n)} \in \mathbb{R}^{|V(\mathrm{Cay}(n))| \times d}$. To construct this, we featurise the first $|V|$ nodes using the data from $\mathbf{X}$, and treat any additional nodes as *virtual nodes*, initialised in some pre-defined way. Specifically:

$$\mathbf{X}^{\mathrm{Cay}(n)}_{1:|V|} = \mathbf{X} \qquad \mathbf{X}^{\mathrm{Cay}(n)}_{|V|+1:|V(\mathrm{Cay}(n)|} \sim \mathrm{InitVirt} \tag{3}$$

where $\mathrm{InitVirt}$ is any sampling procedure for initialising d-dimensional feature vectors. For example, we may choose to sample random features from $\mathcal{N}(0, 1)$, or in our case initialise them to zeros as done in [5, 30, 31]. In Section (Appendix B), we provide additional studies to compare the performance of different initialisation strategies.

Because EGP makes advantage of both the input graph (specified by $\mathbf{A}$) and the generated Cayley graph (specified by $\mathbf{A}^{\mathrm{Cay}(n)}$), we can also appropriately extend the original adjacency matrix, $\mathbf{A}$, to incorporate the new nodes. Since the input graph layers are intended to preserve the input graph topology as much as possible, one approach is to construct such a matrix $\widetilde{\mathbf{A}} \in \mathbb{R}^{|V(\mathrm{Cay}(n)| \times |V(\mathrm{Cay}(n)|}$ by adding *self-edges* to the virtual nodes only:

$$\widetilde{\mathbf{A}}_{1:|V|,1:|V|} = \mathbf{A} \qquad \widetilde{\mathbf{A}}_{|V|+1:|V(\mathrm{Cay}(n)|,|V|+1:|V(\mathrm{Cay}(n)|} = \mathbf{I} \tag{4}$$

$$\widetilde{\mathbf{A}}_{1:|V|,|V|+1:|V(\mathrm{Cay}(n)|} = \widetilde{\mathbf{A}}_{|V|+1:|V(\mathrm{Cay}(n)|,1:|V|} = \mathbf{0} \tag{5}$$

CGP now proceeds in the same manner as EGP: alternating GNN layers, such that every odd layer operates over the input graph—to preserve the topological information therein—and every even layer operates over the generated Cayley graph—to support bottleneck-free global communication. For a two-layer CGP model, this can be depicted as:

$$\mathbf{H} = \mathrm{GNN}(\mathrm{GNN}(\mathbf{X}^{\mathrm{Cay}(n)}, \widetilde{\mathbf{A}}; \theta_1), \mathbf{A}^{\mathrm{Cay}(n)}; \theta_2) \tag{6}$$

where θ_1 and θ_2 are the parameters of the first and second GNN layer, respectively. This implementation works with any choice of base GNN; here we may choose to take advantage of the graph isomorphism network ([32], GIN):

$$\vec{h}_u = \phi\left((1+\epsilon)\,\vec{x}_u + \sum_{v\in\mathcal{N}_u} \vec{x}_v\right) \tag{7}$$

where $\vec{x}_u \in \mathbb{R}^d$ are the features of node u, ϵ is a learnable scalar, and ϕ is a MLP. Our experimentation in Section 5 shows that CGP is MPNN agnostic.

The final node embeddings $\mathbf{H} \in \mathbb{R}^{|V(\mathrm{Cay}(n)|\times k}$ may then be used for downstream node, graph or graph-level tasks. To avoid direct influence of virtual nodes in these predictions, we use only the embeddings corresponding to the original graph's nodes, that is, $\mathbf{H}_{1:|V|}$, in downstream tasks.

The CGP model upholds the requirements of the four criteria (i-iv) set by Deac et al. [14]—arguably, in a more theoretically grounded way than EGP; Figure 1 and 2 provides empirical evidence of this. Specifically, the lower diameter of the graph used in CGP enhances its ability to eliminate over-squashing and bottlenecks, which is further supported by having a higher spectral gap. Furthermore, the CGP model may be able to make up for one of the limitations of the Cayley graph construction: the inability to find the best way to align it to a given input graph, mitigating the potential for *stochastic* effects in the process. The additional virtual nodes act as "bridges" between poorly connected communities in the Cayley graph, ameliorating any poorly-connected regions caused by misalignment.

5 Experimentation

In this section, we empirically validate the efficacy of using the complete Cayley graph structure on a range of graph classification tasks. In particular, we first motivate CGP by extending the results of Deac et al. [14], comparing CGP against approaches that do not incur a computational complexity of having to examine the input graph's structure: master node [5], fully adjacent last layer (FA) [6] and EGP [14]. We then extend our empirical evaluation by comparing the performance of CGP against the state-of-the-art approaches that require *dedicated preprocessing* [9, 10, 12, 33–35] the TUDataset [36] and LRGB [37]. Due to space limitations, we defer the results of the LRGB to Appendix (Section D).

Beyond comparing the performance of CGP against the baselines, we evaluate the practicality of CGP by providing a runtime analysis against the aforementioned approaches that require dedicated processing approaches. In Appendix (Section E), we extend this thorough a scalability analysis. Furthermore, we provide an ablation study to investigate whether the complete Cayley graph structure is a suitable alternative to a fully-connected graph.

Experimental setting. We evaluate on the Open Graph Benchmark (OGB) [38] and TUDataset [36]. In our experiments, we prioritise a fair comparison, following the layer schema for each approach. For EGP and CGP, this includes the interweaving schema as depicted in Section 4. For FA [6], this consists of rewiring the last layer. All other approaches only propagate over base (or rewired) graph structure. Furthermore, we show that our proposed method is MPNN invariant by setting the underlying model to GCN [39] and GIN [32]. The chosen hyperparameters are in line with established foundations [10, 38] for each dataset respectively. Notably, the EGP paper limits its empirical analysis to only the graph classification tasks from OGB with GIN as the backbone GNN. Refer to Appendix

Table 1: Comparative performance evaluation of CGP against the baselines on the OGB. OOM denotes out-of-memory on a NVIDIA RTX 4090.

Model	OGBG-MOLHIV	OGBG-PPA
	Test ROC-AUC ↑	**Test ACC** ↑
GCN	0.7566 ± 0.0104	0.5483 ± 0.0209
+ Master Node	0.7531 ± 0.0128	0.5824 ± 0.0219
+ FA	0.7628 ± 0.0191	OOM
+ EGP	0.7731 ± 0.0081	$\mathbf{0.6821} \pm 0.0045$
+ **CGP**	$\mathbf{0.7794} \pm 0.0122$	0.6782 ± 0.0066
GIN	0.7678 ± 0.0183	0.5888 ± 0.0441
+ Master Node	0.7608 ± 0.0134	0.6069 ± 0.0062
+ FA	0.7718 ± 0.0147	OOM
+ EGP	0.7537 ± 0.0076	0.6533 ± 0.119
+ **CGP**	$\mathbf{0.7899} \pm 0.0090$	$\mathbf{0.6562} \pm 0.0147$

Table 2: Results of CGP compared against EGP, FA and the approaches that require dedicated preprocessing for GCN and GIN on the TUDataset. The experimental setup uses the setting as in Karhadkar et al. [10], and hyperparameters for each baseline from [9, 10, 12, 33–35]. The colours highlight First, Second and Third. OOT indicates out-of-time for the *dedicated prepreprocessing* time.

Model	REDDIT-BINARY	IMDB-BINARY	MUTAG	ENZYMES	PROTEINS	COLLAB
GCN	77.735 ± 1.586	60.500 ± 2.729	74.750 ± 4.030	29.083 ± 2.363	66.652 ± 1.933	70.490 ± 1.628
+ FA	OOM	48.950 ± 1.652	70.250 ± 4.608	28.667 ± 3.693	71.071 ± 1.506	72.039 ± 0.771
+ DIGL	77.350 ± 1.206	49.600 ± 2.435	70.500 ± 5.045	30.833 ± 1.537	72.723 ± 1.420	56.470 ± 0.865
+ SDRF	77.975 ± 1.479	59.000 ± 2.254	74.000 ± 3.462	26.667 ± 2.000	67.277 ± 2.170	71.330 ± 0.807
+ FoSR	77.750 ± 1.385	59.750 ± 2.357	75.250 ± 5.722	24.167 ± 3.005	70.848 ± 1.618	67.220 ± 1.367
+ BORF	OOT	48.900 ± 0.900	76.750 ± 0.037	27.833 ± 0.029	67.411 ± 0.016	OOT
+ GTR	79.025 ± 1.248	60.700 ± 2.079	76.500 ± 4.189	25.333 ± 2.931	72.991 ± 1.956	72.600 ± 1.025
+ PANDA	87.275 ± 1.033	68.350 ± 2.346	76.750 ± 5.531	30.667 ± 2.019	70.134 ± 1.518	73.850 ± 0.695
+ EGP	67.550 ± 1.200	59.700 ± 2.371	70.500 ± 4.738	27.583 ± 3.262	73.304 ± 2.516	69.470 ± 0.970
+ CGP	67.050 ± 1.483	56.200 ± 1.825	83.750 ± 3.597	31.000 ± 2.397	73.036 ± 1.291	69.630 ± 0.730
GIN	84.600 ± 1.454	71.250 ± 1.509	80.500 ± 5.143	35.667 ± 2.803	70.312 ± 1.749	71.490 ± 0.746
+ FA	OOM	69.900 ± 2.332	80.250 ± 5.314	47.833 ± 2.529	72.902 ± 1.419	72.740 ± 0.786
+ DIGL	84.575 ± 1.265	52.650 ± 2.150	78.500 ± 4.189	41.500 ± 3.063	72.321 ± 1.440	57.620 ± 1.010
+ SDRF	84.550 ± 1.396	69.550 ± 2.381	80.500 ± 4.177	37.167 ± 2.709	69.509 ± 1.709	72.958 ± 0.419
+ FoSR	85.750 ± 1.099	69.250 ± 1.810	80.500 ± 4.738	28.083 ± 2.301	71.518 ± 1.767	71.720 ± 0.892
+ BORF	OOT	70.700 ± 0.018	79.250 ± 0.038	34.167 ± 0.029	70.625 ± 0.017	OOT
+ GTR	85.474 ± 0.826	69.550 ± 1.473	79.000 ± 3.847	31.750 ± 2.466	72.054 ± 1.510	71.849 ± 0.710
+ PANDA	90.325 ± 0.867	68.350 ± 2.346	83.250 ± 3.262	42.167 ± 2.286	72.321 ± 1.786	73.320 ± 0.814
+ EGP	77.875 ± 1.563	68.250 ± 1.121	81.500 ± 4.696	40.667 ± 3.095	70.848 ± 1.568	72.330 ± 0.954
+ CGP	78.225 ± 1.268	71.650 ± 1.532	85.250 ± 3.200	50.083 ± 2.242	73.080 ± 1.396	73.350 ± 0.788

(Section C) for more detail on our experimental setting and hyperparameters used for the OGB and TUDataset.

OGB. For real-world comparison and to extend the foundations of EGP, we first provide results on two graph classification tasks, OGBG-MOLHIV and OGBG-PPA, from the OGB [38]. We compare CGP against techniques that do not require *dedicated preprocessing*. OGBG-MOLHIV is among the largest molecule property prediction datasets within the scope of the MoleculeNet benchmark [40], thus providing emulation for real-world analysis. OGBG-PPA focuses on classifying species based on their taxa, using their protein-protein association networks [41]. Our model takes inspiration from the open-source implementation of OGB and hyperparameters as given by [38], including fixing the number of layers to 5, a hidden dimension of 300, a dropout of 50% and with the only modification being a batch size of 64. We report across 10 seeds and 5 seeds for OGBG-MOLHIV and OGBG-PPA respectively. Our results in Table 1 show that overall the CGP model outperforms the other approaches that do not require *dedicated preprocessing*; exemplified in the results for OGBG-MOLHIV. Moreover, CGP consistently outperforms the base GCN and GIN, which is not the case for the other baseline models.

TUDataset. We extend our graph classification task analysis by evaluating on REDDIT-BINARY, IMDB-BINARY, MUTAG, ENZYMES, PROTEINS and COLLAB from the TUDataset [36]. Significantly, these datasets were chosen under the claim of Karhadkar et al. [10] that the topology of the graphs in relation to the tasks require long-range interactions. Accordingly, the TUDataset has become the cornerstone baseline for the *graph-rewiring* approaches investigating over-squashing. Thus, we compare CGP against EGP [14] and FA [29], as well as the following state-of-the-art graph rewiring techniques that required *dedicated preprocessing*: DIGL [33], SDRF [9], FoSR [10], BORF [34] and GTR [12]. Moreover, we include PANDA [35] as a unique approach that dynamically alters the *width* [7] of the model to alleviate *over-squashing* as opposed to rewiring the input graph structure.

The results in Table 2 prioritise a fair comparison to further pinpoint with certainty that the performance gain can be credited to the utilisation of the complete Cayley graph structure. Therefore, in Table 2 we use the hyperparameters and experimental setting as prescribed by Karhadkar et al. [10]. The hyperparameters include a hidden dimension of 64, the number of layers set to 4 and a dropout of 50%. In line with our OGB experimentation for all models, we also use Batch Norm [42]. Unlike EGP, FA and CGP, the *dedicated preprocessing* approaches have an optimisation target and

Table 3: CGP training, evaluation time (seconds per epoch), and memory consumption statistics in comparison to baseline models on REDDIT-BINARY and COLLAB from the TUDataset [36]. OOM signifies out-of-memory on a NVIDIA RTX 4090.

Model	REDDIT-BINARY			COLLAB		
	Train Time	**Eval Time**	**Mem. (GB)**	**Train Time**	**Eval Time**	**Mem. (GB)**
GIN	0.1049 ± 0.0237	0.0741 ± 0.0032	922	0.2787 ± 0.0345	0.2364 ± 0.0094	1722
+ FA	OOM	OOM	OOM	0.4625 ± 0.0507	0.4488 ± 0.0404	4746
+ FoSR	0.1117 ± 0.0268	0.0841 ± 0.0164	906	0.3129 ± 0.0386	0.2619 ± 0.0257	3320
+ PANDA	0.7902 ± 0.0597	0.7489 ± 0.0439	1316	2.1152 ± 0.0964	1.9347 ± 0.0924	4406
+ EGP	0.1215 ± 0.0257	0.0952 ± 0.0128	976	0.3096 ± 0.0372	0.2598 ± 0.0164	1696
+ **CGP**	0.1326 ± 0.0296	0.1147 ± 0.0135	1128	0.3191 ± 0.0321	0.2785 ± 0.0160	2418

thus feature additional approach-specific hyperparameters. For each dataset, we use the optimised hyperparameter setting as stated in each paper respectively. More details are found in Appendix (Section D). Karhadkar et al. [10] reports the results with a 95% confidence interval, thus we respect this for our TUDataset results. The reported results are averaged across 20 random seeds. Notably, we report OOT to indicate out-of-time for the *preprocessing* rewiring procedure for BORF on the REDDIT-BINARY and COLLAB datasets. This is in accordance with Nguyen et al. [34] which do not report results for these two datasets and corresponding hyperparameters. In addition, a time-out is reported in [35] for the aforementioned datasets, whilst in [43] they report out-of-memory for COLLAB.

The results in Table 2 underscore the effectiveness of CGP in comparison with state-of-the-art baselines. In particular, in the case of GIN, our CGP model obtains the highest accuracy for all datasets except for REDDIT-BINARY. The overall performance of CGP when applied to GCN is not as competitive as those of GIN, however, our results are still comparable with the other baselines. This is particularly notable when the sparsity of the Cayley graphs is considered in relation to certain datasets, such as IMDB-BINARY and COLLAB. The sparse nature of the Cayley graph means that unlike many graph rewiring techniques edges may be removed; refer to Appendix (Section C) for the dataset statistics. However, the results of CGP for GIN recover this loss in performance for IMDB-BINARY and COLLAB, emphasising the work of You et al. [44] in which the design space of GNNs can greatly impact a model's results. Finally, of significance is the parity of the hyperparameter's number of layers; Table 1 uses 5 layers, whereas Table 2 uses 4 layers. This demonstrates the performance of CGP is irrespective of the final layer being the input or Cayley graph.

Scalability. In the following, we investigate whether the *additional virtual nodes* as leveraged in CGP will induce an increased *computational complexity*, impacting the training time of the model. Accordingly, we compare CGP directly against EGP, FA and FoSR due to it being a graph-rewiring approach. Moreover, we include PANDA to provide extra detail to the results in Table 2. Even though PANDA obtains state-of-the-art performance, as stated in [35] this framework impacts the runtime. We use REDDIT-BINARY and COLLAB due to these datasets featuring the largest input graphs from the TUDataset. In Table 3 we report the average seconds per epoch for the training and evaluation time, as well as the memory consumption statistics and set the underlying model to GIN. In Appendix (Section E), we extend this discussion by conducting a scalability analysis. This includes comparing the *dedicated preprocessing* time of the techniques reported in Table 2 on the real-world datasets, as well as a synthetic benchmark as used by Karhadkar et al. [10].

The results highlight that *additional virtual nodes* in CGP have a negligible impact on the training and evaluation time. Even though many virtual nodes may be added, they are sparsely connected. The base model, FoSR [10], EGP [14] and CGP are shown to be akin with each other. CGP is shown to increase the memory consumption, however the sparsely connected virtual nodes have a minimal impact when compared with FA [29], FoSR [10] and PANDA [35]. The dataset statistics in Appendix (Section C) provide an explanation, such as REDDIT-BINARY being several times larger than all of the datasets featured in the TUDataset [36]. Notably, the results in Table 3 highlight the strengths of CGP against the approaches that require dedicated preprocessing, as well as the dense fully adjacent layer [29].

Table 4: Results of CGP using the Cayley graph in different layer approaches compared against FA on the TUDataset. † denotes last layer and $*$ denotes every layer. OOM is out-of-memory on a NVIDIA RTX 4090. The colours highlight First, Second and Third.

Model	REDDIT-BINARY	IMDB-BINARY	MUTAG	ENZYMES	PROTEINS	COLLAB
GCN	77.735 ± 1.586	60.500 ± 2.729	74.750 ± 4.030	29.083 ± 2.363	66.652 ± 1.933	70.490 ± 1.628
+ FA†	OOM	48.950 ± 1.652	70.250 ± 4.608	28.667 ± 3.693	71.071 ± 1.506	72.039 ± 0.771
+ FA*	OOM	49.700 ± 1.871	75.250 ± 4.554	27.167 ± 2.770	71.384 ± 1.380	54.990 ± 0.699
+ CGP	67.050 ± 1.483	56.200 ± 1.825	83.750 ± 3.597	31.000 ± 2.397	73.036 ± 1.291	69.630 ± 0.730
+ CGP†	89.600 ± 1.736	61.850 ± 5.850	78.750 ± 9.601	42.000 ± 4.522	71.250 ± 3.989	71.360 ± 1.663
+ CGP*	70.250 ± 3.673	54.550 ± 4.295	88.500 ± 7.921	35.750 ± 5.664	73.259 ± 3.346	56.170 ± 2.412
GIN	84.600 ± 1.454	71.250 ± 1.509	80.500 ± 5.143	35.667 ± 2.803	70.312 ± 1.749	71.490 ± 0.746
+ FA†	OOM	69.900 ± 2.332	80.250 ± 5.314	47.833 ± 2.529	72.902 ± 1.419	72.740 ± 0.786
+ FA*	OOM	54.250 ± 4.784	83.750 ± 7.224	34.250 ± 4.669	71.250 ± 4.721	55.270 ± 1.604
+ CGP	78.225 ± 1.268	71.650 ± 1.532	85.250 ± 3.200	50.083 ± 2.242	73.080 ± 1.396	73.350 ± 0.788
+ CGP†	87.650 ± 0.886	68.749 ± 1.933	83.250 ± 3.760	45.667 ± 3.004	70.848 ± 1.961	72.850 ± 0.985
+ CGP*	71.300 ± 1.855	50.200 ± 1.998	72.250 ± 3.760	30.750 ± 2.882	73.571 ± 1.840	56.910 ± 0.820

Ablation studies. In the following, we answer the question: *'is the complete Cayley graph structure a suitable alternative to a fully adjacent layer [6]?'*. In Table 4 we show that it is a promising avenue as the results for CGP† and FA† are similar for both GCN and GIN. However, the sparsity of CGP is highlighted in the results of REDDIT-BINARY, whereby FA runs out-of-memory (OOM). Accordingly, these results demonstrate the advantages of CGP, due to it being far more scalable. Notably, the results reported in Table 4 use the same experimental setting and hyperparameters as Table 2. Accordingly, these results demonstrate the advantages of CGP, due to it being far more scalable. Notably, the results reported in Table 4 use the same experimental setting and hyperparameters as

Next, we investigate in accordance with the recent work of Bechler-Speicher et al. [45]; we examine if we can ignore the input graph entirely and solely propagate over the Cayley graph structure. Our results in Table 4 conclude that the inductive bias endowed from the input graph is required. The baseline procedure of CGP (interweaving a Cayley graph with an input graph), and using a Cayley graph for the last layer only, outperforms using a Cayley graph solely for each layer. One explanation is that the interweaving schema of CGP aligns with the principles of JK networks [46], facilitating improved structure-aware representations by varying the neighbourhood ranges. However, the tone set by Bechler-Speicher et al. [45] is still a promising line of research, as the results of PROTEINS indicates that the optimal graph-structure used is still task dependent.

6 Conclusion

In this work, we presented Cayley Graph Propagation (**CGP**), an efficient propagation scheme that mitigates over-squashing. CGP utilises the complete Cayley Graph structure to guarantee improved information flow between nodes in the input graph. We highlight the advantageous topological properties of Cayley graphs for message-passing. We show that by truncating the Cayley graphs to align with the input graph, as suggested in Expander Graph Propagation, the resulting graph may contain bottlenecks. This is in contrast to the Cayley graph we use, which is guaranteed to be bottleneck-free. We demonstrate the effectiveness and efficiency of CGP compared to EGP and other rewiring approaches, over multiple real-world datasets, including large-scale and long-range datasets.

Limitations and Future Work. One limitation of our proposed model is the performance on datasets containing graphs with a comparatively higher node-to-edge ratio. For this reason, one such avenue for future work is aligning the Cayley graph edges such that that they retain the inductive bias of the task [47]. Additionally, it would be interesting to see how CGP performs in other tasks that utilise expander graphs, including but not limited to temporal graph rewiring [25]. Furthermore, concurrent work has used *virtual nodes* as the focal point of their proposed methods [48, 49] with Southern et al. [31] analysing the role of virtual nodes within the context of over-squashing. Therefore, we theorise an interesting setting would be applying these authors' approaches to the additional virtual nodes retained from the complete Cayley graph structure.

Acknowledgements

We would like to thank Petar Veličković's co-authors of the Expander Graph Propagation paper, including Andreea Deac and Mark Lackenby. Their original work of leveraging the use of expander graphs within the context of graph neural networks to alleviate bottlenecks and counter oversquashing provided the basis for our work. Furthermore, we would like to thank Alex Vitvitskyi, Csaba Szepesvári, Johannes Vallikivi and Dobrik Georgiev for their invaluable insights prior to submission.

References

[1] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 1

[2] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020. 1

[3] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020. 1

[4] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021. 1

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017. 1, 2, 3, 5, 6, 15

[6] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020. 1, 2, 3, 6, 9, 20

[7] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pages 7865–7885. PMLR, 2023. 1, 3, 5, 7

[8] Francesco Di Giovanni, T Konstantin Rusch, Michael M Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković. How does over-squashing affect the power of gnns? *arXiv preprint arXiv:2306.03589*, 2023. 1, 3, 5

[9] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=7UmjRGzp-A`. 1, 3, 6, 7, 16, 17, 18

[10] Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=3YjQfCLdrzz`. 1, 3, 6, 7, 8, 16, 17, 18, 20

[11] Pradeep Kr Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. Oversquashing in gnns through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022. 1, 3, 4

[12] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, pages 2528–2547. PMLR, 2023. 1, 3, 5, 6, 7, 14, 16, 17, 18

[13] Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lov\'asz bound. *arXiv preprint arXiv:2206.07369*, 2022. 1, 3, 20

[14] Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Learning on Graphs Conference*, pages 38–1. PMLR, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 17, 20

[15] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan

Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 2

[16] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. 2

[17] Peter Sarnak. What is . . . an expander? *Notices of the American Mathematical Society*, 51(7): 762–763, August 2004. ISSN 0002-9920. 2

[18] Emmanuel Kowalski. *An introduction to expander graphs*. Société mathématique de France, 2019. 3

[19] Giuliana P Davidoff, Peter Sarnak, and Alain Valette. *Elementary number theory, group theory, and Ramanujan graphs*, volume 55. Cambridge university press Cambridge, 2003. 3

[20] Dai Shi, Andi Han, Lequan Lin, Yi Guo, and Junbin Gao. Exposition on over-squashing problem on gnns: Current methods, benchmarks and challenges. *arXiv preprint arXiv:2311.07073*, 2023. 3

[21] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021. 3

[22] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021. 3

[23] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023. 4

[24] Thomas Christie and Yu He. Higher-order expander graph propagation. *arXiv preprint arXiv:2311.07966*, 2023. 4

[25] Katarina Petrović, Shenyang Huang, Farimah Poursafaei, and Petar Veličković. Temporal graph rewiring with expander graphs. *arXiv preprint arXiv:2406.02362*, 2024. 4, 9

[26] Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. *arXiv preprint arXiv:2309.04332*, 2023. 4, 19

[27] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Society, 1997. 4, 20

[28] Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems in Analysis*, 625(195-199):110, 1970. 4

[29] Noga Alon and Vitali D Milman. Eigenvalues, expanders and superconcentrators. In *25th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 320–322, 1984. 4, 7, 8

[30] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer. In *International Conference on Machine Learning*, pages 3408–3430. PMLR, 2023. 5, 15

[31] Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes F Lutzeyer. Understanding virtual nodes: Oversmoothing, oversquashing, and node heterogeneity. *arXiv preprint arXiv:2405.13526*, 2024. 5, 9, 15

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 6

[33] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in neural information processing systems*, 2019. 6, 7, 16, 18

[34] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, pages 25956–25979. PMLR, 2023. 7, 8, 16, 17, 18

[35] Jeongwhan Choi, Sumin Park, Hyowon Wi, Sung-Bae Cho, and Noseong Park. Panda: Expanded width-aware message passing beyond rewiring. In *Forty-first International Conference on Machine Learning*, 2024. 6, 7, 8, 16, 17, 20

[36] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020. 6, 7, 8, 15, 16, 18

[37] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022. 6, 16, 17, 18

[38] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020. 6, 7, 16

[39] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 6

[40] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 7, 16

[41] Damian Szklarczyk, Annika Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda Doncheva, John Morris, Peer Bork, Lars Jensen, and Christian von Mering. String v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47, 11 2018. doi: 10.1093/nar/gky1131. 7, 16

[42] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 7, 16, 17

[43] Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni. Locality-aware graph-rewiring in gnns. *arXiv preprint arXiv:2310.01668*, 2023. 8, 17

[44] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020. 8, 17

[45] Maya Bechler-Speicher, Ido Amos, Ran Gilad-Bachrach, and Amir Globerson. Graph neural networks use graphs when they shouldn't. In *Forty-first International Conference on Machine Learning*, 2024. 9

[46] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018. 9

[47] Igor Sterner, Shiye Su, and Petar Veličković. Commute-time-optimised graphs for gnns. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. 9

[48] Chendi Qian, Andrei Manolache, Christopher Morris, and Mathias Niepert. Probabilistic graph rewiring via virtual nodes. *arXiv preprint arXiv:2405.17311*, 2024. 9

[49] Florian Sestak, Lisa Schneckenreiter, Johannes Brandstetter, Sepp Hochreiter, Andreas Mayr, and Günter Klambauer. Vn-egnn: E (3)-equivariant graph neural networks with virtual nodes enhance protein binding site identification. *arXiv preprint arXiv:2404.07194*, 2024. 9

[50] Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022. 14

[51] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: international conference on learning representations*, pages 1–15. ICLR US., 2015. 16

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 16, 17

[53] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 17

[54] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020. 17

[55] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023. 17

[56] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL http://www.sciencemag.org/cgi/content/abstract/286/5439/509. 19

[57] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks, 2021. URL https://arxiv.org/abs/2010.08853. 19

[58] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959. 19

[59] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019. 20

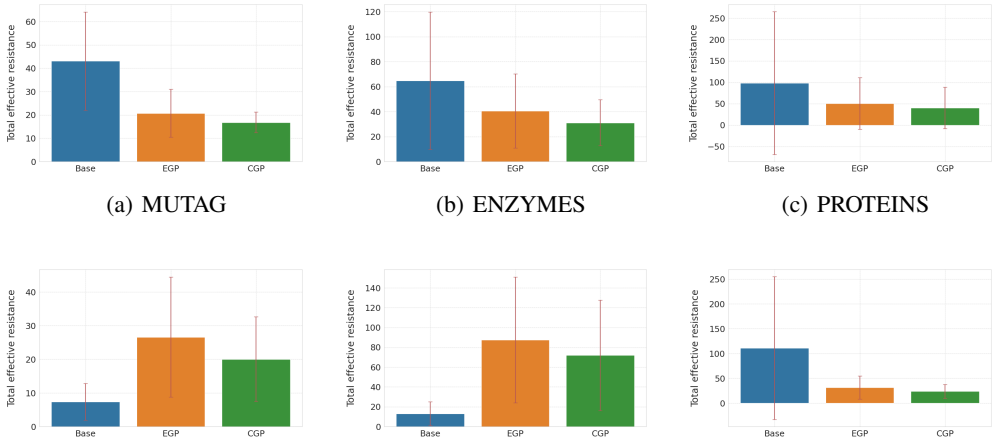
[60] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023. 20

[61] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 20

[62] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019. 20

[63] Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Understanding convolution on graphs via energies. *arXiv preprint arXiv:2206.10991*, 2022. 20

[64] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022. 20



(a) MUTAG (b) ENZYMES (c) PROTEINS

(d) IMDB-BINARY (e) COLLAB (f) OGBG-MOLHIV

Figure 3: Comparison of the total effective resistance R_{tot} for CGP against the baseline model and EGP. A lower total effective resistance indicates that a graph is less susceptible to over-squashing.

A Effective Resistance of Cayley graphs

In this work, we have used the *Cheeger constant* as an approach to measure *bottlenecks* in a graph [50] in regards to *over-squashing*. An alternative closely related approach is measuring *over-squashing* through the lens of *effective resistance* [12]. Stemming from the field of electrical engineering, the effective resistance between two nodes u and v reflects the ease of current flow. In turn, this concept has become analogous to measuring the connectivity between nodes within graph theory. Formally, the effective resistance between two nodes u and v can be expressed using the pseudoinverse $\mathbf{L}^+$ of $\mathbf{L}$, $R_{u,v} = (\mathbf{1}_u - \mathbf{1}_v)^\top \mathbf{L}^+ (\mathbf{1}_u - \mathbf{1}_v)$, where $\mathbf{1}_u$ and $\mathbf{1}_v$ are indicators for nodes u and v.

The *total effective resistance* R_{tot} then builds upon this by measuring the *effective resistance* for all pairs of nodes within a graph, thus providing a metric to quantify *over-squashing* in a graph. As per Black et al. [12], the *total effective resistance* R_{tot} is given by:

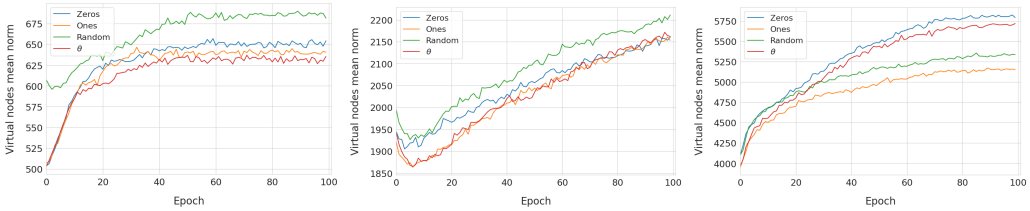
$$R_{tot} = \sum_{u>v} R_{u,v} = n \cdot \mathrm{Tr}(\mathbf{L}^+) = n \sum_{i=1}^{n-1} \frac{1}{\lambda_i}. \tag{8}$$

The results in Figure 3 show the average of the *total effective resistance* R_{tot} for all the corresponding Cayley graphs against the base input graphs and truncated Cayley graphs as found in EGP. Akin to the results presented in Black et al. [12], for a fair evaluation we do not include graphs that may be disconnected. This is because the Cayley graphs used in CGP are a complete and regular graph structure, therefore every node will be connected. The results show that CGP consistently has a lower total effective resistance R_{tot} in comparison to EGP. Significantly, the complete Cayley graph structure is chosen by recalling $|V(\mathrm{Cay}(\mathrm{SL}(2,\mathbb{Z}_n); S_n))| \geq |V|$, therefore the total effective resistance R_{tot} for CGP may be inflated due to it potentially being summed over more pairs of nodes. This further reinforces our claim that it is more beneficial to use the complete Cayley graph structure with the *additional nodes* serving as shortcuts for message passing between nodes along the graph.

In line with our results reported in our empirical evaluation, for certain datasets the total effective resistance for EGP and CGP is higher than the base input graph. The statistics of the datasets reported in Table 6 provide evidence to explain this observation; IMDB-BINARY and COLLAB have a significantly higher edge-to-node ratio when compared to the more sparse Cayley graph's structure. Nevertheless, our results reported in Table 2 illustrate that the CGP model counteracts this by still providing leading performance on these datasets.

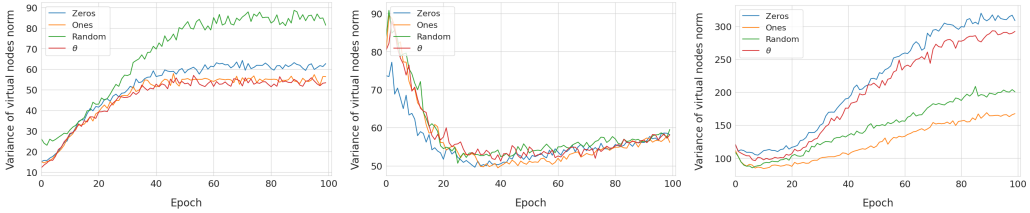
Table 5: Results of CGP using different virtual node initialisation strategies, including ZEROS, ONES, RANDOM and θ on the TUDataset. ZEROS is the baseline CGP and θ denotes the nodes are initialised using a learnable parameter.

Model	REDDIT-BINARY	IMDB-BINARY	MUTAG	ENZYMES	PROTEINS	COLLAB
+ CGP-ZEROS	78.225 ± 1.268	$\mathbf{71.650} \pm 1.532$	$\mathbf{85.250} \pm 3.200$	$\mathbf{50.083} \pm 2.242$	$\mathbf{73.080} \pm 1.396$	$\mathbf{73.350} \pm 0.788$
+ CGP-ONES	79.175 ± 1.992	71.000 ± 1.806	82.250 ± 3.773	47.833 ± 2.428	70.535 ± 1.972	73.200 ± 0.964
+ CGP-RAND	$\mathbf{82.300} \pm 1.368$	71.200 ± 1.532	84.000 ± 3.130	44.083 ± 2.891	70.000 ± 1.937	73.020 ± 0.742
+ CGP-θ	80.150 ± 1.888	69.250 ± 2.034	85.000 ± 3.606	49.750 ± 2.752	70.133 ± 2.053	73.410 ± 0.977



(a) MUTAG (b) ENZYMES (c) PROTEINS

Figure 4: The mean norm of the virtual node embeddings for CGP using different initialisation strategies on the TUDataset, including ZEROS, ONES, RANDOM and θ.



(a) MUTAG (b) ENZYMES (c) PROTEINS

Figure 5: The variance in the norm of the virtual node embeddings for CGP using different initialisation strategies on the TUDataset, including ZEROS, ONES, RANDOM and θ.

B Virtual Nodes Initialisation

In Section 4, we show that virtual nodes in our proposed CGP method provide the flexibility to be initialised in some pre-defined manner. For the baseline CGP model, we opted to initialise them to zeros, which is in line with the work of [5, 30, 31]. In this section, we empirically evaluate whether different node initialisation strategies impact the performance of CGP. We report the results in Table 5, leveraging the TUDataset with the hyperparameter setting as prescribed in Section C. For our other node initialisation strategies, we choose constant ones, random features that are sampled from $\mathcal{N}(0, 1)$ and θ where θ is a learnable parameter. Notably, ones is an interesting strategies as for datasets with no node features they are commonly assigned to a constant of one [36]. The results overall show that our baseline model of CGP-ZERO performs the best across all datasets, however all approaches obtain similar performance within the variance of each other.

Next, we investigate the role of the additional virtual nodes in CGP. To this end, we use the mean norm of the virtual node embeddings, as well as the variance between them as a statistical proxy to measure the diversity of the embeddings. We visualise the results in Figure 4 and 5 respectively for all of the virtual node initialisation strategies found in Table 5, however we fix the number of epochs to 100 and report the average over 10 random seeds. The results show that during the training lifetime of the model, the mean norm and variance of virtual node node embeddings grow, indicating that they are learning distinct representations. Overall, the results in this section show that the initialisation of the virtual nodes impact CGP. However, due to the focus of CGP being to use complete Cayley graph structure and the additional virtual nodes serving as a conduit to this

Table 6: Statistics of the datasets from OGB, TUDataset and LRGB used as part of our emperical evaluation. * denotes the number of additional virtual nodes (VN) added in our CPG model.

Dataset	#Graphs	Average #Nodes	Average #Edges	#Classes	Average #VN*	Max #VN*
OGBG-MOLHIV	41,127	25.5	27.5	2	11.8	191
OGBG-PPA	158,100	243.4	2,266.1	37	50.3	191
REDDIT-BINARY	2,000	429.6	995.5	2	139.4	1809
IMDB-BINARY	1,000	19.8	193.1	2	10.7	71
MUTAG	188	17.9	39.6	2	8.0	23
ENZYMES	600	32.6	124.3	6	14.6	71
PROTEINS	1,113	39.1	145.6	2	20.6	190
COLLAB	5,000	74.5	4,914.4	3	34.2	260
PEPTIDES-FUNC	15,535	150.94	307.3	10	67.0	263
PEPTIDES-STRUCT	15,535	150.94	307.3	11	67.0	263

end. Therefore, a deeper understanding the role of the sparsely connected virtual nodes and their initialisation strategies presents an interesting avenue for future work.

C Experimental Details

In this section, we provide thorough details regarding our experimental setting for each of our datasets. We utilise the well-established experimental settings [10, 38] that are used for OGB and TUDataset respectively. The experimental setting for the LRGB [37] is found in Section D. Refer to Table 6 for the dataset statistics. Of significance, to rule out performance gain due to hyperparameter tuning in our results we use the same setting of GNN and hyperparameters for each task and model. All of our experiments use the default settings of the Adam optimiser [51] with a learning rate of 1×10^{-3}, however the TUDataset and LRGB use the REDUCELRONPLATEAU scheduler with differing parameters.

C.1 Datasets

OGB Datasets. From the OGB [38] we consider two of the graph classification tasks: OGBG-MOLHIV and OGBG-PPA. OGBG-MOLHIV is among the largest molecule property prediction datasets within the scope of the MoleculeNet benchmark [40], and OGBG-PPA focuses on classifying species based on their taxa, using their protein-protein association networks [41]. The OGB provides a unified evaluation protocol for each dataset, including application-specific data splits and evaluation metrics. Accordingly, our experimental setup to empirically evaluate against the OGB datasets leverages the *official* open-source implementation from the OGB authors [38]. OGBG-MOLHIV uses a 80%/10%/10% train/validation/test split and ROC-AUC for the evaluation metric, whilst OGBG-PPA uses a 50%/28%/22% train/validation/test split and accuracy for the evaluation metric. The hyperparameter setting for our models includes 5 layers, a hidden dimension of 300, a dropout of 50% and the use of Batch Norm [42]. In Table 1 the results reported for our model are trained to 100 epochs across 10 seeds and 5 seeds for OGBG-MOLHIV and OGBG-PPA respectively.

TUDataset. The TUDataset [36] is considered under the claim of Karhadkar et al. [10] that the topology of the graphs in relation to the tasks require long-range interactions. Thus, we consider all graph classification tasks from the TUDataset: REDDIT-BINARY, IMDB-BINARY, MUTAG, ENZYMES, PROTEINS and COLLAB. Our setup for all TUDataset experiments is akin to the well-established and open-source setting of Karhadkar et al. [10]. Accordingly, we train our GNNs with 80%/10%/10% train/validation/test split and use a stopping patience of 100 epochs based on the validation loss. We fix the number of layers to 4 with a hidden dimension of 64 and a dropout of 50%. The REDUCELRONPLATEU uses the default setting as found in PyTorch [52]. Additionally, in accordance with Karhadkar et al. [10] the results for the TUDataset are reported to a 95% confidence interval. However unlike Karhadkar et al. [10], we report the accuracy over 20 random seeds, set the maximum number of epochs to 300 [35], as well as apply Batch Norm [42].

For TUDataset experiments, we compare CGP against the state-of-the-art approaches that require *dedicated preprocessing* and use the hyperparameters as in the well-established baselines: DIGL [33], SDRF [9], FoSR [10], BORF [34], GTR [12] and PANDA [35]. By adopting the hyperparameters of these baselines methods, we not only ensure a fair comparison, but demonstrate that CGP remains

competitive even under their optimised settings. For the graph rewiring techniques, we follow the hyperparameters as reported in the respective baselines. This includes the teleport probability (α) and sparsification threshold (ϵ) for DIGL, as well as the number of rewiring iterations for SDRF and FoSR being derived from Karhadkar et al. [10]. For BORF this includes the number of batches (n), number of edges added per batch (h), and number of edges removed per batch (k) from Nguyen et al. [34]. For GTR this includes the number of edges added from Black et al. [12]. Finally, this includes the hyperparameters for PANDA from Choi et al. [35]: the centrality metric $C(G)$, the top-k nodes and the increased width (i.e. hidden dimension) of the model.

C.2 Hardware

All of our experimentation was conducted on a local machine with an AMD Ryzen 9 7950X3D 16-Core Processor (4.20 GHz), NVIDIA RTX 4090 (24 GB) and 64 GB of RAM. The only exception is the OGBG-PPA results in which some of the baselines were processed on an external server with $8\times$ NVIDIA QUADRO RTX 8000 (48 GB). The following libraries were used as part of the implementation PyTorch [52], PyTorch Geometric [53] and NumPy [54].

D Additional Experiments

In this section, we provide additional results to solidify the efficacy of our CGP model by comparing it against state-of-the-art graph rewiring techniques [9, 10] and EGP [14] on the Long Range Graph Benchmark (LRGB) [37].

Datasets. We consider the PEPTIDES datasets from the Long Range Graph Benchmark (LRGB) [37], which have two related tasks PEPTIDES-FUNC and PEPTIDES-STRUCT. The former is a peptide feature classification task in which the objective is to predict the peptide function out of 10 classes with the performance being measured by Average Precision (AP). The latter consists of the same graphs as PEPTIDES-FUNC, however instead it is a graph regression task in which the aim is to predict aggregated 3D properties of the peptides at the graph level; the performance metric is Mean Absolute Error (MAE). The dataset statistics for the LRGB can be found in Table 6.

Table 7: Comparative performance evaluation of CGP against graph rewiring techniques on the LRGB.

Model	PEPTIDES-FUNC	PEPTIDES-STRUCT
	Test AP ↑	**Test MAE** ↓
GCN	0.5029 ± 0.0058	0.3587 ± 0.0006
+ SDRF	0.5041 ± 0.0026	0.3559 ± 0.0010
+ FoSR	0.4534 ± 0.0090	0.3003 ± 0.0007
+ EGP	0.4972 ± 0.0023	0.3001 ± 0.0013
+ CGP	$\mathbf{0.5106} \pm 0.0014$	$\mathbf{0.2931} \pm 0.0006$
GIN	0.5124 ± 0.0055	0.3544 ± 0.0014
+ SDRF	0.5122 ± 0.0061	0.3515 ± 0.0011
+ FoSR	0.4584 ± 0.0079	0.3008 ± 0.0014
+ EGP	0.4926 ± 0.0070	0.3034 ± 0.0027
+ CGP	$\mathbf{0.5159} \pm 0.0059$	$\mathbf{0.2910} \pm 0.0011$

Experimental details. Similar to the OGB, the LRGB [37] provides an experimental setting with the aim to have unified experimental evaluation of their benchmarks. Correspondingly, we leverage the LRGB implementation that is built upon the GraphGym module [44]. For both PEPTIDES tasks it uses a 70%/15%/15% train/validation/test split. The experimental setup of Dwivedi et al. [37] fixes the models number of layers to 5, and does not use dropout, but it does use Batch Norm [42]. However, we reduce the models number of parameters using a hidden dimension of 64 as in Nguyen et al. [34], as opposed to 300 [37]. Additionally, we reduce the number of epochs to 250, which is in line with Tönshoff et al. [55]. A REDUCELRONPLATEAU scheduler is used following Dwivedi et al. [37] settings of a patience of 20 epochs, a decay factor of 0.5 and a minimum learning rate of 1×10^{-5}. We use the graph rewiring hyperparameters from [34, 43].

Results. The results in Table 7 showcase that CGP outperforms the state-of-the-art graph rewiring approaches, as well as EGP for both GCN and GIN. However, it is noted that the work of Tönshoff et al. [55] achieves improved performance through extensive hyperparameter tuning. Nevertheless, this is beyond the scope of evaluating the impact of using the complete Cayley graph structure.

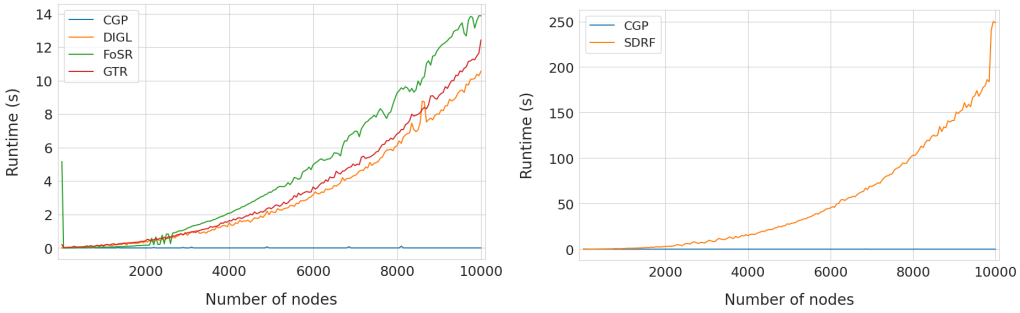


Figure 6: Synthetic preprocessing benchmark for CGP in regards to graph rewiring techniques, using Erdős–Rényi graphs with a probability $p = \frac{5 \log n}{n}$. **Left:** Preprocessing time of CGP against DIGL, FoSR and GTR. **Right:** Preprocessing time of CGP against SDRF.

E Scalability Analysis of CGP

Table 8: Preprocess graph rewiring runtime (in seconds) for each graph in the TUDataset. OOT indicates out-of-time for the *prepreprocessing rewiring* time.

Model	REDDIT-BINARY	IMDB-BINARY	MUTAG	ENZYMES	PROTEINS	COLLAB
DIGL	40.3837	0.411771	0.0342833	0.243485	0.491458	56.3175
SDRF	359.128	5.13257	0.669701	1.71482	3.02873	619.125
FoSR	74.8568	4.54634	4.71567	4.56855	5.04358	9.79994
BORF	OOT	465.408	53.7069	179.573	351.173	OOT
GTR	118.549	3.39839	1.54127	2.87399	6.49714	92.6125
PANDA	6.13925	0.789759	0.246243	0.278594	0.248043	230.850
EGP	0.245215	0.0185697	0.00446963	0.0163198	0.0393348	0.129567
CGP	0.226065	0.0211341	0.00438905	0.0166841	0.0348585	0.131188

Table 9: Preprocess runtime (in seconds) for state-of-the-art graph rewiring techniques for each graph in the LRGB dataset.

Model	PEPTIDES-FUNC	PEPTIDES-STRUCT
SDRF	61.0356	56.1561
FoSR	23.4263	24.1858
EGP	1.36170	1.29376
CGP	1.27776	1.29608

We empirically analyse the scalability of CGP by comparing the computational preprocessing time against the state-of-the-art graph rewiring techniques [9, 10, 12, 33, 34]. We first provide a real-world evaluation by benchmarking the preprocessing time on two real-world datasets: TUDataset [36] and LRGB [37]. The results are reported in Table 8 and 9, using the same graph rewiring techniques as in Table 2 and 7 respectively. To extend our scalability analysis, we create a synthetic benchmark by leveraging Erdős–Rényi with a probability $p = \frac{5 \log n}{n}$ (as used by Karhadkar et al. [10]) to create graphs of up to 10,000 nodes. In line with the results reported in Table 2, we do not conduct the synthetic benchmark for BORF [34], due to the impracticality of the rewiring time which is highlighted in Section 5.

The results show the efficacy of our proposed CGP model, as the preprocessing time is orders of magnitude lower than the graph rewiring techniques. To this end, we examine the lack of computational preprocessing time required to generate the corresponding Cayley graphs for both CGP and EGP. Overall, our results show the practicality of CGP to scale to large graphs when compared to the graph rewiring approaches. This is reinforced by the experimentation being conducted on the local machine with leading hardware as in Section C.2. In particular, both the CPU and GPU deliver

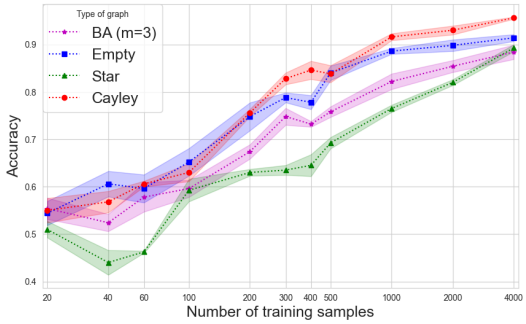


Figure 7: The learning curves of the same GNN model trained on graphs that have the same node features and only differ in their graph structure, which is sampled from different distributions. The label is computed from the node features without the use of any graph structure. The GNN overfits the graph structure instead of ignoring it, and therefore the model performance differ across different graph distributions. Cayley graphs exhibit the best performance, and robustness to overfitting.

top-tier clock speeds, therefore on lower-performing hardware, the graph rewiring techniques could have a more detrimental effect.

F Cayley graphs as Regular graphs

In this section, we explore the additional benefits of Cayley graphs being a regular graph. It was observed in [26] that GNNs tend to overfit the given graph structure, even in cases where it does no provide useful information for the predictive task. Nonetheless, it was shown that regular graphs exhibit robustness to this overfitting. As Cayley graphs are regular graphs, they exhibit this robustness.

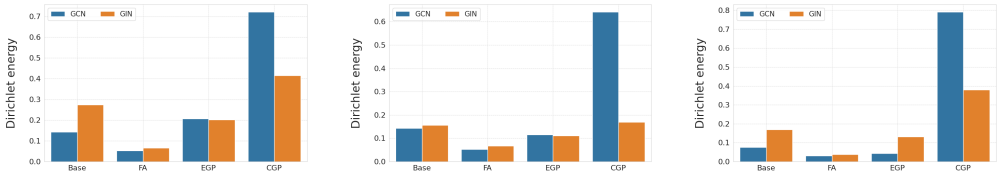
We repeat the experiments from [26] to ensure the robustness of Cayley graphs to graph overfitting. The task is a binary classification task where the label is independent of the graph, and is computed only over the features. We used the Sum task that was presented in [26]: the label is generated using a teacher GNN that simply sums the node features and applies a linear readout to produce a scalar.

We used four different datasets from this baseline by sampling graph-structures from different graph distributions. The set of node feature vectors remains the same across all the datasets, and thus, the datasets differ only in their graph structure. The graph distributions we used are: Cayley graphs over 24 nodes, star-graph (Star) where the only connections are between one specific node and all other nodes, and the preferential attachment model (BA) [56], where the graph is built by incrementally adding new nodes and connecting them to existing nodes with probability proportional to the degrees of the existing nodes. We used the data as is with empty graphs (Empty) as a baseline to compare to. On each dataset, we varied the training set size and evaluated test errors on 5 runs with random seeds.

The results are shown in Figure 7. The GNN trained on the Cayley graphs performs similarly to when trained on empty graphs. Nonetheless, when trained on other distributions, the performance decreases and does not recover even with 4000 training samples. This demonstrate the robustness of Cayley graphs to graph overfitting.

Extrapolation. The ability and failures of GNNs to extrapolate to graphs of sizes larger then the one presented during training was examined in Yehudai et al. [57]. It was shown that size generalisation is dependent on local structures around each node, called *d-patterns*. In particular, if increasing the graph size does not change the distribution of these d-patterns, then extrapolation to larger graph sizes is guaranteed.

Sum Task. This is a binary classification synthetic task with a graph-less ground truth function. To generate the label, we use a teacher GNN that simply sums the node features and applies a linear readout to produce a scalar. The data contains non-informative graph-structures which are drawn from the GNP graph distribution [58], where the edges are sampled i.i.d with probability p (we used $p = 0.5$).



(a) MUTAG (b) ENZYMES (c) PROTEINS

Figure 8: Comparison of the Dirichlet energy for CGP against the baseline model, FA and EGP for the TUDataset. A higher energy indicates that the proposed approach is more robust to the over-smoothing problem.

The teacher readout is sampled once from $\mathcal{N}(0,1)$ and used for all the graphs. All graphs have $n = 20$ nodes, and each node is assigned with a feature vector in $\mathbb{R}^{128}$ sampled i.i.d from $\mathcal{N}(0,1)$. We used a 1-layer "student" GNN following the teacher model, with readout and ReLU activations.

We evaluated the learning curve with an increasing amount of $\{20, 40, 60, 100, 200, 300, 400, 500, 1000, 2000, 4000\}$ samples. We note that the GNN has a total of $\sim$16,000 parameters, and thus, it is over-parameterised and can fit the training data with perfect accuracy.

G Dirichlet Energy of Cayley graphs

Here, we evaluate the impact of propagating over the complete Cayley graph structure in regards to the over-smoothing problem using the Dirichlet energy. Over-smoothing is an independent problem from over-squashing, but another well-known problem impacting the expressivity of GNNs is *over-smoothing* [59, 60]. This phenomenon occurs in GNNs when the number of layers increases [61, 62], such that node features become increasingly similar [63]. However, the over-smoothing problem is correlated with over-squashing due to a common approach to the latter being graph rewiring; too many additional edges lead to over-smoothing [10]. There are varying approaches to measure over-smoothing for a graph with one such notable metric being the Dirichlet energy [10, 13, 64].

The Dirichlet energy quantifies over-smoothing by measuring the deviation of a function on the graph from being constant between connected node pairs, thus indicating the level of non-smoothness in the signals [27]. In turn, the Dirichlet energy has been used to measure the amount of over-smoothing in graph representations [10, 13, 35].

Similar to EGP [14] and FA [6], the CGP model uses an independent graph structure to propagate information over, as opposed to the graph rewiring approaches in which they directly alter the input graph structure. Consequently, we conduct our Dirichlet energy analysis against EGP and FA, which also fall under the category of approaches that do not require *dedicated preprocessing*. The results in Figure 8 show that CGP consistently obtains a higher Dirichlet energy for both GCN and GIN when compared to EGP and FA. This further highlights the strengths of the CGP model to mitigate over-squashing, whilst minimising the negative implications of over-smoothing through the use of the *additional virtual nodes* retained from the complete Cayley graph structure.