Curvature-based Pooling within Graph Neural Networks

Cedric Sanders[1*], Andreas Roth[1*][0000−0002−0515−7635] (✉), and Thomas Liebig[1,2][0000−0002−9841−1101]

[1] TU Dortmund University, Germany
[2] Lamarr Institute for Machine Learning and Artificial Intelligence
{cedric.sanders,andreas.roth,thomas.liebig}@tu-dortmund.de

Abstract. Over-squashing and over-smoothing are two critical issues, that limit the capabilities of graph neural networks (GNNs). While over-smoothing eliminates the differences between nodes making them indistinguishable, over-squashing refers to the inability of GNNs to propagate information over long distances, as exponentially many node states are squashed into fixed-size representations. Both phenomena share similar causes, as both are largely induced by the graph topology. To mitigate these problems in graph classification tasks, we propose CurvPool, a novel pooling method. CurvPool exploits the notion of curvature of a graph to adaptively identify structures responsible for both over-smoothing and over-squashing. By clustering nodes based on the Balanced Forman curvature, CurvPool constructs a graph with a more suitable structure, allowing deeper models and the combination of distant information. We compare it to other state-of-the-art pooling approaches and establish its competitiveness in terms of classification accuracy, computational complexity, and flexibility. CurvPool outperforms several comparable methods across all considered tasks. The most consistent results are achieved by pooling densely connected clusters using the sum aggregation, as this allows additional information about the size of each pool.

Keywords: Graph Neural Networks · Pooling · Over-squashing · Over-smoothing.

1 Introduction

Graph neural networks (GNNs) [17] combine the computational power of neural networks with the structure of graphs to exploit both the topology of graphs and the available graph signal. Their applications are manifold, as they are used to classify single nodes within a graph (node classification) [17,33], classify entire graphs (graph classification) [45], and predict missing edges within the graph (link prediction) [31]. They are inhibited by several problems that impact the achieved results negatively. We propose a method to mitigate two of these

[*] Equal contribution

problems for the graph classification task, namely over-smoothing [27,30,8] and over-squashing [1,36].

Over-smoothing describes a phenomenon that results in node representations becoming overly similar when increasing the depth of the GNN. This leads to a loss of relevant information and leads to worse empirical results across many tasks [17,21,30]. Various theoretical investigations confirmed that this problem is greatly enhanced by the underlying structure of the graph[21,30,7]. Densely connected areas of the graph tend to over-smooth faster than sparsely connected areas [41].

Similarly, over-squashing also leads to loss of information albeit in a different way. It describes the inability of GNNs to propagate information over long distances in a graph. A recent theoretical investigation traced this back to bottlenecks in the graph [1], which describe edges connecting denser regions of the graph. With an increased number of layers, exponentially much information has to get passed through these edges, but the feature vectors are of limited constant size. Since bottlenecks are an inherent attribute of the underlying graph, this problem is also amplified by the graph topology [1,36].

While various directions addressing over-smoothing have been proposed [36,34,41], specifically for the graph classification task, pooling methods are a promising direction [43,23], which cluster sets of nodes in the graph into a single node. This can improve the data flow and change the underlying graph topology to one more suited for the respective task. The difficulty in applying pooling methods is the selection of these groups of nodes. Existing methods provide different criteria by which these nodes can be selected. Yet these are often prohibitively rigid and also not designed with over-smoothing and over-squashing in mind.

Our work addresses these crucial points. The curvature of a graph has been identified to be a meaningful metric for locating structures responsible for over-squashing and over-smoothing [36]. The curvature between two nodes describes the geodesic dispersion of edges starting at these nodes. Based on this metric, we design CurvPool, a novel pooling method that clusters nodes based on a flexible property of the graph topology. By design, the resulting graph has a suitable structure that alleviates the detrimental effects of over-squashing and over-smoothing. Our empirical results on several benchmark datasets for graph classification confirm the effectiveness of our approach. In addition, CurvPool is theoretically and practically efficient to execute.

2 Preliminaries

Notation We consider graphs of the form $G = (\mathcal{V}, \mathcal{E})$ consisting of a set of $n = |\mathcal{V}|$ nodes $\mathcal{V} = \{v_1, \dots, v_n\}$ and edges indicating whether pairs of nodes are connected. For each node v_i, the set of neighboring nodes is denoted by $\mathcal{N}_i$ and its degree by $d_i = |\mathcal{N}_i|$. The graph signal $\mathbf{X} \in \mathbb{R}^{n \times d}$ consists of d features at each node. We consider the task of graph classification, which aims to find a suitable mapping $f_\theta(\mathbf{X}, G) = c$ predicting class likelihoods $\mathbf{c}$ for the entire graph using some parameters θ.

2.1 Graph Neural Networks

Graph neural networks operate on graph-structured data and are designed to extract meaningful node representations. These are structured as layer-wise functions to update the node representation

$$\mathbf{h}_i^{k+1} = \psi(\mathbf{h}_i^k, \phi(\{\mathbf{h}_j^k \mid j \in \mathcal{N}_i\})) \tag{1}$$

in each layer k using some neighbor aggregation function ϕ and some combination function ψ. The graph signal is used for the initial node representations $\mathbf{h}_i^0 = \mathbf{x}_i^0$. Many options for realizing the update functions have been proposed [17].

However, most methods suffer from two phenomena known as over-smoothing [8] and over-squashing [1,36]. Over-smoothing refers to the case that node representations become too similar to carry meaningful information after a few iterations. Over-squashing occurs when exponentially much information is compressed into the representation of a few nodes, preventing information from flowing between distant nodes. This is induced by the structure of the graph as so-called bottlenecks cause over-squashing, which means that two parts of the graph are connected by relatively few edges.

2.2 Pooling within GNNs

A pooling operation reduces the spatial size of the data by aggregating nodes and their representations using some criterion. It results in a new graph $G' = (\mathcal{V}', \mathcal{E}')$ and new node representations $\mathbf{H}'$ with $|\mathcal{V}'| \leq |\mathcal{V}|$. Pooling methods offer various advantages which frequently include an increased memory efficiency and improved expressivity regarding the graph isomorphism problem [3].

Formally, each pool $p_i \subset \mathcal{V}$ contains a subset of nodes, and our goal is to find a suitable complete pooling

$$\mathcal{P} = \{p_i \subset V \mid p_1 \cup \ldots \cup p_n = V\} \tag{2}$$

so that every node of the graph is contained in at least one of the pools. This guarantees that no information that was contained in the initial graph is disregarded. The new set of nodes $\mathcal{V}'$ is given by turning each pool p_i into a new node v_i'. The new set of edges $\mathcal{E}'$ differs between pooling methods. The main challenge towards successful pooling operations within GNNs is finding a suitable pooling criterion $\mathcal{P}$.

2.3 The Curvature of a Graph

Motivated by the Ricci curvature in Riemannian geometry, a recent investigation defined the curvature of a graph determines as the geodesic dispersion of edges starting at two adjacent nodes [36]. Two edges starting at adjacent nodes can meet at a third node, remain parallel, or increase the distance between the endpoints of the edges. Corresponding to these three cases and based on insights from previous edge-based curvatures [13,28,29], they propose the Balanced Forman curvature:

Definition 1. *(Balanced Forman curvature [36].) For any edge (i, j) in a simple, unweighted graph G, we let BFC(i,j) = 0 if $\min\{d_i, d_j\} = 1$ and otherwise*

$$BFC(i,j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\triangle(i,j)|}{\max\{d_i,d_j\}} + \frac{|\triangle(i,j)|}{\min\{d_i,d_j\}} + \frac{\gamma_{max}^{-1}(i,j)}{\max\{d_i,d_j\}}(|\square^i| + |\square^j|) \tag{3}$$

where $\triangle(i,j)$ are the 3-cycles containing edge (i, j), $\square^i(i,j)$ are the neighbors of i forming 4-cycles containing (i, j) without containing a 3-cycle. $\gamma_{max}(i, j)$ is the maximal number of 4-cycles containing (i, j) traversing a common node.

We refer to Topping et al. [36] for a comprehensive definition. This formulation satisfies the desired properties of the geodesic dispersion. The curvature is negative when (i, j) are sparsely connected and positive curvature when redundant paths are available. We provide visualized examples in Figure 1.

The important relationship to over-squashing is that edges with a negative curvature are considered to be the bottlenecks of the graph [36].

3 Related Work

Various methods for pooling within graph neural networks based on clusters of nodes have been proposed [44,46,32,18,37,14,9,20,25,47,10,19,48]. Some are based on the topology of the graph, while others are based on the node representations themselves.

DiffPool [43] is one of the most frequently employed strategies for pooling based on representations. For each node, it predicts a soft assignment within a fixed number of clusters allowing the pooling to be optimized with gradient descent. Several other methods similarly learn a mapping from node representations to pools [26,2,16,22]. However, there are two main concerns with this family of strategies. First, the number of clusters is predefined and fixed for all graphs in the considered task. Second, the structure of the graph is only taken into account using node representations, which do not capture all structural properties, as given by their limitations regarding the Weisfeiler-Leman test [24].

To address this, pooling strategies based on the graph topology were proposed. Fey et al. [12] predefine a fixed set of graph structures and pool only these into single nodes. CliquePool [23] combines each clique in the graph. However, these methods rely on fixed structures in the graph and are unable to provide any pooling when the graph structure does not perfectly align. As an example, a graph could consist of densely connected communities, but these are only pooled when they constitute complete cliques. For comparison throughout this work, we will use one of both categories, namely DiffPool and CliquePool.

4 CurvPool

We aim to construct an adaptive pooling method that can combine arbitrary structures in the graph without explicitly needing the knowledge of which structures we are interested in. In addition, the structure of the pooled graph should also be more resilient to over-smoothing and over-squashing, which then allows for a better flow of information. Using the curvature of the graph as the foundation for our pooling method allows us to achieve these properties.

4.1 Pooling based on the Curvature of a Graph

Since we base our new pooling approach on the Balanced Forman curvature BFC$(\cdot)$, we initially calculate the curvature for every edge $(i, j) \in E$. We then need to convert curvature values of edges to sets of nodes we want to pool together. These values are used to decide if nodes i and j will be assigned to the same pool or not. There are different approaches for making this decision that lead to variations of CurvPool. In the general case, we use some criterion $f(\mathrm{BFC}(i, j))$ on the curvature of each edge to decide whether two nodes are combined. For the initial candidates for pools, this results in a set

$$\mathcal{P}' = \{\{i, j\} \mid (i, j) \in \mathcal{E} \wedge f(\mathrm{BFC}(i, j))\} \cup \{\{i\} \in \mathcal{V}\} \tag{4}$$

that we augment by each node as additional pool candidates to fulfill our requirements for a complete pooling. We describe our choices for the criterion f in the next section. The main challenge arises when combining subsets of $\mathcal{P}'$. Nodes may be contained in multiple pools, as multiple edges of a node may satisfy our criterion for combination. This does not contradict our definition of a pooling but still should be considered since it significantly impacts the resulting graph. Intuitively, we want groups of nodes that are connected by edges of similar curvatures to be combined together. In this way, clusters of densely connected structures can be aggregated into a single node, and sparsely connected regions will be closer connected afterward. The authors of CliquePool chose a different approach and removed duplicate nodes from every non-largest pool they were contained in [23]. This approach doesn't really suit CurvPool since all resulting pools after the initial selection are of the same size. Instead, we merge all pools whose intersections are non-empty, resulting in the final pooling

$$\begin{aligned}\mathcal{P} = \{ \bigcup_{p_i \in S} p_i \mid S \subseteq \mathcal{P}', \forall T \subset S : \left(\bigcup_{p_i \in T} p_i \right) \cap \left(\bigcup_{p_j \in S \setminus T} p_j \right) \neq \emptyset, \\ \forall p_k \in \mathcal{P}' : \exists p_i \in S : p_i \cap p_k \neq \emptyset \Rightarrow p_k \in S\}. \end{aligned} \tag{5}$$

Each element in $\mathcal{P}$ is then mapped to a new node in the pooled graph. Based on the previous node representations $\mathbf{H} \in \mathbb{R}^{n \times g}$ of $\mathcal{V}$ and an aggregation function ω, we construct new node representations

$$\mathbf{h}'_j = \omega(\{\mathbf{h}_i | i \in p_j\}) \tag{6}$$

for each pool $p_j \in \mathcal{P}$. Any aggregation scheme ω can be used to calculate the node features of the resulting pools. We consider the mean (AVG), the sum (SUM), and the maximum (MAX) operators.

This still leaves us with one final question. How is the new set of edges calculated? Since we strive to retain as much of the initial graph structure as possible, we simply remap the old edges from their respective nodes to the new pools they are contained in, resulting in

$$\mathcal{E}' = \{(m,n) \mid \exists\, p_i, p_j \in \mathcal{P} \colon p_i \neq p_j \wedge m \in p_i \wedge n \in p_j\}\,. \tag{7}$$

This same method is used for all considered variations of CurvPool, leaving only the strategy for which curvatures to use for pooling open.

4.2 Curvature-based Strategies for Pooling

We now present our considered strategies for choosing pairs of nodes for our initial pools. Each of the three strategies has slightly differing motivations and carries its own set of advantages and disadvantages.

HighCurvPool The fundamental idea of HighCurvPool is to aggregate nodes that are adjacent to edges with high curvature. This strategy combines all nodes that are connected by an edge with a curvature above a fixed threshold t_{high}. Our initial set of pools

$$\mathcal{P}'_{highCurv} = \{\{i,j\} \mid \forall i,j \in V : BFC(i,j) > t_{\text{high}}\} \cup \{\{i\} \in \mathcal{V}\} \tag{8}$$

considers exactly these, for which overlapping sets will be merged as described in Section 4.1. Nodes are combined along the nodes in dense communities of the graph. As over-smoothing was shown to occur faster in dense communities [41], these sets of nodes already contain similar representations, thereby being redundant when kept as separate nodes. HighCurvPool should alleviate this effect since the most strongly smoothed representations are aggregated, and the new graph contains more diverse neighboring states from each community. The effects of over-squashing should also reduce as the average path lengths become smaller and information from fewer nodes needs to be compressed for connecting edges. While HighCurvPool typically leads to an increase in curvature in bottlenecks, these are not directly removed.

LowCurvPool Analogous to HighCurvPool, LowCurvPool pools nodes that are connected by an edge with low curvature since these directly represent bottlenecks within the graph. Using a different threshold t_{low}, this results in an initial pooling of the form

$$\mathcal{P}'_{lowCurv} = \{\{i,j\} \mid (i,j) \in \mathcal{E} \wedge \mathrm{BFC}(i,j) < t_{\text{low}}\} \cup \{\{i\} \in \mathcal{V}\}. \tag{9}$$

LowCurvPool leads to the removal of exactly those edges that are marked as problematic through the curvature. The aggregation of the two adjacent nodes lets the two separated subgraphs move closer while guaranteeing that all paths through the graph are retained, and no new bottlenecks are created. As a result, the average curvature of the graph rises. Since we assume that the curvature is a good indicator of the over-squashing problem, information will be propagated better through the graph, and over-squashing gets reduced. However, over-smoothing may still be an issue as separate communities of nodes become more closely connected, leading to faster smoothing [41].

MixedCurvPool Finally, MixedCurvPool combines the other approaches. It utilizes both thresholds t_{high} and t_{low}. While one functions as an upper bound, the other works as the lower bound for our initial pooling

$$P'_{mixedCurv} = \{\{i, j\} \mid \forall i, j \in V : BFC(i, j) < t_{low} \vee BFC(i, j) > t_{high}\}. \quad (10)$$

Two nodes are combined along their edge either if the connecting edge represents a bottleneck or they are within the same densely connected community. The idea is that MixedCurvPooll combines the advantages of both approaches to be as effective as possible against over-squashing and over-smoothing. Selecting adequate hyperparameters becomes more important as this approach carries a great risk of simplifying the graph too much and thus losing all the information hidden inside the graph topology that we want to extract in the first place.

4.3 Runtime Complexity

The runtime complexity of CurvPool is mainly given by the complexity of the Balanced Forman curvature. This complexity is $\mathcal{O}(|\mathcal{E}|d^2_{max})$, with d_{max} being the maximum node degree of the graph [36]. The calculation of the pools themselves only has a complexity of $\mathcal{O}(|\mathcal{E}|)$ while the complexity of merging overlapping pools is $\mathcal{O}(2|E|)$. All further operations don't differ between the pooling approaches and thus are not considered further for this comparison. As CurvPool only depends on the graph structure, this step is only executed once before optimization and reused in all settings.

CliquePools complexity is given through the calculation of the cliques via the Bron-Kerbosch algorithm [6] and is $\mathcal{O}(3^{n/3})$ in the worst case for a graph with n nodes [35]. This can frequently be reduced in a practical setting [11].
Since DiffPool requires the calculation of a complete additional GCN its complexity is given by $\mathcal{O}(LN^2F + LNF^2)$ with L being the amount of layers, N being the amount of nodes and F being the amount of features [4].

5 Experiments

To evaluate the effectiveness of our new pooling method, we compare its performance on different benchmark datasets on graph classification to well-established baselines. Our implementation is available online*. We consider datasets that cover diverse graph structures and tasks from different domains. HIV [39] and

* https://gitlab.com/Cedric_Sanders/masterarbeit

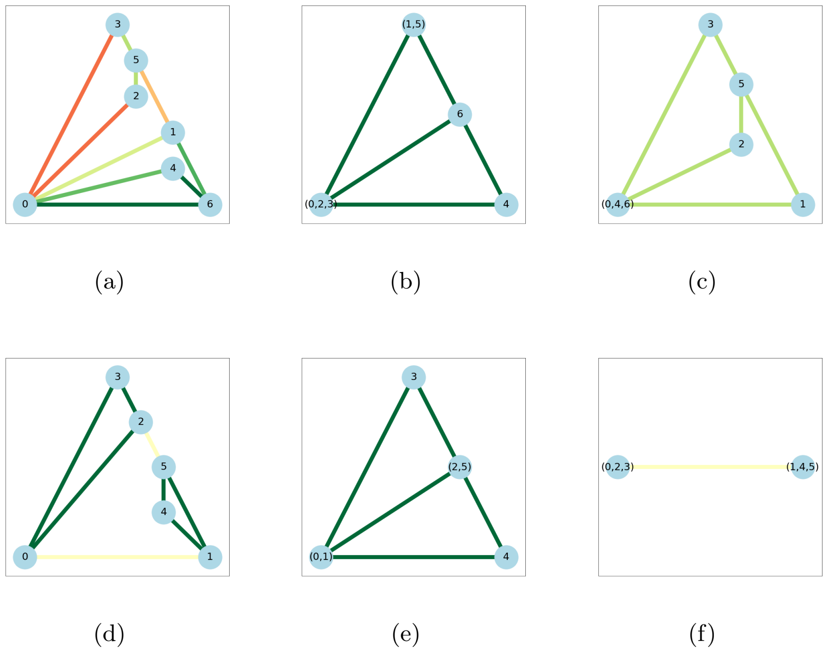


Fig. 1. Example graphs with edges colored according to their curvature from low (red) to high (green). Leftmost are the original graphs while the graphs in the middle represent one step of LowCurvPool and the rightmost graphs represent a step of HighCurvPool.

Proteins [5] are common benchmark datasets that are rooted within biology. They consist of structures of chemical substances that are used to classify the nature or specific properties of these substances. IMDB-BINARY [42] is a common benchmark dataset that contains information about actors and movies. A graph indicates for a specific genre if actors, represented by nodes, have played together in the same movie, represented by edges. The classification task is to predict this genre. In addition, we extend our experiments to a custom dataset containing artificially generated graphs. These are generated using the approach for caveman graphs [38] resulting in multiple dense clique-like areas connected with a small number of edges that represent bottlenecks between these subgraphs. This dataset is referred to as Artificial. This setup allows us to compare different CurvPool variations effectively as the distribution of high and low curvature areas within the graph is smooth. The used features are equivalent to the node degrees.

Table 1. Test accuracies for the best setting for each dataset and method according to the validation scores. The best score for each dataset is marked in bold, and the second-best score is underlined.

Dataset	GCN	DiffPool	CliquePool	CurvPool		
				Mixed	Low	High
HIV	75.83	76.66	78.33	76.31	<u>78.61</u>	**80.06**
Proteins	65.99	**77.81**	74.54	75.27	75.09	**77.81**
IMDB-BINARY	63.80	69.60	68.39	**70.80**	69.40	**70.80**
Artificial	73.20	73.80	73.40	73.30	**74.10**	<u>74.00</u>

5.1 Experimental setup

The classification loss is calculated via the log-likelihood loss, while the used optimizer is the Adam-Optimizer using a learning rate of 0.001 and a batch size of 32. We employ a 10-fold cross-validation and choose the best setting based on validation accuracy. The best run across all hyperparameters is used to calculate the accuracy for held-out test data for each of the folds. All splits are consistent across models.

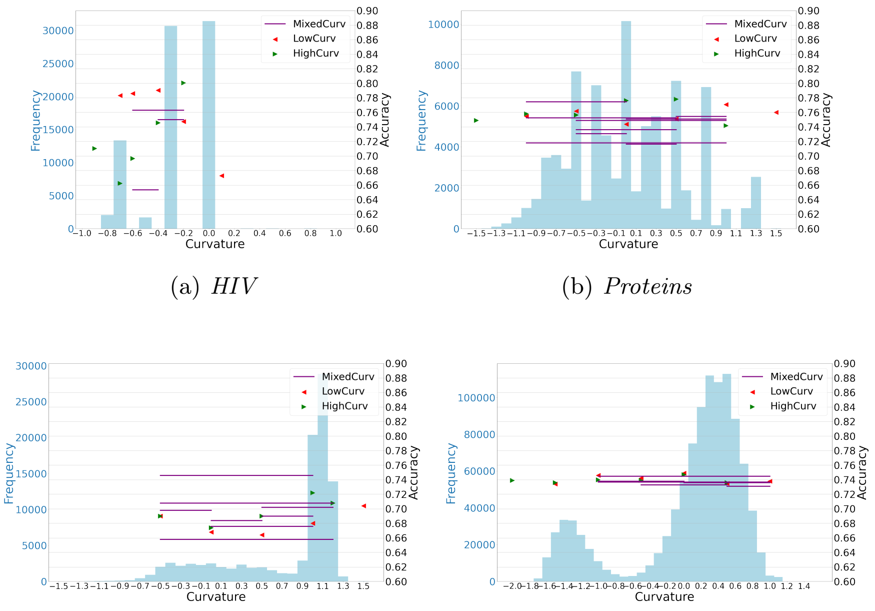
At each scale, our models utilize three convolutional layers, each followed by a ReLU activation and batch normalization [15]. The complete model consists of three of these blocks with the corresponding pooling layers in between. Since we focus on graph classification, a global mean pooling layer and two linear layers are used to calculate the final classification. To keep them as comparable as possible, the pooling approaches only differ in the used pooling operation.

5.2 Results

Table 1 presents the overall results for the experiments. It shows the best parameter constellation per dataset and method. The different variations of CurvPool outperform the established methods on almost all of the datasets, albeit usually by only a few percentage points. Only on the Proteins dataset DiffPool can keep up with CurvPool. Especially HighCurvPool outperforms all other considered methods consistently, with the second-best result typically also going to a variation of CurvPool.

5.3 Ablation Study

The next step is to take a closer look at how some of the parameters impact the results of CurvPool. First up are the thresholds. Figure 2 represents the accuracy scores for the different sets of thresholds, including the different CurvPool variations. To better understand the impact of the thresholds the given histogram represents the distribution of curvature values in the corresponding dataset. This kind of visualization also allows for a good comparison of LowCurvPool, HighCurvPool, and MixedCurvPool.



(a) *HIV* (b) *Proteins*

(c) *IMDB-BINARY* (d) *Artificial*

Fig. 2. Accuracy on the datasets for different thresholds. The histogram represents the distribution of curvature values in the dataset. MixedCurv pools all data points not within the range given via the purple line. LowCurv pools all data points to the left of the red triangle. HighCurv pools all data points to the right of the green triangle.

The thresholds aligned with the center of the histogram seem to achieve the highest scores. This implies that aggregating too many nodes and too few nodes both have detrimental effects on the achieved results. Selecting the correct threshold is a balancing act. For new datasets, we would recommend starting with thresholds that split the dataset into two halves since those seem to present a very good baseline for initial experiments.

In terms of the different CurvPool variations, there is no clear winner. While HighCurvPool tends to achieve the highest scores across the board, it seems to be more sensitive to the choices of the threshold. MixedCurvPool is more consistent with fewer outliers in both directions. MixedCurvPool probably does best on datasets where the curvature values are distributed over a larger area. This explains its weak performance on the HIV dataset. Generally speaking, all three variations seem to be competitive, with differing strengths and weaknesses corresponding to individual datasets and thresholds.

A comparison of the different aggregation schemes is presented in Table 2. For CurvPool, summing all node representations in a pool is clearly on top for all the different datasets. Meanwhile, CliquePool tends to achieve the best results

Table 2. A comparison of different aggregation schemes for CliquePool and the best performing 1.

Accuracy	CliquePool			CurvPool		
	Sum	Avg	Max	Sum	Avg	Max
HIV	75.76	**78.33**	73.05	**80.06**	79.44	78.26
Proteins	74.36	**74.54**	72.72	**77.81**	75.45	76.54
IMDB-BINARY	67.40	**68.39**	63.59	**70.80**	69.20	60.20
Artificial	72.80	73.40	**73.60**	**74.10**	69.70	63.70

Table 3. Runtime per method and dataset. Pre represents the duration for the precomputation of the poolings. Epoch is the training time per epoch. All times are in seconds.

Runtime (s)	HIV		Proteins		IMDB-BINARY		Artificial	
	Epoch	Pre	Epoch	Pre	Epoch	Pre	Epoch	Pre
GCN	**1.6**	-	**2.0**	-	0.7	-	**1.0**	-
DiffPool	9.5	-	47.7	-	2.0	-	4.0	-
CliquePool	2.4	**19**	4.0	**15**	**0.5**	**3**	**1.0**	114
CurvPool	3.0	29	5.0	28	**0.5**	9	**1.0**	**100**

when averaging. Especially notable are the differences between summing and averaging on the Artificial dataset. We explain the success of the sum aggregation by its similarity to more expressive message-passing schemes with respect to the Weisefeiler-Leman test. As pools of different sizes typically occur, the sum provides information about the number of nodes utilized in each pool. In contrast, the mean is unable to determine the number of nodes utilized for pooling, similar to its inability to determine the number of neighbors in message-passing operations [40]. Thus, the sum aggregation provides important additional structural features and offers increased expressivity.

5.4 Runtime

The runtimes presented in Table 3 largely align with the established considerations of Section 4.3. CliquePool and CurvPool can utilize the precalculation of the poolings to drastically reduce the runtime per epoch and outperform DiffPool. Between CliquePool and CurvPool, runtimes are very close and in some cases even faster than the basic GCN. This can be explained through the larger number of aggregated nodes and the resulting smaller adjacency matrices. Especially meaningful is the amount of edges since it impacts the calculation of the Balanced Forman curvature negatively. Though this effect should be largely limited to the precalculation and not affect the time per epoch.

6 Conclusion and Future Work

We introduced CurvPool, a novel pooling approach for graph neural networks that are designed to be effective against over-smoothing and over-squashing. It is based on the Balanced Forman curvature, which represents the connectivity between nodes. A high curvature value occurs for densely connected areas of the graph, which are prone to over-smoothing. Bottlenecks are located at edges with a low curvature value, leading to over-squashing. Our proposed methods HighCurvPool and LowCurvPool directly reduce these critical areas by combining exactly these nodes, resulting in a coarser graph and more effective message-passing. Simultaneously reducing both of these areas is done using our proposed MixedCurvPool. The first outstanding quality of CurvPool is its flexibility. The curvature can be calculated for any graph and always leads to a meaningful pooling metric while still being an inherent attribute of the graph itself. Other approaches like CliquePool on the other hand are servery limited through the need for the existence of specific structures within the graph. A graph without cliques of appropriate size will not lead to a suitable pooling. Meanwhile pooling via DiffPool is almost completely independent of the graph structure itself since it uses an external pooling metric in the form of a clustering. This approach also requires additional knowledge about the graph or extensive hyperparameter tuning to select the fitting clique sizes.

Another important factor is the low theoretical and practical runtime complexity of CurvPool which is linear in the number of edges. As the curvature and thus the pooling can be precomputed, the additional time during training is almost negligible.

Our empirical results on several graph classification tasks show the effectiveness of our approach. CurvPool comes out slightly ahead while comparing classification accuracy for a bunch of different datasets. The experiments have also shown the viability of the different CurvPool variations in highlighting their strengths and weaknesses on specific datasets and parameter combinations. We found HighCurvPool using the sum aggregation to accomplish the strongest results consistently. We explain this as sets of nodes connected by edges with high curvature are prone to over-smoothing. These carry redundant features which HighCurvPool then reduces to a single node. The sum aggregation allows our method to utilize structural properties of the graph as the resulting state is influenced by the number of nodes in each pool. In summary, this flexibility, its slightly improved classification accuracy, and its low runtime complexity make CurvPool a valuable alternative to the established pooling methods.

Limitations Limitations of CurvPool directly stem from limitations in the Balanced Forman curvature itself. While this strategy is very flexible, it may not perfectly align with the nodes that should be pooled in the optimal scenario. However, in case other strategies for ranking pairs of nodes emerge, these can be directly integrated into our method. CurvPool also does not consider node features, which might further enhance its effectiveness, albeit reducing its ability to precompute clusters. Additionally, while our empirical results already cover diverse datasets, CurvPool can be evaluated for additional tasks and against other methods to ensure its generalizability.

Future work Our work opens up several directions for future work. While this paper focused on graph classification, it could be extended to node classification tasks in order to combine distant information. During our work, we noticed that the current theory on over-smoothing and over-squashing is unfit for pooled graphs. Metrics like the Dirichlet energy are not designed for pooling operations, making it challenging to quantify whether a pooling step can reduce over-smoothing. Thus, novel metrics and theoretical investigations are needed. Similarly, the effect of pooling methods in general on the curvature needs to be better understood from a theoretical perspective.

Acknowledgements This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence and by the Federal Ministry of Education and Research of Germany under grant no. 01IS22094E WEST-AI.

References

1. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. arXiv preprint arXiv:2006.05205 (2020)
2. Bianchi, F.M., Grattarola, D., Alippi, C.: Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning. pp. 874–883. PMLR (2020)
3. Bianchi, F.M., Lachi, V.: The expressive power of pooling in graph neural networks. arXiv preprint arXiv:2304.01575 (2023)
4. Blakely, D., Lanchantin, J., Qi, Y.: Time and space complexity of graph convolutional networks. Accessed on: Dec **31** (2021)
5. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21**, i47–i56 (06 2005). https://doi.org/10.1093/bioinformatics/bti1007
6. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM **16**(9), 575–577 (1973)
7. Cai, C., Wang, Y.: A note on over-smoothing for graph neural networks. arXiv preprint arXiv:2006.13318 (2020)
8. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 3438–3445 (2020)
9. Diehl, F., Brunner, T., Le, M.T., Knoll, A.: Towards graph pooling by edge contraction. In: ICML 2019 workshop on learning and reasoning with graph-structured data (2019)
10. Du, J., Wang, S., Miao, H., Zhang, J.: Multi-channel pooling graph neural networks. In: IJCAI. pp. 1442–1448 (2021)

11. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in large sparse real-world graphs. Journal of Experimental Algorithmics (JEA) **18**, 3–1 (2013)
12. Fey, M., Yuen, J.G., Weichert, F.: Hierarchical inter-message passing for learning on molecular graphs. arXiv preprint arXiv:2006.12179 (2020)
13. Forman: Bochner's method for cell complexes and combinatorial ricci curvature. Discrete & Computational Geometry **29**, 323–374 (2003)
14. Gao, H., Chen, Y., Ji, S.: Learning graph pooling and hybrid convolutional operations for text representations. In: The world wide web conference. pp. 2743–2749 (2019)
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. pmlr (2015)
16. Khasahmadi, A.H., Hassani, K., Moradi, P., Lee, L., Morris, Q.: Memory-based graph networks. arXiv preprint arXiv:2002.09518 (2020)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: International conference on machine learning. pp. 3734–3743. PMLR (2019)
19. Lei, F., Liu, X., Jiang, J., Liao, L., Cai, J., Zhao, H.: Graph convolutional networks with higher-order pooling for semisupervised node classification. Concurrency and Computation: Practice and Experience **34**(16), e5695 (2022)
20. Li, J., Ma, Y., Wang, Y., Aggarwal, C., Wang, C.D., Tang, J.: Graph pooling with representativeness. In: 2020 IEEE International Conference on Data Mining (ICDM). pp. 302–311. IEEE (2020)
21. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
22. Liu, N., Jian, S., Li, D., Zhang, Y., Lai, Z., Xu, H.: Hierarchical adaptive pooling by capturing high-order dependency for graph representation learning. IEEE Transactions on Knowledge and Data Engineering (2021)
23. Luzhnica, E., Day, B., Lio, P.: Clique pooling for graph classification. arXiv preprint arXiv:1904.00374 (2019)
24. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 4602–4609 (2019). https://doi.org/10.1609/aaai.v33i01.33014602
25. Nguyen, T., Grishman, R.: Graph convolutional networks with argument-aware pooling for event detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
26. Noutahi, E., Beaini, D., Horwood, J., Giguère, S., Tossou, P.: Towards interpretable sparse graph representation learning with laplacian pooling. arXiv preprint arXiv:1905.11577 (2019)
27. Nt, H., Maehara, T.: Revisiting graph neural networks: All we have is low-pass filters. arXiv preprint arXiv:1905.09550 (2019)
28. Ollivier, Y.: Ricci curvature of metric spaces. Comptes Rendus Mathematique **345**(11), 643–646 (2007)
29. Ollivier, Y.: Ricci curvature of markov chains on metric spaces. Journal of Functional Analysis **256**(3), 810–864 (2009)
30. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947 (2019)

31. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. arXiv preprint arXiv:1802.04407 (2018)
32. Ranjan, E., Sanyal, S., Talukdar, P.: Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5470–5477 (2020)
33. Roth, A., Liebig, T.: Forecasting unobserved node states with spatio-temporal graph neural networks. In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 740–747. IEEE (2022)
34. Roth, A., Liebig, T.: Transforming pagerank into an infinite-depth graph neural network. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 469–484. Springer (2022)
35. Tomita, E., Tanaka, A., Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical computer science **363**(1), 28–42 (2006)
36. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021)
37. Wang, Y.G., Li, M., Ma, Z., Montufar, G., Zhuang, X., Fan, Y.: Haar graph pooling. In: International conference on machine learning. pp. 9952–9962. PMLR (2020)
38. Watts, D.J.: Networks, dynamics, and the small-world phenomenon. American Journal of sociology **105**(2), 493–527 (1999)
39. Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.: Moleculenet: a benchmark for molecular machine learning. Chemical science **9**(2), 513–530 (2018)
40. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
41. Yan, Y., Hashemi, M., Swersky, K., Yang, Y., Koutra, D.: Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In: 2022 IEEE International Conference on Data Mining (ICDM). pp. 1287–1292. IEEE (2022)
42. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1365–1374 (2015)
43. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems **31** (2018)
44. Yuan, H., Ji, S.: Structpool: Structured graph pooling via conditional random fields. In: Proceedings of the 8th International Conference on Learning Representations (2020)
45. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
46. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
47. Zhang, Y.D., Satapathy, S.C., Guttery, D.S., Górriz, J.M., Wang, S.H.: Improved breast cancer classification through combining graph convolutional network and convolutional neural network. Information Processing & Management **58**(2), 102439 (2021)
48. Zhao, H., Xie, J., Wang, H.: Graph convolutional network based on multi-head pooling for short text classification. IEEE Access **10**, 11947–11956 (2022)