Graph Neural Networks Need Cluster-Normalize-Activate Modules

Arseny Skryagin[1] **Felix Divo**[1] **Mohammad Amin Ali**[1]
Devendra Singh Dhami[2] **Kristian Kersting**[1,3,4,5]

[1]AI & ML Group, TU Darmstadt [2]TU Eindhoven [3]Hessian Center for AI (hessian.AI)
[4]German Research Center for AI (DFKI) [5]Centre for Cognitive Science, TU Darmstadt
{arseny.skryagin,felix.divo,kersting}@cs.tu-darmstadt.de
amin.ali@stud.tu-darmstadt.de d.s.dhami@tue.nl

Abstract

Graph Neural Networks (GNNs) are non-Euclidean deep learning models for graph-structured data. Despite their successful and diverse applications, oversmoothing prohibits deep architectures due to node features converging to a single fixed point. This severely limits their potential to solve complex tasks. To counteract this tendency, we propose a plug-and-play module consisting of three steps: Cluster $\rightarrow$ Normalize $\rightarrow$ Activate (CNA). By applying CNA modules, GNNs search and form super nodes in each layer, which are normalized and activated individually. We demonstrate in node classification and property prediction tasks that CNA significantly improves the accuracy over the state-of-the-art. Particularly, CNA reaches 94.18% and 95.75% accuracy on Cora and CiteSeer, respectively. It further benefits GNNs in regression tasks as well, reducing the mean squared error compared to all baselines. At the same time, GNNs with CNA require substantially fewer learnable parameters than competing architectures.

1 Introduction

Graph Neural Networks (GNNs) are a promising approach to leveraging the full extent of the geometric properties of various types of data in many different key domains [Zhou et al., 2020a, Bronstein et al., 2021, Waikhom and Patgiri, 2023]. For instance, they are used to predict the stability of molecules [Wang et al., 2023], aid in drug discovery [Askr et al., 2023], recommend new contacts in social networks [Zhang and Chen, 2018], identify weak points in electrical power grids [Nauck et al., 2022], predict traffic volumes in cities [Jiang and Luo, 2022], and much more [Waikhom and Patgiri, 2023]. To solve such tasks, one typically uses message-passing GNNs, where information from nodes is propagated along outgoing edges to their neighbors, where it is aggregated and then projected by a learned non-linear function. Increasing the expressivity of GNNs is crucial to learning more complex relationships and eventually improving their utility in a plethora of applications.

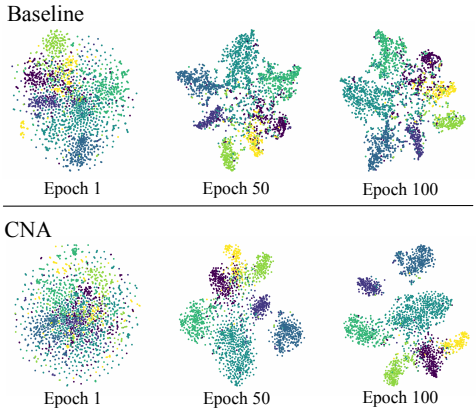


Figure 1: **Evolution of node embeddings for the Cora dataset**. The colors indicate the membership of one of the seven target classes.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-2535

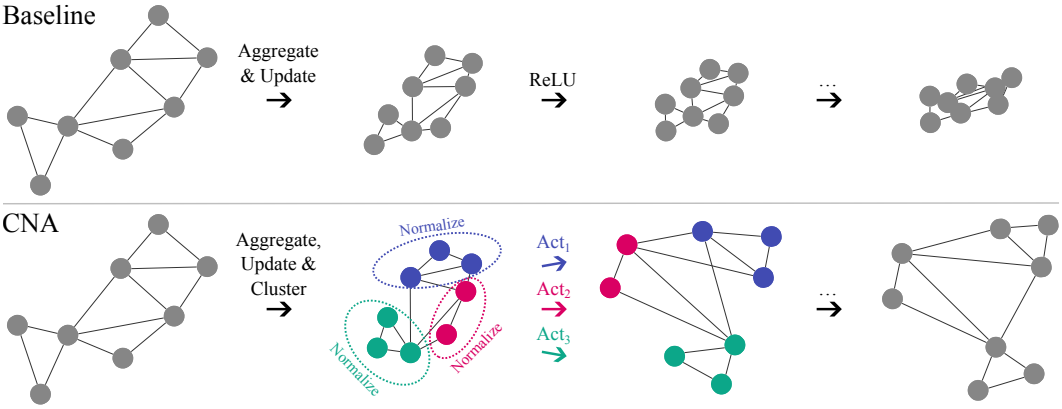


Figure 2: **CNA replaces the activation function in each iteration of any GNN architecture.** When employing classical activations like ReLU to all nodes undifferentiatedly, we observe oversmoothing. With CNA, we cluster the node features and then normalize and project them with a separate learned activation function each, effectively increasing their expressiveness even in deeper networks.

A natural approach to increasing expressivity is to increase depth, effectively enabling further-reaching and higher-level patterns to be captured. This combats under-reaching, where information cannot propagate far enough. For example, this limits the effective radius of information on road crossings in traffic prediction tasks, where information on specific bottlenecks in road networks cannot propagate to the relevant k-hop neighbors. In practice, one wants to increase the depth of the employed GNNs. However, this soon triggers a phenomenon called *oversmoothing*, where node features are converging more and more to a common fix-point with an increasing number of layers [NT and Maehara, 2019, Rusch et al., 2023a]. For example, in the specific task of node classification, node features of different classes become increasingly overlapping and, thus, essentially indistinguishable. There are many attempts to prevent this issue from occurring. Among them is Gradient-Gating (G^2), which gates updates to nodes once features start converging [Rusch et al., 2023b]. However, G^2 adaptively chokes message passing in each node right before oversmoothing can occur, effectively reducing the functionality of deeper GNN layers to an identity mapping. This idea of adaptively controlling the flow of information in each node is still a very promising approach. But, instead of regulating message passing, we propose learning an adaptive node feature update. We argue that it is crucial to ensure that while the node features are iteratively exchanged, aggregated, and projected, they stay sufficiently different from each other to solve the eventual task, like classification or regression. This has the benefit of maintaining effective information propagation even in deeper layers. Figure 1 visualizes the final node features during training, showing how our method improves the separation of the learned classes over the oversmoothed baseline.

To ensure sufficiently distinct nodes, we present Cluster $\rightarrow$ Normalize $\rightarrow$ Activate (CNA) modules,[1] specifically designed to improve the expressivity of GNNs:

- **Cluster** – Transformation of the node features should be shared and yet differ at the same time. For this reason, our first inductive bias is to assume several groups of nodes with shared properties.
- **Normalize** – Stabilization of training in deep architectures, including Transformers [Vaswani et al., 2017], is typically provided by normalization. By employing normalization, CNA effectively maintains beneficial numerical ranges and combats collapse tendencies.
- **Activate** – To preserve distinct representations, the clusters must be transformed individually. By introducing learnable activation functions, we learn separate projections for each of them. This generalizes the typical affine transformation following the normalization to a general learned function that can better adjust to the specific node features.

We use rational Activations [Molina et al., 2019, Delfosse et al., 2024] as powerful yet efficient point-wise non-linearities. The complete procedure is shown in Figure 2.

[1]Code available at `https://github.com/ml-research/cna_modules`.

CNA modules can also be viewed as adding additional hierarchical structure to the problem: By grouping nodes into clusters of similar representations, we effectively introduce super-nodes with different non-linear activation functions. Each of their constituents shares the same activation function yet has distinct node property vectors and neighbors. Moreover, the node features in each super-node are less varied since the members of the clusters share some common characteristics. This divide-and-conquer approach breaks up the challenging task of transforming the node features into many smaller ones.

The presented work introduces the novel CNA modules which limit oversmoothing and thereby improve performance. They allow for many advancements, delivering better performance compared to the state-of-the-art in many tasks and datasets. In summary, we make the following contributions:

(i) We introduce the plug-and-play CNA modules for more expressive GNNs and motivate their construction.

(ii) We show that they empirically allow training much deeper GNNs.

(iii) Our experiments demonstrate the effectiveness of CNA in diverse node and graph-level classification, node-property prediction, and regression tasks.

(iv) Lastly, we show that architectures with CNA are parsimonious, achieving better performance than the state-of-the-art with fewer parameters.

We proceed as follows: We next relate our work to the existing research on GNNs and their specific challenges (Section 2). We then describe and discuss our proposed solution CNA (Section 3) and conduct a comprehensive evaluation in different scenarios (Section 4). Finally, we conclude and suggest promising next steps for further improving the expressiveness of GNNs (Section 5).

2 Related Work

Machine Learning on Graphs and its Challenges. Machine learning on graphs has a long history, with graph neural networks as their more recent incarnations [Gori et al., 2005, Scarselli et al., 2008]. Since then, several new models like Graph Convolutional Networks (GCN) [Kipf and Welling, 2016], Graph Attention Networks (GAT) [Veličković et al., 2018], and GraphSAGE [Hamilton et al., 2017] have been proposed. Gilmer et al. [2017] then unified them into the Message Passing Neural Networks (MPNNs) framework, the most common type of GNNs [Battaglia et al., 2018]. In addition to the typical machine learning pitfalls like overfitting and computationally demanding hyperparameter optimization, MPNNs pose some specific challenges: *oversquasching* is the effect of bottlenecks in the graph's topology, limiting the amount of information that can pass through specific nodes [Alon and Yahav, 2020, Topping et al., 2021]. The other widely studied challenge is *oversmoothing*, where the node features converge to a common fixed point with increasing depth of the MPNN [Li et al., 2018, NT and Maehara, 2019, Rusch et al., 2023a]. This essentially equates to the layers performing low-pass filtering, which is harmful to solving the problem beyond some point. This phenomenon has also been studied in the context of Transformers [Vaswani et al., 2017], where repeated self-attention acts similarly to an MPNN on a fully connected graph [Shi et al., 2021a]. Different metrics have since been proposed to measure oversmoothing: cosine similarity, Dirichlet energy, and mean average distance (MAD). Rusch et al. [2023a] organize the existing mitigation approaches into three main groups. First, as discussed in more detail in the next paragraph, normalization and regularization are beneficial and are also performed by our CNA modules. Second, one can change the propagation dynamics, as done by GraphCON [McCallum et al., 2000], Gradient Gating [Rusch et al., 2023b], and RevGNN [Li et al., 2021]. Finally, residual connections can alleviate some of the effects but cannot entirely overcome them. Solving these challenges is an open task in machine learning on graphs.

Normalization in Deep Learning. In almost all deep learning methods in the many subfields, normalizations have been studied extensively. They are used to improve the training characteristics of neural networks, making them faster to train and better at generalizing [Huang et al., 2023]. The same applies to GNNs, where normalization plays a key role [Zhou et al., 2020a, Cai et al., 2021, Chen et al., 2022, Rusch et al., 2023a]. However, selecting the correct reference group to normalize jointly is key. For example, a learnable grouping is employed in Deep Group Normalization (DGN), where normalization is performed within each cluster separately [Zhou et al., 2020b]. The employed soft clustering of DGN is only of limited suitability to fostering distinct representations of the node

 https://doi.org/10.52202/079017-2535

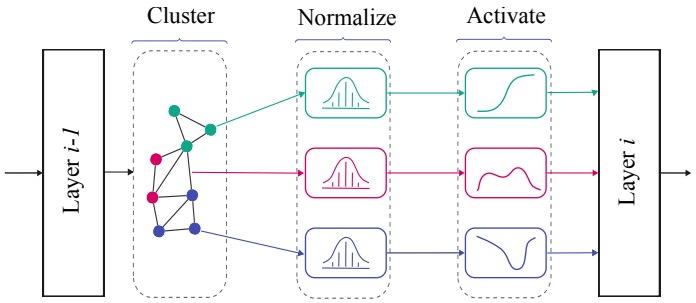


Figure 3: **The components of CNA modules:** They cluster node features without changing the adjacency matrix, normalize them separately, and finally activate with distinct learned functions.

features. Instead, we argue that simple hard clustering, for example, provided by the classic k-means algorithm, is sufficient and more desirable. Zhao and Akoglu [2020] suggest PairNorm, where layerwise normalization ensures a constant total pairwise squared distance of node features. Instead of adjusting the node features against collapse, Caso et al. [2023] rewire the topology based on clusters of node features. For the case of multi-graph datasets, Cai et al. [2021] provide a good overview of existing approaches, and argue that normalization shall be performed per graph.

Learnable Activation Functions. Using non-polynomial activation functions is crucial for neural networks to be universal function approximators [Leshno et al., 1993]. While most works use rectified-based functions like ReLU, GeLU, SiLU, etc., there are also attempts at learning some limited shape parameters as in PReLU or Swish [Apicella et al., 2021]. There has since been further work on learnable activations with reduced flexibility, namely LEAFs [Bodyanskiy and Kostiuk, 2023], a combination of polynomials and exponentials. However, one can even learn the overall shape of the activations, as demonstrated by rational activation functions [Molina et al., 2019, Boulle et al., 2020, Trimmel et al., 2022]. They have proven to be very helpful in a diverse set of applications, in particular, due to their inherently high degree of plasticity during training [Delfosse et al., 2024]. More importantly, rationals are smoothly differentiable universal function approximators [Molina et al., 2019, Telgarsky, 2017], for which reason we select them as flexible activation functions for CNA. Furthermore, changing the activation function has been found beneficial against oversmooting by Kelesis et al. [2023] too, which increased the slope of the classic ReLU activation to reduce oversmoothing in MPNNs. This further motivates taking a closer look at activations such as done by Khalife and Basu [2024] and in this work.

3 Cluster-Normalize-Activate Modules

This section will formally define CNA modules and discuss their design. Adaptive control of the information flow is a promising approach to limit oversmoothing in GNNs. We, therefore, propose learning an adaptive node feature update, ensuring distinct node feature representations during the iterative exchange, aggregation, and projection. This benefits the maintenance of effective information propagation in deeper layers. We start by introducing the notation used throughout this work, proceed to recall message-passing GNNs, and finally highlight the three main components of CNA. The overall module is shown in Figure 3.

Notation. We consider undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are unordered pairs $\{i, j\}$ of nodes $i, j \in \mathcal{V}$. The set of neighbors of a node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} | \{i, j\} \in \mathcal{E}\} \subseteq \mathcal{V}$. We additionally identify each node $i \in \mathcal{V}$ with a feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$. Together, these form the feature matrix $\boldsymbol{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$, where each column represents the features of a single node. Similarly, depending on whether we model a node-level classification, property prediction, or regression task, we have corresponding target vectors $\boldsymbol{y}_i \in \mathbb{R}^t$, with the special case of $t = 1$ for classification. The target matrix for all nodes is $\boldsymbol{Y} \in \mathbb{R}^{t \times |\mathcal{V}|}$ or a vector for graph-level.

Message-Passing Neural Networks (MPNNs). The most prevalent type of GNNs are MPNNs, with GCN, GAT, and GraphSAGE as their best-known representatives. They iteratively transform a graph by a sequence of L layers $\phi = \phi_L \circ \cdots \circ \phi_1$, with $\boldsymbol{Y} = \phi(\mathcal{G}, \boldsymbol{X})$ [Zhou et al., 2020a, Lachaud et al., 2022]. In each layer ℓ, two steps of computation are performed. First, the node

features $\boldsymbol{h}_j^{(l)}$ of the neighbors $j \in \mathcal{N}_i$ of each node $i \in \mathcal{V}$ are aggregated into a single vector $\hat{\boldsymbol{h}}_i^{(\ell)} = \mathrm{Aggregate}(\{\{\boldsymbol{h}_j^{(l)} \mid j \in \mathcal{N}_i\}\})$. Importantly, the Aggregate operation must be invariant to permutations of the neighbors. Popular choices include the point-wise summation or averaging of feature vectors across all neighbors of a node. Second, these features $\hat{\boldsymbol{h}}_i^{(\ell)}$ are projected jointly with the previous node features, as $\boldsymbol{h}_i^{(\ell+1)} = \mathrm{Update}(\boldsymbol{h}_i^{(l)},\ \hat{\boldsymbol{h}}_i^{(\ell)})$. The resulting node features $\boldsymbol{h}_i^{(\ell+1)}$ then form the input to the next layer. Both the Aggregate and the Update steps can be learned, where the latter is often instantiated by Multi-layer Perceptrons (MLPs). Note that the features of the very first layer are simply the node features $\boldsymbol{h}^{(1)} = \boldsymbol{X}$, and the resulting last hidden representation is our target output: $\boldsymbol{h}^{(L)} = \boldsymbol{Y}$.

We propose improving the Update-step to elevate the effectiveness of the overall architecture. Usually, the learned projection ends with a non-linear activation, like ReLU. Instead, we propose performing the three steps of CNA, which we will outline below. We want to emphasize that our general recipe is applicable to any MPNN following the above structure.

3.1 Step 1: Cluster

The node features of typical graph datasets can be clustered into groups of similar properties. In the case of classification problems, a reasonable clustering would at least partially recover class membership. Note that this unsupervised procedure does not require labels and is applicable to a wide range of tasks. So, even in regression tasks, the target output for each node will usually differ; therefore, partitioning nodes into groups of similar patterns is advantageous, too. We, therefore, cluster the nodes by their features $\boldsymbol{x}_i$ to obtain K groups $\mathcal{C}_1, \ldots, \mathcal{C}_K$ at the end of each Update-step. This separation allows us to then normalize representations and learn activation functions that are specific to the characteristics of these subsets of nodes. It is important to note that the geometry, i.e., the arrangement of edges between nodes, does not change in the progression through GNN layers, while the features associated with each node do. Likewise, cluster membership does not necessarily indicate node adjacency and thus allows learning on heterophilic data as well. Note that this approach is, therefore, distinct from the graph partitioning often performed to shard processing of graphs based on its geometry [Chiang et al., 2019].

In principle, any clustering algorithm yielding a fixed number of clusters K can be used to group the node features. Popular choices include the classic k-means [MacQueen, 1967] and Gaussian Mixture Model (GMM) algorithms [Bishop, 2006], which estimate spherical and elliptical clusters, respectively. However, we need to pay attention to the computational costs of such operations. Typical definitions of k-means run in $\mathcal{O}(|\mathcal{V}|Kd)$ per iteration [Manning et al., 2009]. Expectation-maximization can be used to learn GMM clusters in $\mathcal{O}(|\mathcal{V}|Kd^2)$ per iteration [Moore, 1998]. We found that the more expensive execution of GMMs did not materialize in substantial improvements in downstream tasks. We, therefore, opted to use a fast implementation of k-means. This confirms that k-means often provides decent clustering in practical settings and is sufficiently stable [Ben-David et al., 2007]. In our work, we compared nodes by their Euclidean distance, which we found to work reliably in our experiments. However, CNA permits the flexible use of different and even domain-specific data distances.

3.2 Step 2: Normalize

To ensure even scaling of the data across layers, we perform normalization per cluster $\mathcal{C}_k$ and per feature j across all nodes $i \in \mathcal{C}_k$ separately:

$$\widetilde{x}_{ij} = \frac{x_{ij} - \mu_{kj}}{\sqrt{\sigma_{kj}^2 + \epsilon}}, \quad \mu_{kj} = \frac{1}{|\mathcal{C}_k|} \sum_{p \in \mathcal{C}_k} x_{pj}, \quad \sigma_{kj}^2 = \frac{1}{|\mathcal{C}_k|} \sum_{p \in \mathcal{C}_k} \left(x_{pj} - \mu_{kj}\right)^2, \tag{1}$$

where ϵ is introduced for numerical stability. We want to emphasize that this step is similar to Instance Normalization, yet is nonparametric and does not apply the usual affine transformation to restore the unique cluster representation [Huang et al., 2023]. Similarly, it is not required to scale the mean we subtract as in GraphNorm [Cai et al., 2021]. Instead, we learn a much more powerful transformation in the subsequent Activate step, which subsumes the expressivity of a normal affine projection and thus renders it redundant. The idea of normalizing per cluster $\mathcal{C}_k$ is related to GraphNorm. However, instead of normalizing per graph in the batch, we propose normalizing per cluster within each graph, yet with the same motivation of maintaining the expressivity of the individual node features.

https://doi.org/10.52202/079017-2535

3.3 Step 3: Activate

Using an element-wise non-polynomial activation function is crucial for MLPs to be universal function approximators [Leshno et al., 1993]. To maintain distinct representations of node features at large depths, we employ learnable activation functions. Specifically, we use rational activations [Molina et al., 2019] of degree (m, n):

$$R(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{k=0}^{m} a_k x^k}{1 + |\sum_{k=1}^{n} b_k x^k|}. \tag{2}$$

Their purpose is twofold: Firstly, they act as non-polynomial element-wise projections to increase the representational power of the model. Secondly, they replace and subsume the affine transformation in the typical Instance Normalization formulation. Additionally, their strong adaptability allows for appropriate learnable adjustments in the dynamic learning of deep neural networks. This is in line with the findings of Kelesis et al. [2023], who increased the slope of ReLU activations to combat overfitting. Our rationals subsume their approach by further lifting restrictions on the activation function and tuning the slopes automatically while learning the network.

Removing activation functions from GNN layers altogether can–surprisingly–improve overall performance due to reduced oversmoothing [Wu et al., 2019]. Our CNA modules limit oversmoothing further, maintaining strong representational power even in deeper networks. We will demonstrate this in the next section.

3.4 Theoretical Underpinnings

We first show how previous proofs of the necessary occurrence of oversmoothing in vanilla GNNs are not applicable when CNA is used. Next, we explain why these proofs are not easily reinstated by illustrating how CNA breaks free of the oversmoothing curse.

Previous Theoretical Frameworks The Rational activations of CNA trivially break the assumptions of many formalisms due to their potential unboundedness and not being Lipschitz continuous. This includes Prop. 3.1 of Rusch et al. [2023b], where, however, the core proofs on oversmoothing are deferred to Rusch et al. [2022]. Again, the activation σ is assumed to be point-wise and further narrowed to ReLU in the proof in Appendix C.3. Regarding the more recent work of Nguyen et al. [2023], we again note that CNA violates the assumptions neatly discussed in Appendix A. The CNA module can either be modeled as part of the message function ψ_k or as part of the aggregation $\oplus$. However, in both cases, the proof of Prop. 4.3 (which is restricted to regular graphs) breaks down. In the former case, there appears to be no immediate way to repair the proof of Eq. (15) in Appendix C.3. In the latter case, providing upper bounds in Appendix C.2 is much more difficult.

How CNA Escapes Oversmoothing Restoring the proofs for the occurence of oversmoothing is difficult because CNA was built precisely to break free of the current limitations of GNNs. This can be seen by considering two possible extremes that arise as special cases of CNA. Consider a graph with N nodes. On one end of the spectrum, we can consider CNA with $K = N$ clusters and Rationals that approximate some common, fixed activation, such as ReLU. This renders the normalization step ineffective and exactly recovers the standard MPNN architecture, which is known to be doomed to oversmooth under reasonable assumptions [Rusch et al., 2022, Nguyen et al., 2023]. The same holds with only a single cluster ($K = 1$), i.e., MPNNs with global normalization [Zhou et al., 2020b]. Conversely, we can consider $K = N$ clusters, but now with fixed distinct Rational activations given by $R_i(x) = i$ for each cluster $i \in 1, \ldots, N$. The Dirichlet energy of that output is constant, lower-bounded, and, therefore, does not vanish, no matter the number of layers. In practice, we employ, of course, between $K = 1$ one and $K = N$ clusters and thereby trade off the degree to which the GNN is affected by oversmoothing. The following section will investigate this and other questions empirically.

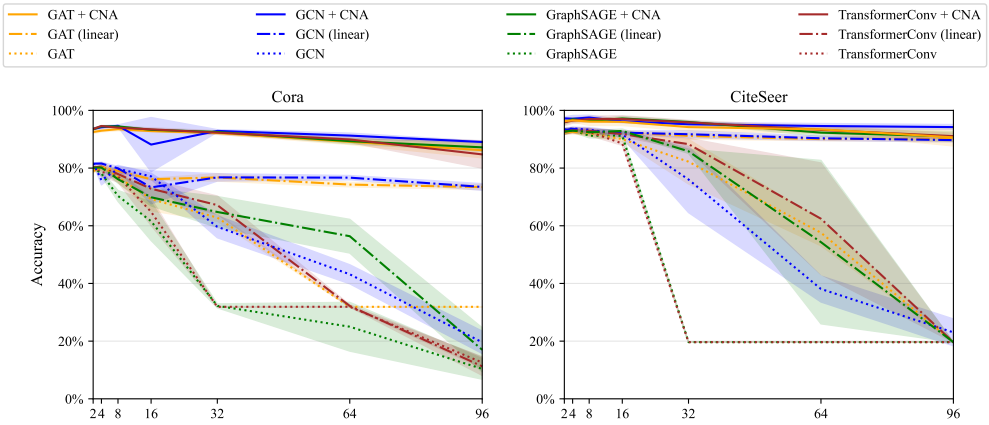


Figure 4: **CNA limits oversmoothing and improves the performance of deep GNNs.**

4 Experiments

To evaluate the effectiveness of CNA with GNNs, we aim to answer the following research questions:

(Q1) Does CNA limit oversmoothing?

(Q2) Does CNA improve the performance in node classification, node regression, and graph classification tasks?

(Q3) Can CNA allow for having fewer parameters while maintaining strong performance when scaling to very large graphs?

(Q4) Model Analysis: How important are each of the three steps in CNA? How do hyperparameters affect the results?

Setup. We implemented CNA based on PyTorch Geometric [Fey and Lenssen, 2019] to answer the above questions. We searched for suitable architectures among Graph Convolutional Network (GCN) [Kipf and Welling, 2016], Graph Attention Network (GAT) [Veličković et al., 2018], Sample and Aggregate (GraphSAGE) [Hamilton et al., 2017], Transformer Convolution (TransformerConv) [Shi et al., 2021b] and Directional GCN (Dir-GNN) [Rossi et al., 2023]. They offer diverse approaches to information aggregation and propagation within graph data, catering to a wide range of application domains and addressing specific challenges inherent to graph-based tasks. Details on the choice of hyperparameters and training settings are provided in Appendix A.2. Average performances and standard deviations are over 5 seeds used for model initialization for all results, except for Tables 1 and 6, where we used 20.

(Q1) Limiting Oversmoothing. Since the phenomenon occurs only within deep GNNs, we systematically increased the number of layers in node classification. We mainly compare vanilla GNNs with ReLU to GNNs with CNA. To complete the analysis, we also consider linearized GNNs without any activation function, since they were found to be more resilient against oversmoothing at the expense of slightly reduced performance [Wu et al., 2019]. Figure 4 shows the resulting accuracies for depths of 2 to 96. We can confirm the strong deterioration of vanilla GNNs at greater depths and the partial resilience of linearized GNNs. On the other hand, CNA modules limit oversmoothing drastically and are even more effective than linearized models. At the same time, they significantly alleviate the model's performance shortcomings, effectively eliminating the practical relevance of oversmoothing.

(Q2) Node Classification, Node Regression, and Graph Classification. We evaluated CNA by incorporating it into existing architectures and compared the resulting performances with the unmodified variants. As the results in Table 1 demonstrate, our CNA modules significantly improve classification performance on the Cora dataset [McCallum et al., 2000] by up to 13.53 percentage points. Moreover, this improvement shows across different architectures, highlighting CNA's versatility. Next, we extend our analysis to many more datasets and compare CNA to the best-known models from the literature. Specifically, we evaluate the performance on the following datasets: Cora,

https://doi.org/10.52202/079017-2535

Table 1: **CNA consistently increases the accuracy** (↑) of each architecture on Cora.

Architecture	Baseline	CNA
GCN	81.59±0.43	**93.66±0.48**
GraphSAGE	80.58±0.49	**93.68±0.50**
TransformerConv	79.97±0.78	**93.50±0.58**
GAT	80.57±0.81	**92.94±0.71**

Table 2: **CNA systematically improves graph classification accuracy** (↑).

Graph Dataset	Baseline	CNA
Mutag	78.42±6.55	**81.60±4.18**
Enzymes	36.97±3.08	**50.00±3.25**
Proteins	72.72±2.60	**74.44±2.49**

Table 3: **CNA reduces the NMSE** (↓) on two multiscale node regression datasets.

Model	**Chameleon**	**Squirrel**
GCN	0.207±0.039	0.143±0.039
GAT	0.207±0.038	0.143±0.039
PairNorm	0.207±0.038	0.140±0.040
GCNII	0.170±0.034	0.093±0.031
G^2-GCN	0.137±0.033	0.070±0.028
G^2-GAT	0.136±0.029	0.069±0.029
Trans.Conv	0.133±0.033	0.072±0.025
Trans.Conv+CNA	**0.131±0.033**	**0.068±0.027**

Table 4: **Comparison of our method CNA with the leaderboard on Papers with Code (PwC)**,[2] as of writing on a diverse set of node classification datasets from five typical collections. CNA outperforms the respective leaders, and thereby all compared methods, in eight out of eleven cases (73%). For some, it does so by a significant margin, e.g., on the popular *Cora* and *CiteSeer* datasets.

	Best CNA Result		**PwC Leaderboard**	
Dataset	**Architecture**	**Accuracy (↑)**	**Architecture**	**Accuracy (↑)**
Chameleon	Dir-GNN	**85.86±1.80**	DJ-GNN [Begga et al., 2023]	80.48±1.46
CiteSeer	GAT	**95.75±0.58**	ACMII-Snowball-2 [Luan et al., 2022]	82.07±1.04
Computers	TransformerConv	**92.68±0.27**	Exphormer [Shirzad et al., 2023]	91.47±0.17
Cora	GraphSAGE	**94.18±0.33**	SSP [Izadi et al., 2020]	90.16±0.59
CoraFull	TransformerConv	**71.82±0.25**	CoLinkDist [Luo et al., 2021]	70.32
DBLP	GCN	**86.90±0.45**	GRACE [Zhu et al., 2020]	84.2±0.1
Photo	TransformerConv	**95.96±0.29**	CGT [Hoang and Lee, 2023]	95.73±0.84
Pubmed	TransformerConv	90.16±0.13	ACM-Snowball-3 [Luan et al., 2022]	**91.44±0.59**
Squirrel	Dir-GNN	**77.47±1.28**	Dir-GNN [Rossi et al., 2023]	75.31±1.92
Texas	GraphSAGE	90.00±3.65	2-HiGCN [Huang et al., 2024]	**92.45±0.73**
Wisconsin	TransformerConv	89.29±2.26	5-HiGCN [Huang et al., 2024]	**94.99±0.65**
#Wins		**8**/11		3/11

CoraFull [Kipf and Welling, 2016], CiteSeer [Bojchevski and Günnemann, 2018], PubMed [Sen et al., 2008], DBLP [Tang et al., 2008], Computers and Photo [Shchur et al., 2019], Chameleon, Squirrel, Texas, and Wisconsin [Pei et al., 2020]. The results in Table 4 demonstrate the effectiveness of CNA. Out of 11 of those datasets, CNA outperforms the SOTA on 8 of them. In particular, for CiteSeer, CNA achieves a classification accuracy of 95.75% compared to 82.07% for ACMII-Snowball-2. This suggests that CNA is particularly effective in dealing with the imbalanced class distribution in CiteSeer. The application of CNA is successful on the famous Cora dataset, achieving 94.18% accuracy compared to the 90.16% of SSP. Considering the results in relation to the dataset properties listed in Appendix A.1, we can see that CNA is particularly effective on larger datasets and such ones with many features. It is largely unaffected by the usually detrimental degree of heterophily and the number of classes due to the clustering step being mostly independent of them.

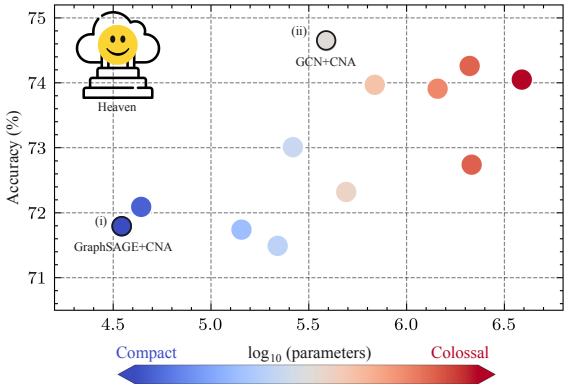
Table 3 displays the comparison in performance in multi-scale node regression task as considered by Rusch et al. [2023b] on the Chameleon and Squirrel datasets [Rozemberczki et al., 2021]. Here, *multi-scale* refers to the wide range of regression targets from 10^{-5} to 1. CNA modules consistently outperform alternative methods in terms of normalized mean squared error (NMSE) based upon the ten pre-defined splits by Pei et al. [2020]. This superior performance highlights the effectiveness of

[2] `https://paperswithcode.com/task/node-classification`

Table 5 & Figure 5: **CNA allows for (i) compact and (ii) accurate models:** The separate treatment of super-nodes boosts expressivity, making GNNs more compact.

Model	Accuracy (↑)	Params (↓)
GraphSAGE*	59.97±0.33	34.8k
GCN*	69.66±0.27	388k
GraphSAGE	71.49±0.27	219k
GCN	71.74±0.29	143k
(i) GraphSAGE+CNA	71.79±0.08	**34.9k**
DAGNN	72.09±0.25	43.9k
DeeperGCN	72.32±0.27	491k
GCNII	72.74±0.16	2.15M
RevGCN-Deep	73.01±0.31	262k
GAT	73.91±0.12	1.44M
UniMP	73.97±0.15	687k
RevGAT-Wide	74.05±0.11	3.88M
RevGAT-SelfKD	74.26±0.17	2.10M
(ii) GCN+CNA	**74.64±0.13**	389.2k

* Reproduced by ourselves with the same hyperparameters as (i) and (ii). Other baselines are taken from the literature.



our approach in handling the complexities of node-level regression tasks. These results suggest that our approach has the potential to provide more accurate predictions in real-world scenarios.

To go beyond node-wise tasks on single graphs, we continue by evaluating CNA on graph-level classification tasks. Namely, we compared CNA with ReLU on the Mutag [Debnath et al., 1991], Enzymes [Borgwardt et al., 2005], and Proteins [Borgwardt et al., 2005] datasets from the TUDataset collection Morris et al. [2020]. Table 2 demonstrates that CNA boosts performance unanimously; for instance, achieving an impressive improvement of 13 percentage points on Enzymes.

A further comparison of CNA to other graph normalization techniques is provided in Appendix A.3. Summarizing the findings on node classification, node regression, and graph classification benchmarks, we can confidently answer (Q2) affirmatively.

(Q3) Parameter Parsimony. CNA creates super-nodes in graphs, each rescaled separately and governed by an individual learnable activation function. This increased specificity and, in turn, expressivity might allow for more compact models. To investigate this, we use the ogbn-arxiv dataset from the *Open Graph Benchmark* (OGB) [Hu et al., 2020] and follow the setup of Li et al. [2021]. We compare GNNs equipped with CNA to a set of baselines without it. The results in Table 5, first of all, clearly show how CNA outperforms a range of existing GNN models (ii). It achieves a test accuracy of 74.64% while estimating a modest number of learnable parameters (389.2k). This indicates that CNA can successfully capture the underlying patterns in the graph data while maintaining a computationally efficient model. The baselines have varying levels of complexity, with some having more layers and/or channels per layer than others. However, CNA outperforms all competitors, even those with more complex architectures. Figure 5 shows that architectures coming close to the performance of CNA need far more parameters that require learning by gradient descent. Namely, improving GraphSAGE + CNA by 2.47 percentage points (the difference to RevGAT-SelfKD) results in a model about 60x bigger. Similarly, the 2.85 percentage point improvement from GraphSAGE + CNA to GCN + CNA is achieved with a model only about eleven times larger. Additional data, such as the underlying abstract texts originally used to generate the citation graph node features, has recently been used with LLMs to distill additional context [He et al., 2023, Duan et al., 2023]. We exclude them to maintain a level playing field, yet recognize it as an interesting avenue for future work. We argue that CNA modules pave the way for a desirable development of GNN modeling when increasing expressivity would not require an explosion in the number of learnable parameters.

(Q4) Model Analysis. We assess the contribution of each of the three operations – Cluster, Normalize, and Activate. To this end, we tested GCN with different subsets of the three operations on the Cora dataset. Table 6 demonstrates that dropping even one of the operations results in minor or no improvement over the plain architecture using ReLU as activation. Cluster-Normalize already improves over the baseline, confirming the findings of Zhou et al. [2020b]. To assess the sensitivity of CNA to the choice of its hyperparameters, we compared the effect of the number of hidden features and the number of clusters per layer on the Cora dataset using GCN, as shown in Figure 6. We

https://doi.org/10.52202/079017-2535

Table 6: **Ablation Study** measured in accuracy (↑) on two datasets.

Cluster	Normalize	Activate	Cora	obgn-arxiv
			81.59±0.43	69.65±0.19
✓			81.25±0.64	69.65±0.19
✓	✓		93.02±0.36	69.47±0.36
✓		✓	81.64±0.61	69.42±0.15
		✓	81.49±0.54	69.36±0.13
	✓		81.60±0.72	69.66±0.21
	✓	✓	81.60±0.70	69.42±0.13
✓	✓	✓	**93.66±0.48**	**74.16±0.33**

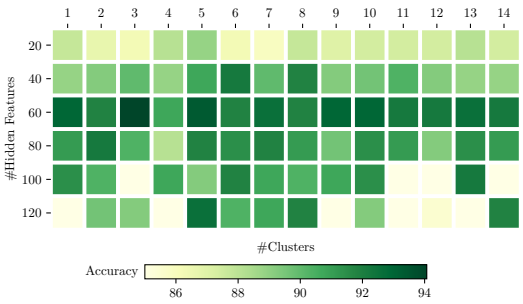


Figure 6: **Hyperparameter sensitivity analysis.**

find that CNA is very robust to the choice of these hyperparameters and works best with moderate numbers of features, as the results from (Q3) would suggest. Answering (Q4), we observed that all three operations of CNA are necessary for the method's efficacy, and it permits practitioners to choose hyperparameters flexibly.

5 Conclusions

In this work, we proposed Cluster-Normalize-Activate modules as a drop-in method to improve the Update step in GNN training. The experimental results demonstrated the effectiveness of CNA modules in various classification, node-property prediction, and regression tasks. Furthermore, we found it to be beneficial across many different GNN architectures. CNA permits more compact models on similar or higher performance levels. Although CNA does not entirely prevent oversmoothing, it does considerably limit its effects in deeper GNNs. Our ablation studies have shown that each step in CNA contributes to the overall efficacy and its overall robustness. CNA provides a simple yet effective way to improve the performance of GNNs, enabling their use in more challenging applications, such as traffic volume prediction, energy grid modeling, and drug design.

Limitations. We focused our evaluation on very popular architectures and datasets. While it is likely that CNA is beneficial in many other configurations, we did not evaluate its effects on GNNs that are not convolutional MPNNs. Similarly, while we did scale or method to the ogbn-arxiv dataset with about 169k nodes and more than a million edges, yet larger datasets might require further work on the speed of the clustering procedure. Our experiments suggest that oversmoothing is of limited practical relevance. Yet, we did not scale this investigation to even greater depth or establish a formal link to existing theories for oversmoothing.

Future Work. The presented results motivate further enhancing CNA in multiple ways. Notably, there are three possible directions. Firstly, regarding clustering, we investigated k-means and GMMs, yet it is important to consider other algorithms. For example, Differentiable Group Normalization [Zhou et al., 2020b] is a promising direction for introducing a learnable clustering step. Further, clustering algorithms need not only to yield a fixed number of clusters k, but should also produce equally sized clusters. Beyond discovering more stable super nodes, this is likely to improve the learning of the rational projections as well. Apart from representational power, investigating faster clustering procedures paves the way toward scaling GNNs via CNA to dynamic and continuous training settings. Secondly, even more potential for improvement lies in combining CNA with other techniques. For example, representing the Aggregate step as learnable sequence models [Hamilton et al., 2017]. These can be beneficial to distill local information to a greater degree, which in turn could further improve performance and limit oversmoothing. Also, combining CNA with established methods like Edge Dropout or Global Pooling can yield compounding benefits. Finally, the abstract idea behind CNA, namely grouping representations and performing distinct updates, is a more general concept and applicable beyond the architectures we have considered in this work. For instance, Transformers [Vaswani et al., 2017] are known to be equivalent to MPNNs on fully connected graphs and can similarly exhibit oversmoothing [Shi et al., 2021a], motivating a closer look at this connection. Unifying the theory about the different clustering-based normalization approaches and their effect on expressivity and phenomena such as oversmoothing might uncover further opportunities for improvements.

Acknowledgments and Disclosure of Funding

This research project was funded by the ACATIS Investment KVG mbH project "Temporal Machine Learning for Long-Term Value Investing" and the German Federal Ministry of Education and Research (BMBF) project KompAKI within the "The Future of Value Creation – Research on Production, Services and Work" program (funding number 02L19C150) managed by the Project Management Agency Karlsruhe (PTKA). The Eindhoven University of Technology authors received support from their Department of Mathematics and Computer Science and the Eindhoven Artificial Intelligence Systems Institute. Authors thank Ponturo Consulting AG for their support.

References

Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *The Ninth International Conference on Learning Representations*, 2020.

Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, June 2021. doi: 10.1016/j.neunet.2021.01.026.

Heba Askr, Enas Elgeldawi, Heba Aboul Ella, Yaseen A. M. M. Elshaier, Mamdouh M. Gomaa, and Aboul Ella Hassanien. Deep learning in drug discovery: an integrative review and future challenges. *Artificial Intelligence Review*, 56:5975–6037, July 2023. doi: 10.1007/s10462-022-10306-1.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, October 2018. URL `http://arxiv.org/abs/1806.01261`.

Ahmed Begga, Francisco Escolano, Miguel Angel Lozano, and Edwin R. Hancock. Diffusion-Jump GNNs: Homophiliation via Learnable Metric Filters, June 2023. URL `http://arxiv.org/abs/2306.16976`.

Shai Ben-David, Dávid Pál, and Hans Ulrich Simon. Stability of k-Means Clustering. In Nader H. Bshouty and Claudio Gentile, editors, *Learning Theory*, pages 20–34, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-72927-3. doi: 10.1007/978-3-540-72927-3_4.

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Yevgeniy Bodyanskiy and Serhii Kostiuk. Learnable Extended Activation Function for Deep Neural Networks. *International Journal of Computing*, 22(3), October 2023. doi: 10.47839/ijc.22.3.3225.

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *The Sixth International Conference on Learning Representations*, 2018.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics (Oxford, England)*, 21 Suppl 1:i47–56, June 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti1007.

Nicolas Boulle, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, May 2021. URL `http://arxiv.org/abs/2104.13478v2`.

Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. GraphNorm: A Principled Approach to Accelerating Graph Neural Network Training. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. ISSN: 2640-3498.

 https://doi.org/10.52202/079017-2535

Francesco Caso, Giovanni Trappolini, Andrea Bacciu, Pietro Liò, and Fabrizio Silvestri. Renormalized Graph Neural Networks, June 2023.

Yihao Chen, Xin Tang, Xianbiao Qi, Chun-Guang Li, and Rong Xiao. Learning graph normalization for graph neural networks. *Neurocomputing*, 493:613–625, July 2022. ISSN 0925-2312.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, July 2019.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, February 1991. ISSN 0022-2623. doi: 10.1021/jm00106a046.

Quentin Delfosse, Patrick Schramowski, Martin Mundt, Alejandro Molina, and Kristian Kersting. Adaptive Rational Activations to Boost Deep Reinforcement Learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. SimTeG: A Frustratingly Simple Approach Improves Textual Graph Learning, August 2023. URL `http://arxiv.org/abs/2308.02565`.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *Representation Learning on Graphs and Manifolds*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, August 2017.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, 2017.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Van Thuy Hoang and O.-Joun Lee. Mitigating Degree Biases in Message Passing Mechanism by Utilizing Community Structures, December 2023. URL `https://arxiv.org/abs/2312.16788`.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, 2020.

Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization Techniques in Training DNNs: Methodology, Analysis and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10173 – 10196, August 2023. doi: 10.1109/TPAMI.2023.3250241.

Yiming Huang, Yujie Zeng, Qiang Wu, and Linyuan Lü. Higher-order Graph Convolutional Network with Flower-Petals Laplacians on Simplicial Complexes. In *The Thirty-Eighth AAAI Conference on Artificial Intelligence*, January 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456, Lille, France, July 2015. JMLR.org.

Mohammad Rasool Izadi, Yihao Fang, Robert Stevenson, and Lizhen Lin. Optimization of Graph Neural Networks with Natural Gradient Descent. In *Proceedings of the IEEE International Conference on Big Data (Big Data)*, 2020.

Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207, November 2022. doi: 10.1016/j.eswa.2022.117921.

Dimitrios Kelesis, Dimitrios Vogiatzis, Georgios Katsimpras, Dimitris Fotakis, and Georgios Paliouras. Reducing Oversmoothing in Graph Neural Networks by Changing the Activation Function. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Rădulescu, editors, *26th European Conference on Artificial Intelligence ECAI 2023*, 2023. doi: 10.3233/FAIA230400.

Sammy Khalife and Amitabh Basu. On the power of graph neural networks and the role of the activation function, May 2024.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*, 2016.

Guillaume Lachaud, Patricia Conde-Cespedes, and Maria Trocan. Mathematical Expressiveness of Graph Neural Networks. *Mathematics. Mathematical Foundations of Deep Neural Networks*, 10 (24):4770, January 2022. doi: 10.3390/math10244770.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, January 1993. doi: 10.1016/S0893-6080(05)80131-5.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training Graph Neural Networks with 1000 Layers. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

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 of *1*, April 2018. doi: 10.1609/aaai.v32i1.11604.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting Heterophily For Graph Neural Networks. *Advances in Neural Information Processing Systems*, 35:1362–1375, December 2022.

Yi Luo, Aiguo Chen, Ke Yan, and Ling Tian. Distilling Self-Knowledge From Contrastive Links to Classify Graph Nodes Without Passing Messages, June 2021. URL `http://arxiv.org/abs/2106.08541`.

J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1, 1967.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, 2009.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3:127–163, July 2000. doi: 10.1023/A:1009953814988.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks. In *The Eighth International Conference on Learning Representations*, 2019.

Andrew Moore. Very Fast EM-Based Mixture Model Clustering Using Multiresolution Kd-Trees. In *Advances in Neural Information Processing Systems*, 1998.

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

Christian Nauck, Michael Lindner, Konstantin Schürholt, Haoming Zhang, Paul Schultz, Jürgen Kurths, Ingrid Isenhardt, and Frank Hellmann. Predicting basin stability of power grids using graph neural networks. *New Journal of Physics*, 24, April 2022. doi: 10.1088/1367-2630/ac54c9.

 https://doi.org/10.52202/079017-2535

Khang Nguyen, Hieu Nong, Vinh Nguyen, Nhat Ho, Stanley Osher, and Tan Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 25956–25979, Honolulu, Hawaii, USA, July 2023. JMLR.org.

Hoang NT and Takanori Maehara. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters, May 2019. URL http://arxiv.org/abs/1905.09550.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks. In *The Eighth International Conference on Learning Representations*. arXiv, February 2020.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael Bronstein. Edge Directionality Improves Learning on Heterophilic Graphs, November 2023. URL http://arxiv.org/abs/2305.10498.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2), April 2021. doi: 10.1093/comnet/cnab014.

T Konstantin Rusch, Benjamin P Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M Bronstein. Graph-Coupled Oscillator Networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, Baltimore, Maryland, USA, 2022. PMLR.

T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A Survey on Oversmoothing in Graph Neural Networks, March 2023a. URL http://arxiv.org/abs/2303.10993.

T. Konstantin Rusch, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. Gradient Gating for Deep Multi-Rate Learning on Graphs. In *The Eleventh International Conference on Learning Representations*, February 2023b.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61 – 80, December 2008. doi: 10.1109/TNN.2008.2005605.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3), September 2008. doi: 10.1609/aimag.v29i3.2157.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation, June 2019. URL http://arxiv.org/abs/1811.05868.

Han Shi, Jiahui Gao, Hang Xu, Xiaodan Liang, Zhenguo Li, Lingpeng Kong, Stephen M. S. Lee, and James Kwok. Revisiting Over-smoothing in BERT from the Perspective of Graph. In *The Tenth International Conference on Learning Representations*, 2021a.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, page 1548, 2021b. ISBN 1045-0823.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse Transformers for Graphs. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2008.

Matus Telgarsky. Neural Networks and Rational Functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3387–3393. PMLR, July 2017.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *The Fortieth International Conference on Machine Learning*, 2021.

Martin Trimmel, Mihai Zanfir, Richard Hartley, and Cristian Sminchisescu. ERA: Enhanced Rational Activations. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Proceedings of the European Conference on Computer Vision*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *The Thirty-fifth International Conference on Machine Learning*, 2018.

Lilapati Waikhom and Ripon Patgiri. A survey of graph neural networks in various learning paradigms: methods, applications, and challenges. *Artificial Intelligence Review*, 56:6295–6364, July 2023. doi: 10.1007/s10462-022-10321-2.

Yuyang Wang, Zijie Li, and Amir Barati Farimani. Graph Neural Networks for Molecules. *Machine Learning in Molecular Sciences*, 36:21–66, 2023.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. ISBN 2640-3498.

Muhan Zhang and Yixin Chen. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems*, 2018.

Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations (ICLR)*, 2020.

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, January 2020a. doi: 10.1016/j.aiopen.2021.01.001.

Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 4917–4928, Red Hook, NY, USA, December 2020b. Curran Associates Inc. ISBN 978-1-71382-954-6.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning, 2020.

 https://doi.org/10.52202/079017-2535

A Appendix

A.1 Details on the datasets

An overview of the datasets used in our evaluation is found in Table 7 and in Table 8. In addition to the number of nodes, edges, features, and classes, we also provided the node homophily ratio and whether the classes are distributed uniformly, as well as number of graphs for graph-level datasets. The node homophily ratio measures how many of a node's neighbors are members of the same class. It is computed as:

$$\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \frac{|\{\{i,j\} \in \mathcal{E} \,|\, j \in \mathcal{N}_i \wedge y_i = y_j\}|}{|\mathcal{N}_i|}.$$

Table 7: **The datasets we used to evaluate CNA.** The table contains statistics for the node classification and property prediction tasks. For regression, we used the Chameleon and Squirrel datasets, but with each page's log average web traffic as the target value.

Name	#Nodes	#Edges	#Features	#Classes	Homophily	Balanced
Chameleon	2,277	36,101	2,325	5	0.10	No
CiteSeer	3,327	9,104	3,703	6	0.71	No
Computers	13,752	491,722	767	10	0.79	No
Cora	2,708	10,556	1,433	7	0.83	No
CoraFull	19,793	126,842	8,710	70	0.59	No
DBLP	17,716	105,734	1,639	4	0.81	No
Photo	7,650	238,162	745	8	0.84	No
PubMed	19,717	88,648	500	3	0.79	No
Squirrel	5,201	217,073	2,089	5	0.09	Yes
Texas	183	309	1,703	5	0.07	No
Wisconsin	251	499	1,703	5	0.17	No
ogbn-arxiv	169,343	1,166,243	128	40	0.43	No

Table 8: **The graph-level datasets we used to evaluate CNA.** The table contains statistics for the graph-level classification.

Name	#Graphs	Avg. #Nodes	Avg. #Edges	#Features	#Classes	Balanced
Mutag	188	~17.9	~39.6	7	2	No
Enzymes	600	~32.6	~124.3	3	6	Yes
Proteins	1,113	~39.1	~145.6	3	2	No

A.2 Details on hyperparameters and training

Throughout the implementation, we used the following software packages:

- PyTorch Geometric (PyG) [Fey and Lenssen, 2019] [3] is widely used for machine learning on Graphs and was our first choice.
- To implement the clustering step, we used Fast PyTorch Kmeans[4], a GPU-based implementation of the renowned algorithm.
- To implement the activate step, we used rationals from the Activation Functions library[5].

The degrees of freedom for the rationals were set to $n = 5$ for the numerator and $m = 4$ for the denominator for all settings. For all experiments, we used the Adam optimizer with weight decay,

[3] `https://www.pyg.org/`

[4] `https://github.com/DeMoriarty/fast_pytorch_kmeans`

[5] `https://github.com/k4ntz/activation-functions`

Table 9: **Hyperparameters** for node classification, node property prediction and graph-level classification.

Dataset	Architecture Type	#Epochs	#Layers	#Clusters	#Hidden	LR	LR Act.	Weight Decay
Chameleon	Dir-GNN	500	2	10	500	10^{-3}	10^{-8}	$5{\cdot}10^{-12}$
CiteSeer	GAT	100	4	12	60	10^{-3}	10^{-5}	$1{\cdot}10^{-1}$
Computers	TransformerConv	100	2	10	20	10^{-3}	10^{-5}	$5{\cdot}10^{-8}$
Cora	GraphSAGE	50	4	12	28	10^{-3}	10^{-5}	$5{\cdot}10^{-6}$
CoraFull	TransformerConv	100	2	14	140	10^{-3}	10^{-5}	$1{\cdot}10^{-9}$
DBLP	GCN	100	4	12	60	10^{-3}	10^{-5}	$5{\cdot}10^{-1}$
Photo	TransformerConv	100	4	8	80	10^{-3}	10^{-5}	$5{\cdot}10^{-2}$
PubMed	TransformerConv	100	2	12	60	10^{-3}	10^{-5}	$5{\cdot}10^{-2}$
Squirrel	Dir-GNN	500	2	10	500	10^{-3}	10^{-8}	$5{\cdot}10^{-12}$
Texas	GraphSAGE	200	2	10	100	10^{-2}	10^{-5}	$1{\cdot}10^{-2}$
Wisconsin	TransformerConv	200	2	10	500	10^{-2}	10^{-5}	$5{\cdot}10^{-2}$
ogbn-arxiv	GCN	300	4	10	400	10^{-3}	10^{-5}	$1{\cdot}10^{-4}$
ogbn-arxiv	GraphSAGE	1000	4	10	60	10^{-3}	10^{-5}	$1{\cdot}10^{-2}$
Mutag	GCN	1000	4	8	128	10^{-3}	10^{-3}	$1{\cdot}10^{-4}$
Enzymes	GCN	1000	4	8	128	10^{-3}	10^{-3}	$1{\cdot}10^{-4}$
Proteins	GCN	1000	4	8	128	10^{-3}	10^{-3}	$1{\cdot}10^{-4}$

Table 10: **Hyperparameters** for node regression (TransformerConv), the ablation study (*), and Table 1 (*).

Architecture Type	#Epochs	#Layers	#Clusters	#Hidden	LR	LR Act.	Weight Decay
TransformerConv	300	2	8	64	$2{\cdot}10^{-3}$	$1{\cdot}10^{-5}$	$1{\cdot}10^{-4}$
*	200	4	14	280	$1{\cdot}10^{-3}$	$1{\cdot}10^{-5}$	$5{\cdot}10^{-6}$

where we set $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We used summation as the aggregation function due to its simplicity and widespread use. Table 9 lists all relevant hyperparameters used for node classification, property prediction, and graph-level classification tasks. Table 10 provides the hyperparameters for Table 1, for the node regression task, as well as for the ablation study. For the sensitivity test, the only difference from (*) in Table 10 was the number of layers, which was set to 32. Regardless of the setting, each experiment was performed on one A100 Nvidia GPU and took between five minutes and two hours, depending on the specific configuration.

A.3 Comparison to other graph normalization techniques

Table 11 shows how CNA compares to other graph normalization methods proposed in the literature. The reference column indicates the origin of the performance indicators, with the remaining results stemming from Table 1. Our method provides the best classification accuracy.

Table 11: **Comparison of CNA to other graph normalization methods** on the Cora dataset.

Architecture and Normalization	Reference	Accuracy (↑)
GCN	Baseline	81.59±0.43
GCN + BatchNorm	Ioffe and Szegedy [2015]	73.9
GCN + Diff. Group Norm.	Zhou et al. [2020b]	82.0
GCN + PairNorm	Zhao and Akoglu [2020]	71.0
GCN + CNA	Ours	**93.66±0.48**

https://doi.org/10.52202/079017-2535

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract summarizes the contributions of our paper. In particular, our claims about our method (cf. Section 3) are backed by the extensive experimental evaluation in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We make assumptions and the focus of our work transparent throughout this manuscript. Additionally, we discuss limitations explicitly in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not state any theoretical results that require proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our new method is completely explained in Section 3. The experimental evaluation in Section 4 starts by providing the basic setup, complemented by the overview of the datasets in Appendix A.1 and training details, including the specific hyperparameters, in Appendix A.2. See also Question 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

https://doi.org/10.52202/079017-2535

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We ensured the exclusive use of open-source datasets and software to aid in reproducibility. Our code is available at `https://github.com/ml-research/cna_modules`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all necessary details not listed in Section 3 in Appendix A.2, like specific hyperparameters and the choice of optimizer. We matched the datasets and train/val/test splits as close to original publications as possible to maintain comparability to the existing literature and reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All experimental results presented in Section 4 (except for the compute-intense hyperparameter sensitivity analysis) are run over multiple seeds, as stated there. We report means as well as standard deviations for all key results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information is provided in the supplemental Appendix A.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We ensured that our research aligns with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: New and improved machine learning models and architectures are inherently dual-use technologies. In particular, our work on CNA can be used for harmful purposes, like any other GNN deployed for surveillance, unfair social recommendations, etc. However, we did not identify societal impacts specific to this work warranting a dedicated discussion.

https://doi.org/10.52202/079017-2535

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We did not identify any such high-risk aspects in the presented work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We acknowledge all immediately relevant third works and cite them in the appropriate sections (namely Section 4 and Appendix A.2). We ensured the legal compatibility of the employed datasets and software components. The datasets we used are provided and publicly documented by PyTorch Geometric [Fey and Lenssen, 2019].

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the implementation of our method used for the experiments at `https://github.com/ml-research/cna_modules`. It contains resources to get started with using the method and reproducing the results.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did not perform experiments with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We did not perform experiments with human subjects.

https://doi.org/10.52202/079017-2535

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.