Multi-view Graph Condensation via Tensor Decomposition

Nícolas R. Santos[1] , Dawon Ahn[1], Diego Minatel[2], Alneu A. Lopes[2], Evangelos Papalexakis[1]
[1]University of California Riverside, [2]University of São Paulo
{nicolasr,dahn017,epapalex}@ucr.edu, dminatel@usp.br, alneu@icmc.usp.br

Abstract

Training Graph Neural Networks (GNNs) on large-scale graphs presents significant computational challenges due to the resources required for their storage and processing. Graph Condensation has emerged as a promising solution to reduce these demands by learning a compact graph that preserves the essential information of the original one while maintaining the GNN's performance. Despite their efficacy, current condensation approaches frequently rely on a computationally intensive bi-level optimization. Moreover, they fail to maintain a mapping between synthetic and original nodes, limiting the interpretability of the model's decisions. In this sense, a wide range of decomposition techniques have been applied to learn linear or multi-linear functions from graphs, offering a more transparent and less resource-intensive alternative. However, their applicability to graph condensation remains unexplored. This paper addresses this gap and proposes a novel method called Multi-view Graph Condensation via Tensor Decomposition (GCTD) to investigate the extent to which such techniques can synthesize a smaller graph while achieving comparable downstream task performance. Experiments on six datasets show that GCTD effectively reduces graph size while preserving GNN performance. Our code is available at https://github.com/nicolasrsantos/gctd.

1 Introduction

Graph Neural Networks (GNNs) have become a pivotal tool for representation learning on graph data, enabling tasks such as antibiotic discovery, estimated time of arrival (ETA) prediction, and fake news detection Stokes et al. [2020], Derrow-Pinion et al. [2021], Song et al. [2021]. Despite their success, GNNs face significant scalability challenges on large-scale graphs, motivating research on graph size reduction Hashemi et al. [2024]. Among the existing approaches, graph condensation has emerged as a promising solution: it learns a smaller graph that allows GNNs to achieve accuracy comparable to training on the original graph while reducing computational cost.

Existing methods generate condensed graphs using a variety of strategies. Some approaches rely on Kernel Ridge Regression Xu et al. [2023], Wang et al. [2024], while others match different types of information, such as gradients Jin et al. [2022], training trajectories Zheng et al. [2023], eigenbasis Liu et al. [2024a], or data distributions Liu et al. [2022]. Beyond these, CGC Gao et al. [2025] proposes a training-free condensation scheme, DisCo Xiao et al. [2025] introduces a disentangled GNN-free framework, and SimGC Xiao et al. [2024] leverages an MLP with a heuristic to preserve the distribution of node features. Methods like SGDD Yang et al. [2023] and GCSR Liu et al. [2024b] further incorporate structural information from the original graph to produce the condensed version.

While existing methods can reduce graphs without major accuracy loss, they often rely on a bi-level optimization and require multiple parameter initializations, leading to a costly triple-loop procedure. Additionally, these methods lack interpretability, losing the notion of how synthetic nodes relate to

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Advancing Graph Machine Learning.

the original ones. At the same time, there exists an extensive line of work that leverages matrix or tensor decomposition for graph-based tasks that learn simpler functions (*i.e.*, linear or multi-linear) from data in contrast to the nonlinear ones learned by the GNNs Kuang et al. [2012], Henderson et al. [2012], Papalexakis et al. [2013], Hua et al. [2022]. Furthermore, they offer a more transparent view of their decisions, providing a more interpretable approach. Importantly, the goals of tensor decomposition and condensation are inherently aligned: both aim to reduce the size and complexity of the original data while preserving its essential information, thereby lowering the computational cost of downstream tasks. Yet, to the best of our knowledge, no prior study has explored the application of decomposition techniques for synthesizing smaller graphs.

In this work, we address this gap by reframing graph condensation as a decomposition problem, investigating whether key information from a large graph can be transferred to a smaller one while preserving downstream performance. To this end, we introduce Multi-view Graph Condensation via Tensor Decomposition (GCTD), a novel method that constructs a multi-view graph by augmenting the original adjacency matrix into a third-order tensor through random edge perturbations. This tensor is then decomposed, and the resulting latent factors are exploited to synthesize a compact graph. The central idea is that decomposition reveals latent co-clusters in the data, allowing nodes to be grouped into synthetic ones based on shared structural patterns Gujral and Papalexakis [2018].

Through extensive experiments on six real-world datasets, we demonstrate that GCTD effectively synthesizes smaller graphs, outperforming existing baselines on three out of six datasets with up to 4.0% higher accuracy. Additionally, we show that using multi-view decomposition leads to better performance than its single-view counterpart. Our contributions are summarized as follows:

- We propose a novel method for graph condensation that leverages a tensor decomposition technique to reduce the original graph;
- We employ multi-view augmented graphs and show through a comprehensive analysis that it improves upon single-view decomposition;
- We conduct an extensive analysis to showcase to what extent decomposition methods can condense graphs;

2 Preliminaries

Given a target graph $\mathcal{G}^{\mathcal{T}} = \mathbf{A}^{\mathcal{T}}, \mathbf{X}^{\mathcal{T}}, \mathbf{Y}^{\mathcal{T}}$, where $\mathbf{A}^{\mathcal{T}} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, $\mathbf{X}^{\mathcal{T}} \in \mathbb{R}^{N \times d}$ is the d-dimensional node feature matrix, and $\mathbf{Y}^{\mathcal{T}} \in 0, \ldots, C-1^{N}$ denotes the node labels over C classes, the goal of graph condensation is to construct a smaller graph $\mathcal{G}^{\mathcal{S}} = \{\mathbf{A}^{\mathcal{S}}, \mathbf{X}^{\mathcal{S}}, \mathbf{Y}^{\mathcal{S}}\}$ with $\mathbf{A}^{\mathcal{S}} \in \mathbb{R}^{N' \times N'}$, $\mathbf{X}^{\mathcal{S}} \in \mathbb{R}^{N' \times d}$, $\mathbf{Y}^{\mathcal{S}} \in 0, \ldots, C-1^{N'}$, and $N' \ll N$, such that training a GNN on $\mathcal{G}^{\mathcal{S}}$ yields comparable performance to training on the much larger $\mathcal{G}^{\mathcal{T}}$. Prior work typically formulates this task as a bi-level optimization problem Hashemi et al. [2024]:

$$\begin{aligned} &\min_{\mathcal{G}^{\mathcal{S}}} \mathcal{L}(\text{GNN}_{\boldsymbol{\theta}_{\mathcal{G}^{\mathcal{S}}}}(\mathbf{A}^{\mathcal{T}}, \mathbf{X}^{\mathcal{T}}), \mathbf{Y}^{\mathcal{T}}) \\ \text{s.t.} \quad &\boldsymbol{\theta}_{\mathcal{G}^{\mathcal{S}}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\text{GNN}_{\boldsymbol{\theta}}(\mathbf{A}^{\mathcal{S}}, \mathbf{X}^{\mathcal{S}}), \mathbf{Y}^{\mathcal{S}}). \end{aligned} \tag{1}$$

Here, the outer loop optimizes the synthetic graph (e.g., via a gradient-matching loss), while the inner loop trains a GNN parameterized by $\boldsymbol{\theta}$ on the synthetic dataset. To prevent overfitting to a particular initialization, this procedure is repeated multiple times, resulting in an expensive triple-loop optimization. In addition to the aforementioned costly step, most methods leverage a GNN to learn a condensed graph. This results in a black-box model, making it difficult to establish a clear connection between the original and synthetic nodes.

3 Proposed Method

We introduce our method GCTD, which generates a tensor from multi-view graphs and decomposes it to derive a condensed graph. The full pipeline is shown in Figure 1, with a detailed algorithm provided in Appendix A. We also describe our initial attempt based on matrix decomposition in Appendix B.

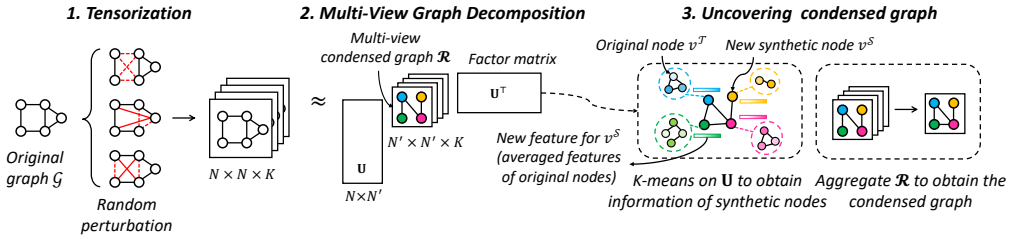


Figure 1: Pipeline of GCTD. We construct a tensor by augmenting the graph's adjacency matrix $\mathbf{A}^\mathcal{T}$ and stacking them together in the third dimension with $\mathbf{A}^\mathcal{T}$. Then, we apply non-negative RESCAL to the given tensor to extract low-rank structures $\mathbf{U}$ and a multi-view condensed graph $\mathcal{R}$. Lastly, we obtain a condensed graph by aggregating $\mathcal{R}$ along the third mode, and we compute the feature and label for each synthetic node by applying K-Means to $\mathbf{U}$.

Tensor creation from multi-view graphs. We build a tensor $\mathcal{X}$ from perturbed versions of the input adjacency matrix $\mathbf{A}^\mathcal{T}$. Specifically, we generate a set of K augmented matrices $S = \{s_1, \ldots, s_K\}$, each obtained by randomly removing and adding edges to $\mathbf{A}^\mathcal{T}$ according to predefined probabilities. This design is motivated by prior work showing that multi-view augmentation improves robustness to adversarial attacks and reduces prediction variance in GNNs Wu et al. [2022].

For each perturbed matrix s_k, edges are first dropped with probability p_r, yielding an expected edge count of $M' = M(1 - p_r)$, where M is the original number of edges. New edges are then added with probability p_a, sampled from the set of non-existing edges. The number of edges after this step becomes $M' = M' + p_a \cdot \left(\frac{N(N-1)}{2} - M'\right)$, where $\frac{N(N-1)}{2}$ is the total number of possible edges in an undirected graph with N nodes. In practice, we set p_r and p_a to small values (i.e., $0.05 \leq p_r, p_a \leq 0.2$) to ensure that the perturbed graphs remain with a similar size to the original one, with $M' \approx M$ on average.

Finally, the perturbed matrices are stacked into a tensor $\mathcal{X}$, where the first slice corresponds to $\mathbf{A}^\mathcal{T}$ and the remaining slices correspond to the matrices in S. The structure of $\mathcal{X}$ is shown in Step 1 of Figure1 and further details on tensors are provided in Appendix C. To reduce runtime overhead, all perturbations are precomputed.

Decomposing a multi-view graph. We reconstruct the multi-view graph $\mathcal{X}$ using the decomposition RESCAL Nickel et al. [2011]. Formally, we compute the following operation for each slice k of $\mathcal{X}$:

$$\tilde{\mathbf{X}}_k = \mathbf{U}\mathbf{R}_k\mathbf{U}^\top, \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{N \times N'}$ is the factor matrix capturing latent components in the given graph and $\mathbf{R}_k$ is the k-th slice of the core tensor $\mathcal{R} \in \mathbb{R}^{N' \times N' \times K}$, indicating relations between latent components existing in the k-th view of the graph. Note that N' is the size of the condensed graph, and K is the total number of augmented graphs.

The formulation presented in Equation 2 consists of an operation that produces K dense matrices, which is computationally expensive for large graphs. To make things worse, keeping them in memory during runtime is resource-intensive since we have to store multiple $N \times N$ matrices. To address this issue, we adopt the sparse version of RESCAL, where we reconstruct only the observed entries of the original tensor as follows:

$$\tilde{x}_{ijk} = \mathbf{u}_i^\top \mathbf{R}_k \mathbf{u}_j. \tag{3}$$

Since this operation considers only observed entries (*i.e.*, nonzero values), providing negative samples is essential to prevent overfitting. Therefore, we randomly generate negative examples in a 1:1 ratio to the available nonzero values, ensuring a balanced set of positive and negative samples. It is noteworthy that this process is precomputed to minimize the computational overhead. In addition to performing sparse decomposition, we employ mini-batching to further reduce memory usage. Finally, we optimize $\mathcal{R}$ and $\mathbf{U}$ using the following reconstruction loss:

$$\mathcal{L}_{rec} = \frac{\frac{1}{n}\sum_{i,j,k}\left(x_{ijk} - \tilde{x}_{ijk}\right)^2}{\sum_{i,j,k} x_{ijk}^2}, \tag{4}$$

where x_{ijk} and $\tilde{x}_{ijk}$ are entries from the original and reconstructed tensor, respectively.

Nonnegativity and sparsity constraint. Given the nonnegative nature of most real-world graphs, we impose a constraint in our method to maintain the nonnegative characteristics of the data. It is important to note that we introduce hard constraints instead of soft regularization-based ones that prior work did Zulaika et al. [2023]. Specifically, we first ensure that the random initialization of the factor matrix and core tensor comprises only nonnegative values by applying the absolute function to them (*i.e.*, $\mathbf{U} \leftarrow |\mathbf{U}|$ and $\mathcal{R} \leftarrow |\mathcal{R}|$). Additionally, since every optimization step of GCTD can shift some values of $\mathbf{U}$ and $\mathcal{R}$ to negative, we apply the nonlinear activation function ReLU to them in every epoch to ensure they remain nonnegative. Besides guaranteeing the constraint mentioned above, applying ReLU inherently induces sparsity in the learned graph since it sets a portion of the values to zero, reducing the storage necessary for the condensed graph.

Computing the structure and features of a condensed graph. The reconstruction in Equation 3 yields a low-rank approximation of the original adjacency matrix, grouping nodes with similar interaction patterns while encoding node and relation-level information in $\mathbf{U}$ and $\mathcal{R}$, respectively. To derive the structure of the condensed graph $\mathcal{G}^{\mathcal{S}}$, we average each slice $\mathcal{R}_{i,j,:}$ of the core tensor to obtain the adjacency matrix $\mathbf{A}^{\mathcal{S}}$. Since this aggregation may produce an asymmetric matrix, we enforce symmetry by adding the corresponding off-diagonal entries.

To extract the synthetic nodes from this decomposition, we apply the K-Means algorithm to the rows of the factor matrix $\mathbf{U}$. Formally, we minimize the following function:

$$\mathcal{L}_{\text{K-Means}} = \sum_{i=1}^{I}\sum_{j=1}^{J} \delta_{ij}\|\mathbf{u}_i - \mu_j\|^2 \quad \text{s.t.} \quad \delta_{ij} = \begin{cases} 1, & \text{if } j = \arg\min_k \|\mathbf{u}_i - \mu_k\|^2 \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $\mathbf{u}_i$ is a row in $\mathbf{U}$, μ_j corresponds to a cluster centroid, and $\delta_{ij} \in \{0, 1\}$ indicates to which of the J clusters $\mathbf{u}_i$ is assigned. We initialize the centroids randomly and the number of clusters is set to match the number of columns of $\mathbf{U}$, denoted N', which corresponds to the size of $\mathcal{G}^{\mathcal{S}}$. This process determines which nodes from the original graph $\mathcal{G}^{\mathcal{T}}$ should be grouped together in the reduced graph, providing the necessary information to assign class labels, splits, and features to the synthetic nodes.

Given the assignments for cluster j, the split of the corresponding synthetic node is chosen as the most frequent split among the original nodes i for which $\delta_{i,j} = 1$. The class label is then determined by considering only the nodes from that split. For example, if five nodes are assigned to cluster j with splits train, val, train, test, train and classes 0, 2, 1, 0, 0, the synthetic node will be assigned to the training split and to class 0, which is the most frequent class among the training nodes. Finally, the features of the synthetic node are computed by averaging the embeddings of the nodes assigned to it that share both its split and class. Throughout this process, we prioritize underrepresented splits and classes to ensure the condensed graph preserves the original distribution.

Complexity Analysis. The complexity of GCTD comprises three parts: tensor decomposition, clustering, and adjacency generation. For decomposition, operations are performed only on observed entries (Eq. 3), giving $\mathcal{O}(N'^2)$ per entry and $\mathcal{O}(N'^2MK)$ overall, where M is the number of edges per view and K the number of views ($2 \leq K \leq 5$). Since $M \ll N^2$ and $N' \ll N$, this step is tractable. KMeans adds $\mathcal{O}(TN'^2N)$ complexity, with $T = 20$ iterations in our experiments, and adjacency generation costs $\mathcal{O}(N'^2)$. Thus, the total complexity is $\mathcal{O}(N'^2MK) + \mathcal{O}(TN'^2N) + \mathcal{O}(N'^2)$.

In practice, this cost remains entirely manageable because the purpose of graph condensation is to reduce the graph size to a small N'. Even for the largest datasets (i.e., Reddit and Ogbn-arxiv), the condensed graphs have fewer than 1,000 nodes (i.e., $N' < 1000$), keeping the cost well within practical limits. For smaller datasets, N' is typically below 100, further alleviating this cost.

4 Experiments

In this section, we describe our experimental settings and present the obtained results. Additional information about baselines, hyperparameters, and datasets is provided in Appendix D, while further experiments are reported in Appendix E.

Baselines. We compared our method against eleven baselines, which include three graph coreset methods (Random, Herding Welling [2009], and K-Center Sener and Savarese [2017]), one coarsening method Huang et al. [2021], two graph distillation methods: SGDD Yang et al. [2023], and GDEM Liu et al. [2024a], and five graph condensation methods: GCond Jin et al. [2022], SFGC Zheng et al. [2023], SNTK Wang et al. [2024], GCDM Liu et al. [2022], and CGC Gao et al. [2025].

Datasets. Following prior work in graph condensation Jin et al. [2022], Yang et al. [2023], Zheng et al. [2023], we evaluate our method on six datasets that range from a few thousand to hundreds of thousands of nodes: four transductive graphs (Cora, Citeseer, Pubmed, and Ogbn-arxiv) and two inductive graphs (Flickr and Reddit). Cora, Citeseer, and Pubmed are taken from the PyTorch Geometric library, Flickr and Reddit from GraphSAINT Zeng et al. [2020], and Ogbn-arxiv from the Open Graph Benchmark Hu et al. [2020]. All experiments utilize the official public splits for each dataset.

Hyperparameter Settings. In the condensation step of GCTD, we considered the decomposition converged either after 200 epochs or when the absolute difference in reconstruction error between epochs t and $t+1$ is below 10^{-7}. Following Jin et al. [2022], during the evaluation phase, we train a 2-layer GCN with 256 hidden units and no dropout for 600 epochs on the condensed graph, selecting the model with the lowest validation loss for final performance evaluation on the original test set.

4.1 Experimental Results

To begin our analysis, we evaluate the performance of our method against selected baselines across six datasets, using three condensation ratios commonly utilized in the literature. We present the average performance across ten runs and the corresponding standard deviation in Table 1. Notably, our method yields superior performance on the Citeseer, Cora, and Pubmed datasets. For example, it achieves a 4.0%, 3.4%, and 4.1% improvement in accuracy across the three condensation ratios applied to Citeseer. Moreover, GCTD exhibits lossless performance in all settings for Citeseer, Pubmed, and Flickr, as well as in the 1.3% and 2.6% ratios for Cora.

Table 1: Node classification accuracy (%) of the baselines and our proposed method. We report the average of ten runs and the standard deviation. Best and second-best results are in bold and underlined, respectively. Out of memory (OOM) is 49GB.

Datasets	Ratio (%)	Traditional Methods				Condensation and Distillation Methods								Full
		Random	Herding	K-Center	Coarse	GCond	SFGC	SGDD	SNTK	GCDM	GDEM	CGC	**GCTD**	Dataset
Cora	1.3	63.6±3.7	67.0±1.3	64.0±2.3	31.2±0.2	79.8±1.3	77.7±1.8	79.1±1.3	78.4±1.4	79.1±0.9	68.0±0.1	**82.6±0.3**	81.4±1.6	81.4±0.6
	2.6	72.8±1.1	73.4±1.0	73.2±1.2	65.2±0.6	80.1±0.6	79.3±0.8	79.0±1.9	79.7±0.9	80.5±0.3	72.8±0.8	81.2±0.6	**84.0±0.4**	
	5.2	76.8±0.1	76.8±0.1	76.7±0.1	70.6±0.1	79.3±0.3	79.4±0.5	80.2±0.8	80.5±0.6	80.2±0.5	77.4±0.6	**82.1±0.9**	79.9±0.3	
Citeseer	0.9	54.4±4.4	57.1±1.5	52.4±2.8	52.2±0.4	70.5±1.2	66.3±2.4	71.5±0.9	66.1±3.0	72.8±0.3	72.3±0.3	72.5±0.5	**76.8±0.4**	71.7±0.4
	1.8	64.2±1.7	66.7±1.0	64.3±1.0	59.0±0.5	70.6±0.9	69.0±1.1	71.2±0.7	69.2±1.2	71.7±0.2	72.6±0.6	73.1±0.2	**76.5±2.5**	
	3.6	69.1±0.1	69.0±0.1	69.1±0.1	65.3±0.5	69.8±1.4	70.8±0.4	70.9±1.2	71.0±0.6	72.5±0.5	72.6±0.5	71.5±0.3	**76.7±0.2**	
Pubmed	0.08	69.5±0.5	73.0±0.7	69.0±0.6	18.1±0.1	78.3±0.2	76.4±1.2	77.1±0.5	78.9±0.7	77.1±0.3	77.7±0.7	77.3±0.1	**79.9±0.2**	77.1±0.3
	0.15	73.8±0.8	75.4±0.7	73.7±0.8	28.7±4.1	77.1±0.3	77.5±0.4	78.0±0.3	79.3±0.3	76.8±0.6	78.4±1.8	76.0±0.5	**79.4±2.8**	
	0.3	77.9±0.4	77.9±0.4	77.8±0.5	42.8±4.1	78.4±0.3	77.9±0.3	77.5±0.5	79.4±0.3	78.1±0.3	78.2±0.8	77.8±0.2	**80.0±1.0**	
Arxiv	0.05	47.1±3.9	52.4±1.8	47.2±3.0	35.4±0.3	59.2±1.1	59.0±1.8	59.6±0.5	58.7±1.7	63.8±0.3	63.7±0.8	**64.1±0.4**	58.2±1.7	71.3±0.1
	0.25	57.3±1.1	58.6±1.2	56.8±0.8	43.5±0.2	63.2±0.3	64.6±0.3	61.7±0.3	64.2±0.5	**66.7±0.4**	63.8±0.6	66.2±0.1	57.9±2.1	
	0.5	60.0±0.9	60.4±0.8	60.3±0.4	50.4±0.1	64.0±0.4	65.2±0.8	58.7±0.6	65.1±0.7	**67.6±0.3**	64.1±0.3	67.0±0.2	57.8±0.7	
Flickr	0.1	41.8±2.0	42.5±1.8	42.0±0.7	41.9±0.2	46.5±0.4	45.5±0.8	46.1±0.3	45.4±0.4	44.8±0.5	**49.9±0.8**	47.1±0.1	48.4±0.8	47.1±0.1
	0.5	44.0±0.4	43.9±0.9	43.2±0.1	44.5±0.1	47.1±0.1	46.0±0.4	45.9±0.4	46.0±0.4	46.4±0.2	**49.4±1.3**	47.1±0.0	48.1±1.4	
	1	44.6±0.2	44.4±0.6	44.1±0.4	44.6±0.1	47.1±0.1	46.1±0.3	46.4±0.2	46.2±0.4	46.7±0.2	**49.9±0.6**	46.8±0.1	48.0±0.9	
Reddit	0.05	46.1±4.4	53.1±2.5	46.6±2.3	40.9±0.5	88.0±1.8	80.0±3.1	84.2±0.7	OOM	91.2±0.1	**92.9±0.3**	90.7±0.1	91.3±0.1	94.1±0.0
	0.1	58.0±2.2	62.7±1.0	53.0±3.3	42.8±0.8	89.6±0.7	84.6±1.6	80.6±0.4	OOM	92.4±0.0	**93.1±0.2**	91.0±0.1	90.1±0.9	
	0.2	66.3±1.9	71.0±1.6	58.5±2.1	47.4±0.9	90.1±0.5	87.9±1.2	84.1±0.3	OOM	92.7±0.1	**93.2±0.4**	90.5±0.0	90.8±0.5	

Analyzing the results on Flickr, one of the most challenging datasets in our evaluation, we observe that GCTD performs robustly, securing the second-best results across all condensation ratios and achieving lossless performance. Similarly, GCTD performs competitively on Reddit, consistently ranking among the top-performing methods. For instance, at the 0.05% condensation ratio, it achieves 91.3% accuracy (the second-best result overall) and maintains comparable performance at higher ratios. These findings highlight the effectiveness of our method in transductive and inductive settings, as well as in large datasets (i.e., Flickr and Reddit). Although results on Ogbn-arxiv do not surpass the strongest baselines, the method maintains competitive performance and demonstrates robustness across diverse datasets, indicating promising potential for further development.

On the number of graph views. Next, we present an experiment we conducted to analyze the impact of the number of augmented multi-views used in condensing the original graph. In this evaluation, we assessed the performance of our method across graphs with varying numbers of views, ranging from 1 to 5. It is important to highlight that the case with one view corresponds to a matrix trifactorization. The results, illustrated in Figure 2, demonstrate how the number of views influences

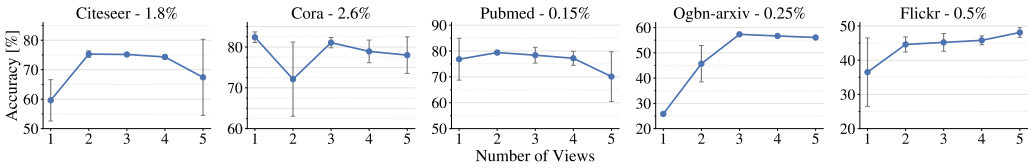


Figure 2: Accuracy scores achieved by our method on graphs with varying numbers of views. The values following each dataset name represent the condensation ratio applied in this ablation study. Experiments were conducted ten times, and we report the average accuracy along with the error bars.

overall performance. Our results indicate that, except for Cora, using a multi-view augmented graph consistently yields better performance than the single-view one. For example, on the Ogbn-arxiv dataset, we observed a 31.5% performance improvement when using three views instead of just one, highlighting the effectiveness of using augmented views.

Table 2: Running time analysis of the proposed method and the SGC version of the baselines. The experiments were performed on a single NVIDIA RTX A6000. OOM is 49GB of memory usage.

Dataset (r%)	GCond	SGDD	SFGC	GCDM	SNTK	GCTD
Citeseer (0.9)	70.8	140.9	63.0	131.3	2.9	32.1
Cora (1.3)	90.3	115.1	62.4	174.7	3.1	42.8
Pubmed (0.08)	57.1	2200.5	61.6	75.4	4.2	48.1
Ogbn-arxiv (0.05)	725.0	2073.3	238.0	480.0	925.4	965.6
Flickr (0.1)	470.4	361.5	42.9	225.0	148.7	222.9
Reddit (0.05)	3780	10350	2420	1617.1	OOM	2515.3

Running time. In Table 2, we compare the running time of our method against GCond, SGDD, SFGC, GCDM, and SNTK. To ensure a fair comparison, we adopted the SGC implementation for all baselines, as it offers faster runtime than their GCN counterparts. Overall, our method strikes a balance between speed and performance. It consistently outperforms SGDD across all datasets, significantly reducing condensation time. For instance, on Citeseer, our method is over four times faster than SGDD and twice as fast as SFGC, one of the fastest baseline models. On larger datasets such as Reddit and Ogbn-arxiv, our method remains considerably faster than SGDD.

When compared to GCond and GCDM, our method shows better efficiency on smaller datasets and Flickr, while also outperforming GCond on Reddit. It is, however, slightly slower than GCDM on Reddit. Notably, SFGC remains among the fastest methods due to its structure-free condensation approach, which aligns with the findings of Han et al. [2023] that matrix multiplication with the adjacency matrix is the most time-consuming operation in GNNs. Lastly, while SNTK performs well on smaller graphs, its runtime advantage diminishes as graph size increases. On Reddit, it runs out of memory (49GB in our setup) due to its expensive kernel computation.

5 Conclusion

In this paper, we introduced GCTD, a novel framework for graph condensation based on the decomposition of multi-view augmented graphs. We demonstrated that tensor decomposition can generate compact graphs that preserve GNN performance on downstream tasks. To validate our approach, we evaluated GCTD on six real-world datasets spanning both transductive and inductive learning settings, complemented by an in-depth analysis of its capabilities.

Our experiments show that GCTD outperforms existing methods on three of the six benchmarks and delivers competitive results on the larger datasets. The framework proves effective in both transductive and inductive scenarios. For future work, we plan to explore alternative decomposition techniques, investigate different augmentation strategies, and replace the current hard assignment with a soft membership approach.

Acknowledgements

Research was supported in part by the National Science Foundation under CAREER grant no. IIS 2046086, grant no. No. 2431569 and CREST Center for Multidisciplinary Research Excellence in CyberPhysical Infrastructure Systems (MECIS) grant no. 2112650, and by the Agriculture and Food

Research Initiative Competitive Grant no. 2020-69012-31914 from the USDA National Institute of Food and Agriculture. This study was also supported by the Coordination for the Improvement of Higher Education Personnel (CAPES) through the Institutional Internationalization Program (PRINT), Call No. 41/2017, and partially funded by CAPES (Finance Code 001), the São Paulo Research Foundation (FAPESP) [grants 20/09835-1, 22/02176-8, and 22/09091-8], and the National Council for Scientific and Technological Development (CNPq) [grants 303588/2022-5 and 406417/2022-9]. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

Broader Impacts: Our framework aims to reduce the computational cost of training Graph Neural Networks, making graph learning research more accessible and environmentally sustainable. It is essential to recognize that condensed graphs can potentially preserve or amplify biases present in the original data.

References

Mohammadreza Babaee, Stefanos Tsoukalas, Maryam Babaee, Gerhard Rigoll, and Mihai Datcu. Discriminative nonnegative matrix factorization for dimensionality reduction. *Neurocomputing*, 173:212–223, 2016. ISSN 0925-2312.

Minsik Cho, Vinod Muthusamy, Brad Nemanich, and Ruchir Puri. Gradzip: Gradient compression using alternating matrix factorization for large-scale deep learning. In *NeurIPS*. 2019.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pages 3844–3852. Curran Associates, Inc., 2016.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. Eta prediction with graph neural networks in google maps. In *CIKM*. ACM, October 2021.

Charles Dickens, Edward Huang, Aishwarya Reganti, Jiong Zhu, Karthik Subbian, and Danai Koutra. Graph coarsening via convolution matching for scalable graph neural network training. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1502–1510, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701726.

Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, page 126–135, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395.

Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

Xinyi Gao, Guanhua Ye, Tong Chen, Wentao Zhang, Junliang Yu, and Hongzhi Yin. Rethinking and accelerating graph condensation: A training-free approach with class partition. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 4359–4373, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

Ekta Gujral and Evangelos E. Papalexakis. Smacd: Semi-supervised multi-aspect community detection. In *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM)*, pages 702–710, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NeurIPS*, 30, 2017.

Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, and Neil Shah. MLPInit: Embarrassingly simple GNN training acceleration with MLP initialization. In *The Eleventh International Conference on Learning Representations*, 2023.

Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=WigDnV-_Gq`.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 1231–1239, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314626.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Chenqing Hua, Guillaume Rabusseau, and Jian Tang. High-order pooling for graph neural networks with tensor decomposition. *Advances in Neural Information Processing Systems*, 35:6021–6033, 2022.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 675–684, 2021.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=WLEx3Jo4QaB`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR, 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Da Kuang, Chris Ding, and Haesun Park. *Symmetric Nonnegative Matrix Factorization for Graph Clustering*, pages 106–117. 2012.

Zechao Li, Jing Liu, and Hanqing Lu. Structure preserving non-negative matrix factorization for dimensionality reduction. *Computer Vision and Image Understanding*, 117(9):1175–1189, 2013. ISSN 1077-3142.

Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697*, 2022.

Yang Liu, Deyu Bo, and Chuan Shi. Graph distillation with eigenbasis matching. In *Forty-first International Conference on Machine Learning*, 2024a.

Zhanyu Liu, Chaolv Zeng, and Guanjie Zheng. Graph data condensation via self-expressive graph structure reconstruction. *arXiv preprint arXiv:2403.07294*, 2024b.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584, 2011.

Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*, pages 899–905. IEEE, 2013.

Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*, 2017.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017. doi: 10.1109/TSP.2017.2690524.

Chenguang Song, Kai Shu, and Bin Wu. Temporally evolving graph neural network for fake news detection. *Information Processing & Management*, 58(6):102712, 2021. ISSN 0306-4573.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 398:185–196, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.02.035.

Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM on Web Conference 2024*, pages 4439–4448, 2024.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pages 1121–1128, 2009.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871. PMLR, 2019.

Zhebin Wu, Lin Shu, Ziyue Xu, Yaomin Chang, Chuan Chen, and Zibin Zheng. Robust tensor graph convolutional networks via t-svd based graph augmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2090–2099, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850.

Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Simple graph condensation, 2024. URL `https://arxiv.org/abs/2403.14951`.

Zhenbang Xiao, Yu Wang, Shunyu Liu, Bingde Hu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Disentangled condensation for large-scale graphs. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 4494–4506, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746.

Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, 2003.

Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2850–2861, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030.

Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. Does graph distillation see like vision dataset counterpart? In *NeurIPS*, 2023.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *ICLR*, 2020.

Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. Navigating complexity: Toward lossless graph condensation via expanding window matching. *ICML 2024*, 2024.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021.

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

Unai Zulaika, Aitor Almeida, and Diego Lopez-de Ipina. Regularized online tensor factorization for sparse knowledge graph embeddings. *Neural Computing and Applications*, 35(1):787–797, 2023.

A Algorithm

We present the detailed algorithm of GCTD in Algorithm 1. Our method takes as input the observed entries of the multi-view graph along with the negative samples. The factor matrix and core tensor are randomly initialized, and the absolute function is applied to ensure their nonnegativity. From lines 6 to 9 of the algorithm, the adjacency matrix is decomposed and the latent components are optimized while maintaining their nonnegativity throughout the process. Convergence is determined either after 200 epochs or when the variation in reconstruction loss between consecutive steps falls below 10^{-7}, as previously described. Finally, the condensed graph is uncovered by averaging the core tensor and applying KMeans clustering to the factor matrix, extracting the necessary information to generate the features, classes, and splits.

The split of a synthetic node is chosen based on the most frequent split among its assigned nodes, with preference given to underrepresented splits according to the target distribution. To assign a class label, we first filter the assigned nodes to those within the selected split and examine their class labels. Instead of simple majority voting, we compute the class frequencies and prioritize underrepresented classes. The first class whose current count is below its target proportion is selected; if all are satisfied, a class is chosen randomly among the candidates. This strategy helps maintain class balance in the condensed graph. Finally, the feature representation of the synthetic node is computed by averaging the embeddings of its assigned nodes.

Algorithm 1: Graph Condensation via Tensor Decomposition (GCTD)

1 **Input:** Pre-computed multi-view graph $\mathcal{X}$ with negative samples.
2 **Output:** Condensed graph $\mathcal{G}^{\mathcal{S}} = (\mathbf{A}^{\mathcal{S}}, \mathbf{X}^{\mathcal{S}}, \mathbf{Y}^{\mathcal{S}})$
3 Initialize matrix $\mathbf{U}$ and core tensor $\mathcal{R}$ randomly.
4 $\mathbf{U} \leftarrow |\mathbf{U}|$; $\mathcal{R} \leftarrow |\mathcal{R}|$
5 **while** *convergence is not achieved* **do**
6 Reconstruct $\mathcal{X}$ according to Equation 3.
7 Compute the reconstruction error $\mathcal{L}_{\text{rec}}$ according to Equation 4.
8 Update $\mathbf{U} \leftarrow \mathbf{U} - \eta \nabla_{\mathbf{U}} \mathcal{L}_{\text{rec}}$ and $\mathcal{R} \leftarrow \mathcal{R} - \eta \nabla_{\mathcal{R}} \mathcal{L}_{\text{rec}}$
9 $\mathbf{U} \leftarrow \text{ReLU}(\mathbf{U})$; $\mathcal{R} \leftarrow \text{ReLU}(\mathcal{R})$
10 Compute $\mathbf{A}^{\mathcal{S}}_{ij} \leftarrow \text{Average}(\mathcal{R}_{i,j,:}), \quad \forall\, i, j$
11 Cluster $\mathbf{U}$ rows according to Equation 5.
12 Average embeddings of the nodes assigned by K-Means to get $\mathbf{X}^{\mathcal{S}}$.
13 Assign to each synthetic node the most frequent split and class among its assigned nodes, focusing on underrepresented classes first.

B First attempt: Single-view Graph Modeling

Motivated by prior work using Matrix Factorization (MF) for clustering Ding et al. [2006], Xu et al. [2003], dimensionality reduction Li et al. [2013], Babaee et al. [2016], and compression Swaminathan et al. [2020], Cho et al. [2019], which aim to simplify or reduce data complexity, our first attempt at graph condensation was to factorize the adjacency matrix $\mathbf{A}^{\mathcal{T}}$ of a single-view graph $\mathcal{G}^{\mathcal{T}}$.

We employed a variant of MF called Matrix Tri-factorization (MTF), which decomposes a matrix into the product of three lower-dimensional factors Ding et al. [2006]. Specifically, it computes $\mathbf{A}^{\mathcal{T}} \approx \mathbf{U}\mathbf{R}\mathbf{V}^{\top}$, where $\mathbf{U}$ and $\mathbf{V}$ capture the row and column spaces of $\mathbf{A}^{\mathcal{T}}$, while $\mathbf{R}$ encodes the interactions between them. However, given the nature of the undirected graphs, we adopted a symmetric formulation of MTF where $\mathbf{V} = \mathbf{U}$, yielding the following reconstruction:

$$\mathbf{A}^{\mathcal{T}} \approx \mathbf{U}\mathbf{R}\mathbf{U}^{\top}, \tag{6}$$

where $\mathbf{U} \in \mathbb{R}^{N \times N'}$ and $\mathbf{R} \in \mathbb{R}^{N' \times N'}$. Here, N is the number of nodes in the original graph, $N' = rN$ is the number of nodes in the condensed graph, r is the condensation ratio, and $N' \ll N$. Note that if the core tensor in Equation 9 is a matrix, Equation 6 is equivalent to the Equation 9.

Reconstruction objective. After computing $\mathbf{A}^{\mathcal{T}}$, we optimize both $\mathbf{U}$ and $\mathbf{R}$ using gradient descent, employing mean squared error (MSE) as the reconstruction loss function. Additionally, we identified empirically that normalizing the MSE by the sum of the squared elements of the original matrix $\mathbf{A}$

enhances convergence speed and stability. Therefore, the final loss function is defined as:

$$\mathcal{L}_{rec} = \frac{\frac{1}{n}\sum_{i,j}\left(a_{ij} - \hat{a}_{ij}\right)^2}{\sum_{ij} a_{ij}^2}, \tag{7}$$

where a_{ij} and $\hat{a}_{ij}$ represent elements from the original and reconstructed matrices, respectively.

Uncovering the condensed graph. Once $\mathbf{U}$ and $\mathbf{R}$ are optimized, we derive the condensed graph in the same manner as outlined in Section 3, except for a minor adjustment to handle $\mathbf{R}$. Specifically, in the single-view setting $\mathbf{R}$ is a matrix instead of a tensor, which allows us to use it directly as the adjacency matrix of the condensed graph, eliminating the need for aggregation.

Remarks on single-view decomposition. While we successfully synthesize a condensed graph through single-view decomposition, we extended this approach to a multi-view setting, as recent studies indicate that it significantly improves model performance, generalization, and robustness Zhao et al. [2021], Wu et al. [2022], Ding et al. [2022]. Additionally, Section 4.1 demonstrates that this strategy yields notable improvements in GNN performance on the condensed graph compared to its single-view counterpart.

C Tensor Background

Notations. Tensors are multi-dimensional arrays that generalize one-dimensional arrays (or vectors) and two-dimensional arrays (or matrices) to higher dimensions. Throughout this paper, we use boldface Euler script letters (*e.g.*, $\boldsymbol{\mathcal{X}}$) to denote tensors, boldface capitals (e.g., $\mathbf{A}$) to denote matrices, boldface lowercases (*e.g.*, $\mathbf{a}$) to denote vectors, and unbolded letters (*e.g.*, A, a) to denote scalars and coefficients. We refer to a tensor's dimension as its *mode* or *order*. The *slices* are two-dimensional sections of a tensor, defined by fixing all but two indices. For example, the k-th frontal slice of a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ is a matrix denoted as $\mathbf{X}_k \in \mathbb{R}^{I\times J}$. Moreover, the n-mode product defines the multiplication of a tensor with a matrix in mode n. For example, the n-mode product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ with a matrix $\mathbf{U} \in \mathbb{R}^{R\times I}$ along the first mode is denoted by $\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} (\in \mathbb{R}^{R\times J\times K})$.

Tensor decomposition. Tensor decomposition techniques aim to decompose tensors into low-rank latent components, facilitating data mining and analysis. Among the most widely used models are CANDECOMP/PARAFAC (CP) and Tucker decompositions, both of which have seen extensive development and application across a diverse range of fields Sidiropoulos et al. [2017], Rabanser et al. [2017]. In this work, we used a variant of Tucker called RESCAL Nickel et al. [2011] and, thus, we describe them next.

Tucker decomposition approximates a given third mode tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ as follows:

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{R}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}, \tag{8}$$

where $\times_1$, $\times_2$, and $\times_3$ denote the n-mode product along the first, second, and third modes, respectively. Factor matrices $\mathbf{U}^{(1)} \in \mathbb{R}^{I\times R_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{J\times R_2}$ and $\mathbf{U}^{(3)} \in \mathbb{R}^{K\times R_3}$ are considered as the principal components in each mode. The core tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{R_1\times R_2\times R_3}$ captures the interaction between each component. Here, R_1, R_2, and R_3 denote the number of components in each mode and are smaller than I, J, K, respectively. Furthermore, it has been demonstrated that $\boldsymbol{\mathcal{R}}$ can be viewed as a compressed version of $\boldsymbol{\mathcal{X}}$ Kolda and Bader [2009].

RESCAL was initially proposed for relational learning and is particularly useful when the frontal slices of the given tensors exhibit symmetry. It is a special case of the Tucker decomposition, as described in Equation 8, with a few key differences. Specifically, the core tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{R_1\times R_2\times K}$ is employed, and the factor matrix $\mathbf{U}^{(3)}$ is set as an identity matrix. Additionally, the factor matrices $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ are identical and denoted by $\mathbf{U}$, as follows:

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{R}} \times_1 \mathbf{U} \times_2 \mathbf{U} \Leftrightarrow \mathbf{X}_k \approx \mathbf{U}\mathbf{R}_k\mathbf{U}^\top, \tag{9}$$

where $\mathbf{R}_k \in \mathbb{R}^{R_1\times R_2}$ indicates the relations between latent components. It is important to highlight that RESCAL does not compress the third mode of the tensor.

D Datasets and Hyperparameters

D.1 Baselines

It is worth noting that although graph coarsening techniques Huang et al. [2021], Loukas [2019], He et al. [2021], Dickens et al. [2024] are conceptually related to graph condensation, numerous studies have consistently shown that coarsening tends to underperform condensation methods on downstream tasks at high condensation ratios Jin et al. [2022], Yang et al. [2023], Zhang et al. [2024], Liu et al. [2024a], Gao et al. [2025]. Consequently, our evaluation focuses on distillation and condensation baselines, which is consistent with the empirical comparison practices adopted in the majority of recent works in this area.

D.2 Datasets

In Table 3, we provide a summary of the datasets used to evaluate our method. Consistent with prior work in graph condensation Jin et al. [2022], Yang et al. [2023], Zheng et al. [2023], our evaluation covers six datasets ranging from a few thousand to several hundred thousand nodes: four transductive graphs (Cora, Citeseer, Pubmed, and Ogbn-arxiv) and two inductive graphs (Flickr and Reddit). Cora, Citeseer, and Pubmed are sourced from the PyTorch Geometric library, Flickr and Reddit from GraphSAINT Zeng et al. [2020], and Ogbn-arxiv from the Open Graph Benchmark Hu et al. [2020]. All experiments rely on the official public splits provided with each dataset.

Table 3: Statistics of the datasets. The first four datasets are transductive, while the last two are inductive.

Dataset	Nodes	Edges	Classes	Features	Train/Val/Test
Cora	2,708	5,429	7	1433	140/500/1000
Citeseer	3,327	4,732	6	3,703	120/500/1000
Pubmed	19,717	44,338	3	500	50/500/1000
Ogbn-arxiv	169,343	1,166,243	40	128	90,941/29,799/48,603
Flickr	89,250	899,756	7	500	44,625/22,312/22,313
Reddit	232,965	57,307,946	210	602	153,932/23,699/55,334

D.3 Hyper-parameters

To optimize the core tensor $\mathcal{R}$ and the factor matrix $\mathbf{U}$, we used the Adam optimizer Kingma and Ba [2015]. We tuned the learning rates for the reconstruction and evaluation steps within {0.1, 0.01, 0.001, 0.0001}, and the weight decay in a range of {0, 0.01, 0.001, 0.0001}. Additionally, the random edge additions and removals used to generate each augmented view of the original adjacency matrix are tuned across {0.05, 0.1, 0.15, 0.2}. Lastly, the number of views was tuned in a range of {1, 2, 3, 4, 5}, where the value 1 corresponds to a single-view decomposition. All hyperparameter optimization is performed using the Bayesian optimizer provided by Weights and Biases[1].

E Additional experiments

In this section, we present additional experiments conducted to evaluate different aspects of our method. Specifically, we investigate the performance of alternative GNN architectures, conduct an ablation study on synthetic node assignment by comparing a simple argmax strategy with K-Means applied to $\mathbf{U}$, provide visualizations of the generated graphs, and compare key statistics between the original and condensed graphs.

E.1 Performance across different GNN architectures

We also evaluated the performance of various GNN architectures on the condensed graphs generated by GCTD and compared them to GCond, SFGC, and GCDM, which represent diverse approaches to graph condensation. Consistent with the evaluation protocol of Jin et al. [2022] and other follow-up

[1]More information at wandb.ai

works, we trained 2-layer versions of APPNP Gasteiger et al. [2019], ChebyNet Defferrard et al. [2016], SGC Wu et al. [2019], and GraphSAGE Hamilton et al. [2017] using the synthesized graphs and measured their performance on the original graph's test set. For this evaluation, we selected datasets across three size categories: Citeseer and Cora (small), Pubmed (medium), and Flickr (large), with condensation ratios of 1.8%, 2.6%, 0.15%, and 0.5%, respectively. We report the average accuracy over ten runs in Table 4.

Table 4: Accuracy (%) of different GNN architectures on the graphs condensed by our method, GCond, and SFGC. The reported values are an average of ten runs. Best and second-best averages are in bold and underlined, respectively.

Datasets (r%)	Methods	Models					Avg.
		GCN	APPNP	Cheby	SGC	SAGE	
Cora (2.6)	GCond	80.1	78.5	76.0	79.3	78.4	78.5
	SFGC	79.3	78.8	79.0	79.1	80.0	<u>79.2</u>
	GCDM	80.5	79.7	75.5	77.7	77.5	78.2
	GCTD	84.0	85.5	76.2	82.5	74.6	**80.5**
Citeseer (1.8)	GCond	70.6	69.6	68.3	70.3	66.2	69.0
	SFGC	69.0	70.5	71.8	71.8	71.7	71.0
	GCDM	71.7	73.6	66.1	73.6	71.1	<u>71.2</u>
	GCTD	76.5	73.8	75.2	77.4	73.5	**75.3**
Pubmed (0.08)	GCond	77.1	76.8	75.9	77.1	76.9	76.8
	SFGC	77.5	76.3	77.7	77.8	76.3	<u>77.1</u>
	GCDM	77.1	78.3	71.5	74.9	77.0	75.8
	GCTD	79.9	79.2	78.0	79.8	79.0	**79.2**
Flickr (0.5)	GCond	47.1	45.9	42.8	46.1	46.2	<u>45.6</u>
	SFGC	46.0	40.7	45.4	42.5	47.0	44.3
	GCDM	46.4	46.0	42.4	45.8	42.6	44.6
	GCTD	48.1	46.9	42.7	45.4	47.2	**46.1**

Focusing on GCTD, we observe that APPNP, SGC, and GCN generally achieve strong results, often surpassing the average performance across datasets. While APPNP performs best on Cora and SGC leads on Citeseer and Pubmed, GCN consistently stays competitive. GraphSAGE also stands out on Flickr, achieving the highest score among the GNNs for that dataset. In contrast, ChebyNet tends to underperform relative to the others, with below-average results on most datasets.

Analyzing the baselines, we observe that GCTD outperforms all methods in terms of average accuracy across datasets. For instance, on Citeseer, GCTD improves over GCDM (the second-best method) by 4.1%. On Pubmed, it surpasses SFGC by 2.1%, and on Cora, it leads by 1.3%. Even on Flickr, where all methods struggle, GCTD still achieves the best average, showing a 0.5% gain over GCond.

E.2 Ablation on the synthetic node assignment

We conducted an ablation study to evaluate the effectiveness of the synthetic node assignment method we leveraged. Specifically, we used the *argmax* operation on each row of the factor matrix $\mathbf{U}$, assigning the selected node to the corresponding synthetic node. We then compared the performance of this approach to that of K-Means, the default method in GCTD. In this ablation study, we follow the same settings used in the previous experiment, employing Citeseer, Cora, Pubmed, and Flickr with 1.8%, 2.6%, 0.15%, and 0.5% condensation ratios, respectively.

As shown in Figure 3, while argmax provides a simple and intuitive method for generating synthetic nodes, its simplicity can sometimes limit its ability to produce high-quality synthetic nodes. Specifically, we can observe that only selecting the maximum value in each row as the cluster membership is insufficient to capture the necessary information for the condensed graph. On the other hand, even though KMeans is computationally complex compared to argmax, it offers greater performance. It typically obtains better clustering of nodes by effectively grouping them into synthetic nodes based on the optimized factor matrix. This matrix, derived during the decomposition process, captures co-clustered nodes exhibiting similar patterns, leading to more meaningful synthetic representations Gujral and Papalexakis [2018].

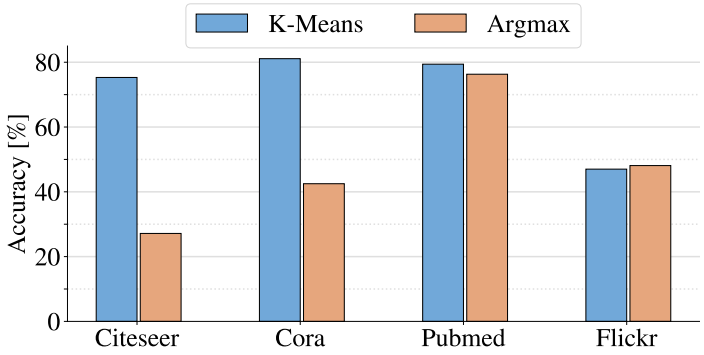
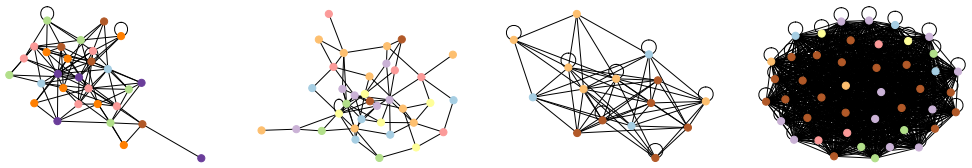


Figure 3: Accuracy of GCTD with K-Means and Argmax as the method employed to compute the synthetic node assignments from factor matrix $\mathbf{U}$. In this ablation study, we used Citeseer, Cora, Pubmed, and Flickr with the condensation ratio set to 1.8%, 2.6%, 0.15%, and 0.5%, respectively.

E.3 Visualization of the condensed graphs

We visualize the condensed graphs generated with the smallest condensation ratio for Citeseer, Cora, Pubmed, and Flickr in Figure 4. Several key observations emerge from these visualizations. First, GCTD produces graphs with well-defined structures and introduces self-loops for certain nodes. Additionally, it reduces the homophily present in the original graphs, indicating a diminished reliance on this property. This is particularly intriguing, given that the GNNs we used typically depend on graph homophily for optimal performance, as discussed in prior work Zhu et al. [2020]. Finally, GCTD generates a complete graph for Flickr, implying that the GNN now relies on information from every node to compute individual node representations. As a result, the GNN relies on the weights and features of nodes to differentiate between information from various neighbors.



(a) Citeseer - 0.9% (b) Cora - 1.3% (c) Pubmed - 0.08% (d) Flickr - 0.1%

Figure 4: Visualization of condensed graphs generated by GCTD. Each node represents a synthetic node, with its color indicating the corresponding class.

E.4 Comparison with original graphs

Table 5 presents a comparison of various properties between the original graphs and the condensed graphs generated by GCTD. The results show that GCTD achieves comparable performance on Citeseer, Cora, Pubmed, and Flickr, while significantly reducing the number of nodes and edges, as well as requiring less storage. Furthermore, the condensed graphs are denser than their original counterparts, and a notable behavior is observed on Flickr, where the learned graph is complete.

Table 5: Comparison between the condensed graphs generated by GCTD and original graphs.

	Citeseer (0.9%)		Cora (1.3%)		Pubmed (0.08%)		Flickr (0.1%)		Ogbn-arxiv (0.05%)	
	Whole	GCTD	Whole	GCTD	Whole	GCTD	Whole	GCTD	Whole	GCTD
Accuracy	71.7	76.8	81.4	81.4	77.1	79.9	47.1	48.4	71.3	58.2
#Nodes	3,327	30	2,708	35	19,717	15	44,625	44	169,343	84
#Edges	4,732	93	5,429	72	44,338	67	218,140	990	1,116,243	37
Sparsity	0.09%	22.91%	0.15%	12.1%	0.01%	63.8%	0.02%	100%	0.09%	1.1%
Storage	47.1 MB	0.51 MB	14.9 MB	0.23 MB	40 MB	0.05 MB	86.8 MB	0.17 MB	100.4 MB	0.08 MB

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: The abstract states that a multi-view tensor-decomposition approach (GCTD) condenses graphs while preserving GNN performance, and the experiments on six datasets substantiate that claim.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We acknowledge settings where GCTD is not best (e.g., Ogbn-arxiv) and that single-view can underperform multi-view (Cora exception).

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: The paper provides algorithmic details and a complexity analysis, but no formal theorems or proofs.

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: An anonymized code repo is provided; datasets, baselines, training protocol, and additional details are presented in the Appendix, enabling reproduction.

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: Code is provided via Anonymous Github, and all datasets used are public (PyG/GraphSAINT/OGB with official public splits). Library requirements and a Wandb configuration file example for training the model are available in the anonymous repository.

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: In both the main text and the Appendix, we report the convergence criteria for decomposition, the GNN's hyperparameter settings, training length, model selection strategy, and the use of official dataset splits.

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: We report results as averages over ten runs and include standard deviations.

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: We provide a runtime table and specify hardware (single NVIDIA RTX A6000).

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: The study uses public benchmarks with official splits and does not involve human subjects or sensitive data; no ethical risks beyond standard ML practice are implied.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We provide a paragraph after the conclusion section discussing the impacts of our work.

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: No high-risk models or scraped datasets are released; work relies on standard public graph benchmark.

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 cite baselines and datasets papers throughout the paper, providing appropriate references.

13. **New assets**

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

    Answer: [Yes]

    Justification: A new code asset is released via an anonymized repository; paper and appendices document methods and settings that accompany the code. Also, a Wandb configuration file is provided to run the model.

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: This work does not involve crowdsourcing nor research with human subjects.

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: This work does not involve crowdsourcing nor research with human subjects.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.