GC-Bench: An Open and Unified Benchmark for Graph Condensation

Qingyun Sun[1*], Ziying Chen[1*], Beining Yang[2], Cheng Ji[1], Xingcheng Fu[3], Sheng Zhou[4], Hao Peng[1], Jianxin Li[1#], Philip S. Yu[5]

[1] Beihang University, [2] University of Edinburgh, [3] Guangxi Normal University, [4] Zhejiang University, [5] University of Illinois, Chicago

Overview of GC-Bench

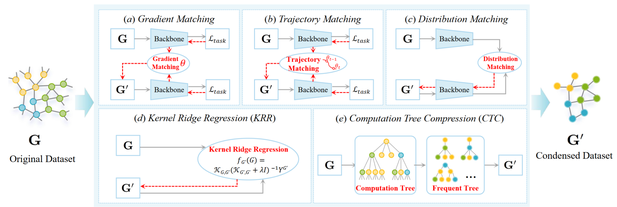


Figure 1: The GC methods can be broadly divided into two categories: The first category depends on the backbone model, refining the condensed graph by aligning it with the backbone's *gradients* (*a*), *trajectories* (*b*), and output *distributions* (*c*) trained on both original and condensed graphs. The second category, independent of the backbone, optimizes the graph by matching its distribution with that of the original graph data (*d*) or by identifying frequently co-occurring computation trees (*e*).

Graph condensation (GC) works are in a promising direction and show tremendous performance by condensing the large-scale dataset into a generated smaller one preserving the original datasets' valuable information. We introduce GC-Bench, an open and unified benchmark to systematically evaluate existing graph condensation methods focusing on the following aspects. ***Effectiveness***: the progress in GC, and the impact of structure and initialization on GC; ***Transferability***: the transferability of GC methods across backbone architectures and downstream tasks; ***Efficiency:*** the time and space efficiency of GC methods. The contributions of GC-Bench are as follows:

- ***Comprehensive benchmark and Key findings***. GC-Bench systematically integrated 12 representative and competitive GC methods on both node-level and graph-level by unified condensation and evaluation, giving multi-dimensional analysis in terms of effectiveness, transferability, and efficiency
- ***Open-sourced benchmark library and future directions***. GC-Bench is open-sourced and easy to extend to new methods and datasets, which can help identify directions for further exploration and facilitate future endeavors.

Table 1: An overview of GC-Bench

Methods	
Traditional core-set methods	*Random*, *Herding* [36], *K-Center* [30]
Gradient matching	*GCond* [15], *DosCond* [14], *SGDD* [42]
Trajectory matching	*SFGC*[47], *GEOM* [45]
Distribution matching	*GCDM* [20], *DM* [24, 22]
Kernel Ridge Regression	*KiDD* [41]
Computation Tree Compression	*Mirage* [11]
Datasets	
Homogeneous datasets	*Cora* [16], *Citeseer* [16], *ogbn-arxiv* [13], *Flickr* [43], *Reddit* [12]
Heterogeneous datasets	*ACM* [46], *DBLP* [7]
Graph-level datasets	*NCI1* [32], *DD* [4], *ogbg-molbace* [13], *ogbg-molbbbp* [13], *ogbg-molhiv* [13]
Downstream Tasks	
Node-level task	Node classification, Link prediction, Anomaly detection
Graph-level task	Graph classification
Evaluations	
Effectiveness	Performance under different condensation ratios, Impact of structural properties, Impact of initialization mechanism
Transferability	Different downstream tasks, Different backbone model architectures
Efficiency	Time and memory consumption

Paper：https://arxiv.org/abs/2407.00615

Code：https://github.com/RingBDStack/GC-Bench

Research Questions and Experimental Results

RQ1: How much progress has been made by existing GC methods?

- **Key Takeaways 1:** Current node-level GC methods can achieve nearly lossless condensation performance. There is still a significant gap between graph-level GC and whole dataset training.
- **Key Takeaways 2:** A large condensation ratio does not necessarily lead to better performance with current methods.

Table 2: **Node classification accuracy (%)** (mean±std) across datasets with varying condensation ratios r. $\mathcal{H}$ denotes the homophily ratio [29]. The best results are shown in **bold** and the runner-ups are shown in underlined. Red color highlights entries that exceed the whole dataset performance.

	Dataset	Ratio(r)	Random	Herding	K-Center	GCDM	DM	DosCond	GCond	SGDD	SFGC	GEOM	Whole Dataset
			Traditional Core-set Methods			Distribution		Gradient			Trajectory		
Homogeneous	*Cora* ($\mathcal{H}$=0.81)	0.26%	31.6±1.2	48.6±1.4	48.6±1.4	39.0±0.4	35.6±3.9	78.7±1.3	**79.8**±0.7	78.6±2.7	78.8±1.2	50.1±1.2	80.8±0.3
		0.52%	47.1±1.7	56.0±0.6	44.7±2.7	42.3±0.7	38.5±4.4	81.1±0.1	80.2±0.7	**81.2**±0.5	79.2±1.8	70.5±0.9	
		1.30%	62.3±1.0	69.9±0.8	62.0±1.3	63.4±0.1	63.2±0.5	81.2±0.4	80.2±2.2	**81.6**±0.5	79.4±0.7	78.5±1.9	
		2.60%	72.4±0.5	74.2±0.6	73.5±0.7	73.4±0.2	72.6±0.7	79.6±0.2	80.5±0.3	**81.8**±0.2	81.7±0.0	76.1±4.7	
		3.90%	74.6±0.6	76.0±0.6	77.4±0.3	76.4±0.5	75.3±0.1	78.9±0.1	79.8±0.4	**82.1**±0.8	81.5±0.5	78.3±2.0	
		5.20%	77.1±0.6	76.7±0.4	76.5±0.6	78.4±0.0	77.9±0.2	79.7±0.6	77.8±3.6	**82.0**±1.4	81.4±0.0	80.4±0.8	
	Reddit ($\mathcal{H}$=0.78)	0.05%	45.2±1.7	54.6±3.4	50.7±2.2	61.1±0.4	60.6±0.7	50.1±1.8	84.4±2.6	83.6±1.7	**87.2**±2.1	73.6±1.2	94.0±0.2
		0.10%	53.3±1.6	62.3±1.6	50.1±1.3	72.3±0.5	68.7±1.2	59.5±2.8	**84.7**±1.9	**83.8**±2.2	81.5±1.2	76.2±3.5	
		0.20%	65.1±1.2	69.4±2.1	54.2±1.9	80.8±0.6	76.7±0.1	65.8±0.4	86.3±2.4	88.3±0.5	**90.4**±0.9	89.9±0.6	
		0.50%	76.1±1.8	81.4±0.7	67.8±1.3	86.4±0.1	84.2±0.2	65.8±0.4	87.9±2.7	91.0±0.1	**91.7**±0.2	**91.7**±0.3	
		1.00%	84.9±0.5	85.9±0.3	74.6±0.8	OOM	78.9±0.7	78.3±1.3	87.2±1.3	85.9±0.9	**91.8**±0.2	OOM	
		5.00%	**92.3**±0.1	91.9±0.1	88.4±0.3	OOM	OOM	OOM	OOM	OOM	91.9±0.7	OOM	
	Citeseer ($\mathcal{H}$=0.74)	0.18%	39.3±1.9	36.6±2.6	36.6±2.6	30.6±2.5	32.1±1.5	71.8±0.1	71.3±0.6	71.7±3.2	70.7±1.1	63.7±0.4	71.7±0.0
		0.36%	44.1±2.0	41.5±2.6	38.7±3.2	36.2±2.0	43.3±1.3	73.1±0.3	70.4±1.8	**73.3**±3.3	70.9±1.6	69.8±0.1	
		0.90%	49.3±1.0	56.5±1.3	51.4±1.4	52.6±1.1	53.7±0.3	**73.0**±0.4	70.7±0.9	72.8±0.6	70.7±0.5	70.5±0.3	
		1.80%	57.4±0.7	68.4±0.8	61.4±1.2	64.7±0.6	66.4±0.2	71.6±0.6	70.2±1.4	**73.6**±0.6	70.4±1.2	67.5±0.6	
		2.70%	66.2±1.0	68.1±0.8	67.5±0.8	69.2±1.6	68.5±0.0	71.7±1.0	67.7±0.2	**72.1**±0.5	71.0±1.0	70.9±0.5	
		3.60%	69.2±0.3	69.1±0.7	69.3±0.7	69.3±0.4	70.4±0.4	72.5±0.3	68.4±1.6	**73.3**±1.2	70.6±0.1	70.7±0.1	
	ogbn-arxiv ($\mathcal{H}$=0.65)	0.05%	10.7±0.9	32.7±0.9	36.9±1.2	51.1±0.2	40.4±3.0	59.5±1.0	60.0±0.0	59.7±0.2	**68.2**±2.3	64.5±0.9	71.5±0.0
		0.10%	36.5±1.1	41.6±1.0	40.3±0.9	55.9±1.5	47.7±2.0	60.9±0.8	59.5±1.1	56.1±3.4	**69.1**±0.7	64.6±1.4	
		0.20%	43.2±0.8	48.9±0.8	42.7±1.0	59.3±0.8	47.1±0.7	62.2±0.4	61.4±0.7	60.9±0.9	69.0±0.4	64.3±0.6	
		0.50%	48.3±0.4	51.4±0.1	48.3±0.5	61.0±0.1	57.7±0.2	62.1±0.3	63.0±0.9	63.2±0.2	67.1±0.6	**67.2**±0.0	
		1.00%	51.2±0.4	52.0±0.4	50.3±0.5	61.0±0.0	59.8±0.2	61.8±1.3	62.8±0.0	62.0±2.8	68.4±1.3	**69.1**±0.3	
		5.00%	57.2±0.2	56.1±0.2	56.5±0.4	OOM	OOM	OOM	OOM	OOM	69.7±0.1	70.6±0.3	
	Flickr ($\mathcal{H}$=0.24)	0.05%	40.5±0.4	41.8±0.6	43.9±0.7	45.8±0.4	44.7±0.5	40.7±1.1	44.2±2.3	**46.6**±0.1	44.3±0.4	44.8±0.8	46.8±0.2
		0.10%	42.0±0.6	41.4±0.6	41.4±0.8	47.5±0.2	45.4±0.1	43.2±0.0	44.2±1.4	46.8±0.2	44.0±1.5	45.5±0.7	
		0.20%	43.1±0.2	41.5±0.7	41.4±0.8	48.8±0.1	42.8±0.6	43.9±0.6	44.4±1.4	46.8±0.3	41.4±5.6	46.1±0.6	
		0.50%	43.1±0.3	44.4±0.3	42.6±0.7	49.1±0.3	48.6±0.3	45.0±0.3	44.7±0.6	45.5±0.9	46.5±0.5	47.1±0.1	
		1.00%	43.0±0.4	44.5±0.6	43.2±0.2	49.3±0.1	49.6±0.3	46.0±0.3	45.1±0.5	47.2±0.2	46.6±0.0	47.0±0.4	
		5.00%	45.2±0.3	44.7±0.4	45.5±0.2	OOM	OOM	OOM	OOM	OOM	45.7±0.6	47.1±0.3	
Heterogeneous	*ACM*	.003%	83.9±1.7	82.5±1.6	76.3±2.4	84.8±0.3	84.8±0.3	89.3±0.6	80.6±0.8	90.3±2.3	**92.2**±0.2	73.4±0.5	91.7±0.4
		.007%	84.7±1.9	84.5±0.8	80.9±1.6	87.1±0.2	87.1±0.2	89.8±0.0	81.9±1.6	91.2±1.9	**91.6**±1.2	73.4±0.5	
		.013%	86.8±0.9	88.6±0.6	87.1±1.2	90.4±0.0	90.4±0.0	89.9±0.1	80.4±3.4	91.9±3.4	**92.0**±0.4	79.3±3.0	
		.033%	87.7±1.4	88.3±1.0	88.6±0.7	91.6±0.2	91.6±0.2	90.9±0.1	89.0±1.2	91.2±1.8	**92.2**±0.2	82.8±1.0	
		.066%	89.1±0.6	87.9±1.1	88.8±1.0	91.6±0.2	91.6±0.2	91.4±0.2	86.6±2.1	**91.9**±2.0	91.8±1.0	71.1±0.6	
		.332%	89.3±0.9	89.4±0.6	88.7±0.9	91.3±0.2	91.3±0.2	91.0±0.5	89.3±0.6	89.4±0.4	**91.8**±2.3	80.9±2.6	
	DBLP	.002%	46.6±2.6	58.9±2.7	54.7±2.6	61.6±0.7	56.6±1.1	71.6±1.8	71.5±2.2	77.8±0.2	**81.9**±2.2	72.3±3.1	80.1±0.9
		.004%	57.0±1.4	62.9±0.9	54.9±2.9	60.4±1.1	62.0±1.5	75.4±2.6	76.2±2.1	80.9±1.4	**81.7**±3.5	72.3±0.5	
		.007%	67.5±0.8	61.3±1.6	60.9±0.4	70.7±0.3	68.3±0.6	71.7±4.4	73.6±0.7	81.1±0.6	**81.2**±0.4	67.3±1.0	
		.019%	68.0±1.1	65.7±1.2	64.5±1.1	73.6±0.2	73.5±0.4	76.6±0.6	74.6±1.1	77.9±0.3	**80.8**±2.1	72.0±0.9	
		.045%	67.6±1.5	65.0±1.1	66.6±1.5	74.6±0.1	75.2±0.1	76.9±1.8	76.4±0.8	78.5±0.6	**81.9**±1.3	71.6±2.3	
		.186%	65.7±1.0	66.3±0.8	66.0±1.0	OOM	OOM	OOM	OOM	OOM	**82.1**±0.3	69.9±0.9	

Table 3: **Graph classification performance on GIN** (mean±std) across datasets with varying condensation ratios r. The best results are shown in **bold** and the runner-ups are shown in underlined. Red color highlights entries that exceed the whole dataset values.

Dataset	Graph /Cls	Ratio(r)	Random	Herding	K-Center	DosCond	KiDD	Mirage	Whole Dataset
			Traditional Core-set methods			Gradient	KRR	CTC	
NCI1 Acc. (%)	1	0.06%	50.90±2.10	51.90±1.60	51.90±1.60	49.20±1.10	**61.40**±0.50	50.80±2.20	80.0±1.8
	5	0.24%	52.10±1.00	50.50±2.40	47.00±1.10	51.10±0.80	**63.20**±0.20	51.30±1.20	
	10	0.49%	55.60±1.90	61.80±1.50	49.40±1.80	50.30±1.30	**64.20**±0.10	51.70±1.40	
	20	0.97%	58.70±1.40	60.90±1.90	55.20±1.60	50.30±1.30	**60.90**±0.70	52.10±2.20	
	50	2.43%	61.10±1.20	59.00±1.50	62.70±1.50	50.30±1.30	**65.40**±0.60	52.40±2.70	
DD Acc. (%)	1	0.21%	49.70±11.30	58.80±6.10	58.80±6.10	46.30±8.50	71.30±1.50	**74.00**±0.40	70.1±2.2
	5	1.06%	40.80±4.30	58.70±5.80	51.30±5.30	57.50±5.60	**70.90**±3.10	-	
	10	2.12%	63.10±5.20	64.10±5.80	53.40±3.10	46.30±8.50	**71.50**±0.50	-	
	20	4.25%	56.40±4.30	67.00±2.60	58.50±5.70	40.70±0.00	**71.20**±0.90	-	
	50	10.62%	58.90±6.30	68.40±4.00	62.30±2.50	44.00±6.70	**71.80**±1.00	-	
ogbg-molbace ROC-AUC	1	0.17%	0.468±.045	0.486±.035	0.486±.035	0.512±.092	0.706±.000	0.590±.004	0.763±.020
	5	0.83%	0.312±.019	0.470±.042	0.553±.024	0.555±.079	**0.847**±.019	**0.838**±.022	
	10	1.65%	0.442±.028	0.532±.031	0.594±.019	0.536±.072	0.594±.000	0.419±.010	
	20	3.31%	0.510±.023	0.509±.052	0.512±.031	0.484±.080	0.640±.011	0.423±.011	
	50	8.26%	0.486±.020	0.625±.026	0.595±.026	0.503±.084	**0.723**±.011	-	
ogbg-molbbbp ROC-AUC	1	0.12%	0.510±.013	0.532±.015	0.532±.015	0.546±.026	0.616±.000	0.592±.004	0.635±.017
	5	0.61%	0.522±.014	0.546±.020	0.581±.022	0.519±.041	0.607±.005	0.431±.013	
	10	1.23%	0.508±.018	0.578±.017	0.619±.027	0.505±.028	0.663±.000	0.465±.036	
	20	2.45%	0.567±.010	0.533±.009	0.546±.012	0.493±.031	0.677±.001	0.610±.022	
	50	6.13%	0.595±.014	0.552±.018	0.594±.016	0.509±.015	**0.684**±.009	0.590±.031	
ogbg-molhiv ROC-AUC	1	0.01%	0.366±.087	0.462±.072	0.462±.072	0.674±.131	0.664±.016	**0.710**±.009	0.701±.028
	5	0.03%	0.501±.051	0.496±.044	0.519±.096	0.369±.175	0.657±.005	**0.703**±.012	
	10	0.06%	0.554±.031	0.458±.058	0.471±.054	0.457±.214	**0.632**±.000	0.513±.052	
	20	0.12%	0.621±.022	0.582±.027	0.627±.050	0.281±.007	0.648±.025	0.633±.048	
	50	0.30%	0.625±.062	0.600±.034	**0.680**±.049	0.455±.214	0.587±.038	0.588±.067	

**Mirage* cannot directly generate graphs with the required ratio. Parameter search aligns generated graphs with *DosCond* disk usage (see Appendix B.1). '-' denotes results unavailable due to recursive limits reached in MP Tree search.

RQ2: How do the potential flaws of the structure affect the GC performance?

- **Key Takeaways 3:** Existing GC methods primarily address simple graph data. There is significant room for improvement in preserving complex structural properties.

Table B4: Homophily ratio comparison of different condensed datasets

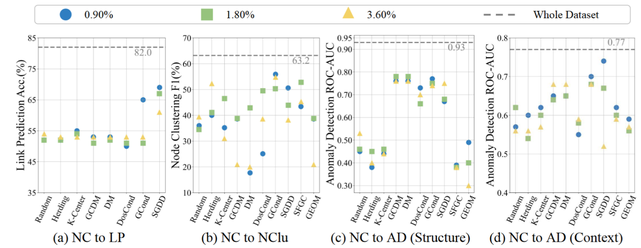
	Whole Dataset	Ratio (r)	GCDM	DM	DosCond	GCond	SGDD
Cora	0.81	1.30%	0.76↓	0.88↑	0.20↓	0.64↓	0.19↓
		2.60%	0.11↓	0.74↓	0.16↓	0.55↓	0.19↓
		5.20%	1.00↑	0.21↓	0.15↓	0.62↓	0.15↓
Citeseer	0.74	0.90%	0.16↓	0.75↑	0.19↓	0.57↓	0.14↓
		1.80%	0.08↓	0.30↓	0.20↓	0.36↓	0.19↓
		3.60%	1.00↑	0.34↓	0.15↓	0.22↓	0.15↓
Flickr	0.24	0.05%	0.28↑	0.29↑	0.25↑	0.28↑	0.32↑
		0.50%	0.29↑	0.22↓	0.08↓	0.28↑	0.30↑
		1.00%	0.36↑	0.18↓	0.06↓	0.28↑	0.26↑

Figure B.1: Visualization of the Condensed *Citeseer* (1.80%) Dataset. Only the top 20% of edges ranked by weight are visualized.

Figure B.2: Visualization of the Condensed *Cora* (2.60%) Dataset. Only the top 20% of edges ranked by weight are visualized.

RQ3: Can the condensed graphs be transferred to different types of tasks?

- **Key Takeaways 4:** All condensed datasets struggle to perform well outside the context of the specific tasks for which they were condensed.

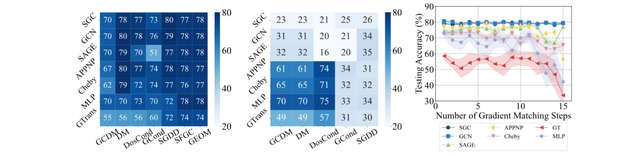


(a) NC to LP (b) NC to NClu (c) NC to AD (Structure) (d) NC to AD (Context)

Figure 2: **Cross-task performance on *Citeseer***. For all downstream tasks, the models are trained solely using data of graphs condensed by node classification. For anomaly detection (c, d), structural and contextual anomalies [3] are injected into both the condensed graph and the original graph.

RQ4: How does the backbone used for condensation affect the performance?

- **Key Takeaways 5:** Current GC methods exhibit significant performance variability when transferred to different backbone architectures. Involving the entire training process potentially may lead to encoding backbone-specific details in the condensed datasets..
- **Key Takeaways 6:** Despite their strong performance in general graph learning tasks, transformers surprisingly yield suboptimal results in graph condensation.



(a) Cross-arch. Acc. from *SGC* (b) Cross-arch. Acc. from *GTrans* (c) *GCond* across different steps.

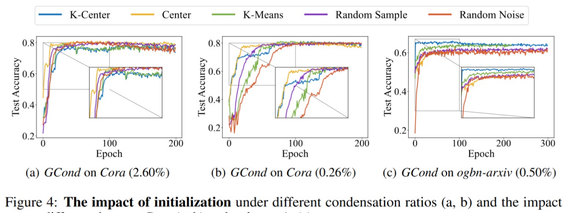
Figure 3: **Cross-architecture performance.** Using *SGC* and Graph Transformer (*GTrans*) to condense *Cora* with a 2.6% ratio, we then test the accuracy on various downstream architectures (a, b). Furthermore, we evaluate the influence of gradient matching steps on *GCond* (c).

RQ5: How does the initialization mechanism affect the performance of GC?

- **Key Takeaways 7:** Different datasets have their preferred initialization methods for optimal performance even for the same GC method.
- **Key Takeaways 8:** The initialization mechanism primarily affects the convergence speed with little impact on the final performance. The smaller the condensed graph, the greater the influence of different initialization strategies on the convergence speed.

RQ6: How efficient are these GC methods in terms of time and space?

- **Key Takeaways 9:** GC methods that rely on backbones and full-scale data training have large time and space consumption.



(a) *GCond* on *Cora* (2.60%) (b) *GCond* on *Cora* (0.26%) (c) *GCond* on *ogbn-arxiv* (0.50%)

Figure 4: **The impact of initialization** under different condensation ratios (a, b) and the impact across different datasets *Cora* (a, b) and *ogbn-arxiv* (c).

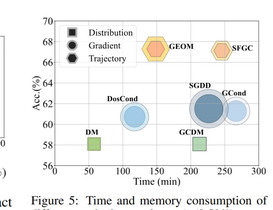


Figure 5: Time and memory consumption of different methods on *ogbn-arxiv* (0.50%).