Data Augmentation in Graph Neural Networks: The Role of Generated Synthetic Graphs

Sümeyye Baş[1,3], Kıymet Kaya[2,3,4], Resul Tugay[5], and Şule Gündüz Öğüdücü[1,3]

[1] Istanbul Technical University, Department of Artificial Intelligence and Data Engineering, Istanbul, Turkey
[2] ITU, Department of Computer Engineering, Istanbul, Turkey
[3] ITU, AI Research and Application Center, Istanbul, Turkey
[4] BTS Group, Istanbul, Turkey
[5] Gazi University, Department of Computer Engineering, Ankara, Turkey
bass20@itu.edu.tr, kayak16@itu.edu.tr, resultugay@gazi.edu.tr, sgunduz@itu.edu.tr

Abstract. Graphs are crucial for representing interrelated data and aiding predictive modeling by capturing complex relationships. Achieving high-quality graph representation is important for identifying linked patterns, leading to improvements in Graph Neural Networks (GNNs) to better capture data structures. However, challenges such as data scarcity, high collection costs, and ethical concerns limit progress. As a result, generative models and data augmentation have become more and more popular. This study explores using generated graphs for data augmentation, comparing the performance of combining generated graphs with real graphs, and examining the effect of different quantities of generated graphs on graph classification tasks. The experiments show that balancing scalability and quality requires different generators based on graph size. Our results introduce a new approach to graph data augmentation, ensuring consistent labels and enhancing classification performance.

Keywords: generative models · graph neural networks · data augmentation · graph sequentialization

1 Introduction

Graphs serve as important representations of interrelated data in various fields, including social networks and chemical sciences, due to their ability to encapsulate complex relationships and facilitate critical tasks such as predictive modeling. Obtaining high-quality graph representations is vital for revealing underlying interrelated patterns and phenomena in fields that rely on graph-related data [1].

However, the progress in this area is being hampered by the lack of useful big datasets and consistent evaluation processes. Despite significant progress in data availability in recent years, this development remains limited in various

application domains for privacy and security reasons [2, 3]. This indicates that future models may intentionally or unintentionally resort to generated synthetic data.

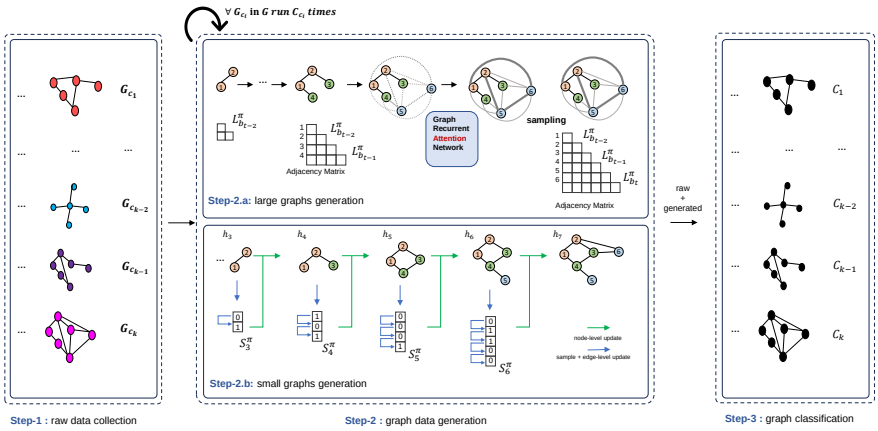

Fig. 1: Graph Classification with Graph Size-Aware Data Augmentation.

The generation of synthetic data is becoming more and more popular for a variety of reasons such as improving data diversity, and enhancing privacy and security [4]. In many domains, it is often difficult to gather sufficient data tailored to specific tasks. Data augmentation emerges as a crucial solution to this challenge. In research areas, involving graph-related data, leveraging proper synthetic graphs proves to be more efficient, cost-effective, and straightforward compared to acquiring additional real-world datasets. Moreover, it offers enhanced privacy protections, particularly in sensitive industries such as healthcare and finance [5].

The *Graph Classification with Graph Size-Aware Data Augmentation* framework we propose in this study is presented in Fig. 1. It's important to clarify that our approach involves data augmentation, encompassing both synthetic and real-world data, such that, the amount of the training set is increased, rather than relying solely on synthetic data. The proposed approach emerges from extensive literature research, revealing the diverse capabilities of different graph generators [6, 7]. Recognizing that raw datasets possess varying characteristics, which can influence generator performance, we examined these generators especially for graph size sensitivity which led us to introduce a novel framework aimed at enhancing graph classification. The framework operates by splitting datasets according to graph labels, training individual generator models for each class, and leveraging both original and generated graphs in the graph classification process. Experimental results demonstrate that this tailored approach significantly boosts classification performance, particularly for datasets with a limited number of graphs.

The main contributions of this study can be summarized as follows:

- Examining the usability of generated synthetic data to improve graph classification model performances.
- Investigating how the ratio of generated data affects the prediction capabilities of the models by training the models with different proportions of synthetic data.
- Addressing graph labeling issue in graph data augmentation.
- Questioning whether it is worth spending time and resources to collect additional real data, comparing the improvements provided by real-world and synthetic data for data augmentation.

The rest of the paper is organized as follows. Section 2 presents related works and Section 3 gives details of the methodology. Section 4 presents the experimental results of the study. Lastly, Section 5 concludes the paper.

2 Literature Review

Data augmentation has recently drawn significant attention in the field of machine learning, especially for deep learning models, [8–10], thanks to its ability to enhance model performance and generalization by incorporating additional training data. Data augmentation is commonly used in computer vision and text applications. Data augmentation with images is relatively simpler due to the Euclidean nature and well-organized structure of image instances. Pixel matrices can be easily transformed using common rule-based techniques like rotation, scaling, and flipping preserving the labels [11]. Similarly, for text-level data augmentation rule-based data augmentation approaches including random insertion, random deletion, and synonym substitution are quite effective [12]. AugGPT rephrases sentences in the training data through rule-based approaches to increase the training data size and ensure accurate labeling in the generated text data [13].

However, compared to text and image data types graphs are irregular and non-Euclidean. Therefore, even small changes in the structure may result in a loss of inter-related information, making the augmentation process more complex for graph data. Graph Data Augmentation methods can be broadly examined under two headings: rule-based and learned (AI-based) approaches. Among these, rule-based methods apply predefined rules to modify graph data, such as random edge removal and graph clipping while learned methods, such as graph structure learning and graph rationalization, exploit learnable parameters for data augmentation that can be trained independently or with downstream tasks.

The conventional rule-based graph augmentation techniques widely applied in the literature, such as arbitrary node removal, edge modification, or the occlusion of node characteristics, rely on random alterations of network structures or attributes. However, these arbitrary changes often compromise label invariance by inadvertently damaging significant label-related information, thus failing to generate appropriate graph data and improve graph prediction models'

performance in practical applications. To address these limitations, GraphAug is offered as a solution by computing label-invariant augmentations through an automated approach, thereby safeguarding crucial label-related data within graph datasets [14]. Furthermore, Sui et al. proposed Adversarial Invariant Augmentation (AIA), a strategy aimed at mitigating covariate shifts in graph representation learning [15]. Yue et al. advanced a label-invariant augmentation method for graph-structured data in graph contrastive learning, generating augmented samples in challenging directions within the representation space while maintaining the original sample labels [16].

Despite advances in data augmentation, studies on using generated data for prediction in graph neural networks are limited in the literature [17]. Zero-shot image classification was performed using data generated with models that had been extensively trained. The performance of prediction models using synthetic data and real data for image classification is compared and similar accuracy values are observed [18]. The effect of synthetic graphs on the model performance of node classification algorithms was examined, and small improvements in performance were obtained through pre-training using graphs with similar characteristics. Sun et al. presented the MoCL framework for learning molecular representations, utilizing both local and global domain expertise to guide the augmentation procedure and guarantee variation without changing graph semantics, as shown on several molecular datasets [19]. Using social network graph datasets, Tang et al. performed cosine similarity-based cross-operation on the initial characteristics to produce additional graph features for node classification tasks [20]. In this study, we investigate the impact of graph data augmentation on the graph classification task. Generator methods proposed in the literature for data augmentation offer different advantages. Our work differs from its contemporaries in that it appropriately combines two state-of-the-art generator models, according to their advantages in terms of graph size.

3 Methodology

The overall structure of our proposed method is depicted in Fig. 1. First, for the graph classification task, the data $G = G_1, ..., G_s$ with E edges and V nodes, is grouped as small or large according to the average number of nodes and edges of the graphs it contains. In this study, the average number of nodes is fifty and above, or an edge number of thousand two hundred twenty-five (edge numbers for a fully connected simple graph with fifty nodes), and above is determined as large, otherwise small. In Step 2 of Fig. 1, the appropriate generation model is selected according to whether the graphs in the data are small or large, and for each graph class C_i, the desired number of synthetic graphs for that class C_{C_i} is generated for data augmentation. Here we recommend GRAN for data consisting of large graphs and GraphRNN for small ones. GRAN, with its focus on balancing efficiency and quality—particularly through its stride parameter—may experience a reduction in generation quality for smaller graphs when compared to GraphRNN. On the other hand, GRAN's methodology is optimized for handling

larger graphs, whereas GraphRNN's architecture and training process are better suited for generating smaller-scale graphs [6] and mostly give out of memory error for larger graphs despite efforts to enhance scalability through techniques like bread-first-search (BFS) node ordering scheme. Graph generation methods GraphRNN, GRAN, and all the classification methods used during the experiments are detailed in the subheadings, and the code repository is also available here[6].

3.1 Graph Data Generation

Learning a distribution $p_{model}(G)$ over graphs is the aim of generative model learning. This is achieved by sampling a collection of observed graphs $G = G_1, ..., G_s$ from the data distribution $p(G)$, where each graph in G may vary in the number of nodes and edges [7]. Instead of directly acquiring knowledge about the probability distribution $p(G)$, which is difficult to define precisely the representation of the sample space, an auxiliary random variable π is sampled to represent node ordering as sequences. This transforms the graph generation process into the generation of node and edge sequences, where nodes and edges are generated autoregressively. An adjacency matrix with a node ordering π maps nodes to rows and columns of the matrix enabling each graph in G

to be depicted by the adjacency matrix $A^{\pi} \in \mathbb{R}^{n \times n}$ 1.

$$A^{\pi}_{i,j} = \mathbb{1}\left[\left(\pi\left(v_i\right), \pi\left(v_j\right)\right) \in E\right]. \tag{1}$$

The aforementioned generator models were designed to work with simple graphs $G = (V, E)$. Initially, graph nodes and edges are represented as sequences and sequences of sequences using a mapping f_S, respectively. For a graph G sampled from $p(G)$ with n nodes under a node ordering π, the sequence S_{π} is obtained as in Equation 2, where S^{π}_i is an adjacency vector representing the edges between node $\pi(v_i)$ and the preceding nodes $\pi(v_j)$, $j \in \{1, ..., i-1\}$, already present in the graph 3.

$$f_S(G, \pi) = (S^{\pi}_1, ..., S^{\pi}_n) \tag{2}$$

$$S^{\pi}_i = \left(A^{\pi}_{1,i}, \ldots, A^{\pi}_{i-1,i}\right)^T, \forall i \in \{2, \ldots, n\} \tag{3}$$

$$p(G) = \sum_{S^{\pi}} p\left(S^{\pi}\right) \mathbb{1}\left[f_G\left(S^{\pi}\right) = G\right] \tag{4}$$

In the case of undirected graphs, S^{π} uniquely determines a graph G, denoted by the mapping $f_G(\cdot)$, where $f_G(S^{\pi}) = G$. This sequentialization process allows generators to observe S^{π} and learn about its probability distribution, $p(S^{\pi})$, which can be analyzed sequentially as S^{π} exhibits a sequential nature. During inference time, generators can derive samples of G without explicitly calculating $p(G)$ by sampling S^{π}, which corresponds to G through the function f_G. With these concepts, $p(G)$ can be expressed as a marginal probability distribution of the joint distribution $p(G, S^{\pi})$ in Equation 4, where $p(S^{\pi})$ is the distribution that the generator aims to learn.

[6] https://github.com/sumeyyebas/AIGraphAugmentation

Large Graphs Generation: GRAN [21] The overall procedure of a generation phase with GRAN is illustrated in Step-2.a of Fig. 1. GRAN promises to provide a strong autoregressive conditioning between the graph's generated and to-be-generated portions as attention-based GNN helps better distinguish multiple newly added nodes. While expressing networks as adjacency matrices, some matrices remain unchanged under certain permutations, resulting in symmetry. To solve this, GRAN constructs a set of symmetric permutations as in Equation 5 and develops a surjective function u that maps permutations to symmetric permutations. Thus, for a graph G, different adjacency matrices for all permutations are modeled. However, for undirected graphs, it is enough to just model the lower triangular portion of the adjacency matrix L^{π}. GRAN generates the lower triangular component L^{π} block by block, adding one block of nodes and associated edges at a time t. This procedure considerably decreases auto-regressive graph creation decisions by a factor of $O(N)$, where $N = |V|$.

$$\Delta(A^{\pi}) = \left\{\tilde{\pi} \mid A^{\tilde{\pi}} = A^{\pi}\right\} \tag{5}$$

$$b_t = \{B(t-1)+1, ..., Bt\}. \tag{6}$$

$$p(L^{\pi}) = \Pi_{t=1}^{T} p(L_{b_t}^{\pi} | L_{b_1}^{\pi}, \cdot, L_{b_{t-1}}^{\pi}) \tag{7}$$

GRAN creates one block of B rows of L^{π} at a time. The t-th block consists of rows with indices as in Equation 6. The number of steps required to create a graph is therefore $T = O(N/B)$. The conditional probability in Equation 7 determines the likelihood of producing the current block. The probability function it needs to learn becomes a long conditional probability as it uses previous blocks to infer the next block. To avoid long-term bottlenecks and use the structural features of graphs, GRAN prefers GNNs over RNNs.

Small Graphs Generation: GraphRNN [7] In GraphRNN, the graph sequentialization is followed by constructing a scalable auto-regressive model that is suitable for small-medium size of graphs and can benefit from graph structure. Its generation process is shown in the Step-2.b of Fig. 1. GraphRNN can be viewed as a hierarchical model where new nodes are constructed by a graph-level RNN and the edges of each newly formed node are generated by an edge-level RNN, all while maintaining the state of the graph.

$$h_i = f_{\text{trans}}(h_{i-1}, S_{i-1}^{\pi}) \tag{8}$$

$$\theta_i = f_{\text{out}}(h_i) \tag{9}$$

$$S^{\pi} = f_S(G, BFS(G, \pi)) \tag{10}$$

The probability distribution $p(S_i\pi | S_{<i}\pi)$ for each i is intricate, requiring an understanding of how node $\pi(v_i)$ connects to preceding nodes based on the previously added nodes. GraphRNN suggests parameterizing $p(S_i\pi | S_{<i}\pi)$ with a neural networks model ensuring scalability with share weights across all time

steps. GraphRNN employs an RNN comprising a state-transition function f_{trans} and an output function f_{out} as in Equations 8 and 9, where h_i in $\mathbb{R}^d$ represents the state encoding of the generated graph up to this point, S_{i-1} is the adjacency vector for the most recently generated node $i-1$, and i denotes the distribution of the adjacency vector for the next node (i.e., S_i follows distribution P_o). Generally, f_{trans} and f_{out} can be any neural network, and P_o can be any distribution over binary vectors.

A key finding of the GraphRNN technique is that, without sacrificing generality, it learns to produce graphs using breadth-first-search (BFS) node orderings instead of learning to generate graphs under any conceivable node permutation. BFS also provide a unique representation of graphs. Therewith, Equation 2 is changed to Equation 10, with the deterministic BFS function represented by $BFS(\cdot)$. Specifically, this BFS function takes a random permutation i as its input, selects node $v1$ as the starting point, and appends each node's neighbors to the BFS queue following the order given by the permutation. It should be noted that the BFS function is many-to-one, meaning that multiple permutations can result in the same ordering once the BFS function is executed.

3.2 Graph Classification

GraphSAGE (Graph Sample and Aggregation) [22] creates node embeddings by sampling and aggregating data from each node's neighborhood. GraphSAGE generates robust graph categorization representations by combining information from several layers of the graph.

GIN0: GIN (Graph Isomorphism Network) generates node embeddings by using message-passing procedures and a learnable set function through a multilayer perceptron network. GIN is suitable for graph classification applications since it can capture higher-order graph structures. GIN0 sets the learnable ε parameter as 0 and depends rather on a set aggregation algorithm for updating node features. Therefore, it is computationally cheaper but less flexible [23].

GINWithJK [23] adds the concept of jumping knowledge (JK) concept to the GIN model. This concept combines representations from several layers to improve the final node embeddings. Therefore, better information flow between layers can be achieved.

GCNWithJK [24] Graph Convolutional Networks (GCN) iteratively aggregate information from neighboring nodes. GCN with Jumping Knowledge (GCNWithJK) directly merges node representations from several GCN layers. So, capturing both local and global graph structures becomes easier.

EdgePool [25] is an edge-level graph pooling technique utilized in Graph Neural Networks (GNNs). It efficiently reduces the size of the graph while maintaining important structural information by selectively aggregating edges. Along with this, every EdgePool layer outputs the mapping between every node in the old graph and every node in the newly-pooled graph. An inverse mapping from pooled nodes to unpooled nodes is produced during unpooling. It is possible to link this mapping via many pooling layers because each node is assigned to exactly one merged node.

4 Experimental Results

We observed the effect of synthetic graph data by GNNs on graph classification on six public datasets from TU[7] - three chemical compounds (MUTAGENICITY, ENZYMES, MUTAG), two social networks (COLLAB, TWITCH EGOS), and one protein interactions (DD). The descriptive statistics of these benchmark datasets are given in Table 1.

Table 1: Benchmark Graph Datasets Statistics

name	#graphs	avg. nodes	avg. edges	avg. degree	avg. density	avg. diameter	#classes	class distributions (%)
DD	518	258.74	650.23	4.99	0.0232	19.75	2	63-36
COLLAB	4001	73.40	2357.18	36.97	0.5076	1.87	3	51-32-15
TWITCH EGOS	101894	29.69	86.51	5.39	0.2020	2.00	2	53-46
MUTAGENICITY	3467	29.52	30.51	2.05	0.0921	9.92	2	55-44
ENZYMES	360	32.28	61.04	3.83	0.1592	11.30	6	16-16-16-16-16-16
MUTAG	152	17.95	19.79	2.19	0.1383	8.24	2	66-33

To investigate the impact of generated graphs on the graph classification task, we partitioned each dataset into three, ensuring the class distributions remained consistent: raw-data (80%), sub-real data (10%), and test data (10%). The raw-data serves as the baseline for comparison, representing the initial data available to researchers. In comparison, the sub-real data (R) imitates the supplementary data that researchers might acquire through additional time and resource investment in real-world scenarios. Lastly, the test data is designated to evaluate the performance of graph classification. Created with real-world application in mind, this strategic data partitioning enables the analysis of how generated graphs affect the accuracy of graph classification models.

Leveraging the initial raw-data available to researchers, we generated datasets comparable in size to the R, ensuring class distributions remained consistent across all generated sets as in all other sets. This approach allowed us to assess the impact of the real and generated data of the same size on graph classification performance. We further extended these experiments by generating data sets twice and three times the size of the R, to examine how change in the volume of generated graphs affects the model performance. Given the diversity in graph sizes within our datasets (see. varying sizes of avg. nodes, avg. edges in Table 1), we employed GRAN, known for its adaptability to large graphs, as our preliminary study indicated that GraphRNN encountered Out of Memory (OOM) errors with bigger graphs. Table 2 presents the prediction results of graph classifier models from various backgrounds for raw-data, with R added to the raw-data (w/ Real), and with the aforementioned generated data sets added to the raw-data namely $w/Gen._1$, $w/Gen._2$, $w/Gen._3$. According to the

[7] https://chrsmrrs.github.io/datasets/docs/datasets/

Table 2: Graph Classification Results - I

		GraphSAGE		GIN0		GINWithJK		GCNWithJK		EdgePool	
		Acc.	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**
DD	raw-data	0.630	6	0.605	6	0.545	27	0.630	5	0.660	4
	w/ Real $= \|R\|$	0.630	12	**0.665**	40	0.630	12	0.630	4	0.630	5
	w/ Gen.$_1 = \|R\|$	0.635	10	0.650	14	0.665	20	0.630	4	0.630	5
	w/ Gen.$_2 = \|R\| * 2$	**0.695**	31	**0.665**	13	0.665	23	0.630	4	0.630	4
	w/ Gen.$_3 = \|R\| * 3$	0.630	14	**0.665**	15	**0.685**	12	0.630	5	0.630	5
COLLAB	raw-data	0.576	7	0.718	40	0.707	11	0.534	8	0.650	5
	w/ Real	0.572	10	0.736	28	**0.736**	10	**0.693**	13	0.656	16
	w/ Gen.$_1$	**0.578**	8	0.705	9	0.718	11	0.534	6	0.666	14
	w/ Gen.$_2$	0.572	14	0.730	10	0.734	11	0.650	24	**0.701**	19
	w/ Gen.$_3$	0.569	13	**0.738**	33	0.716	8	0.631	13	0.627	23
TWITCH EGOS	raw-data	0.576	16	0.681	20	0.701	39	0.689	30	0.682	45
	w/ Real	0.576	7	0.694	31	0.700	30	0.684	13	OOM	
	w/ Gen.$_1$	0.578	10	**0.700**	24	**0.702**	23	0.695	21	OOM	
	w/ Gen.$_2$	**0.579**	12	0.699	12	**0.702**	28	0.696	19	OOM	
	w/ Gen.$_3$	0.575	9	0.699	13	0.690	42	**0.700**	35	OOM	
MUTAGENICITY	raw-data	0.597	20	0.703	15	**0.740**	28	0.551	4	0.701	28
	w/ Real	0.590	14	0.726	16	0.719	12	0.551	4	**0.719**	44
	w/ Gen.$_1$	0.593	29	0.698	10	0.726	22	0.551	4	0.717	30
	w/ Gen.$_2$	**0.609**	18	0.708	11	0.728	14	0.551	4	0.701	32
	w/ Gen.$_3$	0.586	12	**0.744**	32	0.712	31	0.551	4	0.685	22
ENZYMES	raw-data	0.141	9	0.166	7	0.191	46	0.166	13	0.166	10
	w/ Real	0.166	8	0.158	15	0.166	96	0.166	8	0.166	17
	w/ Gen.$_1$	**0.233**	14	0.200	13	**0.266**	41	0.166	10	**0.175**	15
	w/ Gen.$_2$	0.191	7	**0.233**	27	**0.266**	35	0.166	12	**0.175**	15
	w/ Gen.$_3$	0.175	11	0.208	18	0.158	22	0.166	11	0.166	15
MUTAG	raw-data	0.666	22	0.944	25	0.944	26	0.666	10	0.666	11
	w/ Real	0.666	32	0.833	54	0.833	15	0.666	11	0.666	10
	w/ Gen.$_1$	0.666	12	**0.944**	42	0.833	14	0.666	11	0.666	11
	w/ Gen.$_2$	0.666	11	**0.944**	51	**0.944**	24	0.666	11	0.666	10
	w/ Gen.$_3$	0.666	40	0.833	25	**0.944**	36	0.666	13	0.666	11

results in Table 2, the most accurate predictions for each dataset, as highlighted in the table, were achieved by incorporating the generated graphs and mostly with $w/Gen._2$.

In the second part of the experiments, we generated one thousand twenty-four graphs from each class in the datasets and drastically increased the proportion of the generated data size. Here, we aim to explore the feasibility of obtaining a more balanced dataset with a large number of samples with the proposed graph data generation method, avoiding the imbalanced data and data scarcity problems that affect the prediction performance of many learning algorithms. The results of these experiments, which we obtained by data generation with GraphRNN and GRAN on datasets MUTAGENICITY, ENZYMES, and MUTAG, which consist of small graphs and contain a small number of samples, are presented in Table 3. According to the results in Table 3, the most accurate predictions for each dataset, as highlighted in the table, were achieved with $w/Gen.(GraphRNN)$.

The overall summary of the results we obtained with the proposed textit-Graph classification with graph size-aware data augmentation framework is pre-

Table 3: Graph Classification Results - II

		GraphSAGE		GIN0		GINWithJK		GCNWithJK		EdgePool	
		Acc.	Epoch	Acc.	Epoch	Acc.	Epoch	Acc.	Epoch	Acc.	Epoch
MUTAGENICITY	raw-data	0.597	20	0.703	15	0.740	28	0.551	4	0.701	28
	w/ Real	0.590	14	**0.726**	16	0.719	12	0.551	4	**0.719**	44
	w/ Gen. (GraphRNN)	**0,611**	20	0,710	14	**0,744**	23	0,551	6	0,694	35
	w/ Gen. (GRAN)	0,551	7	0,689	21	0.728	23	0,551	5	0,551	6
ENZYMES	raw-data	0.141	9	0.166	7	0.191	46	0.166	13	0.166	10
	w/ Real	0.166	8	0.158	15	0.166	96	0.166	8	0.166	17
	w/ Gen. (GraphRNN)	0,166	12	**0,241**	49	**0,241**	65	0,166	14	0,166	15
	w/ Gen. (GRAN)	**0,183**	7	0,158	31	0,125	75	0,166	22	**0,175**	30
MUTAG	raw-data	0.666	22	**0.944**	25	0.944	26	0.666	10	0.666	11
	w/ Real	0.666	32	0.833	54	0.944	15	0.666	11	0.666	10
	w/ Gen. (GraphRNN)	0,388	70	0,888	85	**0,944**	79	0,666	14	**0,777**	52
	w/ Gen. (GRAN)	**0,944**	27	0,666	30	0,777	60	0,666	14	0,777	21

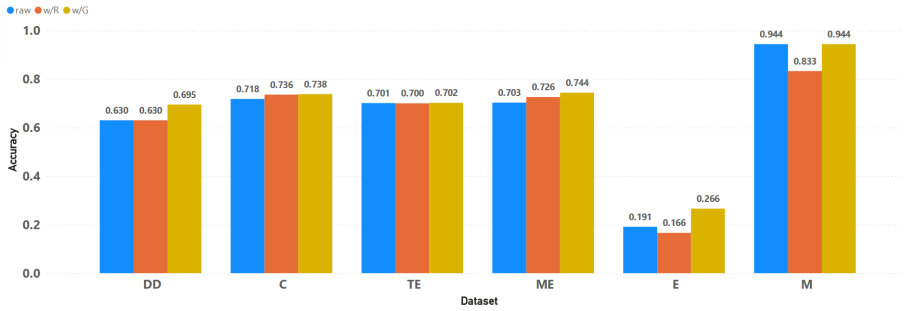

Fig. 2: Proposed Graph Classification with Graph Size-aware Data Augmentation Framework Results Summary

sented in Fig. 2. The results here reflect the raw-data, w/R and w/G accuracy values of the most accurate graph classifier for the relevant dataset (DD, COLLAB (C), TWITCH EGOS (TE), MUTAGENICITY (ME), ENZYMES (E), MUTAG (M)). Mirroring the observations of Touat et al. [6], our study reveals that particularly when working with medium to large-sized graphs, the graphs produced by GRAN are more analogous to the original graphs, however, GraphRNN has scalability problems not applicable to large graphs. However, while working with smaller graphs GRAN tends to overfit and its generation quality drops, hence GrapRNN is better.

5 Conclusion and Future Work

To conclude that, our study demonstrates the substantial impact of synthetic graph data on the performance of graph classification tasks across diverse datasets. The proposed *Graph classification with graph size-aware data augmentation* framework offers a flexible graph data generation process that is applicable for small, medium, and large-sized graphs. Moreover, the experiments involving a signifi-

cant increase in the proportion of generated data further highlighted the potential of our graph generation framework to mitigate issues of data imbalance and scarcity, especially in smaller datasets. This balance was crucial for achieving higher prediction accuracy, as evidenced by the superior performance of models utilizing balanced, generated datasets. Overall, our findings underscore the effectiveness of AI-driven data generation in enhancing graph classification tasks, paving the way for more accurate and reliable machine learning models in diverse applications.

For future work, we plan to investigate the reasons for the differences in the performance of the generated data with explainable AI methods and also to work on generating synthetic graphs for dynamic graphs.

Acknowledgments. This research is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) 1515 Frontier R&D Laboratories Support Program (project number 5239903) and the ITU Scientific Research Projects Fund under grant number YESAP-2024-45920.

Disclosure of Interests. There are no relevant financial or non-financial competing interests to report.

References

1. Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in Big Data*, 2:2, 2019. Article in Research Topic: When Deep Learning Meets Social Networks.
2. Leho Tedersoo, Reena Küngas, Eve Oras, et al. Data sharing practices and data availability upon request differ across scientific disciplines. *Scientific Data*, 8(1):192, 2021.
3. Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, Tianfan Fu, and Wenqi Wei. Machine learning for synthetic data generation: A review, 2024.
4. Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning, 2022.
5. Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard G. Baraniuk. Self-consuming generative models go mad, 2023.
6. Ousmane Touat, Julian Stier, Pierre-Edouard Portier, and Michael Granitzer. Gran is superior to graphrnn: node orderings, kernel- and graph embeddings-based metrics for graph generators, 2023.
7. Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models, 2018.
8. Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey, 2022.
9. Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günnemann, Neil Shah, and Meng Jiang. Graph data augmentation for graph machine learning: A survey, 2023.

10. Lu Lin, Jinghui Chen, and Hongning Wang. Spectral augmentation for self-supervised learning on graphs, 2023.
11. Hritik Bansal and Aditya Grover. Leaving reality to imagination: Robust classification via generated datasets. 2023.
12. Pei Liu, Xuemin Wang, Chao Xiang, and Weiye Meng. A survey of text data augmentation. In *2020 International Conference on Computer Communication and Network Security (CCNS)*, pages 191–195. IEEE, 2020.
13. Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. Auggpt: Leveraging chatgpt for text data augmentation, 2023.
14. Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Automated data augmentations for graph classification, 2023.
15. Yongduo Sui, Qitian Wu, Jiancan Wu, Qing Cui, Longfei Li, Jun Zhou, Xiang Wang, and Xiangnan He. Unleashing the power of graph data augmentation on covariate distribution shift, 2023.
16. Han Yue, Chunhui Zhang, Chuxu Zhang, and Hongfu Liu. Label-invariant augmentation for semi-supervised graph classification, 2022.
17. Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition?, 2023.
18. Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. This dataset does not exist: training models from generated images, 2019.
19. Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. Mocl: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph, 2022.
20. Zhengzheng Tang, Ziyue Qiao, Xuehai Hong, Yang Wang, Fayaz Ali Dharejo, Yuanchun Zhou, and Yi Du. Data augmentation for graph convolutional network on semi-supervised classification, 2021.
21. Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks, 2020.
22. William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.
23. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.
24. Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.
25. Frederik Diehl. Edge contraction pooling for graph neural networks, 2019.