GraphGT: Machine Learning Datasets for Graph Generation and Transformation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph generation, which learns from known graphs and discovers novel graphs, has great potential in numerous research topics like drug design and mobility synthesis and is one of the fastest-growing domains recently due to its promise for discovering new knowledge. Though many benchmark datasets have emerged in the domain of graph representation learning, the real-world datasets for graph generation problem are much fewer and limited to a small number of areas such as molecules and citation networks. To fill the gap, we introduce GraphGT, a large dataset collection for graph generation problem in machine learning, which contains 36 datasets from 9 domains across 6 subjects. To assist the researchers with better explorations of the datasets, we provide a systemic review and classification of the datasets from various views including research tasks, graph types, and application domains. In addition, GraphGT provides an easy-to-use graph generation pipeline that simplifies the process for graph data loading, experimental setup, model evaluation. The community can query and access datasets of interest according to a specific domain, task, or type of graph. GraphGT will be regularly updated and welcome inputs from the community. GraphGT is publicly available at `https://graphgt.github.io/` and can also be accessed via an open Python library.

1 Introduction

Graphs are ubiquitous data structures to capture connections (i.e., edges) between individual units (i.e., nodes). One central problem in machine learning on graphs is the gap between the discrete graph topological information and continuous numerical vectors preferred by data mining and machine learning models [1, 2, 3]. This directly leads to two major directions on graph research in modern machine learning: 1) graph representation learning [2, 4], which aims at encoding graph structural information into a (low-dimensional) vector space, and 2) graph generation [5, 6], which reversely aims at generating novel graph-structured data from the (low-dimensional) vector space. In the past several years, graph representation learning has enjoyed an explosive growth in machine learning. Techniques such as DeepWalk [7], graph convolutional network (GCN) [8], and graph attention networks (GAT) [9] have been proposed for various tasks including node classification [10], link prediction [11, 12], clustering [2, 4] and others [13, 14].

Beyond graph representation learning, graph generation and transformation via machine learning started to obtain fast-increasing attention in even more recent years. It enables end-to-end learning of underlying unknown graph generation or transformation process, which is a significant advancement beyond traditional prescribed graph models such as random graphs and stochastic block models which require strong human prior knowledge and hand-crafted rules. Hence, graph generation and transformation have great potential of many challenging tasks such as molecule design, mobility network synthesis, and protein folding statistical modeling. Over recent few years, substantial efforts have been paid on developing models and algorithms for graph generation and transformation, and a

Submitted to NeurIPS 2021 AI for Science Workshop.

few of them have been studied targeting specific domains, such as GraphVAE [15], MolGAN [16] and JT-VAE [17].

However, different from graph representation learning domain where there are various benchmark datasets such as CORA, CITESEER and PUBMED for node classification [18], OAG for link prediction [19], and Molecule-LENET for graph-level prediction [20], SNAP for general purpose network analysis and graph mining [21], OGB for realistic graph benchmarking [22] that have been developed and well-recognized for model evaluations and comparisons, graph generation via machine learning is still in its nascent stage and lack comprehensive benchmark datasets that well cover different key real-world applications and types of graph patterns. Existing datasets are usually limited to few domains such as citation networks and molecules. Moreover, most of the datasets for graph representation learning research cannot be used as graph generation benchmarks as the latter requires large number of individual whole graphs in order to learn the distributions of graphs and evaluate the learned distributions. Therefore, the gap between the fast-growing body of graph generation research and the paucity of benchmark datasets of this domain may limit its advancement.

In order to fill this gap, we develope and release GraphGT, a large dataset collection for graph generation and transformation via machine learning. The major contributions are as follows.

- 36 datasets are published under various graph types cover 6 disciplines (including biology, physics, chemistry, artificial intelligence (AI), engineering, and social science) and 9 domains (including protein, brain network, physical simulation, vision, molecule, transportation science, electrical and computer engineering (ECE), social network and synthetic data).
- Among all 36 datasets, 18 are collected by us, 8 are processed by us to construct graphs, 10 are reformatted to a unified format for easy access and use. We provide 3 types of APIs including graph generation dataloaders, graph transformation dataloaders, evaluators, and tutorials to use our APIs with 3 lines of code.
- Easy-to-use Python API for users to query and access pre-processed datasets according to specific disciplines, domains, and applications per their interests. We also provide a detailed tutorial for the implementation in the appendix. In addition to the access via the Python API, GraphGT is open-sourced and available for downloading via GitHub at `https://graphgt.github.io/`.

2 Graph Generation and Transformation

In this section, we briefly introduce the two tasks: graph generation and graph transformation, as well as their sub-categories which require different types of datasets.

A graph can be defined as $G = (\mathcal{V}, \mathcal{E}, E, F)$, where $\mathcal{V}$ is the set of N nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ corresponds to a set of edges. $e_{ij} \in \mathcal{E}$ is an edge that connects node v_i and $v_j \in \mathcal{V}$. If the graph is node-attributed or edge-attributed, it has the node attribute matrix $F \in \mathbb{R}^{N \times D}$ that assigns node attributes to each node or edge attribute tensor $E \in \mathbb{R}^{N \times K}$ that assigns attributes to each edge. D and K are dimensions of node attributes and edge attributes, respectively.

2.1 Graph Generation

Thanks to the development of graph representation learning, the surge of the graph-generation field is promoted by first encoding the node and edge attributes into a low-dimensional space to form the distribution of given graphs. Then based on the distribution learned from the given graphs, graph generation aims to sample novel graphs via well-designed probabilistic models [5]. More formally, given a set of observed graphs with arbitrary number of nodes and edges, graph generative models aim to learn the distribution $p(G)$ of the observed graphs and then graph generation can be achieved by sampling a graph G from the learned distribution $G \sim p(G)$.

According to the size of generated graph, graph generation tasks can be classified into two categories: (1) *fixed-size* generation in which the number of nodes is fixed across different graph samples; For example, in human brain networks (e.g., functional connectivity), the number of brain regions is usually the same across different human subjects; and (2) *variable-sized* generation when the number of nodes varies across graph samples. For example, different molecules can be considered as graphs with various numbers of atoms. The two categories are accommodated with different types of datasets.

2.2 Graph Transformation

Graph transformation aims at transforming from one graph in source domain into another graph in target domain. It can also be regarded as the graph generation conditioning on another graph. For instance, in neuroscience, it is interesting to explore the functional connectivity given the corresponding structural connectivity. In hardware design domain, given a integrated circuit design, one may be asked to obfuscate it, by adding additional gates and keys (i.e., can be considered as nodes) but maintain the same functionality. More formally, graph transformation problem can be formalized as learning a generative mapping $\mathcal{T} : (\mathcal{V}_0, \mathcal{E}_0, E_0, F_0) \rightarrow (\mathcal{V}', \mathcal{E}', E', F')$, in which $(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$ corresponds to the graph in source domain and $(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$ represents a graph in target domain.

Based on the entities that are being transformed in the transformation process, problems regarding graph transformation can be further divided into three main scenarios: node transformation, edge transformation, and node-edge co-transformation. As the name suggests, (1) *node transformation* transforms nodes and/or their attributes from the source to the target domain. (2) *Edge transformation* maps graph topology and/or edge attributes from the source domain to the target domain. In the process of (3) *node-edge co-transformation*, both the above node and edge information can change during the transformation process.

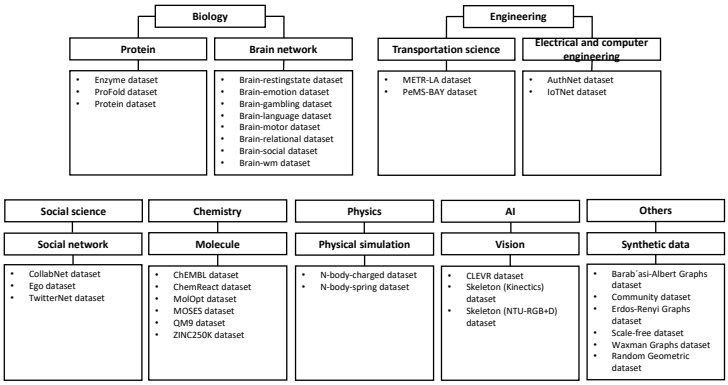

Figure 1: GraphGT Benchmark datasets by domains (alphabetical order under each domain)

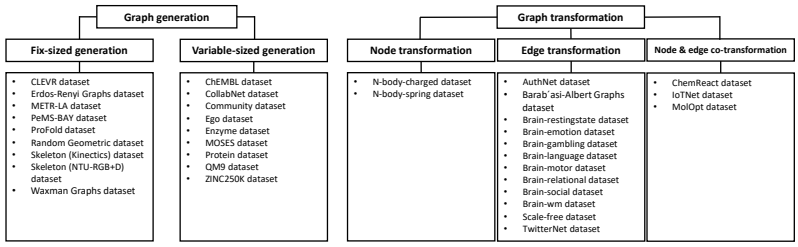

Figure 2: GraphGT benchmark datasets by tasks (alphabetical order under each task)

3 GraphGT Pipeline

3.1 Datasets

Our GraphGT Benchmark covers in total 36 datasets from various domains and different tasks. The taxonomy with respect to different domains is shown in Figure 1, where there are 9 domains, including protein, brain network, physical simulation, vision, molecule, transportation science, electrical and computer engineering, social network and synthetic data, across 6 subjects including biology, physics, artificial intelligence, chemistry, engineering and social science. Moreover, the taxonomy by different

tasks is illustrated in Figure 2. For the graph generation task, they can extract datasets for either fixed-sized generation or variable-sized generation. For the graph transformation task, we provide datasets for node transformation, edge transformation as well as node and edge co-transformation. The general profiles for different datasets are summarized in Table 1. A more detailed description of each dataset and curation method can be found in the appendix.

3.2 Evaluations

There are two main types of evaluations for graph generation and two main types of evaluations for graph transformation. For graph generation task, (1)*statistics-based evaluation* measures the quality of the generated graphs by computing the distance between the graph statistic distribution of real graphs and generated graphs, and (2)*self-quality based evaluation* measures the quality of the generated graphs: validity, uniqueness and novelty. For graph transformation task, (1)*Graph-property-based evaluation* directly compares each generated graph to its label graph by measuring their similarity or distance based on some graph properties or kernels, such as random-walk kernel similarity [23], and (2)*Mapping-relationship-based evaluation* measures whether the learned relationship between the input and the generated graphs is consistent with the true relationship between the input and the real graphs. The detailed elaborations for each type of evaluation metrics and examples can be found in the appendix.

Table 1: Summary of statistics and types of graphs for different GraphGT datasets. (Note: 'Y' stands for 'Yes', 'N' stands for 'No', 'GCS' stands for 'Geographic Coordinate System', '2D/3D' stands for '2D or 3D coordinates under Cartesian Coordinate System'.)

Name	Type	#Graphs	#Nodes	#Edges	Attributed	Directed	Weighted	Signed	Homogeneous	Spatial	Temporal	Labels
QM9 [24]	Molecules	133,885	∼ 9	∼ 19	Y	N	Y	N	Y	3D	N	Y
ZINC250K [25]	Molecules	249,455	∼ 23	∼ 50	Y	N	Y	N	Y	3D	N	Y
MOSES [26]	Molecules	193,696	∼ 22	∼ 47	Y	N	Y	N	Y	3D	N	Y
MolOpt [27]	Molecules	229,473	∼ 24	∼ 53	Y	N	Y	N	Y	3D	N	Y
ChEMBL [28]	Molecules	1,799,433	∼ 27	∼ 58	Y	N	Y	N	Y	3D	N	Y
ChemReact [29]	Molecules	7,180	∼20	∼ 16	Y	N	Y	N	Y	3D	N	Y
Protein [30]	Proteins	1,113	∼39	∼73	Y	N	N	N	Y	N	N	Y
Enzyme [31]	Proteins	600	∼33	∼62	Y	N	N	N	Y	N	N	Y
ProFold [32]	Proteins	76,000	8	∼40	Y	N	N	N	Y	3D	Y	Y
Brain-restingstate [29]	Brain networks	823	68	2274	N	N	Y	Y	Y	N	N	Y
Brain-emotion [29]	Brain networks	811	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-gambling [29]	Brain networks	818	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-language [29]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-motor [29]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-relational [29]	Brain networks	808	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-social [29]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-wm [29]	Brain networks	812	68	2278	N	N	Y	Y	Y	N	N	Y
N-body-charged [33]	Physical simulation networks	3,430,000	25	∼3	Y	N	N	N	Y	2D	Y	Y
N-body-spring [33]	Physical simulation networks	3,430,000	5	∼10	Y	N	N	N	Y	2D	Y	Y
CLEVR [34]	Scene graphs	85,000	6	∼40	Y	Y	Y	N	Y	3D	N	N
Skeleton (Kinectics) [35]	Skeleton graphs	260,000	18	17	N	N	N	N	Y	2D	Y	Y
Skeleton (NTU-RGB+D) [36]	Skeleton graphs	56,000	25	24	N	N	N	N	Y	3D	Y	Y
METR-LA [37]	Traffic networks	34,272	325	2,369	Y	Y	Y	N	Y	GCS	Y	Y
PeMS-BAY [38]	Traffic networks	50,112	207	1,515	Y	Y	Y	N	Y	GCS	Y	Y
AuthNet [39]	Authen. networks	114/412	50/300	∼3/∼7	N	Y	Y	N	Y	N	N	Y
IoTNet [29]	IoT networks	343	20/40/60	∼220/∼630/∼800	Y	N	Y	N	Y	N	N	Y
CollabNet [40]	Collab. networks	2,361	303,308	207,632	N	N	N	N	Y	GCS	Y	Y
TwitterNet [41]	social networks	2,580	300	0.5	N	N	N	N	Y	N	N	N
Barab'asi-Albert Graphs [29]	Synthetic networks	1,000	20/40/60	∼60/∼190/∼300	Y	N	N	N	Y	N	N	N
Erdos-Renyi Graphs [29]	Synthetic networks	1,000	20/40/60	∼100/∼200/∼400	Y	N	N	N	Y	N	N	N
Scale-Free [39]	Synthetic networks	10,000	10/20/50/100/150	20/ 40/ 100/ 200/ 320	N	Y	N	N	Y	N	N	N
Random Geometric [32]	Synthetic networks	9,600	25	∼350	Y	N	N	N	Y	Y	Y	Y
Waxman Graphs [32]	Synthetic networks	9,600	25	∼250	Y	N	N	N	Y	Y	Y	Y

4 Conclusion

Although many benchmark datasets have emerged in the domain of graph representation learning, the real-world datasets for graph generation are much fewer and limited to a small number of areas. To fill this gap, we introduce GraphGT, a large dataset collection for graph generation problem in machine learning. GraphGT covers datasets in 9 domains across 6 subjects, in which 18 are collected by us, 8 are processed by us to construct graphs, 10 are reformatted to a unified format for easy access and use. In addition, we provide 3 types of Python APIs, including dataset downloader, graph generation dataloader, graph transformation dataloader and evaluator, for users to query and access datasets according to specific disciplines, domains and applications per their interests. We believe that GraphGT can advance the community to address significant challenges in graph generation and transformation.

References

[1] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[2] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning: A survey. *APSIPA Transactions on Signal and Information Processing*, 9, 2020.

[3] Pierre Latouche and Fabrice Rossi. Graphs in machine learning: an introduction. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Proceedings of the 23-th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2015)*, pages 207–218, 2015.

[4] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[5] Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *arXiv preprint arXiv:2007.06686*, 2020.

[6] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[8] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[10] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[11] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[12] Tianyu Xia, Yijun Gu, and Dechun Yin. Research on the link prediction model of dynamic multiplex social network based on improved graph representation learning. *IEEE Access*, 9:412–420, 2020.

[13] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows for molecule generation in 3d. *arXiv preprint arXiv:2105.09016*, 2021.

[14] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017.

[15] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN'2018*, pages 412–422, 2018.

[16] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *ICML'2018 Workshop*, 2018.

[17] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML'2018*, pages 2323–2332, 2018.

[18] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[19] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. Netsmf: Large-scale network embedding as sparse matrix factorization. In *The World Wide Web Conference*, pages 1509–1520, 2019.

[20] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: machine learning datasets and tasks for therapeutics. *arXiv preprint arXiv:2102.09548*, 2021.

[21] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[22] 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. *arXiv preprint arXiv:2005.00687*, 2020.

[23] U Kang, Hanghang Tong, and Jimeng Sun. Fast random walk graph kernel. In *SDM'2012*, pages 828–838, 2012.

[24] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[25] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

[26] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:1931, 2020.

[27] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018.

[28] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

[29] Xiaojie Guo, Liang Zhao, Cameron Nowzari, Setareh Rafatirad, Houman Homayoun, and Sai Manoj Pudukotai Dinakarrao. Deep multi-attributed graph translation with node-edge co-evolution. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 250–259. IEEE, 2019.

[30] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[31] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

[32] Xiaojie Guo, Yuanqi Du, and Liang Zhao. Deep generative models for spatial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 505–515, 2021.

[33] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

[34] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[35] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[36] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[37] Hosagrahar V Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[38] Chao Chen. *Freeway performance measurement system (PeMS)*. University of California, Berkeley, 2002.

[39] Xiaojie Guo, Lingfei Wu, and Liang Zhao. Deep graph translation. *arXiv preprint arXiv:1805.09980*, 2018.

[40] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[41] Yuyang Gao and Liang Zhao. Incomplete label multi-task ordinal regression for spatial event scale forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[42] M. Lowe D. Patent reaction extraction: downloads; https://bitbucket.org/dan2097/ patent-reaction-extraction/downloads., 2014.

[43] Mark Jenkinson, Christian F Beckmann, TE Behrens, Mark W Woolrich, and Stephen M Smith. Neuroimage. *Fsl*, 62(2):782–790, 2012.

[44] Alexander D Kent. Comprehensive, multi-source cyber-security events data set. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2015.

[45] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst, Todd M Austin, Trevor Mudge, and Richard B Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*, pages 3–14. IEEE, 2001.

[46] John L Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006.

[47] Hossein Sayadi, Nisarg Patel, Sai Manoj PD, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[48] Sai Manoj Pudukotai Dinakarrao, Hossein Sayadi, Hosein Mohammadi Makrani, Cameron Nowzari, Setareh Rafatirad, and Houman Homayoun. Lightweight node-level malware detection and network-level malware confinement in iot networks. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 776–781. IEEE, 2019.

[49] Hossein Sayadi, Hosein Mohammadi Makrani, Sai Manoj Pudukotai Dinakarrao, Tinoosh Mohsenin, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. 2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 728–733. IEEE, 2019.

[50] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[51] Jiaxuan You, Rex Ying, and Xiang Ren et al. Graphrnn: generating realistic graphs with deep auto-regressive models. In *ICML'2018*, pages 5708–5717, 2018.

[52] Yuyang Gao, Liang Zhao, Lingfei Wu, Yanfang Ye, Hui Xiong, and Chaowei Yang. Incomplete label multi-task deep learning for spatio-temporal event subtype forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3638–3646, 2019.

[53] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[54] Béla Bollobás and Oliver M Riordan. Mathematical results on scale-free random graphs. *Handbook of graphs and networks: from the genome to the internet*, pages 1–34, 2003.

[55] Bernard M Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[56] Mariya Popova, Mykhailo Shvets, and Junier Oliva et al. Molecularrnn: generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

[57] Kaushalya Madhawa, Katushiko Ishiguro, and Kosuke Nakago et al. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[58] Giuseppe Jurman, Roberto Visintainer, and Michele Filosi et al. The him glocal metric and kernel for network comparison and classification. In *DSAA'2015*, pages 1–10, 2015.

[59] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

[60] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[61] Nino Shervashidze, Pascal Schweitzer, and Erik Jan Van Leeuwen et al. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

[62] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: a scalable approach to domain-agnostic labeled graph generation. In *WWW'20*, pages 1253–1263, 2020.

[63] Jiaxuan You, Bowen Liu, and Zhitao Ying et al. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS'2018*, pages 6410–6421, 2018.

A Key Information about GraphGT

A.1 Dataset Documentation

We provide detailed documentation of dataset collection, processing, task for each dataset both in section B and in our website. We provide statistics, taxonomy, detailed description, and task for each dataset and can be tracked in our website `https://graphgt.github.io/`.

A.2 Intended Use

GraphGT is intended for the deep graph learning as well as specific domain (e.g. physics, biology, chemistry, etc.) community to use and develop machine learning algorithms to advance applications in various domains.

A.3 URLs

Official website (`https://graphgt.github.io/`) contains all references of GraphGT, including dataset taxonomy, task, evaluation, visualization, tutorials, papers, GitHub, and other useful resources.

GitHub repository (`https://github.com/yuanqidu/GraphGT`) hosts all source codes, installation instructions, and tutorials of GraphGT.

A.4 Hosting and Maintenance Plan

Our GraphGT Python library is regularly maintained and version-tracked via GitHub. All datasets are currently hosted on Dropbox and will be transferred to Emory University server soon. Our dataset is both directly downloadable with a Dropbox link or from our Python APIs. Our core team commit to maintain this initiative for at least five years. In the meantime, we will expand the community in multiple dimensions and attract external contributors from the whole community. We will regularly update new dataset, task, evaluation and visualization methods to GraphGT.

A.5 Limitations

Graph generation and transformation is a fast-growing, vast, and promising field and their applications cover a wide range of applications. We start this initiative to build the infrastructure for the community which includes most of the mainstream datasets in the graph generation and transformation field and many more new datasets. However, it is an ongoing effort and we strive to continuously include more datasets, evaluation and visualization methods to advance the field.

A.6 Potential Negative Societal Impacts

Graph generation and transformation are motivated by generating novel graph-structured data and understanding the graph-structured data; thus, they have vast applications, such as drug discovery, protein design, mobility synthesis, etc., which could potentially lead to better designed drug, traffic network, etc., and save lives, time, etc. We envision that GraphGT can facilitate algorithmic and scientific advances in various domains across subjects and accelerate machine learning model development and application for real-world use. GraphGT neither involves human subject research nor contains personally identifiable information.

B Dataset Details

We list detailed information for each of the datasets stored in GraphGT.

B.1 Molecules

We have 6 molecule datasets, in which 4 (QM9 [24], ZINC250K [25], MOSES [26], ChEMBL [28]) for graph generation and 2 (MolOpt [27], ChemReact [29]) for graph transformation. For all of the molecule datasets, we store adjacency matrix, node feature (i.e. atoms), edge feature (i.e. bonds), spatial feature (i.e. geometry), and smiles (i.e. string representation). There are in total 4 types of

atoms in QM9, 0 = H, 1 = C, 2 = N, 3 = O, 4 = F. There are in total 14 types of atoms in ZINC250K dataset, MOSES, and ChEMBL dataset, 0 = Br, 1 = C, 2 = Cl, 3 = F, 4 = H, 5 = I, 6 = N, 7 = N, 8 = N, 9 = O, 10 = O, 11 = S, 12 = S, 13 = S. There are in total 4 types of bonds in all the datasets, and we represent them as follows: 0 = Single, 1 = Double, 2 = Triple, 3 = Aromatic.

QM9 [24] dataset is an enumeration of around 134k stable organic molecules with up to 9 heavy atoms (carbon, oxygen, nitrogen and fluorine). As no filtering is applied, the molecules in this dataset only reflect basic structural constraints.

ZINC250K [25] dataset is a curated set of 250k commercially available drug-like chemical compounds. On average, these molecules are bigger (about 23 heavy atoms) and structurally more complex than the molecules in QM9 dataset.

Molecular Sets (MOSES) [26] is a benchmark platform for distribution learning based molecule generation. Within this benchmark, MOSES provides a cleaned dataset of molecules that are ideal of optimization. It is processed from the ZINC Clean Leads dataset.

ChEMBL [28] dataset is a manually curated database of bioactive molecules with drug-like properties. It brings together chemical, bioactivity and genomic data to aid the translation of genomic information into effective new drugs.

MolOpt [27] dataset extracts translation pairs from the ZINC database in terms of three molecular properties, Penalized logP, Drug-likeness, and Dopamine Receptor.

ChemReact [29] dataset has totally 7180 pairs of reactant and product molecule graph in the dataset derived from USPTO dataset [42].

B.1.1 License

QM9: CC BY-NC-SA 4.0.

ZINC250K: Free to use for everyone.

MOSES: The dataset is generated by [26], which is under MIT License. The license of the dataset is not specified.

ChEMBL: CC BY-NC-SA 3.0.

MolOpt: Extracted from ZINC Database.

ChemReact: Not specified.

B.2 Proteins

We have three protein datasets available in GraphGT, which includes protein structures, Enzyme and dynamic protein folding process.

Protein [30] dataset contains 918 protein graphs with $100 \leq \|V\| \leq 500$. Each protein is represented by a graph, where nodes are amino acids and two nodes are connected if they are less than 6 Angstroms apart.

Enzyme [31] dataset contains protein tertiary structures representing 600 Enzyme. Nodes in a graph (protein) represent secondary structure elements, and two nodes are connected if the corresponding elements are interacting. The node labels indicate the type of secondary structure, which is either helices, turns, or sheets.

ProFold [32] dataset contains dynamic folding processes of a protein peptide with sequence AGAAAAGA in 38 steps. The node feature of each protein is the sequence (AGAAAAGA) along with the spatial locations of each amino acid, and the edge feature of each protein is an adjacency matrix constructed by connecting all pairs of nodes with distance < 8 Å.

B.2.1 License

Enzyme: CC-BY-4.0.

ProFold: The dataset is collected by [32]. The license is not specified.

Protein: CC-BY-4.0.

B.3 Brain Networks

The Brain dataset comes from the human connectome project (HCP) [29] and has a few branches: restingstate, emotion, gambling, language, motor, relational, social and wm according to different tasks. In this dataset, the source graphs reflect the structural connectivity (SC), and the target graphs represent the functional connectivity [29]. Specifically, both types of connectivities are processed from the magnetic resonance imaging (MRI) data from HCP. SC is obtained by applying probabilistic tracking on the diffusion MRI data by Probtrackx tool from the FMRIB Software Library [43] with 68 regions of insterest (ROI). The edge attributes of FC are defined as Pearson's correlation between two ROIs blood oxygen level-dependent time obtained from the resting-state functional MRI data. Node attributes is a one-hot vector representing index of each node. In total, 823 pairs of SC and FC samples are enrolled in the dataset.

B.3.1 License

Brain: This dataset comes from the human connectome project. Data collection and sharing for this project was provided by the MGH-USC Human Connectome Project (HCP; Principal Investigators: Bruce Rosen, M.D., Ph.D., Arthur W. Toga, Ph.D., Van J. Weeden, MD). HCP funding was provided by the National Institute of Dental and Craniofacial Research (NIDCR), the National Institute of Mental Health (NIMH), and the National Institute of Neurological Disorders and Stroke (NINDS). HCP data are disseminated by the Laboratory of Neuro Imaging at the University of Southern California.

B.4 N-body Simulations

N-body-charged [33] dataset simulates a system containing 5 particles with positive or negative charges. Particles are located in 2D coordinates without any external forces except attracting force and repelling force. The quantity of electrical charges is sampled from uniform probability. Each particle interacts via Coulomb forces. Every two particles interact, either attract or repel each other. The temporal length of each sequence is 49, which obtains from sub-sampling every 100 steps in a trajectory.

N-body-spring [33] dataset simulates a system containing 5 particles connected by springs. Particles are located in 2D coordinates without any external forces except elastic collisions. Particles are connected via springs with probability of 0.5, and interactions between springs follow Hooke's law. The initial location of each particle is sampled from a Gaussian distribution and the initial velocity of each particle is a random vector of norm 0.5. The trajectories of all springs are calculated by solving Newton's equations of motion PDE. The temporal length of each sequence is 49, which obtains from sub-sampling every 100 steps in a trajectory.

B.4.1 License

N-body-charged: The dataset is simulated by [33], which is under MIT License. The license of the dataset is not specified.

N-body-spring: The dataset is simulated by [33], which is under MIT License. The license of the dataset is not specified.

B.5 Collaboration Networks

CollabNet [40] dataset is collected from DBLP-Citation-network V12, which contains around 4.9 million papers and 45 million citation relationships. We construct graphs by selecting authors as nodes and co-authorships as edges during the time period from 1990 to 2019. To cut the graphs into pieces, we generate sub-graphs based on the Fields of Study attribute from papers. For each field, we generate one spatio-temporal graph. We generate 2361 spatio-tempora graphs with a total of around 9 million nodes and a total of around 6 million of edges.

B.5.1 License

CollabNet: The dataset is collected from DBLP-Citation-network V12. The license is not specified.

B.6 Traffic Networks

METR-LA [37] dataset is collected by Los Angeles Metropolitan Transportation Authority (LA-Metro), and processed by University of Southern California's Integrated Media Systems Center. This dataset contains traffic information collected from 207 loop detectors in the highway of Los Angeles County for 4 months (from Mar 1st 2012 to Jun 30th 2012). Each sensor records traffic speed value per 5 minutes.

PeMS-BAY [38] dataset is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). PeMS-BAY dataset collects traffic information in the Bay Area. The dataset contains traffic information of 325 sensors within 5 months (From Jan 1st 2017 to May 31st 2017). Each sensor records traffic speed value per 5 minutes.

B.6.1 License

METR-LA: The dataset is collected by Los Angeles Metropolitan Transportation Authority (LA-Metro), and processed by University of Southern California's Integrated Media Systems Center. The license is not specified.

PeMS-BAY: The dataset is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). The license is not specified.

B.7 Authentication Networks

AuthNet dataset includes the authentication activities of users on their computers and servers in their enterprise computer network and is published by Los Alamos National Laboratory (LANL). [44, 39]. There are two subsets of different sizes of graphs (e.g., 50 and 300) in AuthNet dataset. For each subset, we train and test folder separately. Train set contains the graph pairs (one-to-one) which are just used for training. Test set contains data for each user. For each user, there are several input graphs (e.g., regular user authentication activity graph) and several target graphs (e.g., malware user authentication activity graph). Input and target graphs in test set are not one-to-one, which can be tested by indirect evaluation. There are no node attributes for this dataset, and only edge attribute is considered. For each graph, the value of the $i-th$ row and the $j-th$ column refers to the edge attribute of node i and j (0 refers to no links).

B.7.1 License

AuthNet: The dataset is publically released by LANL [44]. To the extent possible under law, LANL has waived all copyright and related or neighboring rights to User-Computer Authentication Associations in Time. This work is published from: United States.

B.8 IoT Networks

IoTNet is the malware dataset collected for malware confinement prediction [29]. There are three sets of IoT nodes at different amounts (20, 40 and 60) encompassing temperature sensors connected with Intel ATLASEDGE Board and Beagle Boards (BeagleBone Blue), communicating via Bluetooth protocol. Benign and malware activities are executed on these devices to generate the initial attacked networks as input graphs. Benign activities include MiBench [45] and SPEC2006 [46], Linux system programs, and word processors. The nodes represent devices and node attribute is a binary value referring to whether the device is compromised or not. Edge represents the connection of two devices and the edge attribute is a continuous value reflecting the distance of two devices. The real target graphs are generated by the classical malware confinement method: stochastic controlling with malware detection [47, 48, 49]. We collect 334 pairs of input and target graphs with different contextual parameters (infection rate, recovery rate and decay rate) for each of the three datasets. In this dataset, there are both nodes attributes and edge attributes considered.

B.8.1 License

IoTNet: The dataset is generated by [29]. The license is not specified.

B.9 Skeleton Graphs

Kinetics [35] dataset is a large-scale human action dataset with 300000 videos clips in 400 classes. Those video clips are from YouTube with a great variety. The raw Kinetics dataset doesn't contain skeleton data, and [35] uses OpenPose toolbox to generate skeleton with 18 joints on every frame. Kinetics-Skeleton contains 240000 clips of training data and 20000 clips of test data.

NTU-RGB+D [36] dataset is a large and widely used action recognition dataset with 56000 action clips in 60 classes. These clips are performed by 40 volunteers captured in a constrained lab environment, with three camera views recorded simultaneously. The dataset provides 3D joint locations of each frame and 25 joints for each subject.

B.9.1 License

Skeleton (Kinectics): CC BY 4.0.

Skeleton (NTU-RGB+D): Not specified.

B.10 Social Networks

Ego: Ego dataset contains 757 3-hop ego networks extracted from the Citeseer [50]. The number of nodes of the graph in Ego dataset ranges from 50 to 399. Nodes represent documents and edges represent citation relationships [51].

TwitterNet: The dataset is processed by [41] and obtained from 5 different countries in Latin America, namely Brazil, Colombia, Mexico, Paraguay, and Venezuela. Data sources from Twitter are adopted as the model inputs. In each case the data for the period from July 1, 2013 to February 9, 2014 is used for training and validation, where the validation set consists of a randomly chosen 30% of the data, and the rest is used for training; the data from February 10, 2014 to December 31, 2014 is used for the performance evaluation.

B.10.1 License

Ego: This dataset is extracted from Citeseer [50]. Citeserr is under CC BY-NC-SA 3.0.

TwitterNet: The dataset is obtained from [52]. The license is not specified.

B.11 Scene Graphs

CLEVR [34] dataset provides a dataset for visual question answer, which can be formalized as a spatial-graph dataset. There are 10 objects in the image with different 3D locations. Each object is identified by its shape, such as sphere, cylinder, and cube. The relationship between two objects can be categorized into four types: right, behind, front, left, with directions. Thus, each image can be formalized as a labeled directed graph with different edge types and node types. Thus, the spatial information of each nodes is closely correlated with the edge types between each pair of nodes.

B.11.1 License

CLEVR: CC BY 4.0.

B.12 Synthetic Graphs

Barab'asi-Albert Graphs: This dataset is generated by the Barab'asi-Albert model [29]. It fits the "one-to-one" mapping problem of graph translation. It contains pairs of input and target graphs. The target graph topology is the 2-hop connection of the input graph, where each edge in the target graph refers to the 3-hop reachability in the input graph (e.g., if node i is 3-hop reachable to node j in the input graph, then they are connected in the target graph). There are edge and node attributes for graphs in this dataset: the edge attribute $E_{(i,j)}$ denotes the existence of the edge, and the node attributes

are continuous values computed following the polynomial function: $f(x) : y = ax^2 + bx + c$ $(a = 0; b = 1; c = 5)$, where x is the node degree and $f(x)$ is the node attribute. Here we provide the datasets with three different node sizes.

Community: This dataset is generate by [51] and contains 500 two-community graphs with number of nodes ranging from 60 to 160. Each community is generated by the Erdos-Renyi model (E-R) [53] with $\frac{|V|}{2}$ nodes and the edge probability of 0.3. Then add $0.05|V|$ inter-community edges are added with uniform probability.

Erdos-Renyi graphs: This dataset is generated by the Erdos-Renyi model with the edge probability of 0.2 [29]. It fits the "one-to-one" mapping problem of graph translation. It contains pairs of (input, target) graphs. The target graph topology is the 2-hop connection of the input graph, where each edge in the target graph refers to the 2-hop reachability in the input graph (e.g., if node i is 2-hop reachable to node j in the input graph, then they are connected in the target graph). There are edge and node attributes for graphs in this dataset: the edge attribute $E_{(i,j)}$ denotes the existence of the edge, and node attributes are continuous values computed following the polynomial function: $f(x) : y = ax^2 + bx + c$ $(a = 0; b = 1; c = 5)$, where x is the node degree and $f(x)$ is the node attribute.

Scale-free: This dataset is generated as a directed scale-free network [39], which is a network whose degree distribution follows power-law property [54]. It fits the "one-to-many" mapping graph translation problem. There are no node features in this dataset, and the goal is to learn the mapping from the input graph's topology to the target graph's topology. To generate a target graph, a node will by selected as target node with probability proportional to its in-degree, which will be linked to a new source node with probability of 0.41. Similarly, a node will be selected as the source node with the probability proportional to its out-degree, which will be linked to a new target node with the probability of 0.54. Then, a corresponding target graph is generated by adding m (number of nodes of the input graph) edges between two nodes. Thus, both input and target graphs are directed scale-free graphs.

Waxman graphs: This datase contains graphs generated by the Waxman random graph model that places n nodes uniformly at random in a rectangular domain [55, 32]. There are three types of factors that are related to the generation of Waxman graphs: the independent graph factor b that controls node attributes, the independent spatial factor p that controls the overall node positions, and the graph-spatial correlated factor s that controls both graph and spatial density [32]. There are 80,000 samples for training and 80,000 for testing.

Random Geometric Graphs: This datase contains graphs generated by the random geometric graph model that places n nodes uniformly at random in a rectangular domain [32]. Two nodes are joined by an edge if their distance is larger than a threshold $\beta = 12$. The node attributes among a graph are generated in the same rule as that for generating Waxman graphs. There are 8,000 samples for training and 1,600 for testing in this dataset.

B.12.1 License

Barab'asi-Albert Graphs: The dataset is generated by [29]. The license is not specified.

Community: The dataset is generated by [51], which is under MIT License. The license of the dataset is not specified.

Erdos-Renyi graphs: The dataset is generated by [29]. The license is not specified.

Scale-free: The dataset is generated by [39]. The license is not specified.

Waxman graphs: The dataset is generated by [32]. The license is not specified.

Random geometric: The dataset is generated by [32]. The license is not specified.

C Evaluations

C.1 Graph Generation

Statistics-based evaluation measures the quality of the generated graphs by computing the distance between the graph statistic distribution of real graphs and generated graphs. In the deployed API, seven

typical graph statistics are considered, which are summarized as follows: (1) *Node degree distribution*: the empirical node degree distribution of a graph, which could encode its local connectivity patterns. (2) *Clustering coefficient distribution*: the empirical clustering coefficient distribution of a graph. Intuitively, the clustering coefficient of a node is calculated as the ratio of the potential number of triangles the node could be part of to the actual number of triangles the node is part of. (3) *Orbit count distribution*; the distribution of the counts of node 4-orbits of a graph. Intuitively, an orbit count specifies how many of these 4-orbits substructures the node is part of. This measure is useful in understanding if the model is capable of matching higher-order graph statistics, as opposed to node degree and clustering coefficient, which represent measures of local (or close to local) proximity. (4) *Largest connected component*: the size of the largest connected component of the graphs. (5) *Triangle count*: the number of triangles counted in the graph. (6) *Characteristic path length*: the average number of steps along the shortest paths for all node pairs in the graph. (7) *Assortativity*: the Pearson correlation of degrees of connected nodes in the graph. To calculate the distances regarding the above mentioned statistics, *Average Kullback-Leibler Divergence* and *Maximum Mean Discrepancy (MMD)* are utilized.

Self-quality based evaluation measures the quality of the generated graphs: validity, uniqueness and novelty. The definition and calculation of the three metrics are provided as follows: (1) *Validity*: validity aims to evaluate the graphs by judging whether they preserve some properties. For example, for cycles graphs/Tree graphs, the validity is calculated as what percentage of generated graphs are actually cycles or trees [6]. For molecule graphs, validity is about the percentage of chemically valid molecules based on some domain specific rules [56]. (2) *Uniqueness*: ideally, high-quality generated graphs should be diverse and similar, but not identical. Thus, uniqueness is utilized to capture the diversity of generated graphs [57, 6, 56]. To calculate the uniqueness of a generated graph, the generated graphs that are sub-graph isomorphic to some other generated graphs are first removed. The percentage of graphs remaining after this operation is defined as Uniqueness. For example, if the model generates 100 graphs, all of which are identical, the uniqueness is 1/100 = 1%. (3) *Novelty*. Novelty measures the percentage of generated graphs that are not sub-graphs of the training graphs and vice versa [57]. Note that identical graphs are defined as graphs that are sub-graph isomorphic to each other. In other words, novelty checks if the model has learned to generalize unseen graphs.

C.2 Graph Transformation

Graph-property-based evaluation directly compares each generated graph to its label graph by measuring their similarity or distance based on some graph properties or kernels, such as the following: (1) random-walk kernel similarity by using the random-walk based graph kernel [23]; (2) combination of Hamming and Ipsen-Mikhailov distances(HIM) [58]; (3) spectral entropies of the density matrices; (4) eigenvector centrality distance [59]; (5) closeness centrality distance [60]; (6) Weisfeiler Lehman kernel similarity [61]; (7) Neighborhood Sub-graph Pairwise Distance Kernel [62] by matching pairs of subgraphs with different radii and distances.

Mapping-relationship-based evaluation measures whether the learned relationship between the input and the generated graphs is consistent with the true relationship between the input and the real graphs. There are two kinds of relationship to be considered [5] as follows: (1) *Explicit mapping relationship*. Considering the situation where the true relationship between the input conditions and the generated graphs is known in advance, the evaluation can be conducted as follows: we quantitatively compare the property scores of the generated and input graphs to see if the change indeed meets the requirement. For example, one can compute the improvement of logP scores from the input molecule to the optimized molecule in molecule optimization task [63]. (2) *Implicit mapping relationship*. When the underlying patterns of the mapping from the input graphs to the real target graphs are implicit and complex to define and measure, a classifier-based evaluation metric can be utilized [39]. By regarding the input and target graphs as two classes, it assumes that a classifier that is capable of distinguishing the generated target graphs would also succeed in distinguishing the real target graphs from the input graphs. Specifically, a graph classifier is first trained based on the input and generated target graphs. Then this trained graph classifier is tested to classify the input graph and real target graphs, and the results will be used as the evaluation metrics.

D Tutorials

We provide dataloaders, evaluators, as well as visualizers which simplify the pipeline for graph generation and transformation, as shown in Fig. 3, 4 and 5, respectively.

```
import graphgt
```

```
qm9_data_loader = graphgt.DataLoader(name='qm9', save_path='./', format='numpy')
```

```
Downloading node feature...
100%|| 31.1M/31.1M [00:03<00:00, 9
.13MiB/s]
Done!
Downloading edge feature...
100%|| 49.5M/49.5M [00:04<00:00, 1
0.1MiB/s]
Done!
Downloading spatial feature...
100%|| 35.8M/35.8M [00:03<00:00, 9
.19MiB/s]
Done!
Downloading adjacency matrix...
100%|| 49.5M/49.5M [00:04<00:00, 1
0.2MiB/s]
Done!
Downloading smiles string...
100%|| 15.0M/15.0M [00:02<00:00, 6
.97MiB/s]
Done!
```

```
adj, node_feat, edge_feat, spatial, smile = qm9_data_loader.get_data()
```

Figure 3: Loading generation dataset.

```
import graphgt
```

```
ER20_data_loader = graphgt.DataLoader(name='ER_20', save_path='./', format='numpy')
```

```
Downloading input node feature...
100%|| 80.1k/80.1k [00:00<00:00,
230kiB/s]
Done!
Downloading input edge feature...
100%|| 1.60M/1.60M [00:01<00:00, 1
.51MiB/s]
Done!
Downloading input adjacency matrix...
100%|| 800k/800k [00:00<00:00,
902kiB/s]
Done!
Downloading target node feature...
100%|| 80.1k/80.1k [00:00<00:00, 1
.20MiB/s]
Done!
Downloading target edge feature...
100%|| 1.60M/1.60M [00:01<00:00, 1
.54MiB/s]
Done!
Downloading adjacency matrix...
100%|| 800k/800k [00:00<00:00,
883kiB/s]
Done!
```

```
input_adj, input_node_feat, input_edge_feat, input_spatial, target_adj, target_node_feat, target_edge_feat, target_s
```

Figure 4: Loading transforamtion dataset.

```
import graphgt
import numpy as np
```

```
batch = 1000
x = np.random.rand(batch,1)
y_baseline = np.random.rand(batch,1)
y_pred = np.zeros((batch,1))

print('MMD baseline', graphgt.compute_mmd(x,y_baseline))
print('MMD prediction', graphgt.compute_mmd(x,y_pred))
print ('KLD', graphgt.compute_kld(x,y_baseline))
print ('EMB', graphgt.compute_emd(x,y_baseline))
```

```
MMD baseline 9.684740112247958e-05
MMD prediction 0.3751574658037742
KLD [0.51577211]
EMB 0.01009273634128826
```

Figure 5: Evaluation APIs.